MITSUBISHI ELECTRIC
Changes for the Better

FATEC

Mitsubishi Programmable Controllers Training Manual MELSEC iQ-R Motion Module

SAFETY PRECAUTIONS

(Read these precautions before using this product.)

⚠ **WARNING**	Indicates that incorrect handling may cause hazardous conditions, resulting in death or severe injury.
⚠ **CAUTION**	Indicates that incorrect handling may cause hazardous conditions, resulting in minor or moderate injury or property damage.

When designing a system, always read the relevant manuals and pay full attention to safety.
During the exercise, pay full attention to the following and handle the product correctly.

[EXERCISE PRECAUTIONS]

⚠ WARNING

- To prevent an electric shock, do not touch any terminal while the power is on.
- Before opening the safety cover, turn off the power or ensure the safety.

⚠ CAUTION

- Follow the instructor's directions during the exercise.
- Do not remove the modules from the demonstration machine or change the wiring without permission.
  Doing so may result in failure, malfunction, injury, and/or fire.
- Turn off the power before mounting or removing a module.
  Failure to do so may result in malfunction of the module or electric shock.
- If demonstration machine emits unusual odor or sound, press the "Power switch" or "Emergency switch" to turn off the machine.
- If a problem occurs, notify the instructor immediately.

CONTENTS

APPENDICES 186

INTRODUCTION

This text provides information about hardware used in the system construction and how to make a program in ST language to help users acquire the knowledge necessary for 1-axis control and multi-axis control using the MELSEC iQ-R series Motion module.

RELEVANT MANUALS

Manual name [manual number]	Description	Available form
MELSEC iQ-R Motion Module User's Manual (Startup) [IB-0300406ENG]	Specifications, procedures before operation, system configuration, and wiring of the Motion module	e-Manual PDF
MELSEC iQ-R Motion Module User's Manual (Application) [IB-0300411ENG]	Functions, I/O signals, variables, labels, programming, and troubleshooting of the Motion module	e-Manual PDF
MELSEC iQ-R Motion Module User's Manual (Network) [IB-0300426ENG]	Functions, parameter settings, troubleshooting, and buffer memory of CC-Link IE TSN	e-Manual PDF
MELSEC iQ-R Programming Manual (Motion Module Instructions, Standard Functions/Function Blocks) [IB-0300431ENG]	Instructions for the Motion module and standard functions/function blocks	e-Manual PDF
MELSEC iQ-R Programming Manual (Motion Control Function Blocks) [IB-0300533ENG]	Motion control function blocks, variables, and programming	e-Manual PDF
MELSEC iQ-R Programming Manual (Program Design) [SH-081265ENG]	Specifications and labels of ladder, ST, and other programs	e-Manual PDF
MELSEC iQ-R/MELSEC iQ-F Structured Text (ST) Programming Guide Book [SH-081483ENG]	Programming using Structured Text (ST) in GX Works3 Fundamental operations and functions explained using sample programs	e-Manual PDF
GX Works3 Operating Manual [SH-081215ENG]	System configuration, parameter settings, and online operations of GX Works3	e-Manual PDF
MR-J5-G/MR-J5W-G User's Manual (Introduction) [SH-030294ENG]	Specifications, functions, configuration, startup, maintenance, and compliance with global standards of the servo amplifier	e-Manual PDF
MR-J5 User's Manual (Function) [SH-030300ENG]	Usage of each function to operate a servo amplifier	e-Manual PDF
MR-J5 User's Manual (Troubleshooting) [SH-030312ENG]	Alarms and warnings of the servo amplifier	e-Manual PDF

Point

e-Manual refers to the Mitsubishi Electric FA electronic book manuals that can be browsed using a dedicated tool.

e-Manual has the following features:

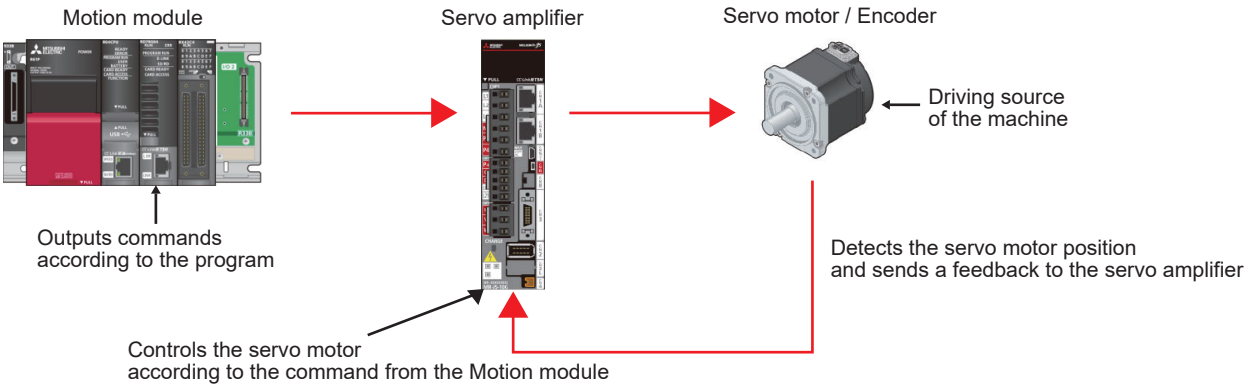
- Required information can be cross-searched in multiple manuals.
- Other manuals can be accessed from the links in the manual.
- The hardware specifications of each part can be found from the product figures.
- Pages that users often browse can be bookmarked.
- Sample programs can be copied to an engineering tool.

1 OVERVIEW OF MOTION MODULE

1.1 Servo System

A servo system is a control system that is configured to track changes in the controlled object by using control variables such as the position, orientation, and attitude of the object.

The Mitsubishi Electric servo system refers to a system that controls servo motors by transmitting position and velocity commands from the programmable controller and Motion module, which serve as the controller, to the servo amplifier. Servo motors have an encoder that can detect their position and velocity and use this information to perform feedback control so that the deviation between the position or velocity command and actual position or velocity is minimized.



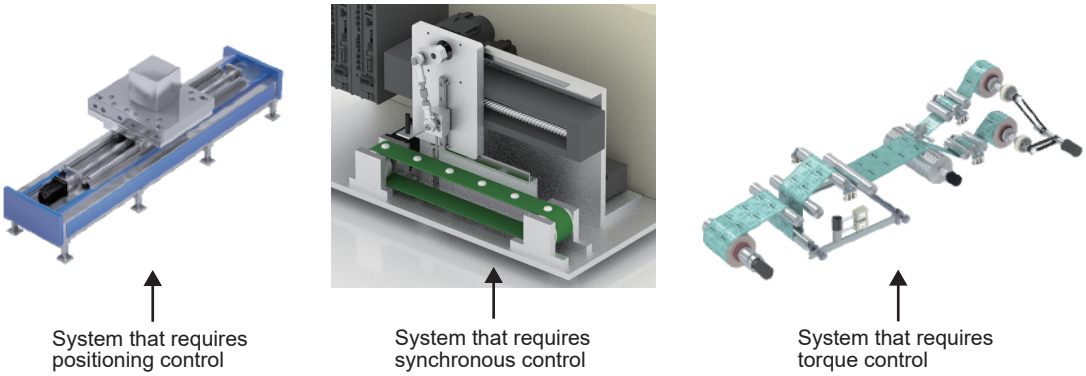
Component

A servo system requires the following devices.

Device	Description
Motion module	Performs various types of motion control such as positioning, synchronization, cam, velocity, and torque control. Sends information necessary for motion control to the servo amplifier and stepping motor driver as commands.
Servo amplifier	Transmits the direction, speed, and amount of rotation to the motor based on the commands from the Motion module.
Servo motor / Encoder	Rotates at a specified speed in the specified direction and stops at the specified position in accordance with the command from the servo amplifier.

Application examples of servo system

Servo systems are widely used in various applications, including machines that require high positioning accuracy, machines that require advanced synchronization such as filling machines, and machines that require tension control (torque control) or speed control such as winding/unwinding machines.

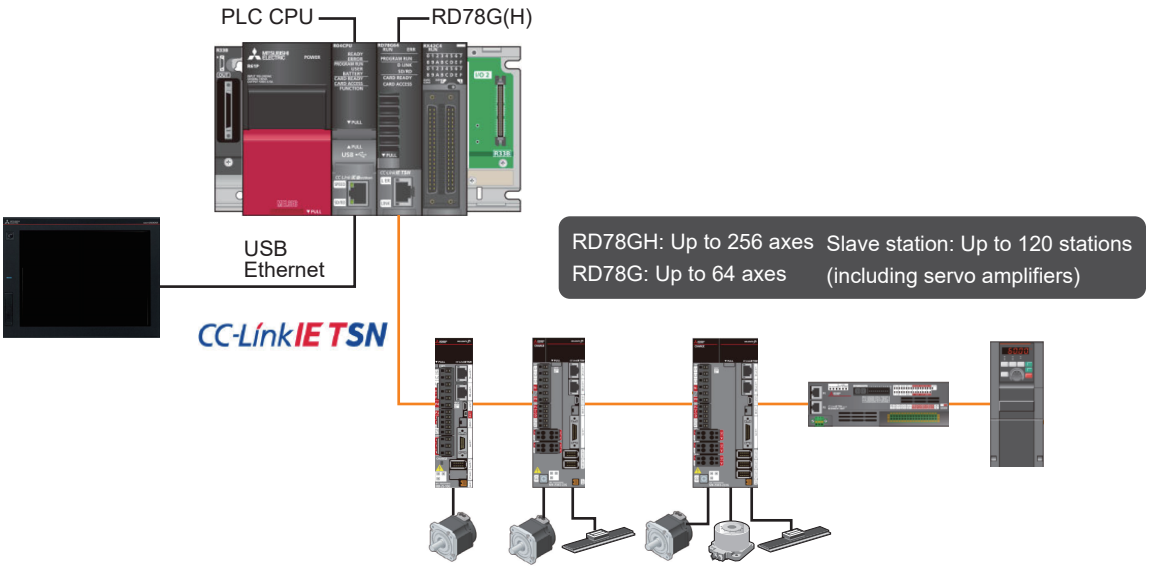


Roles of Motion modules

Motion modules, which output commands for the servo system, not only output position and velocity commands to the servo amplifier, but also generates commands for interpolation and synchronization control using multiple axes.
Another role of Motion modules is to change position and velocity commands in accordance with the sensor values managed by the programmable controller or other means.

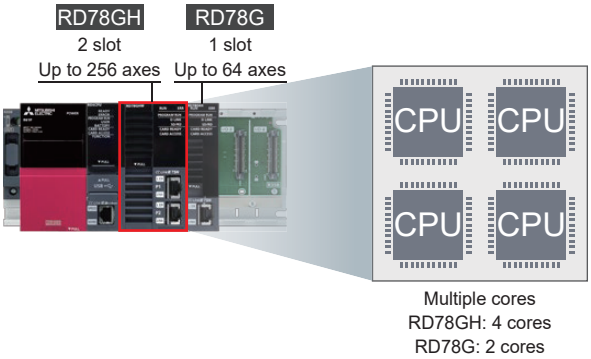
Characteristics of MELSEC iQ-R series Motion module RD78G(H)

MELSEC iQ-R series Motion module RD78G(H) is compatible with CC-Link IE TSN.
While ensuring real-time control performance required for motion control, the Motion module supports CC-Link IE TSN that allows for coexistence with communications with IT systems to achieve connectivity with various devices such as I/O modules and high-speed counter modules as well as servo amplifiers. This allows flexible system configuration using various devices.



High -speed and high-precision Motion modules

RD78G is capable of positioning and synchronization control, and RD78GH is designed for high-performance motion control.
The Motion module performs motion control and PLC CPU performs machine control, thereby achieving load balancing.

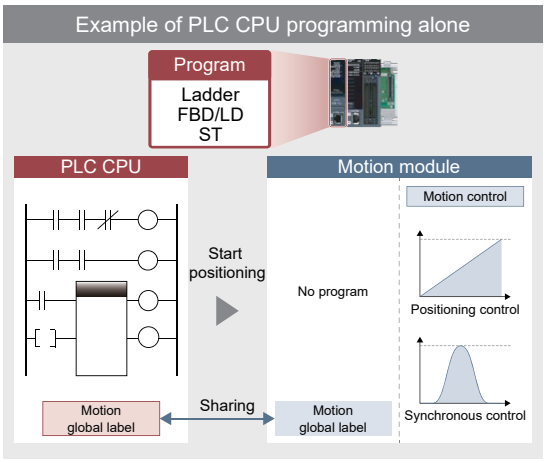


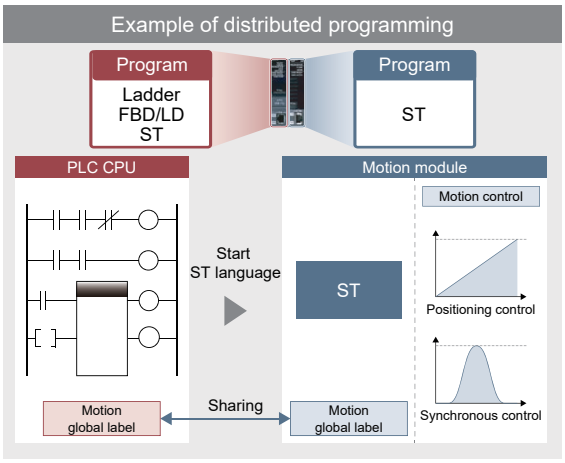
Simple programming for motion control

The PLCopen® Motion Control Function Block (FB) library, which is compliant with an international standard, can be used for programming.

The Motion modules can be programmed in ST (Structured Text) language, and the PLC CPU can be programmed in ladder, FBD/LD, and ST language. Created programs can be written to the PLC CPU and/or Motion module.

Which module(s) to be programmed can be determined to match the needs for high-speed control, complex operations, and other tasks.





■ Example of PLC CPU programming alone

- For users who started with ladder, FBD/LD, and ST language
- Only the PLC CPU needs to be programmed, thereby reducing the engineering effort.

■ Example of distributed programming

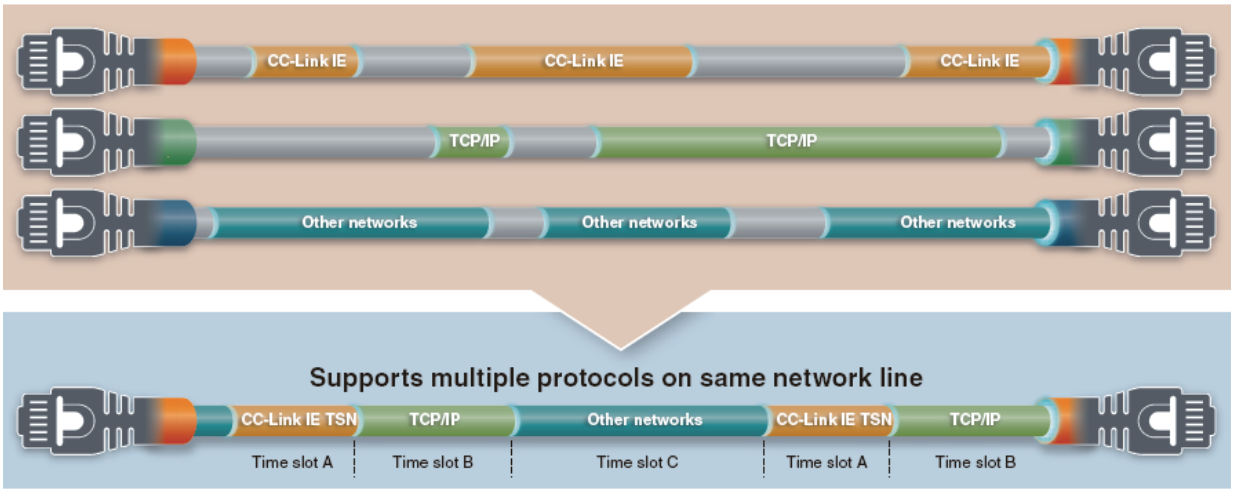
- For users who need high-speed control or complex operations
- Since operations are processed in the Motion module, load is distributed to the Motion module and PLC CPU, which contributes to reducing the cycle time.

CC-Link IE TSN

CC-Link IE TSN is a network that allows for coexistence of communications with IT systems while ensuring real-time control performance through cyclic communication. This network is optimal for building a factory-wide IIoT infrastructure since it enables flexible system construction using various devices and ensures high maintainability.

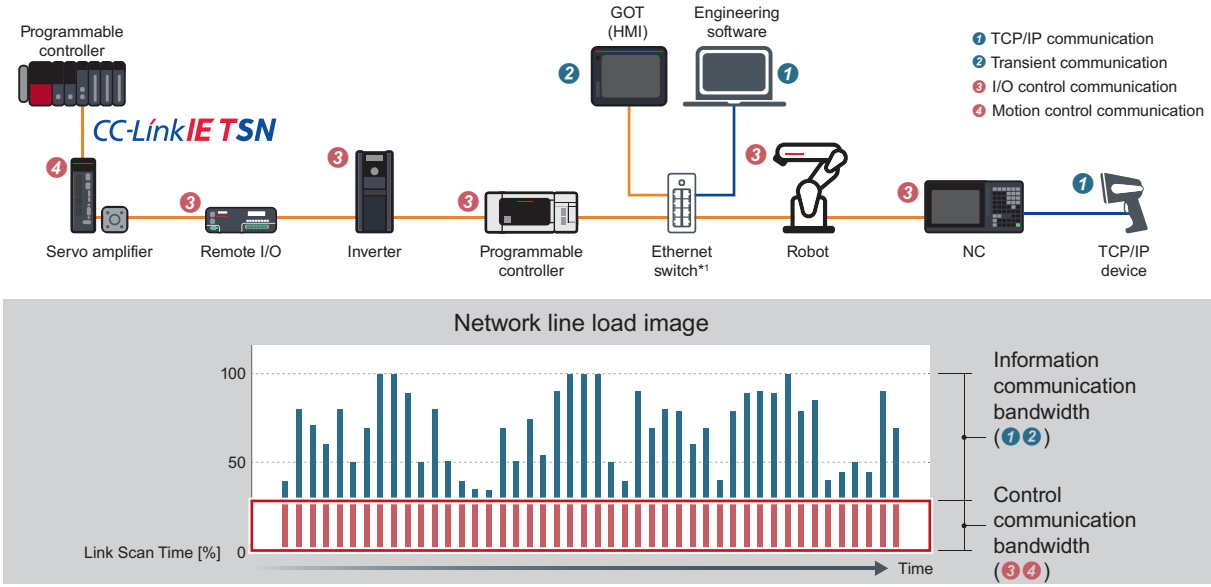
■ Coexistence with other networks

By using the TSN (Time-Sensitive Networking) technology, CC-Link IE TSN, TCP/IP, and other Ethernet networks can be mixed on the same trunk line in different periods of time.



■ Guaranteed periodicity even when TCP/IP communications coexist

Even when TCP/IP communications coexist, periodicity of cyclic communication is guaranteed. Flexible IIoT system construction is possible because general-purpose IP devices can be used without affecting system control.
(They cannot be connected depending on the devices or configuration.)

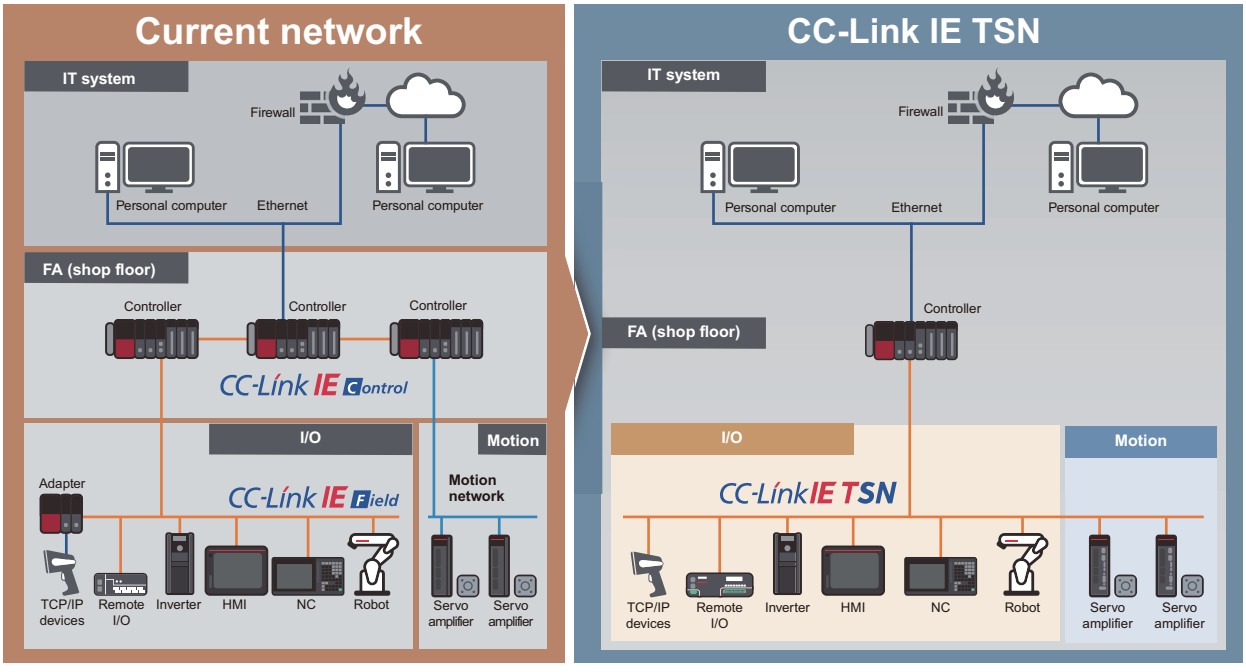


Network configuration example (includes functions and products planned for future support/release.)

*1. Class B switching hub supporting CC-Link IE TSN recommended by the CC-Link Partner Association.

■ Network integration

IT systems and drive systems that have been configured with multiple networks can be integrated.
This increases flexibility of system configuration and reduces wiring cost.



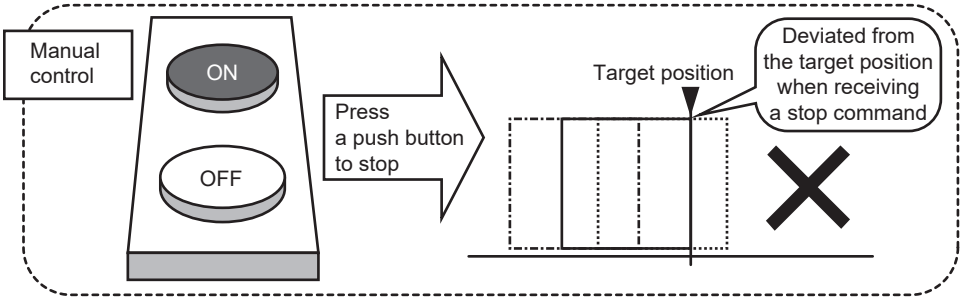
Network configuration example (includes functions and products planned for future support/release.)

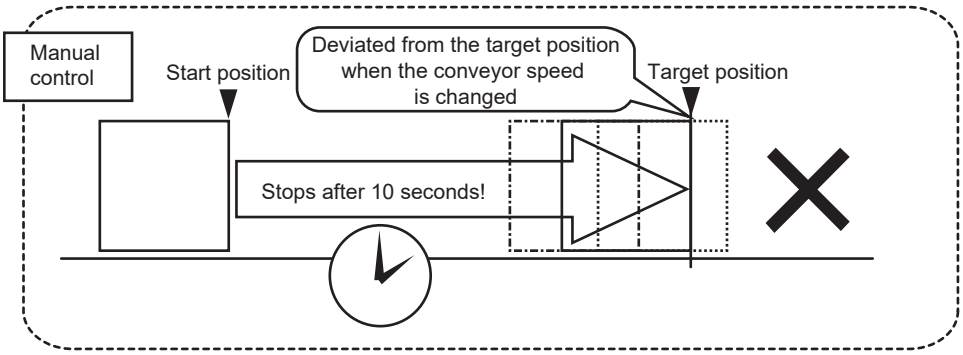
1.2 Control Mechanisms

This section describes positioning control and synchronous control.

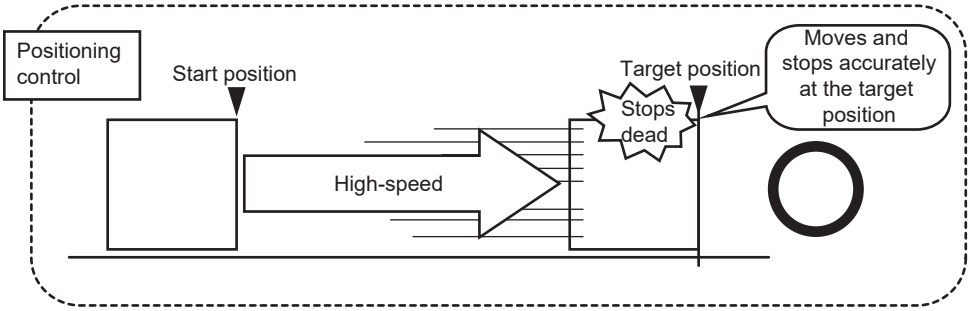
Positioning control

"Positioning control" is to move a workpiece or a tool at a specified speed and stop it exactly at the target position.
Manual control where a workpiece is stopped by pressing a button or automatic control that stops a workpiece based on the timer can be used to move a workpiece to the target position. However, various problems arise when an attempt is made to accurately determine the stop position or to move a workpiece at high speed and then stop it.





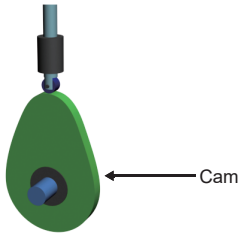
This exercise is for learning the positioning control method to move a workpiece at high speed to the target position and accurately stop it.



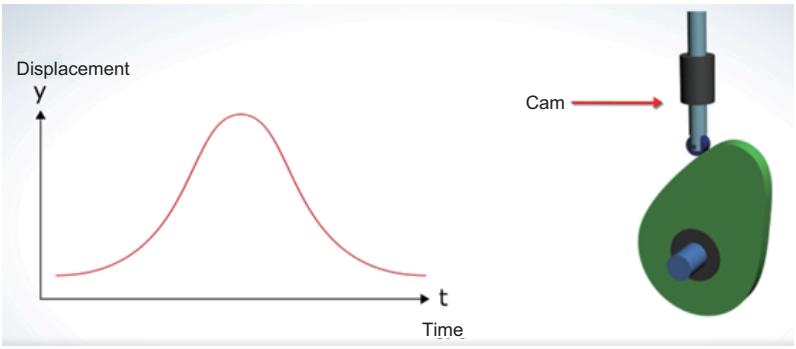
Even for transfer control that requires high precision, repetitive and reliable start/stop operation can be performed by using positioning control of the Motion module. Depending on the device mechanism, the stopping precision can be controlled in units of μm.

Synchronous control

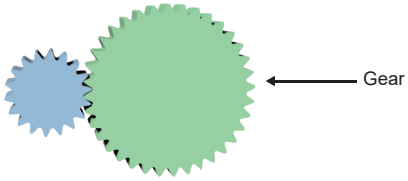
"Synchronous control" is to control the master axis and slave axis in sync with each other.
By using a cam to change the direction of motion and a gear to transmit power, the slave axis follows the motion of master axis.
A cam is a mechanical component that changes the direction of motion of machine parts.



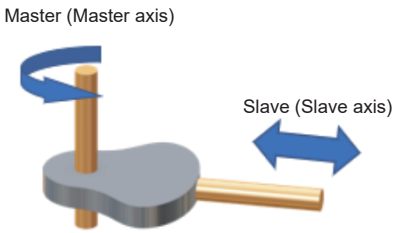
By installing it to a rotating axis, a curve (cam curve) is plotted based on the rotation angle. The cam curve shows the relationship between the cam rotation speed and the follower motion.



A gear is a mechanical component that transmits rotational motion accurately by gear meshing. Power is transmitted from one axis to another, and rotation speed is determined by the ratio of the number of teeth on the gear to the number of teeth on the other gear.



When these mechanical components are used for synchronous control on hardware, complex design and fabrication may be required or errors may occur depending on the accuracy of the machine.
This exercise is for learning how to replace mechanical synchronous control with software-based control.
The synchronous control function of the Motion module uses single-axis synchronous control FB. By transmitting the position information (command) of the slave axis synchronized with the master axis, mechanical systems such as gears can be controlled using software.



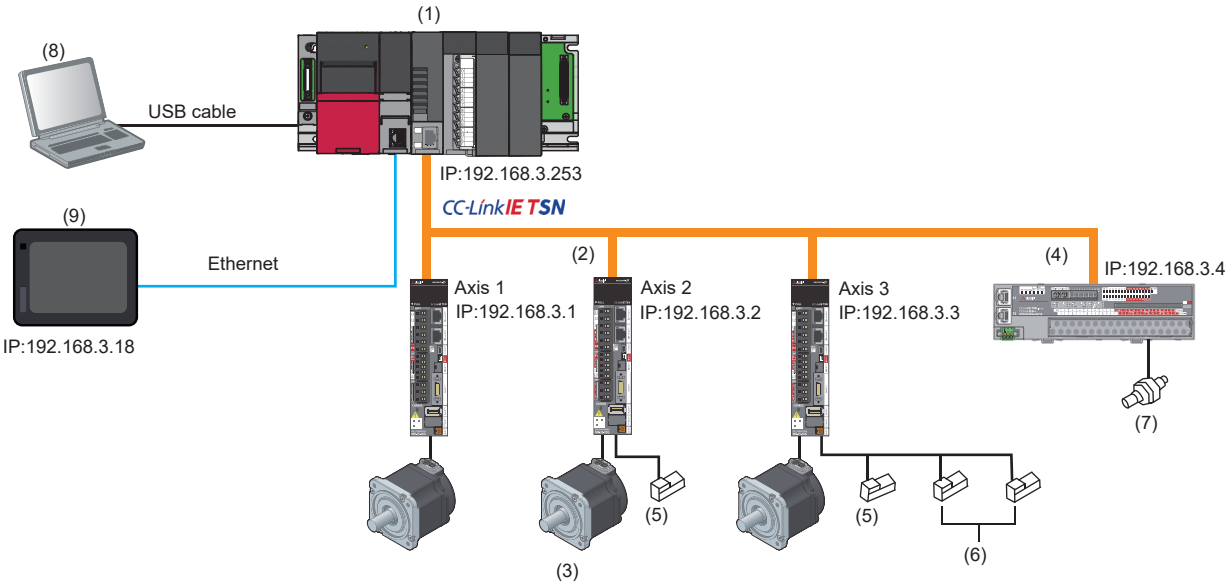
The following table shows the types of synchronous control and the relationship between the master and slave axes when the single-axis synchronous control FB is used.

Function		Description
Synchronous control	Cam operation	Operates the slave axis in sync with the slave axis in accordance with CamTable.
	Gear operation	Starts gear operation after setting the speed ratio between the master axis and slave axis.
	Addition/Subtraction positioning	Transmits the combined travel amount of the two axes.

2 SYSTEM CONFIGURATION

2.1 System Configuration of Demonstration Machine

The following figure shows the system configuration of the demonstration machine.



Device/software		Product name	IP address
(1)	Base unit	R35B	—
	Power supply module	R62P	—
	CPU module	R08CPU	192.168.3.39
	Motion module	RD78G4	192.168.3.253
	Input module	RX40C7	—
	Blank cover module	RG60	—
(2)	Servo amplifier	MR-J5-10G	Axis 1: 192.168.3.1 Axis 2: 192.168.3.2 Axis 3: 192.168.3.3
(3)	Servo motor	HK-KT053W	—
(4)	CC-Link IE TSN remote I/O module	NZ2GN2B1-32D	192.168.3.4
(5)	Dog sensor*1	—	—
(6)	Limit sensor*2	—	—
(7)	Sensor*3	—	—
(8)	GX Works3	SWnDND-GXW3 (n indicates the version.)	—
(9)	GOT2000	GT2708-STBA	192.168.3.18

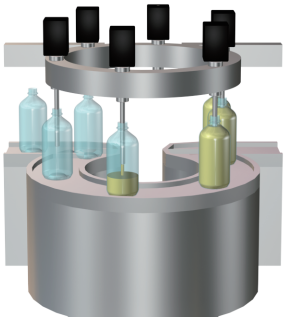
*1 Dog sensors are used for homing.
*2 Limit sensors detect the both ends of travel.
*3 A sensor detects workpieces.

2.2 Operation of Demonstration Machine

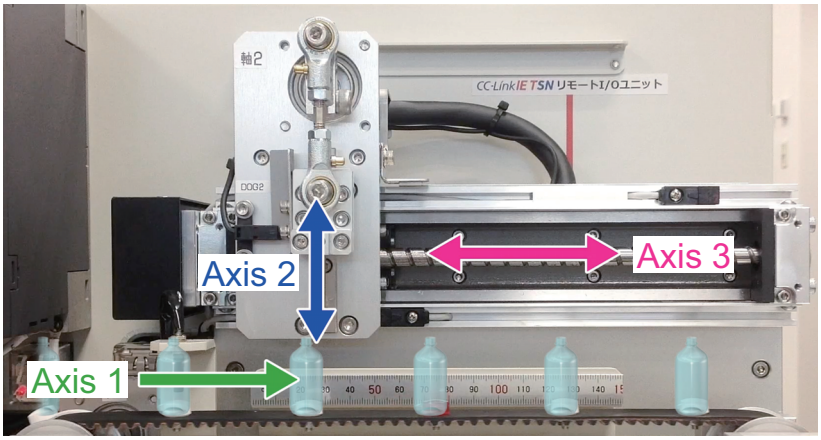
The following describes the operation of motion control by the system using a Motion module.

Outline

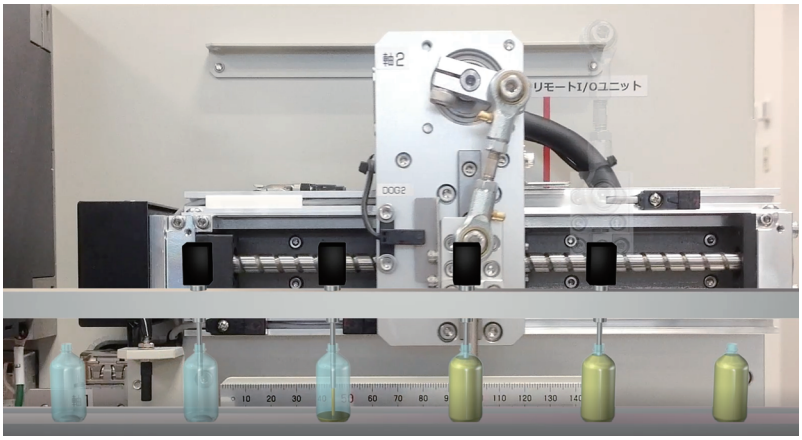
This exercise provides an operation example of axis control for a filling machine to fill containers.
Filling machines are used to fill containers such as bottles and cans with a specified volume of liquid or powder.



In this demonstration machine, axis 1 of the belt conveyor is used as the main axis and synchronized with axis 2 and axis 3 of the nozzle for filling into containers.

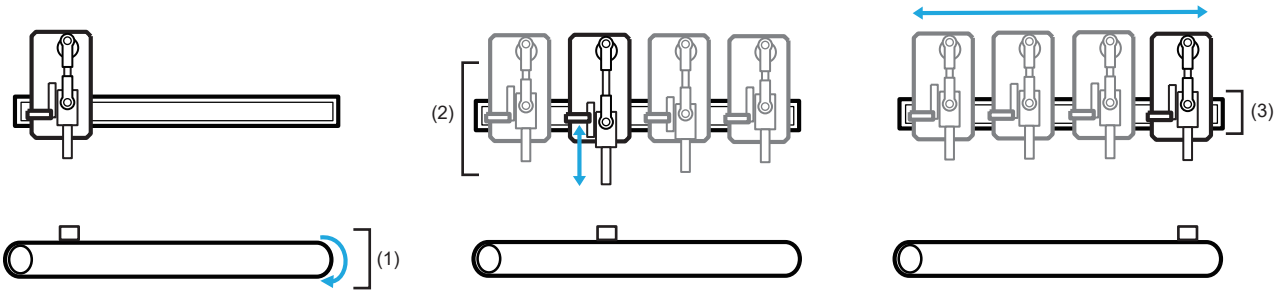


The following shows the image of filling by the demonstration machine.



Operation

The following figures show the operation of each axis of the demonstration machine.



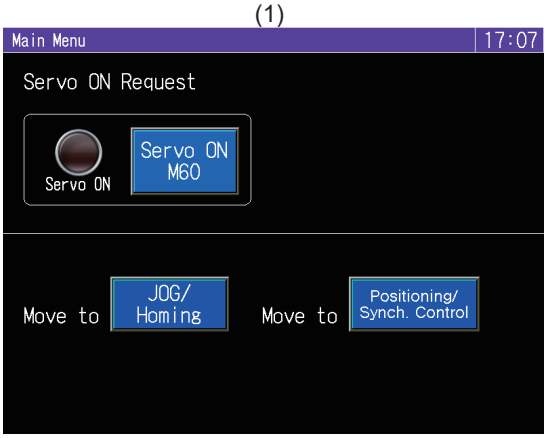
No.	Axis No.	Control	Description
(1)	Axis 1	Transfer control	Controls the speed of the belt conveyor (axis 1).
(2)	Axis 2	Nozzle height control	Controls the height of the nozzle (axis 2) in sync with the speed of the belt conveyor.
(3)	Axis 3	Parallel nozzle motion control	Moves the nozzle (axis 3) in parallel in sync with the speed of the belt conveyor.

2.3 Demonstration Machine Screen

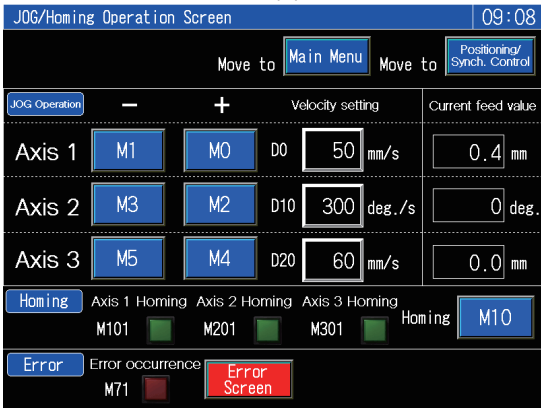
This section describes each screen of GOT.

The demonstration machine is operated using GOT2000.

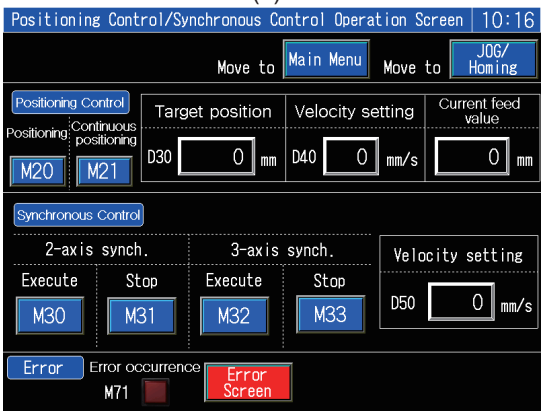
(1)



(2)



(3)



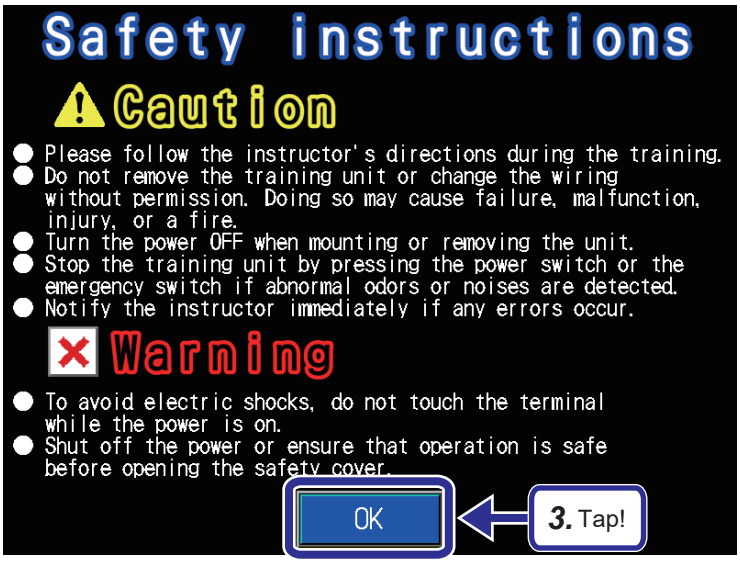
No.	Screen name	Description
(1)	Main Menu	Used to set all axes to the servo ON state. Used to navigate to the JOG Operation/Homing Operation Screen or Positioning Control/Synchronous Control Operation Screen.
(2)	JOG/Homing Operation Screen	Set the desired velocity and perform JOG operation for each axis. The current position is displayed as the current feed value. Used to perform homing control for axis 1, axis 2, and axis 3, and reset the feed current value for each axis after the homing control is complete.
(3)	Positioning Control/Synchronous Control Operation Screen	Used to set the desired target position and velocity and perform positioning or continuous positioning of the workpiece on axis 1 to the target position. Perform two-axis synchronous control of axis 1 and axis 3 and three-axis synchronous control by adding axis 2 to the two-axis synchronous control.

2.4 Operation Check of Demonstration Machine

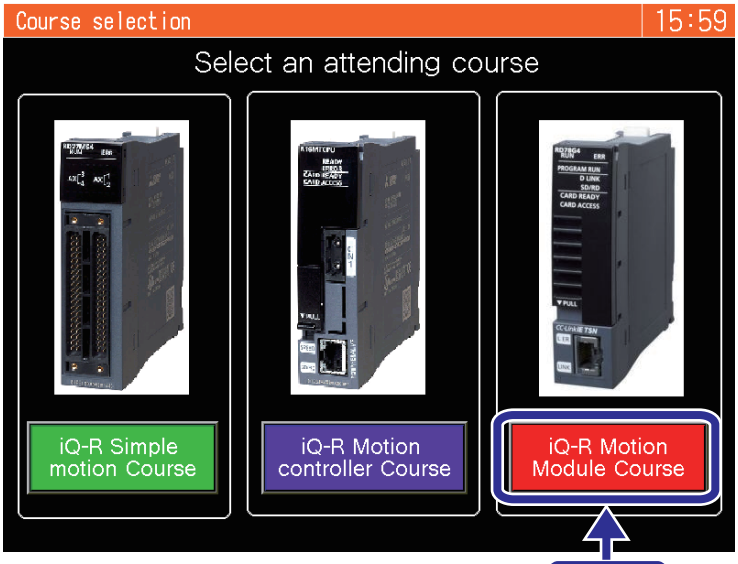
Check the following operation to be demonstrated in advance by using the demonstration machine where a programmed project with the configured parameters is installed.

☞ Page 98 EXERCISE 2 POSITIONING CONTROL☞ Page 151 EXERCISE 3 SYNCHRONOUS CONTROL

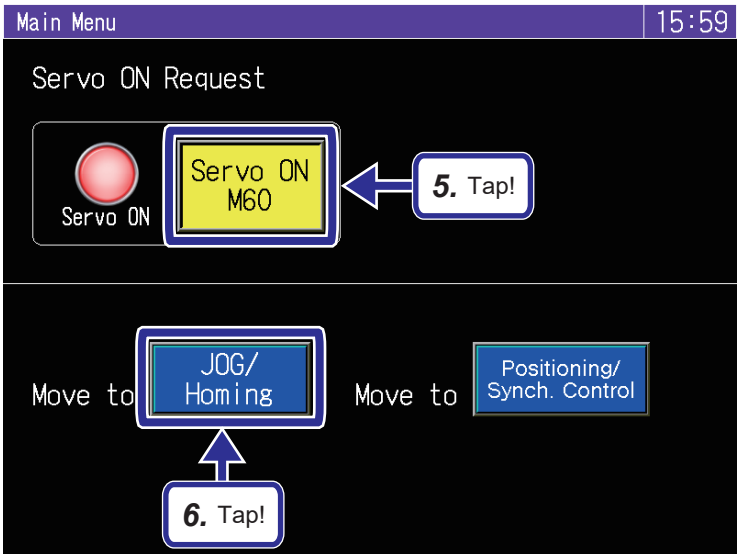
Operating procedure



1. Turn on the power of the control part and drive part of the demonstration machine.
2. Set the RUN/STOP/RESET switch of the CPU module to "RUN".
3. Tap the [OK] button on the safety precautions screen of the GOT.

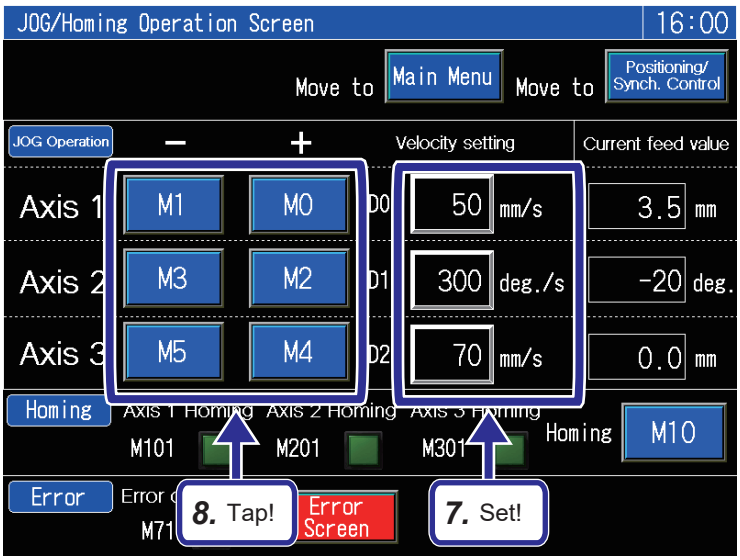


4. Tap the [iQ-R Motion Module Course] button on the course selection screen.



5. Tap the [Servo ON M60] button and check that the Servo ON lamp is turned on.

6. Tap the [JOG/Homing] button.

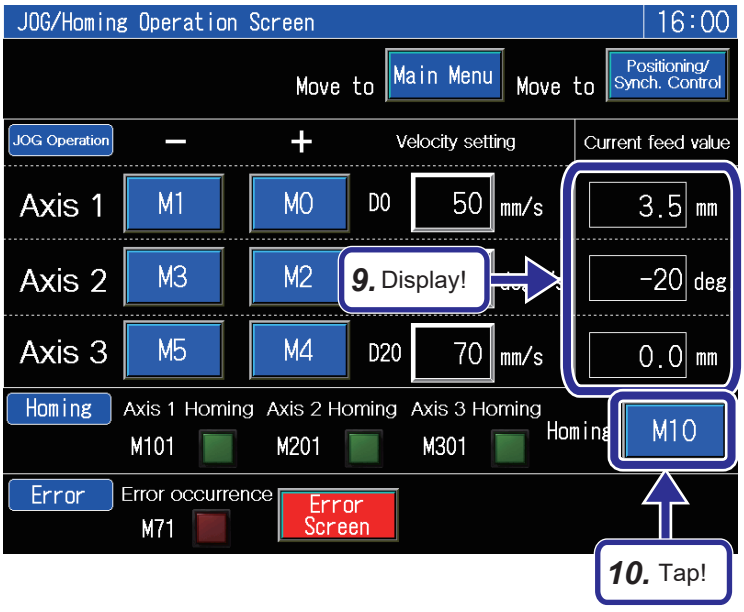


7. Set the desired velocity for each axis in the JOG/Homing Operation Screen.

8. Tap the JOG button for each axis.

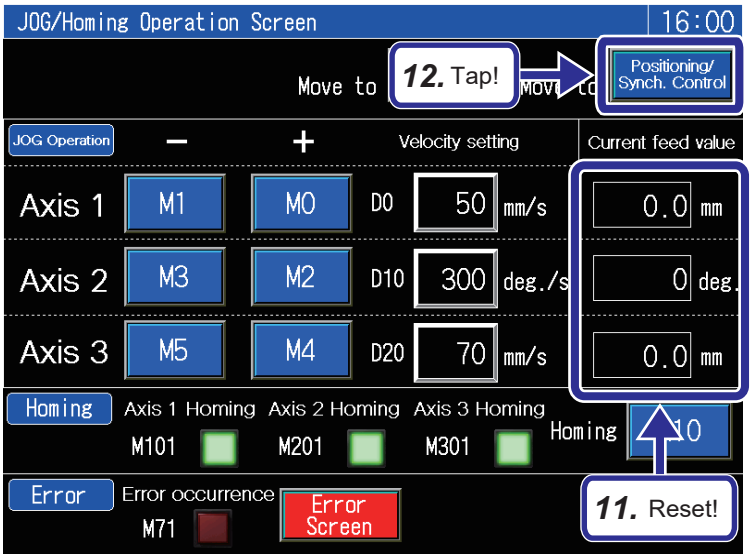
Tapping the - JOG button moves the corresponding axis in the - direction.

Tapping the + JOG button moves the corresponding axis in the + direction.



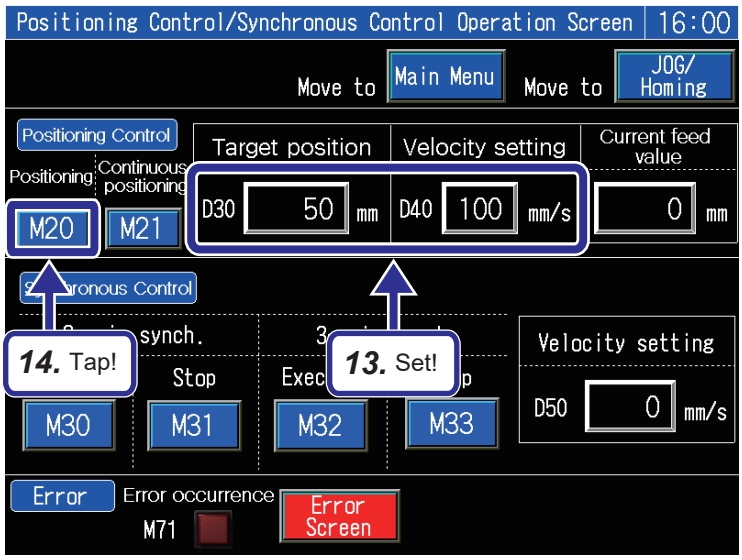
9. The current value of each axis is displayed.

10. Tap the [M10] button for homing.



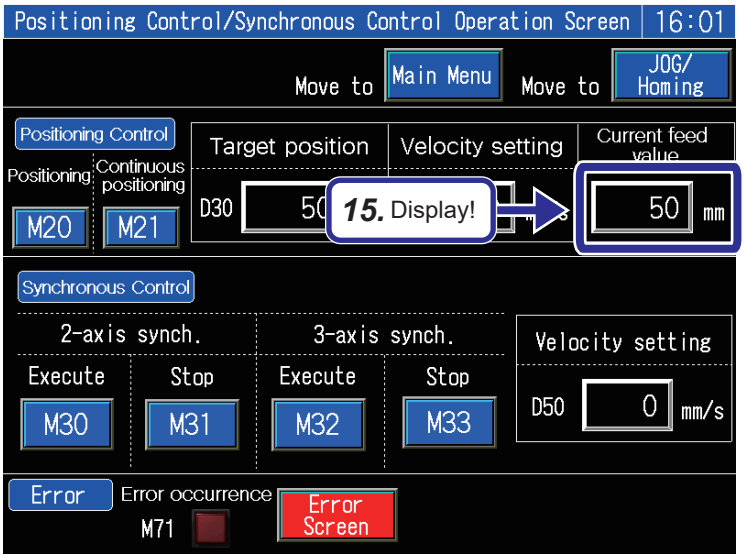
11. After homing of each axis, "M101", "M201", and "M301" lamps turn on and the current feed values are reset.

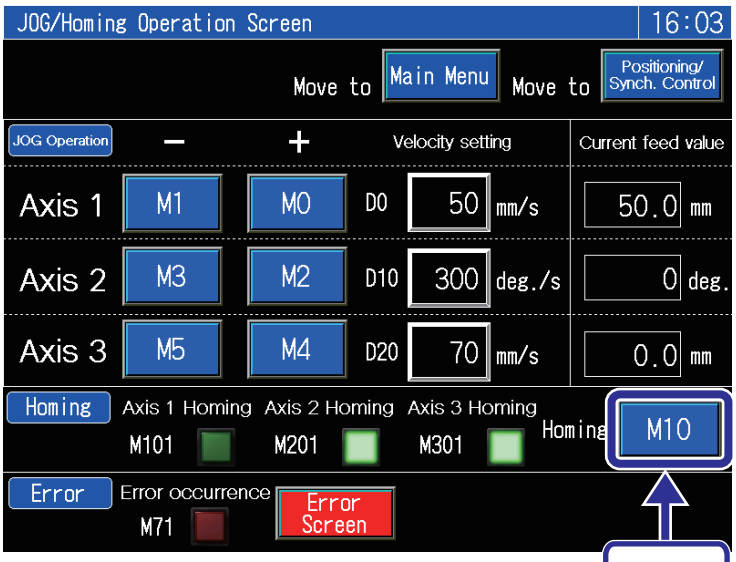
12. Tap the [Positioning/Synch. Control] button.



13. Set the desired values for the target position and velocity setting.

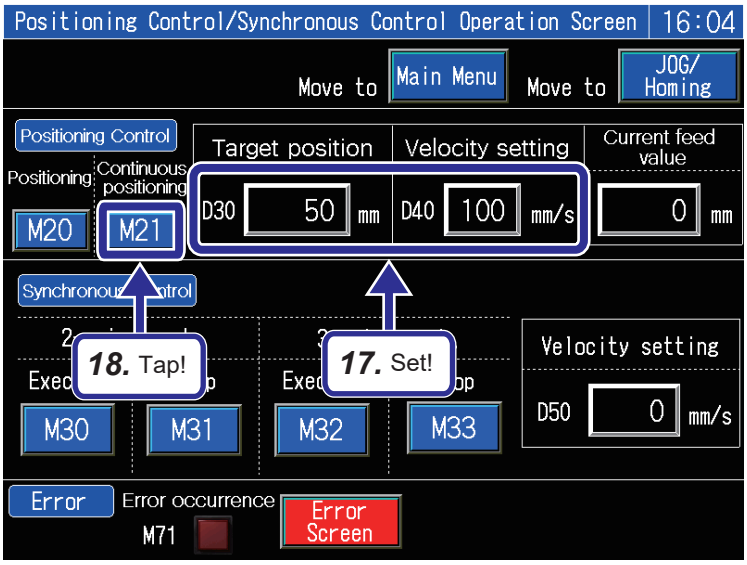
14. Tap the [M20] button for positioning.





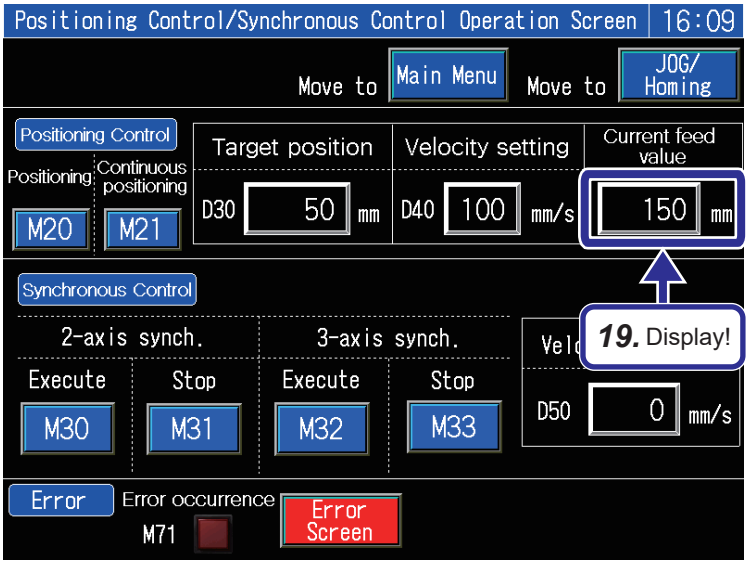
15. Check that the value displayed in the current feed value are the same as the value set for the target position.

16. Go back to the JOG/Homing Operation Screen and perform homing.



17. Go back to the Positioning Control/ Synchronous Control Operation Screen and set the desired values for the target position and velocity setting.

18. Tap the [M21] button for continuous positioning control.



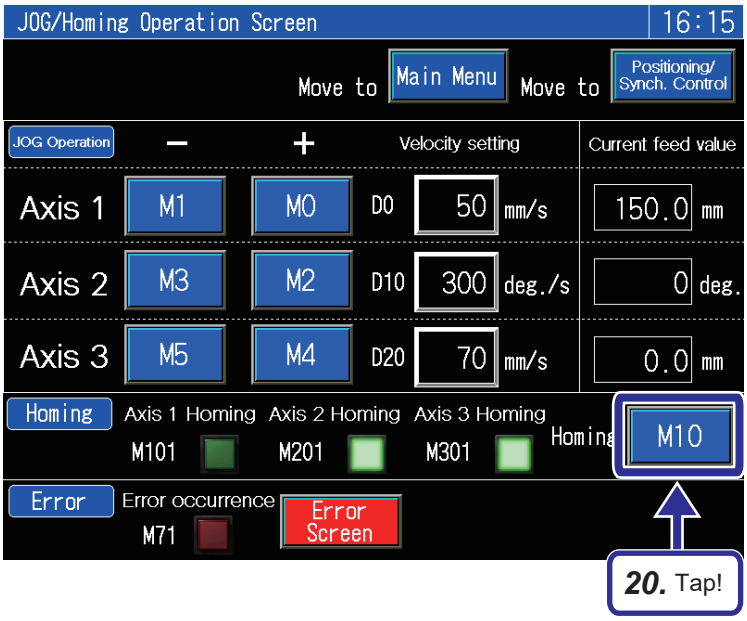
19. Check that the current feed value is displayed as follows.

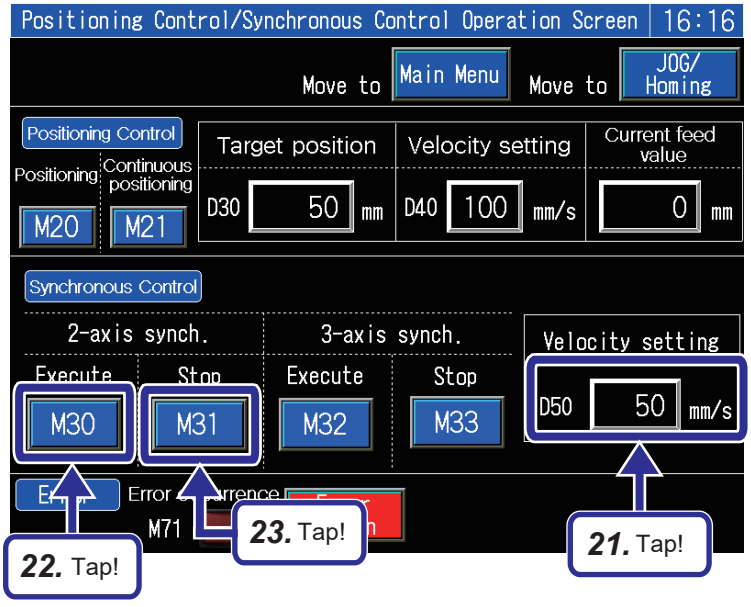
[Continuous positioning operation]

First time: Same as the value set for the target position

Second time: Twice as much as the value set for the target position

Third time: Home position (0mm)





20. Go back to the JOG/Homing Operation Screen and perform homing.

Point

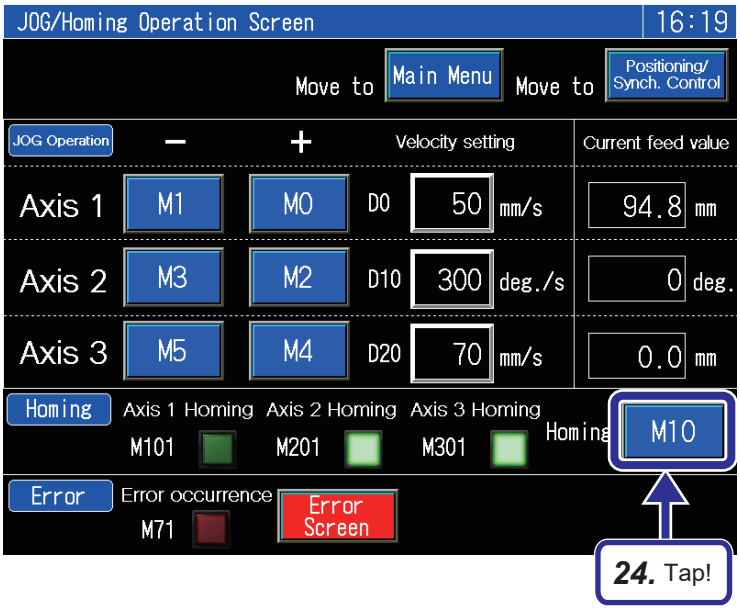
Always perform homing before synchronous control.

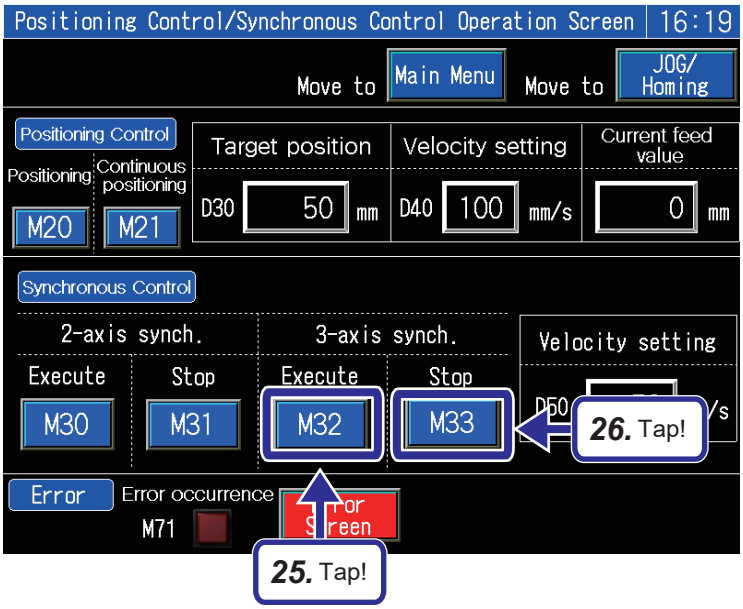
21. Go back to the Positioning Control/ Synchronous Control Operation Screen and set the desired velocity for synchronous control.

22. Tap the [M30] button for 2-axis synchronous execution.

The sensor detects a workpiece and axis 3 operates in sync with axis 1.

23. Stop 2-axis synchronous control by tapping the [M31] button for 2-axis synchronous stop.





24. Go back to the JOG/Homing Operation Screen and perform homing.

Point

Always perform homing before synchronous control.

25. Go back to the Positioning Control/ Synchronous Control Operation Screen and tap the [M32] button for 3-axis synchronous execution.

The sensor detects a workpiece, and axis 2 and axis 3 operate in sync with axis 1.

26. Stop the 3-axis synchronous control by tapping the [M33] button for 3-axis synchronous stop.

3 FUNCTIONS

3.1 Performance Specifications

The following table shows the performance specifications of RD78G(H).

