OMRON

Machine Automation Controller

NJ/NX-series

CPU Unit

Motion Control User's Manual

NX701-1□□□
NX502-1□□□
NX102-1□□□
NX102-90□□
NX1P2-1□□□□□
NX1P2-9□□□□□
NJ501-□□□□
NJ301-1□□□
NJ101-10□□

CPU Unit

SYSMAC
always in control

W507-E1-32

NOTE

1. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, mechanical, electronic, photocopying, recording, or otherwise, without the prior written permission of OMRON.
2. No patent liability is assumed with respect to the use of the information contained herein. Moreover, because OMRON is constantly striving to improve its high-quality products, the information contained in this manual is subject to change without notice.
3. Every precaution has been taken in the preparation of this manual. Nevertheless, OMRON assumes no responsibility for errors or omissions.
   Neither is any liability assumed for damages resulting from the use of the information contained in this publication.

Trademarks

- Sysmac and SYSMAC are trademarks or registered trademarks of OMRON Corporation in Japan and other countries for OMRON factory automation products.
- Microsoft, Windows, Excel, Visual Basic, and Microsoft Edge are either registered trademarks or trademarks of Microsoft Corporation in the United States and other countries.
- EtherCAT® is registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany.
- ODVA, CIP, CompoNet, DeviceNet, and EtherNet/IP are trademarks of ODVA.
- The SD and SDHC logos are trademarks of SD-3C, LLC.

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

Copyrights

- Microsoft product screen shots used with permission from Microsoft.
- 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/.

Introduction

Thank you for purchasing an NJ/NX-series CPU Unit.
This manual contains information that is necessary to use the Motion Control Function Module of an NJ/NX-series CPU Unit. Please read this manual and make sure you understand the functionality and performance of the NJ/NX-series CPU Unit before you attempt to use it in a control system.
Keep this manual in a safe place where it will be available for reference during operation.

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 B 3503.

Applicable Products

This manual covers the following products.
NX-series CPU Units

- NX701-1□□□
- NX502-1□□□
- NX102-1□□□
- NX102-90□□
- NX1P2-1□□□□□
- NX1P2-9□□□□□

NJ-series CPU Units

- NJ501-□□□□
- NJ301-1□□□
- NJ101-10□□

Part of the specifications and restrictions for the CPU Units are given in other manuals. Refer to *Relevant Manuals* on page 2 and *Related Manuals* on page 22.

Relevant Manuals

The following table provides the relevant manuals for the NJ/NX-series CPU Units. Read all of the manuals that are relevant to your system configuration and application before you use the NJ/NX-series CPU Unit.

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.

Purpose of use	Manual																			
	Basic information																			
	NX-series CPU Unit Hardware User's Manual	NX-series NX502 CPU Unit Hardware User's Manual	NX-series NX102 CPU Unit Hardware User's Manual	NX-series NX1P2 CPU Unit Hardware User's Manual	NJ-series CPU Unit Hardware User's Manual	NJ/NX-series CPU Unit Software User's Manual	NX-series NX1P2 CPU Unit Built-in I/O and Option Board User's Manual	NJ/NX-series Instructions Reference Manual	NJ/NX-series CPU Unit Motion Control User's Manual	NJ/NX-series Motion Control Instructions Reference Manual	NJ/NX-series CPU Unit Built-in EtherCAT Port User's Manual	NJ/NX-series CPU Unit Built-in EtherNet/IP Port User's Manual	NJ/NX-series CPU Unit OPC UA User's Manual	NX-series CPU Unit FINS User's Manual	NJ/NX-series Database Connection CPU Units User's Manual	NJ-series SECS/GEM CPU Units User's Manual	NJ-series Robot Integrated CPU Unit User's Manual	NJ-series NJ Robotics CPU Unit User's Manual	NJ/NY-series NC Integrated Controller User's Manual	NJ/NX-series Troubleshooting Manual
Introduction to NX701 CPU Units	○																			
Introduction to NX502 CPU Units		○																		
Introduction to NX102 CPU Units			○																	
Introduction to NX1P2 CPU Units				○																
Introduction to NJ-series Controllers					○															
Setting devices and hardware																				
Using motion control	○	○	○	○	○				○											
Using EtherCAT											○									
Using EtherNet/IP												○								
Using robot control for OMRON robots																	○			

Purpose of use	Manual																							
	Basic information																							
	NX-series CPU Unit Hardware User's Manual	NX-series NX102 CPU Unit Hardware User's Manual	NX-series NX502 CPU Unit Hardware User's Manual	NX-series NX1P2 CPU Unit Hardware User's Manual	NJ-series CPU Unit Hardware User's Manual	NJ/NX-series CPU Unit Software User's Manual	NX-series NX1P2 CPU Unit Built-in I/O and Option Board User's Manual	NJ/NX-series Instructions Reference Manual	NJ/NX-series CPU Unit Motion Control User's Manual	NJ/NX-series Motion Control Instructions Reference Manual	NJ/NX-series CPU Unit Built-in EtherCAT Port User's Manual	NJ/NX-series CPU Unit Built-in EtherNet/IP Port User's Manual	NJ/NX-series CPU Unit User's Manual	NX-series CPU Unit FINS User's Manual	NJ/NX-series Database Connection CPU Units User's Manual	NJ-series SECS/GEM CPU Units User's Manual	NJ-series Robot Integrated CPU Unit User's Manual	NJ-series NJ Robotics CPU Unit User's Manual	NJ/NY-series NC Integrated Controller User's Manual	NJ/NX-series Troubleshooting Manual
Software settings						○																		
Using motion control									○															
Using EtherCAT											○													
Using EtherNet/IP												○												
Using OPC UA													○											
Using FINS														○										
Using the database connection service															○									
Using the GEM Services						○										○								
Using robot control for OMRON robots																	○							
Using robot control by NJ Robotics function																		○						
Using numerical control																			○					
Using the NX1P2 CPU Unit functions							○																	
Writing the user program						○		○																
Using motion control									○	○														
Using EtherCAT											○													
Using EtherNet/IP												○												
Using OPC UA													○											
Using FINS														○										
Using the database connection service															○									
Using the GEM Services						○										○								
Using robot control for OMRON robots																	○							
Using robot control by NJ Robotics function																		○						
Using numerical control																			○					
Programming error processing																								○
Using the NX1P2 CPU Unit functions							○																	

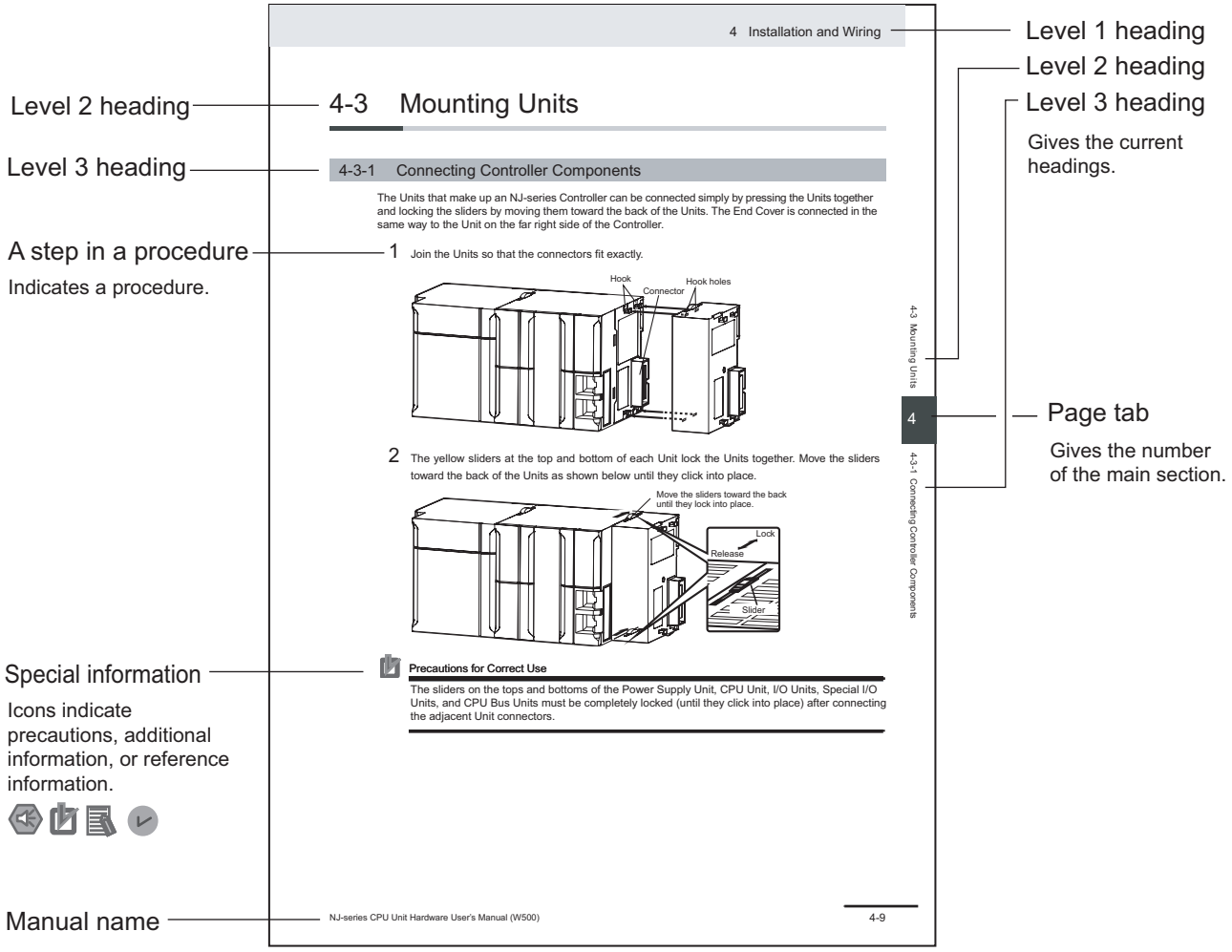
Purpose of use	Manual																				
	Basic information																				
	NX-series CPU Unit Hardware User's Manual	NX-series NX502 CPU Unit Hardware User's Manual	NX-series NX102 CPU Unit Hardware User's Manual	NX-series NX1P2 CPU Unit Hardware User's Manual	NJ-series CPU Unit Hardware User's Manual	NJ/NX-series CPU Unit Software User's Manual	NX-series NX1P2 CPU Unit Built-in I/O and Option Board User's Manual	NJ/NX-series Instructions Reference Manual	NJ/NX-series CPU Unit Motion Control User's Manual	NJ/NX-series Motion Control Instructions Reference Manual	NJ/NX-series CPU Unit Built-in EtherCAT Port User's Manual	NJ/NX-series CPU Unit Built-in EtherNet/IP Port User's Manual	NJ/NX-series CPU Unit OPC UA User's Manual	NX-series CPU Unit FINS User's Manual	NJ/NX-series Database Connection CPU Units User's Manual	NJ-series SECS/GEM CPU Units User's Manual	NJ-series Robot Integrated CPU Unit User's Manual	NJ-series NJ Robotics CPU Unit User's Manual	NJ/NY-series NC Integrated Controller User's Manual	NJ/NX-series Troubleshooting Manual
Testing operation and debugging						○															
Using motion control									○												
Using EtherCAT											○										
Using EtherNet/IP												○									
Using OPC UA													○								
Using FINS														○							
Using the database connection service															○						
Using the GEM Services																○					
Using robot control for OMRON robots																	○				
Using robot control by NJ Robotics function																		○			
Using numerical control																			○		
Using the NX1P2 CPU Unit functions							○														
Learning about error management and corrections*1													△	△	△	△	△	△	△	○	
Maintenance	○	○	○	○	○																
Using motion control									○												
Using EtherCAT											○										
Using EtherNet/IP												○									

*1. Refer to the *NJ/NX-series Troubleshooting Manual (Cat. No. W503)* for the error management concepts and the error items. However, refer to the manuals that are indicated with triangles for details on errors corresponding to the products with the manuals that are indicated with triangles.

Manual Structure

Page Structure

The following page structure is used in this manual.



This illustration is provided only as a sample. It may not literally appear 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.

 **Version Information**

Information on differences in specifications and functionality for Controller with different unit versions and for different versions of the Sysmac Studio is given.

Precaution on Terminology

In this manual, "download" refers to transferring data from the Sysmac Studio to the physical Controller and "upload" refers to transferring data from the physical Controller to the Sysmac Studio.
For the Sysmac Studio, "synchronization" is used to both "upload" and "download" data. Here, "synchronize" means to automatically compare the data for the Sysmac Studio on the computer with the data in the physical Controller and transfer the data in the direction that is specified by the user.

Sections in this Manual

CONTENTS

Section 9 Motion Control Functions

Section 10 Sample Programming

Section 11 Troubleshooting

Appendices

Index

Terms and Conditions Agreement

Warranty, Limitations of Liability

Warranties

- **Exclusive Warranty**

  Omron's exclusive warranty is that the Products will be free from defects in materials and workmanship for a period of twelve months from the date of sale by Omron (or such other period expressed in writing by Omron). Omron disclaims all other warranties, express or implied.

- **Limitations**

  OMRON MAKES NO WARRANTY OR REPRESENTATION, EXPRESS OR IMPLIED, ABOUT NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE OF THE PRODUCTS. BUYER ACKNOWLEDGES THAT IT ALONE HAS DETERMINED THAT THE PRODUCTS WILL SUITABLY MEET THE REQUIREMENTS OF THEIR INTENDED USE.

  Omron further disclaims all warranties and responsibility of any type for claims or expenses based on infringement by the Products or otherwise of any intellectual property right.

- **Buyer Remedy**

  Omron's sole obligation hereunder shall be, at Omron's election, to (i) replace (in the form originally shipped with Buyer responsible for labor charges for removal or replacement thereof) the non-complying Product, (ii) repair the non-complying Product, or (iii) repay or credit Buyer an amount equal to the purchase price of the non-complying Product; provided that in no event shall Omron be responsible for warranty, repair, indemnity or any other claims or expenses 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. Return of any Products by Buyer must be approved in writing by Omron before shipment. Omron Companies shall not be liable for the suitability or unsuitability or the results from the use of Products in combination with any electrical or electronic components, circuits, system assemblies or any other materials or substances or environments. Any advice, recommendations or information given orally or in writing, are not to be construed as an amendment or addition to the above warranty.

  See https://www.omron.com/global/ or contact your Omron representative for published information.

Limitation on Liability; Etc

OMRON COMPANIES SHALL NOT BE LIABLE FOR SPECIAL, INDIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, LOSS OF PROFITS OR PRODUCTION OR COMMERCIAL LOSS IN ANY

WAY CONNECTED WITH THE PRODUCTS, WHETHER SUCH CLAIM IS BASED IN CONTRACT, WARRANTY, NEGLIGENCE OR STRICT LIABILITY.

Further, in no event shall liability of Omron Companies exceed the individual price of the Product on which liability is asserted.

Application Considerations

Suitability of Use

Omron Companies shall not be responsible for conformity with any standards, codes or regulations which apply to the combination of the Product in the Buyer's application or use of the Product. At Buyer's request, Omron will provide applicable third party certification documents identifying ratings and limitations of use which apply to the Product. This information by itself is not sufficient for a complete determination of the suitability of the Product in combination with the end product, machine, system, or other application or use. Buyer shall be solely responsible for determining appropriateness of the particular Product with respect to Buyer's application, product or system. Buyer shall take application responsibility in all cases.

NEVER USE THE PRODUCT FOR AN APPLICATION INVOLVING SERIOUS RISK TO LIFE OR PROPERTY OR IN LARGE QUANTITIES WITHOUT ENSURING THAT THE SYSTEM AS A WHOLE HAS BEEN DESIGNED TO ADDRESS THE RISKS, AND THAT THE OMRON PRODUCT(S) IS PROPERLY RATED AND INSTALLED FOR THE INTENDED USE WITHIN THE OVERALL EQUIPMENT OR SYSTEM.

Programmable Products

Omron Companies shall not be responsible for the user's programming of a programmable Product, or any consequence thereof.

Disclaimers

Performance Data

Data presented in Omron Company websites, catalogs and other materials 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 user must correlate it to actual application requirements. Actual performance is subject to the Omron's Warranty and Limitations of Liability.

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 part numbers when published ratings or features are changed, or when significant construction changes are made. However, some specifications of the Product may

be changed without any notice. When in doubt, special part numbers may be assigned to fix or establish key specifications for your application. Please consult with your Omron's representative at any time to confirm actual specifications of purchased Product.

Errors and Omissions

Information presented by Omron Companies has been checked and is believed to be accurate; however, no responsibility is assumed for clerical, typographical or proofreading errors or omissions.

Statement of security responsibilities for assumed use cases and against threats

OMRON SHALL NOT BE RESPONSIBLE AND/OR LIABLE FOR ANY LOSS, DAMAGE, OR EXPENSES DIRECTLY OR INDIRECTLY RESULTING FROM THE INFECTION OF OMRON PRODUCTS, ANY SOFTWARE INSTALLED THEREON OR ANY COMPUTER EQUIPMENT, COMPUTER PROGRAMS, NETWORKS, DATABASES OR OTHER PROPRIETARY MATERIAL CONNECTED THERETO BY DISTRIBUTED DENIAL OF SERVICE ATTACK, COMPUTER VIRUSES, OTHER TECHNOLOGICALLY HARMFUL MATERIAL AND/OR UNAUTHORIZED ACCESS.

It shall be the users sole responsibility to determine and use adequate measures and checkpoints to satisfy the users particular requirements for (i) antivirus protection, (ii) data input and output, (iii) maintaining a means for reconstruction of lost data, (iv) preventing Omron Products and/or software installed thereon from being infected with computer viruses and (v) protecting Omron Products from unauthorized access.

Safety Precautions

Refer to the following manuals for safety precautions.

- *NX-series CPU Unit Hardware User's Manual (Cat. No. W535)*
- *NX-series NX502 CPU Unit Hardware User’s Manual (Cat. No. W629)*
- *NX-series NX102 CPU Unit Hardware User's Manual (Cat. No. W593)*
- *NX-series NX1P2 CPU Unit Hardware User's Manual (Cat. No. W578)*
- *NJ-series CPU Unit Hardware User’s Manual (Cat No. W500)*

Precautions for Safe Use

Refer to the following manuals for precautions for safe use.

- *NX-series CPU Unit Hardware User's Manual (Cat. No. W535)*
- *NX-series NX502 CPU Unit Hardware User's Manual (Cat. No. W629)*
- *NX-series NX102 CPU Unit Hardware User's Manual (Cat. No. W593)*
- *NX-series NX1P2 CPU Unit Hardware User's Manual (Cat. No. W578)*
- *NJ-series CPU Unit Hardware User's Manual (Cat No. W500)*

Precautions for Correct Use

Refer to the following manuals for precautions for correct use.

- *NX-series CPU Unit Hardware User's Manual (Cat. No. W535)*
- *NX-series NX502 CPU Unit Hardware User’s Manual (Cat. No. W629)*
- *NX-series NX102 CPU Unit Hardware User's Manual (Cat. No. W593)*
- *NX-series NX1P2 CPU Unit Hardware User's Manual (Cat. No. W578)*
- *NJ-series CPU Unit Hardware User’s Manual (Cat No. W500)*

Regulations and Standards

Refer to the following manuals for regulations and standards.

- *NX-series CPU Unit Hardware User's Manual (Cat. No. W535)*
- *NX-series NX502 CPU Unit Hardware User’s Manual (Cat. No. W629)*
- *NX-series NX102 CPU Unit Hardware User's Manual (Cat. No. W593)*
- *NX-series NX1P2 CPU Unit Hardware User's Manual (Cat. No. W578)*
- *NJ-series CPU Unit Hardware User’s Manual (Cat No. W500)*

Versions

Hardware revisions and unit versions are used to manage the hardware and software in NJ/NX-series Units and EtherCAT slaves. The hardware revision or unit version is updated each time there is a change in hardware or software specifications. Even when two Units or EtherCAT slaves have the same model number, they will have functional or performance differences if they have different hardware revisions or unit versions.

Refer to the following manuals for versions.

- *NX-series CPU Unit Hardware User's Manual (Cat. No. W535)*
- *NX-series NX502 CPU Unit Hardware User's Manual (Cat. No. W629)*
- *NX-series NX102 CPU Unit Hardware User's Manual (Cat. No. W593)*
- *NX-series NX1P2 CPU Unit Hardware User's Manual (Cat. No. W578)*
- *NJ-series CPU Unit Hardware User's Manual (Cat No. W500)*

Unit Versions of CPU Units and Sysmac Studio Versions

The functions that are supported depend on the unit version of the NJ/NX-series CPU Unit. The version of Sysmac Studio that supports the functions that were added for an upgrade is also required to use those functions.

Refer to the *NJ/NX-series CPU Unit Software User's Manual (Cat. No. W501)* for the relationship between the unit versions of CPU Unit and the Sysmac Studio versions.

Refer to *A-7 Version Information* on page A-34 for the functions that are supported by each unit version.

Related Manuals

The following are the manuals related to this manual. Use these manuals for reference.

Manual name	Cat. No.	Model numbers	Application	Description
NX-series CPU Unit Hardware User's Manual	W535	NX701-□□□□	Learning the basic specifications of the NX701 CPU Units, including introductory information, designing, installation, and maintenance. Mainly hardware information is provided.	An introduction to the entire NX701 system is provided along with the following information on the CPU Unit. • Features and system configuration • Introduction • Part names and functions • General specifications • Installation and wiring • Maintenance and inspection
NX-series NX502 CPU Unit Hardware User's Manual	W629	NX502-□□□□	Learning the basic specifications of the NX502 CPU Units, including introductory information, designing, installation, and maintenance. Mainly hardware information is provided.	An introduction to the entire NX502 system is provided along with the following information on the CPU Unit. • Features and system configuration • Introduction • Part names and functions • General specifications • Installation and wiring • Maintenance and inspection
NX-series NX102 CPU Unit Hardware User's Manual	W593	NX102-□□□□	Learning the basic specifications of the NX102 CPU Units, including introductory information, designing, installation, and maintenance. Mainly hardware information is provided.	An introduction to the entire NX102 system is provided along with the following information on the CPU Unit. • Features and system configuration • Introduction • Part names and functions • General specifications • Installation and wiring • Maintenance and inspection
NX-series NX1P2 CPU Unit Hardware User's Manual	W578	NX1P2-□□□□	Learning the basic specifications of the NX1P2 CPU Units, including introductory information, designing, installation, and maintenance. Mainly hardware information is provided.	An introduction to the entire NX1P2 system is provided along with the following information on the CPU Unit. • Features and system configuration • Introduction • Part names and functions • General specifications • Installation and wiring • Maintenance and inspection
NJ-series CPU Unit Hardware User's Manual	W500	NJ501-□□□□ NJ301-□□□□ NJ101-□□□□	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 the CPU Unit. • Features and system configuration • Introduction • Part names and functions • General specifications • Installation and wiring • Maintenance and inspection
NJ/NX-series CPU Unit Software User's Manual	W501	NX701-□□□□ NX502-□□□□ NX102-□□□□ NX1P2-□□□□ NJ501-□□□□ NJ301-□□□□ NJ101-□□□□	Learning how to program and set up an NJ/NX-series CPU Unit. Mainly software information is provided.	The following information is provided on a Controller built with an NJ/NX-series CPU Unit. • CPU Unit operation • CPU Unit features • Initial settings • Programming based on IEC 61131-3 language specifications

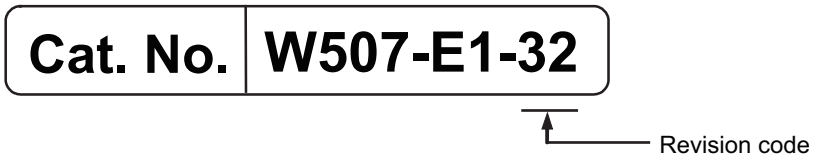
Manual name	Cat. No.	Model numbers	Application	Description
NX-series NX1P2 CPU Unit Built-in I/O and Option Board User's Manual	W579	NX1P2-□□□□	Learning about the details of functions only for an NX-series NX1P2 CPU Unit and an introduction of functions for an NJ/NX-series CPU Unit.	Of the functions for an NX1P2 CPU Unit, the following information is provided. • Built-in I/O • Serial Communications Option Boards • Analog I/O Option Boards An introduction of following functions for an NJ/NX-series CPU Unit is also provided. • Motion control functions • EtherNet/IP communications functions • EtherCAT communications functions
NJ/NX-series Instructions Reference Manual	W502	NX701-□□□□ NX502-□□□□ NX102-□□□□ NX1P2-□□□□ NJ501-□□□□ NJ301-□□□□ NJ101-□□□□	Learning detailed specifications on the basic instructions of an NJ/NX-series CPU Unit.	The instructions in the instruction set (IEC 61131-3 specifications) are described.
NJ/NX-series CPU Unit Motion Control User's Manual	W507	NX701-□□□□ NX502-□□□□ NX102-□□□□ NX1P2-□□□□ NJ501-□□□□ NJ301-□□□□ NJ101-□□□□	Learning about motion control settings and programming concepts.	The settings and operation of the CPU Unit and programming concepts for motion control are described.
NJ/NX-series Motion Control Instructions Reference Manual	W508	NX701-□□□□ NX502-□□□□ NX102-□□□□ NX1P2-□□□□ NJ501-□□□□ NJ301-□□□□ NJ101-□□□□	Learning about the specifications of the motion control instructions.	The motion control instructions are described.
NJ/NX-series CPU Unit Built-in EtherCAT® Port User's Manual	W505	NX701-□□□□ NX502-□□□□ NX102-□□□□ NX1P2-□□□□ NJ501-□□□□ NJ301-□□□□ NJ101-□□□□	Using the built-in EtherCAT port on an NJ/NX-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.
NJ/NX-series CPU Unit Built-in EtherNet/IP™ Port User's Manual	W506	NX701-□□□□ NX502-□□□□ NX102-□□□□ NX1P2-□□□□ NJ501-□□□□ NJ301-□□□□ NJ101-□□□□	Using the built-in EtherNet/IP port on an NJ/NX-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.
NJ/NX-series CPU Unit OPC UA User's Manual	W588	NX701-□□□□ NX502-□□□□ NX102-□□□□ NJ501-1□00	Using the OPC UA.	Describes the OPC UA.
NX-series CPU Unit FINS Function User's Manual	W596	NX701-□□20 NX502-□□□□ NX102-□□□□	Using the FINS function of an NX-series CPU Unit.	Describes the FINS function of an NX-series CPU Unit.
NJ/NX-series Database Connection CPU Units User's Manual	W527	NX701-□□20 NX502-□□□□ NX102-□□20 NJ501-□□20 NJ101-□□20	Using the database connection service with NJ/NX-series Controllers.	Describes the database connection service.
NJ-series SECS/GEM CPU Units User's Manual	W528	NJ501-1340	Using the GEM Services with NJ-series Controllers.	Provides information on the GEM Services.

Manual name	Cat. No.	Model numbers	Application	Description
NJ-series Robot Integrated CPU Unit User's Manual	O037	NJ501-R□□□	Using the NJ-series Robot Integrated CPU Unit.	Describes the settings and operation of the CPU Unit and programming concepts for OMRON robot control.
Sysmac Studio Robot Integrated System Building Function with Robot Integrated CPU Unit Operation Manual	W595	SYSMAC-SE2□□□ SYSMAC-SE200D-64	Learning about the operating procedures and functions of the Sysmac Studio to configure Robot Integrated System using Robot Integrated CPU Unit.	Describes the operating procedures of the Sysmac Studio for Robot Integrated CPU Unit.
Sysmac Studio Robot Integrated System Building Function with IPC Application Controller Operation Manual	W621	SYSMAC-SE2□□□ SYSMAC-SE200D-64	Learning about the operating procedures and functions of the Sysmac Studio to configure Robot Integrated System using IPC Application Controller.	Describes the operating procedures of the Sysmac Studio for IPC Application Controller.
Sysmac Studio 3D Simulation Function Operation Manual	W618	SYSMAC-SE2□□□ SYSMAC-SA4□□□-64	Learning about an outline of the 3D simulation function of the Sysmac Studio and how to use the function.	Describes an outline, execution procedures, and operating procedures for the 3D simulation function of the Sysmac Studio.
NJ-series NJ Robotics CPU Unit User's Manual	W539	NJ501-4□□□ NJ501-R□□□	Controlling robots with NJ-series CPU Units.	Describes the functionality to control robots.
NJ/NY-series NC Integrated Controller User's Manual	O030	NJ501-5300 NY532-5400	Performing numerical control with NJ/NY-series Controllers.	Describes the functionality to perform the numerical control.
NJ/NY-series G code Instructions Reference Manual	O031	NJ501-5300 NY532-5400	Learning about the specifications of the G code/M code instructions.	The G code/M code instructions are described.
NJ/NX-series Troubleshooting Manual	W503	NX701-□□□□ NX502-□□□□ NX102-□□□□ NX1P2-□□□□ NJ501-□□□□ NJ301-□□□□ NJ101-□□□□	Learning about the errors that may be detected in an NJ/NX-series Controller.	Concepts on managing errors that may be detected in an NJ/NX-series Controller and information on individual errors are described.
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.
CNC Operator Operation Manual	O032	SYSMAC-RTNC0□□□D	Learning an introduction of the CNC Operator and how to use it.	An introduction of the CNC Operator, installation procedures, basic operations, connection operations, and operating procedures for main functions are described.
NX-series EtherCAT® Coupler Unit User's Manual	W519	NX-ECC□□□	Learning how to use the NX-series EtherCAT Coupler Unit and EtherCAT Slave Terminals.	The following items are described: the overall system and configuration methods of an EtherCAT Slave Terminal (which consists of an NX-series EtherCAT Coupler Unit and NX Units), and information on hardware, setup, and functions to set up, control, and monitor NX Units through EtherCAT.

Manual name	Cat. No.	Model numbers	Application	Description
NX-series Data Reference Manual	W525	NX-□□□□□□	Referencing lists of the data that is required to configure systems with NX-series Units.	Lists of the power consumptions, weights, and other NX Unit data that is required to configure systems with NX-series Units are provided.
NX-series NX Units User's Manual	W521	NX-ID□□□□ NX-IA□□□□ NX-OC□□□□ NX-OD□□□□ NX-MD□□□□	Learning how to use NX Units.	Describes the hardware, setup methods, and functions of the NX Units. Manuals are available for the following Units. Digital I/O Units, Analog I/O Units, System Units, Position Interface Units, Communications Interface Units, Load Cell Input Unit, IO-Link Master Units, and High-speed Counter Units.
	W522	NX-AD□□□□ NX-DA□□□□		
	W592	NX-HAD□□□		
	W566	NX-TS□□□□ NX-HB□□□□		
	W523	NX-PD1□□□ NX-PF0□□□ NX-PC0□□□ NX-TBX01		
	W524	NX-EC0□□□ NX-ECS□□□ NX-PG0□□□		
	W540	NX-CIF□□□		
	W565	NX-RS□□□□		
	W567	NX-ILM□□□		
	W647	NX-CT□□□□		
GX-series EtherCAT Slave Units User's Manual	W488	GX-ID□□□□ GX-OD□□□□ GX-OC□□□□ GX-MD□□□□ GX-AD□□□□ GX-DA□□□□ GX-EC□□□□ XWT-ID□□ XWT-OD□□	Learning how to use the EtherCAT remote I/O terminals.	Describes the hardware, setup methods and functions of the EtherCAT remote I/O terminals.
AC Servomotors/Servo Drives 1S-series with Built-in EtherCAT® Communications User's Manual	I586	R88M-1□ R88D-1SN□-ECT	Learning how to use the Servomotors/Servo Drives with built-in EtherCAT Communications.	Describes the hardware, setup methods and functions of the Servomotors/Servo Drives with built-in EtherCAT Communications.
	I621	R88M-1AL□/ -1AM□ R88D-1SAN□-ECT		
AC Servomotors/Servo Drives G5 Series with Built-in EtherCAT® Communications User's Manual	I576	R88M-K□ R88D-KN□-ECT	Learning how to use the AC Servomotors/Servo Drives with built-in EtherCAT Communications.	Describes the hardware, setup methods and functions of the AC Servomotors/Servo Drives with built-in EtherCAT Communications. The Linear Motor Type models and dedicated models for position control are available in G5-series.
	I577	R88L-EC-□ R88D-KN□-ECT-L		

Revision History

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



Revision code	Date	Revised content
01	July 2011	Original production
02	January 2012	• Added information on the NJ301-□□□□. • Made changes accompanying release of unit version 1.01 of the CPU Unit.
03	May 2012	• Made changes accompanying release of unit version 1.02 of the CPU Unit. • Corrected mistakes.
04	August 2012	Made changes accompanying release of unit version 1.03 of the CPU Unit.
05	February 2013	Made changes accompanying release of unit version 1.04 of the CPU Unit.
06	April 2013	• Made changes accompanying release of unit version 1.05 of the CPU Unit. • Corrected mistakes.
07	June 2013	• Made changes accompanying release of unit version 1.06 of the CPU Unit. • Corrected mistakes.
08	December 2013	• Made changes accompanying release of unit version 1.08 of the CPU Unit • Corrected mistakes.
09	July 2014	• Made changes accompanying release of unit version 1.09 of the CPU Unit. • Corrected mistakes.
10	January 2015	• Made changes accompanying release of unit version 1.10 of the CPU Unit. • Corrected mistakes.
11	January 2015	• Made changes accompanying release of the NX-series NX701-□□□□ CPU Unit and the NJ-series NJ101-10□□ CPU Unit. • Corrected mistakes.
12	April 2016	• Made changes accompanying release of unit version 1.11 of the CPU Unit. • Corrected mistakes.
13	July 2016	• Made changes accompanying release of unit version 1.12 of the CPU Unit. • Corrected mistakes.
14	October 2016	• Made changes accompanying addition of NX-series NX1P2 CPU Units. • Made changes accompanying release of unit version 1.13 of the CPU Unit. • Corrected mistakes.
15	January 2017	Corrected mistakes.
16	October 2017	Corrected mistakes.
17	April 2018	• Made changes accompanying addition of NX-series NX102 CPU Units. • Made changes accompanying the transfer of explanation for event codes and errors to the NJ/NX-series Troubleshooting Manual. • Corrected mistakes.
18	July 2018	Corrected mistakes.
19	January 2019	Corrected mistakes.

Revision code	Date	Revised content
20	April 2019	• Made changes accompanying addition of the 1S-series AC Servomotor/ Servo Drive models. • Made changes accompanying addition of functions.
21	July 2019	Corrected mistakes.
22	October 2019	• Made changes accompanying addition of NX1P2-9B□□□□. • Corrected mistakes.
23	August 2020	Made changes accompanying addition of NJ501-R□□□.
24	April 2022	Added information to Terms and Conditions Agreement.
25	April 2023	Made changes accompanying addition of NX502-1□□□.
26	July 2023	Corrected mistakes.
27	December 2023	Made changes accompanying the upgrade to Sysmac Studio version 1.57.
28	April 2024	Made changes accompanying the addition of the NX502-1700 and NX502-1600.
29	October 2024	Corrected mistakes.
30	April 2025	Corrected mistakes.
31	July 2025	Made changes accompanying the addition of the MC_SetCamTableEndPointIndex instruction.
32	April 2026	• Made changes accompanying the addition of the MC_CamInCurve and the MC_CamMonitorCurve instructions. • Made changes accompanying the addition of the MC_SyncTorqueControl instruction. • Made changes accompanying the addition of the MC_GearInPos2 instruction. • Made changes accompanying the addition of the MC_SetTorqueLimit2 instruction. • Made changes accompanying the changes to the zero position preset specifications.

1

Introduction to the Motion Control Function Module

This section describes the features, system configuration, and application flow for the Motion Control Function Module.

1-1 Features

The Motion Control Function Module (sometimes abbreviated to “MC Function Module”) is a software function module that is built into the CPU Unit.
The MC Function Module can perform motion control for up to 256 axes through the EtherCAT port that is built into the CPU Unit.
Cyclic communications are performed with Servo Drives and other devices that are connected to the EtherCAT port to enable high-speed, high-precision machine control.

Motion Control Instructions Based on PLCopen®

The motion control instructions of the MC Function Module are based on motion control function blocks that are standardized by PLCopen®.
These instructions allow you to program single-axis PTP positioning, interpolation control, synchronized control (e.g., of electronic cams), velocity control, and torque control.
You can set the velocity, acceleration rate, deceleration rate, and jerk each time a motion control instruction is executed to flexibly control operation according to the application.

Additional Information

PLCopen®

PLCopen® is an association that promotes IEC 61131-3. It has its headquarters in Europe and a world-wide membership. PLCopen® standardizes function blocks for motion control to define a program interface for the languages specified in IEC 61131-3 (JIS B 3503).

Jerk

Jerk is the rate of change in the acceleration rate or deceleration rate. If you specify the jerk, the velocity graph will form an S-curve for acceleration and deceleration.

Data Transmission Using EtherCAT Communications

This function module can be combined with OMRON 1S-series Servo Drives with built-in EtherCAT communications or G5-series Servo Drives with built-in EtherCAT communications to enable exchange of all control information with high-speed data communications.
The various control commands are transmitted via data communications. That means that the Servomotor's operational performance is maximized without being limited by interface specifications, such as the response frequency of the encoder feedback pulses.
You can use the Servo Drive's various control parameters and monitor data on a host controller to unify management of system information.

Additional Information

What Is EtherCAT?

EtherCAT is an open high-speed industrial network system that conforms to Ethernet (IEEE 802.3). Each node achieves a short communications cycle time by transmitting Ethernet frames at high speed. A mechanism that allows sharing clock information enables high-precision synchronized control with low communications jitter.

1-2 System Configuration

The MC Function Module receives sensor signal status from devices and control panels. It receives commands from the motion control instructions that are executed in the user program. It uses both of these to perform motion control with the Servo Drives, Encoder Input Terminals, and NX-series Position Interface Units.

Motion Control Configuration on EtherCAT Network

The EtherCAT network configuration, the Slave Terminal configurations for EtherCAT Coupler Units, and the Sysmac Studio are used for the MC Function Module.

● EtherCAT Network Configuration

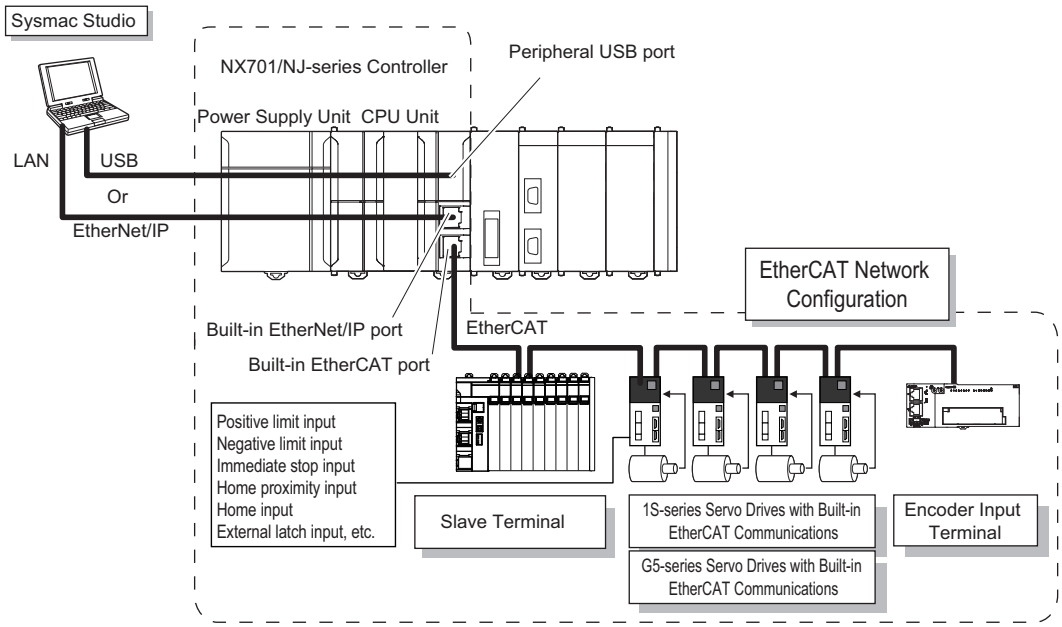
The MC Function Module performs control for Servo Drives and Encoder Input Terminals through the EtherCAT master port that is built into the CPU Unit.
The EtherCAT network configuration is used to perform precise motion control in a fixed period with very little deviation.

● Slave Terminal Configurations of EtherCAT Coupler Units

The MC Function Module uses the Position Interface Units that are mounted under an EtherCAT Coupler Unit to output motor control pulses and read encoder inputs.
You can also use this configuration to perform precise motion control in a fixed period with very little deviation.

● Sysmac Studio

The Sysmac Studio is connected to the peripheral USB port on the CPU Unit with a commercially available USB cable. You can also connect it to the built-in EtherNet/IP port on the CPU Unit with Ethernet cable.



Precautions for Correct Use

Some of the functions of the MC Function Module are different when NX-series Position Interface Units are used. Refer to the *NX-series Position Interface Units User's Manual (Cat. No. W524)* for details.

Version Information

A CPU Unit with unit version 1.05 or later and Sysmac Studio version 1.06 or higher are required to use the NX-series Position Interface Units.

Motion Control Configuration on CPU Unit

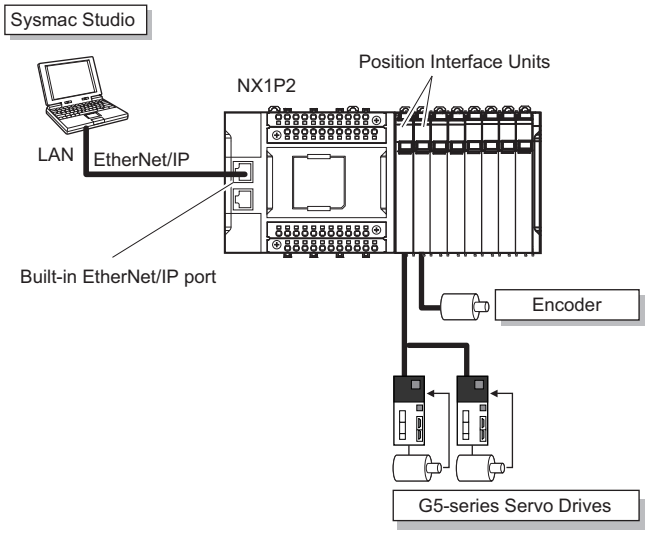
The Position Interface Unit and Sysmac Studio are used for the MC Function Module.

● Position Interface Unit

The MC Function Module uses the Position Interface Units to output motor control pulses and read encoder inputs.

● Sysmac Studio

The Sysmac Studio is connected to the built-in EtherNet/IP port on the CPU Unit with an Ethernet cable.

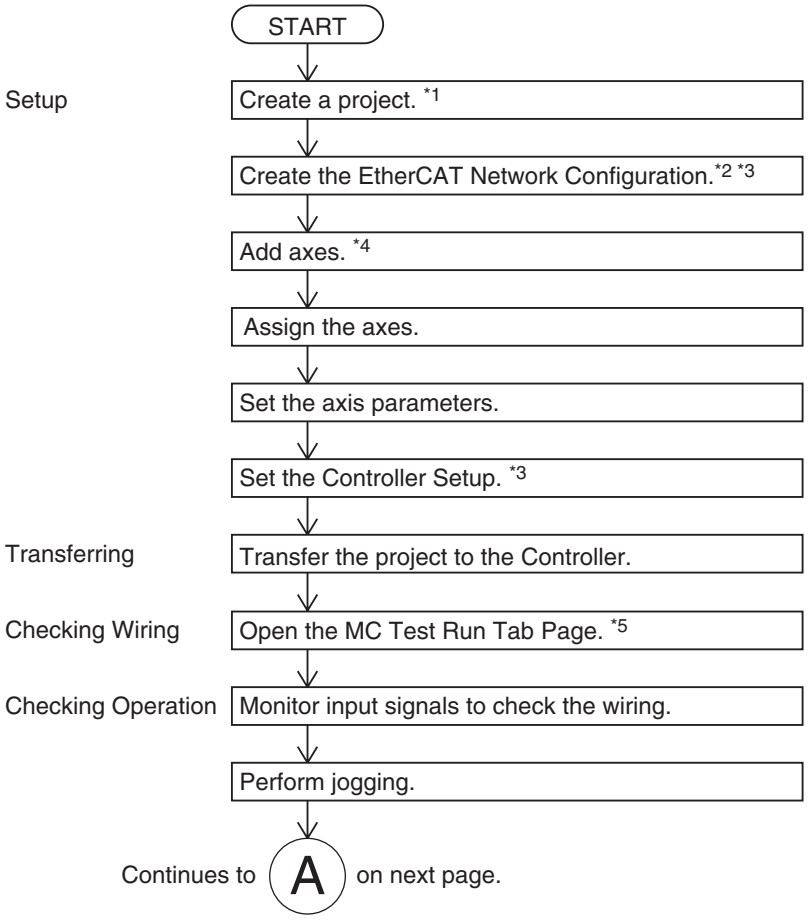


Precautions for Correct Use

Some of the functions of the MC Function Module are different when NX-series Position Interface Units are used. Refer to the *NX-series Position Interface Units User's Manual (Cat. No. W524)* for details.

1-3 Basic Flow of Operation

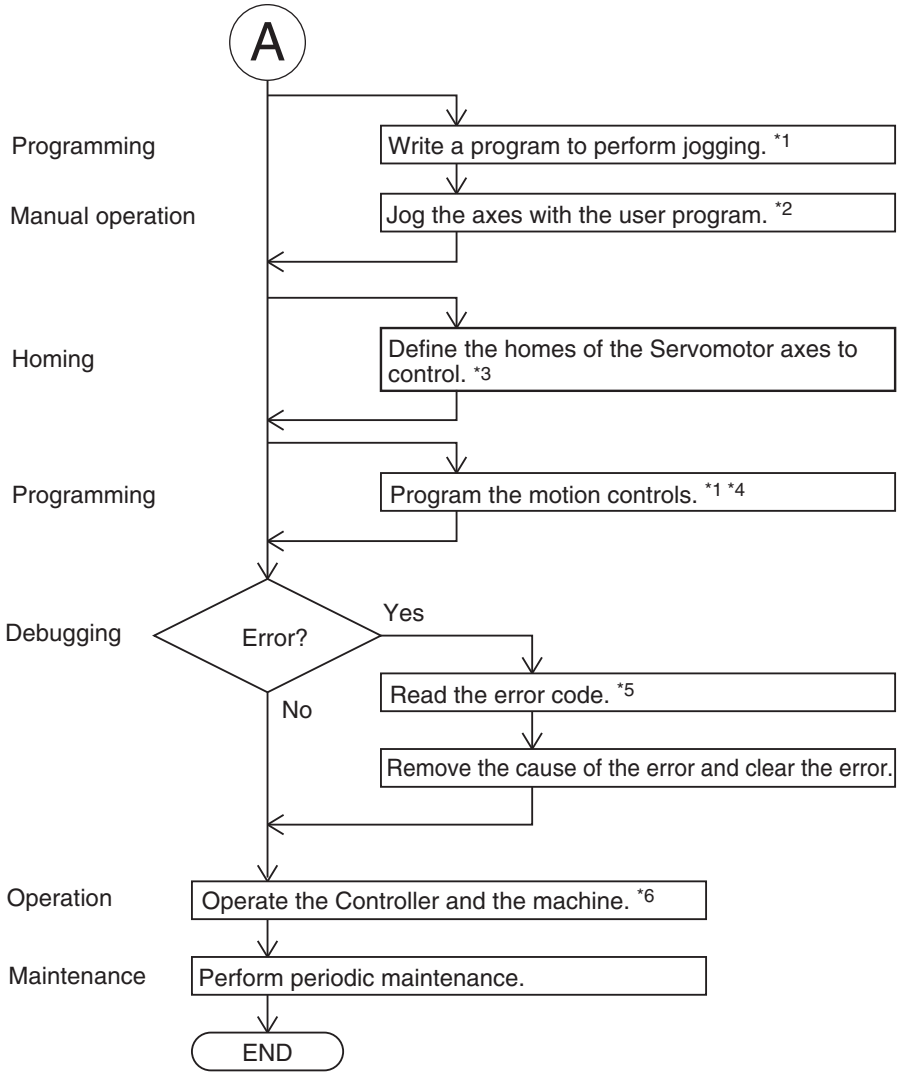
This section provides the basic procedure to perform motion control with the MC Function Module.



*1. Refer to *Sysmac Studio Version 1 Operation Manual (Cat. No. W504)*.
*2. If actual slave devices are connected, you can apply the actual EtherCAT network configuration to the configuration setup online.
If no slave devices are connected, you can select EtherCAT slave devices offline to set up the EtherCAT network configuration.
*3. Refer to *NJ/NX-series CPU Unit Software User's Manual (Cat. No. W501)*.
*4. Refer to *Section 3 Configuring Axes and Axes Groups* on page 3-1.
*5. Refer to *Section 4 Checking Wiring from the Sysmac Studio* on page 4-1.

Additional Information

Refer to the *NX-series Position Interface Units User's Manual (Cat. No. W524)* for the procedures for the NX-series Position Interface Units.



*1. Refer to *Section 6 Motion Control Programming* on page 6-1.
*2. Refer to *Section 7 Manual Operation* on page 7-1.
*3. Refer to *Section 8 Homing* on page 8-1.
*4. Refer to *Section 10 Sample Programming* on page 10-1.
*5. Refer to *Section 11 Troubleshooting* on page 11-1.
*6. Refer to *Section 9 Motion Control Functions* on page 9-1.

1-4 Specifications

This section gives the specifications of the MC Function Module.

Precautions for Correct Use

The NX102-90□□ Units and NJ101-90□□ Units do not provide the Motion Control Function Module.

1-4-1 General Specifications

General specifications conform to the general specifications of the CPU Unit.
Refer to the following manuals for details.

- *NX-series CPU Unit Hardware User's Manual (Cat. No. W535)*
- *NX-series NX502 CPU Unit Hardware User's Manual (Cat. No. W629)*
- *NX-series NX102 CPU Unit Hardware User's Manual (Cat. No. W593)*
- *NX-series NX1P2 CPU Unit Hardware User's Manual (Cat. No. W578)*
- *NJ-series CPU Unit Hardware User's Manual (Cat. No. W500)*

1-4-2 Performance Specifications

The following table describes the performance specifications for each type of CPU Unit.

Item			NX701-	
			17□□	16□□
Number of controlled axes	Maximum number of controlled axes*1		256 axes	128 axes
		Motion control axes*2	256 axes	128 axes
		Single-axis position control axes*3	---	
	Maximum number of used real axes*4		256 axes	128 axes
		Used motion control servo axes	256 axes	128 axes
		Used single-axis position control servo axes	---	
	Maximum number of axes for linear interpolation axis control		4 axes per axes group	
	Number of axes for circular interpolation axis control		2 axes per axes group	
Maximum number of axes groups			64 axes groups	
Override factors			0.00% or 0.01% to 500.00%	
Motion control period			The same control period as that is used for the process data communications cycle for EtherCAT.	
Multi-motion			Supported.	
Cams	Number of cam data points	Maximum points per cam table	65,535 points	
		Maximum points for all cam tables	1,048,560 points	
	Maximum number of cam tables		640 tables	

*1. This is the total for all axis types. Refer to *Axis Types* on page 5-10 for details on axis types.
*2. This means the axes for which all motion control functions can be used.

*3. This means the axes for which only single-axis position control can be used among motion control functions. Refer to *Control Function* on page 5-11 for details on a single-axis position control axis.
*4. This is the total number of axes whose axis type is set to **Servo Axis** or **Encoder Axis** and axis use is set to **Used axis**.

Item			NX502-				
			17□□*1	16□□*1	15□□	14□□	13□□
Number of controlled axes	Maximum number of controlled axes*2		256 axes	128 axes	128 axes	64 axes	32 axes
		Motion control axes*3	256 axes	128 axes	128 axes	64 axes	32 axes
		Single-axis position control axes*4	---				
	Maximum number of used real axes*5		256 axes	128 axes	64 axes	32 axes	16 axes
		Used motion control servo axes	256 axes	128 axes	64 axes	32 axes	16 axes
		Used single-axis position control servo axes	---				
	Maximum number of axes for linear interpolation axis control		4 axes per axes group				
	Number of axes for circular interpolation axis control		2 axes per axes group				
Maximum number of axes groups			64 axes groups		32 axes groups		
Override factors			0.00%, or 0.01% to 500.00%				
Motion control period			The same control period as that is used for the process data communications cycle for EtherCAT.				
Multi-motion			Not supported.				
Cams	Number of cam data points	Maximum points per cam table	65,535 points				
		Maximum points for all cam tables	1,048,560 points				
	Maximum number of cam tables		640 tables				

*1. Models added from the CPU Unit version 1.66.
*2. This is the total for all axis types. Refer to *Axis Types* on page 5-10 for details on axis types.
*3. This means the axes for which all motion control functions can be used.
*4. This means the axes for which only single-axis position control can be used among motion control functions. Refer to *Control Function* on page 5-11 for details on a single-axis position control axis.
*5. This is the total number of axes whose axis type is set to **Servo Axis** or **Encoder Axis** and axis use is set to **Used axis**.

Item			NX102-			
			12□□	11□□	10□□	90□□
Number of controlled axes	Maximum number of controlled axes*1		15 axes			4 axes
		Motion control axes*2	11 axes			---
		Single-axis position control axes*3	4 axes			
	Maximum number of used real axes*4		12 axes	8 axes	6 axes	4 axes
		Used motion control servo axes	8 axes	4 axes	2 axes	---
		Used single-axis position control servo axes	4 axes			
	Maximum number of axes for linear interpolation axis control		4 axes per axes group			---
	Number of axes for circular interpolation axis control		2 axes per axes group			---
Maximum number of axes groups			8 axes groups			---
Override factors			0.00% or 0.01% to 500.00%			
Motion control period			Same as the period for primary periodic task			
Multi-motion			Not supported.			
Cams	Number of cam data points	Maximum points per cam table	65,535 points			
		Maximum points for all cam tables	262,140 points			
	Maximum number of cam tables		160 tables			

*1. This is the total for all axis types. Refer to *Axis Types* on page 5-10 for details on axis types.

*2. This means the axes for which all motion control functions can be used.

*3. This means the axes for which only single-axis position control can be used among motion control functions. Refer to *Control Function* on page 5-11 for details on a single-axis position control axis.

*4. This is the total number of axes whose axis type is set to **Servo Axis** or **Encoder Axis** and axis use is set to **Used Axis**.

Item			NX1P2-			
			11□□	10□□	90□□	9B□□
Number of controlled axes	Maximum number of controlled axes*1		12 axes	10 axes	4 axes	2 axes
		Motion control axes*2	8 axes	6 axes	---	
		Single-axis position control axes*3	4 axes			2 axes
	Maximum number of used real axes*4		8 axes	6 axes	4 axes	2 axes
		Used motion control servo axes	4 axes	2 axes	---	
		Used single-axis position control servo axes	4 axes			2 axes
	Maximum number of axes for linear interpolation axis control		4 axes per axes group		---	
	Number of axes for circular interpolation axis control		2 axes per axes group		---	
Maximum number of axes groups			8 axes groups		---	
Override factors			0.00% or 0.01% to 500.00%			
Motion control period			Same as the period for primary periodic task			
Multi-motion			Not supported.			
Cams	Number of cam data points	Maximum points per cam table	65,535 points		---	
		Maximum points for all cam tables	262,140 points		---	
	Maximum number of cam tables		80 tables		---	

*1. This is the total for all axis types. Refer to *Axis Types* on page 5-10 for details on axis types.

*2. This means the axes for which all motion control functions can be used.

*3. This means the axes for which only single-axis position control can be used among motion control functions. Refer to *Control Function* on page 5-11 for details on a single-axis position control axis.

*4. This is the total number of axes whose axis type is set to **Servo Axis** or **Encoder Axis** and axis use is set to **Used axis**.

Item			NJ501-		
			□5□□	□4□□	□3□□
Number of controlled axes	Maximum number of controlled axes*1		64 axes	32 axes	16 axes
		Motion control axes*2	64 axes	32 axes	16 axes
		Single-axis position control axes*3	---		
	Maximum number of used real axes*4		64 axes	32 axes	16 axes
		Used motion control servo axes	64 axes	32 axes	16 axes
		Used single-axis position control servo axes	---		
	Maximum number of axes for linear interpolation axis control		4 axes per axes group		
	Number of axes for circular interpolation axis control		2 axes per axes group		
Maximum number of axes groups			32 axes groups		
Override factors			0.00% or 0.01% to 500.00%		
Motion control period			The same control period as that is used for the process data communications cycle for EtherCAT.		
Multi-motion			Not supported.		
Cams	Number of cam data points	Maximum points per cam table	65,535 points		
		Maximum points for all cam tables	1,048,560 points		
	Maximum number of cam tables		640 tables		

*1. This is the total for all axis types. Refer to *Axis Types* on page 5-10 for details on axis types.
*2. This means the axes for which all motion control functions can be used.
*3. This means the axes for which only single-axis position control can be used among motion control functions. Refer to *Control Function* on page 5-11 for details on a single-axis position control axis.
*4. This is the total number of axes whose axis type is set to **Servo Axis** or **Encoder Axis** and axis use is set to **Used Axis**.

Item			NJ301-		NJ101-
			12□□	11□□	10□□
Number of controlled axes	Maximum number of controlled axes*1		15 axes*2 *3	15 axes*2 *4	6 axes
		Motion control axes*5	15 axes*2 *3	15 axes*2 *4	6 axes
		Single-axis position control axes*6	---		
	Maximum number of used real axes*7		8 axes	4 axes	2 axes
		Used motion control servo axes	8 axes	4 axes	2 axes
		Used single-axis position control servo axes	---		
	Maximum number of axes for linear interpolation axis control		4 axes per axes group		
	Number of axes for circular interpolation axis control		2 axes per axes group		
Maximum number of axes groups			32 axes groups		
Override factors			0.00% or 0.01% to 500.00%		
Motion control period			The same control period as that is used for the process data communications cycle for EtherCAT.		
Multi-motion			Not supported.		
Cams	Number of cam data points	Maximum points per cam table	65,535 points		
		Maximum points for all cam tables	262,140 points		
	Maximum number of cam tables		160 tables		

*1. This is the total for all axis types. Refer to *Axis Types* on page 5-10 for details on axis types.
*2. Functions with asterisks were added for an upgraded version of the CPU Unit. Refer to *A-7 Version Information* on page A-34 for information on version upgrades.
*3. This number of axes is achieved in a combination of a CPU Unit with unit version 1.06 or later and Sysmac Studio version 1.07 or higher. In other combinations, this is 8 axes.
*4. This number of axes is achieved in a combination of a CPU Unit with unit version 1.06 or later and Sysmac Studio version 1.07 or higher. In other combinations, this is 4 axes.
*5. This means the axes for which all motion control functions can be used.
*6. This means the axes for which only single-axis position control can be used among motion control functions. Refer to *Control Function* on page 5-11 for details on a single-axis position control axis.
*7. This is the total number of axes whose axis type is set to **Servo Axis** or **Encoder Axis** and axis use is set to **Used Axis**.

1-4-3 Function Specifications

The following table describes the functions that are supported for connections to OMRON control devices.

Item	Description
Controllable Servo Drives	OMRON 1S-series Servo Drives with built-in EtherCAT communications or G5-series Servo Drives with built-in EtherCAT communications*1 *2
Controllable encoder input terminals	OMRON GX-series GX-EC0211/EC0241 EtherCAT Remote I/O Terminals*3

Item			Description
Controllable Position Interface Units[*2*4]			OMRON NX-EC0□□□ Incremental Encoder Input Units OMRON NX-ECS□□□ SSI Input Units OMRON NX-PG0□□□ Pulse Output Units
Control method			Control commands using EtherCAT communications
Control modes			Position control, Velocity control, and Torque control
Unit conversions		Position units	Pulse, mm, μm, nm, degree, and inch
		Electronic gear ratio	Pulse per motor rotation/travel distance per motor rotation, or (Pulse per motor rotation × Motor gear ratio)/(Work travel distance per rotation × Work gear ratio)
Positions that can be managed			Command positions and actual positions
Axis types			Servo axes, Virtual servo axes, Encoder axes, and Virtual encoder axes
Position command values			Negative or positive long reals (LREAL) or 0 (command units[*5])
Velocity command values			Negative or positive long reals (LREAL) or 0 (command units/s)
Acceleration command values and deceleration command values			Positive long reals (LREAL) or 0 (command units/s^2)
Jerk command values			Positive long reals (LREAL) or 0 (command units/s^3)
Single axes	Single-axis position control	Absolute positioning	Positioning is performed for a target position that is specified with an absolute value.
		Relative positioning	Positioning is performed for a specified travel distance from the command current position.
		Interrupt feeding	Positioning is performed for a specified travel distance from the position where an interrupt input was received from an external input.
		Cyclic synchronous absolute positioning[*2]	A command position is output each control period in Position Control Mode.
	Single-axis velocity control	Velocity control	Velocity control is performed in Position Control Mode.
		Cyclic synchronous velocity control	A velocity command is output each control period in Velocity Control Mode.
	Single-axis torque control	Torque control	The torque of the motor is controlled.
		Cyclic synchronous torque control[*2]	A torque command is output for each control period in Torque Control Mode.
	Single-axis synchronized control	Starting cam operation	A cam motion is performed using the specified cam profile curve.
		Starting cam operation with specified curve	A cam motion is performed using the specified cam properties and cam nodes.
		Ending cam operation	The cam motion for the axis that is specified with the input parameter is ended.
		Cam monitor, cam monitor with specified curve	Information on cam operation is monitored.
		Starting gear operation	A gear motion with the specified gear ratio is performed between a master axis and slave axis.

Item			Description
		Positioning gear operation	A gear motion with the specified gear ratio and sync position is performed between a master axis and slave axis.
		Ending gear operation	The specified gear motion or positioning gear motion is ended.
		Synchronous positioning	Positioning is performed in sync with a specified master axis.
		Master axis phase shift	The phase of a master axis in synchronized control is shifted.
		Combining axes	The command positions of two axes are added or subtracted and the result is output as the command position.
	Single-axis manual operation	Powering the Servo	The Servo in the Servo Drive is turned ON to enable axis motion.
		Jogging	An axis is jogged at a specified target velocity.
	Auxiliary functions for single-axis control	Resetting axis errors	Axes errors are cleared.
		Homing	A motor is operated and the limit signals, home proximity signal, and home signal are used to define home.
		Homing with parameters*2	The parameters are specified, the motor is operated, and the limit signals, home proximity signal, and home signal are used to define home.
		High-speed homing	Positioning is performed for an absolute target position of 0 to return to home.
		Stopping	An axis is decelerated to a stop.
		Immediately stopping	An axis is stopped immediately.
		Setting override factors	The target velocity of an axis can be changed.
		Changing the current position	The command current position or actual current position of an axis can be changed to any position.
		Enabling external latches	The position of an axis is recorded when a trigger occurs.
		Disabling external latches	The current latch is disabled.
		Zone monitoring	You can monitor the command position or actual position of an axis to see when it is within a specified range (zone).
		Enable Digital Cam Switch*2	The digital outputs are turned ON or turned OFF depending on the axis position.
		Monitoring axis following error	You can monitor whether the difference between the command positions or actual positions of two specified axes exceeds a threshold value.
		Resetting the following error	The error between the command current position and actual current position is set to 0.
		Torque limit	The torque control function of the Servo Drive can be enabled or disabled and the torque limits can be set to control the output torque.
		Changing axis use *2	The **Axis Use** axis parameter can be temporarily changed.
		Start velocity*2	You can set the initial velocity when axis motion starts.
Axes groups	Multi-axes coordinated control	Absolute linear interpolation	Linear interpolation is performed to a specified absolute position.
		Relative linear interpolation	Linear interpolation is performed to a specified relative position.
		Circular 2D interpolation	Circular interpolation is performed for two axes.

Item			Description
		Axes group cyclic synchronous absolute positioning*2	A positioning command is output each control period in Position Control Mode.
	Auxiliary functions for multi-axes coordinated control	Resetting axes group errors	Axes group errors and axis errors are cleared.
		Enabling axes groups	Motion of an axes group is enabled.
		Disabling axes groups	Motion of an axes group is disabled.
		Changing the axes in an axes group*2	The **Composition Axes** parameter in the axes group parameters can be overwritten temporarily.
		Stopping axes groups	All axes in interpolated motion are decelerated to a stop.
		Immediately stopping axes groups	All axes in interpolated motion are stopped immediately.
		Setting axes group override factors	The blended target velocity is changed during interpolated motion.
		Reading axes group positions*2	The command current positions and actual current positions of an axes group can be read.
Common items	Cams*6	Setting cam table properties and cam table end point index	The end point index of the cam table that is specified in the input parameter is changed.
		Saving cam tables	The cam table that is specified with the input parameter is saved in non-volatile memory in the CPU Unit.
		Generating cam tables*2	The cam table that is specified with the input parameter is generated from the cam property and cam node.
	Parameters	Writing MC settings	Some of the axis parameters or axes group parameters are overwritten temporarily.
		Changing axis parameters*2	You can access and change the axis parameters from the user program.
Auxiliary functions	Count modes		You can select either Linear Mode (finite length) or Rotary Mode (infinite length).
	Unit conversions		You can set the display unit for each axis according to the machine.
	Acceleration/ deceleration control	Automatic acceleration/ deceleration control	Jerk is set for the acceleration/deceleration curve for an axis motion or axes group motion.
		Changing the acceleration and deceleration rates	You can change the acceleration or deceleration rate even during acceleration or deceleration.
	In-position check		You can set an in-position range and in-position check time to confirm when positioning is completed.
	Stop method		You can set the stop method to the immediate stop input signal or limit input signal.
	Re-execution of motion control instructions		You can change the input variables for a motion control instruction during execution and execute the instruction again to change the target values during operation.
	Multi-execution of motion control instructions (Buffer Mode)		You can specify when to start execution and how to connect the velocities between operations when another motion control instruction is executed during operation.

Item			Description
	Monitoring functions	Software limits	The movement range of an axis is monitored.
		Following error	The error between the command current value and the actual current value is monitored for an axis.
		Velocity, acceleration rate, deceleration rate, torque, interpolation velocity, interpolation acceleration rate, and interpolation deceleration rate	You can set and monitor warning values for each axis and each axes group.
	Absolute encoder support		You can use an OMRON 1S-series Servomotor or G5-series Servomotor with an Absolute Encoder to eliminate the need to perform homing at startup.*7
	Input signal logic inversion*2		You can inverse the logic of immediate stop input signal, positive limit input signal, negative limit input signal, or home proximity input signal.
External interface signals			The Servo Drive input signals given below are used. • Home signal, home proximity signal, positive limit signal, negative limit signal, immediate stop signal, and interrupt input signal

*1. Unit version 2.1 or later is recommended for G5-series Cylinder-type Servo Drives. Unit version 1.1 or later is recommended for G5-series Linear Motor Types.
*2. Functions were added for an upgraded version of the CPU Unit. Refer to *A-7 Version Information* on page A-34 for information on version upgrades.
*3. The recommended unit version is 1.1 or later.
*4. Some of the functions of the MC Function Module are different when NX-series Position Interface Units are used. Refer to the *NX-series Position Interface Units User's Manual (Cat. No. W524)* for details.
*5. Positions can be set within a 40-bit signed integer range when converted to pulses.
*6. You can create the cam table with the Cam Editor in the Sysmac Studio or with the Generate Cam Table instruction in the user program. Specify the master axis phase and the slave axis displacement. You can change the phase pitch for each range. Cam data can be overwritten from the user program.
*7. Application is possible when you use an absolute external scale for an OMRON G5-series Linear Motor Type Servo Drive with built-in EtherCAT communications.

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.

2-1 Internal Configuration of the CPU Unit

This section provides an overview of the internal mechanisms of the NJ/NX-series CPU Unit.
The CPU Unit has the following software configuration.
The Motion Control Function Module is a software module that performs motion control.

<table>
<tr><td></td><td>Motion Control Function Module</td><td>EtherCAT Master Function Module</td><td>Other Function Modules*1</td></tr>
<tr><td colspan="4">PLC Function Module</td></tr>
<tr><td colspan="4">OS</td></tr>
</table>

*1. Refer to the *NJ/NX-series CPU Unit Software User's Manual (Cat. No. W501)* for details on other Function Modules.

The PLC Function Module runs on top of the OS. The other Function Modules run on top of the PLC Function Module.
A description of each Function Module is given in the following table.

Function Module name	Abbreviation	Description
PLC Function Module	PLC	This module manages overall scheduling, executes the user program, sends commands to the Motion Control Function Module, and provides interfaces to USB and the SD Memory Card.
Motion Control Function Module	MC	This module performs motion control according to the commands from motion control instructions that are executed in the user program. It sends data to the EtherCAT Master Function Module.
EtherCAT Master Function Module	ECAT	This module communicates with the EtherCAT slaves as the EtherCAT master.

Note Refer to the *NJ/NX-series CPU Unit Software User's Manual (Cat. No. W501)* for details on other Function Modules.

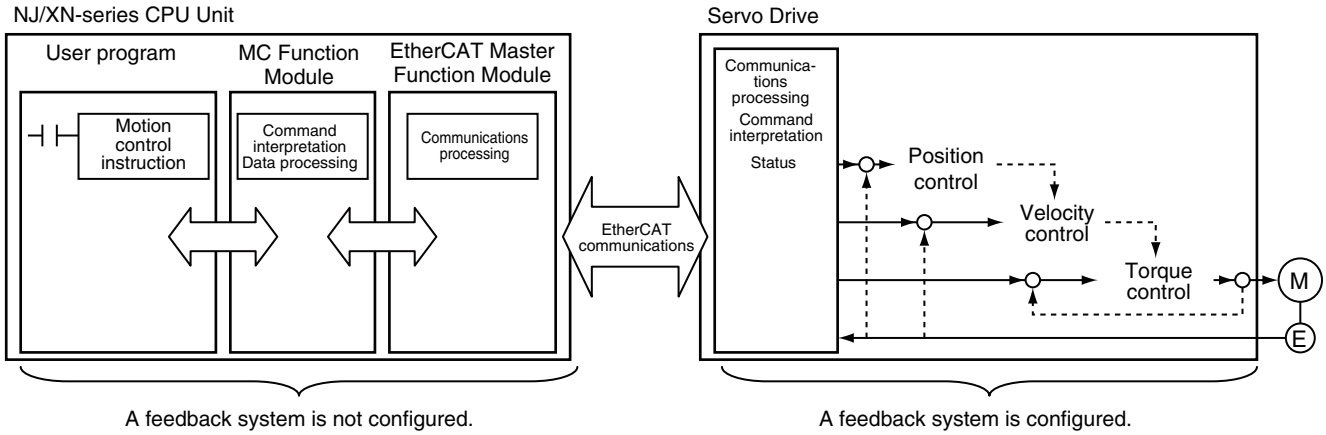
This manual provides the specifications and operating procedures for the Motion Control Function Module (sometimes abbreviated to "MC Function Module").
Refer to the other NJ/NX-series user's manuals as required when using the MC Function Module in an application.

2-2 Motion Control Configuration

A control system built with Servo Drives generally controls motor operation with a semi-closed loop. The semi-closed loop uses an encoder attached to the motor to detect the amount of rotation that has been performed by the motor in response to the command value. This is provided as feedback of the machine's travel distance. The following error between the command value and actual motor rotation is calculated and control is performed to bring the following error to zero.

In a machine configuration that uses the MC Function Module, no feedback information is provided for the commands from the user program in the CPU Unit. A feedback system is built into the Servo Drive.

Configuration on EtherCAT Network

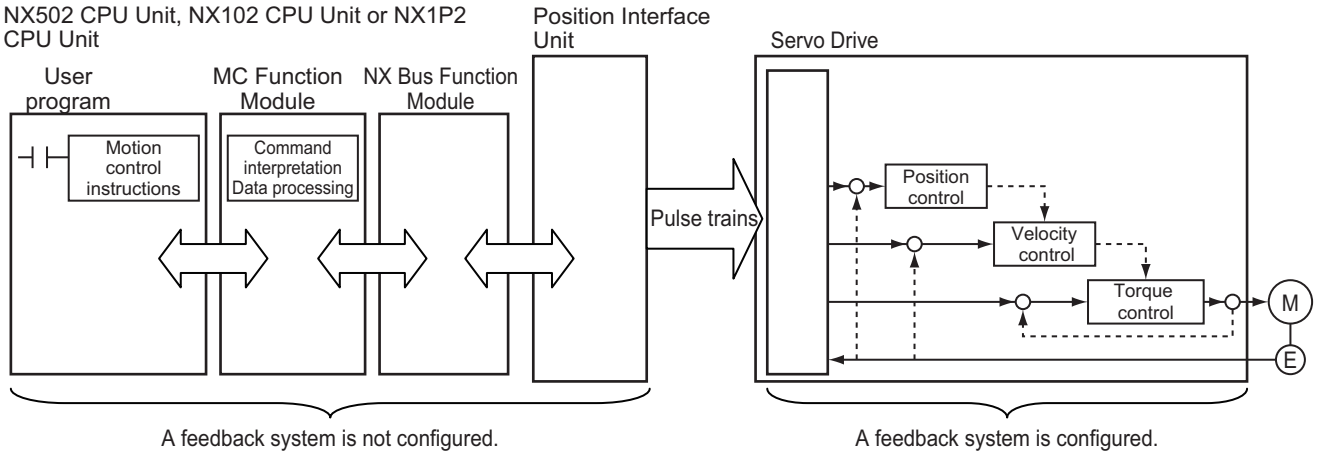


- When motion control instructions are executed in the user program, the MC Function Module interprets the resulting commands.
- The MC Function Module then performs motion control processing at a fixed period based on the results of the command interpretation. It generates command values to send to the Servo Drive. The following command values are generated: target position, target velocity, and target torque.
- The command values are sent by using PDO communications during each process data communications cycle of EtherCAT communications.
- The Servo Drive performs position loop control, velocity loop control, and torque loop control based on the command values received during each process data communications cycle of EtherCAT communications.
- The encoder's current value and the Servo Drive status are sent to the CPU Unit during each process data communications cycle of EtherCAT communications.

Additional Information

- Motion control processing and process data communications in EtherCAT communications are performed during the same time period.
- The MC Function Module controls the Servo Drive, which contains the position control loop, velocity control loop, and torque control loop.
- Refer to the *NX-series Position Interface Units User's Manual (Cat. No. W524)* for information on the configuration to use the NX-series Position Interface Units.

Configuration on CPU Unit



- When motion control instructions are executed in the user program, the MC Function Module interprets the resulting commands.
- The MC Function Module then performs motion control processing at a fixed period based on the results of the command interpretation. It generates command values to send to the Position Interface Unit (Pulse Output Unit). The following command values are generated: target position and target velocity.
- Generated command values are output to the Servo Drive as pulse trains.
- The Servo Drive performs position loop control and velocity loop control based on the command values which are output as pulse trains.

Additional Information

Refer to the *NX-series Position Interface Units User's Manual (Cat. No. W524)* for information on the configuration to use the NX-series Position Interface Units.

2-3 Motion Control Principles

This section provides information on the CPU Unit tasks and how they relate to motion control.

2-3-1 CPU Unit Tasks

Tasks are attributes of programs that determine the execution conditions and sequence of the programs.
The NJ/NX-series CPU Unit support the following tasks.

Type of task	Task name
Tasks that execute programs at a fixed period	Primary periodic task
	Priority-5*1, -16*2, -17, and -18 periodic tasks
Tasks that execute programs only once when the execution conditions for the tasks are met	Event task (execution priority: 8 and 48)*3

*1. You can use the priority-5 periodic task only with NX701 CPU Units.
*2. You cannot use the priority-16 periodic task on NX102 CPU Units and NX1P2 CPU Units.
*3. A CPU Unit with unit version 1.03 or later and Sysmac Studio version 1.04 or higher are required to use event tasks.

Refer to the *NJ/NX-series CPU Unit Software User's Manual (Cat. No. W501)* for details on programs, tasks, and setting methods.

Types of Tasks and Task Priority

The NX701 CPU Unit can execute the primary periodic task and the priority-5 periodic task in parallel with a multi-core processor.
The NX502 CPU Unit, NX102 CPU Unit, NX1P2 CPU Unit, and NJ-series CPU Unit cannot execute more than one task in parallel.
Tasks have an execution priority. Tasks with the highest execution priority are executed first. If the execution conditions are met for another task with a higher execution priority while a task is under execution, the task with the higher execution priority is given priority in execution.
The following table lists the tasks in which you can use motion control instructions and the task priorities for the NJ/NX-series CPU Unit. You cannot use motion control instructions in event tasks.

Type of task	Number of tasks	Priority	Operation
Primary periodic task	1	4	This task executes I/O refreshing, programs, and motion control in the specified task period. This task has the highest execution priority of all tasks and can be executed quickly and precisely. Therefore, this task is best suited for situations when synchronized control or highly responsive control is required. Use the primary periodic task to execute all control with a single task.

Type of task	Number of tasks	Priority	Operation
Periodic tasks	0 or 1	5*1	This task executes I/O refreshing, programs, and motion control in the specified task period. The priority-5 periodic task has the second highest execution priority after the primary periodic task and can be executed quickly and precisely. The priority-5 periodic task is used when you want to divide functions configuring the device for separate control, those functions that need high-speed control with the primary periodic task and others with the priority-5 periodic task. The primary periodic task and priority-5 periodic task are used for the multi-task motion control. *2
		16*3	This task executes programs and I/O refreshing in the specified task period. The execution period for the priority-16 periodic task is longer than the execution period of the primary periodic task and priority-5 periodic task. Therefore, this periodic task is used to execute programs. In the priority-16 periodic task, you can write the user program for some slaves and Units that refresh I/O in the primary periodic task. For example, synchronized control and control requiring a fast response time are placed in the primary periodic task or priority-5 periodic task. Overall device control is separately placed in a priority-16 periodic task.

*1. You can use the priority-5 periodic task only with NX701 CPU Units.

*2. The multi-motion performs the motion controls in parallel using both the primary periodic task and the priority-5 periodic task.

*3. You cannot use the priority-16 periodic task on NX102 CPU Units and NX1P2 CPU Units.

Note The NJ/NX-series CPU Unit has priority-17 and -18 periodic tasks. However, you cannot use motion control instructions in these tasks. These tasks also do not perform I/O refreshing.

Precautions for Correct Use

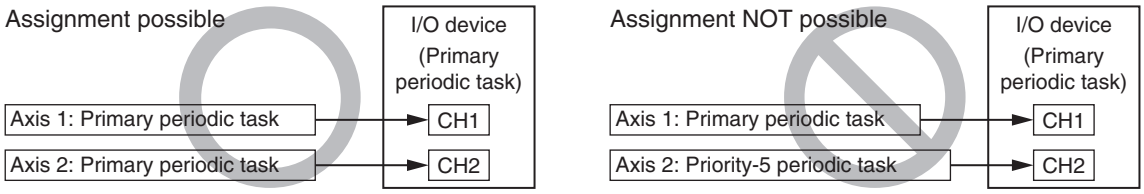
Motion control instructions can be used in the primary periodic task and in a priority-5 or a priority-16 periodic task.
If motion control instructions are used in any other tasks, an error will occur when the user program is built on the Sysmac Studio.

Additional Information

The NX701 CPU Unit allows you to execute the motion control in the primary periodic task and in the priority-5 periodic task. If these two motion controls need to be identified, the motion control in the primary periodic task is called motion control 1, while the motion control in the priority-5 periodic task is called motion control 2.

Task Assignment

- Axes and axes groups can be assigned to either of the primary periodic task and the priority-5 periodic task. The I/O device task that is assigned to an axis must be the same type of task that is assigned to the axis.



- You can execute motion control instructions from the user program that is operated in the priority-16 periodic task for the axes and axes groups that are assigned to the primary periodic task.

Precautions for Correct Use

- You cannot execute motion control instructions from the user program that is operated in the priority-5 periodic task for the axes and axes groups that are assigned to the primary periodic task.
  If you perform the execution, depending on whether the axes that are assigned to the priority-5 periodic task exist or not, the operation differs as follows.
  a) If the axes that are assigned to the priority-5 periodic task exist, an Illegal Axis Specification (event code: 54600000 hex) occurs.
  b) If the axes that are assigned to the priority-5 periodic task do not exist, motion control instructions are executed, however, do not use the instructions due to the variations of execution timing.
  Similarly, you cannot execute motion control instructions from the user program that is operated in the primary periodic task for the axes and axes groups that are assigned to the priority-5 periodic task.
- You cannot execute motion control instructions from the user program that is operated in the priority-16 periodic task for the axes and axes groups that are assigned to the priority-5 periodic task. If you perform the execution, an Illegal Axis Specification (event code: 54600000 hex) occurs.

Additional Information

Refer to *Section 3 Configuring Axes and Axes Groups* on page 3-1 for details on axes and axes groups.

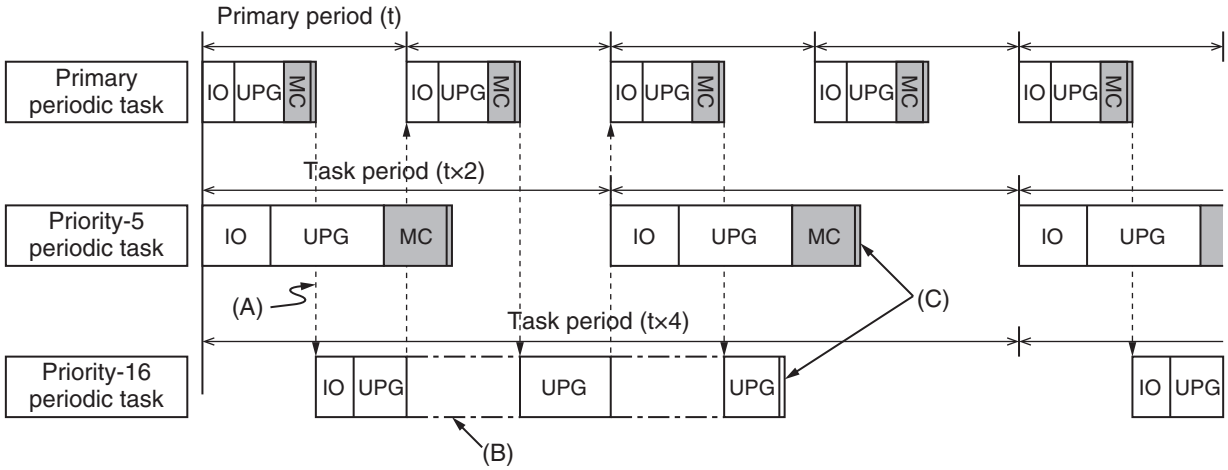
Basic Operation of Tasks

● Overall Task Operation

The primary periodic task and periodic tasks operate based on the task period of the primary periodic task (also called the primary period).

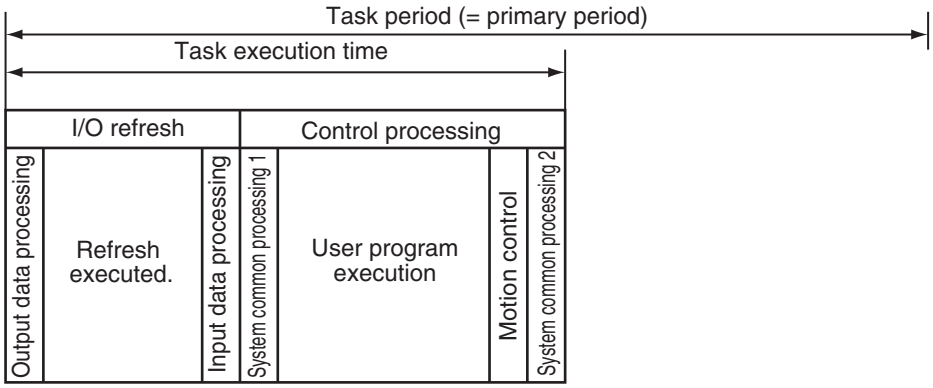
The primary periodic task and priority-5 periodic task include operations such as system common processing and motion control in addition to I/O refreshing and user program execution.

Processing of motion control instructions in the programs is executed during the next motion control (MC) period after the END instruction is executed in the task.



Symbol	Description
IO	I/O refreshing
UPG	User program execution
MC	Motion control
(A)	A dotted line represents a transition to another task.
(B)	A dashed-dotted line means that processing for that task has been interrupted.
(C)	A double line means that all processing for that task has been completed.

● Operation of the Primary Periodic Task



Processing	Processing contents
Output data processing	• Output refresh data is created for Output Units that execute I/O refreshing. • If forced refreshing is set, the forced refreshing values are reflected in the output refresh data.
Refresh execution	This process exchanges data with I/O.
Input data processing	• Whether the condition expression for event task execution is met or not is determined. • Input refresh data is loaded from Input Units that execute I/O refreshing. • If forced refreshing is set, the forced refreshing values are reflected in the input refresh data that was read.
System common processing 1	• Processing for exclusive control of variables in tasks is performed when accessing tasks are set. • Motion input processing is performed. *1 • Data tracing processing (sampling and trigger checking) is performed.

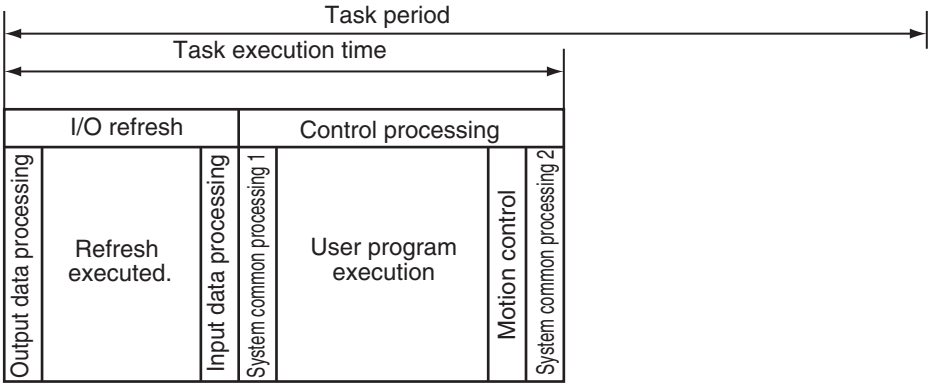
Processing	Processing contents
User program execution	Programs assigned to tasks are executed in the order that they are assigned.
Motion control 1*2	The motion control commands from the motion control instructions in the programs in the primary periodic task and priority-16 periodic task are executed. Motion output processing is performed. *3
System common processing 2	• Processing for exclusive control of variables in tasks is performed when refreshing tasks are set. • Processing for variables accessed from outside of the Controller is performed to maintain concurrency with task execution (executed for the variable access time that is set in the **Task Settings**). • If there is processing for EtherNet/IP tag data links and refreshing tasks are set for the tags (i.e., variables with a Network Publish attribute), variable access processing is performed.

*1. The Servo Drive status, axis current values, and other motion control system-defined variables are updated according to data received from the Servo Drives.

*2. For the system-defined variables of the axes that are assigned to Motion control 1 *_MC_AX[0-255]*, or *_MC1_AX[0-255]* are used. Similarly, for the system-defined variables of the axes groups, *_MC_GRP[0-63]* or *_MC1_GRP[0-63]* are used.
Refer to *3-1-3 Introduction to Axis Variables* on page 3-6 for the system-defined variables of axes and *3-3-3 Introduction to Axes Group Variables* on page 3-24 for the system-defined variables of axes groups.

*3. Data is sent to the Servo Drives during I/O refreshing in the next primary periodic task.

● Operation of a Priority-5 Periodic Task



Processing	Processing contents
Output data processing	• Output refresh data is created for Output Units that execute I/O refreshing. • If forced refreshing is set, the forced refreshing values are reflected in the output refresh data.
Refresh execution	This process exchanges data with I/O.
Input data processing	• Input refresh data is loaded from Input Units that execute I/O refreshing. • If forced refreshing is set, the forced refreshing values are reflected in the input refresh data that was read.
System common processing 1	• Processing for exclusive control of variables in tasks is performed when accessing tasks are set. • Motion input processing is performed.*1 • Data tracing processing (sampling and trigger checking) is performed.
User program execution	Programs assigned to tasks are executed in the order that they are assigned.

Processing	Processing contents
Motion control 2*2	• The motion control commands from the motion control instructions in the programs in the primary periodic task and priority-5 periodic task are executed. • Motion output processing is performed.*3
System common processing 2	• Processing for exclusive control of variables in tasks is performed when refreshing tasks are set. • Processing for variables accessed from outside of the Controller is performed to maintain concurrency with task execution (executed for the *variable access time* that is set in the **Task Settings**).

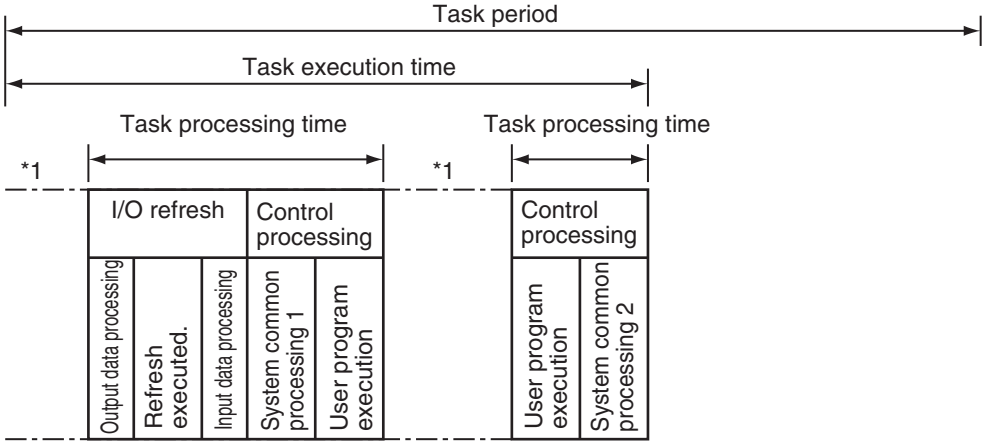
*1. The Servo Drive status, axis current values, and other motion control system-defined variables are updated according to data received from the Servo Drives.

*2. For the system-defined variables of the axes that are assigned to Motion control 2, *_MC2_AX[0-255]* are used. Similarly, for the system-defined variables of the axes groups, *_MC2_GRP[0-63]* are used.
Refer to *3-1-3 Introduction to Axis Variables* on page 3-6 for the system-defined variables of axes and *3-3-3 Introduction to Axes Group Variables* on page 3-24 for the system-defined variables of axes groups.

*3. Data is sent to the Servo Drives during I/O refreshing in the next priority-5 periodic task.

● Operation of a Priority-16 Periodic Task

You can refresh I/O in the priority-16 periodic task.



*1. The CPU Unit will temporarily interrupt the execution of a task in order to execute a task with a higher execution priority.

Task Period

For a single task, the primary period, which is the task period for the primary periodic task, is the standard period for execution.
In this case, the primary period is automatically used as the motion control period. (It is also the same as the process data communications cycle for EtherCAT communications.)
The NX502 CPU Unit, NX102 CPU Unit, NX1P2 CPU Unit, and NJ-series CPU Unit support only the single task control.

For multi-motion, two kinds of period, the primary period and the task period of priority-5 periodic task are the standard periods for execution.
In this case, the motion control takes two kinds of period, while the process data communications cycle for EtherCAT communications automatically takes each of the task periods.

Periodic task execution is synchronized with the primary period. Set the task period of a periodic task as an integer multiple of the primary period.
For example, if the primary period is 1 ms, then you can set the task period of a priority-5 periodic task to 2 ms and the task period of a priority-16 periodic task to 4 ms. In that case, the start of the period for the primary periodic task and the priority-5 periodic task will match once every two primary periods. Similarly, the start of the period for the primary periodic task and the priority-16 periodic task will match once every four primary periods.

Refer to the *NJ/NX-series CPU Unit Software User's Manual (Cat. No. W501)* for details on the task period.

Additional Information

If two process data communications cycles need to be identified, the communications cycle for the primary periodic task is called process data communications cycle 1, while the communications cycle for the priority-5 periodic task is called process data communications cycle 2.

● Valid Task Periods for NX701 CPU Unit

For the NX701 CPU Unit, valid task periods depend on the type of task as shown below.

Task	Valid task periods
Primary periodic task	125 μs, 250 μs to 8 ms (specify in increments of 250 μs)
Priority-5 periodic task	125 μs, 250 μs to 100 ms (specify in increments of 250 μs)
Priority-16 periodic task	1 ms to 100 ms (specify in increments of 250 μs)

Note The setting conditions of task periods of the primary periodic task and the periodic tasks.
The task periods for periodic tasks must be set to integer multiples of the task period of the primary periodic task.
Set the task period so that the least common multiple of task periods of each task must be less than or equal to 600 ms.

● Valid Task Periods for NX502 CPU Unit

For the NX502 CPU Unit, valid task periods depend on the type of task as shown below.

Task	Valid task periods
Primary periodic task	250 μs to 8 ms (specify in increments of 250 μs)
Priority-16 periodic task	1 ms to 100 ms (specify in increments of 250 μs)

Note The setting conditions of task periods of the primary periodic task and the periodic tasks.
The task periods for periodic tasks must be set to integer multiples of the task period of the primary periodic task.
et the task period so that the least common multiple of task periods of each task must be less than or equal to 600 ms.

● Valid Task Periods for NX102 CPU Unit

Only the primary periodic task is valid for an NX102 CPU Unit. Valid task periods are as follows.

Task	Valid task periods
Primary periodic task	1 ms to 32 ms (specify in increments of 250 μs)

● Valid Task Periods for NX1P2 CPU Unit

Only the primary periodic task is valid for an NX1P2 CPU Unit. Valid task periods are as follows.

Task	Valid task periods
Primary periodic task	2 ms to 8 ms (specify in increments of 250 μs)

● Valid Task Periods for NJ-series CPU Unit

The following table lists the possible combinations of primary periodic task and priority-16 periodic task periods for the NJ-series CPU Unit.

Primary period	Valid task periods for periodic tasks
500 μs*1	1 ms, 2 ms, 3 ms, 4 ms, 5 ms, 8 ms, 10 ms, 15 ms, 20 ms, 25 ms, 30 ms, 40 ms, 50 ms, 60 ms, 75 ms, or 100 ms
1 ms	1 ms, 2 ms, 3 ms, 4 ms, 5 ms, 8 ms, 10 ms, 15 ms, 20 ms, 25 ms, 30 ms, 40 ms, 50 ms, 60 ms, 75 ms, or 100 ms
2 ms	2 ms, 4 ms, 8 ms, 10 ms, 20 ms, 30 ms, 40 ms, 50 ms, 60 ms, or 100 ms
4 ms	4 ms, 8 ms, 20 ms, 40 ms, 60 ms, or 100 ms

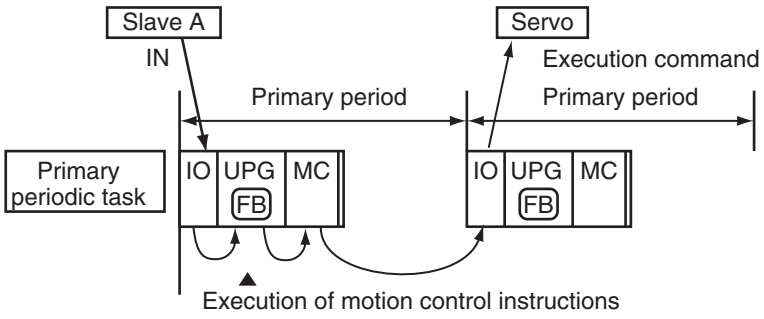
*1. Unit version 1.03 or later is required to use this setting on the NJ301 CPU Unit.
You cannot use this setting on the NJ101-10□□.

2-3-2 Example of Task Operations for Motion Control

Motion control instructions can be used in the primary periodic task, in a priority-5 periodic task, or in a priority-16 periodic task.
This section provides examples of task operations.

Using Motion Control Instructions in the Primary Periodic Task

If high-speed motion control is required, place the motion control instructions (FB) in the primary periodic task.



1 Loading Data
The input data from the EtherCAT slaves (slave A) is loaded during the I/O refresh (IO).

2 Instruction Execution
The motion control instructions (FB) are executed based on the data that was loaded during user program execution (UPG).
The output variables of the motion control instructions are refreshed at this point.

3 Command Generation

Motion processing according to the motion control instructions (FB) that were executed is performed during motion control (MC) immediately after user program execution in the primary periodic task. During this processing, execution commands for the Servo Drives and other devices are generated.

4 Sending Commands

The execution commands that were generated are sent to the Servo Drive or other device during the I/O refresh (IO) in the next period.

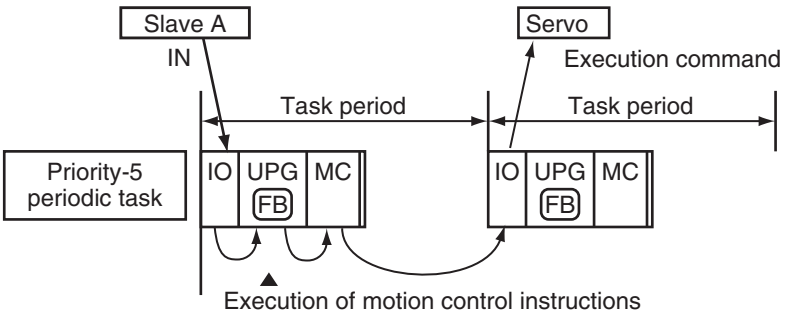
Additional Information

All instructions from inputs to execution command outputs to the Servo Drive or other device are processed quickly in this task. We recommend placing all motion control instructions in the primary periodic task.

Using Motion Control Instructions in a Priority-5 Periodic Task

If second high-speed motion control after the primary periodic task is required, place the motion control instructions (FB) in a priority-5 periodic task.
The basic operation is the same as that of the primary periodic task.

● Timing of Processing



1 Loading Data

The input data from the EtherCAT slaves (slave A) is loaded during the I/O refresh (IO).

2 Instruction Execution

The motion control instructions (FB) are executed based on the data that was loaded during user program execution (UPG).
The output variables of the motion control instructions are refreshed at this point.

3 Command Generation

Motion processing according to the motion control instructions (FB) that were executed is performed during motion control (MC) immediately after user program execution in the primary periodic task. During this processing, execution commands for the Servo Drives and other devices are generated.

4 Sending Commands

The execution commands that were generated are sent to the Servo Drive or other device during the I/O refresh (IO) in the next period.

Additional Information

- You can use the priority-5 periodic task only with NX701 CPU Units.
- The priority-5 periodic task is used when you want to divide functions configuring the device for separate control, those functions that need high-speed control with the primary periodic task and others with the priority-5 periodic task.

● Axis Variable Update Timing in Multi-motion

The multi-motion refers to execution of parallel control using the primary periodic task and the priority-5 periodic task.

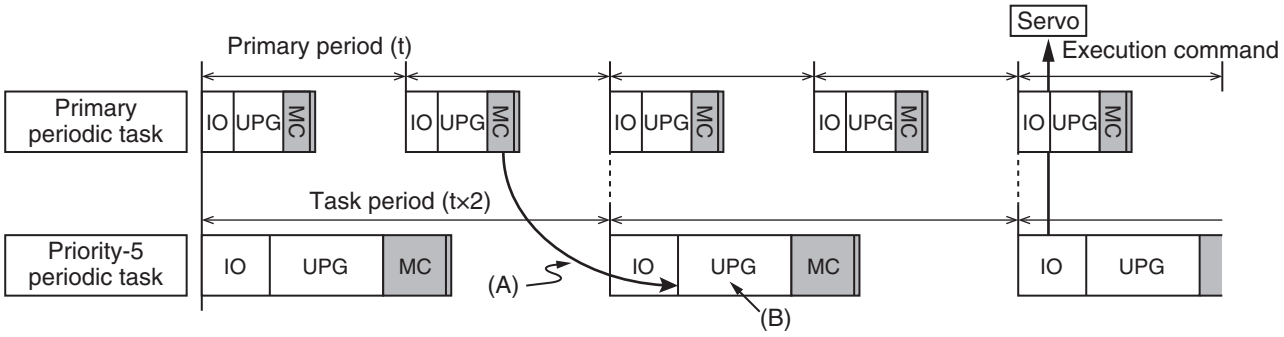
In the multi-motion, the user program for the priority-5 periodic task can access the values of an Axis Variable of an axis that is controlled in the primary periodic task. The reverse is possible: the user program for the primary periodic task can access the values of an Axis Variable of an axis that is controlled in the priority-5 periodic task.

Additional Information

- Refer to *3-1-4 Synchronizing Axis Variables* on page 3-8 for details on synchronization of axis variables.
- The user program for the priority-16 periodic task can access the values of an Axis Variable of an axis that is controlled in the primary periodic task. Refer to *Using Motion Control Instructions in a Priority-16 Periodic Task* on page 2-16 and *3-1-4 Synchronizing Axis Variables* on page 3-8 for details.

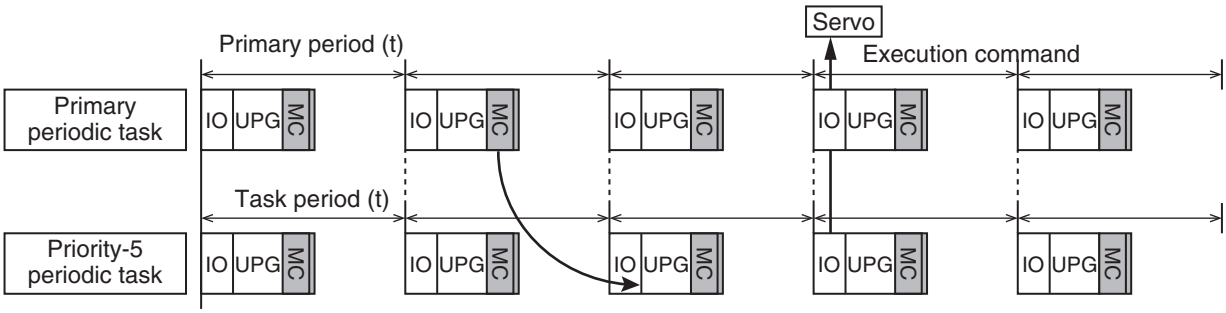
The values of an Axis Variable are updated synchronizing to the task period of accessing task.
The values of an Axis Variable that is accessed are not overwritten during the user program execution for the task that accesses the values of an Axis Variable.

The timing that the values of an Axis Variable in the primary periodic task update to the priority-5 periodic task is shown below.



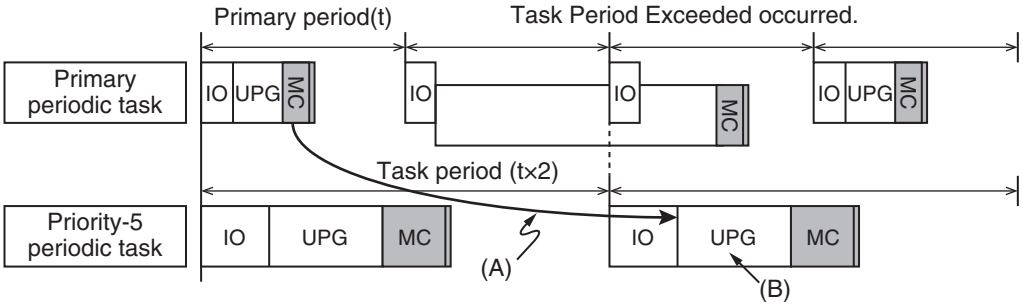
Symbol	Description
(A)	Axis Variable updated. Regardless of where the user program execution for the priority-5 periodic task starts, the Axis Variable is updated with the execution result of the primary periodic task immediately before the start of the task periods matched.
(B)	The values of an Axis Variable are not overwritten during the user program execution.

The update timing is similar if two task periods are the same.



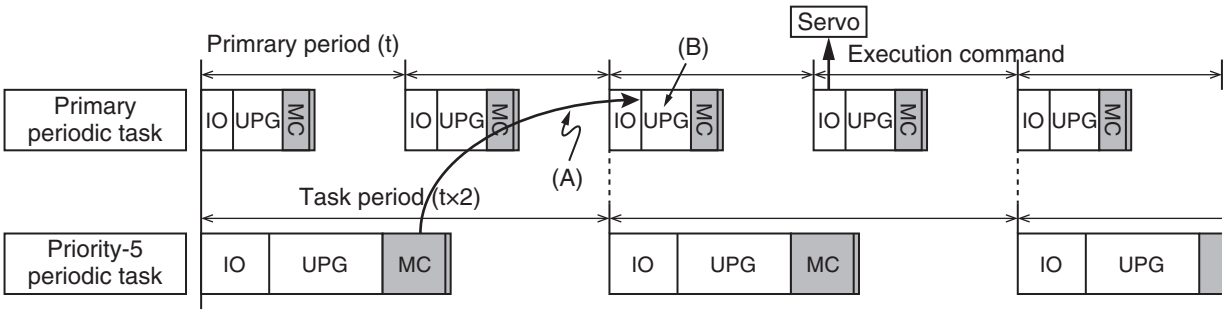
Additional Information

If a Task Period Exceeded error occurs, the Axis Variable is updated with the execution result of the one before the preceding primary periodic task period immediately before the start of the task periods matched, instead of the execution result of the preceding one.



Sym-bol	Description
(A)	Axis Variable updated. If a Task Period Exceeded error occurs, the Axis Variable is updated with the value of the primary periodic task period before the error.
(B)	The values of an Axis Variable are not overwritten during the user program execution.

The timing that the values of an Axis Variable in the priority-5 periodic task update to the primary periodic task is shown below.



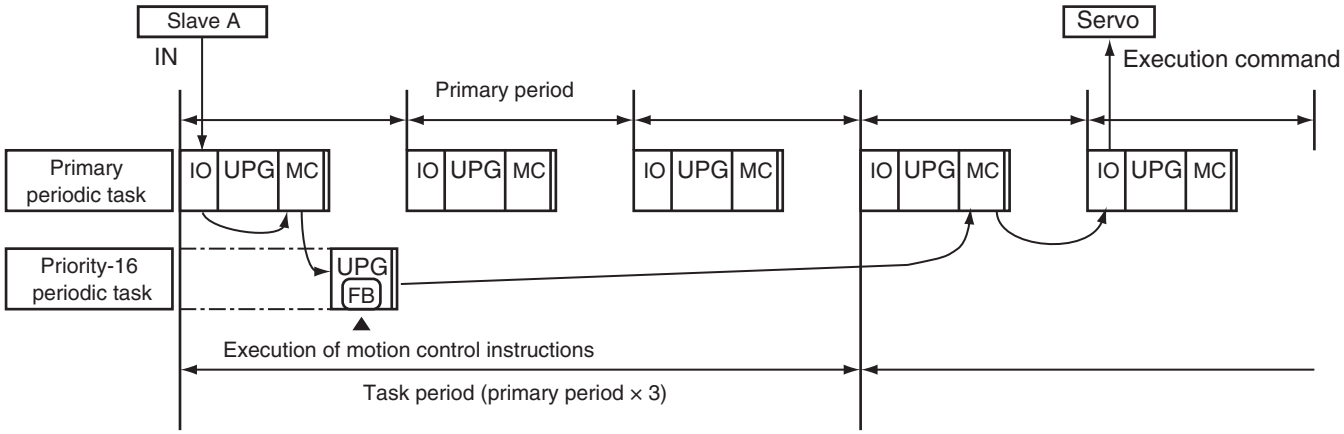
Sym bol	Description
(A)	Axis Variable updated. The Axis Variable is updated with the execution result of the priority-5 periodic task immediately before the start of the task periods matched.
(B)	The values of an Axis Variable are not overwritten during the user program execution.

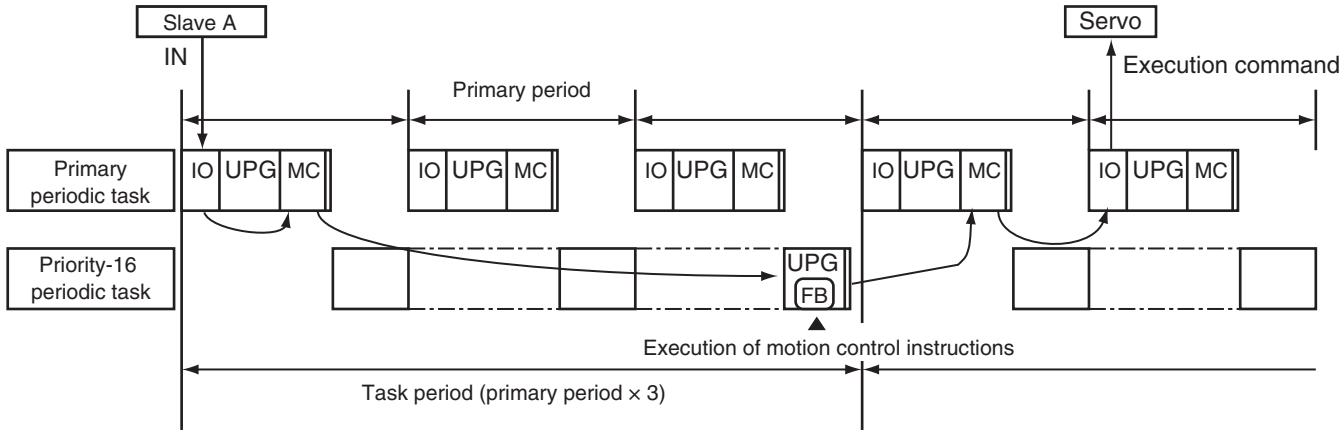
Using Motion Control Instructions in a Priority-16 Periodic Task

If high speed motion control is not required and/or your user program is too large, place motion control instructions in a priority-16 periodic task.

● Timing of Processing

Motion control processing (MC) for the motion control instructions (FB) that are executed in the same task period as the priority-16 periodic task are performed at the same time.
Therefore, processing for multiple axes can be simultaneously executed or stopped.





1 Loading Data
The input data from the EtherCAT slaves (slave A) is loaded during the I/O refresh (IO).

2 Instruction Execution
The motion control instructions (FB) are executed based on the data that was loaded during user program execution (UPG) in the priority-16 periodic task.
The output variables of the motion control instructions are refreshed at this point.

3 Command Generation
Motion control instructions (FB) are executed in the task period of the priority-16 periodic task according to the motion control instructions (FB) that were executed.

Motion processing is performed during motion control processing (MC) in the next primary periodic task after the periodic task. During this processing, execution commands for the Servo Drives or other devices are generated.

4 Sending Commands

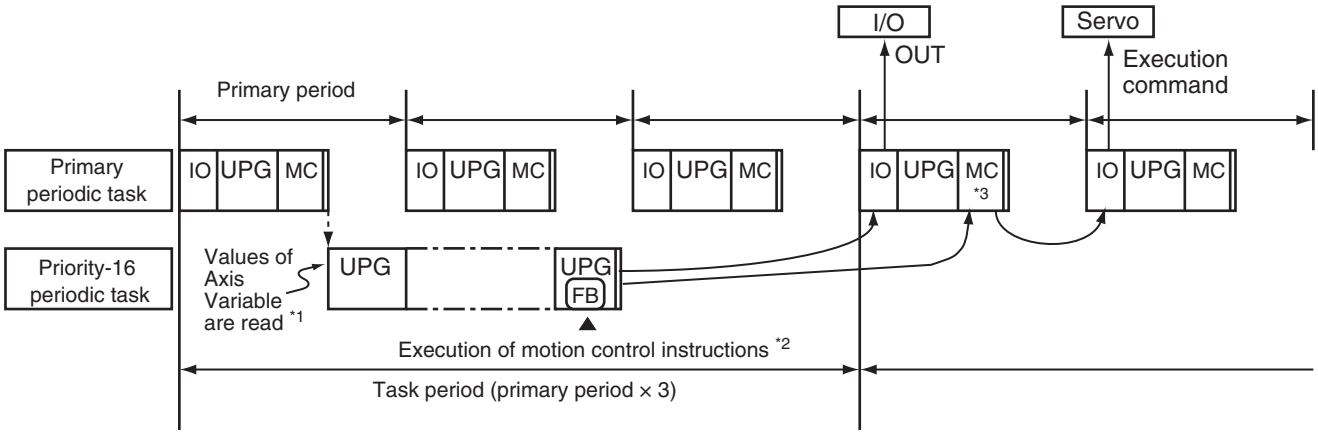
The execution commands that were generated are sent to the Servo Drive or other device during the I/O refresh (IO) in the next period.

● Axis Variable Update Timing

Axis Variables are system-defined variables for some of the axis parameters and for the monitor information, such as the actual position and error information, for the axes controlled by the MC Function Module.

If you access an Axis Variable of the primary periodic task during the priority-16 periodic task, the values of the variable that were read at the start of the priority-16 periodic task are used.

Also, the values of an Axis Variable are not written when a motion control instruction (FB) is executed. They are written in motion control processing (MC) at the start of the next priority-16 periodic task.



*1. The values of an Axis Variable of the primary periodic task are read at the start of user program execution for the priority-16 periodic task.

*2. The values of an Axis Variable are not written when a motion control instruction (FB) is executed in the priority-16 periodic task.

*3. The values are written during this motion control processing (MC).

Precautions for Correct Use

- When motion control instructions are placed in a priority-16 periodic task, the response time of the Servo Drive or other device will increase if the task period of the priority-16 periodic task is lengthened.
- Make sure that all axes can be stopped safely for emergency stops, including emergency stops commanded from external devices.
- The execution timing of motion control instructions in a priority-16 periodic task is not the same as the execution timing for I/O control. Design the user program to allow for this.
- You cannot use the priority-16 periodic task on NX102 CPU Units and NX1P2 CPU Units.

Additional Information

For information on Axis Variables, refer to *3-1-3 Introduction to Axis Variables* on page 3-6.

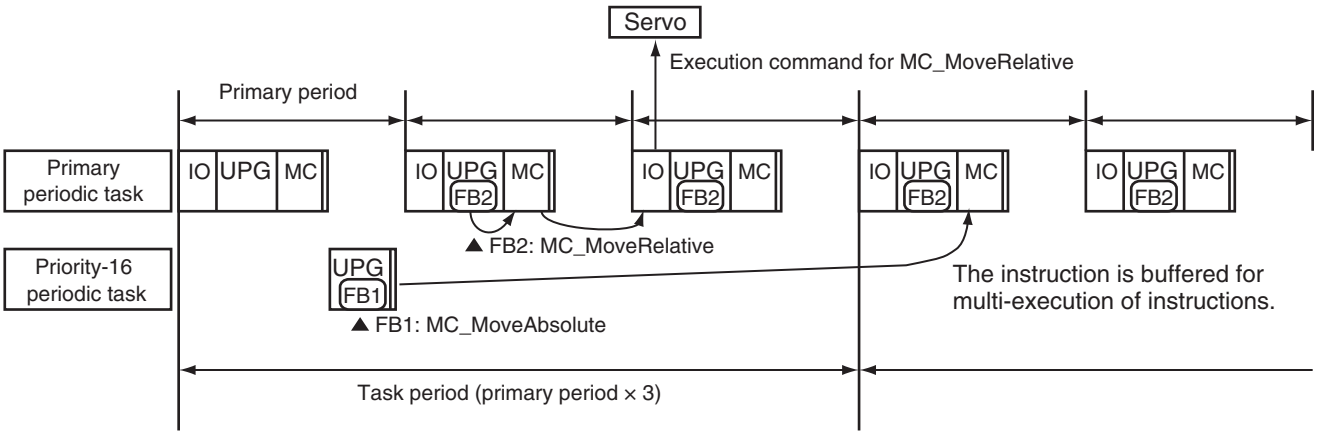
Using Motion Control Instructions in Two Different Types of Tasks

If you have processes that require high-speed motion control and processes that do not require high-speed motion control for the same axis, you can place the motion control instructions (FB) both in the primary periodic task and in a priority-16 periodic task.

If motion control instructions (FB) are executed in both tasks within the period of the priority-16 periodic task, the MC Function Module will perform motion processing for instructions in the primary periodic task first.

For example, the MC_MoveAbsolute instruction is executed in the priority-16 periodic task. Then, the MC_MoveRelative is executed for the same axis in the primary periodic task. The operation for this is shown below.

- The MC Function Module will execute MC_MoveRelative first. MC_MoveAbsolute is executed with multi-execution of instructions.



The values of output variables for a motion control instruction and the values of system-defined variables for motion control will change during the I/O refresh of the task that executed the instruction. Therefore, you may notice different behavior depending on the task if you use motion control instructions for the same axis in different tasks.
Make sure that you thoroughly understand the processes of each task before you start to develop your user program.

Precautions for Correct Use

- If you include motion control instructions for the same axis in both the primary periodic task and the priority-16 periodic task, pay close attention to the following when you develop your user program: the execution order of the motion control instructions, the timing of updates for system-defined variables for motion control, and the output timing of command values.
- If you use system-defined variables for motion control for the same axis in multiple tasks, pay close attention to the differences in timing for updating system-defined variables for motion control when you develop your user program.
- You cannot use the priority-16 periodic task on NX102 CPU Units and NX1P2 CPU Units.

Additional Information

For information on multi-execution of instructions, refer to *9-5-7 Multi-execution of Motion Control Instructions (Buffer Mode)* on page 9-53.

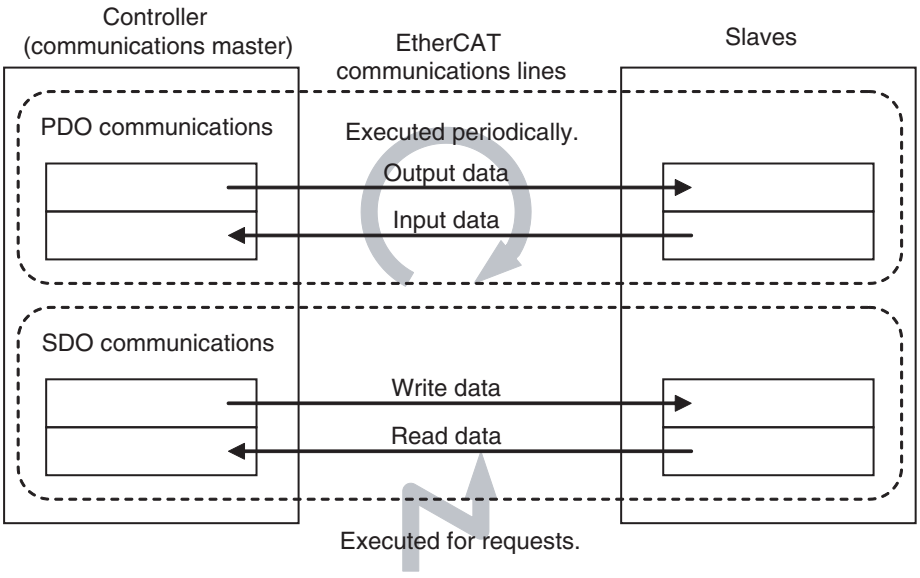
2-4 EtherCAT Communications and Motion Control

The MC Function Module controls Servo Drives, counters, and NX-series Position Interface Units through the PDO communications of the EtherCAT Master Function Module in the CPU Unit.
This section describes EtherCAT communications and other items related to the MC Function Module.

2-4-1 CAN Application Protocol over EtherCAT (CoE)

The MC Function Module exchanges data with the slaves on EtherCAT using the CAN application protocol over EtherCAT (CoE).
With CoE, the parameters and control information held by the slaves are specified according to data specifications of the object dictionary (OD).
To communicate the data between the Controller (communications master) and slaves, two methods are used: process data objects (PDOs), which periodically exchange data in realtime, and service data objects (SDOs), which exchange data when required.

The MC Function Module uses PDO communications for commands to refresh I/O data, such as data for Servomotor position control, on a fixed control period.
It uses SDO communications for commands to read and write data at specified times, such as for parameter transfers.



2-4-2 Relationship between EtherCAT Master Function Module and MC Function Module

The NJ/NX-series CPU Unit can perform sequence control and motion control through connections to EtherCAT slaves.

Sequence Control

- I/O ports for configuration slaves are automatically created when you create the EtherCAT Configuration in EtherCAT Tab Page in the Sysmac Studio.
- You use the I/O Map Tab Page in the Sysmac Studio to assign device variables.
- Perform sequence control through instructions other than motion control instructions.

Motion Control

- I/O ports for configuration slaves are automatically created when you create the EtherCAT Configuration in EtherCAT Tab Page in the Sysmac Studio.
- Create Axis Variables in Motion Control Setup View and assign the EtherCAT slaves for which motion control is performed.
- Perform motion control through motion control instructions.

The following devices can be assigned to Axis Variables: EtherCAT slave Servo Drives, Encoder Input Terminals, and NX-series Position Interface Units.

Additional Information

- Commands are not sent directly through PDO communications to an EtherCAT slave or NX-series Position Interface Unit that is assigned to an Axis Variable for instructions other than motion control instructions. However, the status of such an EtherCAT slave can be accessed indirectly through the Axis Variables.
- You can use SDO communications to read and write the objects of EtherCAT slaves and NX-series Position Interface Units that are assigned to Axis Variables.
  However, do not use SDO communications to write objects that are mapped to PDO communications. If you do, the operation of the slaves will depend on slave specifications. For OMRON slaves, SDO communications will result in errors.
- If EtherCAT slave Servo Drives, Encoder Input Terminals, and NX-series Position Interface Units are not assigned to Axis Variables, you must execute sequence control for them in the same way as for general-purpose EtherCAT slaves.

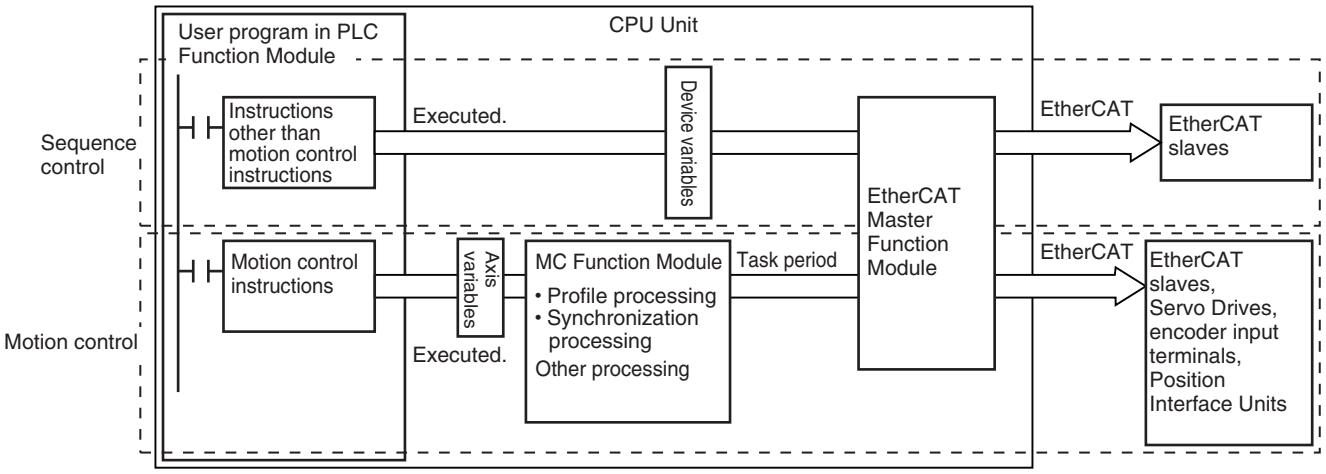
Version Information

With the Sysmac Studio version 1.09 or higher, you can assign device variables to the I/O ports of slaves and Units that are assigned to the Axis Variables.
The following are the conditions of I/O ports to which you can assign device variables.

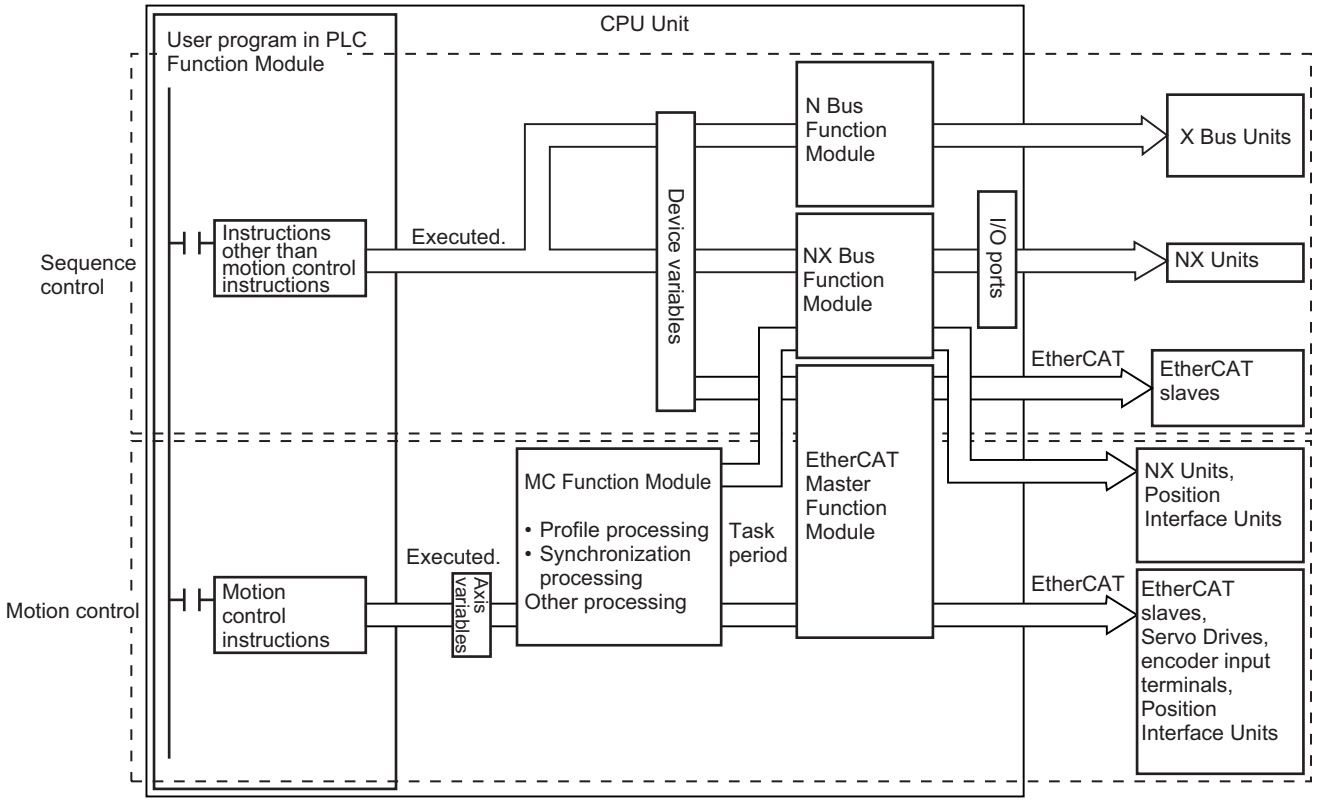
- I/O ports with Read/Write attribute set to Read (R: Read only).
- I/O ports with Read/Write attribute set to Write (W: Write only). Also, for these I/O ports, **<Not assigned>** must be set for the process data field under **Detailed Settings** on the **Axis Basic Settings** Display in the Sysmac Studio.

If you used the Sysmac Studio version 1.09 or higher to create a project and assign device variables to the Axis Variables, and open the project with the Sysmac Studio version 1.08 or lower, the assignment of the device variables will be cleared.

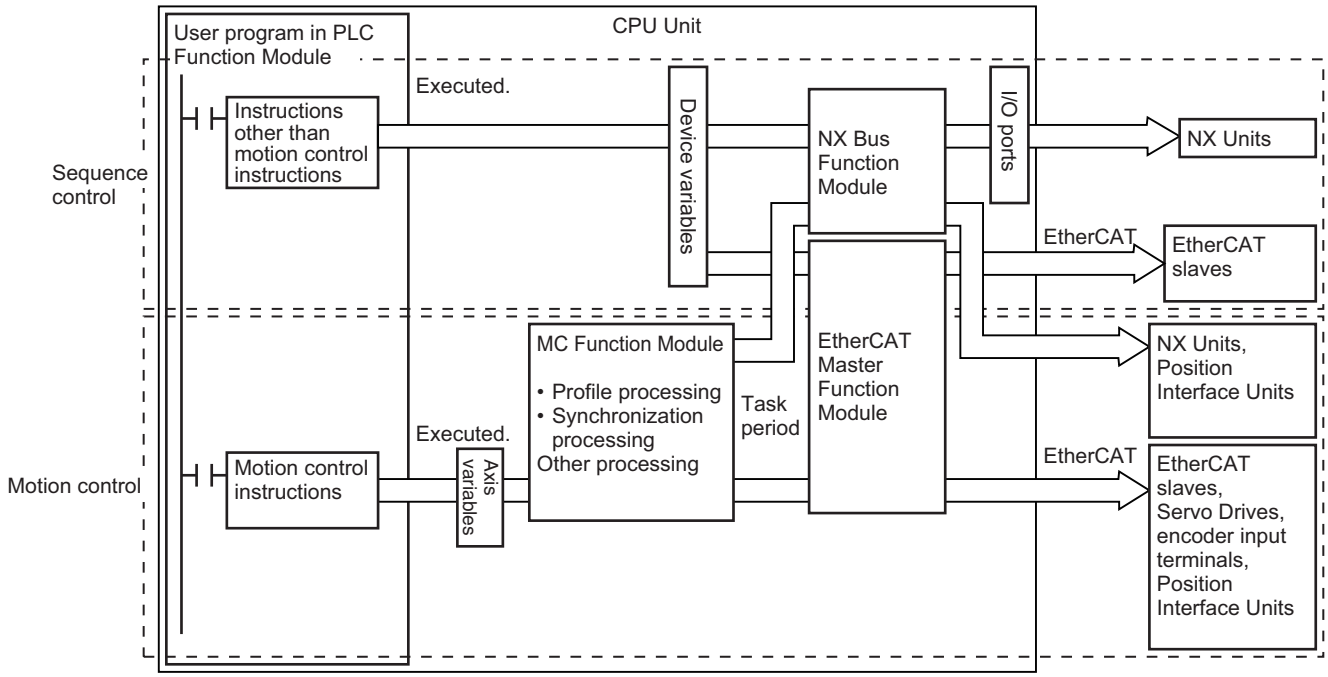
● NX701 CPU Unit



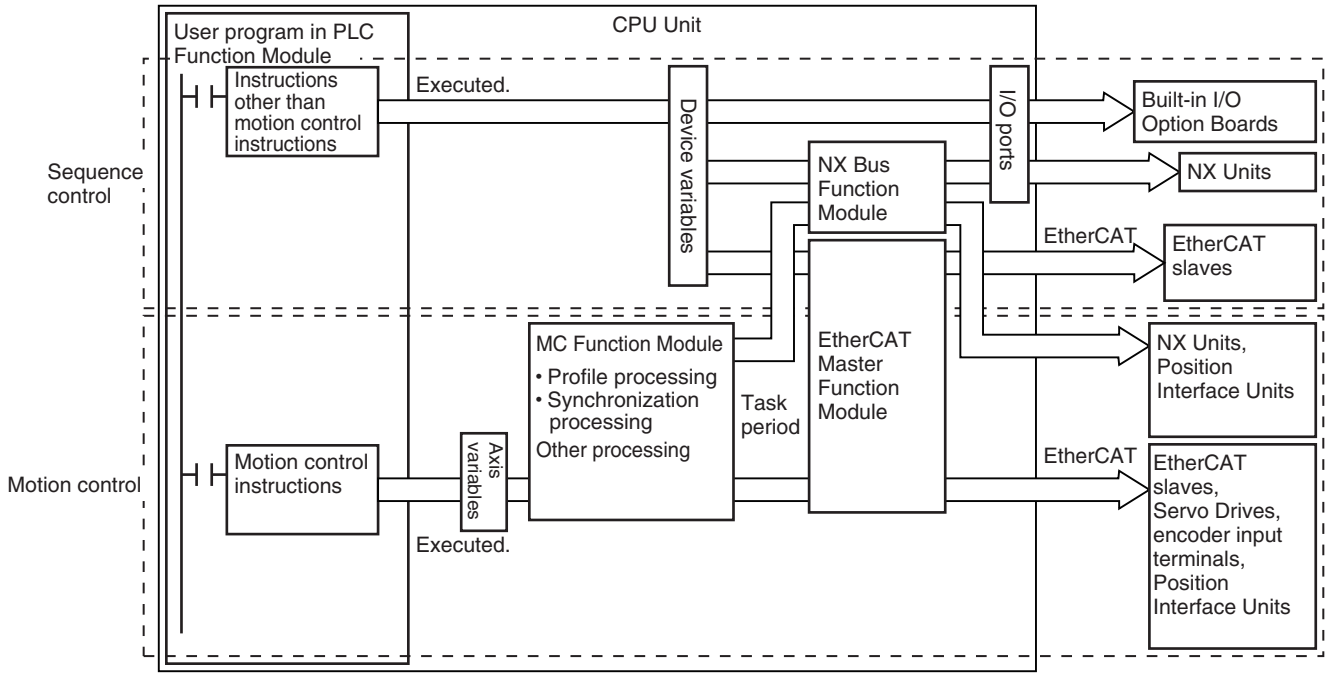
● NX502 CPU Unit



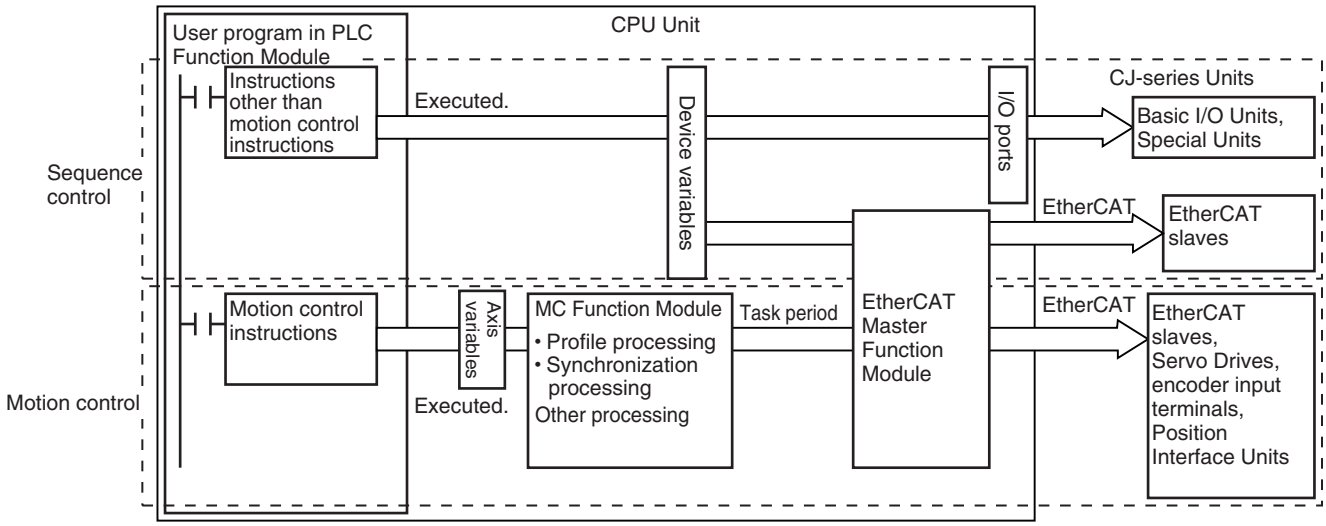
● NX102 CPU Unit



● NX1P2 CPU Unit



● NJ-series CPU Unit



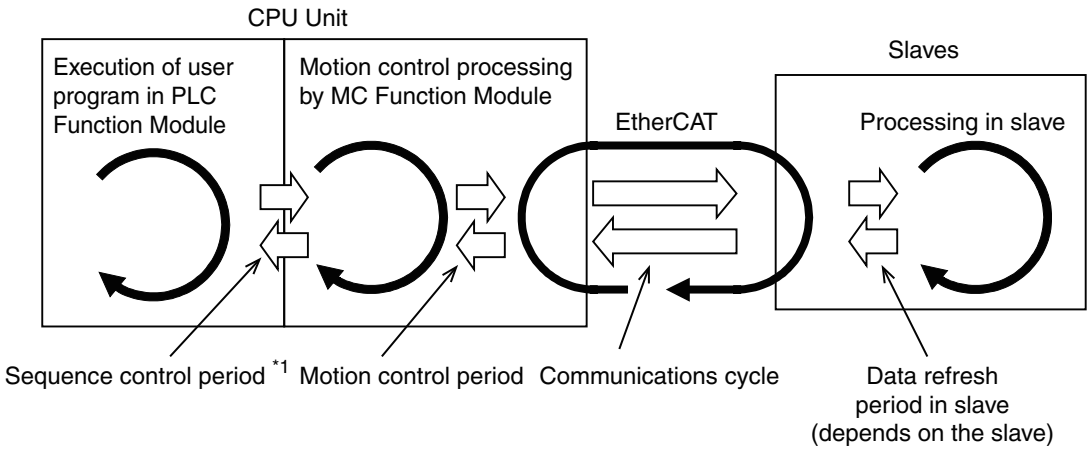
2-4-3 Relationship between Process Data Communications Cycle and Motion Control Period

The PLC Function Module sends motion control commands to the MC Function Module when motion control instructions are executed in the user program. The MC Function Module then performs motion control processing based on those commands and sends the results of processing as commands to the EtherCAT's Servo Drive or other device.

This type of data exchange is updated as shown in the following processing period.

● NX701 CPU Unit

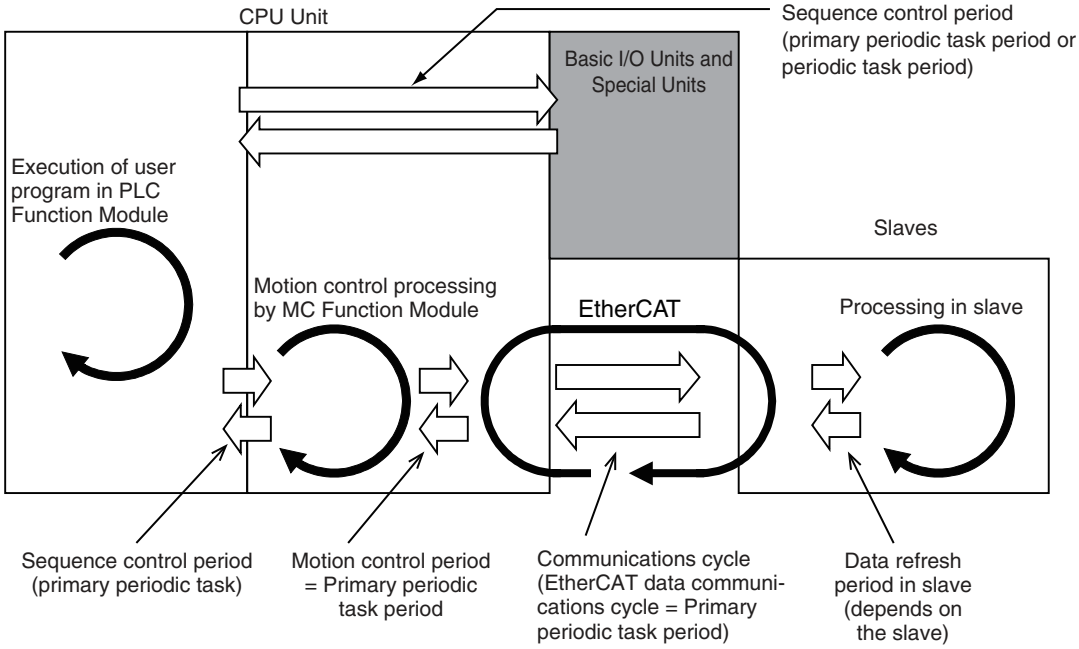
- Primary period = Motion control 1 period = Process data communications cycle for EtherCAT communications 1
- Task period of the priority-5 periodic task = Motion control 2 period = Process data communications cycle for EtherCAT communications 2



*1. If the sequence control period is primary period, the motion control period and the communications cycle are primary period.
If the sequence control period is the task period of the priority-5 periodic task, the motion control period and the communications cycle are the task period of the priority-5 periodic task.