Item		RD78G4	RD78G8	RD78G16	RD78G32	RD78G64	RD78GHV	RD78GHW
Number of control axess	Real drive axis[*1]	4 axes	8 axes	16 axes	32 axes	64 axes	128 axes	256 axes
	Virtual drive axis	1024 axes in total. The number of axes that can be set depends on the system memory capacity setting.[*2]						
	Virtual linked axis							
	Real encoder axis							
	Virtual encoder axis							
Operation cycle[*3]		62.5μs to 8 ms[*4]					31.25μs to 8 ms	
Interpolation function		1-axis to 4-axis linear interpolation 2-axis circular interpolation						
Control method		PTP (Point To Point) control, path control (linear, and arc can be set), speed control, and speed-torque control						
Control unit		pulse, m, degree, Revolution, inch, and character string of the desired unit						
Positioning	Positioning range	-10000000000.0 ≤Positioning range < 10000000000.0						
	Speed command	For position control: 0, +0.0001 to +2500000000.0 For speed control: 0, ±0.0001 to ±2500000000.0						
	Acceleration/deceleration processing	Acceleration/deceleration specification method (acceleration, deceleration, and jerk), acceleration/deceleration time-fixed method						
	Acceleration/deceleration time	Acceleration/deceleration specification method [Unit] U/s^2 [Range] 0.0000, 0.0001 to 2147483647.0 Acceleration/deceleration time-fixed method [Unit] s [Range] 0.000000, 0.000001 to 8400.0						
	Rapid stop deceleration time							
Flash ROM write count		Up to 100000 times						
Number of occupied I/O points		32 points/slot (I/O assignment: 32 points)					48 points, 2 slots (I/O assignment: Empty 16 points + 32 points)	
Internal current consumption (5 V DC)		1.93 A					2.33 A	
External dimensions	Height	106 mm (4.17 inch)						
	Width	27.8 mm (1.09 inch) (1 slot width)					56 mm (2.2 inch) (2 slots width)	
	Depth	110 mm (4.33 inch)						
Mass		0.26 kg					0.44 kg	

*1 When a multi-axis drive unit and a general output device are used as multiple axes, the number of those axes are counted.
Example: The 2-axis drive unit is counted as 2 axes.

*2 For memory capacity, refer to "Memory usage" in the following manual.
MELSEC iQ-R Motion Module User's Manual (Application)

*3 It depends on the number of control axes.

*4 For the version of Add-on baseSystem "1.4" or earlier, the operation cycle is "125 μs to 4 ms".

3.2 Specifications of Interfaces with External Devices

The following lists the external interfaces.

○: Available

Interface name	RD78G	RD78GH	Application
CC-Link IE TSN	1 port	2 ports	Network connection
SD memory card	○	○	Storage of parameters and logs

CC-Link IE TSN

The following table shows the performance specifications of CC-Link IE TSN.

Item			RD78G4	RD78G8	RD78G16	RD78G32	RD78G64	RD78GHV	RD78GHW
Maximum number of link points per network		RX/RY (Slave label)	16K points for each module (16384 points, 2K bytes)						
		RWr/RWw (PDO is included.) (Slave label)	8K points for each module (8192 points, 16K bytes)						
Maximum number of link points per station	Master station	RX/RY (Slave label)	16K points for each module (16384 points, 2K bytes)						
		RWr/RWw (Slave label)	8K points for each module (8192 points, 16K bytes)						
Communication speed			• 1 Gbps • 100 Mbps*9*11						
Minimum synchronization cycle			62.5μs*7					31.25μs	
Time synchronization accuracy			±1μs						
Authentication Class			Authentication Class B device						
Maximum number of connectable stations per network*2			120 stations*3*4*5						
Maximum number of connectable devices per network			256 devices*6						
Communication cable			Ethernet cable which satisfies the standard						
Maximum number of networks			239						
Network topology*8			Line topology, star topology*1, and line plus star topology						
Communication method			Time sharing method						
Transient transmission capacity			1920 bytes at maximum						
Safety communications*10		Maximum number of safety connections per station	Master station: 120 connections						
		Maximum number of safety connections with the same communication destination station	1 connection						
		Maximum number of link points per safety connection	• Input: 8 words • Output: 8 words						

*1 A TSN hub is required in a star topology.

*2 No error occurs if there is neither station nor axis in the network setting.

*3 Even if the station is a multi-axis drive unit device (a device which can control two axes or more), the number of stations is counted as one station when it is recognized as one station. For details, refer to the manual of each drive unit.

*4 When the slave emulation is performed, the number of station is counted according to the communication device setting.

*5 For Add-on baseSystem version "1.4" or earlier, the maximum number of connectable stations is "64 stations".

*6 For Add-on baseSystem version "1.4" or earlier, the maximum number of connectable devices is "64 devices".

*7 For Add-on baseSystem version "1.4" or earlier, the minimum synchronization cycle is "125.00 μs".

*8 When the communication cycle is 31.25 μs or 62.50 μs, only the line topology is available. For connection with the star topology and the line plus star topology, the communication cycle must be set to 125.00 μs or more.

*9 This speed can be used for Add-on baseSystem version "1.5" or later.

*10 It is available depending on the firmware version. For details, refer to "Safety communication" in the following manual.
MELSEC iQ-R Motion Module User's Manual (Network)
*11 MR-J5-_G_(-RJ)/MR-J5W_-_G cannot be connected at 100Mbps.

3.3 Function List

The available functions are limited depending on the version of the Motion module software and engineering tool. For details, refer to "Restrictions by the version" in the following manual.
MELSEC iQ-R Motion Module User's Manual (Application)

Control functions

RD78G has several functions. For details of each function, refer to the following.
MELSEC iQ-R Motion Module User's Manual (Application)

Basic specifications

Function		Description
Axis control function	Technical units	Set the position command unit and velocity command unit used for motion control. The unit can be freely specified in accordance with the controlled property to provide intuitive programming and monitoring.
	Servo ON/OFF	Executes servo ON/OFF of the real axis connected to the motion system. The servo ON enables operation of real axes.
	Follow up	Reflects the input (current position) from the slave station to the set position of the axis.
	Absolute position control	Restores the current position of an axis.
Basic functions	Operation cycle	In the motion system, operation processing related to the motion control is performed in the fixed cycle (operation cycle).
	Add-on function	The motion system functions can be extended by installing the add-on library.
	System memory settings	Set the memory size used in the add-on library on the system memory (RAM) and the system memory (backup RAM).
	Software reboot	The software reboot (reset the system) is executed by writing the reboot command to the control command. When "Clear" is specified to execute the software reboot command, the system is rebooted and all data in the system is deleted.

Motion control

Function		Description
Start and stop	Start	Starts motion control.
	Retrigger/continuous update	Changes the control of the on-going FB with a retrigger/continuous update.
	Multiple start (buffer mode)	Multiple motion control FBs are continuously executed without stopping by executing the motion FB of another instance to the axis and the axes group subject to the motion control FB being executed.
	Stop	Stops motion control.
	Forced stop	Stops axes by using the forced stop signal.
Homing control	Homing request	Determines the start point of positioning control (home position) and carries out positioning toward that start point. This is used to return a machine system at any position other than the home position to the home position, for example, after positioning stop or when the Motion module requires "homing request" at power ON.
	Driver homing method	
	Data set homing method	
	Operation setting for incompletion of homing	Select whether to start the axis or not when the homing request is TRUE.
Axis control function	Single axis positioning control	Executes positioning to the specified position based on the address information.
	Single axis speed control	Controls the speed of the specified axis to the specified speed.
	Single axis manual control	Executes the desired positioning operation by inputting a signal from an external device. The following method is available. • JOG operation: Moves the machine by the specified movement amount.
	Multiple axes positioning control	Executes positioning to the specified position using interpolation control based on the address information.
Direct control	Velocity control	Switches the driver control mode to csv and executes control excluding the position loop.
	Torque control	Switches the driver control mode to cst and executes control.

Function			Description
Functions related to position	Current position change function		Changes the set position and cumulative current position to any specified address.
	Command in-position		Obtains the remaining distance to the target position and changes the command in-position flag to TRUE.
	Software stroke limit		Prevents a given move command to the address outside the setting range from being executed.
	Hardware stroke limit		Stops operation when a signal is input from each limit switch installed at the upper and lower limits of the physical moving range.
Functions related to speed	Acceleration/deceleration processing function		Adjusts the acceleration/deceleration of each motion control to the acceleration/deceleration curve suitable for the device.
	Speed limit		Keeps the command speed within the setting range of the speed limit if the command speed exceeds the speed limit during control.
	Override function		Changes the speed during control.
Functions related to torque	Torque limit		Keeps the generated torque within the torque limit if the torque generated in the servo motor exceeds the torque limit.
	Torque limit value change function		Changes the torque limit value during control.
Sub functions of control	Compensation function	Driver unit conversion function	Converts the cumulative current position to the driver command value and passes it to the target position. Also, it converts the current position of the driver and calculates the actual position.
	Command filter	Smoothing filter	Suppresses load-side vibration, such as work-side vibration and base shake.
		Direction limit filter	Limits the direction of an axis.
		Speed limit filter	Limits the speed of axis.
		Backlash compensation filter	Compensates mechanical backlash (play).
	Input variable change in execution		Changes input variables during control.
Common functions	External signal selection		Set the I/O signals to be used for each type of control.
	Touch probe		Records (latches) the desired data when a trigger input signal is detected.
	Slave emulate		Executes axis control on the real axis without connecting the slave station.
Synchronous control	Cam operation		Operates the slave axis in sync with the slave axis in accordance with CamTable.
	Gear operation		Starts gear operation after setting the speed ratio between the master axis and slave axis.
	Addition/Subtraction positioning		Transmits the combined travel amount of two axes.
Operation profile function	Operation profile data		Opens the cam profile data and reads/writes the cam data.

Control/operation/maintenance

Function			Description
Logging	Data logging function		Collects the motion system data at a specified interval based on the logging setting (trigger condition or data collection condition) written from the engineering tool and saves the results as data logging files.
	Real-time monitor		Used to configure the data collection and monitor the collected data (display waveform) in real time with the engineering tool connected to the motion system.
	Application function	Event detection	Detects triggers without saving file when the file save setting is set to "Disabled" in the logging setting.
		Auto logging	When an SD memory card containing the logging setting is inserted into the motion system, data logging starts automatically based on the logging setting on the SD memory card.
RAS functions	Execution time monitor		Enables monitoring of the operation cycle processing and normal task execution time.
	History data	Event history function	Saves the even history of errors detected by the motion system, operation of the module, and events related to motion control such as start and stop.
		Position data history	Enables monitoring of the position data history of each axis on the engineering tool.
	Servo system recorder		Generates the optimal logging setting file for analyzing error causes and always monitors the error status of the motion system and corresponding slave devices. Records the system status before and after an error occurs for a certain period of time and saves the record as a file.
File control	File transfer function		Executes file operation based on the specified command.
	Parameter read/write function		Allows parameters to be read or written.
Module software install			When updating or changing to the latest software, users have to install the software again.

List of network functions

The following table lists the functions of CC-Link IE TSN. For details of the functions, refer to "Functions" in the following manual.

MELSEC iQ-R Motion Module User's Manual (Network)

Cyclic transmission

This function allows periodic data communications among stations on the network using slave labels.

Function	Description
Communications using slave labels	Exchanges data periodically among the stations on the same network by using the slave labels of the Motion module.
PDO communication	Communicates with slave devices periodically.

Transient transmission

This function is used for communications at any desired timing.

Function	Description
Communications using SLMP	Reads/writes data from/to the device memory of the own station or CPU module in the CC-Link IE remote station and buffer memory of the intelligent function module in the host system by Ethernet.

Ethernet connection

This function connects an Ethernet device to a module without interfering with CC-Link IE TSN.

Function	Description
Connection with MELSOFT products and GOT	Allows programming and monitoring of the CPU module using the engineering tool, and monitoring and testing of the CPU module using the GOT by Ethernet.
Connection with SLMP compatible devices	Connects SLMP compatible devices (such as a personal computer or a vision sensor) to RD78G(H).

Security

This function ensures optimal security for the network environment by restricting access to each communication path to the CPU module.

Function	Description
Remote password	Prevents unauthorized access to the CPU module from a remote location by function.
IP filter	Identifies the IP address of the access source to limit access to the Motion module.

RAS

This function improves reliability, availability, and serviceability to comprehensively enhance usability of automated equipment.

Function	Description
Slave station disconnection	Stops data link of the slave station where an error has occurred and continues data link of the normally operating slave stations in star topology.
Automatic return	Restarts data link automatically when the device station that has been disconnected due to an error returns to normal.
Duplicate detection of station type and IP address	Detects duplicate master stations and IP addresses.
Time synchronization	Sets the clocks in the CPU module and remote devices automatically based on IEEE1588 or IEEE802.1AS synchronization.

Safety communications

This function enables safety data exchange between safety stations on the same network.

Function	Description
Safety communications	Establishes a safety connection and executes "one-on-one" safety communications periodically between safety stations on the same network.

Troubleshooting

This function performs diagnostics and operation test using the engineering tool to check the status of the modules and networks.

Function	Description
CC-Link IE TSN/CC-Link IE Field diagnostics	Monitors the CC-Link IE TSN status. The network configuration, stations where data link is not executed, and communication status monitor of the selected station are displayed on the engineering tool.
Communication test	Checks if transient transmission data is properly routed from the own station to the communication target.

Others

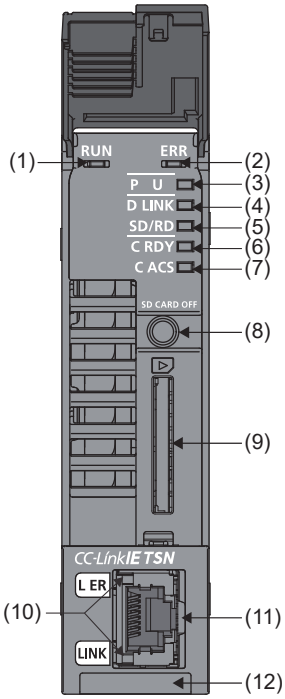
Function		Description
"CC-Link IE TSN Configuration" window	Parameter setting of slave stations	Set the parameters of slave stations (the number of points and assignment of link device) for the master station.
	Detection of connected/ disconnected devices	Detects the connected slave stations and displays it on the "CC-Link IE TSN Configuration" window.
	Parameter processing of slave stations	Reads and saves the parameters from the slave stations, and writes the saved parameters to the slave stations.
	Command execution to slave stations	Executes commands (error clear request and error history clear request) to slave stations.
Network synchronous communication (Motion control station)		Synchronizes the control cycle of slave stations with the communication cycle specified in the master station.
Automatic detection of connected devices (iQSS)		This function reflects the implementation status of network configuration to the network configuration setting of the master station on the engineering tool. It supports the network configuration setting of the master station when a slave station is added at system startup.
Slave station parameter automatic setting		Writes the parameters of the slave stations on CC-Link IE TSN that have been set using the engineering tool into the data memory or SD memory card of the CPU module, and sets the parameters automatically via the master station if the slave station is returned or connected to the network when it is powered ON or changed (slave station parameter automatic setting).
Label access to remote devices		Allows the use of labels and FBs for the slave stations on the network as well as the proximity module.
CPU module search on the network		Searches for modules with the Ethernet function that are connected to a common switching hub with the engineering tool and displays a list of search results

3.4 Part Names

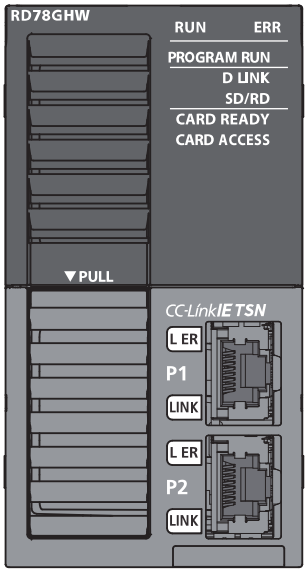
The following shows the name of each part of the Motion module.

RD78G

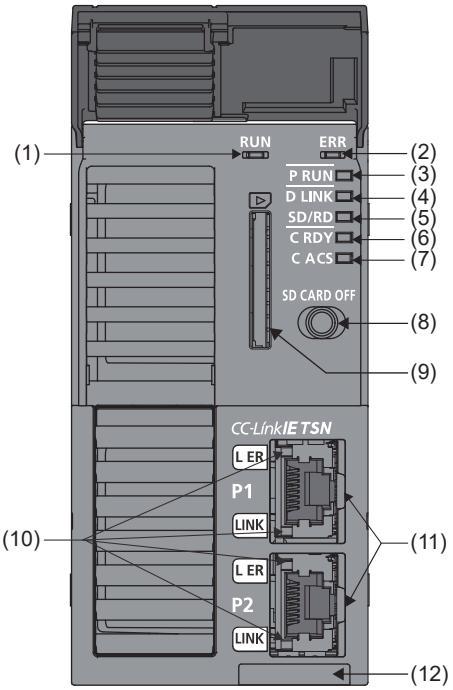
RD78G (with cover open)



RD78GH



RD78GH (with cover open)



No.	Name			Description
(1)	RUN LED			Refer to the following. ☞ Page 33 LED display specifications
(2)	ERR LED			
(3)	PROGRAM RUN LED (Indicated as P RUN inside the cover.)			
(4)	D LINK LED			
(5)	SD/RD LED			
(6)	CARD READY LED (Indicated as C RDY inside the cover.)			
(7)	CARD ACCESS LED (Indicated as C ACS inside the cover.)			
(8)	SD memory card access control switch			
(9)	SD memory card slot			
(10)	CC-Link IE TSN connector LED*1	P1	L ER LED	
			LINK LED	
		P2	L ER LED	
			LINK LED	
(11)	Ethernet port			Connects to a slave station.
(12)	Serial No. marking			Shows the serial No. of the Motion module.

*1 For details of the modules occupying two slots, refer to the following manual.
MELSEC iQ-R Module Configuration Manual

LED display specifications

The following lists the LED display specifications of the Motion module.

The LED display differs during software installation. For details, refer to "MODULE SOFTWARE INSTALLATION" in the following manual.

MELSEC iQ-R Motion Module User's Manual (Application)

□: OFF, ■: ON, ●: Flashing

Name	Description	LED display		Status
RUN LED	Indicates the operating status.	RUN LED■		Operating normally
		RUN LED●		Every 500 ms: Clear / Quick clearing
		RUN LED□		Error, initializing
ERR LED	Indicates the error status.	ERR LED□		Operating normally
		ERR LED■		Error
		ERR LED●		Every 200 ms: Error Every 500 ms: A data link error station has been detected.
PROGRAM RUN LED	Indicates the execution status of the built-in program.	PROGRAM RUN LED■		Program is running.
		PROGRAM RUN LED□		Program is stopped.
D LINK LED	Indicates the data link status.	D LINK LED■		Data link in progress (during cyclic transmission)
		D LINK LED●		Data link in progress (cyclic transmission stopped)
		D LINK LED□		Data link not performed (disconnection)
SD/RD LED	Indicates the data communication status.	SD/RD LED■		Communicating data
		SD/RD LED□		Not communicating data
CARD READY LED	Indicates the SD memory card status.	CARD READY LED■		SD memory card is available.
		CARD READY LED●		In preparation
		CARD READY LED□		Not inserted
CARD ACCESS LED	Indicates the access status of the SD memory card.	CARD ACCESS LED■		Accessing the SD memory card
		CARD ACCESS LED□		Not accessing the SD memory card
L ER LED	Indicates the port status.	P1	L ER LED■	Abnormal data received
			L ER LED□	Normal data received
		P2	L ER LED■	Abnormal data received
			L ER LED□	Normal data received
LINK LED	Indicates the link status.	P1	LINK LED■	Link-up
			LINK LED□	Link-down
		P2	LINK LED■	Link-up
			LINK LED□	Link-down

The error status can be determined as follows according to the status of RUN LED and ERR LED.

RUN LED	ERR LED	Error status	Description
OFF	ON or flashing	Major error	Causes the module to stop operating due to hardware failure or memory failure.
ON	Flashing	Moderate error	Causes the module to stop operating due to parameter errors affecting module operation.
ON	ON	Minor error	Allows the module to continue operation such as communication errors, positioning control errors, and program errors.

When multiple errors occur, the error status is displayed in the order of major, moderate, and minor.

4 DATA TYPE

Various motion control FBs, variables, and parameters are used to perform the motion control.

4.1 Overview of Motion Control FBs

The motion control FBs that can be used in the motion system include FBs defined by PLCopen®. The basic specifications of I/O signals are compliant with the PLCopen®motion control FBs.
PLCopen®is a third-party organization that promotes IEC 61131-3 (JIS B 3503), the international standard for PLC programming, and develops and certifies the specifications of the vendor-independent standard function blocks (FBs) for the purpose of improving the efficiency of PLC application development.
Using the FBs defined by PLCopen® allows programming independent of the PLC manufacturer because the input/output and operation specifications of the FBs are standardized. This enables structured programming, thereby increasing the reusability of programs and reducing engineering costs.
For the Motion module, global labels, local labels, and motion control FBs are used to create motion control programs and perform motion control.

Point

The underlined part of the variable name must be defined by the user.

AxisName.AxisRef.AxisNo
↓
User-defined variable

Type of motion control FB

Motion control FBs are classified in terms of their operation and execution method.

Management FB / Operation FB / Standard FB

Motion control FBs are classified into the following types in terms of their operation.

Type	Description
Management FB	• A motion control FB that takes an axis or an axes group as an argument and does not change the axis status or the axes group status during execution. (There are some exceptions.) • In most cases, a management FB can execute multiple instances for an axis or an axes group at the same time.
Operation FB	• A motion control FB that takes an axis or an axes group as an argument and changes the axis status or the axes group status during execution. • In most cases, an operation FB can be executed to only one axis or axes group. However, some FBs can be executed at the same time. • In most cases, the axis status or the axes group status will not be changed even if a management FB is executed while an operation FB is being executed. However, some FBs cause certain state transitions.
Standard FB	• A motion control FB that does not take an axis or axes group as an argument. • A standard FB can execute multiple instances at the same time. Since it is not related to axes, it does not interact with operation FBs or management FBs.

Execute command (Execute) type / Enable (Enable) type

Some motion control FBs are executed by Execute command (Execute), while others are executed by Enable (Enable).

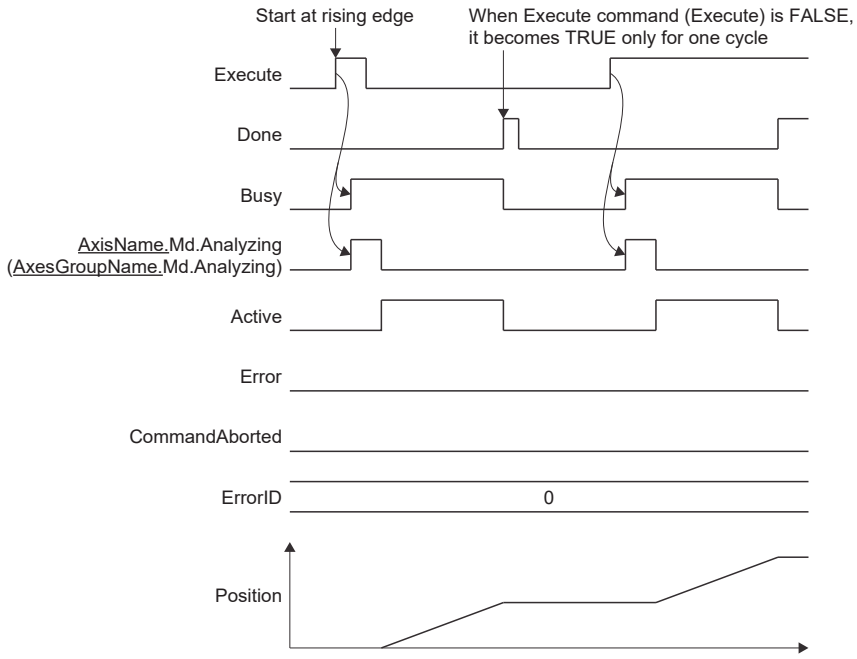
Type	Execute command (Execute) type	Enable (Enable) type	Other types
Management FB	• MC_GroupEnable (Axes Group Enabled) • MC_GroupDisable (Axes Group Disabled) • MC_SetPosition (Current Position Change) • MCv_SetTorqueLimit (Torque Limit Value) • MC_WriteParameter (Parameter Write) • MC_Reset (Axis Error Reset) • MC_GroupReset (Axes Group Error Reset) • MC_TouchProbe (Touch Probe Enabled) • MC_AbortTrigger (Touch Probe Disabled) • MC_CamTableSelect (Cam Table Selection) • MCv_ChangeCycle (Current Value Change per Cycle) • MCv_MotionErrorReset (Motion Error Reset)	• MC_Power (Operation Available) • MC_SetOverride (Override Value Setting) • MC_ReadParameter (Parameter Read) • MCv_AllPower (All Axes Operation Available) • MC_GroupSetOverride (Axes Group Override Value Setting)	—
Operation FB	• MC_Home (OPR) • MC_Stop (Forced Stop) • MC_GroupStop (Group Forced Stop) • MC_MoveAbsolute (Absolute Value Positioning) • MC_MoveRelative (Relative Value Positioning) • MC_MoveVelocity (Speed Control) • MC_TorqueControl (Torque Control) • MCv_SpeedControl (Speed Control (Including Position Loop)) • MCv_MoveLinearInterpolateAbsolute (Absolute Value Linear Interpolation Control) • MCv_MoveLinearInterpolateRelative (Relative Value Linear Interpolation Control) • MCv_MoveCircularInterpolateAbsolute (Absolute Value Circular Interpolation Control) • MCv_MoveCircularInterpolateRelative (Relative Value Circular Interpolation Control) • MC_CamIn (Cam Operation Start) • MC_GearIn (Gear Operation Start) • MC_CombineAxes (Addition/Subtraction Positioning)	• MCv_BacklashCompensationFilter (Backlash Compensation Filter) • MCv_SmoothingFilter (Smoothing Filter) • MCv_DirectionFilter (Moving Direction Restriction Filter) • MCv_SpeedLimitFilter (Speed Limit Filter)	• MCv_Jog (JOG)
Standard FB	• MCv_ReadProfileData (Profile Read) • MCv_WriteProfileData (Profile Write)	—	—

The following describes the basic operation of each motion control FB executed by Execute command (Execute command) and Enable (Enable). Some motion control FBs have different specifications.

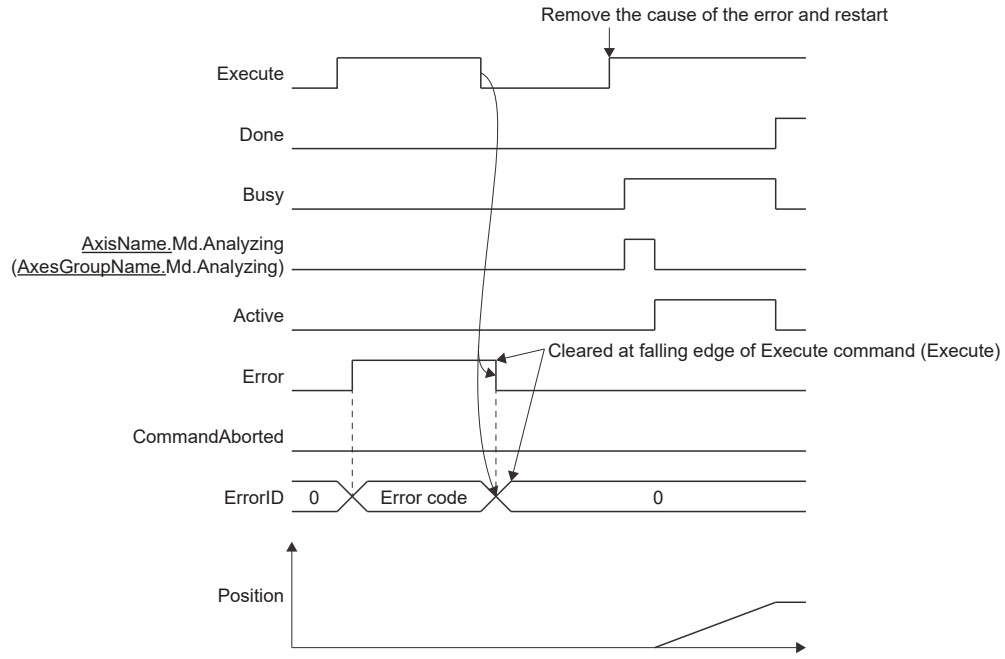
■ Basic operation of Execute command (Execute) type motion control FBs

- Execute command (Execute) type motion control FBs read the input parameters at the rising edge of Execute command (Execute) and then start operation. Once operation has started, the operation will be continued to the end even if Execute command (Execute) is set to FALSE.
- When operation is started, only one output variable among Executing (Busy), Execution completion (Done), Error (Error), and Abortion of execution (CommandAborted) becomes TRUE.
- Execution completion (Done), Error (Error), Error code (ErrorID), and Abortion of execution (CommandAborted) are reset at the falling edge of Execute command (Execute). Executing (Busy) and Controlling (Active) are not affected.
- When the input parameter is changed during operation, the change is applied at restart (retrigger) of Execute command (Execute) or by continuous update using Continuous update (ContinuousUpdate).
- Analyzing (AxisName.Md.Analyzing/AxesGroupName.Md.Analyzing) becomes TRUE at the rising edge of Executing (Busy), and Analyzing (AxisName.Md.Analyzing/AxesGroupName.Md.Analyzing) becomes FALSE after the start of operation.
- When Execute command (Execute) is used in pulse, Execution completion (Done) becomes TRUE only for one cycle.
- The timing chart of Execute command (Execute) type motion control FBs is shown below.

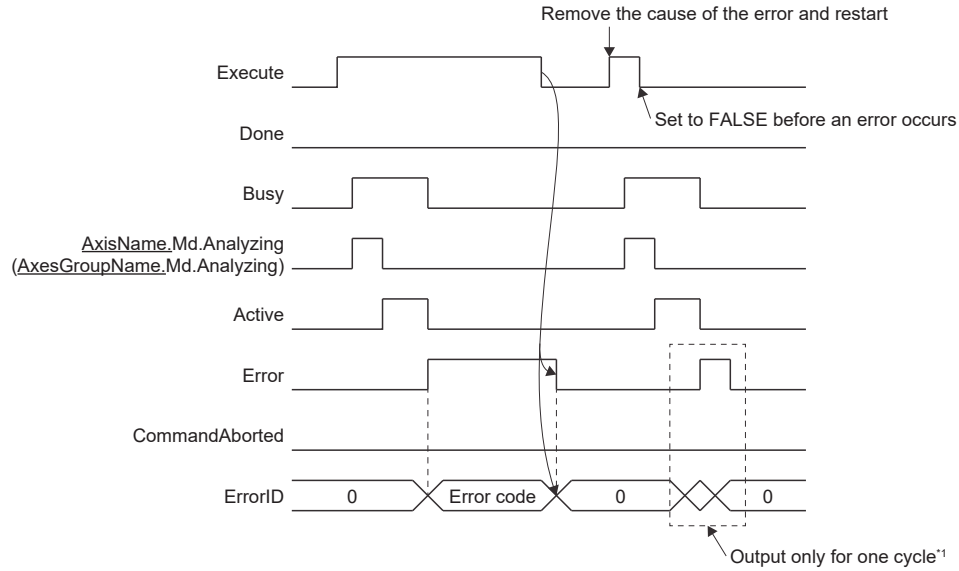
• Normal



• When an I/O variable error occurs



• When an input variable error occurs

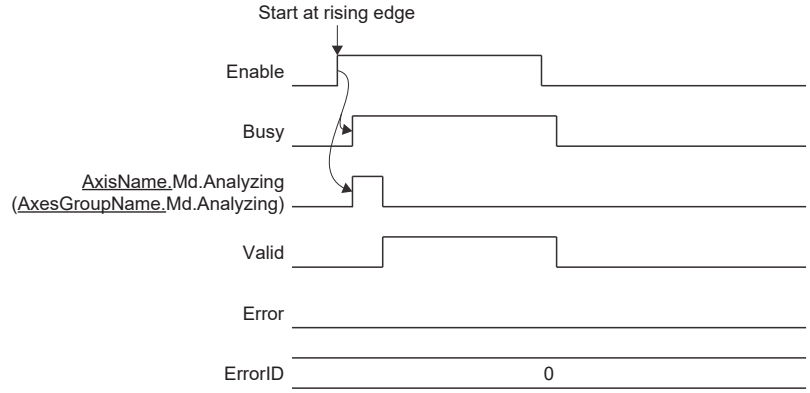


*1 The above operation is carried out because the termination condition of the FB is met (Execute command (Execute) is FALSE).
For FBs not related to axes or FBs which do not decelerate axes to stop, Error (Error) becomes TRUE only for one cycle and Error code (ErrorID) is output.
For FBs that require a deceleration stop, Error (Error) becomes TRUE until the axis is decelerated to stop to maintain Error code (ErrorID). When the axis is completely stopped, Error (Error) becomes FALSE and Error code (Error ID) is cleared.

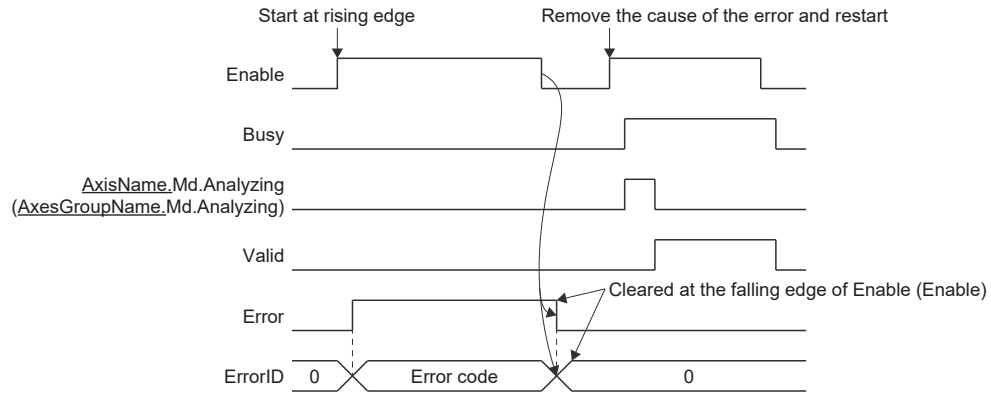
■ Basic operation of Enable (Enable) type motion control FBs

- Enable (Enable) type motion control FBs are continuously executed while Enable (Enable) is TRUE.
- Output value valid (Valid) indicates that the output value is valid. After Output value valid (Valid) becomes FALSE, outputs will not change.
- Only one output variable among Output value valid (Valid)/Enabled (Enabled)/Executing (Busy), Error (Error), and Abortion of execution (CommandAborted) becomes TRUE.
- Analyzing (AxisName.Md.Analyzing/AxesGroupName.Md.Analyzing) becomes TRUE at the rising edge of Executing (Busy), and Analyzing (AxisName.Md.Analyzing/AxesGroupName.Md.Analyzing) becomes FALSE after the start of operation.
- The timing chart of Enable (Enable) type motion control FBs is shown below.

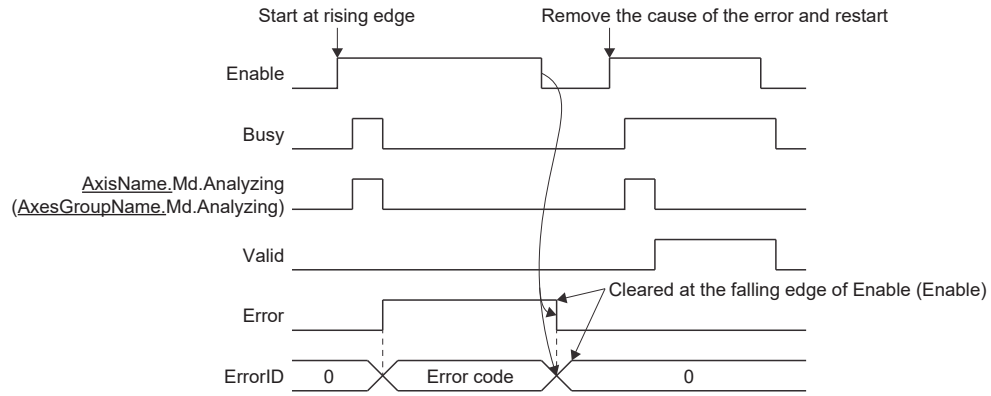
• Normal



• When an I/O variable error occurs



• When an input variable error occurs



Type of motion control

The following types of axis and axes group control can be executed by operation type motion control FBs.

Category	Subcategory	Sub-subcategory	Description
Axis control	Single axis control	Positioning control	Control in which Axis status (AxisName.Md.AxisStatus) is set to "5: During positioning operation (DiscreteMotion)" and the axis is moved to the target position
		Continuous control	Control in which Axis status (AxisName.Md.AxisStatus) is set to "6: During continuous operation (ContinuousMotion)" and continuous control is performed for the axis
		Synchronous control	Control which has Master axis (Master) and Slave axis (Slave) as I/O variables and performs synchronous control for the axis with Axis status (AxisName.Md.AxisStatus) of Slave axis (Slave) set to "7: During synchronous operation (SynchronizedMotion)"
		Homing control	Control in which Axis status (AxisName.Md.AxisStatus) is set to "3: During home position return (Homing)" and continuous control is performed for the axis
Axes group control	Multiple axes control	Positioning control	Control in which Axes group status (AxesGroupName.Md.GroupStatus) is set to "5: Operating (GroupMoving)" and the axis is moved to the target position

Error processing

If an error occurs while the motion control FB is executed, Error (Error) becomes TRUE and the error code is output to Error code (ErrorID). When an axis is used, Axis status (AxisName.Md.AxisStatus) changes to "1: Stopping on error (ErrorStop)" at this time. When an axes group is used, Axes group status (AxesGroupName.Md.GroupStatus) changes to "1: Stopping on error (GroupErrorStop)" at this time.
When the axis status of an available axis changes to "1: Stopping on error (ErrorStop)", all buffering FBs are aborted. Error (Error) of the aborted FBs becomes TRUE.
After that, an error reset must be executed to start the axis or axes group.

Point

One of the following values is output to Error code (ErrorID). (The output value depends on the control.) Note that warning codes are not output.

- Axis error code (AxisName.Md.ErrorID)
- Axes group error code (AxesGroupName.Md.ErrorID)
- Latest motion system error code (System.Md.ErrorID)

Errors (including warnings) during the execution of a motion control FB on the CPU module side will be output as a Motion module error.
Error codes that are output to motion control FB Error code (ErrorID) are as follows.

Error code	Description
0400H	No response was received from the Motion module within the specified time. Execute the FB again.
1C00H	A value outside the range is specified for the request ID.
1C01H	An FB was executed when the dedicated instruction could not be executed.
1C02H	Multiple instructions are being executed at the same time.
1C03H	• The memory capacity of the PlcInstruction add-on is insufficient. • The capacity of the buffer memory (area for MCFB) is insufficient.
1C04H	• Incorrect request data is specified. • An error was detected in the add-on related to MCFB.*1
1C05H	A value outside the range is specified for the request data length.
1C06H	A value outside the range is specified for the allowable number of response data.
1C07H	The add-on required to execute the instruction has not been loaded.
1C0FH	Dedicated instruction execution error

*1 A system error occurs at the same time. Check the error details of each function in the event history.

Precautions

When the I/O No. of the Motion module that is specified in the motion control FB argument is incorrect, or when the Motion module for executing FBs cannot be identified, there will be no operation, or an error code will be output on the CPU module side.

Units used in control

Units of the position, velocity, acceleration/deceleration, and jerk that are used in the motion system follow the technical units of the axis to be used.

The following types of control values, such as position and velocity, are used in the motion system.

Type	Description
Command value	A value (target value) based on an input to the motion control FB. (Commanded position, commanded velocity, etc.)
Set value	The current control value that is generated by motion operation. (Set position, set velocity, etc.)
Actual value	A value obtained by converting the actual value received from the slave devices by the real axis into the technical unit of the axis. (Actual position, actual velocity, etc.)

For details of the technical units and control values related to position and velocity, refer to the following.

MELSEC iQ-R Motion Module User's Manual (Application)

I/O variables in motion control FBs

The following describes the I/O variables in motion control FBs.

The I/O variables, input variables, and output variables must be defined in motion control FBs.

I/O variables

Set the variable names such as Axis information (Axis) and Axes group information (AxesGroup) for the axis and axes group of the driver to be controlled.

The set axis and set axes group are assigned as an axis variable or an axes group variable in the global label data.

Input variables

Set the operation conditions such as the target position and the command velocity.

Output variables

Output the FB status, driver status, error occurrence, etc.

Classes

The following shows the classes of the I/O variables, input variables, and output variables.

Variable	Class
I/O variable	VAR_IN_OUT
Input variable	VAR_INPUT
Output variable	VAR_OUTPUT

Data types

Variables are classified into different types in terms of their bit length, processing method, value range, etc.

Data type	Description
BOOL	Bit
WORD(HEX)	Word [unsigned]/bit string [16-bit] (hexadecimal)
WORD(UINT)	Word [unsigned]/bit string [16-bit]
DWORD(HEX)	Double word [unsigned]/bit string [32-bit] (hexadecimal)
DWORD(UDINT)	Double word [unsigned]/bit string [32-bit]
INT	Word [signed]
DINT	Double word [signed]
REAL	Single-precision real number
LREAL	Double-precision real number
TIME	Time
STRING(□)	Character string*1*2
WSTRING(□)	Character string [Unicode]*1*3

*1 □indicates the number of characters that can be set, excluding Null.
*2 For STRING type, enclose the character string to be set in single quotation marks (').
*3 For WSTRING type, enclose the character string to be set in double quotation marks (").

Omission of input arguments

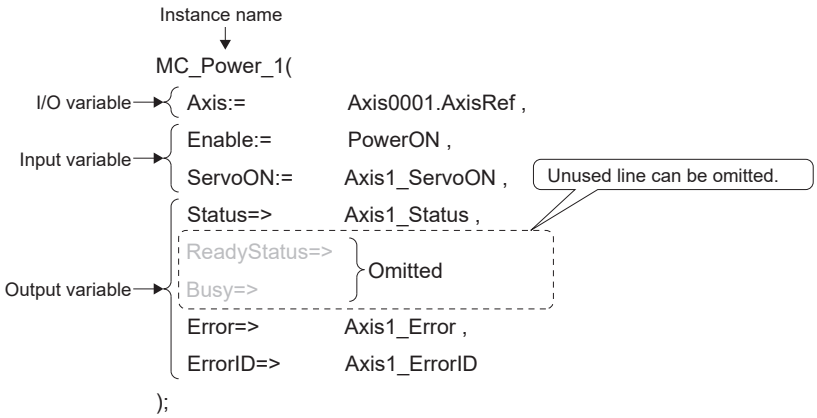
When omitting FB inputs, the default value defined for each FB is applied. For details of the default values, refer to the description of the motion control FB to be used.
When an input such as the velocity is omitted in the FB to be started by multiple start, the input value of the previous FB is inherited.

Refresh timing of inputs/outputs

Each argument of the FB is refreshed when the FB is called. To control the input/output of the FB in sync with the operation cycle, call the FB from a fixed cycle program whose cycle is the same as the operation cycle.

Motion control FB configuration

The structured text (ST) is used to create a program using motion control FBs on the Motion module side. In the ST language, the motion control FB is arranged as follows.



Name	Description
Instance name	The instance name assigned to each FB. The instance name can be changed.
I/O variable	Set the axis variable names such as Axis information (Axis) and Axes group information (AxesGroup) for the driver to be controlled.
Input variable	Set the operation conditions such as the target position and the command velocity. The setting of these variables can be omitted. When omitted, the default values are used.
Output variable	These variables output the FB status and driver status.

4.2 List of Motion Control FBs

The following shows the list of motion control FBs. For details of each motion control FB, refer to the following.

MELSEC iQ-R Programming Manual (Motion Control Function Blocks)

Management FBs

The following shows the list of management motion control FBs.

Motion control FB	Name	Description
MC_GroupEnable	Axes Group Enabled	Changes the specified axes group status from "0: Axes group disabled (GroupDisabled)" to "4: Standby (GroupStandby)".
MC_GroupDisable	Axes Group Disabled	Changes the specified axes group status to "0: Axes group disabled (GroupDisabled)".
MC_Power	Operation Available	Switches a specified axis to the operation possible status.
MC_SetPosition	Current Position Change	Changes the current position (commanded position, actual position) of the specified axis.
MCv_SetTorqueLimit	Torque Limit Value	Executes a torque limit value change.
MC_SetOverride	Override Value Setting	Changes the target velocity, the target acceleration, and the target deceleration of the specified axis.
MC_ReadParameter	Parameter Read	Reads objects of the slave devices.
MC_WriteParameter	Parameter Write	Writes objects of the slave devices.
MC_Reset	Axis Error Reset	Resets errors and warnings of the axis.
MC_GroupReset	Axes Group Error Reset	Resets errors and warnings of the axes group and each axis belonging to the axes group.
MC_TouchProbe	Touch Probe Enabled	Records optional data when the trigger event occurs.
MC_AbortTrigger	Touch Probe Disabled	Disables the latch that is being executed.
MC_CamTableSelect	Cam Table Selection	Stores the specified operation profile data (cam data) in the open area.
MCv_ChangeCycle	Current Value Change per Cycle	Changes the current value per cycle of the specified operation profile data control FB.
MCv_AllPower	All Axes Operation Available	Switches every axis to the operation possible status.
MC_GroupSetOverride	Axes Group Override Value Setting	Changes the target velocity, the target acceleration, and the target deceleration of the specified axes group.
MCv_MotionErrorReset	Motion Error Reset	Resets all errors and warnings of the motion system.

Operation FBs

The following shows the list of operation motion control FBs.

Motion control FB	Name	Description
MC_Home	OPR	Executes homing for the specified axis.
MC_Stop	Forced Stop	Decelerates the specified axis to stop.
MC_GroupStop	Group Forced Stop	Decelerates the specified axes group to stop.
MC_MoveAbsolute	Absolute Value Positioning	Performs positioning with the absolute target position specified.
MC_MoveRelative	Relative Value Positioning	Performs positioning with the relative movement amount specified.
MCv_Jog	JOG	Performs JOG operation at the command velocity.
MC_MoveVelocity	Speed Control	Switches the driver to csv, and controls the speed to the specified speed.
MC_TorqueControl	Torque Control	Switches the driver to cst, and performs torque control according to the specified target torque.
MCv_SpeedControl	Speed Control (Including Position Loop)	Performs speed control including the position loop.
MCv_MoveLinearInterpolateAbsolute	Absolute Value Linear Interpolation Control	Specifies the absolute target position of the specified axes group, and performs positioning through linear interpolation control.
MCv_MoveLinearInterpolateRelative	Relative Value Linear Interpolation Control	Specifies the relative movement amount of the specified axes group, and performs positioning through linear interpolation control.
MCv_MoveCircularInterpolateAbsolute	Absolute Value Circular Interpolation Control	Performs positioning through 2-axis circular interpolation by setting the absolute end point and sub point using the axis configuration of the set axes group.
MCv_MoveCircularInterpolateRelative	Relative Value Circular Interpolation Control	Performs positioning through 2-axis circular interpolation by setting the relative position from the current position at start to the end point and sub point using the axis configuration of the set axes group.
MC_CamIn	Cam Operation Start	Starts cam operation based on the specified cam data.
MC_GearIn	Gear Operation Start	Starts gear operation according to the specified gear ratio.
MC_CombineAxes	Addition/Subtraction Positioning	Adds or subtracts the movement amount of the two specified master axes, and performs positioning using the value as a command position.
MCv_BacklashCompensationFilter	Backlash Compensation Filter	Executes filter processing to compensate mechanical backlash in the movement direction.
MCv_SmoothingFilter	Smoothing Filter	Executes filter processing based on the specified frequency.
MCv_DirectionFilter	Moving Direction Restriction Filter	Executes filter processing to restrict traveling in the set movement direction.
MCv_SpeedLimitFilter	Speed Limit Filter	Executes filter processing to restrict the velocity of the set limit value.

Standard FBs

The following shows the list of motion control FBs for axis control.

Motion control FB	Name	Description
MCv_ReadProfileData	Profile Read	Reads the specified operation profile data from the open area or file.
MCv_WriteProfileData	Profile Write	Writes the specified operation profile data to the open area or file.

4.3 Axis Setting

Axis

The object to be controlled by the motion system is called the axis.
Axes are classified into two categories: real axes that target the drive modules and I/O devices connected on the network, and virtual axes that generate commands and positions virtually in the motion system.

Category	Axis type	Description
Real axis	Real drive axis	An axis that uses the drive module compatible with the CiA402 drive profile connected to CC-Link IE TSN. It is counted as a control axis.
	Real encoder axis	An axis that generates a current position from the output pulse of the synchronous encoder connected to the drive module on CC-Link IE TSN.
Virtual axis	Virtual drive axis	An axis that can generate a command virtually in the motion system. It does not use actual drive modules.
	Virtual encoder axis	An axis that generates the current position from variables of the motion system. It is used as an input axis of single axis synchronous control.
	Virtual linked axis	An axis that connects FBs of single axis synchronous control. Only the minimum data required for single axis synchronous control is defined.

Maximum number of control axes

The maximum number of axes controlled by the motion system is regarded as the number of actual drive axes. Axes of other types are not included in the count of axes. For details, refer to the following.

Axis type	RD78G4	RD78G8	RD78G16	RD78G32	RD78G64	RD78GHV	RD78GHW
Real drive axis*1	4 axes	8 axes	16 axes	32 axes	64 axes	128 axes	256 axes
Virtual drive axis	Up to 1024 axes can be set. The number of axes that can be set depends on the system memory capacity setting.						
Virtual linked axis							
Real encoder axis							
Virtual encoder axis							

*1 When a multi-axis drive unit and a general output device are used as multiple axes, the number of those axes are counted.
Example: The 2-axis drive unit is counted as 2 axes.

If the number of set axes exceeds the maximum number of control axes, the warning "Warning over maximum number of set axes" (warning code: 0F0BH) is output.
Axes are used as control axes in the order in which they are assigned as axis variables to the global label data. When the maximum number of control axes is reached, Axis status (AxisName.Md.AxisStatus) of the remaining axes becomes "-1: Invalid" and they cannot be used for control.

Required axis settings

To set axes, the following items must be set in the axis setting screen of the engineering tool.
For the axis setting method, refer to ☞ Page 86 Axis parameter setting.

Item	Description
Axis name	Set an axis name.
Axis No.	Set the control axis No. of the motion system.
Axis type	Set the axis type.
Station address	Set the station address of the driver device related to the axis. For multi-axis drivers, set the multi-drop No. as well.
Absolute position control setting	Set the absolute position control method of the axis.
Control cycle	Set the cycle to perform control.

Axis variable

Axis variables are variables related to an axis consisting of parameter data such as axis type and monitor data such as the current position of the axis.

The axes set in the engineering tool are assigned to the global label data as axis variables.

Data type

The data type of the axis variable differs depending on the axis type.

Axis type	Data type
Real drive axis	AXIS_REAL
Real encoder axis	AXIS_ENCODER
Virtual drive axis	AXIS_VIRTUAL
Virtual encoder axis	AXIS_VIRTUAL_ENCODER
Virtual linked axis	AXIS_VIRTUAL_LINK

Point

The axis variables that can be set differ depending on the axis type to be used. For the axis variables that can be set, refer to the following.

MELSEC iQ-R Programming Manual (Motion Control Function Blocks)

Each axis data type has the following members.

Member name	Data type*1*2	Description
AxisRef	AXIS_REF	The data structure for input/output of the motion control FBs. The type is fixed regardless of the axis type.
PrConst	AXIS_□_PRM_CONST	Stores the axis parameter data (constant). The setting value is opened when the axis variable is initialized. Re-importing to the control data is not executed after the axis variable is initialized.
Pr	AXIS_□_PRM	Stores the axis parameter data. The default value is opened when the axis variable is initialized. Re-importing to the control data is executed after the axis variable is initialized. The fetch timing to the control changes depending on the parameter.
Md	AXIS_□_MONI	Stores the axis monitor data. Monitor data is refreshed in the fixed cycle for each axis.
Cd	AXIS_□_CMD	Stores the axis control command data. The latest value is acquired every control operation cycle and used for control.

*1 □: Data type of each axis type

*2 The members of the data type differ depending on the axis type.

AxisName.AxisRef. (Axis information)

The following describes the axis variable "AxisName.AxisRef. (Axis information)" used in this exercise.

For the other axis variables, refer to the following.

MELSEC iQ-R Programming Manual (Motion Control Function Blocks)

Variable name	Name	Import	Data type	Attribute	Description
AxisNo	Axis No.	—	WORD(UINT)	LIST_WRITE_BACK	Set the axis No. • 0:Not set • 1 to 10000: Setting axis No.
StartIO	I/O No.	—	WORD(HEX)	LIST_WRITE_BACK	Set the I/O No. • 000H to 0FFH: Start I/O number (the first 3 digits when expressed as 4-digit hexadecimal number) Set this variable when it is used on the CPU module side. The setting is ignored when used from the Motion module side.

Axis variable initialization timing

Axis variables are initialized at the following timing.

Timing	Processing
When the power is turned ON / When the CPU module is reset	The global label data is referenced and all the set axis variables are initialized.
When the PLC READY [Y0] is turned ON	• Uninitialized axis The global label data is referenced and all the axis variables are initialized. • Initialized axis For the axis parameter data, the global label data is referenced and imported again. The axis is not deleted when a parameter error occurs during the fetch. At this time, READY [X0] does not turn ON. For the label initialization processing when the PLC READY [Y0] is turned ON, refer to "Label initialization function" in the following manual. MELSEC iQ-R Programming Manual (Motion Control Function Blocks)

In the case of a real axis, after initializing the axis variables, a network connection of the relevant device is required to actually operate the axis. If the device with the corresponding station address is already connected, it must be disconnected and then returned. (The axis can be emulated without network connection.)

4.4 ENUM Enumerator

Constants such as Axis status (Axis0001.Md.AxisStatus) and Buffer mode (BufferMode) are defined as ENUM enumerators.
The enumeration type constant used for various parameters, monitor data, and motion control FBs actually uses INT type values.
"Enumeration type name__Enumerator name" INT type global labels are available on the engineering tool.
Note that INT type global labels can only be used for programs created on the Motion module side.
Use a constant for programs created on the CPU module side.
○: Available, ×: Not available

Enumerator	CPU module side			Motion module side
	Ladder	FBD/LD	ST	ST
INT type global label	×	×	×	○*1
Constant	○	○	○	○

*1 Use the motion control setting function version "1.010L" or later.

Ex.
When using the MC_DIRECTION type enumerator "mcPositiveDirection" on the engineering tool
- When using an INT type global label: Set "MC_DIRECTION__mcPositiveDirection".*2
- When using with a constant: Set "1".

*2 Two underscores "__" are required between the numeration type name and enumerator name.

MC_DIRECTION

The following describes the ENUM enumerator "MC_DIRECTION" used in this exercise.

Enumerator	Setting value	Description
mcPositiveDirection	1	Positive direction
mcNegativeDirection	2	Negative direction
mcShortestWay	3	Shortest path
mcCurrentDirection	4	Current direction

MC_BUFFER_MODE

The following describes the ENUM enumerator "MC_BUFFER_MODE" used in this exercise.

Enumerator	Setting value	Description
mcAborting	0	Aborts the execution of the running FB and executes the next FB immediately.
mcBuffered	1	Buffers the next FB on the running FB and executes it sequentially after completion of the running FB.
mcBlendingLow	2	Buffers the next FB on the running FB and executes it sequentially after the axis is moved to the target position by the running FB. When the axis is moved to the target position by the running FB, the lower target velocity between the running FB and buffering FB is used as the switching velocity.
mcBlendingPrevious	3	Buffers the next FB on the running FB and executes it sequentially after the axis is moved to the target position by the running FB. When the axis is moved to the target position by the running FB, the target velocity of the running FB is used as the switching velocity.
mcBlendingNext	4	Buffers the next FB on the running FB and executes it sequentially after the axis is moved to the target position by the running FB. When the axis is moved to the target position by the running FB, the target velocity of the buffering FB is used as the switching velocity.
mcBlendingHigh	5	Buffers the next FB on the running FB and executes it sequentially after the axis is moved to the target position by the running FB. When the axis is moved to the target position by the running FB, the higher target velocity between the running FB and buffering FB is used as the switching velocity.

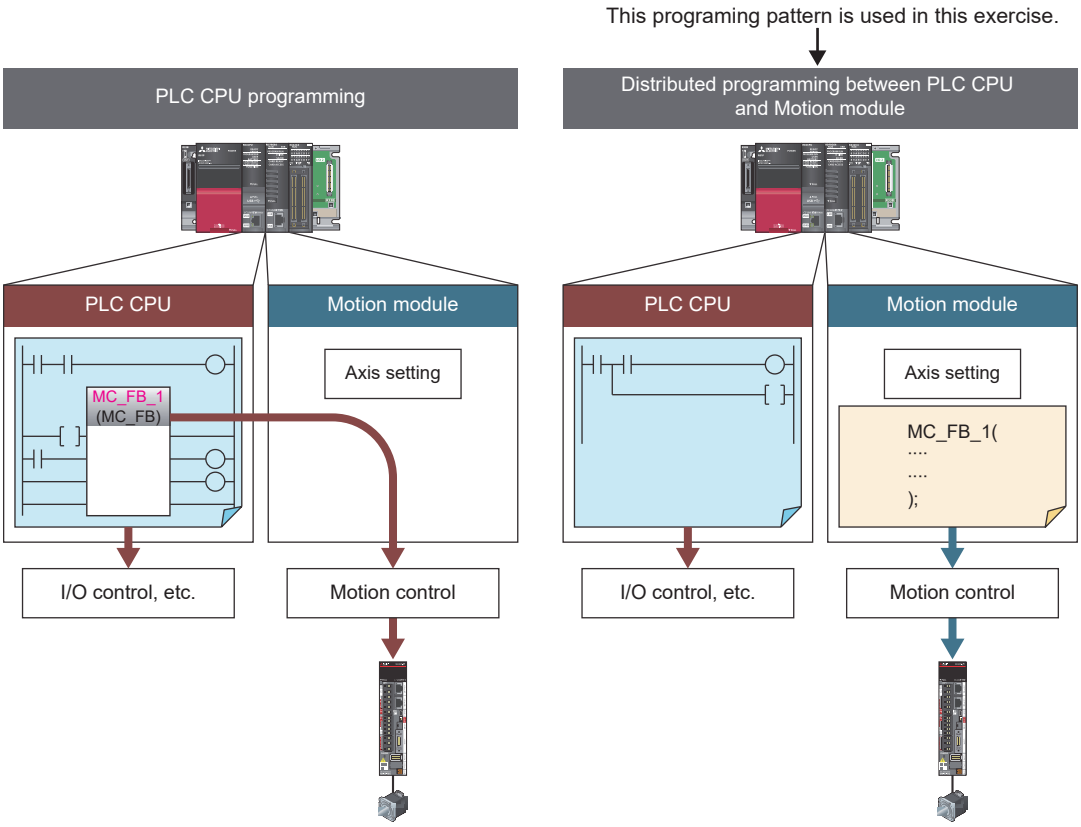
5 PROGRAM

5.1 Programming of Motion Module

The Motion module RD78G(H) can be programmed through PLC CPU programming or distributed programming between the PLC CPU and Motion module.

The PLC CPU is programmed in the ladder, FBD, SFC, and ST language defined in IEC 61131-3, and the Motion module is programmed in the ST language.

This manual describes the distributed programming between the PLC CPU and Motion module using the ST language.



Programming method	Advantage	Disadvantage
PLC CPU programming	Only a single CPU is required for programming, making a program more maintainable.	The scan time is increased because programs are processed by a single CPU.
Distributed programming between PLC CPU and Motion module	Since the programs are separated, motion control does not affect processing time of other sequence programs and the scan time is reduced.	It is necessary to maintain both the PLC CPU program and Motion module program.

5.2 ST Programing

This manual uses the ST language for structured programming.

Structured programming

The following describes the features of structured programs.

Structured design

Structured design is a programming method in which processing is divided into small units (components) so that a set of control by the PLC CPU can be organized in a hierarchical structure. Structured programs can be designed based on the sequence program structure.

The advantages of the hierarchical program are as follows.

- Programming can start with an outline, and can be gradually developed for detail instructions.
- The lowest layer of the hierarchical program is very simple and independent.

The advantages of dividing a program into modules are as follows.

- Visibility of the overall program is enhanced by organizing different actions as separate modules.
- Programming tasks can be assigned to multiple programmers.
- It increases program reusability and improves development efficiency.

Multiple program languages

Multiple program languages are prepared for structured programming. Users can choose and combine the appropriate programming languages for their applications. Each program module can be programmed using different languages.

Name		Description
ST (Structured text)		Text language for computer engineers similar to C and other languages
Structured ladder/FBD	Structured ladder	Graphical language to represent circuits with contacts, coils, and other elements
	FBD	Graphic language to represent circuits by connecting functions and function blocks with lines.

Enhanced program reusability

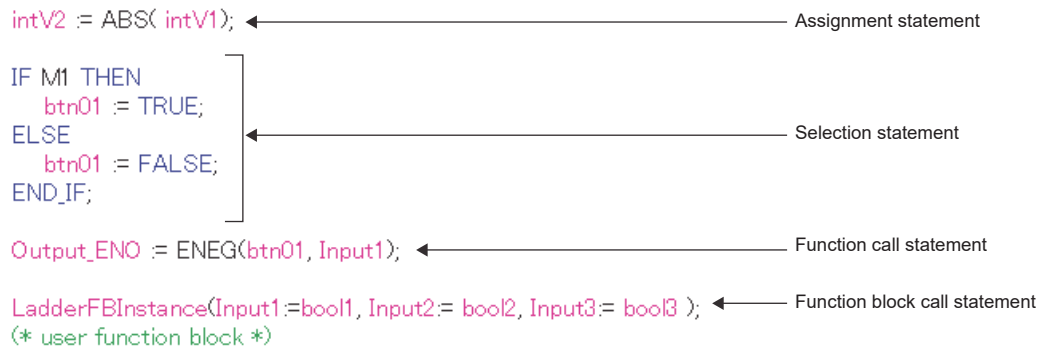
Program modules can be saved in a library. By saving modules in a library, program assets can be shared and its reusability is enhanced.

ST language

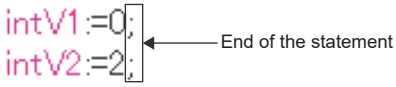
ST language is defined by International Standard IEC61131-3 that defines the logic description system. ST language is a text language having a syntax structure similar to C. This language is suitable for complicated programs that cannot be easily described using the ladder diagram. It improves the visibility of programs because arithmetic operations and data processing can be easily described.

Configuration

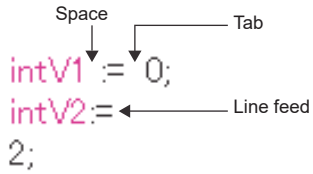
Programs written in ST language consist of operators and control statements.



Each statement must end with a semicolon ";".

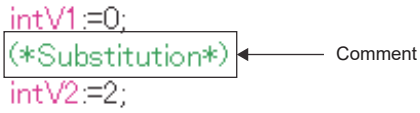


Spaces, tabs, and line feeds can be inserted between an operator and data.



Comments can be inserted into a program.

Start a comment statement with "//" or enclose it with "(*" and "*)" or "/*" and "*/".



■ Program components

An ST program consists of the following components.

For details of each element, refer to the following.

MELSEC iQ-R Programming Manual (Program Design)

Item		Example
Delimiter*1		;, (,)
Operator*1		+, -, <, >, =
Reserved word*1*2	Syntax	IF, CASE, WHILE, RETURN
	Device*3	X0, Y10, M100, ZR0
	Data type	BOOL, DWORD
	Standard function	ADD, REAL_TO_STRING_E
Constant		123, "abc"
Label		Switch_A
Comment		(*Turn on.*)
Other symbols		One-byte space, line feed code, TAB code

*1 Write delimiters, operators, and reserved words in one-byte characters.

*2 For details of the reserved words, refer to the following.

GX Works3 Operating Manual

*3 No device can be described in the ST program for motion control.

Statement and expression

The following describes "statement" and "expression", which constitute the unit for programming in the ST language.

■ Statement

In the ST language, a set of processing to be carried out is called a "statement", which is used for writing a program.

A statement must be written in single-byte characters.

The following table lists the types of statements.

Type		Description
Assignment statement		Assigns the evaluation result of the expression on the right side to the variable on the left side.
Control syntax	Select statement (IF, CASE)	Selects the executable statement according to the conditions.
	Iteration statement (FOR, WHILE, REPEAT)	Executes the executable statement multiple times according to the end condition.
	Interruption of iteration statement (EXIT)	Interrupts the iteration statement.
Subprogram control statement	Call statement	Calls functions and function blocks.
	RETURN statement	Exits the program midway.
Empty statement		No statement is executed.

■ Expression

A description of the values required to process a statement is called an "expression".

An expression consists of variables, operators, and other components. The expression is evaluated during execution of the program.

Operational expressions such as arithmetic and comparison expressions can be described by combining constants and variables with operators, as well as general expressions.

The following table lists the types of expressions.

Type		Data type of expression (operation result)
Operational expression	Arithmetic expression	Integer, real number, etc. (depending on the operation target)
	Boolean expression	Boolean value (TRUE/FALSE)
	Comparison expression	Boolean value (TRUE/FALSE)
Linear expression	Variable, constant	Defined data type
	Function call expression	Data type of the returned value

Operational expression

In the ST language, complex operations including decimal points and exponents can be described as concisely as general arithmetic expressions.

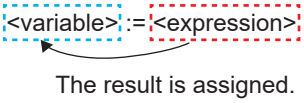
The following describes the operational expressions used in this exercise. For other operational expressions, refer to the following.

MELSEC iQ-R/MELSEC iQ-F Structured Text (ST) Programming Guide Book

■ Assignment (:=)

The result of an expression can be stored in a variable by the assignment statement.

The assignment statement is described with ":=". The result of the expression on the right is stored in the variable on the left.



Point

Although general expressions use "=", the ST language uses the operator with ":" (colon).

■ Four arithmetic operations (+, -, *, /)

Four arithmetic operations are described using the same operators (+, -, *, /) as general arithmetic symbols.

Operations that could not be described at once can be described concisely in a single line of expressions in ladder instructions.

Point

If multiple operational expressions are described in a single statement, the operators are processed in order of precedence.

Order of precedence in four arithmetic operators (in descending order): Multiplication/division (*, /), addition/subtraction (+, -)

If multiple operators have the same precedence, operation is performed from the left.

■ Logical operation (AND, OR, XOR, NOT)

Logical operations are described using operators that are easy to input and understand (AND, OR, XOR, NOT) instead of symbols (∧, ∨, ⊻, etc.).

Point

AND operations can also be described with the operator "&".

If multiple operational expressions are described in a single statement, the operators are processed in order of precedence.

Logical operators in descending order of precedence: Logical negation (NOT), AND operation (AND, &), XOR operation (XOR), OR operation (OR)

If multiple operators have the same precedence, operation is performed from the left.

■ Comparison (<, >, <=, >=), equality/inequality (=, <>)

Comparison operations are described using the same operators as general arithmetic symbols and the inequality signs.

Point

"=" is used in the ST language as an operator that compares whether the values on the right and left are the same.

The assignment statement should be described using ":=" with a semicolon.

Conditional branch

In the ST language, processing that branches according to the conditions can be used in the same way as in high-level programming languages such as C.

The selection statement can be used to describe what is to be done in what case.

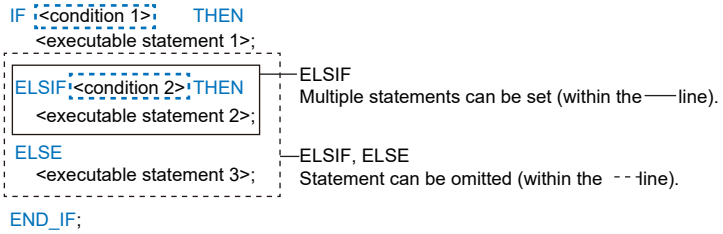
The following describes the conditional branch used in this exercise. For the selection statements, refer to the following.

MELSEC iQ-R/MELSEC iQ-F Structured Text (ST) Programming Guide Book

■ Conditional branch (IF statement) using Boolean values

Processing that branches depending on whether a condition is true (TRUE) or false (FALSE) can be described using an IF statement.

The IF statement performs the following processing.



1. Evaluation of IF

When the conditional expression 1 is true (TRUE), the executable statement 1 is executed.

2. Evaluation of ELSIF

If the previous conditional expressions are false (FALSE), the condition is evaluated.

When the conditional expression 2 is true (TRUE), the executable statement 2 is executed.

3. Evaluation of ELSE

If all the conditional expressions of "IF" and "ELSIF" are false (FALSE), then the executable statement 3 after "ELSE" is executed.

Point

The following can be specified as conditional expressions in the IF statement.

- Operational expression evaluated to a Boolean value
- Boolean variable
- Function call expression that returns a Boolean type value

Multiple conditional branches using ELSIF (ELSIF<conditional expression>THEN<executable statement>;) can be set.

Describe the conditional branches using ELSIF and ELSE as necessary. (They can be omitted.)

Iteration

Iteration statements (WHILE, REPEAT, and FOR) can be used to repeatedly execute the processing until the specified end condition is met.

The following describes the iterations (FOR statements) used in this exercise. For other iteration statements, refer to the following.

MELSEC iQ-R/MELSEC iQ-F Structured Text (ST) Programming Guide Book

■ Iteration until the variable becomes the setting value (FOR)

The FOR statement is used to describe the processing to be iterated until the integral type variable satisfies the condition.

The FOR statement performs the following processing.

The executable statement is executed multiple times according to the end condition of the integer value.

It is executed until the integral type variable where the default value is set becomes the final value.

```
FOR <variable> := <default value (expression)>
 TO  <final value (expression)>
 BY  <increment value (expression)>   ├─ BY (within the - - - line) can be omitted
 DO <executable statement>:
END_FOR;
```

1. Initializing the variable

Set the default value for the variable to be used as the condition.

2. Evaluation of the condition

If the variable is the final value, the processing ends.

3. Executing the processing

Execute the executable statement.

4. Adding the incremental value

Add the incremental value to the variable and iterate the processing.

Point

The following can be specified for the default, final, and increment values of the FOR statement.

- Operational expression whose result is an integer (INT, DINT) value
- Variable of integral data (INT, DINT) type
- Function call expression that returns an integral (INT, DINT) type value

Convert the value to the integral type.

If the increment value is 1, the description of the addition processing for the increment value (BY <increment value(expression)>;) can be omitted.

The variable set as a condition retains its value at the end of the FOR statement.

Comparison between ST language and ladder diagram

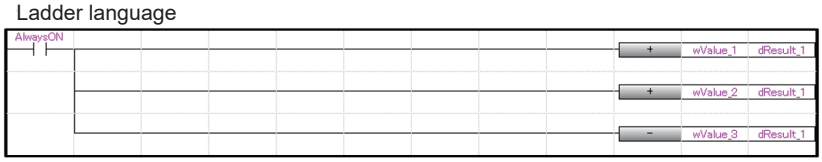
The following compares and explains the structures of the ST language and ladder diagram.

Four arithmetic operations

A four arithmetic operations program can be described in the ladder diagram and ST language as follows.
Compared to the ladder diagram, an expression can be written in a single line in the ST language.

Addition/subtraction

Ladder language



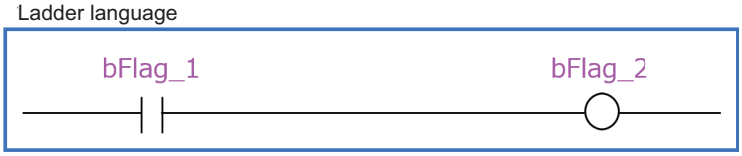
ST language

```
dResult_1 := wValue_1 + wValue_2 - wValue_3;
```

Conditional branch (IF statement)

An IF statement can be described in the ladder diagram and ST language as follows.
Compared to the ladder diagram, the processing flow of the program can be clearly described in the ST language.

Ladder language



ST language

```
IF bFlag_1 = TRUE THEN
   bFlag_2 := TRUE;
END_IF;
```

Logical operation

A logical operation can be described in the ladder diagram and ST language as follows.
The ST language uses logical operators to combine conditions, thereby making programs easy to input and understand.

AND (logical conjunction)

Ladder language

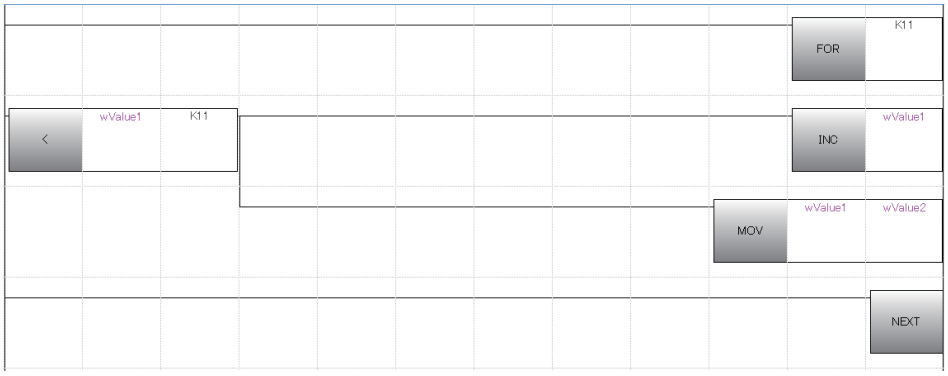
ST language

```
bResult := bFlag0 AND bFlag1;
```

Iteration

An iteration can be described in the ladder diagram and ST language as follows.
In the ladder diagram, the number of iterations is specified; in the ST language, the condition for iteration is specified.
Compared to the ladder diagram, easy-to-understand programs can be created in the ST language because operations can be grouped together like mathematical expressions.

Ladder language



Internal processing of ladder diagram

Count	wValue1	wValue2
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	6
7	7	7
8	8	8
9	9	9
10	10	10
11	11	11

Processing ends after the specified number of iterations

ST language

```
1 FOR wValue1 := 1
2     TO 10
3     BY 1
4     DO wValue2 := wValue1 + 1;
5 END_FOR;
```

Internal processing of ST language

Count	wValue1	wValue2
1	1	2
2	2	3
3	3	4
4	4	5
5	5	6
6	6	7
7	7	8
8	8	9
9	9	10
10	10	11
11	11	-

Executes DO processing up to here

Ends the program by executing only BY processing after the specified value is obtained

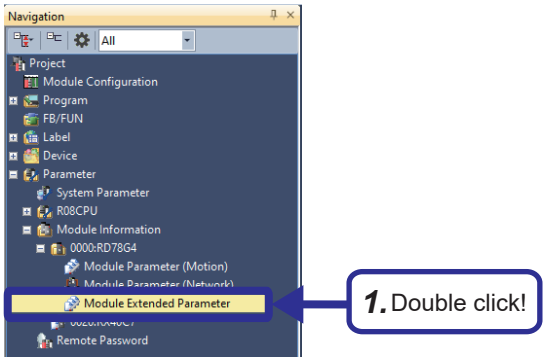
5.3 Creating an ST Program

Create an ST program that executes four arithmetic operations and check the operation.

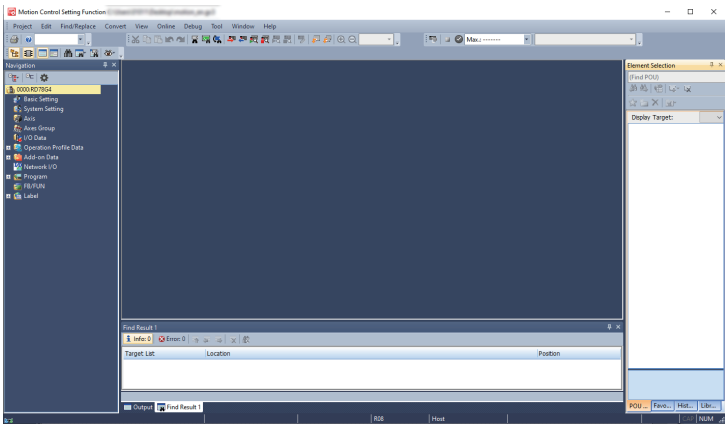
Opening a project

In "school_test.gx3", the module configuration settings and PLC programs are already prepared for this exercise.
Program the Motion module with the motion control setting function.

Operating procedure



1. Open school_test.gx3, select [Parameter] ⇨[Module Information] ⇨ [0000: RD78G4] from the "Navigation" window, and double-click [Module Extended Parameter].

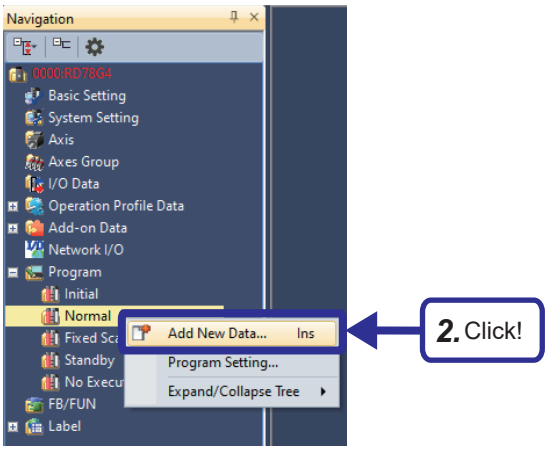


2. The motion control setting function is activated.

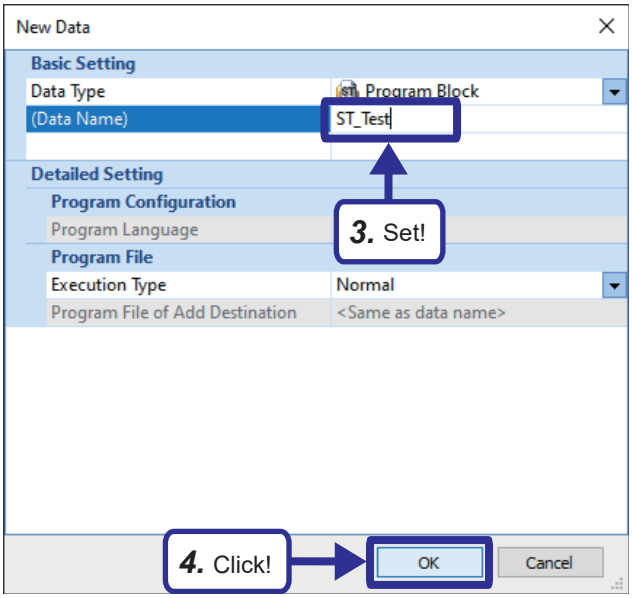
Creating a program block

The following describes how to create a program block.

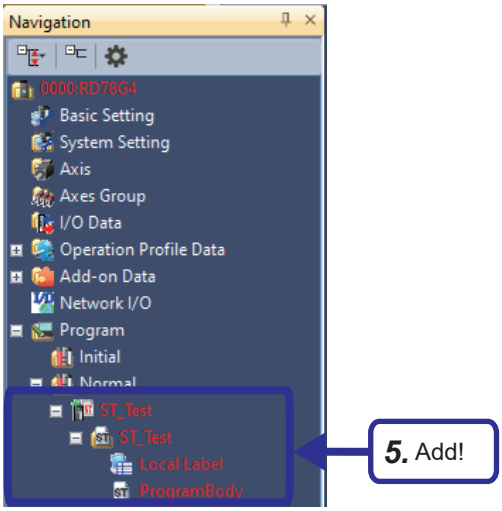
Operating procedure



1. In the "Navigation" window of the motion control setting function, right-click [Normal] under [Program].
2. Click [Add New Data].



3. In the New Data window, set a data name.
4. Click the [OK] button.



5. The program block is added to the Navigation window.

Setting the execution type

Set the execution type to determine when to execute the program.

There are four execution types of the program.

Execution type	Description
Initial	The program is executed only once when the PLC READY [Y0] is turned ON.
Normal	The program is executed by the normal task of the motion system.
Fixed Scan	The program is executed at every specified time (1st operation cycle 62.5[us] to 60000ms]). It must be longer than the operation cycle.
Standby	The program is executed upon request. The PSCAN instruction changes a running program to a normal execution type program and executes it.

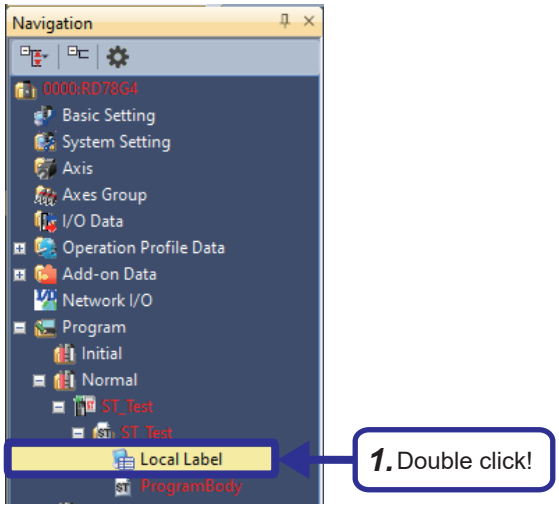
Registering a local label

The following describes how to register a local label.

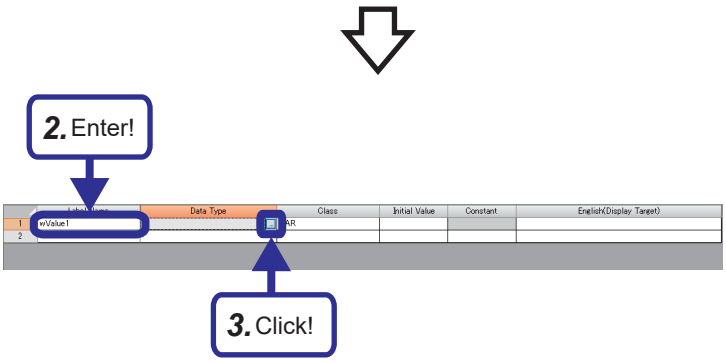
The following table lists the local labels used in this program.

Category	Label name	Data type	Class	Public label	Description
Local label	wValue1	Word [signed]	VAR	—	Variable 1
	wValue2	Word [signed]	VAR	—	Variable 2
	wSumValue	Word [signed]	VAR	—	Stores the sum of variable 1 and variable 2.
	wDifferenceValue	Word [signed]	VAR	—	Stores the difference between variable 1 and variable 2.
	wProductValue	Word [signed]	VAR	—	Stores the product of variable 1 and variable 2.
	wQuotientValue	Word [signed]	VAR	—	Stores the quotient of variable 1 and variable 2.

Operating procedure

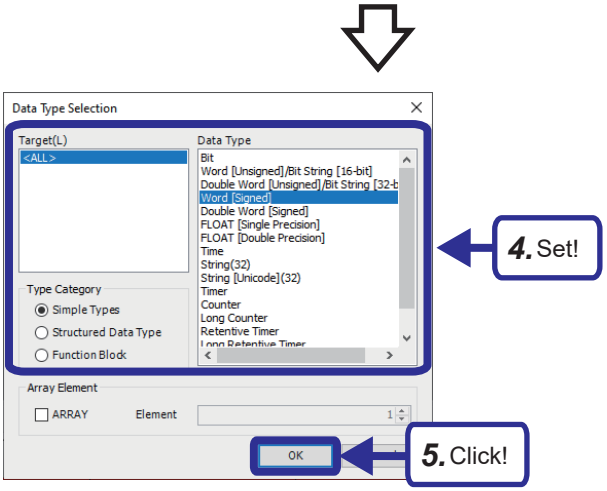


1. In the "Navigation" window, click [Program] ⇨[Normal] ⇨ [ST_Test] ⇨ [ST_Test] and double-click [Local Label].



2. Enter the label name "wValue1".

3. Click the [...] button.



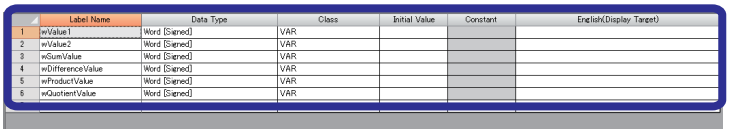
4. The "Data Type Selection" window appears.
Set each item as follows.

[Setting details]

Type Category: Simple Types

Data Type: Word [signed]

5. Click the [OK] button.



	Label Name	Data Type	Class	Initial Value	Constant	English(Display Target)
1	wValue1	Word [Signed]	VAR			
2	wValue2	Word [Signed]	VAR			
3	wSumValue	Word [Signed]	VAR			
4	wDifferenceValue	Word [Signed]	VAR			
5	wProductValue	Word [Signed]	VAR			
6	wQuotientValue	Word [Signed]	VAR			

6. Set the following local labels in the same way.

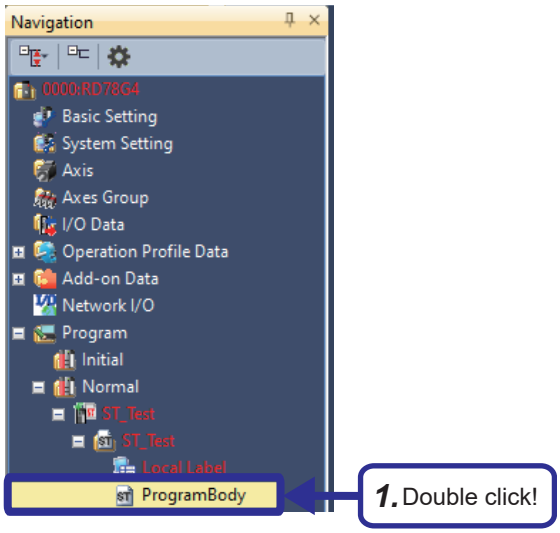
[Setting details]

- wValue2
- wSumValue
- wDifferenceValue
- wProductValue
- wQuotientValue

Creating a program

Create an ST program that executes four arithmetic operations.

Operating procedure



1. In the "Navigation" window, click [Program] ⇨[Normal] ⇨ [ST_Test] ⇨ [ST_Test] and double-click [ProgramBody].

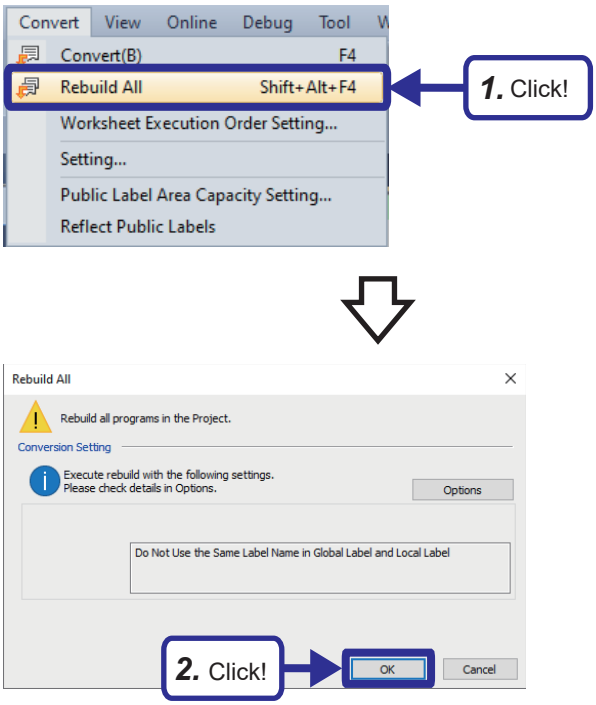
```
1  //set default value for variable
2  wValue1:=100;
3  wValue2:=10;
4
5  wSumValue := wValue1 + wValue2;
6  wDifferenceValue := wValue1 - wValue2;
7  wProductValue := wValue1 * wValue2;
8  wQuotientValue := wValue1 / wValue2;
```

2. Create a program as shown on the left.

Converting all data in the program

After creating a program, convert all data in the program.
Convert the sequence program in GX Works3, and convert the motion control program in the Motion Control Setting Function.
This exercise uses the prepared sequence program and does not require the conversion of the program for the programmable controller.

Operating procedure



1. Click [Convert] ⇨[Rebuild All] from the menu of the motion control setting function.

2. Click the [OK] button.

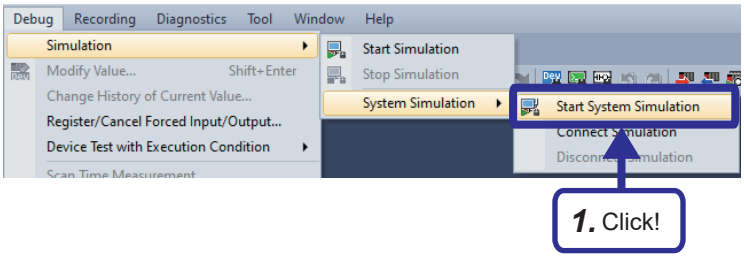
Operation check

Use the simulation function to check the operation of the four arithmetic operations program offline.
Simulation is a function used to debug programs using a virtual programmable controller on the personal computer.
GX Simulator3 is used for the simulation function.

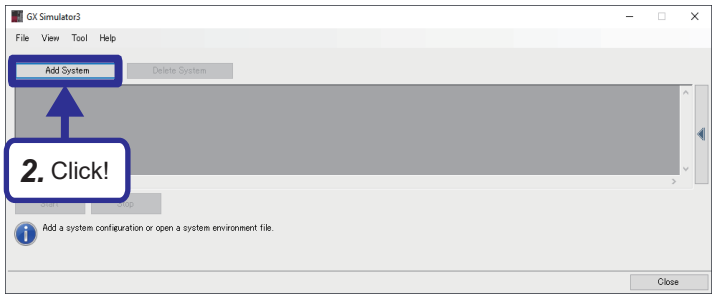
Starting GX Simulator3 and MU Simulator

MU Simulator is a simulator for CC-Link IE TSN-compatible Motion modules.

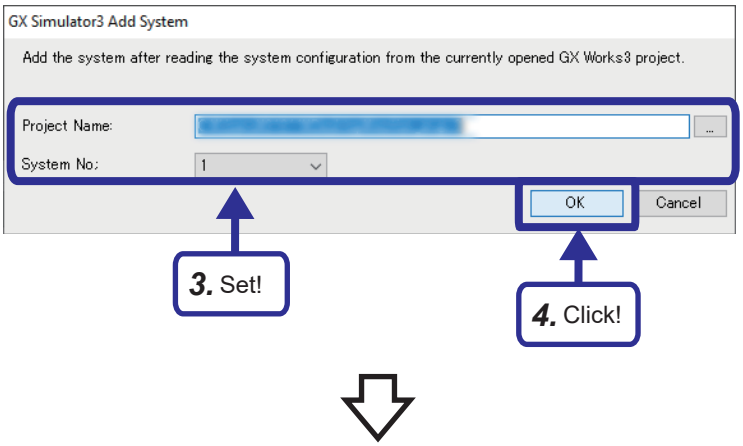
Operating procedure



1. From the menu of GX Works3, click [Debug] ⇨[Simulation] ⇨ [System Simulation] ⇨ [Start System Simulation] to start GX Simulator3.

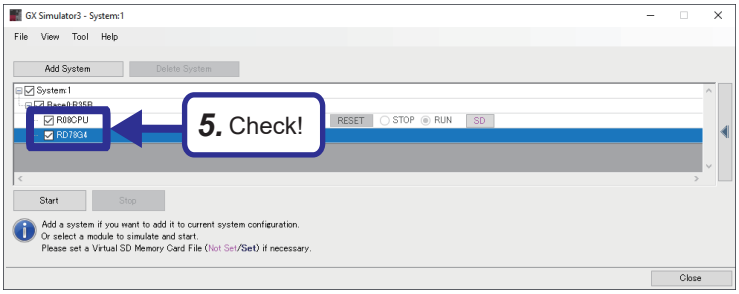


2. Click the [Add System] button in the GX Simulator3 screen.



3. Set the project name and system No. in the GX Simulator3 Add System window.

4. Click the [OK] button.



5. In the GX Simulator3 screen, select the modules to be simulated and click the [Start] button.

[Setting details]
R08CPU:Select
RD78G4:Select

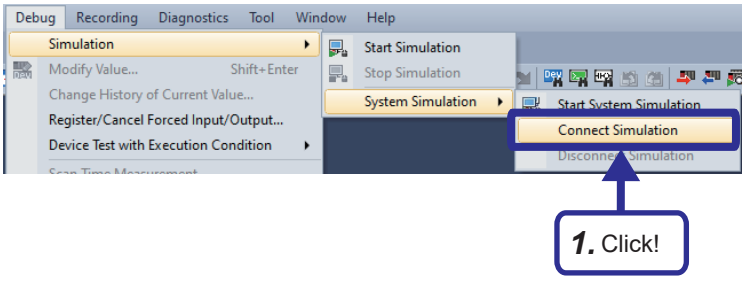
Point

Even if the error LED is on, there is no problem for now. Proceed to the next procedure with the window shown on the left open.

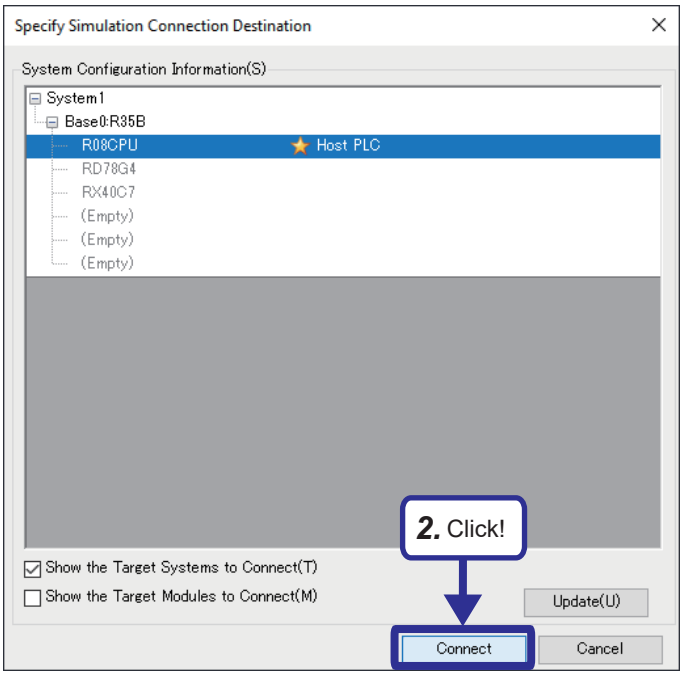
6. MU Simulator starts.

Connection and data writing to the CPU module

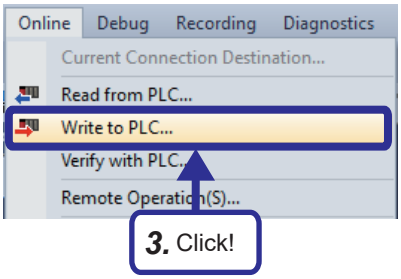
Operating procedure



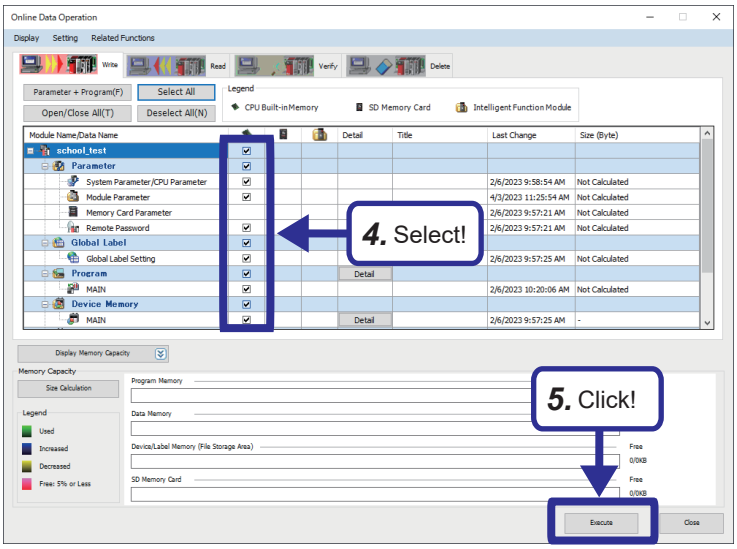
1. Select [Debug] ⇨[Simulation] ⇨ [System Simulation] ⇨ [Connect Simulation] from the menu of GX Works3.



2. In the Specify Simulation Connection Destination window, select the CPU module which was selected in the GX Simulator3 screen, and click the [Connect] button.

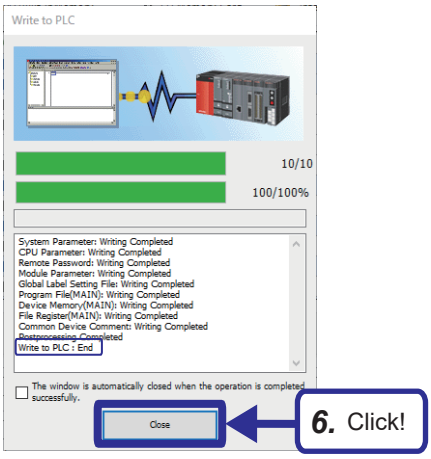


3. Click [Online] ⇨[Write to PLC] from the menu of MELSOFT GX Works3.

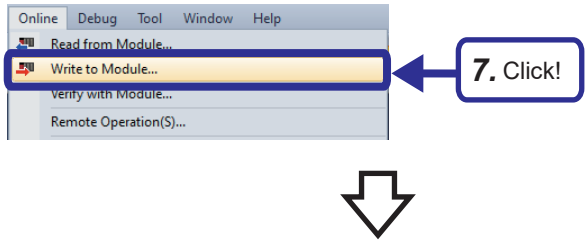


4. The Online Data Operation window appears. Select the items to be written.

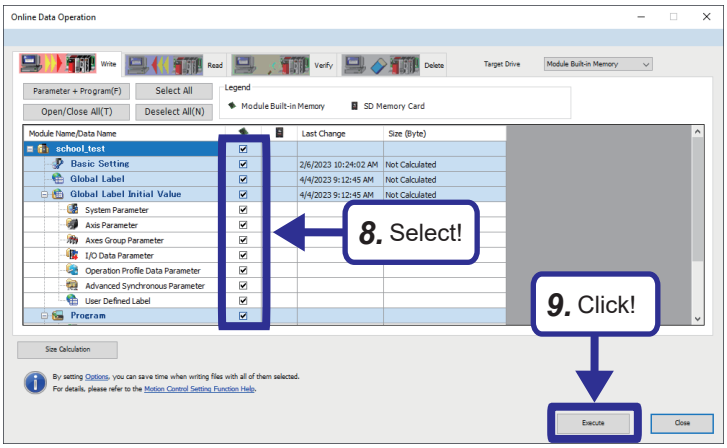
5. Click the [Execute] button.



6. The Write to PLC dialog box appears. Once writing is completed, the message "End" appears. Click [Close].

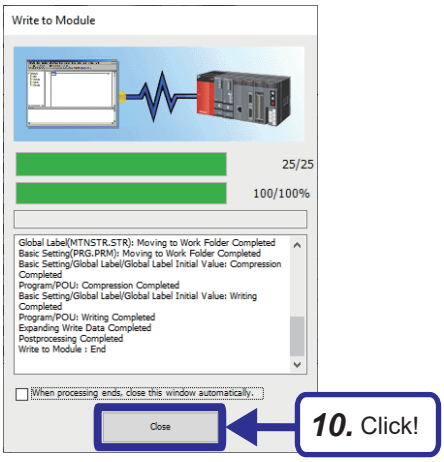


7. Click [Online] ⇨[Write to Module] from the menu of the motion control setting function.



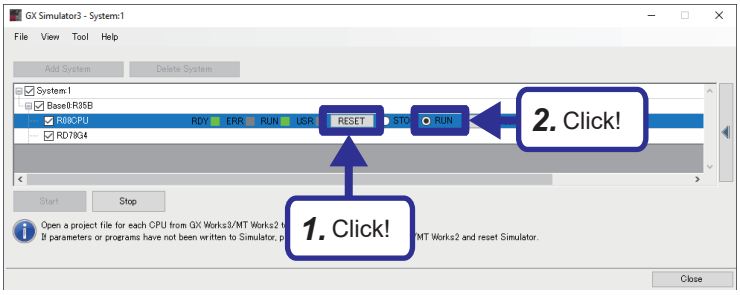
8. The Online Data Operation window appears. Select the items to be written.

9. Click the [Execute] button.



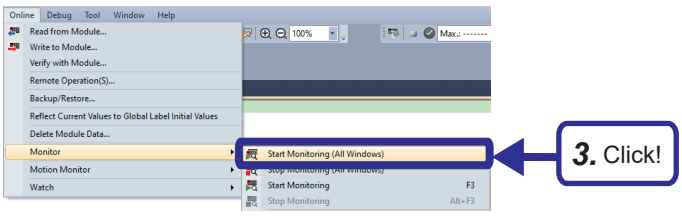
10. The Write to Module dialog box appears. Once writing is completed, the message "End" appears. Click [Close].

Executing the simulation

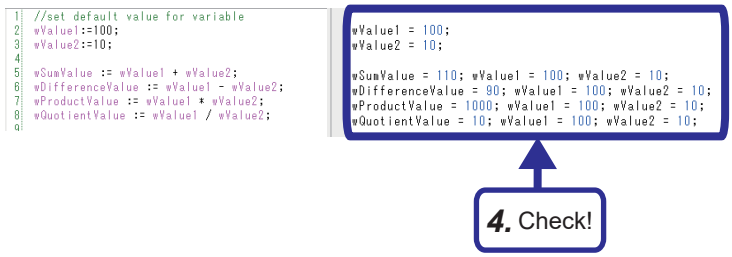


1. Click the [RESET] button of the CPU module in the GX Simulator3 screen.

2. Select "RUN" for the CPU module.

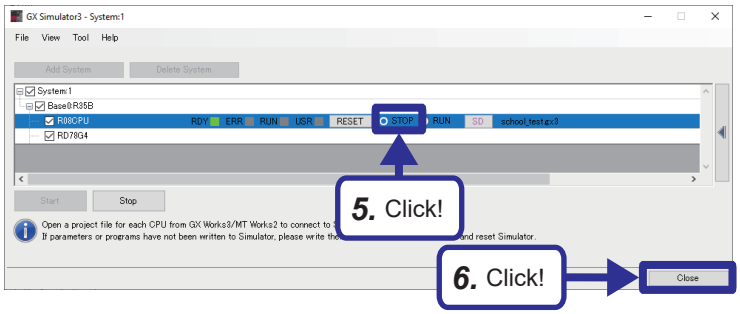


3. Click [Online] ⇨[Monitor] ⇨ [Start Monitoring (All Windows)] from the menu of the motion control setting function.



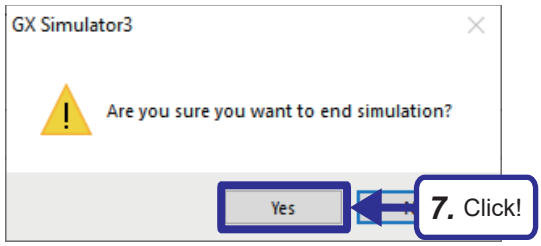
4. Check the results of the four arithmetic operations.

On the right side of the program editor, the same number of lines of device values as the program are automatically displayed as a monitor screen.

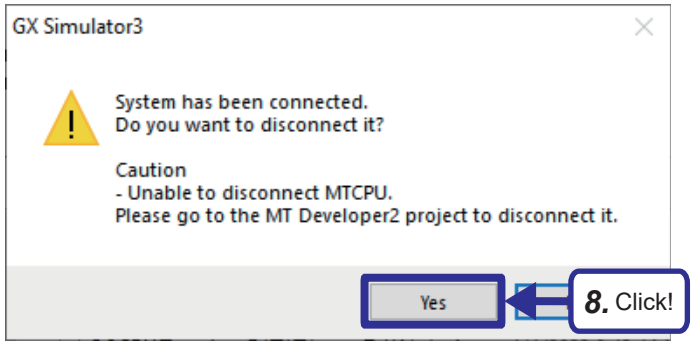


5. Click the [STOP] button of the CPU module in the GX Simulator3 screen.

6. Click the [Close] button.



7. The dialog box shown on the left appears. Click the [Yes] button.



8. The dialog box shown on the left appears. Click the [Yes] button.

9. Change the variable and check the results of the four arithmetic operations.

Practices

Add local labels and programs to the created "school_test.gx3" and check the operation.

■ Local label

Register the following local labels. For how to register the local labels, refer to the following.

☞ Page 59 Registering a local label

Category	Label name	Data type	Class	Public label	Description
Local label	wRemainder	Word [signed]	VAR	—	Stores the remainder of the integer division.
	bFlag1	Bit	VAR	—	Stores the ON/OFF state of Flag1.
	bFlag2	Bit	VAR	—	Stores the ON/OFF state of Flag2.
	bFlag3	Bit	VAR	—	Stores the ON/OFF state of Flag3.
	bFlag4	Bit	VAR	—	Stores the ON/OFF state of Flag4.
	wValue3	Word [signed]	VAR	—	Stores the value according to the condition of the IF statement.
	wValue4	Word [signed]	VAR	—	Stores the absolute value of the set value.
	eValue5	Single-precision real number	VAR	—	Stores the set value in single-precision real number.
	wValue5	Word [signed]	VAR	—	Stores the default value of the variable to be used for the condition of the FOR statement.
	wValue6	Word [signed]	VAR	—	Stores the result of the iteration by the FOR statement.

■ Program

In the program block "ST_Test", change the values and add the program in the red frames below.

```
1  //set default value for variable
2  wValue1:=100;
3  wValue2:=30; //wValue2 10→30
4
5  wSumValue := wValue1 + wValue2;
6  wDifferenceValue := wValue1 - wValue2;
7  wProductValue := wValue1 * wValue2;
8  wQuotientValue := wValue1 / wValue2;
9
10 wRemainder:=(wValue1 MOD wValue2); //calculate remainder
11
12 //IF statement
13 IF bFlag1 THEN;
14     wValue3:=200;
15     ELSIF bFlag2 THEN;
16     wValue3:=100;
17     ELSE
18     wValue3:=0;
19 END_IF;
20
21 wValue4:=ABS(-1234); //absolute value
22
23 eValue5:=33.3; //single-precision real number
24
25 //invert single-precision real number data
26 bFlag4:=ENEG(bFlag3,eValue5);
27
28 //FOR statement
29 FOR wValue5 := 0 //define wValue5 as 0
30     TO 30 //iterate until wValue5 becomes 30
31      BY 1 //add 1 to wValue5 at completion of processing
32      DO wValue6 := wValue5 + 1;  //1st time: wValue6 = 0 + 1, 2nd time: wValue6 = 1 + 1, 3rd time: wValue6 = 2 + 1 ... 31st time: wValue6 = 30 + 1
33 END_FOR;
```

■ Operation check

Check the operation of the program offline using the simulation function.

For how to start the simulator, refer to ☞ Page 63 Starting GX Simulator3 and MU Simulator.

Register bFlag1, bFlag2, and bFlag3 in the watch window, and switch the variables on and off to check that the stored values are changed.

```
1  //set default value for variable
2  wValue1:=100;
3  wValue2:=30; //wValue2 10→30
4
5  wSumValue := wValue1 + wValue2;
6  wDifferenceValue := wValue1 - wValue2;
7  wProductValue := wValue1 * wValue2;
8  wQuotientValue := wValue1 / wValue2;
9
10 wRemainder:=(wValue1 MOD wValue2); //calculate remainder ---------- (1)
11
12 //IF statement
13 IF bFlag1 THEN;
14     wValue3:=200;
15     ELSIF bFlag2 THEN;
16     wValue3:=100;
17     ELSE
18     wValue3:=0;
19 END_IF;
20
21 wValue4:=ABS(-1234); //absolute value ---------- (3)
22
23 eValue5:=33.3; //single-precision real number ---------- (4)
24
25 //invert single-precision real number data
26 bFlag4:=ENEG(bFlag3,eValue5); ---------- (5)
27
28 //FOR statement
29 FOR wValue5 := 0 //define wValue5 as 0
30     TO 30 //iterate until wValue5 becomes 30
31     BY 1 //add 1 to wValue5 at completion of processing
32     DO wValue6 := wValue5 + 1;  //1st time: wValue6 = 0 + 1, 2nd time: wValue6 = 1 + 1,
33 END_FOR;
```

(2)

(6)

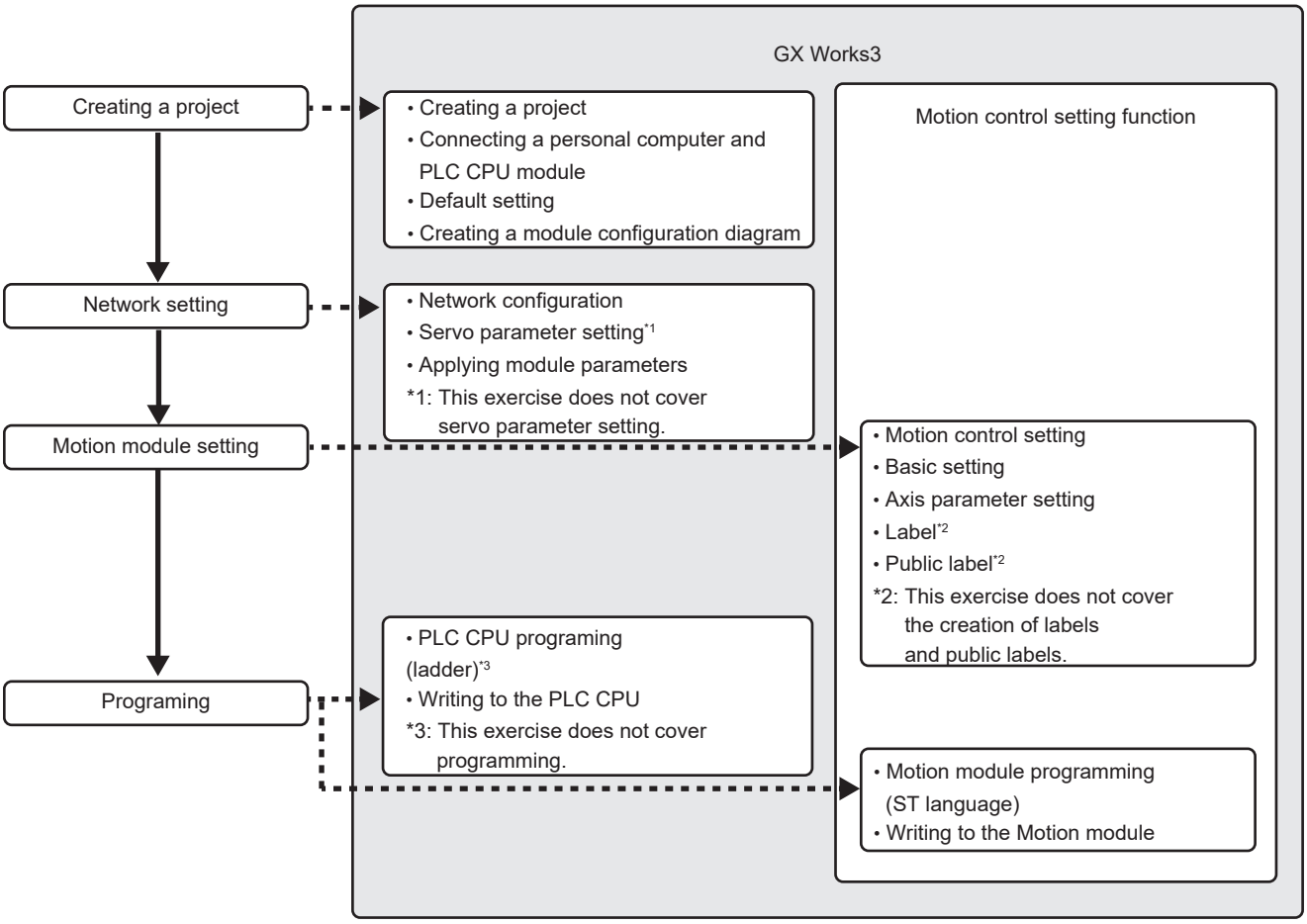
```
wValue1 = 100;
wValue2 = 30;

wSumValue = 130; wValue1 = 100; wValue2 = 30;
wDifferenceValue = 70; wValue1 = 100; wValue2 = 30;
wProductValue = 3000; wValue1 = 100; wValue2 = 30;
wQuotientValue = 3; wValue1 = 100; wValue2 = 30;

wRemainder = 10; wValue1 = 100; wValue2 = 30;



wValue3 = 200;

wValue3 = 200;

wValue3 = 200;


wValue4 = 1234;

eValue5 = 33.300;


eValue5 = 33.300;


wValue5 = 31;

wValue6 = 31; wValue5 = 31;
```

No.	Description
(1)	Stores the remainder of the division of wValue1 and wValue2. In this program, the remainder "10" when 100 is divided by 30 is stored in wRemainder.
(2)	Stores different values in wValue3 depending on whether a condition of the IF statement is true or false. In this program, processing according to the conditions is as follows. • When "bFlag1" of the conditional expression 1 is true (TRUE): Stores "200" in wValue3. • When "bFlag1" of the conditional expression 1 is false (FALSE) and "bFlaf2" of the conditional expression 2 is true (TRUE): Stores "100" in wValue3. • When "bFlag1" of the conditional expression 1 and "bFlaf2" of the conditional expression 2 are false (FALSE): Stores "0" in wValue3.
(3)	Stores the absolute value of the set value (-1234) in wValue4.
(4)	Stores the set value (33.3) as the single-precision real number data in wValue5.
(5)	Inverts the sign of the 32-bit floating type real number data. In this program, by turning on "bFlag3", the sign of the value stored in eValue5 is inverted and the value becomes "-33.300". "bFlag4" stores the execution results.
(6)	Iterates the processing until the set variable satisfies the condition. In this program, the processing is iterated until wValue5 becomes 30. The processing is iterated 31 times because wValue5 is incremented per processing. wValue6 stores the value obtained by adding 1 to the current value of wValue5. When "wValue5 = 30", the processing of "wValue6 := wValue5 + 1" of DO is executed, and after that, 1 is added to wValue5 in the BY processing, resulting in "wValue5 = 31". Since the iteration ends here, the final values of wValue5 and wValue6 are both 31.

6 EXERCISE 1 PROJECT STARTUP

6.1 Setting Procedure Before Operation

The following shows the setting procedure for operating the demonstration machine to be used in this exercise.

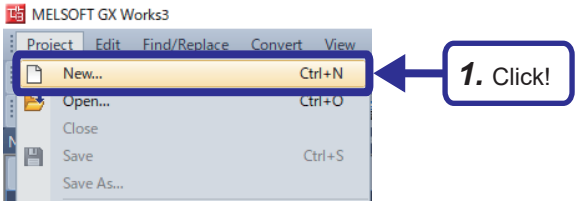


6.2 Creating a Project

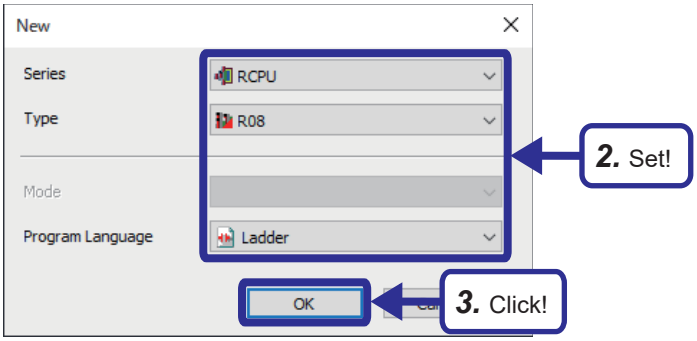
Creating a new project

Create a new project.

Operating procedure



1. Open GX Works3 and click [Project] ⇨[New] from the menu.



2. The New window appears. Set the items as follows.

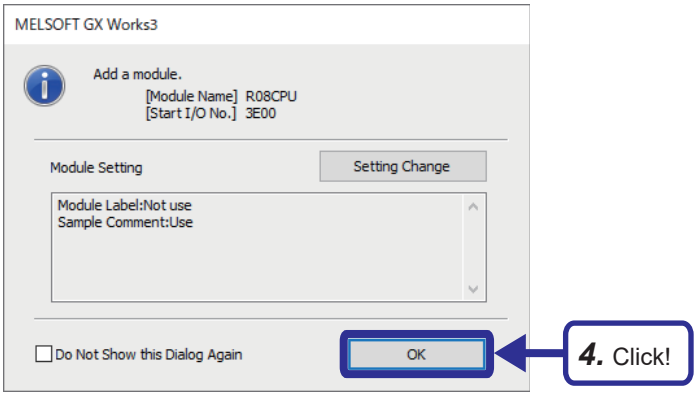
[Setting details]

Series: RCPU

Type: R08

Program Language: Ladder

3. Click the [OK] button.

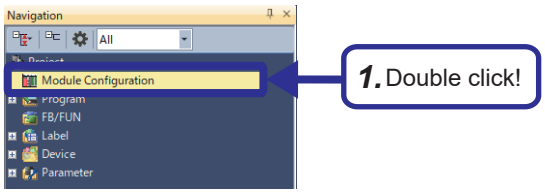


4. Click the [OK] button.

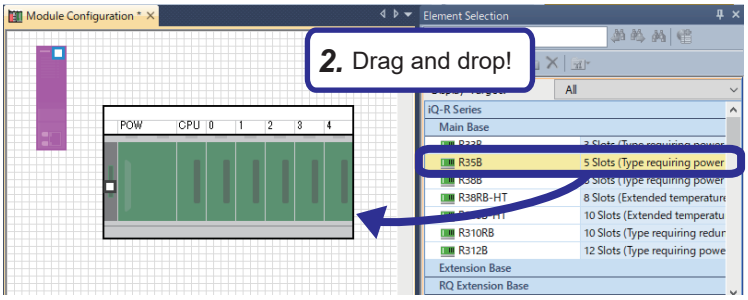
Module configuration diagram setting

Set the module configuration diagram based on the system configuration of the demonstration machine.

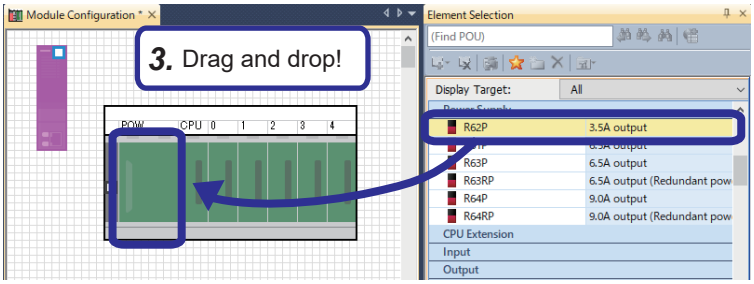
Operating procedure



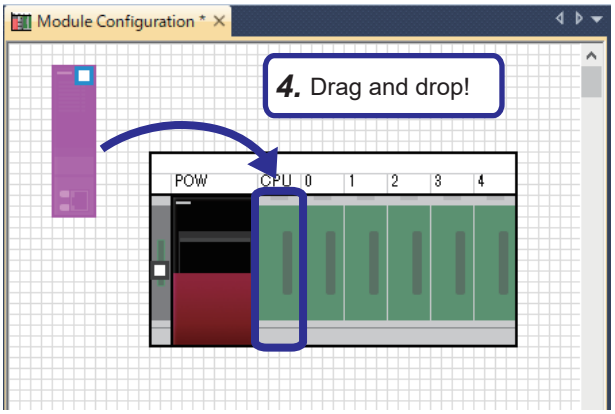
1. Double-click "Module Configuration" from the Navigation window.



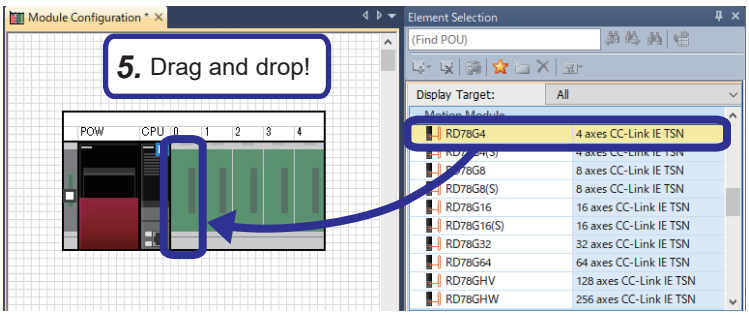
2. Select "R35B" from Main Base in the Element Selection window, and drag and drop it to the module configuration diagram.



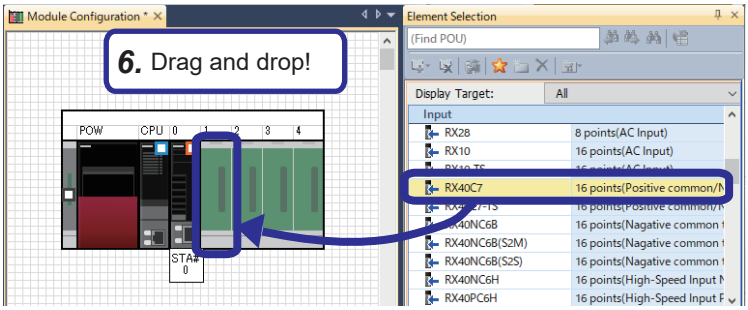
3. Select "R62P" from Power Supply in the Element Selection window, and drag and drop it to the power supply slot of R35B which was added to the module configuration diagram.



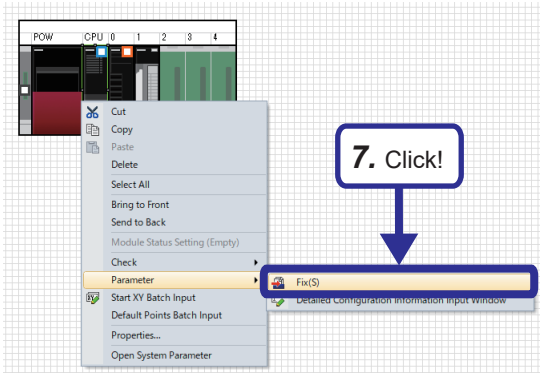
4. Drag and drop R08CPU which is initially displayed in the Module Configuration window to the CPU slot of R35B.



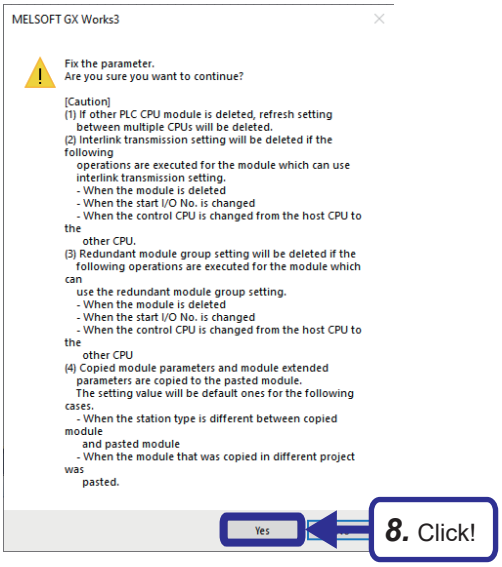
5. Select "RD78G4" from Motion Module in the Element Selection window, and drag and drop it to slot No. 0 of R35B.



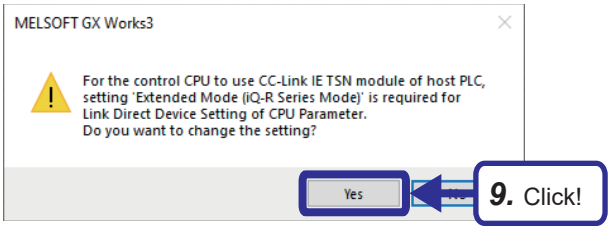
6. Select "RX40C7" from Input in the Element Selection window, and drag and drop it to slot No. 1 of R35B.



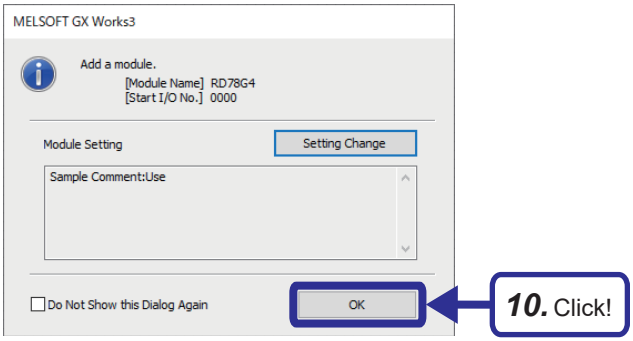
7. Click [Parameter] from the right-click menu and click [Fix(S)] to confirm the parameter.



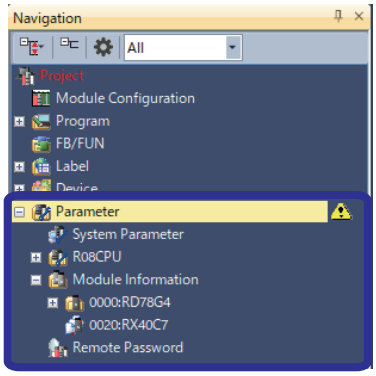
8. A dialog box appears to confirm the parameter.
Click the [Yes] button.



9. Click the [Yes] button.



10. Click the [OK] button.



11. The module information is automatically registered to the "Navigation" window.

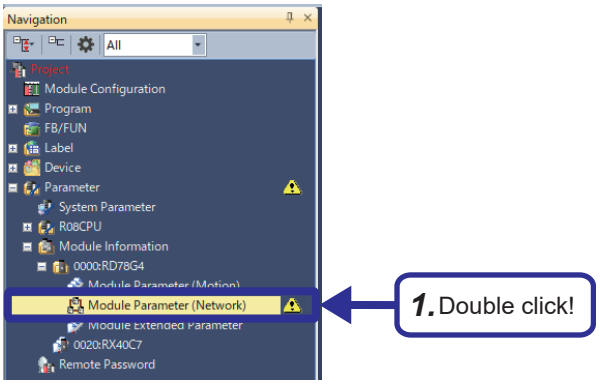
Module parameter setting

The following describes the parameter settings required for communicating with other stations in a Motion module.

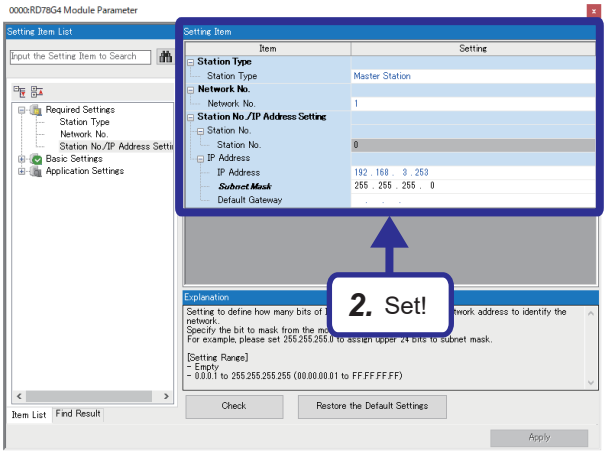
Network configuration setting

Set the slave devices such as servo amplifiers to be connected to CC-Link IE TSN.

Operating procedure



1. Select [Parameter] ⇨[Module Information] ⇨ [0000: RD78G4] from the "Navigation" window, and double-click [Module Parameter (Network)].