● NX502 CPU Unit, NX102 CPU Unit, NX1P2 CPU Unit, and NJ-series CPU Unit

- Primary period = Motion control period = Process data communications cycle for EtherCAT communications



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 function to operate on the Sysmac Studio, and the instructions for creating and configuring axes and axes groups using the Sysmac Studio.

3-1 Axes

This section describes the axes that are used in a MC Function Module.

3-1-1 Introduction to Axes

In a motion control system, the targets of motion control are called axes.
An axis can be an actual Servo Drive, encoder, or other device connected via EtherCAT communications or it can be a virtual Servo Drive or encoder within the MC Function Module.

Although all axes of NX701, NX502, and NJ-series CPU Units are motion control axes, NX102 CPU Units and NX1P2 CPU Units allow the use of single-axis position control axes in addition to motion control axes. You can set each axis to Single-axis Position Control Axis in **Control function** in the Axis Basic Settings.
Refer to *Control Function* on page 5-11 for details on the **Control function**.

The MC Function Module supports the axis types that are given in the following table.

Axis type	Description
Servo axis	These axes are used by the EtherCAT slave Servo Drives and NX-series Position Interface Units.*1 They are assigned to actual Servo Drives or other devices. One Servomotor is used as one axis. If you use NX-series Position Interface Units, you can assign more than one device, such as a Pulse Output Unit and Digital Input Unit, to the same axis.
Virtual servo axis	These virtual axes exist only inside the MC Function Module. They are not used by actual Servo Drives. For example, they are used as master axes for synchronizing control.
Encoder axis	These axes are used by the EtherCAT slave Encoder Input Terminals and NX-series Position Interface Units.*1 An encoder axis is assigned to an actual encoder input terminal or other device. If one encoder input terminal contains two encoder inputs, the individual encoder inputs will act as one axis.
Virtual encoder axis	These axes are used virtually for encoder operation. A virtual encoder axis is used temporarily in place of an encoder axis when there is no physical encoder. *2

*1. Refer to *1-4-3 Function Specifications* on page 1-12 for the controllable devices.
*2. Virtual encoder axes are used in combination with motion control instructions that update the actual position of the virtual encoder axis.
Counting cannot be used with versions of the MC Function Module that do not support these instructions.

The following elements are related to the axes of the MC Function Module.
The number of elements provided is the same as the maximum number of controlled axes for each model. The maximum number of controlled axes varies depending on the model.
Refer to *1-4-2 Performance Specifications* on page 1-7 for details.

Configuration element	Description	Page
Axis parameters	The axis parameters set the maximum velocity, jogging, homing, and other items for the axes operations controlled by the MC Function Module. Use the Sysmac Studio to set the axis parameters.	page 3-3

Configuration element	Description	Page
Axis Variables	Axis Variables are system-defined variables for the actual position, error information, and other monitor information for axes controlled by the MC Function Module. Axis Variables are created when you add an axis from the Multiview Explorer of the Sysmac Studio. The names of the Axis Variables (called the Axis Variable names) are set here.	page 3-6
Specifying axes in the user program	In the user program, motion control is implemented with motion control instructions. Motion control instructions that perform single-axis control are used to create axis commands. To control an axis with axis commands, specify the Axis Variable name of the system-defined variable or the Axis Variable name that was set with the Sysmac Studio for the *Axis* in-out variable of the instruction.	page 3-9

3-1-2 Introduction to Axis Parameters

● Axis Parameters

Classification	Parameter name
Axis Basic Settings	Axis Number
	Motion Control*1
	Axis Use
	Axis Type
	Control Function*2
	Input Device/Output Device
Unit Conversion Settings	Unit of Display
	Command Pulse Count Per Motor Rotation
	Work Travel Distance Per Motor Rotation
	Reducer Use*3
	Work Travel Distance Per Rotation*3
	Work Gear Ratio*3
	Motor Gear Ratio*3
Operation Settings	Maximum Velocity
	Start Velocity*4
	Maximum Jog Velocity
	Maximum Acceleration
	Maximum Deceleration
	Acceleration/Deceleration Over
	Operation Selection at Reversing
	Velocity Warning Value
	Acceleration Warning Value
	Deceleration Warning Value
	Positive Torque Warning Value*5
	Negative Torque Warning Value*5
	Actual Velocity Filter Time Constant
	In-position Range
	In-position Check Time

Classification	Parameter name
	Zero Position Range
Other Operation Settings	Immediate Stop Input Stop Method
	Limit Input Stop Method
	Drive Error Reset Monitoring Time
	Maximum Positive Torque Limit
	Maximum Negative Torque Limit
	Immediate Stop Input Logic Inversion*4
	Positive Limit Input Logic Inversion*4
	Negative Limit Input Logic Inversion*4
	Home Proximity Input Logic Inversion*4
Limit Settings	Software Limits
	Positive Software Limit
	Negative Software Limit
	Following Error Over Value
	Following Error Warning Value
Position Count Settings	Count Mode
	Modulo Maximum Position Setting Value
	Modulo Minimum Position Setting Value
	Encoder Type
Servo Drive Settings	Modulo Maximum Position Setting Value
	Modulo Minimum Position Setting Value
	PDS State Control Method
Homing Settings	Homing Method
	Home Input Signal
	Homing Start Direction
	Home Input Detection Direction
	Operation Selection at Positive Limit Input
	Operation Selection at Negative Limit Input
	Homing Velocity
	Homing Approach Velocity
	Homing Acceleration
	Homing Deceleration
	Homing Jerk
	Home Input Mask Distance
	Home Offset
	Homing Holding Time
	Homing Compensation Value
	Homing Compensation Velocity

*1. Set this parameter when using the NX701 CPU Unit.
*2. Set this parameter when using the NX102 CPU Unit and NX1P2 CPU Unit.
*3. A CPU Unit with unit version 1.11 or later and Sysmac Studio version 1.15 or higher are required to use this parameter.
*4. A CPU Unit with unit version 1.05 or later and Sysmac Studio version 1.06 or higher are required to use this parameter.
*5. This parameter is enabled only for torque control.

Refer to *5-2 Axis Parameters* on page 5-5 for details on axis parameters.

● Settings Required to Use Axes

The following settings must be made to use the axes that are created with the Sysmac Studio.

Classification	Parameter name	Setting	Page
Axis Basic Settings	Axis Number	Axis numbers are automatically set in the order that the axes are created.	page 5-8
	Motion Control*1	Select **Primary periodic task**.	
	Axis Use	Select **Used axis**.	
	Axis Type	Select the type of axis to control.	
	Control Function*2	Select **All**. *3	
	Input Device/ Output Device	Specify the node address of the EtherCAT slave device that is assigned to the axis. The Node Address parameter cannot be selected if the Axis Type parameter is set to use a **virtual axis**.	

*1. Set this parameter when using the NX701 CPU Unit.
*2. Set this parameter when using the NX102 CPU Unit and NX1P2 CPU Unit.
*3. To use the axis as a motion control axis, select **All**. To use the axis as a single-axis position control axis, select **Single-axis position control only**.

● Required Settings to Perform a Servo Drive Test Run from the Sysmac Studio

Make the following settings to operate an EtherCAT-connected Servo Drive or other device using the MC Test Run function of the Sysmac Studio.

Classification	Parameter name	Setting	Page
Axis Basic Settings	Axis Number	Axis numbers are automatically set in the order that the axes are created.	page 5-8
	Motion Control*1	Select **Primary periodic task**.	
	Axis Use	Select **Used axis**.	
	Axis Type	Select **Servo axis**.	
	Control Function*2	Select **All**. *3	
	Input Device/ Output Device	Specify the node address of the EtherCAT slave device that is assigned to the axis. The Node Address parameter cannot be selected if the Axis Type parameter is set to use a **virtual axis**.	
Unit Conversion Settings	Unit of Display	Select the display unit (mm, degrees, etc.).	page 5-13
	Command Pulse Count Per Motor Rotation	Set the number of command pulses per motor rotation according to the encoder resolution. *4	
	Work Travel Distance Per Motor Rotation	Set the workpiece travel distance per motor rotation according to the machine specifications.	
	Reducer Use*5	Specify whether to use the reducer setting or not.	
	Work Travel Distance Per Rotation*5	Set the work travel distance per rotation.	
	Work Gear Ratio*5	Set the gear ratio for the workpiece.	
	Motor Gear Ratio*5	Set the gear ratio of the motor.	

Classification	Parameter name	Setting	Page
Position Count Settings	Count Mode	Set this parameter according to the machine specifications.	page 5-26
Limit Settings	Software Limits	Set this parameter according to the device specifications.	page 5-25

*1. Set this parameter when using the NX701 CPU Unit.
*2. Set this parameter when using the NX102 CPU Unit and NX1P2 CPU Unit.
*3. To use the axis as a motion control axis, select **All**. To use the axis as a single-axis position control axis, select **Single-axis position control only**.
*4. For example, if the encoder resolution is 10,000 pulses/rotation, set *10,000*.
*5. A CPU Unit with unit version 1.11 or later and Sysmac Studio version 1.15 or higher are required to use this parameter.

Precautions for Correct Use

- Select the appropriate values based on the machine's operating conditions for parameters such as the maximum velocity, maximum acceleration/deceleration, or stop settings when the motor is actually operated.
- OMRON 1S-series Servo Drives and G5-series Servo Drives can be set to specific node addresses by using the rotary switches on the front panels.
  If the rotary switches are set to *00*, the node address will be determined by the settings made in the EtherCAT Editor of the Sysmac Studio.
  If the rotary switches are set to *00* for all connected Servo Drives, errors will not occur even if the Servo Drive's connection position is changed. Set the node addresses on the rotary switches to assign specific Servo Drives for each machine control.

3-1-3 Introduction to Axis Variables

Axis Variables are system-defined variables for some of the axis parameters and for the monitor information, such as the actual position and error information, for the axes controlled by the MC Function Module.
When you create axes with the Sysmac Studio, Axis Variables are registered in the variable table in the order that the axes are created.
Axis variables are structures with a data type of *_sAXIS_REF*.

Axis Variable Names

Each Axis Variable in the MC Function Module has two variable names: The Axis Variable name in the system-defined variables and the Axis Variable name that is assigned when the axis is added on the Sysmac Studio.
The Axis Variable names in the system-defined variables are *_MC_AX[0] to _MC_AX[255]*, *_MC1_AX[0] to _MC1_AX[255]*, *_MC2_AX[0] to _MC2_AX[255]*.
When you add axes on the Sysmac Studio, the *MC_Axis000 to MC_Axis255* are set by default for *_MC_AX[0] to _MC_AX[255]*.
The numbers are assigned in the order that the axes are added. You can change each of these Axis Variables as required from the Sysmac Studio.
You can use either the Axis Variables for the system-defined variables or the Axis Variables that are added on the Sysmac Studio to specify the Axis Variables in the user program.

● Example When _MC_AX[0-255] Is Used

Axis Variable name in the system-defined variables (AT specification in global variable table*1)	Default Axis Variable name when axis is added on the Sysmac Studio	Axis number example
_MC_AX[0]	MC_Axis000	Axis 0
_MC_AX[1]	MC_Axis001	Axis 1
:	:	:
_MC_AX[255]	MC_Axis255	Axis 255

*1. An error will occur if you change the names in the AT column in the global variable table on the Sysmac Studio.

Additional Information

- *_MC_AX[0-255]* or *_MC1_AX[0-255]*, and *_MC2_AX[0-255]* are available for the NX701 CPU Unit.
  Only *_MC_AX[0-127]* is available for the NX502 CPU Unit.
  Only *_MC_AX[0-14]* is available for the NX102 CPU Unit.
  Only *_MC_AX[0-11]* is available for the NX1P2 CPU Unit.
  Only *_MC_AX[0-63]* is available for the NJ-series CPU Unit.
- For the NX701 CPU Unit, you can access the same values of *_MC_AX[0-255]* and *_MC1_AX[0-255]* if the axis numbers of them are the same. You can use either of the Axis Variables, or both of them at the same time.
- The Axis Variable assigned to primary periodic task is *_MC_AX[0-255]* or *_MC1_AX[0-255]*. The Axis Variable assigned to priority-5 periodic task is *_MC2_AX[0-255]*.

Examples of Axis Variable Levels and Changing Axis Variable Names

In the descriptions below, _MC_AX[0] is used as an example. The same information applies to the other variables.

_MC_AX[0]	Axis Variable
_MC_AX[0].Status	Level that indicates the axis status
_MC_AX[0].Status.Ready	Variable that indicates that the axis is ready for operation
_MC_AX[0].Status.Disabled	Variable that indicates when the axis is disabled
:	
_MC_AX[0].Details	Level that indicates the axis control status
_MC_AX[0].Details.Idle	Variable that indicates when the axis is idle
_MC_AX[0].Details.InPosWaiting	Variable that indicates in-position waiting
:	
_MC_AX[0].Cmd	Level that indicates the axis command values
_MC_AX[0].Cmd.Pos	Variable that indicates the command current position
_MC_AX[0].Cmd.Vel	Variable that indicates the command current velocity
_MC_AX[0].Cmd.AccDec	Variable that indicates the command current acceleration/deceleration rate in the axis monitor
:	
_MC_AX[0].Act	Level that indicates the axis current values
_MC_AX[0].Act.Pos	Variable that indicates the actual current position
_MC_AX[0].Act.Vel	Variable that indicates the actual current velocity
:	

_MC_AX[0].Cfg	Level that indicates the axis basic settings
_MC_AX[0].Cfg.AxNo	Variable that indicates the axis number
_MC_AX[0].Cfg.AxEnable	Variable that indicates when the axis is enabled
_MC_AX[0].Cfg.AxType	Variable that indicates the axis type
:	
_MC_AX[0].Scale.Units	Variable that indicates the display unit
_MC_AX[1]	Axis Variable
:	

- **Example:**

  If *MC_Axis000* is changed to *MyAxis1*, then either *MyAxis1.Act.Pos* or *_MC_AX[0].Act.Pos* can be used as the variable that indicates the actual current position.

  Refer to *Axis Variables* on page 6-25 for details on Axis Variables.

3-1-4 Synchronizing Axis Variables

The user program for the priority-5 periodic task can access the values of an Axis Variable of an axis that is controlled in the primary periodic task. The reverse is possible: the user program for the primary periodic task can access the values of an Axis Variable of an axis that is controlled in the priority-5 periodic task.
Also, the user program for the priority-16 periodic task can access the values of an Axis Variable that is controlled in the primary periodic task.
The following table shows the relationship between the types of accessing tasks and the Axis Variables.

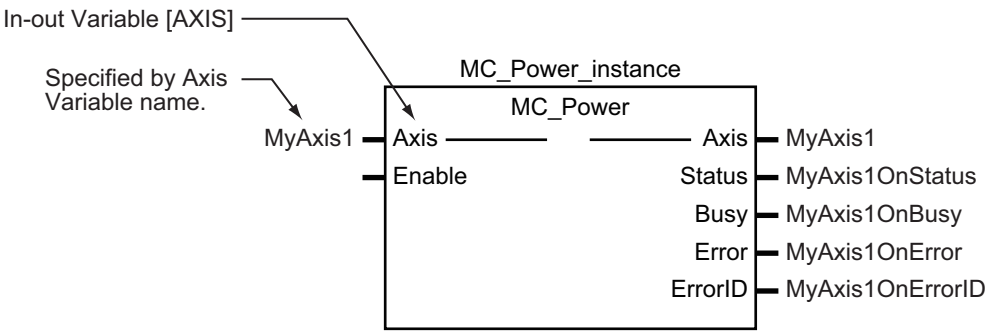
○: Access possible, ×: Access not possible

Task type	Axis Variable for motion control 1		Axis Variable for motion control 2	
	User defined*1	System defined*2	User defined*1	System defined*3
Primary periodic task	○	○	○*4	×
Priority-5 periodic task	○*5	×	○	○
Priority-16 periodic task	○*5	○*6	○*4	×
Priority-17 or priority-18 periodic task	○*5	○*7	○*4	○*8
Priority-8 or priority-48 event task	○*5	○*7	○*4	○*8

*1. The user-defined Axis Variables, such as *MC_Axis000* and *Loader1*, that are automatically generated when you create the axes on the Sysmac Studio. We recommend that you use the user-defined Axis Variables.
*2. *_MC_AX[*]* or *_MC1_AX[*]*
*3. *_MC2_AX[*]*
*4. The Axis Variable for axis 1 is processed every task period of the priority-5 periodic task.
*5. The Axis Variable for axis 1 is processed every primary task period.
*6. The Axis Variables for the maximum number of axes are processed every primary task period. It is not recommended because the task execution time of the primary periodic task is increased.
*7. The Axis Variables for the maximum number of axes are processed every primary task period.
*8. The Axis Variables for the maximum number of axes are processed every task period of the priority-5 periodic task.

3-1-5 Specifying an Axis in the User Program

In the user program, an Axis Variable name is specified for the in-out variable *Axis* in motion control instructions.
In the following example, the Axis Variable name for the axis that was added for the system-defined Axis Variable name of *_MC_AX[0]* has been changed to *MyAxis1* in the Sysmac Studio.



You can also use the *_MC_AX[0]* system-defined variable in place of *MyAxis1*.

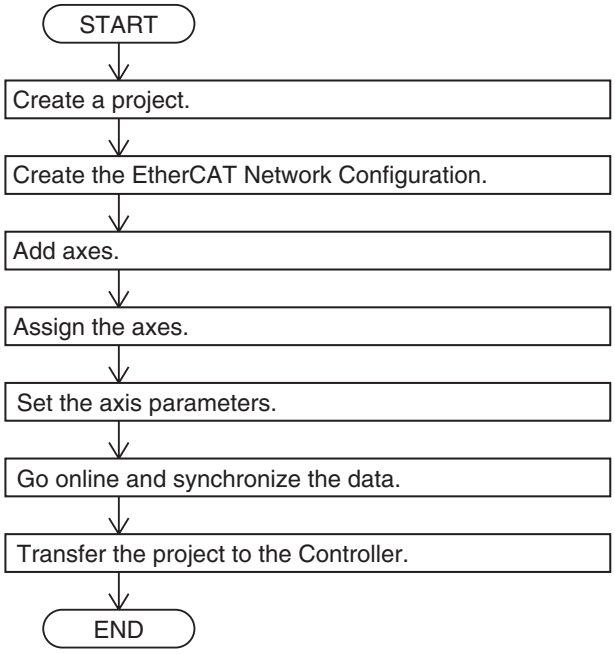
Refer to *6-2 Motion Control Instructions* on page 6-5 for details on motion control instructions.

Refer to the instruction descriptions in the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)* for details on motion control instructions.

3-2 Axis Setting Procedure

This section gives the procedures to set servo axes that are newly created with the Sysmac Studio.

3-2-1 Axis Configuration Procedure

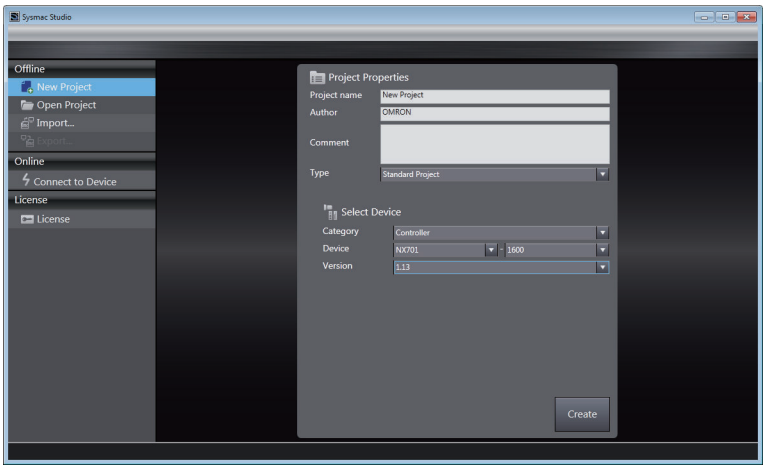


3-2-2 Setting Procedure

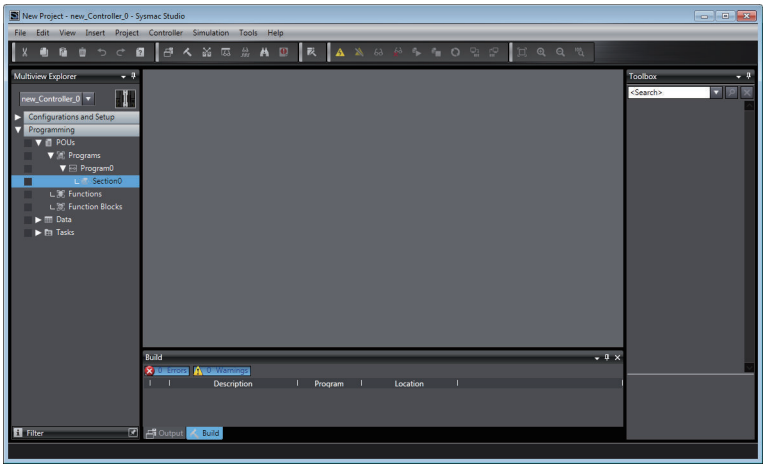
This section describes how to set an axis.

Starting the Sysmac Studio

1. Start the Sysmac Studio and click the **New Project** Button.
2. Set the project properties, select the device, and click the **Create** Button.



A new project is displayed.



Creating the EtherCAT Network Configuration

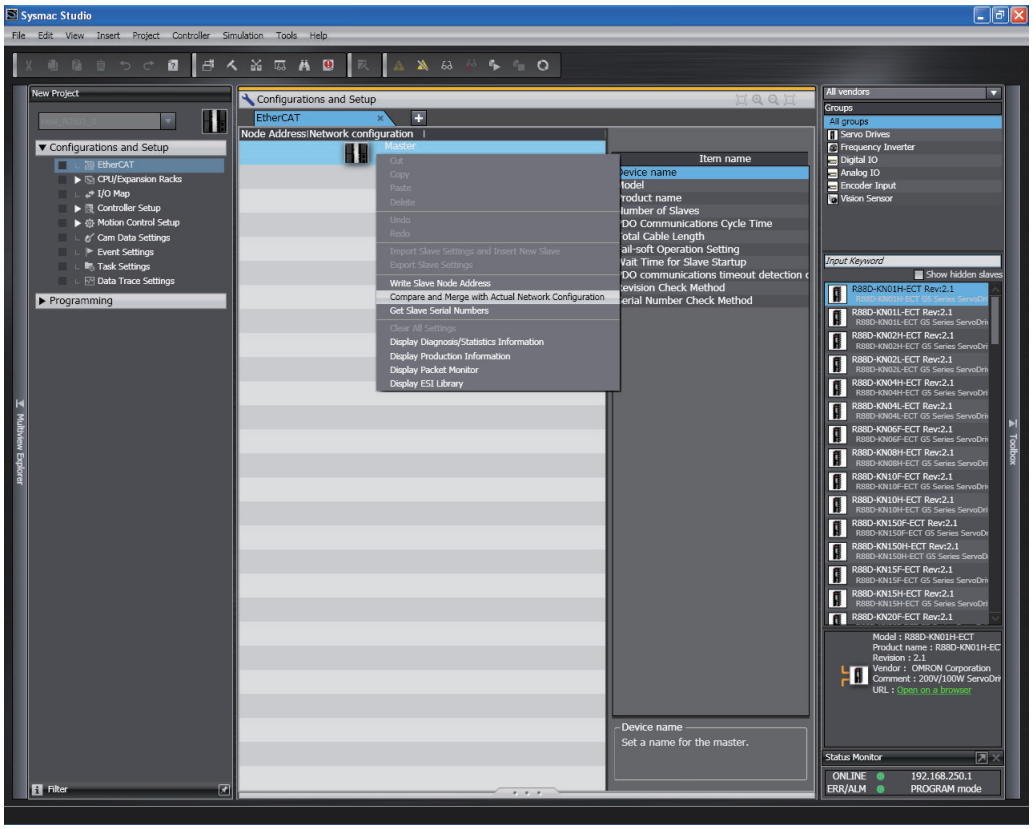
There are two methods to create an EtherCAT Network Configuration: online and offline.

● Online Method

1 Double-click **EtherCAT** in the Multiview Explorer.
The EtherCAT Tab Page is displayed.

2 Select **Online** from the **Controller** Menu. The Sysmac Studio goes online with the Controller.

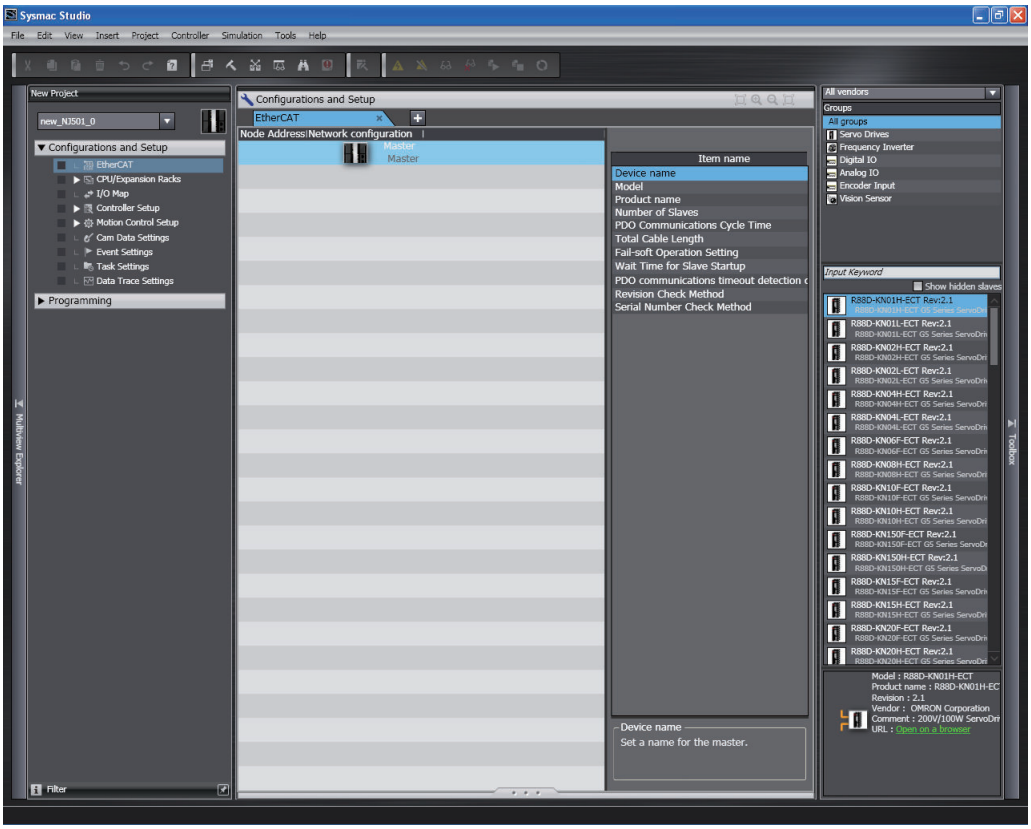
3 Right-click the **Master** Icon in the EtherCAT Tab Page and select **Compare and Merge with Actual Network Configuration** from the menu.



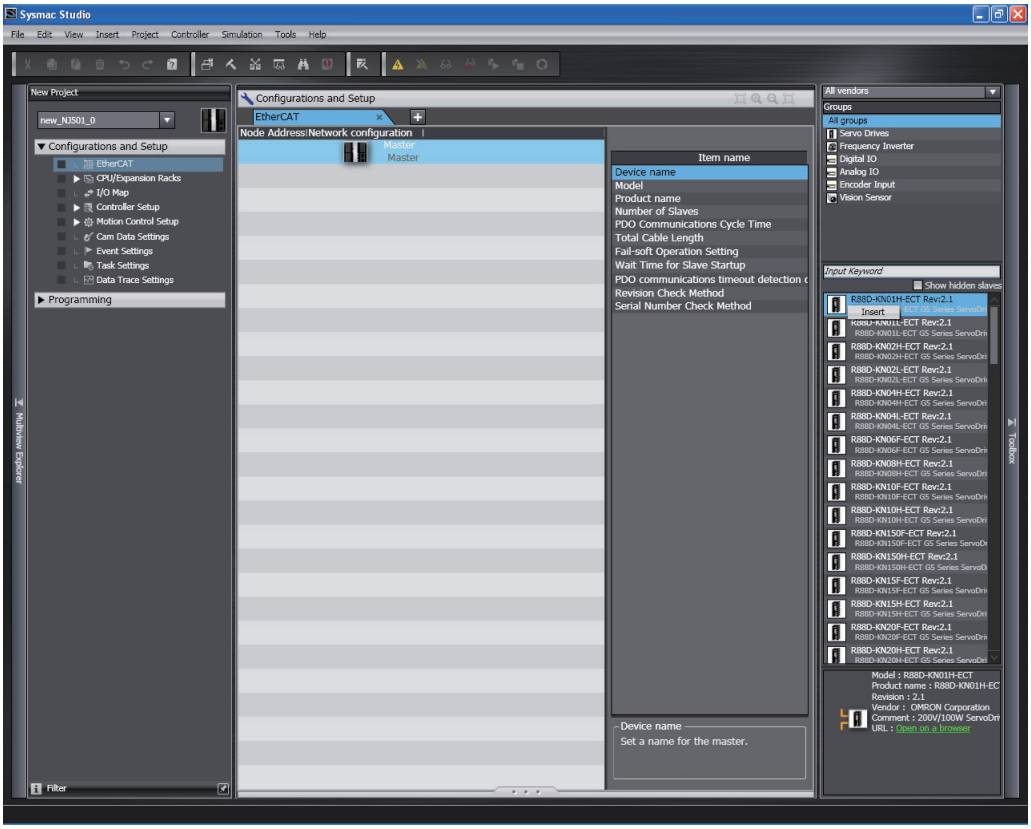
When obtaining the information is completed, the physical slave configuration of the EtherCAT slaves is displayed.
Right-click the displayed physical configuration and select **Apply actual network configuration**.

● Offline Method

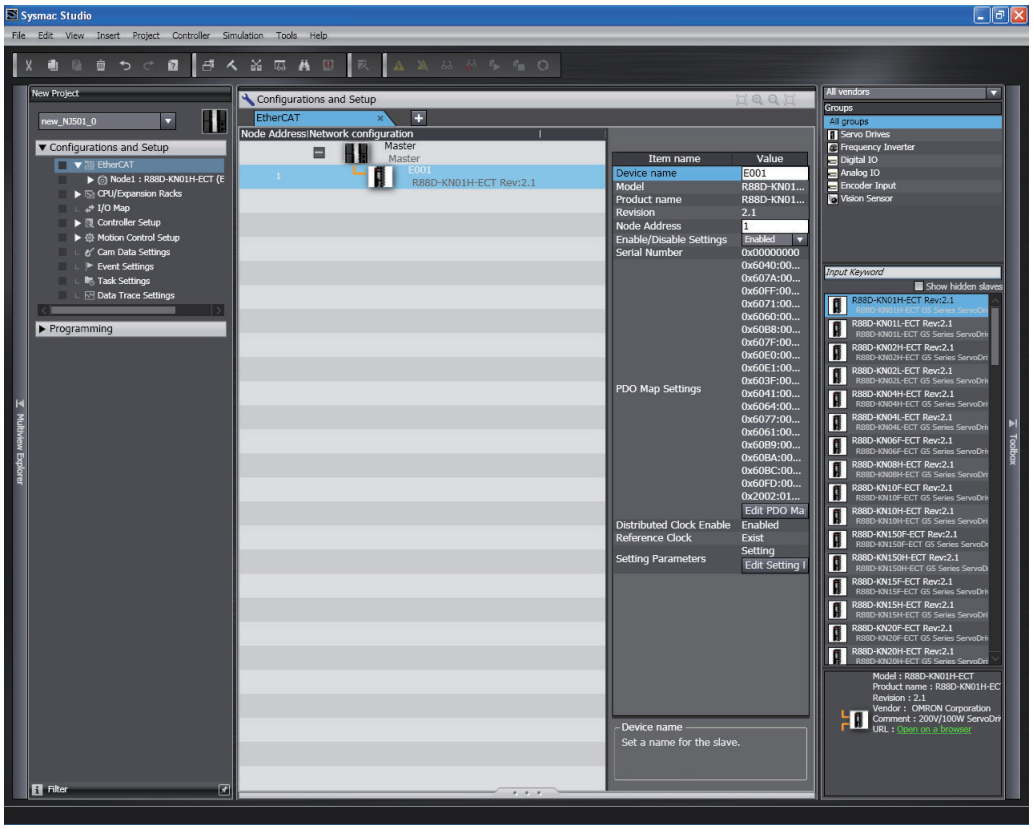
1 Double-click **EtherCAT** in the Multiview Explorer.
The EtherCAT Tab Page is displayed.



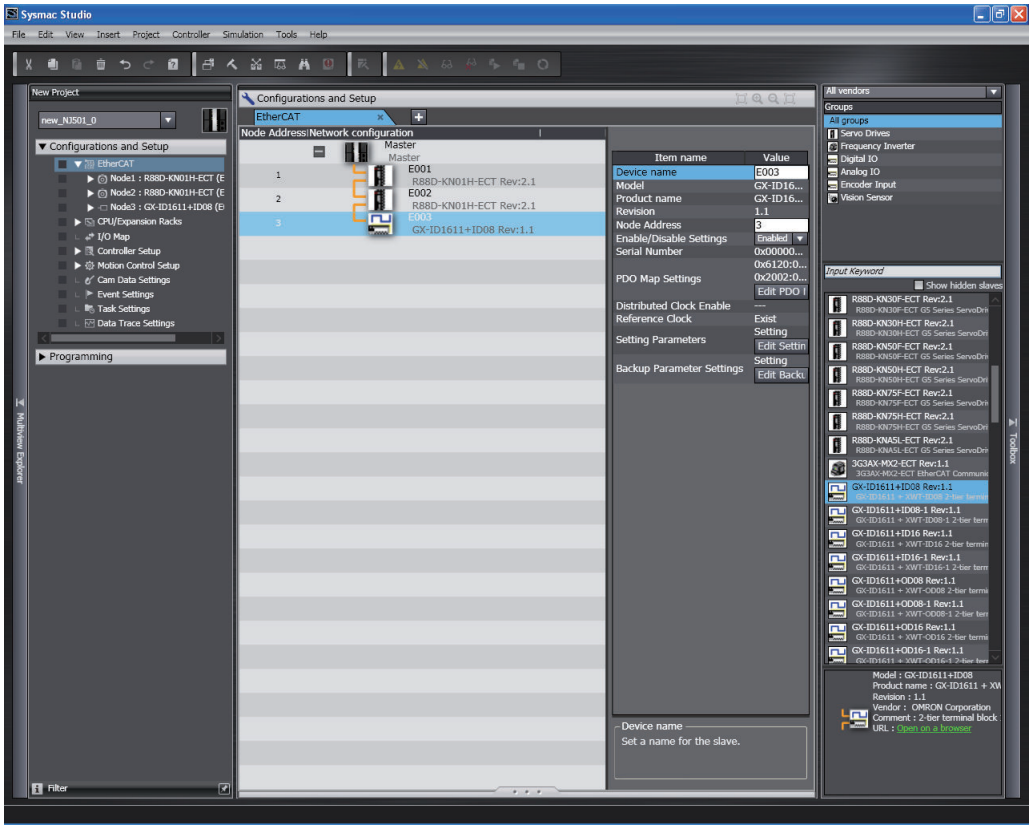
2 Right-click the slave to connect and select **Insert** from the menu.



The slave is inserted on the display.



3 Insert the remaining slaves.

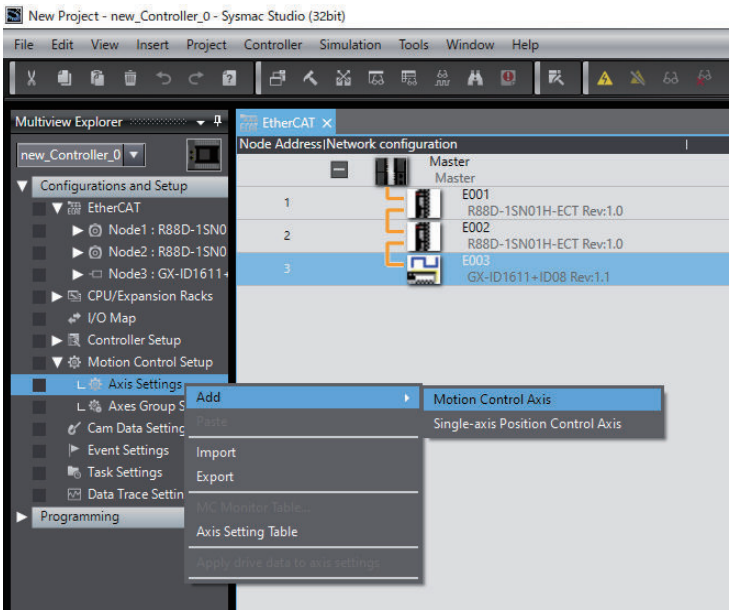


Adding Axes

1 Right-click **Axis Settings** in the Multiview Explorer and select **Motion Control Axis** or **Single-axis Position Control Axis** from the **Add** Menu.

Additional Information

- **Single-axis Position Control Axis** is displayed for the NX102 CPU Unit and NX1P2 CPU Unit.
- Refer to *Control Function* on page 5-11 for details on the differences between **Motion Control Axis** and **Single-axis Position Control Axis**.



An axis is added to the Multiview Explorer. The default name for the new axis is *MC_Axis000*.

Additional Information

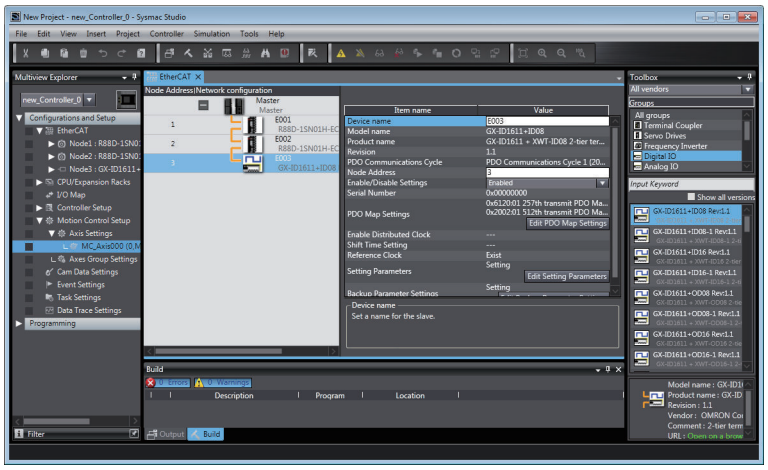
Changing Axis Variable Names in the User Program

Perform the following two procedures to change Axis Variable names that are already used.

- Change the Axis Variable name in the variable table in the variable declarations.
- Change the Axis Variable name in the user program.

Even if you change the Axis Variable names in the variable table, the Axis Variable names in the user program do not change.

An error will occur if you use a variable name that is not declared in the variable table, in the user program. Always change the names in both places.



- **Adding an Axis by Import**

  You can also add **axis settings** by importing a previously created file of axis settings.
  You can create the file of axis settings by exporting the **axis settings**.
  Refer to the *Sysmac Studio Version 1 Operation Manual (Cat. No. W504)* for details.

Version Information

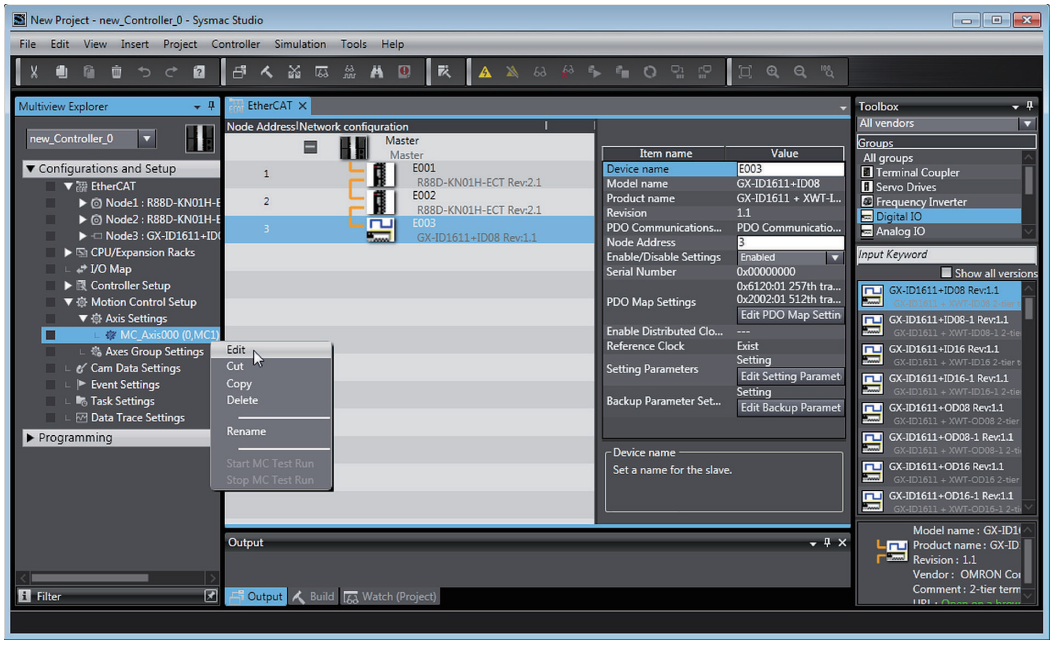
Import and **export** are available in Sysmac Studio Ver.1.57 or higher.

- **Copying an Axis**

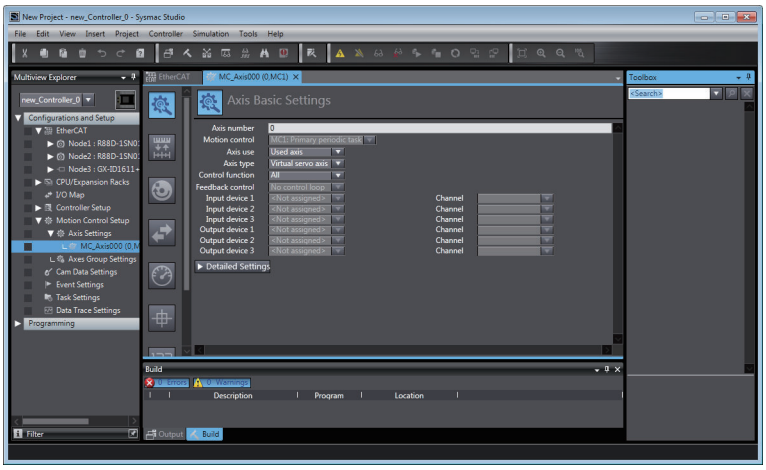
  You can also add an axis by copying the axis settings for an existing axis.

Assigning an Axis

1 Right-click an axis in the Multiview Explorer and select **Edit** from the menu.



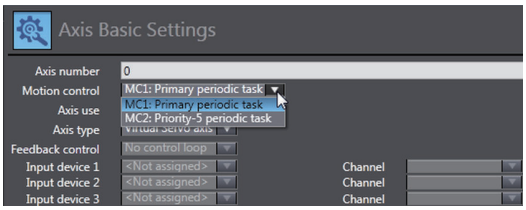
The **Axis Basic Settings** are displayed in the Axis Parameter Settings Tab Page.



Additional Information

Control function is displayed for the NX102 CPU Unit and NX1P2 CPU Unit.

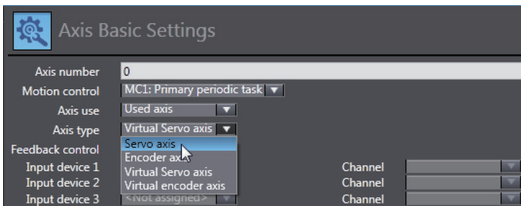
2 Select **Primary periodic task** or **Priority-5 periodic task** from **Motion control**.



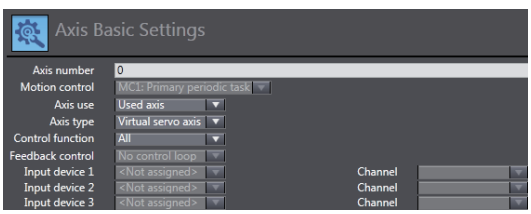
Additional Information

The setting is available for the NX701 CPU Unit.

3 Select **Servo axis** in the **Axis type**.



4 Select **All** in the **Control function**.



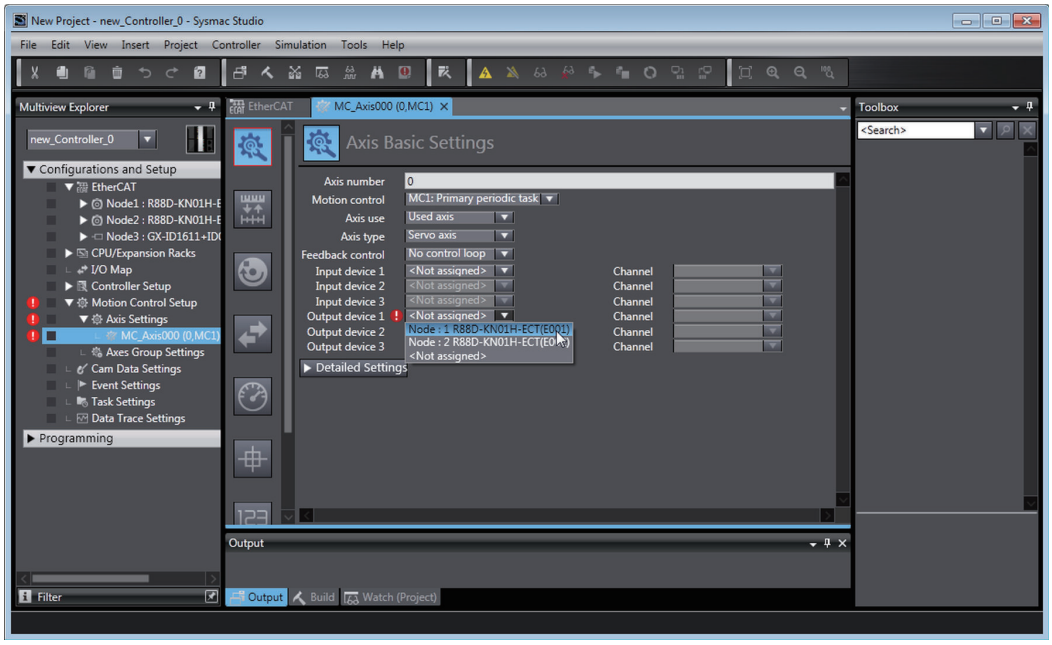
Additional Information

- You can select this parameter for the NX102 CPU Unit and NX1P2 CPU Unit.
- To use the axis as a motion control axis, select **All**. To use the axis as a single-axis position control axis, select **Single-axis position control only**.

5 Select the Servo Drive to use.

Select the Servo Drive to use in **Output device 1**.

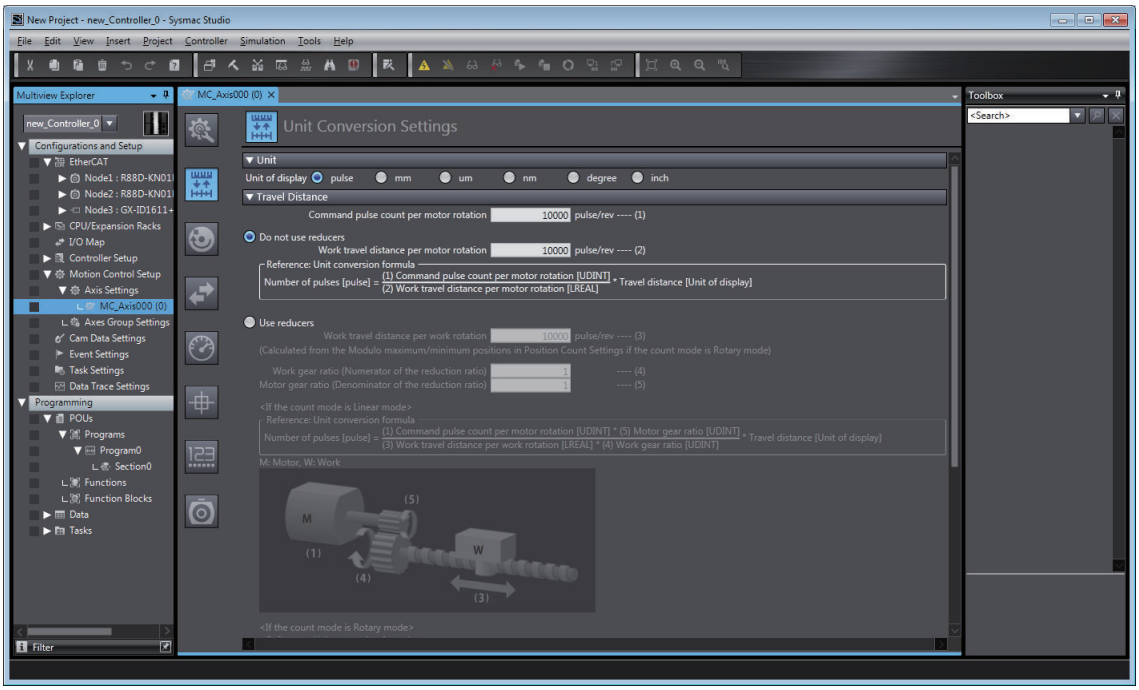
This setting allows you to use the EtherCAT slave Servo Drive as an axis.



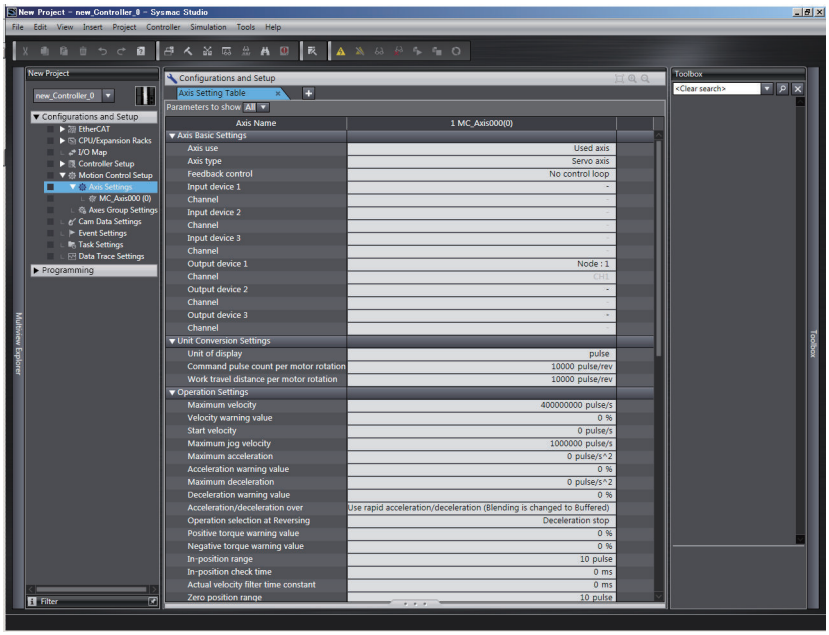
Setting Axis Parameters

Click each of the icons in the Axis Parameter Settings Tab Page.

The settings for each icon are displayed on the Axis Parameter Settings Tab Page.



Right-click **Axis Settings** in the Multiview Explorer and select **Axis Setting Table** to enable setting the axis parameters for all axes at the same time.



Precautions for Correct Use

When making operation settings such as the display unit, electronic gear (unit conversion formula), maximum velocity, or maximum acceleration/deceleration, be sure to use appropriate values for the operating conditions of the device.

Additional Information

Changing Axis Variable Names in the User Program

Perform the following two procedures to change Axis Variable names that are already used.

- Change the Axis Variable name in the variable table in the variable declarations.
- Change the Axis Variable name in the user program.

Even if you change the Axis Variable names in the variable table, the Axis Variable names in the user program do not change.

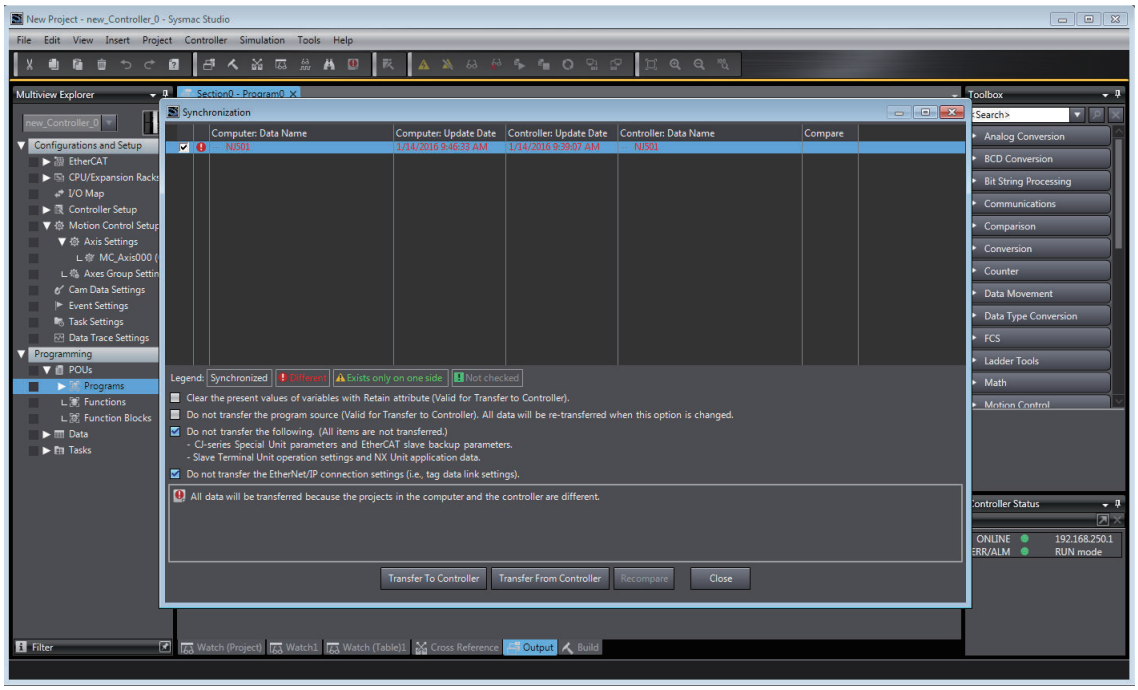
An error will occur if you use a variable name that is not declared in the variable table, in the user program. Always change the names in both places.

Downloading to the CPU Unit

Use the Synchronization menu command of the Sysmac Studio to download the project to the NJ/NX-series CPU Unit.

1 Select **Online** from the **Controller** Menu. The Sysmac Studio goes online with the Controller.

2 Select **Synchronization** from the **Controller** Menu and then click the **Transfer to Controller** Button.



Additional Information

The MC Function Module connects to OMRON 1S-series Servo Drives with built-in EtherCAT communications, G5-series Servo Drives with built-in EtherCAT communications, or NX-series Pulse Output Units.

Connectable Servo Drive Models

- You can connect the R88D-1SN□□□-ECT, R88D-1SAN□□□-ECT, R88D-KN□□□-ECT, and R88D-KN□□□-ECT-L Servo Drives.
- The R88D-KN□□□-ECT-R Servo Drives support only Position Control Mode (Cyclic Synchronous Position Control Mode). Therefore, any functions that use Velocity Control Mode (Cyclic Synchronous Velocity Control Mode) or Torque Control Mode (Cyclic Synchronous Torque Control Mode) cannot be used.

Servo Drive Settings

- The MC Function Module uses some of the input signals and functions of the Servo Drives. Servo Drive signal wiring and object setting are required to use the MC Function Module properly.
  Refer to *A-1 Connecting the 1S-series Servo Drive* on page A-2 or *A-2 Connecting the G5-series Servo Drive* on page A-11 for specific settings.

Refer to the *NX-series Position Interface Units User's Manual (Cat. No. W524)* for the settings to use NX-series Pulse Output Units.

3-3 Axes Groups

This section describes the axes groups of the MC Function Module.

3-3-1 Introduction to Axes Groups

Use axes groups to perform complex operations on multiple axes, such as linear or circular interpolation.
An axes group consists of multiple axes.
Use the Sysmac Studio to set Axes Group Variables to enable execution of axes group motion control instructions or to enable access of the status of the axes group.
The MC Function Module can handle up to 64 groups.
The specifications for axes groups are shown in the following table.

Item	Specifications					
	NX701	NX502-		NX102	NX1P2	NJ-series
		17□□*1 16□□*1	15□□ 14□□ 13□□			
Number of axes groups	64 groups max.	64 groups max.	32 groups max.	8 groups max.	8 groups max.	32 groups max.
Number of composition axes	4 axes max. per axes group					

*1. Models added from the CPU Unit version 1.66.

The following elements are related to the axes groups of the MC Function Module.

Configuration element	Description	Page
Axes group parameters	The axes group parameters set the maximum interpolation velocity, maximum interpolation acceleration/deceleration, and other items for the axes groups controlled by the MC Function Module. Use the Sysmac Studio to set the axes group parameters.	page 3-23
Axes Group Variables	Axes Group Variables are system-defined variables that include a portion of the axes group parameters as well as the command interpolation velocity, error information, and other monitor information for the axes groups controlled by the MC Function Module. Axes Group Variables are created when you add an axes group from the Multiview Explorer of the Sysmac Studio. The names of the Axes Group Variables (called the Axes Group Variable names) are set here.	page 3-24
Specifying axes groups in the user program	In the user program, motion control is implemented with motion control instructions. Motion control instructions that perform multi-axes coordinated control are called *axes group commands*. To control an axes group with axes group commands, specify the axes group variable name of the system-defined variable or the axes group variable name that was set with the Sysmac Studio for the *AxesGroup* in-out variable of the instruction.	page 3-25

Precautions for Correct Use

Do not configure an axes group with axes on the EtherCAT network and axes on the CPU Unit. The timing of commands may not be synchronized because the synchronization between the networks is not performed. This may result in unintended operation.

3-3-2 Introduction to Axes Group Parameters

● Axes Group Parameters

Classification	Parameter name
Axes Group Basic Settings	Axes Group Number
	Motion Control*1
	Axes Group Use
	Composition
	Composition Axes
Axes Group Operation Settings	Maximum Interpolation Velocity
	Maximum Interpolation Acceleration
	Maximum Interpolation Deceleration
	Interpolation Acceleration/Deceleration Over
	Interpolation Velocity Warning Value
	Interpolation Acceleration Warning Value
	Interpolation Deceleration Warning Value
	Axes Group Stop Method
	Correction Allowance Ratio

*1. Set this parameter when using the NX701 CPU Unit.

Refer to *5-3 Axes Group Parameters* on page 5-34 for details on axes group parameters.

● Settings Required to Use an Axes Group

The following settings must be made to use the axes groups that are created with the Sysmac Studio.

Classification	Parameter name	Setting	Page
Axes Group Basic Settings	Axes Group Number	Axes group numbers are automatically set in the order that the axes groups are created.	page 5-35
	Motion Control*1	Select **Primary periodic task**.	
	Axes Group Use	Select **Use**.	
	Composition	Select the axis composition to control.	
	Composition Axes	This parameter sets the axes to assign to the axes group.	

*1. Set this parameter when using the NX701 CPU Unit.

Precautions for Correct Use

- Set appropriate values for the maximum interpolation velocity, stop method, and other items based on the operating conditions.
- An axis for which **Control function** is set to **Single-axis position control only** cannot be allocated as an axis in an axes group.

3-3-3 Introduction to Axes Group Variables

Axes Group Variables are system-defined variables for the setting information and the monitoring information, such as the actual position and error information, for the axes groups controlled by the MC Function Module.
When you create axes groups with the Sysmac Studio, Axes Group Variables are registered in the variable table in the order that the axes groups are created. Axes Group Variables are structures with a data type of *_sGROUP_REF*.

Axes Group Variable Names

Each Axes Group Variable in the MC Function Module has two variable names: The Axes Group Variable name in the system-defined variables and the Axes Group Variable that is assigned when the axes group is added on the Sysmac Studio.
The Axes Group Variable names in the system-defined variables are *_MC_GRP[0] to _MC_GRP[63]*, *_MC1_GRP[0] to _MC1_GRP[63]*, *_MC2_GRP[0] to _MC2_GRP[63]*.
When you add axes groups on the Sysmac Studio, *MC_Group000 to MC_Group063* are set by default for *_MC_GRP[0] to _MC_GRP[63]*.
The numbers are assigned in the order that the axes are added. You can change each of these Axes Group Variable names as required from the Sysmac Studio.
You can use either the Axes Group Variable names for the system-defined variables or the Axes Group Variable names that are set on the Sysmac Studio to specify the Axes Group Variables in the user program.

● Example When _MC_GRP[0-63] Is Used

Axes Group Variable name in the system-defined variables (AT specification in global variable table*1)	Default Axes Group Variable name when axes group is added on Sysmac Studio	Axes group number example
_MC_GRP[0]	MC_Group000	Axes group 0
_MC_GRP[1]	MC_Group001	Axes group 1
:	:	:
_MC_GRP[63]	MC_Group063	Axes group 63

*1. An error will occur if you change the names in the AT column in the global variable table on the Sysmac Studio.

Additional Information

- *_MC_GRP[0-63]*, *_MC1_GRP[0-63]*, and *_MC2_GRP[0-63]* are available for the NX701 CPU Unit.
  Only *_MC_GRP[0-31]* is available for the NX502 CPU Unit.
  Only *_MC_GRP[0-7]* is available for the NX102 CPU Unit and NX1P2 CPU Unit.
  Only *_MC_GRP[0-31]* is available for the NJ-series CPU Unit.
- For the NX701 CPU Unit, you can access the same values of *_MC_GRP[0-63]* and *_MC1_GRP[0-63]* if the axes group numbers of them are the same. You can use either of the Axes Group Variables, or both of them at the same time.
- The Axes Group Variable assigned to primary periodic task is *_MC_GRP[0-63]* or *_MC1_GRP[0-63]*.
  The Axes Group Variable assigned to priority-5 periodic task is *_MC2_GRP[0-63]*.

Examples of Axes Group Variable Levels and Changing Axes Group Variable Names

In the descriptions below, _MC_GRP[0] is used as an example. The same information applies to the other axes group variables.

_MC_GRP[0]	Axes Group Variable
_MC_GRP[0].Status	Level that indicates the axes group status
:	
_MC_GRP[0].Cmd	Level that indicates the axes group command values
_MC_GRP[0].Cmd.Vel	Variable that indicates the command interpolation velocity
_MC_GRP[0].Cmd.AccDec	Variable that indicates the command interpolation acceleration/deceleration rate
:	
_MC_GRP[0].Cfg	Level that indicates the axes group basic settings
_MC_GRP[0].Cfg.GrNo	Variable that indicates the axes group number
_MC_GRP[0].Cfg.GrEnable	Variable that indicates when the axes group is enabled
_MC_GRP[0].Kinematics	Level that indicates the kinematics transformation settings
_MC_GRP[0].Kinematics.GrType	Variable that indicates the axis composition
_MC_GRP[0].Kinematics.Axis[0]	Variable that indicates the axis A0 composition axis
:	
_MC_GRP[0].Kinematics.Axis[3]	Variable that indicates the axis A3 composition axis
_MC_GRP[1]	Axes Group Variable
:	

● **Example:**

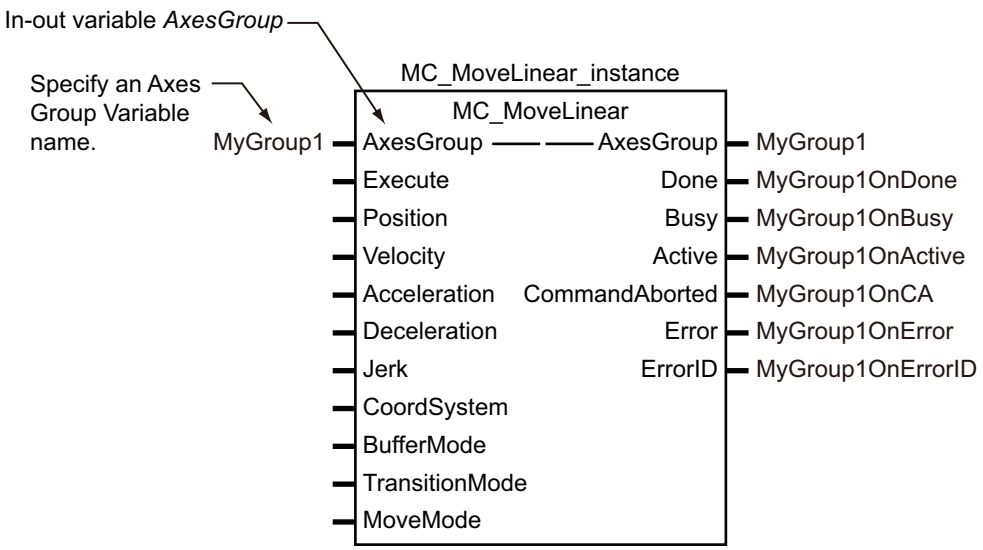
If *MC_Group000* is changed to *MyGroup1*, then either *MyGroup1.Cmd.Vel* or *_MC_GRP[0].Cmd.Vel* can be used as the variable that indicates the command interpolation velocity.

Refer to *Axes Group Variables* on page 6-33 for details on Axes Group Variables.

3-3-4 Specifying an Axes Group in the User Program

In the user program, an axes group variable name is specified for the in-out variable *AxesGroup* in motion control instructions.

In the following example, the Axes Group Variable name for the axes group that was added for the system-defined Axes Group Variable name of *_MC_GRP[0]* has been changed to *MyGroup1* in the Sysmac Studio.



You can also use the *_MC_GRP[0]* system-defined variable in place of *MyGroup1*.

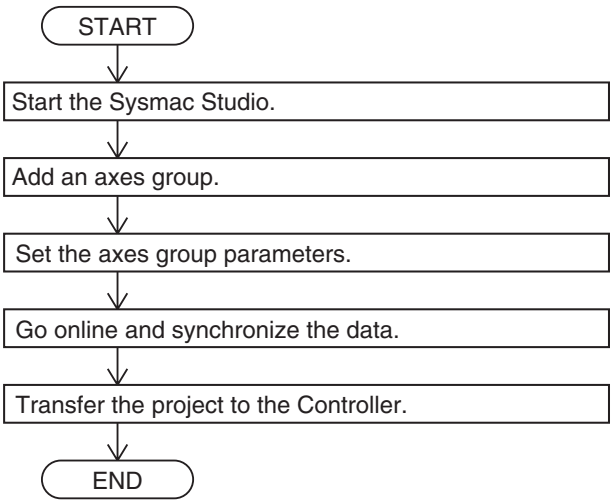
Refer to *6-2 Motion Control Instructions* on page 6-5 for details on motion control instructions.

Refer to the instruction descriptions in the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)* for details on motion control instructions.

3-4 Setting Procedures for Axes Groups

This section gives the procedures to use the Sysmac Studio to set up an axes group.
No configuration is required if you are not going to use any axes group command instructions, such as linear interpolation or circular interpolation.

3-4-1 Setting Procedure for an Axes Group

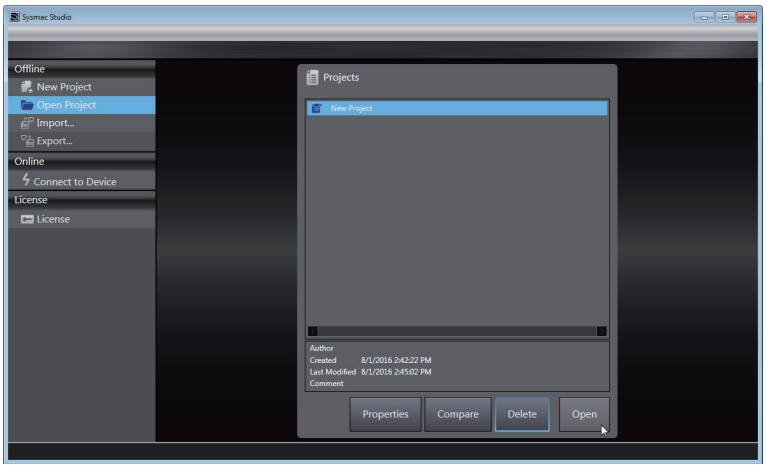


3-4-2 Setting Procedure

This section gives the procedures to use the Sysmac Studio to set up an axes group in a project that already contains the axes.

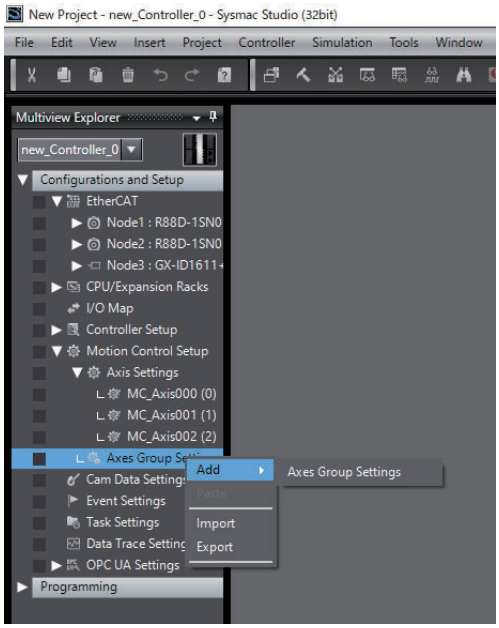
Starting the Sysmac Studio

Start the Sysmac Studio and open the project.

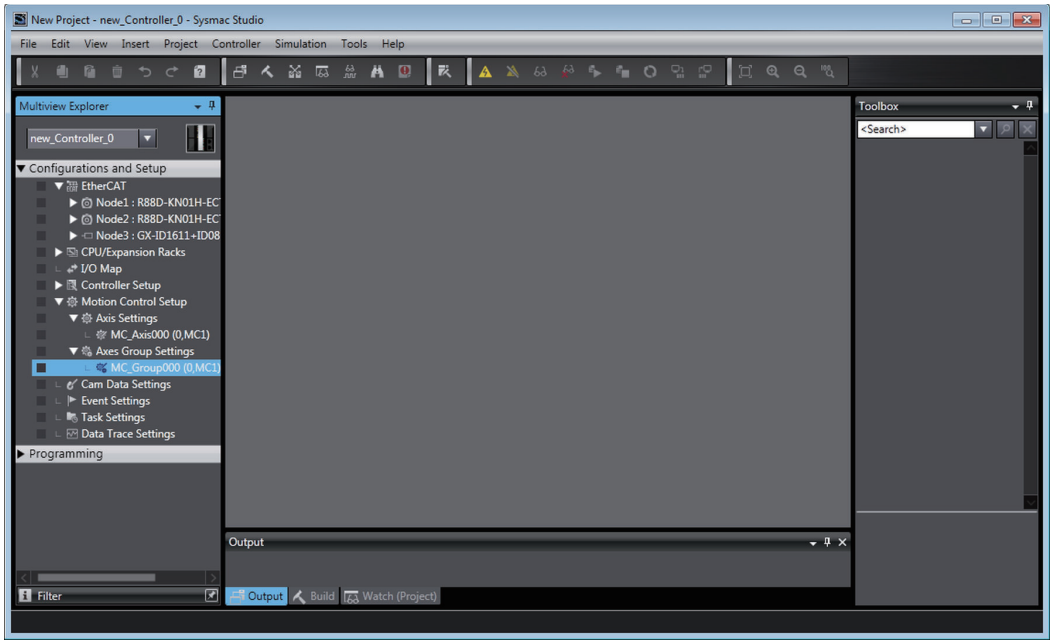


Adding an Axes Group

Right-click **Axes Group Settings** in the Multiview Explorer and select **Axes Group Settings** from the **Add** Menu.



An axes group is added to the Multiview Explorer.
The default name for the new axes group is *MC_Group000*.



● Adding an Axes Group by Import

You can also add **axes group settings** by importing a previously created file of axes group settings.
You can create the file of axes group settings by exporting the **axes group settings**.
Refer to the *Sysmac Studio Version 1 Operation Manual (Cat. No. W504)* for details.

Version Information

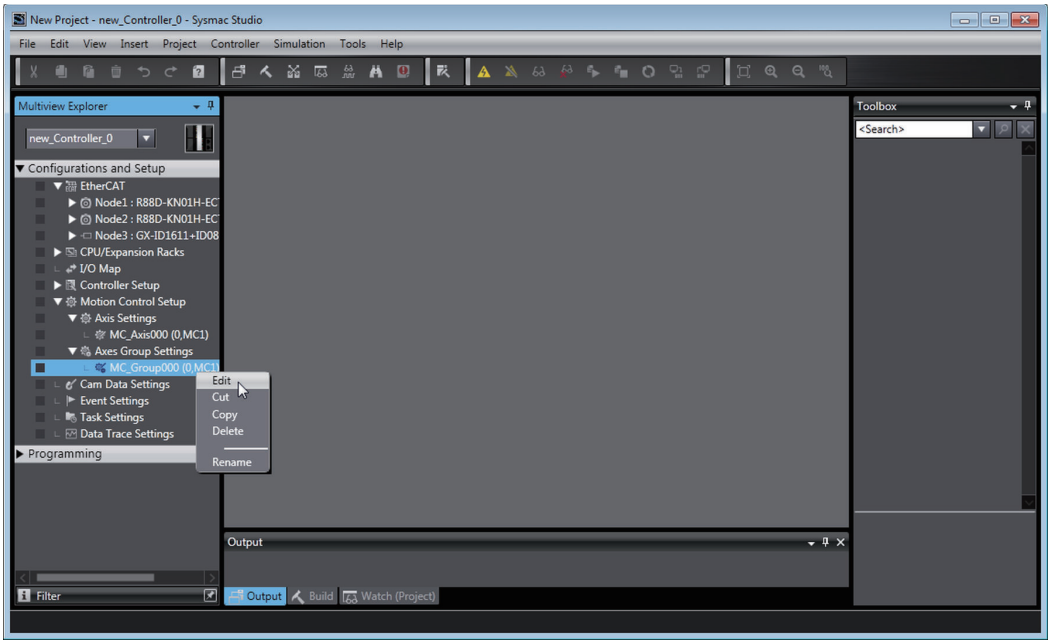
Import and **export** are available in Sysmac Studio Ver.1.57 or higher.

● Copying and Adding an Axes Group

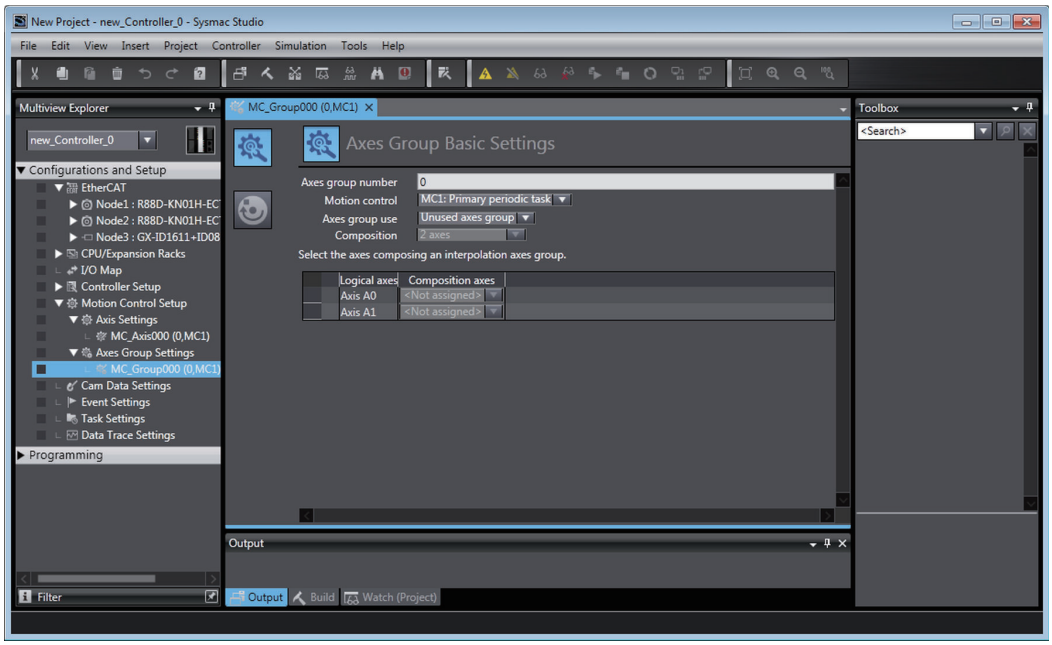
You can also create an axes group by copying and pasting an axes group from a project.

Setting Axes Group Parameters

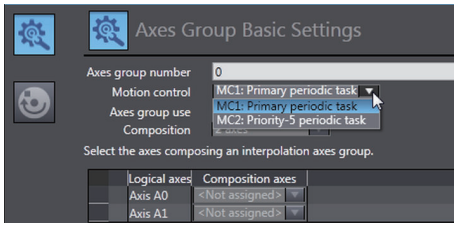
1 Right-click an axes group in the Multiview Explorer and select **Edit** from the menu.



The **Axes Group Basic Settings** are displayed in the Axes Group Parameter Settings Tab Page.



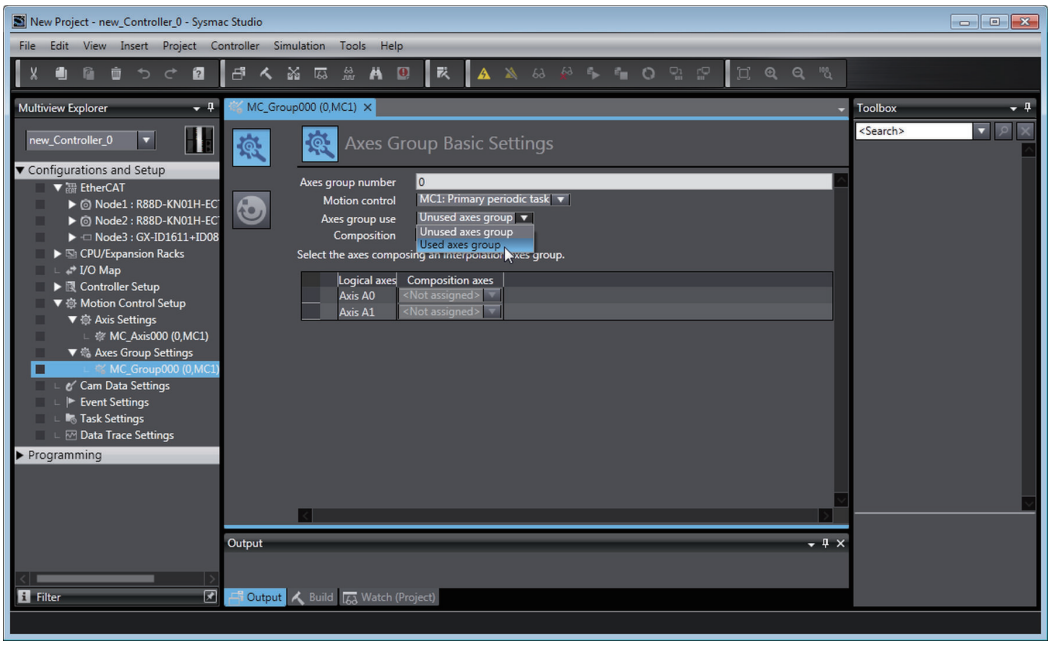
2 Select **Primary periodic task** or **Priority-5 periodic task** from **Motion Control**.



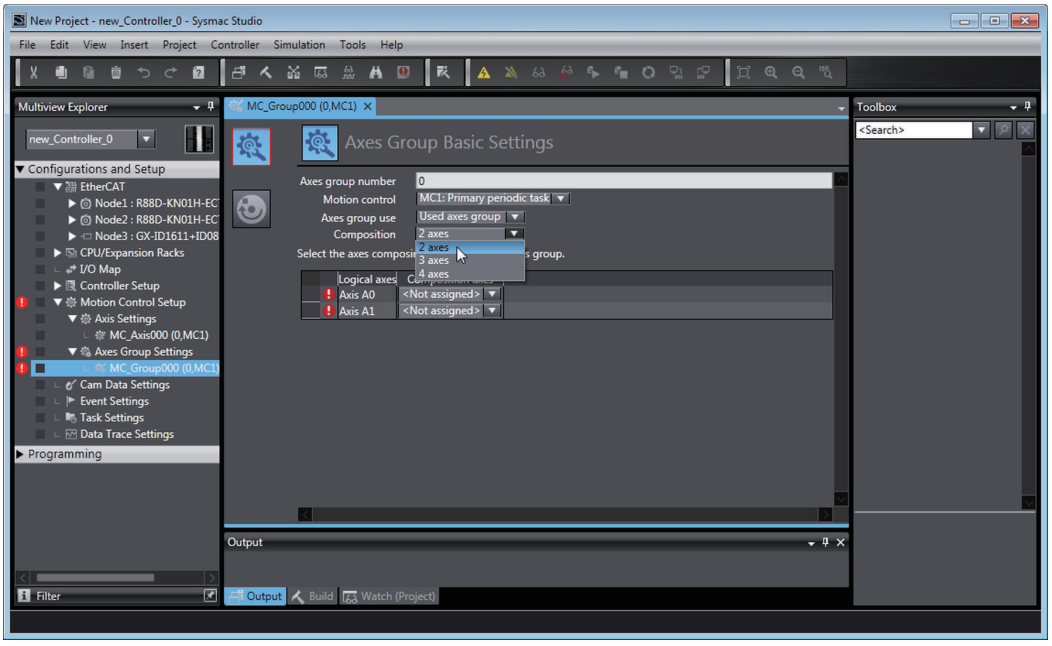
Additional Information

The setting is available for the NX701 CPU Unit.

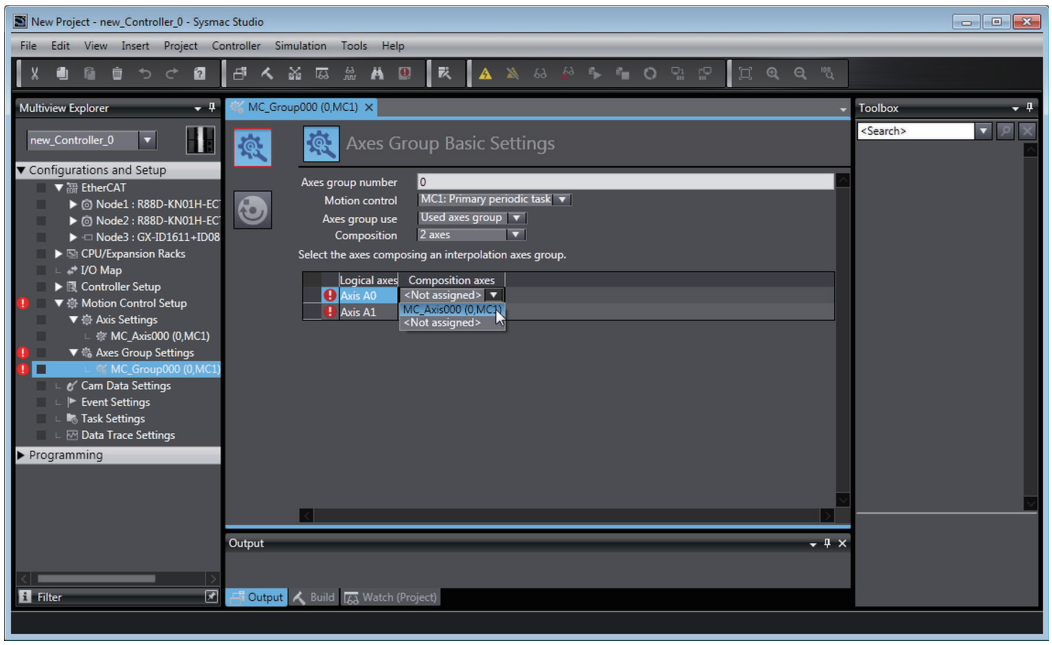
3 Select **Used axes group** in the **Axes group use** Box.



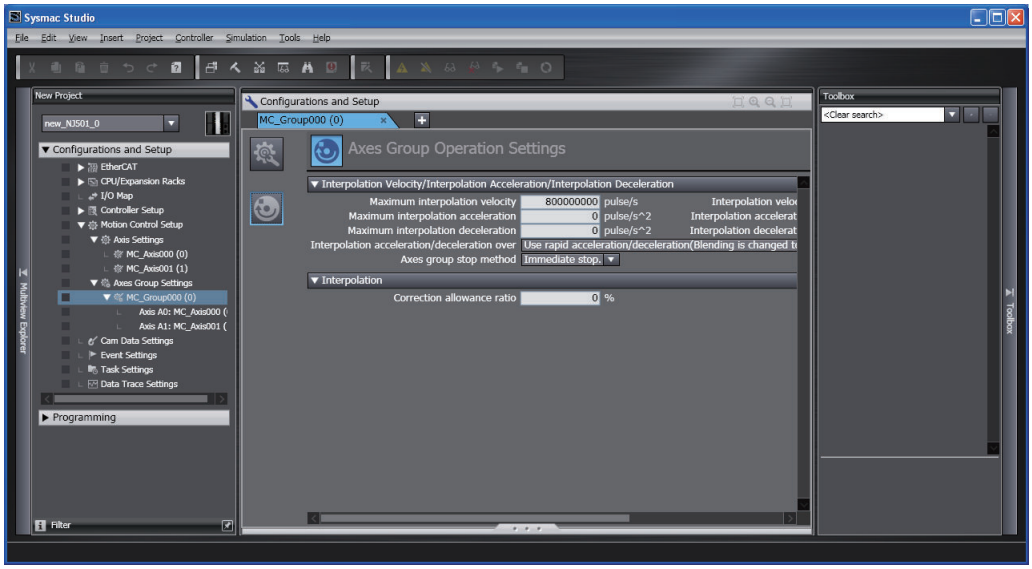
4 Select the composition of the axes group in the **Composition** Box. A 2-axis composition is selected in the following example.



5 Assign the axis to use in the **Logical axes** Box.



6 Click the bottom icon. The **Axes Group Operation Settings** Display is displayed.
Set appropriate values for the settings based on the operating conditions of the device.



Additional Information

Changing Axes Group Variable Names in the User Program

Perform the following two procedures to change Axes Group Variable names that are already used.

- Change the Axes Group Variable name in the variable table in the variable declarations.
- Change the Axes Group Variable name in the user program.

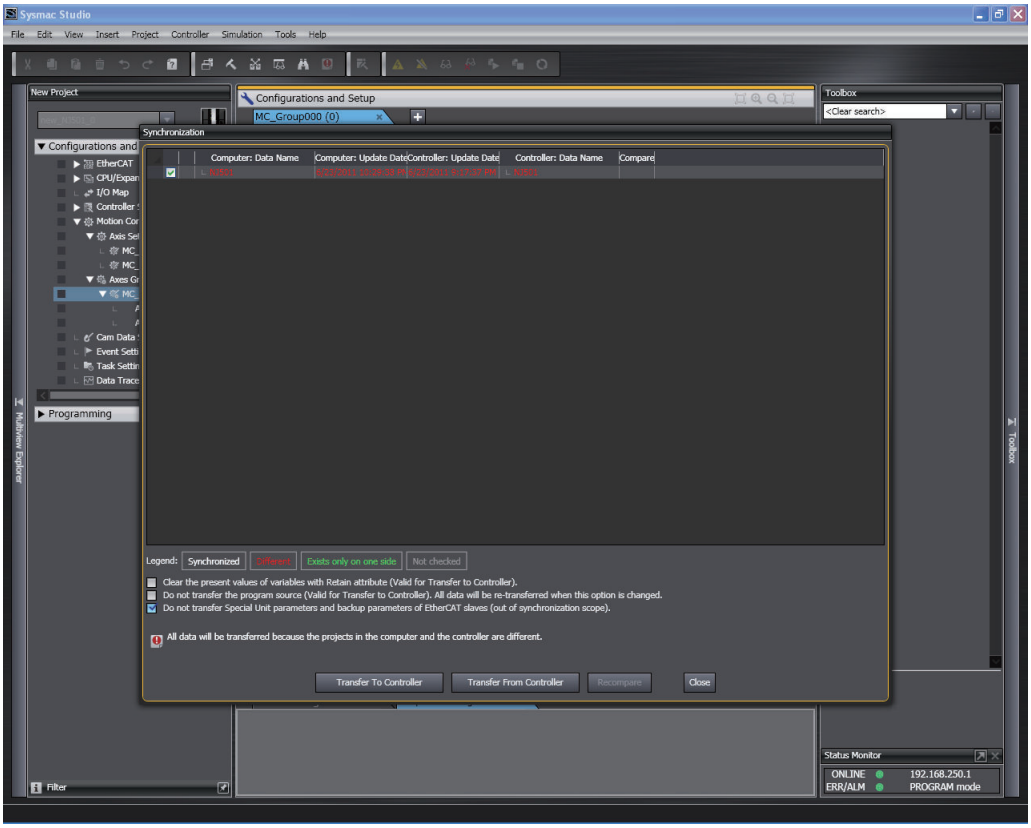
Even if you change the Axes Group Variable names in the variable table, the Axes Group Variable names in the user program do not change.
An error will occur if you use a variable name that is not declared in the variable table, in the user program. Always change the names in both places.

Downloading to the CPU Unit

Use the Synchronization menu command of the Sysmac Studio to download the project to the NJ/NX-series CPU Unit.

1 Select **Online** from the **Controller** Menu. The Sysmac Studio goes online with the Controller.

2 Select **Synchronization** from the **Controller** Menu and then click the **Transfer to Controller** Button.



4

Checking Wiring from the Sysmac Studio

This section describes the MC Test Run function of the Sysmac Studio. You can use the MC Test Run function to monitor sensor signals and check motor wiring without any programming.

4-1 MC Test Run Function

This section describes how to use the MC Test Run function to check wiring and basic settings.
You can use the MC Test Run function in the Sysmac Studio to check wiring without any programming.

Precautions for Correct Use

You can use the MC Test Run function for OMRON 1S-series Servo Drives, G5-series Servo Drives, or NX-series Pulse Output Units. Do not use it with servo drives from any other manufacturer.

Additional Information

- Refer to *A-1 Connecting the 1S-series Servo Drive* on page A-2 for the procedure to connect to the OMRON 1S-series Servo Drive.
- Refer to *A-2 Connecting the G5-series Servo Drive* on page A-11 for the procedure to connect to the OMRON G5-series Servo Drive.

4-1-1 List of MC Test Run Functions

The MC Test Run function supports the following functions.
You can check the wiring of sensors for monitoring and the operations of Servo ON, jogging, and homing for axis operation to check the wiring between the CPU Unit and Servo Drives.

Category	Function	Description	Setting/monitor item
Axis operation	Servo ON/OFF	The Servo is turned ON and OFF.	---
	Jogging	Jogging is performed in the positive or negative direction.	Target Velocity Acceleration/Deceleration
	Homing	Homing is performed using the homing parameter settings.	Homing Parameters
	Absolute positioning	Absolute positioning is performed. *1	Target Position Target Velocity Acceleration/Deceleration Jerk
	Relative positioning	Relative positioning is performed.	Travel Distance Target Velocity Acceleration/Deceleration Jerk
	Deceleration stop	A deceleration stop is performed during the MC Test Run.	---
	Resetting errors	The errors in the MC Function Module are reset.	---
Monitoring	Servo Drive status	The status of the Servo Drive is monitored.	Servo ON/OFF Servo Ready Main Power

Category	Function	Description	Setting/monitor item
	Input signals	The status of the input signals are monitored.	Positive Limit Input/Negative Limit Input Immediate Stop Input Home Proximity Input Home Input External Latch Inputs 1 to 2
	Actual position monitor	The actual position is monitored.	Command and Actual Current Positions
	Actual velocity monitor	The actual velocity is monitored.	Command and Actual Current Velocities
	Axis status	The status of the axes is monitored.	Axis Ready-to-execute Standstill Discrete Motion Continuous Motion Homing Deceleration Stopping Home Defined In Home Position
	Error list	The errors in the MC Function Module are monitored.	MC Common Errors Axis Errors Axes Group Errors

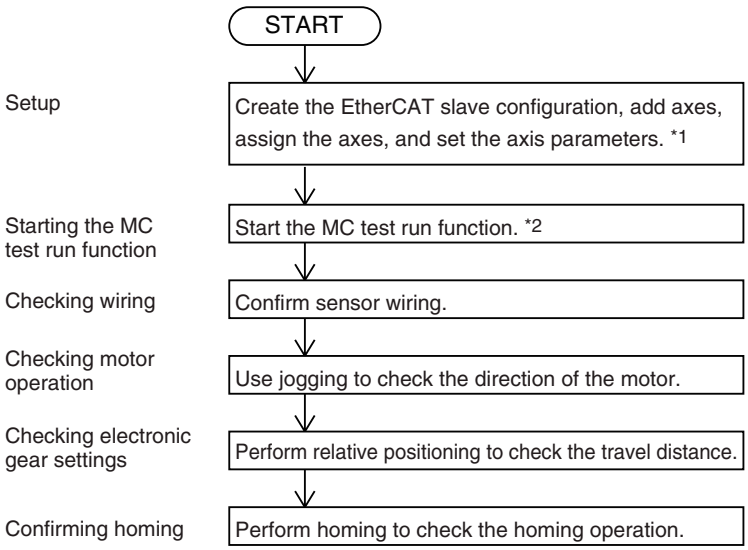
*1. When the **Count Mode** of the axis is set to **Rotary Mode**, positioning is performed toward the target position in the positive direction.
For details, refer to the MC_MoveAbsolute (Absolute Positioning) instruction in the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)*.

4-1-2 Application Procedure

Before you perform an MC Test Run, check the following two items.

- Are the Sysmac Studio and Controller connected and are they online?
- Is the MC Test Run Mode currently not in use from any other copy of the Sysmac Studio?

After you have confirmed these two items, perform the following operations as instructed.



*1. Refer to *Section 3 Configuring Axes and Axes Groups* on page 3-1.
*2. Refer to *4-1-4 Starting the MC Test Run Function* on page 4-5.

Precautions for Correct Use

- When one of the following operations is performed for a command from the Sysmac Studio, the Servomotor will operate at the set velocity: Servo ON, jogging, relative positioning, absolute positioning, or homing.
  Always confirm that it is safe for the Servomotor to operate before executing any of these operations.
- When you operate the Controller from the Sysmac Studio, always install external emergency circuits so that the Servomotor can be stopped safely whenever necessary. The Sysmac Studio may not be able to send commands under some circumstances, e.g., if an error occurs in the computer.
- Set the EtherCAT communications and establish communications before you attempt to perform operation from the Sysmac Studio.
- Refer to the *NX-series Position Interface Units User's Manual (Cat. No. W524)* for the procedures for the NX-series Position Interface Units.

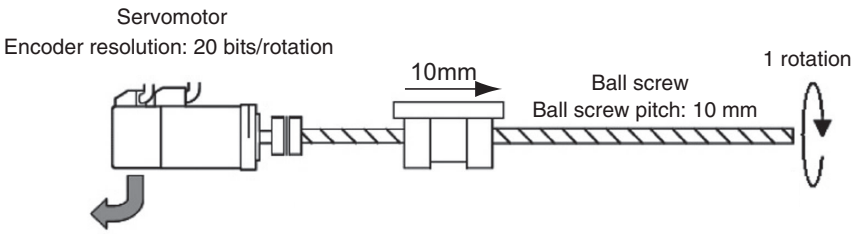
Additional Information

You can perform the following operations to end the MC Test Run function at any time.
- Select **MC Test Run** − **Stop** from the **Controller** Menu of the Sysmac Studio.
- Right-click the axis in the Multiview Explorer of the Sysmac Studio and select **Stop MC Test Run** from the menu.
- Close the MC Test Run Tab Page on the Sysmac Studio.
- Exit the Sysmac Studio.

Refer to the *Sysmac Studio Version 1 Operation Manual (Cat. No. W504)* for specific procedures.

4-1-3 Axis Parameter Setting Example

Set the following axis parameters before you execute the MC Test Run Mode in the Sysmac Studio.
The following setting example is for a one-axis device.



Encoder Output Pulse Count per Motor Rotation
20 bits = 1,048,576

Parameter name	Setting
Axis Variable Name	Axis1*1
Axis Number	1*2
Axis Use	Used axis
Axis Type	Servo axis
Input Device/Output Device	1*3
Unit of Display	μm
Command Pulse Count Per Motor Rotation	1,048,576*4

Parameter name	Setting
Work Travel Distance Per Motor Rotation	10,000*4
Maximum Velocity	500,000*5
Maximum Jog Velocity	50,000*6
Maximum Acceleration	5,000,000*7
Maximum Deceleration	5,000,000*7
Software Limits	Immediate stop for command position
Positive Software Limit	500,000*8
Negative Software Limit	0*8
Count Mode	Linear Mode

*1. If there is more than one axis, a different variable name is set for each axis.
*2. If there is more than one axis, a different value is set for each axis.
*3. Set the same node address as for the Servo Drive. If there is more than one axis, a different value is set for each axis.
*4. The position command unit will be 1 μm.
*5. The maximum velocity will be 3,000 r/min = 30 m/min = 0.5 m/s = 500,000 μm/s.
*6. The maximum jog velocity will be 10% of the maximum velocity, i.e., 0.05 m/s = 50,000 μm/s.
*7. The maximum acceleration and the maximum deceleration will be 5 m/s^2. The acceleration time to the maximum velocity (3,000 r/min) will be 0.1 s.
*8. Set a positioning that is within the movable range of the device. The positive software limit is set to 50 cm = 500,000 μm.

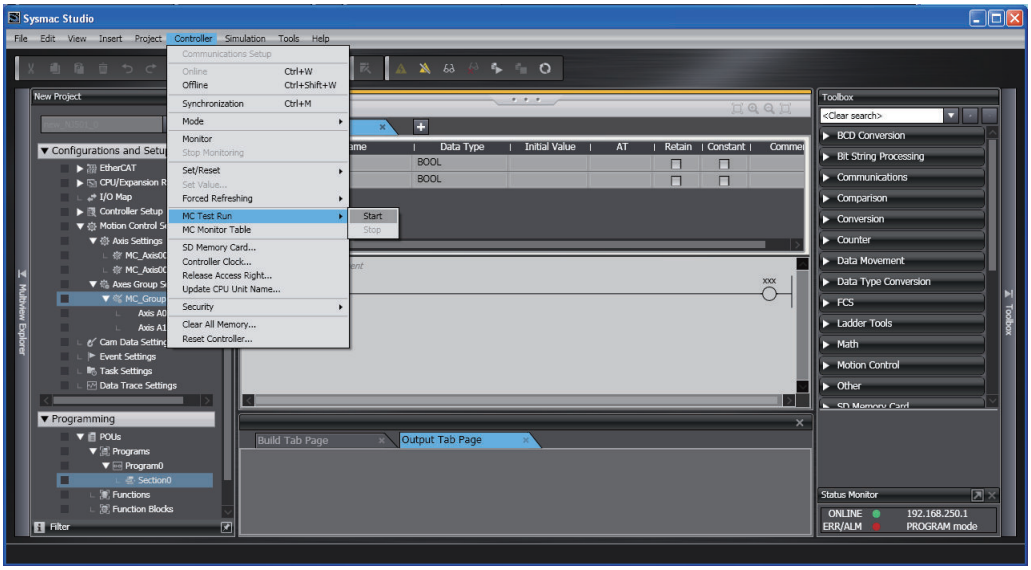
4-1-4 Starting the MC Test Run Function

The MC Test Run Mode is started from the Sysmac Studio.

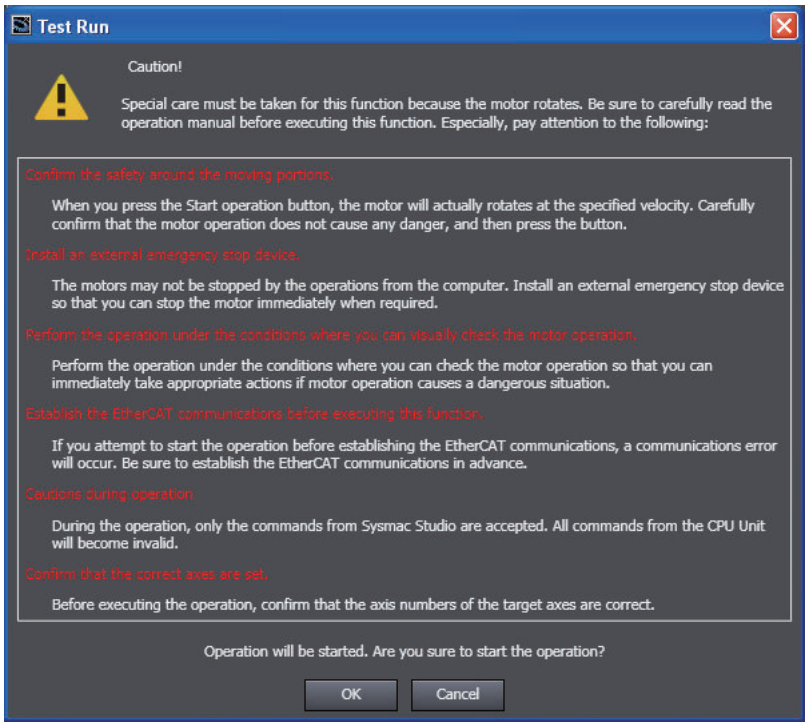
1 Start the Sysmac Studio and open a project in which the axis settings are completed.

2 Select **Online** from the **Controller** Menu. The Sysmac Studio goes online with the Controller.

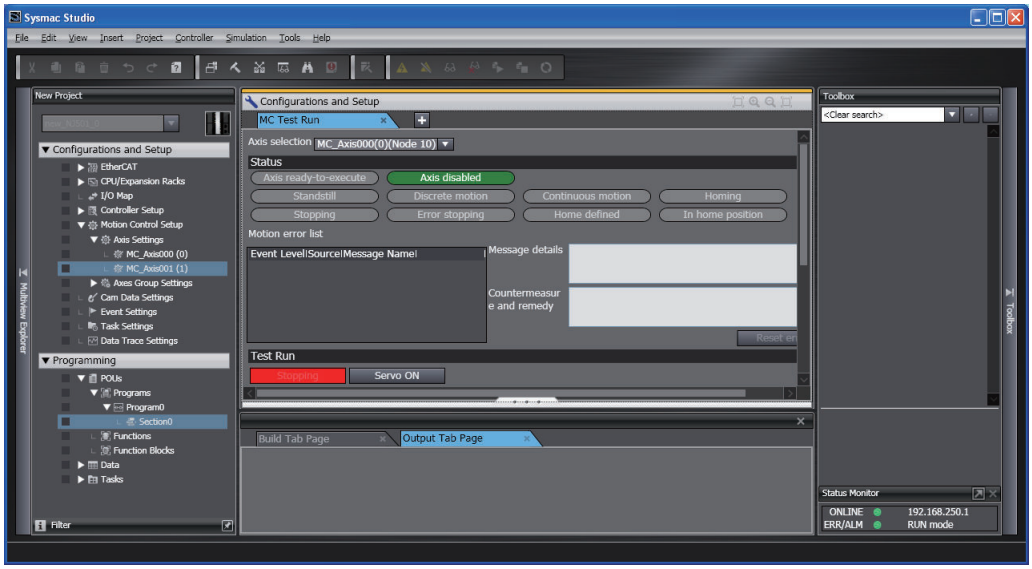
3 Select **MC Test Run** − **Start** from the **Controller** Menu.



When the following caution dialog box appears, read the message carefully. After you confirm safety, click the **OK** Button.



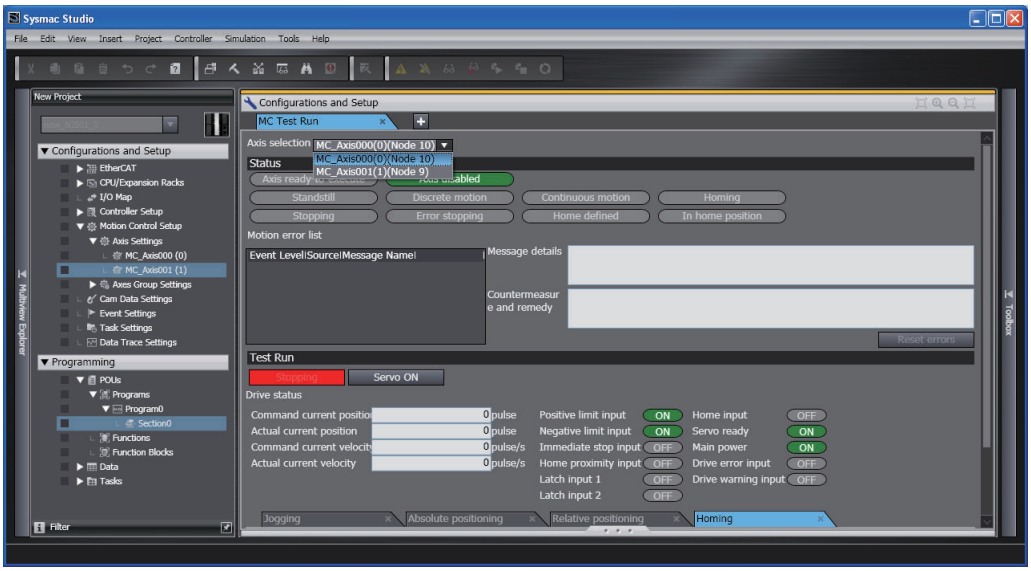
The MC Test Run Tab Page is displayed.



4-2 Monitoring Sensor Signals

You can use the input signal display to check sensor signal wiring.

1 Select the axis to check on the MC Test Run Tab Page.



2 Check to see if the signals turn ON and OFF properly on the monitor screen by turning ON and OFF the sensor connected to each input signal.

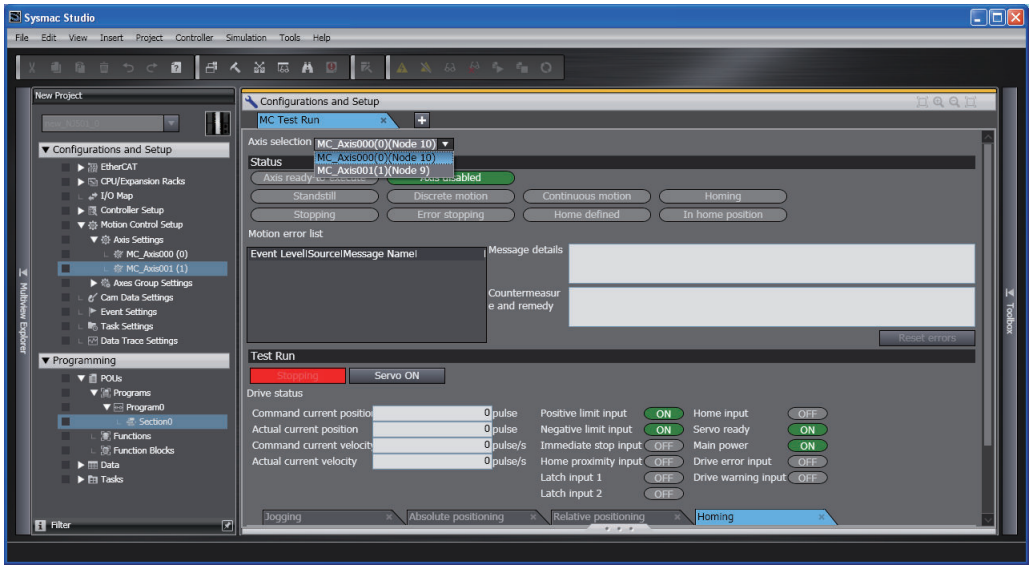
4-3 Checking Motor Operation

Use the functions of the MC Test Run to check motor operation.

4-3-1 Turning ON the Servo

You can use the **Servo ON** Button to turn the Servo ON and OFF.

1 Select the axis for which to turn ON the Servo.



2 Click the **Servo ON** Button to turn ON the Servo.

3 Click the **Servo OFF** Button to turn OFF the Servo.

Precautions for Correct Use

- When one of the following operations is performed for a command from the Sysmac Studio, the Servomotor will operate at the set velocity: Servo ON, jogging, relative positioning, absolute positioning, or homing.
  Always confirm that it is safe for the Servomotor to operate before executing any of these operations.
- When you operate the Controller from the Sysmac Studio, always install external emergency circuits so that the Servomotor can be stopped safely whenever necessary. The Sysmac Studio may not be able to send commands under some circumstances, e.g., if an error occurs in the computer.
- Set the EtherCAT communications and establish communications before you attempt to perform operation from the Sysmac Studio.
- If you use an NX-series Pulse Output Unit, you must provide a separate means to turn the power supply to the motor drive ON and OFF. Refer to the *NX-series Position Interface Units User's Manual (Cat. No. W524)* for details.

4-3-2 Jogging

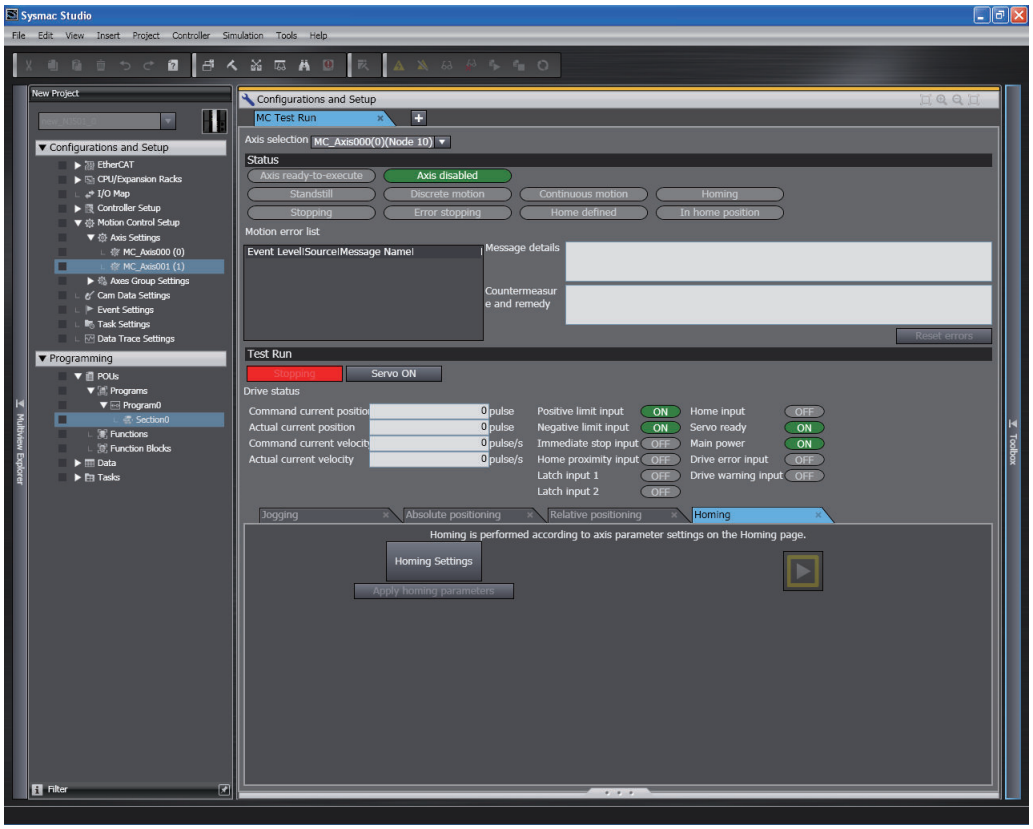
1. Select the axis to jog on the **Jogging** Tab Page of the MC Test Run Tab Page.
2. Click the **Servo ON** Button to turn ON the Servo.
3. Enter the target velocity, acceleration rate, and deceleration rate, and then press the **Apply** Button.
4. Click the  or  Button.

   The motor will operate in either the positive or negative direction while one of these buttons is clicked.
   Check to see if the motor operates in the set direction.

4-3-3 Homing

To perform homing with the MC Test Run function, you must make the **Homing Settings** for axis parameters.
For details on homing, refer to *Section 8 Homing* on page 8-1 and *5-2-9 Homing Settings* on page 5-29.

1. Set the homing parameters in the **Homing Settings** on the Axis Parameter Settings Tab Page.
2. Click the **Homing** Tab on the MC Test Run Tab Page.
   The following dialog box is displayed.



3 Select the axis to home.

4 Click the **Servo ON** Button to turn ON the Servo.

5 Click the **Apply homing parameters** Button.

6 Click the  Button.

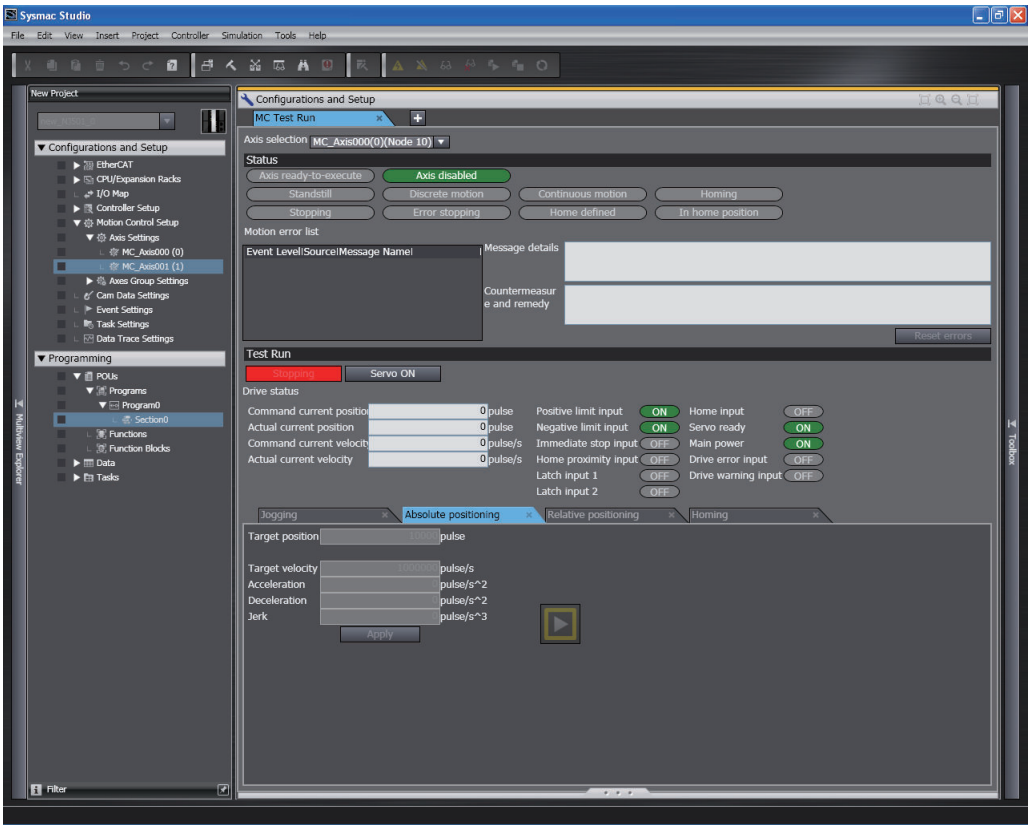
Check to see if the homing operation agrees with the settings.

Additional Information

- When you click the **Homing Settings** Button, the **Homing Settings** are displayed on the **Axis Parameter Settings** Tab Page. Set the **homing parameters**.
- If the **homing parameters** were set in advance, click the **Apply homing parameters** Button to apply those settings.

4-3-4 Absolute Positioning

1 Click the **Absolute positioning** Tab on the MC Test Run Tab Page.
The following dialog box will appear.



2 Select the axis to perform absolute positioning.

3 Click the **Servo ON** Button to turn ON the Servo.

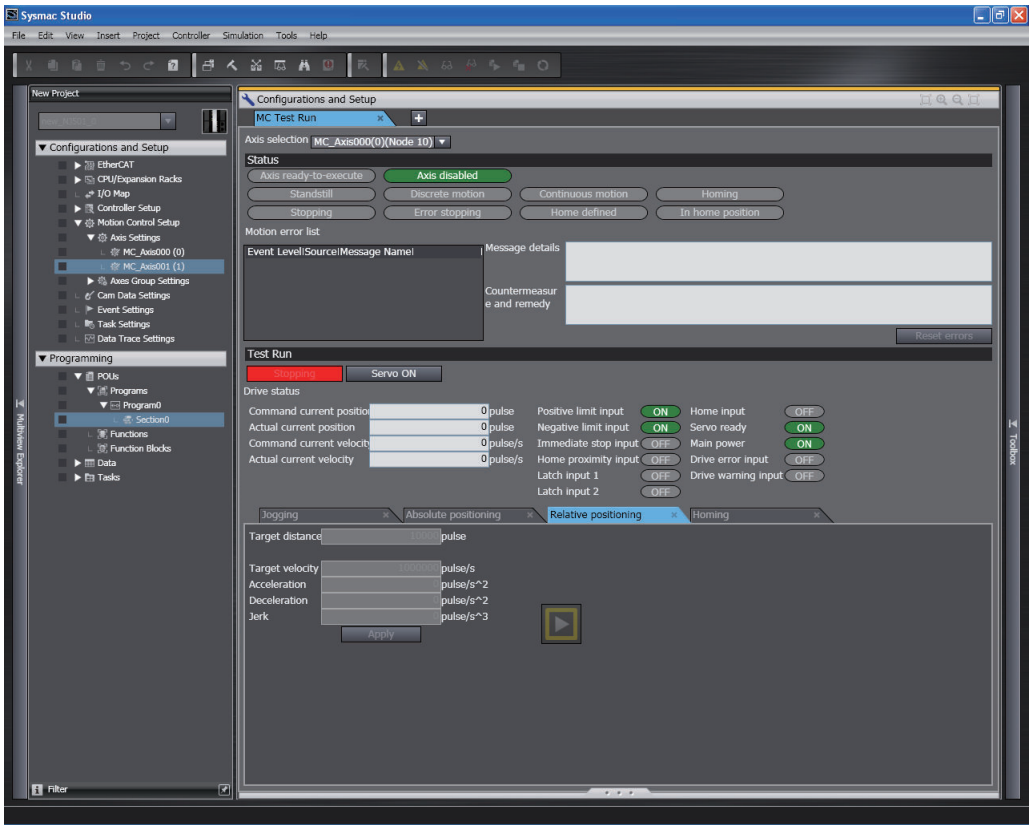
4 Enter the target position, target velocity, acceleration rate, deceleration rate, and jerk, and then click the **Apply** Button.

5 Click the Button. Absolute positioning will start.

Check to see if positioning agrees with the settings.

4-3-5 Relative Positioning

1 Click the **Relative positioning** Tab on the MC Test Run Tab Page.
The following dialog box will appear.



2 Select the axis to perform relative positioning.

3 Click the **Servo ON** Button to turn ON the Servo.

4 Enter the target travel distance, target velocity, acceleration rate, deceleration rate, and jerk, and then click the **Apply** Button.

5 Click the Button. Relative positioning will start.

Check to see if the travel distance agrees with the settings.

5

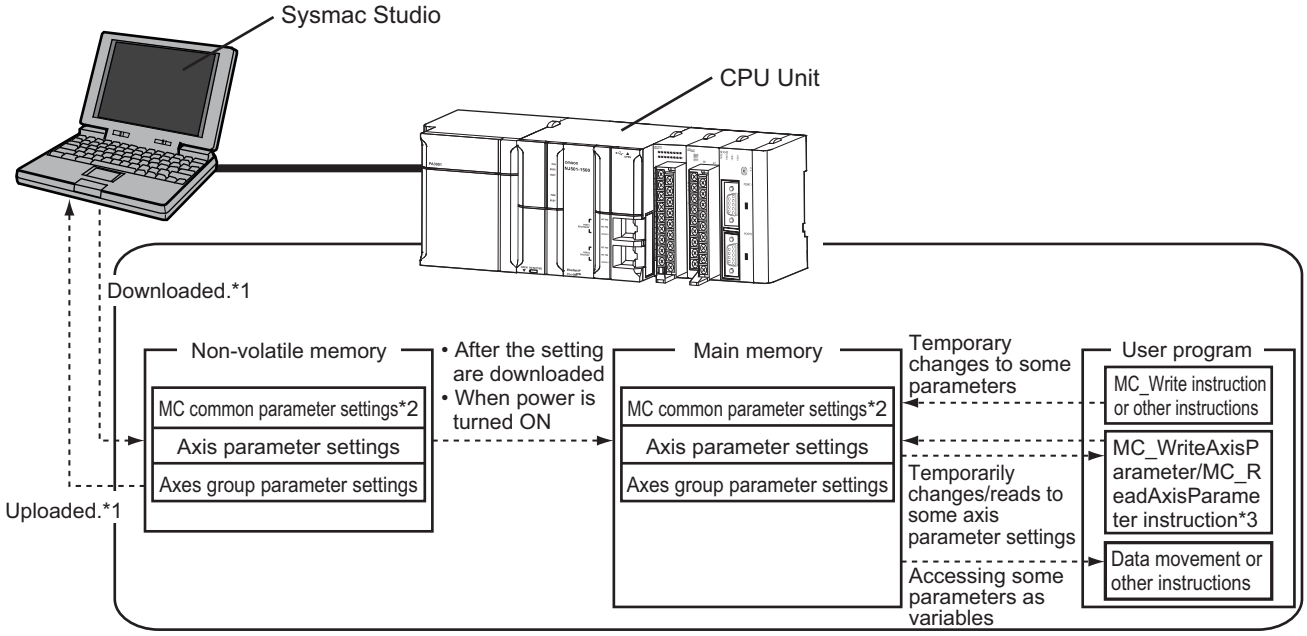
Motion Control Parameters

This section explains about axis parameters and axes group parameters used for motion control.

5-1 Introduction

You can use motion control instructions to perform single-axis operations and multi-axes operations on axes groups with the NJ/NX-series CPU Unit's MC Function Module.
Axis and axes group parameters are used to set these operations.
Axis parameters must be set, but axes group parameters are not required if you do not use multi-axes operations for axes groups.

These parameters are called motion control parameter settings (MC parameter settings).



*1. Use the Synchronization menu command of the Sysmac Studio to upload and download the project.
*2. There are no MC Common Parameter Settings for the current version of the MC Function Module.
*3. A CPU Unit with unit version 1.08 or later and Sysmac Studio version 1.09 or higher are required to use the MC_WriteAxisParameter (Write Axis Parameters) and MC_ReadAxisParameter (Read Axis Parameters) instructions.

Data Flow for Setting MC Parameters

- Download your MC Parameter Settings to the CPU Unit using the Sysmac Studio to save those settings in the CPU Unit's non-volatile memory.
  When you upload the MC Parameter Settings to the Sysmac Studio, the MC Parameter Settings that were saved in the non-volatile memory are uploaded.
- The settings that were saved in the non-volatile memory are applied to the main memory after you download them or when the power is turned ON.
- If there are no problems with the saved settings, the MC Function Module executes control based on the settings in the main memory.
- The settings of some of the parameters can be accessed as system-defined variables for motion control.
- You can upload and download MC parameter settings regardless of the CPU Unit mode or the status of the MC Function Module.

- When you start the download process, all axes in motion will stop immediately.

Version Information

If a CPU Unit with unit version 1.13 or later and Sysmac Studio version 1.17 or higher are combined, commands to the I/O devices can continuously be sent even when the download process is in progress.
For the CPU Unit with unit version 1.12 or earlier, sending commands to the I/O devices is stopped when the download process is executed.
Refer to the *NJ/NX-series CPU Unit Software User's Manual (Cat. No. W501)* for how to set to stop or continue sending commands to the I/O devices when the download process starts.

Stopping sending commands to I/O devices

Servo is turned OFF during the download, and the axis status will be *Disable* (Axis Disabled).

Continuing sending commands to I/O devices

During download, the servo ON state set by the MC_Power instruction immediately before the download and the output torque limits set by the MC_SetTorqueLimit and MC_SetTorqueLimit2 instructions are maintained.
The Servo ON state and torque limits are maintained even if the MC_Power, MC_SetTorqueLimit, and MC_SetTorqueLimit2 instructions are deleted from the user program updated by the download.

Overwriting MC Parameters with Programming Instructions

- You can use motion control instructions like the MC_Write (Write MC Setting), MC_ChangeAxesInGroup (Change Axes in Group) or MC_WriteAxisParameter (Write Axis Parameters) instruction to change the settings of some of the MC parameters in the main memory while the user program is running.
- If the specified set value is outside the valid range, the *Error* output variable from the instruction changes to TRUE and the MC parameter setting is not changed.
- Changes to MC parameter settings become valid in either of the following two situations.
  a) The axis or axes group is stopped and you execute an instruction for an axis command or axes group command.
  b) You set the Buffer Mode Selection for the instruction to Aborting and execute more than one instruction.
- For details on MC_Write (Write MC Setting), MC_ChangeAxesInGroup (Change Axes in Group), MC_WriteAxisParameter (Write Axis Parameters) and other instructions, refer to the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)*.

Precautions for Correct Use

- Changes to the MC Parameter Settings that are made with the MC_Write (Write MC Setting) instruction are saved in the main memory in the CPU Unit. They are not saved in the built-in non-volatile memory in the CPU Unit.
  Therefore, if you cycle the power supply or download the settings from the Sysmac Studio, the parameter settings in the non-volatile memory are restored. Also, you cannot upload the data in the main memory from the Sysmac Studio.
  If you need to save settings to the non-volatile memory, use the Sysmac Studio to change the parameter settings and then download those settings to the CPU Unit.
- To maintain the MC_Power (Power Servo) instruction, the MC_SetTorqueLimit (Set Torque Limit) and MC_SetTorqueLimit2 (Set Torque Limit 2) instructions after downloading, set the Retain attribute of the input bits for the instructions to Retain.
- You can use the following instructions to change the settings of the MC parameters.
  MC_Write (Write MC Setting) instruction
  MC_ChangeAxesInGroup (Change Axes in Group) instruction
  MC_ChangeAxisUse (Change Axis Use) instruction
  MC_WriteAxisParameter (Write Axis Parameters) instruction
- Refer to the *NX-series Position Interface Units User's Manual (Cat. No. W524)* for information on using the NX-series Position Interface Units.

5-2 Axis Parameters

The axis parameters set the maximum velocity, jerk, homing, and other items for the axes controlled by the MC Function Module.
The number of axis parameters provided is the same as the maximum number of controlled axes for each model. The maximum number of controlled axes varies depending on the model.
Refer to *1-4-2 Performance Specifications* on page 1-7 for details.
The same parameter settings are provided for each axis. This section describes only the parameters for axis 1.

5-2-1 Axis Parameters

Use the Sysmac Studio to set the axis parameters for each axis.

Classification	Parameter name	Temporary changes[*1]		Reading variables[*2]	Page
		Support	Applicable instruction		
Axis Basic Settings	Axis Number			OK	page 5-8
	Motion Control[*3]			OK[*4]	
	Axis Use	OK[*5]	MC_ChangeAxisUse	OK	
	Axis Type			OK	
	Control Function[*6]				
	Input Device/Output Device			OK	
Unit Conversion Settings	Unit of Display	OK[*7]	MC_WriteAxisParameter[*8]	OK	page 5-13
	Command Pulse Count Per Motor Rotation			OK	
	Work Travel Distance Per Motor Rotation			OK	
	Reducer Use[*9]				
	Work Travel Distance Per Rotation[*9]				
	Work Gear Ratio[*9]				
	Motor Gear Ratio[*9]				

Classification	Parameter name	Temporary changes[*1]		Reading variables[*2]	Page
		Support	Applicable instruction		
Operation Settings	Maximum Velocity	OK[*7]	MC_WriteAxisParameter[*8]		page 5-21
	Start Velocity[*10]				
	Maximum Jog Velocity				
	Maximum Acceleration				
	Maximum Deceleration				
	Acceleration/Deceleration Over				
	Operation Selection at Reversing				
	Velocity Warning Value	OK	MC_Write MC_WriteAxisParameter[*8]		
	Acceleration Warning Value				
	Deceleration Warning Value				
	Positive Torque Warning Value[*11]				
	Negative Torque Warning Value[*11]				
	In-position Range	OK [*12]			
	In-position Check Time	OK			
	Actual Velocity Filter Time Constant	OK[*7]	MC_WriteAxisParameter[*8]		
	Zero Position Range				
Other Operation Settings	Immediate Stop Input Stop Method	OK[*7]	MC_WriteAxisParameter[*8]		page 5-24
	Limit Input Stop Method				
	Drive Error Reset Monitoring Time				
	Maximum Positive Torque Limit				
	Maximum Negative Torque Limit				
	Immediate Stop Input Logic Inversion[*10]				
	Positive Limit Input Logic Inversion[*10]				
	Negative Limit Input Logic Inversion[*10]				
	Home Proximity Input Logic Inversion[*10]				
Limit Settings	Software Limits	OK	MC_Write MC_WriteAxisParameter[*8]		page 5-25
	Positive Software Limit				
	Negative Software Limit				
	Following Error Over Value				
	Following Error Warning Value				
Position Count Settings	Count Mode	OK[*7]	MC_WriteAxisParameter[*8]		page 5-26
	Modulo Maximum Position Setting Value			OK[*4]	
	Modulo Minimum Position Setting Value			OK[*4]	
	Encoder Type				

Classification	Parameter name	Temporary changes*1		Reading variables*2	Page
		Support	Applicable instruction		
Servo Drive Settings	Modulo Maximum Position Setting Value				page 5-28
	Modulo Minimum Position Setting Value				
	PDS State Control Method*4				
Homing Settings	Homing Method	OK*7	MC_WriteAxisParameter*8		page 5-29
	Home Input Signal				
	Homing Start Direction				
	Home Input Detection Direction				
	Operation Selection at Positive Limit Input				
	Operation Selection at Negative Limit Input				
	Homing Velocity				
	Homing Approach Velocity				
	Homing Acceleration				
	Homing Deceleration				
	Homing Jerk				
	Home Input Mask Distance				
	Home Offset				
	Homing Holding Time				
	Homing Compensation Value				
	Homing Compensation Velocity				

*1. This column indicates if you can use instructions to temporarily change the settings.
*2. Indicates whether you can access the parameter with a system-defined variable for motion control in the user program.
*3. Set this parameter when using the NX701 CPU Unit.
*4. A CPU Unit with unit version 1.10 or later and Sysmac Studio version 1.12 or higher are required to use this parameter.
*5. A CPU Unit with unit version 1.04 or later and Sysmac Studio version 1.05 or higher are required for temporary changes.
*6. Set this parameter when using the NX102 CPU Unit and NX1P2 CPU Unit.
*7. A CPU Unit with unit version 1.08 or later and Sysmac Studio version 1.09 or higher are required for temporary changes.
*8. A CPU Unit with unit version 1.08 or later and Sysmac Studio version 1.09 or higher are required to use the MC_WriteAxisParameter instruction.
The parameters that can be temporarily changed with the MC_WriteAxisParameter instruction can be read with the MC_ReadAxisParameter instruction.
*9. A CPU Unit with unit version 1.11 or later and Sysmac Studio version 1.15 or higher are required to use this parameter.
*10. A CPU Unit with unit version 1.05 or later and Sysmac Studio version 1.06 or higher are required to use this parameter.
*11. This parameter is enabled only for torque control.
*12. A CPU Unit with unit version 1.01 or later and Sysmac Studio version 1.02 or higher are required for temporary changes using the MC_Write instruction.
A CPU Unit with unit version 1.08 or later and Sysmac Studio version 1.09 or higher are required for temporary changes using the MC_WriteAxisParameter instruction.

Refer to *3-2 Axis Setting Procedure* on page 3-10 for details on how to set axis parameters.
Refer to *6-6 System-defined Variables for Motion Control* on page 6-21 for information on system-defined variables for motion control.

For details on instructions including the MC_Write (Write MC Setting) instruction, refer to the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)*.

5-2-2 Axis Basic Settings

The Axis Basic Settings are used to set whether to use the axis. If you use the axis, set the axis type and the node address of the EtherCAT slave device.

Parameter name	Function	Setting range	Default
Axis Number	Set the logical number of the axis. This number indicates the axis number of the corresponding system-defined variable, *_MC_AX[0-255]*, *_MC1_AX[0-255]*, and *_MC2_AX[0-255]*.	---	---
Motion Control*1	Assign to either of the primary periodic task or priority-5 periodic task. 1: Primary periodic task 2 Priority-5 periodic task	1 to 2	1
Axis Use	Set whether to enable or disable the axis. *2 0: Undefined axis 1: Unused axis*3 2: Used axis	0 to 2	0
Axis Type	Set the axis type. I/O wiring is not required for virtual axes. 0: Servo axis 1: Encoder axis 2: Virtual servo axis 3: Virtual encoder axis	0 to 3	2
Control Function*4	Select the function of axis to control. *5 0: All 1: Single-axis position control only	0, 1	0
Input Device/Output Device	Specify the node address of the EtherCAT slave device that is assigned to the axis. *6 The Node Address parameter cannot be selected if the Axis Type parameter is set to use a **virtual axis**.	0 to 65535	---

*1. Set this parameter when using the NX701 CPU Unit.

*2. *Busy* (Executing) changes to TRUE if you execute a motion control instruction for an undefined or unused axis.
Busy changes to FALSE when *Execute* or *Enable* changes to FALSE.
You can set axes as **unused axes** to enable using the same user program for different axis configurations without the need to delete programming for axes that are not used.

*3. With a CPU Unit with unit version 1.04 or later and Sysmac Studio version 1.05 or higher, **Unused axis (changeable to used axis)** and **Unused axis (unchangeable to used axis)** are displayed by the Sysmac Studio.
If you set **Unused axis (changeable to used axis)**, you can set the axis parameters and use the MC_ChangeAxisUse (Change Axis Use) instruction to temporarily change the setting of the **Axis Use** axis parameter.
Refer to *9-8-10 Changing Axis Use* on page 9-89 for details.

*4. Set this parameter when using the NX102 CPU Unit and NX1P2 CPU Unit.

*5. To use the axis as a motion control axis, select **All**. To use the axis as a single-axis position control axis, select **Single-axis position control only**.

*6. For an NX-series Position Interface Unit, select the node address of the EtherCAT Coupler Unit and the NX Unit number of the Position Interface Unit.

Precautions for Correct Use

Using Absolute Encoders

When absolute encoders are used, the **absolute encoder home offset** for each axis is saved to the CPU Unit along with the axis number.

The saved offset is lost if the axis number is changed. If you change the axis number, set the Homing Settings again.

Axis Numbers

You can set the number for axis numbers up to the maximum number of controlled axes. You can change the number of real axes to used axes up to the maximum number of used real axes.

Item	NX701-17□□	NX701-16□□
Settable axis numbers	0 to 255	0 to 127
Maximum number of used real axes	256 axes	128 axes

Item	NX502-17□□*1	NX502-16□□*1	NX502-15□□	NX502-14□□	NX502-13□□
Settable axis numbers	0 to 255	0 to 127	0 to 63	0 to 31	0 to 15
Maximum number of used real axes	256 axes	128 axes	64 axes	32 axes	16 axes

*1. Models added from the CPU Unit version 1.66.

Item	NX102-12□□	NX102-11□□	NX102-10□□	NX102-90□□
Settable axis numbers	0 to 14	0 to 14	0 to 14	0 to 3
Maximum number of used real axes	12 axes	8 axes	6 axes	4 axes
Used motion control servo axes	8 axes	4 axes	2 axes	---
Used single-axis position control servo axes	4 axes	4 axes	4 axes	4 axes

Item	NX1P2-11□□□□	NX1P2-10□□□□	NX1P2-90□□□□	NX1P2-9B□□□□
Settable axis numbers	0 to 11	0 to 9	0 to 3	0 or 1
Maximum number of used real axes	8 axes	6 axes	4 axes	2 axes
Used motion control servo axes	4 axes	2 axes	---	---
Used single-axis position control servo axes	4 axes	4 axes	4 axes	2 axes

Item	NJ501-□5□□	NJ501-□4□□	NJ501-□3□□
Settable axis numbers	0 to 63	0 to 31	0 to 15
Maximum number of used real axes	64 axes	32 axes	16 axes

Item	NJ301-12□□	NJ301-11□□	NJ101-10□□
Settable axis numbers	0 to 14*1	0 to 14*2	0 to 5
Maximum number of used real axes	8 axes	4 axes	2 axes

*1. The range is 0 to 7 for a CPU Unit with unit version 1.05 or earlier.

*2. The range is 0 to 3 for a CPU Unit with unit version 1.05 or earlier.

Motion Control

For the NX701 CPU Unit, the axes to use are assigned to either of the primary periodic task or priority-5 periodic task.

Additional Information

The **Motion control** setting is not provided for the NX502 CPU Unit, NX102 CPU Unit, NX1P2 CPU Unit, and NJ-series CPU Unit, which support only the primary periodic task.

Axis Types

The following table describes the different axis types that you can select in the Axis Type parameter.

Axis type	Description
Servo axis	These axes are used by the EtherCAT slave Servo Drives and NX-series Position Interface Units.*1 They are assigned to actual Servo Drives or other devices. One Servomotor is used as one axis. If you use NX-series Position Interface Units, you can assign more than one device, such as a Pulse Output Unit and Digital Input Unit, to the same axis.
Virtual servo axis	These virtual axes exist only inside the MC Function Module. They are not used by actual Servo Drives. For example, they are used as master axes for synchronizing control.
Encoder axis	These axes are used by the EtherCAT slave Encoder Input Terminals and NX-series Position Interface Units.*1 An encoder axis is assigned to an actual encoder input terminal or other device. If one encoder input terminal contains two counters, each counter will act as one axis.
Virtual encoder axis	These virtual axes are used for encoder operation. A virtual encoder axis is used temporarily in place of an encoder axis when there is no physical encoder. *2

*1. Refer to *1-4-3 Function Specifications* on page 1-12 for the controllable devices.

*2. Virtual encoder axes are used in combination with motion control instructions that update the actual position of the virtual encoder axis.
They cannot be used in place of encoder axes for versions of the MC Function Module that do not support such instructions.

● Virtual Servo Axes

A virtual servo axis does not have a physical encoder or external I/O signals.
Therefore, virtual servo axes differ from servo axes in the following ways.