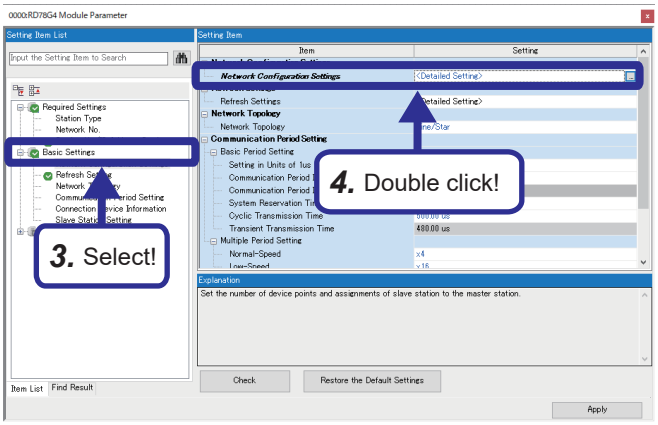
2. Select "Required Settings" from "Setting Item List" and set the items as follows.

[Setting details]

Network No.: 1 (Default)

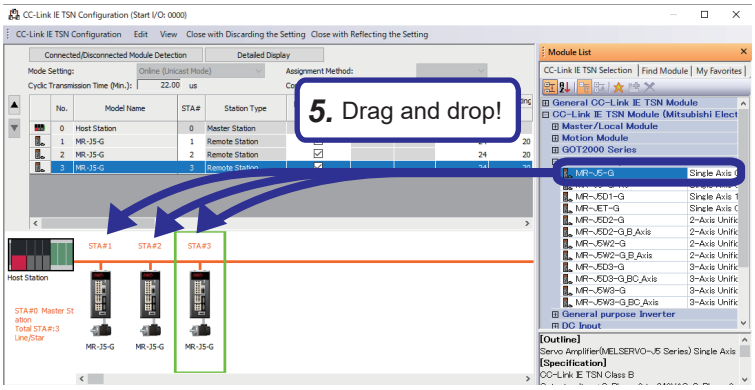
IP Address: 192.168.3.253 (Default)

Subnet Mask: 255.255.255.0



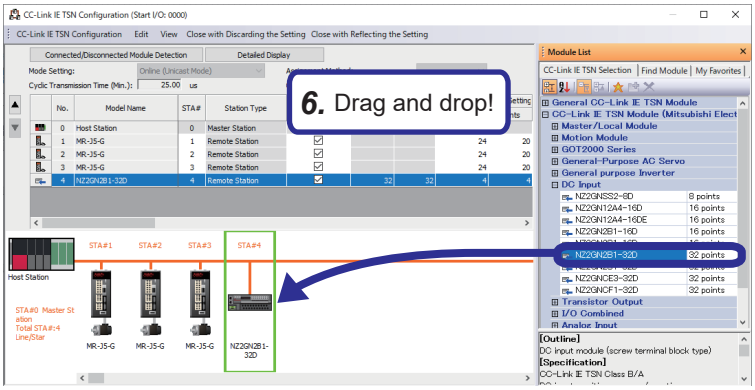
3. Select "Basic Settings" from Setting Item List.

4. Double-click "Network Configuration Settings".

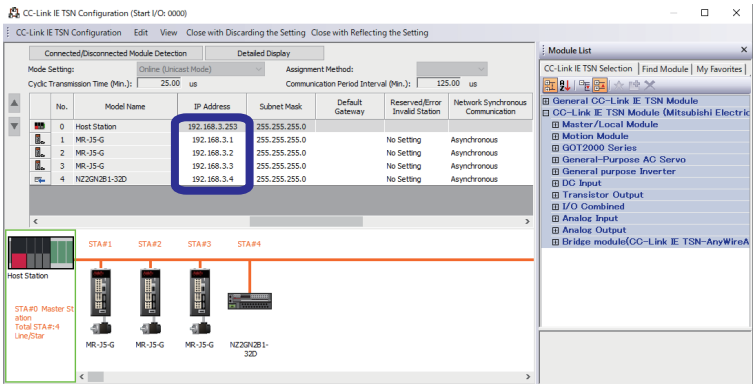


5. Click [CC-Link IE TSN Module (Mitsubishi Electric)] ⇨[General-Purpose AC Servo] in Module List in the "CC-Link IE TSN Configuration" window. Select [MR-J5-G] and drag and drop it.

In this exercise, register three servo amplifiers to control three axes.



6. Click [CC-Link IE TSN Module (Mitsubishi Electric)] ⇨[DC Input] in Module List in the "CC-Link IE TSN Configuration" window. Select [NZ2GN2B1-32D] and drag and drop it.



7. The IP addresses are automatically assigned in the dropped order. Check that the IP addresses are set as follows.

[Setting details]

MR-J5-G:192.168.3.1 (Default)

MR-J5-G:192.168.3.2 (Default)

MR-J5-G:192.168.3.3 (Default)

NZ2GN2B1-32D:192.168.3.4 (Default)

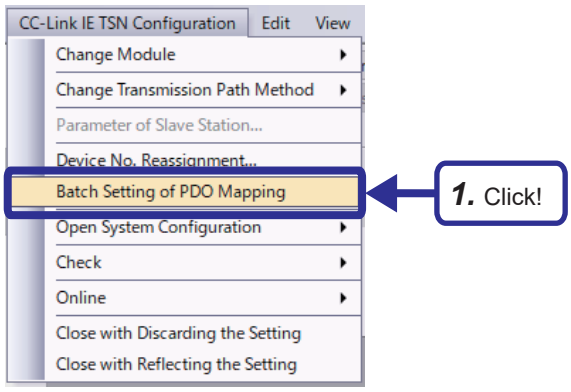
PDO mapping

PDO is an abbreviation for Process Data Object. The PDO communication is equivalent to the conventional CC-Link cyclic communication.

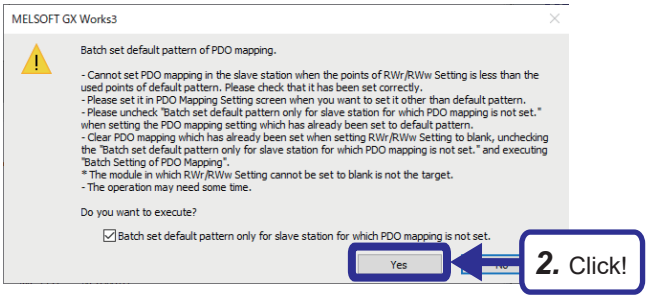
The PDO mapping is the processing of mapping (associating) the data (objects) to be exchanged between the controller and slave through the cyclic communication (PDO communication) in advance.

Set the default data for PDO mapping at once.

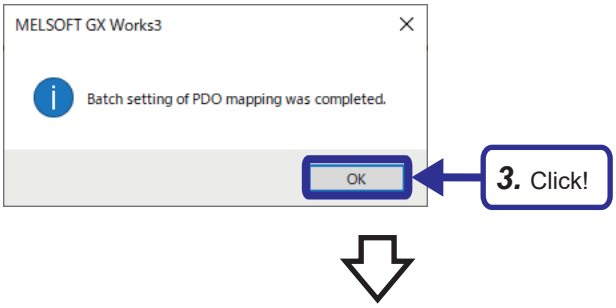
Operating procedure



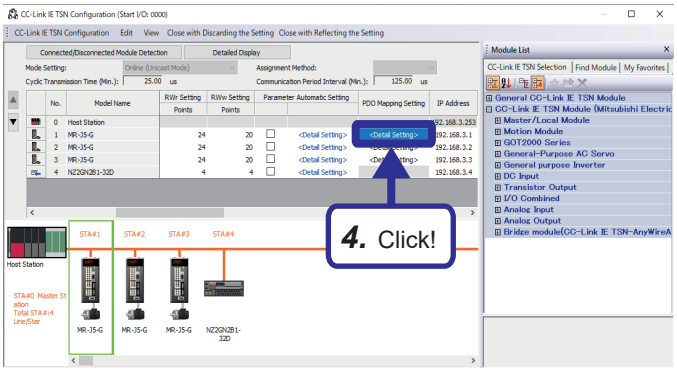
1. Click [CC-Link IE TSN Configuration] ⇨[Batch Setting of PDO Mapping] from the menu in the "CC-Link IE TSN Configuration" window.

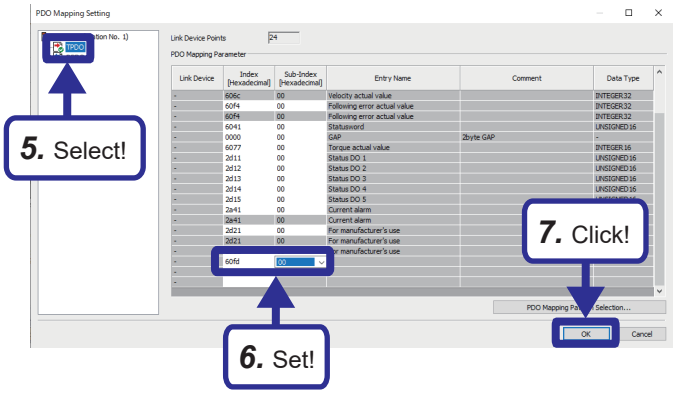


2. A confirmation dialog box appears. Click the [Yes] button.



3. Click the [OK] button.





4. Double-click "Detail Setting" of "PDO Mapping Setting" for MR-J5-G of station No. 1.

Point

Manually add the settings related to the on/off status of the input devices (sensors) connected to the servo amplifier because they are not reflected in the batch setting.

5. Select [TPDO].

6. Add the following settings in the bottom row.

[Setting details]

Index [Hexadecimal]: 60fd

Sub-Index [Hexadecimal]: 00

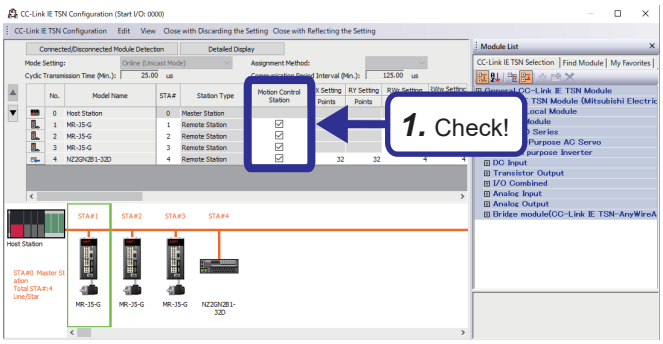
7. Click the [OK] button.

8. Set the servo amplifiers of station No. 2 and No. 3 in the same way.

Motion control station setting

Set the modules (such as servo amplifiers and I/O module) to be controlled by the Motion module.

Operating procedure



1. Check that the "Motion Control Station" check boxes of the servo amplifiers and CC-Link IE TSN remote I/O module are selected.

Servo parameter setting

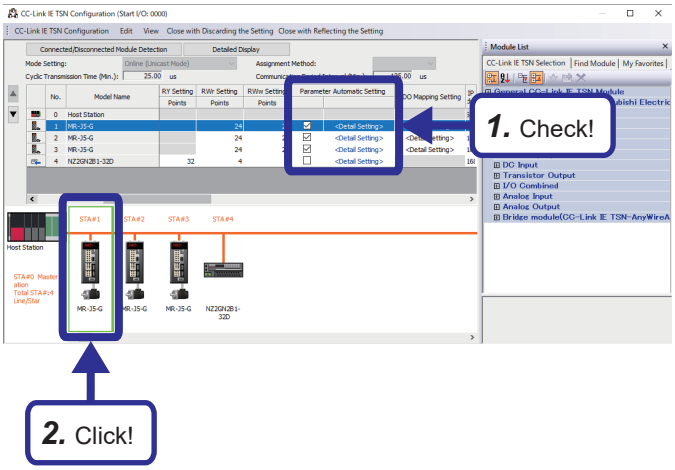
There are two ways to write parameters to the servo amplifier as follows.

Writing method	Advantage	Disadvantage
Writing to the PLC CPU from GX Works3	• Parameters can be collectively managed. • Parameters can be set without connecting devices.	• Initial communication takes time.
Writing to the servo amplifier from MR Configurator2	• The servo amplifier is quickly started because it is not necessary to transfer the parameters at power on.	• Parameters must be written to each servo amplifier.

This manual describes the setting procedure for writing data to the PLC CPU from GX Works3.

Setting procedure of the servo parameter

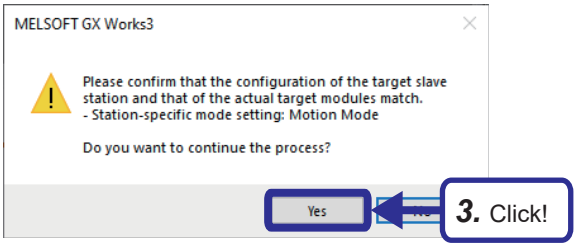
Operating procedure



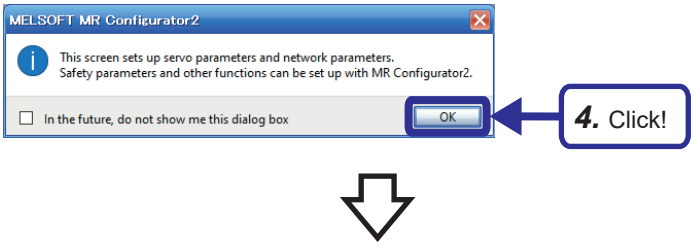
1. Select "Parameter Automatic Setting" of the servo amplifiers in the "CC-Link IE TSN Configuration" window.

When the check box is selected, parameters are written from the master station to the slave stations during initial communication.

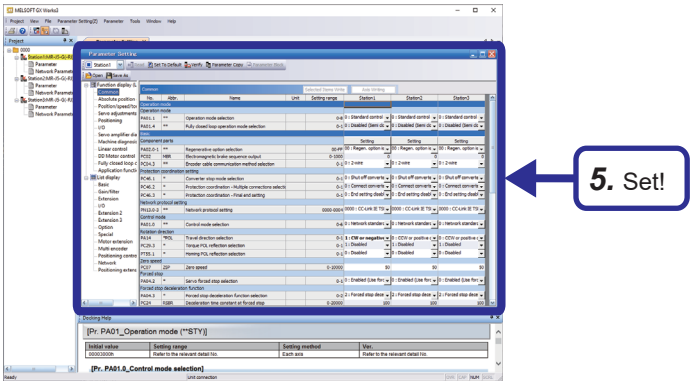
2. Double-click the illustration of the servo amplifier.



3. Click the [Yes] button.



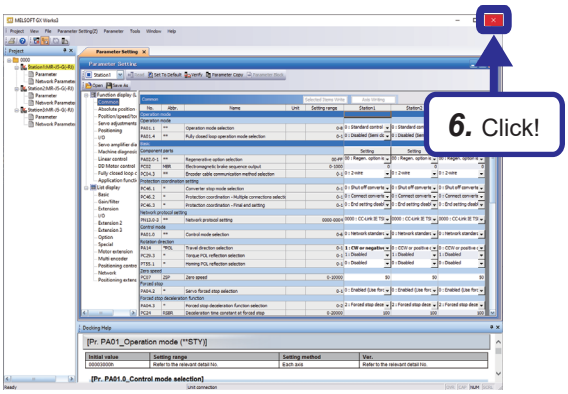
4. Click the [OK] button.



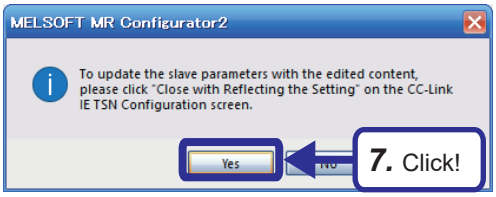
5. The "Parameter Setting" screen appears. Set the parameters of the servo amplifier.

Point

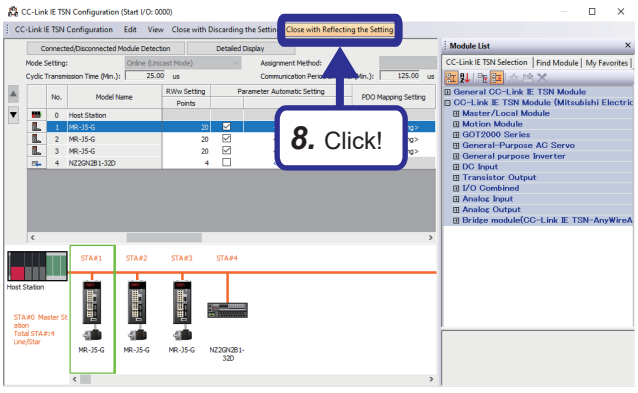
This exercise does not cover the parameter setting.



6. After setting the parameters, click the [×] button at the upper right of the screen.

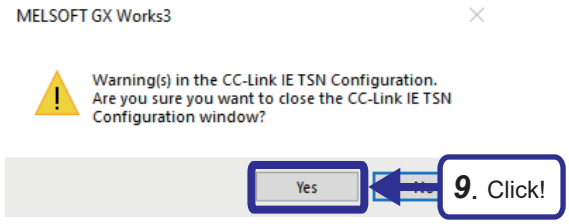


7. Click the [Yes] button to update the parameters.



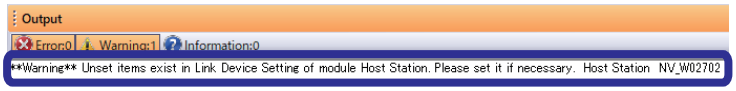
8. Select [Close with Reflecting the Setting] from the menu in the "CC-Link IE TSN Configuration" window.

9. Click the [Yes] button.



Point

The content of the displayed warning message is shown in the output window.
In the following case, there is no problem even if a warning appears.



■ List of servo parameters

The demonstration machine is set as follows.

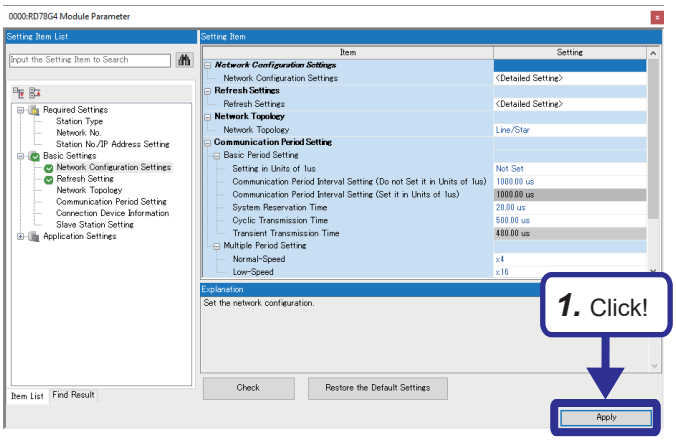
The following shows the changes from the initial values.

No.	Name	Setting value			Remarks
		Station 1	Station 2	Station 3	
PA08.0	Auto tuning mode	0:2 gain adjustment mode 1 (Interpolation mode)	—	—	Set this parameter to stabilize the motion of axis 1.
PA09	Auto tuning response	—	30	30	Set this parameter to prevent delay in the motion of axis 2 and axis 3 during synchronous control.
PA14	Movement direction selection	1:Clockwise or negative direction for forward rotation pulse input, and counterclockwise or positive direction for reverse rotation pulse input	—	—	Set this parameter to change the motor rotation from counterclockwise to clockwise depending on the mounting direction of the motor of axis 1.
PC19.0	[AL. 099 Stroke limit warning] selection	1:Disabled	1:Disabled	0:Enabled	Disable this parameter because axis 1 and axis 2 do not have limit sensors.
PD01.0-7	Input signal automatic ON selection 1	00000C00	00000C00	—	Set this parameter to match the specification of the external signal.
PD41.2	Limit switch enabled status selection	1:Enabled only in the homing mode	1:Enabled only in the homing mode	1:Enabled only in the homing mode	When the Mitsubishi Electric Motion module is used as a controller, set this parameter to enable the limit switch only in the homing mode.
PT06	Creep speed	—	—	50.00	Set this parameter to increase the creep speed during homing of axis 3.
PT07	Home shift amount	—	800000	—	Set this parameter to set the travel amount of home position shift during homing of axis 2.
PT29.0	Device input polarity 1	—	1:When ON, dogs are detected.	—	Set this parameter to match the specification of the dog sensor.
PT45	Homing method	-3:Data set type	-33:Dog type (Rear end detection, Z-phase reference)	-33:Dog type (Rear end detection, Z-phase reference)	For axis 1, set this parameter so that homing is complete in the motion system because axis 1 does not require the home position. For axis 2 and axis 3, set this parameter to perform homing in the negative direction using a dog sensor.

Applying the module parameters

Apply the parameter settings of the Motion module.

Operating procedure



1. Click the [Apply] button.

Point

The parameters are not applied unless the [Apply] button is clicked.

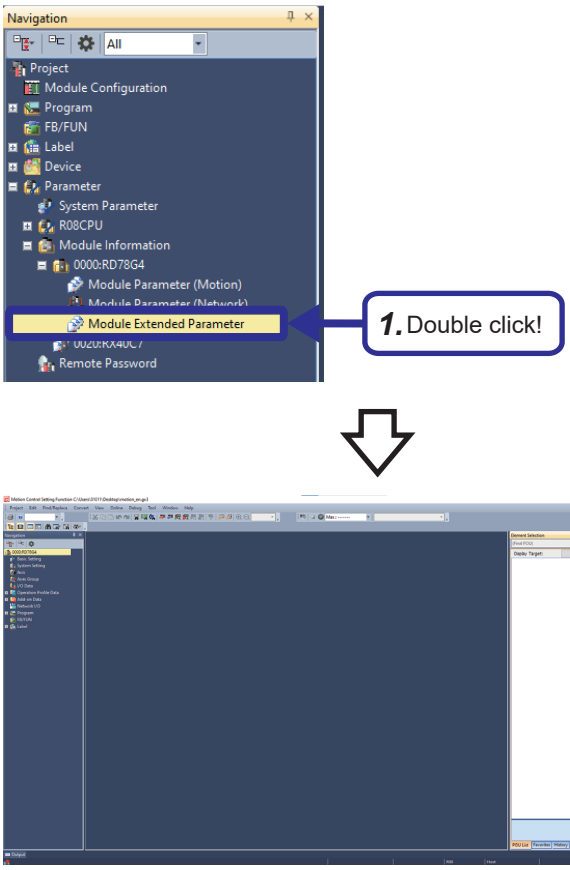
6.3 Motion Module Setting

Use the "Motion Control Setting Function" screen for parameter settings and programming of the Motion module.

Activating the motion control setting function

The following describes how to activate the motion control setting function.

Operating procedure



1. Select [Parameter] ⇨[Module Information] ⇨ [0000: RD78G4] in the "Navigation" window, and double-click [Module Extended Parameter].

2. The "Motion Control Setting Function" screen appears.

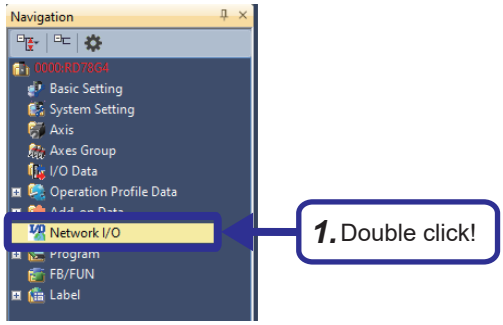
Point

If the "Motion Control Setting Function" screen does not appear after double-clicking "Module Extended Parameter", the motion control setting function may not have been installed. In this case, install the motion control setting function.

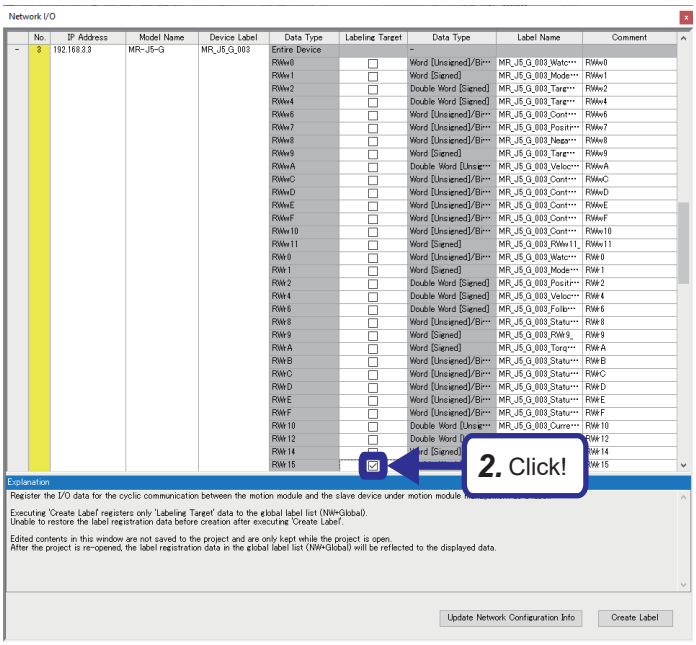
Network I/O setting

Register the labels of I/O data to be exchanged between the slave devices controlled by the Motion module and the Motion module through cyclic communication.

Operating procedure

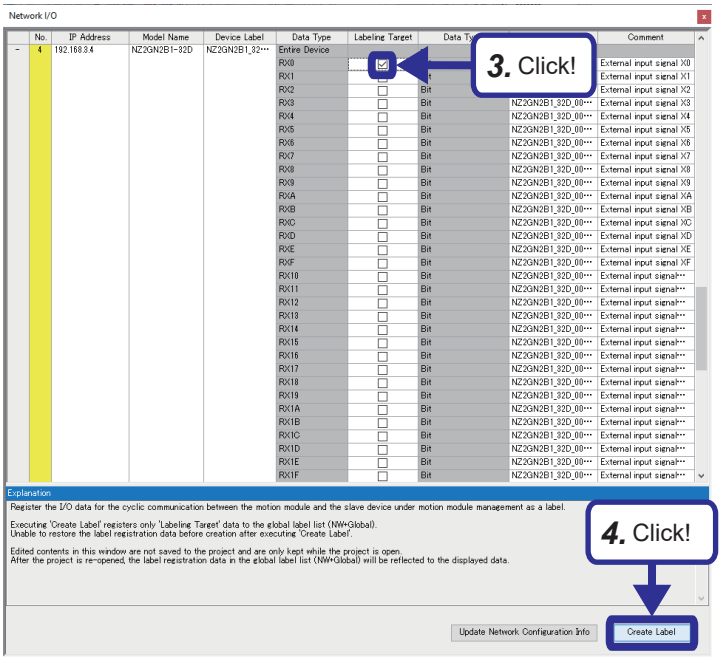


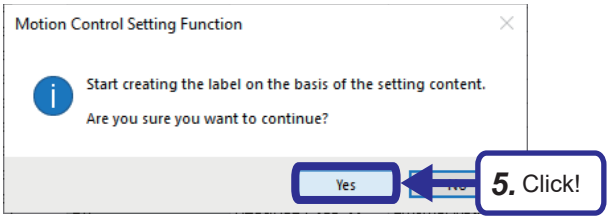
1. Double-click [Network I/O] in the "Navigation" window.

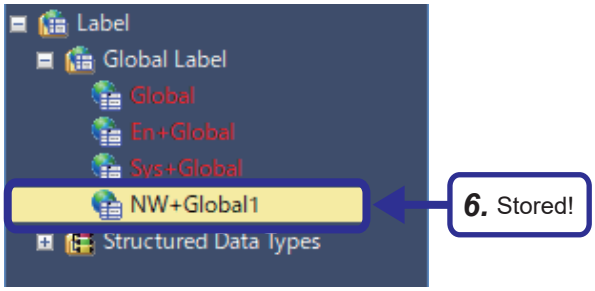


2. Select the "Labeling Target" check box of "RWr15" for the servo amplifier No. 3.

When the check box is selected, the I/O data of the external devices connected to the servo amplifier are labeled. The labeled I/O data are available on the Motion module.







3. Select the "Labeling Target" check box of "RX0" for the remote I/O module (DC input) No. 4.

When the check box is selected, the I/O data of the external devices connected to the remote I/O module (DC input) are labeled. The labeled I/O data are available on the Motion module.

4. Click the [Create Label] button.

5. Click the [Yes] button.

6. The network I/O label is created in [NW+Global1] under [Label] ⇨[Global Label] in the "Navigation" window.

Axis parameter setting

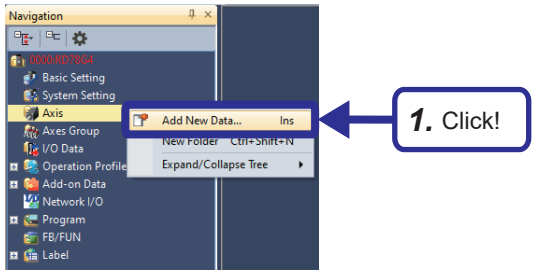
Creating an axis

The following describes the procedure for creating a new axis.

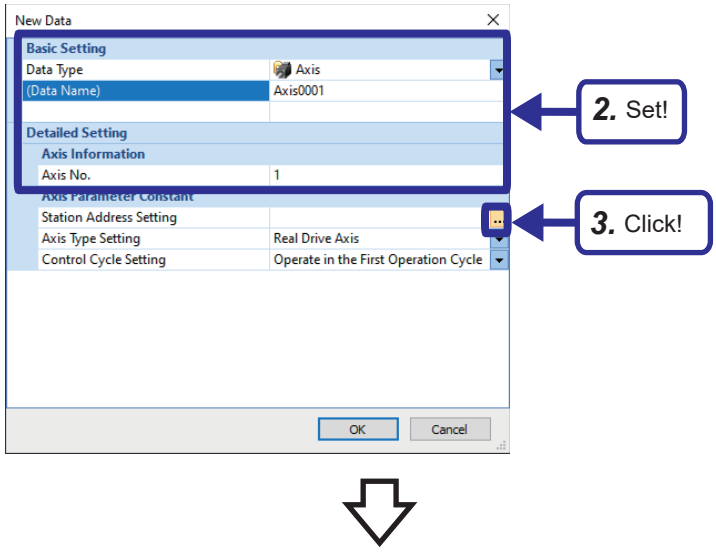
There are five types of axes as follows. In this exercise, register three real drive axes.

Axis type	Description
Real drive axis	An axis that outputs a command using the servo amplifier connected to CC-Link IE TSN
Real encoder axis	An axis that generates a current position from the output pulse of the synchronous encoder connected to the servo amplifier on CC-Link IE TSN
Virtual drive axis	An axis that can generate a command virtually
Virtual encoder axis	An axis that generates a current position virtually from a variable
Virtual linked axis	An axis that connects FBs of the single axis synchronous control virtually

Operating procedure



1. In the "Navigation" window, right-click [Axis] and select [Add New Data].



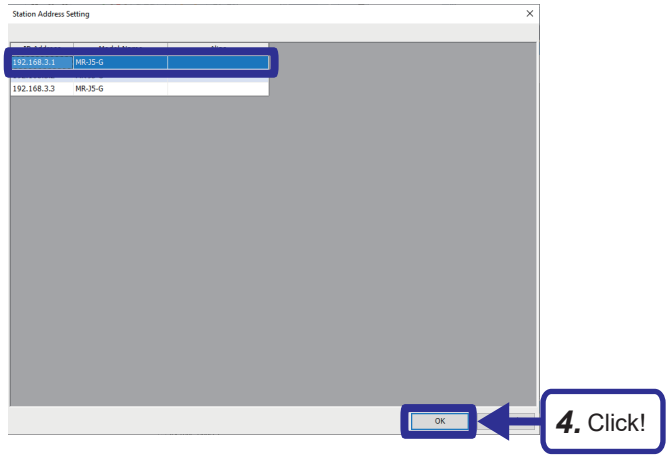
2. Set the data name and axis information as follows.

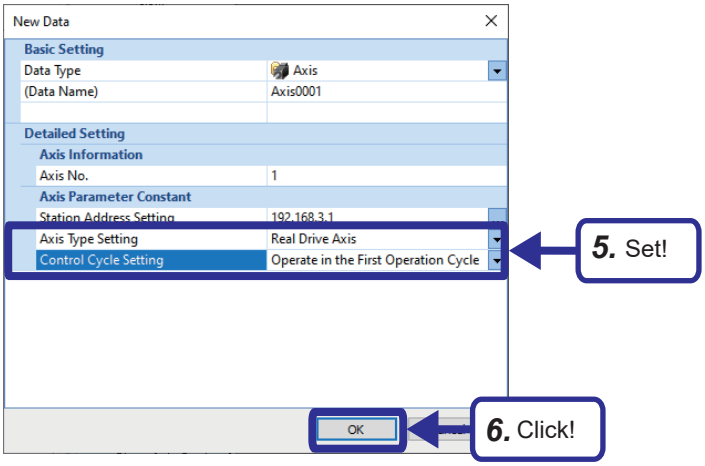
[Setting details]

Data Name: Axis0001 (Default)

Axis No.: 1 (Default)

3. Click the [...] button in Station Address Setting.





4. Select the servo amplifier with the IP address "192.168.3.1", and click the [OK] button.

Point

Set the station address to link the servo amplifier defined in the network configuration and the axis information of the axis parameter. In the station address setting, use the IP address set in the "CC-Link IE TSN Configuration" window.

5. Set the axis type and control cycle as follows.

[Setting details]

Axis Type Setting: Real Drive Axis (Default)

Control Cycle Setting: Operate in the First Operation Cycle (Default)

6. Click the [OK] button.

7. Set axis 2 and axis 3 in the same way as follows.

Setting item	Axis 2	Axis 3
Data Name	Axis0002 (Default)	Axis0003 (Default)
Axis No.	2 (Default)	3 (Default)
Station Address Setting	192.168.3.2	192.168.3.3
Axis Type Setting	Real Drive Axis (Default)	Real Drive Axis (Default)
Control Cycle Setting	Operate in the First Operation Cycle (Default)	Operate in the First Operation Cycle (Default)

Axis parameter setting

When the created axis data is double-clicked in the "Navigation" window, the Axis Parameter Setting screen is displayed. The axis parameters of the demonstration machine are set as follows. The following shows the changes from the initial values.

Item			Description	Setting value		
				Axis0001	Axis0002	Axis0003
Axis Parameter Constant	Station Address Setting		Set the IP address of the servo amplifier.	192.168.3.1	192.168.3.2	192.168.3.3
	Upper Limit Signal	Target	Assign the limit sensor installed at the upper limit of the moving range to an external signal.	—	—	[VAR]MR_J5_G_003_DigitalInputs.1*1
		Signal Detection Method	Specify the signal logic.	1:Detection at FALSE	1:Detection at FALSE	1:Detection at FALSE
	Lower Limit Signal	Target	Assign the limit sensor installed at the lower limit of the moving range to an external signal.	—	—	[VAR]MR_J5_G_003_DigitalInputs.0*1
		Signal Detection Method	Specify the signal logic.	1:Detection at FALSE	1:Detection at FALSE	1:Detection at FALSE
Axis Parameter	Driver Unit Conversion Numerator		Set the numerator to convert the command unit of the motion system to the command unit of the driver. For the setting method, refer to the following. ☞ Page 89 Driver unit conversion (Electronic gear)	1925160336	67108864	67108864
	Driver Unit Conversion Denominator		Set the denominator to convert the command unit of the motion system to the command unit of the driver. For the setting method, refer to the following. ☞ Page 89 Driver unit conversion (Electronic gear)	3157879	360	7999
	Homing Required or Not		Set whether homing is required or not.	0: Homing Not Required	—	—
	Start Permission at Homing Uncompleted		Set whether or not the axis can be started if homing is not complete (if the homing request is TRUE).	1:Enabled	—	—
Stop Signal	Position Command Unit		Set the position command unit to be used for motion control.	um	degree	um

*1 The global labels are used for the upper limit signal and lower limit signal.

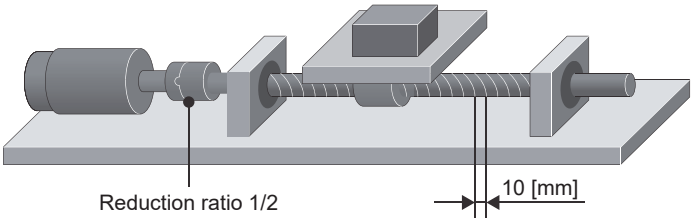
Driver unit conversion (Electronic gear)

The following shows the setting example of the driver unit conversion numerator/denominator.

■ Mechanism of the electronic gear

• Ball screw

67108864[pulse]



Item	Setting value
Servo motor encoder resolution	67108864[pulse]
Ball screw lead	10000[um]
Reduction ratio	1/2 (Load side [NL] / Motor side [NM]) When the motor rotates two revolutions, the load-side ball screw rotates one revolution.

$$\frac{\textit{Driver unit conversion numerator}}{\textit{Driver unit conversion denominator}} = \frac{\textit{Number of encoder pulses}}{\textit{Movement amount} \times \textit{Reduction ratio}} = \frac{67108864}{10000 \times 1/2} = \frac{67108864}{5000}$$

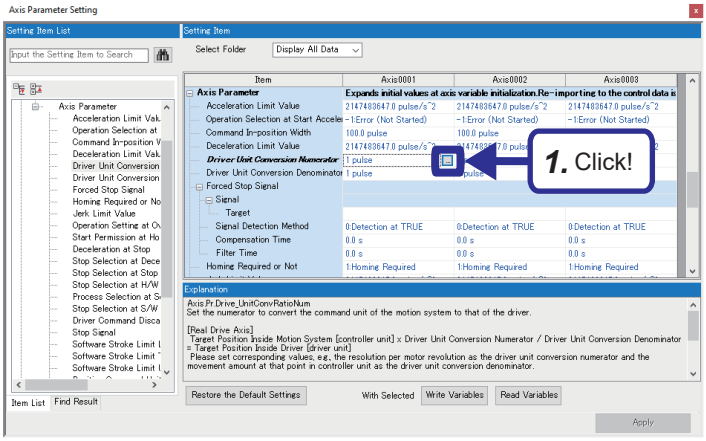
• Driver unit conversion numerator = Number of pulses per revolution 67,108,864

• Driver unit conversion denominator = Travel amount per revolution 5000

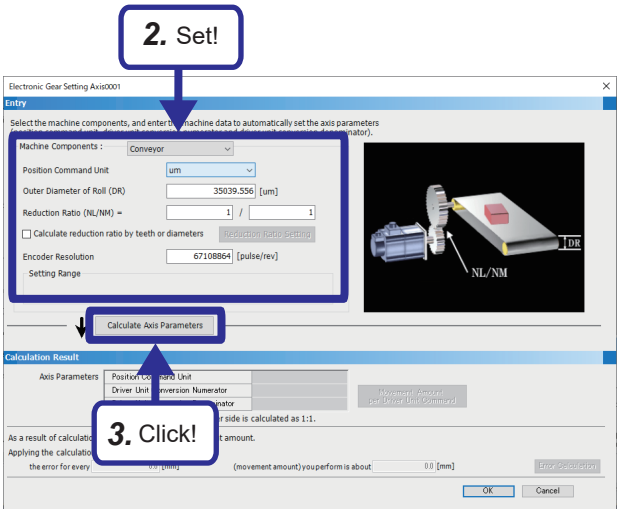
■ Electronic gear setting

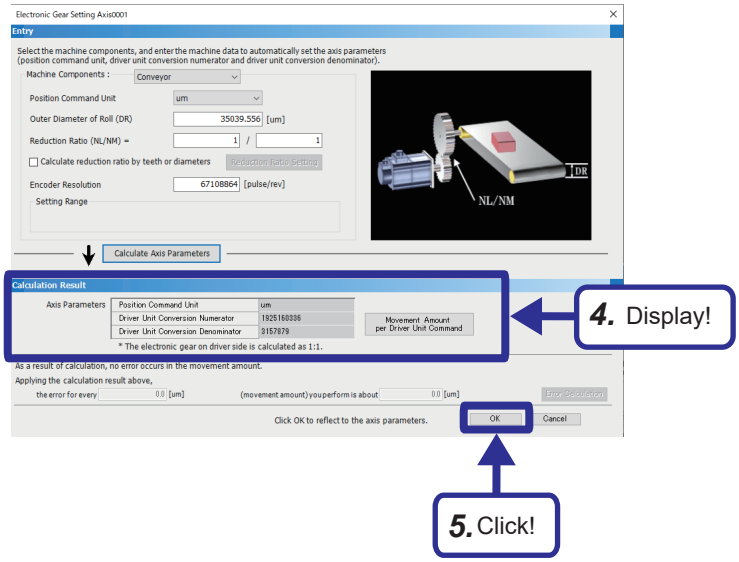
The following describes the setting procedure for the electronic gear.

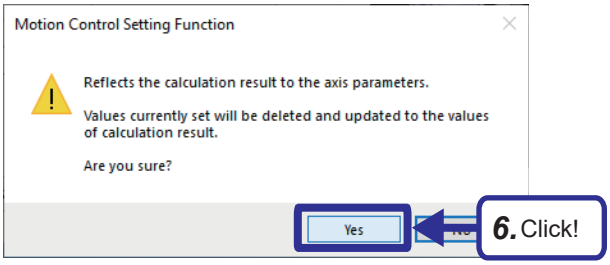
Operating procedure



1. Click the [...] button for Axis0001 under "Axis Parameter" ⇨"Driver Unit Conversion Numerator" or "Driver Unit Conversion Denominator" in the "Axis Parameter Setting" window.







2. The "Electronic Gear Setting" screen appears. Set the items as follows.

[Setting details]

Machine Components: Conveyor

Position Command Unit: um

Outer Diameter of Roll (DR): 35039.556

Reduction Ratio (NL/NM): 1/1

Calculate reduction ratio by teeth or diameters: Not select

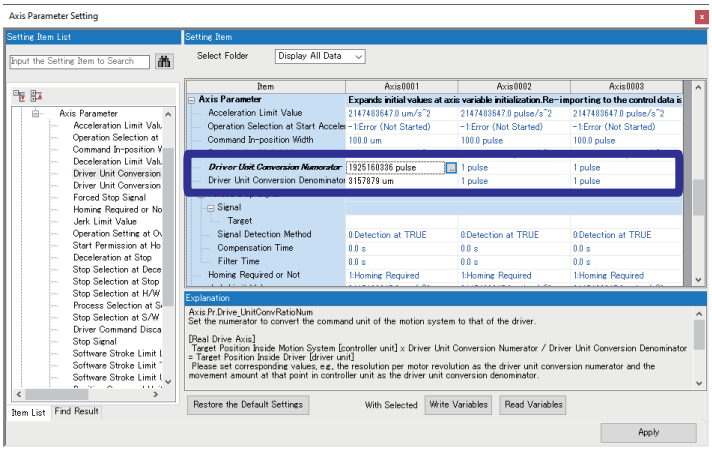
Encoder Resolution: 67108864

3. Click the [Calculate Axis Parameters] button.

4. The values of the numerator and denominatorconverted into the driver unit are displayed in Calculation Result.

5. Click the [OK] button.

6. Click the [Yes] button.



7. The converted values are displayed in "Driver Unit Conversion Numerator" and "Driver Unit Conversion Denominator" in the "Axis Parameter Setting" window.

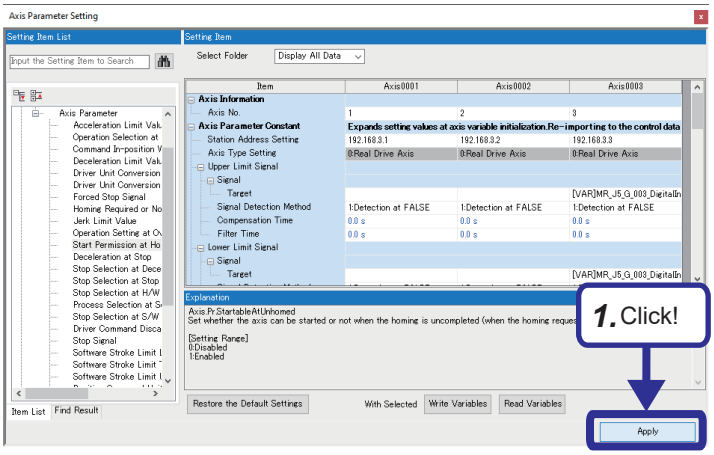
8. Set axis 2 and axis 3 in the same way as follows.

Item	Axis0002	Axis0003
Machine Components	Rotary Table	Ball Screw, Horizontal
Position Command Unit	degree	um
Ball screw lead	—	7999.0
Encoder Resolution	67108864	67108864

Applying the axis parameters

Apply the axis parameter settings.

Operating procedure



1. After setting the parameters for each axis, click the [Apply] button.

Point

Note that the parameters are not applied unless the [Apply] button is clicked.

Label setting

There are two types of labels: local labels and global labels.

Use local labels or global labels depending on the application.

Label	Description
Local label	• Available only in a single program
Global label	• Available in all programs • Available as a public label

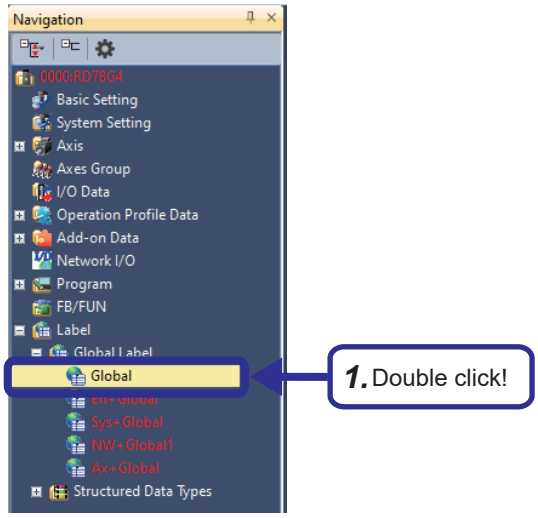
Creating a label

Labels must be created to make a motion control program and perform motion control.

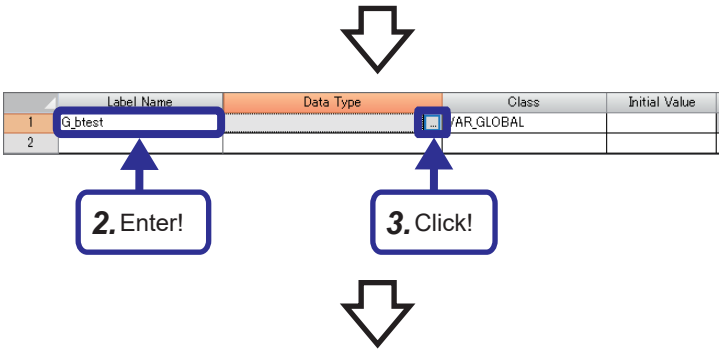
This exercise does not cover the creation of labels.

The following provides an example procedure for creating a new global label.

Operating procedure

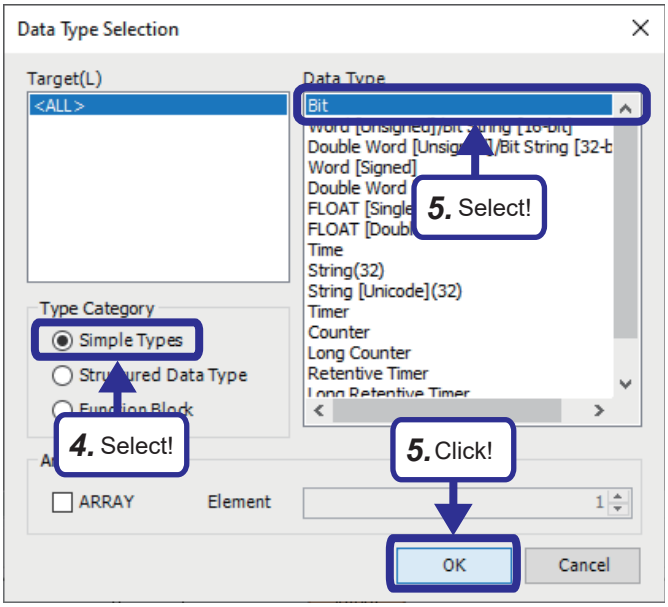


1. In the "Navigation" window of the motion control setting function, double-click [Global] under [Label] ⇨[Global Label].



2. The Global Label Setting screen appears. Enter a name in "Label Name".

3. Click the [...] button in "Data Type".



4. Select "Simple Types" in Type Category in the Data Type Selection window.

5. Select a data type in "Data Type" and click the [OK] button.

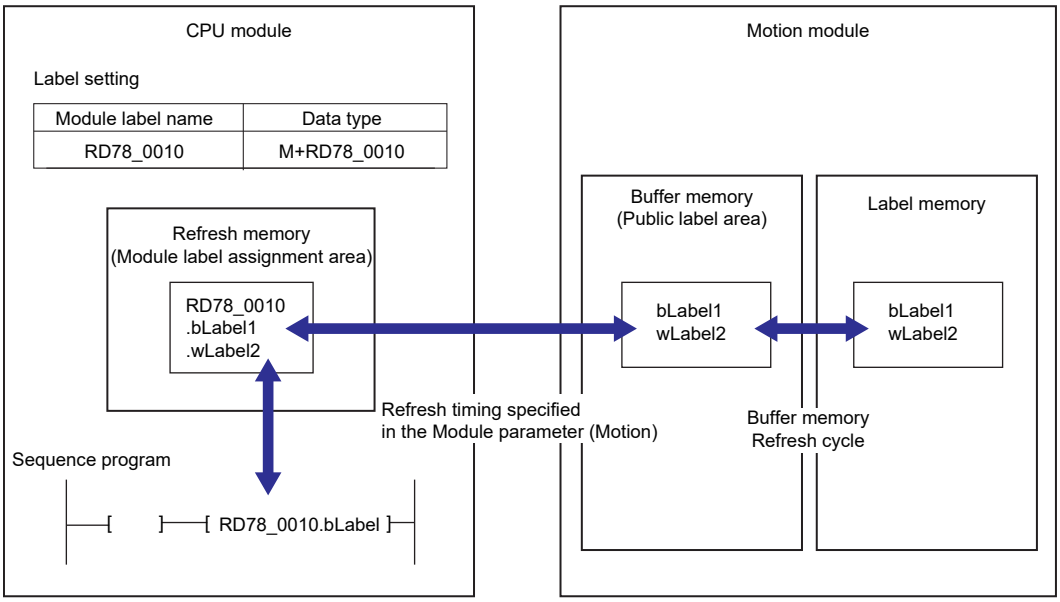
■ Global label type

The following table lists the types of global labels.

Data name	Description
Global	Stores global labels created by users. Enable the public labels.
En+Global	A group for definition of the ENUM enumerator. No operation is required because the system is automatically registered in this group.
Sys+Global	A structure that stores data related to the system. Enable the public labels.
Ax+Global	It is automatically registered when an axis is set in the motion control setting function. No action is required on the label editor.
Prg+Global	It is automatically registered when a program is created in the motion control setting function. No action is required on the label editor.
Gr+Global	It is automatically registered when an axis group is set in the motion control setting function. No action is required on the label editor.
Prf+Global	It is automatically registered when the profile data is registered in the motion control setting function. No action is required on the label editor.
NW+Global1	It is automatically registered when the labeling is registered in the network I/O setting in the motion control setting function. No action is required on the label editor.

Public label

By publishing global labels and structure members, they can be used as module labels by the CPU module.
In addition, data to be monitored such as position and speed are defined as variables (labels) with fixed names in the Motion module.



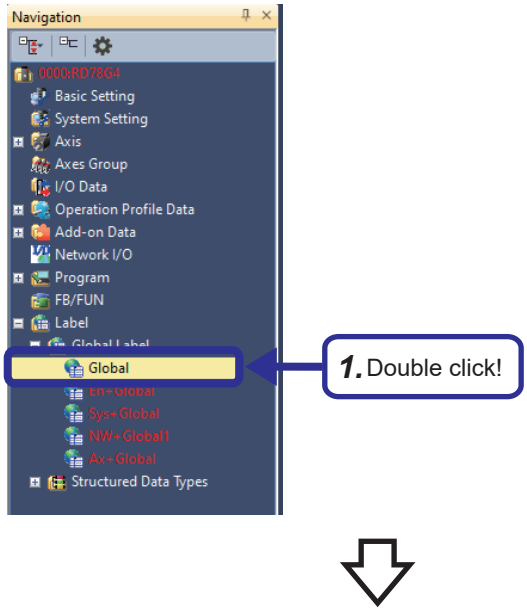
When public labels are used, the following memory areas are used to refresh data.

Location of memory	Name	Refresh timing
CPU module	Refresh memory (Module label assignment area)	Specified by the module parameter (motion).
Motion module	Buffer memory (Public label area)	Buffer memory refresh cycle

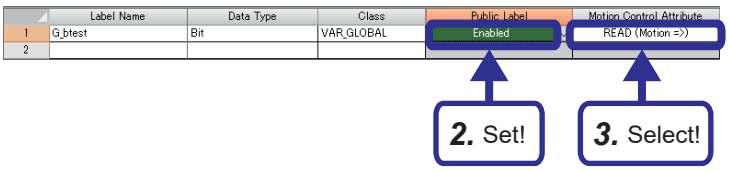
Registering a public label

The following describes how to set a global label as a public label.
This exercise does not cover the public label settings.

Operating procedure



1. In the "Navigation" window of the motion control setting function, double-click [Global] under [Label] ⇨[Global Label].



2. Select the label to be set as the public label and set "Public Label" to "Enabled".

3. Select "Motion Control Attribute" to determine the direction of label refresh.

- Writing from the CPU module to the Motion module: WRITE (=> Motion)
- Writing from the Motion module to the CPU module: READ (Motion =>)

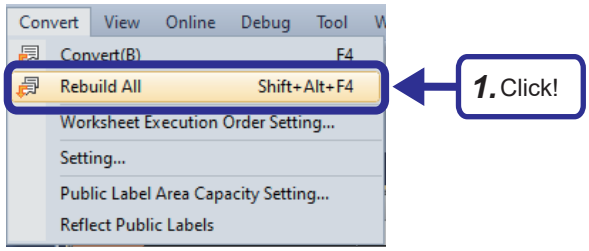
Point

The motion control attribute must be set to use user-created global labels as public labels.

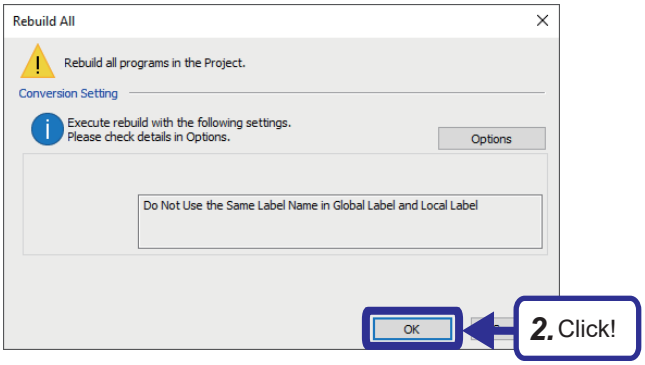
Applying public labels

Reflect the generated public label information to the project on the CPU module.
After applying the public labels, they are automatically registered as module labels to the CPU module.

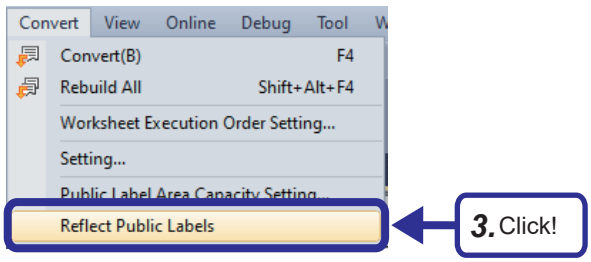
Operating procedure



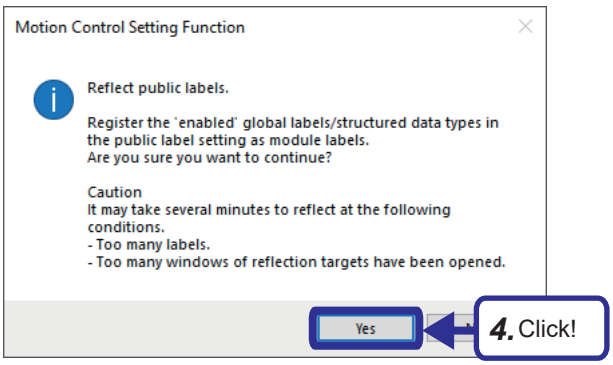
1. Click [Convert] ⇨[Rebuild All] from the menu of the motion control setting function.



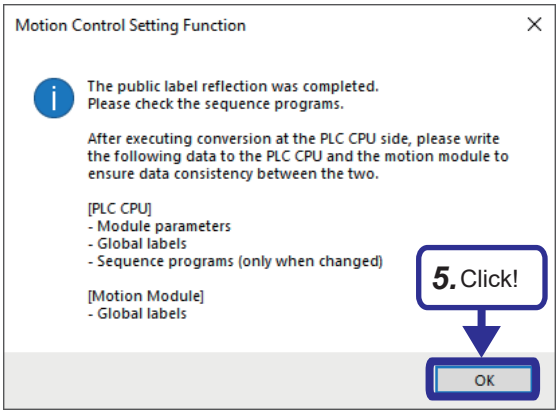
2. Click the [OK] button.



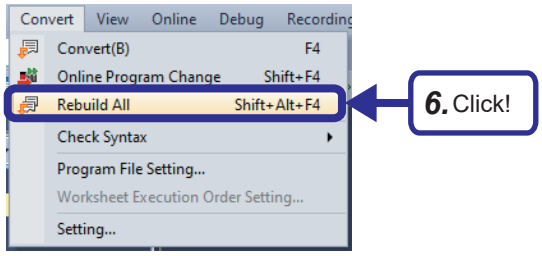
3. Click [Convert] ⇨[Reflect Public Labels] from the menu.



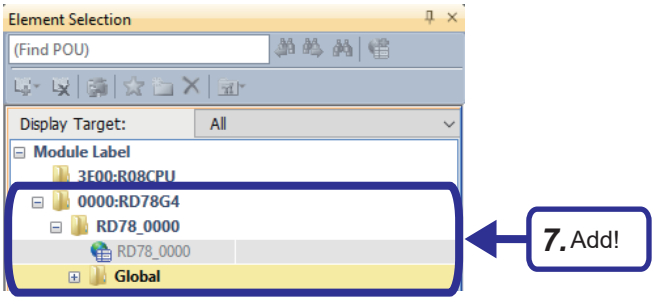
4. Click the [Yes] button.



5. Click the [OK] button.



6. Click [Convert] ⇨[Rebuild All] from the menu of GX Works3 to display the "GX Works3" screen and use the public labels on the CPU module.

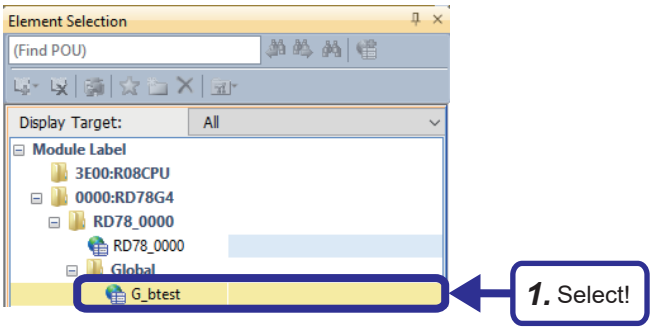


7. They are registered to Module Label in the Element Selection window.

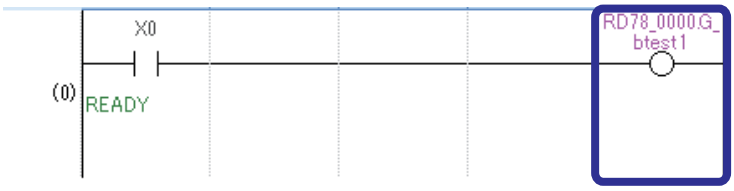
How to use public labels

The following describes how to use the public labels on the CPU module.

Operating procedure



1. Select a public label in [RD78_0000] under [Module Label] ⇨[0000:RD78G4] on the [Module] tab in the Element Selection window, and drag and drop it.



2. It can be used in the program on the PLC as shown on the left.

Point

When the public label of the Motion module is accessed by the PLC CPU, "module + start I/O No." is added at the beginning.

(Ex) When specifying the label of axis 1 (Axis0001)

RD78_0000.Axis0001.AxisRef

- Member
- Axis label name
- Module + Start I/O No.

7 EXERCISE 2 POSITIONING CONTROL

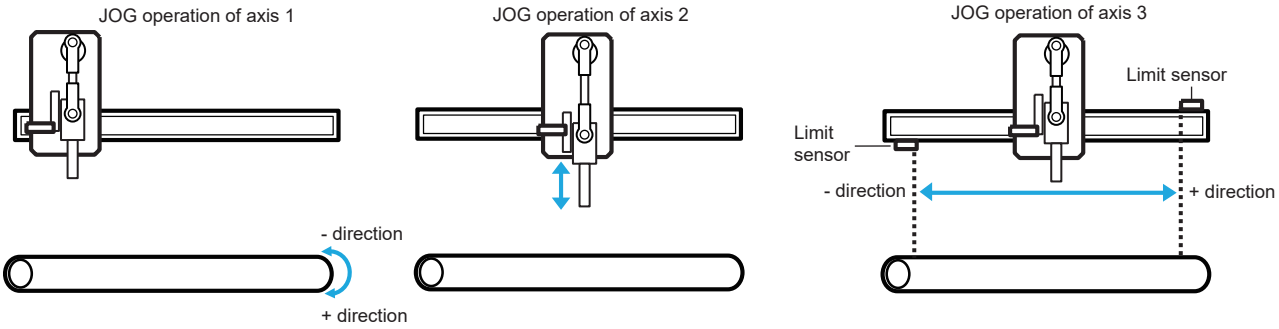
This chapter describes the single axis manual control (JOG operation), homing control, single axis positioning control, and single axis continuous positioning control.

7.1 Exercise

Create a motion program to perform the following operation.

Single axis manual control (JOG operation)

In JOG operation, while the JOG forward rotation command or JOG reverse rotation command is being input, the command is output from the servo system to the axis and the axis moves in the commanded direction. When an input signal is detected by either of the limit sensors installed at both ends, JOG operation stops and an error occurs.



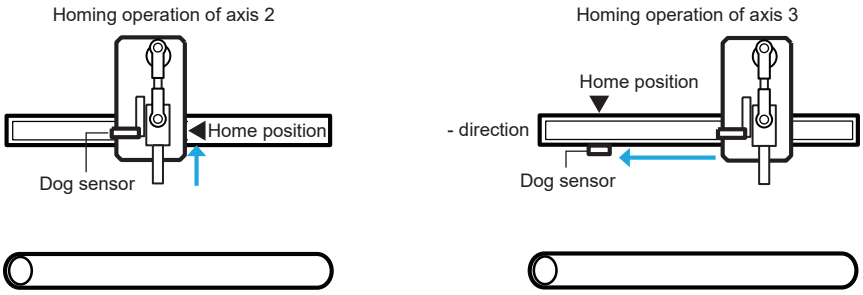
Homing control

In homing control, a machine home position is determined.

None of the address information stored in the motion system or driver is used at this time.

When homing starts, axis 3 moves in the negative direction. Homing ends when the home position is determined to have been reached based on the input signal from the dog sensor.

After homing, the mechanically determined position is regarded as the "home position", which is the start point of positioning control.



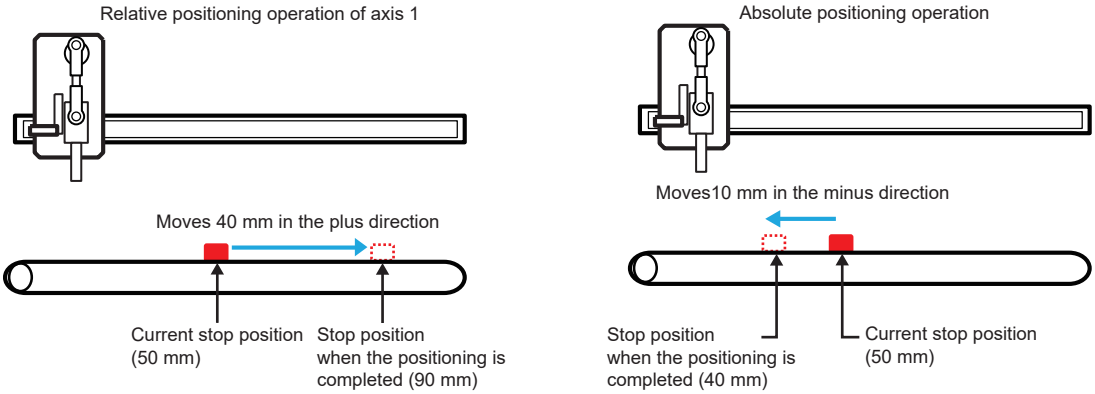
Single axis positioning control

Single axis positioning control executes positioning to the specified position by using address information.

There are two types of single axis positioning control: absolute positioning control and relative positioning control. Specify the travel distance for relative positioning control, and specify the target position for absolute positioning control.

If relative positioning is performed when axis 1 is stopped at 50 mm and the target position is set to 40 mm, the axis will be positioned at 90 mm.

If absolute positioning is performed with the same setting, the axis will be positioned at 40 mm.

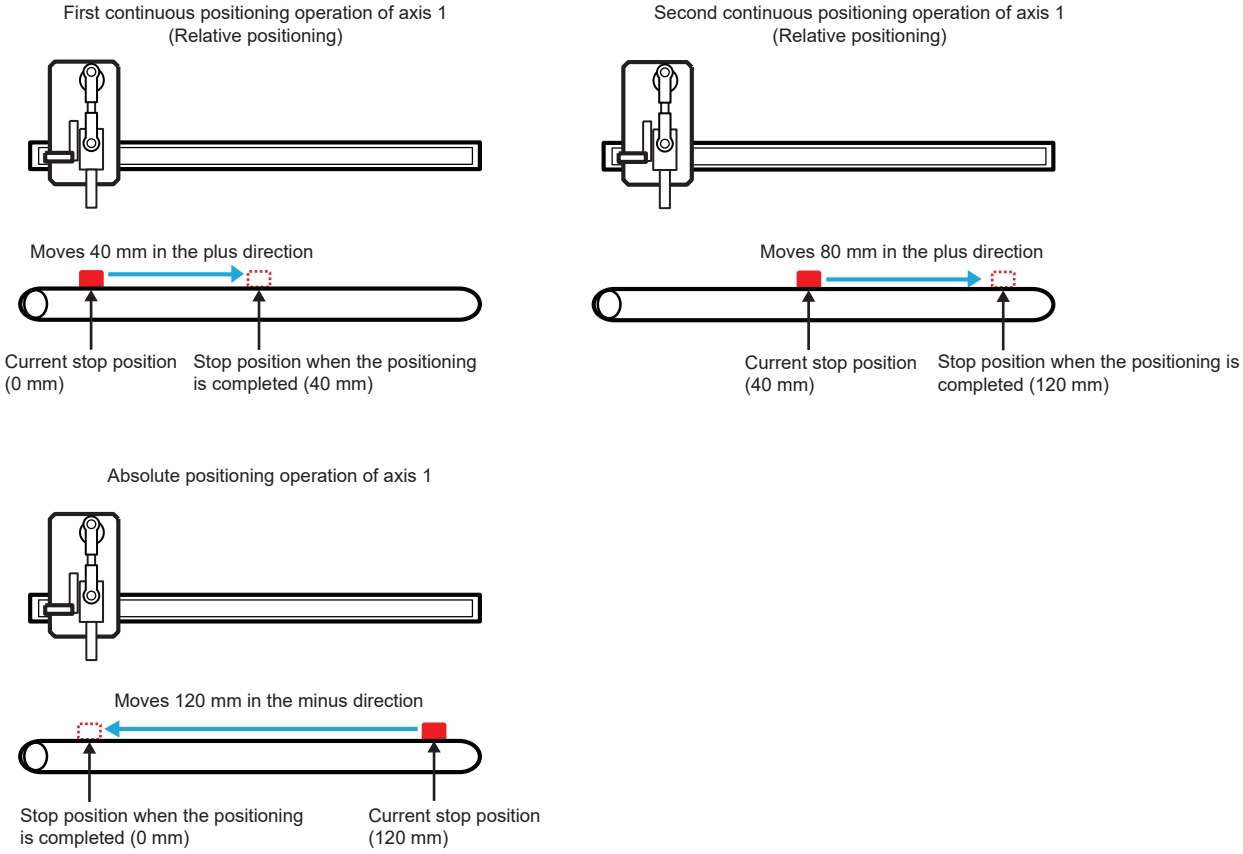


7

Single axis continuous positioning control

By setting the multiple start (buffer mode), multiple motion control FBs can be continuously executed without stopping. When axis 1 is stopped at the home position (0 mm) and the target position is 40 mm, single axis continuous positioning control is performed as follows.

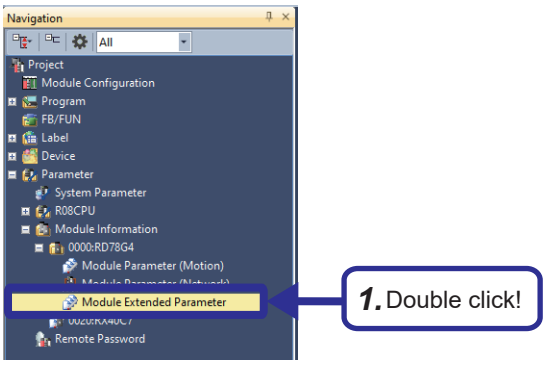
1. Relative positioning is performed to move the axis to the 40 mm position.
2. From the stop position (40 mm), relative positioning is continuously performed by multiple start (buffer mode) to move the axis by 80 mm, twice the movement amount of the target position setting. The red workpiece is stopped at the 120 mm position. This completes single axis continuous positioning control.
3. After the completion of single axis continuous positioning, the workpiece is positioned to the home position (0 mm) by absolute positioning.



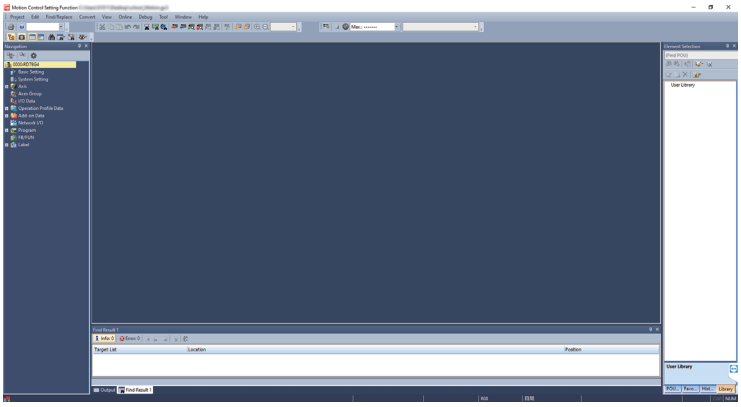
7.2 Opening a Project

"school_Motion.gx3" contains the configured parameters and GOT control programs prepared for this exercise.
Program the Motion module with the motion control setting function.

Operating procedure



1. Open school_Motion.gx3, select [Parameter] ⇨[Module Information] ⇨ [0000: RD78G4] from the "Navigation" window, and double-click [Module Extended Parameter].



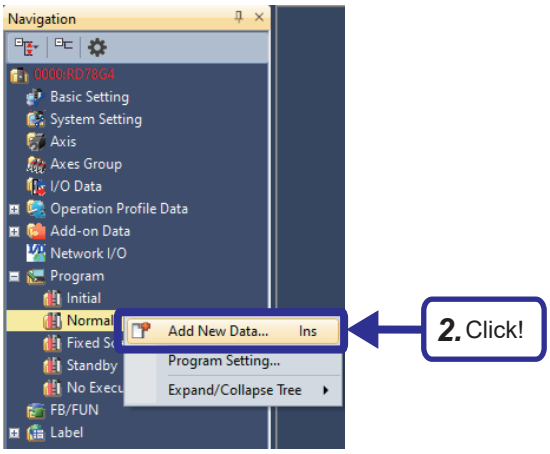
2. The motion control setting function is activated.

7.3 Creating a Program for the Motion Module

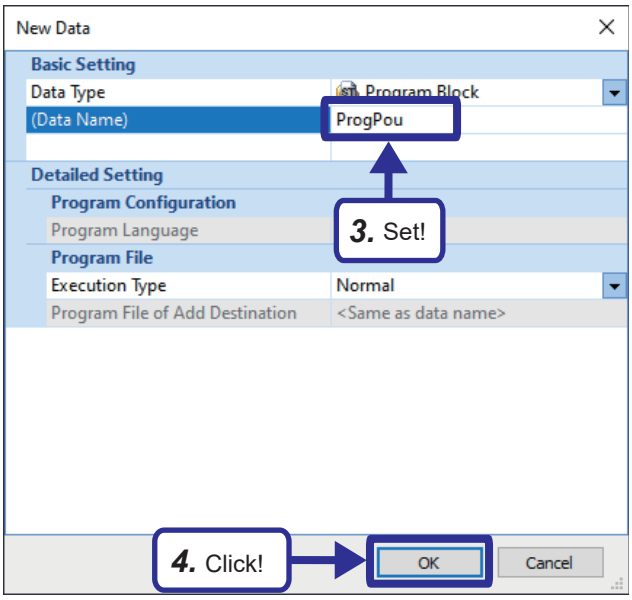
Creating a program block

In the project to be used in this exercise, some program blocks are already prepared.
To create a new project, follow the steps below.

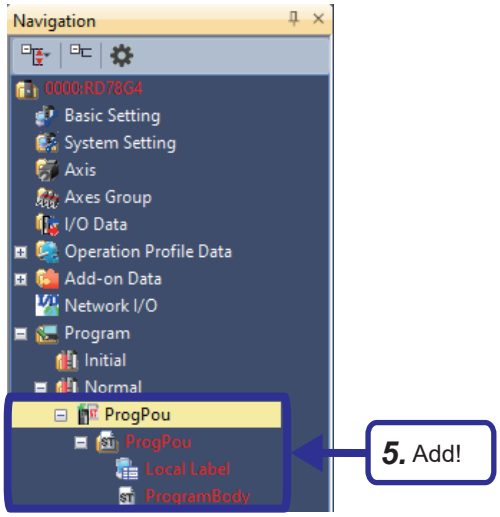
Operating procedure



1. In the "Navigation" window of the motion control setting function, right-click [Normal] under [Program].
2. Click [Add New Data].



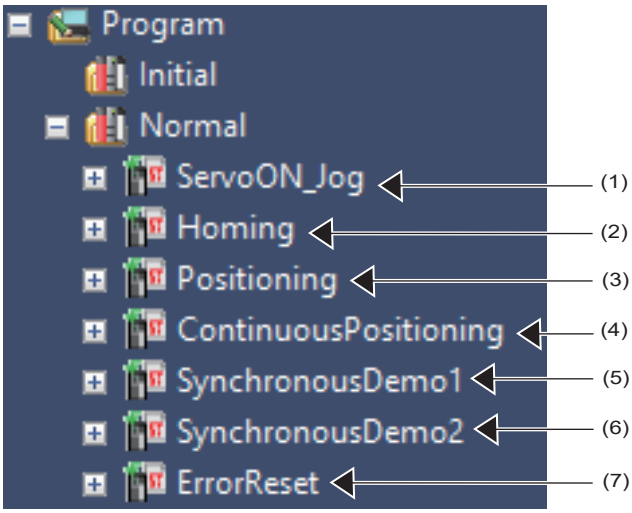
3. In the New Data window, set a data name.
4. Click the [OK] button.



5. The program block is added to the Navigation window.

Preset program blocks

The following table lists preset program blocks.



No.	Name	Description
(1)	ServoON_Jog	Create a program for servo ON and JOG operation.
(2)	Homing	Create a program for homing control.
(3)	Positioning	Create a program for positioning control.
(4)	ContinuousPositioning	Create a program for single axis continuous positioning control.
(5)	SynchronousDemo1	Create a program for 2-axis synchronous control.
(6)	SynchronousDemo2	Create a program for 3-axis synchronous control.
(7)	ErroReset	An error reset program is prepared.

7.4 Positioning Control Program

Create a positioning control program using the following motion control FBs.

Type	FB	Description
Administrative	MCv_AllPower	Performs servo ON for the real drive axes connected to the servo system.
Motion	MCv_Jog	Performs JOG operation at the command velocity.
	MC_Home	Performs homing of the specified axis.
	MC_MoveRelative	Sets the relative movement amount and executes positioning.
	MC_MoveAbsolute	Sets the absolute target position and executes positioning.

Servo ON

FB name: MCv_AllPower

This FB switches all axes to the operation possible state and performs servo ON for the real drive axes connected to the servo system.

The following shows the details of MCv_AllPower.

```
MCv_AllPower(
  Axis:= ?AXIS_REF? ,
  Enable:= ?BOOL? ,
  ServoON:= ?BOOL? ,
  Busy=> ?BOOL? ,
  Error=> ?BOOL? ,
  ErrorID=> ?WORD?
);
```

Name	Number of input area points (byte)	Number of output area points (byte)	Compilation method	FB operation
All Axes Operation Possible	10	4	Subroutine type	Real-time execution

Setting data

■ I/O variable

I/O variable	Name	Data type	Input import	Setting range	Default value	Description
Axis	Axis information	AXIS_REF	At start	—	Can be omitted	When MCv_AllPower (All Axes Operation Possible) is used in the Motion module, this variable can be omitted. The setting is ignored. When MCv_AllPower (All Axes Operation Possible) is used in the CPU module, this variable sets IO No. (StartIO). Axis No. (AxisNo) is ignored. ☞ Page 45 AxisName.AxisRef. (Axis information)

■ Input variables

Input variable	Name	Data type	Import	Setting range	Default value	Description
Enable	Enable	BOOL	Always	TRUE, FALSE	FALSE	When this variable is set to TRUE, axis control is enabled and the axis status switches to the operation possible state. When this variable is set to FALSE, axis control is disabled and operation possible state of the axis is cancelled.
ServoON	Servo ON request	BOOL	Always	TRUE, FALSE	FALSE	When this variable is set to TRUE, the servo ON of the axis is requested.

■ Output variables

Output variable	Name	Data type	Default value	Description
Busy	Executing	BOOL	FALSE	This variable becomes TRUE when MCv_AllPower (All Axes Operation Possible) is executed.
Error	Error	BOOL	FALSE	This variable becomes TRUE when an error occurs.
ErrorID	Error code	WORD(UINT)	0	When an error occurs, this variable returns the error code. For details of error codes, refer to the following. 📖MELSEC iQ-R Motion Module User's Manual (Application)

Processing details

- This FB initializes the information of all axes and switches the axis status to the operation possible state.
- When the Enable (Enable) and Servo ON request (ServoON) inputs are set to TRUE, all axes are switched to the operation possible state.
- When the processing is started, Executing (Busy) becomes TRUE.
- When using this FB for the Motion module, ignore the setting of Axis information (Axis). When using this FB for the CPU module, set I/O No. (StartIO) of Axis information (Axis). For specifying I/O No., refer to the following.

📖MELSEC iQ-R Programming Manual (Motion Control Function Blocks)

- When an error occurs in MCv_AllPower (All Axes Operation Possible), Error (Error) becomes TRUE and the error code is stored in Error code (ErrorID). For details of error codes, refer to the following.

📖MELSEC iQ-R Motion Module User's Manual (Application)

- The servo ON/OFF status and the driver status of all real axes can be switched as follows by inputting Enable (Enable) and Servo ON request (ServoON).

Input variable		Servo ON/OFF status	Driver status (AxisName.Md.Driver_State)
Enable (Enable)	Servo ON request (ServoON)		
TRUE	TRUE	Servo ON	6:Operation Enable
	FALSE	Servo OFF	5:Switched On
FALSE	TRUE	Servo OFF	3:Switch On Disabled
	FALSE	Servo OFF	3:Switch On Disabled

- If the real axis is rotated by external force during the servo OFF state, the follow up processing is performed.
- The servo ON/OFF control can be operated regardless of the control mode. The control mode during the servo OFF state depends on the specification of the driver.
- Since MCv_AllPower (All Axes Operation Possible) is sent to the driver while a drive unit error is occurring, there is no need to turn Enable (Enable) and Servo ON request (ServoON) from FALSE to TRUE again.

Point

To execute servo OFF individually when using MCv_AllPower (All Axes Operation Possible), use MC_Power (Operation Possible) together.
When MCv_AllPower (All Axes Operation Possible) and MC_Power (Operation Possible) are used together, the MC_Power (Operation Possible) command is given priority.

Creating a motion control FB

The following describes the procedure for creating the program shown below.

Create the program in the program block "ServoON_Jog".

```
//------Servo ON/JOG operation------
//Switch to the axis operation enabled state
MCv_AllPower_1(
    Enable:= TRUE ,
    ServoON:= G_bSVONCMD ,
    Busy=> bPowerBussy
);
```

■ Labels used

The following table lists the global and local labels used in this program.

Category	Label name	Data type	Class	Public label	Description
Global label	G_bSVONCMD	Bit	VAR_GLOBAL	Enable	Servo ON request This bit is turned on/off by the program for the PLC CPU. Tapping "Servo ON M60" on the GOT turns on the servo ON request.
Local label	bPowerBussy	Bit	VAR	—	FB is running.

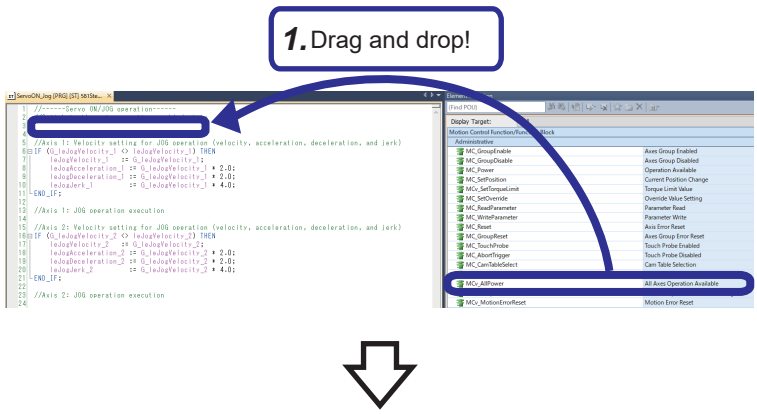
■ How to input a motion control FB

The following describes how to input a motion control FB.

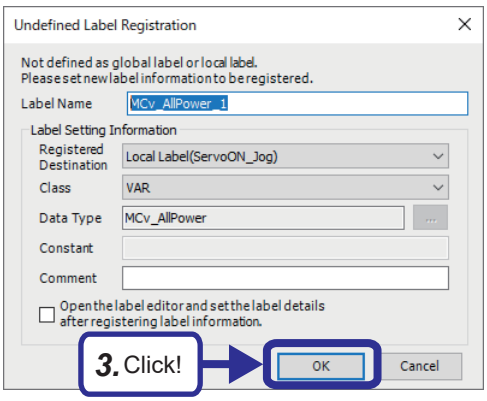
For categories of motion control FBs, refer to the following.

☞ Page 34 Type of motion control FB

Operating procedure



1. From [Administrative] under [Motion Control Function/Function Block] in "POU List" in the Element Selection window, drag and drop [MCv_AllPower].



2. The Undefined Label Registration window appears. Enter the FB label name and registered destination, and if necessary, enter the comment.

Leave this setting as default.

3. Click the [OK] button.

Point

Line breaks and indents can be inserted in the FB as desired.

■ Inputting I/O signals

Change some strings such as "?AXIS_REF?" and "?BOOL?" to the input values and labels.

Operating procedure

```
1  MCv_AllPower_1(
2      Axis:= ?AXIS_REF? ,
3      Enable:= ?BOOL? ,
4      ServoON:= ?BOOL? ,
5      Busy=> ?BOOL? ,
6      Error=> ?BOOL? ,
7      ErrorID=> ?WORD?
8      );
```

1. Delete!

1. Delete "?BOOL?" on the right of "Enable:=".

```
1  MCv_AllPower_1(
2      Axis:= ?AXIS_REF? ,
3      Enable:= TRUE,
4      ServoON:= ?BOOL? ,
5      Busy=> ?BOOL? ,
6      Error=> ?BOOL? ,
7      ErrorID=> ?WORD?
8      );
```

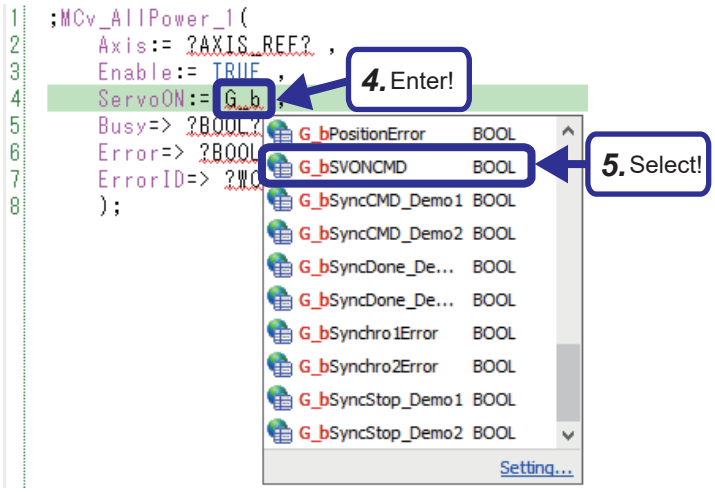
2. Enter!

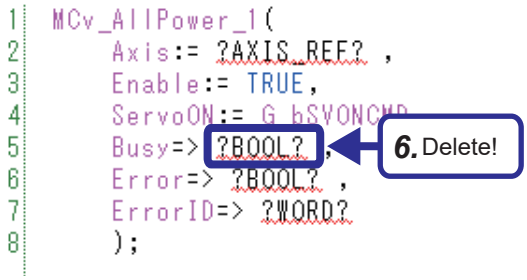
2. Enter "TRUE".

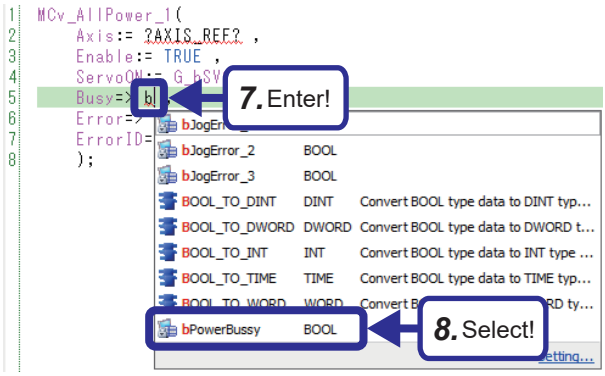
```
1  MCv_AllPower_1(
2      Axis:= ?AXIS_REF? ,
3      Enable:= TRUE,
4      ServoON:= ?BOOL? ,
5      Busy=> ?BOOL? ,
6      Error=> ?BOOL? ,
7      ErrorID=> ?WORD?
8      );
```

3. Delete!

3. Delete "?BOOL?" on the right of "ServoON:=".







4. Enter "G_b".

5. Select "G_bSVONCMD" from the list of registered label options.

Point

Tapping "Servo ON M60" on the GOT turns on G_bSVONCMD as well.

6. Delete "?BOOL?" on the right of "Busy=>".

7. Enter "b".

8. Select "bPowerBussy" from the list of registered label options.

■ Omitting I/O signals

Input signals for FBs not changed from the default values or not used can be omitted.

Operating procedure

```
1  MCv_AllPower_1(
2      Axis:= ?AXIS_REF? ,
3      Enable:= TRUE,
4      ServoON:= G_bSVONCMD ,
5      Busy=> bPowerBussy ,
6      Error=> ?BOOL? ,
7      ErrorID=> ?WORD?
8      );
```

1. Delete!

1. Delete Axis, Error, and ErrorID.

```
1  MCv_AllPower_1(
2      Enable:= TRUE,
3      ServoON:= G_bSVONCMD ,
4      Busy=> bPowerBussy ,
5      );
```

2. Delete!

2. Delete "," at the end of the FB as it is unnecessary.

```
1  //------Servo ON/JOG operation------
2  //Switch to the axis operation enabled state
3  MCv_AllPower_1(
4      Enable:= TRUE ,
5      ServoON:= G_bSVONCMD ,
6      Busy=> bPowerBussy
7  );
```

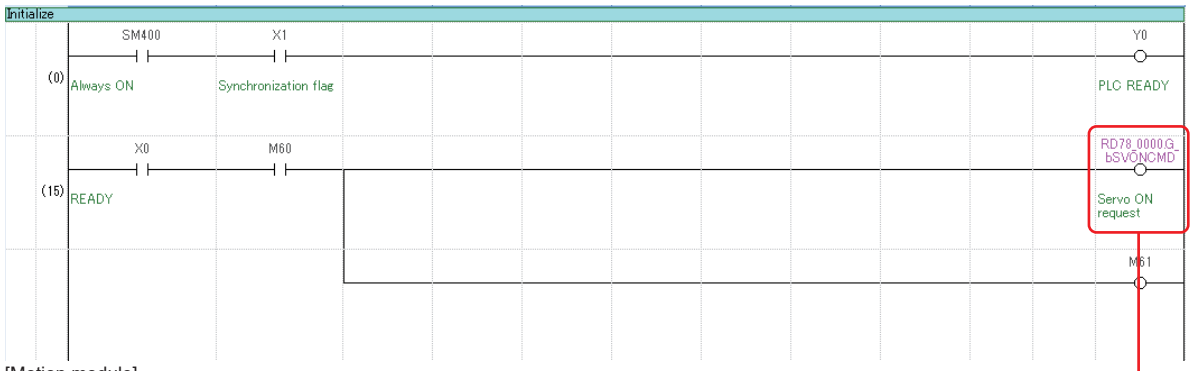
3. Add comments and indents as necessary.

■ PLC READY

Set the PLC READY (Y0) ON and all axes servo ON/OFF. When PLC READY [Y0] turns on, READY [X0] is turned on. Turning on READY [X0] and servo ON [M60] is used as the all axes servo ON signal.

The global label in the Motion module is used as the module label of the CPU module to send the start signal to the Motion module.

[PLC CPU]



[Motion module]

```
1  //------Servo ON/JOG operation------
2  //Switch to the axis operation enabled state
3  MCv_AllPower_1(
4      Enable:= TRUE ,
5      ServoON:= G_bSVONCMD ,
6      Busy=> bPowerBussy
7  );
```

The servo ON start signal from the PLC CPU triggers turning on of G_bSVONCMD and executes all axes ON.

Writing to the programmable controller

Write the sequence program and motion control program to the PLC CPU and Motion module.

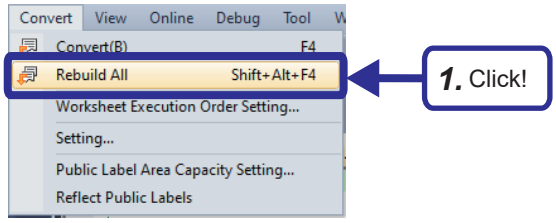
■ Converting programs

After creating a program, convert all data in the program.

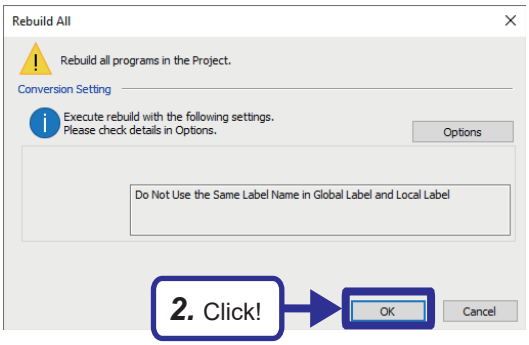
Convert the sequence program in GX Works3, and convert the motion control program in the Motion Control Setting Function.

This exercise uses the prepared sequence program and does not require the conversion of the program for the programmable controller.

Operating procedure



1. Click [Convert] ⇨ [Rebuild All] from the menu of the motion control setting function.



2. Click the [OK] button.

Point

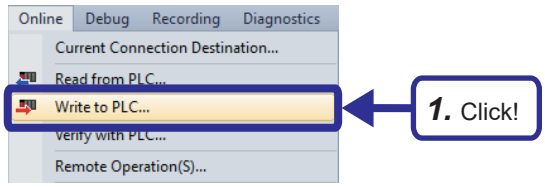
When a public label is used, click [Convert] ⇨ [Reflect Public Labels] from the menu of the motion control setting function to apply public labels to the PLC CPU. Apply public labels before creating a program of the PLC CPU.

This exercise uses the pre-registered labels and does not require public labels to be applied.

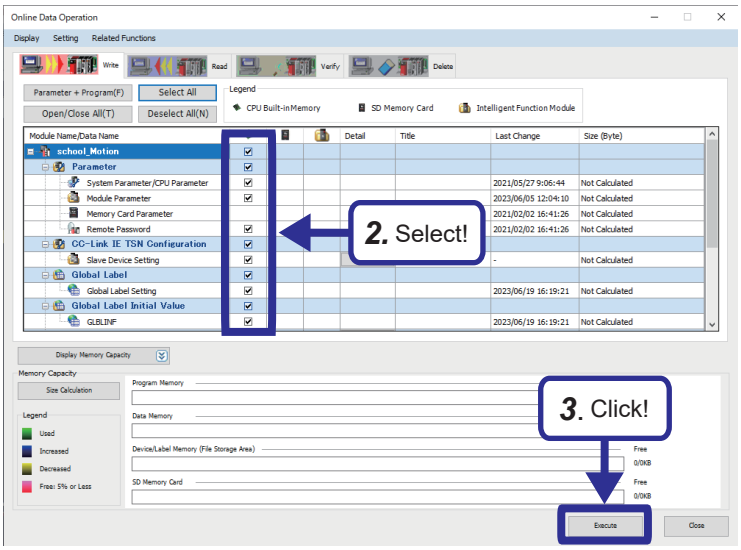
■ Writing

Write the sequence program to the programmable controller using MELSOFT GX Works3, and write the motion control program to the Motion module using the motion control setting function.

Operating procedure

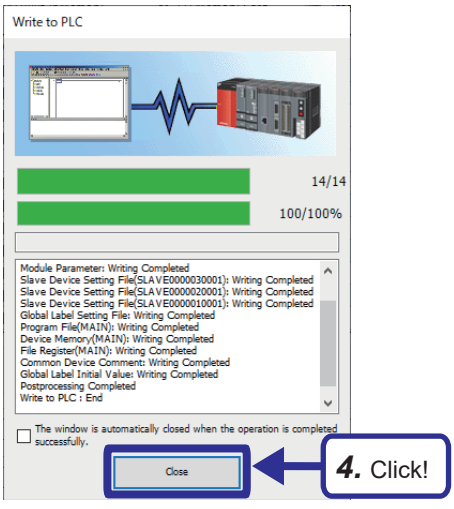


1. Click [Online] ⇨ [Write to PLC] from the menu of MELSOFT GX Works3.

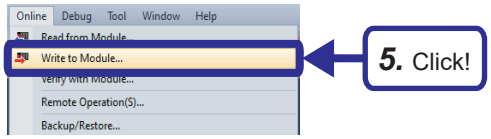


2. The Online Data Operation window appears. Select the items to be written.

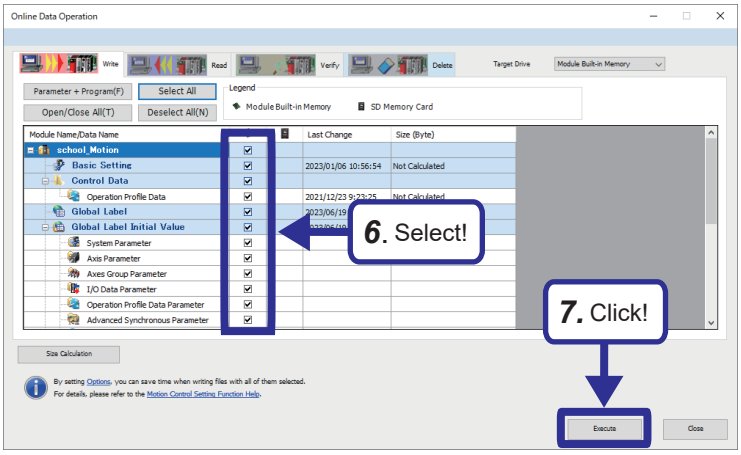
3. Click the [Execute] button.



4. The Write to PLC dialog box appears. Once writing is completed, the message "Completed" appears. Click [Close].

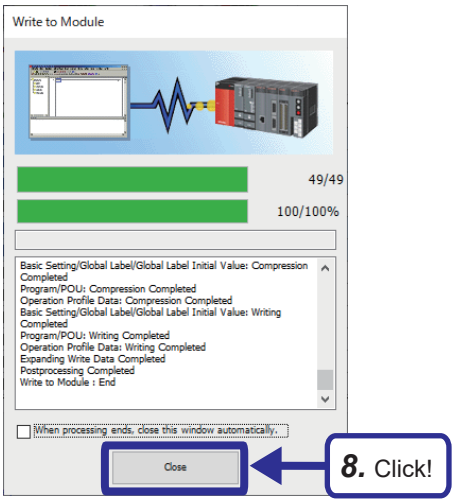


5. Click [Online] ⇨ [Write to Module] from the menu of the motion control setting function.



6. The Online Data Operation window appears. Select the items to be written.

7. Click the [Execute] button.



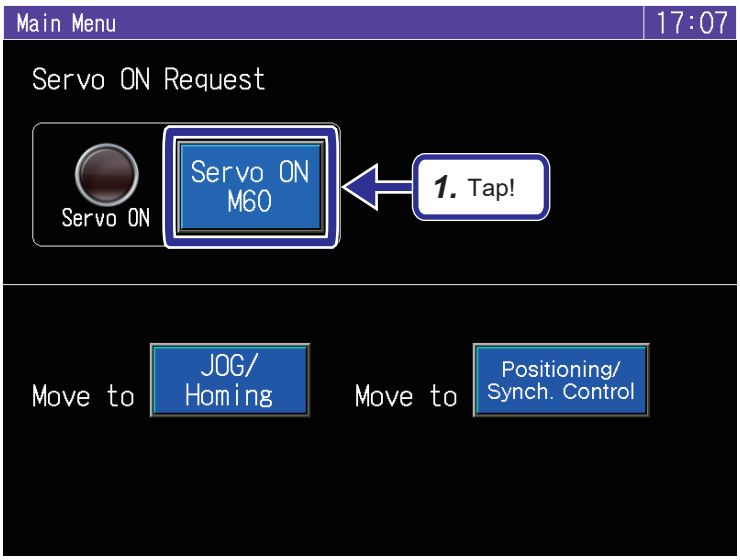
8. The Write to Module dialog box appears. Once writing is completed, the message "Completed" appears. Click [Close].

9. After resetting the CPU module, set the RUN/STOP/RESET switch to "RUN".

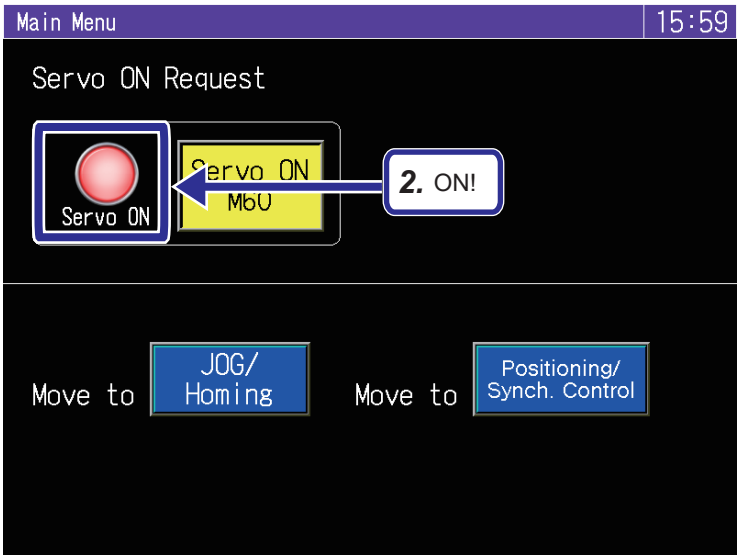
Operation check

After writing the programs, check the operation of servo ON by operating the GOT screen of the demonstration machine.

Operating procedure



1. Tap the [Servo ON M60] button on the main menu screen.



2. The servo ON lamp turns on, and all axes of the servo become the operation possible state.

Single axis manual control (JOG operation)

FB name: MCv_Jog

This FB performs JOG operation at the command velocity.

The following shows the details of MCv_Jog.

```
MCv_Jog(
  Axis:= ?AXIS_REF? ,
  JogForward:= ?BOOL? ,
  JogBackward:= ?BOOL? ,
  Velocity:= ?LREAL? ,
  Acceleration:= ?LREAL? ,
  Deceleration:= ?LREAL? ,
  Jerk:= ?LREAL? ,
  Options:= ?DWORD? ,
  Done=> ?BOOL? ,
  Busy=> ?BOOL? ,
  Active=> ?BOOL? ,
  CommandAborted=> ?BOOL? ,
  Error=> ?BOOL? ,
  ErrorID=> ?WORD?
);
```

Name	Number of input area points (byte)	Number of output area points (byte)	Compilation method	FB operation
JOG	52	8	Subroutine type	Real-time execution

Setting data

■ I/O variable

I/O variable	Name	Data type	Input import	Setting range	Default value	Description
Axis	Axis information	AXIS_REF	At start	—	Mandatory	This variable sets the axis. For the variables used (AxisName.AxisRef.), refer to the following. ☞ Page 45 AxisName.AxisRef. (Axis information)

■ Input variables

Input variable	Name	Data type	Import	Setting range	Default value	Description
JogForward	Forward rotation JOG command	BOOL	Always	TRUE, FALSE	FALSE	When this variable is TRUE, MCv_Jog (JOG) is executed in the positive direction.
JogBackward	Reverse rotation JOG command	BOOL	Always	TRUE, FALSE	FALSE	When this variable is TRUE, MCv_Jog (JOG) is executed in the reverse direction.
Velocity	Velocity	LREAL	At start	0.0, 0.0001 to 2500000000.0	0.0	This variable sets the command velocity.
Acceleration	Acceleration	LREAL	At start	0.0000, 0.0001 to 2147483647.0	0.0	This variable sets the acceleration.
Deceleration	Deceleration	LREAL	At start	0.0000, 0.0001 to 2147483647.0	0.0	This variable sets the deceleration.
Jerk	Jerk	LREAL	At start	0.0000, 0.0001 to 2147483647.0	0.0	This variable sets the jerk.
Options	Options	DWORD(HEX)	At start	00000000H to 00000001H	00000000H	This variable sets the functional options for MCv_Jog (JOG) in terms of bits.

■ Output variables

Output variable	Name	Data type	Default value	Description
Done	Execution completion	BOOL	FALSE	This variable becomes TRUE for only one scan when a deceleration stop is finished by turning off the JOG command.
Busy	Executing	BOOL	FALSE	This variable becomes TRUE when MCv_Jog (JOG) is executed.
Active	Controlling	BOOL	FALSE	This variable becomes TRUE while MCv_Jog (JOG) is controlling the axis.
CommandAborted	Abortion of execution	BOOL	FALSE	This variable becomes TRUE when the execution of MCv_Jog (JOG) is aborted.
Error	Error	BOOL	FALSE	This variable becomes TRUE when an error occurs.
ErrorID	Error code	WORD(UINT)	0	When an error occurs, this variable returns the error code. For details of error codes, refer to the following. 📖MELSEC iQ-R Motion Module User's Manual (Application)

Processing details

- The target axis moves in the specified direction when Forward rotation JOG command (JogForward) or Reverse rotation JOG command (JogBackward) is set to TRUE.
- Axis status (AxisName.Md.AxisStatus) is "6: During continuous operation (ContinuousMotion)" during JOG operation.
- A deceleration stop is performed when Forward rotation JOG command (JogForward) or Reverse rotation JOG command (JogBackward) is set to FALSE.
- Axis status (AxisName.Md.AxisStatus) changes to "4: Standby (Standstill)" at deceleration stop completion.
- If Error (Error) becomes TRUE during deceleration by Forward rotation JOG command (JogForward) or Reverse rotation JOG command (JogBackward) becoming FALSE, Error (Error) remains TRUE until Forward rotation JOG command (JogForward) or Reverse rotation JOG command (JogBackward) is set to TRUE.
- When another operation FB is started during JOG operation, the operation is performed based on the setting of Buffer mode (BufferMode) of the started operation FB.
- When JOG operation is started during another operation FB, the start request is ignored and "Start during Operation Warning (warning code: 0D01H)" occurs. Start JOG operation when Axis status (AxisName.Md.AxisStatus) is "4: Standby (Standstill)".
- To change the velocity during JOG operation, use the override function to perform the velocity change. For details of the override function, refer to the following.

📖MELSEC iQ-R Motion Module User's Manual (Application)

Program example

Create a program to move axis 1, axis 2, and axis 3 in the specified direction at the specified velocity while Forward rotation JOG command or Reverse rotation JOG command is input for each axis.

■ Labels used

The following table lists the global and local labels used in this program.

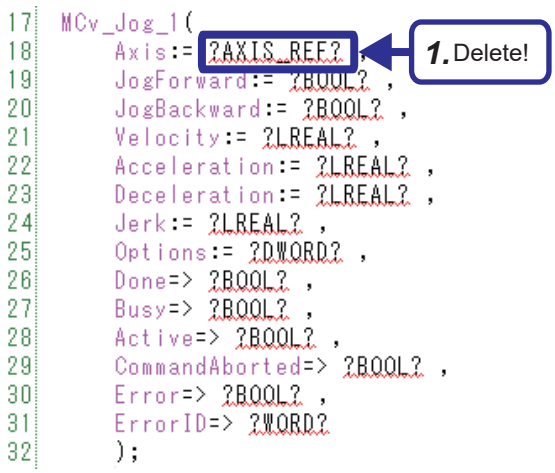
Category	Label name	Data type	Class	Public label	Description
Global label	G_bJogFwb_1	Bit	VAR_GLOBAL	Enable	Axis 1 Forward rotation JOG command This bit is turned on/off by the program for the PLC CPU. Turns on when "M0" is tapped on the GOT.
	G_bJogBwd_1	Bit	VAR_GLOBAL	Enable	Axis 1 Reverse rotation JOG command This bit is turned on/off by the program for the PLC CPU. Turns on when "M1" is tapped on the GOT.
	G_bJogBusy_1	Bit	VAR_GLOBAL	Enable	FB (axis 1) is running.
	G_bJogFwb_2	Bit	VAR_GLOBAL	Enable	Axis 2 Forward rotation JOG command This bit is turned on/off by the program for the PLC CPU. Turns on when "M2" is tapped on the GOT.
	G_bJogBwd_2	Bit	VAR_GLOBAL	Enable	Axis 2 Reverse rotation JOG command This bit is turned on/off by the program for the PLC CPU. Turns on when "M3" is tapped on the GOT.
	G_bJogBusy_2	Bit	VAR_GLOBAL	Enable	FB (axis 2) is running.
	G_bJogFwb_3	Bit	VAR_GLOBAL	Enable	Axis 3 Forward rotation JOG command This bit is turned on/off by the program for the PLC CPU. Turns on when "M4" is tapped on the GOT.
	G_bJogBwd_3	Bit	VAR_GLOBAL	Enable	Axis 3 Reverse rotation JOG command This bit is turned on/off by the program for the PLC CPU. Turns on when "M5" is tapped on the GOT.
	G_bJogBusy_3	Bit	VAR_GLOBAL	Enable	FB (axis 3) is running.
Local label	leJogVelocity_1	Double-precision real number	VAR	—	Axis 1 velocity Stores the JOG operation velocity of axis 1 input from the GOT.
	leJogAcceleration_1	Double-precision real number	VAR	—	Axis 1 acceleration Stores the acceleration based on Axis 1 velocity (leJogVelocity_1).
	leJogDeceleration_1	Double-precision real number	VAR	—	Axis 1 deceleration Stores the acceleration based on Axis 1 velocity (leJogVelocity_1).
	leJogJerk_1	Double-precision real number	VAR	—	Axis 1 jerk Stores the acceleration based on Axis 1 velocity (leJogVelocity_1).
	bJogError_1	Bit	VAR	—	Axis 1 error
	leJogVelocity_2	Double-precision real number	VAR	—	Axis 2 velocity Stores the JOG operation velocity of axis 2 input from the GOT.
	leJogAcceleration_2	Double-precision real number	VAR	—	Axis 2 acceleration Stores the acceleration based on Axis 2 velocity (leJogVelocity_2).
	leJogDeceleration_2	Double-precision real number	VAR	—	Axis 2 deceleration Stores the deceleration based on Axis 2 velocity (leJogVelocity_2).
	leJogJerk_2	Double-precision real number	VAR	—	Axis 2 jerk Stores the acceleration based on Axis 2 velocity (leJogVelocity_2).
	bJogError_2	Bit	VAR	—	Axis 2 error
	leJogVelocity_3	Double-precision real number	VAR	—	Axis 3 velocity Stores the JOG operation velocity of axis 3 input from the GOT.
	leJogAcceleration_3	Double-precision real number	VAR	—	Axis 3 acceleration Stores the acceleration based on Axis 3 velocity (leJogVelocity_3).
	leJogDeceleration_3	Double-precision real number	VAR	—	Axis 3 deceleration Stores the acceleration based on Axis 3 velocity (leJogVelocity_3).
	leJogJerk_3	Double-precision real number	VAR	—	Axis 3 jerk Stores the acceleration based on Axis 3 velocity (leJogVelocity_3).
	bJogError_3	Bit	VAR	—	Axis 3 error

■ Inputting the AxisRef type structure

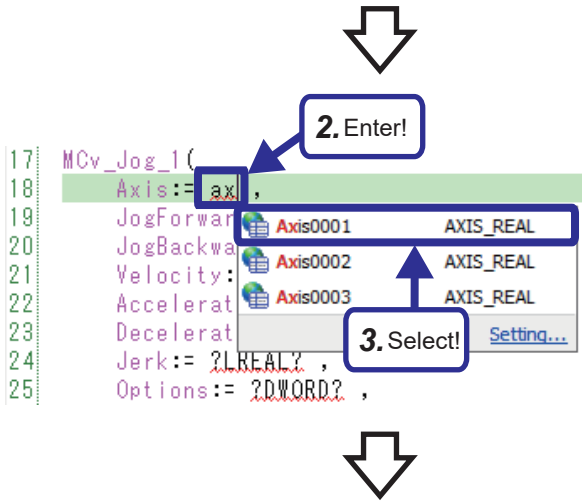
This section describes how to input the AxisRef type structure.

If there is "AXIS_REF" in the input setting/output setting of the motion control FB, the axis can be specified by setting the AXIS_REF type member AxisRef (AxisName.AxisRef) of each axis variable.

Operating procedure

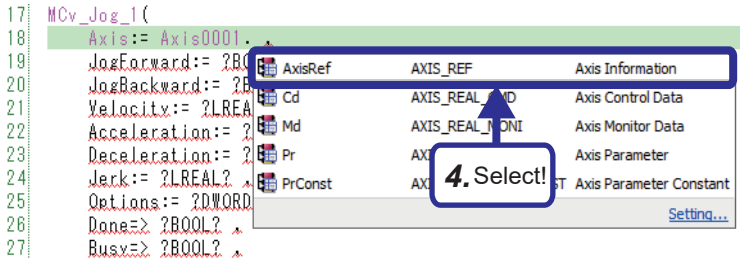


1. Delete "?AXIS_REF?" on the right of "Axis:=".



2. Enter "ax".

3. Select "Axis0001" from the list of registered options.



4. Enter ".", and select "AxisRef" from the list of structure options.

Practice 1

Create the program to execute JOG operation of axis 1 in the program block "ServoON_Jog".
Select "MCv_Jog" from [Motion - Individual] under [Motion Control Function/Function Block] in "POU List" in the Element Selection window.

```
//Axis 1: Velocity setting for JOG operation (velocity, acceleration, deceleration, and jerk)
IF (G_leJogVelocity_1 <> leJogVelocity_1) THEN
    leJogVelocity_1    := G_leJogVelocity_1;
    leJogAcceleration_1 := G_leJogVelocity_1 * 2.0;
    leJogDeceleration_1 := G_leJogVelocity_1 * 2.0;
    leJogJerk_1         := G_leJogVelocity_1 * 4.0;
END_IF;

//Axis 1: JOG operation execution
MCv_Jog_1(
    Axis:= Axis0001.AxisRef ,                    (1)
    JogForward:= G_bJogFwb_1 ,                   (2)
    JogBackward:= G_bJogBwd_1 ,                  (3)
    Velocity:= leJogVelocity_1 ,                 (4)
    Acceleration:= leJogAcceleration_1 ,         (5)
    Deceleration:= leJogDeceleration_1 ,         (6)
    Jerk:= leJogJerk_1 ,                         (7)
    Busy=> G_bJogBusy_1 ,                        (8)
    Error=> bJogError_1                          (9)
);
```

When the JOG speed of axis 1 is changed, acceleration, deceleration, and jerk are recalculated according to the speed.

No.	Description
(1)	Sets the axis information of axis 1.
(2)	Sets the command to perform JOG operation of axis 1 in the positive direction.
(3)	Sets the command to perform JOG operation of axis 1 in the negative direction.
(4)	Sets the command velocity for JOG operation (axis 1).
(5)	Sets the acceleration for JOG operation (axis 1).
(6)	Sets the deceleration for JOG operation (axis 1).
(7)	Sets the jerk for JOG operation (axis 1).
(8)	Stores the execution status of the JOG operation (axis 1) FB.
(9)	Stores errors (axis 1).

Fill in the blanks to complete a program.

```
//Axis 2: Velocity setting for JOG operation (velocity, acceleration, deceleration, and jerk)
IF (G_leJogVelocity_2 <> leJogVelocity_2) THEN
    leJogVelocity_2    := G_leJogVelocity_2;
    leJogAcceleration_2 := G_leJogVelocity_2 * 2.0;
    leJogDeceleration_2 := G_leJogVelocity_2 * 2.0;
    leJogJerk_2         := G_leJogVelocity_2 * 4.0;
END_IF;

//Axis 2: JOG operation execution
MCv_Jog_2(
    Axis:=[        (1)          ]
    JogForward:=[      (2)      ]
    JogBackward:=[      (3)      ]
    Velocity:=[        (4)        ]
    Acceleration:=[           (5)           ]
    Deceleration:=[           (6)           ]
    Jerk:=[      (7)      ]
    Busy=>[      (8)      ]
    Error=>[      (9)      ]
);
```

When the JOG speed of axis 2 is changed, acceleration, deceleration, and jerk are recalculated according to the speed.

No.	Description
(1)	Sets the axis information of axis 2.
(2)	Sets the command to perform JOG operation of axis 2 in the forward rotation direction.
(3)	Sets the command to perform JOG operation of axis 2 in the reverse rotation direction.
(4)	Sets the command velocity for JOG operation (axis 2).
(5)	Sets the acceleration for JOG operation (axis 2).
(6)	Sets the deceleration for JOG operation (axis 2).
(7)	Sets the jerk for JOG operation (axis 2).
(8)	Stores the execution status of the JOG operation (axis 2) FB.
(9)	Stores errors (axis 2).

```
//Axis 3: Velocity setting for JOG operation (velocity, acceleration, deceleration, and jerk)
IF (G_leJogVelocity_3 <> leJogVelocity_3) THEN
    leJogVelocity_3    := G_leJogVelocity_3;
    leJogAcceleration_3 := G_leJogVelocity_3 * 2.0;
    leJogDeceleration_3 := G_leJogVelocity_3 * 2.0;
    leJogJerk_3         := G_leJogVelocity_3 * 4.0;
END_IF;

//Axis 3: JOG operation execution
MCv_Jog_3(
    Axis:= [        (1)        ]
    JogForward:= [      (2)      ]
    JogBackward:=[      (3)      ]
    Velocity:= [        (4)        ]
    Acceleration:=[             (5)             ]
    Deceleration:=[             (6)             ]
    Jerk:=[       (7)       ]
    Busy=>[       (8)       ]
    Error=>[      (9)       ]
);

//Notify the PLC of error occurrence in JOG operation control
G_bJogError := bJogError_1 OR bJogError_2 OR bJogError_3;
```

When the JOG speed of axis 3 is changed, acceleration, deceleration, and jerk are recalculated according to the speed.

Errors of each axis is stored.

No.	Description
(1)	Sets the axis information of axis 3.
(2)	Sets the command to perform JOG operation of axis 3 in the forward rotation direction.
(3)	Sets the command to perform JOG operation of axis 3 in the reverse rotation direction.
(4)	Sets the command velocity for JOG operation (axis 3).
(5)	Sets the acceleration for JOG operation (axis 3).
(6)	Sets the deceleration for JOG operation (axis 3).
(7)	Sets the jerk for JOG operation (axis 3).
(8)	Stores the execution status of the JOG operation (axis 3) FB.
(9)	Stores errors (axis 3).

■ Answer

The following shows the answer program.

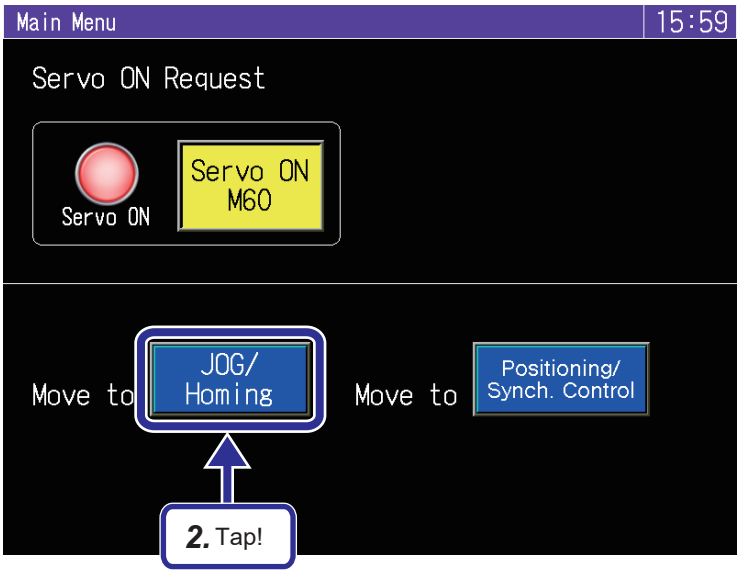
```
 9  //Axis 1: Velocity setting for JOG operation (velocity, acceleration, deceleration, and jerk)
10  IF (G_leJogVelocity_1 <> leJogVelocity_1) THEN
11      leJogVelocity_1   := G_leJogVelocity_1;
12      leJogAcceleration_1 := G_leJogVelocity_1 * 2.0;
13      leJogDeceleration_1 := G_leJogVelocity_1 * 2.0;
14      leJogJerk_1         := G_leJogVelocity_1 * 4.0;
15  END_IF;
16
17  //Axis 1: JOG operation execution
18  MCv_Jog_1(
19      Axis:= Axis0001.AxisRef ,
20      JogForward:= G_bJogFwb_1 ,
21      JogBackward:= G_bJogBwd_1 ,
22      Velocity:= leJogVelocity_1 ,
23      Acceleration:= leJogAcceleration_1 ,
24      Deceleration:= leJogDeceleration_1 ,
25      Jerk:= leJogJerk_1 ,
26      Busy=> G_bJogBusy_1 ,
27      Error=> bJogError_1
28  );
29
30  //Axis 2: Velocity setting for JOG operation (velocity, acceleration, deceleration, and jerk)
31  IF (G_leJogVelocity_2 <> leJogVelocity_2) THEN
32      leJogVelocity_2   := G_leJogVelocity_2;
33      leJogAcceleration_2 := G_leJogVelocity_2 * 2.0;
34      leJogDeceleration_2 := G_leJogVelocity_2 * 2.0;
35      leJogJerk_2         := G_leJogVelocity_2 * 4.0;
36  END_IF;
37
38  //Axis 2: JOG operation execution
39  MCv_Jog_2(
40      Axis:= Axis0002.AxisRef ,
41      JogForward:= G_bJogFwb_2 ,
42      JogBackward:= G_bJogBwd_2 ,
43      Velocity:= leJogVelocity_2 ,
44      Acceleration:= leJogAcceleration_2 ,
45      Deceleration:= leJogDeceleration_2 ,
46      Jerk:= leJogJerk_2 ,
47      Busy=> G_bJogBusy_2 ,
48      Error=> bJogError_2
49  );
50
```

```
//Axis 3: Velocity setting for JOG operation (velocity, acceleration, deceleration, and jerk)
IF (G_leJogVelocity_3 <> leJogVelocity_3) THEN
    leJogVelocity_3   := G_leJogVelocity_3;
    leJogAcceleration_3 := G_leJogVelocity_3 * 2.0;
    leJogDeceleration_3 := G_leJogVelocity_3 * 2.0;
    leJogJerk_3         := G_leJogVelocity_3 * 4.0;
END_IF;

//Axis 3: JOG operation execution
MCv_Jog_3(
    Axis:= Axis0003.AxisRef ,
    JogForward:= G_bJogFwb_3 ,
    JogBackward:= G_bJogBwd_3 ,
    Velocity:= leJogVelocity_3 ,
    Acceleration:= leJogAcceleration_3 ,
    Deceleration:= leJogDeceleration_3 ,
    Jerk:= leJogJerk_3 ,
    Busy=> G_bJogBusy_3 ,
    Error=> bJogError_3
);

//Notify the PLC of error occurrence in JOG operation control
G_bJogError := bJogError_1 OR bJogError_2 OR bJogError_3;
```

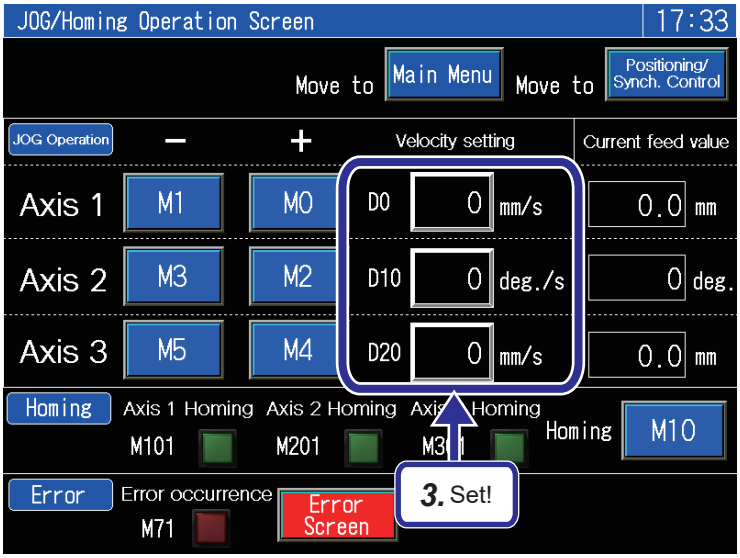
Operation check

After creating the program, write it to the programmable controller by following the same procedure as "☞ Page 110 Writing to the programmable controller" and check the operation of single axis manual control (JOG operation).

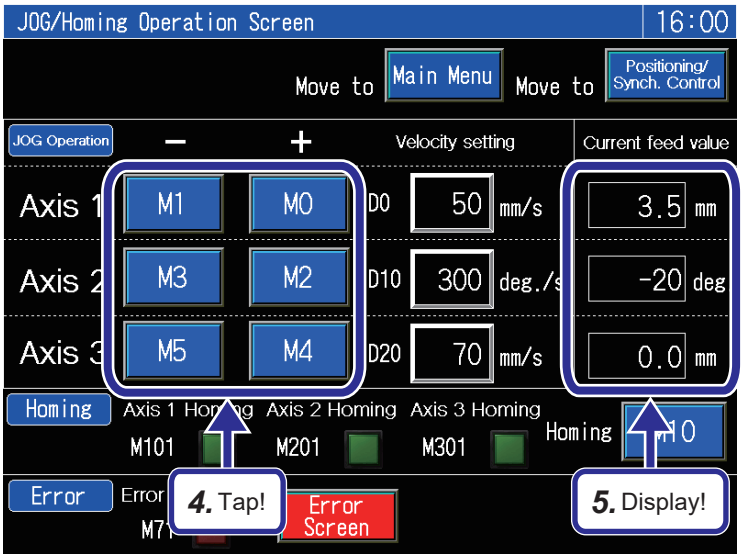
Operating procedure



1. Turn on the servo.
2. Tap the [JOG homing] button on the main menu screen.



3. Set the velocity of each axis in Velocity setting in the JOG/Homing Operation Screen.



4. While the JOG button corresponding to each axis is tapped and held, JOG operation is performed at the velocity specified in Velocity setting.

As for axis 1 and axis 3, tapping the + side button moves the axis in the forward rotation direction (right), and tapping the - side button moves the axis in the reverse rotation direction (left).

As for axis 2, tapping the + side button moves the axis vertically in counterclockwise, and tapping the - side button moves the axis vertically in clockwise.

If a stroke limit is detected in axis 3, an error occurs and the operation stops.

5. The current value of each axis is displayed in Current feed value.

Homing control

FB name: MC_Home

This FB performs homing of the specified axis.

The following shows the details of MC_Home.

```
MC_Home(
  Axis:= ?AXIS_REF? ,
  Execute:= ?BOOL? ,
  Position:= ?LREAL? ,
  AbsSwitch:= ?MC_INPUT_REF? ,
  Options:= ?DWORD? ,
  Done=> ?BOOL? ,
  Busy=> ?BOOL? ,
  Active=> ?BOOL? ,
  CommandAborted=> ?BOOL? ,
  Error=> ?BOOL? ,
  ErrorID=> ?WORD?
);
```

Name	Number of input area points (byte)	Number of output area points (byte)	Compilation method	FB operation
OPR	188	8	Subroutine type	Real-time execution

Setting data

■ I/O variable

I/O variable	Name	Data type	Input import	Setting range	Default value	Description
Axis	Axis information	AXIS_REF	At start	—	Mandatory	This variable sets the axis. For the variables used (AxisName.AxisRef.), refer to the following. ☞ Page 45 AxisName.AxisRef. (Axis information)

■ Input variables

Input variable	Name	Data type	Import	Setting range	Default value	Description
Execute	Execute command	BOOL	At start	TRUE, FALSE	FALSE	When this variable is TRUE, MC_Home (OPR) is executed.
Position	Target position	LREAL	At start	-10000000000.0 to 10000000000.0	0.0	This variable sets the home position address. Set the address within the following range. • -10000000000.0 ≤Setting value < 10000000000.0 When the ring counter is enabled, the address must be within the range of the ring counter.
AbsSwitch	Home position switch	MC_INPUT_REF	At start	—	—	This variable sets the proximity dog signal transmitted to the device station in the driver homing method.
Options	Options	DWORD(HEX)	At start	00000000H	00000000H	Set this variable to "00000000H". When a value other than "00000000H" is set, "Out of Options Range (error code: 1A4EH)" occurs.