- They are always in Servo ON state.
- The actual current position equals the command current position.
  However, there is sometimes calculation error because processing is performed with long reals in the MC Function Module.
- The actual current velocity equals the command current velocity.
  However, there is sometimes calculation error because processing is performed with long reals in the MC Function Module.
- External input signals cannot be used.

- If the MC_Home or MC_HomeWithParameter instruction is executed, the instruction is processed as a zero position preset regardless of the setting of the Homing Method axis parameter.
- If a motion control instruction that uses a latch function is executed, you must set the trigger input condition to Controller Mode. An error does not occur if you set it to Drive Mode, but a latch trigger will not occur, so execution of the instruction will not end.
  Latches are used by the following instructions: MC_TouchProbe (Enable External Latch), MC_MoveFeed (Interrupt Feeding), MC_MoveLink (Synchronous Positioning), and other instructions.
- Errors do not occur for immediate stop inputs or positive/negative limit inputs because the input signals do not exist.

● Encoder Axes and Virtual Encoder Axes

Encoder and virtual encoder axes differ from servo and virtual servo axes in the following ways.

- They do not have command positions. They have only actual positions.
- You cannot use motion-type motion control instructions for them.

Precautions for Correct Use

You cannot set the axis type of a single-axis position control axis to Encoder axis or Virtual encoder axis.

Control Function

Select the function of axis to control.
This setting is selectable only when you use an NX102 CPU Unit or NX1P2 CPU Unit.

Control Function	Description
All	Sets the axis to a *motion control axis* for which all controls can be used.
Single-axis position control only	Sets the axis to a *single-axis position control axis* for which only single-axis position control can be used.

The following table shows the axis types that are available with the selected Control Function.

Axis type	Control Function	
	All	Single-axis position control only
Servo axis	○	○
Virtual servo axis	○	○
Encoder axis	○	×
Virtual encoder axis	○	×

The following table shows the enabled functions.

Motion control function	Control Function	
	All	Single-axis position control only
Single-axis position control	○	○
Single-axis synchronized control	○	×
Single-axis velocity control	○	○*1
Single-axis torque control	○	×

Motion control function	Control Function	
	All	Single-axis position control only
Multi-axes coordinated control	○	×

*1. You can use only the MC_MoveVelocity (Velocity Control) instruction.

Refer to *Section 9 Motion Control Functions* on page 9-1 for details on motion control functions.

Additional Information

Select **Single-axis position control only** when you use NX102-90□□ or NX1P2-9□□□□□.

Input Device/Output Device

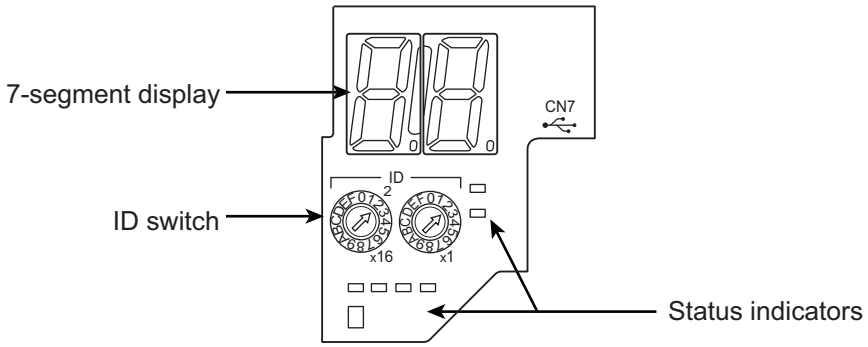
For a servo or encoder axis, the node address specifies the node address of the EtherCAT slave device that is assigned to the axis.
For an NX-series Position Interface Unit, select the node address of the EtherCAT Coupler Unit and the NX Unit number of the Position Interface Unit.
The Node Address parameter cannot be selected if the Axis Type parameter is set to a **virtual axis**.

Precautions for Correct Use

- OMRON 1S-series Servo Drives and G5-series Servo Drives can be set to specific node addresses by using the node address switches on the front panels.
  If the node address switches are set to 00, the node address will be determined by the settings made in the EtherCAT Editor of the Sysmac Studio.
  If the node address switches are set to 00 for all connected Servo Drives, errors will not occur even if the Servo Drive's connection position is changed. Set the node addresses on the node address switches to assign specific Servo Drives for each machine control.
- The value set on the Servo Drive's node address switches is loaded only once when the Servo Drive's control power is turned ON.
  Such changes are enabled only after the power supply is turned ON again.
  Do not change the setting on the node address switches after the power supply has been turned ON.
- An error occurs if the same node address is used more than once.

Additional Information

- The following example shows the EtherCAT device's node address setting for an OMRON 1S-series Servo Drive with built-in EtherCAT communications.



- The rotary switches in the display area on the Servo Drive are used to set the EtherCAT node address.

Rotary switch setting	Node address setting range	
	OMRON slaves	Non-OMRON slaves*1
00	Value set from the Sysmac Studio (1 to 512*2)	Value set from the Sysmac Studio (1 to 512*2)
01 to FF	Node address switch setting	

*1. The value set from the Sysmac Studio will be used for all non-OMRON slaves, regardless of any setting at the slave.

*2. For the NJ-series CPU Unit, the set value is *1 to 192*. However, only for the NJ101 CPU Unit, the maximum number of slaves which can be connected is 64.
For the NX502 CPU Unit, the set value is *1 to 256*, and the maximum number of slaves which can be connected is 256.
For the NX102 CPU Unit, the set value is *1 to 192*, and the maximum number of slaves which can be connected is 64.
For the NX1P2 CPU Unit, the set value is *1 to 192*, and the maximum number of slaves which can be connected is 16.

5-2-3 Unit Conversion Settings

These parameters set position units.

Parameter name	Function	Setting range	Default
Unit of Display	Set the unit for command positions. • pulse • mm • μm • nm • degree • inch	0 to 5	0
Command Pulse Count Per Motor Rotation*1	Set the number of pulses per motor rotation for command positions according to the encoder resolution. *2 The command value is converted to a number of pulses based on the electronic gear ratio.	1 to 4,294,967,295	10,000

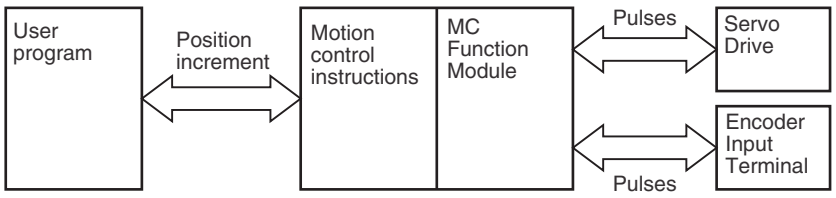
Parameter name	Function	Setting range	Default
Work Travel Distance Per Motor Rotation*3	Set the workpiece travel distance per motor rotation for command positions.	0.000000001 to 4,294,967,295	10,000
Reducer Use*4 *5	Specify whether to use the reducer setting or not. TRUE: Used. FALSE: Not used.	TRUE or FALSE	FALSE
Work Travel Distance Per Rotation*4 *6 *7	Set the work travel distance per rotation.	Positive long reals	10,000
Work Gear Ratio*4 *6	Set the gear ratio for the workpiece.	1 to 4,294,967,295	1
Motor Gear Ratio*4 *6	Set the gear ratio of the motor.	1 to 4,294,967,295	1

*1. This is the numerator of the electronic gear ratio (unit conversion formula).
*2. For example, if the encoder resolution is 10,000 pulses/rotation, set *10,000*.
*3. This is the denominator of the electronic gear ratio (unit conversion formula) when you set **not to use** the reducer.
This parameter is disabled when you set to **use** the reducer.
*4. A CPU Unit with unit version 1.11 or later and Sysmac Studio version 1.15 or higher are required to use this parameter.
*5. When you set to **use** the reducer, some conditions must be met to make the unit conversion settings.
Refer to *Conditions to Use Reducers* on page 5-15 for the conditions.
*6. This parameter is enabled when you set to **use** the reducer.
*7. The setting is possible only when the Count Mode is Linear Mode. When the Count Mode is Rotary Mode, the parameter value is calculated from the modulo maximum position setting value and modulo minimum position setting value.

Precautions for Correct Use

- Set to **use** the reducer if you use the **Count Mode** to **Rotary Mode**.
When you set **not to use** the reducer, the number of pulses for one cycle of the ring counter may not be an expected integer because of a calculation error for one cycle of the ring counter when converted to pulses. The unexpected integer may cause the position offset.
- When you make a change in the unit conversion settings, except for a change in the unit of display conversion, there are some differences between the physical position of the machine and the command current position of the MC Function Module. Therefore, if you made a change in the unit conversion settings, execute the Home instruction to define the home again.

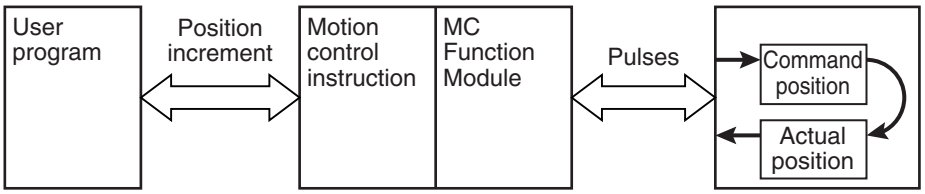
Positions are generally given in pulses between the MC Function Module and Servo Drives or encoder input terminals.
Use a display unit of millimeters or degrees for motion control instructions so that you can easily understand the operation.



You can use the Unit of Display parameter and electronic gear (unit conversion formula) settings to change from a pulse unit to millimeters or degrees.

Additional Information

For a virtual servo axis, the command current value is converted to pulses and then that value is converted to the unit of display. The resulting value is used as the actual current value.



Unit of Display

You can use the Unit of Display parameter to set the unit to display on the Sysmac Studio. The display shows the position's display unit.
The following table describes the units you can set.

Unit	Description
pulse	Use this unit to express values in pulses.
mm	Use this unit for comparatively long-distance direct operation.
μm	Use this unit for precise direct operation.
nm	Use this unit for more precise direct operation than μm.
degree	Use this unit for rotary tables or other rotating axes.
inch	Use this unit for direct operation.

Conditions to Use Reducers

When you set to **use** the reducer, the following condition must be met to use the unit conversion settings.

- The result of the following calculation must be equal to or between 0.000000001 and 4,294,967,295: Work travel distance per rotation × Work gear ratio ÷ Motor gear ratio.

When the **Count Mode** is **Rotary Mode**, the following condition must also be met.

- The result of the following calculation must be equal to or less than 1,099,511,627,775: Command pulse count per motor rotation × Motor gear ratio.

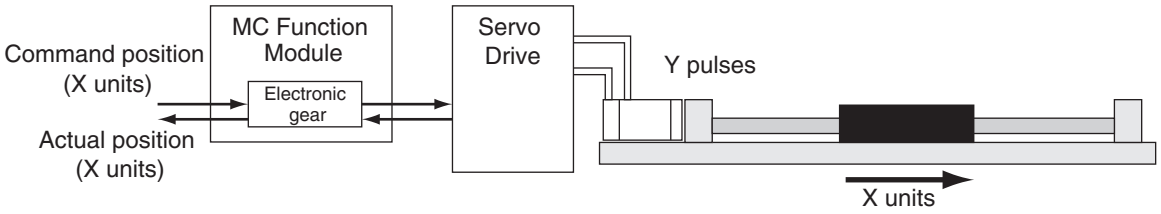
The work gear ratio and the motor gear ratio in the above calculations are determined from the division by the highest common factor of the two.
Therefore, even if the condition is not met by the calculation with the set values, the condition may be met in actual operation.

● Example

When the work gear ratio is 4 and the motor gear ratio is 6, you assume the work gear ratio to be 2 and the motor gear ratio to be 3 in the calculation.

Electronic Gear Ratio (Unit Conversion Formula)

Use the electronic gear to set the relationship between the display unit and pulse unit in the MC Function Module.
Use the Sysmac Studio and set the electronic gear ratio.



Command position value (pulses) = Command position (X units) × Electronic gear ratio

● When Not Using a Reducer

When you set **not to use** the reducer, the following formula is used to express the electronic gear ratio.

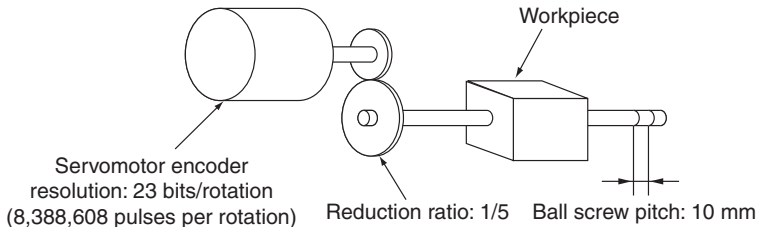
$$\text{Electronic gear ratio} = \frac{\text{Command Pulse Count Per Motor Rotation*1 (Y Pulses)}}{\text{Work Travel Distance Per Motor Rotation*2 (X Units)}}$$

*1. For an encoder axis, this is the number of pulses per encoder rotation.
*2. For an encoder axis, this is the travel distance per encoder rotation.

Precautions for Correct Use

The electronic gear converts units to the values that are used for positioning by the MC Function Module and motion control instructions.
Motion control instructions specify the target position as LREAL data. However, an instruction error will occur if the command position after conversion to pulses by the electronic gear exceeds 40 bits.

In this example, an OMRON 1S-series Servomotor with a 23-bit absolute encoder is used. Mechanically, the reduction ratio of the reducer is 1/5 and the workpiece moves 10 mm for every rotation of the ball screw.



The Unit of Display parameter is set to millimeters. The Command Pulse Count Per Motor Rotation is set to the resolution of the encoder on the Servomotor.
A reducer with a reduction ratio of 1/5 is used, so the ball screw turns once for every five rotations of the Servomotor. The workpiece moves 2 mm (10 mm × 1/5), so the Work Travel Distance Per Motor Rotation is set to 2.

Parameter name	Setting
Unit of Display	mm
Command Pulse Count Per Motor Rotation	8,388,608
Work Travel Distance Per Motor Rotation	2

With these settings, the command unit for positions in the user program is 1 mm.
For example, to move to an absolute position of 100.5 mm, the *Position (Target Position)* input variable to the MC_MoveAbsolute (Absolute Positioning) instruction is set to *100.5*.

Additional Information

Parameter Settings for a Reduction Ratio of 1/9 for the Example
The travel distance of the workpiece for one rotation of the Servomotor is 10 mm × 1/9, or 1.1111... mm (a repeating decimal number).
For numbers that do not divide evenly, multiply the **command pulse count per motor rotation** and the **work travel distance per motor rotation** by the same coefficient and set the parameters to the results.
Here, the reduction ratio is 1/9, so we use 9 as our coefficient.
- **Command Pulse Count Per Motor Rotation**: 75,497,472 (8,388,608 x 9)
- **Work Travel Distance Per Motor Rotation**: 10 (10 x 1/9 x 9)

● When Using a Reducer and the Count Mode Is Linear Mode

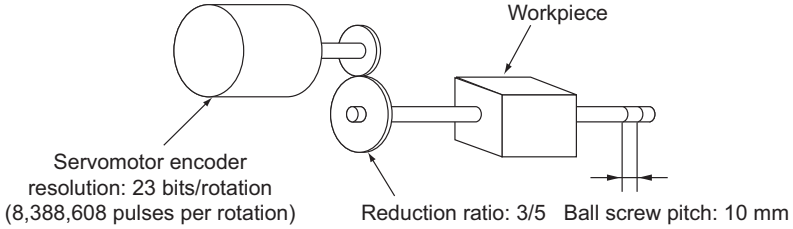
When you set to **use** a reducer and the **Count Mode** is **Linear Mode**, the following calculation formula is used to have the electric gear ratio.

$$\text{Electronic gear ratio} = \frac{\text{Command Pulse Count Per Motor Rotation} \times \text{Motor Gear Ratio}}{\text{Work Travel Distance Per Rotation} \times \text{Work Gear Ratio}}$$

Precautions for Correct Use

The electronic gear converts units to the values that are used for positioning by the MC Function Module and motion control instructions.
Motion control instructions specify the target position as LREAL data. However, an instruction error will occur if the command position after conversion to pulses by the electronic gear exceeds 40 bits.

In this example, an OMRON 1S-series Servomotor with a 23-bit absolute encoder is used.
Mechanically, the reduction ratio of the reducer is 3/5 and the workpiece moves 10 mm for every rotation of the ball screw.



The Unit of Display parameter is set to millimeters. The Command Pulse Count Per Motor Rotation is set to the resolution of the encoder on the Servomotor.
The Work Travel Distance Per Rotation is set to 10 mm, which equals to the ball screw pitch.

A reducer with a reduction ratio of 3/5 is used, so the ball screw turns three times for every five rotations of the Servomotor. For this reduction ratio setting, the work gear ratio is set to *3* and the motor gear ratio is set to *5*.

Parameter name	Setting
Unit of Display	mm
Command Pulse Count Per Motor Rotation	8,388,608
Work Travel Distance Per Rotation	10
Work Gear Ratio	3
Motor Gear Ratio	5

With these settings, the command unit for positions in the user program is 1 mm.
For example, to move to an absolute position of 100.5 mm, the *Position (Target Position)* input variable to the MC_MoveAbsolute (Absolute Positioning) instruction is set to *100.5*.

● When Using a Reducer and the Count Mode Is Rotary Mode

When you set to **use** a reducer and the **Count Mode** is **Rotary Mode**, the following calculation formula is used to have the electric gear ratio.
In **Rotary Mode**, the Work Travel Distance Per Rotation is automatically determined and set by the result of “Modulo maximum position - Modulo minimum position”.

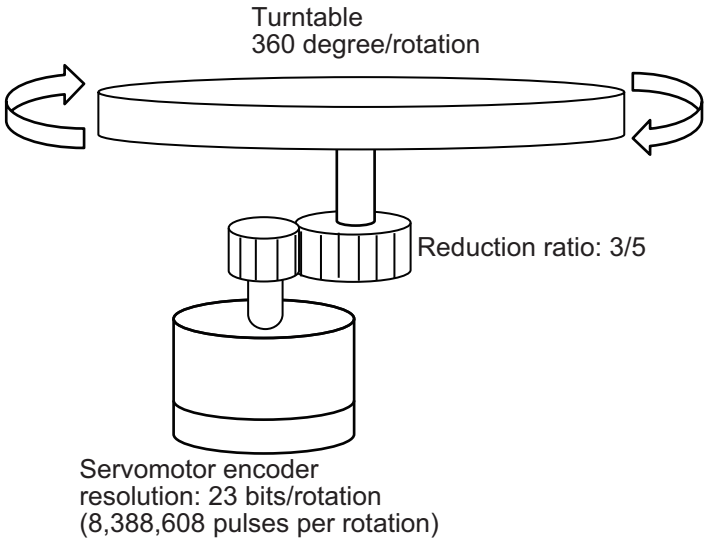
$$\text{Electronic gear ratio} = \frac{\text{Command Pulse Count Per Motor Rotation} \times \text{Motor Gear Ratio}}{\text{Work Travel Distance Per Rotation} \times \text{Work Gear Ratio}}$$

$$= \frac{\text{Command Pulse Count Per Motor Rotation} \times \text{Motor Gear Ratio}}{(\text{Modulo maximum position} - \text{Modulo minimum position}) \times \text{Work Gear Ratio}}$$

Precautions for Correct Use

The electronic gear converts units to the values that are used for positioning by the MC Function Module and motion control instructions.
Motion control instructions specify the target position as LREAL data. However, an instruction error will occur if the command position after conversion to pulses by the electronic gear exceeds 40 bits.

In this example 1, an OMRON 1S-series Servomotor with a 23-bit absolute encoder is used. Mechanically, the reduction ratio of the reducer is 3/5 and the workpiece moves 360 degree for every rotation of the turntable.

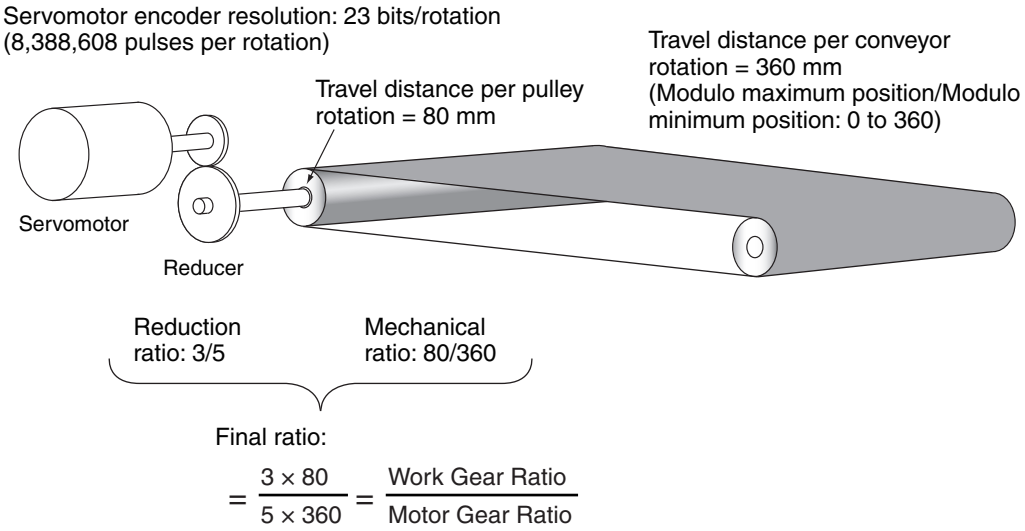


The Unit of Display parameter is set to degree. The Command Pulse Count Per Motor Rotation is set to the resolution of the encoder on the Servomotor.
The Work Travel Distance Per Rotation is automatically determined and set by the result of “Modulo maximum position - Modulo minimum position”.
A reducer with a reduction ratio of 3/5 is used, so the turntable (or workpiece) turns three times for every five rotations of the Servomotor. For this reduction ratio setting, the work gear ratio is set to *3* and the motor gear ratio is set to *5*.

Parameter name	Setting
Unit of Display	degree
Command Pulse Count Per Motor Rotation	8,388,608
Work Travel Distance Per Rotation	Modulo maximum position - Modulo minimum position
Work Gear Ratio	3
Motor Gear Ratio	5

With these settings, the command unit for positions in the user program is 1 degree.
For example, to move to an absolute position of 100.5 degree, the *Position (Target Position)* input variable to the MC_MoveAbsolute (Absolute Positioning) instruction is set to *100.5*.

In this example 2, an OMRON 1S-series Servomotor with a 23-bit absolute encoder is used. Mechanically, the reduction ratio of the reducer is 3/5 and the conveyor moves 80 mm for every rotation of the pulley. The travel distance per conveyor rotation is 360 mm.



The Unit of Display parameter is set to millimeters. The Command Pulse Count Per Motor Rotation is set to the resolution of the encoder on the Servomotor.
The travel distance per conveyor rotation is automatically set by the result of “Modulo maximum position - Modulo minimum position”.

Parameter name	Setting
Modulo Maximum Position	360
Modulo Minimum Position	0

A reducer with a reduction ratio of 3/5 is used, so the pulley turns three times for every five rotations of the Servomotor. The conveyor travel distance per pulley rotation is 80 mm and the travel distance per conveyor rotation is 360. For this reduction ratio setting, the work gear ratio is set to *240* (3 × 80) and the motor gear ratio is set to *1,800* (5 × 360).

Parameter name	Setting
Unit of Display	mm
Command Pulse Count Per Motor Rotation	8,388,608
Work Travel Distance Per Rotation	Modulo maximum position - Modulo minimum position
Work Gear Ratio	240
Motor Gear Ratio	1,800

Additional Information

- If the travel distance per pulley rotation is not an integer (e.g. 80.1 mm), use a ratio multiplied by the coefficient of 10, which converts values to integers, to calculate the work gear ratio and the motor gear ratio. When the above example is used, the work gear ratio is set to *2,403* (3 × 80.1 × 10), and the motor gear ratio is set to *18,000* (5 × 360 × 10).
- In the same way as the above example 2, when a multi-step reducer is used, multiply several ratios together and use a final ratio to set the work gear ratio and the motor gear ratio.

5-2-4 Operation Settings

These parameters set items for axis operation, such as the maximum velocity and maximum acceleration/deceleration rate. Set them according to the specifications of the device you are controlling.

Parameter name	Function	Setting range	Default
Maximum Velocity	Set the maximum velocity for each axis. *1 Do not set a value that exceeds the maximum speed of the motor that you are using. (Unit: command units/s)	Positive long reals*2	400,000,000
Start Velocity*3	Set the start velocity for each axis. Set a value that does not exceed the maximum velocity. (Unit: command units/s)	Positive long reals	0
Maximum Jog Velocity	Set the maximum jog velocity for each axis. *4 Set a value that does not exceed the maximum velocity. (Unit: command units/s)	Positive long reals	1,000,000
Maximum Acceleration	Set the maximum acceleration rate for an axis operation command. There will be no limit to the acceleration rate if *0* is set. (Unit: command units/s^2)	Non-negative long reals	0
Maximum Deceleration	Set the maximum deceleration rate for an axis operation command. There will be no limit to the deceleration rate if *0* is set. (Unit: command units/s^2)	Non-negative long reals	0
Acceleration/Deceleration Over	Set the operation for when the maximum acceleration/deceleration rate would be exceeded after excessive acceleration/deceleration during acceleration/deceleration control of the axis because stopping at the target position is given priority. 0: Use rapid acceleration/deceleration. (Blending is changed to Buffered.) *5 1: Use rapid acceleration/deceleration. 2: Minor fault stop *6	0 to 2	0
Operation Selection at Reversing	Specify the operation for reversing rotation for multi-execution of instructions, re-execution of instructions, and interrupt feeding. *7 0: Deceleration stop 1: Immediate stop	0 to 1	0
Velocity Warning Value	Set the percentage of the maximum velocity at which to output a velocity warning for the axis. No velocity warning is output if *0* is set. (Unit: %)	0 to 100	0
Acceleration Warning Value	Set the percentage of the maximum acceleration rate at which to output an acceleration warning for the axis. No acceleration warning is output if *0* is set. (Unit: %)	0 to 100	0

Parameter name	Function	Setting range	Default
Deceleration Warning Value	Set the percentage of the maximum deceleration rate at which to output a deceleration warning for the axis. No deceleration warning is output if *0* is set. (Unit: %)	0 to 100	0
Positive Torque Warning Value*8	Set the torque command value at which to output a positive torque warning. No positive torque warning is output if *0* is set. (Unit: %)	0 to 1,000	0
Negative Torque Warning Value*8	Set the torque command value at which to output a negative torque warning. No negative torque warning is output if *0* is set. (Unit: %)	0 to 1,000	0
Actual Velocity Filter Time Constant	Set the time period to calculate the average travel of the actual velocity in milliseconds. The average travel is not calculated if *0* is set. (Unit: ms) Use this to reduce variations in the actual current velocity when axis velocity is slow.	0 to 100	0
In-position Range*9	Set the in-position width. (Unit: command units)	Non-negative long reals	10
In-position Check Time*9	Set the in-position check time in milliseconds. Set *0* to check for the end of positioning only when you define the home position during homing and not check positioning at other times. (Unit: ms)	0 to 10,000	0
Zero Position Range	Set the home position detection width. (Unit: command units)	Non-negative long reals	10

*1. The maximum velocity is used as the command velocity if you specify a velocity command value that is greater than the maximum velocity. This parameter also applies to interpolation control operation.
Do not set any value that exceeds the motor maximum velocity (maximum rotation speed) that can be output by the Servomotor/Servo Drive connected to the axis. Check the specifications of the connected Servomotor/Servo Drive before you set this parameter.

*2. The maximum value that you can set is as follows when the value is converted to pulses:
For a CPU Unit with unit version 1.11 or later, the value is 2,147,483,647 [pulses/s]. For a CPU Unit with unit version 1.03 or later and 1.10 or earlier, the value is 500,000,000 [pulses/s]. For a CPU Unit with unit version 1.02 or earlier, the value is 400,000,000 [pulses/s].

*3. A CPU Unit with unit version 1.05 or later and Sysmac Studio version 1.06 or higher are required to use this parameter.

*4. The maximum jog velocity is used as the command velocity if you specify a velocity command value that is greater than the maximum jog velocity.

*5. For a CPU Unit with unit version 1.10 or later, Blending is not changed to Buffered.
Refer to *9-5-7 Multi-execution of Motion Control Instructions (Buffer Mode)* on page 9-53 for details.

*6. For a CPU Unit with unit version 1.10 or later, the axis does not stop with an error when Blending is used for operation.
Refer to *9-5-7 Multi-execution of Motion Control Instructions (Buffer Mode)* on page 9-53 for details.

*7. Refer to *9-5-6 Re-executing Motion Control Instructions* on page 9-48 and *9-5-7 Multi-execution of Motion Control Instructions (Buffer Mode)* on page 9-53 for details on the Operation Selection at Reversing parameter.

*8. This parameter is enabled only for torque control.

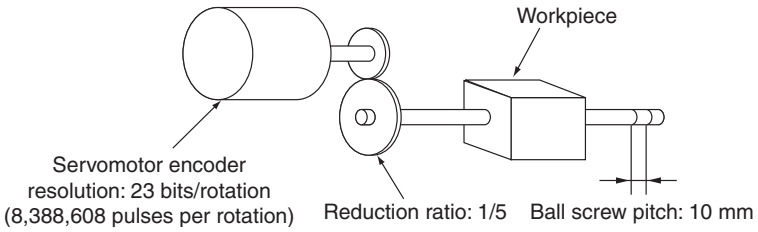
*9. The in-position check is processed by the MC Function Module. The function in the Servo Drive is not used.

Maximum Velocity

This section provides a setting example for the maximum velocity.

● Setting Example for the Maximum Velocity

The same machine as in *When **Not Using** a Reducer* on page 5-16 is described here for a Servomotor with a maximum speed of 6,000 r/min.



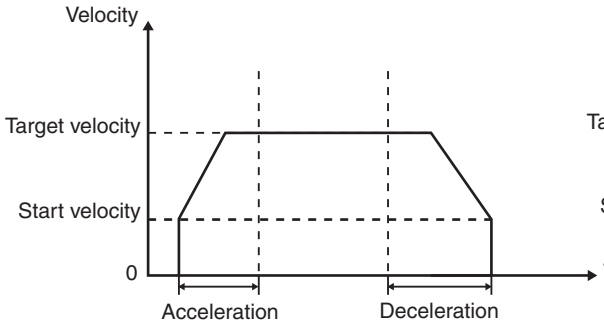
The Maximum Velocity is set to *200* based on a calculation for the conditions (maximum speed: 6,000 r/min, reduction ratio: 1/5, ball screw pitch: 10 mm; 6,000 r/min × 1/5 × 10 mm = 12,000 mm/min = 200 mm/s).
The default setting of 400,000,000 would exceed the maximum speed of the motor, so you must change the setting.

Parameter name	Setting
Unit of Display	mm
Command Pulse Count Per Motor Rotation	8,388,608
Work Travel Distance Per Motor Rotation	2
Maximum Velocity	200

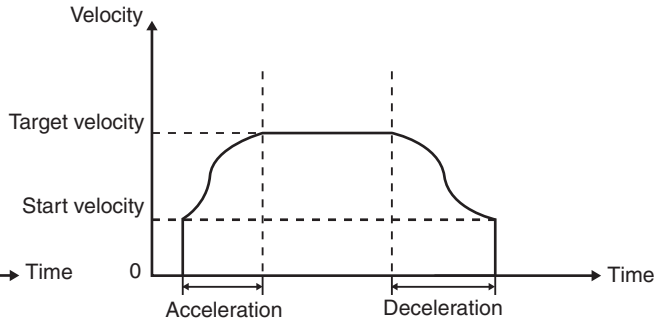
Start Velocity

Set the start velocity to *0* when you use a servomotor.
If you use a stepper motor, use 10% to 50% of the maximum self-start frequency to prevent loosing the sync at startup. However, this depends on the load, so refer to the manual for the stepper motor.

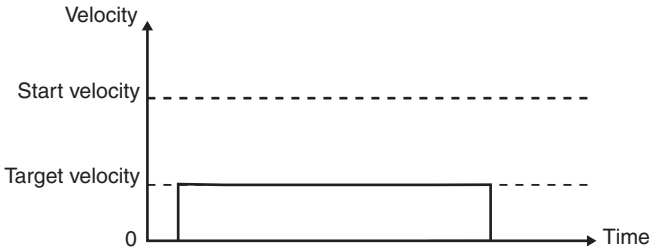
Not Specifying Jerk



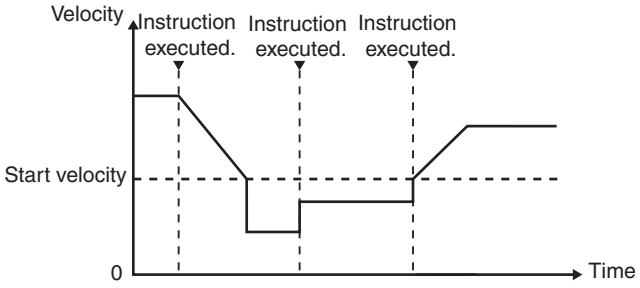
Specifying Jerk



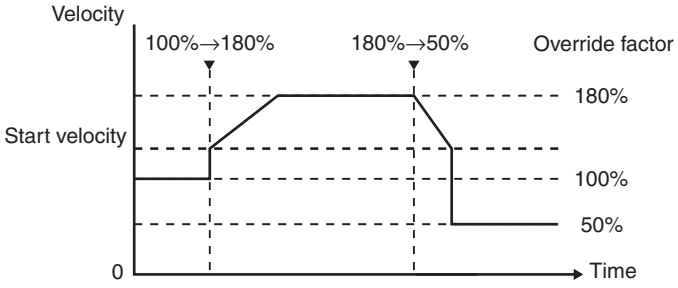
If the target velocity is less than or equal to the start velocity, acceleration/deceleration are not performed and the axis moves at the target velocity.



If the target velocity changes as the result of re-executing the motion control command or as the result of performing multi-execution of instructions for it during motion, the start velocity is used. If the target velocity is greater than the start velocity, acceleration/deceleration are performed at the specified acceleration/deceleration rates. If the target velocity is less than or equal to the start velocity, acceleration/deceleration are not performed and the axis moves.



The start velocity is also used if the velocity is changed by the MC_SetOverride instruction. If the target velocity is greater than the start velocity, acceleration/deceleration are performed at the specified acceleration/deceleration rates. If the target velocity is less than or equal to the start velocity, acceleration/deceleration are not performed and the axis moves.



The start velocity is not used in the following cases.

- Torque control
- Cyclic synchronous torque control
- Cyclic synchronous velocity control
- Cyclic synchronous positioning
- Synchronized control
  However, the start velocity is used for the MC_GearOut and MC_CamOut instructions.
- Multi-axes coordinated control
  However, the start velocity for each axis is used to decelerate the axes to a stop.

5-2-5 Other Operation Settings

These parameters are used to set the stopping methods and torque limits to use when the input signals are enabled.

Parameter name	Function	Setting range	Default
Immediate Stop Input Stop Method	Set the stopping method in the MC Function Module when the immediate stop input is enabled. 0: Immediate stop 2: Immediate stop and error reset 3: Immediate stop and Servo OFF	0, 2, or 3	0
Limit Input Stop Method	Set the stopping method in the MC Function Module when the positive limit input or negative limit input is enabled. 0: Immediate stop 1: Deceleration stop 2: Immediate stop and error reset 3: Immediate stop and Servo OFF	0 to 3	0
Drive Error Reset Monitoring Time	Set the monitor time for a drive error reset. (Unit: ms) After the monitor time has elapsed, reset processing will end even if the drive error is not yet reset.	1 to 1,000	200
Maximum Positive Torque Limit	Set the maximum value of the positive torque limit. *1 (Unit: %)	0.0 to 1000.0	300.0
Maximum Negative Torque Limit	Set the maximum value of the negative torque limit. *1 (Unit: %)	0.0 to 1000.0	300.0
Immediate Stop Input Logic Inversion*2	Set whether to reverse the logic of the immediate stop input signal. FALSE: Do not reverse. TRUE: Reverse.	FALSE or TRUE	FALSE*3
Positive Limit Input Logic Inversion*2	Set whether to reverse the logic of the positive limit input signal. FALSE: Do not reverse. TRUE: Reverse.	FALSE or TRUE	FALSE*3
Negative Limit Input Logic Inversion*2	Set whether to reverse the logic of the negative limit input signal. FALSE: Do not reverse. TRUE: Reverse.	FALSE or TRUE	FALSE*3
Home Proximity Input Logic Inversion*2	Set whether to reverse the logic of the home proximity input signal. FALSE: Do not reverse. TRUE: Reverse.	FALSE or TRUE	FALSE

*1. If **Positive Torque Limit Value (60E0 hex)** and **Negative Torque Limit Value (60E1 hex)** are mapped as PDOs, the set values of these parameters are sent with EtherCAT process data communications.
If a torque limit is enabled with the MC_SetTorqueLimit (Set Torque Limit) or MC_SetTorqueLimit2 (Set Torque Limit 2) instruction, the value that is specified with the input variable to the instruction is sent.

*2. A CPU Unit with unit version 1.05 or later and Sysmac Studio version 1.06 or higher are required to use this parameter. You cannot reverse the logic for a CPU Unit with a unit version of 1.04 or earlier.
This parameter is set for devices, such as NX-series Digital Input Units, for which the logic of the input signals cannot be set. For devices, such as OMRON 1S-series Servo Drives, for which you can set the input signal logic, set this parameter to **not reverse** the signal.

*3. If you assign an NX-series Pulse Output Unit to an axis, the default is *TRUE*.

5-2-6 Limit Settings

Use the following parameters to select functions for limiting the following error and for software limits.

Parameter name	Function	Setting range	Default
Software Limits*1	Select the software limit function. 0: Disabled. 1: Deceleration stop for command position 2: Immediate stop for command position 3: Deceleration stop for actual position 4: Immediate stop for actual position	0 to 4	0
Positive Software Limit	Set the software limit in the positive direction. (Unit: command units)	Long reals	2,147,483,647
Negative Software Limit	Set the software limit in the negative direction. (Unit: command units)	Long reals	-2,147,483,648
Following Error Over Value	Set the excessive following error check value. Set *0* to disable the excessive following error check. (Unit: command units)	Non-negative long reals	0
Following Error Warning Value	Set the following error warning check value. Set *0* to disable the following error warning check. (Unit: command units)	Non-negative long reals that are less than or equal to the Following Error Over Value	0

*1. This function is enabled only when the Count Mode is Linear Mode and the home is defined.
Refer to *9-8-5 Software Limits* on page 9-83 for details on software limits.

5-2-7 Position Count Settings

Set the count mode for the position.
Refer to the *NX-series Position Interface Units User's Manual (Cat. No. W524)* for information on using the NX-series Position Interface Units.

Parameter name	Function	Setting range	Default
Count Mode	Set the count mode for the position. 0: Linear Mode (finite length) 1: Rotary Mode (infinite length)	0 to 1	0
Modulo Maximum Position Setting Value	Set the modulo maximum position when the Count Mode is set to Rotary Mode. (Unit: command units)	Long reals	2,147,483,647
Modulo Minimum Position Setting Value	Set the modulo minimum position when the Count Mode is set to Rotary Mode. (Unit: command units)	Long reals	-2,147,483,648
Encoder Type	Set the encoder type. *1*2 0: Incremental encoder (INC) 1: Absolute encoder (ABS)	0 to 1	0

*1. Set the encoder type to **1 (absolute encoder (ABS))** when you use any of the following.
- When an OMRON 1S-series Servomotor/Servo Drive is used
- When an OMRON G5-series Servomotor with an absolute encoder is used
- When an OMRON G5-series Servomotor/Servo Drive with an absolute external scale for fully-closed control is used
- When an OMRON G5-series Linear Motor Type Servomotor/Servo Drive with built-in EtherCAT communications is used with an absolute external scale

*2. The settings are as follows when you use an OMRON G5-series Servomotor/Servo Drive with an external scale for fully-closed control, or when you use an OMRON G5-series Linear Motor Type Servomotor/Servo Drive with built-in EtherCAT communications.
0: Incremental external scale

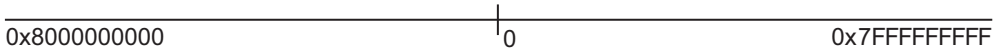
1: Absolute external scale

Count Modes

The Count Mode is the feed mode for the axis.
Select the count mode for the command positions for each axis.
There are two Count Modes: **Linear Mode**, which has a finite axis feed range and **Rotary Mode**, which has an infinite axis feed range.

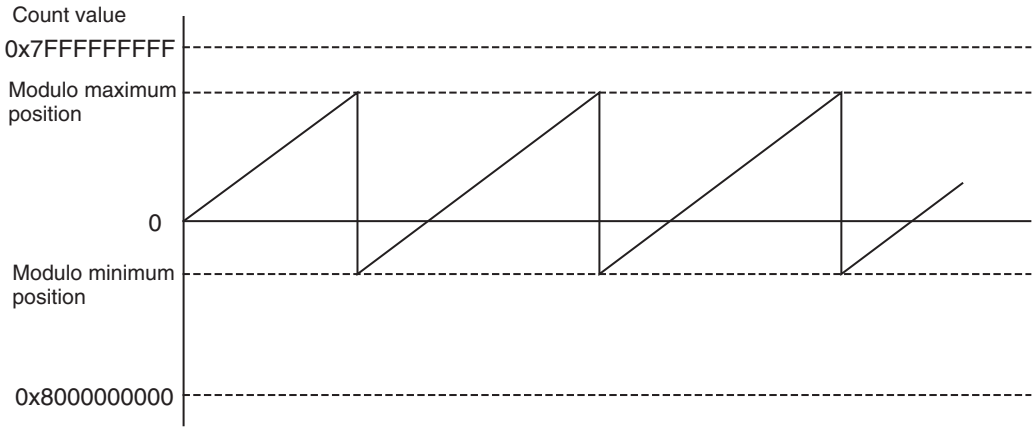
● Linear Mode (Finite Length Axis)

- The linear mode is centered around 0.
  This mode is used for devices with a mechanically limited range of motion, such as an XY stage.
- The setting range when the value is converted to pulses is 40 bits (signed integer: 0x8000000000 to 0x7FFFFFFFFF).
- You cannot specify a target position for relative or absolute positioning that exceeds this range.
- A command position overflow or underflow observation will occur if this range is exceeded for operations that do not have a target position, such as velocity control, homing, or torque control. Command position output will continue, but the actual position is not updated and will be fixed to either the upper limit or the lower limit.
- While the value of the actual position is fixed, you can execute commands and stop the axis with any operation that does not have a target position in the direction toward the linear range. Any command that specifies a direction away from the range will cause an error on execution of the instruction.
- The actual position does not update until the overflow or underflow status is cleared.



● Rotary Mode (Infinite Length Axis)

- This mode repeatedly counts with a ring counter for an infinite amount within the set range.
  Use this mode for rotary tables or winding shafts.
- Use the Sysmac Studio to set the modulo maximum position and the modulo minimum position to define the range of the ring counter.
- The setting range when the value is converted to pulses is 40 bits (signed integer: 0x8000000000 to 0x7FFFFFFFFF).



- The number of pulses for one cycle of the ring counter may not be an expected integer because of a calculation error for one cycle of the ring counter when converted to pulses. The unexpected integer may cause the position offset.

Version Information

Set to **use** the reducer to cancel the cause of the above position offset when you use a CPU Unit with unit version 1.11 or later.
Refer to *5-2-3 Unit Conversion Settings* on page 5-13 on use of reducers.

Modulo Maximum Position and Modulo Minimum Position Setting Values

The settings of these parameters are enabled when the Count Mode is set to Rotary Mode.
Set the upper and lower limits of the ring counter.

Precautions for Correct Use

- If 0 is not included between the upper and lower limits of the ring counter, an error occurs when the MC_MoveZeroPosition (High-speed Home) instruction is executed.
- When you perform absolute positioning with a MC_MoveAbsolute or MC_Move instruction, make sure that the target position is within the range of the ring counter. An error occurs if the target position is not within the range of the ring counter.
  If the *Direction* input variable to the instruction is set to **No direction specified**, you can set a target position that is not within the range of the ring counter. If that occurs, relative positioning is performed using the difference between the target position and the command current position as the target distance.
- When you make a change in the position count settings, there are some differences between the physical position of the machine and the command current position of the MC Function Module. Therefore, if you made a change in the position count settings, execute the Home instruction to define the home again.

Encoder Type

Set the type of encoder to use for feedback input.
Set the encoder type to **1 (absolute encoder (ABS))** when you use any of the following.

- When an OMRON 1S-series Servomotor/Servo Drive is used
- When an OMRON G5-series Servomotor with an absolute encoder is used
- When an OMRON G5-series Servomotor/Servo Drive with an absolute external scale for fully-closed control is used
- When an OMRON G5-series Linear Motor Type Servomotor/Servo Drive with built-in EtherCAT communications is used with an absolute external scale

5-2-8 Servo Drive Settings

Set the value that is set on the Servo Drive or the Encoder Input Terminal that is connected.

Parameter name	Function	Setting range	Default*1
Modulo Maximum Position Setting Value	Set the modulo maximum position that is set on the Servo Drive or the Encoder Input Terminal. *2	-2^{63} to $2^{63}-1$ (Unit: pulses)	2,147,483,647

Parameter name	Function	Setting range	Default*1
Modulo Minimum Position Setting Value	Set the modulo minimum position that is set on the Servo Drive or the Encoder Input Terminal. *2	-2^{63} to $2^{63}-1$ (Unit: pulses)	-2,147,483,648
PDS State Control Method*3	Set the state to which PDS state changes when Servo is turned OFF by the MC_Power instruction. *4 0: Switched on by Servo OFF 1: Ready to switched on by Servo OFF	0 to 1	0

*1. The default range is all DINT integers. You can use the default range with OMRON 1S-series Servo Drives or G5-series Servo Drives.
Refer to the *NX-series Position Interface Units User's Manual (Cat. No. W524)* for information on using the NX-series Position Interface Units.

*2. If you use an OMRON GX-series EtherCAT Slave Encoder Input Terminal, the maximum value of the ring counter (index 0x4003) of the Encoder Input Terminal must agree with the Modulo Maximum Position Setting Value. The modulo minimum position setting value must be set to *0*.

*3. A CPU Unit with unit version 1.10 or later and Sysmac Studio version 1.12 or higher are required to use this parameter.

*4. If you set this parameter to *1*, the Servo Ready (Switched on) status of OMRON G5-series Servo Drives cannot be used. To use the Servo Ready (Switched on) status, set this parameter to *0*. Refer to *A-5 PDS State Transition* on page A-28 for details on the PDS state transition.

5-2-9 Homing Settings

Set the motor operation to use to determine home.

Parameter name	Function	Setting range	Default
Homing Method*1*2	Set the homing operation. 0: Proximity reverse turn/home proximity input OFF 1: Proximity reverse turn/home proximity input ON 4: Home proximity input OFF 5: Home proximity input ON 8: Limit input OFF 9: Proximity reverse turn/home input mask distance 11: Limit inputs only 12: Proximity reverse turn/holding time 13: No home proximity input/holding home input 14: Zero position preset	0, 1, 4, 5, 8, 9, or 11 to 14	14
Home Input Signal	Select the input to use for the home input signal. 0: Use Z-phase input as home 1: Use external home input *3	0 or 1	0
Homing Start Direction	Set the start direction for when homing is started. 0: Positive direction 2: Negative direction	0 or 2	0
Home Input Detection Direction	Set the home input detection direction for homing. 0: Positive direction 2: Negative direction	0 or 2	0
Operation Selection at Positive Limit Input	Set the stopping method when the positive limit input turns ON during homing. 0: No reverse turn/minor fault stop (Stop according to Limit Input Stop Method parameter.) 1: Reverse turn/immediate stop 2: Reverse turn/deceleration stop	0 to 2	1

Parameter name	Function	Setting range	Default
Operation Selection at Negative Limit Input	Set the stopping method when the negative limit input turns ON during homing. 0: No reverse turn/minor fault stop (Stop according to Limit Input Stop Method parameter.) 1: Reverse turn/immediate stop 2: Reverse turn/deceleration stop	0 to 2	1
Homing Velocity	Set the homing velocity. (Unit: command units/s)	Positive long reals	10,000
Homing Approach Velocity	Set the velocity to use after the home proximity input turns ON. (Unit: command units/s)	Positive long reals	1,000
Homing Acceleration	Set the acceleration rate for homing. If the homing acceleration is set to *0*, the homing velocity or other target velocity is used without any acceleration. (Unit: command units/s^2)	Non-negative long reals	0
Homing Deceleration	Set the deceleration rate for homing. If the homing deceleration is set to *0*, the homing approach velocity or other target velocity is used without any deceleration. (Unit: command units/s^2)	Non-negative long reals	0
Homing Jerk	Set the jerk for homing. Set *0* for no jerk. (Unit: command units/s^3)	Non-negative long reals	0
Home Input Mask Distance	Set the home input mask distance when you set the Homing Operation Mode to a proximity reverse turn/home input mask distance. (Unit: command units)	Non-negative long reals	10,000
Home Offset	Preset the actual position for the value that is set after homing. (Unit: command units)	Long reals	0
Homing Holding Time	Set the holding time in milliseconds when you set the Homing Operation Mode to a proximity reverse turn/holding time. (Unit: ms)	0 to 10,000	100
Homing Compensation Value	Set the homing compensation value that is applied after the home is defined. (Unit: command units)	Long reals	0
Homing Compensation Velocity	Set the velocity to use for homing compensation. (Unit: command units/s)	Positive long reals	1,000

*1. These parameters are for homing operation.
Refer to *Section 8 Homing* on page 8-1 for details.

*2. You cannot map the Z-phase input to a PDO for an OMRON G5-series Linear Motor Type Servo Drive. Therefore, if you set the Homing Method to the **No home proximity input/holding home input**, which can use a Z-phase input mapped to a PDO, do not select the Z-phase input for the home input signal.

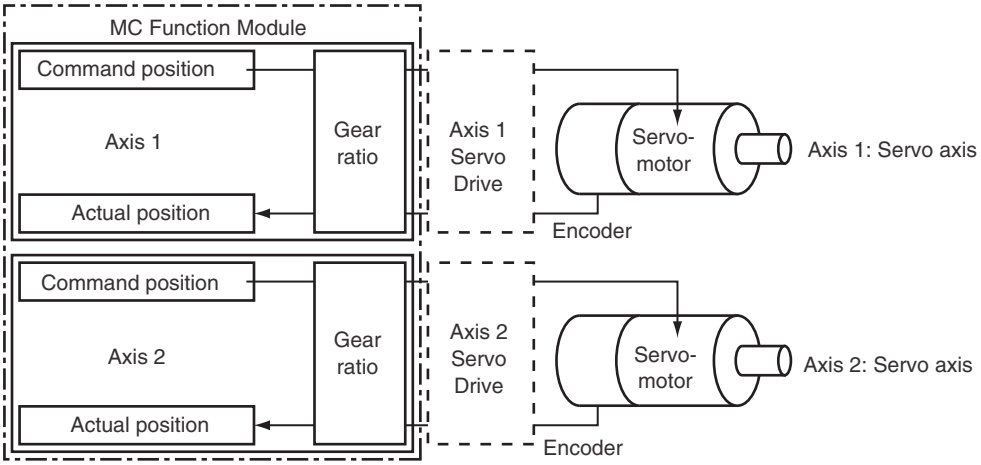
*3. This setting can be used for an OMRON 1S-series Servo Drive or G5-series Servo Drive. The input allocated to latch 1 for the Servo Drive is used as the external home input. In the default setting of the OMRON 1S-series Servo Drives or G5-series Servo Drives, the external latch input 1 is allocated to latch 1.
For details, refer to the *AC Servomotors/Servo Drives 1S-series with Built-in EtherCAT Communications User's Manual (Cat. No. I586)*, *AC Servomotors/Servo Drives 1S-series with Built-in EtherCAT Communications and Safety Functionality User's Manual (Cat. No. I621)*, *AC Servomotors/Servo Drives G5-series with Built-in EtherCAT Communications User's Manual (Cat. No. I576)*, or *AC Servomotors/Servo Drives G5-series with Built-in EtherCAT Communications Linear Motor Type User's Manual (Cat. No. I577)*.
Refer to the *NX-series Position Interface Units User's Manual (Cat. No. W524)* for information on using the NX-series Position Interface Units.

5-2-10 Axis Parameter Setting Example

This section provides examples of axis parameter settings related to positioning.

Single-axis Positioning

The following example is for a device that performs single-axis positioning separately for each of two axes.



Parameter name	Settings	
	Axis 1	Axis 2
Axis Variable Names	Axis1	Axis2
Axis Number	1	2
Enabled Axes	Used axis	Used axis
Axis Type	Servo axis	Servo axis
Input Device/Output Device	1	2
Unit of Display	μm*1	μm
Command Pulse Count Per Motor Rotation	1,048,576	1,048,576
Work Travel Distance Per Motor Rotation	10,000	10,000
Maximum Velocity	500,000*2	500,000*2
Maximum Jog Velocity	50,000*3	50,000*3
Maximum Acceleration	5,000,000*4	5,000,000*4
Maximum Deceleration	5,000,000*4	5,000,000*4
Software Limits	Immediate stop for command position	Immediate stop for command position
Positive Software Limit	500,000*5	500,000*5
Negative Software Limit	0*5	0*5
Count Mode	Linear Mode	Linear Mode

*1. The position command unit will be 1 μm.

*2. The maximum velocity will be 3,000 r/min = 30 m/min = 0.5 m/s = 500,000 μm/s.

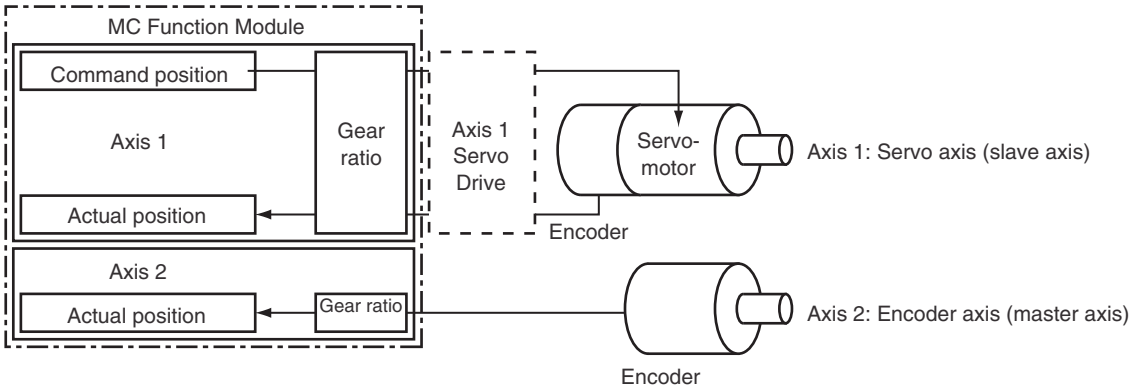
*3. The maximum jog velocity will be 10% of the maximum velocity, i.e., 0.05 m/s = 50,000 μm/s.

*4. The maximum acceleration and the maximum deceleration are 5 m/s^2. The acceleration time to the maximum velocity (3,000 r/min) is 0.1 s.

*5. Set a positioning that is within the movable range of the device.
The positive software limit is set to 50 cm = 500,000 μm.

Synchronized Control with Encoder as Master Axis

The following example is for a device that uses the actual position of axis 2 (an encoder), which is attached to a conveyor, as the master axis. The Servo Drive on axis 1 is synchronized within a finite range.



Parameter name	Settings	
	Axis 1	Axis 2
Axis Variable Name	Axis1	Axis2
Axis Number	1	2
Enabled Axes	Used axis	Used axis
Axis Type	Servo axis	Encoder axis
Input Device/Output Device	1	2
Unit of Display	μm*1	μm*1
Command Pulse Count Per Motor Rotation	1,048,576	1,048,576
Work Travel Distance Per Motor Rotation	10,000	10,000
Maximum Velocity	500,000*2	---
Maximum Jog Velocity	50,000*3	---
Maximum Acceleration	5,000,000*4	---
Maximum Deceleration	5,000,000*4	---
Software Limits	Immediate stop for command position	Disabled.
Positive Software Limit	500,000*5	---
Negative Software Limit	0*5	---
Count Mode	Linear Mode	Rotary Mode
Modulo Maximum Position	---	1,000,000*6
Modulo Minimum Position	---	0*6

*1. The position command unit will be 1 μm.
*2. The maximum velocity will be 3,000 r/min = 30 m/min = 0.5 m/s = 500,000 μm/s.
*3. The maximum jog velocity will be 10% of the maximum velocity, i.e., 0.05 m/s = 50,000 μm/s.

*4. The maximum acceleration and the maximum deceleration are 5 m/s^2. The acceleration time to the maximum velocity (3,000 r/min) is 0.1 s.
*5. Set a positioning that is within the movable range of the device.
The positive software limit is set to 50 cm = 500,000 μm.
*6. The periodic range of the position is 0 to 1 m (1,000,000 μm).

Additional Information

You can select the axis type for the master axis according to the configuration of the device. There are four axis types: servo axes, virtual servo axes, encoder axes, and virtual encoder axes.
In this example, the axis type of the master axis is an encoder axis. Specify the actual position for the motion control instruction input variable *ReferenceType* (Position Type Selection).

5-3 Axes Group Parameters

Use the axes group parameters to set axes group operations related to axes groups that the MC Function Module controls, such as the axis configuration, maximum interpolation velocity, and axes group stopping method.
The axes group parameters are provided for each axes group. The number of axes groups depends on the model.
Refer to *1-4-2 Performance Specifications* on page 1-7 for details.
The same parameter settings are provided for each axes group. This section describes only the parameters for axes group 1.

5-3-1 Axes Group Parameters

Use the Sysmac Studio to set the axes group parameters for each axes group.

Classification	Parameter name	Temporary changes*1		Reading variables*2	Page
		Support	Applicable instruction		
Axes Group Basic Settings	Axes Group Number			OK	page 5-35
	Motion Control*3			OK*4	
	Axes Group Use			OK	
	Composition			OK	
	Composition Axes	OK*5	MC_ChangeAxesInGroup	OK	
Axes Group Operation Settings	Maximum Interpolation Velocity				page 5-37
	Maximum Interpolation Acceleration				
	Maximum Interpolation Deceleration				
	Interpolation Acceleration/Deceleration Over				
	Interpolation Velocity Warning Value	OK	MC_Write		
	Interpolation Acceleration Warning Value				
	Interpolation Deceleration Warning Value				
	Axes Group Stop Method				
	Correction Allowance Ratio				

*1. This column indicates if you can use instructions to temporarily change the settings.
*2. This column indicates whether you can access the parameter with a variable in the user program.
*3. Set this parameter when using the NX701 CPU Unit.
*4. A CPU Unit with unit version 1.10 or later and Sysmac Studio version 1.12 or higher are required to use this parameter.

*5. A CPU Unit with unit version 1.01 or later and Sysmac Studio version 1.02 or higher are required to use this instruction.

Refer to *3-4 Setting Procedures for Axes Groups* on page 3-27 for details on how to set axes group parameters.
For details on the MC_Write (Write MC Setting) and MC_ChangeAxesInGroup instructions, refer to the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)*.
Refer to *6-6 System-defined Variables for Motion Control* on page 6-21 for information on system-defined variables for motion control.

5-3-2 Axes Group Basic Settings

Set whether to use the axes group. If you are going to use the axes group, set the axis composition and the axes to use.

Parameter name	Function	Setting range	Default
Axes Group Number	Set the logical number of the axes group. This number indicates the axes group number of the corresponding system-defined variable *_MC_GRP[0-63]*, *_MC1_GRP[0-63]*, or *_MC2_GRP[0-63]*.	---	---
Motion Control*1	Assign to either of the primary periodic task or priority-5 periodic task. 1: Primary periodic task 2: Priority-5 periodic task	1 or 2	1
Axes Group Use	Set whether to enable or disable the axes group. An error occurs if you execute a motion control instruction for an undefined or unused axes group. *2 0: Undefined axes group 1: Unused axes group 2: Used axes group	0 to 2	0
Composition	Set the axis composition of the axes group. 0: 2 axes 1: 3 axes 2: 4 axes	0 to 2	0
Composition Axes	Sets the axis number to assign to the axes group. Set Axis Variable names from the Sysmac Studio to use for the A0 to A3 axes.	2 to 4 axes	0

*1. Set this parameter when using the NX701 CPU Unit.
*2. An error occurs if you execute the MC_GroupEnable (Enable Axes Group) instruction for an axes group that contains an unused axis.

Composition

The following table lists the axis compositions you can use with the MC Function Module.
Use the Sysmac Studio to set the axis composition according to the actual devices.

Composition	Description
2 axes	A two-axis configuration is used. For example, a machine with a two-axis Cartesian coordinate system is used.

Composition	Description
3 axes	A three-axis configuration is used. For example, a machine with a three-axis Cartesian coordinate system is used.
4 axes	A four-axis configuration is used. For example, a machine with a three-axis Cartesian coordinate system is used with a rotary axis at the end tool.

Composition Axes

The axes that are in an axes group are called composition axes.
To make it easier to reuse programming with interpolation instructions for axes groups commands, logical axes (axis A0 to axis A3) are used instead of axis numbers (axis 0 to axis 255).
For the Composition Axes parameter, set the axis numbers and logical axis numbers for the axes in the axes group.
Servo axes or virtual servo axes can be selected for logical axes.
Use the Sysmac Studio to assign axes from axis A0 for the number of axes you selected in the axis composition.
Set axis numbers from axis A0 for each axes group if you create more than one axes group.
You can also set the same axis number in more than one axes group.

Axis composition setting	Settings in Composition Axes parameter
2 axes	Set Axis Variable names (axis numbers) for axis A0 and axis A1.
3 axes	Set Axis Variable names (axis numbers) for axis A0, axis A1, and axis A2.
4 axes	Set Axis Variable names (axis numbers) for axis A0, axis A1, axis A2, and axis A3.

Precautions for Correct Use

- Assign all of the composition axes to the same task.
- If you set the same axis number for multiple axes groups, you cannot enable these axes groups simultaneously.
- An axis for which **Control function** is set to **Single-axis position control only** cannot be allocated as an axis in an axes group.

Version Information

With a CPU Unit with unit version 1.04 or later and Sysmac Studio version 1.05 or higher, you can set any servo axis or virtual servo axis that is set to a Used axis or an Unused axis (changeable to used axis) in an axes group.

● Composition Axes Setting Examples

- Example 1: Assigning Four Axes with Axis Numbers 1, 2, 5, and 8 to an Axes Group

Logical axis	Axis number
Axis A0	Axis 1
Axis A1	Axis 2
Axis A2	Axis 5
Axis A3	Axis 8

- Example 2: Assigning Three Axes with Axis Numbers 1, 8, and 2 to an Axes Group

Logical axis	Axis number
Axis A0	Axis 1

Logical axis	Axis number
Axis A1	Axis 8
Axis A2	Axis 2
Axis A3	None

5-3-3 Axes Group Operation Settings

These parameters set items for axes group operation, such as the maximum interpolation velocity and axes group stopping method. Set them according to the specifications of the device you are controlling.

Refer to the *NX-series Position Interface Units User's Manual (Cat. No. W524)* for information on using the NX-series Position Interface Units.

Parameter name	Function	Setting range	Default
Maximum Interpolation Velocity	Set the maximum interpolation velocity for the path. Set *0* for no interpolation velocity limit. If a target velocity that exceeds the maximum interpolation velocity is specified for an axes group motion instruction, the axis will move at the maximum interpolation velocity. (Unit: command units/s)	Non-negative long reals	800,000,000
Maximum Interpolation Acceleration	Set the maximum interpolation acceleration for the path. Set *0* for no interpolation acceleration limit. (Unit: command units/s^2)	Non-negative long reals	0
Maximum Interpolation Deceleration	Set the maximum interpolation deceleration for the path. Set *0* for no interpolation deceleration limit. (Unit: command units/s^2)	Non-negative long reals	0
Interpolation Acceleration/Deceleration Over	Set the operation for when the maximum interpolation acceleration/deceleration rate would be exceeded after excessive acceleration/deceleration during acceleration/deceleration control of the axes group because stopping at the target position is given priority. 0: Use rapid acceleration/deceleration. (Blending is changed to Buffered.) *1 1: Use rapid acceleration/deceleration. 2: Minor fault stop *2	0 to 2	0
Interpolation Velocity Warning Value	Set the percentage of the maximum interpolation velocity at which to output an interpolation velocity warning. No interpolation velocity warning is output if *0* is set. (Unit: %)	0 to 100	0
Interpolation Acceleration Warning Value	Set the percentage of the maximum interpolation acceleration rate at which to output an interpolation acceleration warning. No interpolation acceleration warning is output if *0* is set. (Unit: %)	0 to 100	0
Interpolation Deceleration Warning Value	Set the percentage of the maximum interpolation deceleration rate at which to output an interpolation deceleration warning. No interpolation deceleration warning is output if *0* is set. (Unit: %)	0 to 100	0

Parameter name	Function	Setting range	Default
Axes Group Stop Method	Set the stop method of the composition axes for which an error did not occur when an error occurs that forces an immediate stop of an axis that is in a multi-axes coordinated motion. 0: Immediate stop 1: Decelerate axes to a stop at maximum deceleration rate of the axes 3: Immediate stop and Servo OFF	0, 1, or 3	0
Correction Allowance Ratio	This parameter applies when the center designation method is used for a circular interpolation instruction. It compensates the distance when the distance between the start point and the center point does not equal the distance between the end point and the center point. Set the allowable range for that correction as a percentage of the radius. Set the percentage to 0.1% or greater. Error checking is not performed if *0* is set.	Single-precision floating-point number between 0 and 100	0

*1. For a CPU Unit with unit version 1.10 or later, Blending is not changed to Buffered.
Refer to *9-5-7 Multi-execution of Motion Control Instructions (Buffer Mode)* on page 9-53 for details.

*2. For a CPU Unit with unit version 1.10 or later, the axis does not stop with an error when Blending is used for operation.
Refer to *9-5-7 Multi-execution of Motion Control Instructions (Buffer Mode)* on page 9-53 for details.

5-3-4 Enabling an Axes Group

Specify the number of the axes group to enable in the MC_GroupEnable (Enable Axes Group) instruction to enable operation instructions for an axes group in the user program.
An instruction error occurs if you execute a motion control instruction for an axes group that is not enabled.
You can enable more than one axes group at the same time, but if you enable more than one axes group that include the same axis, an instruction error occurs.

If you want to operate the same axis in different axes groups for each work process, create multiple axes groups that include that axis. You can then use the MC_GroupEnable (Enable Axes Group) and MC_GroupDisable (Disable Axes Group) instructions to enable and disable these axes groups as you need to use them.

If you execute the MC_GroupDisable (Disable Axes Group) instruction during multi-axes operation, the axes in the group will decelerate to a stop.

6

Motion Control Programming

This section provides the specifications of a motion control program and the operation procedures that are required up through actual program development.

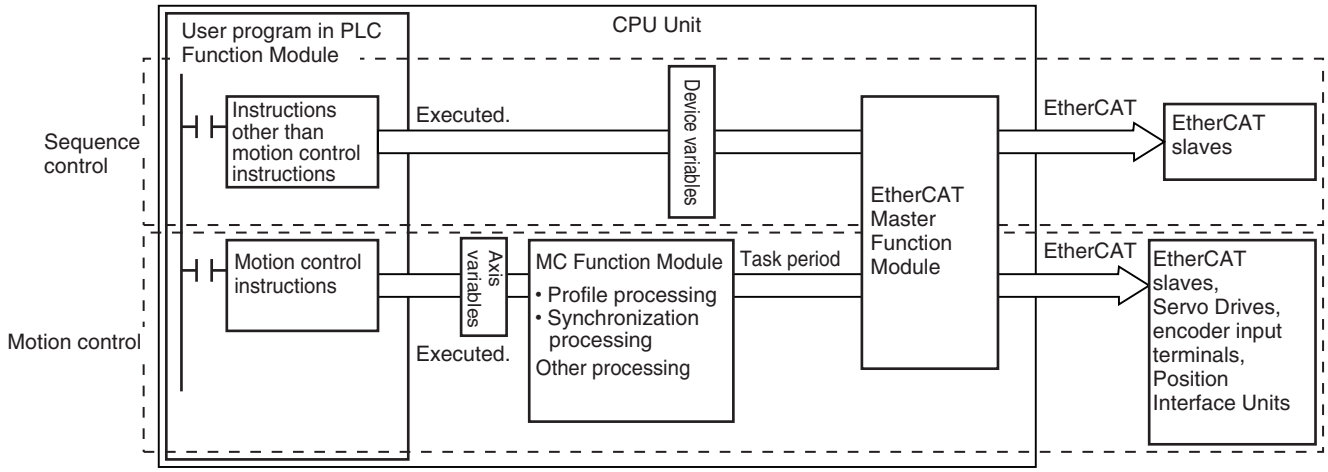
6-1 Introduction

The NJ/NX-series CPU Unit can perform both sequence control and motion control.
Write motion control instructions into the user program to perform motion control with EtherCAT slave Servo Drives, NX-series Position Interface Units, and other devices.
Programs that contain motion control instructions are called motion control programs.
You must assign Axis Variables to EtherCAT slave Servo Drives and NX-series Position Interface Units. If you do not assign Axis Variables, assign I/O device variables in the same way as for a general-purpose slave.
Motion control instructions can be used in the primary periodic task, in a priority-5 periodic task, and in a priority-16 periodic task.

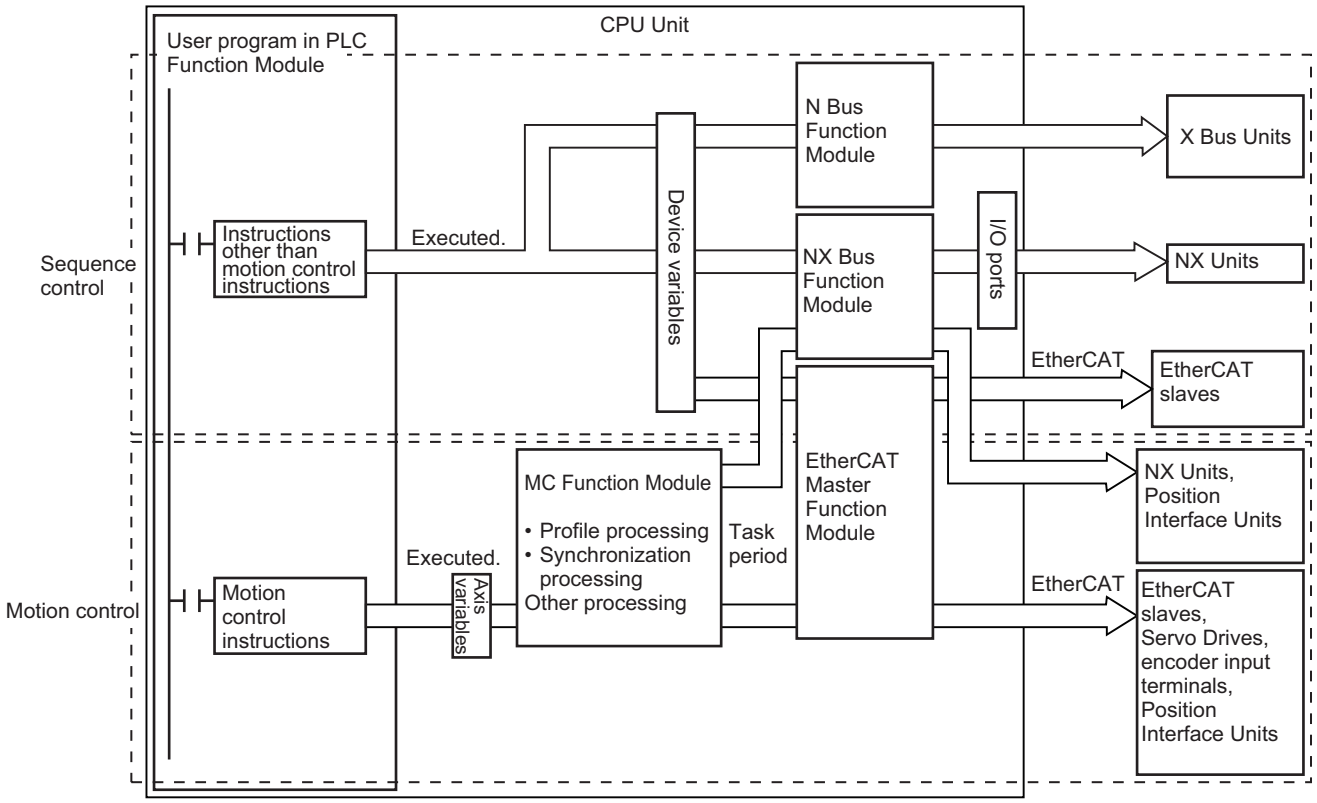
Version Information

With the Sysmac Studio version 1.09 or higher, you can assign device variables to the I/O ports of slaves and Units that are assigned to the Axis Variables.
Refer to *2-4-2 Relationship between EtherCAT Master Function Module and MC Function Module* on page 2-19 for details.

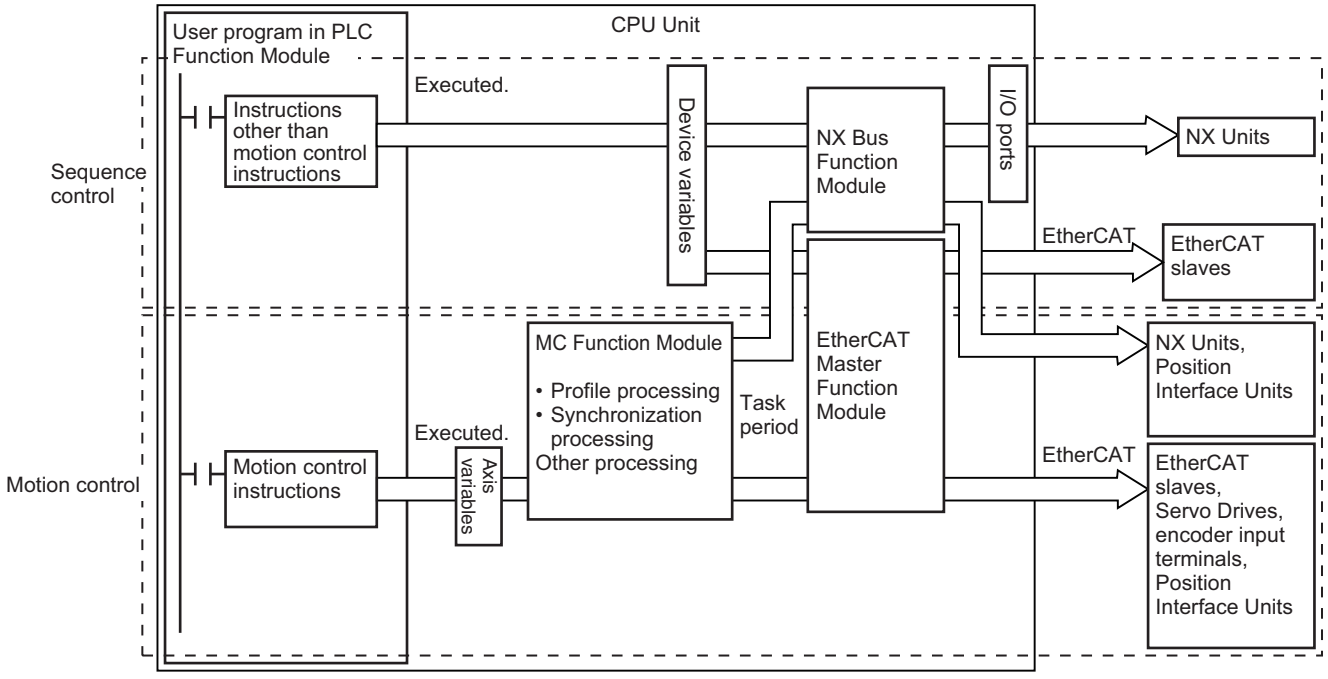
● NX701 CPU Unit



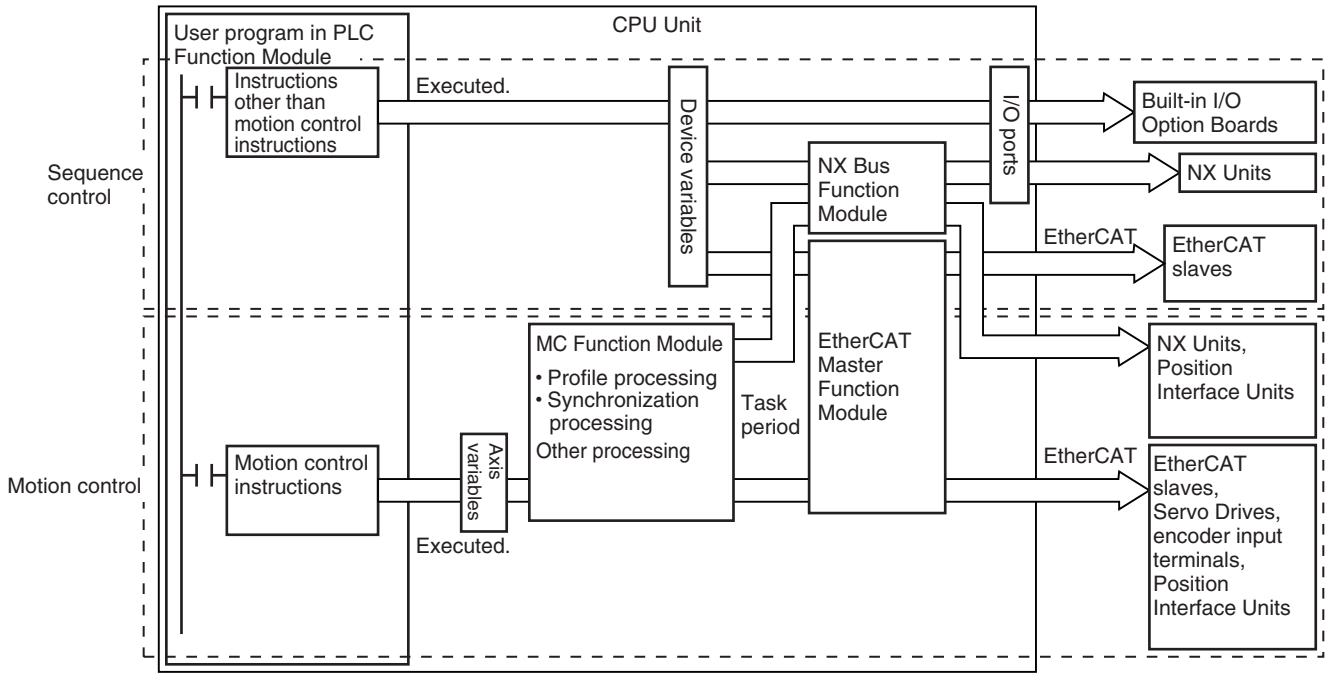
● NX502 CPU Unit



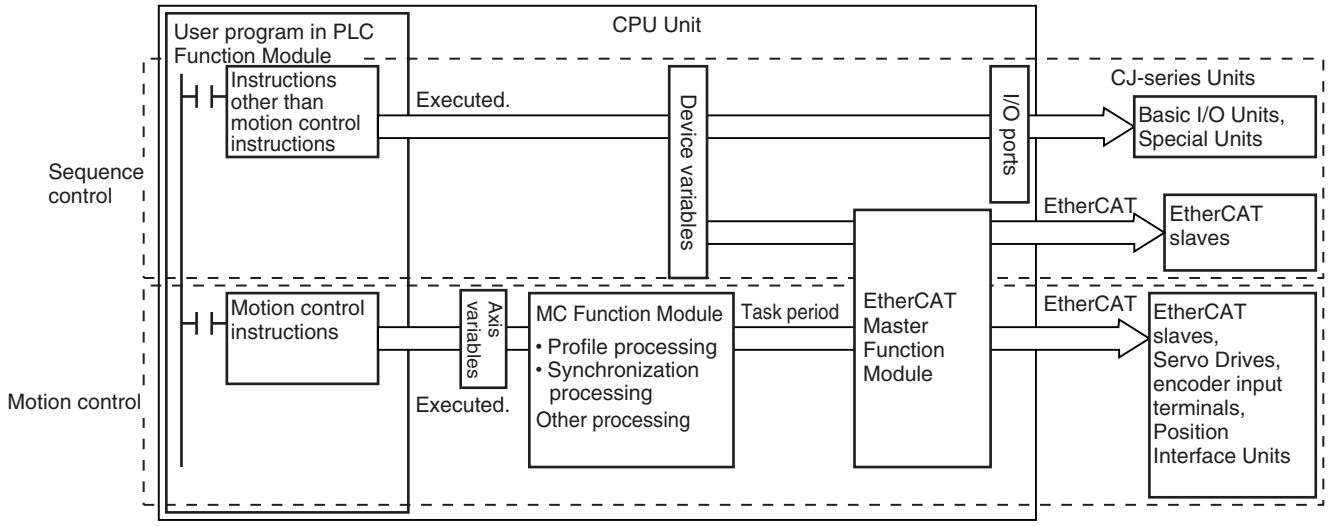
● NX102 CPU Unit



● NX1P2 CPU Unit



● NJ-series CPU Unit



6-2 Motion Control Instructions

Motion control instructions are used in the user program to execute motion controls for an NJ/NX-series CPU Unit. These instructions are defined as function blocks (FBs).
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 gives the specifications of the MC Function Module.

Refer to the *NJ/NX-series CPU Unit Software User's Manual (Cat. No. W501)* for basic information on the NJ/NX-series CPU Unit function blocks (FBs).
Refer to the *NX-series Position Interface Units User's Manual (Cat. No. W524)* for information on using the NX-series Position Interface Units.

Version Information

A CPU Unit with unit version 1.05 or later and Sysmac Studio version 1.06 or higher are required to use NX-series Position Interface Units.

6-2-1 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-axes coordinated control are defined along with basic procedures for executing instructions.
By using PLCopen® motion control function blocks, the user program can be more easily reused without hardware dependence.
Costs for training and support are also reduced.

Additional Information

PLCopen®

PLCopen® is an association that promotes IEC 61131-3. It has its headquarters in Europe and a world-wide membership.
IEC 61131-3 is an international standard for PLC programming.

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

6-2-2 Motion Control Instructions of the MC Function Module

There are three types of motion control 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-axes coordinated control

For a list of the instructions that you can use with the MC Function Module, refer to the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)*.

6-3 State Transitions

The states of axes and axes groups of the MC Function Module and state transitions caused by the execution of instructions are based on the technical specifications of function blocks for PLCopen® motion control.
This section provides an overall description of the MC Function Module, states, and state transitions.

6-3-1 Status of the Motion Control Function Module

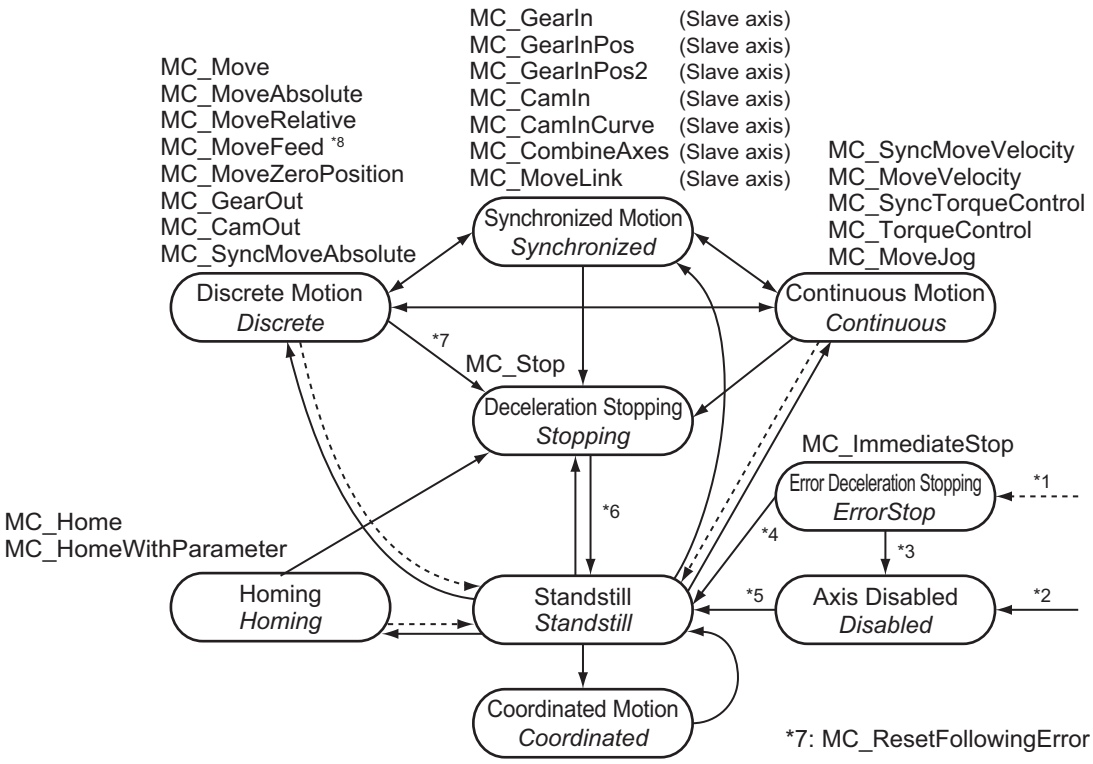
The overall states of the MC Function Module are described in the following table.

State name	Definition
MC Run Mode*1	Motion control instructions are enabled. The motion control instructions in the user program are interpreted and motion control is performed. You can set the *MC Run Mode* state regardless of the operating mode of the CPU Unit.
MC Test Mode*2	In this state, you can execute a test run from the Sysmac Studio.
Saving Cam Table File*3	This state exists while the system performs save or wait processing for a cam table file.
Generating Cam Table*4	This state exists while the system is generating the came table. *5

*1. This state can be monitored with the MC Common Variable *_MC_COM.Status.RunMode*.
*2. This state can be monitored with the MC Common Variable *_MC_COM.Status.TestMode*.
*3. This state can be monitored with the MC Common Variable *_MC_COM.Status.CamTableBusy*.
*4. This state can be monitored with the MC Common Variable *_MC_COM.Status.GenerateCamBusy*.
*5. When you turn OFF the power supply for the CPU Unit, make sure that generation of the cam table is not in progress. If you turn OFF the power supply for the CPU Unit while generation of the cam table is in progress, the cam table will not be generated correctly.

6-3-2 Axis States

The operation of an axis when motion control instructions are executed for it is shown in the following figure.
Motion control instructions are executed in sequence and axes enter one of the states listed in the following table.



*1. Transition into this state occurs when there is an axis error in any state except for *Coordinated Motion* state.
*2. Transition into this state occurs when there are no axis errors and the *Status* output to the MC_Power instruction is FALSE. (The Servo is OFF.)
*3. Transition into this state occurs if an error is reset with the MC_Reset or ResetMCError instruction when the Servo is OFF.
*4. Transition into this state occurs if an error is reset with the MC_Reset or ResetMCError instruction when the Servo is ON.
*5. Transition into this state occurs when the *Enable* input to the MC_Power instruction changes to TRUE and the *Status* (Servo ON) output from the MC_Power instruction changes to TRUE. (The Servo is ON.)
*6. Transition into this state occurs when the *Done* output from the MC_Stop instruction is TRUE and the *Execute* input to the MC_Stop instruction changes to FALSE.
*7. Transition into the *Deceleration Stopping* state occurs when the MC_ResetFollowingError instruction is executed.
*8. The *Continuous Motion* state exists from when **2: _mcVelocity** (velocity control) is set for the *MoveMode* input variable of the MC_MoveFeed instruction until a trigger input is detected.

State name		Definition
Servo OFF		In this state, the Servo is OFF for the axis. When this state is moved to, the buffered status for multi-execution of instructions is cleared.
	Axis Disabled	In this state, the Servo is OFF for the axis, the axis is stopped, and execution preparations are completed.
	Error Deceleration Stopping*1	In this state, the Servo is OFF for the axis, the axis is stopped, and an axis error has occurred.
Servo ON		In this state, the Servo is ON for the axis.
	Stopped	In this state, the Servo is ON for the axis and the axis is stopped.
	Discrete Motion	In this state, positioning is performed for the specified target position. This includes when waiting the in-position status and when the velocity is *0* because the override factor was set to 0 during a discrete motion.

State name	Definition
Continuous Motion	In this state, continuous motion control is executed with no specified target position. This state exists during velocity control or torque control. 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 Motion	In this state, the synchronized control is performed for the axis with synchronized control commands. This includes waiting for synchronization after changing to synchronized control instructions.
Deceleration Stopping	In this state, the axis is stopping due to a MC_Stop or MC_TouchProbe (Enable External Latch) instruction. This includes when Execute is TRUE after stopping for the MC_Stop instruction. In this state, it is not possible to execute axis operation commands. If an attempt is made to execute one, CommandAborted for the instruction changes to TRUE.
Error Deceleration Stopping*1	In this state, the Servo is ON for the axis and an axis error has occurred. This includes during execution of the MC_ImmediateStop (Immediate Stop) instruction and during a deceleration stop for an axis error. It is not possible to execute axis operation commands in this state. The instruction will enter the aborted (CommandAborted = TRUE) status if executed.
Homing	In this state, home is being searched for by the MC_Home or MC_HomeWithParameter instruction.
Coordinated Motion	In this state, the axes group was enabled by an instruction for an axes group command. In this state, the axis is in motion for an axes group state of *GroupMoving*, *GroupStopping*, or *GroupErrorStop*.

*1. The *Error Deceleration Stopping* state occurs both when the Servo is ON and when the Servo is OFF for the axis.

Note You can monitor the axis status in the member variables of the Axis Variables *_MC_AX[0-255].Status*.

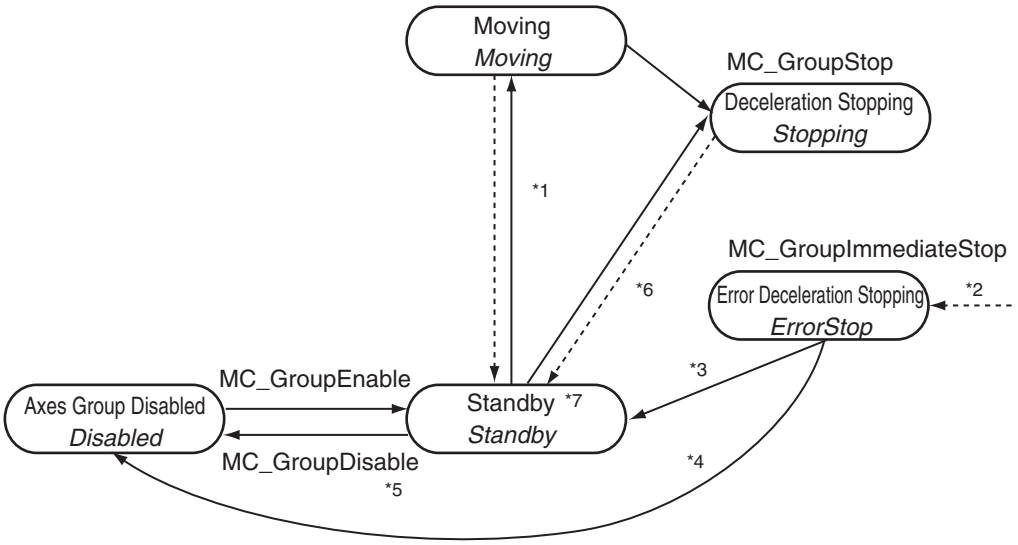
Version Information

For the NX701 CPU Unit, the variable names that start with _MC_AX[*] may apply to those with _MC1_AX[*] and _MC2_AX[*].

Refer to the *NX-series Position Interface Units User's Manual (Cat. No. W524)* for information on using the NX-series Position Interface Units.

6-3-3 Axes Group States

The following figure shows how an axes group operates when motion control instructions are executed.



*1. The axes group enters the Moving state whenever any axes group motion control instruction is executed.
*2. The axes group can move to the ErrorStop state from any other states. When an error occurs, the axes group enters this state even if it is disabled.
*3. The axes group moves to the Standby state if the MC_GroupReset or ResetMCError instruction is executed while it is enabled.
*4. The axes group moves to the Disabled state if the MC_GroupReset or ResetMCError instruction is executed while it is disabled.
*5. If the MC_GroupDisable instruction is executed while the axes group is in the ErrorStop state, the axes group will remain in the same state.
*6. The axes group moves from the Stopping state to the Standby state when the MC_GroupStop instruction has the output variable *Done* set to TRUE and the input variable *Execute* set to FALSE.
*7. The axes group is ready to execute if the following conditions are all met while it is in the Standby state.
The Servo is ON for all composition axes.
Execution of the MC_Stop instruction is not in progress for any composition axis.
Home is defined for all composition axes.

State name		Definition
Axes Group Disabled		The axes group is disabled in this state. When the axes group enters this state, the buffered status for multi-execution of instructions is cleared.
	Error Deceleration Stopping*1	In this state, an error occurred in an axes group that is disabled.
Axes Group Enabled		The axes group is enabled in this state.
	Standby	In this state, no instructions for axes group commands are executing. This is independent of the Servo ON/OFF status of the composition axes in the axes group.
	Moving	In this state, positioning is performed for the specified target position due to a motion instruction for an axes group command. This includes during the in-position check and when the velocity is 0 because the override factor was set to *0* while the group was in motion.
	Deceleration Stopping	In this state, the MC_GroupStop instruction is executing. This includes when *Execute* is TRUE after stopping for the MC_GroupStop instruction. In this state, it is not possible to execute a motion for an axes group command. If one is executed, *CommandAborted* for the instruction will change to TRUE.

State name		Definition
	Error Deceleration Stopping*1	In this state, an axes group error has occurred. This includes during execution of the MC_GroupImmediateStop (Axes Group Immediate Stop) instruction and during a deceleration stop for an axes group error. It is not possible to execute multi-axes coordinated control commands in this state. If an attempt is made to execute one of them, *CommandAborted* for the instruction will change to TRUE.

*1. The *Error Deceleration Stopping* state occurs both when the axes group is enabled and when it is disabled.

Note You can monitor the axes group status in the member variables of the Axes Group Variables *_MC_GRP[0-63].Status*, *_MC1_GRP[0-63].Status*, and *_MC2_GRP[0-63].Status*.

6-4 Execution and Status of Motion Control Instructions

Variables that represent the execution status of instructions and variables that are used to execute motion control instructions are defined in the MC Function Module.
There are two input variables that you use to execute motion control instruction functions *Execute* and *Enable*. The following output variables indicate the execution status of an instruction *Busy*, *Done*, *CommandAborted*, and *Error*.

6-4-1 Basic Rules for Execution of Instructions

The basic rules for the MC Function Module are listed in the following table.
You can find execution examples in *6-4-2 Execution Timing Charts* on page 6-14.
Refer to these examples as well.

Item	Rule
Exclusiveness of outputs	The following output variables are mutually exclusive, and only one of them can be TRUE at one time: *Busy*, *Done*, *Error*, and *CommandAborted*. Similarly, only one of the following output variables can be TRUE at the same time: *Active*, *Done*, *Error*, and *CommandAborted*. *Busy* and *Active* may be TRUE at the same time in some cases.
Output status	The output variables *Done*, *InGear* (Gear Ratio Achieved), *InSync*, *InVelocity* (Target Velocity Reached), and *CommandAborted* change to FALSE when the input variable *Execute* changes to FALSE. The actual execution of a motion control instruction is not stopped when *Execute* changes to FALSE. Even if *Execute* changes to FALSE before the instruction finishes execution, the corresponding output variable will be TRUE for at least one period if the status of the instruction instance changes. The output variable *Error* will not reset to FALSE and the output variable *ErrorID* (Error Code) will not reset to 0 until you execute one of the following instructions: MC_Reset, MC_GroupReset, or ResetMCError. *1 If the *Execute* variable of the same instruction instance changes to TRUE again (i.e., if the instruction is restarted) during the execution of a motion control instruction, the *CommandAborted* variable will not change to TRUE.
Input parameters	For motion control instructions that are started with the input variable *Execute*, the values of the input parameters when *Execute* changes to TRUE are used. For motion control instructions that start for the input variable *Enable*, the current values of the input parameters during each period when *Enable* is TRUE are used.
Omitting input parameters	The default value applies if you omit an input parameter for an instruction instance. *2
Position (Target Position) and *Distance (Travel Distance)*	The input variable *Position* is defined as a value in the coordinate system. The input variable *Distance* is the relative length, i.e., it is the difference between two positions.
Sign rules	The input variables *Acceleration*, *Deceleration*, and *Jerk* are non-negative values. *Position* (Target Position), *Distance* (Travel Distance), and *Velocity* (Target Velocity) can be positive, negative, or 0.

Item	Rule
Error processing	There are two output variables that represent an error when a problem occurs during the execution of an instruction instance. These outputs are defined as follows: • Error: The output variable *Error* changes to TRUE to indicate that an error occurred during the execution of the instruction instance. • ErrorID (Error Code): This is an error code that represents the cause of the error. The output variables *Done*, *InVelocity* (Target Velocity Reached), *InGear* (Gear Ratio Achieved), and *InSync* all represent normal completion or normal operation and therefore will never be TRUE when the output variable *Error* is TRUE. Types of errors: • Instruction instance errors (e.g., parameter out of range and illegal condition for state transition) • Axis errors (e.g., Following Error Limit Exceeded and Servo Drive errors) Some instruction instance errors may not cause an axis error but will cause the axis to stop.
Operation of output variable *Done*	The output variable *Done*, *InGear* (Gear Ratio Achieved), or *InSync* will change to TRUE when the instruction ends operation normally or when the commanded condition is reached. For movement instructions for which a target position is specified, the timing of when the output variable Done changes to TRUE depends on the setting of the **In-position Check Time** axis parameter. • If the In-position Check Time axis parameter is set to *any value but 0*, output variable Done changes to TRUE in the next period after the period in which positioning is completed. • If the In-position Check Time axis parameter is set to *0*, output variable Done changes to TRUE in the next period after the period in which pulse distribution is completed. When working with multiple instructions that operate on the same axis, the output variable *Done* from the first instruction will not change to TRUE if another operation instruction takes over before the axis operation for the first instruction reaches the target position.
Operation of output variable *CommandAborted*	The output variable *CommandAborted* will change to TRUE when another operation instruction interrupts the commanded operation. For the MC Function Module, this variable will change to TRUE when a motion control instruction is executed and the target axis or axes group causes an error or is decelerating to a stop. All other output variables change to FALSE when *CommandAborted* changes to TRUE.
Input variables outside of valid range	The instruction instance will output an error when it is executed with an input variable that is outside of the valid range.
Operation of output variable *Busy*	The output variable *Busy* is TRUE when the instruction instance is executing. *Busy* will change to TRUE when the input variable *Execute* changes to TRUE, and will change to FALSE when the output variable *Done*, *CommandAborted*, or *Error* changes to TRUE. It is impossible to know when the above output variables will change. Write your programs so that the instruction instance executes every period[*3] while *Busy* is TRUE so that you can monitor for changes in the output variables. For a single axis or single axes group, the *Busy* variable of more than one instruction instance can be TRUE at the same time. However, the output variable *Active* of only one instruction instance can be TRUE at one time. However, the MC_Phasing (Shift Master Axis Phase) instruction is an exception to this rule.
Output variable *Active*	The output variable *Active* changes to TRUE when the instruction instance obtains permission to control the applicable axis. The output variable *Active* may change slower than the *Busy* variable. [*4]

*1. This behavior is different from the PLCopen® specifications. Under the PLCopen® specifications, *Error* changes to FALSE and *ErrorID* changes to *0* 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 if their *Enable* variable is TRUE when an error is reset.

*2. When you program the instruction in a ladder diagram, insert an input between the input variable *Execute* or *Enable* 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.
Set the initial value for or omit any input variable that is reserved.
*3. If the condition expressions or set values for ST Structure instructions do not match, the instructions in that statement are not executed. For details, refer to the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)*.
*4. Refer to the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)* for an output variable timing chart.

Precautions for Correct Use

- Confirm that EtherCAT process data communications are active and normal before you execute motion control instructions.
  Refer to *10-2-1 Monitoring EtherCAT Communications and Turning ON Servos* on page 10-3 for details.
- Write the user program so that *Execute* is FALSE during the first period in which the instruction is executed.

6-4-2 Execution Timing Charts

The motion control instructions in the MC Function Module are function blocks that are unconditionally executed.
This section calls instructions that are executed according to the *Execute* input variable "execute-type instructions" and instructions that are executed according to the *Enable* input variable "enable-type instructions."

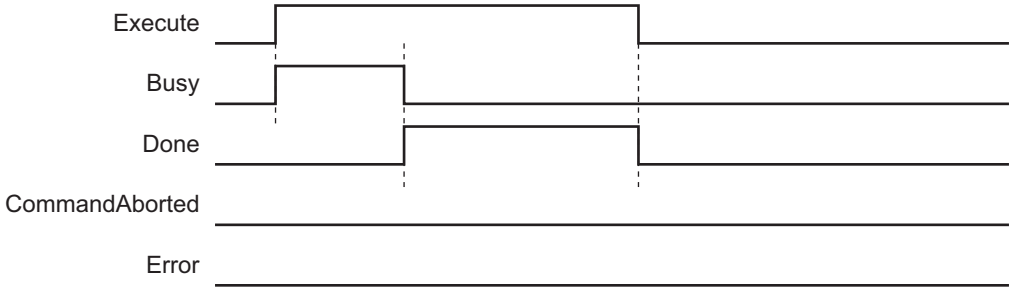
Execution condition	Description
Execute-type instructions	These motion control instructions are executed when the input variable *Execute* to the instruction changes to TRUE. These instructions will continue execution until one of the following status occurs. • The specified operation is completed. • Another motion control instruction is executed and interrupts operation. • The instruction is restarted when Execute changes from FALSE to TRUE again. Values for the other input variables are input when *Execute* changes to TRUE.
Enable-type instructions	These motion control instructions are executed every period while the input variable *Enable* to the motion control instruction is TRUE. As long as *Enable* is TRUE, the other input variables are also input every period. However, MC_MoveJog input variables *Velocity*, *Acceleration*, and *Deceleration* are an exception to this rule. The values when *PositiveEnable* or *NegativeEnable* changes to TRUE are used for these input variables.

Precautions for Correct Use

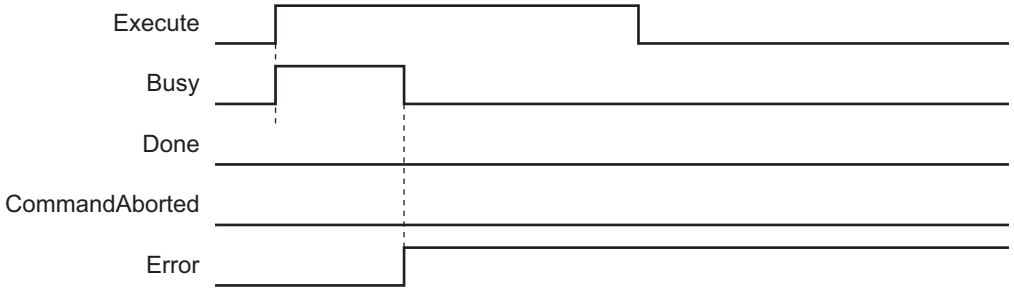
The timing in the timing charts that are given in this manual may not necessarily be the same as the timing displayed for data traces on the Sysmac Studio.
Refer to the *NJ/NX-series CPU Unit Software User's Manual (Cat. No. W501)* for details on data tracing.

Timing Charts for Execute-type Instructions

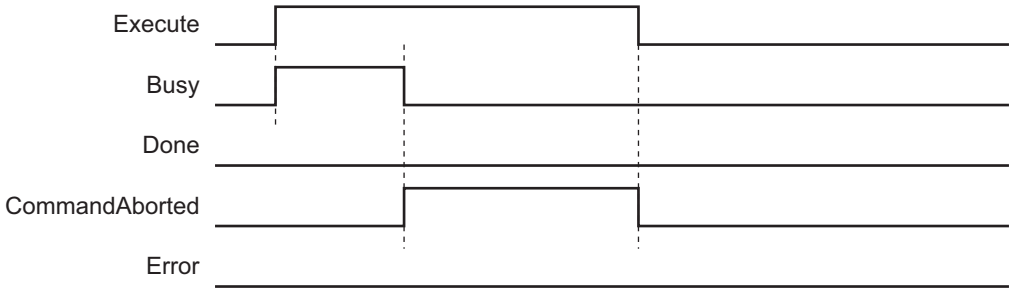
- The following timing chart shows the operation of the instruction is completed while the input variable *Execute* is TRUE.
  In this example, no error occurs after the completion of the instruction, and then *Execute* is changed to FALSE.



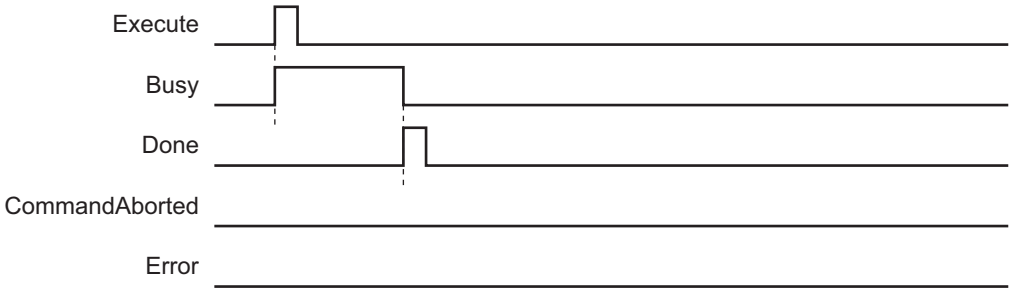
- The following timing chart shows an example where an error occurs while the input variable *Execute* is TRUE.
  The output variable *Error* remains TRUE even after *Execute* changes to FALSE.



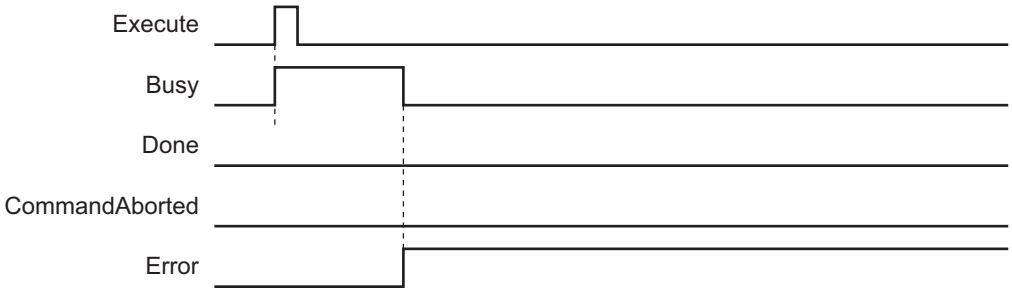
- The following timing chart shows an example where the instruction is interrupted during execution while the input variable *Execute* is true.



- The following timing chart shows an example where the input variable *Execute* is TRUE for only one period and an error does not occur for the instruction.
  The output variable *Done* changes to TRUE for only one period after the instruction operation is completed.

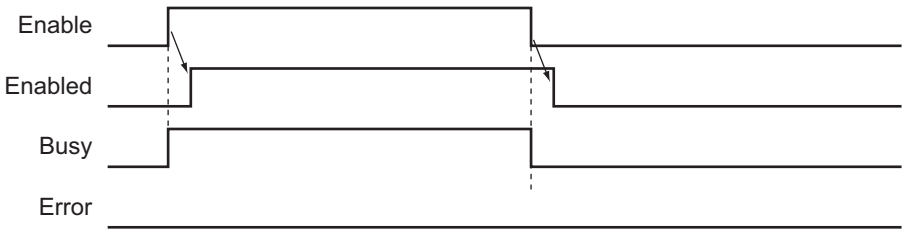


- The following timing chart shows an example where the input variable *Execute* is TRUE for only one period and an error occurs for the instruction.
  The output variable *Error* remains TRUE.

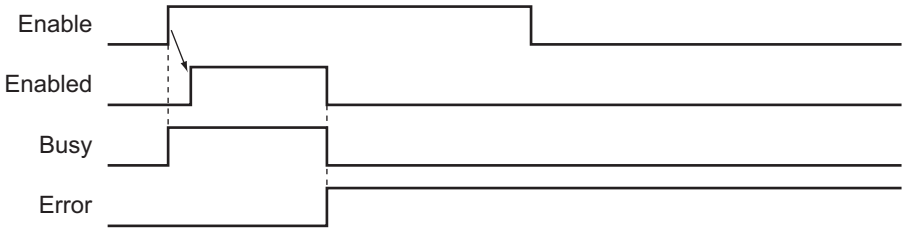


Timing Charts for Enable-type Instructions

- The following timing chart shows an example where the input variable *Enable* changes to TRUE and an error does not occur for the instruction.



- The following timing chart shows an example where the input variable *Enable* changes to TRUE and an error occurs for the instruction.



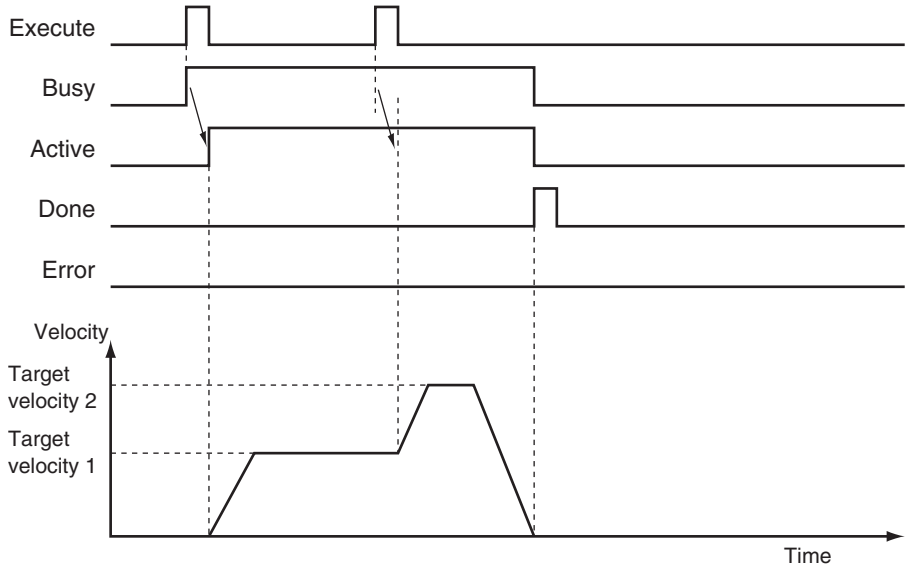
Additional Information

Enable and *Enabled* change at the same time for instructions such as MC_ZoneSwitch (Zone Monitor) and MC_AxesObserve (Monitor Axis Following Error). For details on the timing of individual instructions, refer to the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)*.

6-4-3 Timing Chart for Re-execution of Motion Control Instructions

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

The following timing chart shows an example where the velocity is changed for MC_MoveAbsolute (Absolute Positioning) instruction.

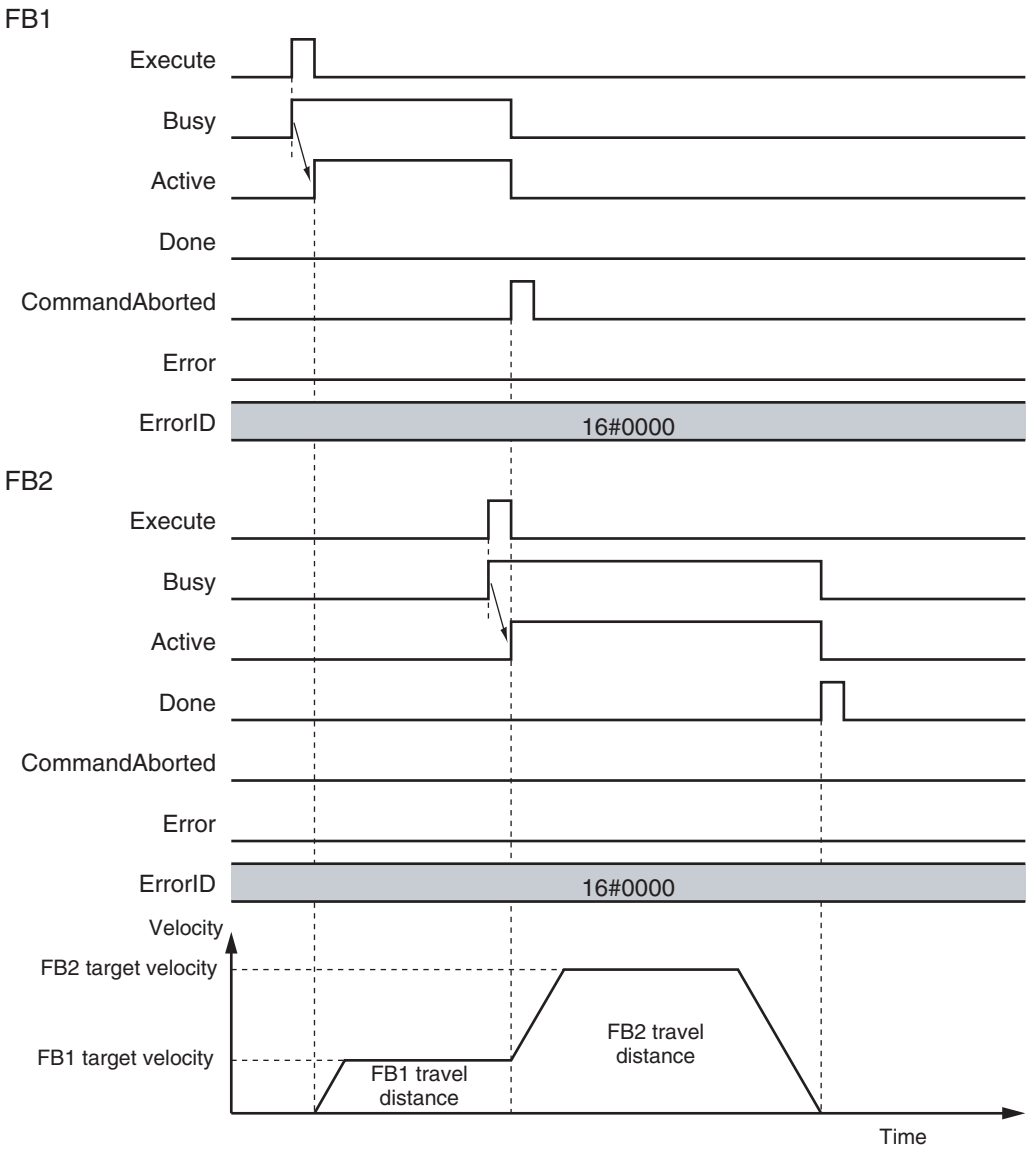


For details on re-executing instructions for the MC Function Module, refer to *9-5-6 Re-executing Motion Control Instructions* on page 9-48 and *9-7-4 Re-executing Motion Control Instructions for Multi-axes Coordinated Control* on page 9-71.

6-4-4 Timing Chart for Multi-execution of Motion Control Instructions

Another instance can be executed for an axis during axis motion.
Set the input variable *BufferMode* to specify when to start operation.

The following figure shows an example in which *BufferMode* (Buffer Mode Selection) is set to aborting when MC_MoveAbsolute (Absolute Positioning) instructions are executed with multi-execution of instructions.
“FB1” and “FB2” in the following figure are the instance names of the instructions.



For details on multi-execution of instructions for the MC Function Module, refer to *9-5-7 Multi-execution of Motion Control Instructions (Buffer Mode)* on page 9-53 and *9-7-5 Multi-execution of Motion Control Instructions (Buffer Mode) for Multi-axes Coordinated Control* on page 9-72.

6-5 Positions

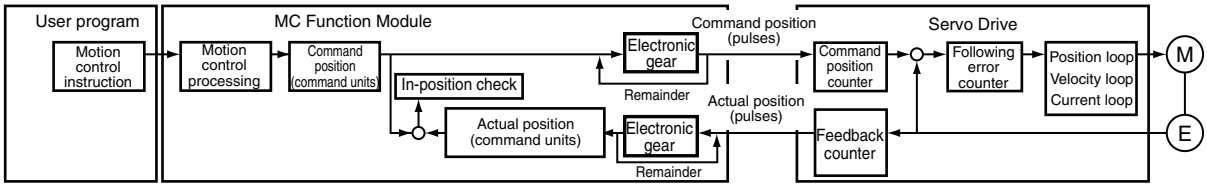
This section describes the positions that are used in motion control programming.

6-5-1 Types of Positions

The MC Function Module uses the following two types of positions.

Type of position	Definition
Command position	This is the position that the MC Function Module outputs to control an axis.
Actual position	The actual position as input from the Servo Drive or encoder input.

The following figure shows the relationship between the command position and the actual position for an EtherCAT slave Servo Drive.



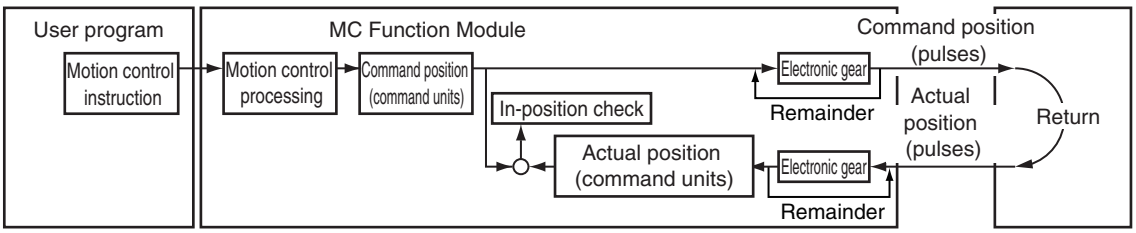
The command position and actual position share the following items.

Item	Command position	Actual position
Count Mode	You can set Linear Mode or Rotary Mode.	The same Count Mode is used as for the command position.
Position increment	You can set one of the following: mm, μm, nm, inch, degree, or pulse.	The unit is the same as the unit of the command position.
Software limits	You can set the range of operation of the software.	The range is the same as the range for the command position.
Changing the current position	You can change the actual position to any desired position.	This value will be set to the same position as the command position. *1
Defining home	Home is either defined or undefined.	The status of home is the same as the command position.

*1. If there is any following error before the change, the following error value is maintained in the actual position.

Additional Information

- For a virtual servo axis, the command current value is converted to pulses and then that value is converted to the unit of display. The resulting value is used as the actual current value.



- Refer to the *NX-series Position Interface Units User's Manual (Cat. No. W524)* for information on the NX-series Position Interface Units.

6-5-2 Valid Positions for Each Axis Type

The following table lists the valid positions for each axis type.

Axis type	Types of positions	
	Command position	Actual position
Servo axis	Applicable	Applicable
Virtual servo axis	Applicable	Applicable*1
Encoder axis	Cannot be used.	Applicable
Virtual encoder axis	Cannot be used.	Applicable*2

*1. For a virtual servo axis, the actual position is the same as the command position.
However, there is sometimes calculation error because processing is performed with long reals in the MC Function Module.

*2. This is used when there is no actual encoder.

6-6 System-defined Variables for Motion Control

This section describes the variables of the MC Function Module.

6-6-1 Overview of System-defined Variables for Motion Control

The NJ/NX-series CPU Unit is compliant with the IEC 61131-3 standard.
Parameter settings, status information, and other data are handled as variables in the user program in the NJ/NX-series CPU Unit.
Of these, system-defined variables that belong to the MC Function Module are called system-defined variables for motion control.

Types of System-defined Variables for Motion Control

The following table lists all of the types of system-defined variables for motion control.

Level 1	Level 2	Level 3	Description
System-defined variables	System-defined variables for motion control	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 Variables	You can monitor axes group status and the settings of part of the axes group parameters.

● MC Common Variable

You can monitor the overall status of the MC Function Module with the MC Common Variable. The variable name is *_MC_COM*.

● Axis Variables

Use these variables to handle EtherCAT slaves, Servo Drives, encoder input terminals, NX-series Position Interface Units, virtual Servo Drives, and virtual encoder input terminals.
You can use either the system-defined variables or the variables that are set on the Sysmac Studio to specify the Axis Variables in the user program.
You can change any of the Axis Variables that you create on the Sysmac Studio.

a. Axis Variables in the system-defined variables
   - _MC_AX[0] to _MC_AX[255]
   - _MC1_AX[0] to _MC1_AX[255]
   - _MC2_AX[0] to _MC2_AX[255]

b. Default Axes Group Variables when axes groups are created on the Sysmac Studio
   - MC_Axis000 to MC_Axis255 (default)

● Axes Group Variables

Use these variables to handle multiple axes as a single group.
You can use either the system-defined variables or the variables that are set on the Sysmac Studio to specify the Axes Group Variables in the user program.

You can change any of the Axes Group Variables that you create on the Sysmac Studio.

a. Axes Group Variables in the system-defined variables
   - _MC_GRP[0] to _MC_GRP[63]
   - _MC1_GRP[0] to _MC1_GRP[63]
   - _MC2_GRP[0] to _MC2_GRP[63]

b. Default Axes Group Variables when axes groups are created on the Sysmac Studio
   - MC_Group000 to MC_Group063 (default)

Refer to the *NJ/NX-series CPU Unit Software User's Manual (Cat. No. W501)* for details on the variables that are used by an NJ/NX-series CPU Unit.

Data Types Used for System-defined Variables for Motion Control

System-defined variables for motion control use both basic data types and derivative data types.

● Basic Data Types

Category	Data type	Size	Range of values	Notation
Boolean	BOOL	2*1	TRUE or FALSE	TRUE or FALSE
Integer	UINT	2	0 to +65,535	Binary notation: "2#" is added to the front of the number *2 Octal notation: "8#" is added to the front of the number *3 Decimal notation: "10#" is added to the front of the number *4 Hexadecimal notation: "16#" is added to the front of the number *5 If you do not add any notation to the beginning of a number, that number is treated as a decimal number.
	UDINT	4	0 to +4,294,967,295	
Floating-point numbers	LREAL	8	-1.79769313486231e+308 to -2.22507385850721e-308, 0, 2.22507385850721e−308 to 1.79769313486231e+308, positive infinity, or negative infinity	Written as (sign) + integer_part + (decimal_point) + (decimal_part) + (exponent). *6 You can omit items in parentheses.

*1. BOOL data is only 1 bit in size but it takes up 2 bytes of memory.
*2. Binary notation examples: 2#1111_1111, 2#1110_0000
*3. Octal notation examples: 8#377, 8#340
*4. Decimal notation examples: −12, 0, 123_456, +986, 10#1234
*5. Hexadecimal notation examples: 16#FF, 16#ff, 16#E0, 16#e0
*6. Examples: 2, -12.0, 0.0, 0.4560, 3.14159_26, -1.34E-12, -1.34e-12, 1.0E+6, 1.0e+6, 1.234E6, 1.234e6

● Derivative Data Types

Type	Description
Enumerated data type	This data type uses one item from a prepared name list as its value. Variables with this data type start with "_e."

Type	Description
Structure data type	This data type consists of multiple data types placed together into a single layered structure. Variables with this data type start with "_s."

Refer to the *NJ/NX-series CPU Unit Software User's Manual (Cat. No. W501)* for details on the other data types that are used by an NJ/NX-series CPU Unit.

Attributes of System-defined Variables for Motion Control

The attributes that are shown in the following table are the same for all system-defined variables for motion control.

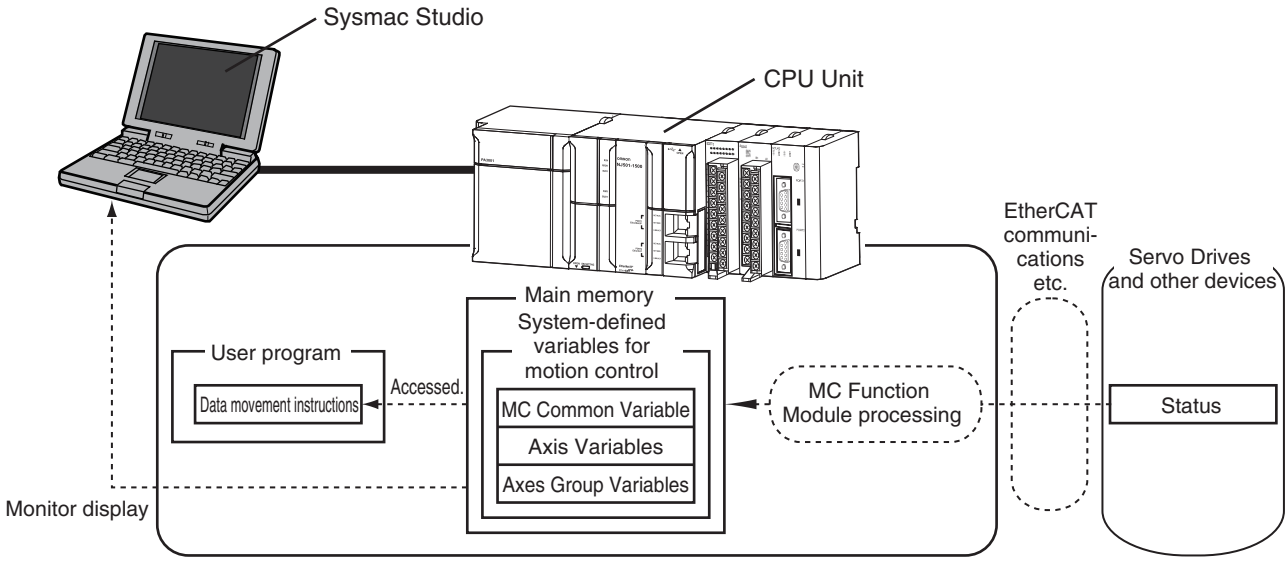
Attribute	Attribute of system-defined variables for motion control
Global/Local	Global variable
R/W access	Read only
Retain	Non-retain
Network Publish	Publish*1

*1. Variables are published on the network with the variable names of the system-defined variables. The variable names that are created when the axes or axes groups are created on the Sysmac Studio are not published to the network.

6-6-2 System for System-defined Variables for Motion Control

System-defined variables for motion control consist of information representing the status of the MC Function Module, status information for slave devices connected via EtherCAT communications and NX Units mounted on the CPU Unit, and the portion of the MC parameter settings used to perform motion control.

You can access system-defined variables for motion control as variables in the user program and monitor them from the Sysmac Studio.



Update Timing of System-defined Variables for Motion Control

The update timings of the system-defined variables for motion control are different in the NX701 CPU Unit, NX502 CPU Unit, NX102 CPU Unit, NX1P2 CPU Unit, and NJ-series CPU Unit as follows.

● NX701 CPU Unit

When the primary periodic task and priority-5 periodic task are assigned to axes, the motion control variables are updated as follows:

- Only the task area that is assigned to the primary periodic task and the basic setting area of the priority-5 periodic task are updated every primary task period.
- The other area of the priority-5 periodic task is not updated from the default settings and will be updated in the task period of priority-5 periodic task.

Variable area to update	Primary periodic task	Priority-5 periodic task
_MC_AX[0-255]	○	
_MC1_AX[0-255]	○	
_MC2_AX[0-255]		○
_MC_GRP[0-63]	○	
_MC1_GRP[0-63]	○	
_MC2_GRP[0-63]		○

● NX502 CPU Unit, NX102 CPU Unit, NX1P2 CPU Unit, and NJ-series CPU Unit

The system-defined variables for motion control are updated every primary task period.

6-6-3 Tables of System-defined Variables for Motion Control

This section provides tables that describe the system-defined variables for motion control.

MC Common Variable

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

Variable 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 File Save Busy	TRUE while the Cam Table is being saved or on standby.
GenerateCamBusy*1	BOOL	Generating Cam Table	TRUE while the cam table is being generated.

Variable name	Data type	Meaning	Function
PFaultLvl	_sMC_REF_EVENT	MC Common Partial Fault	
Active	BOOL	MC Common Partial Fault Occurrence	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 Occurrence	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 Occurrence	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.

*1. A CPU Unit with unit version 1.08 or later and Sysmac Studio version 1.09 or higher are required.

Axis Variables

_MC_AX[0-255], _MC1_AX[0-255], and _MC2_AX[0-255] are the Axis Variables in the system-defined variables. The data type is _sAXIS_REF, which is a structure variable.

This section describes the configuration of the Axis Variables and provides details on the members using _MC_AX[0-255] as an example.

The same information applies to _MC1_AX[0-255] and _MC2_AX[0-255].

Variable name	Data type	Meaning	Function
_MC_AX[0-255]	_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 indicates the same status as when _MC_AX[*].Status.Standstill is TRUE (Standstill).
Disabled	BOOL	Axis Disabled	TRUE while the Servo is OFF for the axis. The following axis states are mutually exclusive. Only one can be TRUE at the same time. Disabled, Standstill, Discrete, Continuous, Synchronized, Homing, Stopping, ErrorStop, or Coordinated

Variable name	Data type	Meaning	Function
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. 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. This state exists during velocity control or torque control. 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. This includes waiting for synchronization after changing to synchronized control instructions.
Homing	BOOL	Homing	TRUE when homing for the MC_Home or MC_HomeWithParameter 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_ImmdediateStop instruction or a minor fault (_MC_AX[*].MFaultLvl.Active) is TRUE (Axis Minor Fault Occurrence). Axis motion instructions are not executed in this state. (CommandAborted is TRUE.)
Coordinated	BOOL	Coordinated Motion	TRUE when an axes group is enabled by a multi-axes coordinated control instruction.
Details	_sAXIS_REF_DET	Axis Control Status	Gives the control status of the command.
Idle	BOOL	Idle	TRUE when processing is not currently performed for the command value, except when waiting for in-position state. *1 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 performed when positioning for the in-position check.

Variable name	Data type	Meaning	Function
Homed	BOOL	Home Defined	TRUE when home is defined. *2 FALSE: Home not defined TRUE: Home is defined
InHome	BOOL	In Home Position	TRUE when the axis is in the range for home. It gives an AND of the following conditions. • Home defined • The actual current position is in the **zero position range** with home as the center. TRUE also when the zero position is passed by while the axis is moving in command status.
VelLimit*3	BOOL	Command Velocity Saturation	TRUE while the command velocity is limited to the maximum velocity during synchronized control.
Dir	_sAXIS_REF_DIR	Command Direction	Gives the command travel direction.
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	Gives the status of the Servo Drive.
ServoOn	BOOL	Servo ON	TRUE when the Servomotor is powered.
Ready	BOOL	Servo Ready	TRUE when the Servo is ready.*4
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*6
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. *7
CSP	BOOL	Cyclic Synchronous Position (CSP) Control Mode	TRUE when the Servo is ON at the Servo Drive and the current mode is CSP Mode. *8
CSV	BOOL	Cyclic Synchronous Velocity (CSV) Control Mode	TRUE when the Servo is ON at the Servo Drive and the current mode is CSV Mode. *8

Variable name	Data type	Meaning	Function
CST	BOOL	Cyclic Synchronous Torque (CST) Control Mode	TRUE when the Servo is ON at the Servo Drive and the current mode is CST Mode. *8
Cmd	_sAXIS_REF_CMD_DATA	Axis Command Values	
Pos	LREAL	Command Current Position	Contains the current value of the command position. (Unit: command units) When the Servo is OFF and the mode is not position control mode, this variable contains the actual current position. *9
Vel	LREAL	Command Current Velocity	Contains the current value of the command velocity. (Unit: command units/s) A plus sign is added during travel in the positive direction, and a minus sign during travel 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) 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. The value is 0 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) A plus sign is added when the absolute value of acceleration/deceleration is increasing, and a minus sign is added when it is decreasing. The value is 0 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: %) A plus sign is added during travel in the positive direction, and a minus sign during travel in the negative direction. Contains the same value as the actual current torque except in torque control mode. *10

Variable name	Data type	Meaning	Function
Act	_sAXIS_REF_ACT_DATA	Axis Current Value	
Pos	LREAL	Actual Current Position	Contains the actual current position. (Unit: command units)*9
Vel	LREAL	Actual Current Velocity	Contains the actual current velocity. (Unit: command units/s) A plus sign is added during travel in the positive direction, and a minus sign during travel in the negative direction.
Trq	LREAL	Actual Current Torque	Contains the current value of the actual torque. (Unit: %) A plus sign is added during travel in the positive direction, and a minus sign during travel in the negative direction.
TimeStamp*11	ULINT	Time Stamp	Contains the time when the current position of the axis was updated. This variable is valid for an axis for which time stamping is operating. (Unit: ns)
MFaultLvl	_sMC_REF_EVENT	Axis Minor Fault	
Active	BOOL	Axis Minor Fault Occurrence	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.
Obsr	_sMC_REF_EVENT	Axis Observation	
Active	BOOL	Axis Observation Occurrence	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	Gives the settings of the Axis Basic Settings parameters.
AxNo	UINT	Axis Number	Contains the logical number of the axis.
AxEnable	_eMC_AXIS_USE	Axis Use	Shows if the axis is enabled or disabled. 0: _mcNoneAxis (Undefined Axis) 1: _mcUnusedAxis (Unused Axis) 2: _mcUsedAxis (Used Axis)
AxType	_eMC_AXIS_TYPE	Axis Type	Contains the axis type. I/O wiring is not required for virtual axes. 0: _mcServo (Servo Axis) 1: _mcEncdr (Encoder Axis) 2: _mcVirServo (Virtual Servo Axis) 3: _mcVirEncdr (Virtual Encoder Axis)
NodeAddress	UINT	Node Address	Contains the EtherCAT slave address. *12 A value of 16#FFFF indicates that there is no address.
ExecID*13	UINT	Execution ID	Contains the task execution ID. 0: Not assigned to task (undefined axis) 1: Assigned to primary periodic task 2: Assigned to priority-5 periodic task

Variable name	Data type	Meaning	Function
Scale	_sAXIS_REF_SCALE	Unit Conversions	Gives settings of the electronic gear ratio.
Num	UDINT	Command Pulse Count Per Motor Rotation	Contains the number of pulses per motor rotation for command positions. 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. *14
Units	_eMC_UNITS	Unit of Display	Contains the display unit for command positions. 0: _mcPls (pulse) 1: _mcMm (mm) 2: _mcUm (μm) 3: _mcNm (nm) 4: _mcDeg (degree) 5: _mcInch (inch)
CountMode*13	_eMC_COUNT_MODE	Count Mode	Contains the count mode. 0: _mcCountModeLinear (linear mode) 1: _mcCountModeRotary (rotary mode)
MaxPos*13	LREAL	Maximum Current Position	Contains the maximum value of the current position indication. *15
MinPos*13	LREAL	Minimum Current Position	Contains the minimum value of the current position indication. *16

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

*2. Even if the variable is TRUE, the home must be defined again in the following cases.
When you make a change in the position count settings or the unit conversion settings.
If an error or erroneous operation occurs on the Servo Drive, which leads to loss of absolute position data. Examples of errors and erroneous operations include breaks of encoder cables and clear of absolute encoder data.

*3. Use VelLimit only for a slave axis that is currently in synchronized control.

*4. This variable is TRUE when the PDS state of the Servo Drive is either *Ready to switch on*, *Switched on* or *Operation enabled* and the main circuit power supply (voltage enabled) is ON. However, if **Main circuit power supply OFF detection** is set to **Do not detect**, ON/OFF status of the main circuit power supply is ignored.
For details on the PDS state and the Main circuit power supply OFF detection, refer to *A-5 PDS State Transition* on page A-28.

*5. The **Detailed Settings** Area on the **Axis Basic Settings** Display of the Sysmac Studio gives the signal that is set for **encoder Z-phase detection digital input**. You may not be able to map this signal to a PDO for servo drives from other manufacturers. Refer to the manual for the connected servo drive for details.

*6. You cannot map this signal to a PDO for an OMRON G5-series Linear Motor Type Servo Drive with built-in EtherCAT communications.

*7. This variable gives the status of bit 11 (internal limit enabled) in the Status Word (6041 hex) that is mapped to a PDO. The conditions for this variable changing to TRUE depend on the specifications of the Servo Drive. Refer to the manual for the connected servo drive for details.
For an OMRON 1S-series Servo Drive or G5-series Servo Drive, this variable gives an OR of the following four: torque limits, velocity limits, drive prohibit inputs, and software limits.

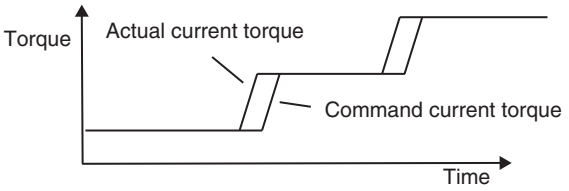
*8. This variable gives the value of the Modes of Operation Display (6061 hex) that is mapped to a PDO. The conditions for CSP, CSV, and CST changing to TRUE depend on the specifications of the Servo Drive. Refer to the manual for the connected servo drive for details.
If the Modes of Operation Display (6061 hex) is not mapped to a PDO, the values of these variables depend on the unit version of the CPU Unit as follows:

For a CPU Unit with unit version 1.10 or later, they are TRUE when the status of the Statusword (6041 hex) that was mapped to a PDO is Operation Enabled.
For a CPU Unit with unit version 1.09 or earlier, they are always FALSE.

*9. If the process data communications between the CPU Unit and an EtherCAT slave or NX Unit that is assigned to an axis changed to a non-established state, the variables contain different values as follows depending on the unit version of the CPU Unit.
For a CPU Unit with unit version 1.10 or later, the actual current position and the command current position axis variables will contain the actual current position output that is just before process data communications change to a non-established state.
For a CPU Unit with unit version 1.09 or earlier, the actual current position and the command current position axis variables will contain 0 or the lower limit. The lower limit is contained when the Count Mode is set to Rotary Mode and 0 is not included in the range of position.

*10. If you display a data trace that compares the command current torque and the actual current torque in any mode other than Torque Control Mode, the command current torque will change after one task period. This is caused by the timing of processing the data trace. Refer to *Data Tracing* in the *NJ/NX-series CPU Unit Software User's Manual (Cat. No. W501)* for information on the timing of processing data tracing.

Data Trace Display Example



Changes in the display of the command current torque are delayed in respect to the actual current torque.

*11. A CPU Unit with unit version 1.06 or later and Sysmac Studio version 1.07 or higher are required to use this variable.

*12. For an NX-series Position Interface Unit, this is the node address of the EtherCAT Coupler Unit under which the Position Interface Unit is mounted.

*13. A CPU Unit with unit version 1.10 or later and Sysmac Studio version 1.12 or higher are required to use this variable.

*14. The parameter is disabled if you set to **use** a reducer in the unit conversion settings, which is the function added for CPU Units with unit version 1.11 or later.
To confirm alternatively enabled parameters, i.e. **Work Travel Distance Per Rotation**, **Work Gear Ratio**, and **Motor Gear Ratio**, use the MC_ReadAxisParameter (Read Axis Parameters) instruction.

*15. If the Count Mode is set to Linear Mode, the position just before an overflow is given. In Rotary Mode, the modulo maximum position is given.

*16. If the Count Mode is set to Linear Mode, the position just before an underflow is given. In Rotary Mode, the modulo minimum position is given.

● Relationship between Axis Variables and Axis Types

Axis Variables are enabled or disabled according to the axis type. Disabled members are FALSE or 0.

In the descriptions, _MC_AX[0-255] is used as an example. The same information applies to _MC1_AX[0-255] and _MC2_AX[0-255].
Also, the following table shows which members are enabled for each axis type.

○: Enabled, ---: Disabled

Variable name	Data type	Meaning	Servo axis	Virtual servo axis	Encoder axis	Virtual encoder axis
_MC_AX[0-255]	_sAXIS_REF	Axis Variable				
Status	_sAXIS_REF_STA	Axis Status				
Ready	BOOL	Axis Ready-to-execute	○	○	---	---

Variable name	Data type	Meaning	Servo axis	Virtual servo axis	Encoder axis	Virtual encoder axis
Disabled	BOOL	Axis Disabled	○	○	○	○
Standstill	BOOL	Standstill	○	○	---	---
Discrete	BOOL	Discrete Motion	○	○	---	---
Continuous	BOOL	Continuous Motion	○	○	---	---
Synchronized	BOOL	Synchronized Motion	○	○	---	---
Homing	BOOL	Homing	○	○	---	---
Stopping	BOOL	Deceleration Stopping	○	○	---	---
ErrorStop	BOOL	Error Deceleration Stopping	○	○	---	---
Coordinated	BOOL	Coordinated Motion	○	○	---	---
Details	_sAXIS_REF_DET	Axis Control Status				
Idle	BOOL	Idle	○	○	○	○
InPosWaiting	BOOL	In-position Waiting	○	○	---	---
Homed	BOOL	Home Defined	○	○	---	---
InHome	BOOL	In Home Position	○	○	---	---
VelLimit	BOOL	Command Velocity Saturation	○	○	---	---
Dir	_sAXIS_REF_DIR	Command Direction				
Posi	BOOL	Positive Direction	○	○	---	---
Nega	BOOL	Negative Direction	○	○	---	---
DrvStatus	_sAXIS_REF_STA_DRV	Servo Drive Status				
ServoOn	BOOL	Servo ON	○	---	---	---
Ready	BOOL	Servo Ready	○	---	---	---
MainPower	BOOL	Main Power	○	---	---	---
P_OT	BOOL	Positive Limit Input	○	---	---	---
N_OT	BOOL	Negative Limit Input	○	---	---	---
HomeSw	BOOL	Home Proximity Input	○	---	---	---
Home	BOOL	Home Input	○	---	---	---
ImdStop	BOOL	Immediate Stop Input	○	---	---	---
Latch1	BOOL	External Latch Input 1	○	---	---	---
Latch2	BOOL	External Latch Input 2	○	---	---	---
DrvAlarm	BOOL	Drive Error Input	○	---	---	---
DrvWarning	BOOL	Drive Warning Input	○	---	---	---
ILA	BOOL	Drive Internal Limiting	○	---	---	---
CSP	BOOL	Cyclic Synchronous Position (CSP) Control Mode	○	---	---	---
CSV	BOOL	Cyclic Synchronous Velocity (CSV) Control Mode	○	---	---	---
CST	BOOL	Cyclic Synchronous Torque (CST) Control Mode	○	---	---	---
Cmd	_sAXIS_REF_CMD_DATA	Axis Command Values				
Pos	LREAL	Command Current Position	○	○	---	---

Variable name	Data type	Meaning	Servo axis	Virtual servo axis	Encoder axis	Virtual encoder axis
Vel	LREAL	Command Current Velocity	○	○	---	---
AccDec	LREAL	Command Current Acceleration/Deceleration	○	○	---	---
Jerk	LREAL	Command Current Jerk	○	○	---	---
Trq	LREAL	Command Current Torque	○	○	---	---
Act	_sAXIS_REF_ACT_DATA	Axis Current Value				
Pos	LREAL	Actual Current Position	○	○	○	○
Vel	LREAL	Actual Current Velocity	○	○	○	○
Trq	LREAL	Actual Current Torque	○	○	---	---
TimeStamp	ULINT	Time Stamp	○	---	○	---
MFaultLvl	_sMC_REF_EVENT	Axis Minor Fault				
Active	BOOL	Axis Minor Fault Occurrence	○	○	○	○
Code	WORD	Axis Minor Fault Code	○	○	○	○
Obsr	_sMC_REF_EVENT	Axis Observation				
Active	BOOL	Axis Observation Occurrence	○	○	○	○
Code	WORD	Axis Observation Code	○	○	○	○
Cfg	_sAXIS_REF_CFG	Axis Basic Settings				
AxNo	UINT	Axis Number	○	○	○	○
AxEnable	_eMC_AXIS_USE	Axis Use	○	○	○	○
AxType	_eMC_AXIS_TYPE	Axis Type	○	○	○	○
NodeAddress	UINT	Node Address	○	---	○	---
ExecID*1	UINT	Execution ID	○	○	○	○
Scale	_sAXIS_REF_SCALE	Unit Conversions				
Num	UDINT	Command Pulse Count per Motor Rotation	○	○	○	○
Den*2	LREAL	Work Travel Distance per Motor Rotation	○	○	○	○
Units	_eMC_UNITS	Unit of Display	○	○	○	○
CountMode*1	_eMC_COUNT_MODE	Count Mode	○	○	○	○
MaxPos*1	LREAL	Maximum Current Position	○	○	○	○
MinPos*1	LREAL	Minimum Current Position	○	○	○	○

*1. A CPU Unit with unit version 1.10 or later and Sysmac Studio version 1.12 or higher are required to use this variable.

*2. The parameter is disabled if you set to use a reducer in the unit conversion settings, which is the function added for CPU Units with unit version 1.11 or later.

Axes Group Variables

_MC_GRP[0-63], _MC1_GRP[0-63], and _MC2_GRP[0-63] are the system-defined Axes Group Variables. The data type is _sGROUP_REF, which is a structure variable.

This section describes the configuration of the Axes Group Variables and provides details on the members using _MC_GRP[0-63] as an example.
The same information applies to _MC1_GRP[0-63] and _MC2_GRP[0-63].
Also, in the descriptions of functions, _MC_AX[*] is used as an example, but the same information applies to _MC1_AX[*] and _MC2_AX[*].

Variable name	Data type	Meaning	Function
_MC_GRP[0-63]	_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. • Execution of the MC_Stop instruction is not in progress for any composition axis. • _MC_GRP[*].Status.Standby (standby) is TRUE. • The Servo is ON for the composition axes. • _MC_AX[*].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. Disabled, Standby, Moving, Stopping, or ErrorStop
Standby	BOOL	Standby	TRUE when the axes group motion instruction is stopped. This is independent of 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. 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 axes stop for an MC_GroupStop instruction. Axes group motion instructions are not executed in this state. (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[*].MFaultLvl.Active is TRUE). Axes group motion instructions are not executed in this state. (CommandAborted is TRUE.)
Details	_sGROUP_REF_DET	Axes Group Control Status	Gives the control status of the instruction.

Variable name	Data type	Meaning	Function
Idle	BOOL	Idle	TRUE when processing is not currently performed for the command value, except when waiting for in-position state. *1 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 for any composition axis. The in-position check performed when positioning for the in-position check. *2
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 during travel in the positive direction, and a minus sign is added during travel 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 Occurrence	TRUE while there is an axes group minor fault.
Code	WORD	Axes Group Minor Fault Code	Contains the error 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 Occurrence	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	Gives the settings of the Axes Group Basic Settings parameters.
GrpNo	UINT	Axes Group Number	Contains the logical number of the axes group.

Variable name	Data type	Meaning	Function
GrpEnable	_eMC_GROUP_USE	Axes Group Use	Shows if the axes group is enabled or disabled. 0: _mcNoneGroup (Undefined Axes Group) 1: _mcUnusedGroup (Unused Axes Group) 2: _mcUsedGroup (Used Axes Group)
ExecID*3	UINT	Execution ID	Contains the assigned task execution ID. 0: Not assigned to task (undefined axes group) 1: Assigned to primary periodic task 2: Assigned to priority-5 periodic task
Kinematics	_sGROUP_REF_KIM	Kinematics Transformation Settings	Contains the definition of the kinematic conversions for the axes group.
GrpType	_eMC_TYPE	Composition	Gives the axis composition of multi-axes coordinated control. 0: _mcXY (two axes) 1: _mcXYZ (three axes) 2: _mcXYZU (four axes)
Axis[0]	UINT	Composition Axis for Axis A0	Contains the axis number that is assigned to axis A0.
Axis[1]	UINT	Composition Axis for Axis A1	Contains the axis number that is assigned to axis A1.
Axis[2]	UINT	Composition Axis for Axis A2	Contains the axis number that is assigned to axis A2.
Axis[3]	UINT	Composition Axis for Axis A3	Contains the axis number that is assigned to axis A3.

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

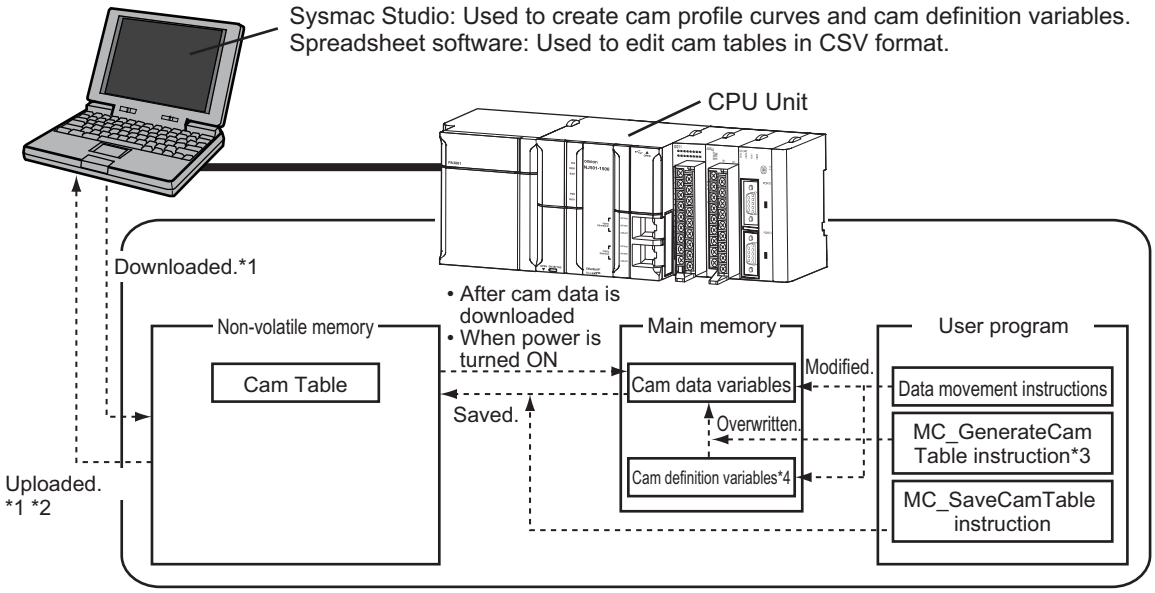
*2. This variable is FALSE when all composition axes in the axes group are within the in-position ranges set in the axis parameters.

*3. A CPU Unit with unit version 1.10 or later and Sysmac Studio version 1.12 or higher are required to use this variable.

6-7 Cam Tables and Cam Data Variables

The MC Function Module uses the cam profile curves that you create on the Cam Editor of the Sysmac Studio as cam tables.
The cam table data is handled as cam data variables in the user program in the NJ/NX-series CPU Unit.

Creating and Saving Cam Tables



*1. Use the Synchronization menu command of the Sysmac Studio to upload and download the project.
*2. The cam data variables that are uploaded cannot be changed on the Cam Editor. Refer to the *Editing a Cam Data Variable on the Computer after Editing It from the User Program* on page 6-38 for the procedure to edit a cam data variable on the computer after editing it from the user program.
Also, if the project is rebuilt or the cam profile curve is changed from the Sysmac Studio, the cam data variable that was uploaded is overwritten with the cam profile curve data.
*3. A CPU Unit with unit version 1.08 or later and Sysmac Studio version 1.09 or higher are required to use the MC_GenerateCamTable (Generate Cam Table) instruction.
*4. “Cam definition variable” is the generic term for cam property variables and cam node variables.

● Cam Table Data Flow

- Use the Sysmac Studio to download the cam profile curves that you created in the Sysmac Studio to the CPU Unit to save them as cam tables in the non-volatile memory in the CPU Unit. When you upload a cam table to the Sysmac Studio, the cam table that was saved in the non-volatile memory is uploaded.
- The cam tables that were saved in non-volatile memory are implemented as cam data variables in the main memory after you download them or when the power is turned ON.
- You can use the user program to edit cam data variables and cam definition variables in the main memory. Refer to *9-2-5 Cam Tables and Start Cam Operation Instruction* on page 9-17 for information on cam data variables and cam definition variables.
- The MC_GenerateCamTable (Generate Cam Table) instruction in the user program can overwrite the cam data variable in main memory according to the value of the cam definition variable.

- The motion control instruction MC_SaveCamTable saves the cam data variables in the main memory to non-volatile memory.
- You can upload the cam definition variable that were created in the Cam Data Settings of the Sysmac Studio even after the variable is changed in the user program. If the cam definition variable was created as a user-defined variable, you cannot upload it after it is changed in the user program.
- Refer to the *Sysmac Studio Version 1 Operation Manual (Cat. No. W504)* for information on creating and transferring the cam definition variable in the Sysmac Studio.
- For details on the MC_GenerateCamTable (Generate Cam Table) instruction and MC_SaveCamTable instruction, refer to the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)*.
- You can upload and download cam tables regardless of the operating mode of the NJ/NX-series CPU Unit mode or the status of the MC Function Module. You cannot upload cam data, download cam data, start online operation, perform online editing, or start data traces during a cam table save operation. The MC_SaveCamTable instruction is not executed during online editing.
- All axes in motion will decelerate at the maximum deceleration rate when you start the download process.

Precautions for Correct Use

If you change any cam data in the user program, those changes are lost and the cam table in non-volatile memory is restored if you restart the power or download cam data from the Sysmac Studio. Also, you cannot upload the data in the main memory from the Sysmac Studio.

Version Information

If a CPU Unit with unit version 1.13 or later and Sysmac Studio version 1.17 or higher are combined, commands to the I/O devices can continuously be sent even when the download process is in progress.
For the CPU Unit with unit version 1.12 or earlier, sending commands to the I/O devices is stopped when the download process is executed.
Refer to the *NJ/NX-series CPU Unit Software User's Manual (Cat. No. W501)* for how to set to stop or continue sending commands to the I/O devices when the download process starts.

Stopping sending commands to I/O devices

Servo is turned OFF during the download, and the axis status will be *Disable* (Axis Disabled).

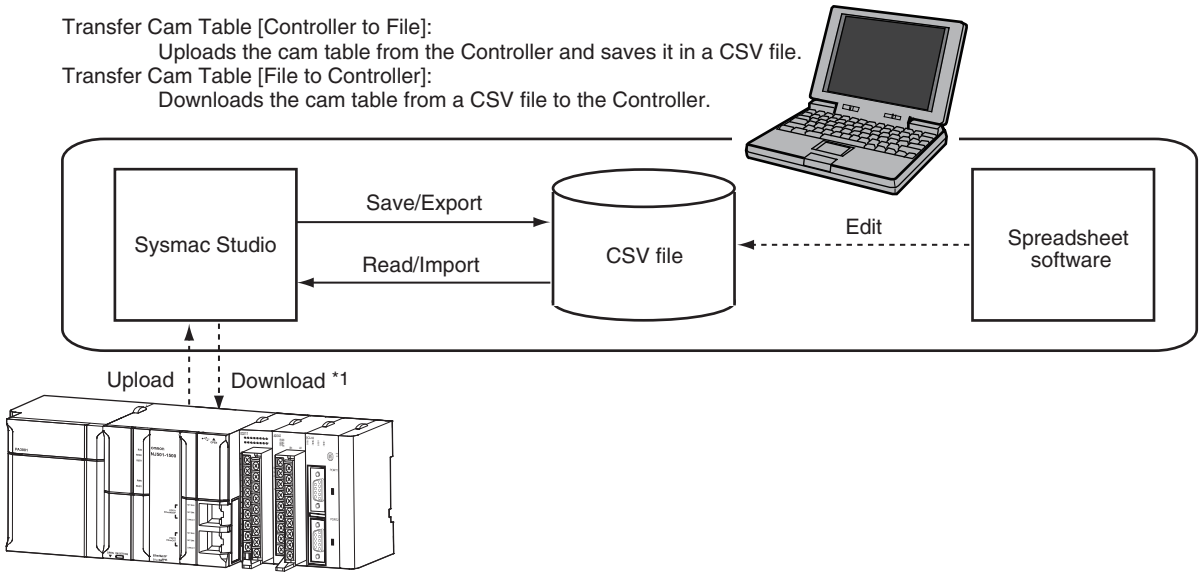
Continuing sending commands to I/O devices

The MC_Power (Power Servo), MC_SetTorqueLimit (Set Torque Limit), and MC_SetTorqueLimit2 (Set Torque Limit 2) instructions can run even when the download process is in progress.
During download, the servo ON state set by the MC_Power instruction immediately before the download and the output torque limits set by the MC_SetTorqueLimit and MC_SetTorqueLimit2 instructions are maintained.
The Servo ON state and torque limits are maintained even if the MC_Power, MC_SetTorqueLimit, and MC_SetTorqueLimit2 instructions are deleted from the user program updated by the download.

Editing a Cam Data Variable on the Computer after Editing It from the User Program

If you edit or overwrite a cam data variable from the user program and then use the MC_SaveCamTable instruction to save the cam table to non-volatile memory, you cannot edit the data with the Cam

Editor of the Sysmac Studio. This section describes how to use spreadsheet software to edit the data and then use it as a cam table.



*1. Use the Synchronization menu command of the Sysmac Studio to upload and download the project.

● Saving a Cam Table from Non-volatile Memory to a CSV File

- Right-click a cam profile that you edited in the **Cam Data Settings** of the Sysmac Studio and select **Transfer Cam Table [Controller to File]** from the menu.
- The **Save** Dialog Box is displayed. Enter the file save location and file name, and then click the **Save** Button.

● Editing CSV Files

- Use spreadsheet software or other CSV-compatible software to edit the CSV file.

● Transferring the CSV File to the CPU Unit

- Right-click the cam profile to download and select **Transfer Cam Table [File to Controller]** from the menu.
- The **Open File** Dialog Box is displayed. Specify the file to transfer, and then click the **OK** Button.
- To enable the cam table that you transferred, reset the Controller or cycle the power supply to the Controller after the cam table is transferred.

Precautions for Correct Use

- Synchronize the data with the Controller before executing **Transfer Cam Table [File to Controller]**.
- If you transfer the Cam Data Settings to the Controller through the synchronization after executing **Transfer Cam Table [File to Controller]**, the cam table in the Controller is replaced with the data in the Cam Data Settings.
  In this case, you need to execute **Transfer Cam Table [File to Controller]** again, or do not include the Cam Data Settings in the synchronization data.

You can also export the Cam Data Settings that were entered from the Cam Editor to a CSV file. Refer to the *Sysmac Studio Version 1 Operation Manual (Cat. No. W504)* for information on the Cam Data Settings and the export procedure.

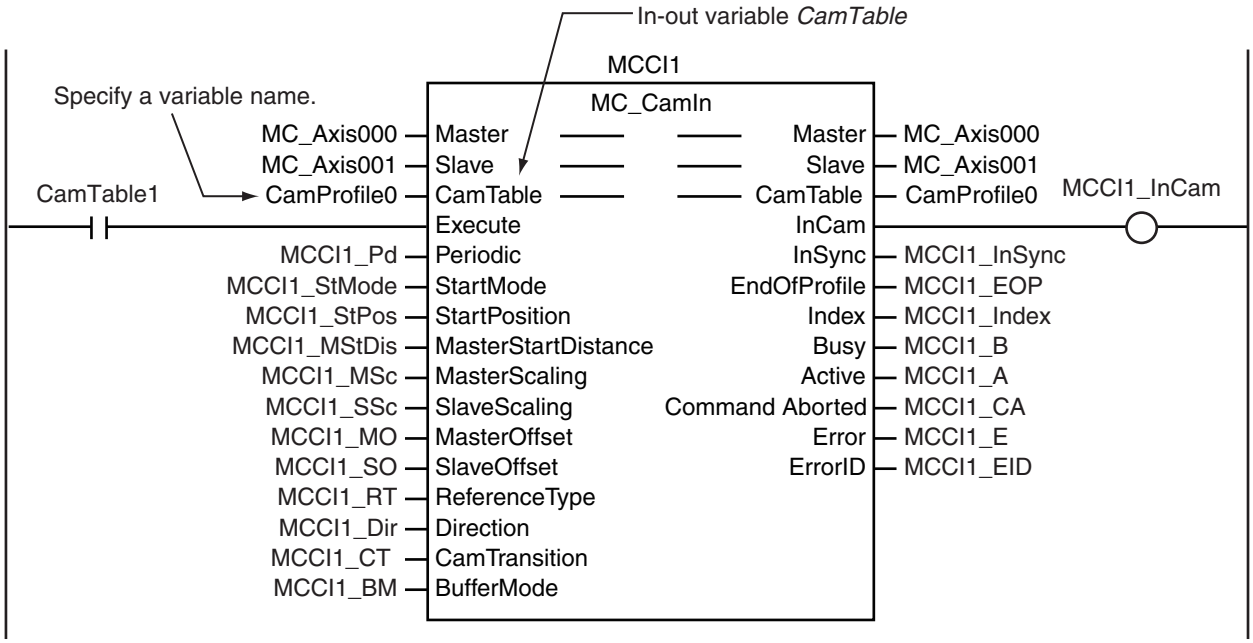
Cam Profile Curve Names

When a cam profile is created in the Sysmac Studio, **CamProfile0** is used as the default name. Each time you create another cam profile, the number on the end of the name is incremented.
You can change the name of any cam profile as required from the Sysmac Studio.

The cam profile names that are set on the Sysmac Studio are used as the cam table names.

Specifying Cam Tables in the User Program

In the user program, the cam table name is specified for the in-out variable *CamTable* in motion control instructions.



6-8 Programming Motion Controls

Place motion control instructions in the user program of the NJ/NX-series CPU Unit to perform motion control. Programs that contain motion control instructions are called motion control programs.

Precautions for Correct Use

- You can set and program up to 256 axes on the Sysmac Studio for any model of CPU Unit. You cannot download a project to the CPU Unit if the project contains more than the maximum number of controlled axes for that CPU Unit.
- When you reuse a project, make sure that the maximum number of controlled axes for the CPU Unit model is not exceeded.
- Even axes that are set as **unused axes** are included in the number of controlled axes.

Refer to the *NJ/NX-series CPU Unit Software User's Manual (Cat. No. W501)* for details on programming.

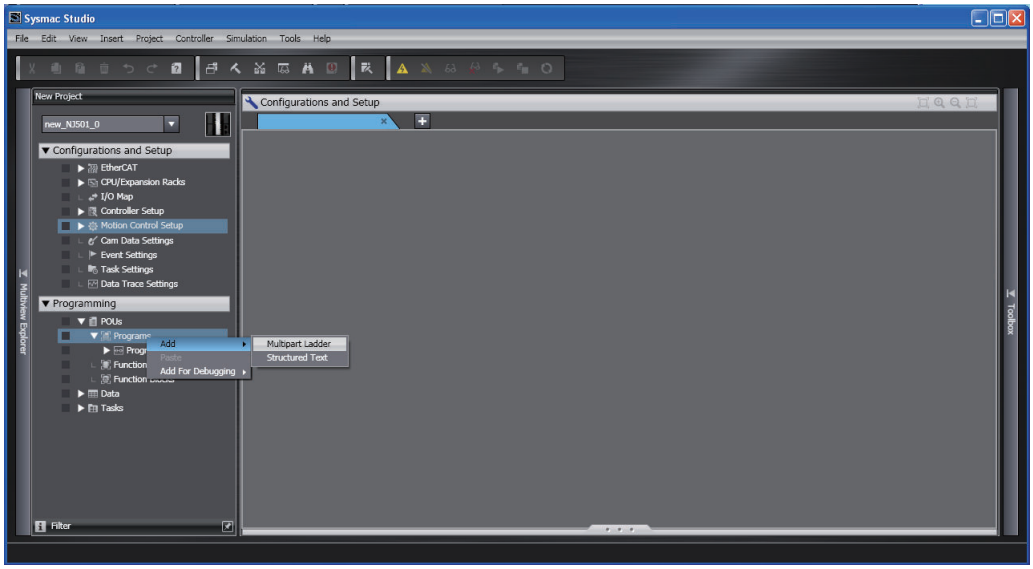
This section gives the procedure to create a program in an existing project on the Sysmac Studio.

1 Starting the Sysmac Studio

Start the Sysmac Studio and open the project.

2 Adding a Program

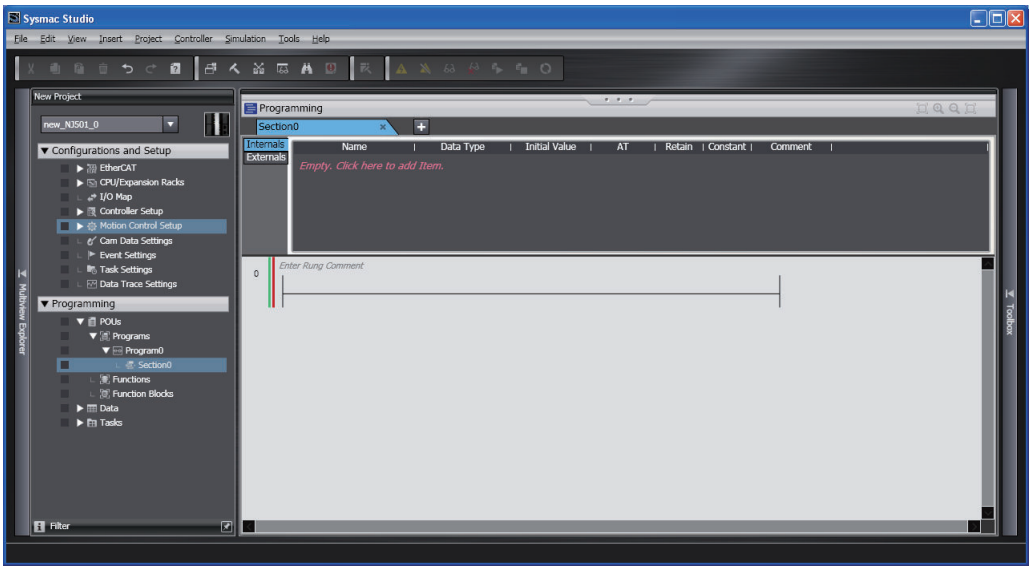
Right-click **Programs** in the Multiview Explorer and select **Multipart Ladder** or **Structured Text** from the **Add** Menu.



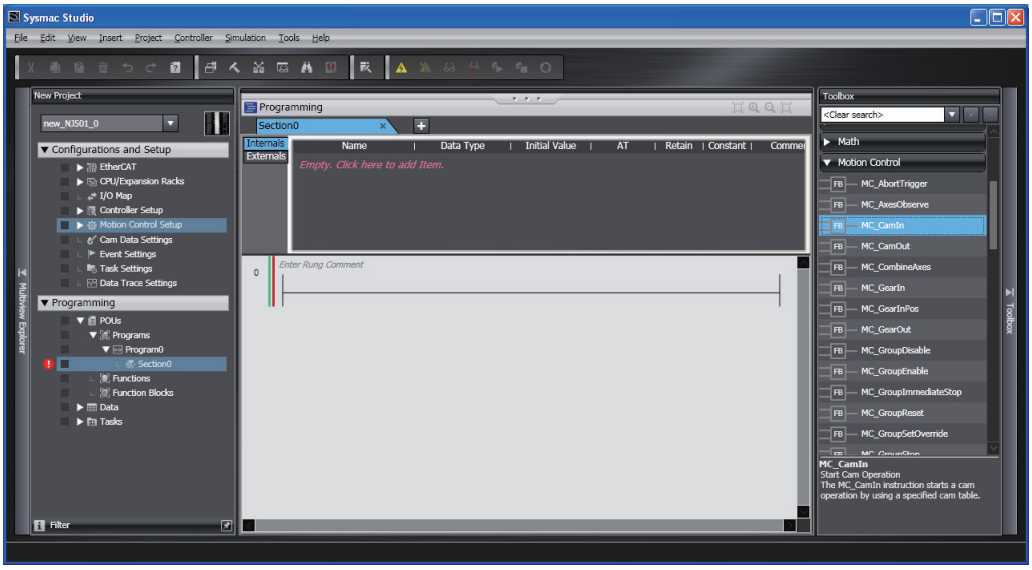
A program is added to the Multiview Explorer.

3 Editing the Program

Right-click a section in the new program and select **Edit** from the menu. The Program Edit Tab Page is displayed.



Select the required instructions from the Toolbox and enter the program.



Refer to the *NJ/NX-series CPU Unit Software User's Manual (Cat. No. W501)* for details on programming.

Refer to the *Sysmac Studio Version 1 Operation Manual (Cat. No. W504)* for specific procedures.

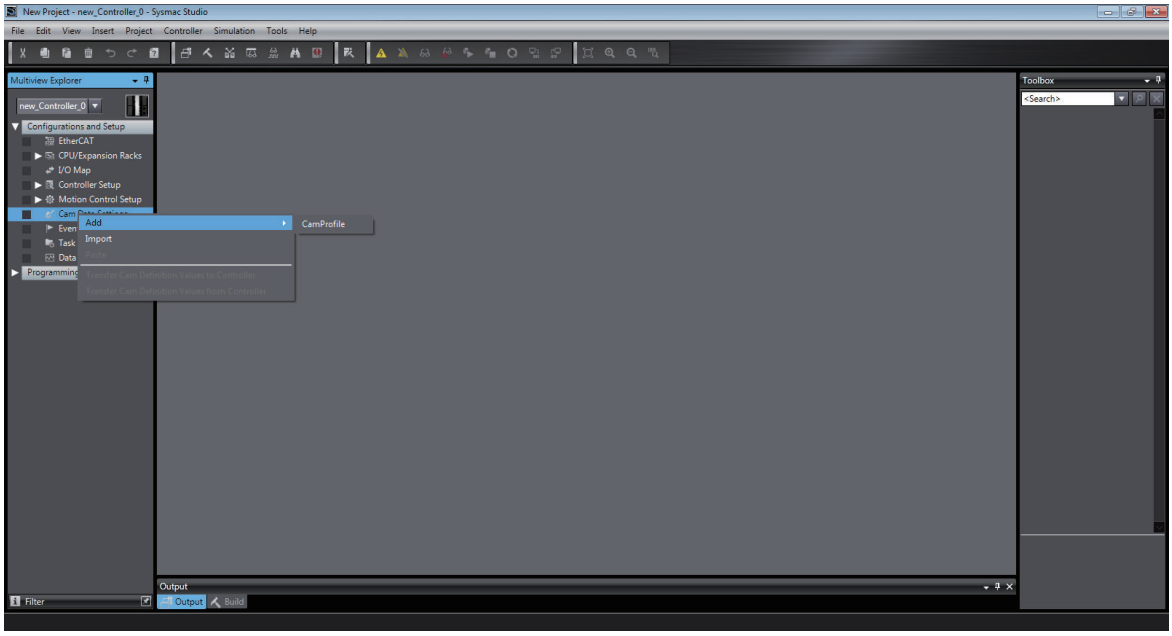
6-9 Creating Cam Tables

This section will explain how to use the Cam Editor of the Sysmac Studio to create a cam table.

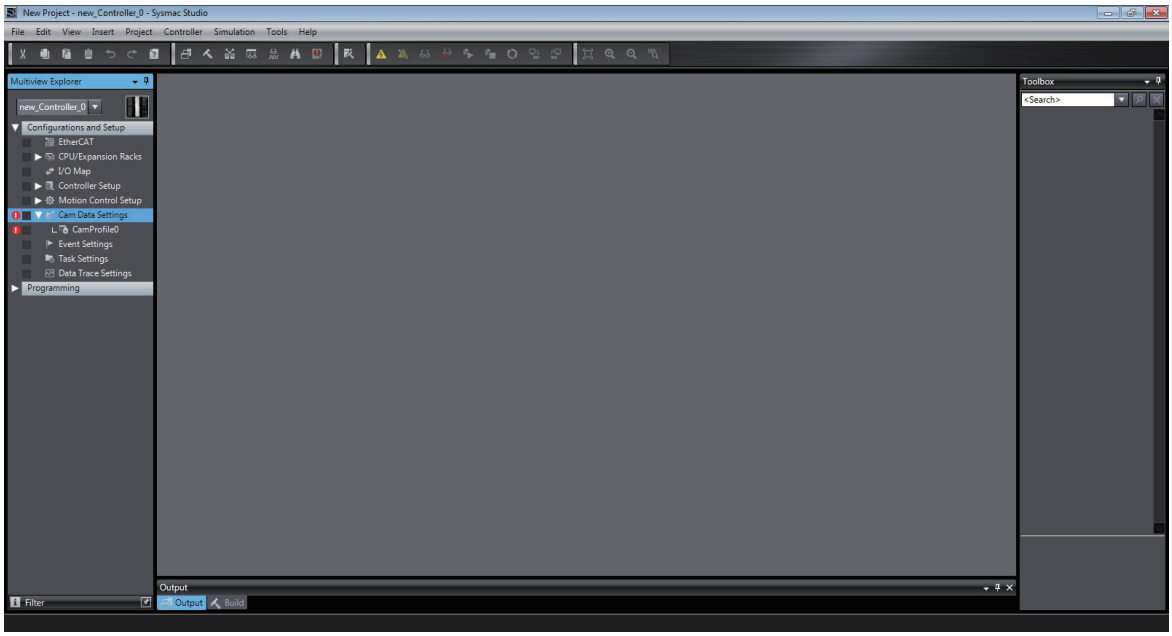
Refer to the *Sysmac Studio Version 1 Operation Manual (Cat. No. W504)* for details on the Cam Editor.

1 Adding a Cam Profile

Right-click **Cam Data Settings** in the Multiview Explorer and select **CamProfile** from the **Add** Menu.

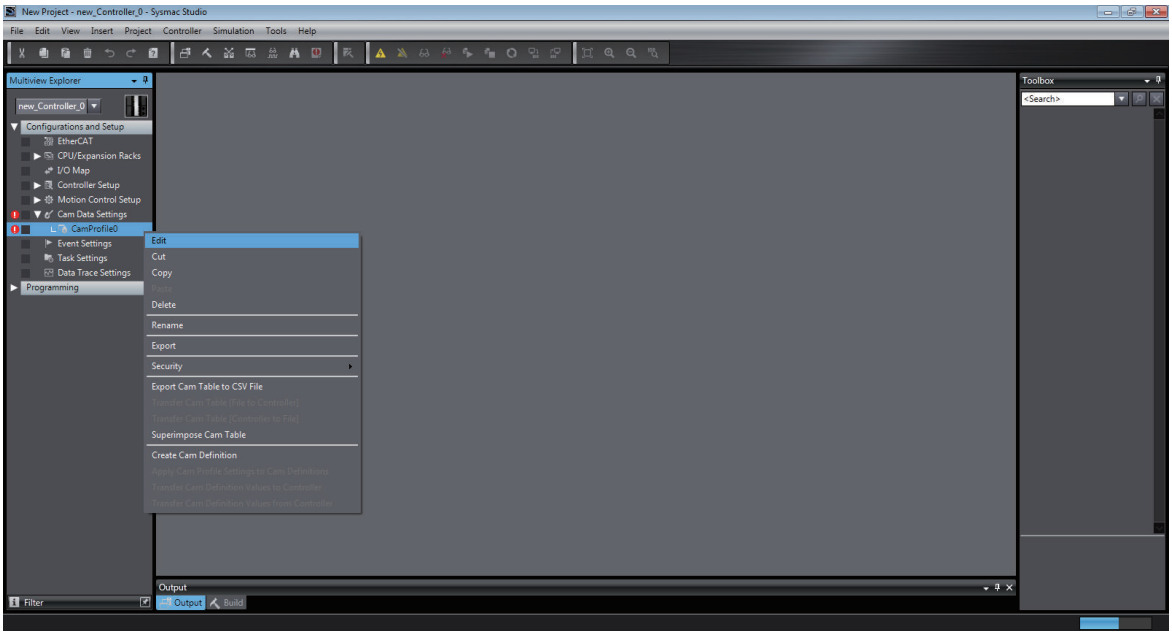


A cam profile is added to the Multiview Explorer. You can change the name of the cam profile as required from the default name of *CamProfile0*.

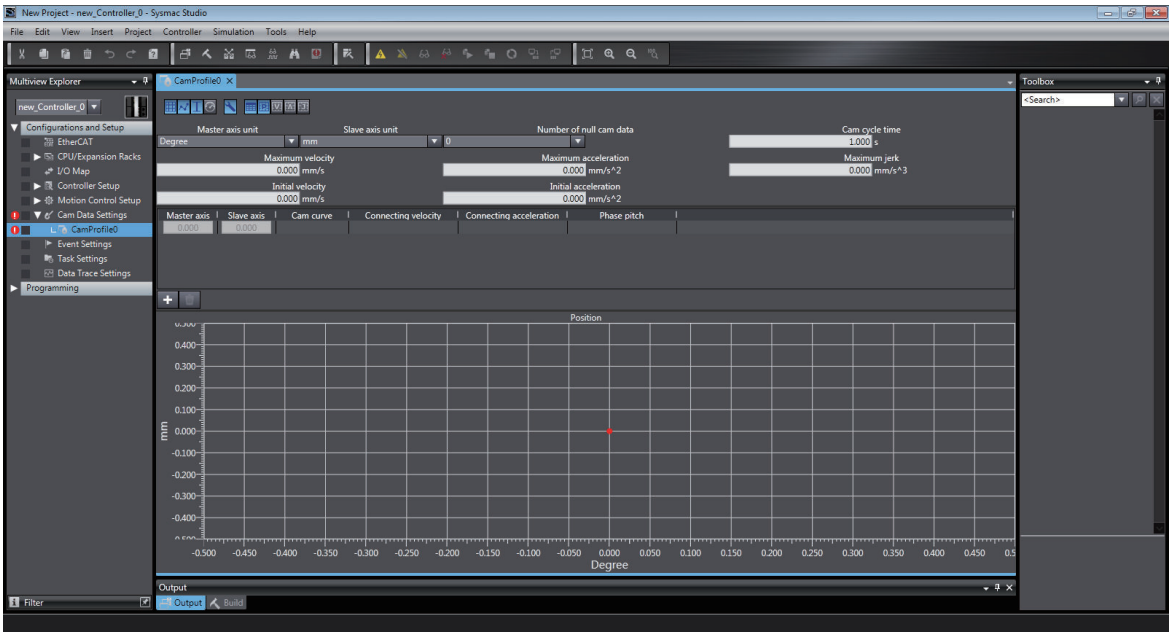


2 Editing the Cam Profile

Right-click the cam profile in the Multiview Explorer and select **Edit** from the menu.



The Cam Profile Edit Tab Page is displayed.



Make the settings and enter the cam profile.

Refer to the *Sysmac Studio Version 1 Operation Manual (Cat. No. W504)* for specific procedures.

7

Manual Operation

This section describes manual operation when the MC Function Module is used together with an OMRON 1S-series Servo Drive.

7-1 Outline

This section describes how to combine the MC Function Module and OMRON 1S-series Servo Drives together and use motion control instructions from the user program to perform manual operations.

The motion control instructions for manual operation are MC_Power and MC_MoveJog. MC_Power changes the Servo Drive to the Servo ON state and MC_MoveJog performs jogging.

Precautions for Correct Use

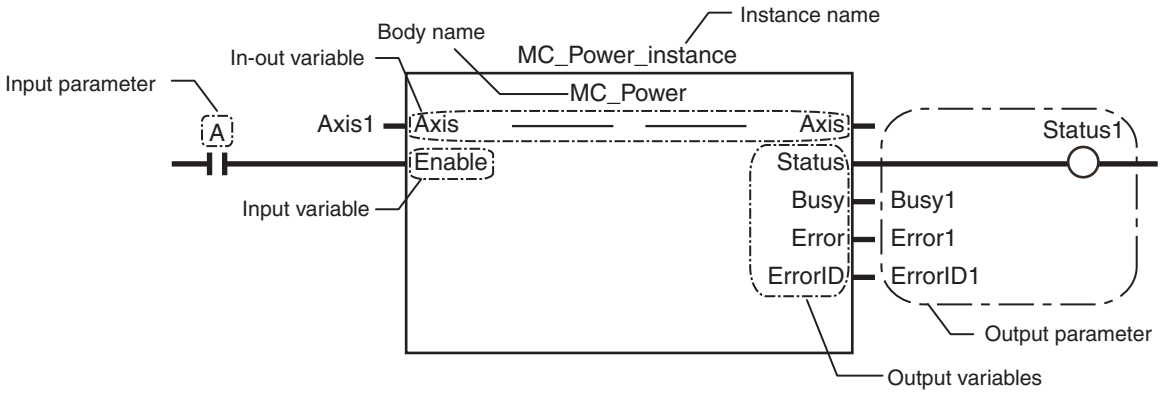
You must set the axes to perform manual operation.
Refer to *Section 3 Configuring Axes and Axes Groups* on page 3-1 for details on how to set axes.

Additional Information

- Use the MC Test Run function of the Sysmac Studio if you perform manual operation without programming.
  Refer to *4-3 Checking Motor Operation* on page 4-8 for information on how to use the Sysmac Studio to perform manual operation.
- Refer to *Section 6 Motion Control Programming* on page 6-1 for information on how to create user programs.
- The same procedures and operations are applicable to an OMRON G5-series Servo Drive.

7-2 Turning ON the Servo

You can turn the Servo ON or OFF to enable or disable sending operation commands to the Servo Drive.
The MC_Power (Power Servo) motion control instruction is used.



Specify the axis to move with the *Axis* in-out variable.
Change the *Enable* input variable for MC_Power to TRUE to turn ON the Servo. Change *Enable* to FALSE to turn OFF the Servo.

Precautions for Correct Use

- If you change *Enable* to FALSE while the axis is moving, the command stops immediately and all motion control instructions for that axis are disabled.
- If you use an NX-series Pulse Output Unit, you must provide a separate means to turn the power supply to the motor drive ON and OFF. Refer to the *NX-series Position Interface Units User's Manual (Cat. No. W524)* for details.

Additional Information

If an OMRON 1S-series Servomotor or G5-series Servomotor with an absolute encoder is used, or if an OMRON G5-series Linear Motor Type Servomotor/Servo Drive with built-in EtherCAT communications is used with an absolute external scale, home is defined when the *Enable* input variable to the MC_Power instruction changes to TRUE.

Version Information

For a CPU Unit with unit version 1.10 or later, if an OMRON 1S-series Servomotor or G5-series Servomotor with an absolute encoder is used, or if an OMRON G5-series Linear Motor Type Servomotor/Servo Drive with built-in EtherCAT communications is used with an absolute external scale, home is also defined when EtherCAT process data communications change from a non-established to an established state, in addition to the step shown in the above Additional Information.

7-2-1 Turning ON the Servo

1 Adding and Setting an Axis

Add and set an axis from the Sysmac Studio.

For details, refer to *3-2-2 Setting Procedure* on page 3-10.

2 Setting Axis Parameters
Set the axis parameters from the Sysmac Studio.
For details, refer to *3-2-2 Setting Procedure* on page 3-10.

3 Writing the User Program
Create the user program from the Sysmac Studio.
For details, refer to *6-8 Programming Motion Controls* on page 6-41.

4 Downloading Axis Parameters and the User Program
Download the axis parameters and user program to the CPU Unit.
Use the Synchronization menu command of the Sysmac Studio to download the project to the CPU Unit.
For details, refer to *3-2-2 Setting Procedure* on page 3-10.

5 Executing the User Program
Execute the user program and change the *Enable* input variable for MC_Power to TRUE.
The Servo Drive will change to the Servo ON state.

7-2-2 Setting Axis Parameters

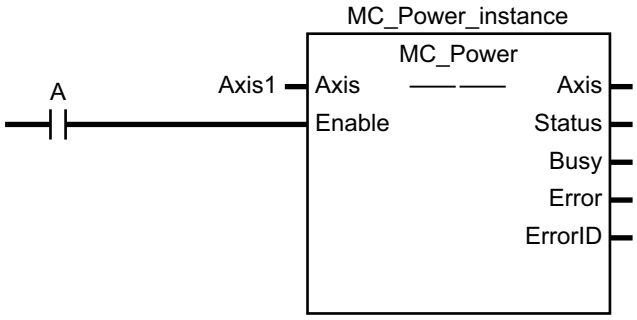
Only the following axis parameter settings are required if you want only to change to the Servo ON state.
The following table provides examples of the settings.

Parameter name	Setting
Axis Variable Name	Axis1*1
Axis Number	1*2
Axis Use	Used axis
Axis Type	Servo axis
Input Device/Output Device	1*3

*1. If there is more than one axis, a different variable name is set for each axis.
*2. If there is more than one axis, a different value is set for each axis.
*3. Set the node address to the same value as the node address that is set on the Servo Drive. If there is more than one axis, a different value is set for each axis.

7-2-3 Programming Example

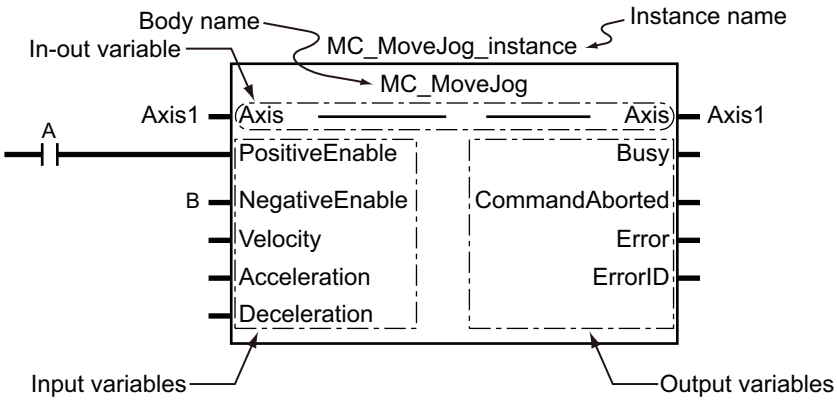
The following sample programming turns the Servo ON and OFF for an axis named Axis1 based on the value of bit A.



For details on the MC_Power (Power Servo) instruction, refer to the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)*.

7-3 Jogging

Use the motion control instruction MC_MoveJog for jogging.



Specify the axis to jog with the *Axis* in-out variable.
Change the *PositiveEnable* input variable to TRUE to start the axis with the specified positive *Velocity* (Target Velocity) and *Acceleration* (Acceleration Rate). Change *PositiveEnable* to FALSE to decelerate and stop the axis at the specified *Deceleration* (Deceleration Rate).
Similarly, if you change the *NegativeEnable* input variable to TRUE, the axis will start in the negative direction. Change the variable to FALSE to stop the axis.
You can perform jogging even if the home has not yet been defined.

7-3-1 Jogging Procedure

1 Adding and Setting an Axis
Add and set an axis from the Sysmac Studio.
For details, refer to *3-2-2 Setting Procedure* on page 3-10.

2 Setting Axis Parameters
Set the axis parameters from the Sysmac Studio.
For details, refer to *3-2-2 Setting Procedure* on page 3-10.

3 Writing the User Program
Create the user program from the Sysmac Studio.
For details, refer to *6-8 Programming Motion Controls* on page 6-41.

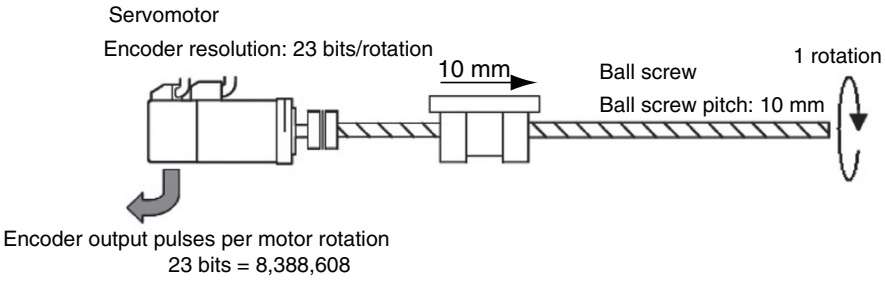
4 Downloading Axis Parameters and the User Program
Download the axis parameters and user program to the CPU Unit.
Use the Synchronization menu command of the Sysmac Studio to download the project to the CPU Unit.
For details, refer to *3-2-2 Setting Procedure* on page 3-10.

5 Executing the User Program
Execute the user program and change the *Enable* input variable for MC_Power to TRUE to change the Servo Drive to the Servo ON state.

Change either the *PositiveEnable* or *NegativeEnable* input variable for the MC_MoveJog instruction to TRUE to jog.

7-3-2 Setting Axis Parameters

Set the following axis parameters if you want to jog when home is not defined.
The following setting example is for a one-axis device.



Parameter name	Setting
Axis Variable Name	Axis1*1
Axis Number	1*2
Axis Use	Used axis
Axis Type	Servo axis
Input Device/Output Device	1*3
Command Pulse Count Per Motor Rotation	8,388,608*4
Work Travel Distance Per Motor Rotation	10,000*4
Software Limits	Enabled for actual position
Unit of Display	μm
Count Mode	Linear Mode
Maximum Velocity	500,000*5
Maximum Jog Velocity	50,000*6
Maximum Acceleration	5,000,000*7
Maximum Deceleration	5,000,000*7

*1. If there is more than one axis, a different variable name is set for each axis.
*2. If there is more than one axis, a different value is set for each axis.
*3. Set the node address to the same value as the node address that is set on the Servo Drive. If there is more than one axis, a different value is set for each axis.
*4. The position command unit will be 1 μm.
*5. The maximum velocity will be 3,000 r/min = 30 m/min = 0.5 m/s = 500,000 μm/s.
*6. The maximum jog velocity will be 10% of the maximum velocity, i.e., 0.05 m/s = 50,000 μm/s.
*7. The maximum acceleration and the maximum deceleration are 5 m/s^2. The acceleration time to the maximum velocity (3,000 r/min) is 0.1 s.

7-3-3 Setting Example for Input Variables

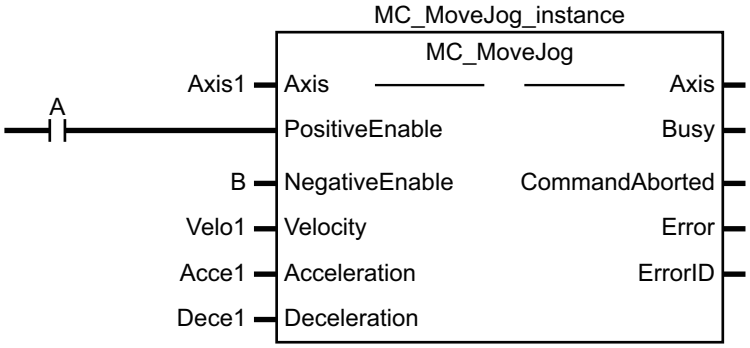
This section describes the settings for the MC_MoveJog input variables *Velocity* (Target Velocity), *Acceleration* (Acceleration Rate), and *Deceleration* (Deceleration Rate).

- For example, set *Velocity* to *30,000* to jog at a velocity of 0.03 m/s (30,000 μm/s).

- Set *Acceleration* and *Deceleration* to *3,000,000* to accelerate and decelerate at 3 m/s^2 (3,000,000 μm/s^2).

7-3-4 Programming Example

The following programming example jogs an axis named Axis1 in the positive direction for the value of bit A and in the negative direction for the value of bit B.



In this example, *Velocity* (Target Velocity) is *Velo1*, *Acceleration* is *Acce1*, and *Deceleration* is *Dece1*.

Set the values for each variable in the user program in advance to operate the axis with the example input variable settings.

- Velo1 = 30,000
- Acce1 = 3,000,000
- Dece1 = 3,000,000

For details on the MC_MoveJog (Jog) instruction, refer to the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)*.

8

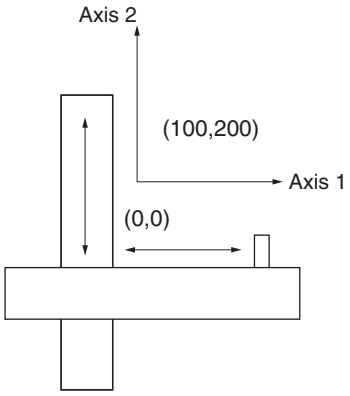
Homing

This section describes homing.

8-1 Outline of Homing

To perform positioning to absolute positions in a positioning system, you must first perform homing.

For example, if you want to perform positioning to the position (axis 1, axis 2) = (100 mm, 200 mm) in the XY plane shown below, you must define the position of home (0,0).
The process of defining home is called homing.



In the MC Function Module, use the motion control instruction MC_Home or MC_HomeWithParameter to define home.

Name	Description
Homing	Home is defined by actually moving the motor and using the limit sensors, home proximity sensor, and home input signal to determine the position of home. Use a proximity sensor or the encoder's Z phase signal as the home input signal.

Precautions for Correct Use

- The defined home is lost in the following cases.
  When MC_SetPosition is executed.
  When an overflow or underflow occurs in Linear Mode.
  When homing is started.
  The control state of EtherCAT communications is not Operational state.
- Some of the homing functions are restricted for the NX-series Position Interface Units. Refer to the *NX-series Position Interface Units User's Manual (Cat. No. W524)* for details.

Additional Information

If an OMRON 1S-series Servomotor or G5-series Servomotor with an absolute encoder is used, or if an OMRON G5-series Linear Motor Type Servomotor/Servo Drive with built-in EtherCAT communications is used with an absolute external scale, home is defined when the *Enable* input variable to the MC_Power instruction changes to TRUE.
Refer to *8-4 Homing with an Absolute Encoder* on page 8-14 for details on the home with absolute encoder.

Version Information

For a CPU Unit with unit version 1.10 or later, if an OMRON 1S-series Servomotor or G5-series Servomotor with an absolute encoder is used, or if an OMRON G5-series Linear Motor Type Servomotor/Servo Drive with built-in EtherCAT communications is used with an absolute external scale, home is also defined when EtherCAT process data communications change from a non-established to an established state, in addition to the step shown in the above Additional Information.

The MC_MoveZeroPosition (High-speed Home) instruction is also provided to perform positioning to home as defined for the previously described method.

Name	Description
High-speed Homing	The axis returns to home using an absolute position of 0 as the target position.

The MC Function Module can operate the motor even when home is undefined (excluding MC_MoveZeroPosition).

Function	Operation
Jogging and velocity control	If home is not defined, the position at startup is defined as 0 to control movement.
High-speed homing	High-speed homing cannot be used. If it is used, an instruction error will occur.
Positioning	If home is not defined, the position at startup is defined as 0 to control movement.
Interrupt feeding	
Starting cam operation	
Starting gear operation	
Synchronous positioning	
Combining axes	
Torque control	
Zone monitoring	
Linear interpolation	Interpolation cannot be used. If it is used, an instruction error will occur.
Circular interpolation	

Additional Information

Software limits are not valid when home is not defined.

Changes in the home definition status for operations and events are listed in the following table.

Operation or event	Condition for change	Home definition status change
Servo turned ON or axis enabled	Incremental encoder	No change
	When absolute data is read normally from the absolute encoder	Home is defined.
	When absolute data cannot be read from the absolute encoder	Home is undefined.
Changing the current position	When starting	Home is undefined.
Homing	When starting	Home is undefined.
	When ending	Home is defined.
Overflows and underflows	When overflow or underflow occurs	Home is undefined.

Precautions for Correct Use

- For a virtual servo axis, home is always defined with a **zero position preset**. The setting of the **Homing Method** axis parameter is ignored.
- The positive drive prohibit input (POT), negative drive prohibit input (NOT), and home proximity input (DEC) of the Servo Drive are used by the MC Function Module as the positive limit input, negative limit input, and home proximity input.
  Make sure that the signal widths for all of these input signals are long enough for the Servo Drive to detect them and longer than the control period of the MC Function Module.
  If the input signal widths are shorter than the control period, the MC Function Module may not be able to detect the input signals, resulting in incorrect operation.
- You must set the Servo Drive parameters for each Servo Drive input signal. Refer to the manual for your Servo Drive and the appendices and make the proper settings.

8-2 Homing Procedure

This section describes the procedure to perform homing.

1 Adding and Setting an Axis
Add and set an axis from the Sysmac Studio.

2 Setting Axis Parameters
Set the homing method with the homing parameters.

3 Writing the User Program
Create the user program from the Sysmac Studio.
For details, refer to *6-8 Programming Motion Controls* on page 6-41.

4 Downloading Axis Parameters and the User Program
Download the axis parameters and user program to the CPU Unit.
Use the Synchronization menu command of the Sysmac Studio to download the project to the CPU Unit.
For details, refer to *3-2-2 Setting Procedure* on page 3-10.

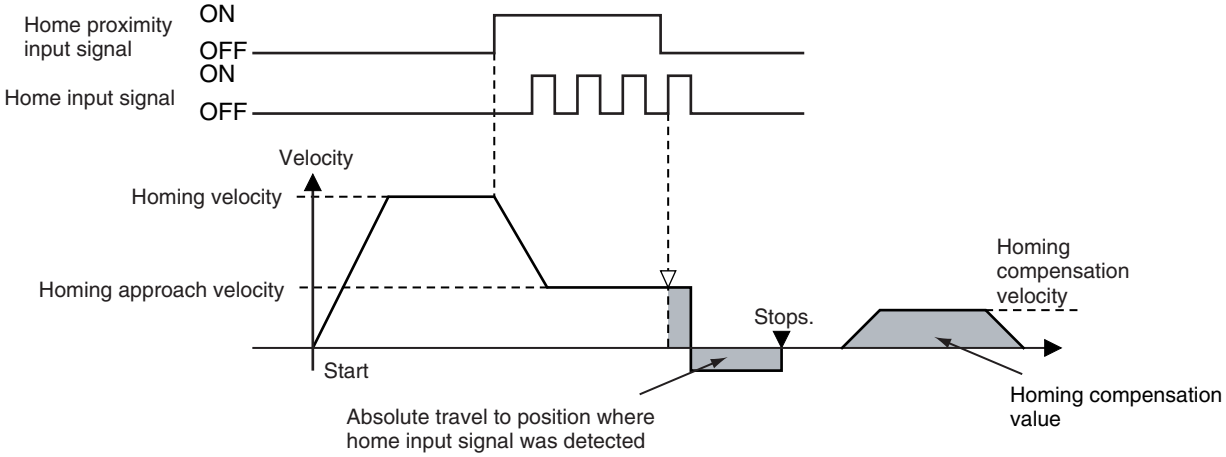
5 Executing the User Program
Execute the user program and change the *Enable* input variable for MC_Power to TRUE to change the Servo Drive to the Servo ON state.
Homing is performed when the *Execute* input variable to the MC_Home or MC_HomeWith-Parameter instruction changes to TRUE.

8-2-1 Setting Homing Parameters

Set the homing parameters to specify the homing procedure.
Set the homing parameters from the Sysmac Studio.

Parameter name	Description
Homing Method	Set the homing operation.
Home Input Signal	Select the input to use for the home input signal.
Homing Start Direction	Set the start direction for when homing is started.
Home Input Detection Direction	Set the home input detection direction for homing.
Operation Selection at Positive Limit Input	Set the stopping method when the positive limit input turns ON during homing.
Operation Selection at Negative Limit Input	Set the stopping method when the negative limit input turns ON during homing.
Homing Velocity	Set the homing velocity. (Unit: command units/s)
Homing Approach Velocity	Set the velocity to use after the home proximity input turns ON. (Unit: command units/s)
Homing Acceleration	Set the acceleration rate for homing. If this parameter is set to *0*, the homing velocity is used without any acceleration. (Unit: command units/s^2)

Parameter name	Description
Homing Deceleration	Set the deceleration rate for homing. If this parameter is set to *0*, the zero velocity is used without any deceleration. (Unit: command units/s^2)
Homing Jerk	Set the jerk for homing. Set *0* for no jerk. (Unit: command units/s^3)
Home Input Mask Distance	Set the home input mask distance when you set the Homing Operation Mode to the proximity reverse turn/home input mask distance. (Unit: command units)
Home Offset	Preset the actual position for the value that is set after homing. (Unit: command units)
Homing Holding Time	Set the holding time in milliseconds when you set the Homing Operation Mode to the proximity reverse turn/holding time. (Unit: ms)
Homing Compensation Value	Set the homing compensation value that is applied after the home is defined. (Unit: command units)
Homing Compensation Velocity	Set the velocity to use for homing compensation. (Unit: command units/s)



Homing Method

You can select any of the ten operation modes to define home.

- Proximity reverse turn/home proximity input OFF
- Proximity reverse turn/home proximity input ON
- Home proximity input OFF
- Home proximity input ON
- Limit input OFF
- Proximity reverse turn/home input mask distance
- Limit inputs only
- Proximity reverse turn/holding time
- No home proximity input/holding home input
- Zero position preset

The following table shows the homing parameters that are used for each Homing Operation Mode.

(○: Parameter is used, ×: Parameter is not used.)

Homing Operation Mode	Homing parameters														
	Home Input Signal	Homing Start Direction	Home Input Detection Direction	Operation Selection at Positive Limit Input	Operation Selection at Negative Limit Input	Homing Velocity	Homing Approach Velocity	Homing Acceleration	Homing Deceleration	Homing Jerk	Home Input Mask Distance	Home Offset	Homing Holding Time	Homing Compensation Value	Homing Compensation Velocity
Proximity reverse turn/home proximity input OFF	○	○	○	○	○	○	○	○	○	○	×	○	×	○	○
Proximity reverse turn/home proximity input ON	○	○	○	○	○	○	○	○	○	○	×	○	×	○	○
Home proximity input OFF	○	○	○	○	○	○	○	○	○	○	×	○	×	○	○
Home proximity input ON	○	○	○	○	○	○	○	○	○	○	×	○	×	○	○
Limit input OFF	○	○	○	○	○	○	○	○	○	○	×	○	×	○	○
Proximity reverse turn/home input mask distance	○	○	○	○	○	○	○	○	○	○	○	○	×	○	○
Limit inputs only	×	○	○	○	○	○	○	○	○	○	×	○	×	○	○
Proximity reverse turn/holding time	×	○	○	○	○	○	○	○	○	○	×	○	○	○	○
No home proximity input/holding home input	○	○	○	○	○	○	○	○	○	○	×	○	×	○	○
Zero position preset	×	×	×	×	×	×	×	×	×	×	×	○	×	×	×

For details on the Homing Operation Modes, refer to the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)*.

Precautions for Correct Use

You cannot map the Z-phase input to a PDO for an OMRON G5-series Linear Motor Type Servo Drive with built-in EtherCAT communications. Therefore, if you use the **No Home Proximity Input/Holding Home Input** Homing Operation Mode, which can use a Z-phase input mapped to a PDO, do not select the Z-phase input for the home input signal.

Home Input Signal

In a Homing Operation Mode that uses a home input signal, select either the Z phase signal of the Servo Drive or an external home signal as the signal to define home.

Precautions for Correct Use

This parameter can be used to set a home input signal only when you are connected to an OMRON 1S-series Servo Drive or G5-series Servo Drive.

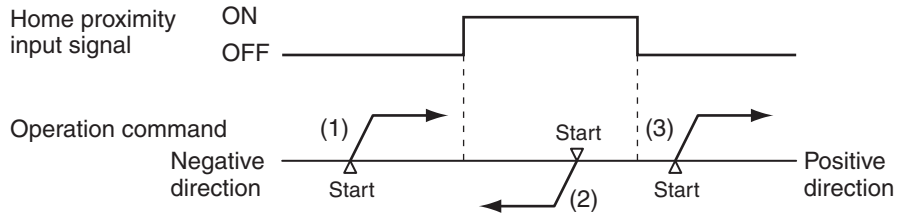
Homing Start Direction

Select the direction (positive or negative) 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 for a proximity reverse turn, 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 proximity reverse turn. These are listed below.

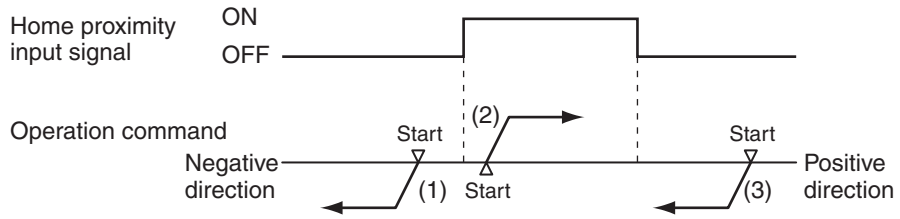
- 0: Proximity reverse turn/home proximity input OFF
- 1: Proximity reverse turn/home proximity input ON
- 9: Proximity reverse turn/home input mask distance
- 12: Proximity reverse turn/holding time

Homing start direction: Positive
Home input detection direction: Positive



(1) and (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.

Homing start direction: Negative
Home input detection direction: Negative



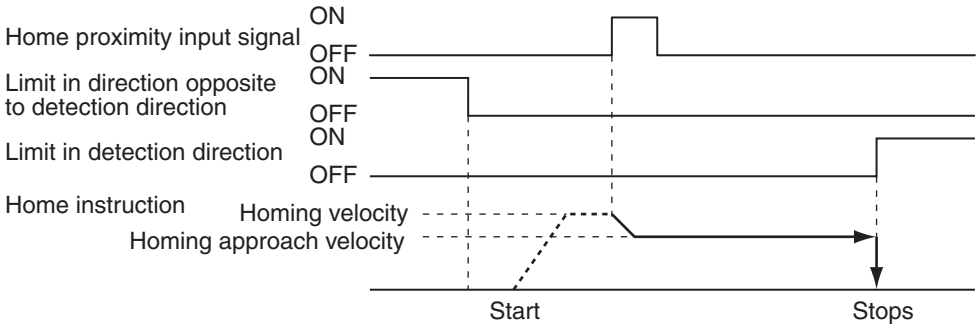
(1) and (3) : The home proximity signal is OFF, so the axis starts moving in the homing start direction.
(2) : The home proximity signal is ON, so the axis starts moving in the direction opposite to the home input detection direction.

Home Input Detection Direction

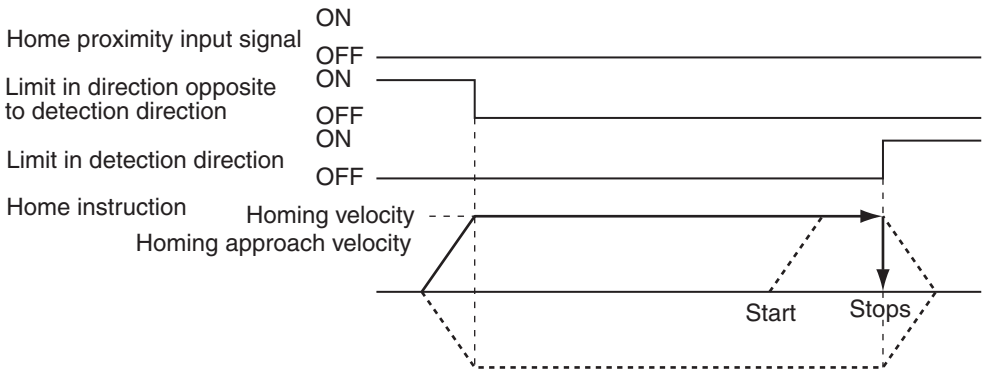
Select the direction (positive or negative) in which to detect home.
Refer to *Homing Start Direction* on page 8-8 for the relationship between the home detection method and the initial direction when homing starts.

Operation Selection at Positive Limit Input and Operation Selection 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 Homing Operation 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** for 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 Velocity

Set the homing velocity in command units per second (command units/s).

Homing Approach Velocity

Set the velocity after the home proximity input turns ON in command units per second (command units/s).

Homing Acceleration

Set the homing acceleration rate in command units per seconds squared (command units/s^2).
If the homing acceleration is set to *0*, the homing velocity or other target velocity is used without any acceleration.

Homing Deceleration

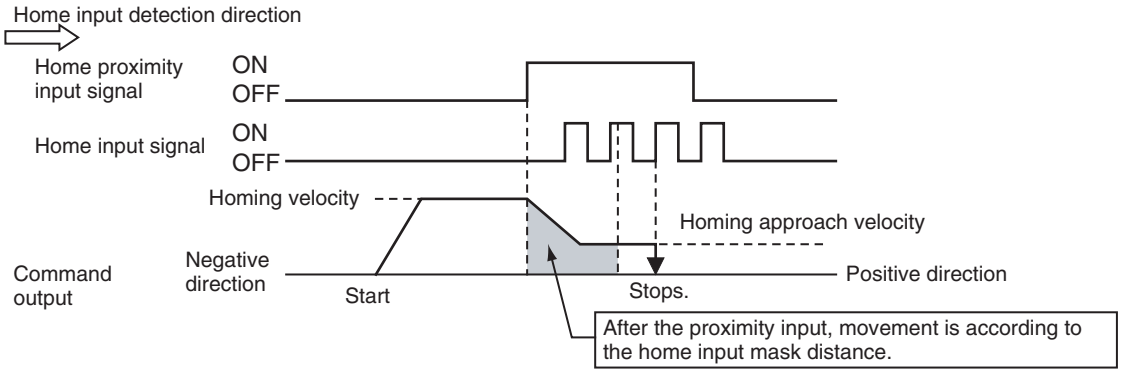
Set the homing deceleration rate in command units per seconds squared (command units/s^2).
If the homing deceleration is set to *0*, the homing approach velocity or other target velocity is used without any deceleration.

Homing Jerk

Set the homing jerk in command units per seconds cubed (command units/s^3).
If the homing jerk is set to *0*, acceleration and deceleration are performed without jerk.

Home Input Mask Distance

Set the home input mask distance in command units when you set Homing Operation Mode **9 (proximity reverse turn/home input mask distance)**.
This is the distance that the axis travels after starting deceleration when the home proximity input signal changes from OFF to ON.



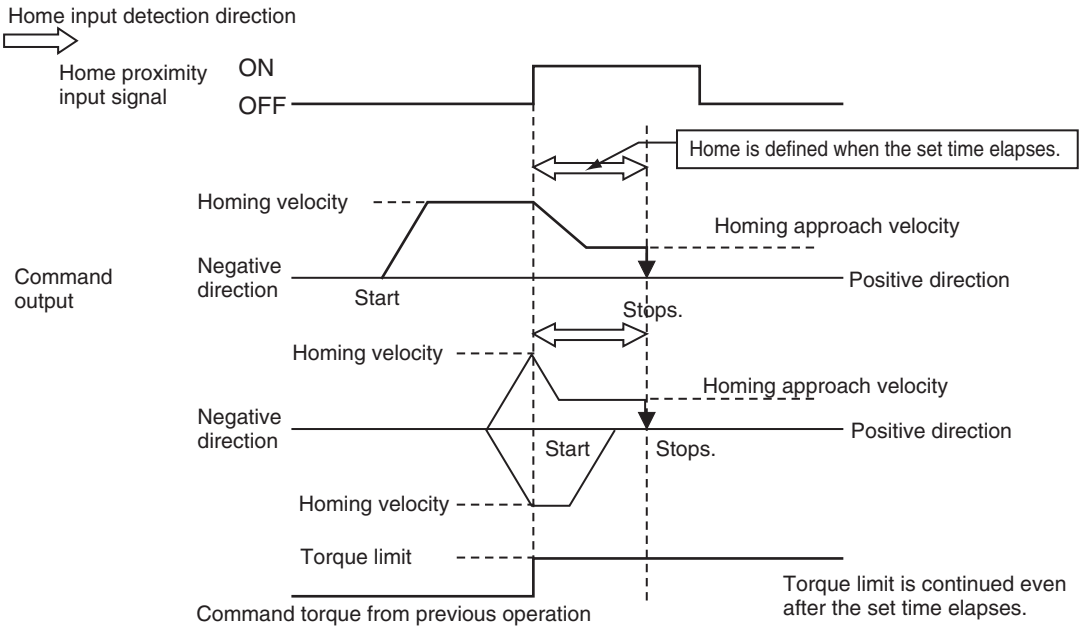
Home Offset

After home is defined, the operation for the homing compensation value is completed if a homing compensation value is set, and then the actual value is preset to the set value.
This means that you can set home to any specified value rather than to *0*.
For systems with absolute encoders, also the absolute encoder home offset value is calculated and saved in the CPU Unit.

Homing Holding Time

Set the holding time in milliseconds when you set Homing Operation Mode **12 (proximity reverse turn/holding time)**.

This is a period of time which elapses from when deceleration is started as the home proximity input signal changes from OFF to ON.



Homing Compensation Value

After home is defined, relative positioning is performed at the set value to adjust the position of home. This homing compensation is performed at the homing compensation velocity.

Adjusting the workpiece is sometimes difficult after home is defined. The homing compensation can be used to fine-tune the position of home after it is determined.

This is useful when you cannot easily replace the home proximity sensor or when home has moved after a motor replacement.

Homing Compensation Velocity

If you set a homing compensation value, set the velocity to use for the compensation in command units per second (command units/s).

8-2-2 Monitoring the Homing Operation

You can read Axis Variables in the user program to monitor the homing status and the input signal status.

In the descriptions, a variable name _MC_AX[*] is used as an example, but the same information applies to _MC1_AX[*] and _MC2_AX[*].

Variable name	Data type	Meaning	Function
MC_AX[0-255].Status.Homing	BOOL	Homing	TRUE when homing for the MC_Home or MC_Home-WithParameter instruction.
_MC_AX[0-255].Dtails.Homed	BOOL	Home Defined	TRUE when home is defined. FALSE: Home not defined TRUE: Home is defined
_MC_AX[0-255].Dtails.InHome	BOOL	In Home Position	TRUE when the axis is in the range for home. It gives an AND of the following conditions. • Home defined. • Actual current position is within the **zero position range** with home as the center. This variable is also TRUE when the zero position is passed by while the axis is moving for a command.
_MC_AX[0-255].DrvStatus.P_OT	BOOL	Positive Limit Input	TRUE when the positive limit input is enabled.
_MC_AX[0-255].DrvStatus.N_OT	BOOL	Negative Limit Input	TRUE when the negative limit input is enabled.
_MC_AX[0-255].DrvStatus.HomeSw	BOOL	Home Proximity Input	TRUE when the home proximity input is enabled.
_MC_AX[0-255].DrvStatus.Home	BOOL	Home Input	TRUE when the home input is enabled.

8-3 Homing Operation

Select the home definition method based on the configuration of the positioning system and its purpose.

There are 10 Homing Operation Modes supported by the MC Function Module.

You can also fine-tune the home that is detected with a homing compensation value.

Additional Information

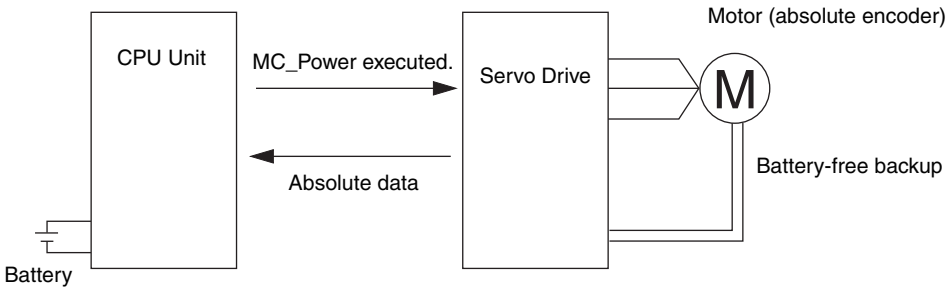
- The most suitable mode depends on the configuration of the positioning system and the application.
  In Linear Mode (finite length), **Proximity Reverse Turn and Home Proximity Input OFF** is normally used if there is a home proximity sensor, positive limit input, and negative limit input.
- The override factors are ignored for homing.
- The in-position check will follow the in-position check settings only for the completion of the home definition and homing compensation motions.
- Buffering and blending are not performed if you use multi-execution of other motion control instructions during homing.

For details on homing, refer to the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)*.

8-4 Homing with an Absolute Encoder

This section describes how to use an OMRON 1S-series Servomotor or G5-series Servomotor with an absolute encoder, or an OMRON G5-series Linear Motor Type Servomotor/Servo Drive with built-in EtherCAT communications that is used with an absolute external scale.

If you use an absolute encoder, the absolute data can be retained even when the power supply to the CPU Unit is turned OFF.
When you execute the MC_Power (Power Servo) instruction, the position is determined by reading the actual position from the absolute encoder.
Unlike when using an incremental encoder, after home is defined, you do not need to perform the homing operation again.



Precautions for Correct Use

- If you use an OMRON 1S-series Servo Drive with a CPU Unit which is not NX502, NX102, or NX1P2, connect the CPU Unit battery.
- If you use an absolute encoder for the OMRON G5-series Servo Drive with an NX502, NX102, or NX1P2 CPU Unit, connect the absolute encoder backup battery of the Servo Drive.
- If you use an absolute encoder for the OMRON G5-series Servo Drive with a CPU Unit which is not NX502, NX102, or NX1P2, connect both the CPU Unit battery and the absolute encoder backup battery of the Servo Drive.
- Always execute the MC_Home or MC_HomeWithParameter instruction to define home when you use the absolute encoder for the first time, after you replace the motor, when you use an OMRON G5-series Servo Drive, when the battery in the absolute encoder expires, or at any other time when the absolute value data is lost.
  After you complete these steps, home is defined when the *Enable* input variable to the MC_Power instruction changes to TRUE.
  For details on homing, refer to *8-1 Outline of Homing* on page 8-2.
  For a CPU Unit with unit version 1.10 or later, home is also defined when EtherCAT process data communications change from a non-established to an established state, in addition to the previously mentioned step.
- For an NX502, NX102, or NX1P2 CPU Unit, if an error occurs for the CPU Unit memory when the power supply is turned OFF and then the absolute value data cannot be saved, an Absolute Encoder Home Offset Read Error (event code: 14600000 hex) will occur when the power is turned ON again.
  If you turn ON the power supply to a CPU Unit which is not NX502, NX102, or NX1P2 while an error is occurring on the CPU Unit battery, an Absolute Encoder Home Offset Read Error (event code: 14600000 hex) will occur.
  You can use the ResetMCError instruction to reset the error and turn ON the Servo.
  For a CPU Unit with unit version 1.10 or later, home will be left undefined until it is defined by the MC_Home or MC_HomeWithParameter instruction executed after error reset.
  In order to operate with correct positions, execute the MC_Home or MC_HomeWithParameter instruction to define the correct home position.
  For a CPU Unit with unit version 1.09 or earlier, when the error is reset and the Servo is turned ON, home is defined with an **absolute encoder home offset** value of 0.
- If the power supply to the CPU Unit is turned OFF, home will become undefined.

Additional Information

If you use an OMRON G5-series Linear Motor Type Servomotor/Servo Drive with built-in EtherCAT communications, you can set the absolute encoder home position. If you use a Linear Motor Type, observe the following points when reading this section.

- A Linear Motor Type does not use an encoder. It uses an external scale, which functions in a similar way.
- “Absolute encoder” in this section means “absolute external scale” for a Linear Motor Type.
- An absolute external scale does not have the rotation data of an absolute encoder. Any rotation data setting procedures that are required for an absolute encoder are not required. A battery to back up the rotation data is also not required.
- Refer to the *G5-series AC Servomotors/Servo Drives with Built-in EtherCAT Communications Linear Motor Type User’s Manual (Cat. No. I577)* for the specifications of Linear Motor Type.

8-4-1 Outline of Function

To define home with an absolute encoder system, the absolute encoder offset compensation is performed when the MC_Power (Power Servo) instruction is executed.

For a CPU Unit with unit version 1.10 or later, home is also defined when EtherCAT process data communications change from a non-established to an established state, in addition to the previously mentioned step.
Home can also be defined by performing a homing operation in the same way as for an incremental encoder.
After home is defined, the difference between the command position and the absolute value data read from absolute encoder is saved in the CPU Unit as the **absolute encoder home offset**.
For details on homing, refer to *8-1 Outline of Homing* on page 8-2.

The **absolute encoder home offset** is also set to the difference (i.e., the offset) between the command position after defining home and the absolute value when the MC_Home or MC_HomeWithParameter instruction is executed.
The MC Function Module saves the **absolute encoder home offset** in the CPU Unit when the power supply to the CPU Unit is turned OFF.

Precautions for Correct Use

- When absolute encoders are used, the **absolute encoder home offset** for each axis is saved to the CPU Unit along with the axis number. The saved offset is lost if the axis number is changed. If you change the axis number, set the Homing Settings again.
- If you replace the CPU Unit or the Battery for the CPU Unit, make sure home is defined and back up the **absolute encoder home offset** before you start the replacement procedure.
- You can restore the backed up data after finishing the replacement procedure to use the home that was previously defined.
- Use the Sysmac Studio to back up and restore the data.
  Refer to the *Sysmac Studio Version 1 Operation Manual (Cat. No. W504)* for specific procedures.

Applicable Servomotors

The following table lists the Servomotors that use the absolute encoder home setting.

Manufacturer	Series	Servo Drive	Servomotor
OMRON	1S Series	R88D-1SN□□□-ECT	R88M-1□□□□□□S
		R88D-1SAN□□□-ECT	R88M-1□□□□□□T R88M-1□□□□□□C
	G5 Series	R88D-KN□□□-ECT	R88M-K□□□□□S R88M-K□□□□□T R88M-K□□□□□C
		R88D-KN□□□-ECT-L	R88L-EC□□□□□□□□

Precautions for Correct Use

You cannot use this absolute encoder for an NX-series Pulse Output Unit.

Connecting the Servo Drive

Connect the Servo Drive correctly according to information in the *NJ/NX-series CPU Unit Built-in EtherCAT Port User's Manual (Cat. No. W505)*.

Additional Information

- Refer to *A-1 Connecting the 1S-series Servo Drive* on page A-2 for setting examples for connection to an OMRON 1S-series Servo Drive.
- Refer to *A-2 Connecting the G5-series Servo Drive* on page A-11 for setting examples for connection to an OMRON G5-series Servo Drive.

8-4-2 Setting Procedure

This section describes the procedure to set the home of an absolute encoder system.

1 Absolute Encoder Setup
Refer to the *Sysmac Studio Version 1 Operation Manual (Cat. No. W504)* for the setup procedures.

2 Setting Axis Parameters
Set the **Encoder Type** in the **Position Count Settings** of the axis parameters in the MC Function Module to **1 (absolute encoder (ABS))**.
For details, refer to *5-2-7 Position Count Settings* on page 5-26.

3 Execute Homing
Execute homing.
Set the **Homing Method** in the **Homing Settings** axis parameters of the MC Function Module.
After home is defined, the difference between the command position and the absolute value data read from the absolute encoder is saved to the CPU Unit in the CPU Unit as the **absolute encoder home offset**.

Absolute Encoder Setup

The absolute encoder must be set up the first time it is used, to initialize the rotation data to 0, when the absolute encoder is stored for an extended period of time without a battery connected, etc.
Refer to the *Sysmac Studio Version 1 Operation Manual (Cat. No. W504)* for detailed setup procedures.

Precautions for Correct Use

After the absolute encoder is set up, the power supply to the OMRON 1S-series Servo Drive or G5-series Servo Drive must be cycled.
When setup processing for the absolute encoder is completed, an Absolute Value Clear Error (A27.1) will occur in the Servo Drive. Cycle the control power supply to the Servo Drive to clear this error and complete the absolute encoder setup procedure.

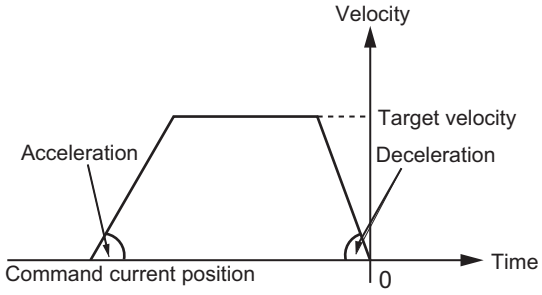
Using an Absolute Encoder in Rotary Mode

If you set the **Count Mode** axis parameter to **Rotary Mode**, the actual position will be a **ring-shaped counter in the range between the modulo maximum position setting value** and the **modulo minimum position setting value**.

When using an absolute encoder in **Rotary Mode**, the **absolute encoder home offset** is automatically calculated and updated in the MC Function Module each motion control period.
The updated **absolute encoder home offset** is saved in the CPU Unit when the power supply to the CPU Unit is turned OFF.
This enables recovering the actual position of a rotating axis from the absolute encoder the next time power is turned ON even if the power to the CPU Unit or Servo Drive is turned OFF.

8-5 High-speed Homing

This function performs quick positioning to the home. Home is defined in advance.
Use the MC_MoveZeroPosition (High-speed Homing) instruction and specify the target velocity, acceleration rate, deceleration rate, and jerk.
If you execute this instruction when home is not defined, an instruction error will occur.



Additional Information

For details on the MC_MoveZeroPosition (High-speed Homing) instruction, refer to the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)*.

9

Motion Control Functions

This section describes the motion control functions that are used when connected to OMRON 1S-series Servo Drives with built-in EtherCAT communications.

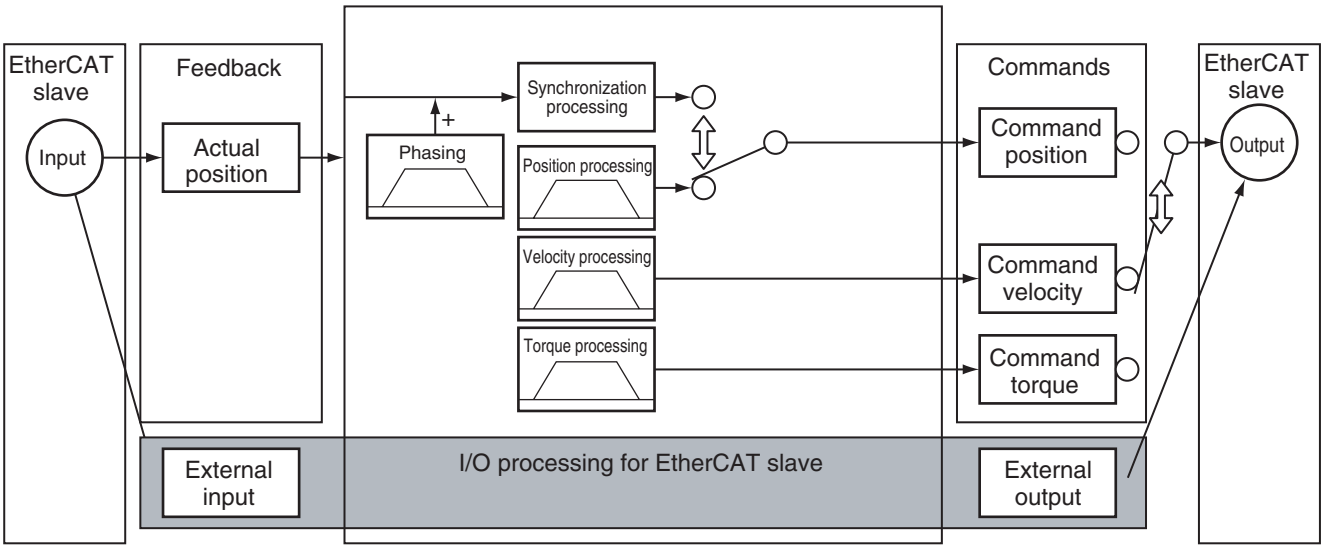
9-1 Single-axis Position Control

The MC Function Module can be connected to OMRON 1S-series Servo Drives with built-in EtherCAT communications or G5-series Servo Drives with built-in EtherCAT communications to implement position control, velocity control, and torque control.
This section describes positioning operation for single axes.

Some of the functions of the MC Function Module are different when NX-series Pulse Output Units are used. Refer to the *NX-series Position Interface Units User's Manual (Cat. No. W524)* for details.

9-1-1 Outline of Operation

The single-axis control function of the MC Function Module consists of control for motion profile commands and synchronized control.
There are three Control Modes for motion profile commands: position control, velocity control, and torque control.
In synchronized control, the slave axis (i.e., the axis being controlled) operates in a synchronized relationship to the master axis, as expressed by a cam profile curve or a gear ratio.
Manual operations such as jogging and homing are also supported.



Note You can use the command position or actual position as the input to the synchronization processing.

Resetting Axis Errors

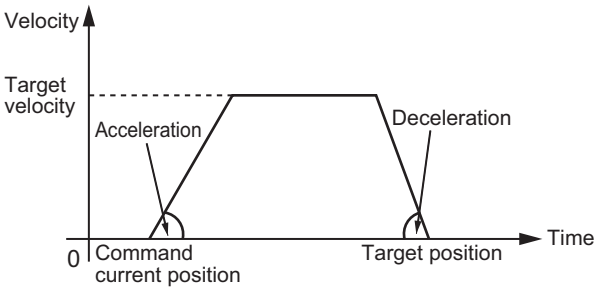
If an error occurs in an axis, you can use the MC_Reset instruction to remove the error once you have eliminated the cause.

For details on resetting axis errors, refer to the MC_Reset (Reset Axis Error) instruction in the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)*.

Refer to the *NX-series Position Interface Units User's Manual (Cat. No. W524)* for the differences when you use NX-series Pulse Output Units.

9-1-2 Absolute Positioning

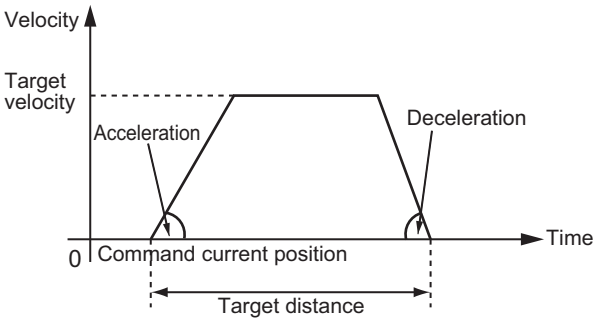
Absolute positioning specifies the absolute coordinates of the target position in relation to home.
You can perform positioning, such as shortest way positioning on a rotary table, by setting the **Count Mode** to **Rotary Mode** and specifying the operation direction.



For details, refer to the MC_MoveAbsolute (Absolute Positioning) and MC_Move (Positioning) instructions in the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)*.

9-1-3 Relative Positioning

Relative positioning specifies the distance from the actual position.
You can specify a travel distance that exceeds the ring counter range by setting the **Count Mode** to **Rotary Mode**.



For details, refer to the MC_MoveRelative (Relative Positioning) and MC_Move (Positioning) instructions in the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)*.

9-1-4 Interrupt Feeding

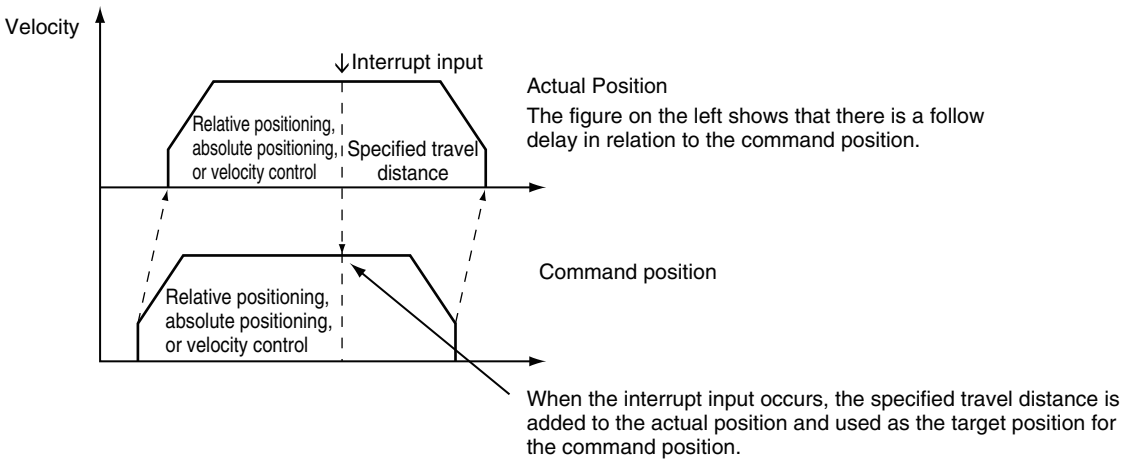
Interrupt feeding feeds the axis at the specified velocity and for the specified distance from the actual position when a trigger signal occurs.
You can also select to output an error if the trigger signal does not occur within the specified travel distance when you specify either absolute or relative positioning.
Feeding is not affected by following error. This is achieved by using the latch function of the Servo Drive to determine the actual position when the trigger signal occurs. You can also use the window function to disable trigger signals that occur outside of a specified position range. For applications such as wrapping machines, this enables feeding only on trigger signals for printed marks on films and eliminates other influences.

Motion Relative to the Actual Position

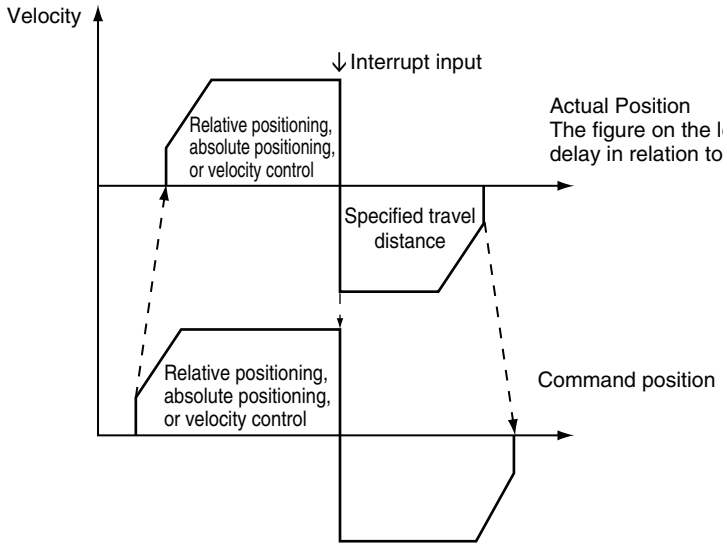
For details, refer to the MC_MoveFeed (Interrupt Feeding) instruction in the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)*.

Refer to the *NX-series Position Interface Units User's Manual (Cat. No. W524)* for the differences when you use NX-series Pulse Output Units.

● Feeding for a Specified Distance in the Moving Direction



● Feeding for a Specified Distance in the Direction Opposite to the Moving Direction

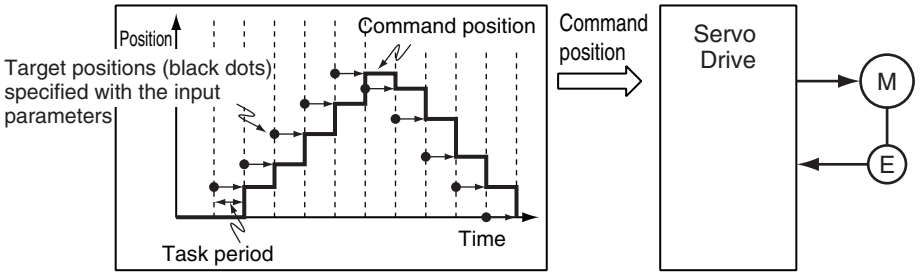


If decelerating to a stop after a reverse turn is specified for the Operation Selection at Reversing axis parameter, an acceleration/deceleration curve is used when reversing.

9-1-5 Cyclic Synchronous Positioning

Cyclic synchronous positioning is used to output a target position to a specified axis each control period in the primary periodic task or a periodic task. The target position is specified as an absolute position.

You can use it to move in a specific path that you create.



For details, refer to the MC_SyncMoveAbsolute (Cyclic Synchronous Absolute Positioning) instruction in the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)*.

Version Information

A CPU Unit with unit version 1.03 or later and Sysmac Studio version 1.04 or higher are required to use cyclic synchronous positioning.

9-1-6 Stopping

Functions to stop axis operation include immediate stop input signal and limit input signals connected to the Servo Drive, stop functions of motion control instructions in the user program, and stopping due to errors.

Refer to the *NX-series Position Interface Units User's Manual (Cat. No. W524)* for the differences when you use NX-series Pulse Output Units.

Stopping for Servo Drive Input Signals

Axis motion is stopped for the immediate stop input signal or a limit input signal from the Servo Drive. You can select the stop method with the Sysmac Studio.

● Immediate Stop Input

Stop processing in the MC Function Module is executed according to the state of the Servo Drive input signals.

You can select one of the following stopping methods for the MC Function Module.

- Immediate stop
- Immediate stop and error reset
- Immediate stop and Servo OFF

Precautions for Correct Use

The immediate stop input for the OMRON 1S-series Servo Drive or G5-series Servo Drive also causes an error and executes stop processes in the Servo Drive itself.

● Limit Inputs (Positive Limit Input or Negative Limit Input)

Stop processing in the MC Function Module is executed according to the state of the Servo Drive input signals.

You can select one of the following stopping methods for the MC Function Module.

- Immediate stop
- Deceleration stop
- Immediate stop and error reset
- Immediate stop and Servo OFF

Precautions for Correct Use

- If a limit input signal turns ON, do not execute an instruction for an axis command of the axis in the same direction as the limit input signal.
- If a limit input signal is ON for any axis in an axes group, do not execute an instruction for an axes group command for that axes group.
- If the signal to decelerate to a stop is input during execution of a synchronous movement instruction that has a *Deceleration* input variable, the axis decelerates to a stop at the deceleration rate given by *Deceleration*.
- If the signal to decelerate to a stop is input during execution of a synchronous movement instruction that does not have a *Deceleration* input variable, the axis decelerates to a stop at the maximum deceleration rate that is set in the axis parameters.

Additional Information

- You must set up the Servo Drive in order to use the input signals from the Servo Drive. An OMRON 1S-series Servo Drive with built-in EtherCAT communications or G5-series Servo Drive with built-in EtherCAT communications has an immediate stop input and limit input assigned in its default settings.
- Refer to *A-1 Connecting the 1S-series Servo Drive* on page A-2 for setting examples for connection to an OMRON 1S-series Servo Drive.
- Refer to *A-2 Connecting the G5-series Servo Drive* on page A-11 for setting examples for connection to an OMRON G5-series Servo Drive.

Stopping with Motion Control Instructions

Use the MC_Stop or MC_ImmediateStop instruction to stop single-axis operation.

For details, refer to the MC_Stop and MC_ImmediateStop instructions in the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)*.

Additional Information

When the input variable *Enable* to the MC_Power (Servo ON) instruction changes to FALSE, the MC Function Module immediately stops the command value and turns OFF the Servo. When the Servo is turned OFF, the Servo Drive will operate according to the settings in the Servo Drive.

● MC_Stop Instruction

You can specify the deceleration rate and jerk for single-axis control and synchronized control to decelerate to a stop.

Specify a deceleration rate of *0* to send a command that immediately stops the Servo Drive.

Other operation commands are not acknowledged while decelerating to a stop for this instruction and while the input variable *Execute* is TRUE.

● MC_ImmediateStop Instruction

You can perform an immediate stop for single-axis control or synchronized control functions.
You can also execute this instruction on axes that are enabled in an axes group.

Stopping Due to Errors or Other Problems

● Stopping for Errors during Single-axis Operation

When an error occurs during single-axis operation, the axis will stop immediately or decelerate to a stop depending on the error.
Refer to the *NJ/NX-series Troubleshooting Manual (Cat. No. W503)* for details on the stop method for each error.

● Stopping for a Software Limit

To stop for a software limit, set the **Software Limits** axis parameter.
You can select from the following stop methods for the software limits.

- Enabled for command position. Decelerate to a stop.
- Enabled for command position. Immediate stop.
- Enabled for actual position. Decelerate to a stop.
- Enabled for actual position. Immediate stop.

Refer to *9-8-5 Software Limits* on page 9-83 for details on software limits.

● Stopping Due to Motion Control Period Exceeded

If motion control processing does not end within two periods, a Motion Control Period Exceeded occurs. All axes stop immediately.

Precautions for Correct Use

When you use an NX701 CPU Unit and operate in the multi-motion, all axes in both tasks will stop immediately if a Motion Control Period Exceeded occurs in either of the tasks.
Refer to *A-6-2 Motion Control* on page A-31 for multi-motion.

● Errors That Cause the Servo to Turn OFF

An immediate stop is performed if an error occurs that causes the Servo to turn OFF.
When the Servo is turned OFF, the Servo Drive will operate according to the settings in the Servo Drive.

● Stopping Due to Start of MC Test Run

All axes will decelerate to a stop at their *maximum deceleration* if a MC Test Run is started from the Sysmac Studio.

● Stopping Due to End of MC Test Run

All axes will decelerate to a stop at their *maximum deceleration* if a MC Test Run is stopped from the Sysmac Studio.

- Click the **Stop MC Test Run** Button on the MC Test Run Tab Page of the Sysmac Studio.
- Close the MC Test Run Tab Page on the Sysmac Studio.
- Exit the Sysmac Studio.

● Stopping Due to Change in CPU Unit Operating Mode

All axes will decelerate to a stop at their *maximum deceleration* if the NJ/NX-series CPU Unit operating mode changes.

Precautions for Correct Use

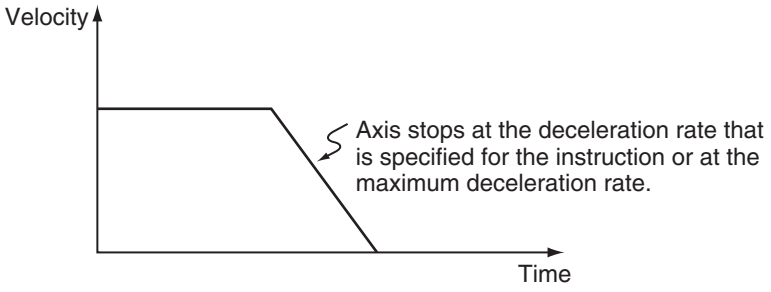
- If an error that results in deceleration to a stop occurs during execution of a synchronous movement instruction that has a *Deceleration* input variable, the axis decelerates to a stop at the deceleration rate given by *Deceleration*.
- If an error that results in deceleration to a stop occurs during execution of a synchronous movement instruction that does not have a *Deceleration* input variable, the axis decelerates to a stop at the maximum deceleration rate that is set in the axis parameters.

Additional Information

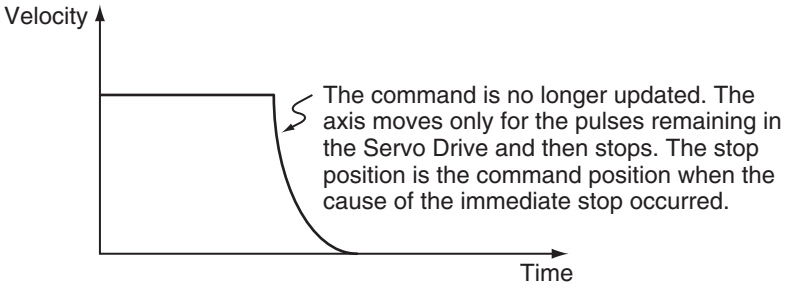
- When RUN mode changes to PROGRAM mode, any motion control instructions for current motions are aborted. The *CommandAborted* output variable from the instructions remains FALSE. The Servo ON/OFF status remains the same even after changing to PROGRAM mode.
- If the operating mode returns to RUN mode while a deceleration stop is in progress after the operating mode changes from RUN to PROGRAM mode, the output variables from motion control instructions are cleared. The *CommandAborted* output variables from the motion control instructions therefore remain FALSE.
- The save process will continue during a save for the MC_SaveCamTable instruction.
- The generation process will continue when generation of the cam table is in progress for the MC_GenerateCamTable (Generate Cam Table) instruction.

Stop Method

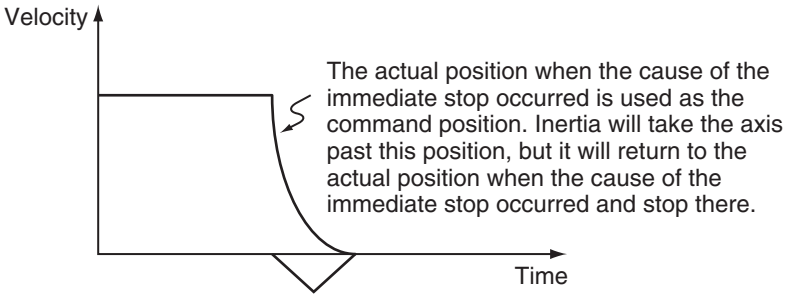
● Deceleration Stop



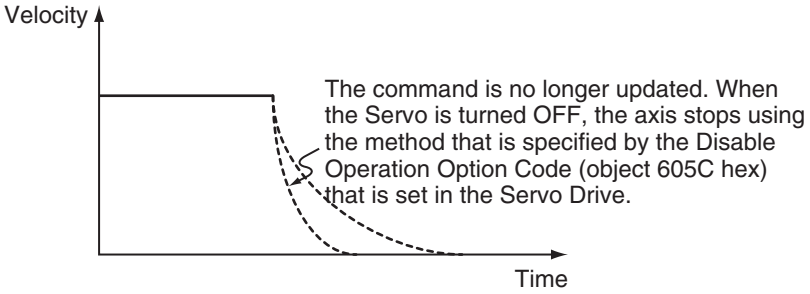
● Immediate Stop



● Immediate Stop and Error Reset



● Immediate Stop and Servo OFF



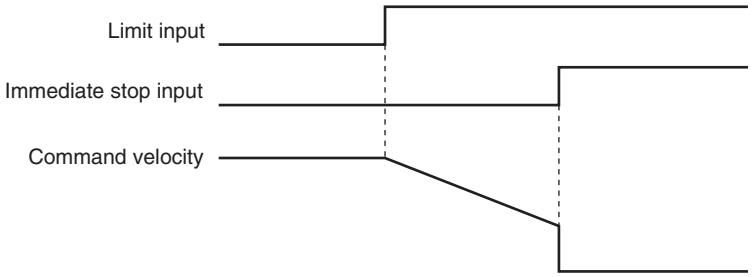
Stop Priorities

The priorities for each stop method are listed in the following table. If a stop with a higher priority stop method occurs while stopping, the stop method will switch to the higher priority method.

Stop method	Priority (higher numbers mean higher priority)
Immediate stop and Servo OFF	4
Immediate stop and error reset	3
Immediate stop	2
Deceleration stop	1

● Example:

The following figure is an example of an immediate stop when the limit input signal is ON and the immediate stop input changes to ON during a deceleration to a stop.



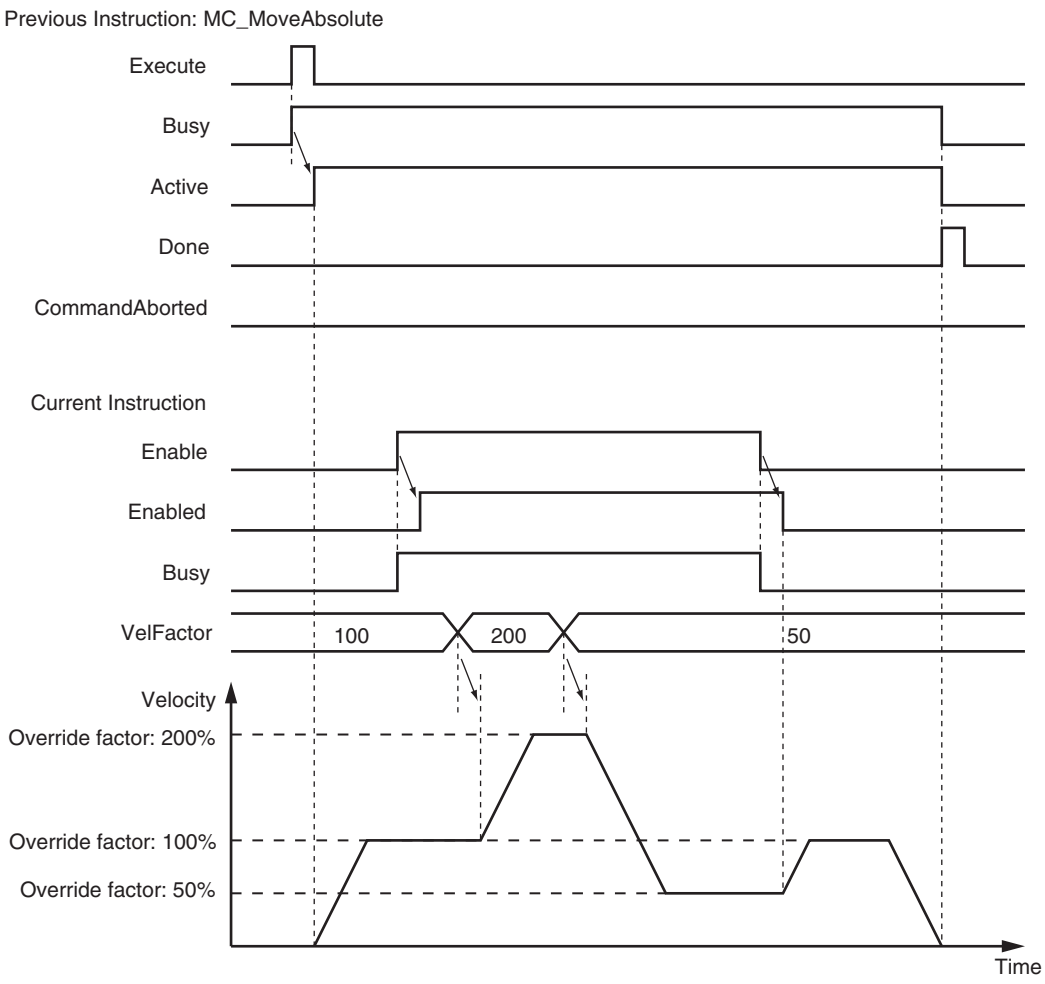
9-1-7 Override Factors

You can use the MC_SetOverride instruction to set override factors for the motion of the axes that are currently in motion.
The velocity override factor is set as a percentage of the target velocity. It can be set between 0% and 500%.
If an override factor of 0% is set for the target velocity, operating status will continue with the axis stopped as a velocity of 0.

The set override factor is read as long as the overrides are enabled.
If the overrides are disabled, the override factors return to 100%.
If the maximum velocity is exceeded when an override factor is changed, the *maximum velocity* for the axis is used.

● 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.



For details, refer to the MC_SetOverride (Set Override Factors) instruction in the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)*.

9-2 Single-axis Synchronized Control

This section describes the operation of synchronized control for single axes.

9-2-1 Overview of Synchronized Control

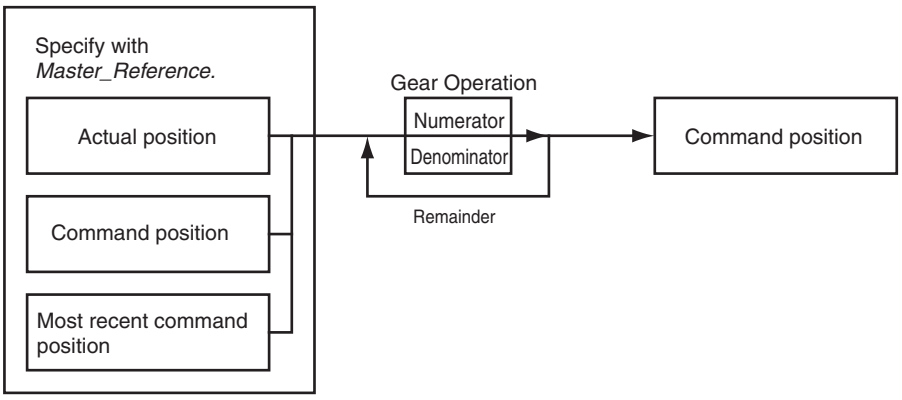
Synchronous control synchronizes the position of a slave axis with the position of a master axis.
The command position or actual position of any axis can be specified for the master axis.
If the command velocity for the slave axis exceeds the **maximum velocity** that is set in the axis parameters, the command is performed at the maximum velocity of the axis. If this occurs, any insufficient travel distance is distributed and output in the following periods.

Precautions for Correct Use

- You cannot specify an encoder axis, virtual encoder axis or single-axis position control axis for the slave axis.
- When you use an NX701 CPU Unit and operate in the multi-motion, assign the master axis and slave axis to the same task.
  If you specify the master axis in a different task from the slave axis by executing the synchronized control instructions such as the MC_GearIn (Start Gear Operation) instruction or the MC_Camin (Start Cam Operation) instruction, an Illegal Master Axis Specification (event code: 54620000 hex) occurs.
  Refer to *9-2-11 Achieving Synchronized Control in Multi-motion* on page 9-30 if you desire to specify the master axis in a different task from the slave axis.

9-2-2 Gear Operation

This function specifies the gear ratio between the master axis and the slave axis and starts operation. Start gear operation with the MC_GearIn (Start Gear Operation) instruction. End synchronization with the MC_GearOut (End Gear Operation) instruction or the MC_Stop instruction.



You can set the gear ratio numerator, gear ratio denominator, position type, acceleration rate, and deceleration rate for the slave axis to operate. For the master axis, you can specify the command position, actual position, or most recent command position.

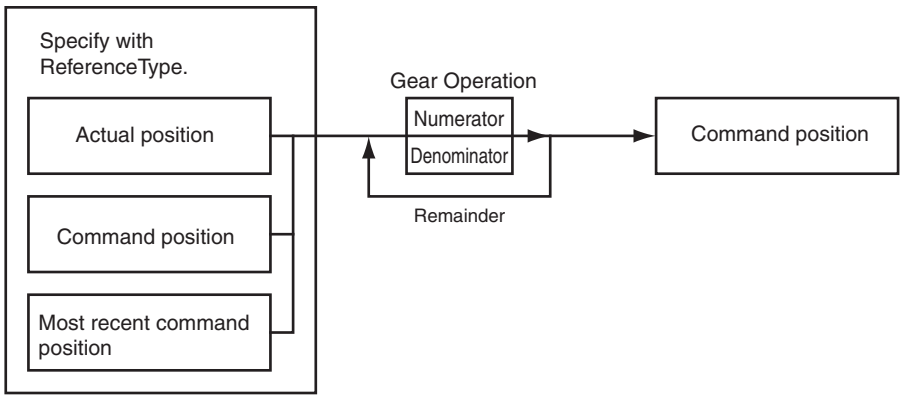
After operation starts, the slave axis uses the velocity of the master axis times the gear ratio for its target velocity, and accelerates/decelerates accordingly.
The catching phase exists until the target velocity is reached. The InGear phase exists after that.

If the gear ratio is positive, the slave axis and master axis move in the same direction. If the gear ratio is negative, the slave axis and master axis move in the opposite directions.

For details on gear operation, refer to the MC_GearIn (Start Gear Operation), MC_GearOut (End Gear Operation), and MC_Stop instructions in the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)*.

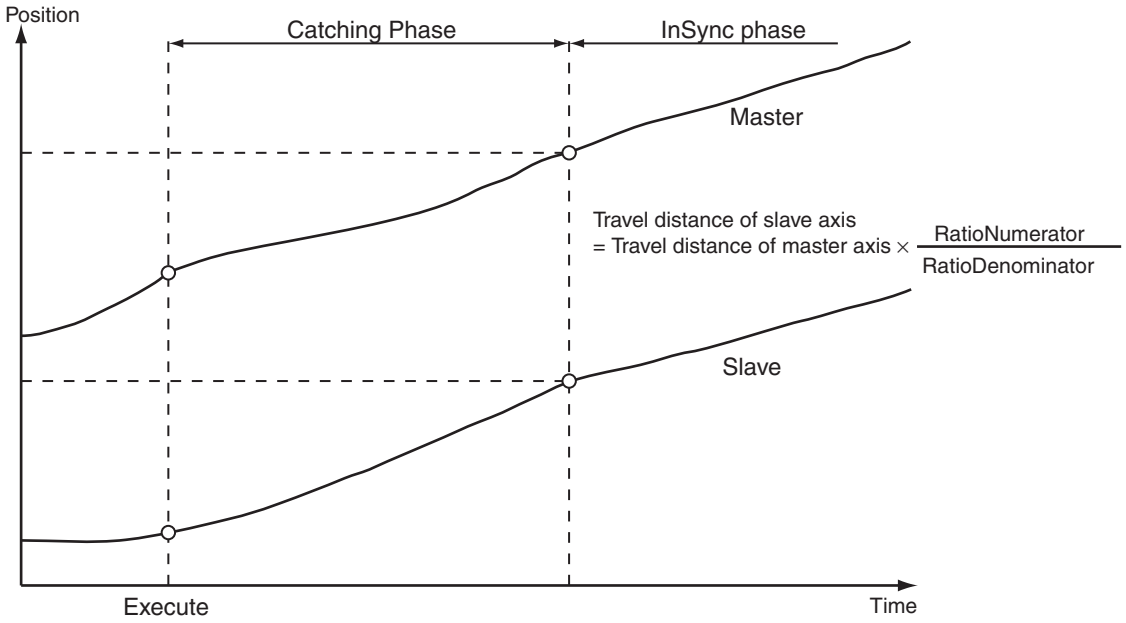
9-2-3 Positioning Gear Operation

This function specifies the gear ratio between the master axis and the slave axis and starts operation. Positioning gear operation allows you to set the positions of the master and slave axes at which to start synchronization.
Start positioning gear operation with the MC_GearInPos (Positioning Gear Operation) or MC_GearInPos2 (Positioning Gear Operation2) instruction. End synchronization with the MC_GearOut (End Gear Operation) instruction or the MC_Stop instruction.



You can set the gear ratio numerator, gear ratio denominator, position type, acceleration rate, and deceleration rate for the slave axis to operate. 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 times the gear ratio for its target velocity, and accelerates/decelerates accordingly.
The slave axis is in the catching phase until it reaches the slave sync position. The slave axis enters the *InSync* phase after it reaches the slave sync position. For either, the position of the slave axis is synchronized with the master axis.
If the gear ratio is positive, the slave axis and master axis move in the same direction. If the gear ratio is negative, the slave axis and master axis move in the opposite directions. The following figure shows the operation when the gear ratio is positive.
The MC_GearInPos2 (Positioning Gear Operation2) instruction can be executed even while the slave axis is in motion. In that case, the slave axis velocity is maintained until the slave axis enters the catching phase. At this time, there is no synchronized relationship between the master axis and the slave axis. Therefore, even if the master axis stops, the slave axis does not stop.



For details on positioning gear operation, refer to the MC_GearInPos (Positioning Gear Operation), MC_GearInPos2 (Positioning Gear Operation2), MC_GearOut (End Gear Operation), and MC_Stop instructions in the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)*.

9-2-4 Cam Operation

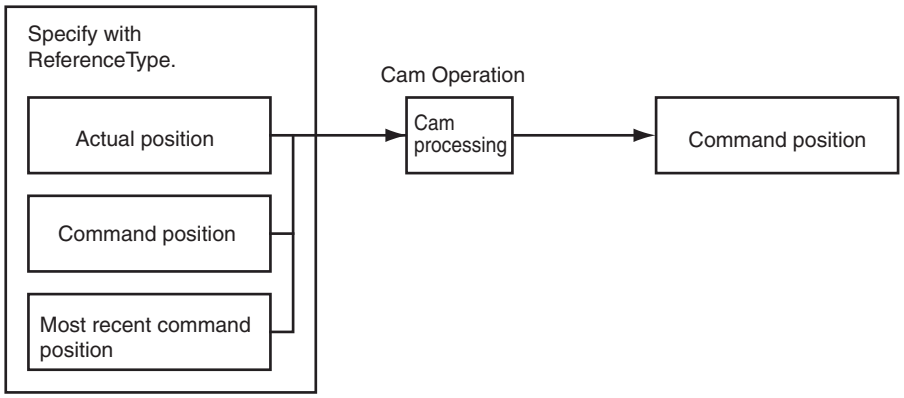
Cam operation synchronizes the position of the slave axis with the master axis according to a cam table or cam definition variable.

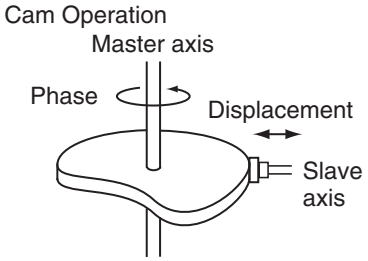
Start cam operation with the MC_CamIn (Start Cam Operation) instruction or the MC_CamInCurve (Start Cam Operation With Specified Curve) instruction. End cam operation with the MC_CamOut (End Cam Operation) instruction or the MC_Stop instruction.

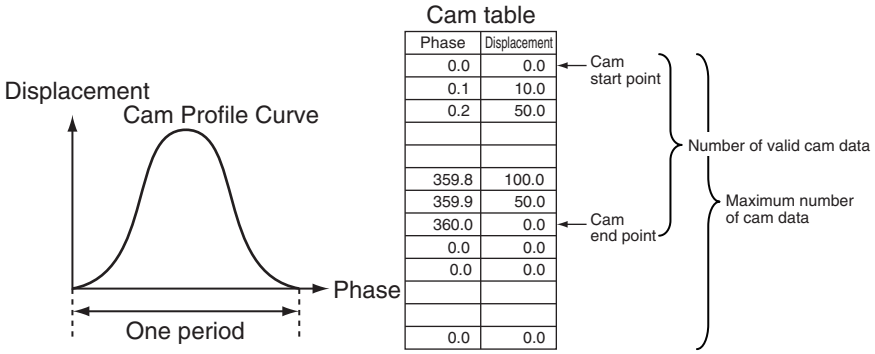
To monitor information on cam operation, use the MC_CamMonitor (Cam Monitor) instruction for the MC_CamIn (Start Cam Operation) instruction, and the MC_CamMonitorCurve (Cam Monitor With Specified Curve) instruction for the MC_CamInCurve (Start Cam Operation With Specified Curve) instruction.

The MC_CamIn (Start Cam Operation) instruction uses a cam table to start cam operation. Create a cam table using the Sysmac Studio Cam Editor and download it to the CPU Unit.

Use the Synchronization menu command of the Sysmac Studio to download the project to the CPU Unit.







In a combination of a CPU Unit with unit version 1.06 or later and Sysmac Studio version 1.07 or higher, the following operation is possible: if another MC_CamIn (Start Cam Operation) instruction is executed by using multi-execution with the Buffer Mode set for blending while the current MC_CamIn (Start Cam Operation) instruction is executed, the operation can continue using the switched cam table and the slave axis does not stop.

The MC_CamInCurve (Start Cam Operation With Specified Curve) instruction uses a cam definition variable, instead of a cam table, to start cam operation.
For details on cam operation, refer to the MC_CamIn (Start Cam Operation), MC_CamInCurve (Start Cam Operation With Specified Curve), MC_CamOut (End Cam Operation), and MC_Stop instructions in the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)*.
For details on the Cam Editor, refer to the *Sysmac Studio Version 1 Operation Manual (Cat. No. W504)*.

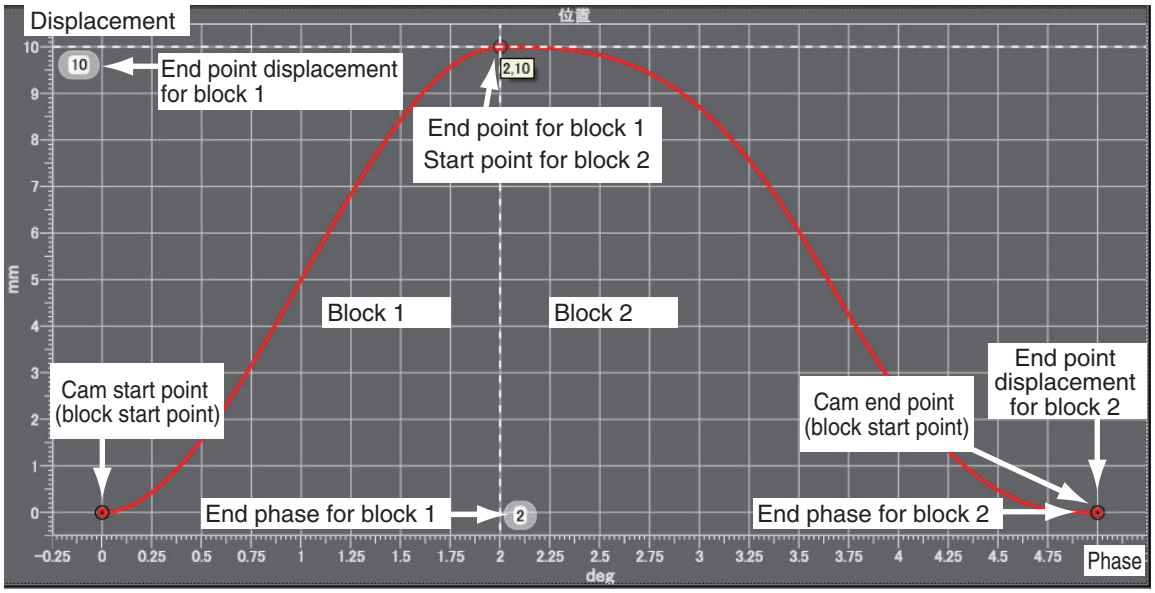
9-2-5 Cam Tables and Start Cam Operation Instruction

This section describes the cam tables that are used for cam operation.

Cam Table Terminology

Term	Description
cam operation	An operation that takes one master axis and one slave axis and follows the cam profile curve to derive the displacement of the slave axis from the phase of the master axis.
cam profile curve	A curve that shows the relationship between phases and displacements in a cam operation. The cam profile curve is created using the Sysmac Studio Cam Editor. You can download the cam profile curve to the CPU Unit and use it as a cam data variable or cam definition variable in your user program. Use the Synchronization menu command of the Sysmac Studio to download the project to the CPU Unit.
cam block	You can select a cam curve in this block. It represents the area between the end point of the previous cam block and the end point of the current cam block.
cam curve	A curve that represents the cam characteristics. You can select a cam curve for each cam block. The Sysmac Studio calculates the phase widths and displacement widths from the specified points and creates the actual cam profile curve. You can choose from different curves, such as straight line, parabolic, and trapecloid.
cam data	Data made up of phases (master axis) and displacements (slave axis) for cam operation.
cam data variable	A variable that represents the cam data as a structure array.
cam table	A data table that contains cam data. If phase data is not in ascending order the cam table is treated as an illegal cam table.
cam start point	The first point in the cam data.
cam end point	The last point of valid cam data in the cam data. If the cam end point is less than the number of cam data, all phases and displacements after the cam end point will be 0.
cam block start point	The start point for a cam block. It is the same as the cam start point at the start of the cam operation. If the cam profile curve continues, this will be the same as the cam block end point.
cam block end point	The end point for a cam block. It is the same as the *cam end point* at the end of the cam operation. If the cam profile curve continues, this will be the same as the *cam block start point*. The *cam block end point* is defined as (horizontal axis, vertical axis) = (phase end point, displacement end point).
original cam data	Cam data that is created by dividing up the cam profile curve in the Cam Editor.
program-modified cam data	The cam data changed by the user program while the CPU Unit is in operation.
master axis	The axis that serves as the input to the cam operation. You can specify either **Linear Mode** or **Rotary Mode**.
slave axis	The axis that serves as the output from the cam operation. You can specify either **Linear Mode** or **Rotary Mode**.
phase	The relative distance on the master axis from the start point of the cam table.

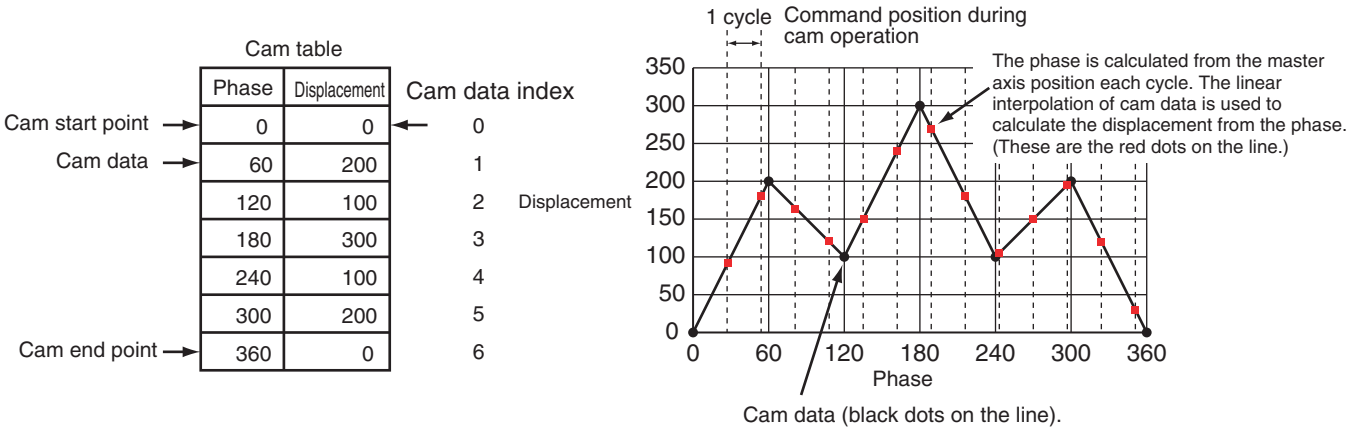
Term	Description
displacement	The relative distance on the slave axis from the master following distance.
valid cam data	The cam data other than the cam start point and other than data where the phase is 0.
invalid cam data	The cam data other than the cam start point where the phase is 0.
number of valid cam data	The number of sets of cam data.
maximum number of cam data	The maximum number of sets of cam data that the cam table can contain.
cam data index	The number of the cam data that is executed.
cam table start position	The absolute position of the master axis that corresponds to the cam start point (phase = 0).
master following distance	The master start distance where the slave axis starts cam operation represented as either an absolute position or relative position. The relative position is based on the cam start point position.
start mode	A specification of whether to represent the master following distance as an absolute position or relative position.
null cam data	Cam data that can be set after the end point where the phase and displacement are 0.
connecting velocity	The connecting velocity that is used to connect cam profile curves. The connecting velocity cannot be specified for some curves.
connecting acceleration	The acceleration rate that is used to connect cam profile curves. The connecting acceleration cannot be specified for some curves.
phase pitch	The width when dividing the cam profile curve by phases (horizontal axis). The points after dividing the curve into the phase pitch correspond to the cam data in the cam table. You must specify the phase pitch for each cam block.



Cam Tables

The MC Function Module defines a single element of data consisting of the phase of the master axis and the displacement of the slave axis as one cam data. A cam table is defined as the combination of multiple sets of cam data.

The cam table is created with the Cam Editor in the Sysmac Studio. You can modify cam data in the cam table from the user program.
The phases and displacements in the cam data that makes up the cam table are represented as relative distances from the start point 0.0.
During cam operation, the command position sent to the slave axis is the displacement determined by interpolating linearly between the two cam data elements adjacent to the phase of the master axis.
The more cam data there is in the cam table, the more accurate the trajectory and the smoother the cam profile curve will be.



Precautions for Correct Use

- Make sure that the cam data is arranged in the cam table so that the phases are in ascending order. An instruction error occurs if a cam operation instruction is executed when 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 in a task that does not control the variable. An Incorrect Cam Table Specification (event code: 54390000 hex) will occur.

Cam Table Specifications

Item	Description
Maximum number of cam data per cam table	65,535 points
Maximum size of all cam data	1,048,560 points*1
Maximum number of cam tables	640 tables*2
Switching cam operation	You can switch to a different cam operation by executing a motion control instruction.
Changing cam data	Cam data can be edited from the user program. Cam data can be overwritten with the Generate Cam Table instruction.*3
Saving cam data	Cam data can be saved to non-volatile memory by using the Save Cam Table instruction.

Item	Description
Information attached to the cam data	Information can be downloaded or uploaded for display in the Cam Editor*4
Timing to load cam data to main memory	• When the data is downloaded from the Sysmac Studio • When power is turned ON

*1. If 65,535 points are used for each cam table, there will be a maximum of 16 cams. A resolution of 0.1° allows for a maximum of 3,600 points per cam table for a maximum of 291 cams.
*2. The total size is 10 MB max.
*3. A CPU Unit with unit version 1.08 or later and Sysmac Studio version 1.09 or higher are required to use the Generate Cam Table instruction.
*4. Use the Synchronization menu command of the Sysmac Studio to upload and download the project.

Data Type of Cam Tables

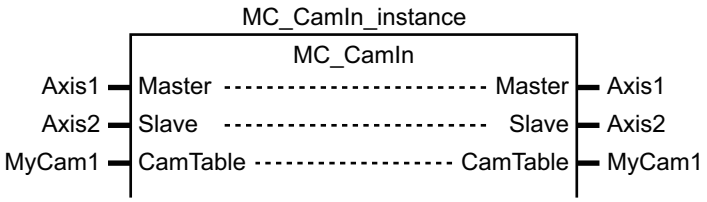
A cam table is declared as an array of cam data structures. The type declaration for the cam data structure is shown below.

```
TYPE
  (*Cam data structure*)
  _sMC_CAM_REF :
  STRUCT
    Phase      : REAL;   (*Phase*)
    Distance   : REAL;   (*Displacement*)
  END_STRUCT;
END_TYPE
```

You must create the cam data with the Cam Editor in the Sysmac Studio and then specify the name of the cam table and the number of cam data (i.e., the size of the array).
For example, to make a cam table called *MyCam1* with 1,000 points use the following declaration.

```
VAR
  (*Cam table*)
  MyCam1   :   ARRAY [0..999] OF _sMC_CAM_REF;
END_VAR
```

The following notation is used to specify *MyCam1* for a cam operation instruction. In this example, the master axis is Axis1 and the slave axis is Axis2.

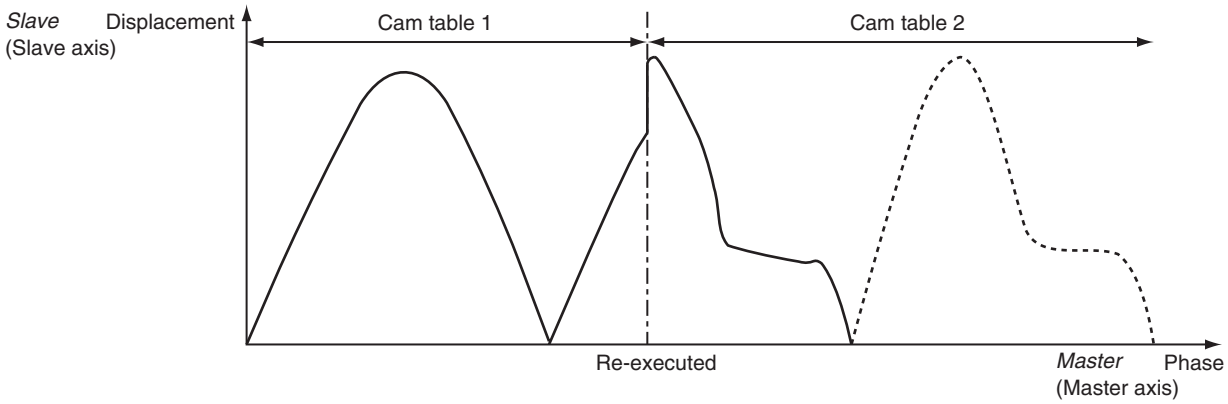


An error will occur if the specified cam table does not exist in the Controller. You can also specify the same cam table for more than one axis.

Switching Cam Tables

You can switch cam tables by re-executing the cam operation instruction during cam operation.

After switching, cam operation will be performed with the cam table you specified for re-execution of the instruction.
The *EndOfProfile* and *Index* output variables from the MC_CamIn instruction are output according to the new cam table.



Precautions for Correct Use

- The cam table you want to switch to must be saved to non-volatile memory before it can be used.
- 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 before or after the cam table is switched. Since this may affect the mechanical machine, be sure to thoroughly verify that there will be no excessive changes in velocity or acceleration before switching the cam table.

Loading/Saving Cam Data and Saving Cam Tables

Cam data can be loaded and saved from the user program just like any other variables.
For example, you can use *MyCam1[0].Phase* to specify the phase and *MyCam1[0].Distance* to specify the displacement in the first array elements of a cam table named*MyCam1*.
Cam data overwritten from the user program can be saved to the non-volatile memory in the CPU Unit as a cam table by executing the MC_SaveCamTable instruction.

Precautions for Correct Use

- Overwritten cam data will be lost if the CPU Unit is turned OFF or the cam data is downloaded from the Sysmac Studio before the Save Cam Table instruction is executed or if the instruction fails to save the data for any reason.
- Be careful not to lose the overwritten data when overwriting cam data from the user program in the CPU Unit.
- Cam data saved to non-volatile memory can be loaded by using the upload function of the Sysmac Studio.
- Use the Synchronization menu command of the Sysmac Studio to upload and download the project.

For details on arrays, refer to the *NJ/NX-series CPU Unit Software User's Manual (Cat. No. W501)*.

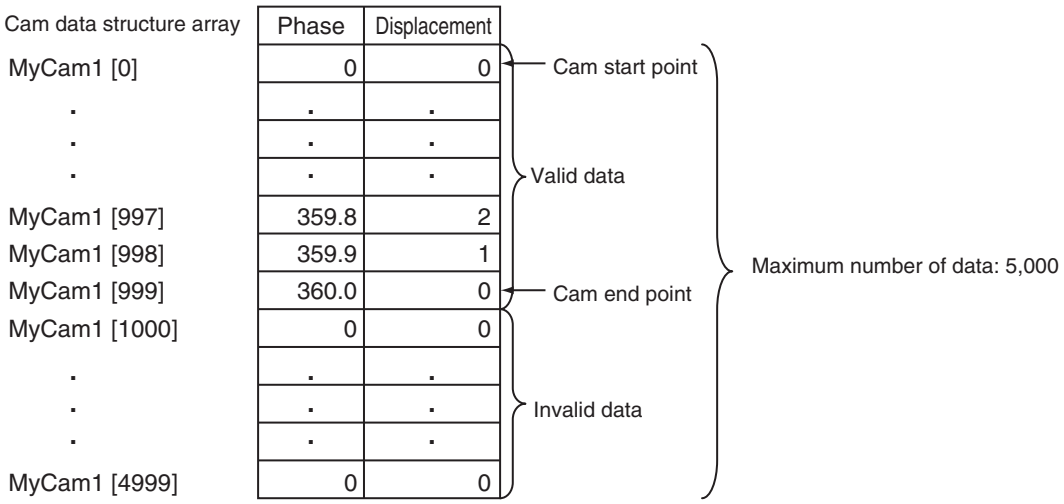
For details on the Save Cam Table instruction, refer to the MC_SaveCamTable instruction in the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)*.

Setting Cam Table Properties and Cam Table End Point Index

The MC Function Module must identify the cam end point of the cam table.
If an overwrite is performed from the user program during cam operation and the number of valid cam data changes, you must update the number of valid cam data to the latest value.
To do so, use the MC_SetCamTableProperty or MC_SetCamTableEndPointIndex instruction.
Although both of these instructions perform the same operation and produce the same result for the cam table, the MC_SetCamTableEndPointIndex instruction allows you to adjust the task period required from the execution to completion of the instruction.

The cam 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. All cam data after phase 0 is detected will be invalid.