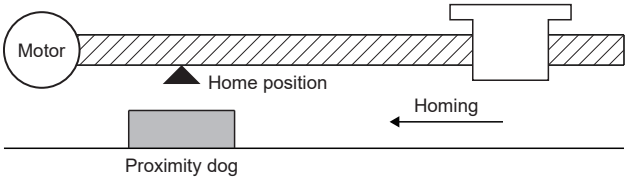
■ Output variables

Output variable	Name	Data type	Default value	Description
Done	Execution completion	BOOL	FALSE	This variable becomes TRUE when homing is complete.
Busy	Executing	BOOL	FALSE	This variable becomes TRUE when MC_Home (OPR) is executed.
Active	Controlling	BOOL	FALSE	This variable becomes TRUE while MC_Home (OPR) is controlling the axis.
CommandAborted	Abortion of execution	BOOL	FALSE	This variable becomes TRUE when the execution of MC_Home (OPR) is aborted.
Error	Error	BOOL	FALSE	This variable becomes TRUE when an error occurs.

Output variable	Name	Data type	Default value	Description
ErrorID	Error code	WORD(UINT)	0	When an error occurs, this variable returns the error code. For details of error codes, refer to the following. MELSEC iQ-R Motion Module User's Manual (Application)

Processing details

- In homing control, a machine home position is determined. None of the address information stored in the motion system or driver is used at this time. After homing, the mechanically determined position is regarded as the "home position", which is the start point of positioning control.



- For the homing method, "Driver homing method" and "Data set homing method" are available.

The homing method at the start of homing is determined by the following conditions.

Homing method	Condition for homing method
Driver homing method	Driver homing method is used when all of the following conditions are satisfied. • The axis type is real drive axis. • The driver supports Homing mode. • "Home offset (607CH)" is set to a slave object.
Data set homing method	Data set homing method is used when the above conditions are not satisfied.

Program example

Create a program that resets the current feed value of each axis and performs homing of axis 1, axis 2, and axis 3.

■ Labels used

The following table lists the global and local labels used in this program.

Category	Label name	Data type	Class	Public label	Description
Global label	G_bHomingReq_1	Bit	VAR_GLOBAL	Enable	Axis 1 homing execution command Turns on when "M10" is tapped on the GOT.
	G_bHomingReq_2	Bit	VAR_GLOBAL	Enable	Axis 2 homing execution command Turns on when "M10" is tapped on the GOT.
	G_bHomingReq_3	Bit	VAR_GLOBAL	Enable	Axis 3 homing execution command Turns on when "M10" is tapped on the GOT.
Local label	bHomingDone_1	Bit	VAR	—	Execution completion (axis 1)
	bHomingAborted_1	Bit	VAR	—	Axis 1 execution aborted
	bHomingError_1	Bit	VAR	—	Axis 1 error
	bHomingDone_2	Bit	VAR	—	Execution completion (axis 2)
	bHomingAborted_2	Bit	VAR	—	Axis 2 execution aborted
	bHomingError_2	Bit	VAR	—	Axis 2 error
	bHomingDone_3	Bit	VAR	—	Execution completion (axis 3)
	bHomingAborted_3	Bit	VAR	—	Axis 3 execution aborted
	bHomingError_3	Bit	VAR	—	Axis 3 error

Practice 2

Create a program that executes homing of axis 1 in the program block "Homing".
Select "MC_Home" from [Motion - Individual] under [Motion Control Function/Function Block] in "POU List" in the Element Selection window.

```
1  //------Homing------
2  //Axis 1: Homing execution request
3  IF G_bHomingCMD_1 THEN
4      G_bHomingReq_1 := TRUE;
5      ELSE
6      G_bHomingReq_1 := FALSE;
7  END_IF;
8
9  //Axis 2: Homing execution request
10 IF G_bHomingCMD_2 THEN
11     G_bHomingReq_2 := TRUE;
12     ELSE
13     G_bHomingReq_2 := FALSE;
14 END_IF;
15
16 //Axis 3: Homing execution request
17 IF G_bHomingCMD_3 THEN
18     G_bHomingReq_3 := TRUE;
19     ELSE
20     G_bHomingReq_3 := FALSE;
21 END_IF;
22
23 //Axis 3: Temporarily disable hardware stroke limit error detection only during homing
24 IF G_bHomingReq_3 THEN
25     Axis0003.Cd.HwStrokeLimit_Override := 'DISABLE';
26     ELSE
27     Axis0003.Cd.HwStrokeLimit_Override := '';
28 END_IF;
29
30 //Axis 1: Homing execution
31 MC_Home_1(
32 Axis:= Axis0001.AxisRef ,          (1)
33 Execute:= G_bHomingReq_1 ,         (2)
34 Position:= 0.0 ,                  (3)
35 Done=> bHomingDone_1 ,            (4)
36 CommandAborted=> bHomingAborted_1 , (5)
37 Error=> bHomingError_1             (6)
38 );
39
```

When the homing requests for axis 1, axis 2, and axis 3 are turned on, homing execute commands for axis 1, axis 2, and axis 3 turn on.

No.	Description
(1)	Sets the axis information of axis 1.
(2)	Sets the command to perform homing of axis 1.
(3)	Sets the home position address of axis 1.
(4)	Stores "Execution completion" of the homing (axis 1) FB.
(5)	Stores "Abortion of execution" of the homing (axis 1) FB.
(6)	Stores errors (axis 1).

Fill in the blanks to complete a program.

```
40  //Axis 2: Homing execution
41  MC_Home_2(
42  Axis:= [        (1)        ],
43  Execute:= [       (2)        ],
44  Position:=[ (3) ],
45  Done=>[       (4)        ],
46  CommandAborted=> [        (5)        ],
47  Error=>[       (6)        ]
48  );
49
```

No.	Description
(1)	Sets the axis information of axis 2.
(2)	Sets the command to perform homing of axis 2.
(3)	Sets the home position address of axis 2.
(4)	Stores "Execution completion" of the homing (axis 2) FB.
(5)	Stores "Abortion of execution" of the homing (axis 2) FB.
(6)	Stores errors (axis 2).

```
50  //Axis 3: Homing execution
51  MC_Home_3(
52  Axis:=[        (1)        ] ,
53  Execute:=[       (2)        ] ,
54  Position:= [(3)] ,
55  Done=>[      (4)       ] ,
56  CommandAborted=>[        (5)        ]
57  Error=>[      (6)       ]
58  );
59
60  //Notify the PLC of error occurrence in homing control
61  G_bHomingError := bHomingError_1 OR bHomingError_2 OR bHomingError_3;   } Errors of each axis is stored.
62
63  //Notify the PLC of homing control execution completion
64  G_bHomingDone_1 := bHomingDone_1 OR bHomingAborted_1 OR bHomingError_1;
65  G_bHomingDone_2 := bHomingDone_2 OR bHomingAborted_2 OR bHomingError_2;   } Each axis status is stored.
66  G_bHomingDone_3 := bHomingDone_3 OR bHomingAborted_3 OR bHomingError_3;
```

No.	Description
(1)	Sets the axis information of axis 3.
(2)	Sets the command to perform homing of axis 3.
(3)	Sets the home position address of axis 3.
(4)	Stores "Execution completion" of the homing (axis 3) FB.
(5)	Stores "Abortion of execution" of the homing (axis 3) FB.
(6)	Stores errors (axis 3).

■ Answer

The following shows the answer program.

```
//------Homing------
//Axis 1: Homing execution request
IF G_bHomingCMD_1 THEN
    G_bHomingReq_1 := TRUE;
    ELSE
    G_bHomingReq_1 := FALSE;
END_IF;

//Axis 2: Homing execution request
IF G_bHomingCMD_2 THEN
    G_bHomingReq_2 := TRUE;
    ELSE
    G_bHomingReq_2 := FALSE;
END_IF;

//Axis 3: Homing execution request
IF G_bHomingCMD_3 THEN
    G_bHomingReq_3 := TRUE;
    ELSE
    G_bHomingReq_3 := FALSE;
END_IF;

//Axis 3: Temporarily disable hardware stroke limit error detection only during homing
IF G_bHomingReq_3 THEN
    Axis0003.Cd.HwStrokeLimit_Override := 'DISABLE';
    ELSE
    Axis0003.Cd.HwStrokeLimit_Override := '';
END_IF;

//Axis 1: Homing execution
MC_Home_1(
Axis:= Axis0001.AxisRef ,
Execute:= G_bHomingReq_1 ,
Position:= 0.0 ,
Done=> bHomingDone_1 ,
CommandAborted=> bHomingAborted_1 ,
Error=> bHomingError_1
);

//Axis 2: Homing execution
MC_Home_2(
Axis:= Axis0002.AxisRef ,
Execute:= G_bHomingReq_2 ,
Position:= 0.0 ,
Done=> bHomingDone_2 ,
CommandAborted=> bHomingAborted_2 ,
Error=> bHomingError_2
);

//Axis 3: Homing execution
MC_Home_3(
Axis:= Axis0003.AxisRef ,
Execute:= G_bHomingReq_3 ,
Position:= 0.0 ,
Done=> bHomingDone_3 ,
CommandAborted=> bHomingAborted_3 ,
Error=> bHomingError_3
);

//Notify the PLC of error occurrence in homing control
G_bHomingError := bHomingError_1 OR bHomingError_2 OR bHomingError_3;

//Notify the PLC of homing control execution completion
G_bHomingDone_1 := bHomingDone_1 OR bHomingAborted_1 OR bHomingError_1;
G_bHomingDone_2 := bHomingDone_2 OR bHomingAborted_2 OR bHomingError_2;
G_bHomingDone_3 := bHomingDone_3 OR bHomingAborted_3 OR bHomingError_3;
```

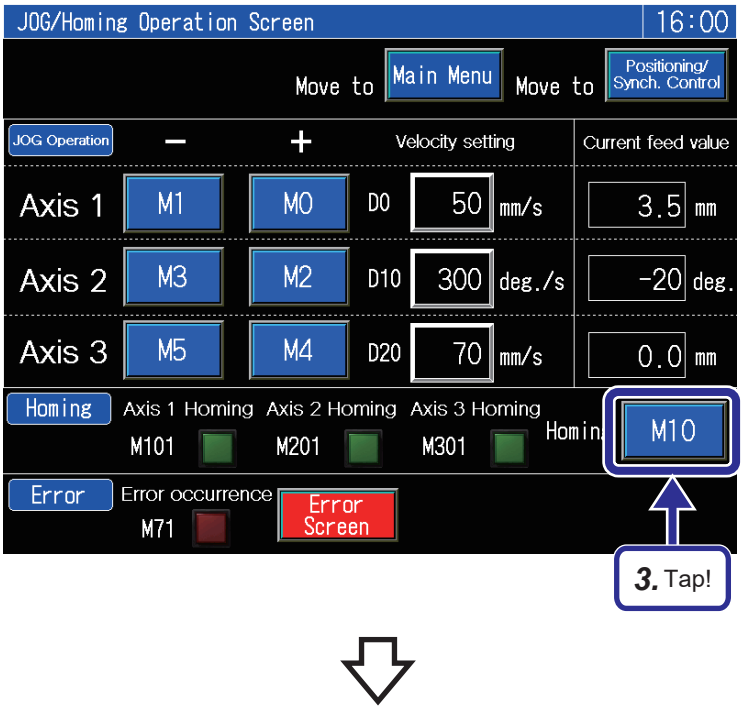
Operation check

After creating the program, write it to the programmable controller by following the same procedure as "☞ Page 110 Writing to the programmable controller" and check the operation of homing.

Operating procedure

1. Turn on the servo, and open the JOG/Homing Operation Screen.

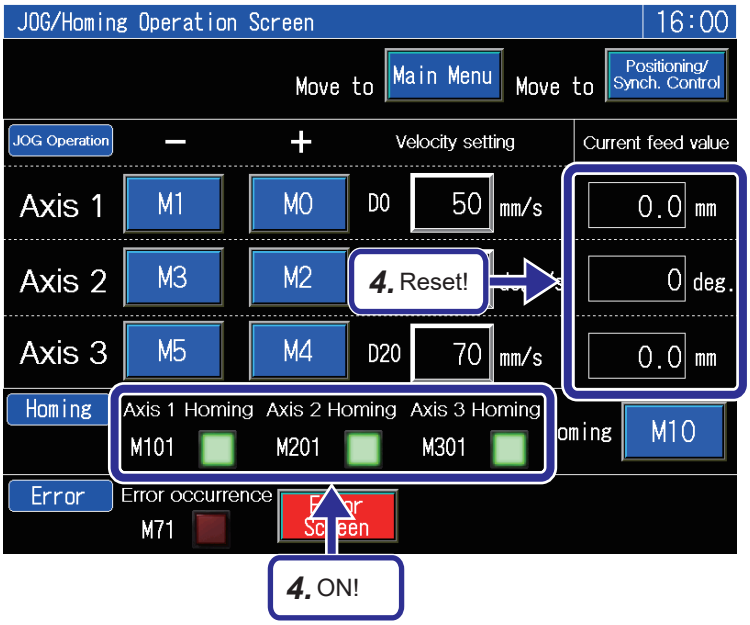
2. Move each axis to the desired position by JOG operation.



3. Tap the [M10] button for homing.

Axis 2 moves vertically and homing ends by the input signal from the dog sensor.

Axis 3 moves in the negative direction (left) and homing ends by the input signal from the dog sensor.



4. When homing ends, "M101", "M201", and "M301" lamps turn on. In addition, the current feed values are reset.

7

Single axis positioning control

FB name: MC_MoveRelative

Sets the relative movement amount and executes positioning.

The following shows the details of MC_MoveRelative.

```
MC_MoveRelative(
  Axis:= ?AXIS_REF? ,
  Execute:= ?BOOL? ,
  ContinuousUpdate:= ?BOOL? ,
  Distance:= ?LREAL? ,
  Velocity:= ?LREAL? ,
  Acceleration:= ?LREAL? ,
  Deceleration:= ?LREAL? ,
  Jerk:= ?LREAL? ,
  BufferMode:= ?INT? ,
  Options:= ?DWORD? ,
  Done=> ?BOOL? ,
  Busy=> ?BOOL? ,
  Active=> ?BOOL? ,
  CommandAborted=> ?BOOL? ,
  Error=> ?BOOL? ,
  ErrorID=> ?WORD?
);
```

Name	Number of input area points (byte)	Number of output area points (byte)	Compilation method	FB operation
Relative Value Positioning	64	8	Subroutine type	Real-time execution

Setting data

■ I/O variable

I/O variable	Name	Data type	Input import	Setting range	Default value	Description
Axis	Axis information	AXIS_REF	At start	—	Mandatory	This variable sets the axis. For the variables used (AxisName.AxisRef.), refer to the following. ☞ Page 45 AxisName.AxisRef. (Axis information)

■ Input variables

Input variable	Name	Data type	Import	Setting range	Default value	Description
Execute	Execute command	BOOL	At start	TRUE, FALSE	FALSE	When this variable is TRUE, MC_MoveRelative (Relative Value Positioning) is executed.
ContinuousUpdate	Continuous update	BOOL	At start	TRUE, FALSE	FALSE	This variable sets whether to enable or disable continuous change of Movement amount (Distance), Velocity (Velocity), Acceleration (Acceleration), and Deceleration (Deceleration). • FALSE:Disable • TRUE:Enable
Distance	Movement amount	LREAL	At start / Retrigger possible / Continuous update possible	-10000000000.0 to 10000000000.0	0.0	This variable sets the relative position from the current position at start to the end point.
Velocity	Velocity	LREAL	At start / Retrigger possible / Continuous update possible	0.0, 0.0001 to 2500000000.0	0.0	This variable sets the velocity.

Input variable	Name	Data type	Import	Setting range	Default value	Description
Acceleration	Acceleration	LREAL	At start / Retrigger possible / Continuous update possible	0.0000, 0.0001 to 2147483647.0	0.0	This variable sets the acceleration.
Deceleration	Deceleration	LREAL	At start / Retrigger possible / Continuous update possible	0.0000, 0.0001 to 2147483647.0	0.0	This variable sets the deceleration.
Jerk	Jerk	LREAL	At start	0.0000, 0.0001 to 2147483647.0	0.0	This variable sets the jerk.
BufferMode	Buffer mode	INT (MC_BUFFER_MODE)	At start	0 to 5	0	This variable sets the buffer mode. • 0:Aborting (mcAborting) • 1:Buffered (mcBuffered) • 2:BlendingLow (mcBlendingLow) • 3:BlendingPrevious (mcBlendingPrevious) • 4:BlendingNext (mcBlendingNext) • 5:BlendingHigh (mcBlendingHigh) For details of the buffer mode, refer to ☞ Page 141 Multiple start (buffer mode).
Options	Options	DWORD(HEX)	At start	00000000H to 00000021H	00000000H	This variable sets the functional options for MC_MoveRelative (Relative Value Positioning).

■ Output variables

Output variable	Name	Data type	Default value	Description
Done	Execution completion	BOOL	FALSE	This variable becomes TRUE when the relative position is reached.
Busy	Executing	BOOL	FALSE	This variable becomes TRUE when MC_MoveRelative (Relative Value Positioning) is executed. This variable becomes FALSE after the axis reaches the relative position.
Active	Controlling	BOOL	FALSE	This variable becomes TRUE while MC_MoveRelative (Relative Value Positioning) is controlling the axis. This variable becomes FALSE after the axis reaches the relative position.
CommandAborted	Abortion of execution	BOOL	FALSE	This variable becomes TRUE when the execution of MC_MoveRelative (Relative Value Positioning) is aborted. This variable becomes FALSE when Execute command (Execute) becomes FALSE.
Error	Error	BOOL	FALSE	This variable becomes TRUE when an error occurs.
ErrorID	Error code	WORD(UINT)	0	When an error occurs, this variable returns the error code. For details of error codes, refer to the following. MELSEC iQ-R Motion Module User's Manual (Application)

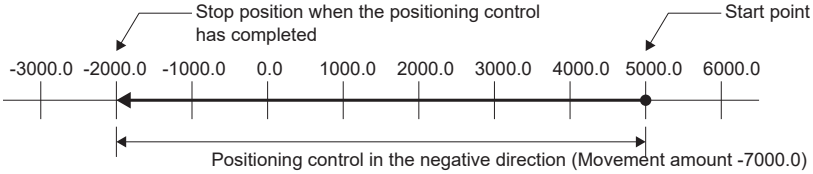
Processing details

- This FB sets Movement amount (Distance), Velocity (Velocity), Acceleration (Acceleration), Deceleration (Deceleration), Jerk (Jerk), and Buffer mode (BufferMode), then executes positioning from the current position at start (start point position) based on the movement amount set in Movement amount (Distance). The movement direction is determined by the sign of the movement amount. Axis status (AxisName.Md.AxisStatus) becomes "5: During positioning operation (DiscreteMotion)".

Ex.

When the start point position (current stop position) is "5000.0" and the movement amount is set to "-7000.0"

- The axis will be positioned at "-2000.0".



Program example

Create a program that sets the target position and moves axis 1 to the target position.

■ Labels used

The following table lists the global and local labels used in this program.

Category	Label name	Data type	Class	Public label	Description
Global label	G_lePosition	Double-precision real number	VAR_GLOBAL	Enable	Target position Stores the positioning target position input from the GOT.
Local label	bPositionReq	Bit	VAR	—	Positioning execution command Turns on when "M20" is tapped on the GOT.
	leVelocity	Double-precision real number	VAR	—	Positioning velocity Stores the positioning velocity input from the GOT.
	leAcceleration	Double-precision real number	VAR	—	Acceleration Stores the acceleration based on Positioning velocity (leVelocity).
	leDeceleration	Double-precision real number	VAR	—	Deceleration Stores the acceleration based on Positioning velocity (leVelocity).
	leJerk	Double-precision real number	VAR	—	Jerk Stores the acceleration based on Positioning velocity (leVelocity).
	bMoveRelaDone	Bit	VAR	—	Positioning execution complete
	bMoveRelaBusy	Bit	VAR	—	Positioning in progress
	bMoveRelaActive	Bit	VAR	—	Positioning controlling
	bMoveRelaAborted	Bit	VAR	—	Positioning execution aborted
	bMoveRelaError	Bit	VAR	—	Positioning error

Practice 3

Create a program that executes positioning control in the program block "Positioning".
Select "MC_MoveRelative" from [Motion - Individual] under [Motion Control Function/Function Block] in "POU List" in the Element Selection window.

```
1  //------Positioning control------
2  //Positioning control execution request
3  IF G_bPositionCMD THEN
4      bPositionReq := TRUE;
5      ELSE
6      bPositionReq := FALSE;
7  END_IF;
8
9  //Velocity setting for positioning control (velocity, acceleration, deceleration, and jerk)
10 IF (G_leSetVelocity <> leVelocity) THEN
11     leVelocity     := G_leSetVelocity;
12     leAcceleration := G_leSetVelocity * 2.0;
13     leDeceleration := G_leSetVelocity * 2.0;
14     leJerk         := G_leSetVelocity * 4.0;
15     //Acceleration, deceleration, and jerk are negative during the travel in the negative direction
16     IF (leVelocity < E0) THEN
17         EDNEG(TRUE,leAcceleration);
18         EDNEG(TRUE,leDeceleration);
19         EDNEG(TRUE,leJerk);
20     END_IF;
21 END_IF;
22
23 //Positioning control execution
24 MC_MoveRelative_1(
25     Axis:= Axis0001.AxisRef ,                    (1)
26     Execute:= bPositionReq ,                     (2)
27     ContinuousUpdate:= TRUE ,                    (3)
28     Distance:= G_lePosition ,                    (4)
29     Velocity:= leVelocity ,                      (5)
30     Acceleration:= leAcceleration ,              (6)
31     Deceleration:= leDeceleration ,              (7)
32     Jerk:= leJerk ,                              (8)
33     Done=> bMoveRelaDone ,                       (9)
34     Busy=> bMoveRelaBusy ,                       (10)
35     Active=> bMoveRelaActive ,                   (11)
36     CommandAborted=> bMoveRelaAborted ,          (12)
37     Error=> bMoveRelaError                       (13)
38     );
```

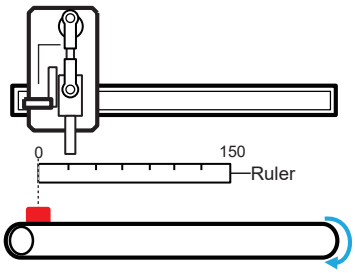
When the positioning speed of axis 1 is changed, acceleration, deceleration, and jerk are recalculated according to the speed.

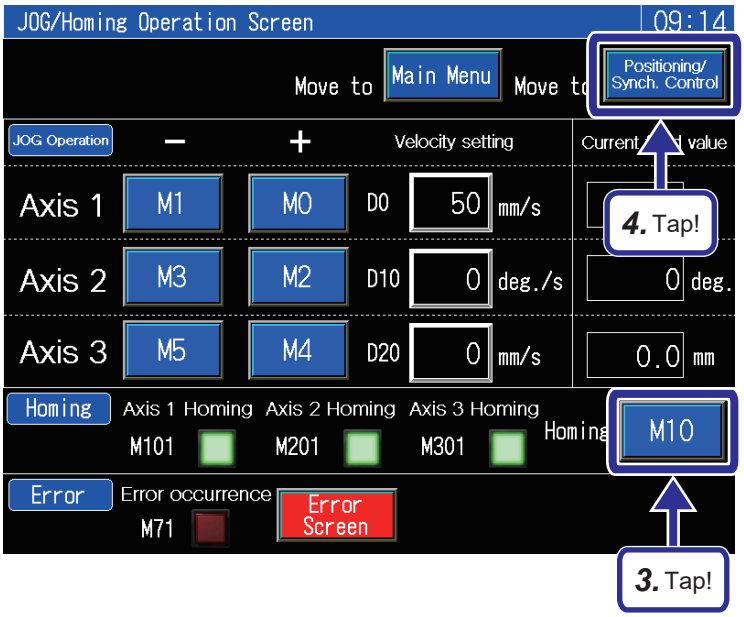
No.	Description
(1)	Sets the axis information of axis 1.
(2)	Sets the command to perform relative positioning of axis 1.
(3)	Enables the continuous update in relative positioning (axis 1).
(4)	Sets the target position of relative positioning (axis 1).
(5)	Sets the velocity of relative positioning (axis 1).
(6)	Sets the acceleration of relative positioning (axis 1).
(7)	Sets the deceleration of relative positioning (axis 1).
(8)	Sets the jerk of relative positioning (axis 1).
(9)	Stores "Execution completion" of the relative positioning (axis 1) FB.
(10)	Stores "Executing" of the relative positioning (axis 1) FB.
(11)	Stores "Controlling" of the relative positioning (axis 1) FB.
(12)	Stores "Abortion of execution" of the relative positioning (axis 1) FB.
(13)	Stores errors (axis 1).

Operation check

Check the operation of single axis positioning control.

Operating procedure





1. Turn on the servo.

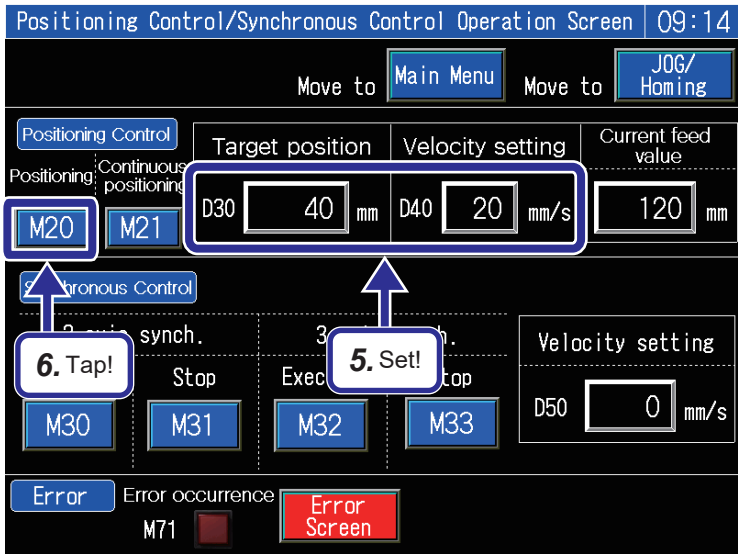
2. Perform JOG operation of axis 1 on the JOG/ Homing Operation Screen to move the red workpiece to the point near 0 on the ruler.

3. Tap the [M10] button for homing.

Point

Always perform homing before executing single axis positioning control.

4. Tap the [Positioning/Synch. Control] button.



5. Set the desired values for the target position "D30" and velocity setting "D40".

Point

The target position set in this step should be within the range of the ruler (1 to 150 mm).

6. Tap the [M20] button for positioning.

When positioning control starts, the red workpiece on axis 1 is moved to the target position at the set velocity.

7. Check that the workpiece has been moved to the target position with the ruler.

Point

The movement amount can be obtained from the current feed value as well.

However, the current value should be reset by homing because it is updated by JOG operation.

Single axis continuous positioning control

FB name: MC_MoveAbsolute

Sets the absolute target position and executes positioning.

The following shows the details of MC_MoveAbsolute.

```
MC_MoveAbsolute(
  Axis:= ?AXIS_REF? ,
  Execute:= ?BOOL? ,
  ContinuousUpdate:= ?BOOL? ,
  Position:= ?LREAL? ,
  Velocity:= ?LREAL? ,
  Acceleration:= ?LREAL? ,
  Deceleration:= ?LREAL? ,
  Jerk:= ?LREAL? ,
  Direction:= ?INT? ,
  BufferMode:= ?INT? ,
  Options:= ?DWORD? ,
  Done=> ?BOOL? ,
  Busy=> ?BOOL? ,
  Active=> ?BOOL? ,
  CommandAborted=> ?BOOL? ,
  Error=> ?BOOL? ,
  ErrorID=> ?WORD?
);
```

Name	Number of input area points (byte)	Number of output area points (byte)	Compilation method	FB operation
Absolute Value Positioning	64	8	Subroutine type	Real-time execution

Setting data

■ I/O variable

I/O variable	Name	Data type	Input import	Setting range	Default value	Description
Axis	Axis information	AXIS_REF	At start	—	Mandatory	This variable sets the axis. For the variables used (AxisName.AxisRef.), refer to the following. ☞ Page 45 AxisName.AxisRef. (Axis information)

■ Input variables

Input variable	Name	Data type	Import	Setting range	Default value	Description
Execute	Execute command	BOOL	At start	TRUE, FALSE	FALSE	When this variable is TRUE, MC_MoveAbsolute (Absolute Value Positioning) is executed.
ContinuousUpdate	Continuous update	BOOL	At start	TRUE, FALSE	FALSE	This variable sets whether to enable or disable continuous change of Target position (Position), Velocity (Velocity), Acceleration (Acceleration), and Deceleration (Deceleration). • FALSE:Disable • TRUE:Enable
Position	Target position	LREAL	At start / Retrigger possible / Continuous update possible	-10000000000.0 to 10000000000.0	0.0	This variable sets the absolute target position. Different setting ranges are applied to each setting.
Velocity	Velocity	LREAL	At start / Retrigger possible / Continuous update possible	0.0, 0.0001 to 2500000000.0	0.0	This variable sets the velocity.

Input variable	Name	Data type	Import	Setting range	Default value	Description
Acceleration	Acceleration	LREAL	At start / Retrigger possible / Continuous update possible	0.0000, 0.0001 to 2147483647.0	0.0	This variable sets the acceleration.
Deceleration	Deceleration	LREAL	At start / Retrigger possible / Continuous update possible	0.0000, 0.0001 to 2147483647.0	0.0	This variable sets the deceleration.
Jerk	Jerk	LREAL	At start	0.0000, 0.0001 to 2147483647.0	0.0	This variable sets the jerk.
Direction	Direction selection	INT (MC_DIRECTION)	At start	1 to 3	0	When software stroke limit is disabled, this variable sets the movement direction from the current position to the target position. • 1:Positive direction (mcPositiveDirection) • 2:Negative direction (mcNegativeDirection) • 3: Shortest path (mcShortestWay) When this setting is omitted, "Out of Direction Selection Range (error code: 1A37H)" occurs.
BufferMode	Buffer mode	INT (MC_BUFFER_MODE)	At start	0 to 5	0	This variable sets the buffer mode. • 0:Aborting (mcAborting) • 1:Buffered (mcBuffered) • 2:BlendingLow (mcBlendingLow) • 3:BlendingPrevious (mcBlendingPrevious) • 4:BlendingNext (mcBlendingNext) • 5:BlendingHigh (mcBlendingHigh)
Options	Options	DWORD(HEX)	At start	00000000H to 00000021H	00000000H	This variable sets the functional options for MC_MoveAbsolute (Absolute Value Positioning).

■ Output variables

Output variable	Name	Data type	Default value	Description
Done	Execution completion	BOOL	FALSE	This variable becomes TRUE when the axis reaches the target position.
Busy	Executing	BOOL	FALSE	This variable becomes TRUE when MC_MoveAbsolute (Absolute Value Positioning) is being executed. This variable becomes FALSE after reaching the target position.
Active	Controlling	BOOL	FALSE	This variable becomes TRUE while MC_MoveAbsolute (Absolute Value Positioning) is controlling the axis. This variable becomes FALSE after the axis reaches the target position.
CommandAborted	Abortion of execution	BOOL	FALSE	This variable becomes TRUE when the execution of MC_MoveAbsolute (Absolute Value Positioning) is aborted. This variable becomes FALSE when Execute command (Execute) becomes FALSE.
Error	Error	BOOL	FALSE	This variable becomes TRUE when an error occurs.
ErrorID	Error code	WORD(UINT)	0	When an error occurs, this variable returns the error code. For details of error codes, refer to the following. MELSEC iQ-R Motion Module User's Manual (Application)

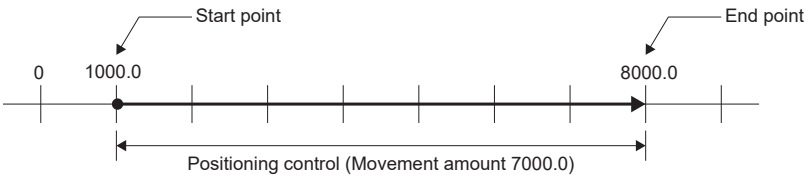
Processing details

- This FB sets Target position (Position), Velocity (Velocity), Acceleration (Acceleration), Deceleration (Deceleration), Jerk (Jerk), Direction selection (Direction), Buffer mode (BufferMode), and Options (Options), then executes positioning from the current position at start (start point position) to the specified position (end point position) set in Target position (Position).

Ex.

When the start point position (current stop position) is "1000.0" and Target position (Position) is set to "8000.0"

- Positioning is performed in the positive direction for the movement amount of "7000.0 (8000.0 - 1000.0)".



Multiple start (buffer mode)

Multiple motion control FBs can be continuously executed without stopping by executing the motion FB of another instance to the axis and the axes group subject to the motion control FB being executed.

Point

- "Multiple start" is to execute the motion FB of another instance when Axis status (AxisName.Md.AxisStatus) and Axes group status (AxesGroupName.Md.GroupStatus) are as follows.
  [Axis status (AxisName.Md.AxisStatus) where multiple start is available]
  - 3: During homing (Homing) (Only MC_Stop (Forced Stop) is possible)
  - 5: During positioning operation (DiscreteMotion)
  - 6: During continuous operation (ContinuousMotion)
  - 7: During synchronous operation (SynchronizedMotion)

  [Axes group status (AxesGroupName.Md.GroupStatus) where multiple start is available]
  - 5: Operating (GroupMoving)
- Multiple start of the single axis control FB cannot be executed to an axis operated in the axes group. It will cause "Motion FB Issue Error to the Axis during Axes Group Operating (error code: 1A7CH)".

■ Buffer mode type

The following lists the types of buffer mode and available buffer mode types differ depending on the FB.

Setting value	Buffer mode type	Description
0:mcAborting	Aborting	Aborts (cancels) the execution of the running FB and executes the next FB immediately.
1:mcBuffered	Buffered	Buffers the next FB on the running FB. If an FB is already buffered on the running FB, subsequent FBs are consecutively buffered. (Up to 2.) Buffering FBs are sequentially executed at the completion of the running FB.
2:mcBlendingLow	BlendingLow	Buffers the next FB on the running FB.*1 If an FB is already buffered on the running FB, subsequent FBs are consecutively buffered. (Up to 2.) Buffering FBs are sequentially executed after the axis is moved to the target position by the running FB. The lower target velocity between the running FB and buffering FB is used as the switching velocity.
3:mcBlendingPrevious	BlendingPrevious	Buffers the next FB on the running FB.*1 If an FB is already buffered on the running FB, subsequent FBs are consecutively buffered. (Up to 2.) Buffering FBs are sequentially executed after the axis is moved to the target position by the running FB. The target velocity of the running FB is used as the switching velocity.
4:mcBlendingNext	BlendingNext	Buffers the next FB on the running FB.*1 If an FB is already buffered on the running FB, subsequent FBs are consecutively buffered. (Up to 2.) Buffering FBs are sequentially executed after the axis is moved to the target position by the running FB. The switching speed changes to the target velocity of the buffering FB.
5:mcBlendingHigh	BlendingHigh	Buffers the next FB on the running FB.*1 If an FB is already buffered on the running FB, subsequent FBs are consecutively buffered. (Up to 2.) Buffering FBs are sequentially executed after the axis is moved to the target position by the running FB. The higher target velocity between the running FB and buffering FB is used as the switching velocity.

*1 In this mode, the FB that is running and the buffering FB are switched without stopping.

Point

- Up to two motion FBs can be buffered after multiple start in one axis and an axes group. If multiple start is executed when multiple start of two FBs has already been executed, "Warning starting over number of buffering FBs (warning code: 0D22H)" occurs and the analysis of the buffering FB will wait for the completion of the running FB. Each multiple start triggers the warning, however, it is possible to configure the filter setting not to detect the warnings. For details of the filter setting, refer to the following. MELSEC iQ-R Motion Module User's Manual (Application) When an error or a stop cause has occurred in the running FB, the FBs waiting for analysis are also aborted.
- When "Warning starting over number of buffering FBs (warning code: 0D22H)" occurs, do not execute multiple start until the running FB finishes. If the multiple FBs are waiting for analysis due to multiple start, the subsequent FBs may not be buffered in the order in which they are started.
- Since the FBs started by multiple start are immediately executed when Aborting has been specified, the FBs are not buffered. When the running FBs include a buffering FB, all buffering FBs will be aborted. However, as the FBs waiting for analysis are not aborted, they start after the completion of FBs started by multiple start with Aborting specified.
- When an error or a stop cause has occurred in the running FB, all buffering FBs are aborted. (The output of Abortion of execution (CommandAborted) becomes TRUE.)

Program example

Create a program that sets the buffer mode, executes relative positioning to move axis 1 continuously to the specified target position, and executes absolute positioning to move it to the home position after single axis continuous positioning control is finished.

■ Labels used

The following table lists the global and local labels used in this program.

Category	Label name	Data type	Class	Public label	Description
Global label	G_lePosition	Double-precision real number	VAR_GLOBAL	Enable	Target position Stores the positioning target position input from the GOT.
Local label	bContStartReq	Bit	VAR	—	Execution command of single axis continuous positioning Turns on when "M21" is tapped on the GOT.
	lePosition	Double-precision real number	VAR	—	Target position of absolute positioning
	leVelocity	Double-precision real number	VAR	—	Positioning velocity Stores the positioning velocity input from the GOT.
	leAcceleration	Double-precision real number	VAR	—	Acceleration Stores the acceleration based on Positioning velocity (leVelocity).
	leDeceleration	Double-precision real number	VAR	—	Deceleration Stores the acceleration based on Positioning velocity (leVelocity).
	leJerk	Double-precision real number	VAR	—	Jerk Stores the acceleration based on Positioning velocity (leVelocity).
	bContPosReq1	Bit	VAR	—	Execution request for relative positioning 1 and relative positioning 2
	bContPosReq2	Bit	VAR	—	Execution request for absolute positioning
	bMoveRela1Done	Bit	VAR	—	Relative positioning 1 execution complete
	bMoveRela1Active	Bit	VAR	—	Under control of relative positioning 1
	bMoveRela1Aborted	Bit	VAR	—	Relative positioning 1 execution aborted
	bMoveRela1Error	Bit	VAR	—	Relative positioning 1 error
	bMoveRela2Done	Bit	VAR	—	Relative positioning 2 execution complete
	bMoveRela2Active	Bit	VAR	—	Under control of relative positioning 2
	bMoveRela2Aborted	Bit	VAR	—	Relative positioning 2 execution aborted
	bMoveRela2Error	Bit	VAR	—	Relative positioning 2 error
	bMoveAbs1Done	Bit	VAR	—	Absolute positioning execution complete
	bMoveAbs1Active	Bit	VAR	—	Under control of absolute positioning
	bMoveAbs1Aborted	Bit	VAR	—	Absolute positioning execution aborted
	bMoveAbs1Error	Bit	VAR	—	Absolute positioning error

■ ENUM enumerators to be used

They are used in "Enumeration type name__Enumerator name" INT type global labels.

For details of ENUM enumerators, refer to the following.

☞ Page 47 ENUM Enumerator

Enumeration type name	Enumerator	Description
MC_BUFFER_MODE	mcBuffered	Buffers the next FB on the running FB. Buffering FBs are sequentially executed at the completion of the running FB.
MC_DIRECTION	mcShortestWay	Shortest path

Practice 4

Create a program that executes positioning control in the program block "ContinuousPositioning".

Select "MC_MoveRelative" and "MC_MoveAbsolute" from [Motion - Individual] under [Motion Control Function/Function Block] in "POU List" in the Element Selection window.

```
1  //------Continuous positioning control------
2
3  //Execution request (default value setting / continuous positioning control)
4  R_TRIG_1(
5      CLK:= G_bContPositionCMD ,
6      Q=> bContStartReq
7      );
8
9  //Target position and velocity setting for positioning control (velocity, acceleration, deceleration, and jerk)
10 IF bContStartReq THEN
11     lePosition     := 0.0;                                    (1)
12     leVelocity     := G_leSetVelocity;
13     leAcceleration := G_leSetVelocity * 2.0;
14     leDeceleration := G_leSetVelocity * 2.0;
15     leJerk         := G_leSetVelocity * 4.0;
16     bContPosReq1 := TRUE;
17     ELSE
18     bContPosReq1 := FALSE;
19 END_IF;
20
21 //Relative positioning 1 control execution
22
23 MC_MoveRelative_1(
24     Axis:=  Axis0001.AxisRef,                                 (2)
25     Execute:= bContPosReq1 ,                                  (3)
26     Distance:= G_lePosition ,                                 (4)
27     Velocity:= leVelocity,                                    (5)
28     Acceleration:=  leAcceleration,                           (6)
29     Deceleration:=  leDeceleration,                           (7)
30     Jerk:=  leJerk,                                           (8)
31     Done=> bMoveRela1Done ,                                   (9)
32     Active=> bMoveRela1Active ,                               (10)
33     CommandAborted=> bMoveRela1Aborted ,                      (11)
34     Error=> bMoveRela1Error                                   (12)
35     );
36
```

When the positioning speed of axis 1 is changed, acceleration, deceleration, and jerk are recalculated according to the speed.

No.	Description
(1)	Sets the target position of absolute positioning 1 (axis 1).
(2)	Sets the axis information of axis 1.
(3)	Sets the command to perform relative positioning of axis 1 under single axis continuous positioning control.
(4)	Sets the target position of relative positioning 1 (axis 1).
(5)	Sets the velocity of relative positioning 1 (axis 1).
(6)	Sets the acceleration of relative positioning 1 (axis 1).
(7)	Sets the deceleration of relative positioning 1 (axis 1).
(8)	Sets the jerk of relative positioning 1 (axis 1).
(9)	Stores "Execution completion" of the relative positioning 1 (axis 1) FB.
(10)	Stores "Controlling" of the relative positioning 1 (axis 1) FB.
(11)	Stores "Abortion of execution" of the relative positioning 1 (axis 1) FB.
(12)	Stores errors of the relative positioning 1 (axis 1) FB.

```
//Relative positioning 2 control execution

MC_MoveRelative_2(
    Axis:= Axis0001.AxisRef ,                                    (1)
    Execute:= bContPosReq1 ,                                     (2)
    Distance:= G_lePosition*2.0 ,                                (3)
    Velocity:= leVelocity/2.0 ,                                  (4)
    Acceleration:= leAcceleration ,                              (5)
    Deceleration:= leDeceleration,                               (6)
    Jerk:=  leJerk,                                              (7)
    BufferMode:= MC_BUFFER_MODE__mcBuffered ,                    (8)
    Done=> bMoveRela2Done ,                                      (9)
    Active=> bMoveRela2Active ,                                  (10)
    CommandAborted=> bMoveRela2Aborted ,                         (11)
    Error=> bMoveRela2Error                                      (12)
    );

```

No.	Description
(1)	Sets the axis information of axis 1.
(2)	Sets the command to perform relative positioning of axis 1 under single axis continuous positioning control.
(3)	Sets the target position of relative positioning 2 (axis 1). In this program, the axis is positioned at twice the distance of the target position setting input from the GOT.
(4)	Sets the velocity of relative positioning 2 (axis 1). In this program, positioning is performed at 0.5 times as fast as the positioning velocity input from the GOT.
(5)	Sets the acceleration of relative positioning 2 (axis 1).
(6)	Sets the deceleration of relative positioning 2 (axis 1).
(7)	Sets the jerk of relative positioning 2 (axis 1).
(8)	Sets the buffer mode to continuously execute the relative positioning 1 (axis 1) FB and relative positioning 2 (axis 1) FB. "FB for relative positioning 2 (MC_MoveRelative_2)" is buttered on "FB for relative positioning 1 (MC_MoveRelative_1)" that is running. "FB for relative positioning 2 (MC_MoveRelative_2)" waits for the completion of the running FB. The buffered FB is executed after the completion of the running FB.
(9)	Stores "Execution completion" of the relative positioning 2 (axis 1) FB.
(10)	Stores "Controlling" of the relative positioning 2 (axis 1) FB.
(11)	Stores "Abortion of execution" of the relative positioning 2 (axis 1) FB.
(12)	Stores errors (axis 1).

```
54  //Set dwell time
55      SET(bMoveRela2Done,bDwell_In);
56      TON_1(IN:= bDwell_In ,PT:= T#2000ms ,Q=>  bContPosReq2);
57
58  //Absolute positioning 1 control execution
59
60  MC_MoveAbsolute_1(
61      Axis         := Axis0001.AxisRef ,
62      Execute      := bContPosReq2 ,
63      Position     := lePosition ,
64      Velocity     := leVelocity,
65      Acceleration:= leAcceleration ,
66      Deceleration:= leDeceleration ,
67      Jerk         := leJerk ,
68      Direction    := MC_DIRECTION__mcShortestWay ,
69      Done             => bMoveAbs1Done ,
70      Active           => bMoveAbs1Active ,
71      CommandAborted   => bMoveAbs1Aborted ,
72      Error            => bMoveAbs1Error
73      );
74
75  //Notify positioning control interruption
76  bAborted := bMoveRela1Aborted OR bMoveRela2Aborted OR bMoveAbs1Aborted;
77
78  //Notify error occurrence in positioning control
79  G_bContPositionError := bMoveRela1Error OR bMoveRela2Error OR bMoveAbs1Error;
80
81  //Notify the PLC of positioning control execution completion
82  G_bContPosDone := bMoveAbs1Done OR G_bContPositionError OR bAborted;
83
84  //Reset Dwell_In at completion of positioning control execution
85  RST(bMoveAbs1Done,bDwell_In);
```

Sets the time interval between the end of single-axis continuous positioning control of axis 1 and the start of the next positioning control.

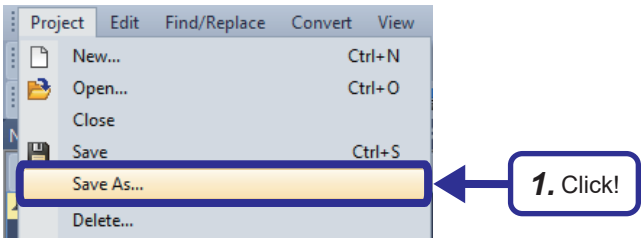
(1) (2) (3) (4) (5) (6) (7) (8) (9) (10) (11) (12)

No.	Description
(1)	Sets the axis information of axis 1.
(2)	Sets the command to perform absolute positioning of axis 1.
(3)	Sets the target position of absolute positioning (axis 1).
(4)	Sets the velocity of absolute positioning (axis 1).
(5)	Sets the acceleration of absolute positioning (axis 1).
(6)	Sets the deceleration of absolute positioning (axis 1).
(7)	Sets the jerk of absolute positioning (axis 1).
(8)	Sets the shortest path to perform positioning in the direction of the shortest path to the target position from the current position.
(9)	Stores "Execution completion" of the relative positioning 2 (axis 1) FB.
(10)	Stores "Controlling" of the relative positioning 2 (axis 1) FB.
(11)	Stores "Abortion of execution" of the relative positioning 2 (axis 1) FB.
(12)	Stores errors (axis 1).

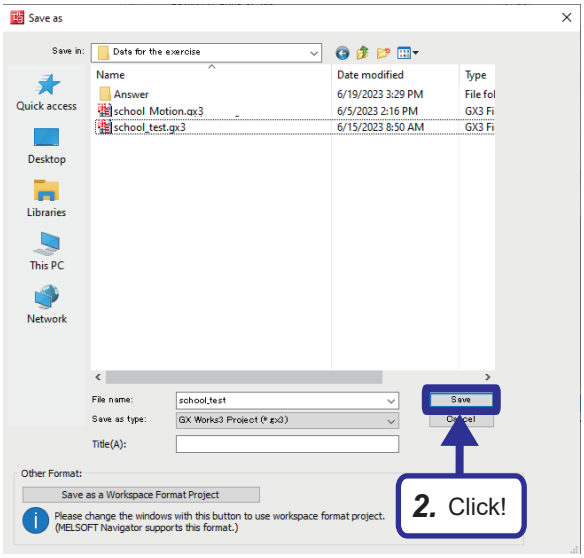
Saving the project

After creating the program, write it to the programmable controller by following the same procedure as "☞ Page 110 Writing to the programmable controller" to save the created project.

Operating procedure



1. Click [Project] ⇨[Save As] from the menu of GX Works3.

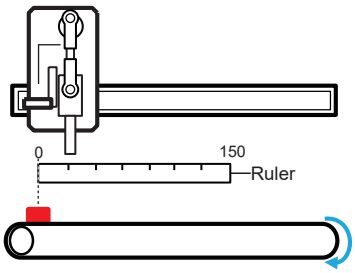


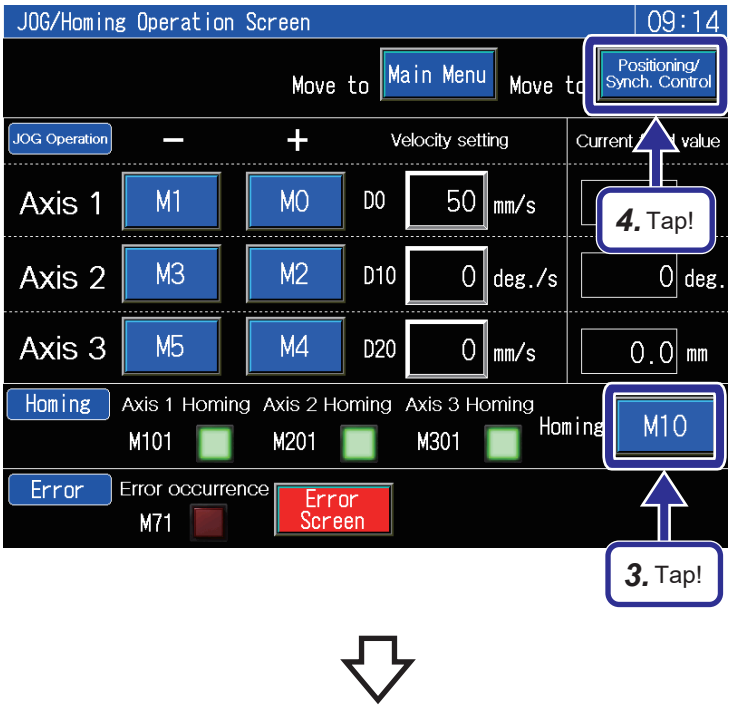
2. Enter the file name, and click the [Save] button.

Operation check

Check the operation of single axis continuous positioning control.

Operating procedure





1. Turn on the servo.

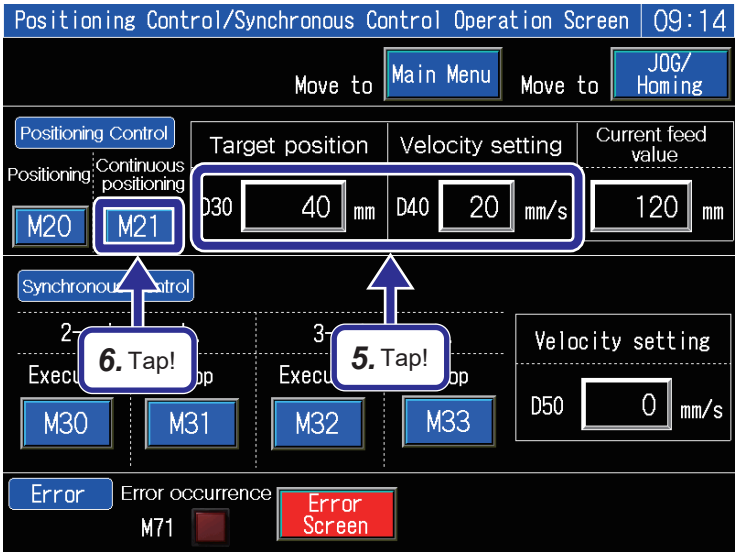
2. Perform JOG operation of axis 1 on the JOG/ Homing Operation Screen to move the red workpiece to the point near 0 on the ruler.

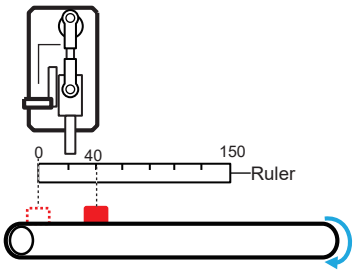
3. Tap the [M10] button for homing.

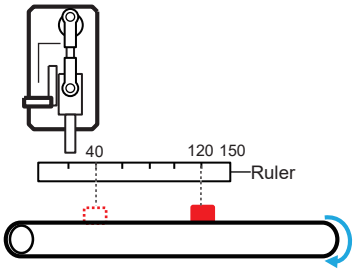
Point

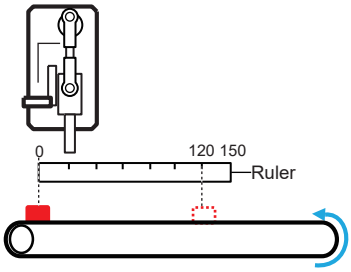
Always perform homing before executing single axis continuous positioning control.

4. Tap the [Positioning/Synch. Control] button.









5. Set the desired values for the target position "D30" and velocity setting "D40".

Point

If you cannot see how continuous positioning control works, decrease the setting velocity.

6. Tap the [M21] button for continuous positioning control.

7. When single axis continuous positioning control starts, the red workpiece on axis 1 is moved to the target position at the set velocity by relative positioning 1 control.

8. Continuously, relative positioning 2 control is executed. The workpiece is positioned at twice the distance of the set target position from the current stop position at 0.5 times the set velocity.

This is the operation of single axis continuous positioning control.

9. Two seconds after completion of single axis continuous positioning control, the workpiece is moved to the home position by absolute positioning.

7.5 Troubleshooting

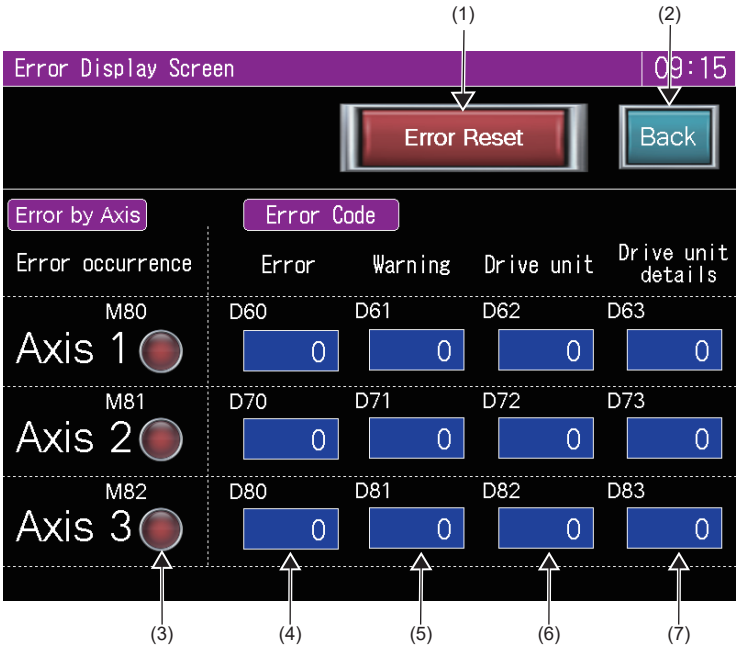
This section provides troubleshooting information on the Motion module.

If the module does not operate, check the following.

Check item	Action
Is the servo amplifier turned on (all axes servo ON)?	If the servo amplifier is not turned on, tap the [Servo ON M60] button on the menu.
Is the CPU module RUN?	If it is not RUN, set the RUN/STOP/RESET switch to "RUN".
Are there any missing parameters in the programmable controller?	Refer to ☞ Page 110 Writing to the programmable controller, write the parameter again.
Is there any error (ERR LED of RD78G turned on)?	If any error has occurred, refer to the following to clear the error. ☞ Page 149 Checking errors
Is the error occurrence lamp "M71" turned on?	

Checking errors

The Error Display Screen shows the error axis and error code.

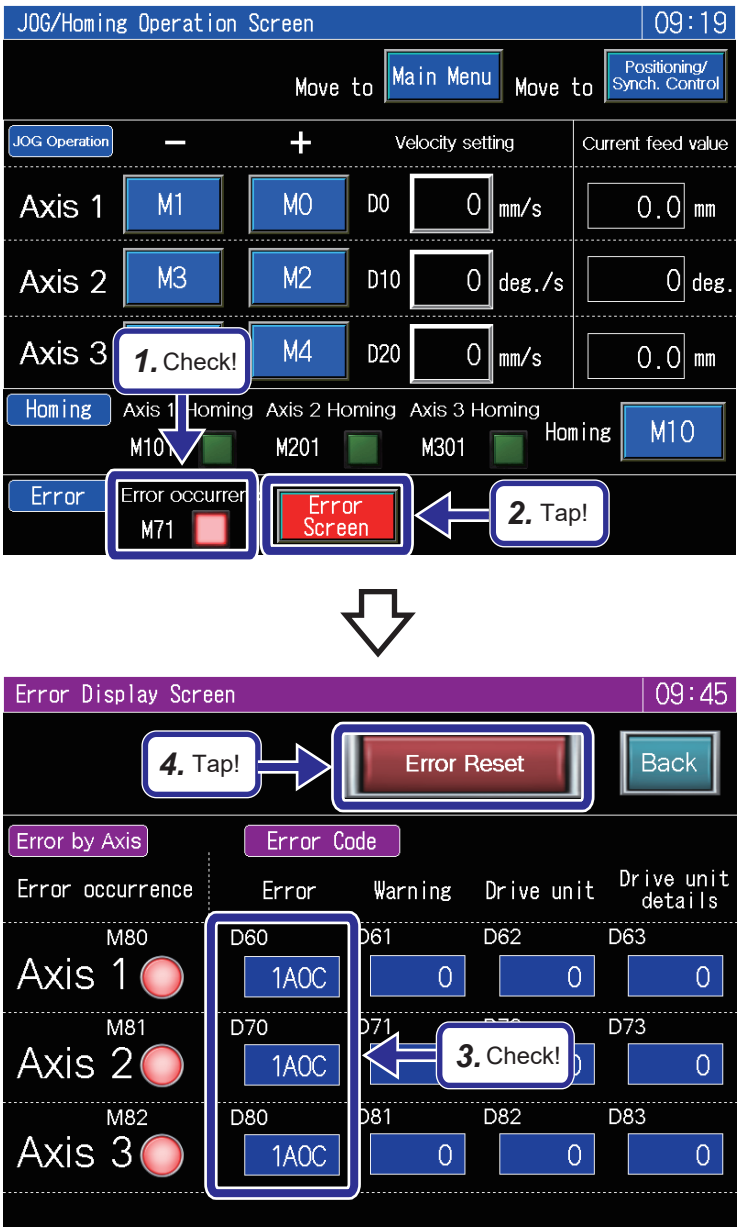


Displayed items

No.	Name	Device	Description
(1)	Error Reset	—	Resets the occurring error.
(2)	Back	—	Returns to the previous screen.
(3)	Error occurrence	M80, M81, M82	Turns on if any error occurs in each axis.
(4)	Error	D60, D70, D80	Displays the axis error code.
(5)	Warning	D61, D71, D81	Displays the axis warning code.
(6)	Drive unit	D62, D72, D82	Displays the alarm number and warning number of the servo amplifier. For example, "098" is displayed when "AL. 098.1 Forward rotation-side software stroke limit reached" has occurred.
(7)	Drive unit details	D63, D73, D83	Displays the detailed numbers of the alarm and warning of the servo amplifier. For example, "1" is displayed when "AL. 098.1 Forward rotation-side software stroke limit reached" has occurred.

The following describes how to check the occurring errors.

Operating procedure



1. Check that the error occurrence lamp "M71" is turned on.
2. Tap the [Error Screen] button.
3. Check the error code in the Error Display Screen, and eliminate the error.

For error codes, refer to ☞ Page 150 Error code.

4. Tap the [Error Reset] button to clear the error.

■ Error code

The following table lists the error codes frequently generated during operation of the demonstration machine.

For other error codes, refer to the following manual.

📖MELSEC iQ-R Motion Module User's Manual (Application)

📖MR-J5 User's Manual (Troubleshooting)

Error code	Error name	Error details	Action
1A0C	Acceleration/ Deceleration 0 Specified Operation Error at Start	Occurs when JOG operation, positioning control, or synchronous control is executed without changing the velocity from "0".	Set the velocity.
1A22	Start at Homing Incomplete	Occurs when positioning control is executed without homing completed.	Execute homing.
1A2F	FLS Signal Detection (Controlling)	Occurs when the axis reaches the + side limit during control.	Move the error axis to the position within the range of control.
1A30	RLS Signal Detection (Controlling)	Occurs when the axis reaches the - side limit during control.	

8 EXERCISE 3 SYNCHRONOUS CONTROL

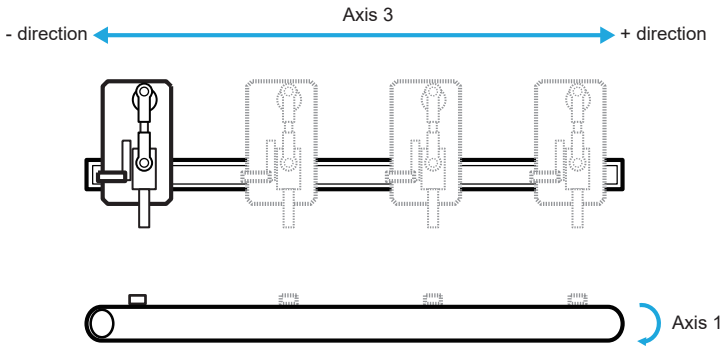
This chapter describes synchronous control.
Synchronous control can be achieved using software instead of using mechanical components such as gears, shafts, speed change gears and cams.

8.1 Exercise

Create a motion program to perform the following operation.

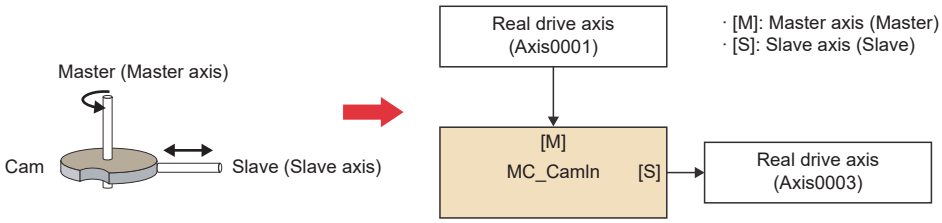
2-axis synchronous control

Perform 2-axis synchronous control of axis 1 and axis 3.
Axis 1 conveys the workpiece at a constant speed, and when the sensor detects a workpiece, axis 3 moves horizontally in sync with axis 1.
Once axis 3 moves 10 cm horizontally from the home position, it returns to the home position and then moves in sync with axis 1.



■ Axis configuration

A cam table (cam) is used to synchronize the motion of the real drive axis 1 (input axis) and real drive axis 3 (output axis).
The FB (MC_CamIn) can be used to replace mechanical cams with electronic cams for synchronous control.
A mechanical cam is converted to the FB as follows.



The following table lists the axes that can be specified as Master and Slave in the single-axis synchronous control FB.

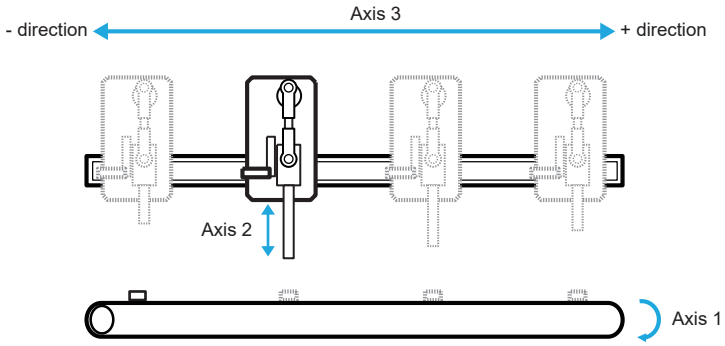
Axis type		Master	Slave	Remarks
Real axis	Real drive axis	○	○	When control by multiple motion FBs is required, configure the system that executes an FB for each virtual linked axis and transmits the result (command) to the real drive axis.
	Real encoder axis	○	×	It is used as a master axis (Master). If it is used as a slave axis (Slave), "Necessary Slave Object Unset (error code: 1AA8H)" occurs and it does not start.
Virtual axis	Virtual drive axis	○	○	It can generate commands mainly through positioning control.
	Virtual encoder axis	○	×	It is used as a master axis (Master). If it is used as a slave axis (Slave), "Necessary Slave Object Unset (error code: 1AA8H)" occurs and it does not start.
	Virtual linked axis	○	○	It is used as an intermediate axis to transmit a command to the real drive axis. Assign virtual linked axes to use multiple motion FBs such as for gears.

3-axis synchronous control

Perform 3-axis synchronous control between axis 1 and axis 2 and between axis 1 and axis 3.

Axis 1 conveys the workpiece at a constant speed, and when the sensor detects a workpiece, axis 2 moves vertically and axis 3 moves horizontally in sync with axis 1.

After the synchronous control is done, it is started again when the sensor detects a workpiece.

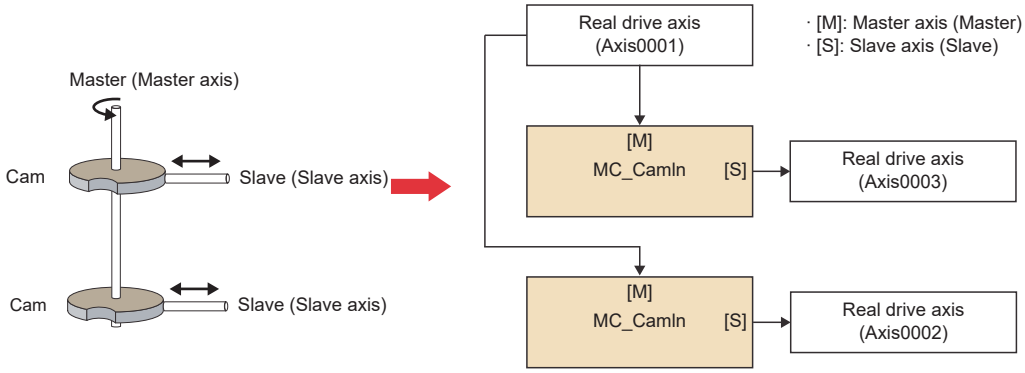


■ Axis configuration

A cam table (cam) is used to synchronize the motion of the real drive axis 3 (output axis) and real drive axis 2 (output axis) with the real drive axis 1 (input axis).

The FB (MC_CamIn) can be used to replace mechanical cams with electronic cams for synchronous control.

A mechanical cam is converted to the FB as follows.



8.2 2-Axis Synchronous Control

Cam data setting

This section describes how to create operation profile data (cam data).
Operation profile data is a generic term for the waveform data used for control.
To perform cam operation, create operation profile data (cam data) that serves as a cam pattern in advance.

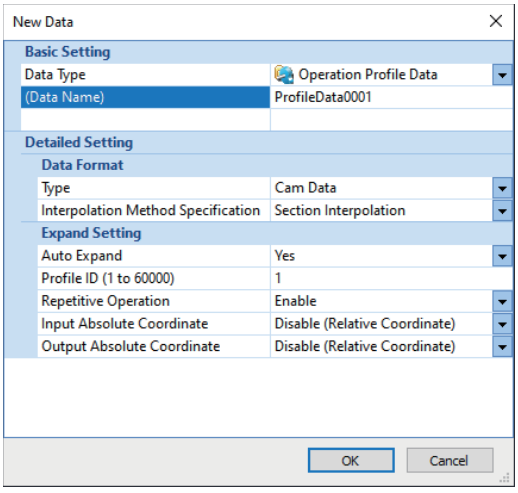
Creating new operation profile data

Create new operation profile data using the motion control setting function.
When operation profile data is created with the motion control setting function, a profile data type global label whose name is the operation profile data name is automatically added and available in the program. The settings (storage location and ID) are used as the default values for the profile data type.

Point

The profile data type label created with the motion control setting function cannot be edited on the global label editor. Be sure to create and edit it on the operation profile data create window.

Window



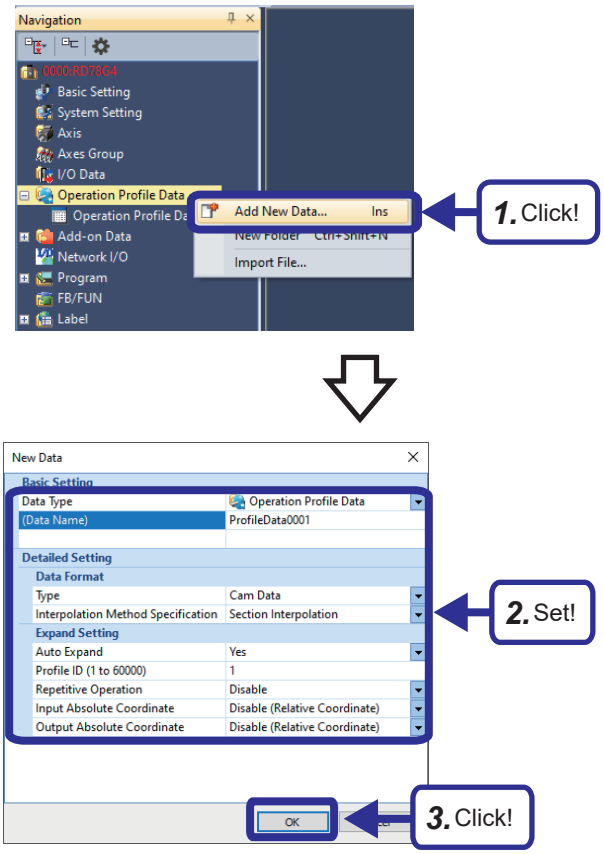
Displayed items

Item	Description
Data Name	Enter the operation profile data name.
Type	Select the operation profile data type.
Interpolation Method Specification	Specify the interpolation method of data. • Linear Interpolation: Displays the data of a cam curve for one cycle defined with two or more points in the open area. • Section Interpolation: Displays the stroke data defined by interpolating the data with specified curves and dividing a cam curve for one cycle by the number of points of the cam resolution in the open area. • Spline Interpolation: Displays the stroke data defined by interpolating that data with splines and dividing a cam curve for one cycle by the number of points of the cam resolution in the open area.
Auto Expand	Set whether to automatically open the operation profile data at power ON/programmable controller ON. • Yes: The operation profile data is automatically opened in the motion system at power ON/PLC READY [Y0] ON. • No: The operation profile data expand FB must be executed to use the operation profile data.
Profile ID	Set the profile ID.
Repetitive Operation	Set the repetitive mode of the open instruction. • Disable: Stops operation at the end point of the operation profile data. • Enable: Repeats the execution of the operation profile data.
Input Absolute Coordinate	Set the input absolute coordinate of the open instruction.

Item	Description
Output Absolute Coordinate	Set the output absolute coordinate of the open instruction. • Disable (Relative Coordinate): Calculates the output value based on the current value when the execution of the operation profile data (cam data) is started. Select this option to perform the feed cam operation. • Enable (Absolute Coordinate): Calculates the output value at the start of the operation profile data (cam data) execution so that it is always the start point at the start of each cycle of the operation profile data. When the start point and end point of the operation profile data are different in the repetitive operation, a command is output in every operation cycle to return to the first output value at the start of the next cycle.

The following shows how to create new operation profile data with the motion control setting function.

Operating procedure



1. In the Navigation window of the motion control setting function, right-click [Operation Profile Data] and click [Add New Data].

2. Configure the Basic Setting and Detailed Setting as follows.

[Setting details]

Data Type: Operation Profile Data (Default)

(Data Name): ProfileData0001 (Default)

Type: Cam Data (Default)

Interpolation Method Specification: Section Interpolation (Default)

Auto Expand: Yes (Default)

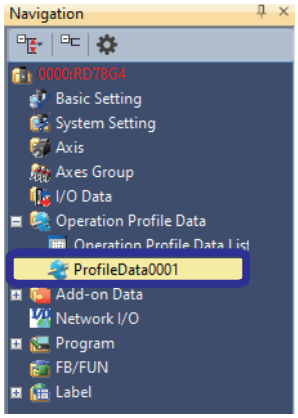
Profile ID: 1 (Default)

Repetitive Operation: Disable

Input Absolute Coordinate: Disable (Relative Coordinate) (Default)

Output Absolute Coordinate: Disable (Relative Coordinate) (Default)

3. Click the [OK] button.



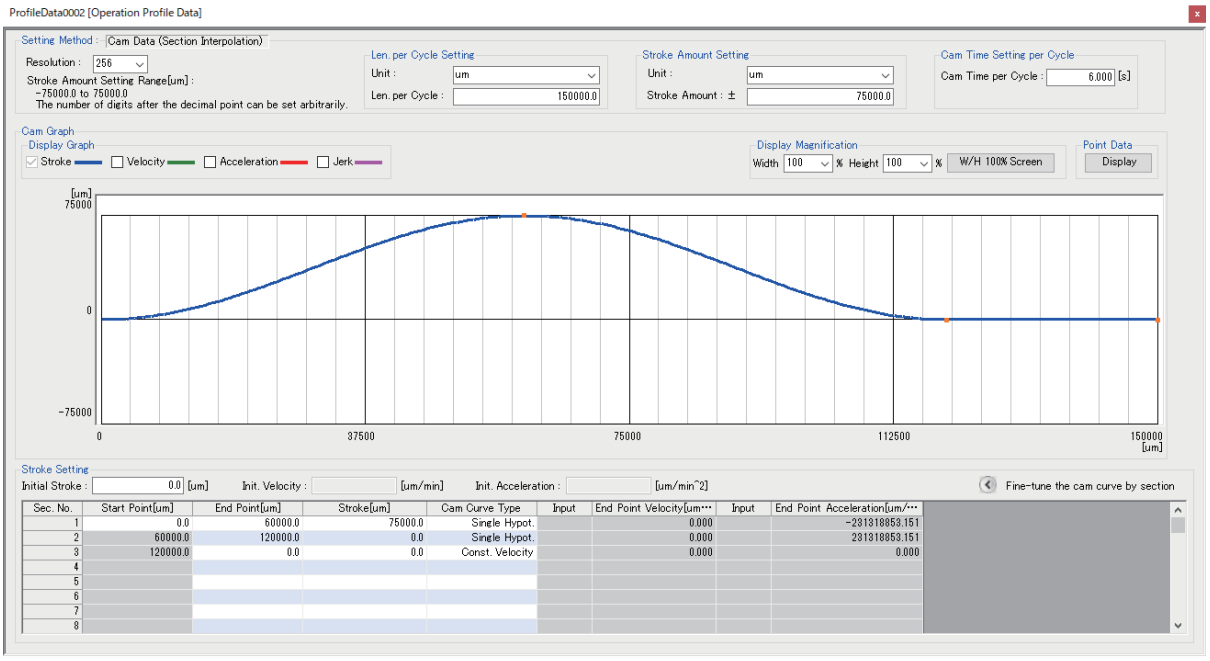
4. The operation profile data is added to the Navigation window.

Creating cam data

Configure the waveform setting of the operation profile data.

Cam data defines the general relationship between input values and output values.

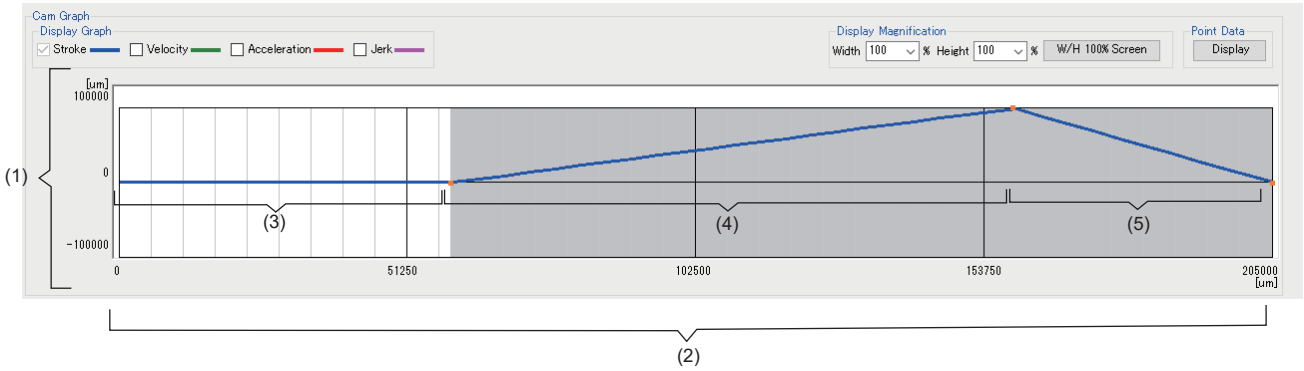
Window



Displayed items

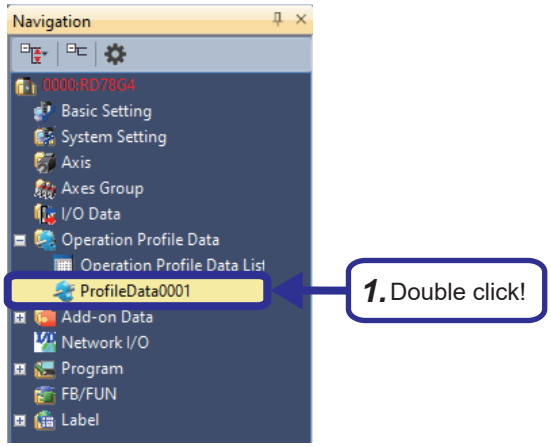
Item		Description
Setting Method	Resolution	Select the resolution of the cam data.
	Len. per Cycle Setting	Set the length per cycle and its unit. (Set the travel distance of the master axis required for one rotation of the cam.)
	Stroke Amount Setting	Set the stroke amount and its unit. (Set the maximum distance through which the slave axis moves during one rotation of the cam.)
	Cam Time Setting per Cycle	Set the time required for one cycle of the cam. It is used for calculating the velocity, acceleration, and jerk.
Cam Graph	Display Graph	Select the type of the cam graph to be displayed.
	Display Magnification	Set the display magnification for the height and width of the cam graph.
	[W/H 100% Screen] button	Adjusts the display magnification for the height and width of the cam graph to 100%.
	Point Data	Click the [Display] button to display the list of data points in the cam curve.
Stroke Setting	Initial Stroke	Displays the stroke amount at the start point.
	Init. Velocity	Set the initial velocity.
	Init. Acceleration	Set the initial acceleration of Section No. 1 in the stroke data.
	Sec. No.	Displays Section Nos. for which the stroke data is set.
	Start Point	Set the start point of the stroke data.
	End Point	Set the end point of the stroke data.
	Stroke	Set the stroke position.
	Cam Curve Type	Set the curve type of the cam data.
	Input	Input for the end point velocity. Set whether to automatically set the end point velocity to the calculated initial velocity for the next section.
	End Point Velocity	Set the end point velocity of the stroke data.
	Input	Input for the end point acceleration. Set whether to automatically set the end point acceleration to the calculated initial acceleration for the next section.
	End Point Acceleration	Set the end point acceleration of the stroke data.

The following shows the waveform of the cam data to be created.

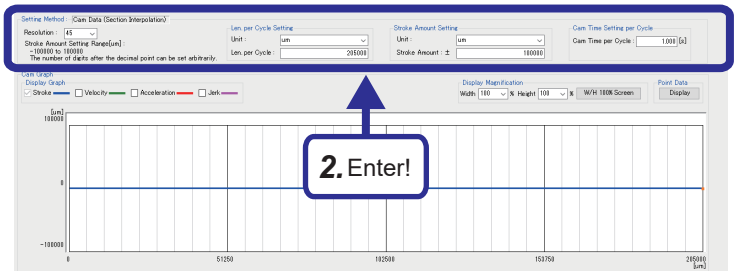


No.	Description
(1)	Indicates the movement amount of the slave axis (axis 3).
(2)	Indicates the movement amount of the master axis (axis 1).
(3)	Indicates the movement amount to the home position of axis 3 after the sensor detects a workpiece on axis 1. Although the movement amount of axis 1 increases because it is operating at a constant velocity, the movement amount of axis 3 does not change because it is stopped.
(4)	Indicates the travel distance of axis 3 that follows the workpiece on axis 1. The movement amount of axis 3 increases along with axis 1.
(5)	Indicates the travel distance of axis 3 to return to the home position. The movement amount of axis 3 decreases (moves in the reverse direction) because it returns to the home position.

Operating procedure



1. In the Navigation window of the motion control setting function, double-click [ProfileData0001] under [Operation Profile Data].



2. Set each item as follows.

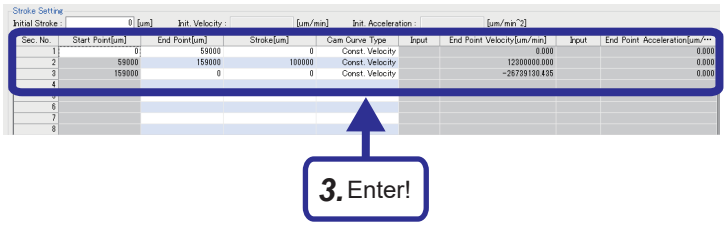
[Setting details]

Resolution: 45

Unit: um (Manual input), Len. per Cycle Setting: 205000

Unit: um (Manual input), Stroke Amount: 100000

Cam Time per Cycle: 1.000 (Default)

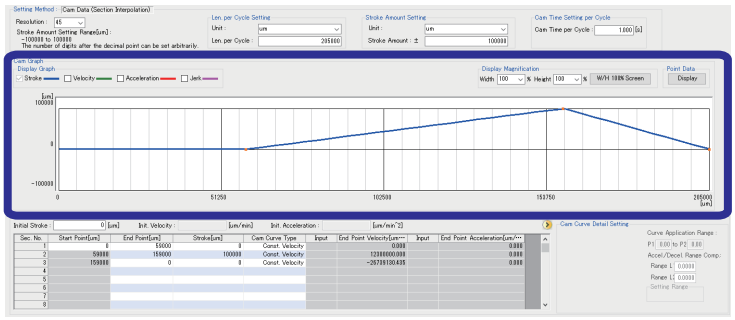


3. Configure the stroke settings as follows.

Sec. No.	Start Point	End Point	Stroke	Cam Curve Type	Description
1	0	59000	0	Const. Velocity	Set the distance from the sensor to the dog sensor of axis 3.
2	59000	159000	100000	Const. Velocity	Set the travel distance of axis 3 that follows the workpiece on axis 1 from the dog sensor of axis 3.
3	159000	0	0	Const. Velocity	Set the distance from axis 3 to the sensor.

Point

Cam data can also be created by dragging the cursor to a point of the stroke curve in the cam graph. The changed point is reflected in the stroke setting field.



4. The waveform as shown on the left is created.

2-axis synchronous control program

Create the following program: Axis 1 starts conveying a workpiece at a constant speed, and when the sensor detects the workpiece, axis 3 is synchronized with axis 1 according to the cam pattern created as the operation profile data.

FB to be used

The following table lists the FB used in the 2-axis synchronous control program.

Create the 2-axis synchronous control program by combining the FB that controls the velocity of axis 1 (MC_MoveVelocity), FB that executes the cam operation of axis 1 and axis 3 according to the created cam data (MC_CamIn), and FB that stops the synchronous control of axis 1 and axis 3 (MC_Stop).

Type	FB	Description
Motion	MC_MoveVelocity	Switches the driver to csv and controls the velocity control according to the specified velocity.
	MC_CamIn	Executes cam operation.
	MC_Stop	Decelerates the specified axis to stop. This FB is used to stop synchronization.

FB name: MC_MoveVelocity

Switches the driver to csv and controls the velocity control according to the specified velocity.

The following shows the details of MC_MoveVelocity.

```
MC_MoveVelocity(
  Axis:= ?AXIS_REF? ,
  Execute:= ?BOOL? ,
  ContinuousUpdate:= ?BOOL? ,
  Velocity:= ?LREAL? ,
  Acceleration:= ?LREAL? ,
  Deceleration:= ?LREAL? ,
  Jerk:= ?LREAL? ,
  Direction:= ?INT? ,
  BufferMode:= ?INT? ,
  Options:= ?DWORD? ,
  InVelocity=> ?BOOL? ,
  Busy=> ?BOOL? ,
  Active=> ?BOOL? ,
  CommandAborted=> ?BOOL? ,
  Error=> ?BOOL? ,
  ErrorID=> ?WORD?
);
```

Name	Number of input area points (byte)	Number of output area points (byte)	Compilation method	FB operation
Velocity Control	56	8	Subroutine type	Real-time execution

Setting data

■ I/O variable

I/O variable	Name	Data type	Input import	Setting range	Default value	Description
Axis	Axis information	AXIS_REF	At start	—	Mandatory	This variable sets the axis. For the variables used (AxisName.AxisRef.), refer to the following. ☞ Page 45 AxisName.AxisRef. (Axis information)

■ Input variables

Input variable	Name	Data type	Import	Setting range	Default value	Description
Execute	Execute command	BOOL	At start	TRUE, FALSE	FALSE	When this variable is TRUE, MC_MoveVelocity (Speed Control) is executed.

Input variable	Name	Data type	Import	Setting range	Default value	Description
ContinuousUpdate	Continuous update	BOOL	At start	TRUE, FALSE	FALSE	This variable sets whether to enable or disable continuous change of Velocity (Velocity), Acceleration (Acceleration), and Deceleration (Deceleration). • FALSE:Disable • TRUE:Enable
Velocity	Velocity	LREAL	At start / Retrigger possible / Continuous update possible	0.0, ±0.0001 to ±2500000000.0	0.0	This variable sets the command velocity. When the velocity is negative, the axis moves in the negative direction. When "0.0" is set, the axis does not operate, but Axis status (AxisName.Md.AxisStatus) changes to "6: During continuous operation (ContinuousMotion)".
Acceleration	Acceleration	LREAL	At start / Retrigger possible / Continuous update possible	0.0000, 0.0001 to 2147483647.0	0.0	This variable sets the acceleration.
Deceleration	Deceleration	LREAL	At start / Retrigger possible / Continuous update possible	0.0000, 0.0001 to 2147483647.0	0.0	This variable sets the deceleration.
Jerk	Jerk	LREAL	At start	0.0000, 0.0001 to 2147483647.0	0.0	This variable sets the jerk.
Direction	Direction selection	INT (MC_DIRECTION)	At start	1, 2	0	This variable sets the selected direction. • 1:Positive direction (mcPositiveDirection) • 2:Negative direction (mcNegativeDirection) If "2: Negative direction (mcNegativeDirection)" is set when Velocity (Velocity) is negative, the motor moves in the positive direction. When this setting is omitted, "Out of Direction Selection Range (error code: 1A37H)" occurs.
BufferMode	Buffer mode	INT (MC_BUFFER_MODE)	At start	0, 1	0	This variable sets the buffer mode. • 0:Aborting (mcAborting) • 1:Buffered (mcBuffered)
Options	Options	DWORD(HEX)	At start	00000000H to 0002001H	00000000H	This variable sets the functional options for MC_MoveVelocity (Speed Control) in terms of bits.

■ Output variables

Output variable	Name	Data type	Default value	Description
InVelocity	Target velocity reached	BOOL	FALSE	This variable becomes TRUE when the command velocity calculated by the motion system reaches the target velocity. The value is retained until Execute command (Execute) becomes FALSE or the control is aborted. When the target velocity has been changed due to a change while Continuous update (ContinuousUpdate) is TRUE, this variable remains FALSE until the velocity reaches the target velocity after change.
Busy	Executing	BOOL	FALSE	This variable becomes TRUE when MC_MoveVelocity (Speed Control) is executed.
Active	Controlling	BOOL	FALSE	This variable becomes TRUE while MC_MoveVelocity (Speed Control) is controlling the axis.
CommandAborted	Abortion of execution	BOOL	FALSE	This variable becomes TRUE when execution of MC_MoveVelocity (Speed Control) is aborted. This variable becomes FALSE when Execute command (Execute) becomes FALSE.

Output variable	Name	Data type	Default value	Description
Error	Error	BOOL	FALSE	This variable becomes TRUE when an error occurs.
ErrorID	Error code	WORD(UINT)	0	When an error occurs, this variable returns the error code. For details of error codes, refer to the following. 📖MELSEC iQ-R Motion Module User's Manual (Application)

Processing details

- MC_MoveVelocity (Speed Control) switches the control mode of the driver to csv and performs control. This function controls the command velocity based on the set Acceleration (Acceleration), Deceleration (Deceleration), and Jerk (Jerk). To stop MC_MoveVelocity (Speed Control), start MC_Stop (Forced Stop).

Program example

Create a velocity control program to operate an axis at a constant velocity.

■ Labels used

The following table lists the global and local labels used in this program.

Category	Label name	Data type	Class	Public label	Description
Local label	bSyncReq_Demo1	Bit	VAR	—	2-axis synchronous control execution request
	leVelocity	Double-precision real number	VAR	—	Axis 1 velocity Stores the velocity setting for synchronous control input from the GOT.
	leAcceleration	Double-precision real number	VAR	—	Axis 1 acceleration Stores the acceleration based on Axis 1 velocity (leVelocity).
	leDeceleration	Double-precision real number	VAR	—	Axis 1 deceleration Stores the deceleration based on Axis 1 velocity (leVelocity).
	leJerk	Double-precision real number	VAR	—	Axis 1 jerk Stores the jerk based on Axis 1 velocity (leVelocity).
	bMove1InVelo	Bit	VAR	—	Axis 1 target velocity reached
	bMove1Aborted	Bit	VAR	—	Axis 1 execution aborted
	bMove1Error	Bit	VAR	—	Axis 1 error

■ ENUM enumerator to be used

It is used in "Enumeration type name__Enumerator name" INT type global labels.

For details of ENUM enumerators, refer to the following.

☞ Page 47 ENUM Enumerator

Enumeration type name	Enumerator	Description
MC_DIRECTION	mcPositiveDirection	Positive direction

Practice 5-1

Create the "1-axis: Velocity control execution" program in the program block "SynchronousDemo1".
Select "MC_MoveVelocity" from [Motion - Individual] under [Motion Control Function/Function Block] in "POU List" in the Element Selection window.

```
1  //------2-axis synchronous control------
2  //Synchronous control execution request
3  IF  (G_bSyncCMD_Demo1 AND NOT G_bSyncStop_Demo1)  THEN
4      bSyncReq_Demo1 := TRUE;
5      ELSE
6      bSyncReq_Demo1 := FALSE;
7  END_IF;
8
9  //Velocity setting for synchronous control (velocity, acceleration, deceleration, and jerk)
10 IF (G_leSyncSetVelocity <> leVelocity) THEN
11     leVelocity     := G_leSyncSetVelocity;
12     leAcceleration := G_leSyncSetVelocity * 2.0;
13     leDeceleration := G_leSyncSetVelocity * 2.0;
14     leJerk         := G_leSyncSetVelocity * 4.0;
15 END_IF;
16
17 //1-axis: Velocity control execution
18 MC_MoveVelocity_1(
19     Axis:= Axis0001.AxisRef ,
20     Execute:= bSyncReq_Demo1 ,
21     Velocity:= leVelocity ,
22     Acceleration:= leAcceleration ,
23     Deceleration:= leDeceleration ,
24     Jerk:= leJerk ,
25     Direction:= MC_DIRECTION_mcPositiveDirection ,
26     InVelocity=> bMove1InVelo ,
27     CommandAborted=> bMove1Aborted ,
28     Error=> bMove1Error
29     );
30
```

When the 2-axis synchronous control request is turned on, the synchronous control command turns on.

When the synchronous control speed is changed, acceleration, deceleration, and jerk are recalculated according to the speed.

(1) (2) (3) (4) (5) (6) (7) (8) (9) (10)

No.	Description
(1)	Sets the axis information of axis 1.
(2)	Sets the execution request for 2-axis synchronous control (axis 1 and axis 3).
(3)	Sets the velocity of the velocity control (axis 1).
(4)	Sets the acceleration of the velocity control (axis 1).
(5)	Sets the deceleration of the velocity control (axis 1).
(6)	Sets the jerk of the velocity control (axis 1).
(7)	Sets the control direction of the velocity control (axis 1) to the positive direction.
(8)	Stores "Target velocity reached" of the velocity control (axis 1) FB.
(9)	Stores "Abortion of execution" of the velocity control (axis 1) FB.
(10)	Stores errors (axis 1).

FB name: MC_CamIn

This FB starts the cam operation based on the specified cam data.

The following shows the details of MC_CamIn.

```
MC_CamIn(
  Master:= ?AXIS_REF? ,
  Slave:= ?AXIS_REF? ,
  Execute:= ?BOOL? ,
  ContinuousUpdate:= ?BOOL? ,
  MasterOffset:= ?LREAL? ,
  SlaveOffset:= ?LREAL? ,
  MasterScaling:= ?LREAL? ,
  SlaveScaling:= ?LREAL? ,
  MasterStartDistance:= ?LREAL? ,
  MasterSyncPosition:= ?LREAL? ,
  StartMode:= ?INT? ,
  MasterValueSource:= ?INT? ,
  CamTableID:= ?MC_CAM_ID? ,
  BufferMode:= ?INT? ,
  Options:= ?DWORD? ,
  InSync=> ?BOOL? ,
  Busy=> ?BOOL? ,
  Active=> ?BOOL? ,
  CommandAborted=> ?BOOL? ,
  Error=> ?BOOL? ,
  ErrorID=> ?WORD? ,
  EndOfProfile=> ?BOOL?
);
```

Name	Number of input area points (byte)	Number of output area points (byte)	Compilation method	FB operation
Cam Operation Start	96	48	Subroutine type	Real-time execution

Setting data

■ I/O variable

I/O variable	Name	Data type	Input import	Setting range	Default value	Description
Master	Master axis	AXIS_REF	At start	—	Mandatory	This variable sets the axis. For the variables used (AxisName.AxisRef.), refer to the following. ☞ Page 45 AxisName.AxisRef. (Axis information)
Slave	Slave axis	AXIS_REF	At start	—	Mandatory	This variable sets the axis. For the variables used (AxisName.AxisRef.), refer to the following. ☞ Page 45 AxisName.AxisRef. (Axis information)

■ Input variables

Input variable	Name	Data type	Import	Setting range	Default value	Description
Execute	Execute command	BOOL	At start	TRUE, FALSE	FALSE	When this variable is TRUE, MC_CamIn (Cam Operation Start) is executed.
ContinuousUpdate	Continuous update	BOOL	At start	TRUE, FALSE	FALSE	This variable sets whether to enable or disable continuous change of Master axis offset (MasterOffset), Slave axis offset (SlaveOffset), Master axis scaling (MasterScaling), Slave axis scaling (SlaveScaling), and Cam table ID (CamTableID). • FALSE:Disable • TRUE:Enable

Input variable	Name	Data type	Import	Setting range	Default value	Description
MasterOffset	Master axis offset	LREAL	At start / Retrigger possible / Continuous update possible	-10000000000.0 to 10000000000.0	0.0	This variable shifts the phase of the master axis (Master) by the offset amount. (This does not affect Master axis follow-up distance (MasterStartDistance) or Master axis synchronization start position (MasterSyncPosition).) If a value other than "0" is set at the start of operation, at the rising edge of In synchronization (InSync), the offset amount to the master axis (Master) is added to the cam 1-cycle position. The same applies when the value is changed while In synchronization (InSync) is TRUE.
SlaveOffset	Slave axis offset	LREAL	At start / Retrigger possible / Continuous update possible	-10000000000.0 to 10000000000.0	0.0	This variable shifts the displacement of the slave axis (Slave) by the offset amount. If a value other than "0.0" is set before the rising edge of In synchronization (InSync), the command is output with the offset amount added to the slave axis (Slave) position at the rising edge of In synchronization (InSync). The same applies when the value is changed while In synchronization (InSync) is TRUE.
MasterScaling	Master axis scaling	LREAL	At start / Retrigger possible / Continuous update possible	0.01 to 10.0	1.0	This variable expands/reduces the cam table.
SlaveScaling	Slave axis scaling	LREAL	At start / Retrigger possible / Continuous update possible	0.01 to 10.0	1.0	This variable increases/reduces the stroke amount of the cam table.
MasterStartDistance	Master axis follow-up distance	LREAL	At start	-10000000000.0 to 10000000000.0	0.0	This variable sets the position (the relative position from Master axis synchronization start position (MasterSyncPosition)) of the master axis (Master) where the output axis (OutputData) starts synchronization.
MasterSyncPosition	Master axis synchronization start position	LREAL	At start	-10000000000.0 to 10000000000.0	0.0	This variable sets the position of the master axis (Master) to start synchronization of Current value per cycle (InputPerCycle).
StartMode	Start mode	INT (MC_START_MODE)	At start	0, 1	0	This variable sets the start timing of cam operation. • 0:Immediate (mcImmediate) • 1:Absolute (mcAbsolute)
MasterValueSource	Master axis data source selection	INT (MC_SOURCE)	At start	1, 2, 101, 102	1	This variable sets the data source of the master axis (Master). • 1:Set value (mcSetValue) • 2:Actual value (mcActualValue) • 101:Latest set value (mcLatestSetValue) • 102:Latest actual value (mcLatestActualValue)
CamTableID	Cam table ID	MC_CAM_ID	At start / Retrigger possible / Continuous update possible	1 to 6000	0	This variable sets the cam ID. The cam ID is shared to the open area by MC_CamTableSelect (Cam Table Selection) in advance.
BufferMode	Buffer mode	INT (MC_BUFFER_MODE)	At start	0, 1	0	This variable sets the buffer mode. • 0:Aborting(mcAborting) • 1:Buffered(mcBuffered)
Options	Options	DWORD(HEX)	At start	00000000H to 00010000H	00000000H	This variable sets the functional options for MC_CamIn (Cam Operation Start) in terms of bits.

■ Output variables

Output variable	Name	Data type	Default value	Description
InSync	In synchronization	BOOL	FALSE	This variable becomes TRUE when Output value (OutputData) starts synchronization.
Busy	Executing	BOOL	FALSE	This variable becomes TRUE when MC_ CamIn (Cam Operation Start) is executed.
Active	Controlling	BOOL	FALSE	This variable becomes TRUE when Current value per cycle (InputPerCycle) starts synchronization.
CommandAborted	Abortion of execution	BOOL	FALSE	This variable becomes TRUE when execution of MC_CamIn (Cam Operation Start) is aborted. This variable becomes FALSE when Execute command (Execute) becomes FALSE.
Error	Error	BOOL	FALSE	This variable becomes TRUE when an error occurs.
ErrorID	Error code	WORD(UINT)	0	When an error occurs, this variable returns the error code. For details of error codes, refer to the following. MELSEC iQ-R Motion Module User's Manual (Application)
EndOfProfile	Cam cycle completion	BOOL	FALSE	After Controlling (Active) becomes TRUE, this variable becomes TRUE only for 1 scan of the program cycle each time the axis moves the one cycle length.

■ Public variables

Public variable	Name	Data type	Default value	Description
InputPerCycle	Current value per cycle	LREAL	0.0	This variable stores the current value per cycle.
Reference	Reference value	LREAL	0.0	This variable stores the reference value.
OutputData	Output value	LREAL	0.0	This variable stores the output value.
InstanceID	Instance ID	WORD(UINT)	0	The instance ID. This variable is automatically set by the system when an instance is created. This instance ID is used in FB input, etc.

Processing details

- MC_CamIn (Cam Operation Start) sets Master axis offset (MasterOffset), Slave axis offset (SlaveOffset), Master axis scaling (MasterScaling), Slave axis scaling (SlaveScaling), Master axis follow-up distance (MasterStartDistance), Master axis synchronization start position (MasterSyncPosition), Start mode (StartMode), Master axis data source selection (MasterValueSource), Cam tableID (CamTableID), and Buffer mode (BufferMode), then executes the cam operation.
- To stop the operation, perform MC_Stop (Forced Stop).

Program example

Create a program that executes cam operation based on the specified cam data and performs 2-axis synchronous control.

■ Labels used

The following table lists the global and local labels used in this program.

Category	Label name	Data type	Class	Public label	Description
Local label	bMoveCam	Bit	VAR	—	Cam Operation Start
	bInSync1	Bit	VAR	—	In synchronization
	bCamInAborted	Bit	VAR	—	Abortion of execution
	bCamInError	Bit	VAR	—	Error

■ Operation profile data

The following table lists the operation profile data used in this program.

Profile ID	Label name	Data type	Description
1	ProfileData0001	MC_CAM_REF	2-axis synchronous control cam data

■ Auto expansion of cam data

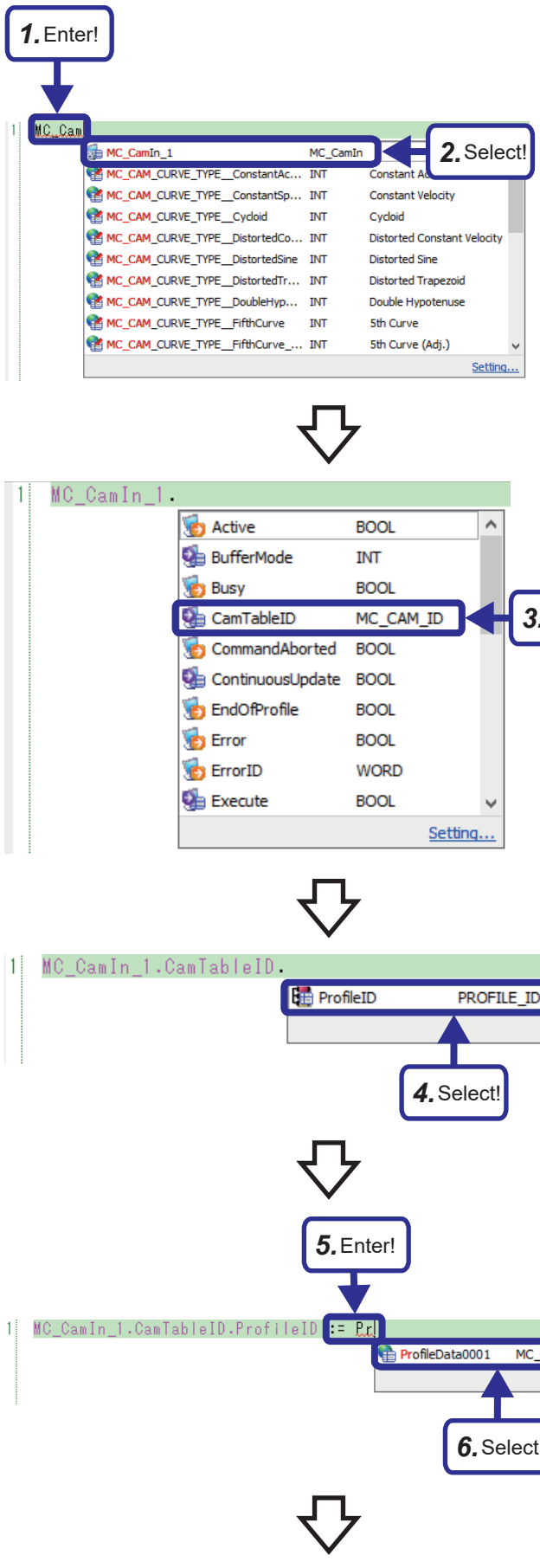
The cam data created as the operation profile data must be stored in the cam open area in the Motion module.

If "Auto Expand" for the cam data is set to "Yes" in ☞ Page 153 Creating new operation profile data, the cam data is automatically stored in the cam open area when [Y0] is turned on.

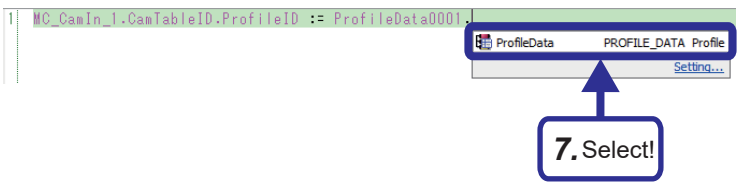
At this time, for "CamTableID" specified for MC_CamIn, specify "ProfileData name.ProfileData.ID" for the member ProfileID of the FB name.CamTableID structure. This specifies the profile data to be used.

Delete or comment out the CamTableID input in the FB.

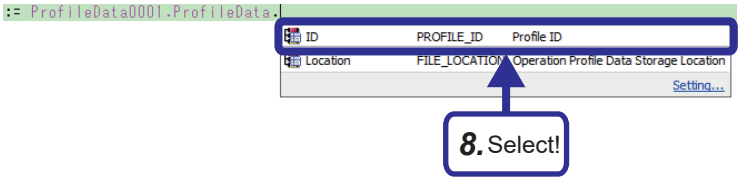
Operating procedure



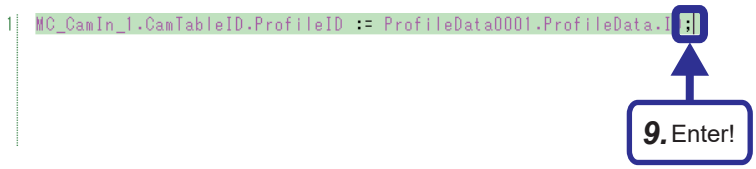
1. Enter "MC_Cam".
2. Select the FB name "MC_CamIn_1" from the displayed options.
3. Enter ".", and select "CamTableID" from the displayed options.
4. Enter ".", and select "ProfileID" from the displayed structure variables.
5. Enter ":=" and "Pr".
6. Select "ProfileData0001" from the displayed structure variables.



7. Enter ".", and select "ProfileData" from the displayed structures.



8. Enter ".", and select "ID" from the displayed structures.



9. Enter ";".

Practice 5-2

Create a program that specifies operation profile data and executes synchronous control in the program block "SynchronousDemo1".

Select "MC_CamIn" from [Motion - Individual] under [Motion Control Function/Function Block] in "POU List" in the Element Selection window.

```
31  //Detect the rising edge of workpiece detection
32  R_TRIG_1(
33      CLK:= NZ2GN2B1_32D_001_RX0 ,
34      Q=> bWorkON
35      );
36
37  //Start synchronization along with the workpiece detection timing
38  bMoveCam := bMove1InVelo AND bWorkON;
39
40  //Specify operation profile data of the motion to be performed by synchronous control of Axis 1 and Axis 3
41  MC_CamIn_1.CamTableID.ProfileID := ProfileData0001.ProfileData.ID; ------(1)
42  //Synchronous control execution Master axis: Axis 1 Slave axis: Axis 3
43  MC_CamIn_1(
44      Master:= Axis0001.AxisRef , ------------------------------------(2)
45      Slave:= Axis0003.AxisRef , -------------------------------------(3)
46      Execute:= bMoveCam , -------------------------------------------(4)
47      InSync=> bInSync1 , --------------------------------------------(5)
48      CommandAborted=> bCamInAborted , -------------------------------(6)
49      Error=> bCamInError  -------------------------------------------(7)
50      );
51
```

Starts synchronous control when a workpiece is detected.

No.	Description
(1)	Sets the auto expansion of cam data for synchronous control (axis 1 and axis 3).
(2)	Sets the axis information of axis 1 for the master axis.
(3)	Sets the axis information of axis 3 for the slave axis.
(4)	Sets the command to start the cam operation of synchronous control (axis 1 and axis 3).
(5)	Stores "Execution start" of the cam operation start (axis 1 and axis 3) FB.
(6)	Stores "Abortion of execution" of the cam operation start (axis 1 and axis 3) FB.
(7)	Stores errors (axis 1 and axis 3).

Point

Delete or comment out the CamTableID in the "MC_CamIn" FB when the operation profile data is specified by auto expansion of the cam data.

FB name: MC_Stop

Decelerates the specified axis to stop.

The following shows the details of MC_Stop.

```
MC_Stop(
  Axis:= ?AXIS_REF? ,
  Execute:= ?BOOL? ,
  Deceleration:= ?LREAL? ,
  Jerk:= ?LREAL? ,
  Options:= ?DWORD? ,
  Done=> ?BOOL? ,
  Busy=> ?BOOL? ,
  Active=> ?BOOL? ,
  CommandAborted=> ?BOOL? ,
  Error=> ?BOOL? ,
  ErrorID=> ?WORD?
);
```

Name	Number of input area points (byte)	Number of output area points (byte)	Compilation method	FB operation
Forced Stop	36	8	Subroutine type	Real-time execution

Setting data

■ I/O variable

I/O variable	Name	Data type	Input import	Setting range	Default value	Description
Axis	Axis information	AXIS_REF	At start	—	Mandatory	This variable sets the axis. For the variables used (AxisName.AxisRef.), refer to the following. ☞ Page 45 AxisName.AxisRef. (Axis information)

■ Input variables

Input variable	Name	Data type	Import	Setting range	Default value	Description
Execute	Execute command	BOOL	At start	TRUE, FALSE	FALSE	When this variable is TRUE, MC_Stop (Forced Stop) is executed.
Deceleration	Deceleration	LREAL	At start / Retrigger possible	0.0000, 0.0001 to 2147483647.0	0.0	This variable sets the deceleration.
Jerk	Jerk	LREAL	At start / Retrigger possible	0.0	0.0	Set this variable to "0.0". If a value other than "0.0" is set, "Out of Jerk Range (error code: 1A13H)" occurs.
Options	Options	DWORD(HEX)	At start	00000000H	00000000H	Set this variable to "00000000H". If a value other than "00000000H" is set, "Out of Options Range (error code: 1A4EH)" occurs.

■ Output variables

Output variable	Name	Data type	Default value	Description
Done	Execution completion	BOOL	FALSE	This variable becomes TRUE when the velocity becomes 0.
Busy	Executing	BOOL	FALSE	This variable becomes TRUE when MC_Stop (Forced Stop) is executed.
Active	Controlling	BOOL	FALSE	This variable becomes TRUE while MC_Stop (Forced Stop) is controlling the axis.
CommandAborted	Abortion of execution	BOOL	FALSE	This variable becomes TRUE when execution of MC_Stop (Forced Stop) is aborted.
Error	Error	BOOL	FALSE	This variable becomes TRUE when an error occurs.
ErrorID	Error code	WORD(UINT)	0	When an error occurs, this variable returns the error code. For details of error codes, refer to the following. 📖MELSEC iQ-R Motion Module User's Manual (Application)

Processing details

- MC_Stop (Forced Stop) sets Deceleration (Deceleration) and decelerates the FB being executed to stop.
- When MC_Stop (Forced Stop) is executed, Abortion of execution (CommandAborted) becomes TRUE in the FB being executed and Axis status (AxisName.Md.AxisStatus) changes to "2: Decelerating to Stop (Stopping)". While Execute command (Execute) is TRUE or if the velocity has not reached "0.0", "2: Decelerating to Stop (Stopping)" is maintained. When Execution completion (Done) becomes TRUE and Execute command (Execute) becomes FALSE at stop completion, the axis status changes to "4: Standby (Standstill)".

Program example

Create a program to decelerate the operation of 2-axis synchronous control to stop.

■ Labels used

The following table lists the global and local labels used in this program.

Category	Label name	Data type	Class	Public label	Description
Global label	G_bSyncStop_Demo1	Bit	VAR_GLOBAL	Enable	2-axis synchronous control stop Turns on when "M31" is tapped on the GOT.
	G_bSynchro1Error	Bit	VAR_GLOBAL	Enable	2-axis synchronous control error Stores errors of velocity control or cam control.
Local label	bAborted	Bit	VAR	—	2-axis synchronous control execution aborted Stores "Abortion of execution" of velocity control or cam control.
	bStop1Done	Bit	VAR	—	Axis 3 deceleration stop execution complete
	bStop2Done	Bit	VAR	—	Axis 1 deceleration stop execution complete

Practice 5-3

Create the following program in the program block "SynchronousDemo1".
Select "MC_Stop" from [Motion - Individual] under [Motion Control Function/Function Block] in "POU List" in the Element Selection window.

```
52  //Axis 3: Control stop
53  MC_Stop_1(
54      Axis:= Axis0003.AxisRef ,                                          (1)
55      Execute:= G_bSyncStop_Demo1 OR bAborted OR G_bSynchro1Error,       (2)
56      Done=> bStop1Done                                                  (3)
57      );
58
```

No.	Description
(1)	Sets the axis information of axis 3.
(2)	Sets the command to forcibly stop 2-axis synchronous control (2-axis synchronous control stop, 2-axis synchronous control abortion of execution, or 2-axis synchronous control error).
(3)	Stores deceleration stop execution completion of forced stop (axis 3).

Fill in the blanks to complete a program.

```
59  //Axis 1: Control stop
60  MC_Stop_2(
61      Axis:=[        (1)        ],
62      Execute:= [                              (2)                              ],
63      Done=> [      (3)      ]
64      );
65
66  //Notify the PLC of error occurrence in synchronous control
67  G_bSynchro1Error := bMove1Error OR bCamInError;
68
69  //Notify the PLC of synchronous control execution completion
70  bAborted := bMove1Aborted OR bCamInAborted;
71  G_bSyncDone_Demo1 := (bStop1Done AND bStop2Done) OR G_bSynchro1Error OR bAborted;
```

Errors of each axis is stored.

Each axis status is stored.

No.	Description
(1)	Sets the axis information of axis 1.
(2)	Sets the command to forcibly stop 2-axis synchronous control (2-axis synchronous control stop, 2-axis synchronous control abortion of execution, or 2-axis synchronous control error).
(3)	Stores deceleration stop execution completion of forced stop (axis 1).

■ Answer

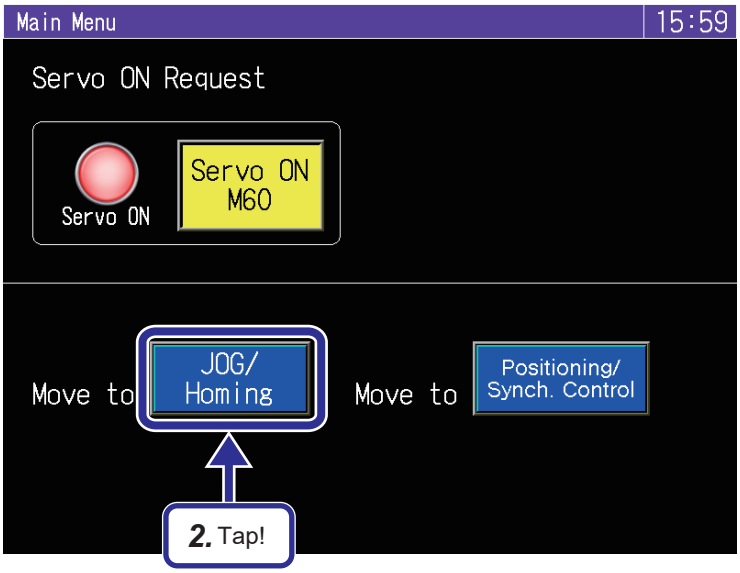
The following shows the answer program.

```
52  //Axis 3: Control stop
53  MC_Stop_1(
54      Axis:= Axis0003.AxisRef ,
55      Execute:= G_bSyncStop_Demo1 OR bAborted OR G_bSynchro1Error,
56      Done=> bStop1Done
57      );
58
59  //Axis 1: Control stop
60  MC_Stop_2(
61      Axis:= Axis0001.AxisRef ,
62      Execute:= G_bSyncStop_Demo1 OR bAborted OR G_bSynchro1Error ,
63      Done=> bStop2Done
64      );
65
66  //Notify the PLC of error occurrence in synchronous control
67  G_bSynchro1Error := bMove1Error OR bCamInError;
68
69  //Notify the PLC of synchronous control execution completion
70  bAborted := bMove1Aborted OR bCamInAborted;
71  G_bSyncDone_Demo1 := (bStop1Done AND bStop2Done) OR G_bSynchro1Error OR bAborted;
```

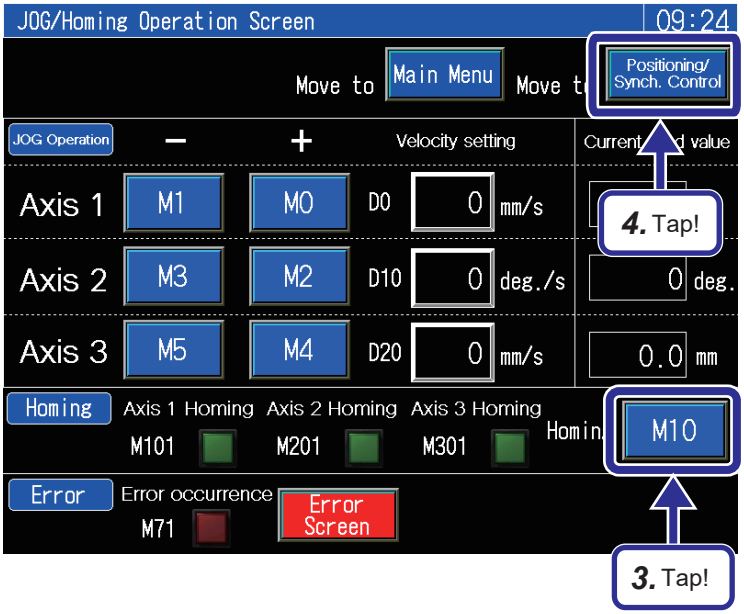
Operation check

After creating the program, write it to the programmable controller by following the same procedure as "☞ Page 110 Writing to the programmable controller" and check the operation of 2-axis synchronous control.

Operating procedure



1. Turn on the servo.
2. Tap the [JOG homing] button on the main menu screen.

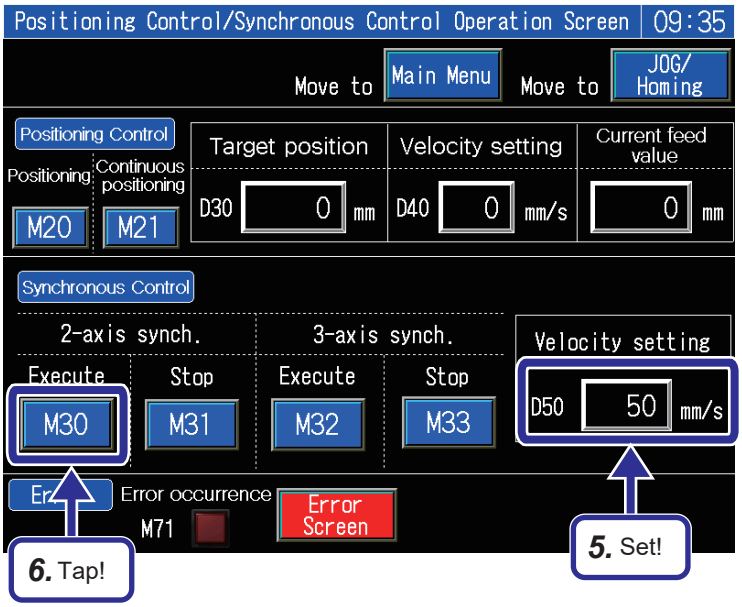


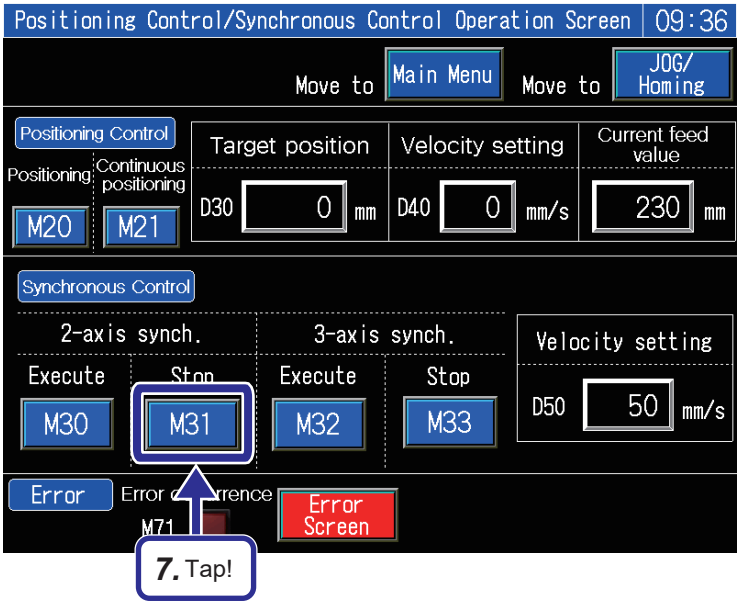
3. Tap the [M10] button for homing.

Point

Always perform homing before synchronous control.

4. Tap the [Positioning/Synch. Control] button.





5. Set the desired velocity in the velocity setting "D50".

6. Tap the [M30] button for 2-axis synchronous execution.

When the sensor detects a workpiece on axis 1, axis 3 moves 10 cm horizontally in sync with the velocity of axis 1.

As the synchronous control comes to the end (axis 3 moves 10 cm), axis 3 returns to the home position. When a workpiece is detected again, synchronous control is started.

7. Stop 2-axis synchronous control by tapping the [M31] button for 2-axis synchronous stop.

8.3 3-Axis Synchronous Control

Cam data setting

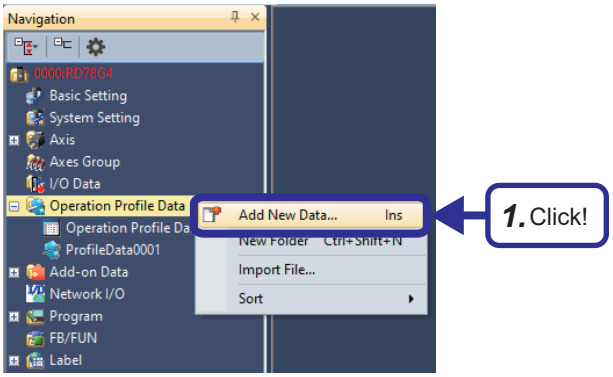
For operation profile data (cam data), refer to the following.

☞ Page 153 Cam data setting

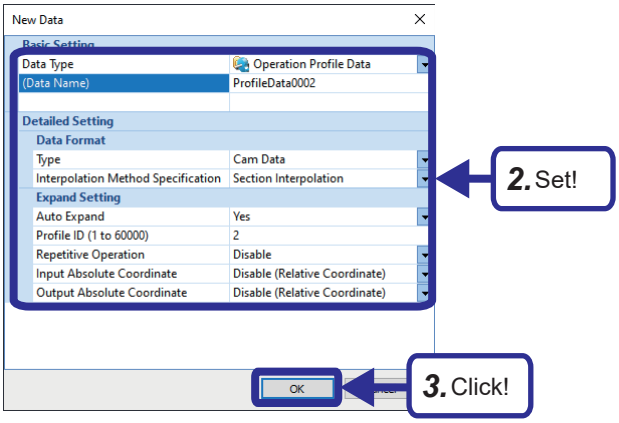
Adding operation profile data

The following shows how to add operation profile data with the motion control setting function.

Operating procedure



1. In the Navigation window of the motion control setting function, right-click [Operation Profile Data] and click [Add New Data].



2. Configure the Basic Setting and Detailed Setting as follows.

[Setting details]

Data Type: Operation Profile Data (Default)

(Data Name): ProfileData0002 (Default)

Type: Cam Data (Default)

Interpolation Method Specification: Section Interpolation (Default)

Auto Expand: Yes (Default)

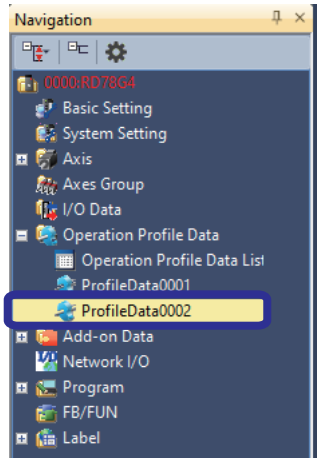
Profile ID: 2 (Default)

Repetitive Operation: Disable

Input Absolute Coordinate: Disable (Relative Coordinate) (Default)

Output Absolute Coordinate: Disable (Relative Coordinate) (Default)

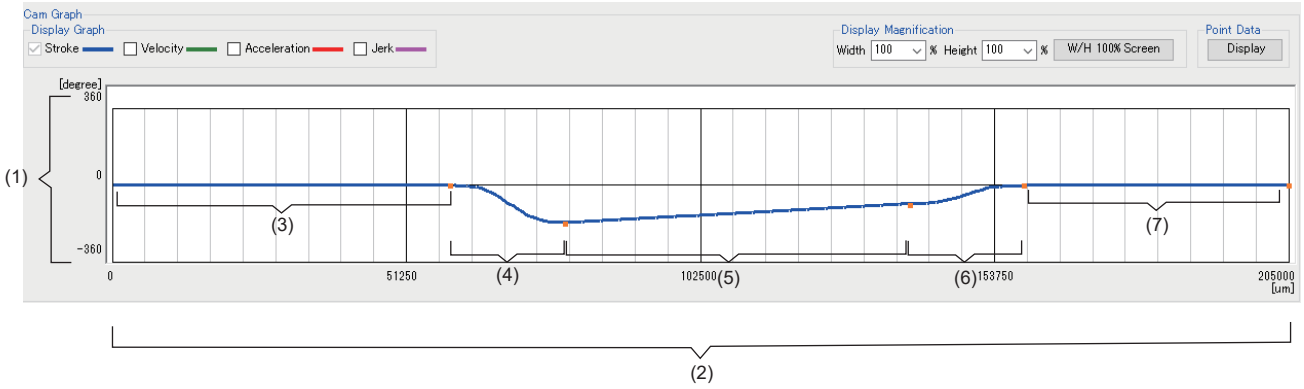
3. Click the [OK] button.



4. The operation profile data is added to the Navigation window.

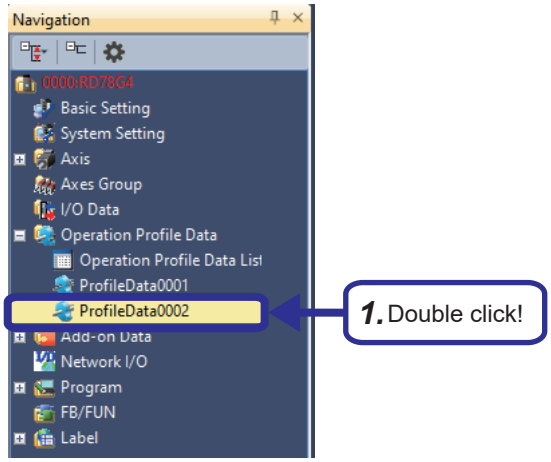
Creating cam data

The following shows the waveform of the cam data to be created.

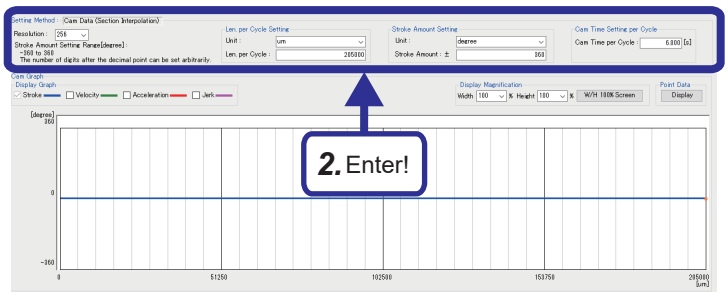


No.	Description
(1)	Indicates the movement amount of the slave axis (axis 2).
(2)	Indicates the movement amount of the master axis (axis 1).
(3)	Indicates the movement amount to the dog sensor of axis 3 after the sensor detects a workpiece on axis 1. Although the movement amount of axis 1 increases because it is operating at a constant velocity, the movement amount of axis 2 does not change because it is stopped.
(4)	Indicates the distance through which axis 2 rotates 180° (moves down) while following the workpiece on axis 1. Although the movement amount of axis 1 increases, the movement amount axis 2 decreases (the axis descends).
(5)	Indicates the distance through which axis 2 rotates 90° (moves up) from its lowered position while following the workpiece on axis 1. The movement amount of axis 2 increases (the axis moves up) along with axis 1.
(6)	Indicates the travel distance of axis 2 to return to the home position (rotates 90°) while following the workpiece on axis 1. The movement amount of axis 2 increases (the axis moves up) along with axis 1.
(7)	Indicates the standby distance of axis 2. Although the movement amount of axis 1 increases because it is operating at a constant velocity, the movement amount of axis 2 does not change because it is stopped.