For example, in the case of the following cam table, the output variable *EndPointIndex* (end point index) of the instruction is 999 and the *MaxDataNumber* (maximum number of data) is 5,000.



Precautions for Correct Use

- You cannot change the maximum number of cam data from the user program.
- Execute this instruction after overwriting the cam data in any way that changes the number of valid cam data.
  If the number of valid cam data is not updated, the cam operation and the operation of the EndOfProfile (End of Cam Cycle) of the MC_CamIn instruction may not be as expected.
- If you execute an instruction that uses the processing target cam table while this instruction is being processed, an unintended execution result may occur.
  Make sure that this instruction is completed before you execute an instruction that uses the cam table.

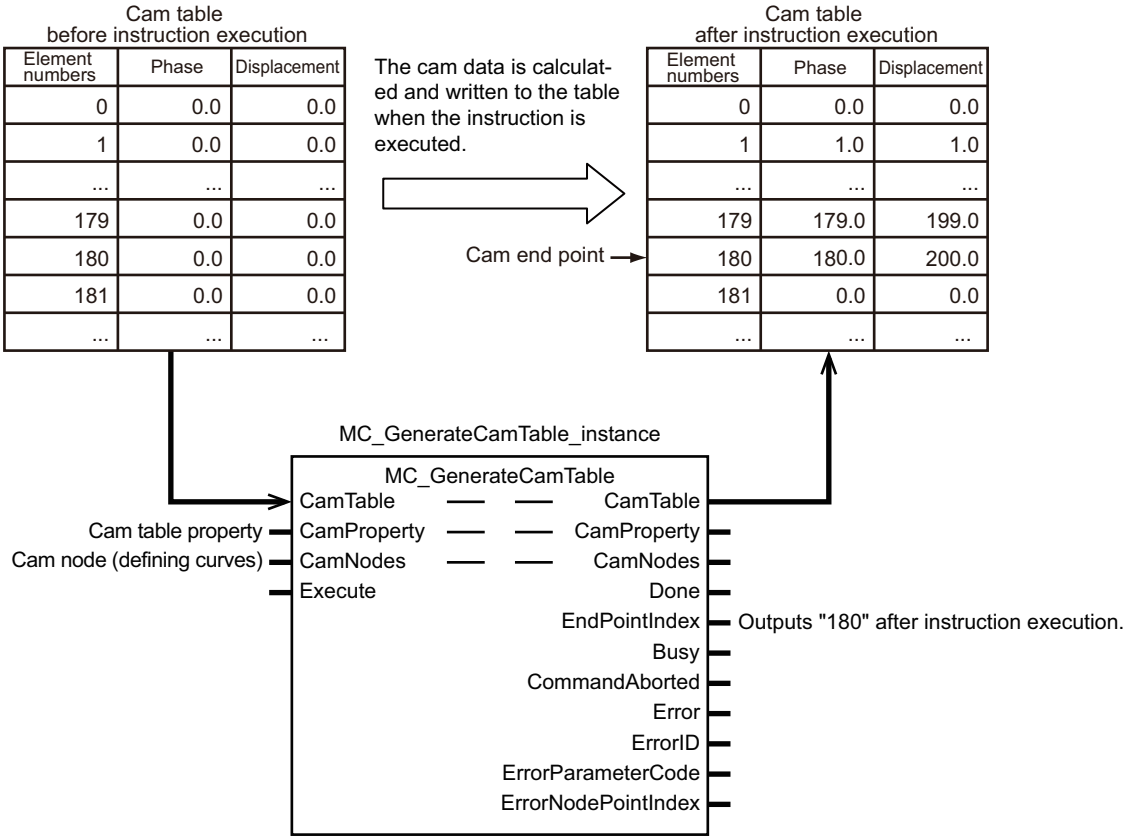
For details on the Set Cam Table Properties instruction, refer to the MC_SetCamTableProperty instruction in the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)*.
For details on the processing time required from the execution to completion of the instruction, refer to the MC_SetCamTableEndPointIndex instruction in the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)*.

Generating Cam Table

With a CPU Unit with unit version of 1.08 or later and the Sysmac Studio version 1.09 or higher, you can generate the cam table by executing the MC_GenerateCamTable (Generate Cam Table) instruction.

The MC_GenerateCamTable instruction calculates the cam data using the values specified for CamProperty (Cam Properties) and CamNodes (Cam Nodes), and rewrites the cam data variable specified for the CamTable (Cam Table) in-out variable.

When rewriting is completed, the MC_GenerateCamTable instruction updates the end point index of the cam table and outputs the element number of the cam end point to EndPointIndex (End Point Index).
It is not necessary to execute the MC_SetCamTableProperty or MC_SetCamTableEndPointIndex instruction after the MC_GenerateCamTable instruction is completed.



The cam data variable is an array variable with the data type of cam data structure _sMC_CAM_REF. You create the cam data variable on the Cam Editor of the Sysmac Studio.
For CamProperty, specify the cam property variable. The cam property variable is an array variable with the data type of cam property structure _sMC_CAM_PROPERTY. You create the cam property variable as a user-defined variable on the global variable table of the Sysmac Studio. Or, you create the variable using the cam data settings on the Sysmac Studio.
For CamNodes, specify the cam node variable. The cam node variable is an array variable with the data type of cam node structure _sMC_CAM_NODE. You create the cam node variable as a user-defined variable on the global variable table of the Sysmac Studio. Or, you create the variable using the cam data settings on the Sysmac Studio.

The cam property variable and the cam node variable are collectively called "cam definition variable".

If the cam definition variable is created as a user-defined variable, the default of its Retain attribute is Non-retain. You must set the Retain attribute of variable to Retain, if you want to reuse the variable after changing its value and switching the operating mode to PROGRAM mode or cycling the power supply. If you set the variable each time of use from the HMI, etc., the attribute can be left Non-retain. If the cam definition variable is created with the cam data settings on the Sysmac Studio, the Retain attribute of variable will be fixed to Retain.

By using the HMI, etc., to set the values for the MC_GenerateCamTable instruction, you can create the cam data variable and adjust the cam operation without using the Sysmac Studio.
The following is the procedure used to adjust the cam operation.

1 Create a user program, in advance, that includes the following processing.
- Assigning the value of the cam definition variable that is set from the HMI to the Generate Cam Table instruction.
- Displaying the cam variable that is created by the Generate Cam Table instruction graphically on the HMI.
- Displaying the value of EndPointIndex (End Point Index) on the HMI.

2 Set the value of the cam definition variable from the HMI.

3 Execute the Generate Cam Table instruction.

4 Verify the curve shape of the generated cam table and the value of the end point index displayed on the HMI.

5 If there is no problem with the curve shape of the cam table and the number of the cam data, then execute the cam operation.

6 Verify the result of the cam operation and consider changing the value of the cam definition variable.

7 Repeat steps 2 to 6.

For details on the cam definition variable and the Generate Cam Table instruction, refer to the MC_GenerateCamTable instruction in the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)*.

Refer to the *Sysmac Studio Version 1 Operation Manual (Cat. No. W504)* for information on creating and transferring the cam definition variables using the Sysmac Studio.

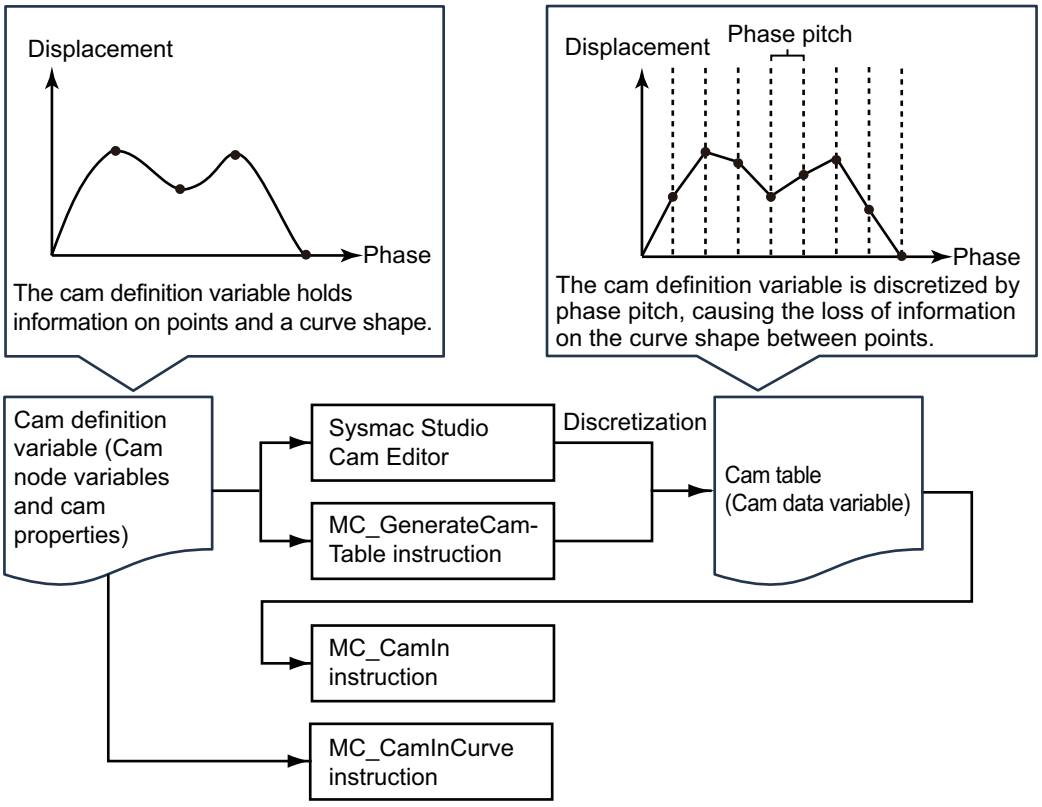
9-2-6 Cam Definition Variables and Start Cam Operation With Specified Curve Instruction

The combination of a CPU Unit version 1.70 or later and Sysmac Studio version 1.66 or higher enables the MC_CamInCurve (Start Cam Operation With Specified Curve) instruction to reference cam definition variables and start cam operation.

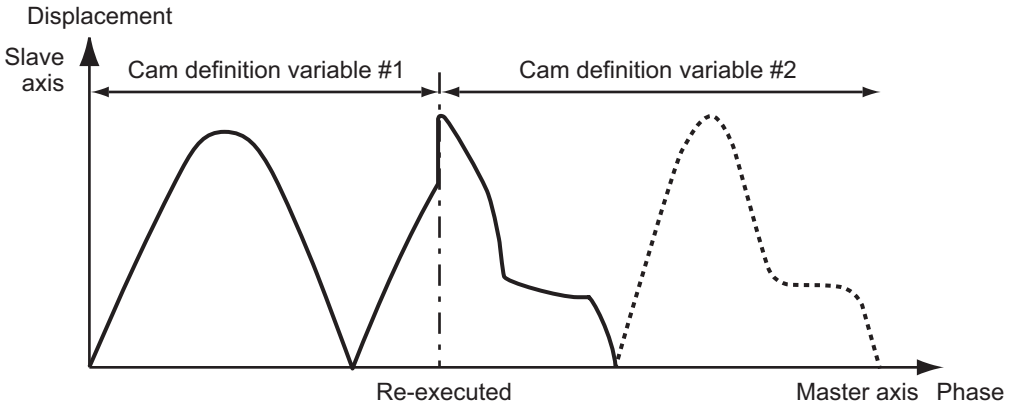
Cam definition variables contain data on points and a curve shape between the points with respect to phase and displacement. The Sysmac Studio Cam Editor or the MC_GenerateCamTable instruction discretizes the cam definition variable data into a cam table for use with the MC_CamIn instruction. The MC_CamInCurve (Start Cam Operation With Specified Curve) instruction provides the following improvements since it uses the cam definition variable as an input variable.

- There is no need to wait for completion of cam table generation by the MC_GenerateCamTable (Generate Cam Table) instruction.
- More precise slave axis command values can be output because cam definition variables are not discretized by phase pitch.

This instruction identifies the end point of the cam definition variable when it is executed or re-executed, eliminating the need to execute the Set Cam Table Properties instruction or the Cam Table End Point Index instruction.
To change the value of the cam definition variable in the user program after executing this instruction, re-execute it.



You can switch the cam definition variable by re-executing the instruction during cam operation. After switching, cam operation will be performed with the cam definition variable that you specified when re-executing the instruction. The *EndOfProfile* and *NodePointIndex* output variables of this instruction are output according to the new cam definition variable.

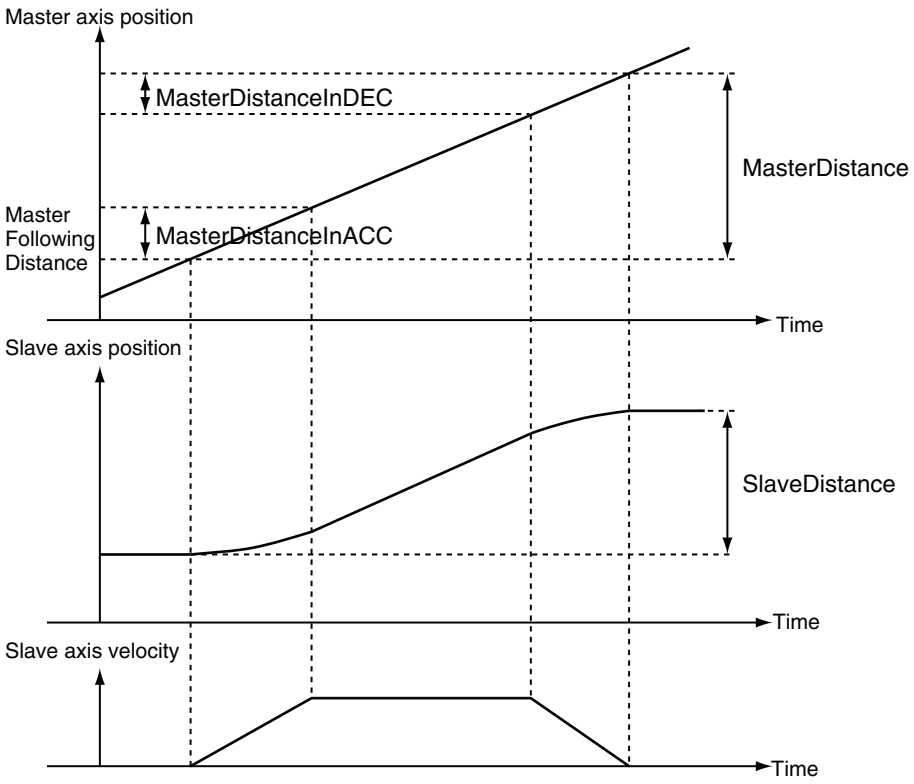


For details on the MC_CamInCurve (Start Cam Operation With Specified Curve) instruction, refer to the MC_CamInCurve (Start Cam Operation With Specified Curve) instruction in the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)*.

9-2-7 Synchronous Positioning

This function performs positioning using a trapezoidal curve while synchronizing the specified slave axis to the specified master axis.
This is a type of electronic cam, but it does not use cam tables created in the Cam Editor.
Operation starts when the MC_MoveLink (Synchronous Positioning) instruction is executed.
Use the MC_Stop instruction to stop the axes in motion.
Operation is performed for the *Slave* (Slave Axis) and the following are set: *Master* (Master Axis), *MasterDistance* (Master Axis Travel Distance), *MasterDistanceInACC* (Master Distance In Acceleration), *MasterDistanceInDEC* (Master Distance In Deceleration), *SlaveDistance* (Slave Axis Travel Distance), and *MasterStartDistance* (Master Following Distance).
The command position or actual position can be specified for the master axis.
You can specify one of the following as the start condition for synchronous operation: **start of instruction**, **when trigger is detected**, or **when master axis reaches the master following distance**.

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 sync start position shown in the following figure represents the position where the sync start condition is met.

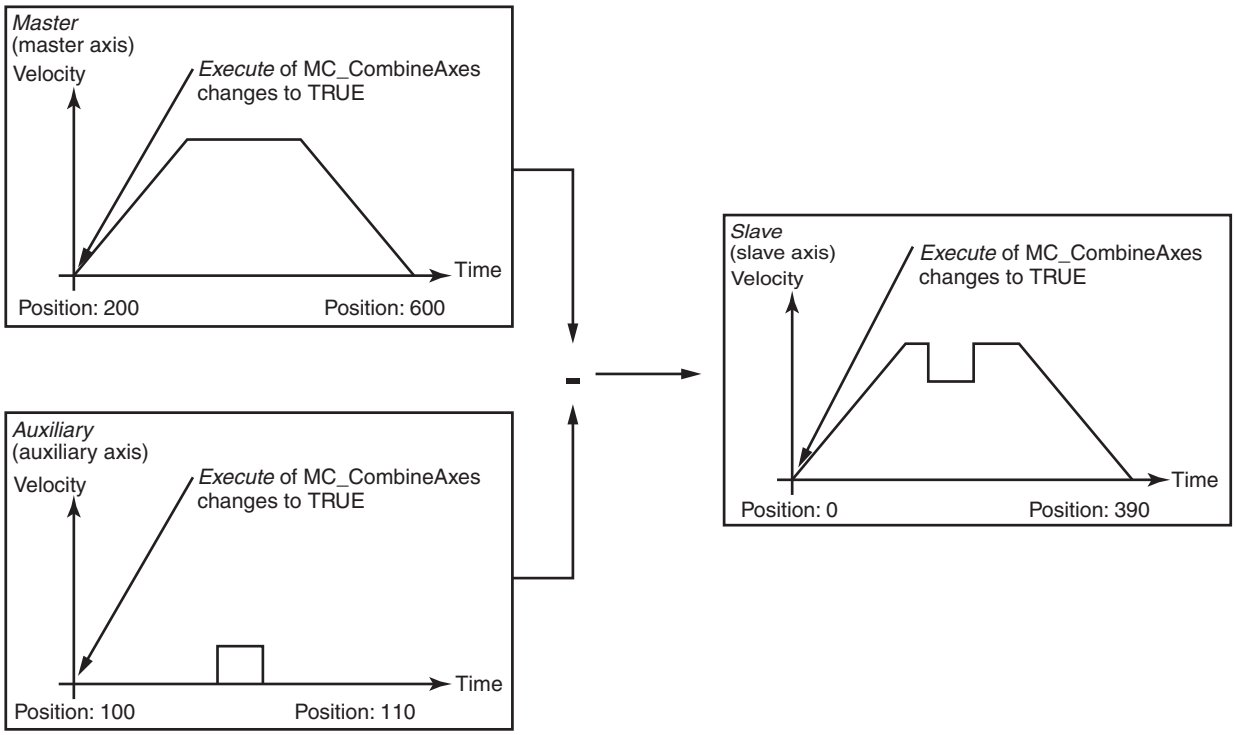


For details on synchronous positioning, refer to the MC_MoveLink (Synchronous Positioning) and MC_Stop instructions in the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)*.

9-2-8 Combining Axes

The sum or difference of two positions can be used as the command position for the slave axis.
Operation starts when the MC_CombineAxes instruction is executed.
Use the MC_Stop instruction to stop the axes in motion.

The following figure is an example where the command position for the slave axis is determined based on the operation of subtraction.
Slave (Slave Axis) command current position = Master (Master Axis) command current position − Auxiliary (Auxiliary Axis) command current position

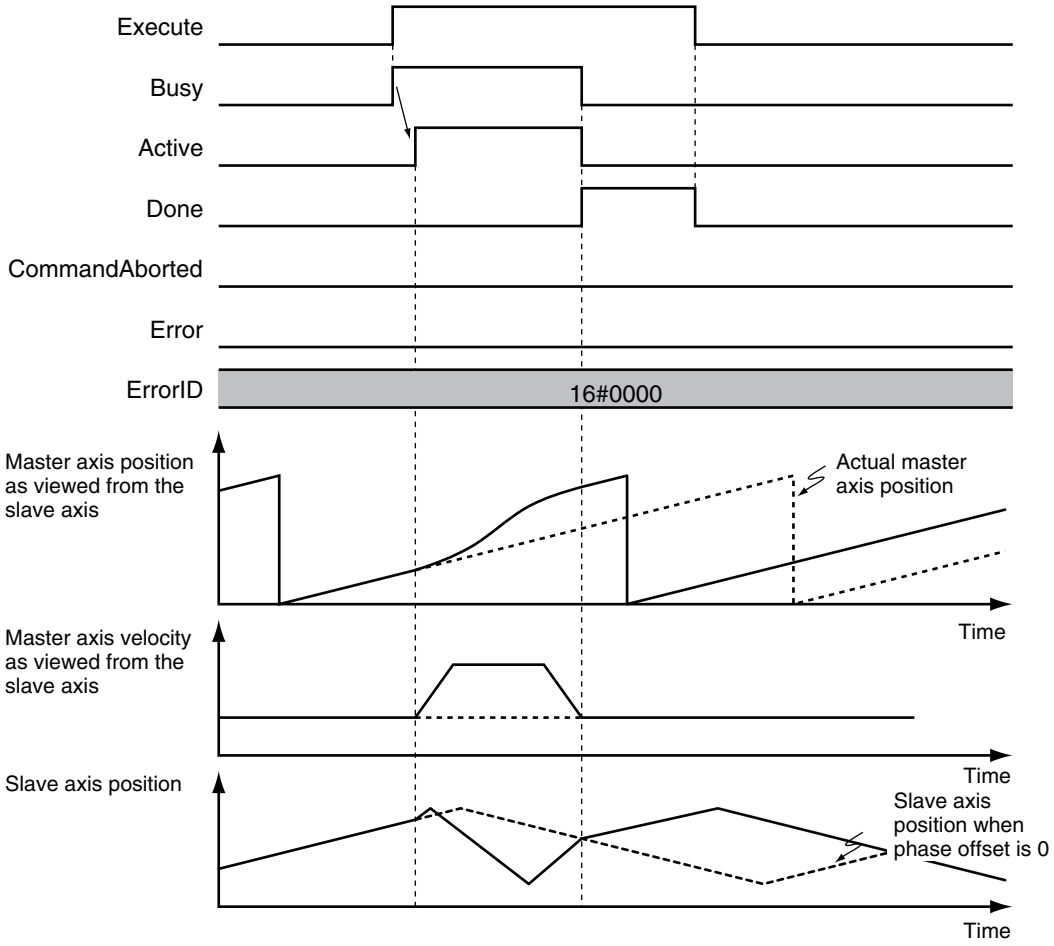


For details on combining axes, refer to the MC_CombineAxes and MC_Stop instructions in the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)*.

9-2-9 Master Axis Phase Shift

The phase of the master axis as viewed from the slave axis can be shifted for the current instruction. The shift amount as viewed from the slave axis is a relative amount. During synchronization, the slave axis will synchronize to the relative distance of the master axis.
You can execute the MC_Phasing (Shift Master Axis Phase) instruction to shift the phase for a synchronized control instruction.

You can specify the *phase shift amount*, *target velocity*, *acceleration rate*, *deceleration rate*, and *jerk* for the MC_Phasing (Shift Master Axis Phase) instruction.



For details on the shift master axis phase function and the synchronized control instructions for which a master axis phase shift can be applied, refer to the MC_Phasing (Shift Master Axis Phase) instruction in the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)*.

9-2-10 Slave Axis Position Compensation

This function compensates the position of the slave axis currently in synchronized control.
An offset calculated from the value of the input variable is added to the command current position. The result is output to the Servo Drive to compensate the position of the slave axis in synchronized control. Even when the MC Function Module commands the same travel distance to two axes, their actual travel distance may be different due to mechanical strain or other factors. This function can perform compensation in such a case.
To perform position compensation cyclically for the slave axis in synchronized control, execute the MC_SyncOffsetPosition (Cyclic Synchronous Position Offset Compensation) instruction.
Also, to perform position compensation with the acceleration/deceleration curve for the slave axis in synchronized control, execute the MC_OffsetPosition (Position Offset Compensation) instruction.

For details on slave axis position compensation, refer to the MC_SyncOffsetPosition (Cyclic Synchronous Position Offset Compensation) instruction and MC_OffsetPosition (Position Offset Compensation) instruction in the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)*.

9-2-11 Achieving Synchronized Control in Multi-motion

When you use the standard functions of the MC Function Module, if the synchronized control instructions are executed between axes assigned to different tasks in the multi-motion, an Illegal Master Axis Specification (event code: 54620000 hex) occurs.
However, you can perform synchronized control of the master axis that is controlled in the primary periodic task and the slave axis that is controlled in the priority-5 periodic task by using the MC_PeriodicSyncVariables (Periodic Axis Variable Synchronization between Tasks) instruction.

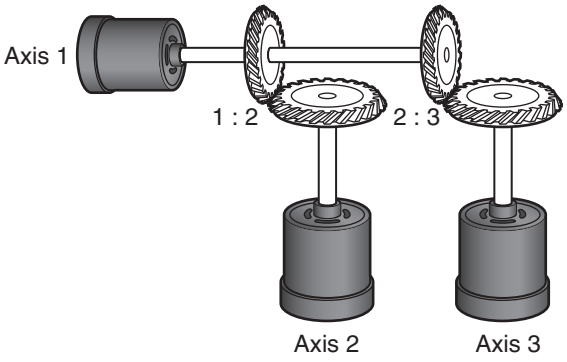
The cam operation and gear operation synchronized with the master axis and slave axis are available for the following combinations.

Mster axis task	Slave axis task	
	Primary periodic task	Priority-5 periodic task
Primary periodic task	Synchronized by motion control instructions	Synchronized control is achieved by executing the MC_PeriodicSyncVariables (Periodic Axis Variable Synchronization between Tasks) instruction and using the **virtual master axis** in the priority-5 periodic task.
Priority-5 periodic task	Not available.	Synchronized by motion control instructions

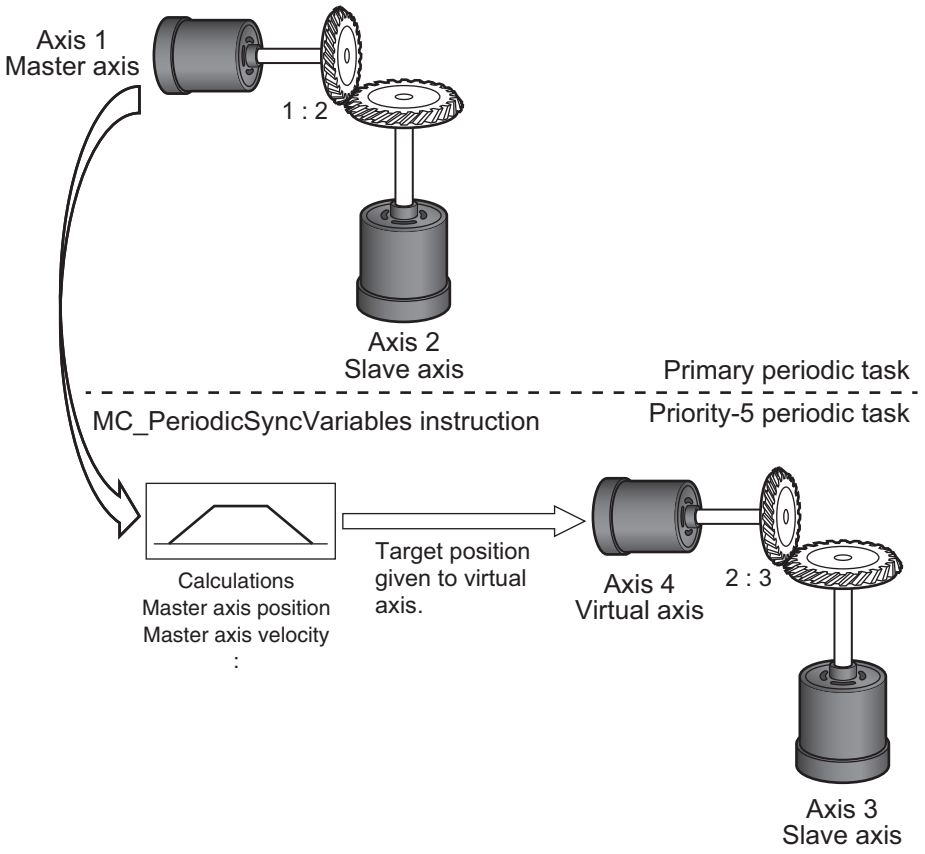
Axis Composition in Operation Examples

In the following figure, axis 1 is the master axis. Axis 2 is a slave axis that requires high-speed and high-precision control. It is assigned to the primary periodic task. Axis 3 is a slave axis that does not require precision. It is assigned to a priority-5 periodic task. The master axis (axis 1) is assigned to the primary periodic task.

● Physical Axis Composition



● Logical Axis Composition



Programming is placed in both the primary periodic task and priority-5 periodic task to achieve the operation for the above application.

Refer to the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)* for details on the MC_PeriodicSyncVariables (Periodic Axis Variable Synchronization between Tasks) instruction.

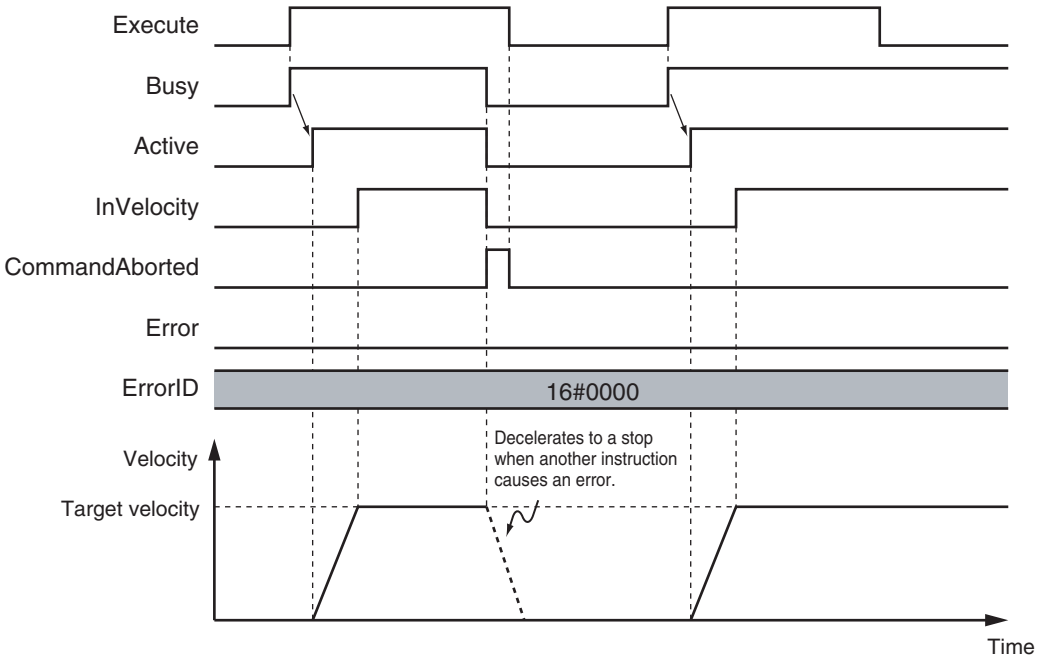
9-3 Single-axis Velocity Control

This section describes the operation of velocity control for single axes.

Refer to the *NX-series Position Interface Units User's Manual (Cat. No. W524)* for the differences when you use NX-series Pulse Output Units.

9-3-1 Velocity Control

Velocity control is used to constantly move an axis at the specified velocity.
You can also specify the acceleration rate, deceleration rate, and jerk.
To stop an axis, use the MC_Stop instruction or execute another motion instruction.
If you specify a target velocity of *0*, the axis will not move but the axis status will indicate that it is moving.
If any other motion control instruction is executed with multi-execution of instructions during velocity control, the operation will switch only after reaching the target velocity.



The MC Function Module uses Position Control Mode of the Servo Drive or other device and sends target position commands to achieve the specified target velocity.
The position control loop is enabled in the Servo Drive or other device. Therefore, as the command velocity slows down, e.g., due to disturbance, and the following error increases, the velocity will change to eliminate this following error.

For details, refer to the MC_MoveVelocity (Velocity Control) instruction in the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)*.

9-3-2 Cyclic Synchronous Velocity Control

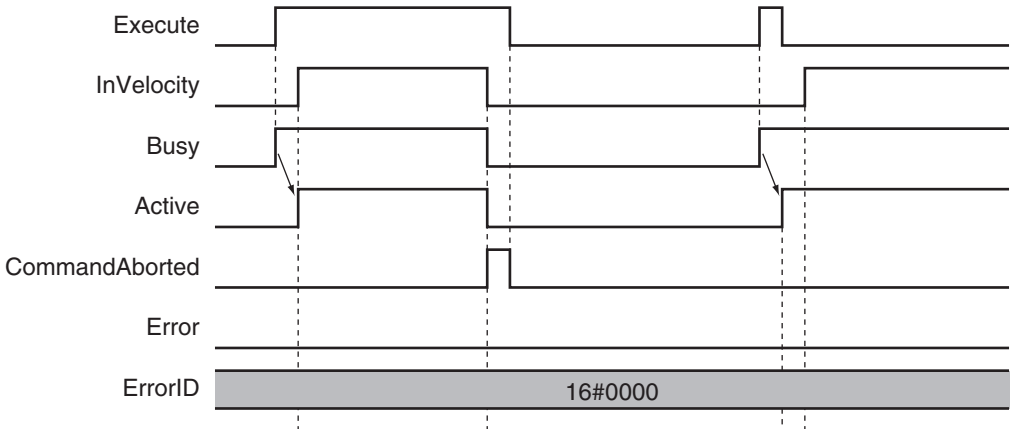
The control mode of the Servo Drive is set to Velocity Control Mode and a command speed is output every control period.

Precautions for Correct Use

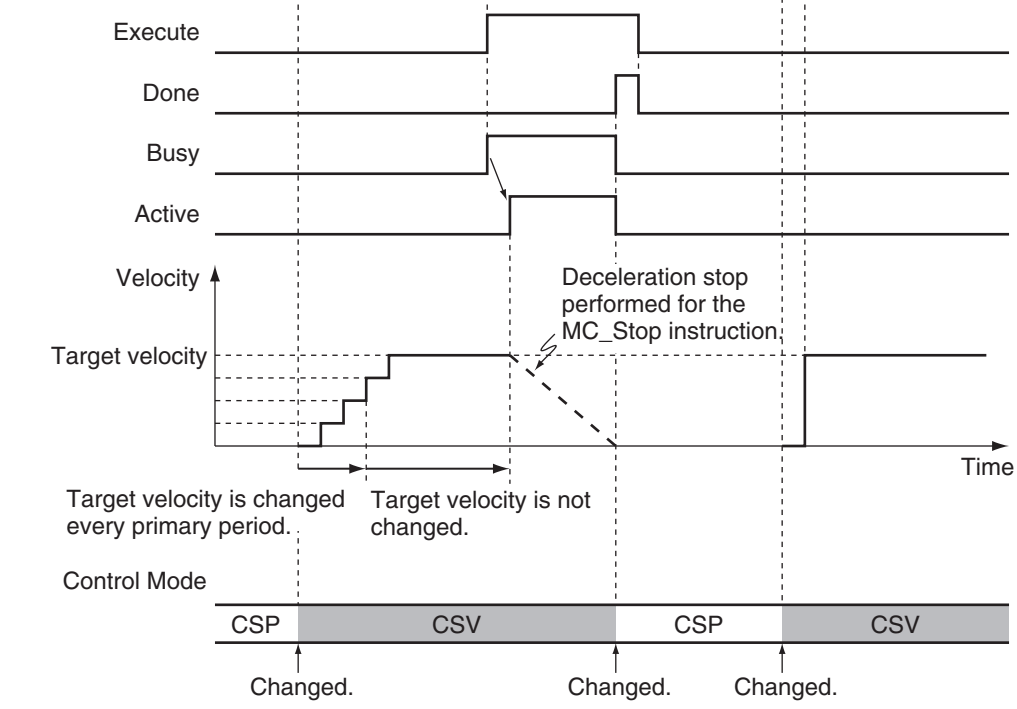
You cannot use cyclic synchronous velocity control for an NX-series Pulse Output Unit.

To stop an axis, use the MC_Stop instruction or execute another motion instruction.
If you specify a target velocity of *0*, the axis will not move but the axis status will indicate that it is moving.

MC_SyncMoveVelocity Instruction



MC_Stop Instruction



The Servo Drive will receive commands in the velocity control loop. Therefore, if any disturbance causes the velocity to decrease below the command velocity, no change in velocity will occur to remove the following error.

For details, refer to the MC_SyncMoveVelocity (Cyclic Synchronous Velocity Control) instruction in the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)*.

9-4 Torque Control

Torque control continuously applies the specified amount of torque.

The MC Function Module uses the Torque Control Mode of the Servo Drive. The Motion Control Function Module outputs a torque command value to the Servo Drive.
The Servo Drive receives the torque command value from the MC Function Module in the torque control loop and to control the torque.
The torque command is specified as a percentage [%] of the rated torque.
The *Velocity* (Velocity Limit) input variable can be used to specify the *velocity limit value* on the Servo Drive. You can use this to limit high-speed revolution of the motor when the load on the motor is low in Torque Control Mode.

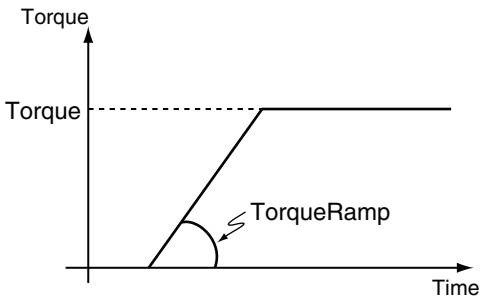
Precautions for Correct Use

- To be safe, always set a velocity limit value for torque control.
- You cannot use torque control for an NX-series Pulse Output Unit.

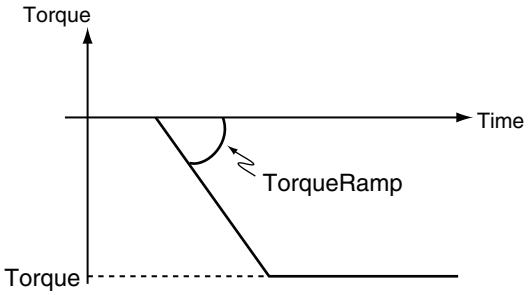
9-4-1 Torque Control

You can use the *TorqueRamp* input variable to specify the rate of change of the torque until the Torque (Target Torque) is reached.
To interrupt the control, use the MC_Stop instruction or execute another motion instruction.
If you specify *0* for Torque (Target Torque), the axis will not move but the axis status will indicate that it is moving.

Example 1: Direction Designation = Positive Direction



Example 2: Direction Designation = Negative Direction



9-4-2 Cyclic Synchronous Torque Control

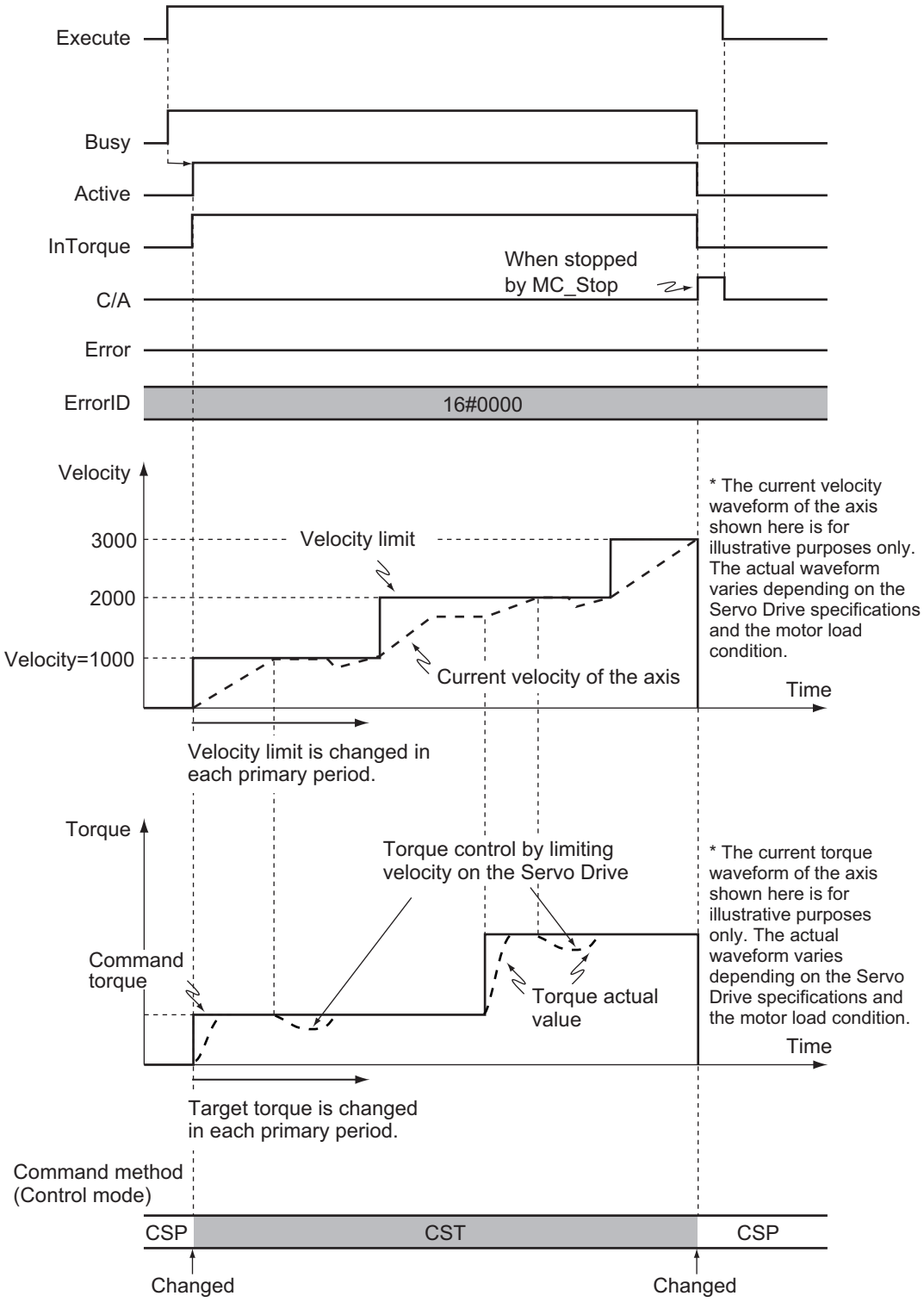
Cyclic synchronous torque control outputs the target torque in each control period.
When this instruction is written in the primary periodic task, the user program instructs the Servo Drive to reach the target torque in the next period after the target torque is given in Cyclic Synchronous Torque Control Mode. If the target torque is not changed, the target torque from the previous period is applied.

Cyclic synchronous torque control differs from torque control in the following points.

Item	MC_SyncTorqueControl	MC_TorqueControl
How the Target Torque input parameter is given	Every period	At instruction execution
Presence of Torque Ramp input parameter	No	Yes
Re-execution	Not possible	Possible

To interrupt the control, use the MC_Stop instruction or execute another motion instruction.
If you specify *0* for Torque (Target Torque), the axis will not move but the axis status will indicate that it is moving.

MC_SyncTorqueControl Instruction



9-5 Common Functions for Single-axis Control

This section describes the common functions used for single-axis control.

Refer to the *NX-series Position Interface Units User's Manual (Cat. No. W524)* for the differences when you use NX-series Pulse Output Units.

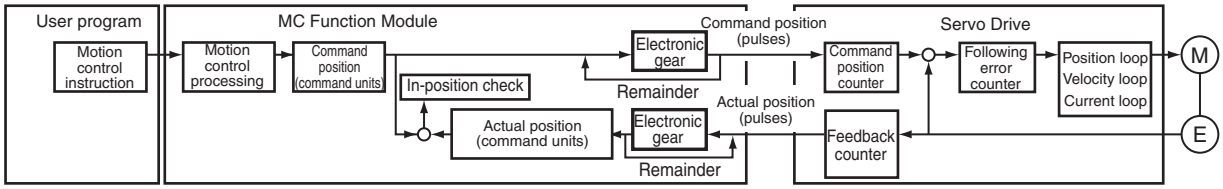
9-5-1 Positions

Types of Positions

The MC Function Module uses the following two types of positions.

Type of position	Definition
Command position	This is the position that the MC Function Module outputs to control an axis.
Actual position	The actual position as input from the Servo Drive or encoder input.

The following figure shows the relationship between the command position and the actual position for an EtherCAT slave Servo Drive.



The command position and actual position share the following items.

Item	Command position	Actual position
Count Mode	You can set Linear Mode or Rotary Mode.	The same Count Mode is used as for the command position.
Position increment	You can set one of the following: mm, μm, nm, inch, degree, or pulse.	The unit is the same as the unit of the command position.
Software limits	You can set the range of operation of the software.	The range is the same as the range for the command position.
Changing the current position	You can change the actual position to any desired position.	This value will be set to the same position as the command position. *1
Defining home	Home is either defined or undefined.	The status of home is the same as the command position.

*1. If there is any following error before the change, the following error value is maintained in the actual position.

Additional Information

Refer to the *NX-series Position Interface Units User's Manual (Cat. No. W524)* for information on the NX-series Position Interface Units.

Axis Parameters That Are Related to Positions

Parameter name	Function	Setting range	Default
In-position Range	Set the in-position width. (Unit: command units)	Non-negative long reals	10
In-position Check Time	Set the in-position check time in milliseconds. Set *0* to check for the end of positioning only when you define the home position during homing and not check positioning at other times. (Unit: ms)	0 to 10,000	0
Software Limits	Select the software limit function. 0: Disabled. 1: Deceleration stop for command position 2: Immediate stop for command position 3: Deceleration stop for actual position 4: Immediate stop for actual position	0 to 4	0
Positive Software Limit	Set the software limit in the positive direction. (Unit: command units)	Long reals	2,147,483,647
Negative Software Limit	Set the software limit in the negative direction. (Unit: command units)	Long reals	-2,147,483,648
Following Error Over Value	Set the excessive following error check value. Set *0* to disable the excessive following error check. (Unit: command units)	Non-negative long reals	0
Following Error Warning Value	Set the following error warning check value. Set *0* to disable the following error warning check. (Unit: command units)	Non-negative long reals that are less than or equal to the Following Error Over Value	0

Specifying Target Positions for Axis Operations

The actual position or distance for a positioning motion is specified with the *Position* (Target Position) and *Distance* (Travel Distance) input variables to the motion control instruction.

Monitoring Positions

You can read Axis Variables in the user program to monitor positions.
In the descriptions, a variable name _MC_AX[*] is used as an example, but the same information applies to _MC1_AX[*] and _MC2_AX[*].

Variable name	Data type	Meaning	Function
_MC_AX[0-255].Cmd.Pos	LREAL	Command Current Position	This is the current value of the command position. When the Servo is OFF and the mode is not the position control mode, the actual current position is output.
_MC_AX[0-255].Act.Pos	LREAL	Actual Current Position	This is the actual current position.

9-5-2 Velocity

Types of Velocities

The following two types of axis velocities are used in the MC Function Module.

Velocity type	Definition
Command velocity	This is the velocity that the MC Function Module outputs to control an axis.
Actual velocity	This is the velocity calculated in the MC Function Module based on the actual position input from the Servo Drive or encoder input. *1

*1. This value is given if the Velocity actual value (606C hex) is mapped in the PDOs and assigned to the **Actual Current Velocity**.

Velocity Unit

A velocity is given in command units/s.
The command unit is the value obtained from unit conversion of the position display unit and the electronic gear.

Axis Parameters That Are Related to Velocities

Parameter name	Function	Setting range	Default
Maximum Velocity	Specify the maximum velocity for the axis. If a target velocity that exceeds the maximum velocity is specified for an axis motion instruction, the axis will move at the maximum velocity.	Positive long reals	400,000,000
Start Velocity*1	Set the start velocity for each axis. Set a value that does not exceed the maximum velocity. (Unit: command units/s)	Positive long reals	0
Maximum Jog Velocity	Set the maximum jog velocity for each axis. *2 Set a value that does not exceed the maximum velocity. (Unit: command units/s)	Positive long reals	1,000,000
Velocity Warning Value	Set the percentage of the maximum velocity at which to output a velocity warning for the axis. No velocity warning is output if *0* is set. (Unit: %)	0 to 100	0
Actual Velocity Filter Time Constant	Set the time period to calculate the average travel of the actual velocity in milliseconds. The average travel is not calculated if *0* is set. (Unit: ms) Use this to reduce variations in the actual current velocity when axis velocity is slow.	0 to 100	0

*1. A CPU Unit with unit version 1.05 or later and Sysmac Studio version 1.06 or higher are required to use this parameter.
*2. The maximum jog velocity is used as the command velocity if you specify a velocity command value that is greater than the maximum jog velocity.

Specifying Target Velocities for Axis Operations

The velocity used in an actual positioning motion is specified by the *Velocity* (Target Velocity) input variable to the motion control instruction.

Monitoring Velocities

You can read Axis Variables in the user program to monitor velocities.
In the descriptions, a variable name _MC_AX[*] is used as an example, but the same information applies to _MC1_AX[*] and _MC2_AX[*].

Variable name	Data type	Meaning	Function
_MC_AX[0-255].Cmd.Vel	LREAL	Command Current Velocity	This is the current value of the command velocity. A plus sign is added during travel in the positive direction, and a minus sign is added during travel in the negative direction.
_MC_AX[0-255].Act.Vel	LREAL	Actual Current Velocity	This is the actual current velocity. A plus sign is added during travel in the positive direction, and a minus sign is added during travel in the negative direction.

9-5-3 Acceleration and Deceleration

Unit of Acceleration and Deceleration Rates

Acceleration rates and deceleration rates are given in command units/s^2.
The command unit is the value obtained from unit conversion of the position display unit and the electronic gear.

Axis Parameters That Are Related to Acceleration and Deceleration

Parameter name	Function	Setting range	Default
Maximum Acceleration	Set the maximum acceleration rate for an axis operation command. There will be no limit to the acceleration rate if *0* is set. (Unit: command units/s^2)	Non-negative long reals	0
Maximum Deceleration	Set the maximum deceleration rate for an axis operation command. There will be no limit to the deceleration rate if *0* is set. (Unit: command units/s^2)	Non-negative long reals	0

Parameter name	Function	Setting range	Default
Acceleration/ Deceleration Over	Set the operation for when the maximum acceleration/ deceleration rate would be exceeded after excessive acceleration/deceleration during acceleration/deceleration control of the axis because stopping at the target position is given priority. 0: Use rapid acceleration/deceleration. (Blending is changed to Buffered.)*1 1: Use rapid acceleration/deceleration. 2: Minor fault stop*2	0 to 2	0
Acceleration Warning Value	Set the percentage of the maximum acceleration rate at which to output an acceleration warning for the axis. No acceleration warning is output if *0* is set. (Unit: %)	0 to 100	0
Deceleration Warning Value	Set the percentage of the maximum deceleration rate at which to output a deceleration warning for the axis. No deceleration warning is output if *0* is set. (Unit: %)	0 to 100	0

*1. For a CPU Unit with unit version 1.10 or later, Blending is not changed to Buffered. Refer to *9-5-7 Multi-execution of Motion Control Instructions (Buffer Mode)* on page 9-53 for details.

*2. For a CPU Unit with unit version 1.10 or later, the axis does not stop with an error when Blending is used for operation. Refer to *9-5-7 Multi-execution of Motion Control Instructions (Buffer Mode)* on page 9-53 for details.

Specifying Acceleration and Deceleration Rates for Axis Operation

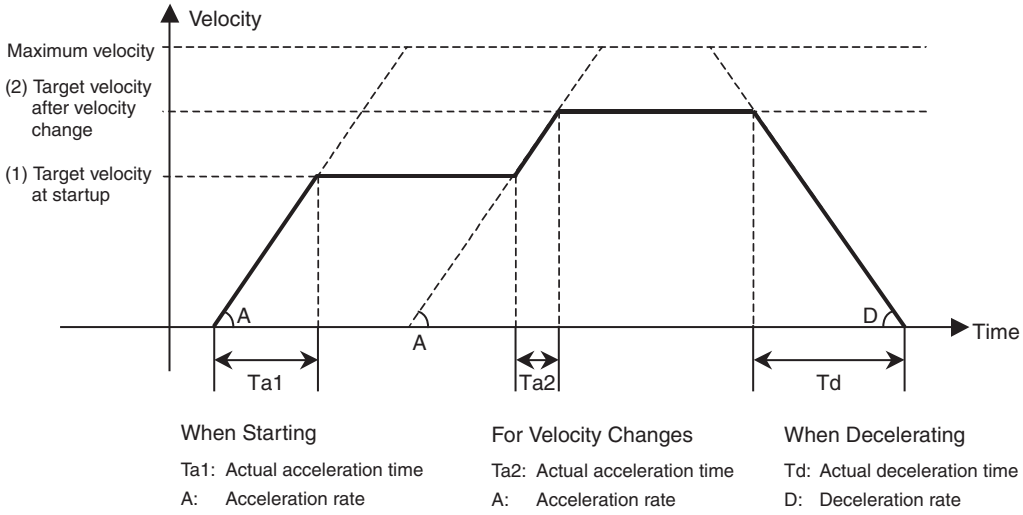
The acceleration and deceleration rates used in an actual positioning motions are specified by the *Acceleration* (Acceleration Rate) and *Deceleration* (Deceleration Rate) input variables to the motion control instruction.

Monitoring Acceleration and Deceleration Rates

You can read Axis Variables in the user program to monitor acceleration and deceleration rates.
In the descriptions, a variable name _MC_AX[*] is used as an example, but the same information applies to _MC1_AX[*] and _MC2_AX[*].

Variable name	Data type	Meaning	Function
_MC_AX[0-255].Cmd.AccDec	LREAL	Command Current Acceleration/ Deceleration	This is the current value of the command acceleration/deceleration rate. A plus sign is added for acceleration, and a minus sign is added for deceleration.

Example of Acceleration/Deceleration Operation



If you specify a short travel distance or a low acceleration/deceleration rate, the target velocity may not be reached.
If the target position is exceeded after re-execution of the motion control instruction with the newly updated acceleration or deceleration rate, positioning is performed at an acceleration or deceleration rate that will enable stopping at the target position.

9-5-4 Jerk

The jerk specifies the rate of change in the acceleration rate or deceleration rate. If the jerk is specified, the velocity waveform during acceleration will be an S-curve, which will reduce the shock and vibration on the machine.

Additional Information

Jerk is also called jolt, surge and lurch.

Jerk Unit

Jerk is given in command units/s^3.
The command unit is the value obtained from unit conversion of the position display unit and the electronic gear.

Specifying Jerk for Axis Motion

The jerk used in an actual positioning motion is specified with the *Jerk* input variable to the motion control instruction.
The same value is used for acceleration and deceleration.

Use the following formula to calculate the value to set for the jerk.

Jerk = Acceleration rate/(Time of acceleration × Ratio of time to apply jerk during acceleration/2)

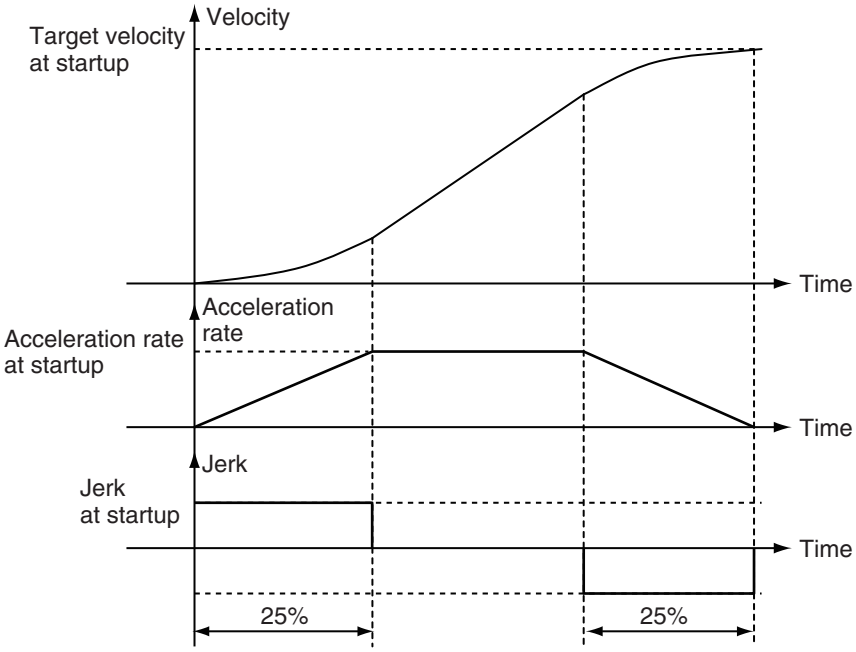
Jerk is applied in two sections: at the start of acceleration and at the end of acceleration. The time that jerk is applied is therefore divided by 2.

● Example of Velocity Control When Jerk Is Specified

The acceleration will change at a constant rate over the range where jerk is specified. The command velocity will form a smooth S curve. A fixed acceleration rate is used in areas where the jerk is set to 0. This command velocity will form a straight line.

Example: Acceleration of 25,000 mm/s^2, Acceleration Time of 0.1 s, and a Jerk Application Rate of 50%

Jerk = 25,000/(0.1 × 0.5/2) = 1,000,000 (mm/s$^{3)}$



Monitoring Jerk

You can read Axis Variables in the user program to monitor jerk.
In the descriptions, a variable name _MC_AX[*] is used as an example, but the same information applies to _MC1_AX[*] and _MC2_AX[*].

Variable name	Data type	Meaning	Function
_MC_AX[0-255].Cmd.Jerk	LREAL	Command Current Jerk	This is the current value of the command jerk.

9-5-5 Specifying the Operation Direction

If you want to specify a rotation direction, such as shortest way, using an index table, set the Count Mode to Rotary Mode. Next, set the operation direction with the *Direction* input variable to the motion control instruction for an absolute position.

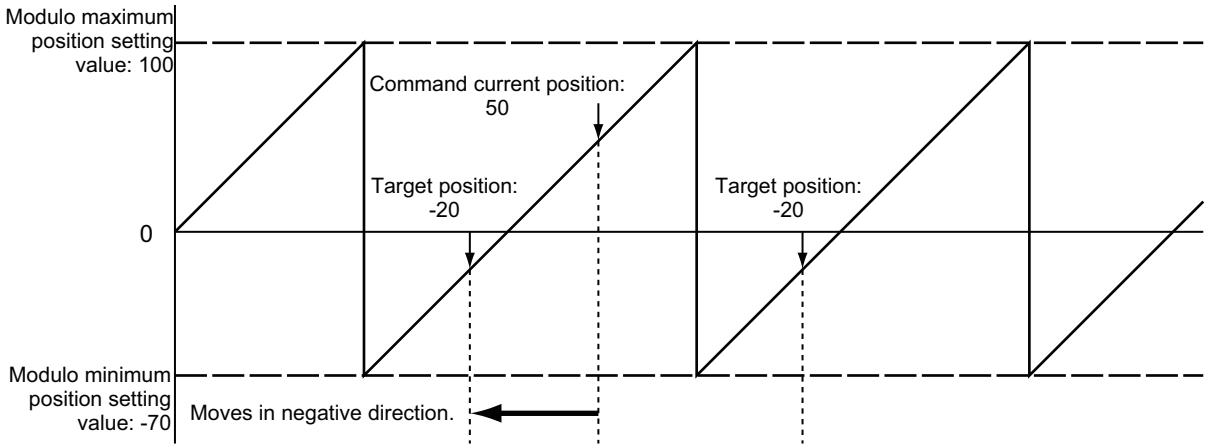
If you set the direction to the **shortest way**, **positive direction**, **negative direction**, or **current direction**, you can specify a position that is greater than or equal to the modulo minimum position and less than the modulo maximum position within one turn of the ring counter.
The Direction input variable will be ignored when the Count Mode is set to Linear Mode. Positioning will be performed to the target position.

The following table lists the different directions you can specify in the MC Function Module.

Direction	Operation
Shortest way	Motion starts in the direction where the command current position and the target position are closer to each other.
Positive direction	Motion starts in the positive direction.
Negative direction	Motion starts in the negative direction.
Current direction	Motion starts in the same direction as the previous operation.
No direction specified	Motion starts in the direction that does not pass through the upper and lower limits of the ring counter. With this direction specification, you can specify a target position that exceeds the upper or lower limits of the ring counter. If that occurs, relative positioning is performed using the difference between the target position and the command current position as the target distance. This enables you to perform multi-turn positioning on the ring counter.

Example for Shortest Way

The following example illustrates when positioning is performed towards a target position of −20 when the command current position is 50.

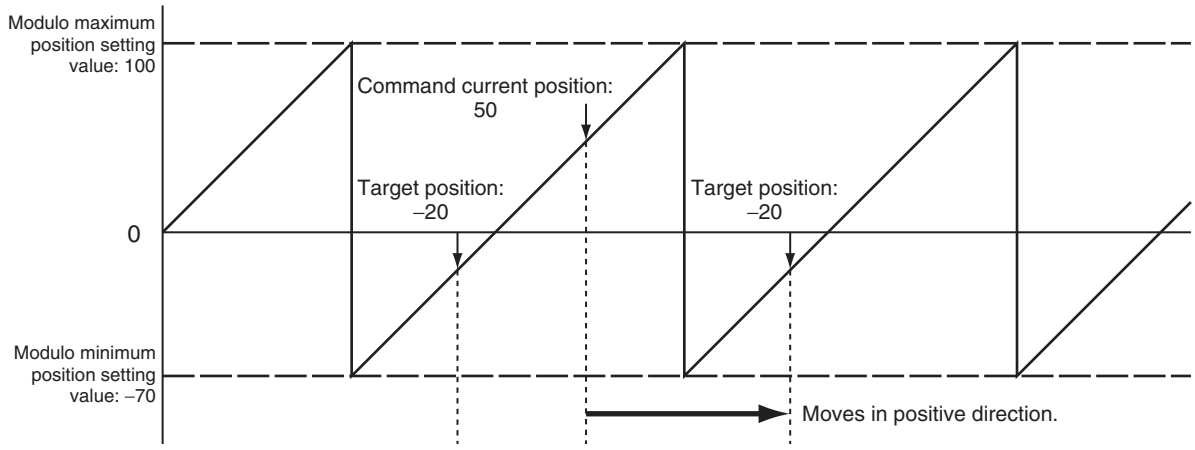


Additional Information

Moves in the same direction as the **Current Direction** specification if the travel distance is the same in the positive and negative directions.

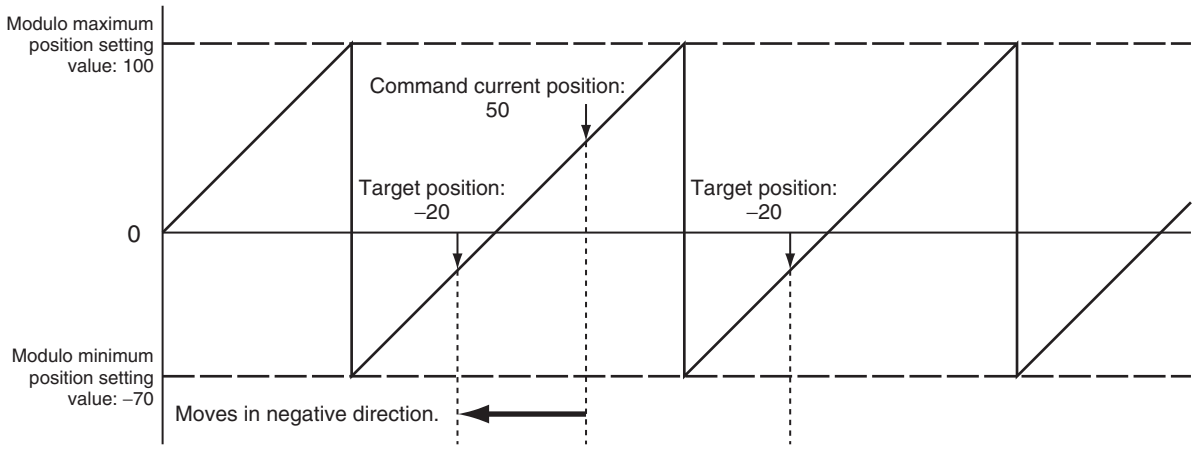
Example for Positive Direction

The following example illustrates when positioning is performed towards a target position of −20 when the command current position is 50.



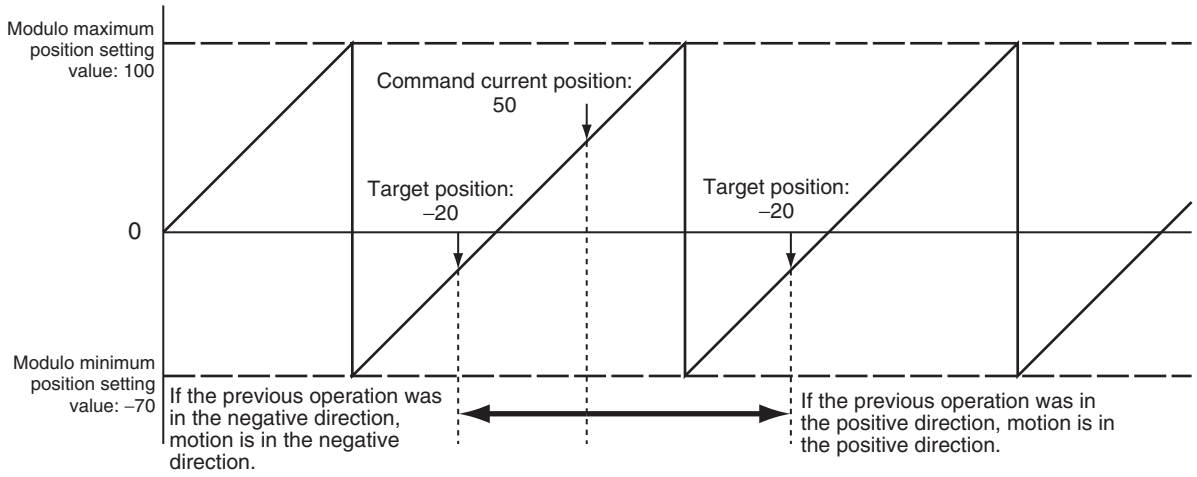
Example for Negative Direction

The following example illustrates when positioning is performed towards a target position of −20 when the command current position is 50.



Example for Current Direction

The following example illustrates when positioning is performed towards a target position of −20 when the command current position is 50.



The direction of the previous operation is given in the *Command Direction* in the Axis Variable.

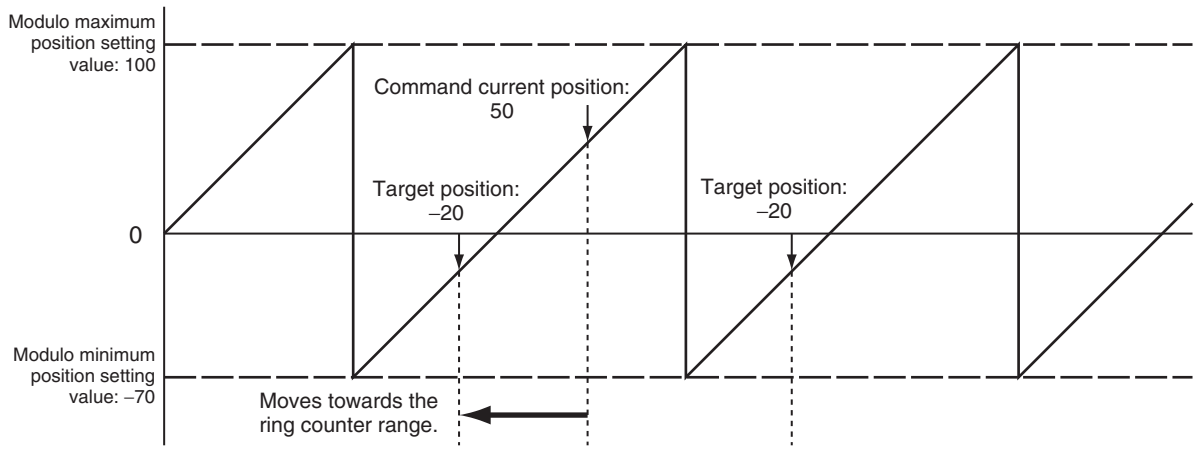
Precautions for Correct Use

Attention should be paid to the direction of operation in the following cases.

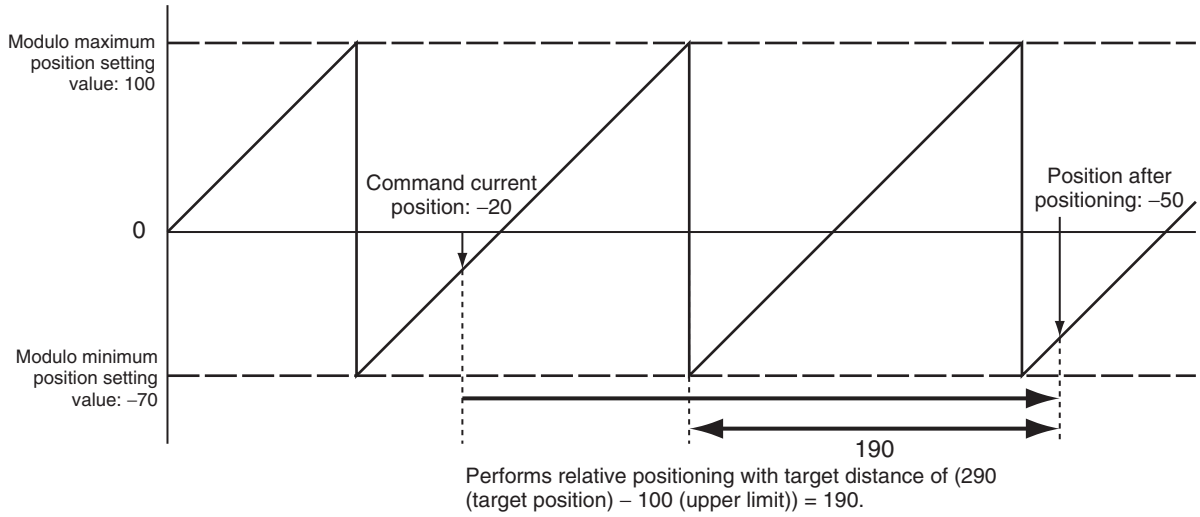
- If the MC_Home or MC_HomeWithParameter instruction exceeds the point where the home input was detected and reverses operation, the opposite direction of the home input detection direction is used.
- If a homing compensation value is set for the MC_Home or MC_HomeWithParameter instruction, the axis will move in the direction of the compensation value.
- If an immediate stop is specified for the MC_TouchProbe (Enable External Latch) instruction, the latch position may be exceeded and the direction may be reversed.
- The direction may be reversed for the MC_MoveFeed (Interrupt Feeding) instruction.
- When the MC_ResetFollowingError instruction is executed, the error is set to zero, so the command direction is used.
- If an immediate stop is specified for an external input signal or resetting the error counter is specified for stopping for a limit input, the operation may reverse direction toward the position where the external input signal was received.

Example for No Direction Specification

The following example illustrates when positioning is performed towards a target position of −20 when the command current position is 50.



Similarly, the following example illustrates when the ring counter upper limit is 100, the lower limit is −70, the command current position is −20, and positioning is performed towards a target position of 290.



9-5-6 Re-executing Motion Control Instructions

This section describes how to modify input variables of the same instance of a motion control instruction during operation of a single axis and re-execute that instruction.
The input variables *Position* (Target Position), *Distance* (Travel Distance), *Velocity* (Target Velocity), *Acceleration* (Acceleration Rate), *Deceleration* (Deceleration Rate), and *Torque* (Target Torque) and sometimes other input variables can be changed by re-execution.
An instruction error will occur if you change an input variable that cannot be changed and attempt to re-execute the instruction.
If you re-execute an instruction that has been buffered due to multi-execution of instructions, the input variables for the instruction in the buffer will change.

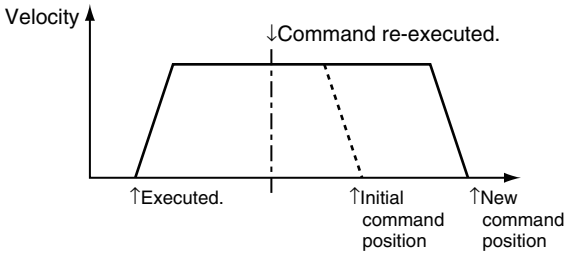
For details on input variables that can be changed, refer to the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)*.

Changing the Target Position

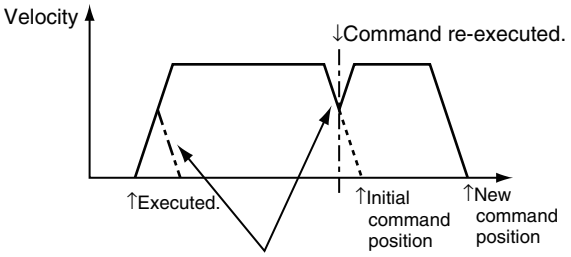
If you change the target position with re-execution, the operation may change depending on the timing of the change and the new target position.
If the direction of motion reverses due to a change in the target position, you can choose to **Deceleration stop** or **Immediate stop** with the **Operation Selection at Reversing** axis parameter.

● When a Reverse Turn Does Not Occur for the New Command Value

Re-executing Instruction during Constant-velocity Motion



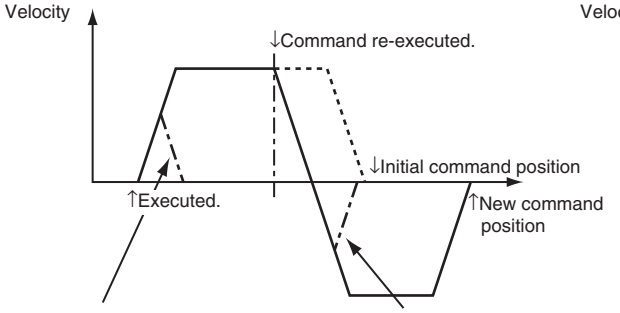
Re-executing Instruction during Acceleration/deceleration



If you re-execute an instruction during triangular control or during deceleration, acceleration to the target velocity will occur again. In some cases, the axis will not reach the target velocity.

● When a Reverse Turn Occurs for the New Command Value

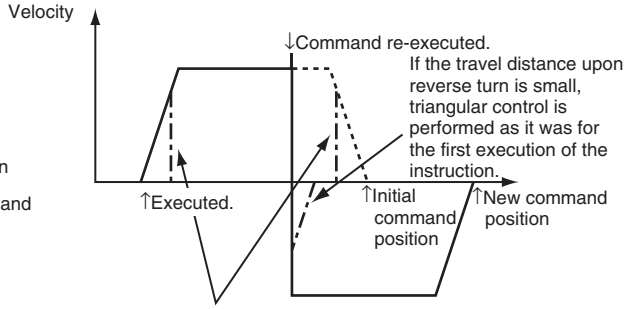
Decelerating to a Stop after Reverse Turn



If the instruction is re-executed during acceleration, the axis starts deceleration as soon as the instruction is re-executed.

If the travel distance upon reversal is small, triangular control is performed as it was for the first execution of the instruction.

Stopping Immediately after Reverse Turn

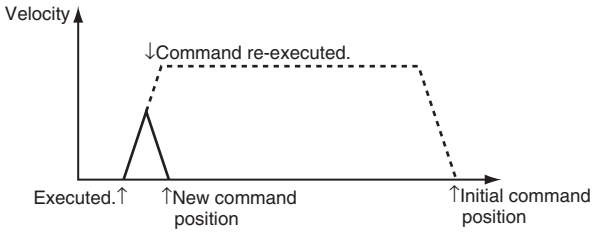


If the instruction is re-executed during acceleration or deceleration, the axis stops immediately upon re-execution. This also occurs during deceleration.

● Triangular Control Patterns

The triangular control shown in the figure below may result if the travel distance is shortened due to a change in the target position.

No Reverse Turn

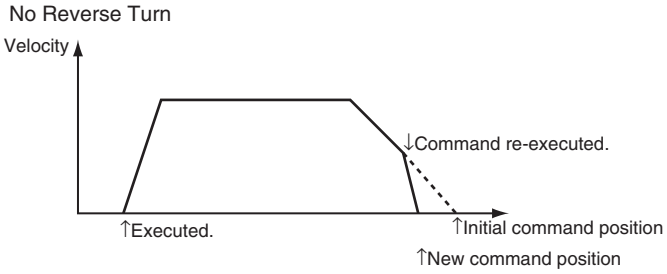


● Excessive Deceleration Patterns

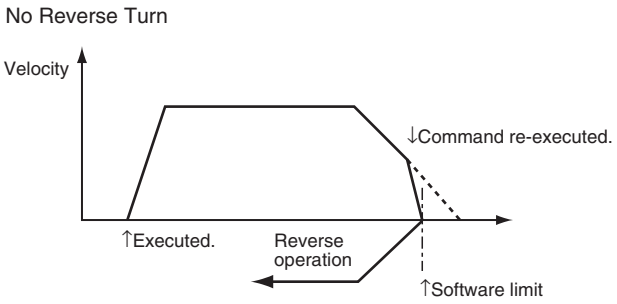
In the following cases, priority is given to stopping at the target position. Therefore, the deceleration rate will exceed the specified deceleration rate.

If the deceleration rate exceeds the rate that is set in the **Maximum Deceleration** axis parameter, the operation set in the **Acceleration/Deceleration Over** axis parameter setting is performed.

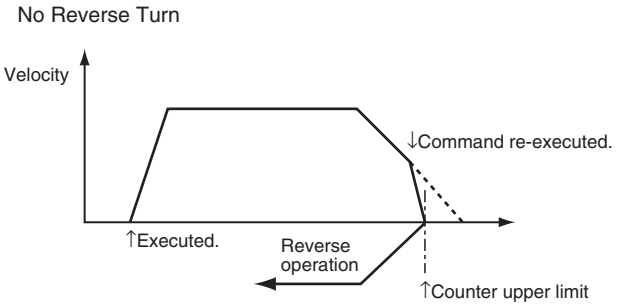
- If There Is No Reverse Turn and the Target Position Would Be Exceeded at the Specified Deceleration Rate



- If There Is a Reverse Turn and Decelerating to a Stop Would Result in an Excess over the Software Limit

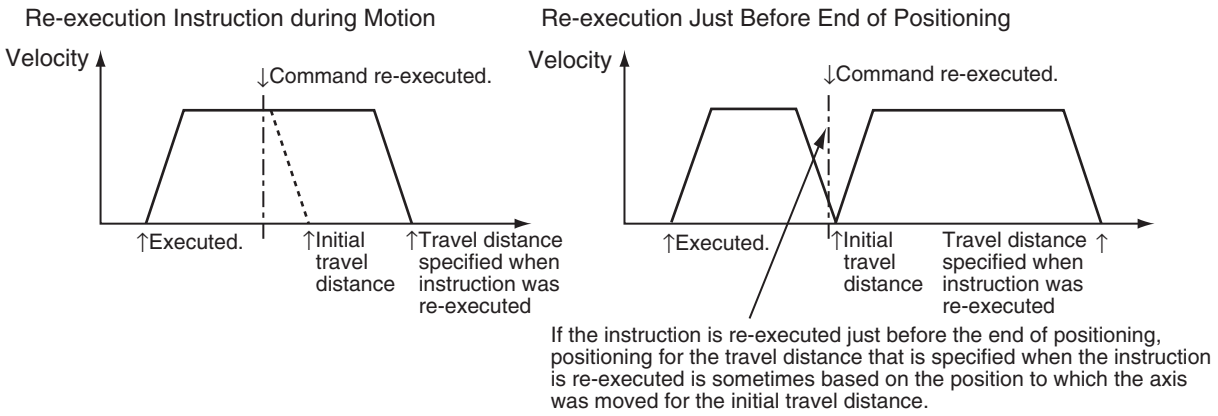


- If There Is a Reverse Turn and Decelerating to a Stop Would Result in Command Current Position Overflow or Underflow



Changing the Travel Distance

Even if you change the travel distance and re-execute the MC_MoveRelative (Relative Positioning) instruction, positioning is performed for the new travel distance in reference to the position where the motion first started.
However, if the instruction is executed again just before positioning is completed, it may be executed as a new instruction rather than as a re-execution of the same instruction.



Precautions for Correct Use

Do not change the travel distance and re-execute the instruction just before the end of positioning.

Changing the Target Velocity

The operation is changed only during acceleration (including acceleration for triangular control) and constant-velocity motion.
Changes are also accepted when the axis is decelerating, but operation is not affected.

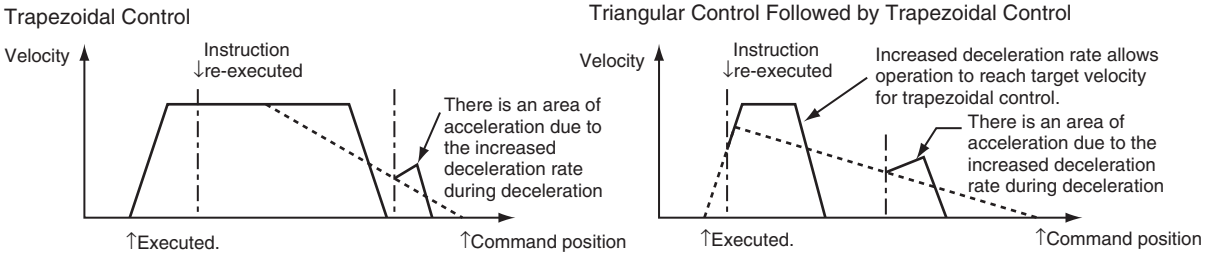
Changing the Acceleration Rate

The operation is changed only during acceleration and acceleration during triangular control.
If it is changed when moving at a constant speed, the changed rate applies to acceleration for an override.
Changes are also accepted when the axis is decelerating, but operation is not affected.

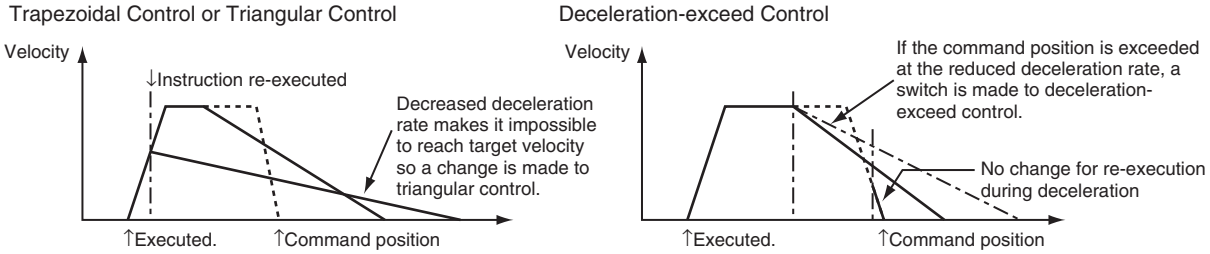
Changing the Deceleration Rate

The deceleration rate is changed only during acceleration, constant-velocity motion, deceleration, triangular control, or during deceleration-exceed control.
If the new deceleration rate causes the axis to exceed the target position, stopping at the target position is given the highest priority. Therefore, in this case, the actual deceleration rate will exceed the specified deceleration rate.

● Patterns Where Deceleration Rate Increases



● Patterns Where Deceleration Rate Decreases

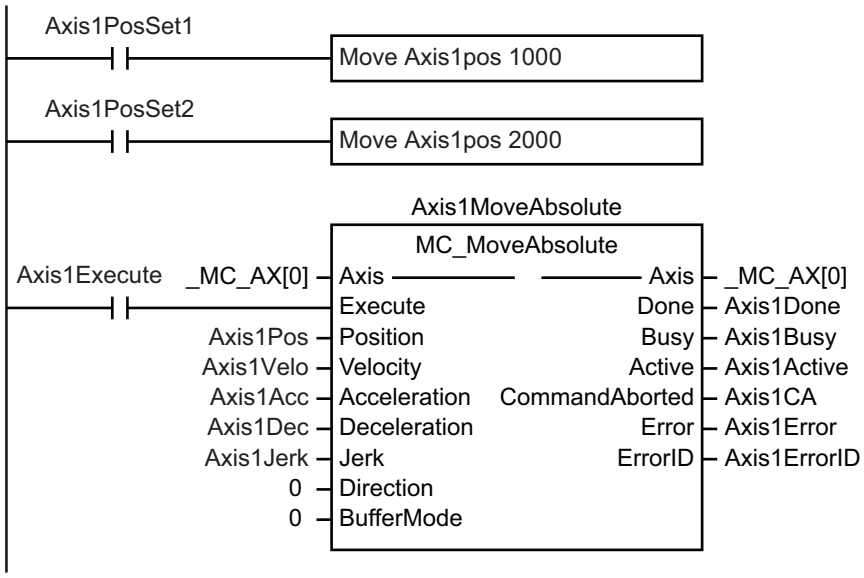


Changing the Torque Command

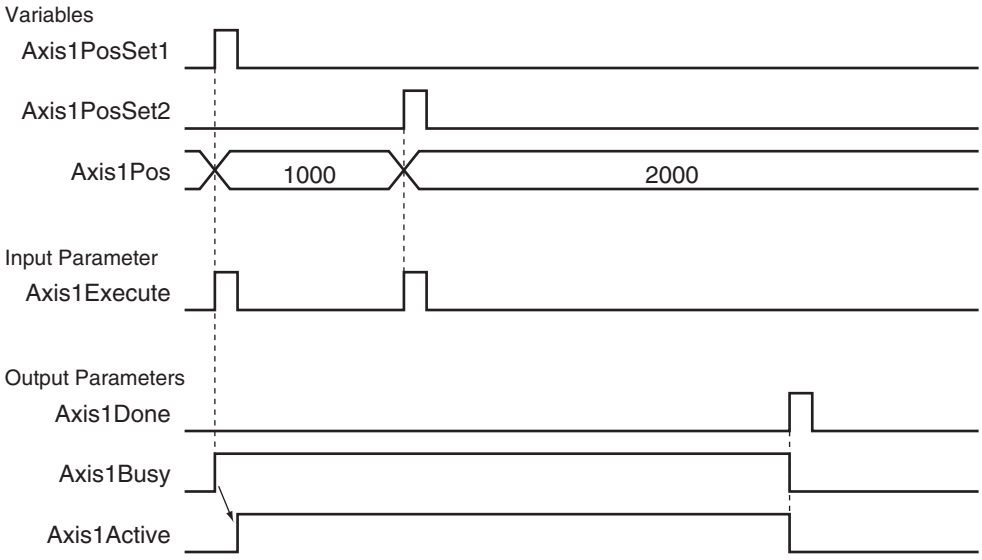
The torque command value will change based on the torque ramp specification when you re-execute a motion control instruction.

Programming Example for Re-execution

This example demonstrates changing the target position from 1000 to 2000 for absolute positioning.
In this example, the variable *Axis1Pos* is used as the input parameter to the target position.
Specify the target position to *1000* with the MOV instruction and change Axis1Execute to TRUE to begin positioning.
Specify the target position to *2000* during operation and change Axis1Execute to TRUE again to switch to a positioning operation for the new target position of 2000.



● Timing Charts



Precautions for Correct Use

For input variables that are not changed, always use the same values as before re-execution of the instruction.

9-5-7 Multi-execution of Motion Control Instructions (Buffer Mode)

You can trigger another motion control instruction while an axis is moving.

In the PLCopen® technical specifications, this functionality is defined as Buffer Mode, but in the MC Function Module this is sometimes referred to as multi-execution of instructions.
You can use multi-execution of instructions to execute multiple motion control instructions in sequence without stopping the overall motion.

The following terms are used in relation to multi-execution of instructions in the MC Function Module.

Term		Meaning
This manual	PLCopen®	
Current instruction	Previous function block	The motion control instruction that is in execution just before multi-execution is triggered.
Buffered instruction	Next function block	A motion control instruction that is triggered while the axis is in motion, and is waiting to be executed.
Transit velocity	Blending	When blending is specified, it specifies the command velocity for the current instruction to move to the specified target position.

You can set the *BufferMode* (Buffer Mode Selection) input variable to motion control instruction to select one of the following Buffer Modes.
The main difference between these modes is the timing at which the buffered instructions are executed and the transit velocity.

Buffer Mode		Description of operation
Aborting		The current instruction is aborted and the next motion control instruction is executed.
Buffered		The buffered instruction is executed after the operation for the current instruction is normally finished.
Blending		The buffered instruction is executed after the target position of the current instruction is reached. In this mode, no stop is performed between the current instruction and the buffered instruction. You can select one of the following transit velocities for when the current instruction reaches the target position.
	Blending Low (low velocity)	The transit velocity is set to the target velocity of the current instruction or the buffered instruction, whichever is lower.
	Blending Previous (previous velocity)	The target velocity of the current instruction is used as the transit velocity.
	Blending Next (next velocity)	The target velocity of the buffered instruction is used as the transit velocity.
	Blending High (high velocity)	The transit velocity is set to the target velocity of the current instruction or the buffered instruction, whichever is higher.

If Buffered or Blending Mode is specified, the next instruction is buffered in the MC Function Module, and executed at the specified BufferMode timing and transit velocity.
There is one buffer for each axis.
If Aborting is specified, the next instruction is executed immediately, so it is not buffered.

Precautions for Correct Use

- Only one instruction can be buffered for each axis. If multi-execution is performed for two or more instructions, an instruction error will occur.
- Multi-execution of multi-axes coordinated control instructions (axes group instructions) is not possible for axes operating as a single axis. Similarly, multi-execution of single-axis control instructions is not possible for axes operating under multi-axes coordinated control (axes group instructions). An instruction error will occur if these rules are broken.

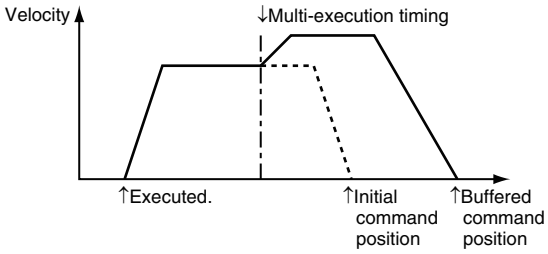
Aborting

This is the default mode. No buffering is performed in this mode.

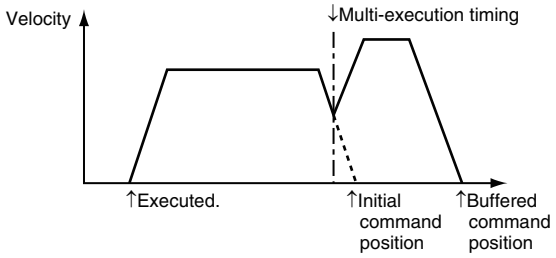
The current command is aborted and the new instruction is executed.
Aborting Mode can be used for multi-execution of motion control instructions for single-axis control and synchronized control.

● When a Reverse Turn Does Not Occur for the Command Position of the Next Instruction for Multi-execution

Multi-execution during Constant-velocity Motion



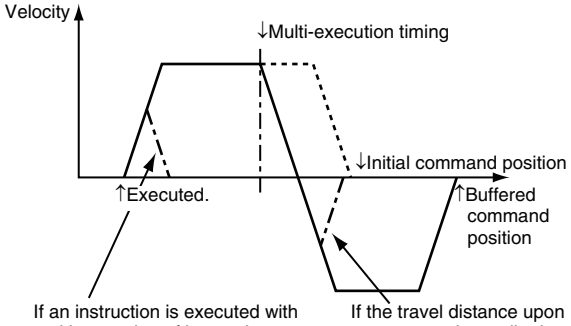
Multi-execution during Acceleration/Deceleration



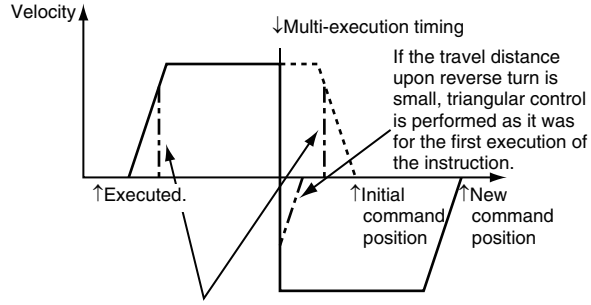
If you use multi-execution of an instruction during triangular control or during deceleration, the axis will accelerate to the target velocity of the buffered instruction. In some cases, the axis will not reach the target velocity.

● When a Reverse Turn Occurs for the Command Position of the Next Instruction for Multi-execution

Decelerating to a Stop after Reverse Turn

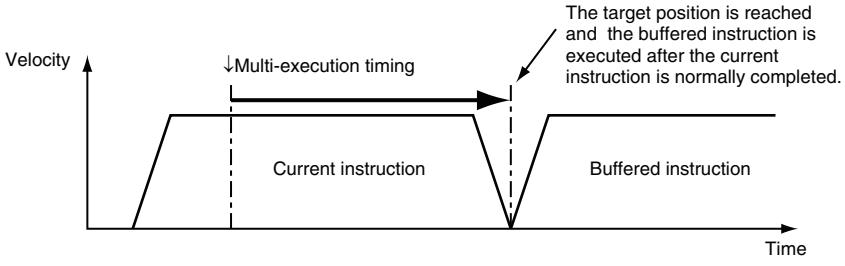


Stopping Immediately after Reverse Turn



Buffered

The next instruction remains in the buffer until the operation of the current instruction is finished.
The buffered instruction is executed after the operation for the current instruction is normally ended.



Blending

The buffered instruction remains in the buffer until the target position of the current instruction is reached.

The buffered instruction is executed after the current instruction's target position is reached. However, motion does not stop at this time. Operation transitions to the next instruction at the velocity specified with the *BufferMode* (Buffer Mode Selection) input variable.

For relative travel, the final position will be the total of the values for both instructions.

For absolute travel, the final position will be the target position of the second instruction.

The **Acceleration/Deceleration Over** axis parameter is used to select one of the following operations for when the target position would be exceeded with the values that are set in the **Maximum Acceleration** and **Maximum Deceleration** axis parameters.

- Use rapid acceleration/deceleration. (Blending is changed to Buffered.)
- Use rapid acceleration/deceleration.
- Minor fault stop

Precautions for Correct Use

- In a blending mode, you cannot combine single-axis and synchronized control.

Version Information

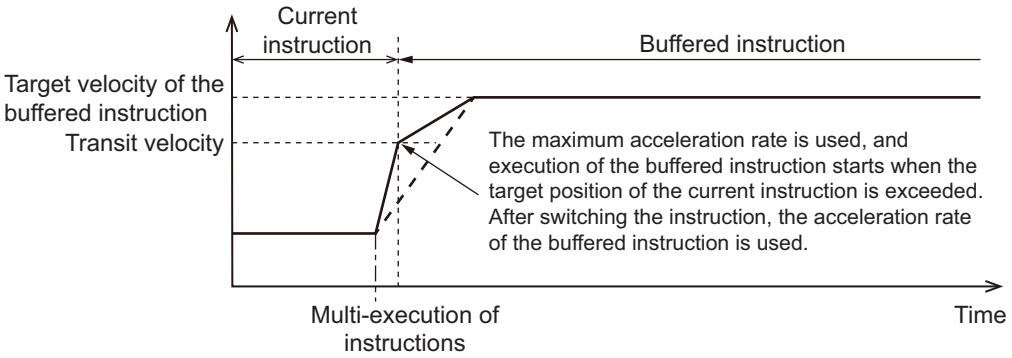
For a CPU Unit with unit version 1.10 or later, Blending is not changed to Buffered even if you select **Use rapid acceleration/deceleration (Blending is changed to Buffered)**. In this case, the maximum acceleration/deceleration rate is used and the blending operation is continued. Also, the axis does not stop with an error even if you select **Minor fault stop**. Similar to the previous case, the maximum acceleration/deceleration rate is used and the blending operation is continued.

● Use Rapid Acceleration/Deceleration (Blending Is Changed to Buffered)

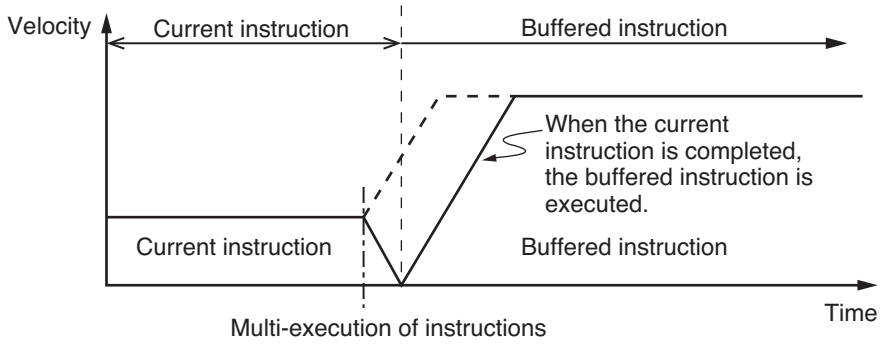
- For a CPU Unit with Unit Version 1.10 or Later

  The operation will be the same even if you select **Minor fault stop**.

  Here, BufferMode is set to Blending Next.

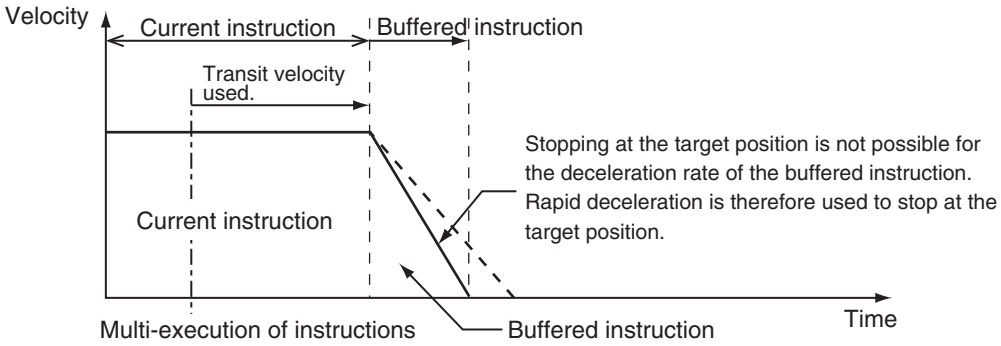


- For a CPU Unit with Unit Version 1.09 or Earlier

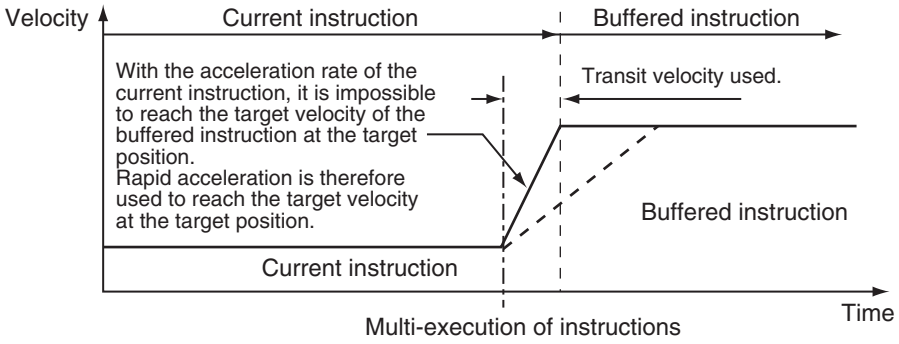


● Use Rapid Acceleration/Deceleration

- *BufferMode* Is Set to Blending Previous



- *BufferMode* Is Set to Blending Next



Blending Low (Low Velocity)

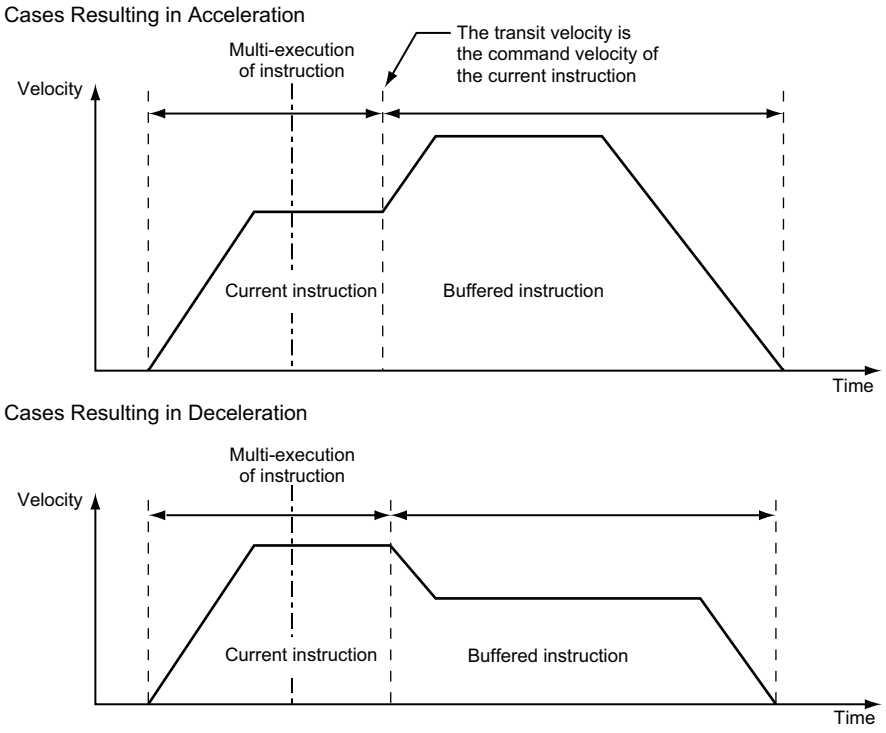
Operation is performed using the target position of the current instruction and the target velocity that is the lower of the target velocities for the current instruction and buffered instruction.

Blending Previous (Previous Velocity)

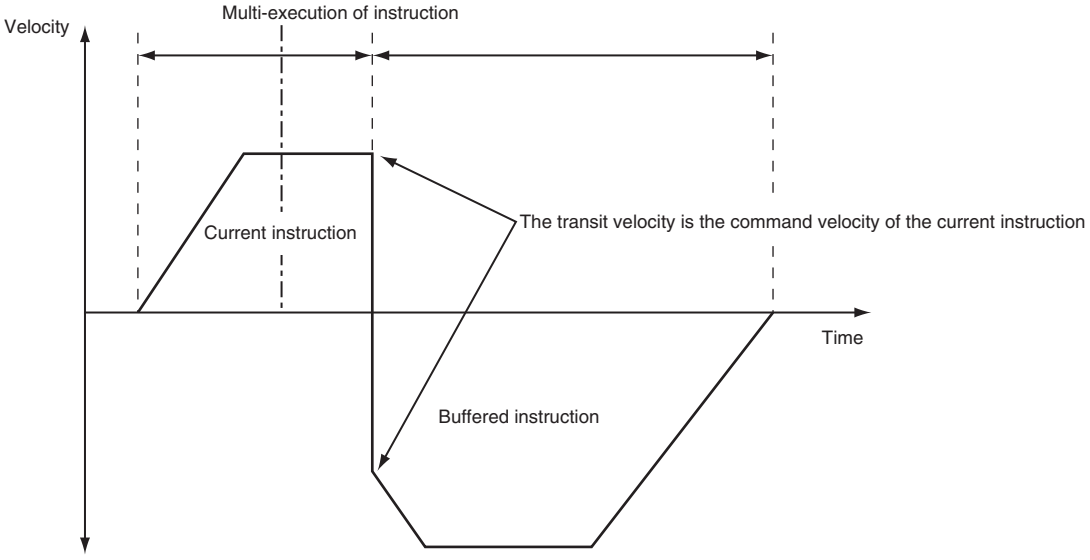
Operation is performed with the target velocity of the current instruction until the target position of the current instruction is reached.

Operation is performed after acceleration/deceleration to the target velocity of the buffered instruction once the target position is reached.

● When the Direction of Operation Does Not Change

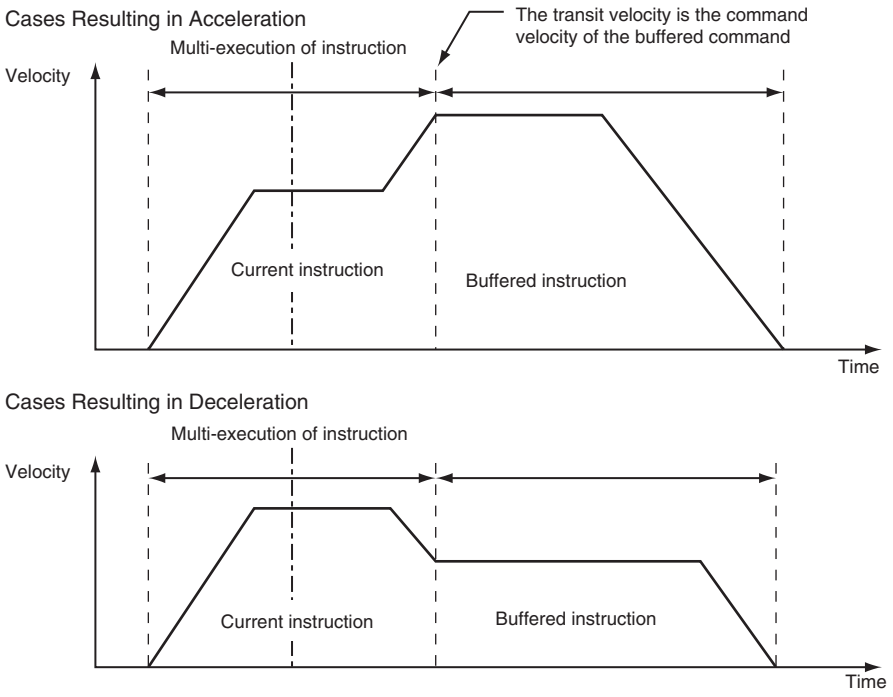


● When the Direction of Operation Changes



Blending Next (Next Velocity)

Operation is performed using the target position of the current instruction and the target velocity of the buffered instruction.



Blending High (High Velocity)

Operation is performed using the target position of the current instruction and the target velocity that is the higher of the target velocities for the current instruction and buffered instruction.

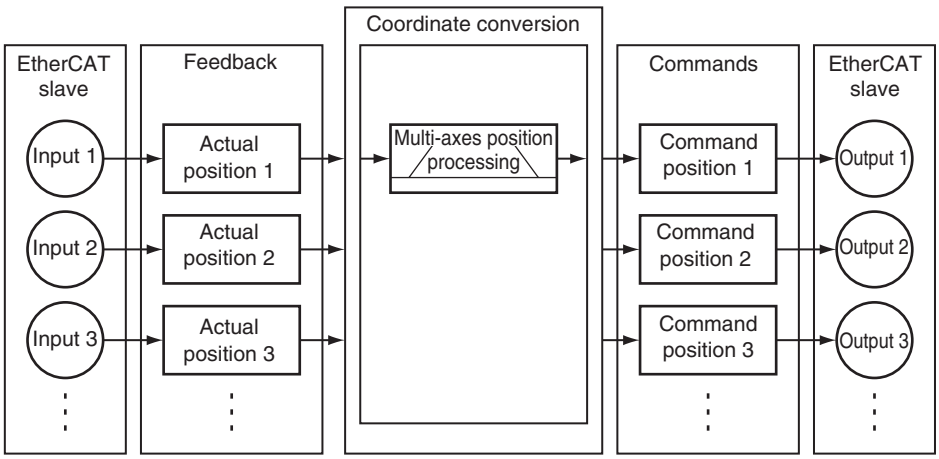
9-6 Multi-axes Coordinated Control

This section describes the operation of multi-axes coordinated control.
With the MC Function Module, you can set an axes group in advance from the Sysmac Studio to perform interpolation control for multiple axes.

9-6-1 Outline of Operation

Multi-axes coordinated control performs a motion with multiple related axes together as a single group to control the path of the target control object.
The MC Function Module treats all axes that perform coordinated operation as an axes group. Axes groups are set from the Sysmac Studio.
In the user program, turn ON the Servo for each axis and then enable the axes group that is going to perform the multi-axes coordinated control.
The purpose of multi-axes coordinated control is the coordinated operation of all axes belonging to the target axes group. Therefore, you cannot execute any single-axis operation motion control instructions on the axes in an enabled axes group. Furthermore, if any error occurs for any axis in an axes group, all axes in the axes group will stop according to the setting of the **Axes Group Stop Method** group axes parameter.

The MC Function Module can perform linear interpolation with two to four axes or circular interpolation with two axes.



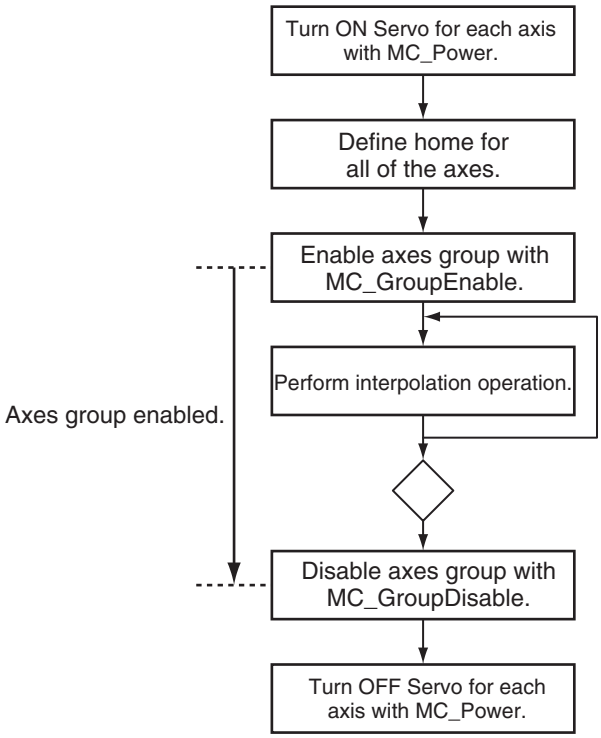
Additional Information

For devices that require you to modify the grouping of axes in motion to perform interpolation control, you must create multiple axes groups that include the axes to modify from the Sysmac Studio beforehand. After completing this step, you can execute by specifying the enabled axes groups from the user program during operation.
With a CPU Unit with unit version of 1.01 or later and Sysmac Studio version 1.02 or higher, you can use the MC_ChangeAxesInGroup (Change Axes in Group) instruction to change the composition axes for an axes group that is disabled.

For details on axes groups, refer to *3-3 Axes Groups* on page 3-22.

Enabling and Disabling Axes Groups

To enable an axes group, specify the axes group for the MC_GroupEnable (Enable Axes Group) instruction. An instruction error will occur if you try to execute an axes group instruction when the axes group is still disabled.
To disable an axes group, specify the axes group for the MC_GroupDisable (Disable Axes Group) instruction.
When you disable an axes group that is in operation, all axes in that axes group will decelerate to a stop at the maximum deceleration rate that is specified in their axis parameter settings.



For details on enabling and disabling axes groups, refer to the MC_GroupEnable (Enable Axes Group) and MC_GroupDisable (Disable Axes Group) instructions in the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)*.

Changing the Axes in an Axes Group

You can use the MC_ChangeAxesInGroup (Change Axes in Group) instruction to temporarily change the composition axes for an axes group that is disabled. If the axes group is enabled, use the MC_GroupDisable (Disable Axes Group) instruction to disable the axes group before you change the composition axes.
A CPU Unit with unit version 1.01 or later and Sysmac Studio version 1.02 or higher are required to use this instruction.

Precautions for Correct Use

Changes made using the MC_ChangeAxesInGroup (Change Axes in Group) instruction will not be saved to non-volatile memory in the CPU Unit. If you cycle the power supply or download the settings from the Sysmac Studio, the parameter settings in the non-volatile memory are restored.

For details on changing the composition axes of an axes group, refer to the MC_ChangeAxesInGroup (Change Axes in Group) instruction in the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)*.

Reading Axes Group Positions

You can use the MC_GroupReadPosition (Read Axes Group Position) instruction to read the command current positions and the actual current positions of an axes group.
A CPU Unit with unit version 1.01 or later and Sysmac Studio version 1.02 or higher are required to use this instruction.

For details on reading the axis positions for an axes group, refer to the MC_GroupReadPosition (Read Axes Group Position) instruction in the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)*.

Resetting Axes Group Errors

If an error occurs in an axes group, you can use the MC_GroupReset instruction to remove the error once you have eliminated the cause.

For details on resetting axes group errors, refer to the MC_GroupReset (Group Reset) instruction in the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)*.

Refer to the *NX-series Position Interface Units User's Manual (Cat. No. W524)* for the differences when you use NX-series Pulse Output Units.

9-6-2 Linear Interpolation

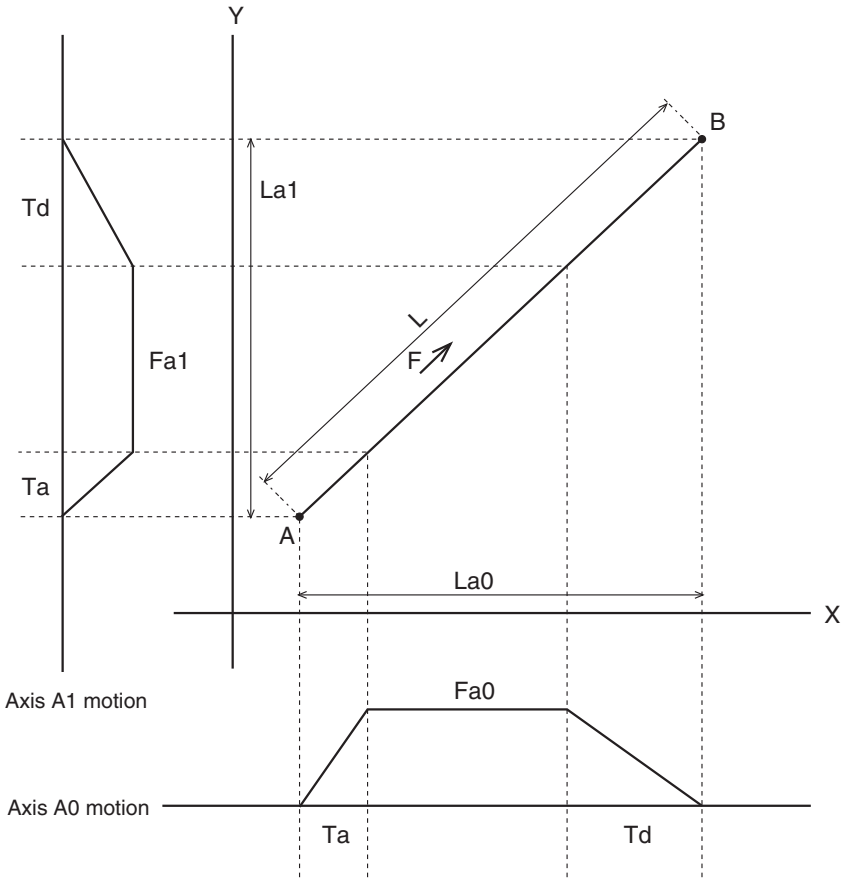
Linear interpolation is used to move 2 to 4 of the logical axes A0 to A3 in a straight line between a start point and an end point.
Either absolute or relative positioning is possible. You can specify the interpolation velocity, interpolation acceleration, interpolation deceleration, and jerk.

The MC Function Module uses the following three kinds of linear interpolation instructions.

- MC_MoveLinear (Linear Interpolation)
  You can specify the *MoveMode* input variable to select between linear interpolation to an absolute value or linear interpolation to a relative value. This instruction is unique to the MC Function Module.
- MC_MoveLinearAbsolute (Absolute Linear Interpolation)
  This instruction performs linear interpolation to an absolute value. This instruction is defined in the PLCopen® technical specifications.
- MC_MoveLinearRelative (Relative Linear Interpolation)

This instruction performs linear interpolation to a relative value. This instruction is defined in the PLCopen® technical specifications.

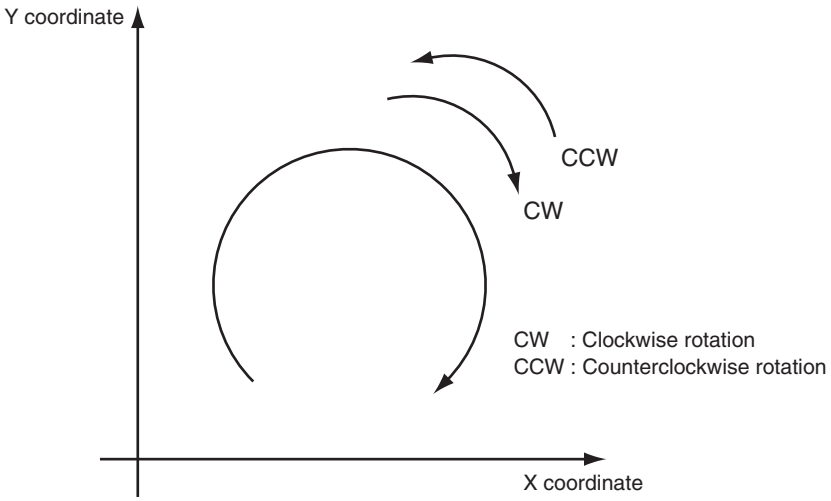
The following figure shows linear interpolation of 2 axes from point A to point B.



For details on linear interpolation, refer to the MC_MoveLinear (Linear Interpolation), MC_MoveLinear-Absolute (Absolute Linear Interpolation), and MC_MoveLinearRelative (Relative Linear Interpolation) instructions in the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)*.

9-6-3 Circular Interpolation

Circular interpolation is used to move two of the logical axes A0 to A3 in a circular motion on a 2D plane.
Either absolute or relative positioning is possible. You can specify the circular interpolation mode, path direction, interpolation velocity, interpolation acceleration, interpolation deceleration, and combined jerk for the two axes.



With the MC Function Module, you can specify the following three kinds of circular interpolation methods with the input variable *CircMode* (Circular Interpolation Mode).

- Border point
- Center
- Radius

Precautions for Correct Use

Set the Count Mode to **Linear Mode** for the axis that you use for circular interpolation.
If the instruction is executed with this axis in **Rotary Mode**, an instruction error will occur.

9-6-4 Axes Group Cyclic Synchronous Positioning

You can cyclically output specified target positions for the axes in an axes group.
You can specify target positions that are calculated in the user program as absolute positions to move the axes in any desired path.
A CPU Unit with unit version 1.01 or later and Sysmac Studio version 1.02 or higher are required to use this instruction.

For details on axes group cyclic synchronous positioning for an axes group, refer to the MC_GroupSyncMoveAbsolute (Axes Group Cyclic Synchronous Absolute Positioning) instruction in the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)*.

9-6-5 Stopping Under Multi-axes Coordinated Control

Multi-axes coordinated control of axes groups will stop when you execute certain motion control instructions in the user program or when an error or some other problem occurs.

Stopping with Motion Control Instructions

Use the MC_GroupStop or MC_GroupImmediateStop instruction to stop axes group operation.

For details, refer to the MC_GroupStop and MC_GroupImmediateStop instructions in the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)*.

● MC_GroupStop Instruction

For linear interpolation or circular interpolation performed on an axes group, you can decelerate to a stop along the control path. You specify the deceleration rate and jerk.
Specify a deceleration rate of *0* to send a command that immediately stops the Servo Drive or other devices.
Other operation commands are not acknowledged while decelerating to a stop for this instruction and while the input variable *Execute* is TRUE.

● MC_GroupImmediateStop Instruction

You can perform an immediate stop for all axes in the axes group. The immediate stopping method is determined by the setting of the **Immediate Stop Input Stop Method** axis parameter for each axis.
The MC_GroupImmediateStop instruction can also be executed for an axes group that is decelerating to a stop for an MC_GroupStop instruction.

Stopping Due to Errors or Other Problems

● Stopping for Errors during Axes Group Motion

If an error that results in a deceleration stop occurs for any composition axis in the axes group during an axes group motion, all of the axes will decelerate to a stop on the interpolation path at the interpolation deceleration rate. The interpolation deceleration rate is determined by the deceleration rate that is specified for the controlling instruction.
If an error that results in an immediate stop occurs for any composition axis in the axes group during an axes group motion, the other axes in the axes group will stop according to the setting of the **Axes Group Stop Method** parameter in the axes group parameters.

You can select one of the following stop methods for axes groups.

- Immediate stop
- Decelerate axes to a stop at maximum deceleration rate of the axes.
- Immediate stop and Servo OFF

● Stopping Due to Motion Control Period Exceeded

If motion control processing does not end within two periods, a Motion Control Period Exceeded occurs. All axes stop immediately.

Precautions for Correct Use

When you use an NX701 CPU Unit and operate in the multi-motion, all axes in both tasks will stop immediately if a Motion Control Period Exceeded occurs in either of the tasks.
Refer to *A-6-2 Motion Control* on page A-31 for multi-motion.

● Stopping Due to Start of MC Test Run

All axes will decelerate to a stop at their **maximum deceleration** if a MC Test Run is started from the Sysmac Studio.

● Stopping Due to Change in CPU Unit Operating Mode

All the axes will decelerate to a stop at their **maximum deceleration** if the CPU Unit operating mode changes.

Additional Information

- If you execute the MC_GroupDisable (Disable Axes Group) instruction during axes group operation, the axes in the group will decelerate to a stop at their maximum deceleration rates.
- If you execute the MC_Stop instruction while an axes group is in operation, an error will occur for the axes and axes group and the axes group operation will decelerate to a stop with interpolation. The interpolation deceleration rate is determined by the deceleration rate that is specified for the controlling instruction.
- When the input variable *Enable* to the MC_Power (Servo ON) instruction changes to FALSE during axes group motion, the MC Function Module immediately stops the command value for that axis and turns OFF the Servo.
  When the Servo is turned OFF, the Servo Drive or other device will operate according to the settings in the Servo Drive or other device.
  Other axes in that axes group will stop with the stop method that is set in the **Axes Group Stop Method** axes group parameter. An error will occur for the axes group if this happens.
- When RUN mode changes to PROGRAM mode, any motion control instructions for current motions are aborted. The *CommandAborted* output variable from the instructions remains FALSE. The Servo ON/OFF status remains the same even after changing to PROGRAM mode.
- If the operating mode returns to RUN mode while a deceleration stop is in progress after the operating mode changes from RUN to PROGRAM mode, the output variable *CommandAborted* from the current motion control instructions changes to TRUE.
- The save process will continue during a save for the MC_SaveCamTable instruction.
- The generation process will continue when generation of the cam table is in progress for the MC_GenerateCamTable (Generate Cam Table) instruction.

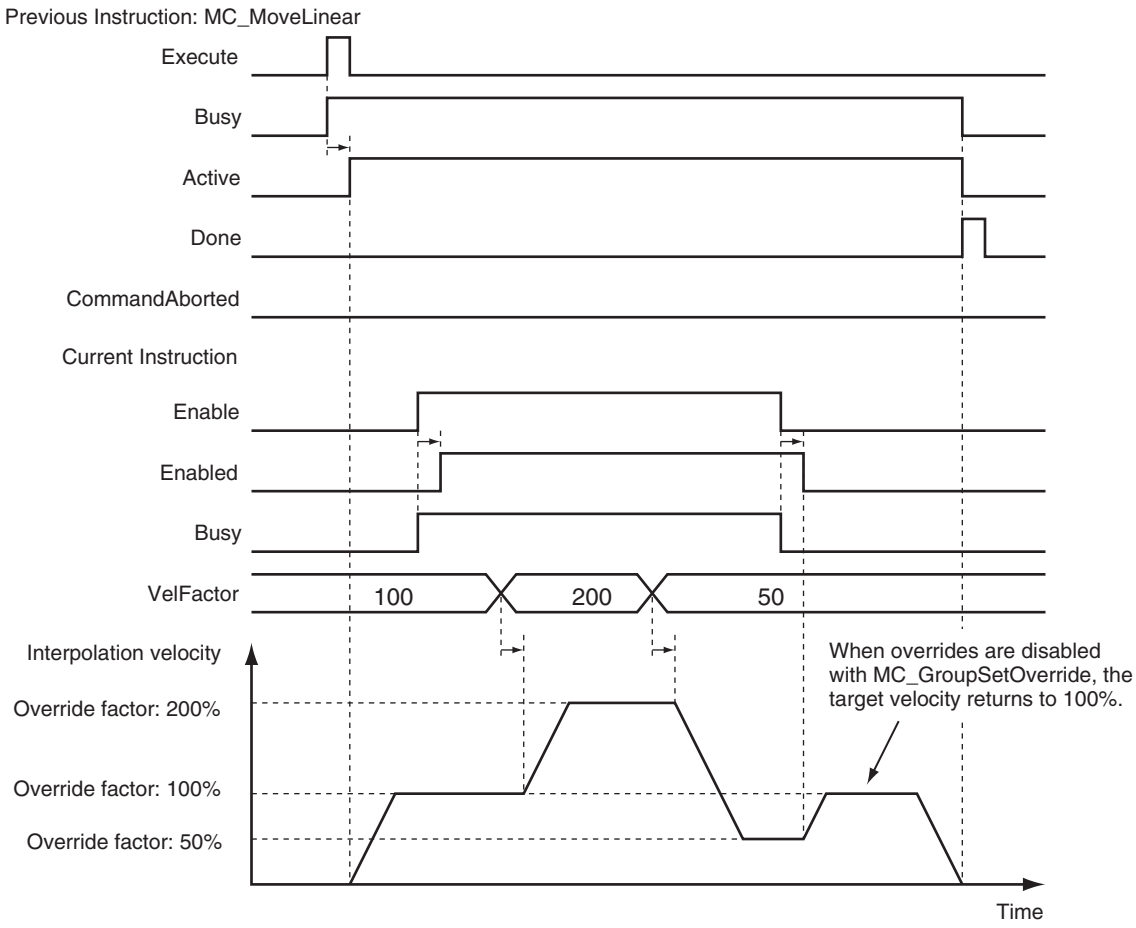
9-6-6 Overrides for Multi-axes Coordinated Control

You can use the MC_GroupSetOverride (Set Group Overrides) instruction to set override factors for multi-axes coordinated control of the axes group in the current interpolation operation.
The velocity override factor is set as a percentage of the target velocity for interpolation. It can be set between 0% and 500%.
If an override factor of 0% is set for the interpolation target velocity, operating status will continue with the axis stopped at a velocity of 0.

The set override factor is read as long as the overrides are enabled. If the overrides are disabled, the override factors return to 100%.
If the maximum interpolation velocity is exceeded when an override factor is changed, the **maximum interpolation velocity** for the axes group is used.

● Overrides for the MC_MoveLinear (Linear Interpolation) Instruction

An example of a time chart for using the Set Override Factors instruction for the MC_MoveLinear (Linear Interpolation) instruction is given below.



For details, refer to the MC_GroupSetOverride (Set Group Overrides) instruction in the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)*.

9-7 Common Functions for Multi-axes Coordinated Control

This section describes the common functions for multi-axes coordinated control.

9-7-1 Velocity Under Multi-axes Coordinated Control

To specify the velocity for multi-axes coordinated control, specify the interpolation velocity on the path. The unit is the same as for single axes, command units/s.

Types of Velocities

The following is the only type of interpolation velocity for axes groups supported by the MC Function Module.

Velocity type	Definition
Command interpolation velocity	This is the actual value of the command interpolation velocity output by the MC Function Module to control an axes group.

Axis Parameters That Are Related to Velocities

Parameter name	Function	Setting range	Default
Maximum Interpolation Velocity	Set the maximum interpolation velocity for the path. Set *0* for no interpolation velocity limit. If a target velocity that exceeds the maximum interpolation velocity is specified for an axes group motion instruction, the axis will move at the maximum interpolation velocity.	Non-negative long reals	800,000,000
Interpolation Velocity Warning Value	Set the percentage of the maximum interpolation velocity at which to output an interpolation velocity warning. No interpolation velocity warning is output if *0* is set. (Unit: %)	0 to 100	0

Specifying Target Velocities for Axis Operations

The interpolation velocity used in an actual positioning motion is specified by the *Velocity* (Target Velocity) input variable to the motion control instruction.

Monitoring Velocities

You can read Axes Group Variables from the user program to monitor the interpolation velocity.
In the descriptions, a variable name _MC_GRP[*] is used as an example, but the same information applies to _MC1_GRP[*] and _MC2_GRP[*].

Variable name	Data type	Meaning	Function
_MC_GRP[0-63].Cmd.Vel	LREAL	Command Interpolation Velocity	This is the current value of the command interpolation velocity. A plus sign is added during travel in the positive direction, and a minus sign is added during travel in the negative direction.

9-7-2 Acceleration and Deceleration Under Multi-axes Coordinated Control

Multi-axes coordinated control performs control on the path for the interpolation acceleration and interpolation deceleration rates.

The unit is the same as for single axes, command units/s^2.

Axis Parameters That Are Related to Interpolation Acceleration and Interpolation Deceleration

Parameter name	Function	Setting range	Default
Maximum Interpolation Acceleration	Set the maximum interpolation acceleration for the path. Set *0* for no interpolation acceleration limit. (Unit: command units/s^2)	Non-negative long reals	0
Maximum Interpolation Deceleration	Set the maximum interpolation deceleration for the path. Set *0* for no interpolation deceleration limit. (Unit: command units/s^2)	Non-negative long reals	0
Interpolation Acceleration/Deceleration Over	Set the operation for when the maximum interpolation acceleration/deceleration rate would be exceeded after excessive acceleration/deceleration during acceleration/deceleration control of the axes group because stopping at the target position is given priority. 0: Use rapid acceleration/deceleration. (Blending is changed to Buffered.)*1 1: Use rapid acceleration/deceleration. 2: Minor fault stop*2	0 to 2	0
Interpolation Acceleration Warning Value	Set the percentage of the maximum interpolation acceleration rate at which to output an interpolation acceleration warning. No interpolation acceleration warning is output if *0* is set. (Unit: %)	0 to 100	0
Interpolation Deceleration Warning Value	Set the percentage of the maximum interpolation deceleration rate at which to output an interpolation deceleration warning. No interpolation deceleration warning is output if *0* is set. (Unit: %)	0 to 100	0

*1. For a CPU Unit with unit version 1.10 or later, Blending is not changed to Buffered. Refer to *9-5-7 Multi-execution of Motion Control Instructions (Buffer Mode)* on page 9-53 for details.

*2. For a CPU Unit with unit version 1.10 or later, the axis does not stop with an error when Blending is used for operation. Refer to *9-5-7 Multi-execution of Motion Control Instructions (Buffer Mode)* on page 9-53 for details.

Specifying an Interpolation Acceleration and Interpolation Deceleration for an Axes Group

The interpolation acceleration and interpolation deceleration rates used in an actual positioning motion are specified by the *Acceleration* (Acceleration Rate) and *Deceleration* (Deceleration Rate) input variables to the motion control instruction.

Monitoring Interpolation Acceleration and Interpolation Deceleration Rates

You can read Axes Group Variables in the user program to monitor interpolation acceleration and interpolation deceleration rates.
In the descriptions, a variable name _MC_GRP[*] is used as an example, but the same information applies to _MC1_GRP[*] and _MC2_GRP[*].

Variable name	Data type	Meaning	Function
_MC_GRP[0-63].Cmd.AccDec	LREAL	Command Interpolation Acceleration/ Deceleration	This is the current value of the command interpolation acceleration/deceleration rate. A plus sign is added for acceleration, and a minus sign is added for deceleration.

9-7-3 Jerk for Multi-axes Coordinated Control

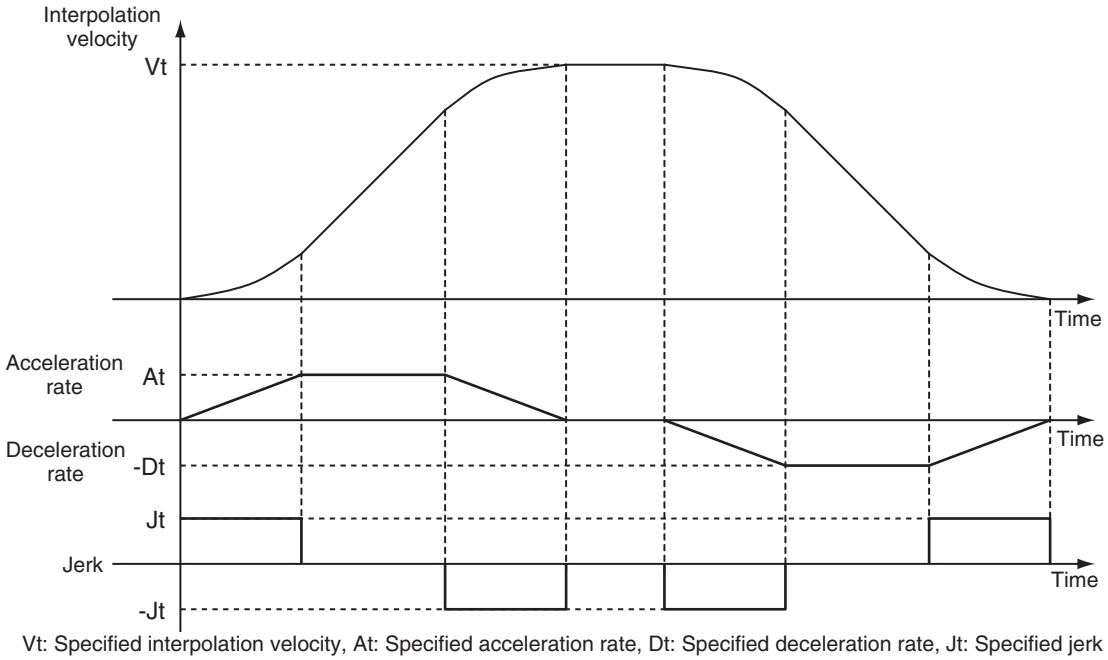
Jerk for multi-axes coordinated control is used to reduce shock and vibration on the machine by smoothing the interpolation acceleration/deceleration rate along the interpolation path into an S-curve.
The unit is the same as for single axes, command units/s^3.

Specifying Jerk for Axes Group Motion

The jerk used in an actual interpolation is specified by the *Jerk* input variable to the motion control instruction.

Jerk Example (Setting Other than *0*)

The acceleration/deceleration rate will change at a constant rate over the range where jerk is specified. The command interpolation velocity will form a smooth S-curve.
A fixed interpolation acceleration rate is used in areas where the jerk is set to *0*. This command interpolation velocity will form a straight line.

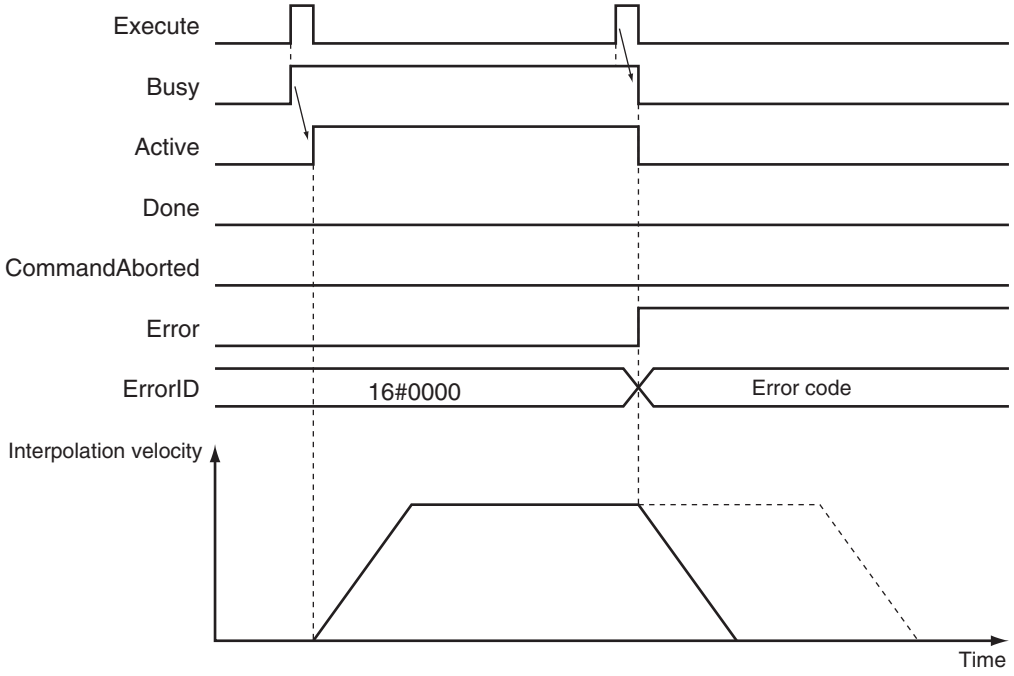


Vt: Specified interpolation velocity, At: Specified acceleration rate, Dt: Specified deceleration rate, Jt: Specified jerk

Vt: Specified interpolation velocity, At: Specified acceleration rate, Dt: Specified deceleration rate, Jt: Specified jerk

9-7-4 Re-executing Motion Control Instructions for Multi-axes Coordinated Control

If you re-execute a linear interpolation or circular interpolation instruction, an instruction error will occur.



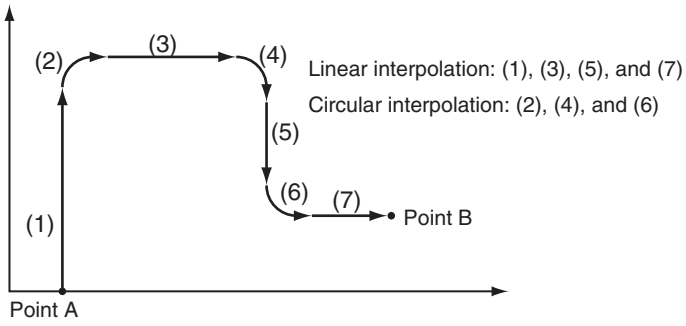
You can change the deceleration rate if you re-execute the MC_GroupStop instruction, but you cannot change the jerk in this way.

If you re-execute the MC_GroupReset instruction, the re-execution command will be ignored and error reset processing will continue.

For details on re-executing motion control instructions, refer to each instruction in the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)*.

9-7-5 Multi-execution of Motion Control Instructions (Buffer Mode) for Multi-axes Coordinated Control

You can perform multi-execution for multi-axes coordinated control in axes groups the same way as you can for axis operations.
You can perform path control for multiple continuous lines and/or arcs if you use Buffer Mode under multi-axes coordinated control.



You can set the *BufferMode* input variable to motion control instruction to select one of the same Buffer Modes as are supported for single-axis operations.
There are a total of eight instruction buffers for axes groups. Each axes group has one buffer for the instruction currently in operation and seven buffers for multi-execution of instructions.
Multi-execution of instruction cannot be used from an axis motion instruction to an axes group motion instruction and vice-versa.

Precautions for Correct Use

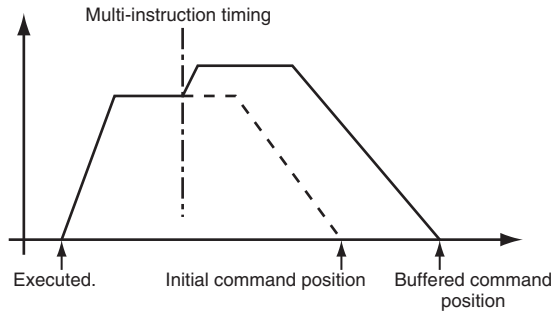
- Up to seven instructions can be buffered at the same time for a single axes group. If multi-execution is performed for eight or more instructions, an instruction error will occur.
- Multi-execution of multi-axes coordinated control instructions (axes group instructions) is not possible for axes operating as a single axis. Similarly, multi-execution of single-axis control instructions is not possible for axes operating under multi-axes coordinated control (axes group instructions). An instruction error will occur if these rules are broken.

Aborting

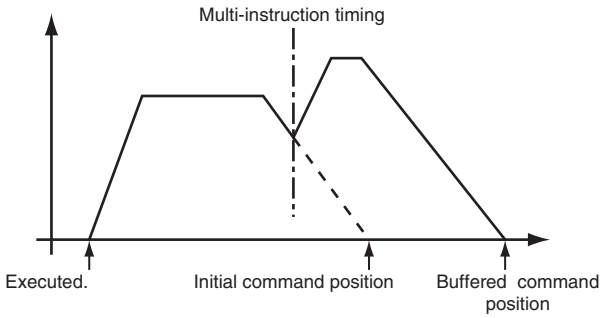
This is the default mode. No buffering is performed in this mode.
The current command is aborted and the new instruction is executed.
Multi-execution of motion control instructions that have no *BufferMode* input variable will operate in Aborting Mode.
Operation of the new instruction starts at the current interpolation velocity when the instruction is triggered.
With Aborting Mode you cannot combine single-axis control, including synchronized single-axis control, and axes group control.

An instruction error will occur at the time of multi-execution if you execute an axes group operation on an axis currently in a single-axis motion.
This will stop both the axes group and the single axis.

Multi-execution during Constant-velocity Motion

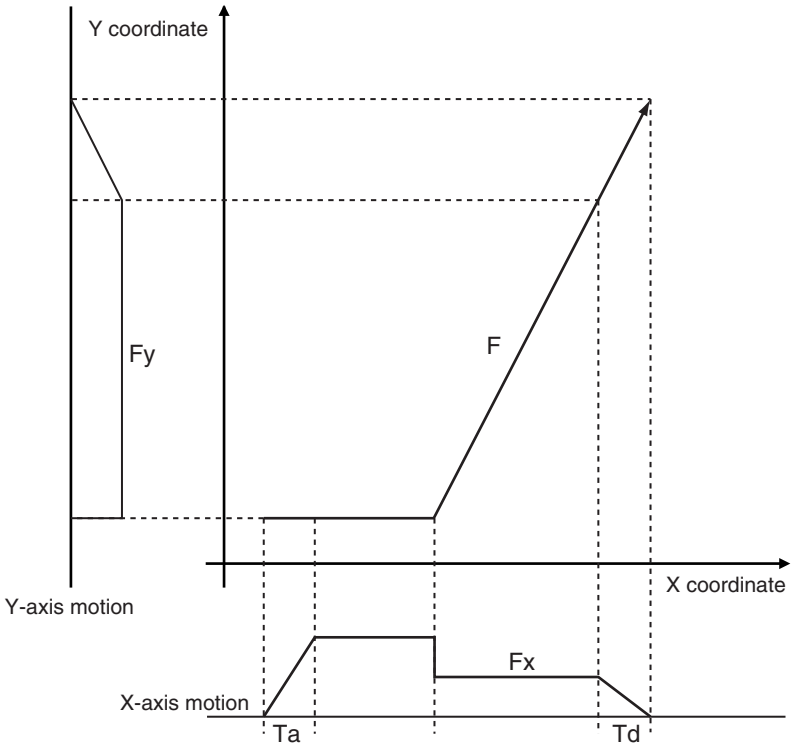


Multi-execution during Acceleration/Deceleration



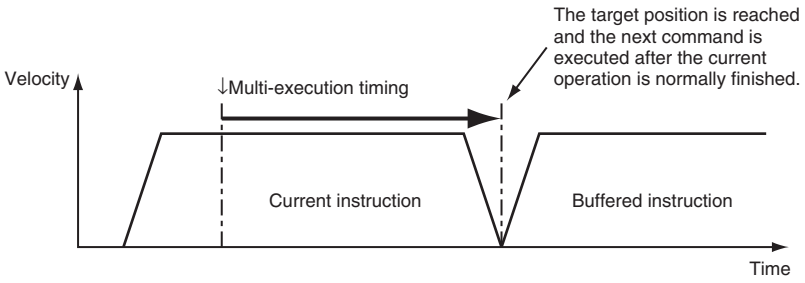
Multi-execution for axes groups is done so that the interpolation velocity remains continuous between instructions. If continuous operation is performed with an instruction with a travel distance of 0, the velocity changes for the axes will not be continuous.

● Example: Interpolation Velocity and Velocities of Axes for Two-axis Cartesian Coordinates



Buffered

The next instruction remains in the buffer until the current operation is finished.
The buffered instruction is executed after the operation for the current instruction is normally ended.



Blending

Blending for axes groups works in the same way as blending for single-axis operations.
The buffered instruction remains in the buffer until the target position of the current instruction is reached.
The buffered instruction is executed after the target position of the current instruction is reached. The axes do not stop at the target position. The two motions are blended together at the interpolation velocity specified with the *BufferMode* input variable
The **Interpolation Acceleration/Deceleration Over** axes group parameter is used to select one of the following operations for when the acceleration/deceleration that is specified in the buffered instruction would exceed the target position.

- Use rapid acceleration/deceleration. (Blending is changed to Buffered.)
- Use rapid acceleration/deceleration.
- Minor fault stop

Version Information

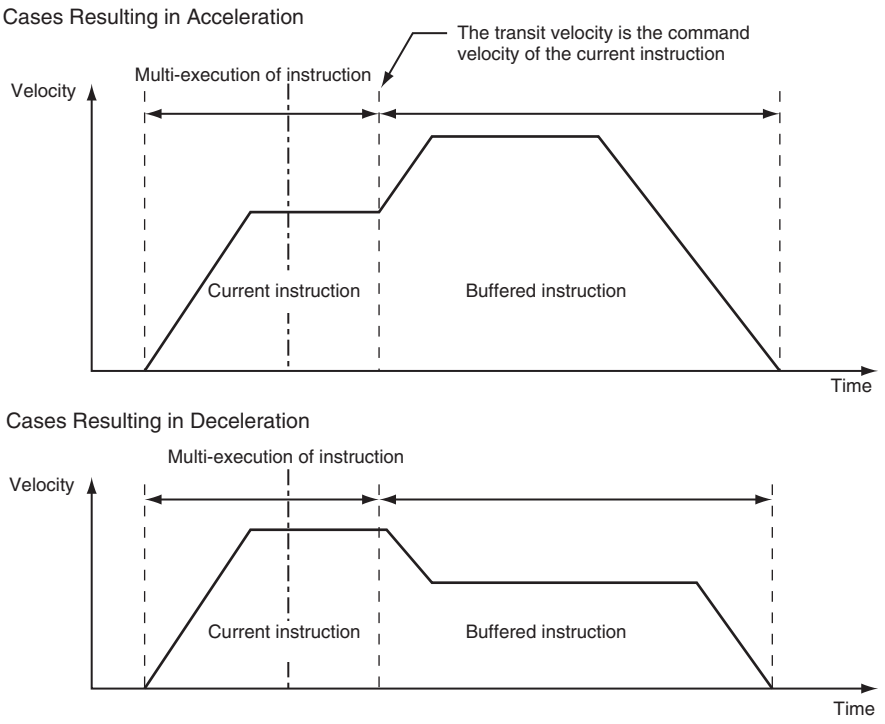
- For a CPU Unit with unit version 1.10 or later, Blending is not changed to Buffered even if you select **Use rapid acceleration/deceleration (Blending is changed to Buffered)**. In this case, the maximum acceleration/deceleration rate is used and the blending operation is continued.
  Also, the axis does not stop with an error even if you select **Minor fault stop**. Similar to the previous case, the maximum acceleration/deceleration rate is used and the blending operation is continued.
- Note that the following restriction applies to CPU Units with unit version 1.09 or earlier.
  For blending in multi-axes coordinated control, buffered operation is used if the results of profile processing show that the execution time of the current instruction is less than four control periods. A Notice of Insufficient Travel Distance to Achieve Blending Transit Velocity observation will occur.
  The above restriction does not apply to CPU Units with unit version 1.10 or later.

● Blending Low (Low Velocity)

Operation is performed using the target position of the current instruction and the target velocity that is the lower of the target velocities for the current instruction and buffered instruction.

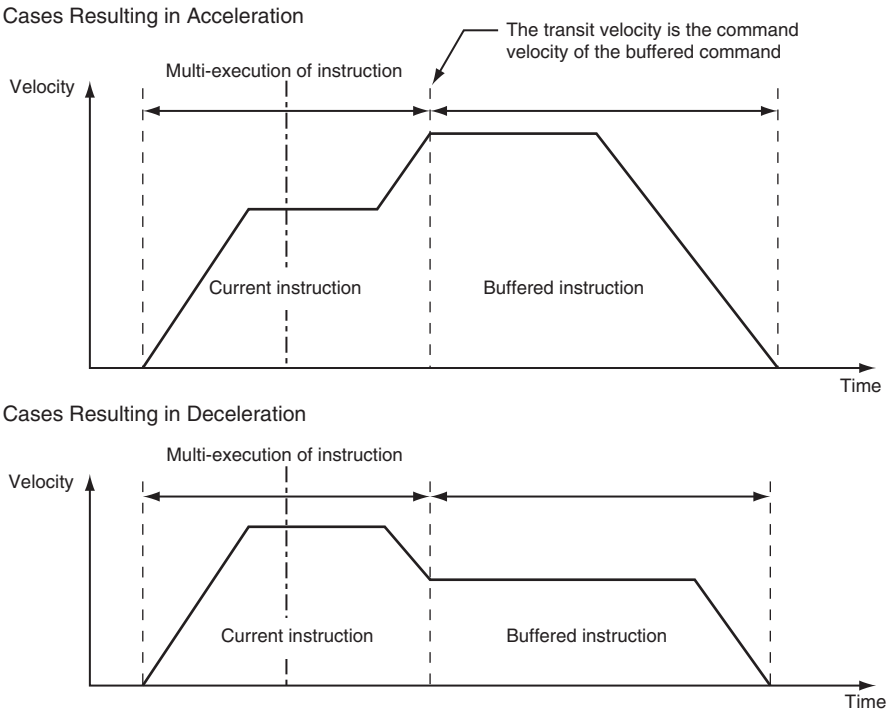
● Blending Previous (Previous Velocity)

Operation is performed with the target velocity of the current instruction until the target position of the current instruction is reached. Operation is performed after acceleration/deceleration to the target velocity of the buffered instruction once the target position is reached.



● Blending Next (Next Velocity)

Operation is performed using the target position of the current instruction and the target velocity of the buffered instruction.



● Blending High (High Velocity)

Operation is performed using the target position of the current instruction and the target velocity that is the higher of the target velocities for the current instruction and buffered instruction.

Transition Modes

Multi-execution of instructions for axes groups may create some shock on the device and/or workpiece due to changes in the direction of the interpolation path. You can specify the *TransitionMode* input variable to the motion control instruction to select a transition method to use between instructions in order to lessen this shock.

You can choose from the following transition modes in the MC Function Module.

No.	Transition mode	Description
0	transition disabled (_mcTMNone)	Do not perform any processing for transitions (default). No attempt is made to lessen the shock, but this results in a shorter operation time.
10	Superimpose Corners (_mcTMCornerSuperimposed)	The deceleration of the current instruction is superimposed on the acceleration of the buffered instruction. You can keep the linear velocity of the interpolation path constant.

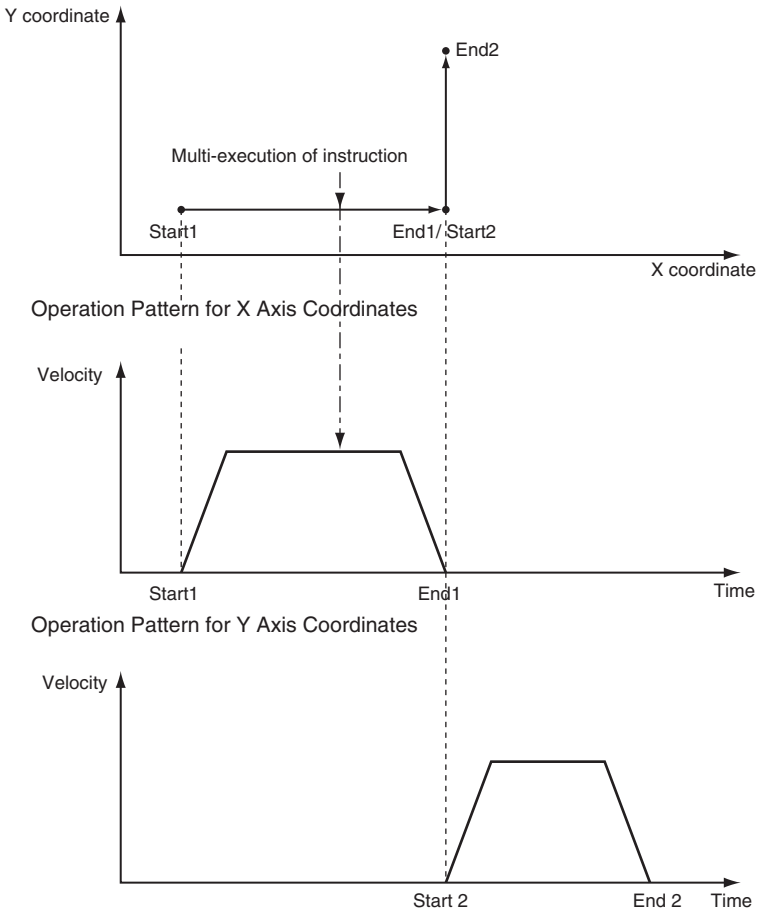
Additional Information

The PLCopen® technology specifications define numbers 0 through 9. Number 10 is unique to the MC Function Module.

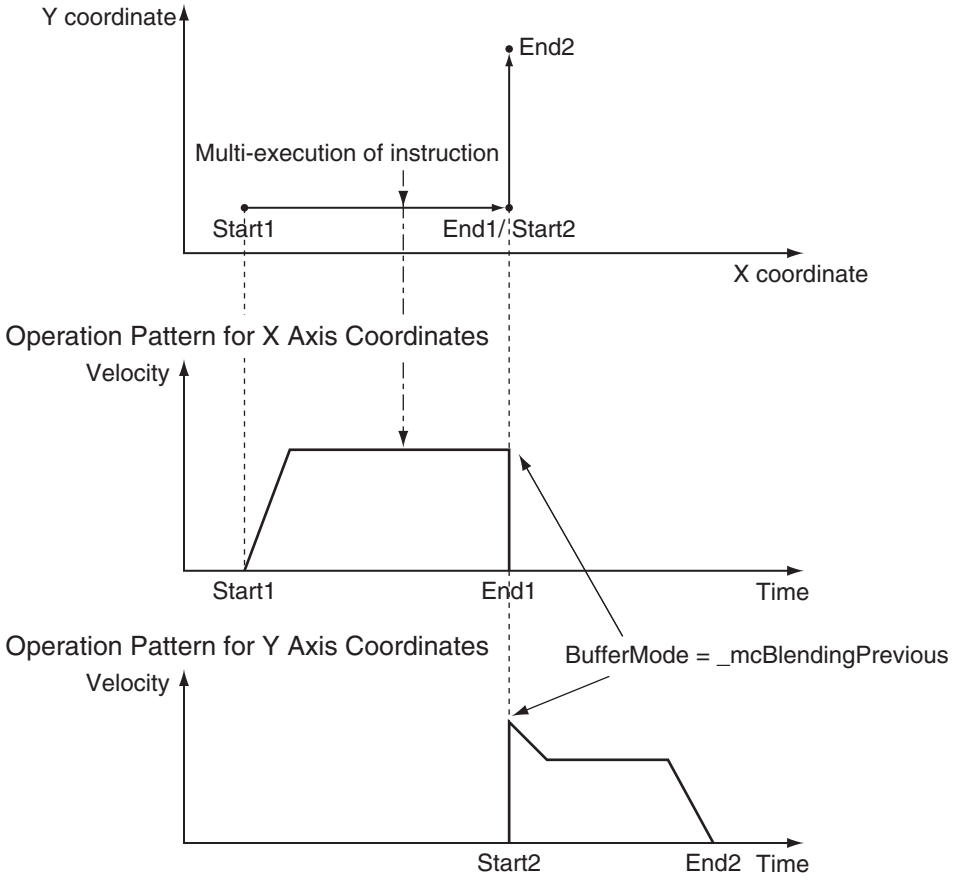
● Transition Disabled (0: _mcTMNone)

No processing is performed to connect the two positions.

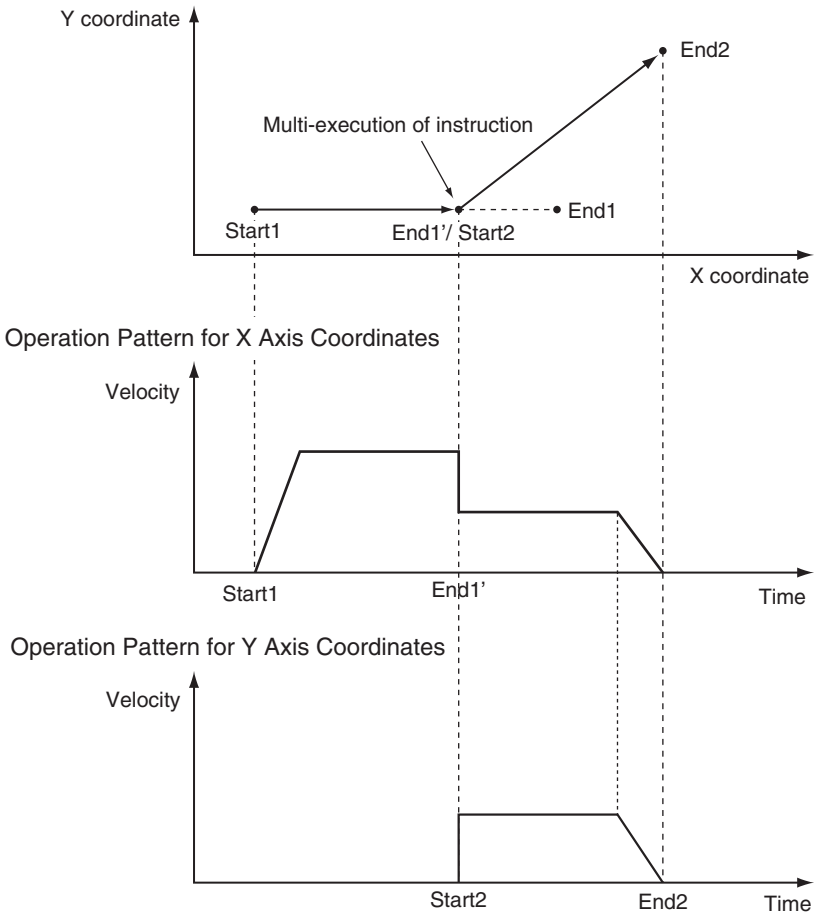
- When **BufferMode** (Buffer Mode Selection) is **1: _mcBuffered**, the axis moves to position End1, stops, and then moves to position End2.



- When **BufferMode** (Buffer Mode Selection) is Blending, the axis moves to position End1, and then moves to position End2.



- When **BufferMode** (Buffer Mode Selection) is **0: _mcAborting**, the axis moves from End1' (multi-execution of instruction) to End2.

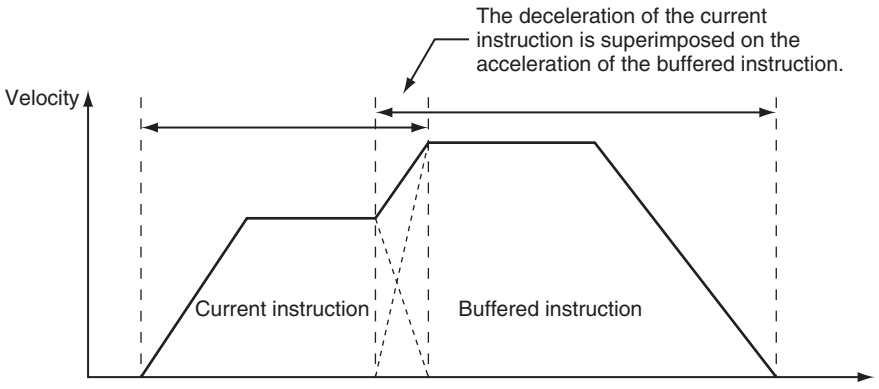


● Superimpose Corners (10: _mcTMCornerSuperimposed)

The deceleration of the current instruction is superimposed on the acceleration of the buffered instruction.

Operation is executed in the same amount of time as for the deceleration of the current instruction, no matter what is specified as the acceleration for the buffered instruction.

The superimposed area will apply no jerk even if jerk is specified.



The output variable *Done*, which indicates the end of a motion control instruction, will change to TRUE for **_mcTMCornerSuperimposed** when the area of superimposition is completed.

Additional Information

The path linear velocity is constant if the following two conditions are met.
- The target velocities of the current instruction and the buffered instruction are the same.
- The deceleration rate of the current instruction and the acceleration rate of the buffered instruction are the same.

Combining Transition Modes and Multi-execution of Instructions

The following table shows the combinations of Transition Modes and Buffer Modes.

○: Operation possible, ---: Generates an error and stops

Transition Mode	Buffer Mode					
	Aborting	Buffered	Blending Low	Blending Previous	Blending Next	Blending High
Transition Disabled (_mcTMNone)	○	○	○	○	○	○
Superimpose Corners*1 (_mcTMCornerSuperimposed)	---	---	○	○	○	○

*1. For superimpose corners, the deceleration for the current instruction and the acceleration for the buffered instruction will be superimposed.

9-8 Other Functions

This section describes other functions of the MC Function Module.
Refer to the *NX-series Position Interface Units User's Manual (Cat. No. W524)* for the differences when you use NX-series Pulse Output Units.

9-8-1 Changing the Current Position

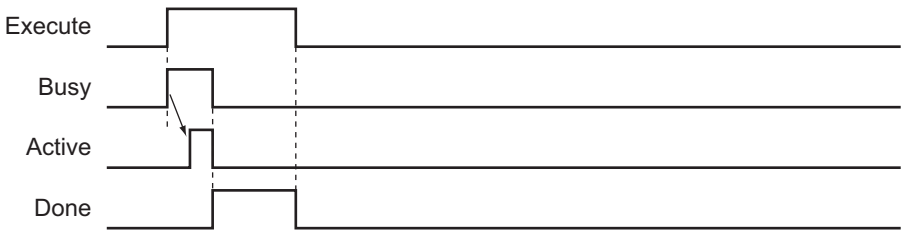
The command current position of a Servo axis can be changed to a specified value. The actual current position changes to a value that maintains the current following error with the command current position.
For an encoder axis, you can change the actual current position.
Use the MC_SetPosition instruction to specify the actual position you want to modify.

You can change the actual position even while an axis is in motion.
If positioning to an absolute value is being executed, positioning will be performed to the target position using the new absolute coordinates.
However, the travel distance will stay the same when you position to a relative value.

Precautions for Correct Use

- When the **Count Mode** is **Rotary Mode**, an instruction error will occur if you specify a position outside the ring counter range.
- After changing the current position the home will be undefined and you will not be able to use the following functions and instructions.
  Software limits
  High-speed homing
  Interpolation instructions (linear and circular interpolation)

● Timing Chart for Execution While Axis Is Stopped



Additional Information

You can change the actual position while home is defined by specifying a zero position preset for the MC_Home or MC_HomeWithParameter instruction.

For details on the MC_SetPosition instruction, refer to the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)*.

9-8-2 Torque Limit

The output torque is limited by enabling and disabling the torque limit function of the Servo Drive and by setting the torque limit value.
Different limits can be specified for the positive torque limit and negative torque limit.

For details, refer to the MC_SetTorqueLimit (Set Torque Limit) and MC_SetTorqueLimit2 (Set Torque Limit 2) instructions in the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)*.

Precautions for Correct Use

You cannot use the torque limit function for an NX-series Pulse Output Unit.

9-8-3 Latching

Latching is used to control positioning based on the position 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.
You can set up to two trigger signals for each axis.
Use the MC_TouchProbe (Enable External Latch) instruction to specify the *Trigger Input Condition* variable, *Window Only* variable, and *Stopping Mode Selection* variable for the axis you want to latch.
In addition to signals that connect to the Servo Drive, you can also specify variables in the user program to use as a trigger.
Use the MC_AbortTrigger (Disable External Latch) instruction to abort latching.
You can use latching only with a Servo Drive that support latching (touch probe), such as the OMRON 1S-series Servo Drives, or a GX-EC0211/EC0241 Encoder Input Terminal.

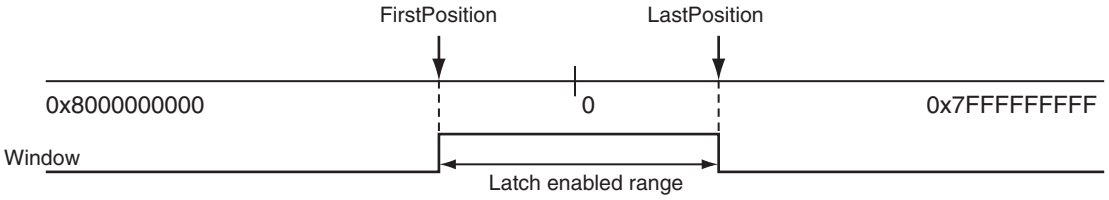
Use WindowOnly to detect only trigger signals within a specific start point and end point.

For details on latching, refer to the MC_TouchProbe (Enable External Latch) and MC_AbortTrigger (Disable External Latch) instructions in the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)*.

Refer to the *NX-series Position Interface Units User's Manual (Cat. No. W524)* for the differences when you use NX-series Pulse Output Units.

● Linear Mode

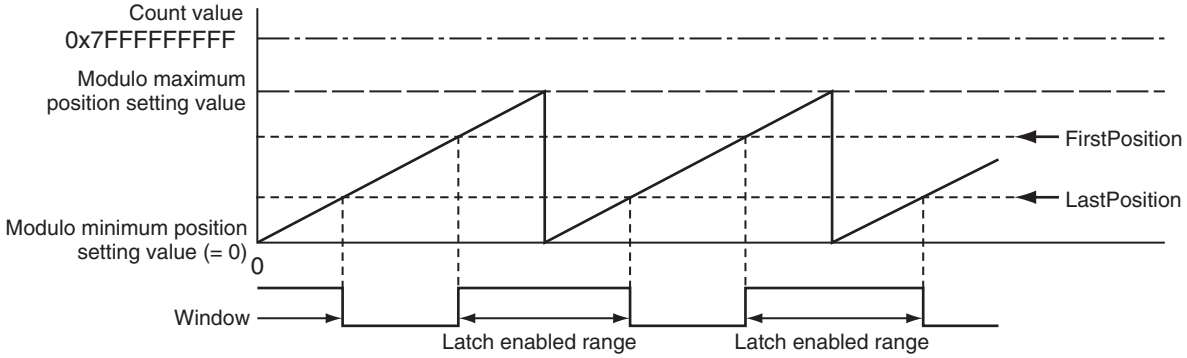
- The FirstPosition must be less than or equal to the LastPosition.
- An instruction error will occur if the FirstPosition is greater than the LastPosition.
- An instruction error will occur if a position beyond the position range of Linear Mode is specified.

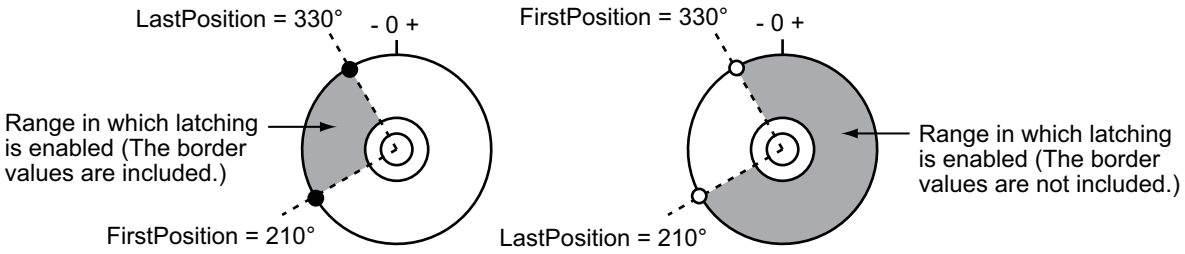


● Rotary Mode

- The FirstPosition can be either equal to or less than the LastPosition, or greater than the LastPosition.
  If the FirstPosition is greater than the LastPosition, the setting will straddle the modulo minimum position setting value.
- An instruction error will occur if a position beyond the upper and lower limits of the ring counter is specified.

	First Position ≤ Last Position	First Position > Last Position
Valid range	FirstPosition to LastPosition	LastPosition to FirstPosition





9-8-4 Zone Monitoring

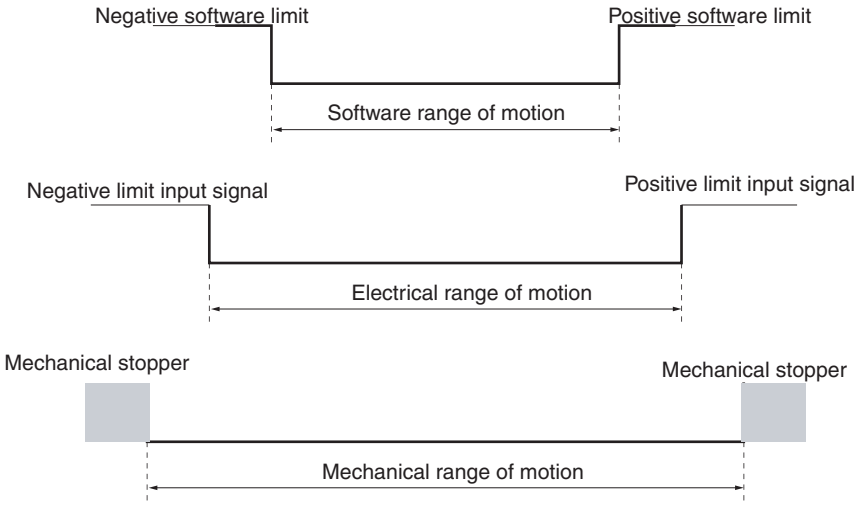
This function detects whether the command position or actual position of an axis is in the specified range (zone).
Use the MC_ZoneSwitch (Zone Monitor) instruction to specify the first position and last position of the zone to check.
The *InZone* output variable for the Zone Monitor instruction will change to TRUE when the position of the axis enters the specified zone.
You can also specify multiple zones for a single axis. Zones can overlap.

For details on zone monitoring, refer to the MC_ZoneSwitch (Zone Monitor) instruction in the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)*.

9-8-5 Software Limits

Actual positions can be monitored in the MC Function Module software. This function is separate from the hardware-based limit input signals.

Set the range to monitor by setting the software limits in the **Positive Software Limit** and **Negative Software Limit** axis parameters.
During normal positioning, motion is possible within the range of these software limits.
Set software limits to prevent potential damage to machinery caused by mistakes in the user program or improper operation.



Axis Parameters That Are Related to Software Limits

Parameter name	Function	Setting range	Default
Software Limits	Select the software limit function. 0: Disabled 1: Deceleration stop for command position*1 2: Immediate stop for command position 3: Deceleration stop for actual position*1 4: Immediate stop for actual position	0 to 4	0
Positive Software Limit	Set the software limit in the positive direction. The unit is command units.	Long reals*2	2,147,483,647
Negative Software Limit	Set the software limit in the negative direction. The unit is command units.		-2,147,483,648

*1. If the actual position goes beyond a software limit during execution of a movement instruction that has a *Deceleration* input variable, the axis decelerates to a stop at the deceleration rate given by *Deceleration*. If the actual position goes beyond a software limit during execution of a movement instruction that does not have a *Deceleration* input variable, the axis decelerates to a stop at the maximum deceleration that is set in the axis parameters.
*2. Positions can be set within a 40-bit signed integer range when converted to pulses.

You can use the axis settings of the Sysmac Studio, the MC_Write (Write MC Setting) instruction, or the MC_WriteAxisParameter (Write Axis Parameters) instruction to set the above axis parameters.
If any setting values are changed for an axis or axes group in operation, those settings are enabled when the next operation begins.
Depending on the operating state of the axis and the motion control instruction, the software limit works when the motion instruction is executed and when the axis is operating.

For details on the instruction to write the MC settings and the instruction to write the axis parameters, refer to the MC_Write instruction and MC_WriteAxisParameter instruction in the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)*.

● Executing Motion Instructions

- When the actual position is within the software limits, an instruction error will occur if the target position is outside the software limit range.
- When the actual position is outside the software limits, motion is allowed only toward the software limit range. As long as the motion is toward the range, the target position does not need to be within the software limit range.

Precautions for Correct Use

Do not execute an instruction for an axis command for a target position that is outside of the software limit range.

● During Axis Motion

When the axis is in discrete motion, synchronized motion, continuous motion, or coordinated motion:

- An axis error will occur if the software limits are **enabled for the command position** and the command position leaves the range.
- An axis error will occur if the software limits are **enabled for the actual position** and the actual position leaves the range.

Additional Information

Software limits can be enabled when the **Count Mode** is set to **Linear Mode** and home is defined.
Software limits are disabled in the following situations no matter what axis parameters have been set.

- When **Count Mode** is set to **Rotary Mode**.
- When home is not defined.
- During homing.

9-8-6 Following Error Monitoring

Following error is the difference between the command position and the actual position of an axis. The MC Function Module monitors the following error every motion control period.
If the value of the following error exceeds the **Following Error Over Value** that is set in the axis parameters, Following Error Limit Exceeded minor fault level error occurs. If it exceeds the **Following Error Warning Value**, a Following Error Warning observation occurs. Monitoring the following error is disabled during execution of the holding operation for homing.

● Axis Parameters That Are Related to Monitoring the Following Error

You can set the check values for monitoring the following error by setting the appropriate axis parameters. Set the **Following Error Warning Value** so that it is less than the **Following Error Over Value**.
Set the axis parameters from the Sysmac Studio.

Parameter name	Function	Setting range	Default
Following Error Over Value	Set the excessive following error check value. Set *0* to disable the excessive following error check. (Unit: command units)	Non-negative long reals	0
Following Error Warning Value	Set the following error warning check value. Set *0* to disable the following error warning check. (Unit: command units)	Non-negative long reals that are less than or equal to the Following Error Over Value	0

Refer to the *NX-series Position Interface Units User's Manual (Cat. No. W524)* for the differences when you use NX-series Pulse Output Units.

9-8-7 Following Error Counter Reset

Resetting the following error counter resets the following error to 0.
Use the MC_ResetFollowingError instruction in the user program to reset the following error counter.
You can use the MC_ResetFollowingError instruction for each axis during positioning or during homing.
If you execute a following error counter reset while the axis is in motion, the current motion control instruction will be aborted and the command position will be set to the same value as the actual position.
The home will remain defined even after executing a following error counter reset.

For details on resetting the following error counter, refer to the MC_ResetFollowingError instruction in the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)*.

● Axis Parameters That Are Related to Resetting the Following Error Counter

You can choose to reset the following error counter on an immediate stop, on a limit input stop, or after homing is completed by setting the appropriate axis parameters.
Set the axis parameters from the Sysmac Studio.

Parameter name	Function	Setting range	Default
Immediate Stop Input Stop Method	Set the stopping method in the MC Function Module when the immediate stop input is enabled. 0: Immediate stop 2: Immediate stop and error reset 3: Immediate stop and Servo OFF	0, 2, or 3	0
Limit Input Stop Method	Set the stopping method in the MC Function Module when the positive limit input or negative limit input is enabled. 0: Immediate stop 1: Deceleration stop 2: Immediate stop and error reset 3: Immediate stop and Servo OFF	0 to 3	0

Refer to the *NX-series Position Interface Units User's Manual (Cat. No. W524)* for the differences when you use NX-series Pulse Output Units.

9-8-8 Axis Following Error Monitoring

You can monitor the amount of following error for the command position or the actual position between two axes.
Use the MC_AxesObserve (Monitor Axis Following Error) instruction to specify the permitted following error and the two axes to monitor.
If the permitted following error is exceeded, the *Invalid* output variable for the Monitor Axis Following Error instruction will change to TRUE.

You can use this monitoring function to program the actions to take when the following error between axes grows too large for gantry control and other devices where both axes perform the same operation.

Precautions for Correct Use

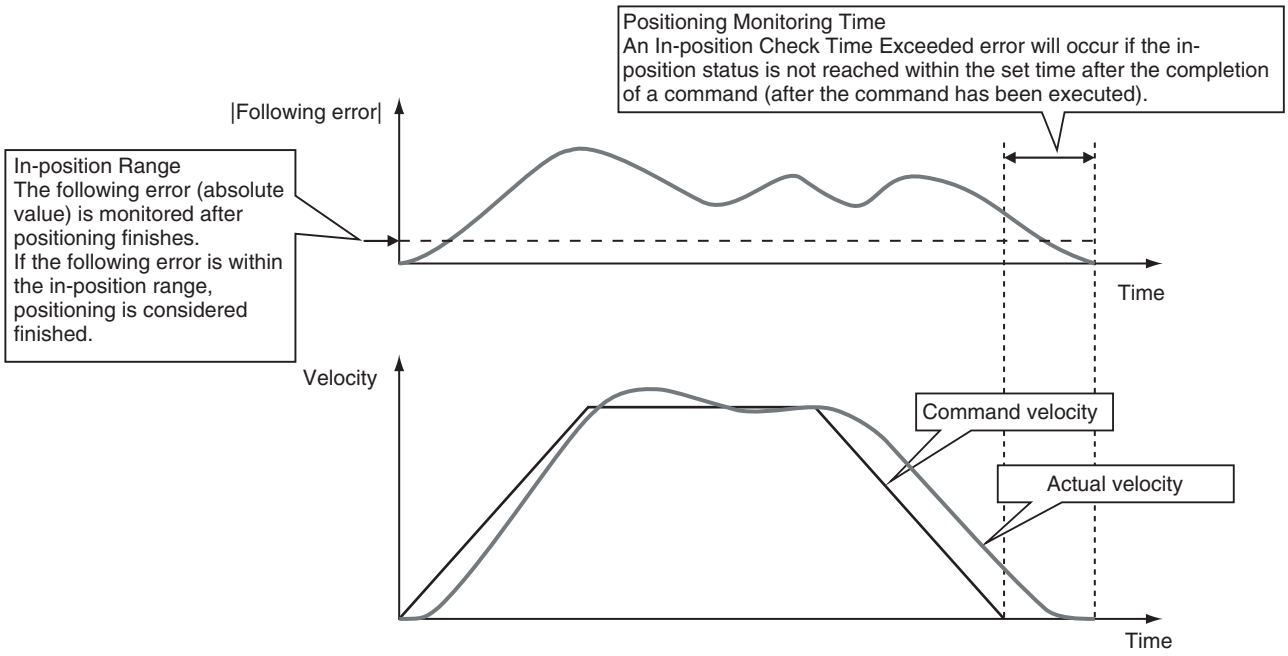
Even if the permitted following error between axes is exceeded, no error will occur in the MC Function Module.
Check the *Invalid* output variable to stop axis operation or to take some other action as appropriate in the user program.

For details on axis following error monitoring, refer to the MC_AxesObserve (Monitor Axis Following Error) instruction in the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)*.

Refer to the *NX-series Position Interface Units User's Manual (Cat. No. W524)* for the differences when you use NX-series Pulse Output Units.

9-8-9 In-position Check

You can check to see if the actual current position has reached the specified range for the target position during positioning or homing.
After command output of the target position is completed, positioning is considered to be finished when the difference between the target position and the actual current position is within the **in-position range**.
An instruction error occurs if positioning is not finished within the **in-position check time**.



● Axis Parameters That Are Related to In-position Checks

You can set the check conditions for the in-position check by setting the appropriate axis parameters.

Set the in-position check time if you want to start any of the following operations only after confirming that axes are in position.

Parameter name	Function	Setting range	Default
In-position Range	Set the in-position width. (Unit: command units)	Non-negative long reals	10
In-position Check Time	Set the in-position check time in milliseconds. Set *0* to check for the end of positioning only when you define the home position during homing and not check positioning at other times. (Unit: ms)	0 to 10,000	0

You can use the axis settings of the Sysmac Studio, the MC_Write (Write MC Setting) instruction, or the MC_WriteAxisParameter (Write Axis Parameters) instruction to set the above axis parameters.

Additional Information

- The in-position check is processed by the MC Function Module. The function in the Servo Drive is not used.
- Do not set an in-position check time if you want to start the next operation as quickly as possible without waiting for positioning to finish.
- The value set from the Sysmac Studio is restored if power to the CPU Unit is cycled or the user program is downloaded with the Synchronization menu command of the Sysmac Studio. Use the MC_Write (Write MC Setting) and MC_WriteAxisParameter (Write Axis Parameters) instructions only when you need to temporarily change the in-position check time.

● Monitor Information That Is Related to In-position Checks

You can read Axis Variables from the user program to monitor when positioning finishes.

In the descriptions, a variable name _MC_AX[*] is used as an example, but the same information applies to _MC1_AX[*] and _MC2_AX[*].

Variable name	Data type	Meaning	Function
_MC_AX[0-255].Details.Idle	BOOL	Idle	TRUE when processing is not currently performed for the command value, except when waiting for in-position state. *1 Idle and InPosWaiting are mutually exclusive. They cannot both be TRUE at the same time.
_MC_AX[0-255].Details.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.

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

You can read Axes Group Variables from the user program to monitor when positioning finishes for the axes group.
In the descriptions, a variable name _MC_GRP[*] is used as an example, but the same information applies to _MC1_GRP[*] and _MC2_GRP[*].

Variable name	Data type	Meaning	Function
_MC_GRP[0-63].Details.Idle	BOOL	Idle	TRUE when processing is not currently performed for the command value, except when waiting for in-position state. *1 Idle and InPosWaiting are mutually exclusive. They cannot both be TRUE at the same time.
_MC_GRP[0-63].Details.InposWaiting	BOOL	In-position Waiting	TRUE when waiting for in-position state for any composition axis. *2 The in-position check is performed when positioning for the in-position check.

*1. This also includes states where processing is performed while in motion at a velocity of 0.
*2. This variable is FALSE when all composition axes in the axes group are within the in-position ranges set in the axis parameters.

For details on the instruction to write the MC settings and the instruction to write the axis parameters, refer to the MC_Write (Write MC Setting) and MC_WriteAxisParameter (Write Axis Parameters) instruction in the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)*.

Refer to the *NX-series Position Interface Units User's Manual (Cat. No. W524)* for the differences when you use NX-series Pulse Output Units.

9-8-10 Changing Axis Use

You can use the MC_ChangeAxisUse (Change Axis Use) instruction to temporarily change the setting of the **Axis Use** axis parameter. To change an axis in this way, it must be set as a **Used axis** or as an **Unused axis (changeable to used axis)** in the **Axis Use** axis parameter.

If the **Axis Use** axis parameter is set to **Unused axis (changeable to used axis)** and the **Axis Type** parameter is set to a **servo axis** or **virtual servo axis**, you can set the axis in an axes group.
A CPU Unit with unit version 1.04 or later and Sysmac Studio version 1.05 or higher are required.

Precautions for Correct Use

- Do not attempt to change an axis that is set to **Unused axis (unchangeable to used axis)** to a **Used axis**.
- You cannot set an axis in an axes group if the **Axis Use** axis parameter is set to **Unused axis (unchangeable to used axis)**.

For details, refer to the MC_ChangeAxisUse instruction in the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)*.

For an application example of the MC_ChangeAxisUse instruction, refer to the *NJ/NX-series CPU Unit Software User's Manual (Cat. No. W501)*.

9-8-11 Enabling Digital Cam Switch

You can use the MC_DigitalCamSwitch (Enable Digital Cam Switch) instruction to turn the digital outputs ON or OFF according to the axis position.
The setting of the *ValueSource* input variable to the instruction also allows you to adjust for the acceleration or deceleration rate.
Always use this function together with the NX_AryDOutTimeStamp instruction and with a Digital Output Unit that supports time stamp refreshing. The NX_AryDOutTimeStamp instruction turns the specified digital outputs ON or OFF at specified timing of the time stamp.
A CPU Unit with unit version 1.06 or later and Sysmac Studio version 1.07 or higher are required to use this function.

Precautions for Correct Use

You can use this instruction for an axis that is assigned to an NX-series Position Interface Unit. The NX Units that can be used are NX-EC0□□□ and NX-ECS□□□, also must be running the time stamping.

Refer to the MC_DigitalCamSwitch (Enable Digital Cam Switch) instruction in the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)* for details on enabling digital cam switch.

Refer to the *NJ/NX-series Instructions Reference Manual (Cat. No. W502)* for details on NX_AryDOutTimeStamp instruction.

Refer to the *NX-series Digital I/O Units User's Manual (Cat. No. W521)* for Digital Output Unit that supports time stamp refreshing.

Refer to the *NX-series Position Interface Units User's Manual (Cat. No. W524)* for time stamping and time stamps.

9-8-12 Displaying 3D Motion Monitor for User Coordinate System

In the case that coordinate systems (such as SCARA robot and vertical articulated robot) other than orthogonal coordinate system are implemented by user programs, this function can be used to display the path of robot hands, etc. in 3D with Sysmac Studio.
You can create an _sMC_POSITION_REF type user-defined variable and display in 3D Motion Monitor Display Mode.

A CPU Unit with unit version 1.06 or later and Sysmac Studio version 1.07 or higher are required to use this function.

● _sMC_POSITION_REF

The followings are the members of _sMC_POSITION_REF type data.

Member	Data type	Meaning
CommandPosition	ARRAY[0..5] OF LREAL	Command Current Position
ActualPosition	ARRAY[0..5] OF LREAL	Actual Current Position

The following list describes each member.

Member	Description
User-defined variable.CommandPosition[0]	This is an X-axis component for the command current position. This member is assigned a user-defined variable that indicates the X-axis position of the command current position generated by a user program.
User-defined variable.CommandPosition[1]	This is a Y-axis component for the command current position. This member is assigned a user-defined variable that indicates the Y-axis position of the command current position generated by a user program.
User-defined variable.CommandPosition[2]	This is a Z-axis component for the command current position. This member is assigned a user-defined variable that indicates the Z-axis position of the command current position generated by a user program.
User-defined variable.CommandPosition[3] to [5]	Not used.
User-defined variable.ActualPosition[0]	This is an X-axis component for the actual current position. This member is assigned a user-defined variable that indicates the X-axis position of the actual current position handled in a user program.
User-defined variable.ActualPosition[1]	This is a Y-axis component for the actual current position. This member is assigned a user-defined variable that indicates the Y-axis position of the actual current position handled in a user program.
User-defined variable.ActualPosition[2]	This is a Z-axis component for the actual current position. This member is assigned a user-defined variable that indicates the Z-axis position of the actual current position handled in a user program.
User-defined variable.ActualPosition[3] to [5]	Not used.

Each member is assigned a user-defined variable. The followings are the examples.

Name	Data type	Description
3D_position	_sMC_POSITION_REF	User-defined variable for 3D display
MCS_Cmd_TransX	LREAL	User-defined variable that indicates the X-axis position of the command current position generated by a user program
MCS_Cmd_TransY	LREAL	User-defined variable that indicates the Y-axis position of the command current position generated by a user program
MCS_Cmd_TransZ	LREAL	User-defined variable that indicates the Z-axis position of the command current position generated by a user program
MCS_Act_TransX	LREAL	User-defined variable that indicates the X-axis position of the actual current position handled in a user program
MCS_Act_TransY	LREAL	User-defined variable that indicates the Y-axis position of the actual current position handled in a user program
MCS_Act_TransZ	LREAL	User-defined variable that indicates the Z-axis position of the actual current position handled in a user program

```
3D_position.CommandPosition[0] := MCS_Cmd_TransX;
3D_position.CommandPosition[1] := MCS_Cmd_TransY;
3D_position.CommandPosition[2] := MCS_Cmd_TransZ;
3D_position.ActualPosition[0] := MCS_Act_TransX;
3D_position.ActualPosition[1] := MCS_Act_TransY;
3D_position.ActualPosition[2] := MCS_Act_TransZ;
```

● Overview of Operating Procedures

1 Create an _sMC_POSITION_REF type user-defined variable.

2 Create a program in which user-defined variables that indicate the command current position and actual current position for 3D display are assigned to each member of the created user-defined variable.

3 Select **Specified coordinate** in the Type Box in the 3D Machine Model List.
The _sMC_POSITION_REF data type is displayed in the 3D Machine Model Parameter Settings section.

4 Set the created user-defined variable in the Value Column in the 3D Machine Model Parameter Settings section.

5 Execute the user program.

6 Start tracing the data with the data trace to sample the data.

7 Check the trace results on the Data Trace Tab Page.

Refer to the *Sysmac Studio Version 1 Operation Manual (Cat. No. W504)* for details on 3D Motion Monitor Display Mode.

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.

10-1 Overview of Sample Programming

This section provides information that applies to all of the sample programming.

Precautions for Correct Use

- The sample programming that is provided includes only programming that uses the MC Function Module.
- When programming actual applications, also program device interlocks, I/O with other devices, and other control procedures.
- Create a user program that will produce the intended device operation.
- Check the user program for proper execution before you use it for actual operation.
- Refer to the *NX-series Position Interface Units User's Manual (Cat. No. W524)* for application examples for the NX-series Position Interface Units.

10-1-1 Devices

The following devices are used in the sample programming.

Device	Servo configuration example
CPU Unit	NJ501-1□00 (unit version 1.0)
Power Supply Unit	NJ-Px3001
Servo Drive	R88D-1SN□□□-ECT
Servomotor	R88M-1□□□□□□□
Encoder Input Terminal	GX-EC0211 (version 1.1)

10-1-2 Installation and Wiring

Refer to the following manual for details on installing and wiring the devices.

Device	Manual
CPU Unit and Power Supply Unit	NJ-series CPU Unit Hardware User's Manual (Cat. No. W500)
Servo Drive and Servomotor	AC Servomotors/Servo Drives 1S-series with Built-in EtherCAT Communications User's Manual (Cat. No. I586)
Encoder Input Terminal	GX-series EtherCAT Slave Units User's Manual (Cat. No. W488)
EtherCAT communications cables	

10-1-3 Setup

Refer to the following manual for details on settings.

Setup	Manual
Controller Setup	NJ/NX-series CPU Unit Software User's Manual (Cat. No. W501)
Motion Control Setup	*3-2 Axis Setting Procedure* on page 3-10, *5-2 Axis Parameters* on page 5-5, and *A-1 Connecting the 1S-series Servo Drive* on page A-2 in this manual.
Servo parameters	AC Servomotors/Servo Drives 1S-series with Built-in EtherCAT Communications User's Manual (Cat. No. I586)

10-2 Basic Programming Samples

This section provides programming samples for the basic functions of the MC Function Module.

Precautions for Correct Use

- When you use these programming samples for reference, be sure to add programming for suitable interlocks that suit the operating conditions of the devices.
- Enter the variables that are used in the programming samples from the **Programming** Layer in the Edit Pane of the Sysmac Studio.

10-2-1 Monitoring EtherCAT Communications and Turning ON Servos

In this sample, the MC_Power (Power Servo) instruction is executed to turn ON the Servo for the Servo Drive when EtherCAT process data communications are established with the Servo Drive.

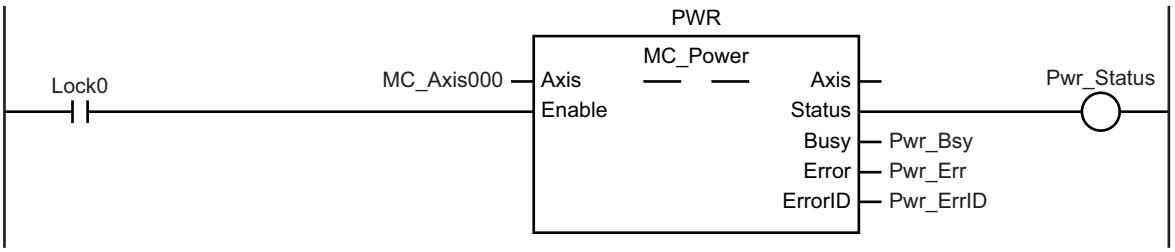
Main Variables Used in the Programming Samples

Variable name	Data type	Default	Comment
MC_Axis000	_sAXIS_REF	---	This is the Axis Variable for axis 0.
MC_Axis000.MFaultLvl.Active	BOOL	FALSE	TRUE when there is a minor fault level error for axis 0.
MC_Axis000.Cfg.NodeAddress	UINT	---	This is the node address.
_EC_PDSlavTbl[N]	BOOL	FALSE	TRUE when EtherCAT process data communications for node address N are in Operational state.
StartPg	BOOL	FALSE	When this variable is TRUE, the Servo is turned ON if EtherCAT process data communications are active and normal.

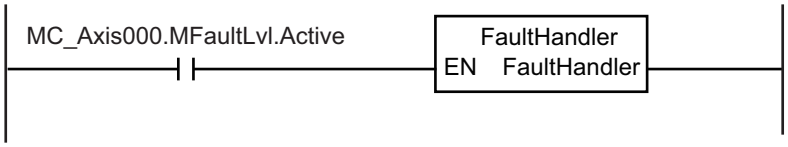
Ladder Diagram

Check if the Servo Drive is ready when *StartPg* is TRUE.

If the Servo Drive is ready, turn ON the Servo for axis 0.



If a minor fault level error occurs for axis 0, the error handler for the device (FaultHandler) is executed.
Program the FaultHandler according to the device.



ST Programming

```
// If the Servo Drive is ready when StartPg is TRUE, turn ON the Servo for axis 0.
IF (StartPg=TRUE)
  AND (MC_Axis000.DrvStatus.Ready=TRUE) THEN
    Pwr_En:=TRUE;
  ELSE
    Pwr_En:=FALSE;
END_IF;

// If a minor fault level error occurs for axis 0, the error handler for the device
 (FaultHandler) is executed.
// Program the FaultHandler according to the device.
IF MC_Axis000.MFaultLvl.Active=TRUE THEN
  FaultHandler();
END_IF;

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

10-2-2 Interlocking Axis Operation with Master Control Instructions

You can place the MC_Power (Power Servo) instruction between the MC (Master Control Start) and MCR (Master Control End) instructions in ladder diagrams to interlock axis operation.
When Mc_On is FALSE in this sample, the MC_Power (Power Servo) instruction between the MC and MCR instructions is disabled to turn OFF the Servo.
The *CommandAborted* output variable from the current motion control instruction changes to TRUE at the same time, and axis motion stops.

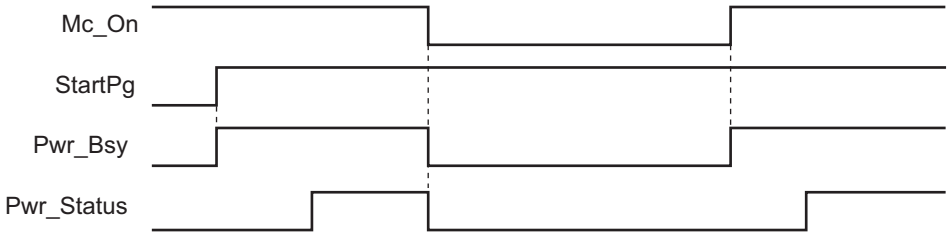
Precautions for Correct Use

You cannot use the MC instruction in ST.

Main Variables Used in the Programming Samples

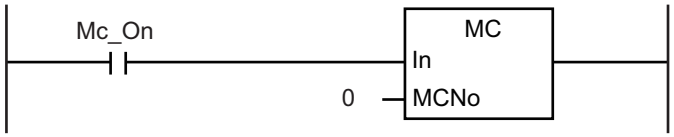
Variable name	Data type	Default	Comment
MC_Axis000	_sAXIS_REF	---	This is the Axis Variable for axis 0.
MC_Axis000.MFaultLvl.Active	BOOL	FALSE	TRUE when there is a minor fault level error for axis 0.
Mc_On	BOOL	FALSE	This variable enables and disables the MC instruction. Control programming is not given in this sample. In actual programming, program controls for the required device operation.
StartPg	BOOL	FALSE	When this variable is TRUE, the Servo is turned ON if EtherCAT process data communications are active and normal.

Timing Chart

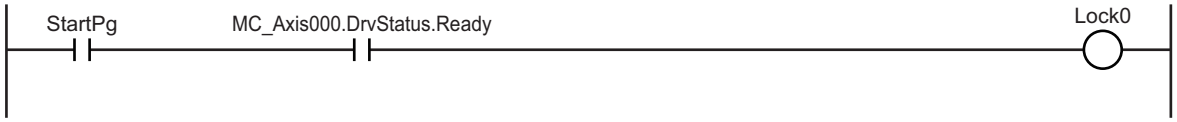


Ladder Diagram

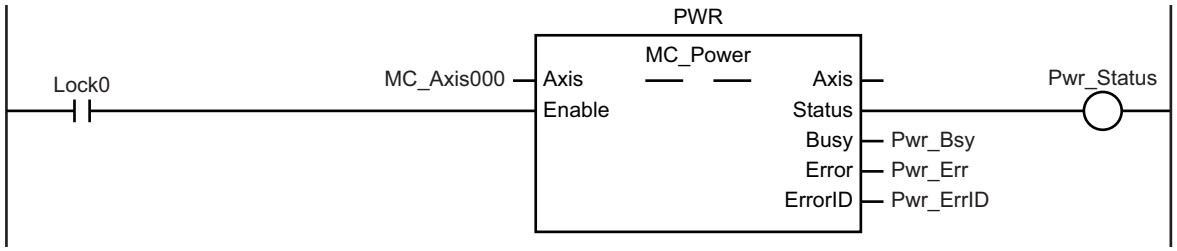
When *Mc_On* is TRUE, master control is started.



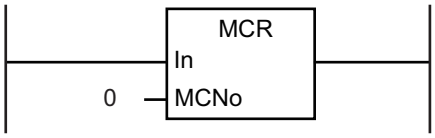
Check if the Servo Drive is ready when *StartPg* is TRUE.



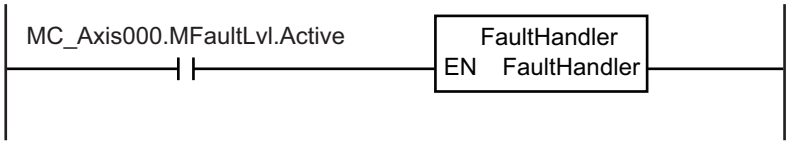
If the Servo Drive is ready, turn ON the Servo for axis 0.



Master control is ended.



If a minor fault level error occurs for axis 0, the error handler for the device (FaultHandler) is executed.
Program the FaultHandler according to the device.



10-2-3 Error Monitoring and Error Resetting for Single-axis Operation and Synchronized Operation

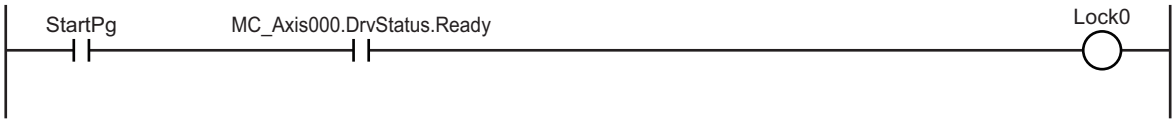
You can monitor error status by monitoring the status of Axis Minor Fault Occurrence in the Axis Variable.
If a minor fault level error occurs in this sample, the *Enable* input variable for the MC_Power instruction changes to FALSE to turn OFF the Servo.
If the external button is ON and the command current velocity is zero, the error is reset with the MC_Reset (Reset Axis Error) instruction.

Main Variables Used in the Programming Samples

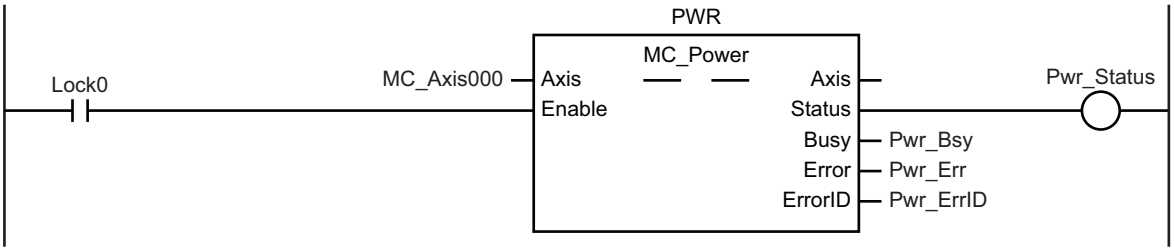
Variable name	Data type	Default	Comment
MC_Axis000	_sAXIS_REF	---	This is the Axis Variable for axis 0.
MC_Axis000.MFaultLvl.Active	BOOL	FALSE	TRUE when there is a minor fault level error for axis 0.
MC_Axis000.Status.ErrorStop	BOOL	FALSE	TRUE while there is a minor fault level error for axis 0 and the axis is decelerating to a stop or stopped.
MC_Axis000.Details.Idle	BOOL	FALSE	TRUE when the command current velocity for axis 0 is zero, except when waiting for in-position state.
StartPg	BOOL	FALSE	When this variable is TRUE, the Servo is turned ON if EtherCAT process data communications are active and normal.
ResetON	BOOL	FALSE	This variable gives the status of the external button that is used to reset errors.

Ladder Diagram

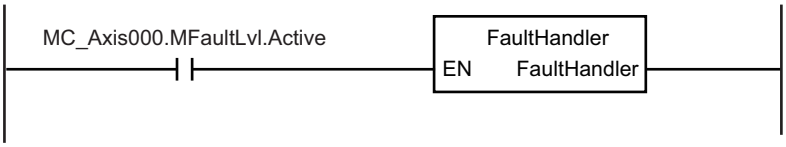
Check if the Servo Drive is ready when *StartPg* is TRUE.



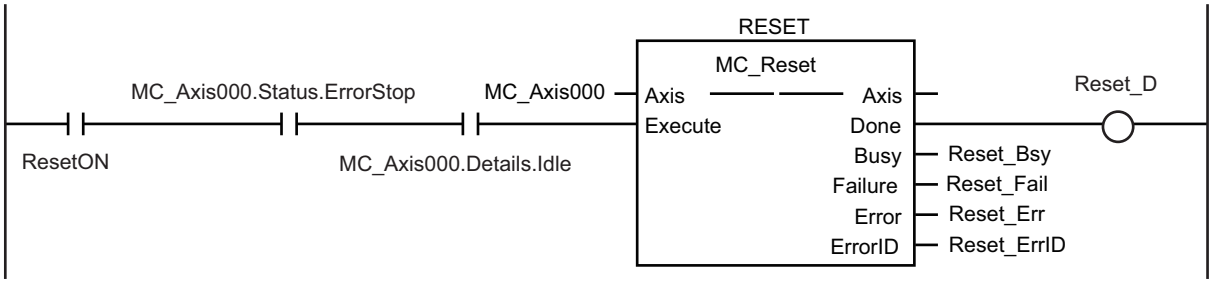
If the Servo Drive is ready, turn ON the Servo for axis 0.



If a minor fault level error occurs for axis 0, the error handler for the device (FaultHandler) is executed. Program the FaultHandler according to the device.



If *ResetON* is TRUE (i.e., when the external button is ON) and the command current velocity is zero, the error is reset.



ST Programming

```
// Check that the Servo Drive is ready when StartPg is TRUE and turn ON the Servo f
or axis 0.
// If the Servo Drive is not ready, turn OFF the Servo for axis 0.
IF (StartPg=TRUE)
  AND (MC_Axis000.DrvStatus.Ready=TRUE) THEN
    Pwr_En:=TRUE;
  ELSE
    Pwr_En:=FALSE;
END_IF;

// If a minor fault level error occurs for axis 0, the error handler for the device
 (FaultHandler) is executed.
// Program the FaultHandler according to the device.
IF MC_Axis000.MFaultLvl.Active=TRUE THEN
  FaultHandler();
END_IF;

// If ResetON is TRUE (i.e., when the external button is ON) and the command curren
t velocity is zero, the error is reset.
```

```
IF (ResetOn=TRUE)
  AND (MC_Axis000.Status.ErrorStop=TRUE)
  AND (MC_Axis000.Details.Idle=TRUE) THEN
    Reset_Ex := TRUE; // Minor fault is reset.
END_IF;

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

// MC_Reset
RESET(
  Axis := MC_Axis000,
  Execute := Reset_Ex,
  Done => Reset_D,
  Busy => Reset_Bsy,
  Failure => Reset_Fai,
  Error => Reset_Err,
  ErrorID => Reset_ErrID
);
```

10-2-4 Error Monitoring and Error Resetting for Multi-axes Coordinated Operation

You can monitor error status by monitoring the status of *Axis Minor Fault Occurrence* in the Axis Variables and *Axes Group Minor Fault Occurrence* in the Axes Group Variable.
If a minor fault level error occurs in this sample, the *Execute* input variable for the MC_GroupDisable (Disable Axes Group) instruction changes to TRUE to disable the axes group.
If the external button is ON and the command current velocity for the axes group is zero, the error is reset with the MC_GroupReset (Reset Axes Group Error) instruction.

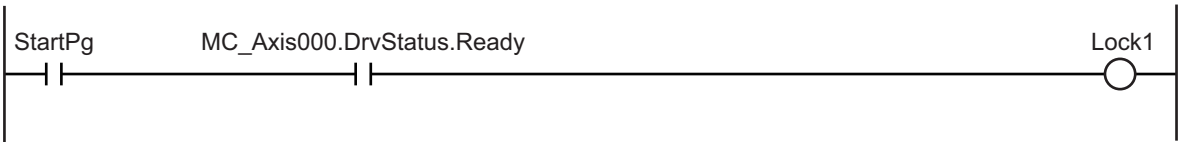
Main Variables Used in the Programming Samples

Variable 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 when there is a minor fault level error for axes group 0.
MC_Group000.Details.Idle	BOOL	FALSE	TRUE when the command interpolation velocity for axes group 0 is zero, except when waiting for in-position state.

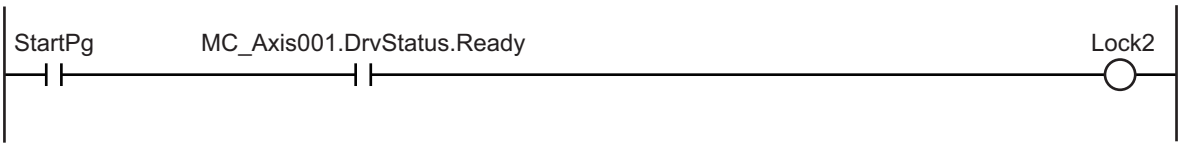
Variable name	Data type	Default	Comment
MC_Axis000	_sAXIS_REF	---	This is the Axis Variable for axis 0.
MC_Axis000.MFaultLvl.Active	BOOL	FALSE	TRUE when there is a minor fault level error for axis 0.
MC_Axis000.Details.Homed	BOOL	FALSE	TRUE when home is defined for axis 0.
MC_Axis001	_sAXIS_REF	---	This is the Axis Variable for axis 1.
MC_Axis001.MFaultLvl.Active	BOOL	FALSE	TRUE when there is a minor fault level error for axis 1.
MC_Axis001.Details.Homed	BOOL	FALSE	TRUE when home is defined for axis 1.
Pwr1_Status	BOOL	FALSE	This variable is assigned to the Status output variable from the PWR1 instance of the MC_Power instruction. It is TRUE when the Servo is ON.
Pwr2_Status	BOOL	FALSE	This variable is assigned to the Status output variable from the PWR2 instance of the MC_Power instruction. It is TRUE when the Servo is ON.
StartPg	BOOL	FALSE	When this variable is TRUE, the Servo is turned ON if EtherCAT process data communications are active and normal.
ResetON	BOOL	FALSE	This variable gives the status of the external button that is used to reset errors.

Ladder Diagram

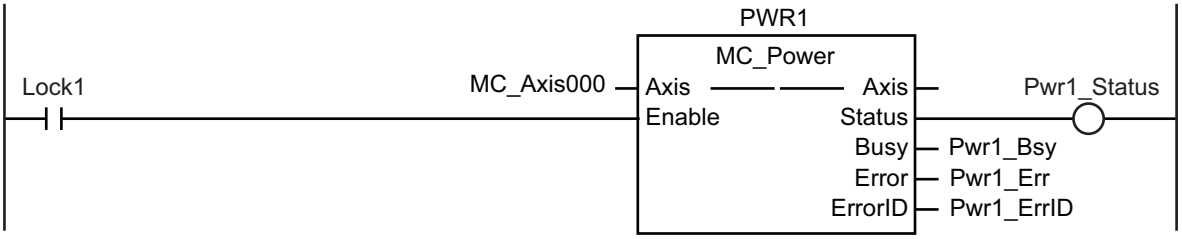
Check if the Servo Drive for axis 0 is ready when *StartPg* is TRUE.



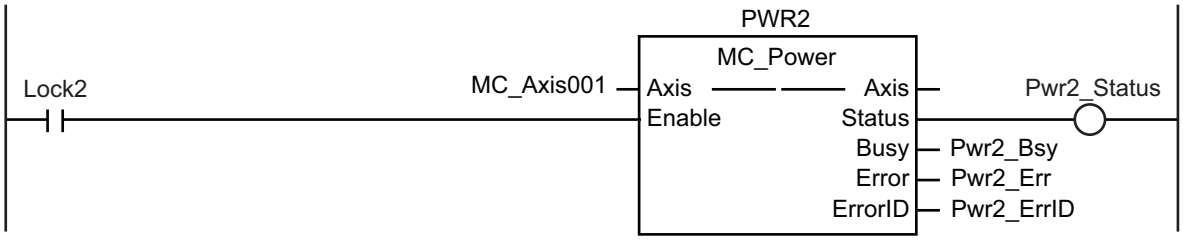
Check if the Servo Drive for axis 1 is ready when *StartPg* is TRUE.



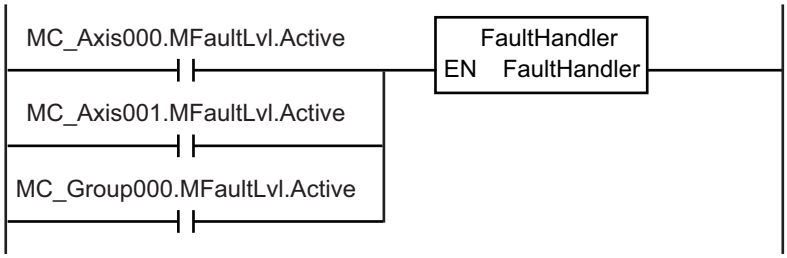
If the Servo Drive for axis 0 is ready, turn ON the Servo for axis 0.



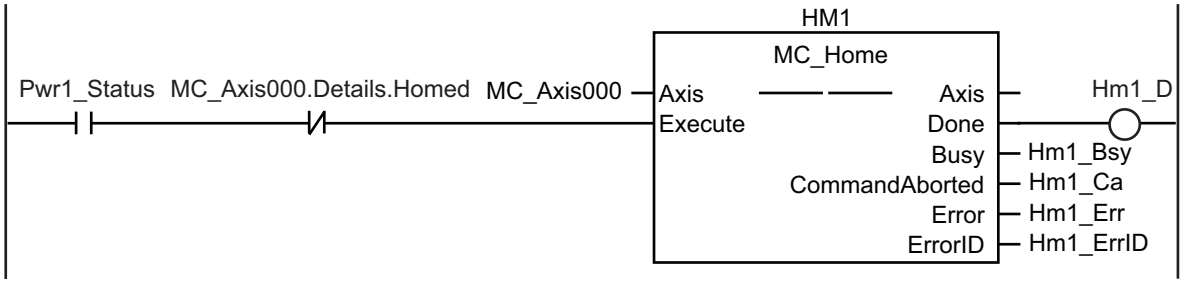
If the Servo Drive for axis 1 is ready, turn ON the Servo for axis 1.



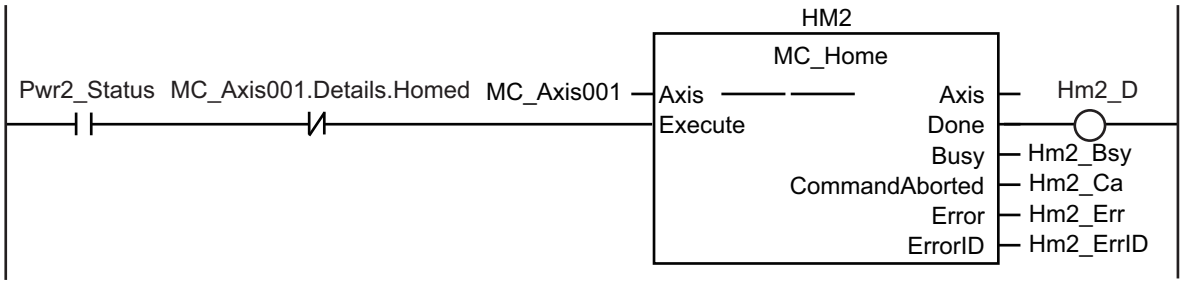
If a minor fault level error occurs for any of the composition axes in the axes group, the error handler for the device (FaultHandler) is executed. Program the FaultHandler according to the device.



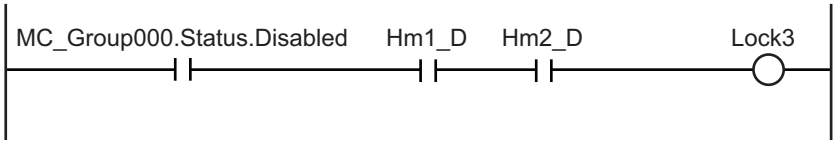
If the Servo is ON for axis 0 and home is not defined, the MC_Home instruction is executed.



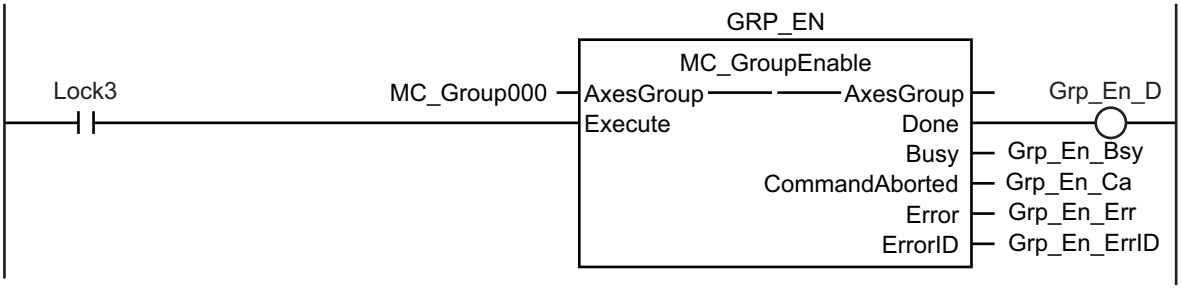
If the Servo is ON for axis 1 and home is not defined, the MC_Home instruction is executed.



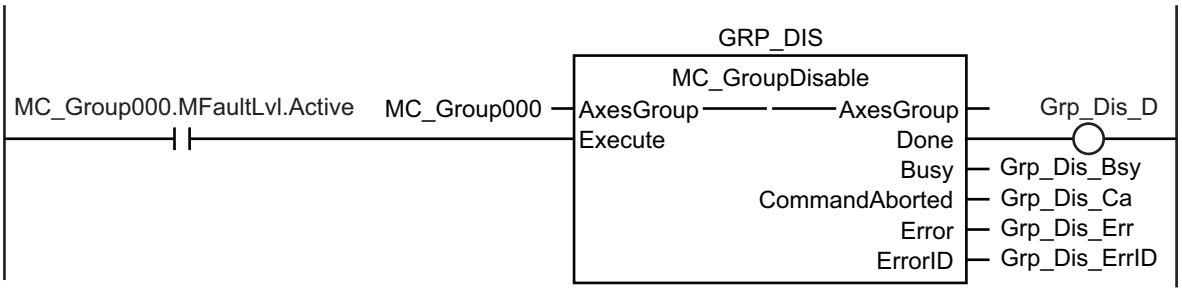
The status of the axes group and the status of home for axis 0 and axis 1 are checked.



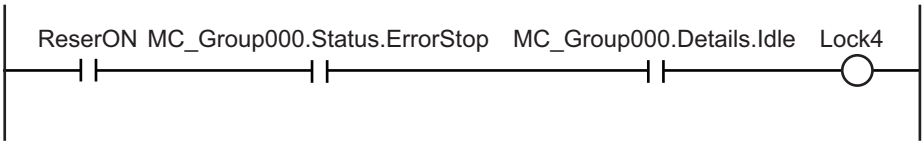
If home is defined for axis 0 and axis 1, the axes group is enabled.



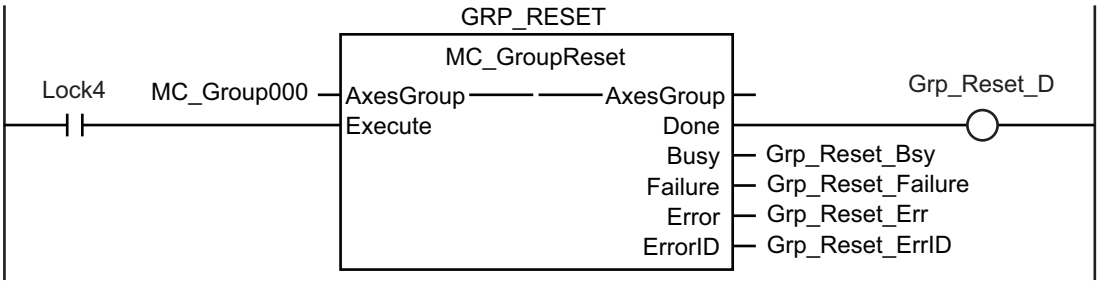
If there is a minor fault level error for the axes group, the axes group is disabled.



If the external button is ON, the status of *ResetON* and the status of axes group motion is checked.



If *ResetON* is TRUE and the axes group is stopped, the error is reset.



ST Programming

```
// Check that the Servo Drive is ready when StartPg is TRUE and turn ON the Servo f
or axis 0.
// If the Servo Drive is not ready, turn OFF the Servo for axis 0.
IF (StartPg =TRUE)
  AND (MC_Axis000.DrvStatus.Ready=TRUE) THEN
    Pwr1_En :=TRUE; // Turn ON the Servo for axis 0.
  ELSE
    Pwr1_En :=FALSE;// Turn OFF the Servo for axis 0.
END_IF;
```

```
// Check that the Servo Drive is ready when StartPg is TRUE and turn ON the Servo f
or axis 1.
// If the Servo Drive is not ready, turn OFF the Servo for axis 1.
IF (StartPg =TRUE)
  AND (MC_Axis001.DrvStatus.Ready=TRUE) THEN
    Pwr2_En :=TRUE; // Turn ON the Servo for axis 1.
  ELSE
    Pwr2_En :=FALSE; // Turn OFF the Servo for axis 1.
END_IF;

// If there is a minor fault level error for a composition axis in the axes group,
execute the error handler (FaultHandler).
IF (MC_Axis000.MFaultLvl.Active=TRUE)
  OR (MC_Axis001.MFaultLvl.Active=TRUE)
  OR (MC_Group000.MFaultLvl.Active=TRUE) THEN
    FaultHandler(); // Program the FaultHandler according to the device.
END_IF;

// If the Servo is ON for axis 0 and home is not defined, the MC_Home instruction i
s executed.
IF (Pwr1_Status=TRUE) AND (MC_Axis000.Details.Homed=FALSE) THEN
  Hm1_Ex := TRUE;
END_IF;

// If the Servo is ON for axis 1 and home is not defined, the MC_Home instruction i
s executed.
IF (Pwr2_Status=TRUE) AND (MC_Axis001.Details.Homed=FALSE) THEN
  Hm2_Ex := TRUE;
END_IF;

// If the axes group is disabled and home is defined for axis 0 and axis 1, the axe
s group is enabled.
IF (MC_Group000.Status.Disabled=TRUE) AND (Hm1_D=TRUE) AND (Hm2_D=TRUE) THEN
  Grp_En_Ex := TRUE;
END_IF;

// If there is a minor fault level error for the axes group, the axes group is disa
bled.
IF MC_Group000.MFaultLvl.Active=TRUE THEN
  Grp_Dis_Ex :=TRUE;
END_IF;

// If ResetON is TRUE (i.e., if the external button is ON) and the axes group is st
opped, the error is reset.
IF (ResetON=TRUE)
  AND (MC_Group000.Status.ErrorStop=TRUE)
  AND (MC_Group000.Details.Idle=TRUE) THEN
```

```
    Grp_Reset_Ex := TRUE;
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_GroupEnable
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
);

//MC_GroupDisable
GRP_DIS(
  AxesGroup := MC_Group000,
  Execute := Grp_Dis_Ex,
  Done => Grp_Dis_D,
  Busy => Grp_Dis_Bsy,
  CommandAborted => Grp_Dis_Ca,
  Error => Grp_Dis_Err,
  ErrorID => Grp_Dis_ErrID
);

//MC_GroupReset
GRP_RESET(
  AxesGroup := MC_Group000,
  Execute := Grp_Reset_Ex,
  Done => Grp_Reset_D,
  Busy => Grp_Reset_Bsy,
  Failure => Grp_Reset_Fai,
  Error => Grp_Reset_Err,
  ErrorID => Grp_Reset_ErrID
);
```

10-2-5 Monitoring for Instruction Errors

In this sample, further processing is not performed if there is an error when the MC_Power (Power Servo) instruction is executed.
The *UpgOn* variable specifies whether subsequent processing is allowed.

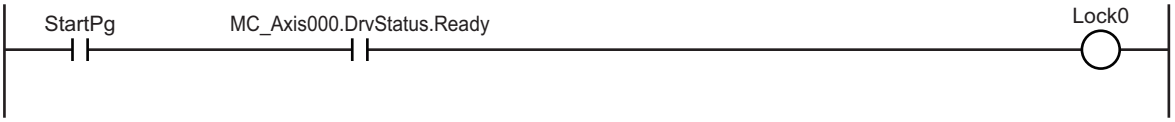
Main Variables Used in the Programming Samples

Variable name	Data type	Default	Comment
MC_Axis000	_sAXIS_REF	---	This is the Axis Variable for axis 0.
Pwr_Status	BOOL	FALSE	This variable is assigned to the Status output variable from the PWR instance of the MC_Power instruction. It is TRUE when the Servo is ON.
Pwr_Bsy	BOOL	FALSE	This variable is assigned to the Busy output variable from the PWR instance of the MC_Power instruction. It is TRUE when the Servo is ON.

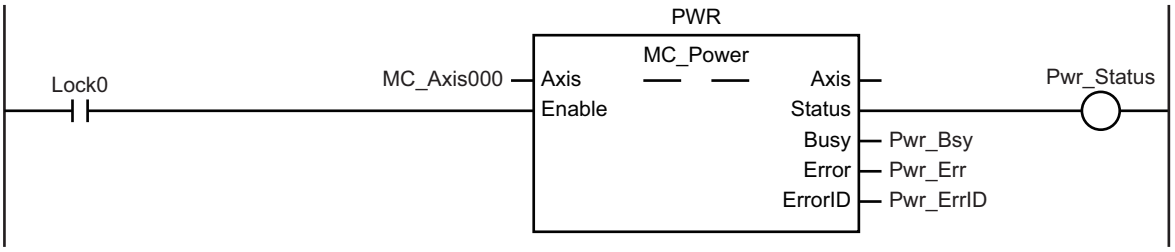
Variable name	Data type	Default	Comment
StartPg	BOOL	FALSE	When this variable is TRUE, the Servo is turned ON if EtherCAT process data communications are active and normal.
UpgOn	BOOL	FALSE	When this variable is TRUE, subsequent processing is performed.

Ladder Diagram

Check if the Servo Drive is ready when *StartPg* is TRUE.



If the Servo Drive is ready, turn ON the Servo for axis 0.



A check is made to see if any errors occurred when MC_Power was executed before execution of further processing.

ST Programming

```
// If the Servo Drive is ready when StartPg is TRUE, turn ON the Servo for axis 0.
// If the Servo Drive is not ready, turn OFF the Servo for axis 0.
IF (StartPg=TRUE)
  AND (MC_Axis000.DrvStatus.Ready=TRUE) THEN
    Pwr_En :=TRUE;
  ELSE
    Pwr_En :=FALSE;
END_IF;

IF (Pwr_Status=TRUE) OR (Pwr_Bsy=TRUE) THEN
    UpgOn := TRUE; // Further processing executed.
  ELSE
    UpgOn := FALSE;// Further processing not executed.
END_IF;

// MC_Power
PWR(
```

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

10-2-6 Checking to See If Errors Are Reset

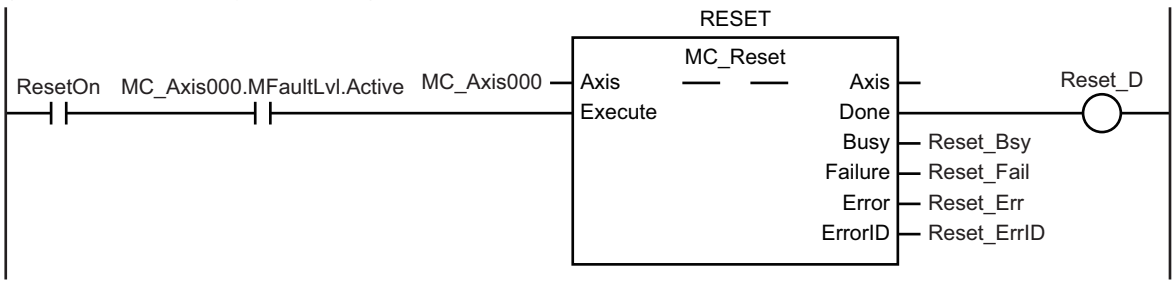
In this sample, the MC_Reset (Reset Axis Error) instruction is executed if an external button turns ON while there is a minor fault level error. Further normal processing is not executed until the *Done* output variable from the MC_Reset instruction changes to TRUE.
If the *Failure* output variable changes to TRUE, the axis decelerated to a stop or an MC common error has occurred. The cause that made the *Failure* output variable from the MC_Reset instruction turn ON is read.

Main Variables Used in the Programming Samples

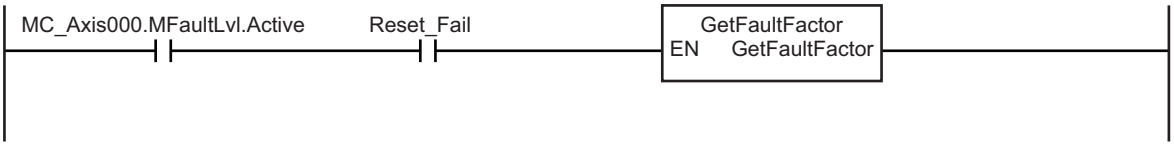
Variable name	Data type	Default	Comment
MC_Axis000	_sAXIS_REF	---	This is the Axis Variable for axis 0.
MC_Axis000.MFaultLvl.Active	BOOL	FALSE	TRUE when there is a minor fault level error for axis 0.
Pwr_Status	BOOL	FALSE	This variable is assigned to the Status output variable from the PWR instance of the MC_Power instruction. It is TRUE when the Servo is ON.
ResetON	BOOL	FALSE	This variable gives the status of the external button that is used to reset errors.
GetFaultFactor	---	---	This is the process to read the cause of the error. Program it according to the device.
RegularProcess	---	---	This is the normal processing. Program it according to the device.

Ladder Diagram

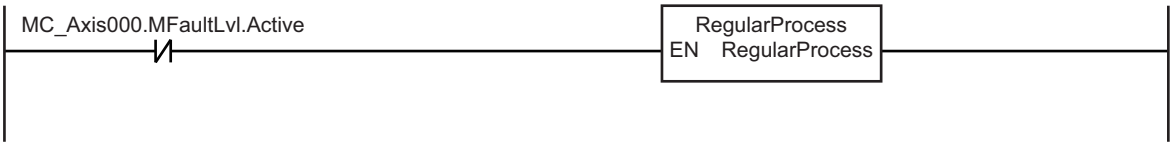
In this sample, the MC_Reset (Reset Axis Error) instruction is executed if an external button turns ON (i.e., if *ResetOn* changes to TRUE) while there is a minor fault level error.



If the *Failure* output variable from the MC_Reset instruction changes to TRUE, processing is performed to read the cause of the device error with GetFaultFactor. Program GetFaultFactor according to the device.



If a minor fault level error did not occur or was reset, normal device processing (RegularProcess) is performed. Program GetFaultFactor according to the device.



ST Programming

```
// If the external button is ON (i.e., if ResetOn changes to TRUE) while there is a
 minor fault level error, the MC_Reset (Reset Axis Error) instruction is executed.
IF (MC_Axis000.MFaultLvl.Active=TRUE) AND (ResetOn=TRUE) THEN
    Reset_Ex := TRUE; // Minor fault is reset.
  ELSE
    Reset_Ex := FALSE;
END_IF;

// If the Failure output variable from the MC_Reset instruction changes to TRUE, pr
ocessing is performed to read the cause of the error with GetFaultFactor.
// Program GetFaultFactor according to the device.
IF (MC_Axis000.MFaultLvl.Active=TRUE) AND (Reset_Fail=TRUE) THEN
  GetFaultFactor();
END_IF;

// If a minor fault level error did not occur or was reset, normal device processin
g (RegularProcess) is performed.
// Program RegularProcess according to the device.
IF MC_Axis000.MFaultLvl.Active=FALSE THEN
  RegularProcess();
END_IF;

// MC_Reset
RESET(
  Axis := MC_Axis000,
  Execute := Reset_Ex,
  Done => Reset_D,
  Busy => Reset_Bsy,
  Failure => Reset_Fail,
  Error => Reset_Err,
```

```
  ErrorID => Reset_ErrID
);
```

10-2-7 Stopping Axes during Single-axis Operation

In this sample, the MC_Stop instruction is executed to decelerate to a stop if an external button turns ON during execution of the MC_MoveAbsolute (Absolute Positioning) instruction. If there is a minor fault level error, the *CommandAborted* output variable from the MC_Stop instruction changes to TRUE.
In that case, the MC_ImmediateStop instruction is executed to stop immediately.
If for any reason the *Error* output variable from the MC_Stop instruction changes to TRUE, the MC_ImmediateStop instruction is executed to stop immediately. If the MC_ImmediateStop instruction is executed, the axis status is Error Deceleration Stopping.

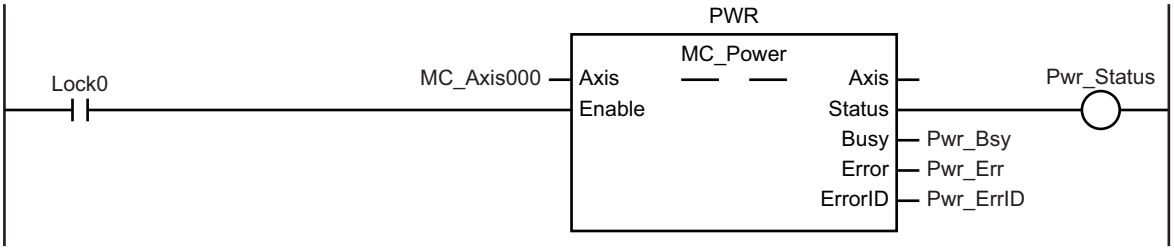
Main Variables Used in the Programming Samples

Variable name	Data type	Default	Comment
MC_Axis000	_sAXIS_REF	---	This is the Axis Variable for axis 0.
MC_Axis000.MFaultLvl.Active	BOOL	FALSE	TRUE when there is a minor fault level error for axis 0.
MC_Axis000.Details.Homed	BOOL	FALSE	TRUE when home is defined for axis 0.
Pwr_Status	BOOL	FALSE	This variable is assigned to the Status output variable from the PWR instance of the MC_Power instruction. It is TRUE when the Servo is ON.
Stp_Ca	BOOL	FALSE	This variable is assigned to the CommandAborted output variable from the STP instance of the MC_Stop instruction.
Stp_Err	BOOL	FALSE	This variable is assigned to the Error output variable from the STP instance of the MC_Stop instruction.
StartPg	BOOL	FALSE	When this variable is TRUE, the Servo is turned ON if EtherCAT process data communications are active and normal.
StopOn	BOOL	FALSE	This variable gives the status of the external button that is used to stop. The MC_Stop instruction is executed to stop the axis if this variable is TRUE.

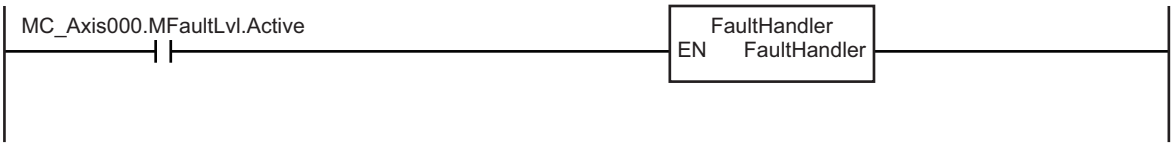
Ladder Diagram

Check if the Servo Drive for axis 0 is ready when *StartPg* is TRUE.

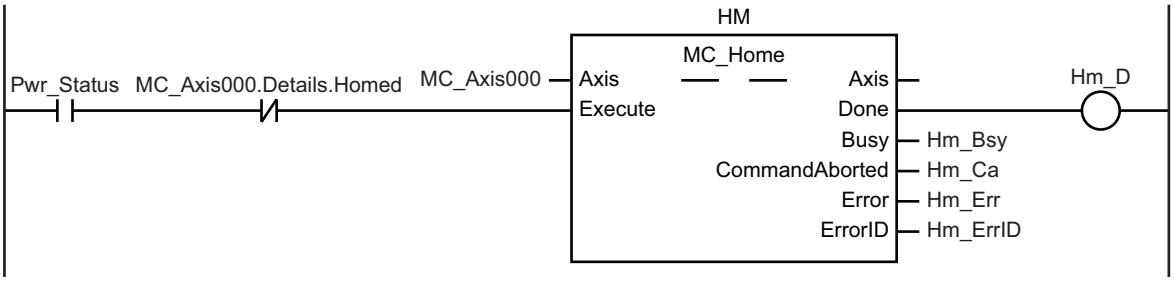
If the Servo Drive for axis 0 is ready, turn ON the Servo for axis 0.



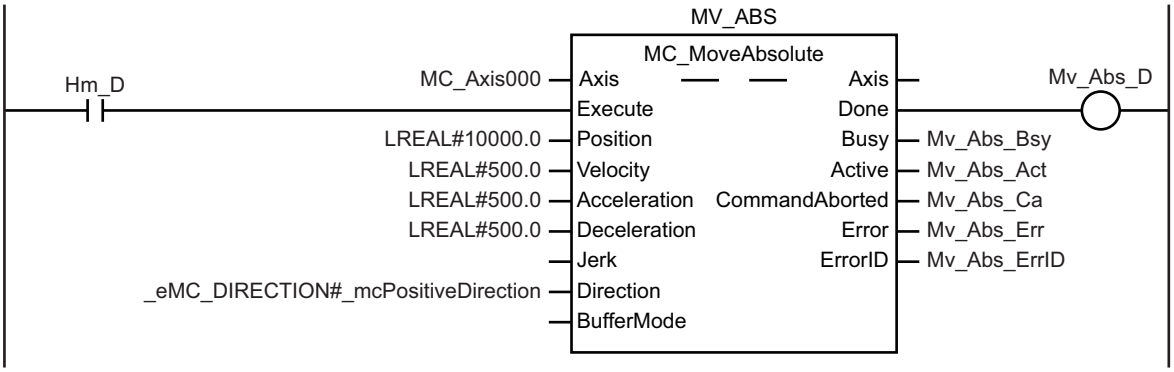
If a minor fault level error occurs for axis 0, the error handler for the device (FaultHandler) is executed.
Program the FaultHandler according to the device.



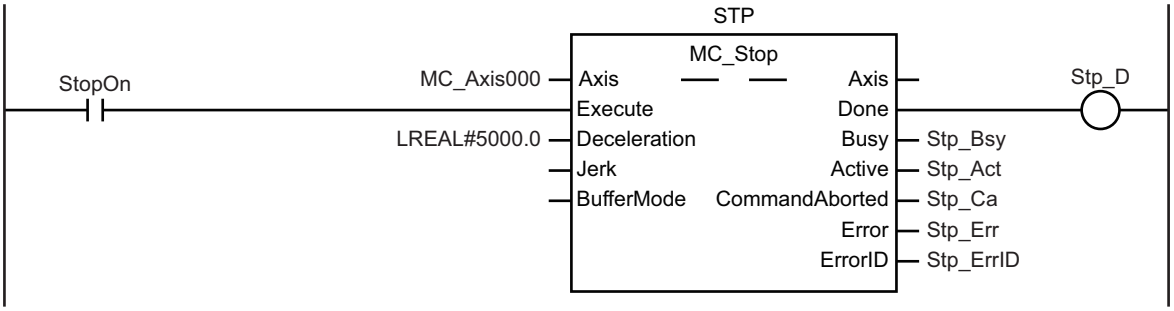
If the Servo is ON for axis 0 and home is not defined, the MC_Home instruction is executed for axis 0.



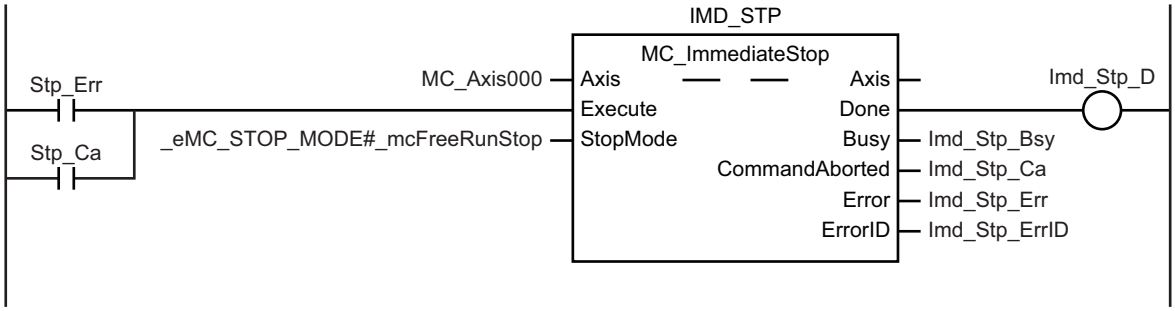
After homing is completed for axis 0, the MC_MoveAbsolute (Absolute Positioning) instruction is executed.



If *StopOn* is TRUE, the MC_Stop instruction is executed.



If the *Error* or *CommandAborted* output variable from the MC_Stop instruction changes to TRUE, the MC_ImmediateStop instruction is executed to stop immediately.



ST Programming

```
// If the input parameters for absolute positioning and stopping are not set, the t
arget values and other parameters are set.
IF InitFlag=FALSE THEN
  // The input parameters for the MC_MoveAbsolute (Absolute Positioning) instructio
n are set.
    Mv_Abs_Pos := LREAL#10000.0;
    Mv_Abs_Vel := LREAL#500.0;
    Mv_Abs_Acc := LREAL#500.0;
    Mv_Abs_Dec := LREAL#500.0;
    Mv_Abs_Dir := _eMC_DIRECTION#_mcPositiveDirection;
  // The input parameters for the MC_Stop instruction are set.
    Stp_Dec:=LREAL#5000.0;
  // The input parameters for the MC_Immediate Stop instruction are set.
    Imd_Stp_SM :=_eMC_STOP_MODE#_mcFreeRunStop;
  // The Input Parameter Initialization Completed Flag is changed to TRUE.
    InitFlag := TRUE;
END_IF;

// If the Servo Drive is ready when StartPg is TRUE, turn ON the Servo for axis 0.
IF (StartPg=TRUE)
  AND (MC_Axis000.DrvStatus.Ready=TRUE) THEN
    Pwr_En :=TRUE;
  ELSE
    Pwr_En :=FALSE;
END_IF;

// If a minor fault level error occurs for axis 0, the error handler for the device
 (FaultHandler) is executed.
// Program the FaultHandler according to the device.
IF MC_Axis000.MFaultLvl.Active=TRUE THEN
  FaultHandler();
END_IF;

// If the Servo is ON for axis 0 and home is not defined, the MC_Home instruction i
```

```
s executed.
IF (Pwr_Status=TRUE) AND (MC_Axis000.Details.Homed=FALSE) THEN
  Hm_Ex :=TRUE;
END_IF;

// If homing is completed, absolute positioning is executed.
IF Hm_D=TRUE THEN
  Mv_Abs_Ex := TRUE;
END_IF;

// If StopOn is TRUE, stopping is executed.
IF StopOn=TRUE THEN
  Stp_Ex :=TRUE;
END_IF;

// If the Error or CommandAborted output variable from the MC_Stop instruction chan
ges to TRUE, the MC_ImmediateStop instruction is executed to stop immediately.
IF (Stp_Err=TRUE) OR (Stp_Ca=TRUE) THEN
  Imd_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_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_ABS(
  Axis := MC_Axis000,
  Execute := Mv_Abs_Ex,
  Position := Mv_Abs_Pos,
```

```
  Velocity := Mv_Abs_Vel,
  Acceleration := Mv_Abs_Acc,
  Deceleration := Mv_Abs_Dec,
  Direction := Mv_Abs_Dir,
  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_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_ImmediateStop
IMD_STP(
  Axis := MC_Axis000,
  Execute := Imd_Stp_Ex,
  StopMode := Imd_Stp_SM,
  Done => Imd_Stp_D,
  Busy => Imd_Stp_Bsy,
  CommandAborted => Imd_Stp_Ca,
  Error => Imd_Stp_Err,
  ErrorID => Imd_Stp_ErrID
);
```

10-2-8 Stopping an Axes Group in Coordinated Motion

In this sample, the MC_GroupStop instruction is executed to decelerate to a stop if an external button turns ON during execution of the MC_MoveLinearAbsolute (Absolute Linear Interpolation) instruction. If there is a minor fault level error, the *CommandAborted* output variable from the MC_GroupStop instruction changes to TRUE. In that case, the MC_GroupImmediateStop instruction is executed to stop immediately.

If for any reason the *Error* output variable from the MC_GroupStop instruction changes to TRUE, the MC_GroupImmediateStop instruction is executed to stop immediately. If the MC_GroupImmediateStop instruction is executed, the axes group status is Error Deceleration Stopping.

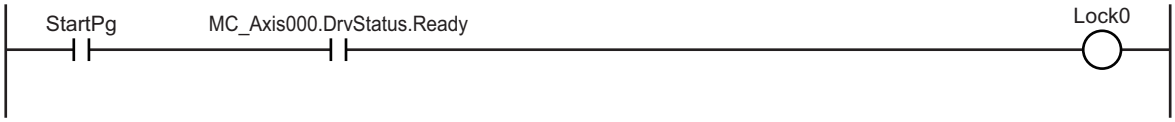
Main Variables Used in the Programming Samples

Variable name	Data type	Default	Comment
MC_Group000	_sGROUP_REF	---	This is the Axes Group Variable for axes group 0.
MC_Group000.Status.Disabled	BOOL	FALSE	TRUE when axes group 0 is disabled.
MC_Group000.MFaultLvl.Active	BOOL	FALSE	TRUE when there is a minor fault level error for axes group 0.
MC_Axis000	_sAXIS_REF	---	This is the Axis Variable for axis 0.
MC_Axis000.MFaultLvl.Active	BOOL	FALSE	TRUE when there is a minor fault level error for axis 0.
MC_Axis000.Details.Homed	BOOL	FALSE	TRUE when home is defined for axis 0.
MC_Axis001	_sAXIS_REF	---	This is the Axis Variable for axis 1.
MC_Axis001.MFaultLvl.Active	BOOL	FALSE	TRUE when there is a minor fault level error for axis 1.
MC_Axis001.Details.Homed	BOOL	FALSE	TRUE when home is defined for axis 1.
Pwr1_Status	BOOL	FALSE	This variable is assigned to the Status output variable from the PWR1 instance of the MC_Power instruction. It is TRUE when the Servo is ON.
Pwr2_Status	BOOL	FALSE	This variable is assigned to the Status output variable from the PWR2 instance of the MC_Power instruction. It is TRUE when the Servo is ON.
Grp_Stp_Ca	BOOL	FALSE	This variable is assigned to the CommandAborted output variable from the GRP_EN instance of the MC_GroupStop instruction.
Grp_Stp_Err	BOOL	FALSE	This variable is assigned to the Error output variable from the GRP_EN instance of the MC_GroupStop instruction.
StartPg	BOOL	FALSE	When this variable is TRUE, the Servo is turned ON if EtherCAT process data communications are active and normal.
StopOn	BOOL	FALSE	This variable gives the status of the external button that is used to stop. The MC_GroupStop instruction is executed to stop the axes group if this variable is TRUE.
InitFlag	BOOL	FALSE	TRUE if the input parameters are set for the MC_MoveLinearAbsolute and MC_GroupStop instructions.
Grp_En_Ex	BOOL	FALSE	This variable is used to execute the GRP_EN instance of the MC_GroupEnable instruction. It is used in ST programming.
Mv_Lin_Abs_Ex	BOOL	FALSE	This variable is used to execute the MV_LIN_ABS instance of the MC_MoveLinear instruction. It is used in ST programming.
Grp_Stp_Ex	BOOL	FALSE	This variable is used to execute the GRP_STP instance of the MC_GroupStop instruction. It is used in ST programming.

Variable name	Data type	Default	Comment
Grp_Imd_Stp_Ex	BOOL	FALSE	This variable is used to execute the GRP_IMD_STP instance of the MC_GroupImmediateStop instruction. It is used in ST programming.

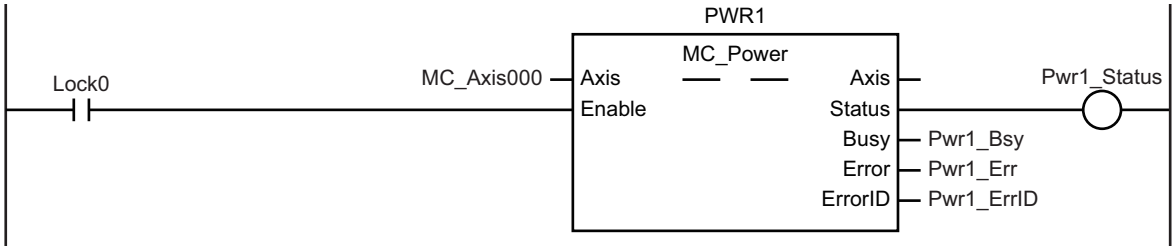
Ladder Diagram

Check if the Servo Drive for axis 0 is ready when *StartPg* is TRUE.

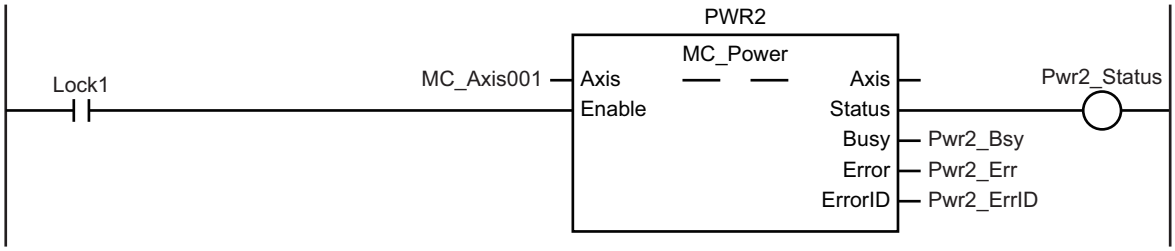


Check if the Servo Drive for axis 1 is ready when *StartPg* is TRUE.

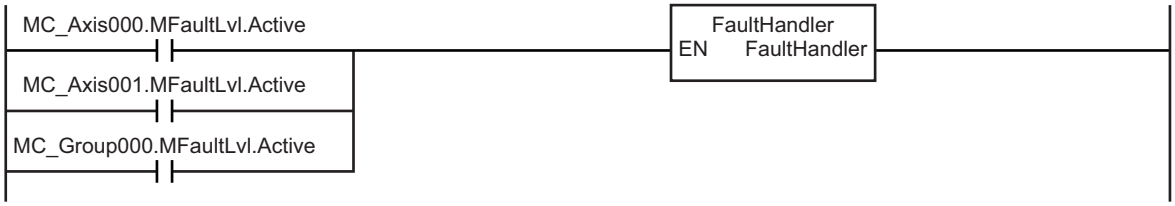
If the Servo Drive for axis 0 is ready, turn ON the Servo for axis 0.



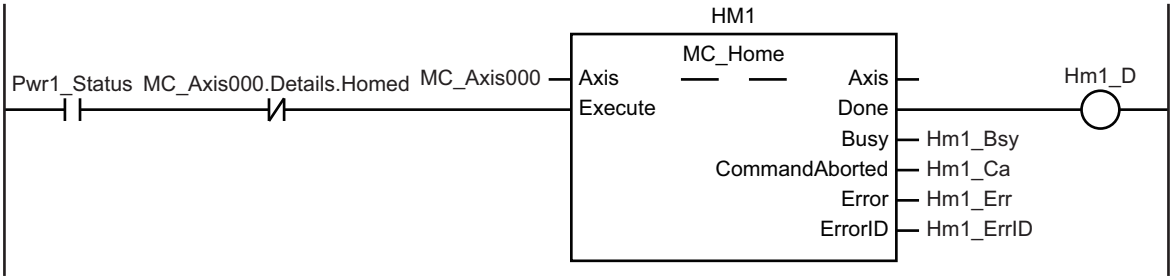
If the Servo Drive for axis 1 is ready, turn ON the Servo for axis 1.



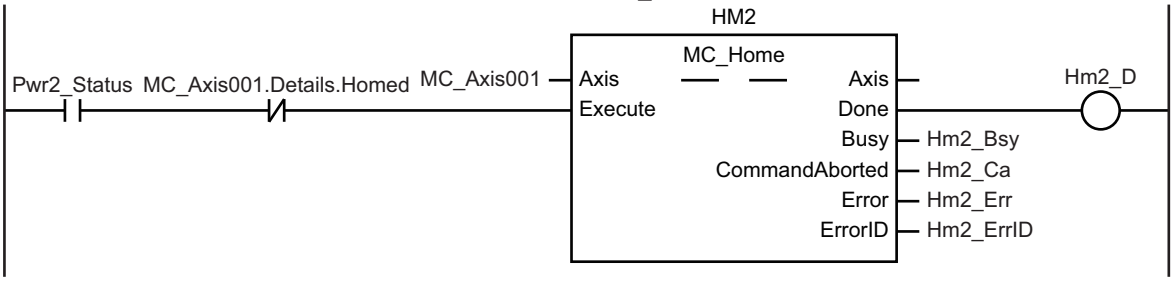
If a minor fault level error occurs for the axes group, the error handler for the device (FaultHandler) is executed. Program the FaultHandler according to the device.



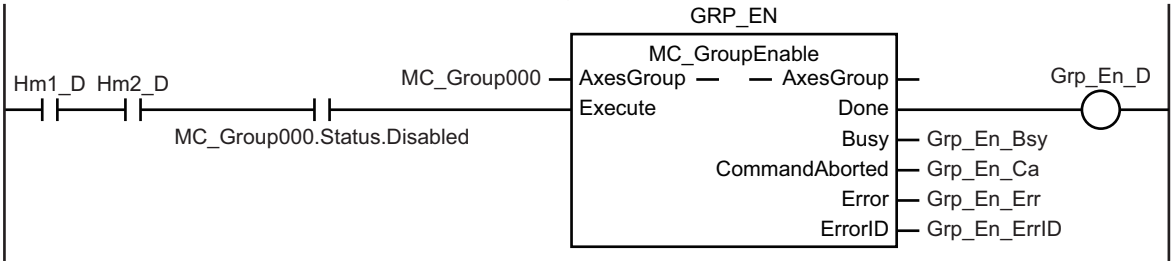
If the Servo is ON for axis 0 and home is not defined, the MC_Home instruction is executed.



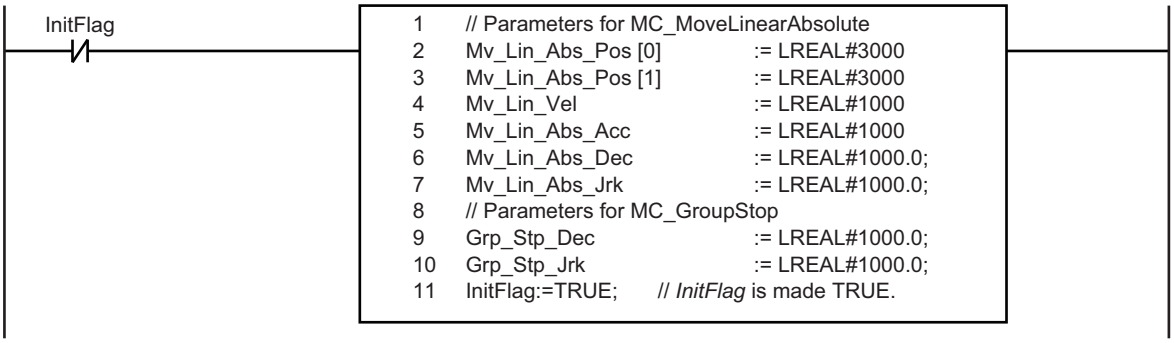
If the Servo is ON for axis 1 and home is not defined, the MC_Home instruction is executed.



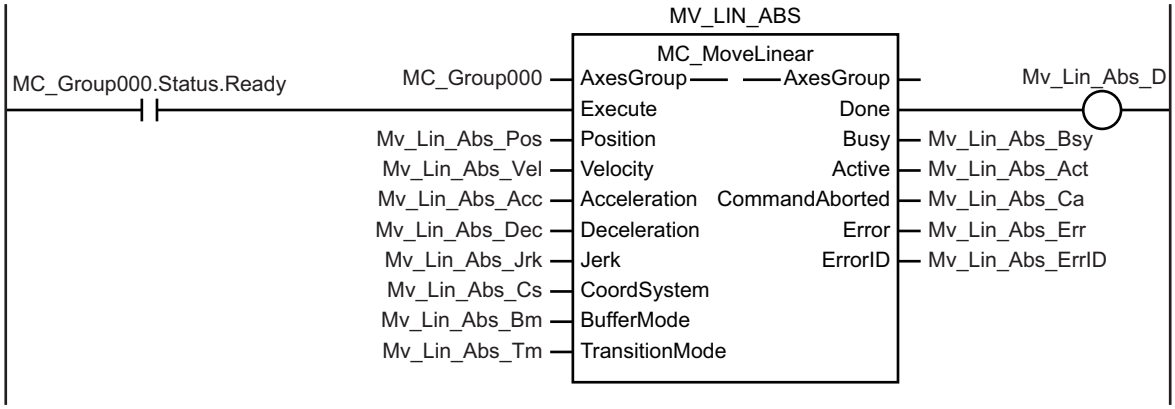
After home is defined for axis 0 and axis 1, the axes group is enabled.



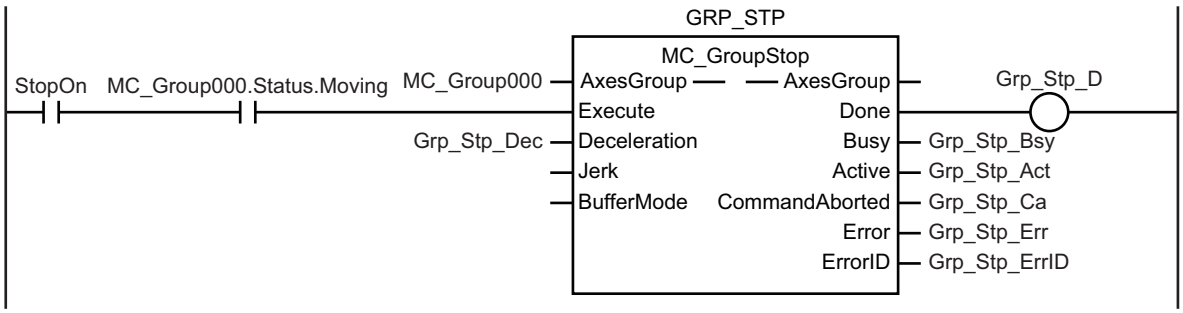
The input parameters for the MC_MoveLinearAbsolute and MC_GroupStop instructions are set.



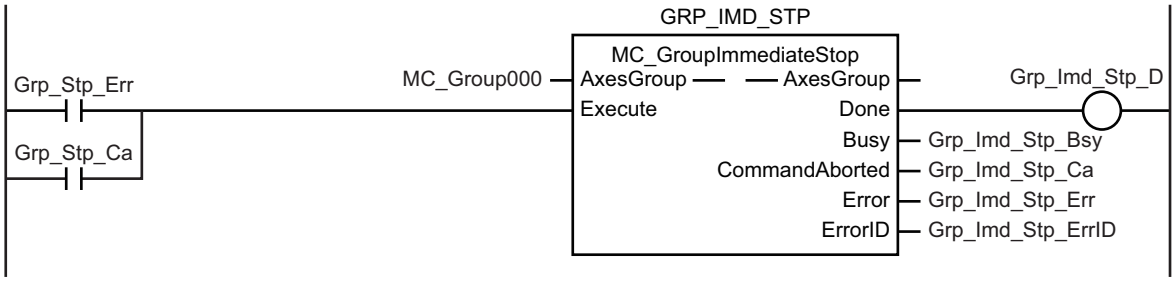
If the axes group is enabled, linear interpolation is executed.



If the external button turns ON (i.e., *StopOn* changes to TRUE) during execution of linear interpolation, the MC_GroupStop instruction is executed to decelerate the axes to a stop.



If the *Error* or *CommandAborted* output variable of the MC_GroupStop instruction is TRUE, the MC_GroupImmediateStop instruction is executed to stop immediately.



ST Programming

```
// If the input parameters for absolute linear interpolation and stopping the axes
group are not set, the target values and other parameters are set.
IF InitFlag=FALSE THEN
  // The input parameters for the MC_MoveLinearAbsolute (Absolute Linear Interpolat
ion) instruction are set.
    Mv_Lin_Abs_Pos[0] := LREAL#3000.0;
    Mv_Lin_Abs_Pos[1] := LREAL#3000.0;
    Mv_Lin_Abs_Vel := LREAL#1000.0;
    Mv_Lin_Abs_Acc := LREAL#1000.0;
    Mv_Lin_Abs_Dec := LREAL#1000.0;
    Mv_Lin_Abs_Jrk := LREAL#1000.0;
```

```
  // The input parameters for the MC_GroupStop instruction are set.
    Grp_Stp_Dec := LREAL#1000.0;
    Grp_Stp_Jrk := LREAL#1000.0;
  // The Input Parameter Initialization Completed Flag is changed to TRUE.
    InitFlag := TRUE;
END_IF;

// If the Servo Drive is ready when StartPg is TRUE, turn ON the Servo for axis 0.
// If the Servo Drive is not ready, turn OFF the Servo for axis 0.
IF (StartPg =TRUE)
  AND (MC_Axis000.DrvStatus.Ready=TRUE) THEN
    Pwr1_En :=TRUE; // Turn ON the Servo for axis 0.
  ELSE
    Pwr1_En :=FALSE;// Turn OFF the Servo for axis 0.
END_IF;

// If the Servo Drive is ready when StartPg is TRUE, turn ON the Servo for axis 1.
// If the Servo Drive is not ready, turn OFF the Servo for axis 1.
IF (StartPg =TRUE)
  AND (MC_Axis001.DrvStatus.Ready=TRUE) THEN
    Pwr2_En :=TRUE; // Turn ON the Servo for axis 1.
  ELSE
    Pwr2_En :=FALSE; // Turn OFF the Servo for axis 1.
END_IF;

// If a minor fault level error occurs, the error handler for the device (FaultHand
ler) is executed.
// Program the FaultHandler according to the device.
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 0 and home is not defined, the MC_Home instruction i
s executed for axis 0.
IF (Pwr1_Status=TRUE) AND (MC_Axis000.Details.Homed=FALSE) THEN
  Hm1_Ex :=TRUE;
END_IF;

// If the Servo is ON for axis 1 and home is not defined, the MC_Home instruction i
s executed for axis 1.
IF (Pwr2_Status=TRUE) AND (MC_Axis001.Details.Homed=FALSE) THEN
  Hm2_Ex := TRUE;
END_IF;

// If home is defined for axis 0 and axis 1 and the axes group is disabled, the axe
```

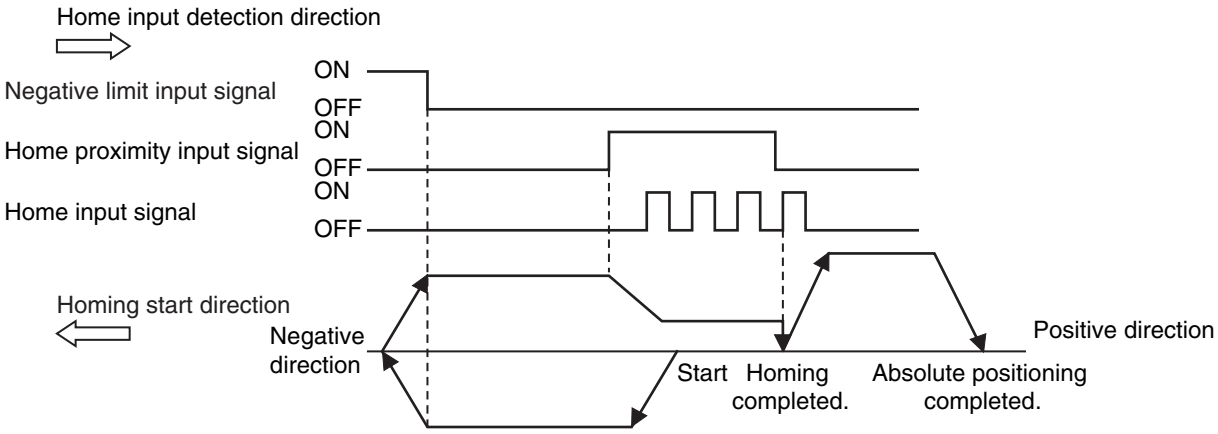
```
s group is enabled.
IF (MC_Group000.Status.Disabled=TRUE) AND (Hm1_D=TRUE) AND (Hm2_D=TRUE) THEN
  Grp_En_Ex := TRUE;
END_IF;

// If the axes group is enabled, absolute linear interpolation is executed.
IF MC_Group000.Status.Ready=TRUE THEN
  Mv_Lin_Abs_Ex :=TRUE;
END_IF;

// If the external button turns ON (i.e., StopOn changes to TRUE) during execution
of absolute linear interpolation, the axes group is stopped.
IF (MC_Group000.Status.Moving=TRUE) AND (StopOn=TRUE) THEN
  Grp_Stp_Ex := TRUE;
END_IF;

// If the CommandAborted or Error output variable from the Group Stop instruction a
re TRUE, the axes group is stopped immediately.
IF (Grp_Stp_Ca=TRUE) OR (Grp_Stp_Err=TRUE) THEN
  Grp_Imd_Stp_Ex :=TRUE;
END_IF;

//MC_Power
PWR1(
  Axis := MC_Axis000,
  Enable := Pwr1_En,
  Status => Pwr1_Status,
  Busy => Pwr1_Bsy,
  Error => Pwr1_Err,
  ErrorID => Pwr1_ErrID
);

PWR2(
  Axis := MC_Axis001,
  Enable := Pwr2_En,
  Status => Pwr2_Status,
  Busy => Pwr2_Bsy,
  Error => Pwr2_Err,
  ErrorID => Pwr2_ErrID
);

//MC_Home
HM1(
  Axis := MC_Axis000,
  Execute := Hm1_Ex,
  Done => Hm1_D,
  Busy => Hm1_Bsy,
```

```
  CommandAborted => Hm1_Ca,
  Error => Hm1_Err,
  ErrorID => Hm1_ErrID
);

HM2(
  Axis := MC_Axis001,
  Execute := Hm2_Ex,
  Done => Hm2_D,
  Busy => Hm2_Bsy,
  CommandAborted => Hm2_Ca,
  Error => Hm2_Err,
  ErrorID => Hm2_ErrID
);

//MC_GroupEnable
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
);

//MC_MoveLinearAbsolute
MV_LIN_ABS(
  AxesGroup := MC_Group000,
  Execute := Mv_Lin_Abs_Ex,
  Position := Mv_Lin_Abs_Pos,
  Velocity := Mv_Lin_Abs_Vel,
  Acceleration := Mv_Lin_Abs_Acc,
  Deceleration := Mv_Lin_Abs_Dec,
  Jerk := Mv_Lin_Abs_Jrk,
  Done => Mv_Lin_Abs_D,
  Busy => Mv_Lin_Abs_Bsy,
  Active => Mv_Lin_Abs_Act,
  CommandAborted => Mv_Lin_Abs_Ca,
  Error => Mv_Lin_Abs_Err,
  ErrorID => Mv_Lin_Abs_ErrID
);

//MC_GroupStop
GRP_STP(
  AxesGroup := MC_Group000,
  Execute := Grp_Stp_Ex,
```

```
  Deceleration := Grp_Stp_Dec,
  Done => Grp_Stp_D,
  Busy => Grp_Stp_Bsy,
  Active => Grp_Stp_Act,
  CommandAborted => Grp_Stp_Ca,
  Error => Grp_Stp_Err,
  ErrorID => Grp_Stp_ErrID
);

//MC_GroupImmediateStop
GRP_IMD_STP(
  AxesGroup := MC_Group000,
  Execute := Grp_Imd_Stp_Ex,
  Done => Grp_Imd_Stp_D,
  Busy => Grp_Imd_Stp_Bsy,
  CommandAborted => Grp_Imd_Stp_Ca,
  Error => Grp_Imd_Stp_Err,
  ErrorID => Grp_Imd_Stp_ErrID
);
```

10-2-9 Homing and Absolute Positioning

In this sample, the starting point for homing is assumed to be where the home proximity input is ON. The Homing Operation Mode is set to **4 (Home proximity input OFF)**. After homing is completed to define home, absolute positioning is executed.



Axis Parameter Settings That Are Related to Homing

Parameter name	Setting	Description
Homing Method	4: Home proximity input OFF	Home is defined where the home proximity input turns OFF.
Operation Selection at Positive Limit Input	1: Reverse turn/immediate stop	The positive limit input is not used, so the default setting is used for this parameter.
Operation Selection at Negative Limit Input	2: Reverse turn/deceleration stop	The axis decelerates to a stop and reverses direction when the negative limit input is detected.

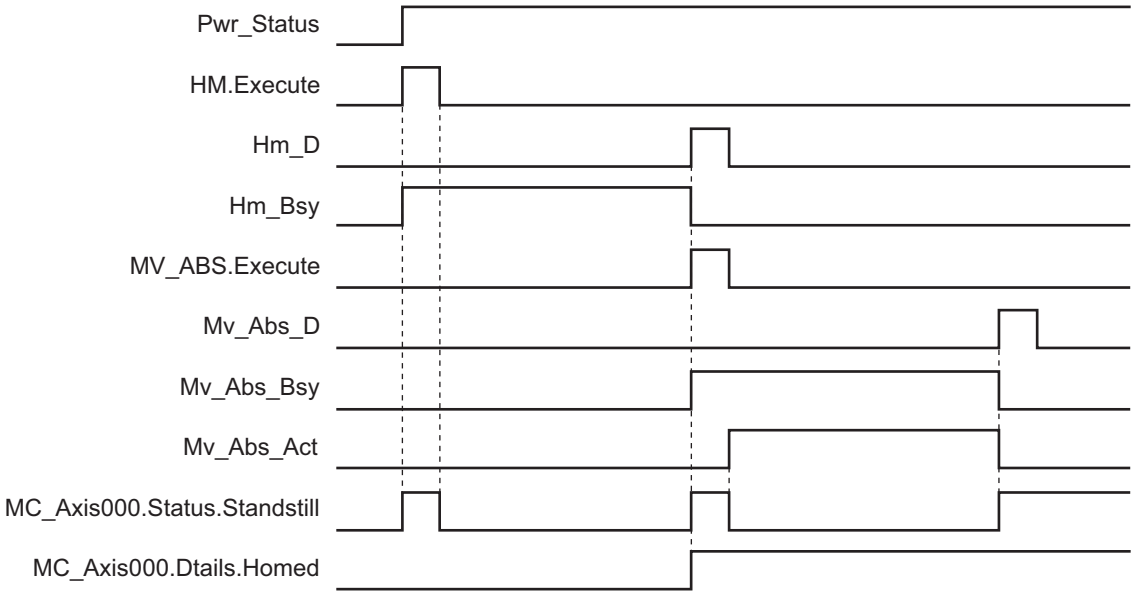
Parameter name	Setting	Description
Homing Start Direction	2: Negative direction	When homing is performed, the axis starts moving in the negative direction.
Home Input Detection Direction	1: Positive direction	Home is detected while the axis moves in the positive direction.

Main Variables Used in the Programming Samples

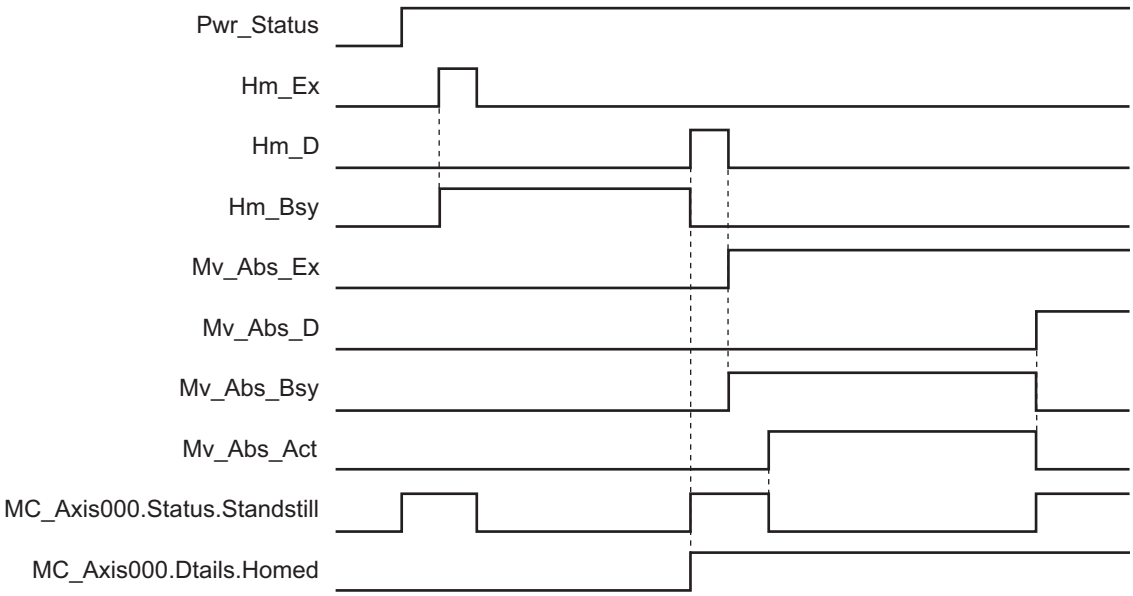
Variable name	Data type	Default	Comment
MC_Axis000	_sAXIS_REF	---	This is the Axis Variable for axis 0.
MC_Axis000.Status.StandStill	BOOL	FALSE	TRUE while the Servo is OFF for axis 0.
MC_Axis000.MFaultLvl.Active	BOOL	FALSE	TRUE when there is a minor fault level error for axis 0.
MC_Axis000.Details.Homed	BOOL	FALSE	TRUE when home is defined for axis 0.
Pwr_Status	BOOL	FALSE	This variable is assigned to the Status output variable from the PWR instance of the MC_Power instruction. It is TRUE when the Servo is ON.
StartPg	BOOL	FALSE	When this variable is TRUE, the Servo is turned ON if EtherCAT process data communications are active and normal.
Hm_Ex	BOOL	FALSE	This variable is used to execute the MC_Home instruction. It is used in ST programming.
Mv_Abs_Ex	BOOL	FALSE	This variable is used to execute the MC_MoveAbsolute (Absolute Positioning) instruction. It is used in ST programming.

Timing Chart

● Ladder Diagram

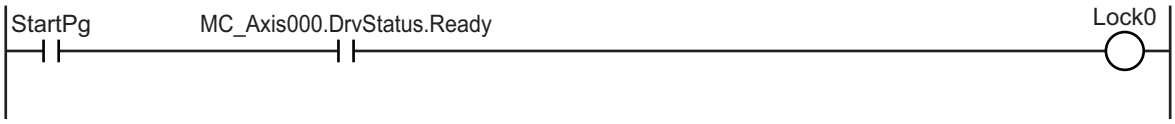


● ST Programming

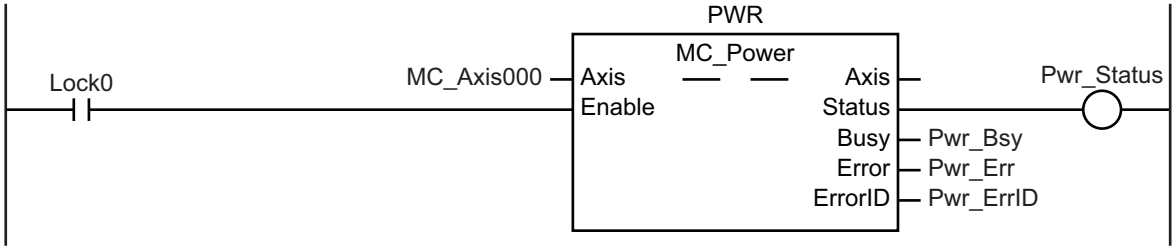


Ladder Diagram

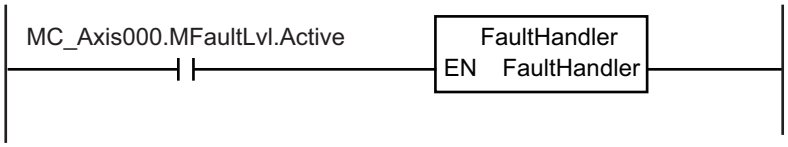
Check if the Servo Drive for axis 0 is ready when *StartPg* is TRUE.



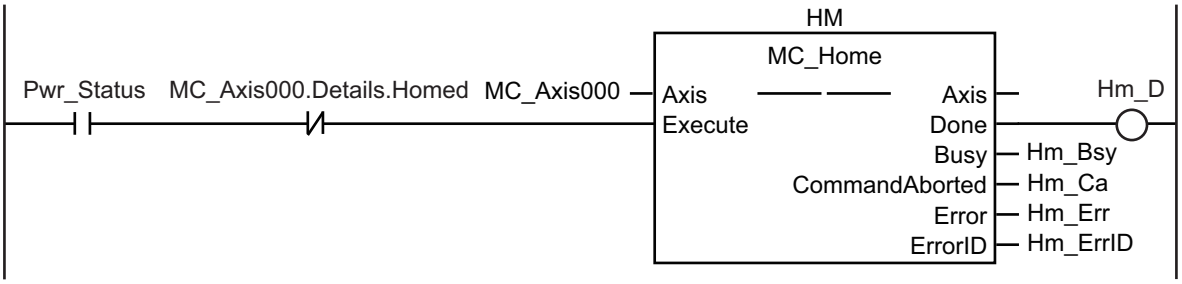
If the Servo Drive for axis 0 is ready, turn ON the Servo for axis 0.



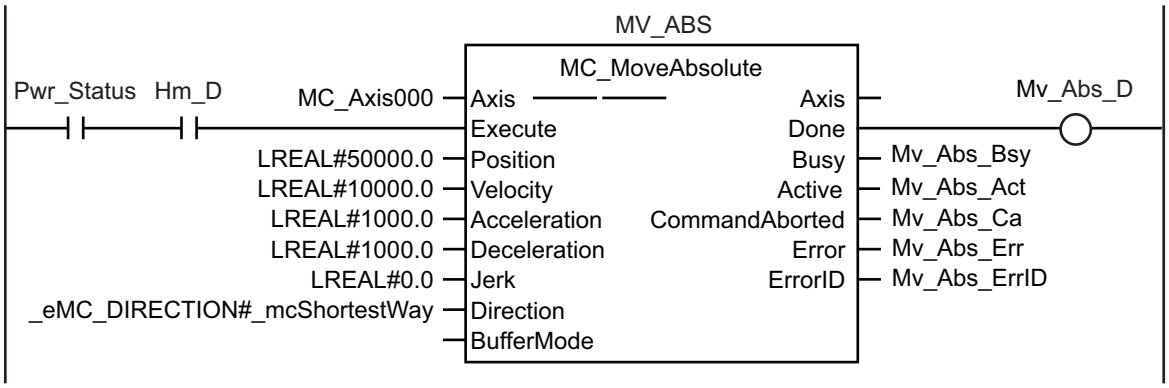
If a minor fault level error occurs for axis 0, the error handler for the device (FaultHandler) is executed. Program the FaultHandler according to the device.



If the Servo is ON for axis 0 and home is not defined, the MC_Home instruction is executed.



If the Servo is ON and home is defined, absolute positioning is executed.



ST Programming

```
// If the input parameters for absolute positioning are not set, the target values
and other parameters are set.
IF InitFlag=FALSE THEN
  // The input parameters for the MC_MoveAbsolute (Absolute Positioning) instructio
n are set.
    Mv_Abs_Pos := LREAL#50000.0;
    Mv_Abs_Vel := LREAL#10000.0;
    Mv_Abs_Acc := LREAL#1000.0;
    Mv_Abs_Dec := LREAL#1000.0;
    Mv_Abs_Dir := _eMC_DIRECTION#_mcShortestWay;
  // The Input Parameter Initialization Completed Flag is changed to TRUE.
    InitFlag := TRUE;
END_IF;

// If the Servo Drive is ready when StartPg is TRUE, turn ON the Servo for axis 0.
IF (StartPg=TRUE)
  AND (MC_Axis000.DrvStatus.Ready=TRUE) THEN
    Pwr_En:=TRUE;
  ELSE
    Pwr_En:=FALSE;
END_IF;

// If a minor fault level error occurs for axis 0, the error handler for the device
```

```
 (FaultHandler) is executed.
// Program the FaultHandler according to the device.
IF MC_Axis000.MFaultLvl.Active=TRUE THEN
  FaultHandler();
END_IF;

// If the Servo is ON for axis 0 and home is not defined, the MC_Home instruction i
s executed.
IF (Pwr_Status=TRUE) AND (MC_Axis000.Details.Homed=FALSE) THEN
    Hm_Ex :=TRUE;
  ELSE
    Hm_Ex :=FALSE;
END_IF;

// If the Servo is ON and home is defined, absolute positioning is executed.
IF (Pwr_Status=TRUE) AND (Hm_D=TRUE) THEN
  Mv_Abs_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_ABS(
Axis := MC_Axis000,
Execute := Mv_Abs_Ex,
Position := Mv_Abs_Pos,
Velocity := Mv_Abs_Vel,
Acceleration := Mv_Abs_Acc,
```

```
Deceleration := Mv_Abs_Dec,
Direction := Mv_Abs_Dir,
Done => Mv_Abs_D,
Busy => Mv_Abs_Bsy,
Active => Mv_Abs_Act,
CommandAborted => Mv_Abs_Ca,
Error => Mv_Abs_Err,
ErrorID => Mv_Abs_ErrID
);
```

10-2-10 Changing the Target Position by Re-execution of an Instruction

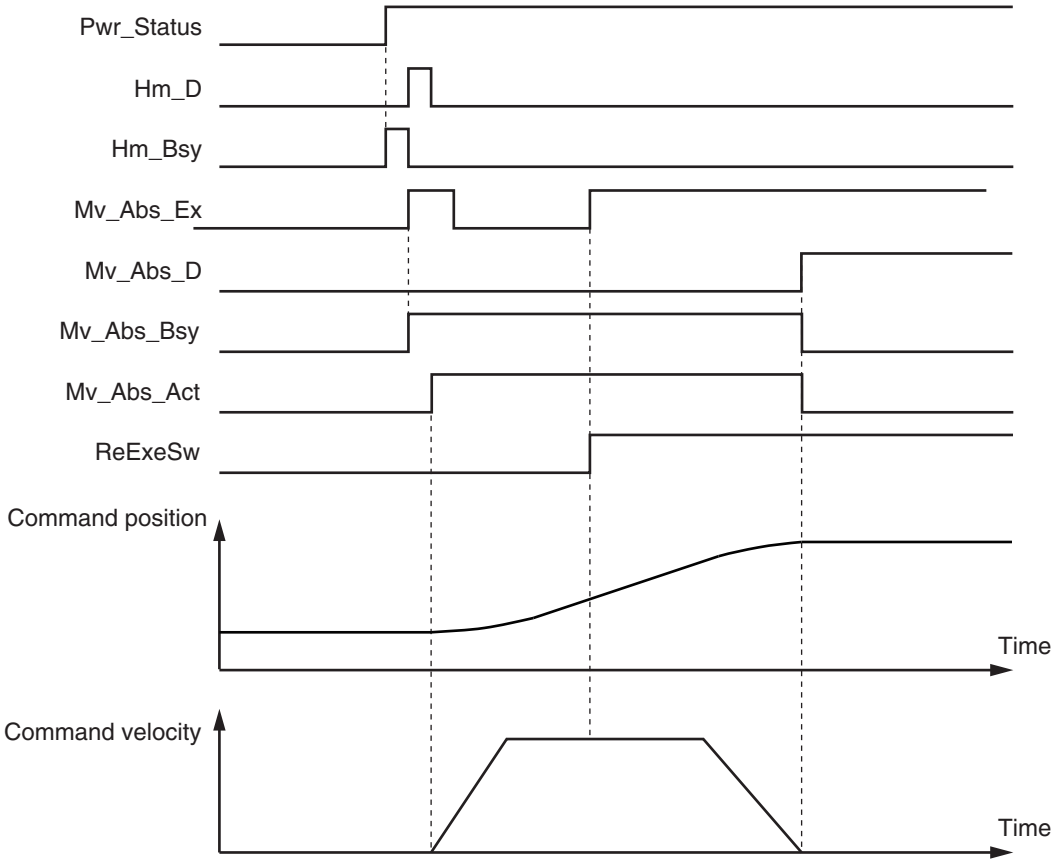
This sample starts absolute positioning to a target position of 1000 and then uses the same instance of the absolute positioning instruction to change the target position to 2000.

Main Variables Used in the Programming Samples

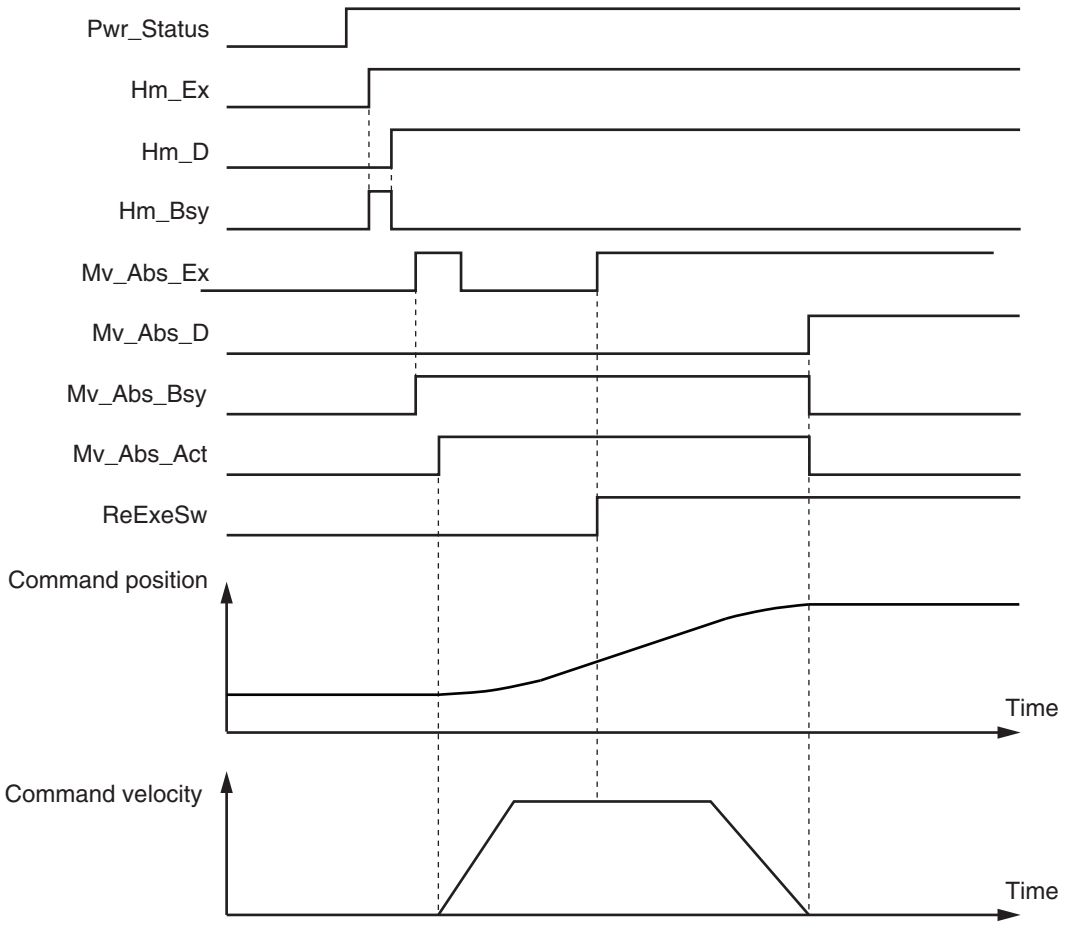
Variable name	Data type	Default	Comment
MC_Axis000	_sAXIS_REF	---	This is the Axis Variable for axis 0.
MC_Axis000.MFaultLvl.Active	BOOL	FALSE	TRUE when there is a minor fault level error for axis 0.
MC_Axis000.Details.Homed	BOOL	FALSE	TRUE when home is defined for axis 0.
Pwr_Status	BOOL	FALSE	This variable is assigned to the Status output variable from the PWR instance of the MC_Power instruction. It is TRUE when the Servo is ON.
StartPg	BOOL	FALSE	When this variable is TRUE, the Servo is turned ON if EtherCAT process data communications are active and normal.
ReExeSw	BOOL	FALSE	This variable is used to re-execute the instruction.
Hm_Ex	BOOL	FALSE	This variable is used to execute the MC_Home instruction. It is used in ST programming.
Mv_Abs_Ex	BOOL	FALSE	This variable is used to execute the MC_MoveAbsolute (Absolute Positioning) instruction. It is used in ST programming.

Timing Chart

● Ladder Diagram

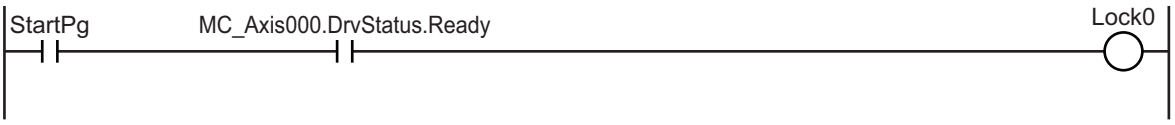


● ST Programming

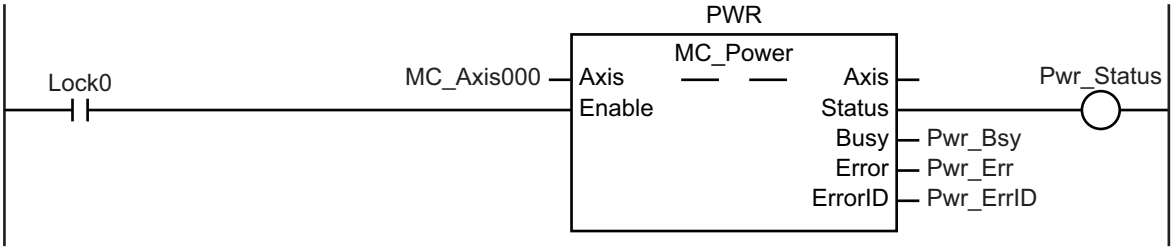


Ladder Diagram

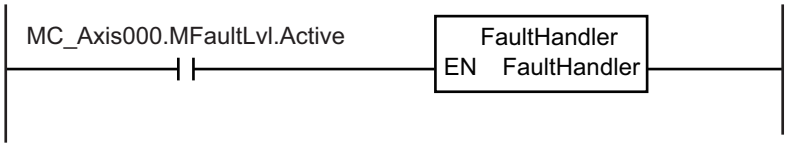
Check if the Servo Drive for axis 0 is ready when *StartPg* is TRUE.



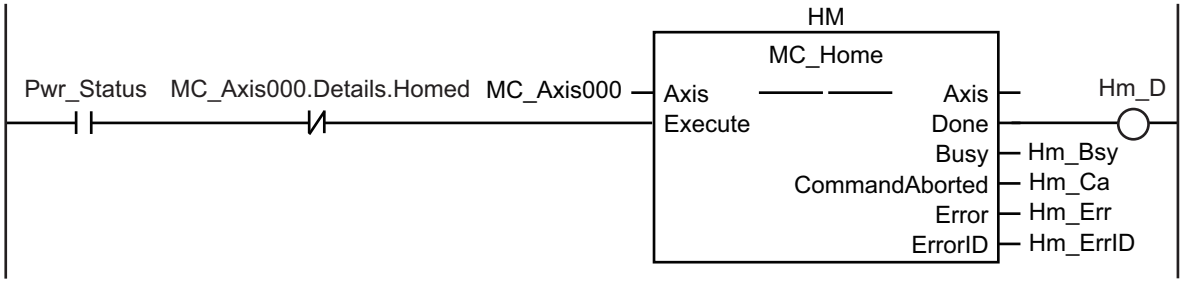
If the Servo Drive for axis 0 is ready, turn ON the Servo for axis 0.



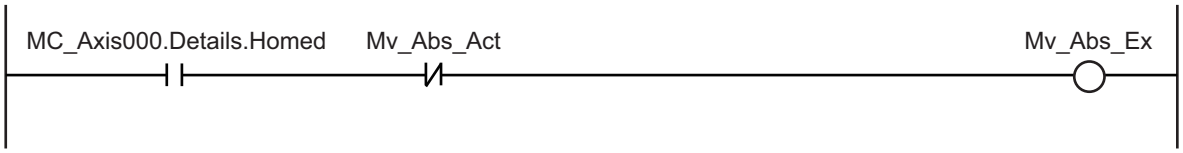
If a minor fault level error occurs for axis 0, the error handler for the device (FaultHandler) is executed.
Program the FaultHandler according to the device.



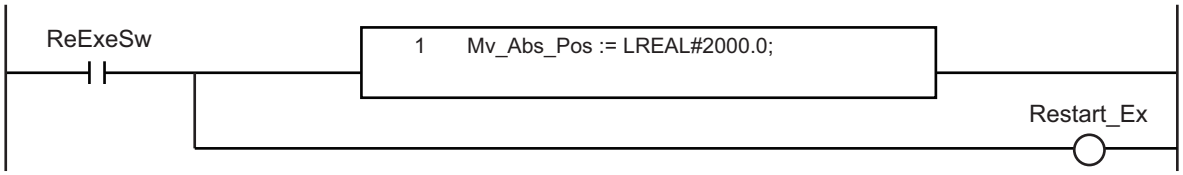
If the Servo is ON for axis 0 and home is not defined, the MC_Home instruction is executed.



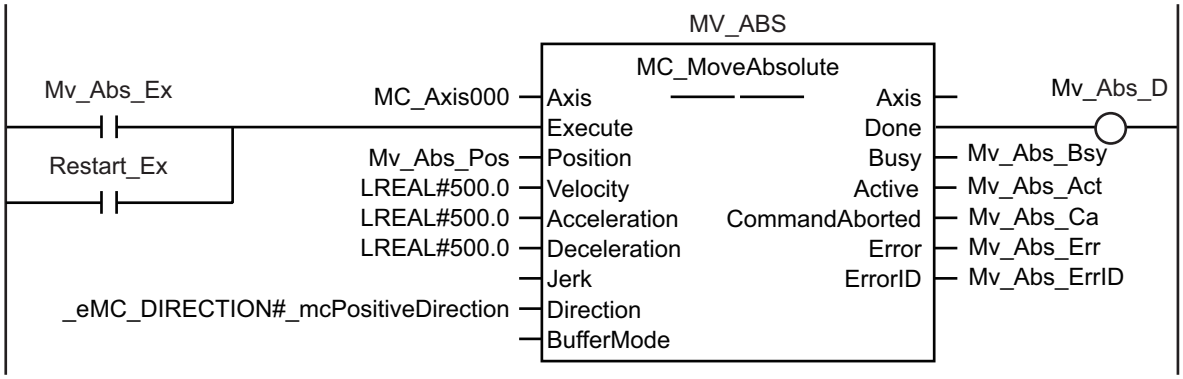
After home is defined for axis 0, absolute positioning is executed if it is not already in progress.



When *ReExeSw* changes to TRUE, the absolute positioning instruction is re-executed to change the target position to 2000.



Absolute positioning is executed according to the status of Mv_Abs_Ex.



ST Programming

```
// If the input parameters for absolute positioning are not set, the target values
and other parameters are set.
IF InitFlag = FALSE THEN
  // Parameters for MC_MoveAbsolute
```

```
    Mv_Abs_Pos := LREAL#1000.0;
    Mv_Abs_Vel := LREAL#500.0;
    Mv_Abs_Acc := LREAL#500.0;
    Mv_Abs_Dec := LREAL#500.0;
    Mv_Abs_Dir := _eMC_DIRECTION#_mcPositiveDirection;
  // The Input Parameter Initialization Completed Flag is changed to TRUE.
    InitFlag := TRUE;
END_IF;

// If the Servo Drive is ready when StartPg is TRUE, turn ON the Servo for axis 0.
IF (StartPg=TRUE)
  AND (MC_Axis000.DrvStatus.Ready=TRUE) THEN
    Pwr_En:=TRUE;
  ELSE
    Pwr_En:=FALSE;
END_IF;

// If a minor fault level error occurs for axis 0, the error handler for the device
 (FaultHandler) is executed.
// Program the FaultHandler according to the device.
IF MC_Axis000.MFaultLvl.Active=TRUE THEN
  FaultHandler();
END_IF;

// If the Servo is ON for axis 0 and home is not defined, the MC_Home instruction i
s executed.
IF (Pwr_Status=TRUE) AND (MC_Axis000.Details.Homed=FALSE) THEN
  Hm_Ex :=TRUE;
END_IF;

// After homing is completed for axis 0, absolute positioning is executed if it is
not already in progress.
IF (Hm_D=TRUE) AND (Mv_Abs_Act=FALSE) THEN
    Mv_Abs_Ex := TRUE;
  ELSE
    Mv_Abs_Ex := FALSE;
END_IF;

// When ReExeSw changes to TRUE, the absolute positioning instruction is re-execute
d to change the target position to 2000.
IF ReExeSw=TRUE THEN
  Mv_Abs_Pos :=LREAL#2000.0;
  Mv_Abs_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_ABS(
  Axis := MC_Axis000,
  Execute := Mv_Abs_Ex,
  Position := Mv_Abs_Pos,
  Velocity := Mv_Abs_Vel,
  Acceleration := Mv_Abs_Acc,
  Deceleration := Mv_Abs_Dec,
  Direction := Mv_Abs_Dir,
  Done => Mv_Abs_D,
  Busy => Mv_Abs_Bsy,
  Active => Mv_Abs_Act,
  CommandAborted => Mv_Abs_Ca,
  Error => Mv_Abs_Err,
  ErrorID => Mv_Abs_ErrID
);
```

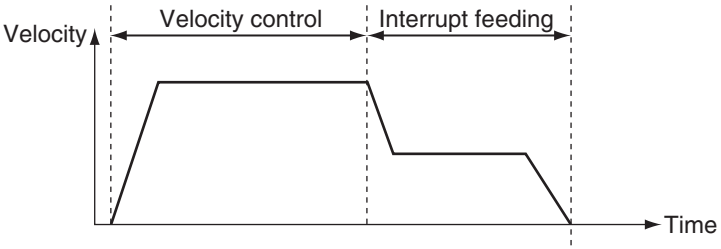
10-2-11 Interrupt Feeding

This sample performs interrupt feeding when an interrupt occurs during velocity control.
One of the following is specified for the *Direction* variable when velocity control is performed in **Rotary Mode**.

- _mcPositiveDirection
- _mcNegativeDirection
- _mcCurrentDirection

This sample uses _mcCurrentDirection.

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



Axis Parameter Settings

Parameter name	Setting	Description
Count Mode	Rotary Mode	Rotary Mode is set as the count mode for the position.
Modulo Maximum Position Setting Value	360	The Modulo Maximum Position is set to *360*.
Modulo Minimum Position Setting Value	0	The Modulo Minimum Position is set to *0*.
Homing Method	Zero position preset	A zero position preset is performed to define home.

Main Variables Used in the Programming Samples

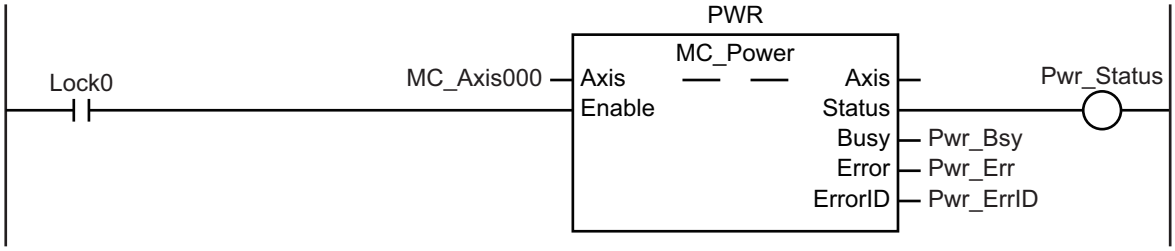
Variable name	Data type	Default	Comment
MC_Axis000	_sAXIS_REF	---	This is the Axis Variable for axis 0.
MC_Axis000.MFaultLvl.Active	BOOL	FALSE	TRUE when there is a minor fault level error for axis 0.
MC_Axis000.Details.Homed	BOOL	FALSE	TRUE when home is defined for axis 0.
Pwr_Status	BOOL	FALSE	This variable is assigned to the Status output variable from the PWR instance of the MC_Power instruction. It is TRUE when the Servo is ON.
StartPg	BOOL	FALSE	When this variable is TRUE, the Servo is turned ON if EtherCAT process data communications are active and normal.
TrigRef	_sTRIGGER_REF	---	This parameter specifies the trigger input condition to use for the interrupt input. Latch 1 of the Servo Drive is used in this sample.
Hm_Ex	BOOL	FALSE	This variable is used to execute the MC_Home instruction. It is used in ST programming.

Variable name	Data type	Default	Comment
Mv_Feed_Ex	BOOL	FALSE	This variable is used to execute the MC_MoveFeed (Interrupt Feeding) instruction. It is used in ST programming.
InitFlag	BOOL	FALSE	TRUE if the input parameters are set for the MC_MoveFeed instruction.

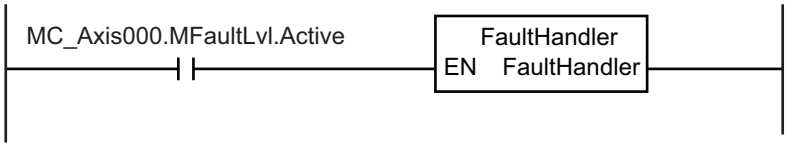
Ladder Diagram

Check if the Servo Drive for axis 0 is ready when *StartPg* is TRUE.

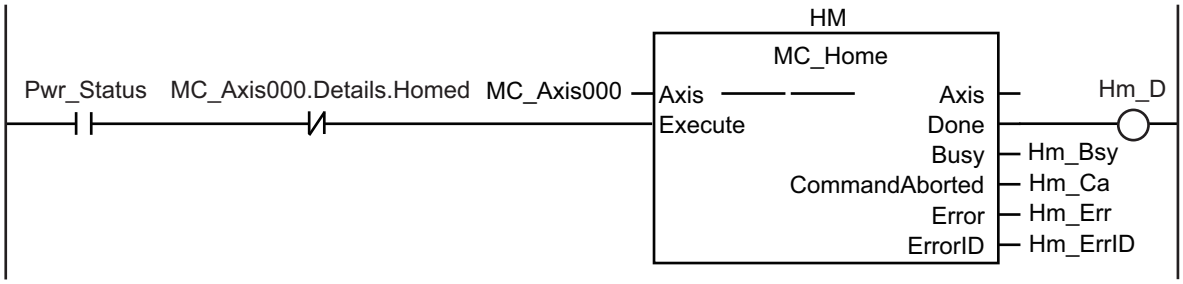
If the Servo Drive for axis 0 is ready, turn ON the Servo for axis 0.



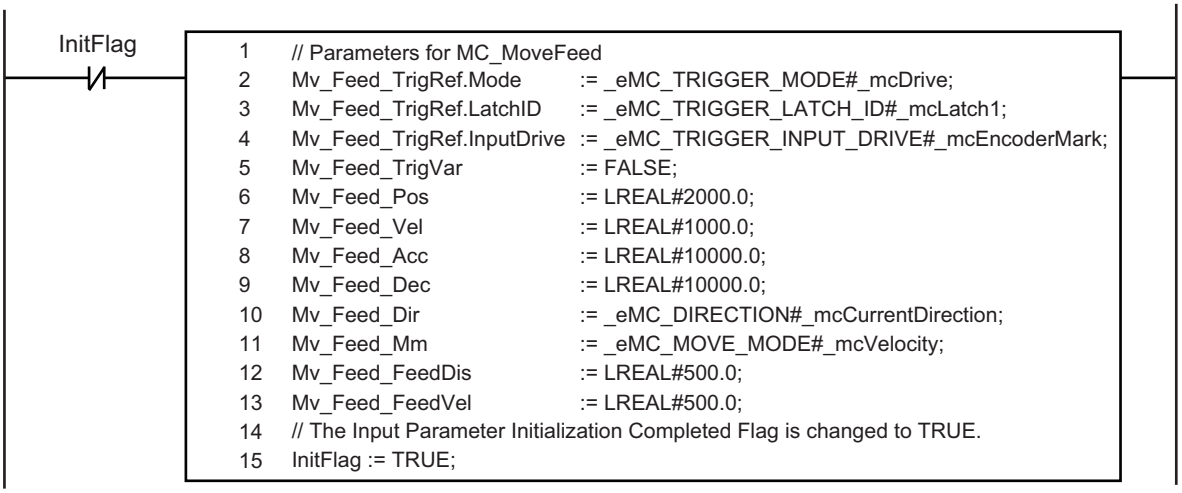
If a minor fault level error occurs for axis 0, the error handler for the device (FaultHandler) is executed.
Program the FaultHandler according to the device.



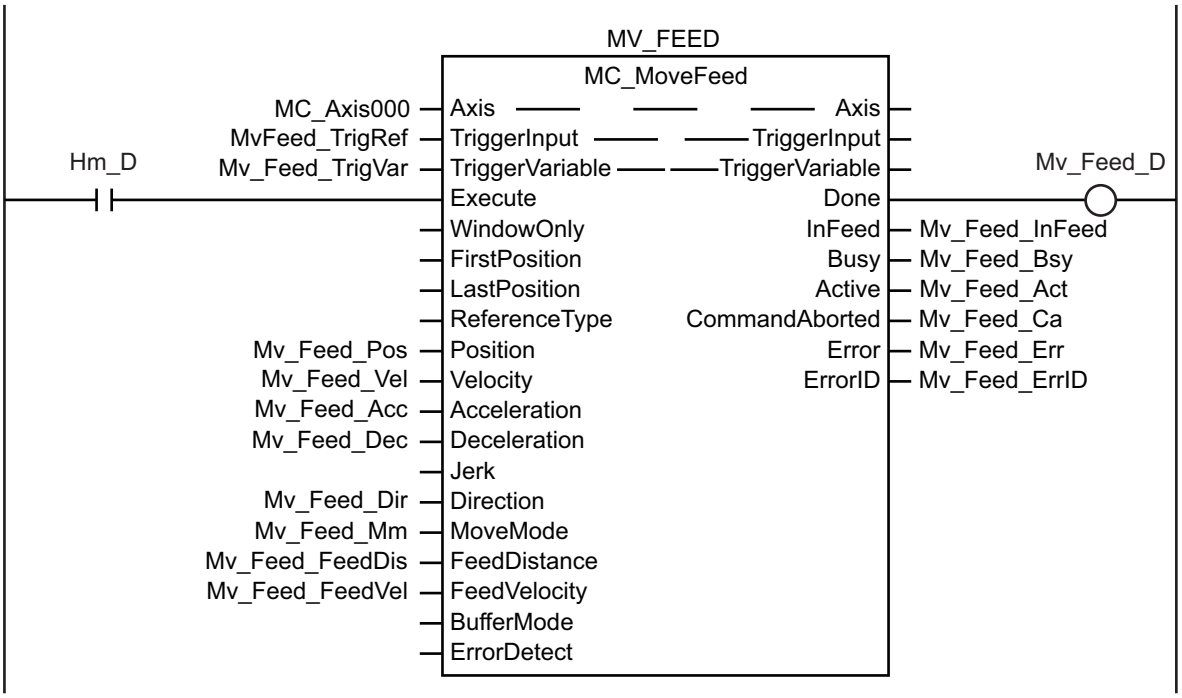
If the Servo is ON for axis 0 and home is not defined, the MC_Home instruction is executed.



The input parameters for interrupt feeding are set.



If homing is completed, interrupt feeding is executed.



ST Programming

```
// If the input parameters for interrupt feeding are not set, the target values and
 other parameters are set.
IF InitFlag=FALSE THEN
  // Parameters for MC_MoveFeed
    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#_mcEncoderMark;
    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_Dir := _eMC_DIRECTION#_mcCurrentDirection;
    Mv_Feed_Mm := _eMC_MOVE_MODE#_mcVelocity;
    Mv_Feed_FeedDis := LREAL#500.0;
    Mv_Feed_FeedVel := LREAL#500.0;
  // The Input Parameter Initialization Completed Flag is changed to TRUE.
    InitFlag := TRUE;
END_IF;

// If the Servo Drive is ready when StartPg is TRUE, turn ON the Servo for axis 0.
IF (StartPg=TRUE)
  AND (MC_Axis000.DrvStatus.Ready=TRUE) THEN
    Pwr_En:=TRUE;
  ELSE
    Pwr_En:=FALSE;
END_IF;

// If a minor fault level error occurs for axis 0, the error handler for the device
 (FaultHandler) is executed.
// Program the FaultHandler according to the device.
IF MC_Axis000.MFaultLvl.Active=TRUE THEN
  FaultHandler();
END_IF;

// If the Servo is ON for axis 0 and home is not defined, the MC_Home instruction i
s executed.
IF (Pwr_Status=TRUE) AND (MC_Axis000.Details.Homed=FALSE) THEN
  Hm_Ex :=TRUE;
END_IF;

// If homing is defined, interrupt feeding is executed.
IF Hm_D=TRUE THEN
  Mv_Feed_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_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,
  Direction := Mv_Feed_Dir,
  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
);
```

10-2-12 Changing the Cam Table by Re-execution of an Instruction

This sample changes the cam table during cam motion.

CamProfile0 is used when the command position for axis 0 is 5000 or less and CamProfile1 is used when it is over 5000.

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 before or after the cam table is switched. Since this may affect the mechanical machine, be sure to thoroughly verify that there will be no excessive changes in velocity or acceleration before switching the cam table.

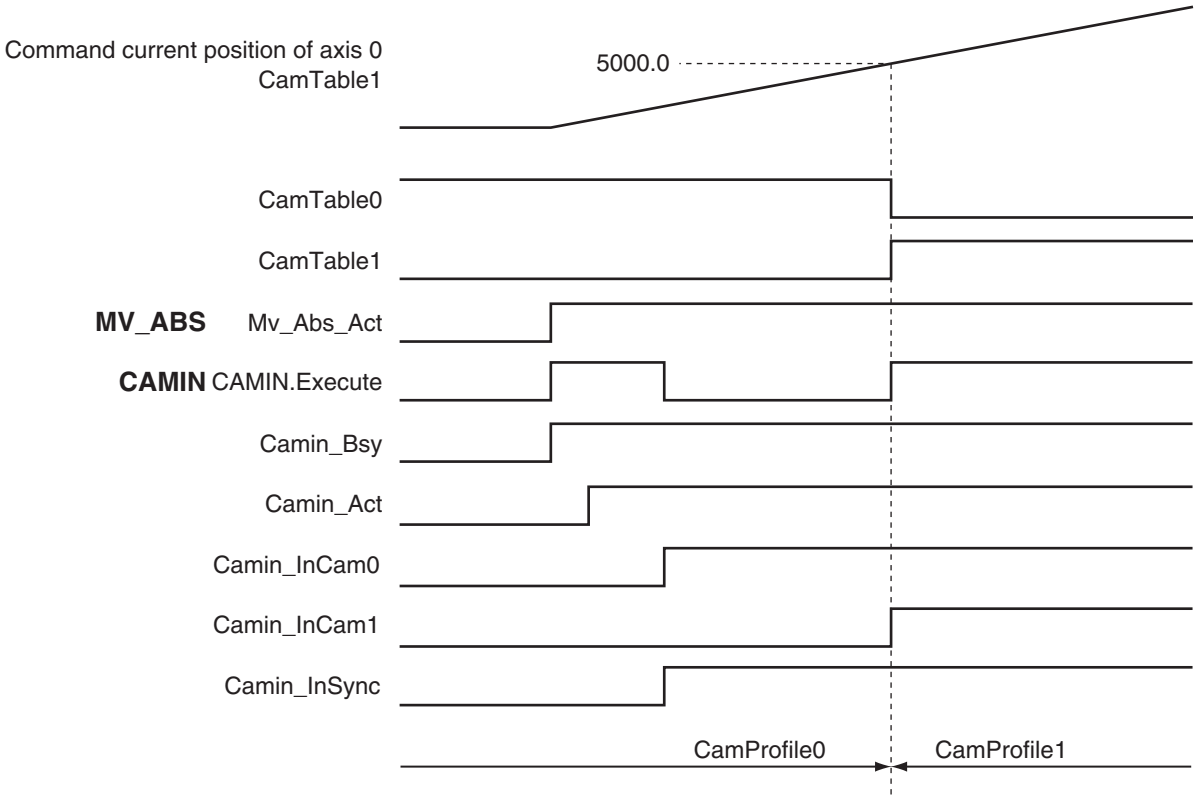
Main Variables Used in the Programming Samples

Variable name	Data type	Default	Comment
MC_Axis000	_sAXIS_REF	---	This is the Axis Variable for axis 0.
MC_Axis001	_sAXIS_REF	---	This is the Axis Variable for axis 1.
CamProfile0	ARRAY[0..100] OF _sMC_CAM_REF	---	This is the cam data variable. *1
CamProfile1	ARRAY[0..10] OF _sMC_CAM_REF	---	This is the cam data variable. *1
Pwr1_S	BOOL	FALSE	This variable is assigned to the Status output variable from the PWR1 instance of the MC_Power instruction. It is TRUE when the Servo is ON.
Pwr2_S	BOOL	FALSE	This variable is assigned to the Status output variable from the PWR2 instance of the MC_Power instruction. It is TRUE when the Servo is ON.
CamTable0	BOOL	FALSE	TRUE when CamProfile0 is used for the cam table.
CamTable1	BOOL	FALSE	TRUE when CamProfile1 is used for the cam table.
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 motion for CamProfile0. After this variable changes to TRUE, it remains TRUE until the operation is completed or canceled.
Camin_InCam1	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 motion for CamProfile1. After this variable changes to TRUE, it remains TRUE until the operation is completed or canceled.
Mv_Abs_Act	BOOL	FALSE	This variable is assigned to the Active output variable from the MV_ABS instance of the MC_MoveAbsolute instruction.
StartPg	BOOL	FALSE	When this variable is TRUE, the Servo is turned ON if EtherCAT process data communications are active and normal.
Hm1_Ex	BOOL	FALSE	This variable is used to re-execute the MC_Home instruction for axis 0.
Hm2_Ex	BOOL	FALSE	This variable is used to re-execute the MC_Home instruction for axis 1.
Mv_Abs_Ex	BOOL	FALSE	This variable is used to execute the MC_MoveAbsolute (Absolute Positioning) instruction.
Camin_Ex	BOOL	FALSE	This variable is used to execute the MC_CamIn (Start Cam Operation) instruction. It is used in ST programming.

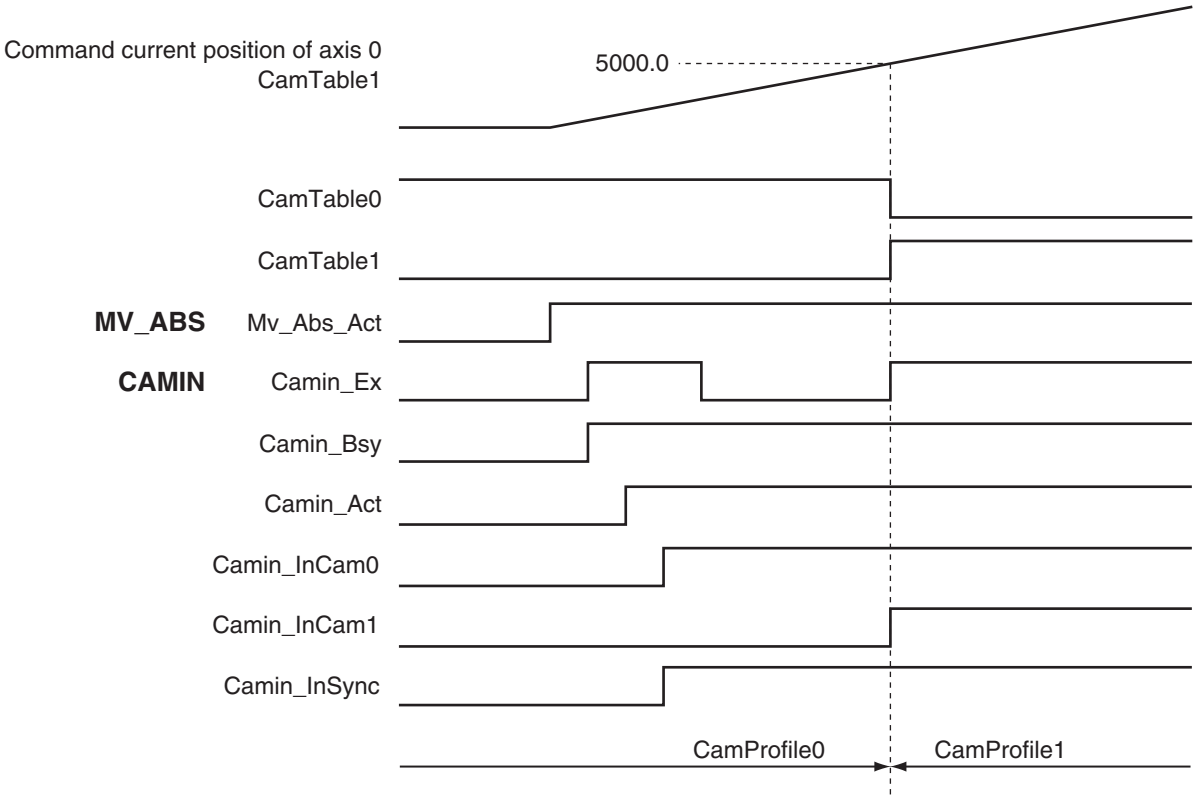
*1. The array elements ARRAY[0..N] are set with the Cam Editor in the Sysmac Studio. The range of the array is 0 to 109 in this sample.

Timing Chart

● Ladder Diagram



● ST Programming

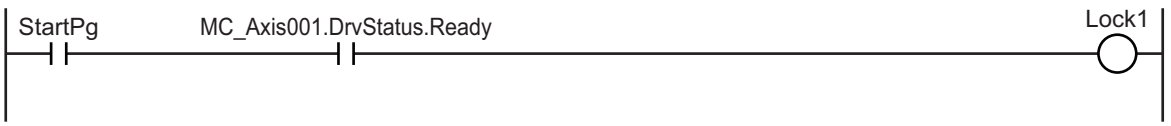


Ladder Diagram

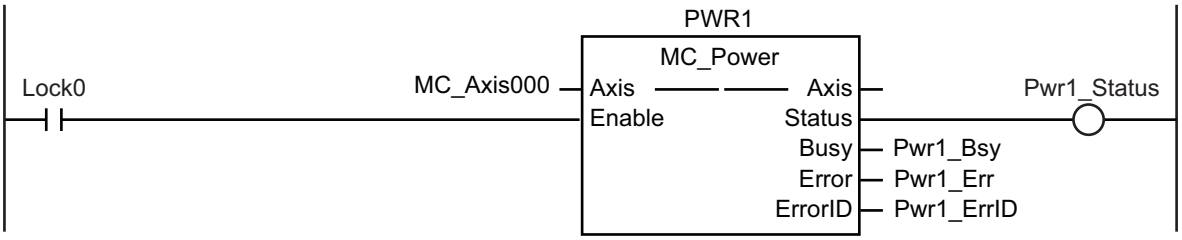
To change from one cam table to another, two instances of the MC_CamIn (Start Cam Operation) instruction with the same instance name are used.
A different output parameter is assigned to the *InCam* (Cam Motion) output variable from each instance. An error will occur if you assign the same output parameter.
In this sample, a JMP (Jump) instruction is used so that both instances are not executed at the same time.

Check if the Servo Drive for axis 0 is ready when *StartPg* is TRUE.

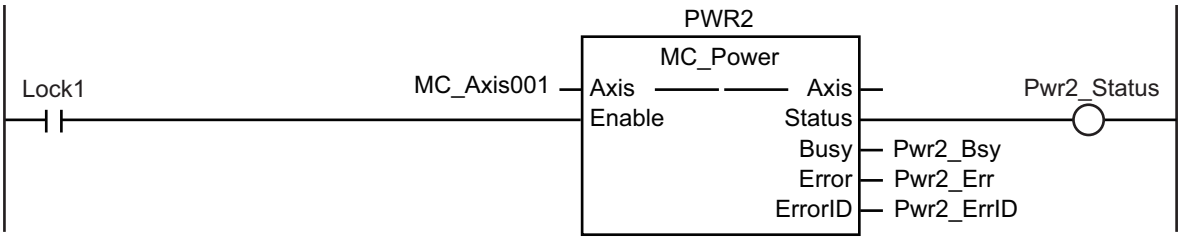
Check if the Servo Drive for axis 1 is ready when *StartPg* is TRUE.



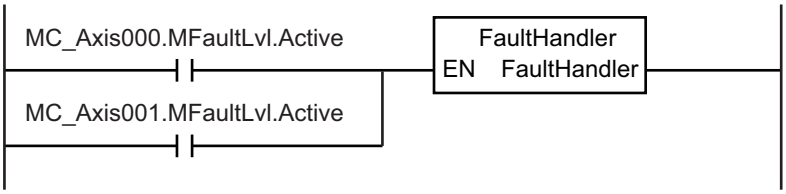
If the Servo Drive for axis 0 is ready, turn ON the Servo for axis 0.



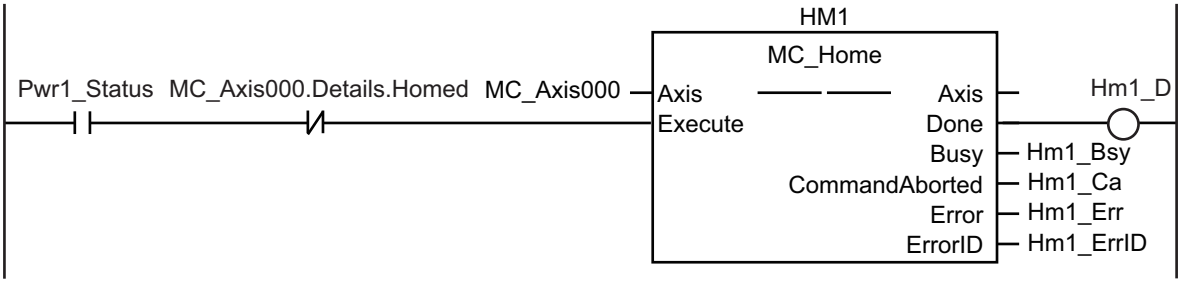
If the Servo Drive for axis 1 is ready, turn ON the Servo for axis 1.



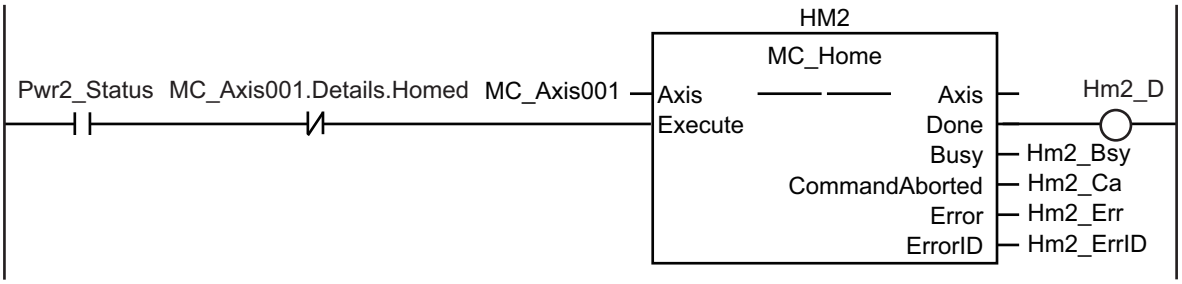
If a minor fault level error occurs for axis 0 or axis 1, the error handler for the device (FaultHandler) is executed. Program the FaultHandler according to the device.



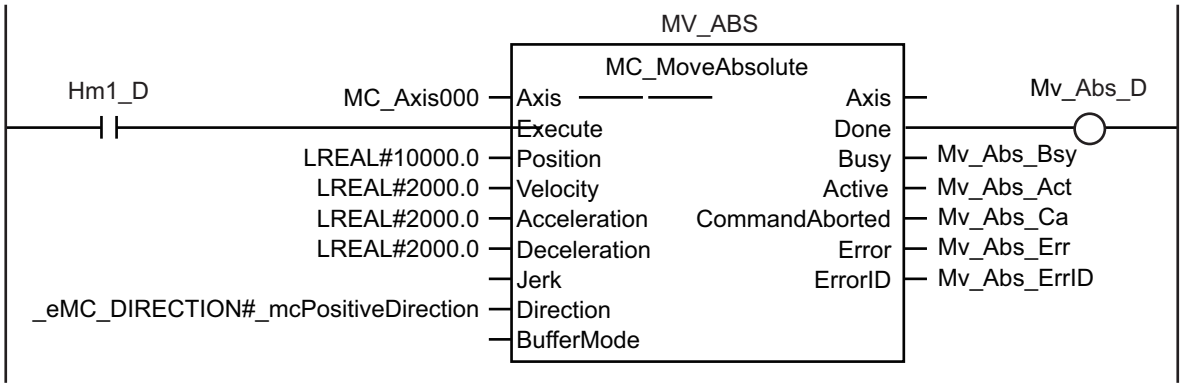
If the Servo is ON for axis 0 and home is not defined, the MC_Home instruction is executed.



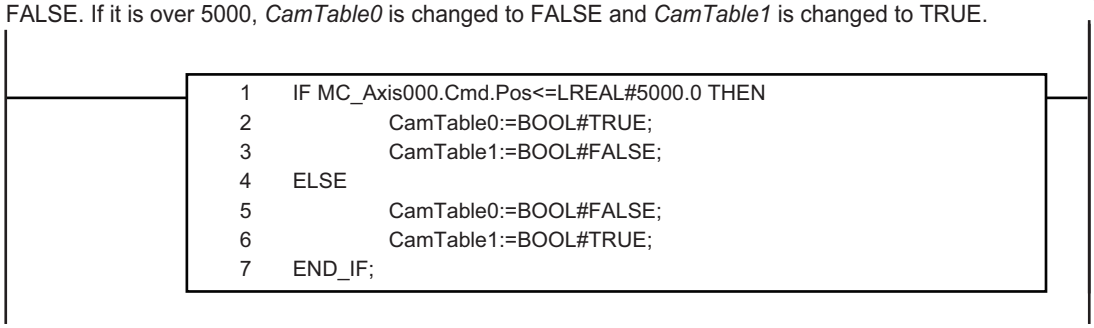
If the Servo is ON for axis 1 and home is not defined, the MC_Home instruction is executed.



If homing is completed for axis 0, absolute positioning is executed.

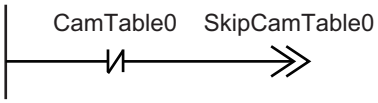


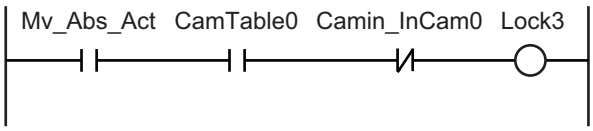
If the command position for axis 0 is 5000 or less, *CamTable0* is changed to TRUE and *CamTable1* is changed to FALSE. If it is over 5000, *CamTable0* is changed to FALSE and *CamTable1* is changed to TRUE.

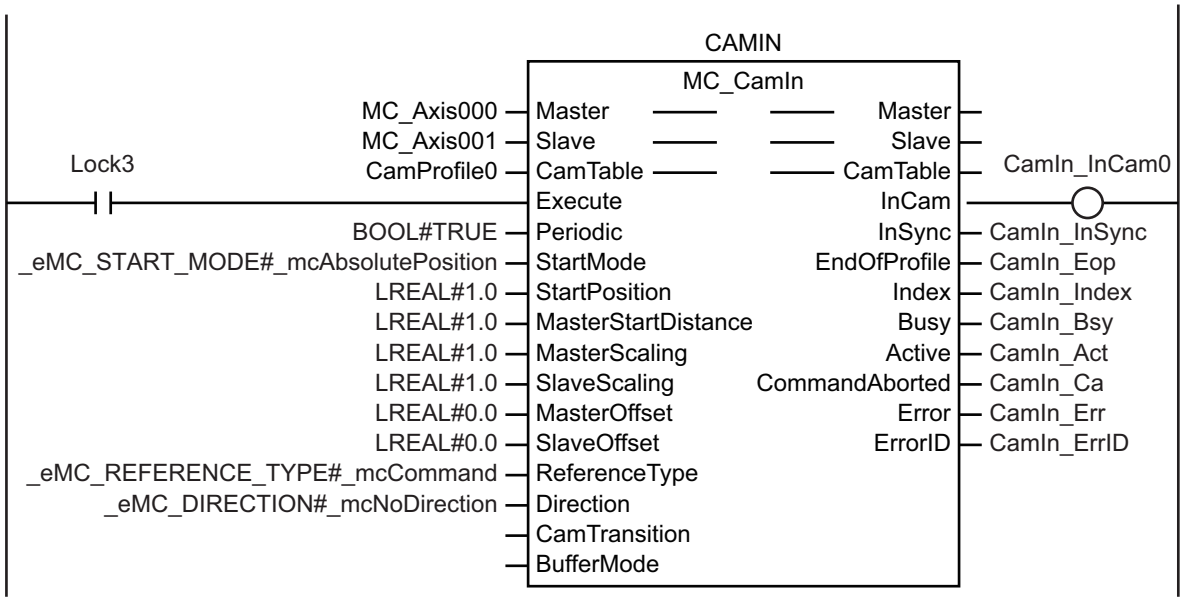


```
1   IF MC_Axis000.Cmd.Pos<=LREAL#5000.0 THEN
2           CamTable0:=BOOL#TRUE;
3           CamTable1:=BOOL#FALSE;
4   ELSE
5           CamTable0:=BOOL#FALSE;
6           CamTable1:=BOOL#TRUE;
7   END_IF;
```

If *CamTable0* is TRUE during absolute positioning, then the instance that uses *CamProfile0* for the cam table is executed.
If *InCam* is TRUE, then *Execute* is changed to FALSE.

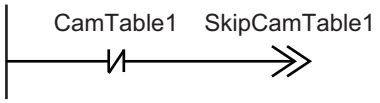


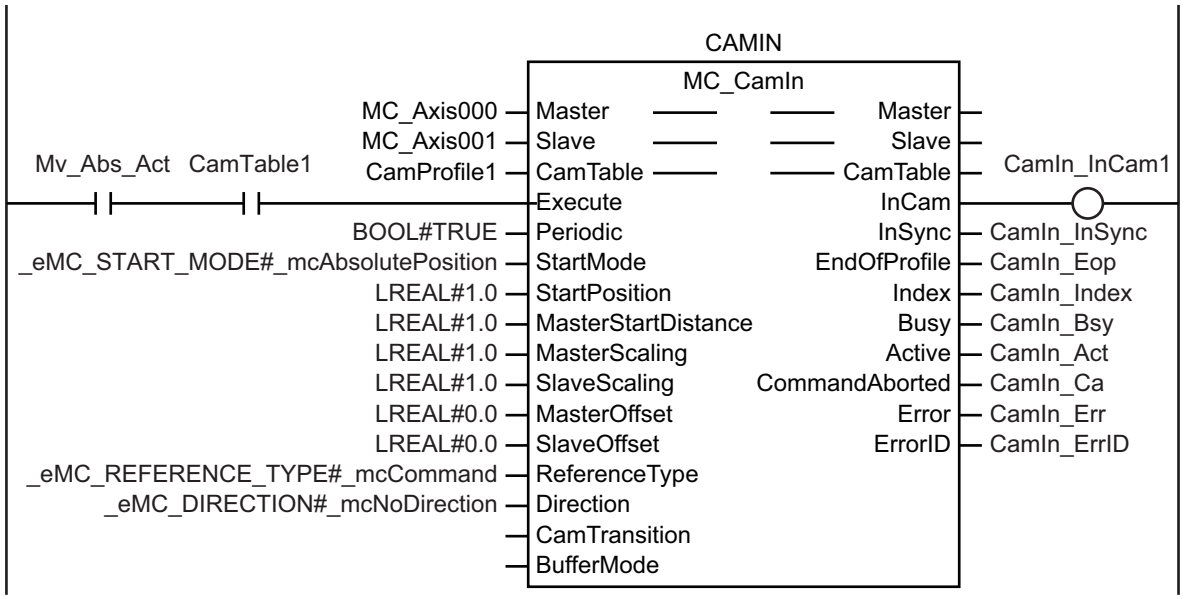




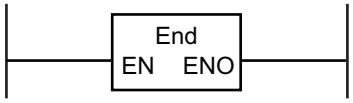
If *CamTable1* is TRUE during absolute positioning, then the instance that uses *CamProfile1* for the cam table is executed.

SkipCamTable0





SkipCamTable1



ST Programming

```
// If the input parameters for absolute positioning and starting cam operation are
not set, the target values and other parameters are set.
IF InitFlag=FALSE THEN
  // The input parameters for the MC_MoveAbsolute (Absolute Positioning) instructio
n are set.
    Mv_Abs_Pos := LREAL#10000.0;
    Mv_Abs_Vel := LREAL#2000.0;
    Mv_Abs_Acc := LREAL#2000.0;
    Mv_Abs_Dec := LREAL#2000.0;
  // The input parameters for the MC_CamIn (Start Cam Operation) instruction are se
t.
    Camin_EM := TRUE;
    Camin_StMode := _eMC_START_MODE#_mcAbsolutePosition;
    Camin_StPos := LREAL#1.0;
    Camin_MStDis := LREAL#1.0;
    Camin_MSc := LREAL#1.0;
    Camin_SSc := LREAL#1.0;
    Camin_MO := LREAL#0.0;
    Camin_SO := LREAL#0.0;
    Camin_RT := _eMC_REFERENCE_TYPE#_mcCommand;
    Camin_Dir := _eMC_DIRECTION#_mcNoDirection;
  // The cam table is selected.
    CamTable0 :=BOOL#TRUE;
    CamTable1 :=BOOL#FALSE;
  // The Input Parameter Initialization Completed Flag is changed to TRUE.
    InitFlag := TRUE;
END_IF;

// If the Servo Drive is ready when StartPg is TRUE, turn ON the Servo for axis 0.
IF (StartPg=TRUE)
  AND (MC_Axis000.DrvStatus.Ready=TRUE) THEN
    Pwr1_En:=TRUE;
  ELSE
    Pwr1_En:=FALSE;
END_IF;

// If the Servo Drive is ready when StartPg is TRUE, turn ON the Servo for axis 1.
IF (StartPg=TRUE)
  AND (MC_Axis001.DrvStatus.Ready=TRUE) THEN
    Pwr2_En :=TRUE;
  ELSE
    Pwr2_En :=FALSE;
END_IF;

// If a minor fault level error occurs for axis 0 or axis 1, the error handler for
```

```
the device (FaultHandler) is executed.
// Program the FaultHandler according to the device.
IF (MC_Axis000.MFaultLvl.Active=TRUE) OR (MC_Axis001.MFaultLvl.Active=TRUE) THEN
  FaultHandler();
END_IF;

// If the Servo is ON for axis 0 and home is not defined, the MC_Home instruction i
s executed.
IF (Pwr1_S=TRUE) AND (MC_Axis000.Details.Homed=FALSE) THEN
  Hm1_Ex :=TRUE;
END_IF;

// If the Servo is ON for axis 1 and home is not defined, the MC_Home instruction i
s executed.
IF (Pwr2_S=TRUE) AND (MC_Axis001.Details.Homed=FALSE) THEN
Hm2_Ex :=TRUE;
END_IF;

// If homing is completed for axis 0, absolute positioning is executed.
IF Hm1_D=TRUE THEN
  Mv_Abs_Ex := TRUE;
END_IF;

// If the command position for axis 0 is 5000 or less, CamTable0 is changed to TRUE
 and CamTable1 is changed to FALSE.
// If it exceeds 5000, CamTable0 is changed to FALSE and CamTable1 is changed to TR
UE.
IF MC_Axis000.Cmd.Pos<=LREAL#5000.0 THEN
    CamTable0 :=BOOL#TRUE;
    CamTable1 :=BOOL#FALSE;
  ELSE
    CamTable0 :=BOOL#FALSE;
    CamTable1 :=BOOL#TRUE;
END_IF;

// If CamTable0 is TRUE during absolute positioning, use CamProfile0 for the cam ta
ble.
// Execute the instance.
// If InCam is TRUE, Execute is changed to FALSE.
IF (Mv_Abs_Act=TRUE)
  AND (CamTable0=TRUE)
  AND (Camin_InCam0=FALSE) THEN
    Camin_Ex := TRUE;
  ELSE
    Camin_Ex := FALSE;
END_IF;
```

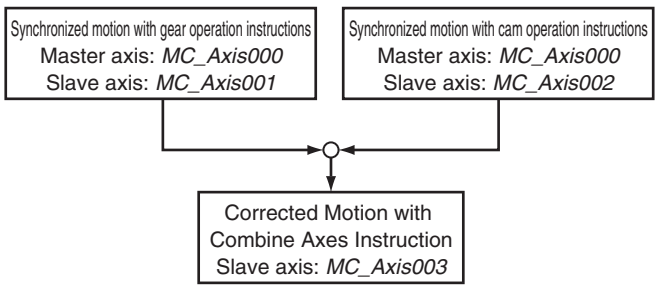
```
// If CamTable1 is TRUE during absolute positioning, use CamProfile1 for the cam ta
ble.
// Execute the instance.
IF (Mv_Abs_Act=TRUE) AND (CamTable1=TRUE) THEN
  Camin_Ex := TRUE;
END_IF;

//MC_Camin
IF CamTable0=TRUE THEN
  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,
    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
  );
END_IF;

IF CamTable1=TRUE THEN
  CAMIN(
    Master := MC_Axis000,
    Slave := MC_Axis001,
    CamTable := CamProfile1,
    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,
    InCam => Camin_InCam1,
    InSync => Camin_InSync,
    EndOfProfile => Camin_EOP,
    Index => Camin_Index,
    Busy => Camin_Bsy,
    Active => Camin_Act,
    CommandAborted => Camin_Ca,
    Error => Camin_Err,
    ErrorID => Camin_ErrID
  );
END_IF;

// MC_Power for axis 0
PWR1(
  Axis := MC_Axis000,
  Enable := Pwr1_En,
  Status => Pwr1_S,
  Busy => Pwr1_Bsy,
  Error => Pwr1_Err,
  ErrorID => Pwr1_ErrID
);

// MC_Power for axis 1
PWR2(
  Axis := MC_Axis001,
  Enable := Pwr2_En,
  Status => Pwr2_S,
  Busy => Pwr2_Bsy,
  Error => Pwr2_Err,
  ErrorID => Pwr2_ErrID
);

// MC_Home for axis 0
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 1
HM2(
  Axis := MC_Axis001,
  Execute := Hm2_Ex,
  Done => Hm2_D,
  Busy => Hm2_Bsy,
  CommandAborted => Hm2_Ca,
  Error => Hm2_Err,
  ErrorID => Hm2_ErrID
);

//MC_MoveAbsolute
MV_ABS(
Axis := MC_Axis000,
Execute := Mv_Abs_Ex,
Position := Mv_Abs_Pos,
Velocity := Mv_Abs_Vel,
Acceleration := Mv_Abs_Acc,
Deceleration := Mv_Abs_Dec,
Direction := Mv_Abs_Dir,
Done => Mv_Abs_D,
Busy => Mv_Abs_Bsy,
Active => Mv_Abs_Act,
CommandAborted => Mv_Abs_Ca,
Error => Mv_Abs_Err,
ErrorID => Mv_Abs_ErrID
);
```

10-2-13 Using a Cam Profile Curve to Correct the Sync Position

This sample uses a cam profile curve to correct a slave axis in a gear motion.
The slave axis for gear motion is MC_Axis001, a virtual Servo axis, and the slave axis for cam motion is MC_Axis002, also a virtual Servo axis. These slave axes are combined with MC_CombineAxes and the results is output to MC_Axis003, a Servo axis. The master axis is MC_Axis000, a Servo axis.
The processing flow is as follows:



● Axis Type Settings

The axes types are set in the axis parameters for each axis as given below.

Parameter name	Setting			
	Axis 1	Axis 2	Axis 3	Axis 4
Axes variable name	MC_Axis000	MC_Axis001	MC_Axis002	MC_Axis003
Axis type	Servo axis	Virtual servo axis	Virtual servo axis	Servo axis

Main Variables Used in the Programming Samples

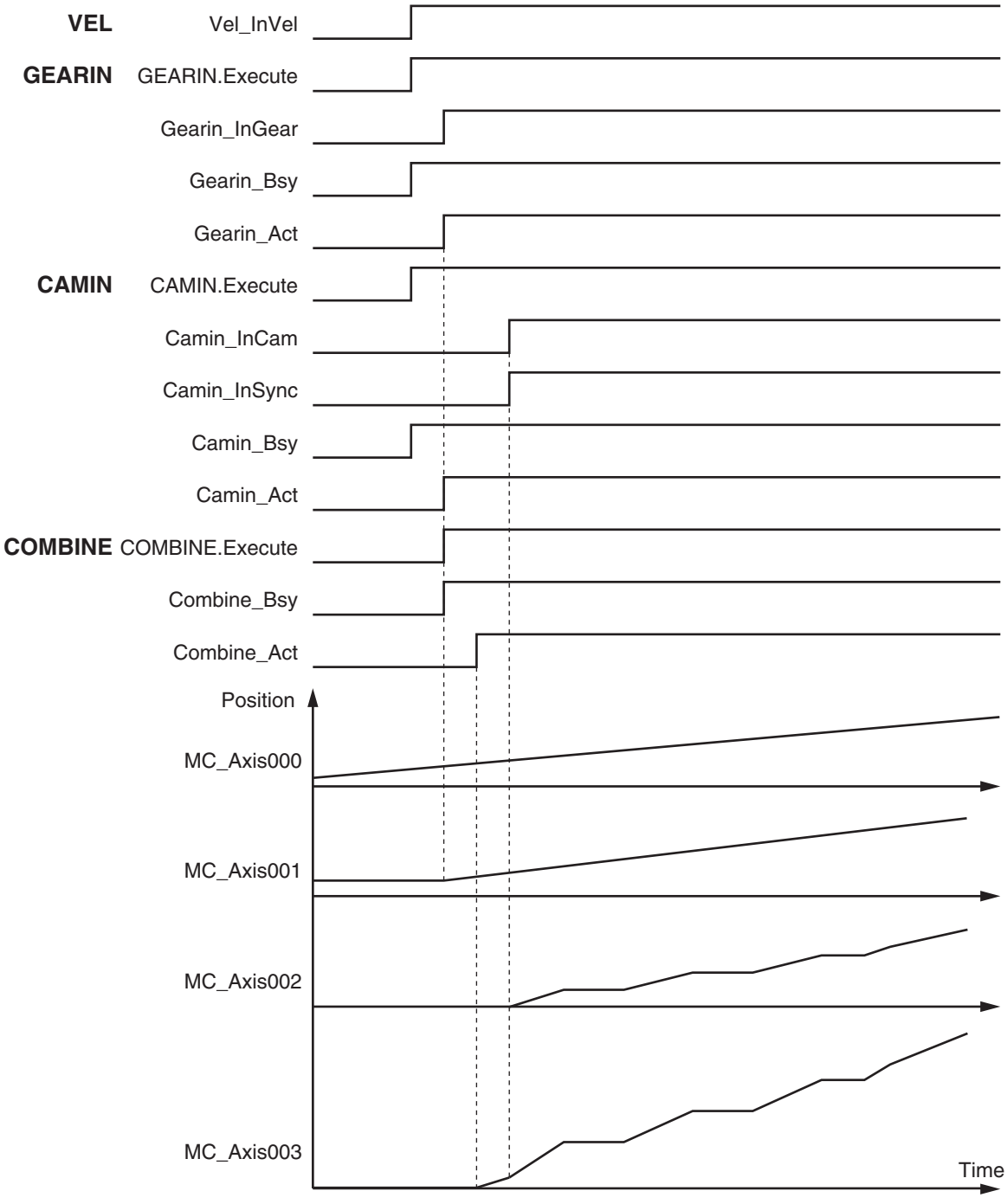
Variable name	Data type	Default	Comment
MC_Axis000	_sAXIS_REF	---	This is the Axis Variable for axis 0.
MC_Axis000.MFaultLvl.Active	BOOL	FALSE	TRUE when there is a minor fault level error for axis 0.
MC_Axis000.Details.Homed	BOOL	FALSE	TRUE when home is defined for axis 0.
MC_Axis001	_sAXIS_REF	---	This is the Axis Variable for axis 1.
MC_Axis001.MFaultLvl.Active	BOOL	FALSE	TRUE when there is a minor fault level error for axis 1.
MC_Axis002	_sAXIS_REF	---	This is the Axis Variable for axis 2.
MC_Axis002.MFaultLvl.Active	BOOL	FALSE	TRUE when there is a minor fault level error for axis 2.
MC_Axis003	_sAXIS_REF	---	This is the Axis Variable for axis 3.
MC_Axis003.MFaultLvl.Active	BOOL	FALSE	TRUE when there is a minor fault level error for axis 3.
MC_Axis003.Details.Homed	BOOL	FALSE	TRUE when home is defined for axis 3.
CamProfile0	ARRAY[0..109] OF _sMC_CAM_REF	---	This is the cam data variable. *1
Pwr1_Status	BOOL	FALSE	This variable is assigned to the Status output variable from the PWR1 instance of the MC_Power instruction. It is TRUE when the Servo is ON.
Pwr4_Status	BOOL	FALSE	This variable is assigned to the Status output variable from the PWR4 instance of the MC_Power instruction. It is TRUE when the Servo is ON.
Vel_InVel	BOOL	FALSE	TRUE when the target velocity for MC_MoveVelocity for axis 0 is reached.
StartPg	BOOL	FALSE	When this variable is TRUE, the Servo is turned ON if EtherCAT process data communications are active and normal.
Gearin_Ex	BOOL	FALSE	This variable is used to execute the MC_GearIn (Start Gear Operation) instruction. *2
Camin_Ex	BOOL	FALSE	This variable is used to execute the MC_CamIn (Start Cam Operation) instruction. *2
Combine_Ex	BOOL	FALSE	This variable is used to execute the MC_CombineAxes (Combine Axes) instruction. *2

*1. The array elements ARRAY[0..N] are set with the Cam Editor in the Sysmac Studio. The range of the array is 0 to 109 in this sample.

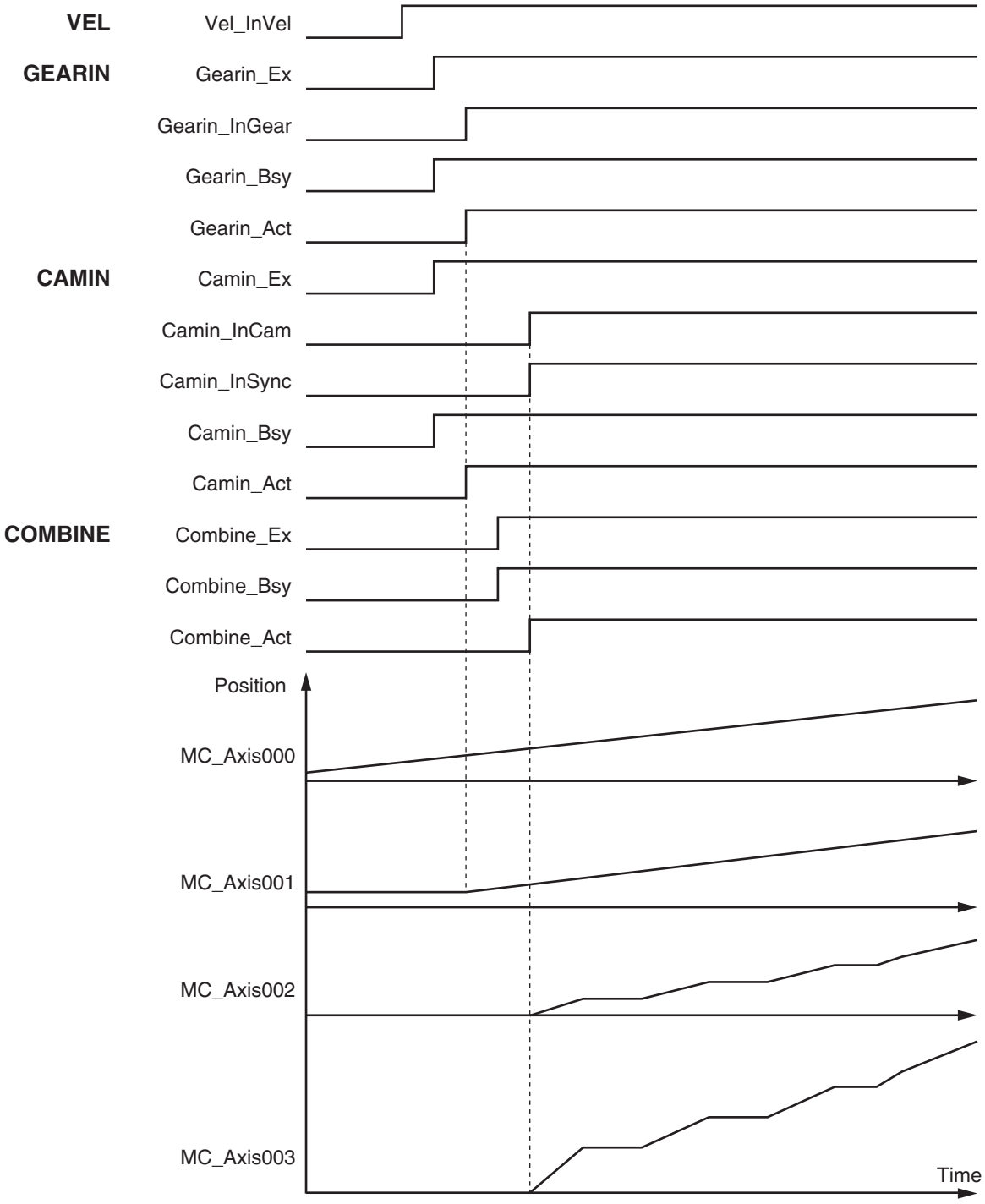
*2. The variable is used in ST programming.

Timing Chart

● Ladder Diagram



● ST Programming

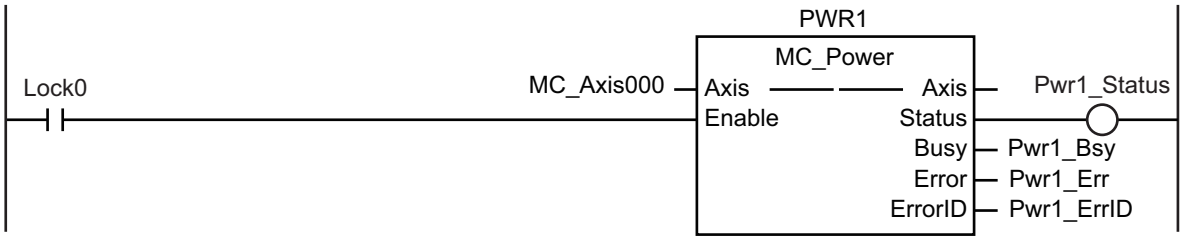


Ladder Diagram

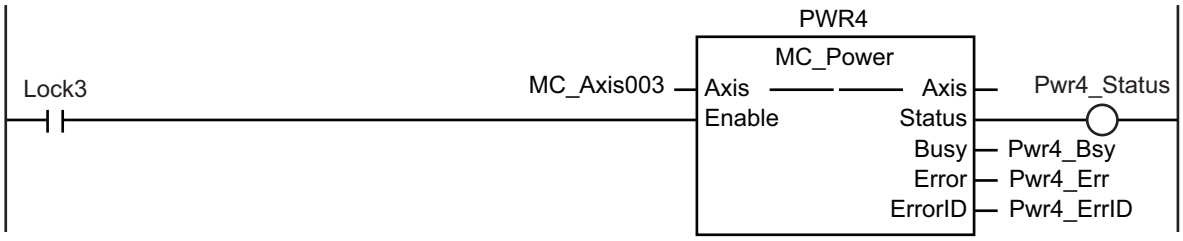
Check if the Servo Drive for axis 0 is ready when *StartPg* is TRUE.

Check if the Servo Drive for axis 3 is ready when *StartPg* is TRUE.

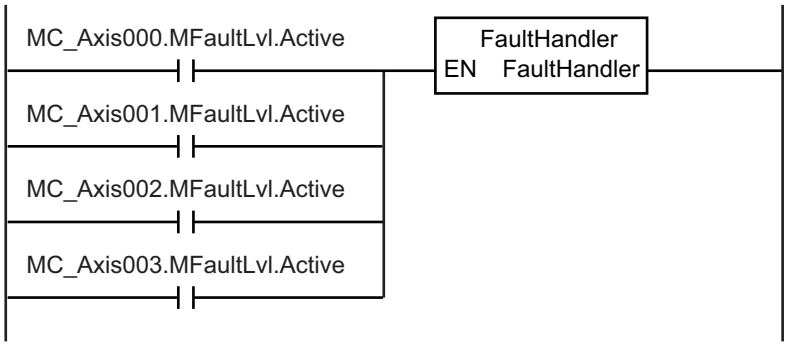
If the Servo Drive for axis 0 is ready, turn ON the Servo for axis 0.



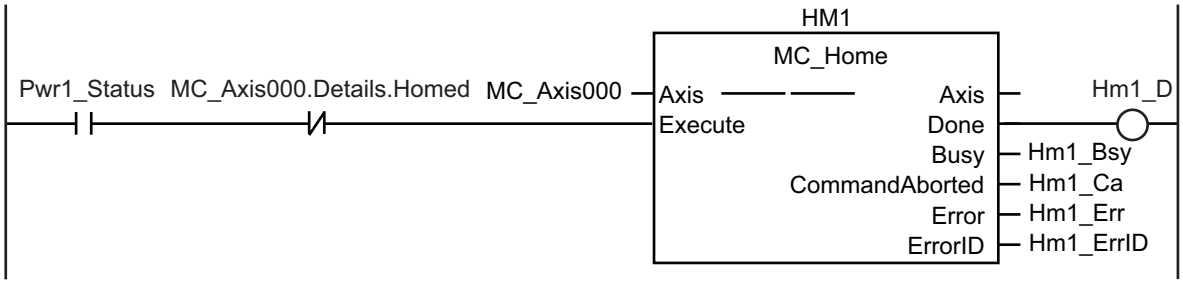
If the Servo Drive for axis 3 is ready, turn ON the Servo for axis 3.



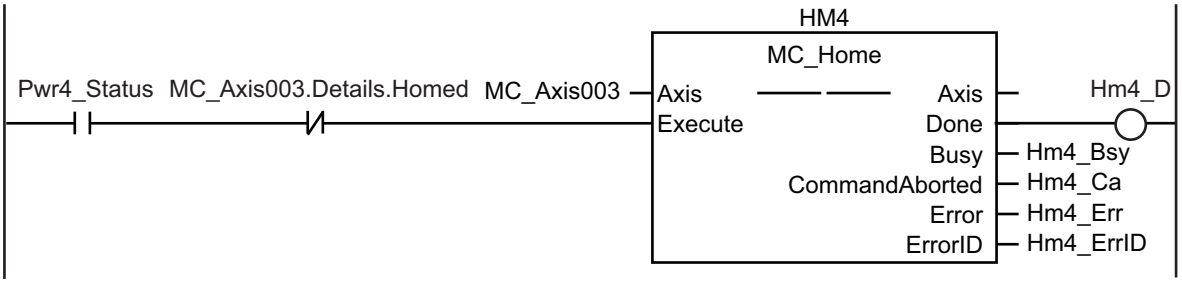
If a minor fault level error occurs for any of the composition axes in the axes group, the error handler for the device (FaultHandler) is executed. Program the FaultHandler according to the device.



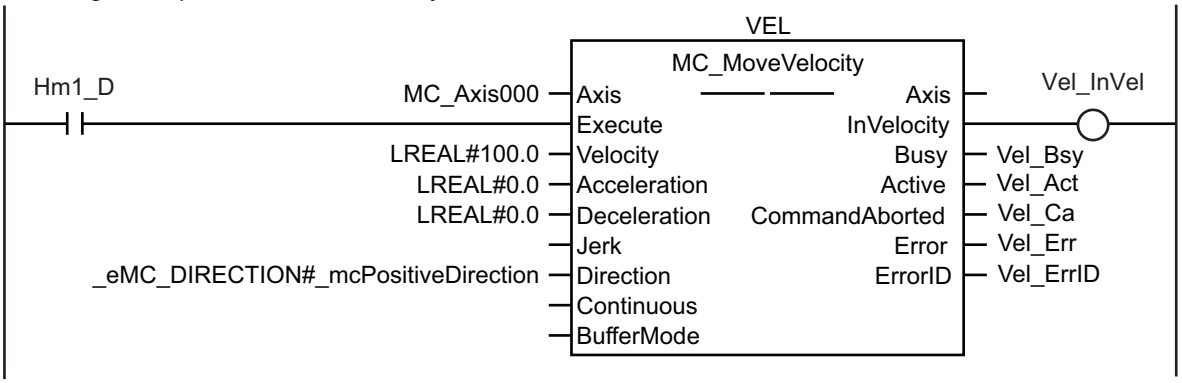
If the Servo is ON for axis 0 and home is not defined, the MC_Home instruction is executed.



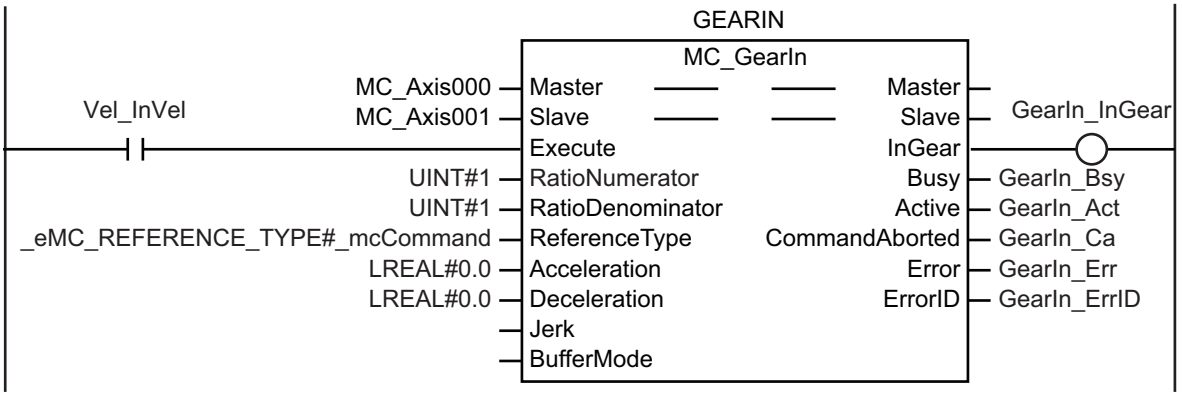
If the Servo is ON for axis 3 and home is not defined, the MC_Home instruction is executed.



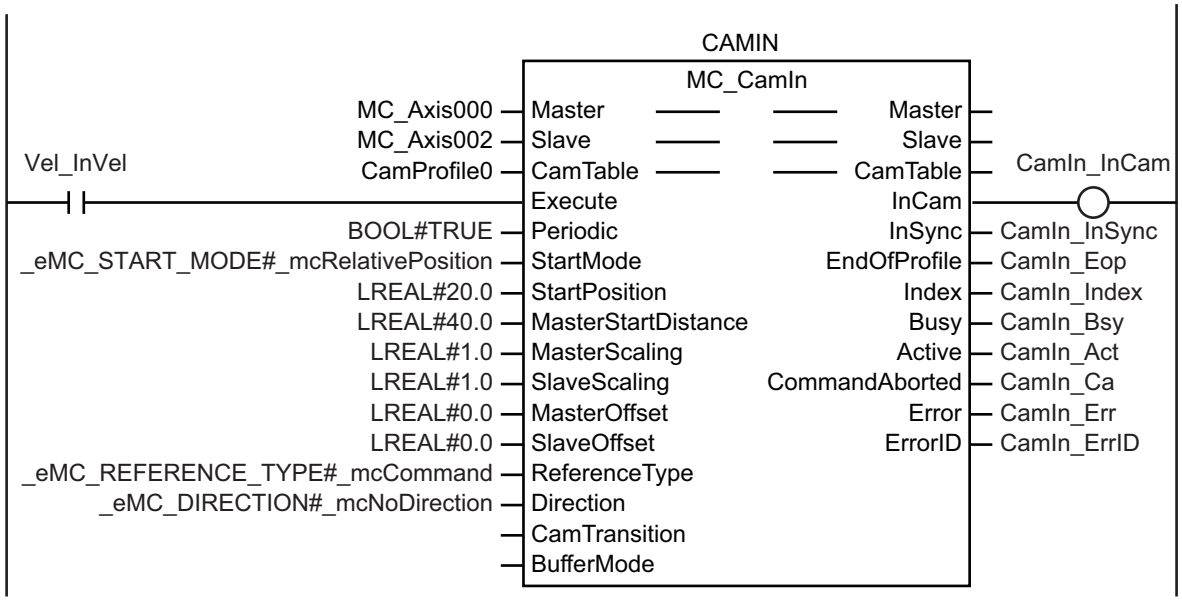
If homing is completed for axis 0, velocity control is executed.



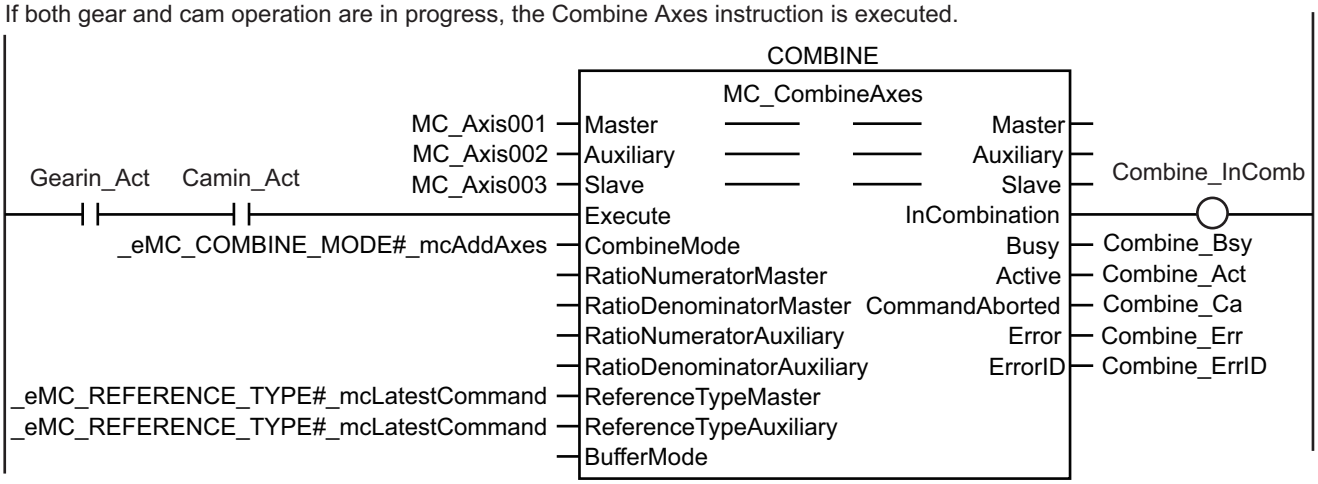
If homing is completed for axis 0, gear operation is executed.



When axis 0 reaches the target velocity, cam operation is executed.



If both gear and cam operation are in progress, the Combine Axes instruction is executed.



ST Programming

```
// If the input parameters for the motion instructions are not set, the target valu
es and other parameters are set.
IF InitFlag=FALSE THEN
  // The input parameters for the MC_MoveVelocity (Velocity Control) instruction ar
e set.
    Vel_Vel := LREAL#100.0;
    Vel_Acc := LREAL#0.0;
    Vel_Dec := LREAL#0.0;
    Vel_Dir := _eMC_DIRECTION#_mcPositiveDirection;
  // The input parameters for the MC_CamIn (Start Cam Operation) instruction are se
t.
    Camin_Em := TRUE;
    Camin_Sm := _eMC_START_MODE#_mcRelativePosition;
    Camin_Sp := LREAL#20.0;
```

```
    Camin_Msd := LREAL#40.0;
    Camin_Ms := LREAL#1.0;
    Camin_Ss := LREAL#1.0;
    Camin_Mo := LREAL#0.0;
    Camin_So := LREAL#0.0;
    Camin_Rt := _eMC_REFERENCE_TYPE#_mcCommand;
    Camin_Dir := _eMC_DIRECTION#_mcNoDirection;
  // The input parameters for the MC_GearIn (Start Gear Operation) instruction are
set.
    Gearin_RatN := UINT#1;
    Gearin_RatD := UINT#1;
    Gearin_RefTyp := _eMC_REFERENCE_TYPE#_mcCommand;
    Gearin_Acc := LREAL#0.0;
    Gearin_Dec := LREAL#0.0;
  // The input parameters for the MC_CombineAxes (Combine Axes) instruction are set
.
    Combine_Cm := _eMC_COMBINE_MODE#_mcAddAxes;
    Combine_RefMas :=_eMC_REFERENCE_TYPE#_mcLatestCommand;
    Combine_RefAux :=_eMC_REFERENCE_TYPE#_mcLatestCommand;
  // The Input Parameter Initialization Completed Flag is changed to TRUE.
    InitFlag := TRUE;
END_IF;

// If the Servo Drive is ready when StartPg is TRUE, turn ON the Servo for axis 0.
IF (StartPg=TRUE)
  AND (MC_Axis000.DrvStatus.Ready=TRUE) THEN
    Pwr1_En:=TRUE;
  ELSE
    Pwr1_En:=FALSE;
END_IF;

// If the Servo Drive is ready when StartPg is TRUE, turn ON the Servo for axis 3.
IF (StartPg=TRUE)
  AND (MC_Axis003.DrvStatus.Ready=TRUE) THEN
    Pwr4_En :=TRUE;
  ELSE
    Pwr4_En :=FALSE;
END_IF;

// If a minor fault level error occurs for axis 0 to axis 3, the error handler for
the device (FaultHandler) is executed.
// Program the FaultHandler according to the device.
IF (MC_Axis000.MFaultLvl.Active=TRUE)
  OR (MC_Axis001.MFaultLvl.Active=TRUE)
  OR (MC_Axis002.MFaultLvl.Active=TRUE)
  OR (MC_Axis003.MFaultLvl.Active=TRUE) THEN
    FaultHandler();
```

```
END_IF;

// If the Servo is ON for axis 0 and home is not defined, the MC_Home instruction i
s executed for axis 0.
IF (Pwr1_Status=TRUE) AND (MC_Axis000.Details.Homed=FALSE) THEN
  Hm1_Ex :=TRUE;
END_IF;

// If the Servo is ON for axis 3 and home is not defined, the MC_Home instruction i
s executed for axis 3.
IF (Pwr4_Status=TRUE) AND (MC_Axis003.Details.Homed=FALSE) THEN
  Hm4_Ex :=TRUE;
END_IF;

// If homing is completed for axis 0, velocity control is executed.
IF Hm1_D=TRUE THEN
  Vel_Ex :=TRUE;
END_IF;

// When axis 0 reaches the target velocity, gear operation is executed.
IF Vel_InVel=TRUE THEN
Gearin_Ex := TRUE;
END_IF;

// When axis 0 reaches the target velocity, cam operation is executed.
IF Vel_InVel=TRUE THEN
  Camin_Ex := TRUE;
END_IF;

// If both gear and cam operation are in progress, the Combine Axes instruction is
executed.
IF (Gearin_Act=TRUE) AND (Camin_Act=TRUE) THEN
  Combine_Ex :=TRUE;
END_IF;

// MC_Power for axis 0
PWR1(
  Axis := MC_Axis000,
  Enable := Pwr1_En,
  Status => Pwr1_Status,
  Busy => Pwr1_Bsy,
  Error => Pwr1_Err,
  ErrorID => Pwr1_ErrID
);

// MC_Power for axis 3
PWR4(
```

```
  Axis := MC_Axis003,
  Enable := Pwr4_En,
  Status => Pwr4_Status,
  Busy => Pwr4_Bsy,
  Error => Pwr4_Err,
  ErrorID => Pwr4_ErrID
);

// MC_Home for axis 0
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 3
HM4(
  Axis := MC_Axis003,
  Execute := Hm4_Ex,
  Done => Hm4_D,
  Busy => Hm4_Bsy,
  CommandAborted => Hm4_Ca,
  Error => Hm4_Err,
  ErrorID => Hm4_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_Axis002,
CamTable := CamProfile0,
Execute := Camin_Ex,
Periodic := Camin_Em,
StartMode := Camin_Sm,
StartPosition := Camin_Sp,
MasterStartDistance := Camin_Msd,
MasterScaling := Camin_Ms,
SlaveScaling := Camin_Ss,
MasterOffset := Camin_Mo,
SlaveOffset := Camin_So,
ReferenceType := Camin_Rt,
Direction := Camin_Dir,
InCam => Camin_InCam,
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_GearIn
GEARIN(
  Master := MC_Axis000,
  Slave := MC_Axis001,
  Execute := Gearin_Ex,
  RatioNumerator := Gearin_RatN,
  RatioDenominator := Gearin_RatD,
  ReferenceType := Gearin_RefTyp,
  Acceleration := Gearin_Acc,
  Deceleration := Gearin_Dec,
  InGear => Gearin_InGear,
  Busy => Gearin_Bsy,
  Active => Gearin_Act,
  CommandAborted => Gearin_Ca,
  Error => Gearin_Err,
  ErrorID => Gearin_ErrID
);

//MC_CombineAxes
COMBINE(
Master := MC_Axis001,
```

```
Auxiliary := MC_Axis002,
Slave := MC_Axis003,
Execute := Combine_Ex,
CombineMode := Combine_CM,
ReferenceTypeMaster := Combine_RefMas,
ReferenceTypeAuxiliary := Combine_RefAux,
InCombination => Combine_InComb,
Busy => Combine_Bsy,
Active => Combine_Act,
CommandAborted => Combine_Ca,
Error => Combine_Err,
ErrorID => Combine_ErrID
);
```

10-2-14 Shifting the Phase of a Master Axis in Cam Motion

This sample synchronizes a slave axis in cam motion with a master axis in velocity control. If StartOn is TRUE, the phase of the master axis is shifted with the MC_Phasing (Shift Master Axis Phase) instruction. The slave axis is synchronized with the shifted phase.

Main Variables Used in the Programming Samples

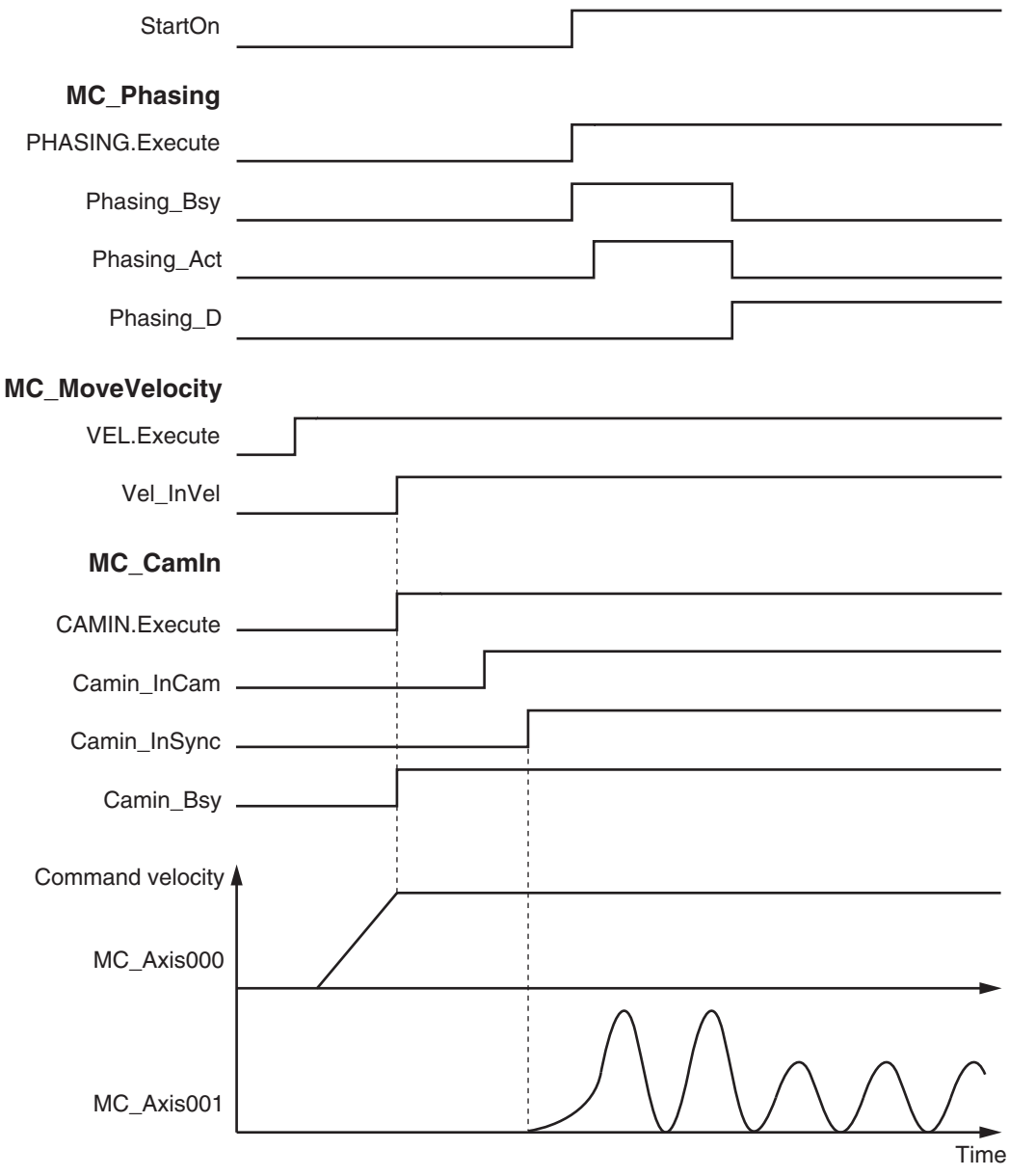
Variable name	Data type	Default	Comment
MC_Axis000	_sAXIS_REF	---	This is the Axis Variable for axis 0.
MC_Axis000.MFaultLvl.Active	BOOL	FALSE	TRUE when there is a minor fault level error for axis 0.
MC_Axis000.Details.Homed	BOOL	FALSE	TRUE when home is defined for axis 0.
MC_Axis001	_sAXIS_REF	---	This is the Axis Variable for axis 1.
MC_Axis001.MFaultLvl.Active	BOOL	FALSE	TRUE when there is a minor fault level error for axis 1.
MC_Axis001.Details.Homed	BOOL	FALSE	TRUE when home is defined for axis 1.
CamProfile0	ARRAY[0..360] OF _sMC_CAM_REF	---	This is the cam data variable. *1
Pwr1_Status	BOOL	FALSE	This variable is assigned to the Status output variable from the PWR1 instance of the MC_Power instruction. It is TRUE when the Servo is ON.
Pwr2_Status	BOOL	FALSE	This variable is assigned to the Status output variable from the PWR2 instance of the MC_Power instruction. It is TRUE when the Servo is ON.
StartOn	BOOL	FALSE	This variable is used to start shifting the phase of the master axis.
StartPg	BOOL	FALSE	When this variable is TRUE, the Servo is turned ON if EtherCAT process data communications are active and normal.
Camin_Ex	BOOL	FALSE	This variable is used to execute the MC_CamIn (Start Cam Operation) instruction. It is used in ST programming.

Variable name	Data type	Default	Comment
Vel_Ex	BOOL	FALSE	This variable is used to execute the MC_MoveVelocity (Velocity Control) instruction. It is used in ST programming.
Phasing_Ex	BOOL	FALSE	This variable is used to execute the MC_Phasing (Shift Master Axis Phase) instruction. It is used in ST programming.

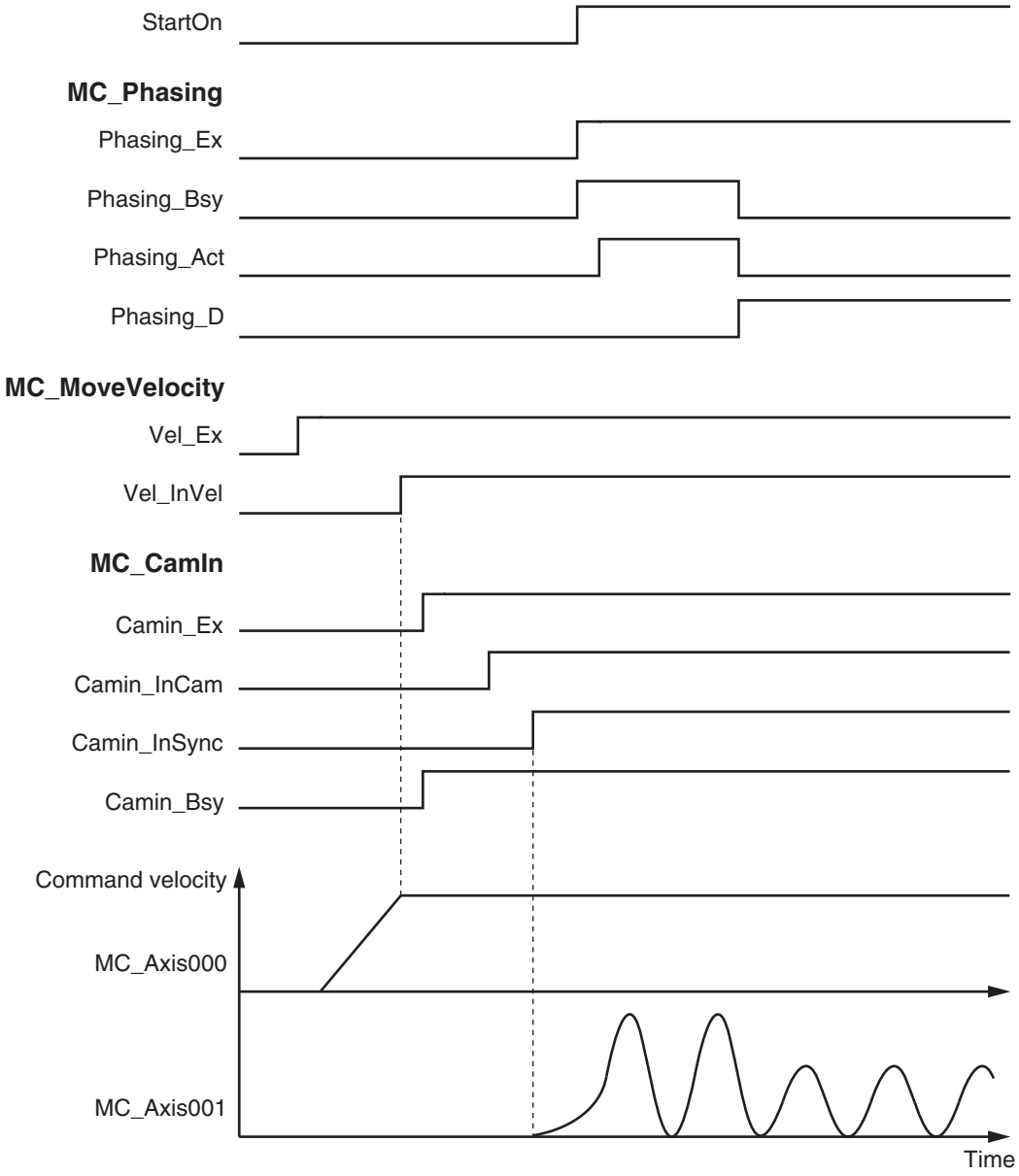
*1. The array elements ARRAY[0..N] are set with the Cam Editor in the Sysmac Studio. The range of the array is 0 to 360 in this sample.

Timing Chart

● Ladder Diagram



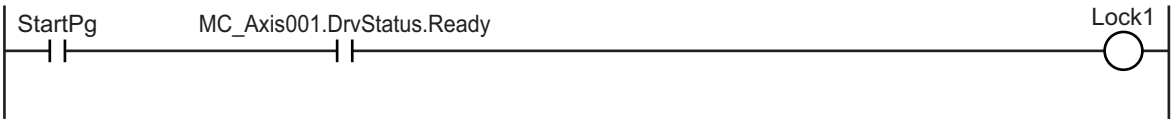
● **ST Programming**



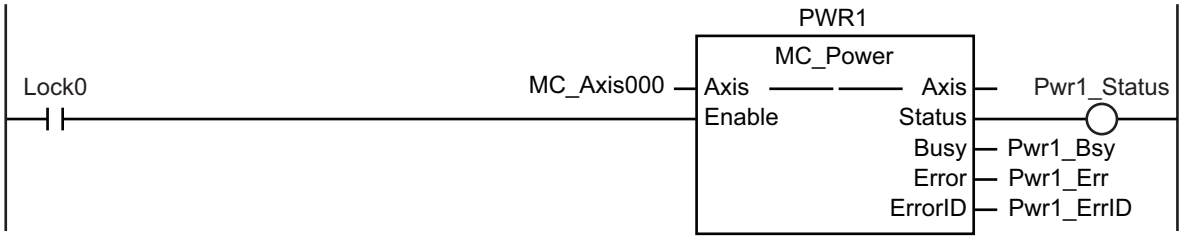
Ladder Diagram

Check if the Servo Drive for axis 0 is ready when *StartPg* is TRUE.

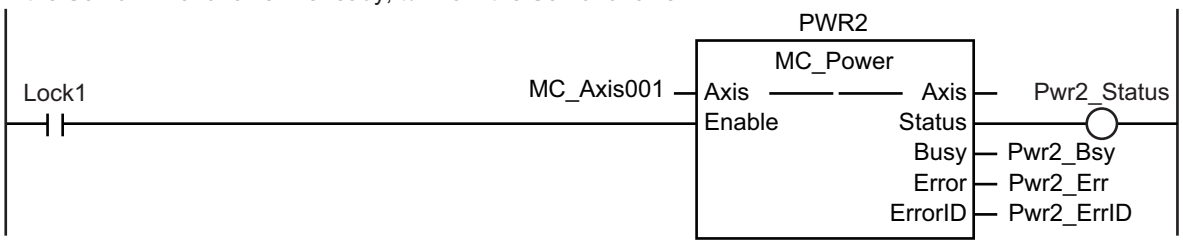
Check if the Servo Drive for axis 1 is ready when *StartPg* is TRUE.



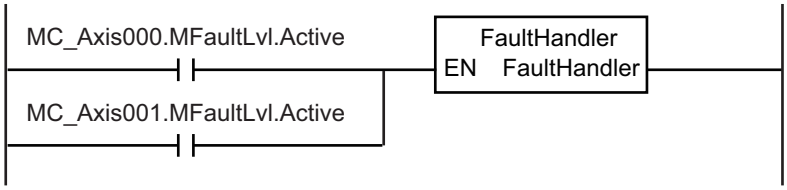
If the Servo Drive for axis 0 is ready, turn ON the Servo for axis 0.



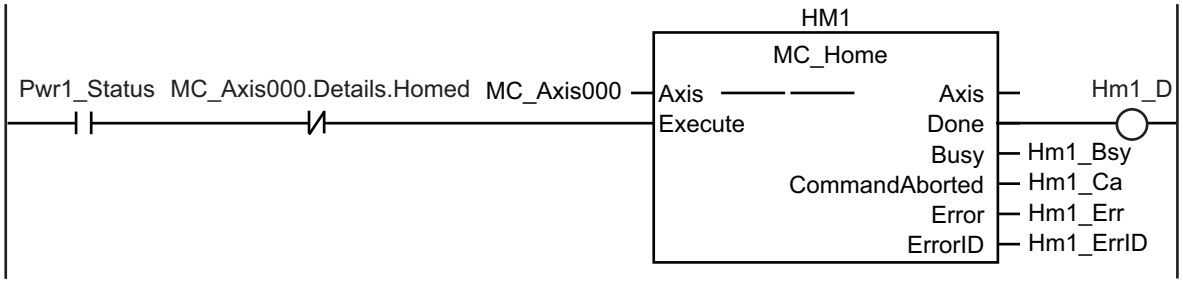
If the Servo Drive for axis 1 is ready, turn ON the Servo for axis 1.



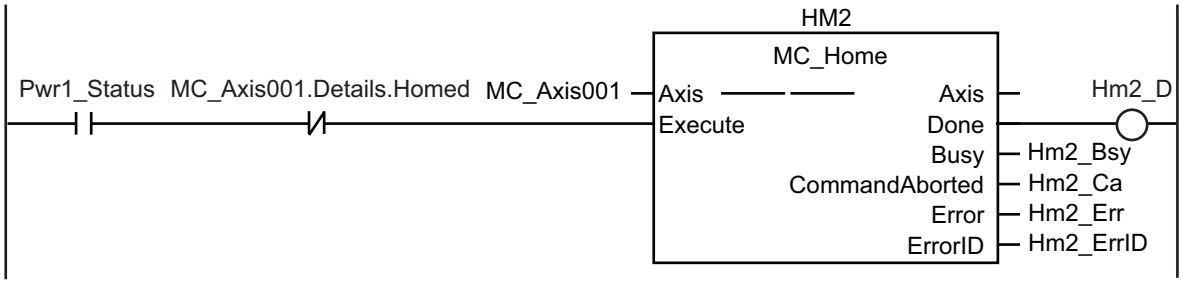
If a minor fault level error occurs for any of the composition axes in the axes group, the error handler for the device (FaultHandler) is executed. Program the FaultHandler according to the device.



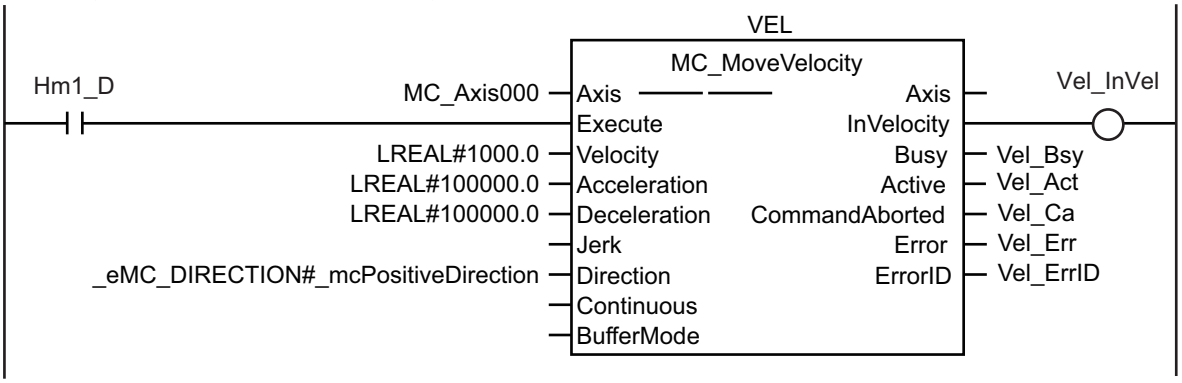
If the Servo is ON for axis 0 and home is not defined, the MC_Home instruction is executed.



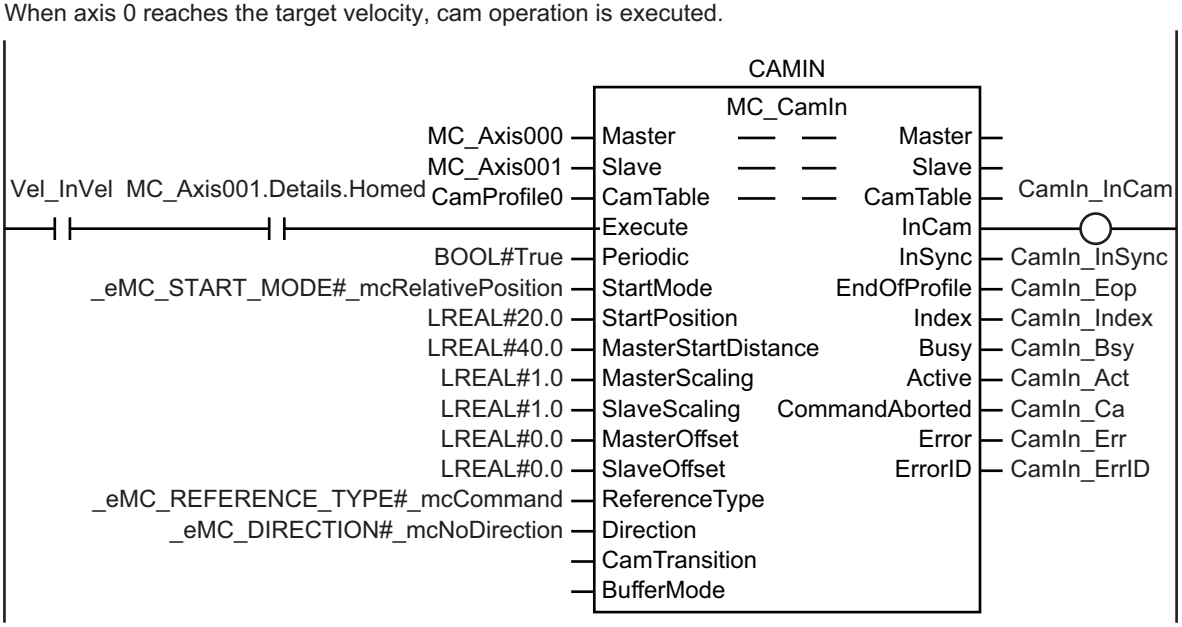
If the Servo is ON for axis 1 and home is not defined, the MC_Home instruction is executed.



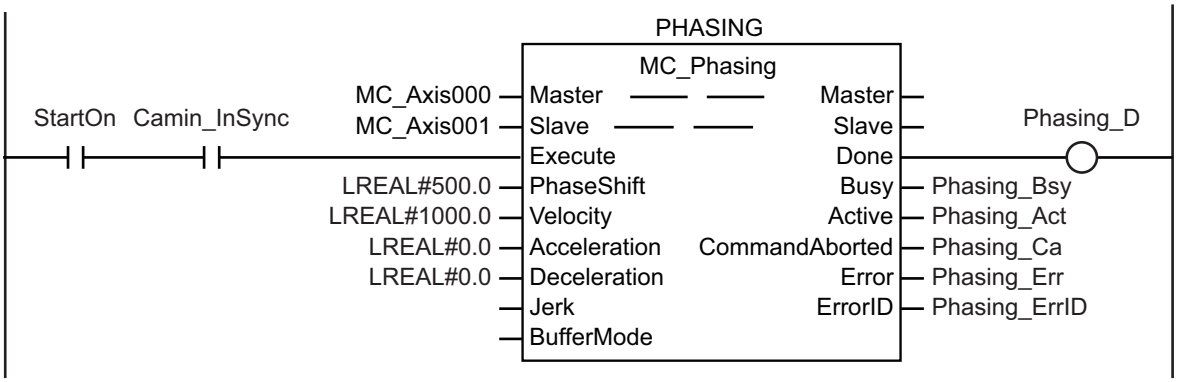
If homing is completed for axis 0, velocity control is executed.



When axis 0 reaches the target velocity, cam operation is executed.



If *StartOn* is TRUE and cam motion is in sync, shifting the phase of the master axis is started.



ST Programming

```
// If the input parameters for the motion instructions are not set, the target valu
es and other parameters are set.
IF InitFlag=FALSE THEN
  // The input parameters for the MC_MoveVelocity (Velocity Control) instruction ar
```

```
e set.
    Vel_Vel := LREAL#1000.0;
    Vel_Acc := LREAL#100000.0;
    Vel_Dec := LREAL#100000.0;
    Vel_Dir := _eMC_DIRECTION#_mcPositiveDirection;
  // The input parameters for the MC_Phasing (Shift Master Axis Phase) instruction
are set.
    Phasing_Ps := LREAL#500.0;
    Phasing_Vel := LREAL#1000.0;
    Phasing_Acc := LREAL#0.0;
    Phasing_Dec := LREAL#0.0;
  // The input parameters for the MC_CamIn (Start Cam Operation) instruction are se
t.
    Camin_Em := TRUE;
    Camin_Sm := _eMC_START_MODE#_mcRelativePosition;
    Camin_Sp := LREAL#20.0;
    Camin_Msd := LREAL#40.0;
    Camin_Ms := LREAL#1.0;
    Camin_Ss := LREAL#1.0;
    Camin_Mo := LREAL#0.0;
    Camin_So := LREAL#0.0;
    Camin_Rt := _eMC_REFERENCE_TYPE#_mcCommand;
    Camin_Dir := _eMC_DIRECTION#_mcNoDirection;
  // The Input Parameter Initialization Completed Flag is changed to TRUE.
    InitFlag := TRUE;
END_IF;

// If the Servo Drive is ready when StartPg is TRUE, turn ON the Servo for axis 0.
IF (StartPg=TRUE)
  AND (MC_Axis000.DrvStatus.Ready=TRUE) THEN
    Pwr1_En:=TRUE;
  ELSE
    Pwr1_En:=FALSE;
END_IF;

// If the Servo Drive is ready when StartPg is TRUE, turn ON the Servo for axis 1.
IF (StartPg=TRUE)
  AND (MC_Axis001.DrvStatus.Ready=TRUE) THEN
    Pwr2_En:=TRUE;
  ELSE
    Pwr2_En:=FALSE;
END_IF;

// If a minor fault level error occurs for axis 0 or axis 1, the error handler for
the device (FaultHandler) is executed.
// Program the FaultHandler according to the device.
IF (MC_Axis000.MFaultLvl.Active=TRUE) OR (MC_Axis001.MFaultLvl.Active=TRUE) THEN
```

```
  FaultHandler();
END_IF;

// If the Servo is ON for axis 0 and home is not defined, the MC_Home instruction i
s executed for axis 0.
IF (Pwr1_Status=TRUE) AND (MC_Axis000.Details.Homed=FALSE) THEN
  Hm1_Ex:=TRUE;
END_IF;

// If the Servo is ON for axis 1 and home is not defined, the MC_Home instruction i
s executed for axis 1.
IF (Pwr2_Status=TRUE) AND (MC_Axis001.Details.Homed=FALSE) THEN
  Hm2_Ex:=TRUE;
END_IF;

// If homing is completed for axis 0, velocity control is executed.
IF Hm1_D=TRUE THEN
  Vel_Ex :=TRUE;
END_IF;

// When axis 0 reaches the target velocity and the home is defined for axis 1, cam
operation is executed.
IF (Vel_InVel=TRUE) AND (MC_Axis001.Details.Homed=TRUE) THEN
  Camin_Ex := TRUE;
END_IF;

// If StartOn is TRUE and cam motion is in sync, shifting the phase of the master a
xis is started.
IF (StartOn=TRUE) AND (Camin_InSync=TRUE) THEN
  Phasing_Ex :=TRUE;
END_IF;

// MC_Power for axis 0
PWR1(
  Axis := MC_Axis000,
  Enable := Pwr1_En,
  Status => Pwr1_Status,
  Busy => Pwr1_Bsy,
  Error => Pwr1_Err,
  ErrorID => Pwr1_ErrID
);

// MC_Power for axis 1
PWR2(
  Axis := MC_Axis001,
  Enable := Pwr2_En,
  Status => Pwr2_Status,
```

```
  Busy => Pwr2_Bsy,
  Error => Pwr2_Err,
  ErrorID => Pwr2_ErrID
);

// MC_Home for axis 0
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 1
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_Phasing
PHASING(
Master := MC_Axis000,
Slave := MC_Axis001,
```

```
Execute := Phasing_Ex,
PhaseShift := Phasing_Ps,
Velocity := Phasing_Vel,
Acceleration := Phasing_Acc,
Deceleration := Phasing_Dec,
Done => Phasing_D,
Busy => Phasing_Bsy,
Active => Phasing_Act,
CommandAborted => Phasing_Ca,
Error => Phasing_Err,
ErrorID => Phasing_ErrID
);

//MC_CamIn
CAMIN(
Master := MC_Axis000,
Slave := MC_Axis001,
CamTable := CamProfile0,
Execute := Camin_Ex,
Periodic := Camin_Em,
StartMode := Camin_Sm,
StartPosition := Camin_Sp,
MasterStartDistance := Camin_Msd,
MasterScaling := Camin_Ms,
SlaveScaling := Camin_Ss,
MasterOffset := Camin_Mo,
SlaveOffset := Camin_So,
ReferenceType := Camin_Rt,
Direction := Camin_Dir,
InCam => Camin_InCam,
InSync => Camin_InSync,
EndOfProfile => Camin_Eop,
Index => Camin_Index,
Busy => Camin_Bsy,
Active => Camin_Act,
CommandAborted => Camin_Ca,
Error => Camin_Err,
ErrorID => Camin_ErrID
);
```

10-2-15 Changing the Actual Position during Velocity Control

This sample changes the absolute values of the command current position and the actual current position for an axis in velocity control.

Precautions for Correct Use

- When you use the MC_SetPosition instruction for an axis in motion, the travel distance between execution of the instruction and changing the actual position will remain as error.
- Home will become undefined when the MC_Set Position instruction is executed.

Axis Parameter Settings

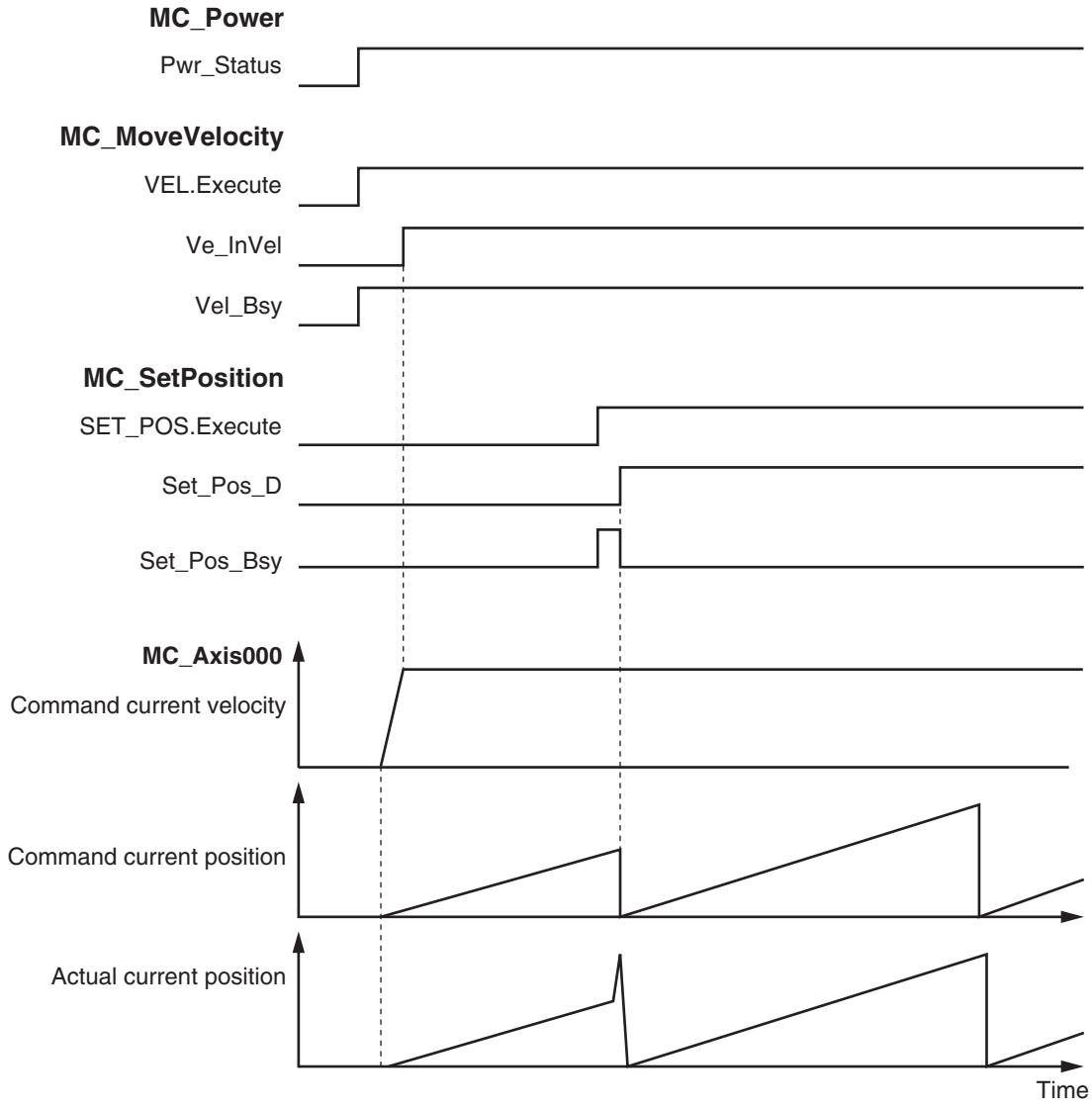
Parameter name	Setting	Description
Count Mode	Rotary Mode	Rotary Mode is set as the count mode for the position.
Modulo Maximum Position Setting Value	360	The Modulo Maximum Position is set to *360*.
Modulo Minimum Position Setting Value	0	The Modulo Minimum Position is set to *0*.
Homing Method	Zero position preset	A zero position preset is performed to define home.

Main Variables Used in the Programming Samples

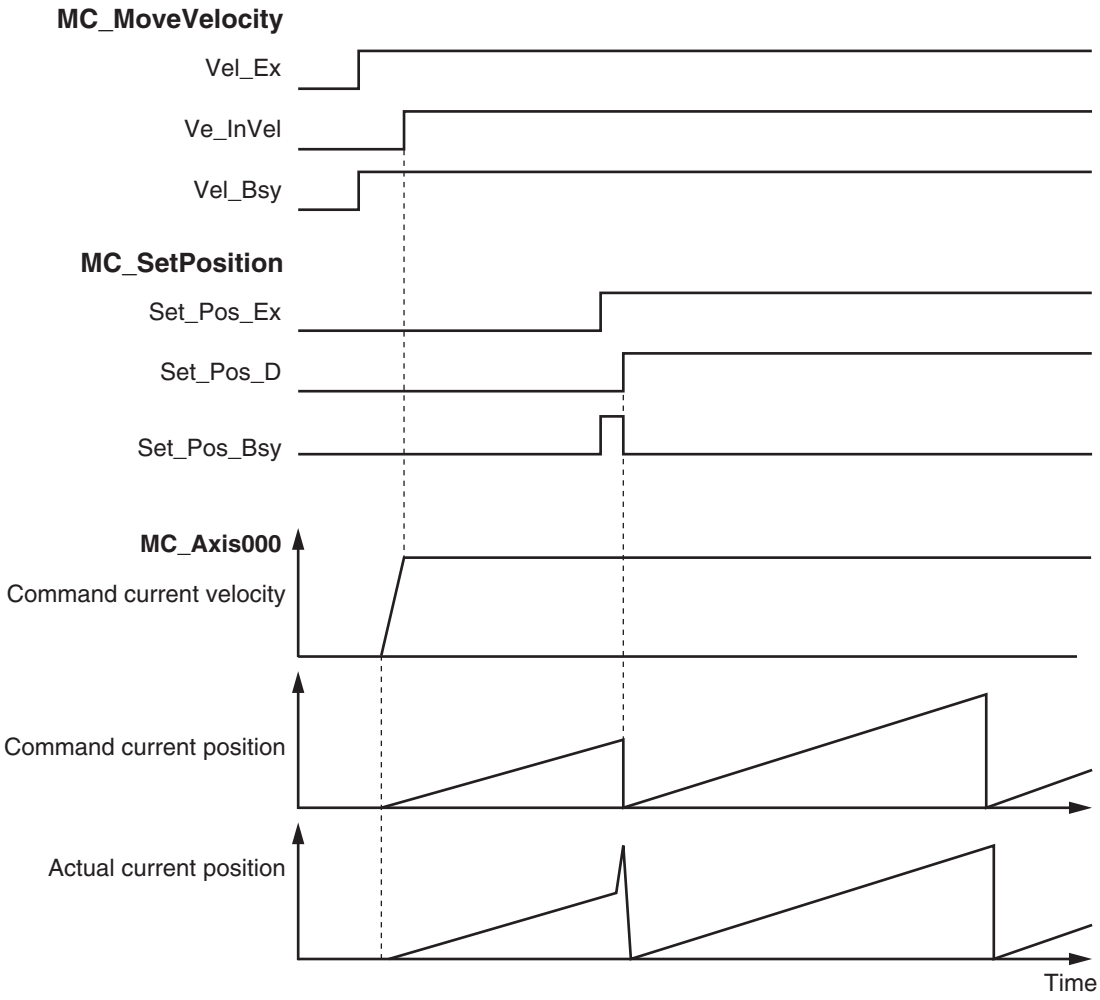
Variable name	Data type	Default	Comment
MC_Axis000	_sAXIS_REF	---	This is the Axis Variable for axis 0.
MC_Axis000.MFaultLvl.Active	BOOL	FALSE	TRUE when there is a minor fault level error for axis 0.
Pwr_Status	BOOL	FALSE	This variable is assigned to the Status output variable from the PWR instance of the MC_Power instruction. It is TRUE when the Servo is ON.
StartSetPos	BOOL	FALSE	This variable gives the status of the external button that is used to change the actual position.
StartPg	BOOL	FALSE	When this variable is TRUE, the Servo is turned ON if EtherCAT process data communications are active and normal.
Vel_Ex	BOOL	FALSE	This variable is used to execute the MC_MoveVelocity (Velocity Control) instruction. It is used in ST programming.
SetPos_Ex	BOOL	FALSE	This variable is used to execute the MC_SetPosition instruction. It is used in ST programming.

Timing Chart

● Ladder Diagram

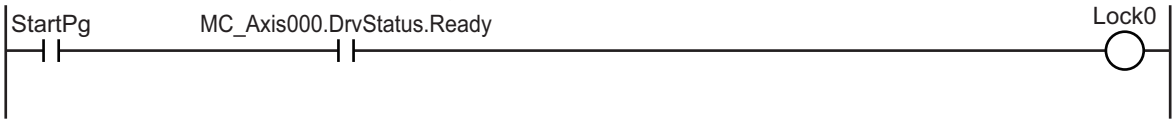


● **ST Programming**

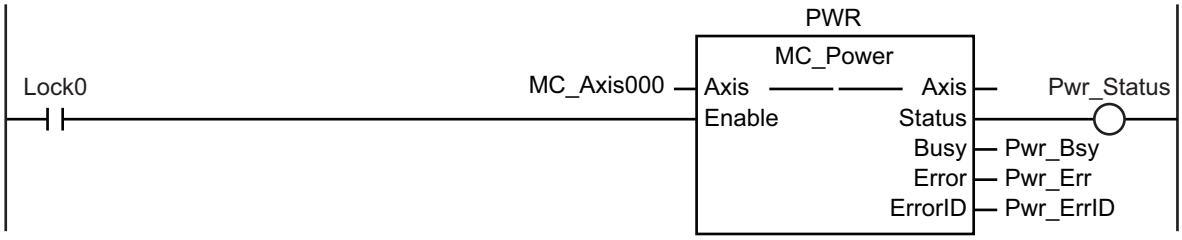


Ladder Diagram

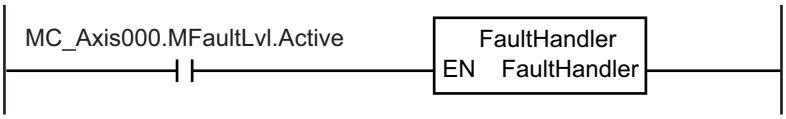
Check if the Servo Drive for axis 0 is ready when *StartPg* is TRUE.



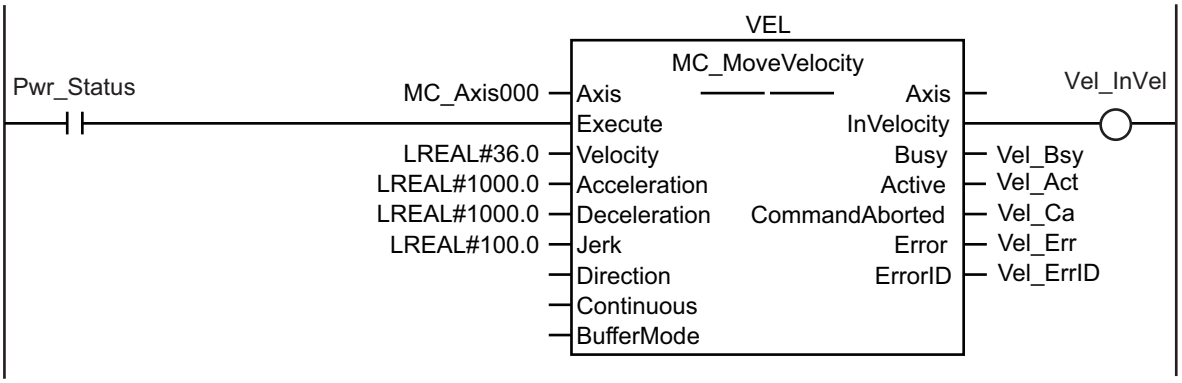
If the Servo Drive for axis 0 is ready, turn ON the Servo for axis 0.



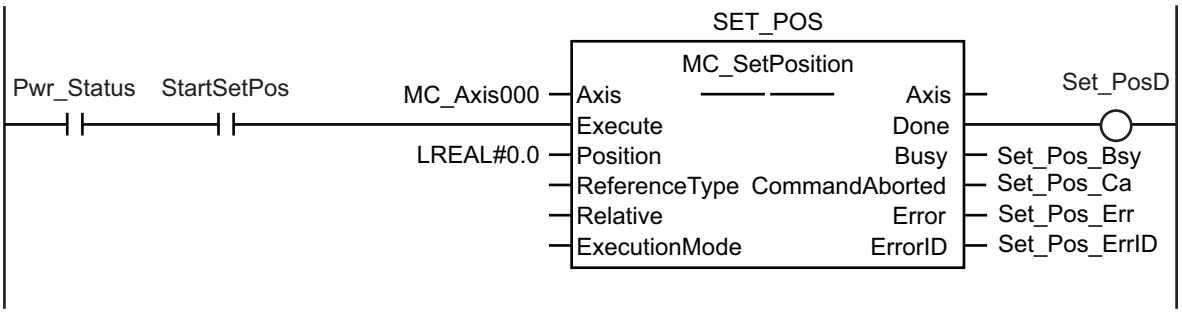
If a minor fault level error occurs for axis 0, the error handler for the device (FaultHandler) is executed. Program the FaultHandler according to the device.



If the Servo is ON for axis 0, velocity control is executed.



If *StartSetPos* is TRUE while the Servo is ON, the Set Position instruction is executed.



ST Programming

```
// If the input parameters for the instructions are not set, the target values and
other parameters are set.
IF InitFlag=FALSE THEN
  // The input parameters for the MC_MoveVelocity (Velocity Control) instruction ar
e set.
    Vel_Vel := LREAL#36.0;
    Vel_Acc := LREAL#1000.0;
    Vel_Dec := LREAL#1000.0;
    Vel_Jrk := LREAL#100.0;
  // The input parameters for the MC_SetPosition instruction are set.
    Set_Pos_Pos := LREAL#0.0;
  // The Input Parameter Initialization Completed Flag is changed to TRUE.
    InitFlag := TRUE;
END_IF;

// If the Servo Drive is ready when StartPg is TRUE, turn ON the Servo for axis 0.
// If the Servo Drive is not ready, turn OFF the Servo.
```

```
IF (StartPg=TRUE)
  AND (MC_Axis000.DrvStatus.Ready=TRUE) THEN
    Pwr_En:=TRUE;
  ELSE
    Pwr_En:=FALSE;
END_IF;

// If a minor fault level error occurs for axis 0, the error handler for the device
 (FaultHandler) is executed.
// Program the FaultHandler according to the device.
IF MC_Axis000.MFaultLvl.Active=TRUE THEN
  FaultHandler();
END_IF;

// If the Servo is ON for axis 0, velocity control is executed for axis 0.
IF Pwr_Status=TRUE THEN
  Vel_Ex :=TRUE;
END_IF;

// If StartSetPos is TRUE while the Servo is ON, the Set Position instruction is ex
ecuted.
IF (Pwr_Status=TRUE) AND (StartSetPos=TRUE) THEN
  Set_Pos_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_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_SetPosition
SET_POS(
  Axis := MC_Axis000,
  Execute := Set_Pos_Ex,
  Position := Set_Pos_Pos,
  Done => Set_Pos_D,
  Busy => Set_Pos_Bsy,
  CommandAborted => Set_Pos_Ca,
  Error => Set_Pos_Err,
  ErrorID => Set_Pos_ErrID
);
```

10-2-16 Changing a Cam Data Variable and Saving the Cam Table

This sample uses the user program to change a cam data variable that was created on Cam Editor of the Sysmac Studio.
The displacements for phases of 0° to 180° are multiplied by 2 and the displacements for phases of 181° to 360° are multiplied by 0.5.

If the changes to the cam data are completed, the motion control instruction MC_SaveCamTable is used to save the cam data variable to non-volatile memory in the CPU Unit.
When saving the data is completed, the MC_CamIn (Start Cam Operation) instruction is executed to start cam motion.

Precautions for Correct Use

- If the phases are not in ascending order, an error occurs when the MC_CamIn (Start Cam Operation) instruction is executed.
  The order of the phases are not checked in this sample. To check the order of the phases, execute the MC_SetCamTableProperty or MC_SetCamTableEndPointIndex instruction.
- There is a limit to the number of times that you can write non-volatile memory in the CPU Unit.
  Save cam table data only when necessary.
- If the power supply to the CPU Unit is turned OFF before the data is saved with the MC_SaveCamTable instruction, the cam data variable will revert to the contents from before it was changed by the user program.

Main Variables Used in the Programming Samples

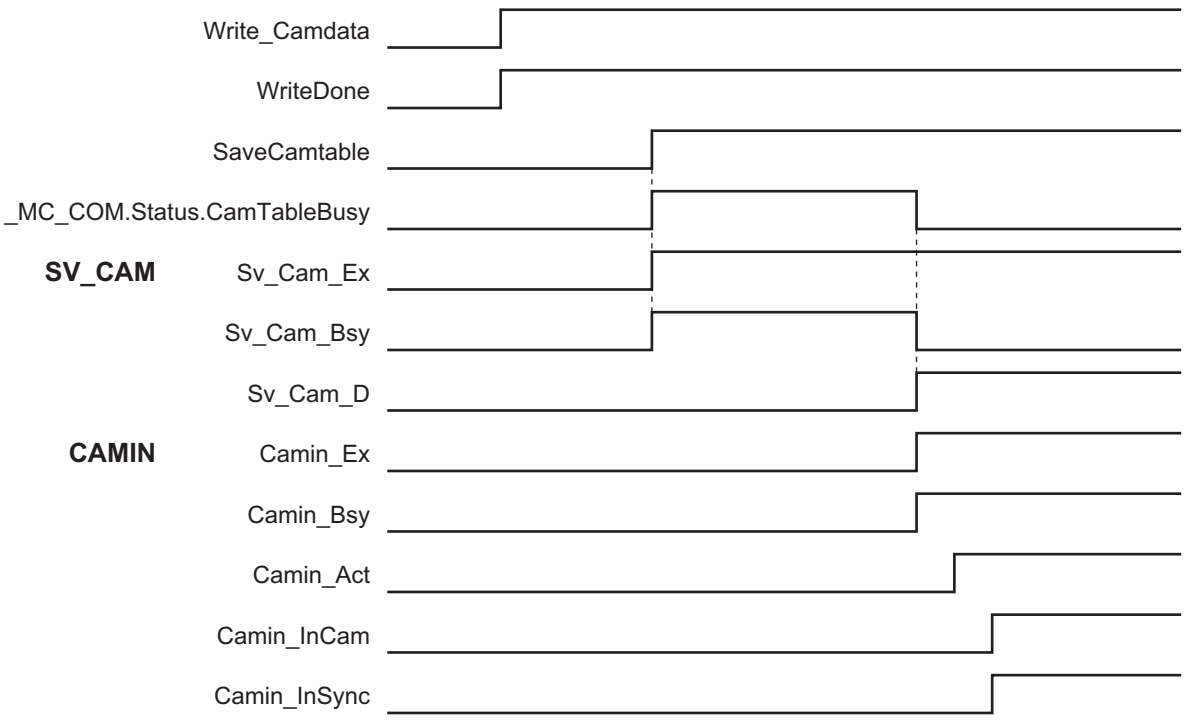
Variable name	Data type	Default	Comment
MC_Axis000	_sAXIS_REF	---	This is the Axis Variable for axis 0.
MC_Axis001	_sAXIS_REF	---	This is the Axis Variable for axis 1.
CamProfile0	ARRAY[0..360] OF _sMC_CAM_REF	---	This is the cam data variable. *1

Variable name	Data type	Default	Comment
Pwr1_Status	BOOL	FALSE	This variable is assigned to the Status output variable from the PWR1 instance of the MC_Power instruction. It is TRUE when the Servo is ON.
Pwr2_Status	BOOL	FALSE	This variable is assigned to the Status output variable from the PWR2 instance of the MC_Power instruction. It is TRUE when the Servo is ON.
StartPg	BOOL	FALSE	When this variable is TRUE, the Servo is turned ON if EtherCAT process data communications are active and normal.
WriteCamdata	BOOL	FALSE	This variable is used to start changing the cam data. It is changed to TRUE to start editing.
WriteDone	BOOL	FALSE	This variable is used to indicate that the changes to the cam data are completed. It is changed to TRUE when the changes to the cam data are completed.
SaveCamtable	BOOL	FALSE	This variable is used to execute the Save Cam Table instruction.
_MC_COM.Status.CamTableBusy	BOOL	FALSE	This system-defined variable is TRUE while cam table data is being saved.
Sv_Cam_Ex	BOOL	FALSE	This variable is used to execute the MC_SaveCamTable instruction.
Camin_Ex	BOOL	FALSE	This variable is used to execute the MC_CamIn (Start Cam Operation) instruction. It is used in ST programming.

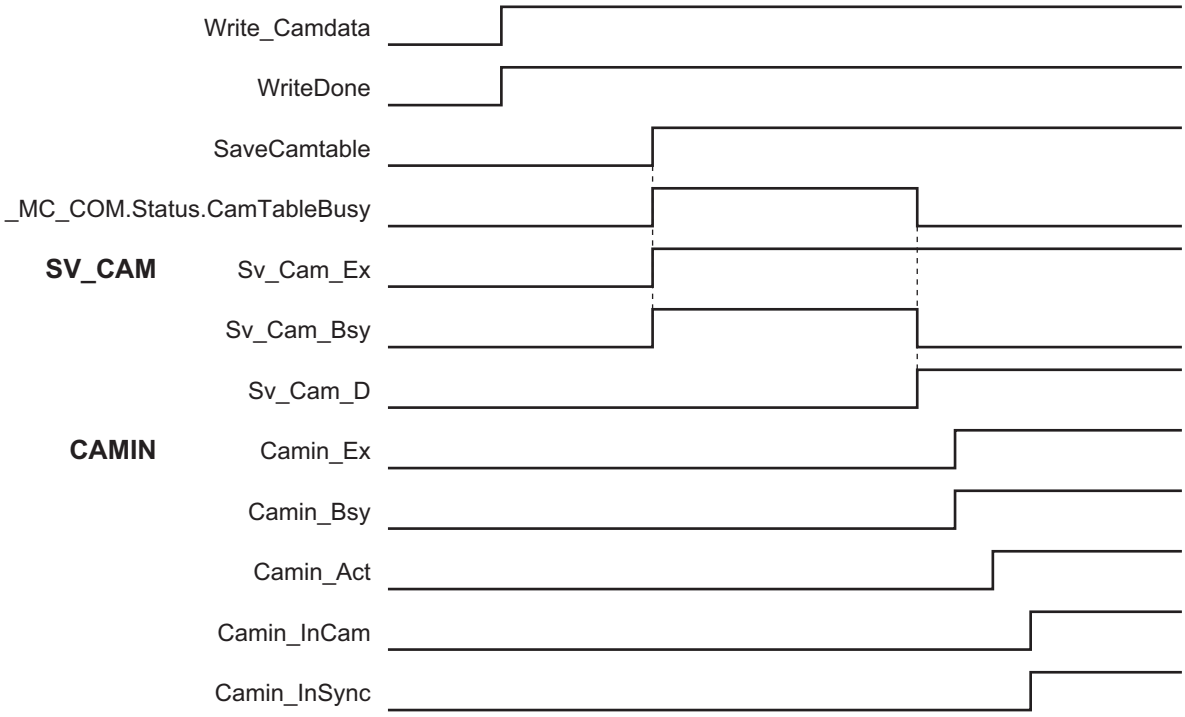
*1. The array elements ARRAY[0..N] are set with the Cam Editor in the Sysmac Studio. The range of the array is 0 to 360 in this sample.

Timing Chart

● Ladder Diagram



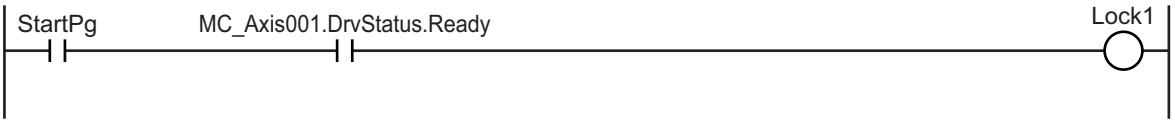
● ST Programming



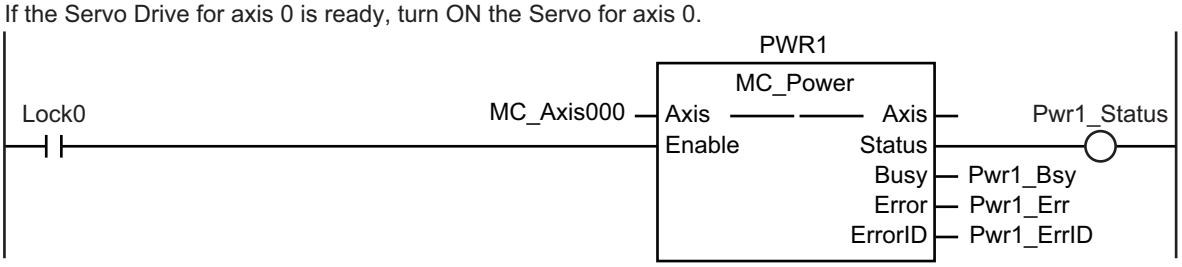
Ladder Diagram

Check if the Servo Drive for axis 0 is ready when *StartPg* is TRUE.

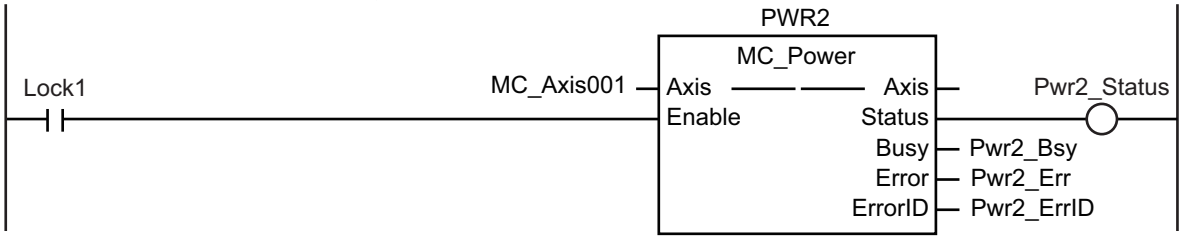
Check if the Servo Drive for axis 1 is ready when *StartPg* is TRUE.



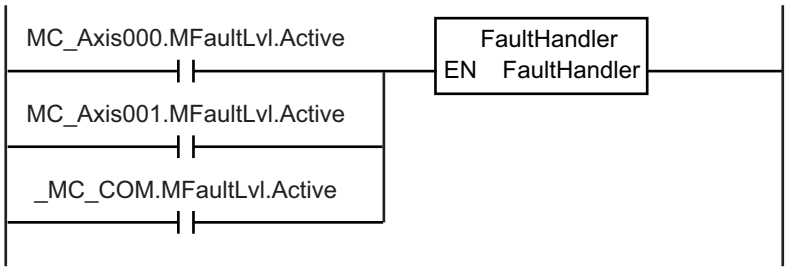
If the Servo Drive for axis 0 is ready, turn ON the Servo for axis 0.



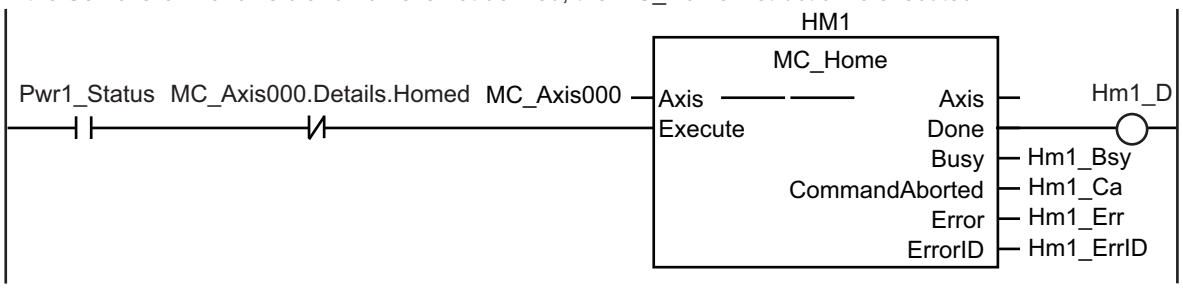
If the Servo Drive for axis 1 is ready, turn ON the Servo for axis 1.



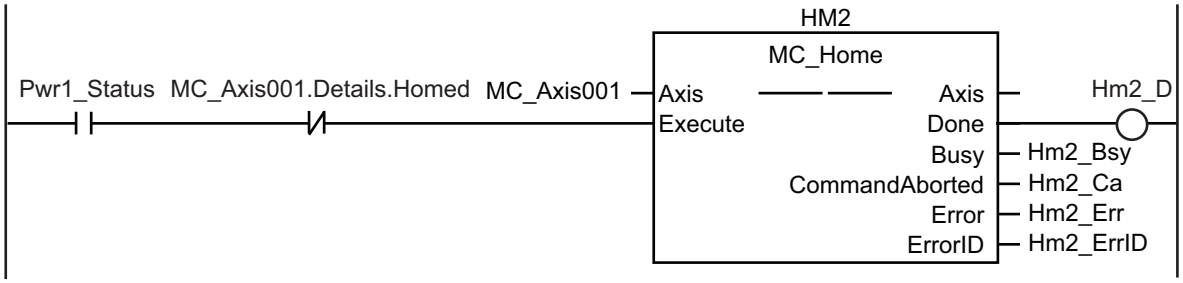
If a minor fault level error occurs in the MC Common Error Status variable or for any of the axes, the error handler for the device (FaultHandler) is executed. Program the FaultHandler according to the device.



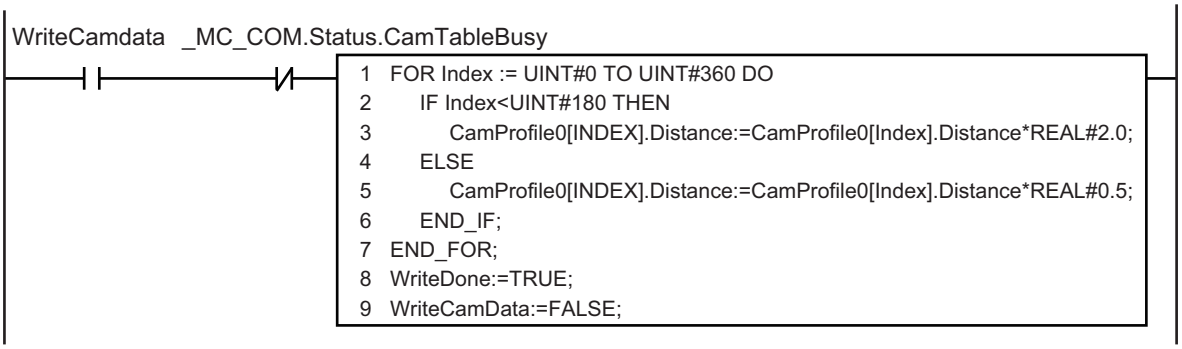
If the Servo is ON for axis 0 and home is not defined, the MC_Home instruction is executed.



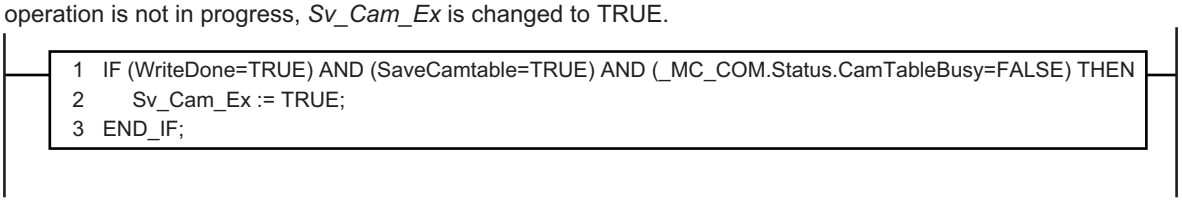
If the Servo is ON for axis 1 and home is not defined, the MC_Home instruction is executed.



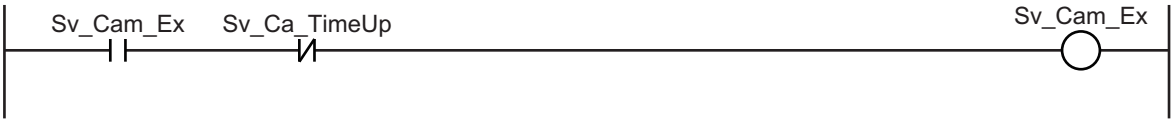
If *WriteCamData* is TRUE and a cam table file is not being saved, the values in the cam data variable are changed.
The displacements for phases from 0° to 180° are multiplied by 2 and the displacements for phases from 181° to 360° are multiplied by 0.5. When the changes to the displacements are completed, *WriteDone* is changed to TRUE.



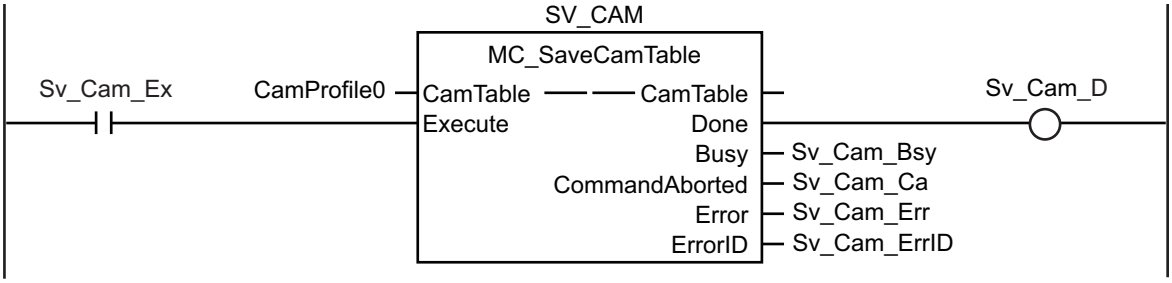
If the changes to the cam data variable are completed, *SaveCamtable* is TRUE, and a cam table file save operation is not in progress, *Sv_Cam_Ex* is changed to TRUE.



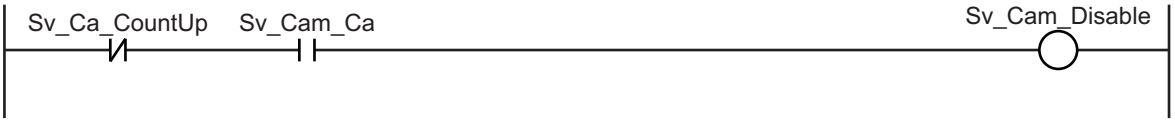
If *Sv_Ca_TimeUp* is TRUE, *Sv_Cam_Ex* is changed to FALSE.
If *Sv_Cam_Ex* changes to FALSE, *Sv_Ca_TimeUp* changes to FALSE and *Sv_Cam_Ex* changes to TRUE.
The MC_SaveCamTable instruction is executed again.



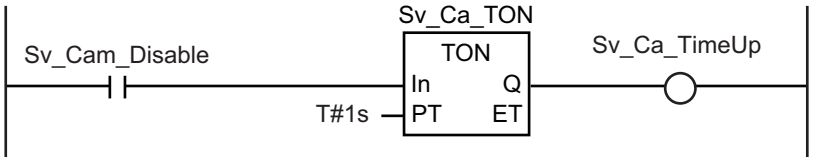
If *Sv_Cam_Ex* changes to TRUE, the MC_SaveCamTable instruction is executed.



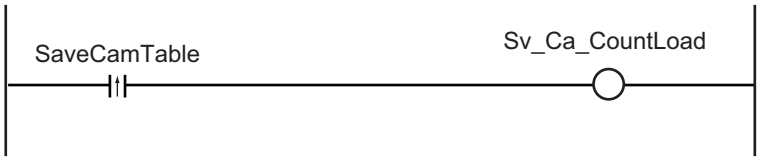
If Sv_Ca_CountUp is FALSE, a Cannot Execute Save Cam Table error occurs and *Sv_Cam_Disable* is changed to TRUE.



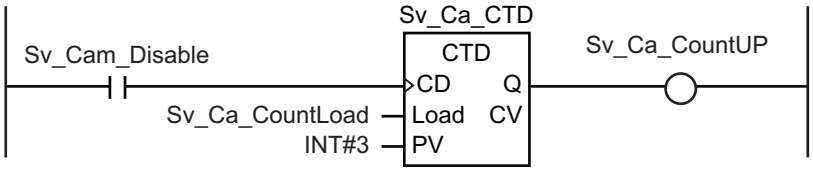
One second after a Cannot Execute Save Cam Table error occurs, *Sv_Ca_TimeUp* is changed to TRUE.
When *Sv_Ca_TimeUp* changes to TRUE, *Sv_Cam_Ex* changes to FALSE.



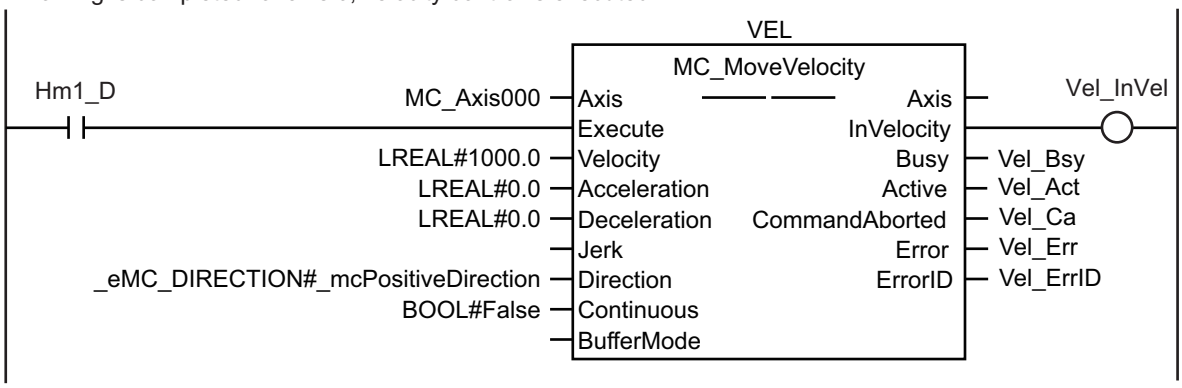
Sv_Ca_CountLoad changes to TRUE for one period when the cam table is saved.
If *Sv_Ca_CountLoad* is TRUE, the retry counter is reset.



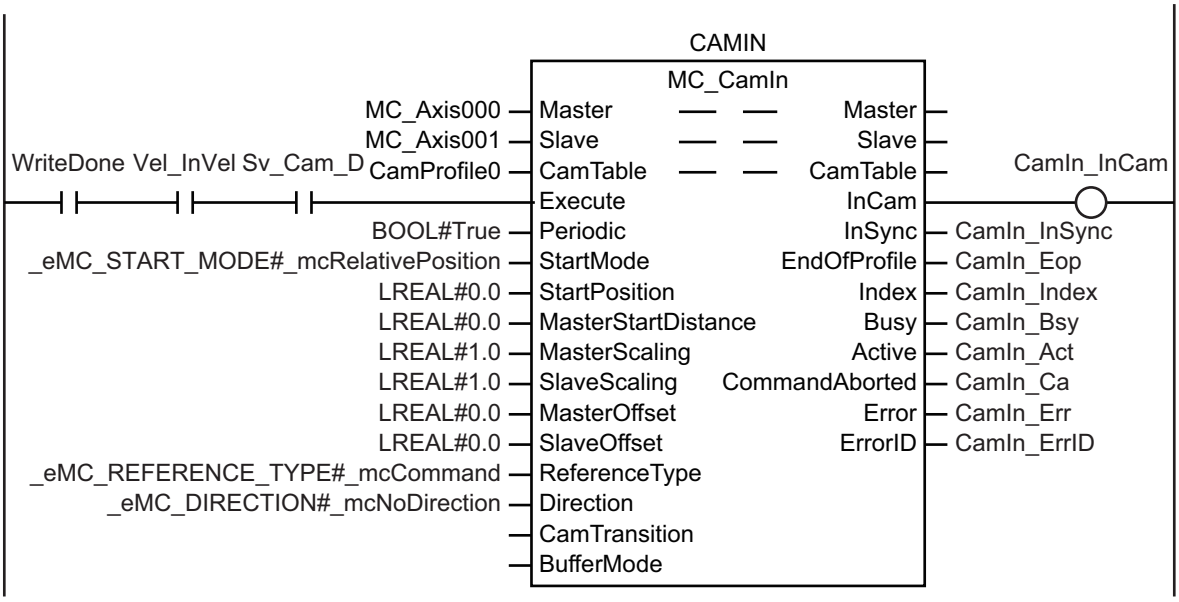
If a Cannot Execute Save Cam Table error occurs three times, *Sv_Ca_CountUp* is changed to TRUE.
When *Sv_Ca_CountUp* changes to TRUE, *Sv_Cam_Disable* is changed to FALSE.
Retry processing for the MC_SaveCamTable instruction is completed.



If homing is completed for axis 0, velocity control is executed.



If the changes to the cam data variable and saving the cam table are completed and axis 0 is at the target velocity, the cam operation is executed.



ST Programming

```
// If the input parameters for the instructions are not set, the target values and
other parameters are set.
IF InitFlag=FALSE THEN
  // The input parameters for the MC_MoveVelocity (Velocity Control) instruction ar
e set.
    Vel_Vel := LREAL#1000.0;
    Vel_Acc := LREAL#0.0;
    Vel_Dec := LREAL#0.0;
    Vel_Dir := _eMC_DIRECTION#_mcPositiveDirection;
  // The input parameters for the MC_CamIn (Start Cam Operation) instruction are se
t.
    Camin_Em := TRUE;
    Camin_Sm := _eMC_START_MODE#_mcRelativePosition;
    Camin_Sp := LREAL#0.0;
    Camin_Msd := LREAL#0.0;
    Camin_Ms := LREAL#1.0;
    Camin_Ss := LREAL#1.0;
    Camin_Mo := LREAL#0.0;
    Camin_So := LREAL#0.0;
    Camin_Rt := _eMC_REFERENCE_TYPE#_mcCommand;
    Camin_Dir := _eMC_DIRECTION#_mcNoDirection;
  // The Input Parameter Initialization Completed Flag is changed to TRUE.
    InitFlag := TRUE;
END_IF;

// If the Servo Drive is ready when StartPg is TRUE, turn ON the Servo for axis 0.
```

```
// If the Servo Drive is not ready, turn OFF the Servo.
IF (StartPg=TRUE)
  AND (MC_Axis000.DrvStatus.Ready=TRUE) THEN
    Pwr1_En :=TRUE;
  ELSE
    Pwr1_En :=FALSE;
END_IF;

// If the Servo Drive is ready when StartPg is TRUE, turn ON the Servo for axis 1.
// If the Servo Drive is not ready, turn OFF the Servo.
IF (StartPg=TRUE)
  AND (MC_Axis001.DrvStatus.Ready=TRUE) THEN
    Pwr2_En :=TRUE;
  ELSE
    Pwr2_En :=FALSE;
END_IF;

// If a minor fault level error occurs in the MC Common Error Status variable or fo
r any of the axes, the error handler for the device (FaultHandler) is executed.
// Program the FaultHandler according to the device.
IF (MC_Axis000.MFaultLvl.Active=TRUE)
  OR (MC_Axis001.MFaultLvl.Active=TRUE)
  OR (_MC_COM.MFaultLvl.Active=TRUE) THEN
    FaultHandler();
END_IF;

// If the Servo is ON for axis 0 and home is not defined, the MC_Home instruction i
s executed for axis 0.
IF (Pwr1_Status=TRUE) AND (MC_Axis000.Details.Homed=FALSE) THEN
  Hm1_Ex :=TRUE;
END_IF;

// If the Servo is ON for axis 1 and home is not defined, the MC_Home instruction i
s executed for axis 1.
IF (Pwr2_Status=TRUE) AND (MC_Axis001.Details.Homed=FALSE) THEN
  Hm2_Ex :=TRUE;
END_IF;

// If WriteCamData is TRUE and a cam table file is not being saved, the values in t
he cam data variable are changed.
// The displacements for phases of 0° to 180° are multiplied by 2 and the displacem
ents for phases of 181° to 360° are multiplied by 0.5.
// When the changes are completed, WriteDone is changed to TRUE.
IF (WriteCamdata=TRUE) AND (_MC_COM.Status.CamTableBusy=FALSE) THEN
  FOR Index := UINT#0 TO UINT#360 DO
    IF Index<UINT#180 THEN
        CamProfile0[Index].Distance:=CamProfile0[Index].Distance*REAL#2.0;
```

```
      ELSE
        CamProfile0[Index].Distance:=CamProfile0[Index].Distance*REAL#0.5;
    END_IF;
  END_FOR;
  WriteDone :=TRUE;
  WriteCamdata :=FALSE;
END_IF;

// If homing is completed for axis 0, velocity control is executed.
IF Hm1_D=TRUE THEN
  Vel_Ex :=TRUE;
END_IF;

// If the changes to the cam data variable are completed, SaveCamtable is TRUE, and
 a cam table file save operation is not in progress, Sv_Cam_Ex is changed to TRUE.
//  If Sv_Cam_Ex is TRUE, the MC_SaveCamTable instruction is executed.
IF (WriteDone=TRUE)
  AND (SaveCamtable=TRUE)
  AND (_MC_COM.Status.CamTableBusy=FALSE) THEN
    Sv_Cam_Ex := TRUE;
END_IF;

// If Sv_Ca_TimeUp is TRUE, Sv_Cam_Ex is changed to FALSE.
// If Sv_Cam_Ex is FALSE, Sv_Ca_TimeUp changes to FALSE and Sv_Cam_Ex changes to TR
UE.
// The MC_SaveCamTable instruction is executed again.
IF (Sv_Cam_Ex=TRUE) AND (Sv_Ca_TimeUp=FALSE) THEN
    Sv_Cam_Ex := TRUE;
  ELSE
    Sv_Cam_Ex := FALSE;
END_IF;

// If Sv_Ca_CountUp is FALSE and a Cannot Execute Save Cam Table error occurs, Sv_C
am_Disable is changed to TRUE.
IF (Sv_Ca_CountUP=FALSE) AND (Sv_Cam_Ca=TRUE) THEN
    Sv_Cam_Disable := TRUE;
  ELSE
    Sv_Cam_Disable := FALSE;
END_IF;

// One second after the Cannot Execute Save Cam Table error occurs, Sv_Ca_TimeUp is
 changed to TRUE.
// If Sv_Ca_TimeUp changes to TRUE, Sv_Cam_Ex is changed to FALSE.
Sv_Ca_TON(
  In := Sv_Cam_Disable,
  PT := T#1s,
  Q => Sv_Ca_TimeUp
```

```
);

// Sv_Ca_CountLoad is changed to TRUE for one period when the cam table is saved.
// If Sv_Ca_CountLoad changes to TRUE, the retry counter is reset.
R_TRIG1(SaveCamtable, Sv_Ca_CountLoad);

// If a Cannot Execute Save Cam Table error occurs three times, Sv_Ca_CountUP is ch
anged to TRUE.
// If Sv_Ca_CountUP changes to TRUE, Sv_Cam_Disable is changed to FALSE.
// Retry processing for the MC_SaveCamTable instruction is completed.
Sv_Ca_CTD(
  CD := Sv_Cam_Disable,
  LOAD := Sv_Ca_CountLoad,
  PV := INT#3,
  Q => Sv_Ca_CountUP
);

// If the changes to the cam data variable and saving the cam table are completed a
nd axis 0 is at the target velocity, the cam operation is executed.
IF (Vel_InVel=TRUE)
  AND (WriteDone=TRUE)
  AND (Sv_Cam_D=TRUE) THEN
    Camin_Ex :=TRUE;
END_IF;

//MC_SaveCamTable
SV_CAM(
  CamTable :=CamProfile0,
  Execute := Sv_Cam_Ex,
  Done => Sv_Cam_D,
  Busy => Sv_Cam_Bsy,
  CommandAborted => Sv_Cam_Ca,
  Error => Sv_Cam_Err,
  ErrorID => Sv_Cam_ErrID
);

CAMIN(
  Master := MC_Axis000,
  Slave := MC_Axis001,
  CamTable := CamProfile0,
  Execute := Camin_Ex,
  Periodic := Camin_Em,
  StartMode := Camin_Sm,
  StartPosition := Camin_Sp,
  MasterStartDistance := Camin_Msd,
  MasterScaling := Camin_Ms,
  SlaveScaling := Camin_Ss,
```

```
  MasterOffset := Camin_Mo,
  SlaveOffset := Camin_So,
  ReferenceType := Camin_Rt,
  Direction := Camin_Dir,
  InCam => Camin_InCam,
  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_Power for axis 0
PWR1(
  Axis := MC_Axis000,
  Enable := Pwr1_En,
  Status => Pwr1_Status,
  Busy => Pwr1_Bsy,
  Error => Pwr1_Err,
  ErrorID => Pwr1_ErrID
);

// MC_Power for axis 1
PWR2(
  Axis := MC_Axis001,
  Enable := Pwr2_En,
  Status => Pwr2_Status,
  Busy => Pwr2_Bsy,
  Error => Pwr2_Err,
  ErrorID => Pwr2_ErrID
);

// MC_Home for axis 0
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 1
```

```
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
);
```

10-2-17 Temporarily Changing Axis Parameters

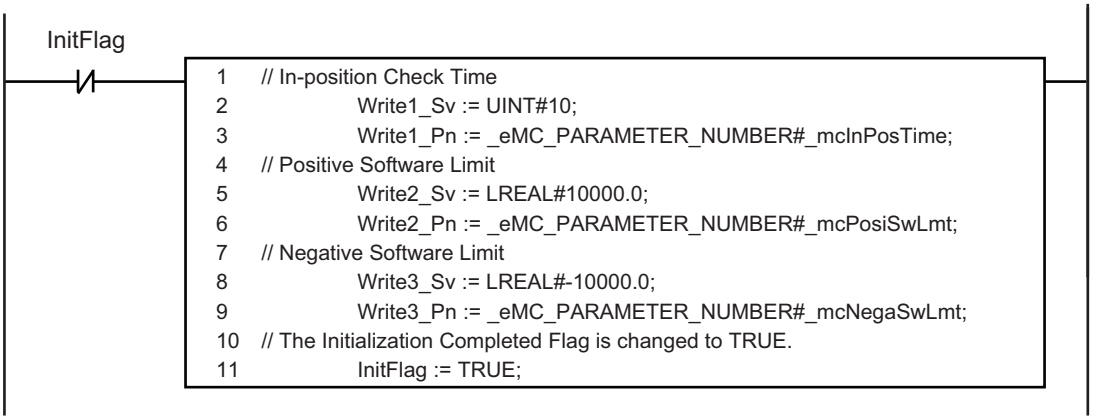
This sample uses the MC_Write (Write MC Setting) instruction to change the settings of the In-Position Check Time, Positive Software Limit, and Negative Software Limit.

Main Variables Used in the Programming Samples

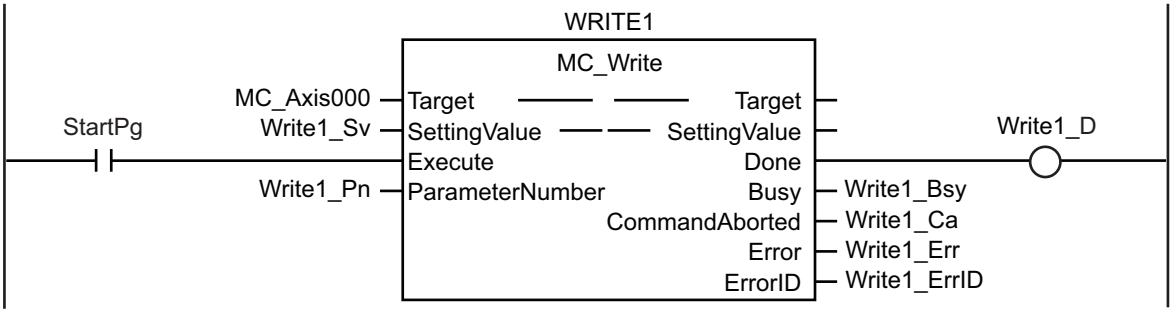
Variable name	Data type	Default	Comment
MC_Axis000	_sAXIS_REF	---	This is the Axis Variable for axis 0.
InitFlag	BOOL	FALSE	This variable indicates the status of parameter settings. FALSE while parameters are changed. TRUE after the changes to the parameters are completed.
StartPg	BOOL	FALSE	This variable is used to execute the MC_Write instruction.

Ladder Diagram

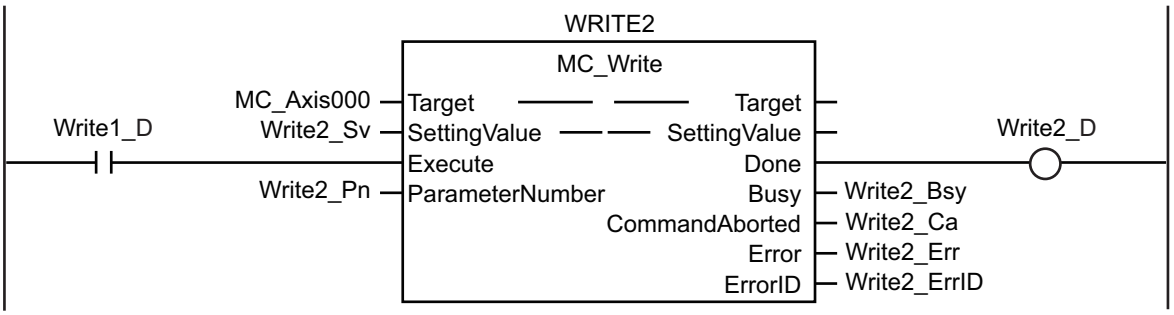
The axis parameters are set. When setting the parameters is completed, *InitFlag* is changed to TRUE.



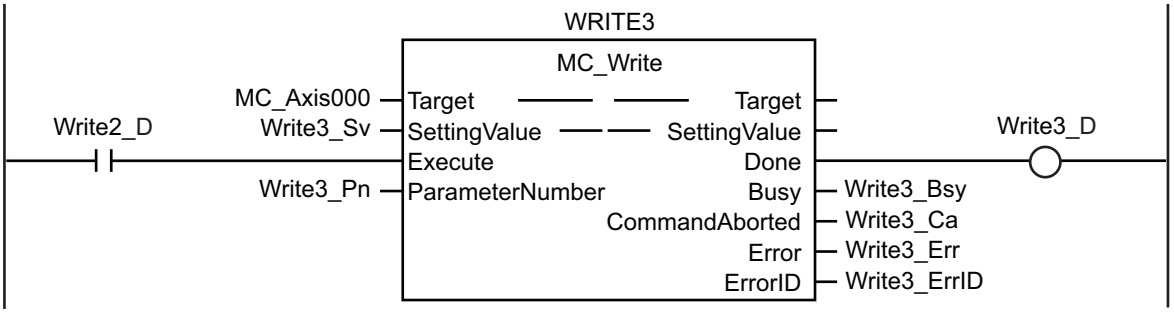
If *StartPg* is TRUE, the setting of the In-position Check Time is changed.



If changing the setting of the In-Position Check Time is completed, the setting of the Positive Software Limit is changed.



If changing the setting of the Positive Software Limit is completed, the setting of the Negative Software Limit is changed.



ST Programming

```
// The axis parameters are set. When setting the parameters is completed, InitFlag
is changed to TRUE.
IF InitFlag=FALSE THEN
  // In-position Check Time
    Write1_Sv := UINT#10;
    Write1_Pn := _eMC_PARAMETER_NUMBER#_mcInPosTime;
  // Positive Software Limit
    Write2_Sv := LREAL#10000.0;
    Write2_Pn := _eMC_PARAMETER_NUMBER#_mcPosiSwLmt;
  // Negative Software Limit
    Write3_Sv := LREAL#-10000.0;
    Write3_Pn := _eMC_PARAMETER_NUMBER#_mcNegaSwLmt;
  // The Input Parameter Initialization Completed Flag is changed to TRUE.
    InitFlag := TRUE;
END_IF;

// If StartPg is TRUE, the setting of the In-position Check Time is changed.
IF StartPg =TRUE THEN
  Write1_Ex := TRUE;
END_IF;

// If changing the setting of the In-Position Check Time is completed, the setting
of the Positive Software Limit is changed.
IF Write1_D = TRUE THEN
  Write2_Ex := TRUE;
END_IF;

// If changing the setting of the Positive Software Limit is completed, the setting
 of the Negative Software Limit is changed.
IF Write2_D = TRUE THEN
Write3_Ex := TRUE;
END_IF;

// MC_Write
WRITE1(
  Target := MC_Axis000,
  SettingValue := Write1_Sv,
  Execute := Write1_Ex,
  ParameterNumber := Write1_Pn,
  Done => Write1_D,
  Busy => Write1_Bsy,
  CommandAborted => Write1_Ca,
  Error => Write1_Err,
  ErrorID => Write1_ErrID
);
```

```
WRITE2(
  Target := MC_Axis000,
  SettingValue := Write2_Sv,
  Execute := Write2_Ex,
  ParameterNumber := Write2_Pn,
  Done => Write2_D,
  Busy => Write2_Bsy,
  CommandAborted => Write2_Ca,
  Error => Write2_Err,
  ErrorID => Write2_ErrID
);

WRITE3(
  Target := MC_Axis000,
  SettingValue := Write3_Sv,
  Execute := Write3_Ex,
  ParameterNumber := Write3_Pn,
  Done => Write3_D,
  Busy => Write3_Bsy,
  CommandAborted => Write3_Ca,
  Error => Write3_Err,
  ErrorID => Write3_ErrID
);
```

10-2-18 Updating the Cam Table End Point Index

This sample increases the valid number of data points by 10 in a cam table with a maximum number of data points of 110 and a valid number of data points of 100. It also updates the end point index.

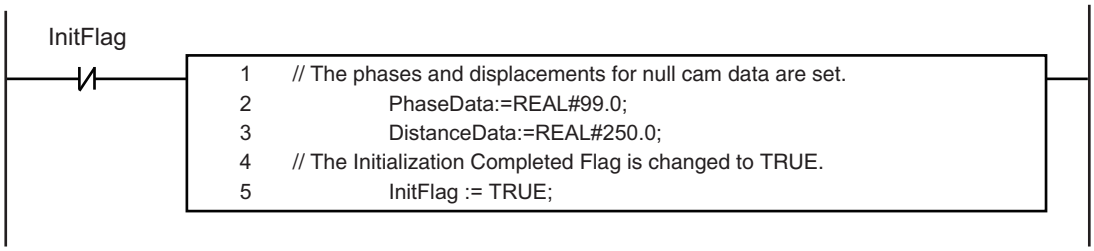
Main Variables Used in the Programming Samples

Variable name	Data type	Default	Comment
CamProfile0	ARRAY[0..109] OF _sMC_CAM_REF	---	This is a cam data variable with a maximum number of data points of 110. *1 It contains 100 valid cam data points and 10 null cam data points.
WriteCamdata	BOOL	FALSE	This variable is used to start changing the cam data. It is changed to TRUE to start editing.
WriteDone	BOOL	FALSE	This variable is used to indicate that the changes to the cam data are completed. It is changed to TRUE when the changes to the cam data are completed.

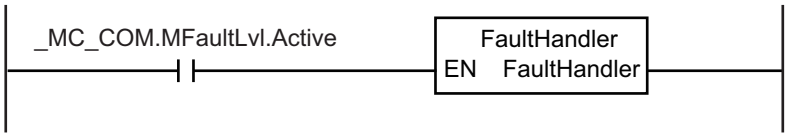
*1. The array elements ARRAY[0..N] are set with the Cam Editor in the Sysmac Studio. The range of the array is 0 to 109 in this sample.

Ladder Diagram

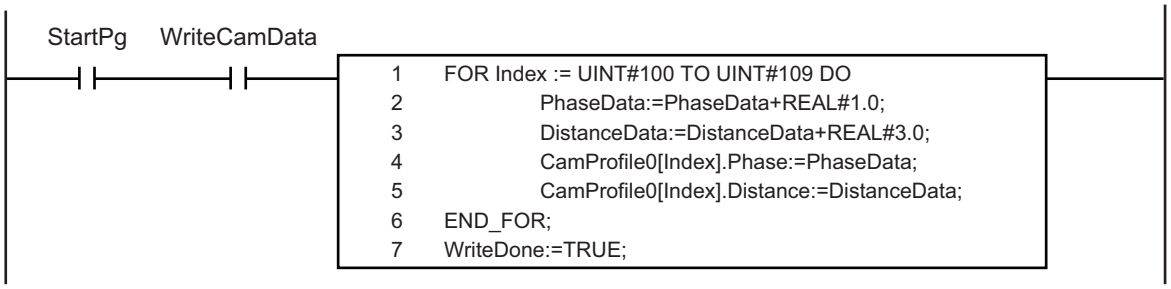
The axis parameters are set. When setting the parameters is completed, *InitFlag* is changed to TRUE.



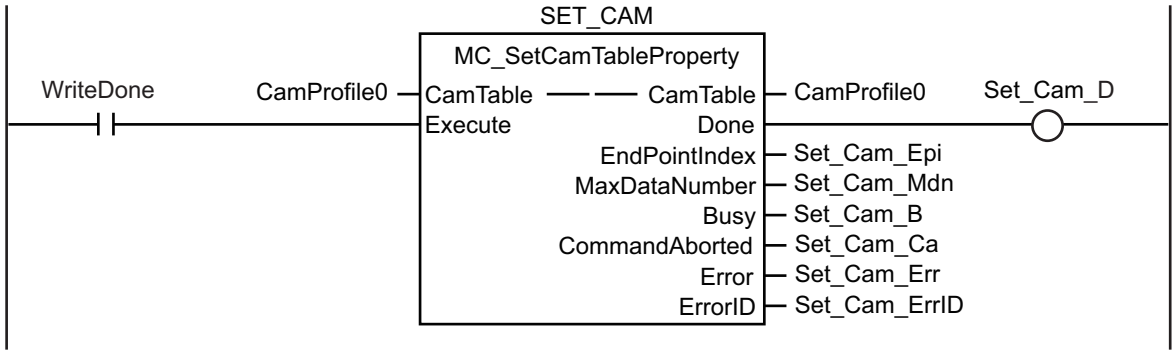
If a minor fault level error occurs in the MC Common Error Status variable, the error handler for the device (FaultHandler) is executed. Program the FaultHandler according to the device.



If *StartPg* and *WriteCamData* are TRUE, the values in the cam data variable are changed.
Phases and displacements are set for *CamProfile[100]* to *CamProfile[109].*
When the changes to the cam data variable are completed, *WriteDone* is changed to TRUE.



If the changes to the cam data variable are completed, the Set Cam Table Properties instruction is executed.



ST Programming

```
// The axis parameters are set. When setting the parameters is completed, InitFlag
is changed to TRUE.
IF InitFlag=FALSE THEN
  // The phases and displacements for null cam data are set.
    PhaseData :=REAL#99.0;
```

```
    DistanceData :=REAL#250.0;
  // The Initialization Completed Flag is changed to TRUE.
    InitFlag := TRUE;
END_IF;

// If a minor fault level error occurs in the MC Common Error Status variable, the
error handler for the device is executed.
// Program the FaultHandler according to the device.
IF _MC_COM.MFaultLvl.Active=TRUE THEN
FaultHandler();
END_IF;

// If StartPg and WriteCamData are TRUE, the values in the cam data variable are ch
anged.
// The phases and displacements are set in CamProfile[100] to CamProfile[109].
// When the changes to the cam data variable are completed, WriteDone is changed to
 TRUE.
IF StartPg=TRUE
  AND WriteCamData=TRUE THEN
    FOR Index := UINT#100 TO UINT#109 DO
      PhaseData :=PhaseData+REAL#1.0;
      DistanceData :=DistanceData+REAL#3.0;
      CamProfile0[Index].Phase :=PhaseData;
      CamProfile0[Index].Distance :=DistanceData;
    END_FOR;
    WriteDone :=TRUE;
END_IF;

// If the changes to the cam data variable are completed, the Set Cam Table Propert
ies instruction is executed.
IF WriteDone=TRUE THEN
  Set_Cam_Ex := TRUE;
END_IF;

//MC_SetCamTableProperty
SET_CAM(
  CamTable := CamProfile0,
  Execute := Set_Cam_Ex,
  Done => Set_Cam_D,
  EndPointIndex => Set_Cam_Epi,
  MaxDataNumber => Set_Cam_Mdn,
  Busy => Set_Cam_B,
  CommandAborted => Set_Cam_Ca,
  Error => Set_Cam_Err,
  ErrorID => Set_Cam_ErrID
);
```

11

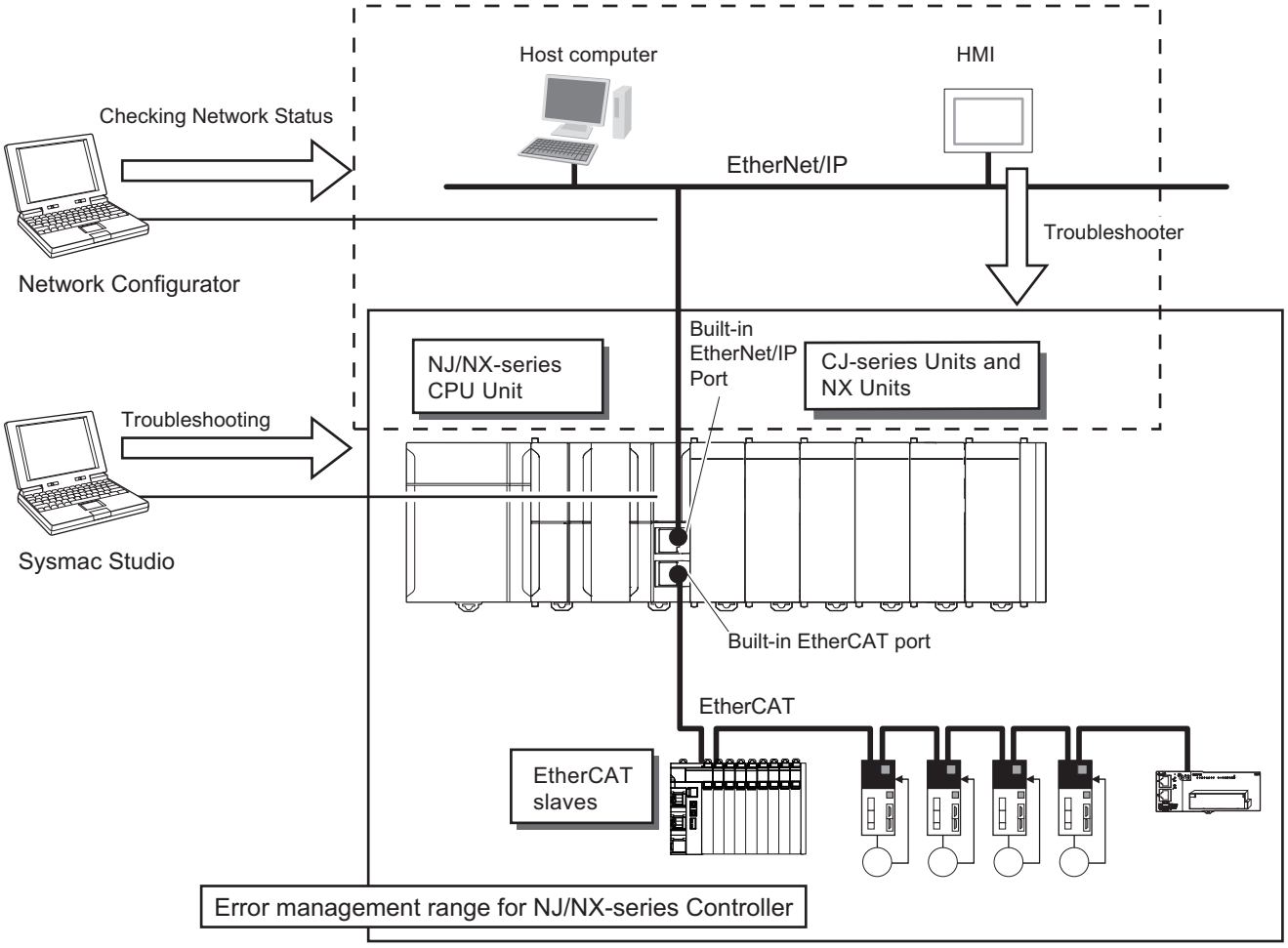
Troubleshooting

This section describes the overview of the methods for checking errors.

11-1 Overview of Troubleshooting

You manage all of the errors that occur on the NJ/NX-series Controller as events.
This allows you to see what errors have occurred and find corrections for them with the same methods for the entire range of errors that is managed (i.e., CPU Unit, NX Units, NX-series Slave Terminals, EtherCAT slaves,*1 and CJ-series Units).

*1. Only Sysmac devices are supported.



You can use the troubleshooting functions of the Sysmac Studio or the Troubleshooter on an HMI to quickly check for errors that have occurred and find corrections for them.

Refer to the *NJ/NX-series Troubleshooting Manual (Cat. No. W503)* for error types and details, specific corrections when errors occur, and troubleshooting information on the entire NJ/NX-series Controllers.

A

Appendices

This section describes settings and connection methods for OMRON 1S-series Servo Drive and G5-series Servo Drive objects.

A-1 Connecting the 1S-series Servo Drive

This appendix describes connections to an OMRON 1S-series Servo Drive with built-in EtherCAT communications.

A-1-1 Wiring the Servo Drive

Servo Drives are connected using EtherCAT communications.
Refer to the *NJ/NX-series CPU Unit Built-in EtherCAT Port User's Manual (Cat. No. W505)* for information on the connection methods.

A-1-2 Servo Drive Settings

This section outlines the Servo Drive settings that are used when connected to OMRON 1S-series Servo Drives with built-in EtherCAT communications (i.e., the applicable Servo Drives for the MC Function Module).
For details on the Servo Drives, refer to the *AC Servomotors/Servo Drives 1S-series with Built-in EtherCAT Communications User's Manual (Cat. No. I586)* or *AC Servomotors/Servo Drives 1S-series with Built-in EtherCAT Communications and Safety Functionality User's Manual (Cat. No. I621)*.

Assigning External Input Signals

The MC Function Module uses the general-purpose inputs of the Servo Drive for the following input signals.

- Immediate stop input
- Positive limit input
- Negative limit input
- Home proximity input
- External latch trigger signals (latch input 1 and latch input 2)

● Assigning Positive Limit Inputs, Negative Limit Inputs, and Home Proximity Input

The default settings of the input signals of an OMRON 1S-series Servo Drive are listed in the following table.

Signal name	Input signal	Wiring of R88D-1SN□	Wiring of R88D-1SAN□
Immediate stop input	Servo Drive general-purpose input 1 (IN1)	Pin 12 on connector CN1, NC*1	Pin 5 on connector CN1, NC*1
Positive limit input	Servo Drive general-purpose input 2 (IN2)	Pin 32 on connector CN1, NC*2	Pin 19 on connector CN1, NC*2
Negative limit input	Servo Drive general-purpose input 3 (IN3)	Pin 13 on connector CN1, NC*3	Pin 6 on connector CN1, NC*3
Home proximity input	Servo Drive general-purpose input 4 (IN4)	Pin 33 on connector CN1, NO	Pin 20 on connector CN1, NO

*1. The signal name for the Servo Drive is the error stop input.
*2. The signal name for the Servo Drive is the positive drive prohibit input.
*3. The signal name for the Servo Drive is the negative drive prohibit input.

● Trigger Signal Assignments for External Latches

The input signals in the following table are assigned to external latch trigger signals by default for the OMRON 1S-series Servo Drive.

Settings for the TriggerInput (Trigger Input Condition) input variable of the MC_TouchProbe instruction			External latch trigger signal	Wiring of R88D-1SN□	Wiring of R88D-1SAN□
Mode	InputDrive	LatchID			
0: mcDrive	0: mcEncoder-Mark	---	Encoder Z phase	---	---
	1: mcEXT	1: mcLatch1	Servo Drive general-purpose input 7 (IN7)	Pin 15 on connector CN1, NO*1	Pin 8 on connector CN1, NO*1
		2: mcLatch2	Servo Drive general-purpose input 8 (IN8)	Pin 35 on connector CN1, NO*2	Pin 22 on connector CN1, NO*2
1: mcController	---	---	Variable specified by TriggerVariable	---	---

*1. The signal name for the Servo Drive is the external latch input 1.
*2. The signal name for the Servo Drive is the external latch input 2.

Backlash Compensation

The MC Function Module does not perform backlash compensation.
If you require backlash compensation, use the compensation function on the Servo Drive.
The objects that must be set on the Servo Drive are listed in the following table.

Index	Subindex	Name	Description
3001 hex	---	Machine	---
	02 hex	Backlash Compensation Selection	Selects whether to enable or disable backlash compensation in the position control, and the operation direction for the compensation.*1 0: Backlash compensation disabled 1: Compensate at the first positive operation after servo ON 2: Compensate at the first negative operation after servo ON
	03 hex	Backlash Compensation Amount	Sets the backlash compensation amount in the position control.
	04 hex	Backlash Compensation Time Constant	Sets the backlash compensation time constant in the position control.

*1. The default setting is 0: Backlash compensation disabled.

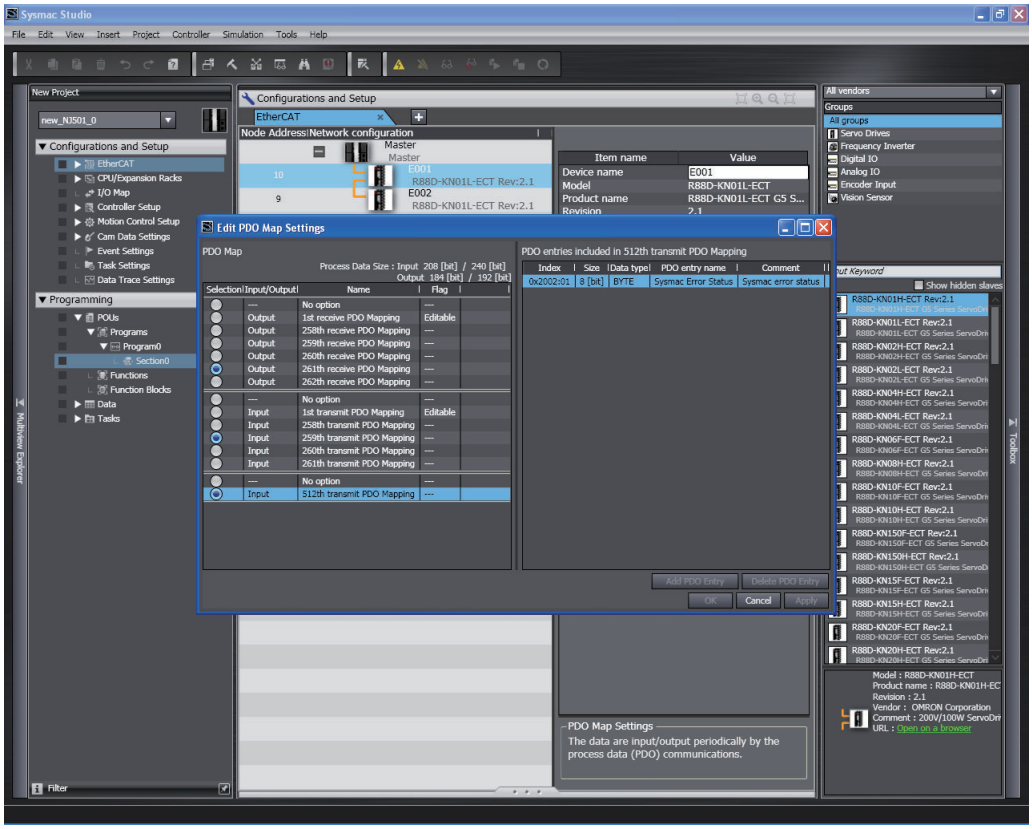
For details on the backlash function, refer to the *AC Servomotors/Servo Drives 1S-series with Built-in EtherCAT Communications User's Manual (Cat. No. I586)* or *AC Servomotors/Servo Drives 1S-series with Built-in EtherCAT Communications and Safety Functionality User's Manual (Cat. No. I621)*.

PDO Mapping

This section describes mapping PDOs to control servo axes from the MC Function Module.

To use motion control functions, you must map the objects that are required for those functions to PDOs.
The PDO map is a list of required objects that is prepared in advance.

You select the PDOs to use in the **Edit PDO Map Settings Window** of the **EtherCAT** Tab Page in the Sysmac Studio.



The following PDOs are mapped by default in the Sysmac Studio.

RxPDO: 261th Receive PDO Mapping (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: 259th Transmit PDO Mapping (1B02 hex)	Error Code(603F hex), Statusword (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)

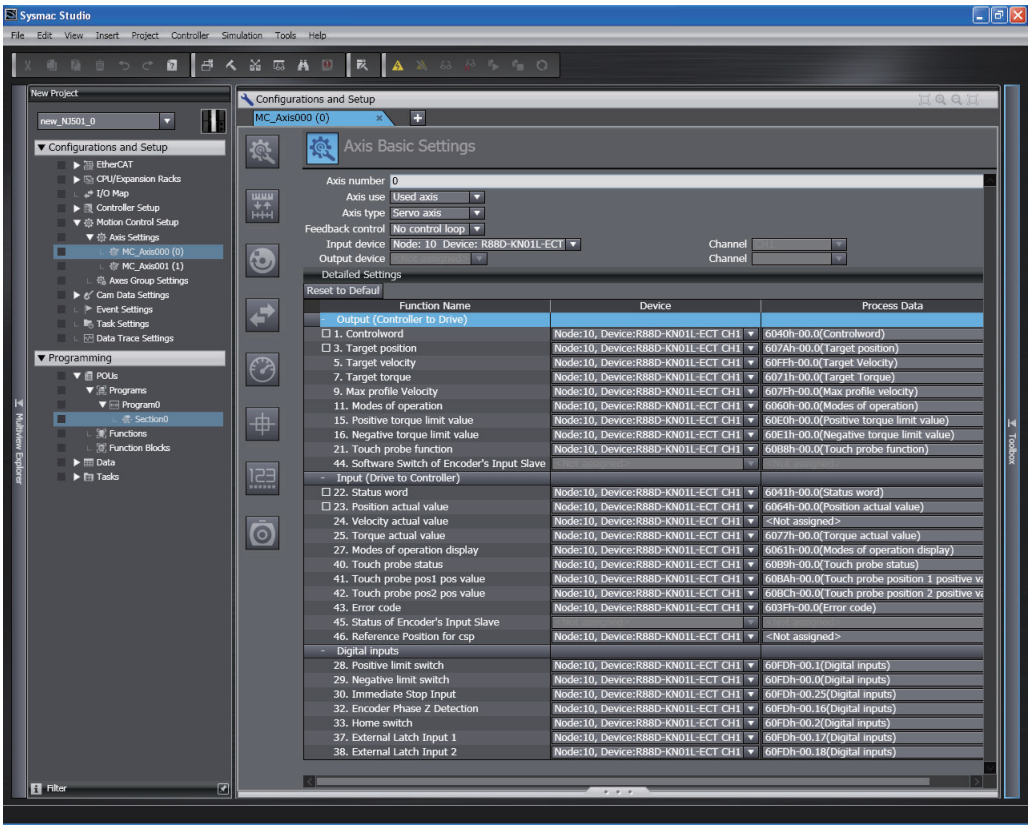
Additional Information

If you use the recommended OMRON Servo Drives, R88D-1SN□□□-ECT or R88D-1SAN□□□-ECT, it is not necessary to change the default PDO map on the Sysmac Studio.

Relationships between MC Function Module and Process Data

The functions of the MC Function Module are related to the information in the process data objects. Depending on the EtherCAT slave configuration and functions that are used by the MC Function Module, you sometimes must change the relationships between the MC Function Module and the PDOs.

To access the settings, click the **Detailed Settings** Button on the **Axis Basic Settings** Display in the Sysmac Studio.



Additional Information

If you use the recommended OMRON Servo Drives, R88D-1SN□□□-ECT or R88D-1SAN□□□-ECT, it is not necessary to change the the default relationships between MC Function Module functions and the PDOs on the Sysmac Studio.

● Output Settings (Controller to Servo Drive)

The input settings apply to the command data that is sent from the MC Function Module to the Servo Drive.

The default settings in the Sysmac Studio are listed in the following table. (Required objects are marked with a circle.)

Function name	Process data	Description
○ Control word	6040 hex-00.0 (Controlword)	This data is used to control the status of the Servo Drive. Set 6040 hex: Controlword.
○ Target position	607A hex-00.0 (Target position)	The target position for position control. Set **607A hex: Target position**.
Target velocity	60FF hex-00.0 (Target velocity)	The target velocity for velocity control. This object is necessary to output to the Servo Drive in Cyclic Synchronous Velocity Control Mode by the MC_SyncMoveVelocity (Cyclic Synchronous Velocity Control) and other instructions. Normally set **60FF hex: Target velocity**.

Function name	Process data	Description
Target torque	6071 hex-00.0 (Target torque)	The target torque for torque control. This object is necessary to output to the Servo Drive in Cyclic Synchronous Torque Control Mode by the MC_TorqueControl (Torque Control) and other instructions. Normally set **6071 hex: Target torque**.
Max profile velocity	607F hex-00.0 (Max profile velocity)	The velocity limit value for torque control. This object is necessary for velocity control in Cyclic Synchronous Torque Control Mode by the MC_TorqueControl (Torque Control) and other instructions. Normally set **607F hex: Max profile velocity**.
Modes of operation	6060 hex-00.0 (Modes of operation)	This data is required to change the control mode. This object is necessary to change to a control mode other than Cyclic Synchronous Position Control Mode for the MC_SyncMoveVelocity (Cyclic Synchronous Velocity Control), MC_TorqueControl (Torque Control), and other instructions. Normally set **6060 hex: Modes of operation**. *1
Positive torque limit value	60E0 hex-00.0 (Positive torque limit value)	This is the torque limit value in the positive direction. This object is necessary to control the output torque of the Servo Drive by the MC_SetTorqueLimit and other instructions. Normally set **60E0 hex: Positive torque limit value**.
Negative torque limit value	60E1 hex-00.0 (Negative torque limit value)	This is the torque limit value in the negative direction. This object is necessary to control the output torque of the Servo Drive by the MC_SetTorqueLimit and other instructions. Normally set **60E1 hex: Negative torque limit value**.
Touch probe function	60B8 hex-00.0 (Touch probe function)	This data is used to control the touch probe function. It is required for the touch probe function for the MC_Home, MC_HomeWithParameter, MC_MoveFeed (Interrupt Feeding), MC_TouchProbe (Enable External Latch), MC_MoveLink (Synchronous Positioning), and other instructions. Normally set **60B8 hex: Touch probe function**.

*1. If you set **6060 hex: Modes of operation**, also set **6061 hex: Modes of operation display**. Normal operation is not possible if only one of these two is set.

Precautions for Correct Use

- If you change the settings, make sure that the desired operations are performed for the MC Function Module and process data settings.
- If you are not using an OMRON 1S-series Servo Drive with built-in EtherCAT communications, always set the Modes of Operation (6060 hex).

● Input Settings (Servo Drive to Controller)

This is the status data from the Servo Drive to the MC Function Module.
The default settings in the Sysmac Studio are listed in the following table. (Required objects are marked with a circle.)

Function name	Process data	Description
○ Statusword	6041 hex-00.0 (Statusword)	The status of the Servo Drive. Set **6041 hex: Statusword**.

Function name		Process data	Description
○	Position actual value	6064 hex-00.0 (Position actual value)	Shows the actual position. Set **6064 hex: Position actual value**.
	Velocity actual value	Not set.*1	Shows the actual velocity. If you use it, normally set **606C hex: Velocity actual value**.
	Torque actual value	6077 hex (Torque actual value)	Shows the actual torque. This object is necessary to output to the Servo Drive in Cyclic Synchronous Torque Control Mode by the MC_TorqueControl (Torque Control) and other instructions. Normally set **6077 hex: Torque actual value**.
	Modes of operation display	6061 hex-00.0 (Modes of operation display)	Shows the operation mode. This object is necessary to change to a control mode other than Cyclic Synchronous Position Control Mode for the MC_SyncMoveVelocity (Cyclic Synchronous Velocity Control), MC_TorqueControl (Torque Control), and other instructions. Normally set **6061 hex: Modes of operation display**. *2
	Touch probe status	60B9 hex-00.0 (Touch probe status)	Shows the status of the touch probe function. It is required for the touch probe function for the MC_Home, MC_HomeWithParameter, MC_MoveFeed (Interrupt Feeding), MC_TouchProbe (Enable External Latch), MC_MoveLink (Synchronous Positioning), and other instructions. Normally set **60B9 hex: Touch probe status**.
	Touch probe pos1 pos value	60BA hex-00.0 (Touch probe pos1 pos value)	The latched position for touch probe 1. It is required for the touch probe function for the MC_Home, MC_HomeWithParameter, MC_MoveFeed (Interrupt Feeding), MC_TouchProbe (Enable External Latch), MC_MoveLink (Synchronous Positioning), and other instructions. Normally set 60BA hex: Touch probe pos1 pos value.
	Touch probe pos2 pos value	60BC hex-00.0 (Touch probe pos2 pos value)	The latched position for touch probe 2. It is required for the touch probe function for the MC_Home, MC_HomeWithParameter, MC_MoveFeed (Interrupt Feeding), MC_TouchProbe (Enable External Latch), MC_MoveLink (Synchronous Positioning), and other instructions. Normally set 60BC hex: Touch probe pos2 pos value.
	Error code	603F hex-00.0 (Error code)	The error code in the Servo Drive. Normally set **603F hex: Error code**.
	Reference position for csp	Not set.	The reference position for changing the csp mode. This data is accessed by instructions that are used in Velocity Control Mode (CSV) or Torque Control Mode (CST). This object is supported for OMRON 1S-series Servomotors/ Servo Drives. *3

*1. If required, map the selected process data to a PDO before setting it.
The standard setting is 606C hex-00.0 (Velocity actual value).

*2. If you set **6061 hex: Modes of operation display**, also set **6060 hex: Modes of operation**. Normal operation is not possible if only one of these two is set.

*3. Map **3010-87 hex: Reference Position for CSP** to a PDO when you use an OMRON 1S-series Servomotor/Servo Drive.

Precautions for Correct Use

- If you change the settings, make sure that the desired operations are performed for the MC Function Module and process data settings.
- If you are not using an OMRON 1S-series Servo Drive with built-in EtherCAT communications, always set the Modes of Operation Display (6061 hex).
- To use the MC_SyncMoveVelocity (Cyclic Synchronous Velocity Control) instruction to change the control mode of an OMRON 1S-series Servo Drive, you need to map Reference Position for CSP.

● Digital Input Settings

The MC Function Module uses the following input signals of the Servo Drive.

Function name	Process data	Description
Positive drive prohibit input	60FD hex-00.1 (Digital inputs)	This signal is used for the positive limit input. Normally set Bit 1: Positive limit switch of 60FD hex-00: Digital inputs.
Negative drive prohibit input	60FD hex-00.0 (Digital inputs)	This signal is used for the negative limit input. Normally set Bit 0: Negative limit switch of 60FD hex-00: Digital inputs.
Error stop input	60FD hex-00.25 (Digital inputs)	This signal is used for the immediate stop input. Set Bit 25: Error stop input of 60FD hex-00: Digital inputs for an OMRON 1S-series Servo Drive.
Encoder Z-phase detection	60FD hex-00.16 (Digital inputs)	Shows the status of detecting the Z-phase input. Set **Bit 16: Encoder phase Z detection** of **60FD hex-00: Digital inputs** for an OMRON 1S-series Servomotor/Servo Drive.
Home proximity input	60FD hex-00.2 (Digital inputs)	This signal is used for the home proximity input. Normally set Bit 2: Home switch of 60FD hex-00: Digital inputs.
External latch input 1	60FD hex-00.17 (Digital inputs)	Shows the status of the signal that is used for external latch input 1. Set Bit 17: External latch input 1 of 60FD hex-00: Digital inputs for an OMRON 1S-series Servo Drive.
External latch input 2	60FD hex-00.18 (Digital inputs)	Shows the status of the signal that is used for external latch input 2. Set Bit 18: External latch input 2 of 60FD hex-00: Digital inputs for an OMRON 1S-series Servo Drive.

Precautions for Correct Use

If you change the settings, make sure that the desired operations are performed for the MC Function Module and process data settings.

Version Information

- If you are using a CPU Unit with unit version 1.10 or later, operation is as described in the following table depending on whether Modes of Operation (6060 hex) and Modes of Operation Display (6061 hex) are mapped.

	Modes of Operation Display	
	(6061 hex) mapped	**(6061 hex) not mapped**
Modes of Operation (6060 hex) mapped	• You can execute instructions that use CSP*1, CSV*2, or CST*3. • The servo is OFF in any control mode other than CSP, CSV, or CST.	• You can execute instructions that use CSP. If you execute any instruction that uses any other control mode, a Process Data Object Setting Missing error (error code 3461 hex) occurs. • The MC Function Module assumes that the CSP Servo Drive control mode is used. Command the Servo Drive to use CSP.
Modes of Operation (6060 hex) not mapped	• You can execute instructions that use CSP. If you execute any instruction that uses any other control mode, a Process Data Object Setting Missing error (error code 3461 hex) occurs. • The servo is OFF in any control mode other than CSP.	You can execute instructions that use CSP. If you execute any instruction that uses any other control mode, a Process Data Object Setting Missing error (error code 3461 hex) occurs.

*1. CSP is the Cyclic Synchronous Position Control Mode of the Servo Drive.
*2. CSV is the Cyclic Synchronous Velocity Control Mode of the Servo Drive.
*3. CST is the Cyclic Synchronous Torque Control Mode of the Servo Drive.

- If you are using a CPU Unit with unit version 1.09 or earlier and you are not using an OMRON 1S-series Servo Drive with built-in EtherCAT communications for the servo axis, Modes of Operation (6060 hex) and Modes of Operation Display (6061 hex) are required.

Object Settings

The OMRON 1S-series Servo Drive settings required to use the control functions of the MC Function Module are listed in the following table.
Consult the manual for your Servo Drive and set all related objects for the Servo Drive functions that you are going to use.

Index	Subindex	Name	Recommended setting	Description
3001 hex	---	Machine	---	The gear ratio on the Servo is 1:1. Set the user unit on the Controller. *1
	05 hex	Electronic Gear Ratio Numerator	1	
	06 hex	Electronic Gear Ratio Denominator	1	
3330 hex	---	Torque Limit	---	Use **60E0 hex** and **60E1 hex** as the torque limit values when PCL and NCL are OFF.
	01 hex	Switch Selection	2	
3A00 hex	---	Homing	---	Use *0* as the offset value on the Servo.
	06 hex	Encoder Home Offset	0	
3B10 hex	---	Drive Prohibit	---	Disable the drive prohibit input at the Servo. It is processed by the Controller.
	01 hex	Enable	0	

Index	Subindex	Name	Recommended setting	Description
3B11 hex	---	Software Limits	---	Disable both the positive and negative software limits.
	01 hex	Enable Selection	0	
3B30 hex	---	Touch Probe Function 1	---	Set this as follows: Touch probe1 = External latch signal 1, Touch probe2 = External latch signal 2.
	01 hex	Latch 1 Trigger Selection	1	
3B31 hex	---	Touch Probe Function 2	---	
	01 hex	Latch 2 Trigger Selection	2	
4020 hex	---	Warning Customization	---	Set this to automatically reset warnings when the cause is removed.
	04 hex	Warning Hold Selection	0	
4510 hex	---	Encoder	---	Use as absolute encoders. Ignore multi-rotation counter overflow.
	01 hex	Absolute Encoder Operation Selection	2	
4630 hex	---	Positive Drive Prohibit	---	Assign the Positive Drive Prohibit to the general-purpose input 2 (IN2) as a negative logic (NC input).
	01 hex	Port Selection	2	
	02 hex	Logic Selection	1	
4631 hex	---	Negative Drive Prohibit	---	Assign the Negative Drive Prohibit to the general-purpose input 3 (IN3) as a negative logic (NC input).
	01 hex	Port Selection	3	
	02 hex	Logic Selection	1	
4632 hex	---	External Latch Input 1	---	Assign the External Latch Input 1 to the general-purpose input 7 (IN7) as a positive logic (NO input).
	01 hex	Port Selection	7	
	02 hex	Logic Selection	0	
4633 hex	---	External Latch Input 2	---	Assign the External Latch Input 2 to the general-purpose input 8 (IN8) as a positive logic (NO input).
	01 hex	Port Selection	8	
	02 hex	Logic Selection	0	
4634 hex	---	Home Proximity Input	---	Assign the Home Proximity Input to the general-purpose input 4 (IN4) as a positive logic (NO input).
	01 hex	Port Selection	4	
	02 hex	Logic Selection	0	

*1. With a CPU Unit of unit version 1.10 or earlier, you cannot operate OMRON 1S-series Servomotors in the maximum rotation speed. To operate the 1S-series Servomotors in the maximum rotation speed, set the electronic gear ratio to 2:1 or greater.

A-2 Connecting the G5-series Servo Drive

This appendix describes connections to an OMRON G5-series Servo Drive with built-in EtherCAT communications.

A-2-1 Wiring the Servo Drive

Servo Drives are connected using EtherCAT communications.
Refer to the *NJ/NX-series CPU Unit Built-in EtherCAT Port User's Manual (Cat. No. W505)* for information on the connection methods.

A-2-2 Servo Drive Settings

This section outlines the Servo Drive settings that are used when connected to OMRON G5-series Servo Drives with built-in EtherCAT communications (i.e., the applicable Servo Drives for the MC Function Module).
For details on the Servo Drives, refer to the *AC Servomotors/Servo Drives G5-series with Built-in EtherCAT Communications User's Manual (Cat. No. I576)* or the *AC Servomotors/Servo Drives G5-series with Built-in EtherCAT Communications Linear Motor Type User's Manual (Cat. No. I577)*.

Recommended Servo Drives

All of the functions of the MC Function Module can be used for Servo Drives with the unit versions that are given in the following table.

Manufacturer	Compatible models	Applicable unit versions
OMRON	R88D-KN□□□-ECT	Unit version 2.1 or later
	R88D-KN□□□-ECT-L	Unit version 1.1 or later

Additional Information

- You can also use unit versions of the OMRON G5-series Servo Drives with built-in EtherCAT communications other than the recommended unit versions. The functions that you can use depend on the specifications of the Servo Drive. Set the functions to use and the object dictionary on Sysmac Studio.
- The R88D-KN□□□-ECT-R (unit version 1.0) of the OMRON G5-series Servo Drives support only position control (Cyclic Synchronous Position Control Mode).
  You can use them for applications that do not require velocity control (Cyclic Synchronous Velocity Control Mode) or torque control (Cyclic Synchronous Torque Control Mode).
  Refer to the *AC Servomotors/Servo Drives G5-series with Built-in EtherCAT Communications User's Manual (Cat. No. I573)* for details on functions.
- When you use unit version 2.0 or earlier of an OMRON G5-series Cylinder-type Servomotor/Servo Drive, do not set the node address switches to *00*. If you set them to *00*, a network configuration error occurs.
- Refer to the *AC Servomotors/Servo Drives G5-series with Built-in EtherCAT Communications User's Manual (Cat. No. I576)* for details on the differences between the unit versions of the OMRON G5-series Servo Drives.

Assigning External Input Signals

The MC Function Module uses the general-purpose inputs of the Servo Drive for the following input signals.

- Immediate stop input
- Positive limit input
- Negative limit input
- Home proximity input
- External latch trigger signals (latch input 1 and latch input 2)

● Assigning Positive Limit Inputs, Negative Limit Inputs, and Home Proximity Input

The default settings of the input signals of an OMRON G5-series Servo Drive are listed in the following table.

Signal name	Input signal
Immediate stop input	Servo Drive general-purpose input 1 (IN1: pin 5 on connector CN1, NC)
Positive limit input	Servo Drive general-purpose input 2 (IN2: pin 7 on connector CN1, NC)*1
Negative limit input	Servo Drive general-purpose input 3 (IN3: pin 8 on connector CN1, NC)*2
Home proximity input	Servo Drive general-purpose input 4 (IN4: pin 9 on connector CN1, NO)

*1. The signal name for the Servo Drive is the positive drive prohibit input.
*2. The signal name for the Servo Drive is the negative drive prohibit input.

● Trigger Signal Assignments for External Latches

The input signals in the following table are assigned to external latch trigger signals by default for the OMRON G5-series Servo Drive.

Settings for the *TriggerInput* (Trigger Input Condition) input variable of the MC_TouchProbe instruction			External latch trigger signal
Mode	InputDrive	LatchID	
0: mcDrive	0: mcEncoderMark	---	Encoder Z phase
	1: mcEXT	1: mcLatch1	Servo Drive general-purpose input 7 (IN7: pin 12 on connector CN1, NO)*1
		2: mcLatch2	Servo Drive general-purpose input 6 (IN6: pin 11 on connector CN1, NO)*2
1: mcController	---	---	Variable specified by TriggerVariable

*1. The signal name for the Servo Drive is the external latch input 1.
*2. The signal name for the Servo Drive is the external latch input 2.

Backlash Compensation

The MC Function Module does not perform backlash compensation.
If you require backlash compensation, use the compensation function of the Servo Drive.
The objects that must be set on the Servo Drive are listed in the following table.

Index	Name	Description
3704 hex	Backlash Compensation Selection	This object is used to select whether to enable or disable backlash compensation during position control, and to set the compensation direction. The default value is to disable compensation.
3705 hex	Backlash Compensation Amount	Set the backlash compensation amount during position control.
3706 hex	Backlash Compensation Time Constant	Set the backlash compensation time constant during position control.

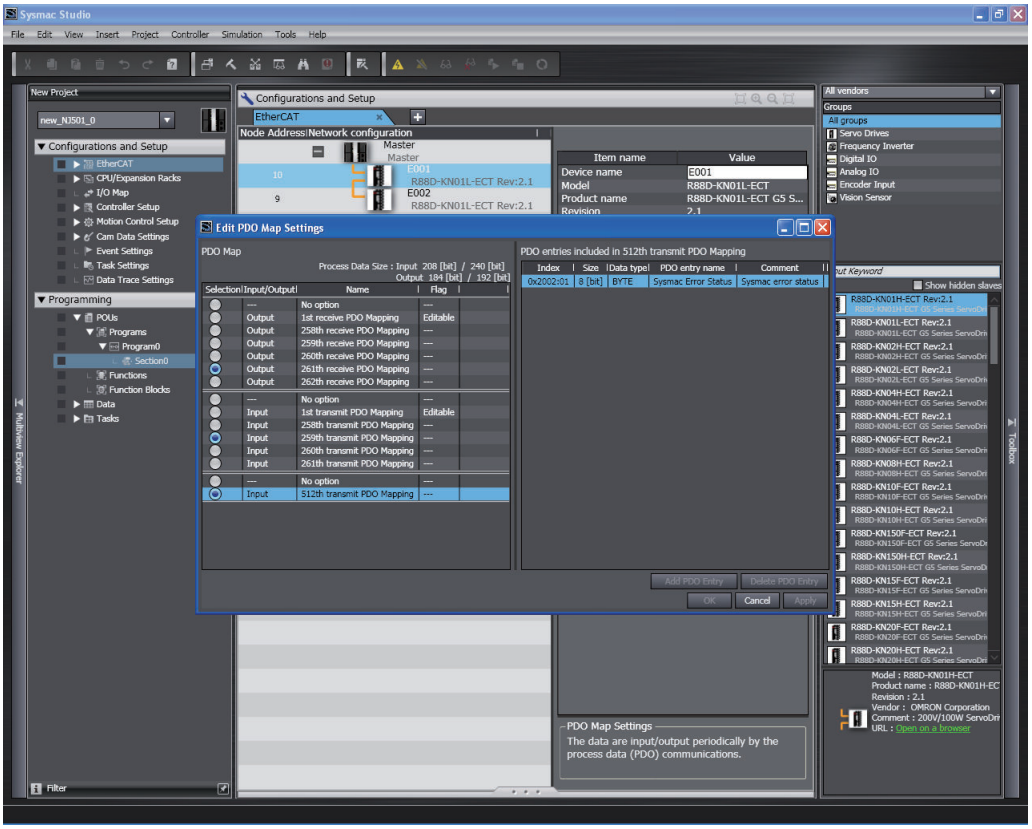
For details on the backlash function, refer to the *AC Servomotors/Servo Drives G5-series with Built-in EtherCAT Communications User's Manual (Cat. No. I576)* or the *AC Servomotors/Servo Drives G5-series with Built-in EtherCAT Communications Linear Motor Type User's Manual (Cat. No. I577)*.

PDO Mapping

This section describes mapping PDOs to control servo axes from the MC Function Module.

To use motion control functions, you must map the objects that are required for those functions to PDOs.
The PDO map is a list of required objects that is prepared in advance.

You select the PDOs to use in the **Edit PDO Map Settings Window** of the **EtherCAT** Tab Page in the Sysmac Studio.



The following PDOs are mapped by default in the Sysmac Studio.

RxPDO: 261th Receive PDO Mapping (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: 259th Transmit PDO Mapping (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)

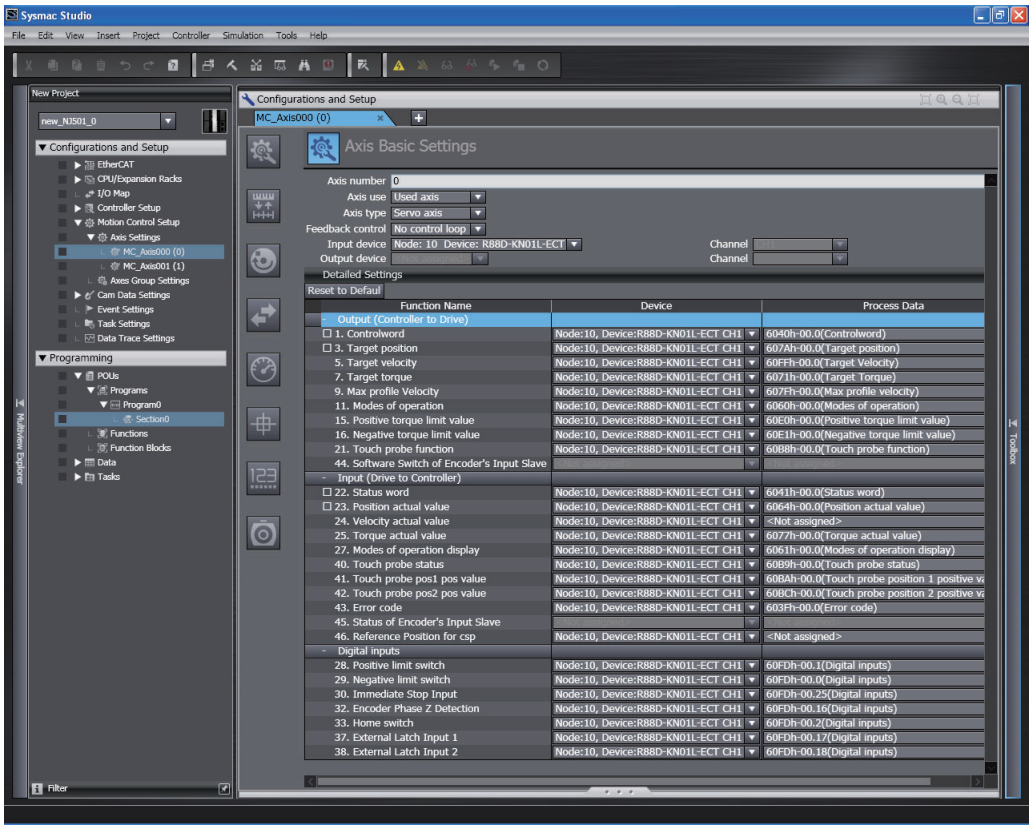
Additional Information

- If you use the recommended OMRON Servo Drives (R88D-KN□□□-ECT, unit version 2.1 or later, or R88D-KN□□□-ECT-L, unit version 1.1 or later), then it is not necessary to change the default PDO map on the Sysmac Studio.
- To perform fully-closed control with OMRON R88D-KN□□□-ECT, select 1701 hex or 1600 hex for RxPDO. For 1600 hex, the total size of objects must be set to 12 bytes or less (for unit version 2.1 or later).

Relationships between MC Function Module and Process Data

The functions of the MC Function Module are related to the information in the process data objects. Depending on the EtherCAT slave configuration and functions that are used by the MC Function Module, you sometimes must change the relationships between the MC Function Module and the PDOs.

To access the settings, click the **Detailed Settings** Button on the **Axis Basic Settings** Display in the Sysmac Studio.



Additional Information

If you use the recommended OMRON Servo Drives (R88D-KN□□□-ECT, unit version 2.1 or later, or R88D-KN□□□-ECT-L, unit version 1.1 or later), then it is not necessary to change the default relationships between MC Function Module functions and the PDOs on the Sysmac Studio.

● Output Settings (Controller to Servo Drive)

The input settings apply to the command data that is sent from the MC Function Module to the Servo Drive.

The default settings in the Sysmac Studio are listed in the following table. (Required objects are marked with a circle.)

Function name	Process data	Description
○ Control word	6040 hex-00.0 (Controlword)	This data is used to control the status of the Servo Drive. Set **6040 hex: Controlword**.
○ Target position	607A hex-00.0 (Target position)	The target position for position control. Set **607A hex: Target position**.
Target velocity	60FF hex-00.0 (Target velocity)	The target velocity for velocity control. This object is necessary to output to the Servo Drive in Cyclic Synchronous Velocity Control Mode by the MC_SyncMoveVelocity (Cyclic Synchronous Velocity Control) and other instructions. Normally set **60FF hex: Target velocity**.
Target torque	6071 hex-00.0 (Target torque)	The target torque for torque control. This object is necessary to output to the Servo Drive in Cyclic Synchronous Torque Control Mode by the MC_TorqueControl (Torque Control) and other instructions. Normally set **6071 hex: Target torque**.
Maximum profile velocity	607F hex-00.0 (Max profile velocity)	The velocity limit value for torque control. This object is necessary for velocity control in Cyclic Synchronous Torque Control Mode by the MC_TorqueControl (Torque Control) and other instructions. Normally set **607F hex: Max profile velocity**.
Modes of operation	6060 hex-00.0 (Modes of operation)	This data is required to change the control mode. This object is necessary to change to a control mode other than Cyclic Synchronous Position Control Mode for the MC_SyncMoveVelocity (Cyclic Synchronous Velocity Control), MC_TorqueControl (Torque Control), and other instructions. Normally set **6060 hex: Modes of operation**.*1
Positive torque limit value	60E0 hex-00.0 (Positive torque limit value)	This is the torque limit value in the positive direction. This object is necessary to control the output torque of the Servo Drive by the MC_SetTorqueLimit and other instructions. Normally set **60E0 hex: Positive torque limit value**.
Negative torque limit value	60E1 hex-00.0 (Negative torque limit value)	This is the torque limit value in the negative direction. This object is necessary to control the output torque of the Servo Drive by the MC_SetTorqueLimit and other instructions. Normally set **60E1 hex: Negative torque limit value**.

Function name	Process data	Description
Touch probe function	60B8 hex-00.0 (Touch probe function)	This data is used to control the touch probe function. It is required for the touch probe function for the MC_Home, MC_HomeWithParameter, MC_MoveFeed (Interrupt Feeding), MC_TouchProbe (Enable External Latch), MC_MoveLink (Synchronous Positioning), and other instructions. Normally set **60B8 hex: Touch probe function**.

*1. If you set **6060 hex: Modes of operation**, also set **6061 hex: Modes of operation display**. Normal operation is not possible if only one of these two is set.

Precautions for Correct Use

- Some functions may not be supported if you connect a unit version of the OMRON G5-series Servo Drives with built-in EtherCAT communications other than the recommended unit versions.
  Refer to the manual for the connected servo drive for details.
- If you change the settings, make sure that the desired operations are performed for the MC Function Module and process data settings.
- If you are not using an OMRON G5-series Servo Drive with built-in EtherCAT communications, always set the Modes of Operation (6060 hex).
- To perform fully-closed control with OMRON G5-series Servomotors/Servo Drives with built-in EtherCAT communications, make settings so that the size of objects totals 12 bytes or less.

● Input Settings (Servo Drive to Controller)

This is the status data from the Servo Drive to the MC Function Module.
The default settings in the Sysmac Studio are listed in the following table. (Required objects are marked with a circle.)

	Function name	Process data	Description
○	Status word	6041 hex-00.0 (Statusword)	The status of the Servo Drive. Set **6041 hex: Statusword**.
○	Position actual value	6064 hex-00.0 (Position actual value)	Shows the actual position. Set **6064 hex: Position actual value**.
	Velocity actual value	Not set.*1	Shows the actual velocity. Normally set **606C hex: Velocity actual value**.
	Torque actual value	6077 hex (Torque actual value)	Shows the actual torque. This object is necessary to output to the Servo Drive in Cyclic Synchronous Torque Control Mode by the MC_TorqueControl (Torque Control) and other instructions. Normally set **6077 hex: Torque actual value**.
	Modes of operation display	6061 hex-00.0 (Modes of operation display)	Shows the operation mode. This object is necessary to change to a control mode other than Cyclic Synchronous Position Control Mode for the MC_SyncMoveVelocity (Cyclic Synchronous Velocity Control), MC_TorqueControl (Torque Control), and other instructions. Normally set **6061 hex: Modes of operation display**. *2

Function name	Process data	Description
Touch probe status	60B9 hex-00.0 (Touch probe status)	Shows the status of the touch probe function. It is required for the touch probe function for the MC_Home, MC_HomeWithParameter, MC_MoveFeed (Interrupt Feeding), MC_TouchProbe (Enable External Latch), MC_MoveLink (Synchronous Positioning), and other instructions. Normally set **60B9 hex: Touch probe status**.
Touch probe position 1 position value	60BA hex-00.0 (Touch probe pos1 pos value)	The latched position for touch probe 1. It is required for the touch probe function for the MC_Home, MC_HomeWithParameter, MC_MoveFeed (Interrupt Feeding), MC_TouchProbe (Enable External Latch), MC_MoveLink (Synchronous Positioning), and other instructions. Normally set 60BA hex: Touch probe pos1 pos value.
Touch probe position 2 position value	60BC hex-00.0 (Touch probe pos2 pos value)	The latched position for touch probe 2. It is required for the touch probe function for the MC_Home, MC_HomeWithParameter, MC_MoveFeed (Interrupt Feeding), MC_TouchProbe (Enable External Latch), MC_MoveLink (Synchronous Positioning), and other instructions. Normally set 60BC hex: Touch probe pos2 pos value.
Error code	603F hex-00.0 (Error code)	The error code in the Servo Drive. Normally set **603F hex: Error code**.
Reference position for csp	Not set.	The reference position for changing the csp mode. This data is accessed by instructions that are used in Velocity Control Mode (CSV) or Torque Control Mode (CST). This object is supported for OMRON G5-series Cylinder-type Servomotors/Servo Drives with unit version 2.1 or later. *3 OMRON G5-series Linear Motor Type Servo Drives do not support this object.

*1. If required, map the selected process data to a PDO before setting it.
The standard setting is **606C hex-00.0 (Velocity actual value)**.

*2. If you set **6061 hex: Modes of operation display**, also set **6060 hex: Modes of operation**. Normal operation is not possible if only one of these two is set.

*3. Map **4020 hex: Reference Position for CSP** to a PDO when you use an OMRON G5-series Servomotor/Servo Drive.

Precautions for Correct Use

- Some functions may not be supported if you connect a unit version of the OMRON G5-series Servo Drives with built-in EtherCAT communications other than the recommended unit versions. Refer to the manual for the connected servo drive for details.
- If you change the settings, make sure that the desired operations are performed for the MC Function Module and process data settings.
- If you are not using an OMRON G5-series Servo Drive with built-in EtherCAT communications, always set the Modes of Operation Display (6061 hex).
- To use the MC_SyncMoveVelocity (Cyclic Synchronous Velocity Control) instruction to change the control mode of an OMRON G5-series Servo Drive, you need to map Reference Position for CSP.
- Set the primary period to 1 ms or greater when you use Reference Position for CSP for an OMRON G5-series Servo Drive. Also, set the electronic gear ratio to 1:1 for the G5-series Servo Drive.
  For details, refer to the *AC Servomotors/Servo Drives G5-series with Built-in EtherCAT Communications User's Manual (Cat. No. I576)* or the *AC Servomotors/Servo Drives G5-series with Built-in EtherCAT Communications Linear Motor Type User's Manual (Cat. No. I577)*.

● Digital Input Settings

The MC Function Module uses the following input signals of the Servo Drive.

Function name	Process data	Description
Positive limit switch (positive drive prohibit input)	60FD hex-00.1 (Digital inputs)	This signal is used for the positive limit input. Normally set Bit 1: Positive limit switch of 60FD hex-00: Digital inputs.
Negative limit switch (negative drive prohibit input)	60FD hex-00.0 (Digital inputs)	This signal is used for the negative limit input. Normally set Bit 0: Negative limit switch of 60FD hex-00: Digital inputs.
Immediate stop input	60FD hex-00.25 (Digital inputs)	This signal is used for the immediate stop input. Set Bit 25: Immediate Stop Input of 60FD hex-00: Digital inputs for an OMRON G5-series Servo Drive.
Encoder Phase Z Detection (encoder Z-phase detection)	60FD hex-00.16 (Digital inputs)	Shows the status of detecting the Z-phase input. For OMRON G5-series Cylinder-type Servomotors/Servo Drives with unit version 2.1 or later, set **Bit 16: Encoder Phase Z Detection** of **60FD hex-00: Digital inputs**. G5-series Linear Motor Type Servo Drives do not support this object.
Home switch (home proximity input)	60FD hex-00.2 (Digital inputs)	This signal is used for the home proximity input. Normally set Bit 2: Home switch of 60FD hex-00: Digital inputs.
External Latch Input 1	60FD hex-00.17 (Digital inputs)	Shows the status of the signal that is used for external latch input 1. Set **Bit 17: External Latch Input 1** of **60FD hex-00: Digital inputs** for an OMRON G5-series Servo Drive.
External Latch Input 2	60FD hex-00.18 (Digital inputs)	Shows the status of the signal that is used for external latch input 2. Set **Bit 18: External Latch Input 2** of **60FD hex-00: Digital inputs** for an OMRON G5-series Servo Drive.

Precautions for Correct Use

- Some functions may not be supported if you connect a unit version of the OMRON G5-series Servo Drives with built-in EtherCAT communications other than the recommended unit versions. Refer to the manual for the connected servo drive for details.
- If you change the settings, make sure that the desired operations are performed for the MC Function Module and process data settings.

Version Information

- If you are using a CPU Unit with unit version 1.10 or later, operation is as described in the following table depending on whether Modes of Operation (6060 hex) and Modes of Operation Display (6061 hex) are mapped.

	Modes of Operation Display	
	(6061 hex) mapped	**(6061 hex) not mapped**
Modes of Operation (6060 hex) mapped	• You can execute instructions that use CSP*1, CSV*2, or CST*3. • The servo is OFF in any control mode other than CSP, CSV, or CST.	• You can execute instructions that use CSP. If you execute any instruction that uses any other control mode, a Process Data Object Setting Missing error (error code 3461 hex) occurs. • The MC Function Module assumes that the CSP Servo Drive control mode is used. Command the Servo Drive to use CSP.
Modes of Operation (6060 hex) not mapped	• You can execute instructions that use CSP. If you execute any instruction that uses any other control mode, a Process Data Object Setting Missing error (error code 3461 hex) occurs. • The servo is OFF in any control mode other than CSP.	You can execute instructions that use CSP. If you execute any instruction that uses any other control mode, a Process Data Object Setting Missing error (error code 3461 hex) occurs.

*1. CSP is the Cyclic Synchronous Position Control Mode of the Servo Drive.
*2. CSV is the Cyclic Synchronous Velocity Control Mode of the Servo Drive.
*3. CST is the Cyclic Synchronous Torque Control Mode of the Servo Drive.

- If you are using a CPU Unit with unit version 1.09 or earlier and you are not using an OMRON G5-series Servo Drive with built-in EtherCAT communications for the servo axis, Modes of Operation (6060 hex) and Modes of Operation Display (6061 hex) are required.

Object Settings

The OMRON G5-series Servo Drive settings required to use the control functions of the MC Function Module are listed in the following table.
Consult the manual for your Servo Drive and set all related objects for the Servo Drive functions that you are going to use.

Index	Subindex	Name	Recommended setting	Description
3013 hex	00 hex	No. 1 External Torque Limit*1	1388 hex	Default setting = 500.0%
3015 hex*2	00 hex	Operation Switch for Using Absolute Encoder	0002 hex	Use absolute values and ignore multi-rotation counter overflow.

Index	Subindex	Name	Recommended setting	Description
3317 hex	00 hex	Speed Limit Selection	0001 hex	The velocity limit method used during torque control is either **607F hex: Max profile velocity** or **3321 hex: Velocity limit value setting**, whichever value is smaller.
3323 hex	00 hex	External Feedback Pulse Type Selection	0000, 0001, or 0002 hex*3	Set the type of external scale to use. The default value is 0000 hex (**90° Phase Difference Output**).
3324 hex*2	00 hex	External Feedback Pulse Dividing Numerator	00000000 hex	Set the encoder resolution per motor rotation [pulses]. Set to *0* for automatic setting.
3401 hex	00 hex	Input Signal Selection 2	00818181 hex	Positive Drive Prohibit Input (NC)
3402 hex	00 hex	Input Signal Selection 3	00828282 hex	Negative Drive Prohibit Input (NC)
3403 hex	00 hex	Input Signal Selection 4	00222222 hex	Home proximity input (NO)
3404 hex	00 hex	Input Signal Selection 5	002B2B2B hex	External Latch Signal 3 (NO)
3405 hex	00 hex	Input Signal Selection 6	00212121 hex	External Latch Signal 2 (NO)
3406 hex	00 hex	Input Signal Selection 7	00202020 hex	External Latch Signal 1 (NO)
3504 hex	00 hex	Drive Prohibit Input Selection	0001 hex	The drive prohibit input is disabled at the Servo. This is performed by the MC Function Module instead.
3508 hex	00 hex	Undervoltage Error Selection	0001 hex	Operation is stopped for an insufficient main power voltage.
3521 hex	00 hex	Torque Limit Selection*1	0006 hex	There are two limit values, one for positive and one for negative. Switch between them by using PCL and NCL.
3522 hex	00 hex	No. 2 External Torque Limit*1	1388 hex	Default setting = 500.0%
3703 hex	00 hex	Torque Limit Output Setting*1	0001 hex	Output turns ON for the torque limit value excluding the torque command value.
3801 hex	00 hex	Software Limit Function	0003 hex	Disable the software limits in both directions.
3758 hex	00 hex	Latch Trigger Selection	0100 hex	Touch probe1 = External latch signal 1, Touch probe2 = External latch signal 2
3759 hex	00 hex	Warning Hold Selection	0000 hex	The warnings are automatically cleared when the cause of the warning is eliminated.
607C hex	00 hex	Encoder Home Offset	00000000 hex	An offset value of *0* is used by the Servo Drive.
6091 hex	01 hex	Electronic Gear Ratio Numerator	00000001 hex	The gear ratio on the Servo Drive is 1:1. A similar function is set in the MC Function Module.
	02 hex	Electronic Gear Ratio Denominator	00000001 hex	

*1. For OMRON G5-series Linear Motor Type Servo Drives with built-in EtherCAT communications, “force” applies instead of “torque.”

*2. OMRON G5-series Linear Motor Type Servo Drives with built-in EtherCAT communications do not support this object.

*3. Select a value according to the type of external scale to use when you use fully-closed control with an OMRON G5-series Servomotor/Servo Drive, or when you use an OMRON G5-series Linear Motor Type Servomotor/Servo Drive with built-in EtherCAT communications.

A-3 Connecting to Encoder Input Terminals

This appendix describes connections to an OMRON GX-series EtherCAT Slave Encoder Input Terminals.

A-3-1 Wiring to Encoder Input Terminals

Encoder Input Terminals are connected using EtherCAT communications.

Refer to the *NJ/NX-series CPU Unit Built-in EtherCAT Port User's Manual (Cat. No. W505)* for information on the connection methods.

A-3-2 Settings for Encoder Input Terminals

This section outlines the Encoder Input Terminal settings that are used when connected to OMRON GX-series GX-EC0211/EC0241 EtherCAT Remote I/O Terminals (i.e., the applicable Encoder Input Terminals for the MC Function Module).

Refer to the *GX-series EtherCAT Slave Units User's Manual (Cat. No. W488)* for detailed information on the Encoder Input Terminals.

Recommended Encoder Input Terminals

All of the functions of an encoder axis of the MC Function Module can be used for Encoder Input Terminals with the unit versions that are given in the following table.

Manufacturer	Compatible models	Applicable unit versions
OMRON	GX-EC0211	Unit version 1.1 or later
	GX-EC0241	Unit version 1.1 or later

Additional Information

- Only the OMRON GX-EC0211/EC0241 can be used for encoder axes of EtherCAT slaves.
- Unit version 1.0 of the OMRON GX-EC0211/EC0241 can also be used for encoder axes, but they are not Sysmac devices.
  When you use unit version 1.0, do not set the node address switches to *00*. If you set them to *00*, a network configuration error occurs.
  Refer to the *GXseries EtherCAT Slave Units User's Manual (Cat. No. W488)* for detailed information on functions.
  Refer to the *NJ/NX-series CPU Unit Software User's Manual (Cat. No. W501)* for details on Sysmac devices.

External Input Signals

When all of the functions of an encoder axis are used for an Encoder Input Terminal, the following input signals are used at the Encoder Input Terminal.

- Counter A phase

- Counter B phase
- Counter Z phase
- Latch Inputs (A/B)

There are two counter channels, and there are two external latches for each channel. Wire the input signals that are required for your application.

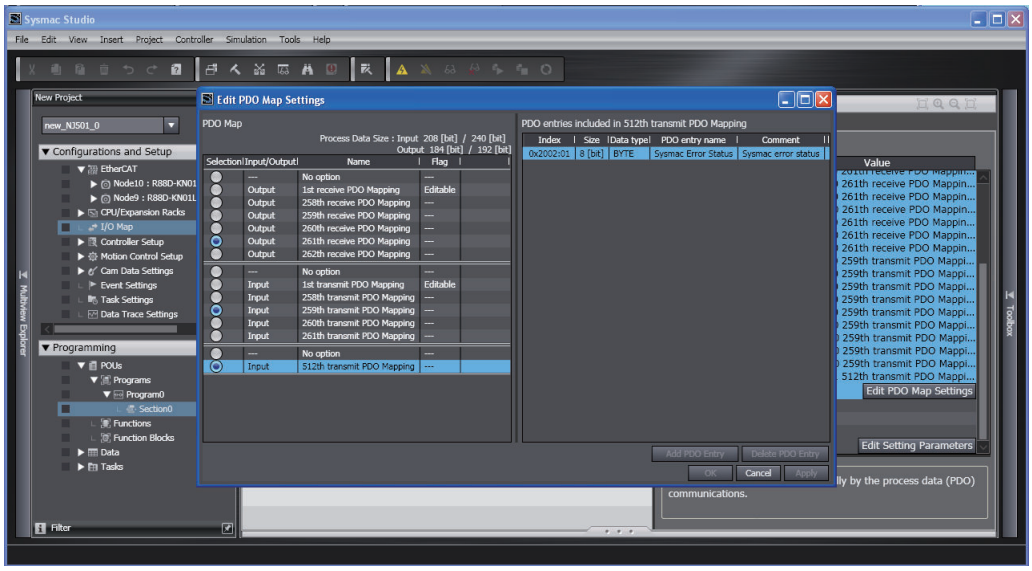
Refer to the *GX-series EtherCAT Slave Units User's Manual (Cat. No. W488)* for input signal wiring methods.

PDO Mapping

This section describes mapping PDOs to control encoder axes from the MC Function Module.

You must map the objects that are required for the motion control functions that you will use to process data communications.
The PDO map is a list of required objects that is prepared in advance.

You select the PDOs to use in the **Edit PDO Map Settings Window** of the **EtherCAT** Tab Page in the Sysmac Studio.



The following PDOs are mapped by default in the Sysmac Studio.

RxPDO (1700 hex)	Channel 1 Instruction Bits (4020 hex-01 hex) and Channel 2 Instruction Bits (4020 hex-02 hex)
RxPDO (1701 hex)	Channel 1 Preset Value (4011 hex-01 hex) and Channel 2 Preset Value (4011 hex-02 hex)
TxPDO (1B00 hex)	Channel 1 Position Value (4010 hex-01 hex) and Channel 2 Position Value (4010 hex-02 hex)
TxPDO (1B01 hex)	Channel 1 Latch Value A (4012 hex-01 hex) and Channel 2 Latch Value A (4012 hex-02 hex)
TxPDO (1B02 hex)	Channel 1 Latch Value B (4013 hex-01 hex) and Channel 2 Latch Value B (4013 hex-02 hex)
TxPDO (1B03 hex)	Channel 1 Status Bits (4030 hex-01 hex) and Channel 2 Status Bits (4030 hex-02 hex)
TxPDO (1BFF hex)	Sysmac Error Status (2002 hex-01 hex)

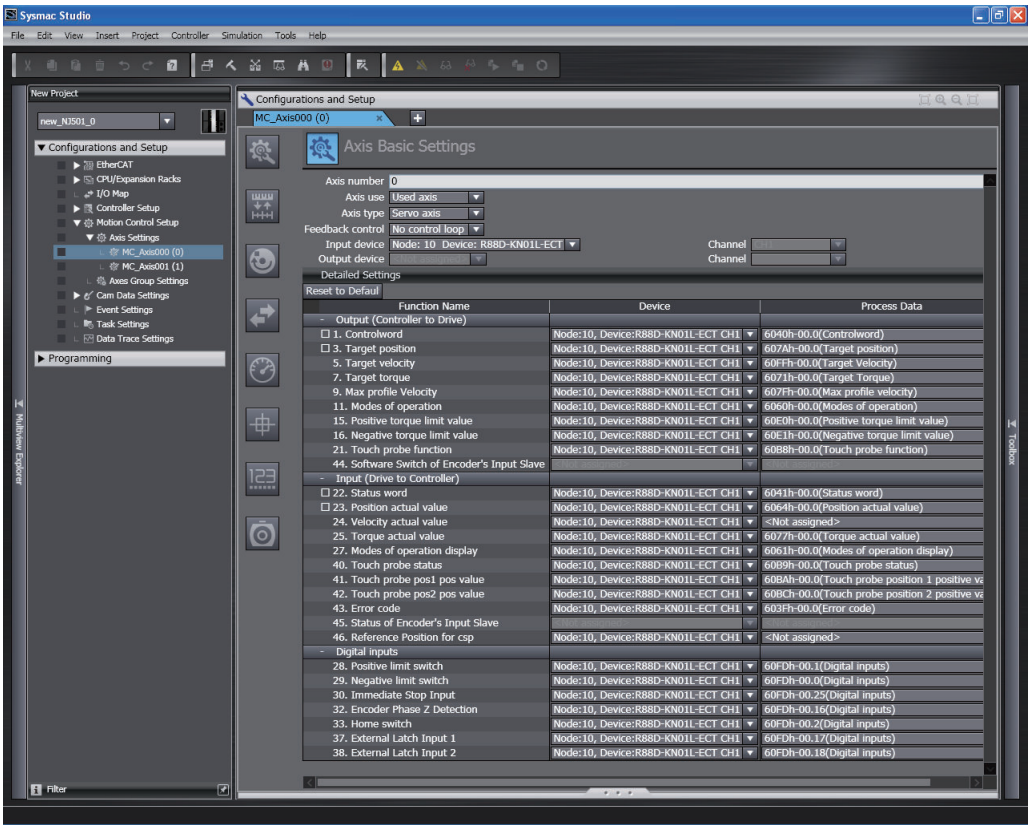
Additional Information

If you use the recommended Encoder Input Terminals (GX-EC0211/EC0241, version 1.1 or later), then it is not necessary to change the default PDO map on the Sysmac Studio.

Relationships between MC Function Module and Process Data

The functions of the MC Function Module are related to the information in the process data objects. Depending on the EtherCAT slave configuration and functions that are used by the MC Function Module, you sometimes must change the relationships between the MC Function Module and the PDOs.

To access the settings, click the **Detailed Settings** Button on the **Axis Basic Settings** Display in the Sysmac Studio.



Additional Information

If you use the recommended Encoder Input Terminals (GX-EC0211/EC0241, version 1.1 or later), then it is not necessary to change the default relationships between the functions and process data on the Sysmac Studio.

● Output Settings (Controller to Servo Drive)

The input settings apply to the command data that is sent from the MC Function Module to the Encoder Input Terminal.

The default settings in the Sysmac Studio are listed in the following table. (Required objects are marked with a circle.)

	Function name	Process data		Description
		Channel 1	Channel 2	
○	Software Switch of Encoder's Input Slave	4020 hex-01.0 (Instruction Bits)	4020 hex-02.0 (Instruction Bits)	Set the instruction bits. Set the objects given at the left for each channel.

Precautions for Correct Use

If you change the settings, make sure that the desired operations are performed for the MC Function Module and process data settings.

● Input Settings (Servo Drive to Controller)

This is the status data from the Encoder Input Terminal to the MC Function Module.
The default settings in the Sysmac Studio are listed in the following table. (Required objects are marked with a circle.)

	Function name	Process data		Description
		Channel 1	Channel 2	
○	Position actual value	4010 hex-01.0 (Position Value)	4010 hex-02.0 (Position Value)	Store the current values from the counters. Set the objects given at the left for each channel.
	Touch probe position 1 position value	4012 hex-01.0 (Latch Value A)	4012 hex-02.0 (Latch Value A)	The latched position for touch probe 1. Store the values of latch positions A. You must map these objects to use the touch probe function, i.e., to use the MC_TouchProbe (Enable External Latch) instruction. Set the objects given at the left for each channel.
	Touch probe position 2 position value	4013 hex-01.0 (Latch Value B)	4013 hex-02.0 (Latch Value B)	The latched position for touch probe 2. Store the values of latch positions B. You must map these objects to use the touch probe function, i.e., to use the MC_TouchProbe (Enable External Latch) instruction. Set the objects given at the left for each channel.
	Status of Encoder's Input Slave	4030 hex-01.0 (Status Bits)	4030 hex-02.0 (Status Bits)	Store the status bits. You must map these objects to use the touch probe function, i.e., to use the MC_TouchProbe (Enable External Latch) instruction. Set the objects given at the left for each channel.

Precautions for Correct Use

If you change the settings, make sure that the desired operations are performed for the MC Function Module and process data settings.

● Digital Input Settings

Settings are not required to use an encoder axis.

Object Settings in the Encoder Input Terminals

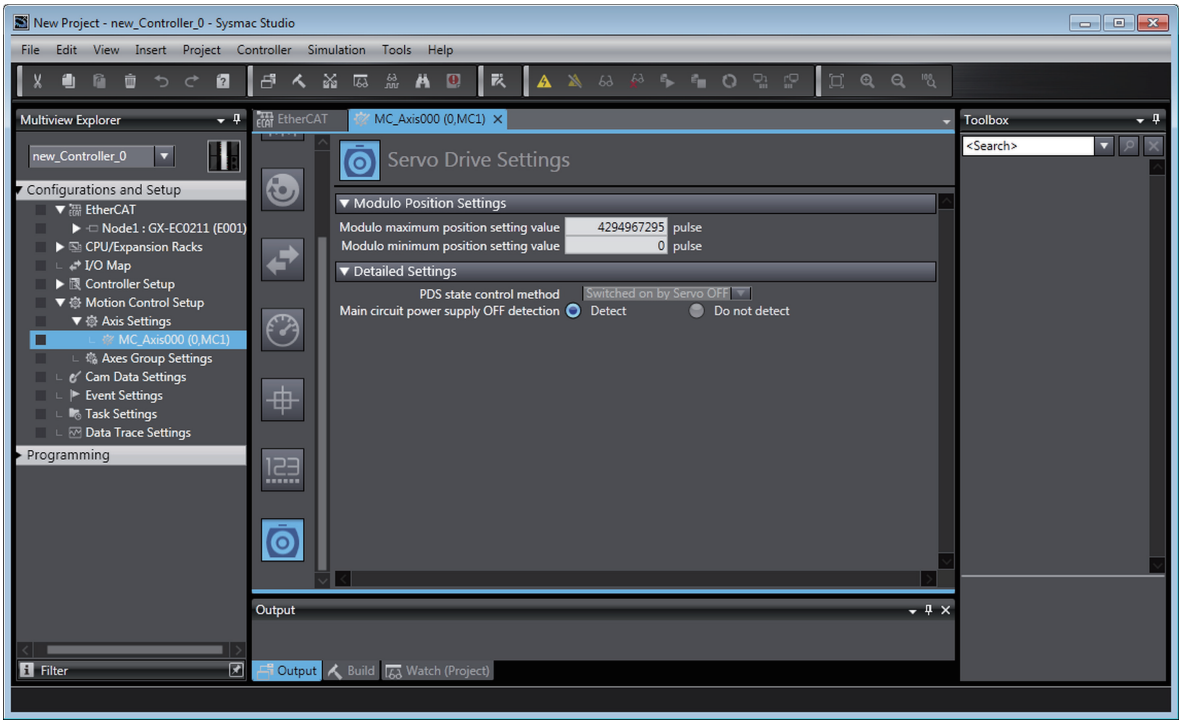
There are no objects that you must set at the Encoder Input Terminal.

Relationship between the MC Function Module and the Ring Counter of an Encoder Input Terminal

The Modulo Minimum Position Setting Value and Modulo Maximum Position Setting Value in the Servo Drive Settings in the axis parameters of the MC Function Module must agree with the maximum value setting of the ring counter in the Encoder Input Terminal.

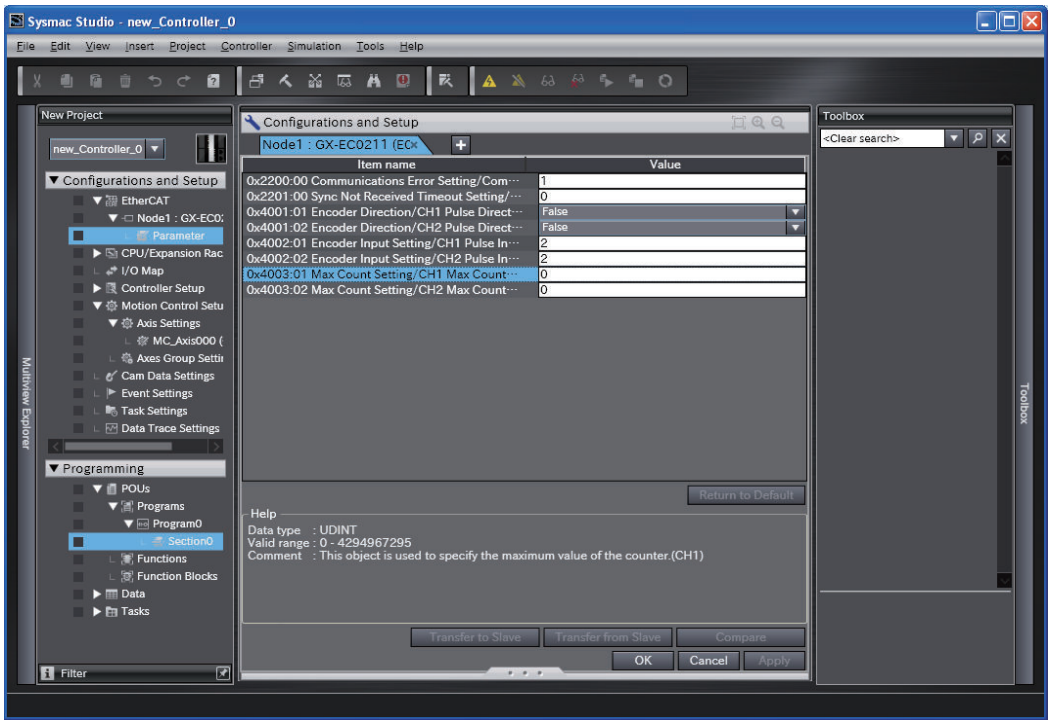
The Modulo Minimum Position Setting Value and Modulo Maximum Position Setting Value are set on the **Servo Drive Settings** View on the Sysmac Studio.
The settings are as follows:

Parameter	Meaning	Set value
Modulo Maximum Position Setting Value	Set the modulo maximum position that is set on the Servo Drive or the Encoder Input Terminal.	This setting must agree with the maximum value that is set for the ring counter in the Encoder Input Terminal.
Modulo Minimum Position Setting Value	Set the modulo minimum position that is set on the Servo Drive or the Encoder Input Terminal.	Set this parameter to *0*.



The maximum value of the ring counter for the Encoder Input Terminal is set on the **EtherCAT** Tab Page in the Sysmac Studio.
The setting is as follows:

Index	Object name	Set value
0x4003	Max Count Setting (maximum value of the ring counter)	Set this parameter to the same value as the Modulo Maximum Position Setting Value in the Servo Drive Settings of the axis parameters of the MC Function Module.



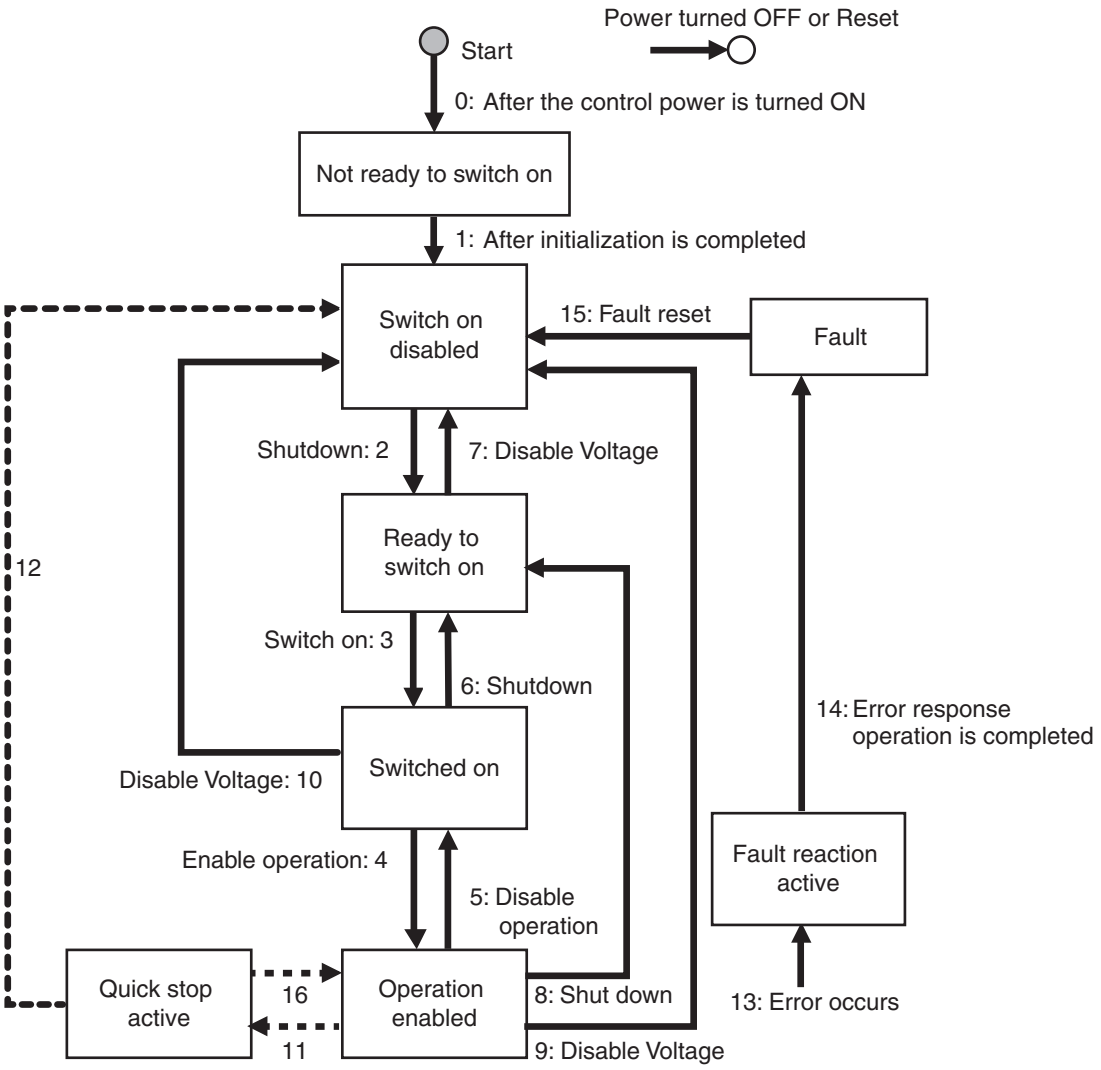
A-4 Connecting to NX Units

Refer to the *NX-series Position Interface Units User's Manual (Cat. No. W524)* for information on connecting to the NX-series Position Interface Units.

A-5 PDS State Transition

The PDS state transition is defined in CiA402 drive profile. Use the Controlword (6040 hex) process data to command PDS state transitions. To check actual PDS states, use the Statusword (6041 hex) process data.

The following diagram shows the state transition defined in CiA402 drive profile.
Each box indicates a state, while numbers indicate the state control commands.



A-5-1 PDS State Control Method

This section describes the relationship between the setting values of the **PDS State Control Method** axis parameter and the PDS states.

● When PDS State Control Method Is Set to 0

The following operation is performed when **PDS State Control Method** is set to **0: Switched on by Servo OFF**.

- After communications with the Servo Drive is established, MC Function Module automatically changes the PDS state to Servo Ready (Switched on).
- If *Enable* of the MC_Power (Power Servo) instruction changes to TRUE in the Servo Ready (Switched on) state, the PDS state changes to Servo ON (Operation enabled). This operation corresponds to **4: Enable operation** in the state transition diagram.
- If *Enable* of the MC_Power (Power Servo) instruction changes to FALSE in the Servo ON (Operation enabled) state, the PDS state changes to Servo Ready (Switched on). This operation corresponds to **5: Disable operation** in the state transition diagram.

● When PDS State Control Method Is Set to 1

The following operation is performed when **PDS State Control Method** is set to **1: Ready to switched on by Servo OFF**.

- After communications with the Servo Drive is established, MC Function Module automatically changes the PDS state to Main Power OFF (Ready to switch on).
- If *Enable* of the MC_Power (Power Servo) instruction changes to TRUE in the Main Power OFF (Ready to switch on) state, the PDS state changes to Servo ON (Operation enabled). This operation corresponds to **3: Switch on** and **4: Enable operation** in the state transition diagram. The steps 3 and 4 are performed simultaneously.
- If *Enable* of the MC_Power (Power Servo) instruction changes to FALSE in the Servo ON (Operation enabled) state, the PDS state changes to Main Power OFF (Ready to switch on). This operation corresponds to **5: Disable operation** and **6: Shutdown** in the state transition diagram. The steps 5 and 6 are performed simultaneously.

Version Information

- For a CPU Unit with unit version 1.10 or later, **PDS State Control Method** is selectable. **PDS State Control Method** is set to *0* by default.
- For a CPU Unit with unit version 1.09 or earlier, **PDS State Control Method** is set to *0*.

A-5-2 Main Circuit Power Supply OFF Detection

You can select whether to detect the OFF state of the main circuit power supply while the Servo is ON. Select the **Do not detect** option if a Servo Main Circuit Power OFF error occurs even after the MC_Power (Power Servo) instruction is executed.

Precautions for Correct Use

You cannot select the **Do not detect** option when you use an OMRON 1S-series Servo Drive or G5-series Servo Drive.
A Servo Main Circuit Power OFF error will occur if you select the **Do not detect** option and turn off the main power supply to Servo Drive when the Servo is ON.

Version Information

- For a CPU Unit with unit version 1.10 or later, **Main circuit power supply OFF detection** is selectable. **Main circuit power supply OFF detection** is set to **Detect** by default.
- For a CPU Unit with unit version 1.09 or earlier, **Main circuit power supply OFF detection** is set to **Detect**.

A-6 Terminology

This appendix provides definitions of terms related to motion control.

A-6-1 NJ/NX-series Controller

Term	Description
main memory	The memory inside the CPU Unit that is used by the CPU Unit to execute the OS and user program.
periodic task	Tasks for which user program execution and I/O refreshing are performed each period.
primary periodic task	The task with the highest priority.
task period	The interval at which the primary periodic task or a periodic task is executed.
I/O refreshing	Cyclic data exchange with external devices that is performed with predetermined memory addresses.
program	One of three POUs. The others are functions and function blocks. Programs are assigned to tasks to execute them.
user program	All of the programs in one project.
Inline ST	ST programming that is included within a ladder diagram program.
system-defined variable	A variable for which all attributes are defined by the system and cannot be changed by the user.
global variable	Reading and writing global variables are possible from all POUs (programs, functions, and function blocks).
local variable	A variable that can be accessed only from the POU in which it is defined. Local variables include internal variables, input variables, output variables, in-out variables, and external variables.
download	To transfer data from the Sysmac Studio to the Controller with the synchronization operation of the Sysmac Studio.
upload	To transfer data from the Controller to the Sysmac Studio with the synchronization operation of the Sysmac Studio.
major fault level Controller error	An error for which all control operations from the NJ/NX-series Controller are not possible. The CPU Unit stops user program execution immediately and turns OFF the loads for all slaves and Units (including remote I/O).
partial fault level Controller error	An error for which all control operations for one of the function modules in the NJ/NX-series Controller are stopped. The NJ/NX-series CPU Unit continues operation even after a partial fault level Controller error occurs.
minor fault level Controller error	An error for which some of the control operations for one of the function modules in the NJ/NX-series Controller are stopped. The NJ/NX-series CPU Unit continues operation even after a minor fault level Controller error occurs.
observation	One of the event levels for Controller information and user-defined information. Observations represent minor errors that do not affect control operations. They are recorded in an event log to inform the user.

A-6-2 Motion Control

Term	Description
used real axis	Axis of which axis type is set to **Servo Axis** or **Encoder Axis** and axis use is set to **Used Axis**.
used virtual axis	Axis of which axis type is set to **Virtual Servo Axis** or **Virtual Encoder Axis** and axis use is set to **Used Axis**.
Motion Control Function Module	A software component that executes motion control. It performs motion control based on commands from the motion control instructions that are executed in the user program. (Abbreviation: MC Function Module)
motion control instruction	An instruction that is defined as a function block to execute a motion control function. The MC Function Module supports instructions that are based on function blocks for PLCopen® motion control as well as instructions developed specifically for the MC Function Module.
motion control axes	Axes for which all motion control functions can be used.
single-axis position control axes	Axes for which only single-axis position control can be used.
single-axis position control	Controlling the position of one axis.
single-axis velocity control	Controlling the velocity of one axis. For single-axis velocity control, the MC Function Module sometimes outputs velocity commands to the Servo Drive and sometimes outputs position commands to the Servo Drive.
single-axis torque control	Controlling the torque of one axis.
single-axis synchronized control	Synchronizing the control of one slave axis with one master axis. There are two types of single-axis synchronized control: gear operation, in which the axes are synchronized with a gear ratio, and cam operation, in which the axes are synchronized according to the relationship between phases and displacements in a cam table.
single-axis manual operation	Controlling an axis with manual operation, such as jogging.
auxiliary functions for single-axis control	Functions that aid in controlling an axis, such as override factor settings and resetting errors.
multi-axes coordinated control	Controlling the motion of more than one axis, such as linear interpolation and circular interpolation. You specify an axes group to specify the axes to coordinate.
auxiliary functions for multi-axes coordinated control	Functions that aid in controlling an axes group, such as override factor settings and resetting errors.
motion control parameters	Parameters that define the operation of the MC Function Module. The motion control parameters include the MC common parameters, axis parameters, and axes group parameters.
axis parameters	Parameters that apply to a single axis.
axes group parameters	Parameters that apply to an axes group.
system-defined variables for motion control	System-defined variables that provide status information for the MC Function Module. The system-defined variables for motion control include the MC Common Variable, Axis Variables, and Axes Group Variables.
MC common variable	A system-defined variable that is defined as a structure and provides status information for the overall operation of the MC Function Module.
axis variables	System-defined variables that are defined as structures and provide status information and some of the axis parameters for individual axes.
axes group variables	System-defined variables that are defined as structures and provide status information and some of the axes group parameters for individual axes groups.

Term	Description
homing	The process of defining home. Homing is also called home positioning, home searching, calibration, and datum.
home	The zero position of the mechanical system. Home is determined by the home input signal during the homing operation.
zero position	The position that is based on home and is treated as the zero position in the user program. This is the same position as home if the home position is not changed.
following error	The difference between the command current position and actual current position. There is a following error only in position control mode. (Other modes do not have a command current position.) The following error is also called the following error counter value and the remaining pulses.
following error reset	Setting the following error to zero.
cam profile curve	A curve that shows the relationship between phases and displacements in a cam operation. The cam profile curve is created on the Sysmac Studio.
cam data	Data made up of phases (master axis) and displacements (slave axis) for cam operation.
cam data variable	A structure array variable for cam data. It contains phases and displacements and is defined as a structure array.
cam table	A data table that contains cam data. Use the Sysmac Studio to download the cam profile curves that you created with the Cam Editor to the CPU Unit to save them as cam tables in the non-volatile memory in the CPU Unit.
override	A function that allows the operator to temporarily change programmed values during operation.
jerk	The rate of change in the acceleration or deceleration rate. If you specify the jerk, the velocity graph will form an S-curve for acceleration and deceleration. Jerk is also called jolt, surge, and lurch.
Multi-motion	Performing the motion controls in parallel using both of the primary periodic task and the priority-5 periodic task.

A-6-3 EtherCAT Communications

Term	Description
CAN application protocol over EtherCAT (CoE)	A CAN application protocol service implemented on EtherCAT.
CAN in Automation (CiA)	CiA is the international users' and manufacturers' group that develops and supports higher-layer protocols.
EtherCAT Technology Group	The ETG is a global organization in which OEM, End Users and Technology Providers join forces to support and promote the further technology development.
Object	An abstract representation of a particular component within a device, which consists of data, parameters, and methods.
Object Dictionary	A data structure addressed by Index and Subindex that contains description of data type objects, communications objects and application objects.
Process Data	Collection of application objects designated to be transferred cyclically or acyclically for the purpose of measurement and control.

Term	Description
Process Data Object	A process data (I/O data) object that exchanges data at regular periods with CoE.
Service Data Object	CoE asynchronous mailbox communications where all objects in the object dictionary can be read and written.
Receive PDO	A process data object received by an EtherCAT slave.
Transmit PDO	A process data object sent from an EtherCAT slave.
Device Profile	A collection of device dependent information and functionality providing consistency between similar devices of the same device type.

A-7 Version Information

This section describes the functions that are supported for each unit version.

Refer to the *NJ/NX-series CPU Unit Software User's Manual (Cat. No. W501)* for the relationship between the unit versions of CPU Units and the Sysmac Studio versions.

Refer to the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)* for version information on the motion control instructions.

Motion Control Functions That Were Added for Unit Version 1.01

Version 1.02 or higher of the Sysmac Studio is required to use the functions that were added for unit version 1.01 of the CPU Unit.

Function	Overview
Changing the axes in an axes group	You can use the MC_ChangeAxesInGroup (Change Axes in Group) instruction to temporarily change the **Composition Axes** axes group parameter for an axes group.
Reading axes group positions	You can use the MC_GroupReadPosition (Read Axes Group Position) instruction to read the command current positions and the actual current positions of an axes group.
Axes group cyclic synchronous absolute positioning	You can use the MC_GroupSyncMoveAboslute (Axes Group Cyclic Synchronous Absolute Positioning) instruction to cyclically output the specified target positions for the axes.
Controllable Servo Drives	Support was added for OMRON G5-series Linear Motor Type Servomotors/Servo Drives with built-in EtherCAT communications.

Motion Control Functions That Were Added for Unit Version 1.02

Version 1.03 or higher of the Sysmac Studio is required to use the functions that were added for unit version 1.02 of the CPU Unit.

No motion control functions were added for unit version 1.02, but the specifications of some instructions were improved. Refer to the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)* for version information on the motion control instructions.

Motion Control Functions That Were Added for Unit Version 1.03

Version 1.04 or higher of the Sysmac Studio is required to use the functions that were added for unit version 1.03 of the CPU Unit.

Function	Overview
Cyclic synchronous absolute positioning	The MC_SyncMoveAbsolute (Cyclic Synchronous Absolute Positioning) instruction can be used to output a command position each control period in Position Control Mode.

Function	Overview
Homing with parameters	The MC_HomeWithParameter (Home with Parameters) instruction can be used to specify the homing parameters and operate the motor to determine home. It uses the limit signals, home proximity signal, and home signal.

Motion Control Functions That Were Added for Unit Version 1.04

Version 1.05 or higher of the Sysmac Studio is required to use the functions that were added for unit version 1.04 of the CPU Unit.

Function	Overview
Changing axis use	You can use the MC_ChangeAxisUse (Change Axis Use) instruction to temporarily change the setting of the **Axis Use** axis parameter.

Motion Control Functions That Were Added for Unit Version 1.05

Version 1.06 or higher of the Sysmac Studio is required to use the functions that were added for unit version 1.05 of the CPU Unit.

Function	Overview
Start velocity	You can set the initial velocity when axis motion starts.
Input signal logic inversion	You can inverse the logic of immediate stop input signal, positive limit input signal, negative limit input signal, or home proximity input signal.

Note With a CPU Unit with unit version 1.05 or later, you can perform motion control by assigning axes to NX-series Position Interface Units.

Motion Control Specifications That Were Added or Changed for Unit Version 1.06

Version 1.07 or higher of the Sysmac Studio is required to use the performance specifications and function specifications that were added or changed for unit version 1.06 of the CPU Unit.

Function	Overview
Maximum number of controlled axes	The maximum number of controlled axes for NJ301-□□□□ was increased to 15 axes. (No change in maximum number of used real axes.)
Maximum number of axes for single-axis control	The maximum number of axes for single-axis control for NJ301-□□□□ was increased to 15 axes. (No change in maximum number of used real axes.)
Enable digital cam switch	You can use the MC_DigitalCamSwitch (Enable Digital Cam Switch) instruction to turn the digital outputs ON or OFF according to the axis position.
Time stamp to axis position calculation	You can use the MC_TimeStampToPos (Time Stamp to Axis Position Calculation) instruction to calculate the position of the axis for the specified time stamp.
Adding blending options to Start Cam Operation	The blending options were added to Buffer Mode Selection for the MC_CamIn (Start Cam Operation) instruction.
_sMC_POSITION_REF	You can use this data type to display the path of user coordinate systems in 3D Motion Monitor Display Mode.

Motion Control Functions That Were Added or Changed for Unit Version 1.08

Version 1.09 or higher of the Sysmac Studio is required to use the functions that were added or changed for unit version 1.08 of the CPU Unit.

Function	Overview
Generating cam tables	You can use the MC_GenerateCamTable (Generate Cam Table) instruction and generate the cam table according to the cam property and cam node specified for the input parameters.
Changing axis parameters	You can access and change the axis parameters from the MC_WriteAxisParameter (Write Axis Parameters) instruction and the MC_ReadAxisParameter (Read Axis Parameters) instruction in the user program.
Assigning device variables	You can assign the device variables to the I/O ports of slaves and Units that are assigned to the Axis Variables. *1 Refer to *2-4-2 Relationship between EtherCAT Master Function Module and MC Function Module* on page 2-19 for details.

*1. This function is independent of the version of the CPU Unit. Using the Sysmac Studio version 1.09 or higher allows you use this assignment function.

Motion Control Functions That Were Added or Changed for Unit Version 1.09

Version 1.10 or higher of the Sysmac Studio is required to use the functions that were added for unit version 1.09 of the CPU Unit.

No motion control functions were added for unit version 1.09, but the specifications of some instructions were improved. Refer to the *NJ/NX-series Motion Control Instructions Reference Manual (Cat. No. W508)* for version information on the motion control instructions.

Motion Control Functions That Were Added or Changed for Unit Version 1.10

Version 1.12 or higher of the Sysmac Studio is required to use the following functions that were added for unit version 1.10 of the CPU Unit.

Function	Overview
Slave axis position compensation	This function compensates the position of the slave axis currently in synchronized control.
Change in the blending operation	The maximum acceleration/deceleration rate is used and the blending operation is continued even if you set the **Acceleration/Deceleration Over** parameter to **Use rapid acceleration/deceleration (Blending is changed to Buffered)** or **Minor fault stop**.
Home definition timing for absolute encoders	In addition to the existing home definition timing, home is also defined when EtherCAT slave process data communications change from a non-established to an established state.
Current position when process data communications are in a non-established state	The actual current position and the command current position axis variables will contain the actual current position output that is just before process data communications change to a non-established state.

Function	Overview
With or without Modes of Operation (6060 hex) and Modes of Operation Display (6061 hex) mapping	The operation depends on whether Modes of Operation (6060 hex) and Modes of Operation Display (6061 hex) are mapped.
PDS state control method	You can set the state to which PDS state changes when Servo is turned OFF by the MC_Power instruction.
Main circuit power supply OFF detection	You can select whether to detect the OFF state of the main circuit power supply while the Servo is ON or when the Servo ON command is sent.

Version 1.13 or higher of the Sysmac Studio is required to use the following functions that were added for unit version 1.10 of the CPU Unit.

Function	Overview
Synchronized control in multi-motion	You can execute the synchronized control instructions between axes assigned to different tasks in the multi-motion.

Motion Control Functions That Were Added or Changed for Unit Version 1.11

Version 1.15 or higher of the Sysmac Studio is required to use the following functions that were added for unit version 1.11 of the CPU Unit.

Function	Overview
Adding the parameters for the unit conversion settings	The following parameters were added: **Reducer Use**, **Work Travel Distance Per Rotation**, **Work Gear Ratio**, and **Motor Gear Ratio**.

Version 1.16 or higher of the Sysmac Studio is required to use the following functions that were added for unit version 1.11 of the CPU Unit.

Function	Overview
Controllable Servo Drives	Support was added for OMRON 1S-series Servo Drives with built-in EtherCAT communications.

Motion Control Functions That Were Added or Changed for Unit Version 1.13

Version 1.17 or higher of the Sysmac Studio is required to use the following functions that were added for unit version 1.13 of the CPU Unit.

Function	Overview
Control Function	Select the function of axis to control.
Commands to I/O devices during the download process	The MC Function Module operation differs depending on the setting that specifies to stop or continue sending commands to the I/O devices when the download process starts.

Motion Control Functions That Were Added for Unit Version 1.21 or 1.32

The specified unit version of the CPU Unit and version 1.28 or higher of the Sysmac Studio are required to use the functions described below.

CPU Unit	Unit version
NX102 CPU Unit	Version 1.32 or later
NX1P2 CPU Unit	Version 1.21 or later
NJ-series CPU Unit	Version 1.21 or later

Function	Overview
Cam Monitor	You can use the MC_CamMonitor (Cam Monitor) instruction to monitor the phase and displacement of the cam table based on the current master axis position and the position and displacement of the current slave axis.
Slave axis position compensation	You can use the MC_OffsetPosition (Position Offset Compensation) to add the specified position offset with the acceleration/deceleration curve to the slave axis currently in synchronized control and compensates the command position.
Controllable Servo Drives	Support was added for OMRON 1S-series Servo Drives with built-in EtherCAT Communications and Safety Functionality.

Motion Control Function That Was Added for Unit Version 1.36 or 1.69

The specified unit version of the CPU Unit and version 1.63 or higher of the Sysmac Studio are required to use the function described below.

CPU Unit	Unit version
NX102 CPU Unit	Version 1.69 or later
NX1P2 CPU Unit	Version 1.69 or later
NX502 CPU Unit	Version 1.69 or later
NJ-series CPU Unit	Version 1.69 or later
NX701 CPU Unit	Version 1.36 or later

Function	Overview
Setting the cam table end point index	The MC_SetCamTableEndPointIndex instruction is provided as an alternative to the MC_SetCamTableProperty instruction. Although the operation performed on the cam table and the result obtained from it are the same as those of the MC_SetCamTableProperty instruction, it allows you to adjust the task period required from the execution to completion of the instruction.

Motion Control Functions That Were Added for Unit Version 1.37 or 1.70

The specified unit version of the CPU Unit and version 1.66 or higher of the Sysmac Studio are required to use the functions described below.

CPU Unit	Unit version
NX102 CPU Unit	Version 1.70 or later
NX1P2 CPU Unit	Version 1.70 or later
NX502 CPU Unit	Version 1.70 or later
NJ-series CPU Unit	Version 1.70 or later

CPU Unit	Unit version
NX701 CPU Unit	Version 1.37 or later

Function	Description
Cyclic Synchronous Torque Control	You can use the MC_SyncTorqueControl (Cyclic Synchronous Torque Control) instruction to output the command torque for each control period in Torque Control Mode.
Zero position preset for encoder axes	This is a functional extension to enable zero position preset for encoder axes.
Positioning gear operation2	The MC_GearInPos2 (Positioning Gear Operation2) instruction has been added as a functional extension to positioning gear operation to improve takt time.
Set torque limit 2	The MC_SetTorqueLimit2 (Set Torque Limit 2) instruction has been added as a functional extension to change the torque limit method at Servo OFF.
Starting cam operation with specified curve	The MC_CamInCurve (Start Cam Operation With Specified Curve) instruction executes cam operation based on a specified curve. The MC_CamMonitorCurve (Cam Monitor With Specified Curve) instruction monitors the cam operation started by the MC_CamInCurve (Start Cam Operation With Specified Curve) instruction.

I

Index

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

Z

OMRON Corporation **Industrial Automation Company**

Kyoto, 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
2895 Greenspoint Parkway, Suite 200
Hoffman Estates, IL 60169 U.S.A.
Tel: (1) 847-843-7900 Fax: (1) 847-843-7787

OMRON ASIA PACIFIC PTE. LTD.
438B Alexandra Road, #08-01/02 Alexandra
Technopark, Singapore 119968
Tel: (65) 6835-3011 Fax: (65) 6835-3011

OMRON (CHINA) CO., LTD.
Room 2211, Bank of China Tower,
200 Yin Cheng Zhong Road,
PuDong New Area, Shanghai, 200120, China
Tel: (86) 21-6023-0333 Fax: (86) 21-5037-2388

Authorized Distributor:

Cat. No. W507-E1-32 0426