Operating procedure



1. In the Navigation window of the motion control setting function, double-click [ProfileData0002] under [Operation Profile Data].



2. Set each item as follows.

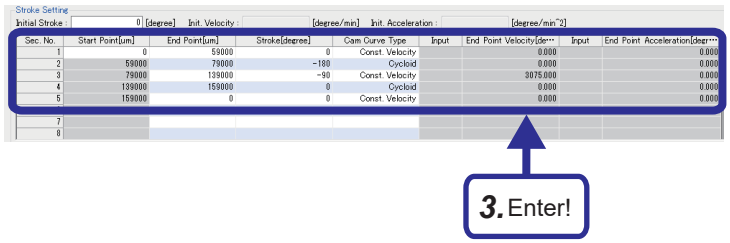
[Setting details]

Resolution: 256

Unit: um (Manual input), Len. per Cycle Setting: 205000

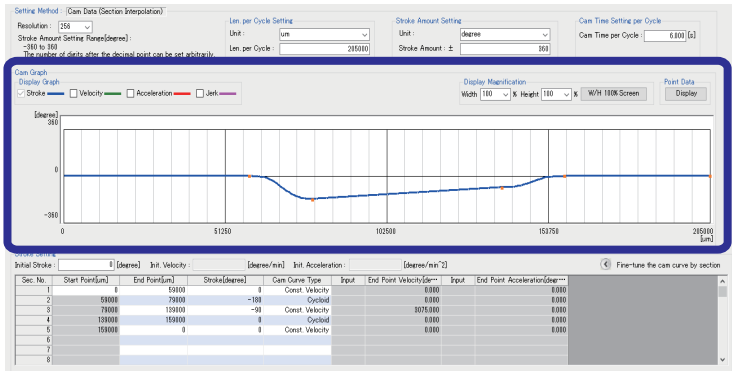
Unit: degree, Stroke Amount: 360

Cam Time per Cycle: 6.000



3. Configure the stroke settings as follows.

Sec. No.	Start Point	End Point	Stroke	Cam Curve Type	Description
1	0	59000	0	Const. Velocity	Set the distance from the sensor to the dog sensor of axis 3.
2	59000	79000	-180	Cycloid	Set the distance through which axis 2 rotates 180° (moves down) while following the workpiece on axis 1 from the dog sensor of axis 3.
3	79000	139000	-90	Const. Velocity	Set the distance through which axis 2 rotates 90° (moves up) while following the workpiece on axis 1.
4	139000	159000	0	Cycloid	Set the travel distance of axis 2 to return to the home position while following the workpiece on axis 1.
5	159000	0	0	Const. Velocity	Set the distance over which axis 2 stops.



4. The waveform as shown on the left is created.

3-axis synchronous control program

Create the following program: Axis 1 starts conveying the workpiece at a constant speed, and when the sensor detects the workpiece, axis 3 and axis 2 are synchronized with axis 1 according to the cam pattern created as the operation profile data.

FB to be used

The following table lists the FB used in the 3-axis synchronous control program.

Create the 3-axis synchronous control program by combining the FB that controls the velocity of axis 1 (MC_MoveVelocity), FB that executes the cam operation based on the cam data created by axis 1 and axis 2 as well as axis 1 and axis 3 (MC_CamIn), and FB that stops the synchronous control of axis 1, axis 2, and axis 3 (MC_Stop).

Type	FB	Description
Motion	MC_MoveVelocity	Switches the driver to csv and controls the velocity control according to the specified velocity.
	MC_CamIn	Executes cam operation.
	MC_Stop	Decelerates the specified axis to stop. This FB is used to stop synchronization.

Program example

Create the 3-axis synchronous control program for axis 1, axis 2, and axis 3.

■ Labels used

The following table lists the global and local labels used in this program.

Category	Label name	Data type	Class	Public label	Description
Global label	G_bSyncStop_Demo2	Bit	VAR_GLOBAL	Enable	3-axis synchronous control stop
	G_bSynchro2Error	Bit	VAR_GLOBAL	Enable	3-axis synchronous control error Stores errors of velocity control or cam control.
Local label	bSyncReq_Demo2	Bit	VAR	—	3-axis synchronous control execution request
	leVelocity	Double-precision real number	VAR	—	Axis 1 velocity Stores the velocity setting for synchronous control input from the GOT.
	leAcceleration	Double-precision real number	VAR	—	Axis 1 acceleration Stores the acceleration based on Axis 1 velocity (leVelocity).
	leDeceleration	Double-precision real number	VAR	—	Axis 1 deceleration Stores the deceleration based on Axis 1 velocity (leVelocity).
	leJerk	Double-precision real number	VAR	—	Axis 1 jerk Stores the jerk based on Axis 1 velocity (leVelocity).
	bMove1InVelo	Bit	VAR	—	Axis 1 target velocity reached
	bMove1Aborted	Bit	VAR	—	Axis 1 execution aborted
	bMove1Error	Bit	VAR	—	Axis 1 error
	bMoveCam	Bit	VAR	—	Cam Operation Start
	bAborted	Bit	VAR	—	3-axis synchronous control execution aborted Stores "Abortion of execution" of velocity control or cam control.
	bInSync1	Bit	VAR	—	In synchronization (axis 1 and axis 3)
	bCamIn1Aborted	Bit	VAR	—	Abortion of execution (axis 1 and axis 3)
	bCamIn1Error	Bit	VAR	—	Error (axis 1 and axis 3)
	bInSync2	Bit	VAR	—	In synchronization (axis 1 and axis 2)
	bCamIn2Aborted	Bit	VAR	—	Abortion of execution (axis 1 and axis 2)
	bCamIn2Error	Bit	VAR	—	Error (axis 1 and axis 2)
	bStop1Done	Bit	VAR	—	Axis 2 deceleration stop execution complete
	bStop2Done	Bit	VAR	—	Axis 3 deceleration stop execution complete
	bStop3Done	Bit	VAR	—	Axis 1 deceleration stop execution complete

Practice 6

Fill in the blanks to complete a program in the program block "SynchronousDemo2".

Add the synchronous control program for axis 1 and axis 3 in the same manner as the program created for 2-axis synchronous control, and add synchronous control program for axis 1 and axis 2 as well.

Select "MC_MoveVelocity", "MC_CamIn", and "MC_Stop" from [Motion - Individual] under [Motion Control Function/Function Block] in "POU List" in the Element Selection window.

```
1  //------3-axis synchronous control------
2  //Synchronous control execution request
3  IF  (G_bSyncCMD_Demo2 AND NOT G_bSyncStop_Demo2)  THEN
4      bSyncReq_Demo2 := TRUE;
5      ELSE
6      bSyncReq_Demo2 := FALSE;
7  END_IF;
8
9  //Velocity setting for synchronous control (velocity, acceleration, deceleration, and jerk)
10 IF (G_leSyncSetVelocity <> leVelocity) THEN
11     leVelocity     := G_leSyncSetVelocity;
12     leAcceleration := G_leSyncSetVelocity * 2.0;
13     leDeceleration := G_leSyncSetVelocity * 2.0;
14     leJerk         := G_leSyncSetVelocity * 4.0;
15 END_IF;
16
17 //1-axis: Velocity control execution
18 MC_MoveVelocity_1(
19     Axis:=[      (1)       ],
20     Execute:=[      (2)      ],
21     Velocity:= [   (3)   ] ,
22     Acceleration:= [     (4)      ] ,
23     Deceleration:= [     (5)      ] ,
24     Jerk:=[  (6)  ] ,
25     Direction:[               (7)                ] ,
26     InVelocity=>[     (8)      ],
27     CommandAborted=>[     (9)      ],
28     Error=>[      (10)      ]
29     );
30
31 //Detect the rising edge of workpiece detection
32 R_TRIG_1(
33     CLK:= NZ2GN2B1_32D_001_RX0 ,
34     Q=> bWorkON
35     );
36
37 //Start synchronization along with the workpiece detection timing
38 bMoveCam := bMoveInVelo AND bWorkON;
39
```

When the 3-axis synchronous control request is turned on, the synchronous control command turns on.

When the synchronous control speed is changed, acceleration, deceleration, and jerk are recalculated according to the speed.

Starts synchronous control when a workpiece is detected.

No.	Description
(1)	Sets the axis information of axis 1.
(2)	Sets the execution request for performing 3-axis synchronous control (axis 1, axis 2, and axis 3).
(3)	Sets the velocity of the velocity control (axis 1).
(4)	Sets the acceleration of the velocity control (axis 1).
(5)	Sets the deceleration of the velocity control (axis 1).
(6)	Sets the jerk of the velocity control (axis 1).
(7)	Sets the direction of the control for the velocity control (axis 1) to the positive direction.
(8)	Stores "Target velocity reached" of the velocity control (axis 1) FB.
(9)	Stores "Abortion of execution" of the velocity control (axis 1) FB.
(10)	Stores errors (axis 1).

```
//Specify operation profile data of the motion to be performed by synchronous control of Axis 1 and Axis 3
[                          (1)                          ];
//Synchronous control execution Master axis: Axis 1 Slave axis: Axis 3
MC_CamIn_1(
    Master:=[        (2)        ] ,
    Slave:=[         (3)        ] ,
    Execute:=[   (4)    ],
    InSync=>[   (5)    ],
    CommandAborted= [       (6)       ] ,
    Error=>[         (7)         ]
    );

//Specify operation profile data of the motion to be performed by synchronous control of Axis 1 and Axis 2
[                                    (8)                                    ]
//Synchronous control execution Master axis: Axis 1 Slave axis: Axis 2
MC_CamIn_2(
    Master:=[        (9)        ] ,
    Slave:=[        (10)        ] ,
    Execute:= [  (11)  ] ,
    InSync=>[  (12)  ] ,
    CommandAborted=>[      (13)      ] ,
    Error=>[        (14)         ]
    );
```

No.	Description
(1)	Sets the auto expansion of cam data for synchronous control (axis 1 and axis 3).
(2)	Sets the axis information of axis 1 for the master axis.
(3)	Sets the axis information of axis 3 for the slave axis.
(4)	Sets the command to start the cam operation of synchronous control (axis 1 and axis 3).
(5)	Stores "Execution start" of the cam operation start (axis 1 and axis 3) FB.
(6)	Stores "Abortion of execution" of the cam operation start (axis 1 and axis 3) FB.
(7)	Stores errors (axis 1 and axis 3).
(8)	Sets the auto expansion of cam data for synchronous control (axis 1 and axis 2).
(9)	Sets the axis information of axis 1 for the master axis.
(10)	Sets the axis information of axis 2 for the slave axis.
(11)	Sets the command to start the cam operation of synchronous control (axis 1 and axis 2).
(12)	Stores "Execution start" of the cam operation start (axis 1 and axis 2) FB.
(13)	Stores "Abortion of execution" of the cam operation start (axis 1 and axis 2) FB.
(14)	Stores errors (axis 1 and axis 2).

```
//Axis 2: Control stop
MC_Stop_1(
    Axis:=[      (1)      ],
    Execute:=[                    (2)                    ] ,
    Done=>[      (3)      ]
    );

//Axis 3: Control stop
MC_Stop_2(
    Axis:=[      (4)      ] ,
    Execute:=[                    (5)                    ] ,
    Done=>[      (6)      ]
    );

//Axis 1: Control stop
MC_Stop_3(
    Axis:=[      (7)      ]
    Execute:=[                    (8)                    ] ,
    Done=>[      (9)      ]
    );


//Notify the PLC of error occurrence in synchronous control
G_bSynchro2Error := bMove1Error OR bCamIn1Error OR bCamIn2Error;
//Notify the PLC of synchronous control execution completion
bAborted := bMove1Aborted OR bCamIn1Aborted OR bCamIn2Aborted;
G_bSyncDone_Demo2 := (bStop1Done AND bStop2Done AND bStop3Done) OR G_bSynchro2Error OR bAborted;
```

Errors of each axis is stored.

Each axis status is stored.

No.	Description
(1)	Sets the axis information of axis 2.
(2)	Sets the command to forcibly stop 3-axis synchronous control (3-axis synchronous control stop, 3-axis synchronous control abortion of execution, or 3-axis synchronous control error).
(3)	Sets deceleration stop execution completion of forced stop (axis 2).
(4)	Sets the axis information of axis 3.
(5)	Sets the command to forcibly stop 3-axis synchronous control (3-axis synchronous control stop, 3-axis synchronous control abortion of execution, or 3-axis synchronous control error).
(6)	Sets deceleration stop execution completion of forced stop (axis 3).
(7)	Sets the axis information of axis 1.
(8)	Sets the command to forcibly stop 3-axis synchronous control (3-axis synchronous control stop, 3-axis synchronous control abortion of execution, or 3-axis synchronous control error).
(9)	Sets deceleration stop execution completion of forced stop (axis 1).

■ Answer

The following shows the answer program.

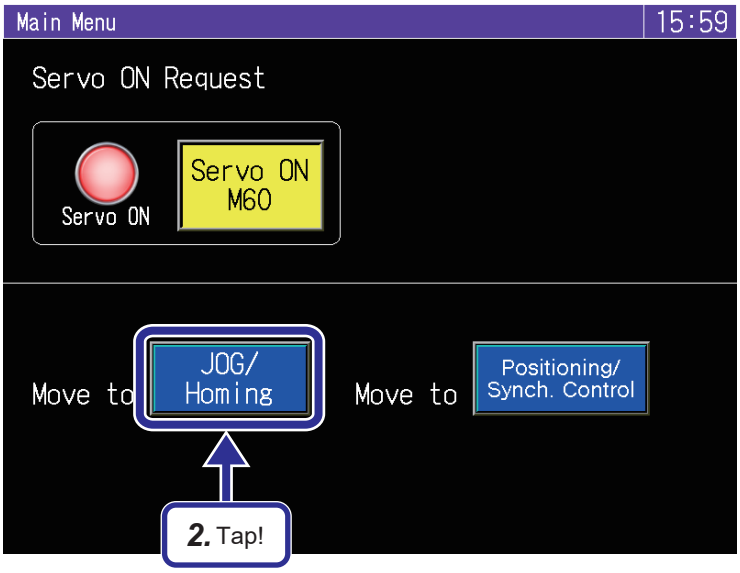
```
//------3-axis synchronous control------
//Synchronous control execution request
IF  (G_bSyncCMD_Demo2 AND NOT G_bSyncStop_Demo2)  THEN
    bSyncReq_Demo2 := TRUE;
    ELSE
    bSyncReq_Demo2 := FALSE;
END_IF;

//Velocity setting for synchronous control (velocity, acceleration, deceleration, and jerk)
IF (G_leSyncSetVelocity <> leVelocity) THEN
    leVelocity    := G_leSyncSetVelocity;
    leAcceleration := G_leSyncSetVelocity * 2.0;
    leDeceleration := G_leSyncSetVelocity * 2.0;
    leJerk         := G_leSyncSetVelocity * 4.0;
END_IF;

//1-axis: Velocity control execution
MC_MoveVelocity_1(
    Axis:= Axis0001.AxisRef ,
    Execute:= bSyncReq_Demo2 ,
    Velocity:= leVelocity ,
    Acceleration:= leAcceleration ,
    Deceleration:= leDeceleration ,
    Jerk:= leJerk ,
    Direction:= MC_DIRECTION__mcPositiveDirection ,
    InVelocity=> bMove1InVelo ,
    CommandAborted=> bMove1Aborted ,
    Error=> bMove1Error
    );

//Detect the rising edge of workpiece detection
R_TRIG_1(
    CLK:= NZ2GN2B1_32D_001_RX0 ,
    Q=> bWorkON
    );

//Start synchronization along with the workpiece detection timing
bMoveCam := bMove1InVelo AND bWorkON;

```

```
//Specify operation profile data of the motion to be performed by synchronous control of Axis 1 and Axis 3
MC_CamIn_1.CamTableID.ProfileID := ProfileData0001.ProfileData.ID;
//Synchronous control execution Master axis: Axis 1 Slave axis: Axis 3
MC_CamIn_1(
    Master:= Axis0001.AxisRef ,
    Slave:= Axis0003.AxisRef ,
    Execute:= bMoveCam ,
    InSync=> bInSync1 ,
    CommandAborted=> bCamIn1Aborted ,
    Error=> bCamIn1Error
    );

//Specify operation profile data of the motion to be performed by synchronous control of Axis 1 and Axis 2
MC_CamIn_2.CamTableID.ProfileID := ProfileData0002.ProfileData.ID;
//Synchronous control execution Master axis: Axis 1 Slave axis: Axis 2
MC_CamIn_2(
    Master:= Axis0001.AxisRef ,
    Slave:= Axis0002.AxisRef ,
    Execute:= bMoveCam ,
    InSync=> bInSync2 ,
    CommandAborted=> bCamIn2Aborted ,
    Error=> bCamIn2Error
    );


//Axis 2: Control stop
MC_Stop_1(
    Axis:= Axis0002.AxisRef ,
    Execute:= G_bSyncStop_Demo2 OR bAborted OR G_bSynchro2Error ,
    Done=> bStop1Done
    );

//Axis 3: Control stop
MC_Stop_2(
    Axis:= Axis0003.AxisRef ,
    Execute:= G_bSyncStop_Demo2 OR bAborted OR G_bSynchro2Error ,
    Done=> bStop2Done
    );

//Axis 1: Control stop
MC_Stop_3(
    Axis:= Axis0001.AxisRef ,
    Execute:= G_bSyncStop_Demo2 OR bAborted OR G_bSynchro2Error ,
    Done=> bStop3Done
    );
```

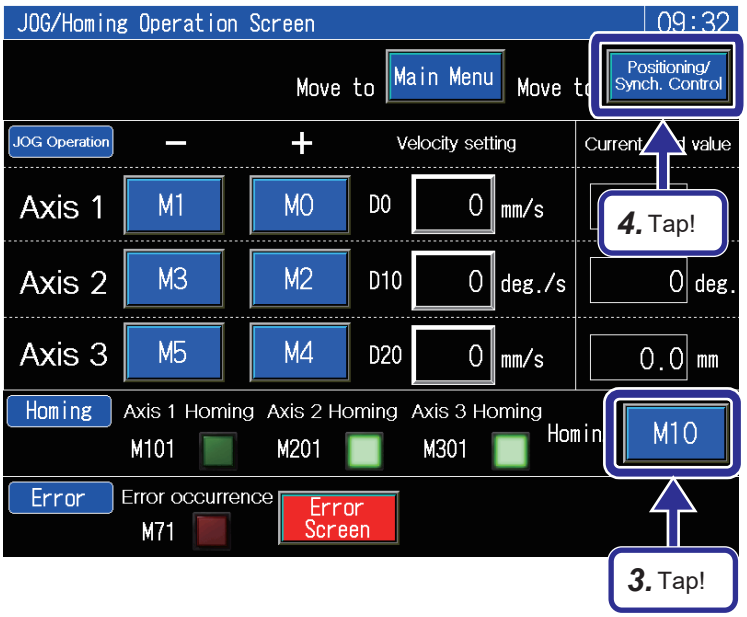
Operation check

After creating the program, write it to the programmable controller by following the same procedure as "☞ Page 110 Writing to the programmable controller" and check the operation of 3-axis synchronous control.

Operating procedure



1. Turn on the servo.
2. Tap the [JOG homing] button on the main menu screen.

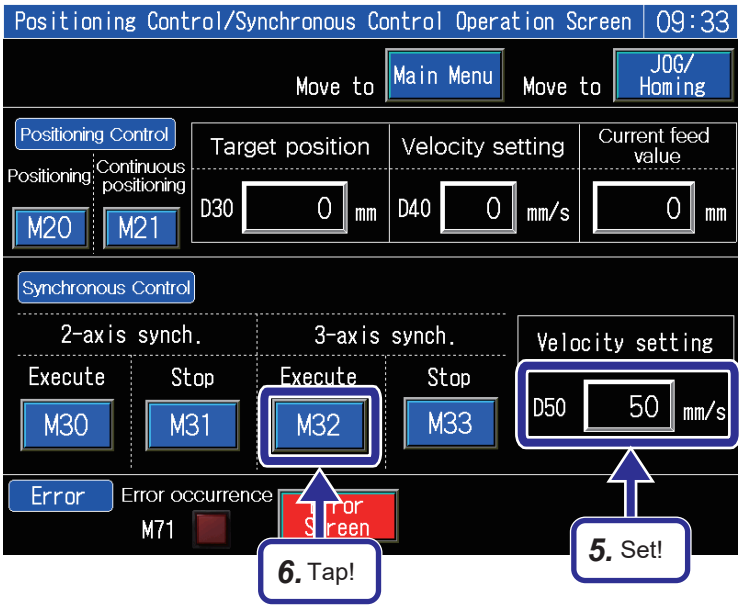


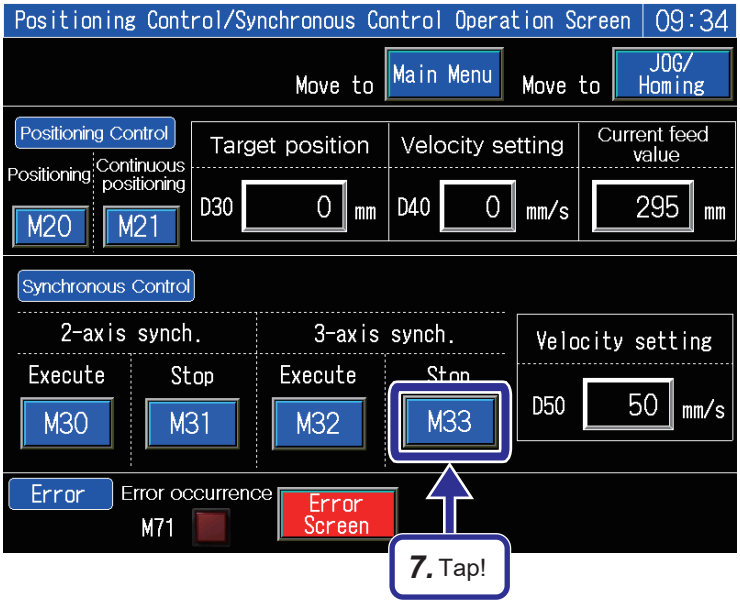
3. Tap the [M10] button for homing.

Point

Always perform homing before synchronous control.

4. Tap the [Positioning/Synch. Control] button.





5. Set the desired velocity in the velocity setting "D50".

6. Tap the [M32] button for 2-axis synchronous execution.

When the sensor detects a workpiece on axis 1, axis 3 moves 10 cm horizontally in sync with the velocity of axis, and axis 2 moves vertically in sync with the velocity of axis 1. As the synchronous control comes to the end (axis 3 moves 10 cm), axis 3 and axis 2 return to the home position.
When a workpiece is detected again, synchronous control is started.

7. Stop 2-axis synchronous control by tapping the [M33] button for 2-axis synchronous stop.

APPENDICES

Appendix 1 3-Axis Synchronous Control Program (Axis 1 Velocity Change)

Create a program that changes the velocity of axis 1 during synchronous control and synchronizes the velocity of axis 2 and axis 3, and check the operation.

FB to be used

The following table shows the FB used in the 3-axis synchronous control program (Axis 1 velocity change).
The 3-axis synchronous control program (Axis 1 velocity change) can be created by combining the FB that changes the velocity of axis 1 (MC_SetOverride) with the 3-axis synchronous control program. For the 3-axis synchronous control, refer to ☞ Page 174 3-Axis Synchronous Control.

Type	FB	Description
Administrative	MC_SetOverride	Changes the target velocity, target acceleration, and target deceleration of the specified address.

FB name: MC_SetOverride

This FB changes the target velocity, target acceleration, and target deceleration of the specified address.
The following shows the details of MC_SetOverride.

```
MC_SetOverride(
  Axis:= ?AXIS_REF? ,
  Enable:= ?BOOL? ,
  VelFactor:= ?LREAL? ,
  AccFactor:= ?LREAL? ,
  Acceleration:= ?LREAL? ,
  JerkFactor:= ?LREAL? ,
  Enabled=> ?BOOL? ,
  Busy=> ?BOOL? ,
  Error=> ?BOOL? ,
  ErrorID=> ?WORD?,
);
```

Name	Number of input area points (byte)	Number of output area points (byte)	Compilation method	FB operation
Override Value Setting	40	6	Subroutine type	Real-time execution

Setting data

■ I/O variable

I/O variable	Name	Data type	Input import	Setting range	Default value	Description
Axis	Axis information	AXIS_REF	At start	—	Mandatory	This variable sets the axis. For the variables used (AxisName.AxisRef.), refer to the following. ☞ Page 45 AxisName.AxisRef. (Axis information)

■ Input variables

Input variable	Name	Data type	Import	Setting range	Default value	Description
Enable	Enable	BOOL	At start	TRUE, FALSE	FALSE	When this variable is TRUE, it executes MC_SetOverride (Override Value Setting).
VelFactor	Velocity override factor	LREAL	Always	0.00 to 10.00	0.00	This variable sets the velocity override factor. When Enable (Enable) is TRUE, values are always imported.

Input variable	Name	Data type	Import	Setting range	Default value	Description
AccFactor	Acceleration override factor	LREAL	Always	0.00, 0.01 to 10.00	0.0	This variable sets the acceleration override factor. When Enable (Enable) is TRUE, values are always imported. When this variable is set to "0.00", the acceleration override factor is not changed and the previous value is used for control.
JerkFactor	Jerk override factor	LREAL	Always	0.0	0.0	Set this variable to "0.0". If a value other than "0.0" is set, "Out of Jerk Override Coefficient (JerkFactor) Range (error code: 349EH)" occurs.

■ Output variables

Output variable	Name	Data type	Default value	Description
Enabled	Enabled	BOOL	FALSE	This variable becomes TRUE when the correct override value is set.
Busy	Executing	BOOL	FALSE	This variable becomes TRUE when MC_SetOverride (Override Value Setting) is executed.
Error	Error	BOOL	FALSE	This variable becomes TRUE when an error occurs.
ErrorID	Error code	WORD (UINT)	0	When an error occurs, this variable returns the error code. For details of error codes, refer to the following. MELSEC iQ-R Motion Module User's Manual (Application)

Processing details

- This FB changes the target velocity, target acceleration, and target deceleration of the specified address.
- The target velocity, target acceleration, and target deceleration currently in operation are multiplied by the override factor.
- MC_SetOverride (Override Value Setting) is executed when Enable (Enable) becomes TRUE. Enabled (Enabled) is TRUE while the override factor is valid.
- If the override factor value is changed while Enable (Enable) is TRUE, the new override factor is applied.
- When an error occurs in MC_SetOverride (Override Value Setting), Error (Error) becomes TRUE and the error code is stored in Error code (ErrorID). For details of error codes, refer to the following.
  MELSEC iQ-R Motion Module User's Manual (Application)
- When the value for Velocity override factor (VelFactor) is set to "0.00", the axis stops without changing Axis status (AxisName.Md.AxisStatus) to "4: Standby (Standstill)".
- When the value for Acceleration override factor (AccFactor) is set to "0.00", the acceleration override factor is not changed and the previous acceleration override factor is maintained.

Program example

Add a motion control FB that executes the override value setting of axis 1 to the program block "SynchronousDemo2" in "school_Motion_appendix.gx3".
"school_Motion_appendix.gx3" contains the configured parameters and GOT control programs prepared for this exercise.

■ Labels used

The following table lists the local labels used in this program.

Category	Label name	Data type	Class	Public label	Description
Local label	bSyncReq_Demo2	Bit	VAR	—	3-axis synchronous control execution request
	leVelocity	Double-precision real number	VAR	—	Axis 1 velocity Stores the velocity setting for synchronous control input from the GOT.
	leVelFactor	Double-precision real number	VAR	—	Sets the velocity override factor. When Enable (Enable) is TRUE, values are always imported.

Additional assignment

Add the program in red frame below to the program block "SynchronousDemo2" in "school_Motion_appendix.gx3".
Select "MC_SetOverride" from [Administrative] under [Motion Control Function/Function Block] in "POU List" in the Element Selection window.

```
//------3-axis synchronous control------
//Synchronous control execution request
IF  (G_bSyncCMD_Demo2 AND NOT G_bSyncStop_Demo2)  THEN
    bSyncReq_Demo2 := TRUE;
    ELSE
    bSyncReq_Demo2 := FALSE;
END_IF;

//Velocity setting for synchronous control (velocity, acceleration, deceleration, and jerk)
IF (G_leSyncSetVelocity <> leVelocity) THEN
    leVelocity     := G_leSyncSetVelocity;
    leAcceleration := G_leSyncSetVelocity * 2.0;
    leDeceleration := G_leSyncSetVelocity * 2.0;
    leJerk         := G_leSyncSetVelocity * 4.0;
END_IF;

//1-axis: Velocity control execution
MC_MoveVelocity_1(
    Axis:= Axis0001.AxisRef ,
    Execute:= bSyncReq_Demo2 ,
    Velocity:= leVelocity ,
    Acceleration:= leAcceleration ,
    Deceleration:= leDeceleration ,
    Jerk:= leJerk ,
    Direction:= MC_DIRECTION__mcPositiveDirection ,
    InVelocity=> bMove1InVelo ,
    CommandAborted=> bMove1Aborted ,
    Error=> bMove1Error
    );

//[Add] Axis 1 velocity change
IF 100000.0 < leVelocity*leVelFactor THEN  // Disable MC_SetOverride if velocity setting exceeds 100 mm/s
    G_bSyncStop_Demo2:= TRUE;
END_IF;

IF bSyncReq_Demo2 THEN
    MC_SetOverride_1(
        Axis:= Axis0001.AxisRef, ---------- (1)
        Enable:= bSyncReq_Demo2 ,---------- (2)
        VelFactor:= leVelFactor  ---------- (3)
        );
END_IF;
```

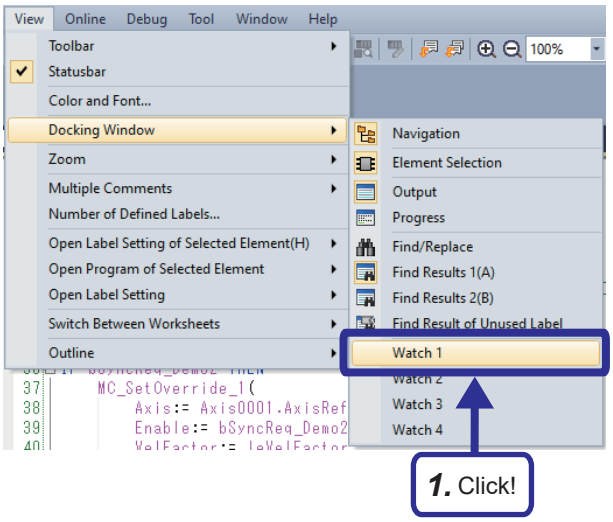
If the value obtained by multiplying the velocity input in GOT by the override factor exceeds 100mm/s, the FB (MC_SetOverride) is not executed.

No.	Description
(1)	Sets the axis information of axis 1.
(2)	Executes MC_SetOverride (Override Value Setting).
(3)	Sets the velocity override factor.

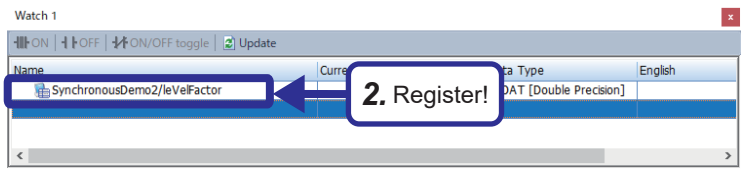
Operation check

After creating the program, write it in the same way as "☞ Page 110 Writing to the programmable controller" and check the operation of Axis 1 velocity change.

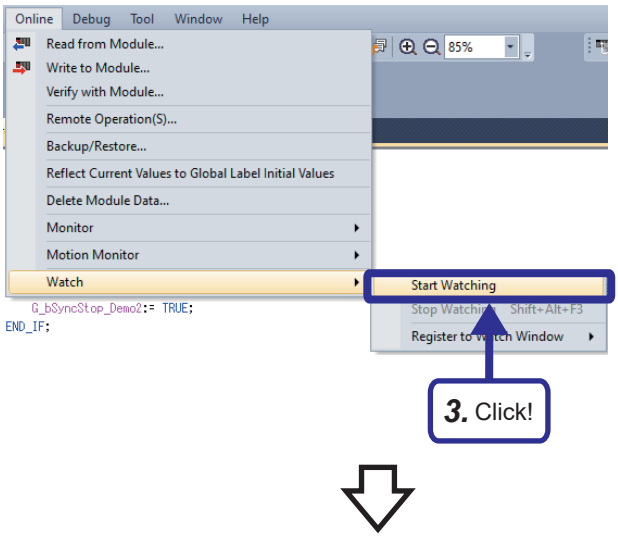
Operating procedure



1. Click [View] ⇨ [Docking Window] ⇨ [Watch1] from the menu of the motion control setting function.

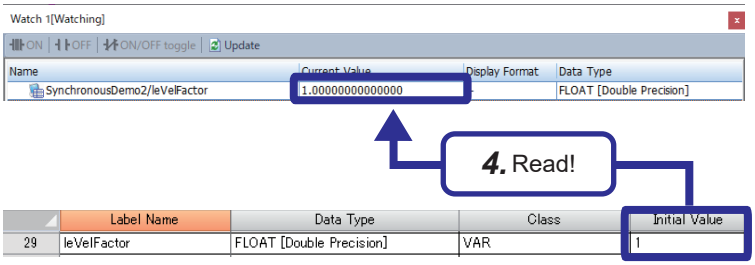


2. Enter "SynchronousDemo2/leVelFactor" as the name of Watch 1 to register the label of the velocity override factor.



3. Click [Online] ⇨ [Watch] ⇨ [Start Watching] from the menu of the motion control setting function.

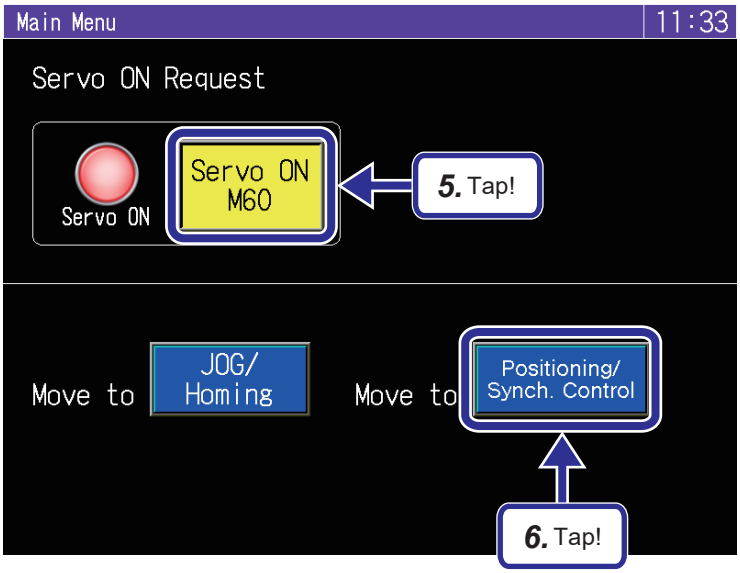
A



4. The default value "1" for the local label will be read.

Point

If the current value is 0, leVelFactor ×leVelocity will become 0 mm/s, and thus axis 1 will not move.

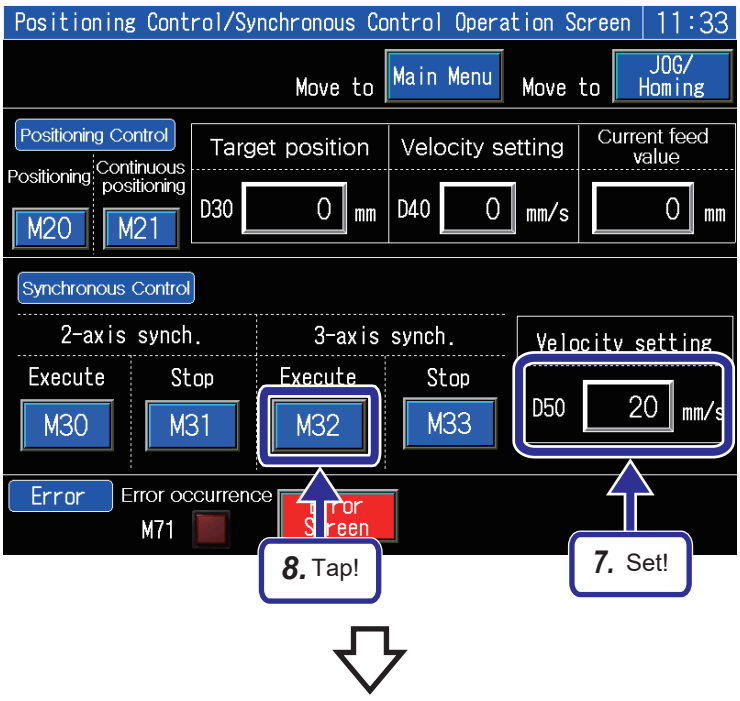


5. Turn on the servo.

6. Tap the [Positioning/Synch. Control] button on the main menu screen.

Point

Always perform homing before synchronous control.



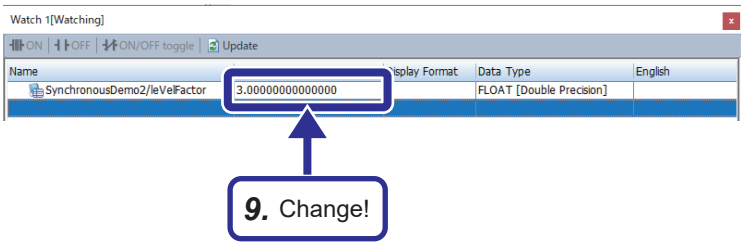
7. Set the desired velocity in the velocity setting "D50".

Point

At this time, set a velocity so that the value obtained by multiplying the velocity by the override factor does not exceed 100 mm/s.

8. Tap the [M32] button for 3-axis synchronous execution.

In the example shown on the left, 20 mm/s (leVelocity) is multiplied by 1 (leVelFactor), and thus the synchronous control is executed at the velocity set with the GOT.



9. Change the current value of "SynchronousDemo2/leVelFactor".

In this case, 20 mm/s (leVelocity) is multiplied by 3 (leVelFactor), and thus the synchronous control is executed at the velocity of 60 mm/s.

Point

- Always set a factor so that leVelFactor ×leVelocity is smaller than 100 mm/s.
- The current value of leVelFactor can be changed during synchronous control.

10. The velocity of axis 2 and axis 3 is synchronized with the changed velocity of axis 1.

Check that the velocity of each axis in the demonstration machine has increased.

Precautions

Axis 1 velocity change affects other controls of the demonstration machine. Before performing operation other than velocity change after changing the velocity, delete the program added for this assignment, and then write the data to the Motion module again.

A

Appendix 2 Monitor

Axis monitor

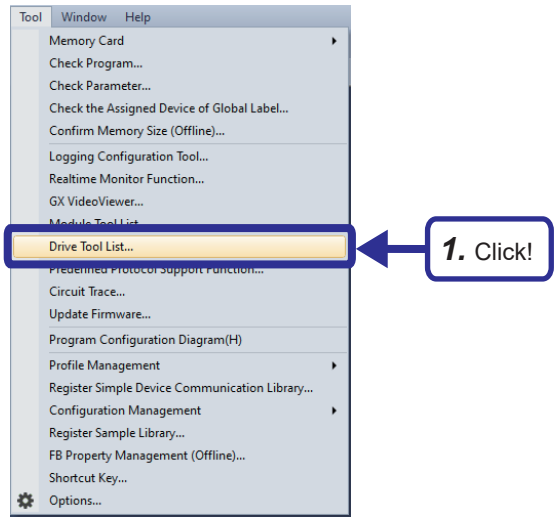
In the axis monitor, the current values and error codes of all operation axes are monitored and displayed.
This monitor shows the current values and error occurrence while the system is running.

How to display

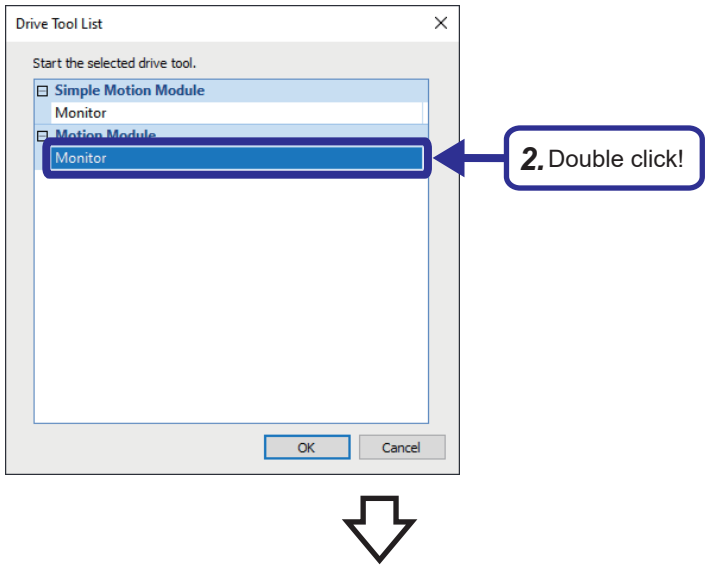
The axis monitor window can be displayed as follows.

■ Display from GX Works3

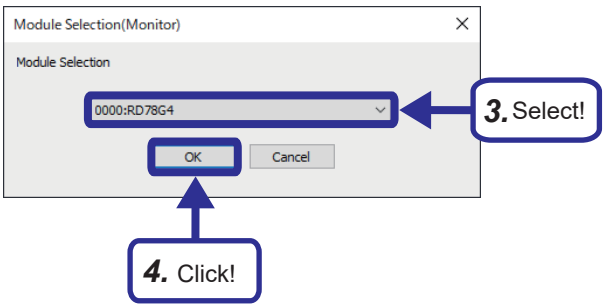
Operating procedure



1. From the menu of GX Works3, click [Tool] ⇨[Drive Tool List].



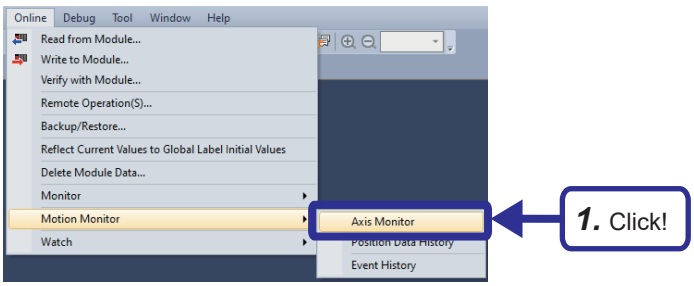
2. The "Drive Tool List" window appears. Double-click [Monitor] under "Motion Module".



3. The "Module Selection(Monitor)" window appears. Select a Motion module (for example, 0000:RD78G4).
4. Click the [OK] button.

■ Display from the motion control setting function

Operating procedure

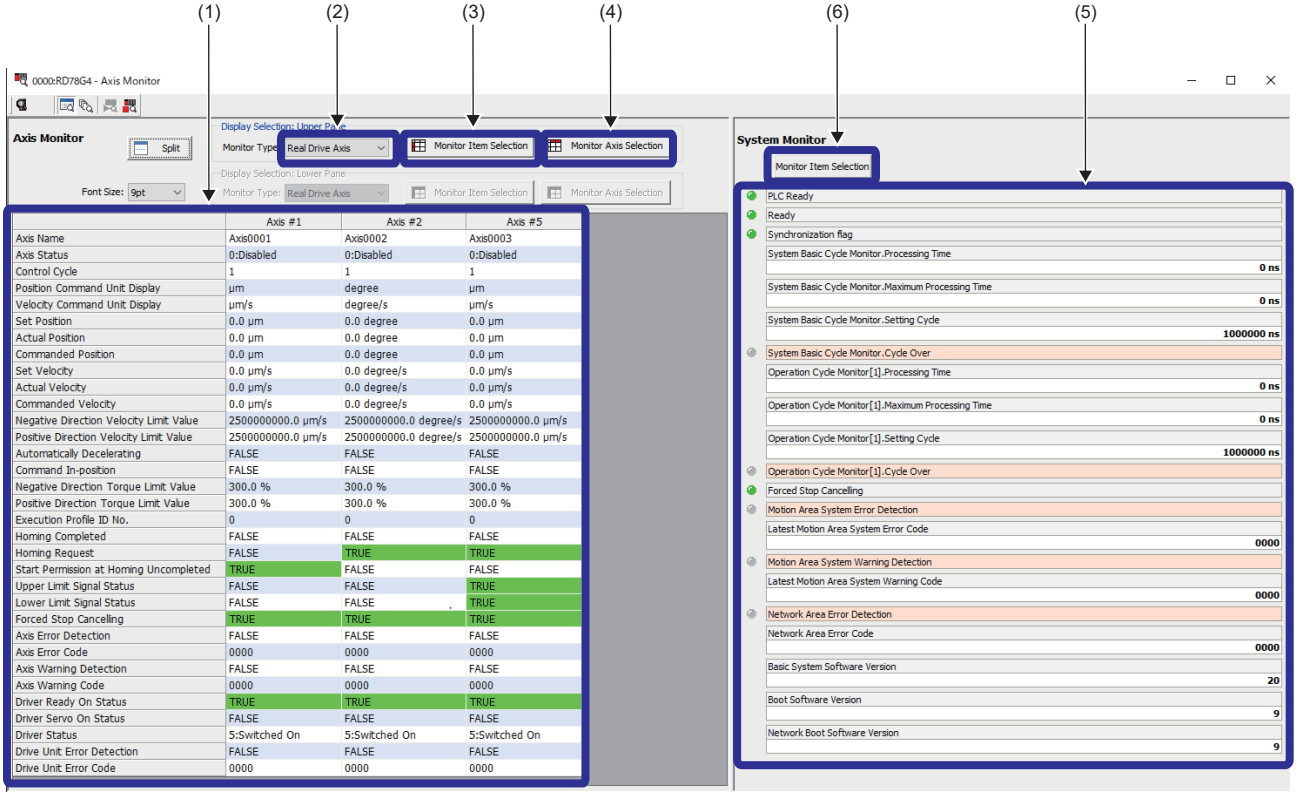


1. Click [Online] ⇨[Motion Monitor] ⇨ [Axis Monitor] from the menu of the motion control setting function.

A

Displayed items

The following shows the display of the axis monitor.



	Axis #1	Axis #2	Axis #5
Axis Name	Axis0001	Axis0002	Axis0003
Axis Status	0:Disabled	0:Disabled	0:Disabled
Control Cycle	1	1	1
Position Command Unit Display	µm	degree	µm
Velocity Command Unit Display	µm/s	degree/s	µm/s
Set Position	0.0 µm	0.0 degree	0.0 µm
Actual Position	0.0 µm	0.0 degree	0.0 µm
Commanded Position	0.0 µm	0.0 degree	0.0 µm
Set Velocity	0.0 µm/s	0.0 degree/s	0.0 µm/s
Actual Velocity	0.0 µm/s	0.0 degree/s	0.0 µm/s
Commanded Velocity	0.0 µm/s	0.0 degree/s	0.0 µm/s
Negative Direction Velocity Limit Value	2500000000.0 µm/s	2500000000.0 degree/s	2500000000.0 µm/s
Positive Direction Velocity Limit Value	2500000000.0 µm/s	2500000000.0 degree/s	2500000000.0 µm/s
Automatically Decelerating	FALSE	FALSE	FALSE
Command In-position	FALSE	FALSE	FALSE
Negative Direction Torque Limit Value	300.0 %	300.0 %	300.0 %
Positive Direction Torque Limit Value	300.0 %	300.0 %	300.0 %
Execution Profile ID No.	0	0	0
Homing Completed	FALSE	FALSE	FALSE
Homing Request	FALSE	TRUE	TRUE
Start Permission at Homing Uncompleted	TRUE	FALSE	FALSE
Upper Limit Signal Status	FALSE	FALSE	TRUE
Lower Limit Signal Status	FALSE	FALSE	TRUE
Forced Stop Cancelling	TRUE	TRUE	TRUE
Axis Error Detection	FALSE	FALSE	FALSE
Axis Error Code	0000	0000	0000
Axis Warning Detection	FALSE	FALSE	FALSE
Axis Warning Code	0000	0000	0000
Driver Ready On Status	TRUE	TRUE	TRUE
Driver Servo On Status	FALSE	FALSE	FALSE
Driver Status	5:Switched On	5:Switched On	5:Switched On
Drive Unit Error Detection	FALSE	FALSE	FALSE
Drive Unit Error Code	0000	0000	0000

Displayed items

No.	Name	Description
(1)	Items displayed in the axis monitor	Displays the monitored items of the axis selected in "Monitor Type."
(2)	Monitor Type	Select the type of the axis to be monitored.
(3)	Monitor Item Selection	Used to add or delete the monitored items displayed in the axis monitor.
(4)	Monitor Axis Selection	Used to add or delete the monitored axes displayed in the axis monitor.
(5)	Items displayed in the system monitor	Displays the monitored items of the system.
(6)	Monitor Item Selection	Used to add or delete the monitored items displayed in the system monitor.

Program monitor

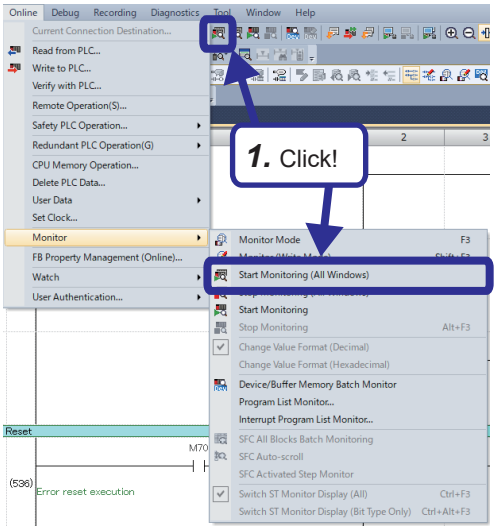
By using the monitor function, execution programs can be displayed on the program editor.

How to display

The program monitor window can be displayed as follows.

■ PLC CPU

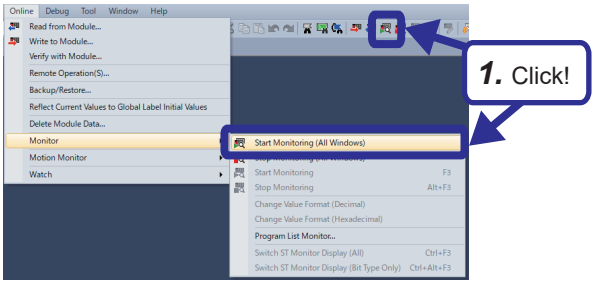
Operating procedure



1. From the menu of GX Works3, click [Online] ⇨[Monitor] ⇨ [Start Monitoring (All Windows)], or click the "Start Monitoring (All Windows)" icon on the toolbar.

■ Motion module

Operating procedure



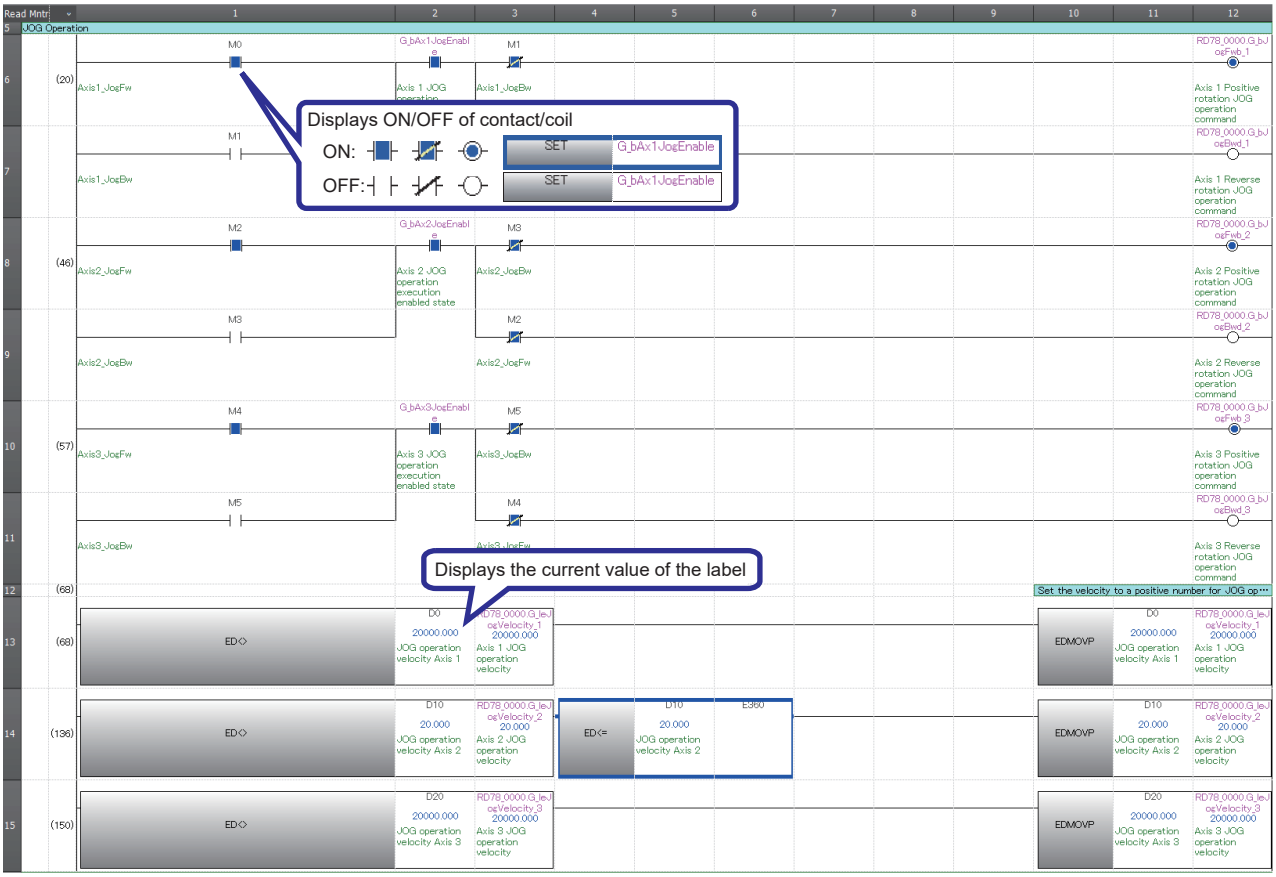
1. From the menu of the Motion Control Setting Function, click [Online] ⇨[Monitor] ⇨ [Start Monitoring (All Windows)], or click the "Start Monitoring (All Windows)" icon on the toolbar.

A

Displayed items

The following shows the display of the program monitor.

- PLC CPU



- Motion module

```
1  //------Servo ON/JOG operation------
2  //Switch to the axis operation enabled state
3  MCv_AllPower_1(
4      Enable:= TRUE ,
5      ServoON:= G_bSVONCMD ,
6      Busy=> bPowerBussy
7  );
8
9  //Axis 1: Velocity setti                                    leration, de
10 IF (G_leJogVelocity_1 <>
11     leJogVelocity_1   :=
12     leJogAcceleration_1
13     leJogDeceleration_1
14     leJogJerk_1
15 END_IF;
16
17 //Axis 1: JOG operation ex ution
18 MCv_Jog_1(
19     Axis:= Axis0001.AxisRef ,
20     JogForward:= G_bJogFwb_1 ,
21     JogBackward:= G_bJogBwd_1 ,
22     Velocity:= leJogVelocity_1 ,
23     Acceleration:= leJogAcceleration_1 ,
24     Deceleration:= leJogDeceleration_1 ,
25     Jerk:= leJogJerk_1 ,
26     Busy=> G_bJogBusy_1 ,
27     Error=> bJogError_1
28 );
29
30 //Axis 2: Velocity setting for JOG operation (velocity, acceleration, de
31 IF (G_leJogVelocity_2 <> leJogVelocity_2) THEN
32     leJogVelocity_2   := G_leJogVelocity_2;
33     leJogAcceleration_2 := G_leJogVelocity_2 * 2.0;
34     leJogDeceleration_2 := G_leJogVelocity_2 * 2.0;
35     leJogJerk_2         := G_leJogVelocity_2 * 4.0;
36 END_IF;
```

Displays TRUE/FALSE of bit type labels and bit devices
TRUE: JogForward
FALSE: JogBackward

Displays the value stored in the word device

```
G_leJogVelocity_1 = 4000.000; leJogVelocity_1 = 4000.000;
leJogVelocity_1 = 4000.000; G_leJogVelocity_1 = 4000.000;
leJogAcceleration_1 = 8000.000; G_leJogVelocity_1 = 4000.000;
leJogDeceleration_1 = 8000.000; G_leJogVelocity_1 = 4000.000;
leJogJerk_1 = 16000.000; G_leJogVelocity_1 = 4000.000;

MCv_Jog_1.Velocity = 4000.000; leJogVelocity_1 = 4000.000;
MCv_Jog_1.Acceleration = 8000.000; leJogAcceleration_1 = 8000.000;
MCv_Jog_1.Deceleration = 8000.000; leJogDeceleration_1 = 8000.000;
MCv_Jog_1.Jerk = 16000.000; leJogJerk_1 = 16000.000;

G_leJogVelocity_2 = 4.000; leJogVelocity_2 = 4.000;
leJogVelocity_2 = 4.000; G_leJogVelocity_2 = 4.000;
leJogAcceleration_2 = 8.000; G_leJogVelocity_2 = 4.000;
leJogDeceleration_2 = 8.000; G_leJogVelocity_2 = 4.000;
leJogJerk_2 = 16.000; G_leJogVelocity_2 = 4.000;
```

Appendix 3 Monitor Event History

The event history provides detailed information when an error occurs.
The date and time of occurrence recorded in the event history is synchronized with the date and time of alarm occurrence recorded in the servo amplifier.

How to display

The event history window can be displayed as follows.

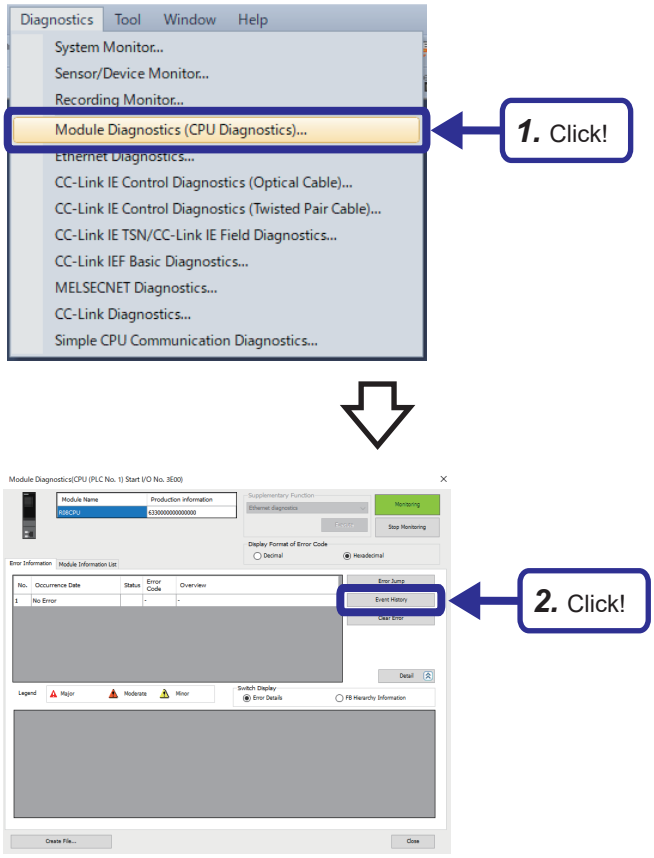
■ Display from GX Works3

There are following two ways to display the event history from GX Works3.

- Click [Diagnostics] ⇨[System Monitor] ⇨ [Event History] from the menu of GX Works3.
- Click [Diagnostics] ⇨[Module Diagnostics (CPU Diagnostics)] ⇨ [Event History] from the menu of GX Works3.

The following describes the procedure using [Module Diagnostics (CPU Diagnostics)].

Operating procedure



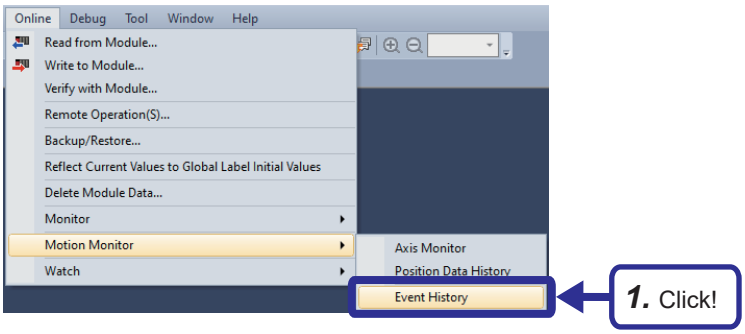
1. Click [Diagnostics] ⇨[Module Diagnostics (CPU Diagnostics)] from the menu of GX Works3.

2. The "Module Diagnostics" window appears. Click the [Event History] button.

A

■ Display from the motion control setting function

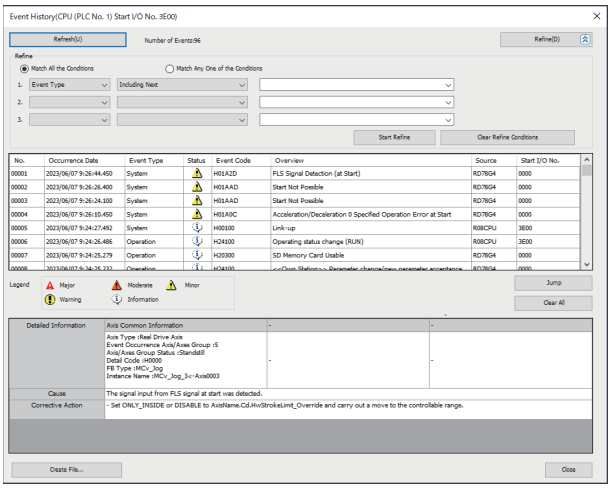
Operating procedure



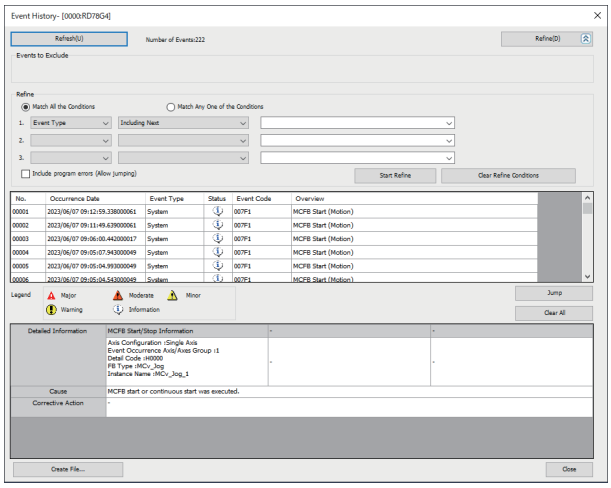
1. Click [Online] ⇨[Motion Monitor] ⇨ [Event History] from the menu of the motion control setting function.

Displayed items

<Display from MELSOFT GX Works3>



<Display from the motion control setting function>

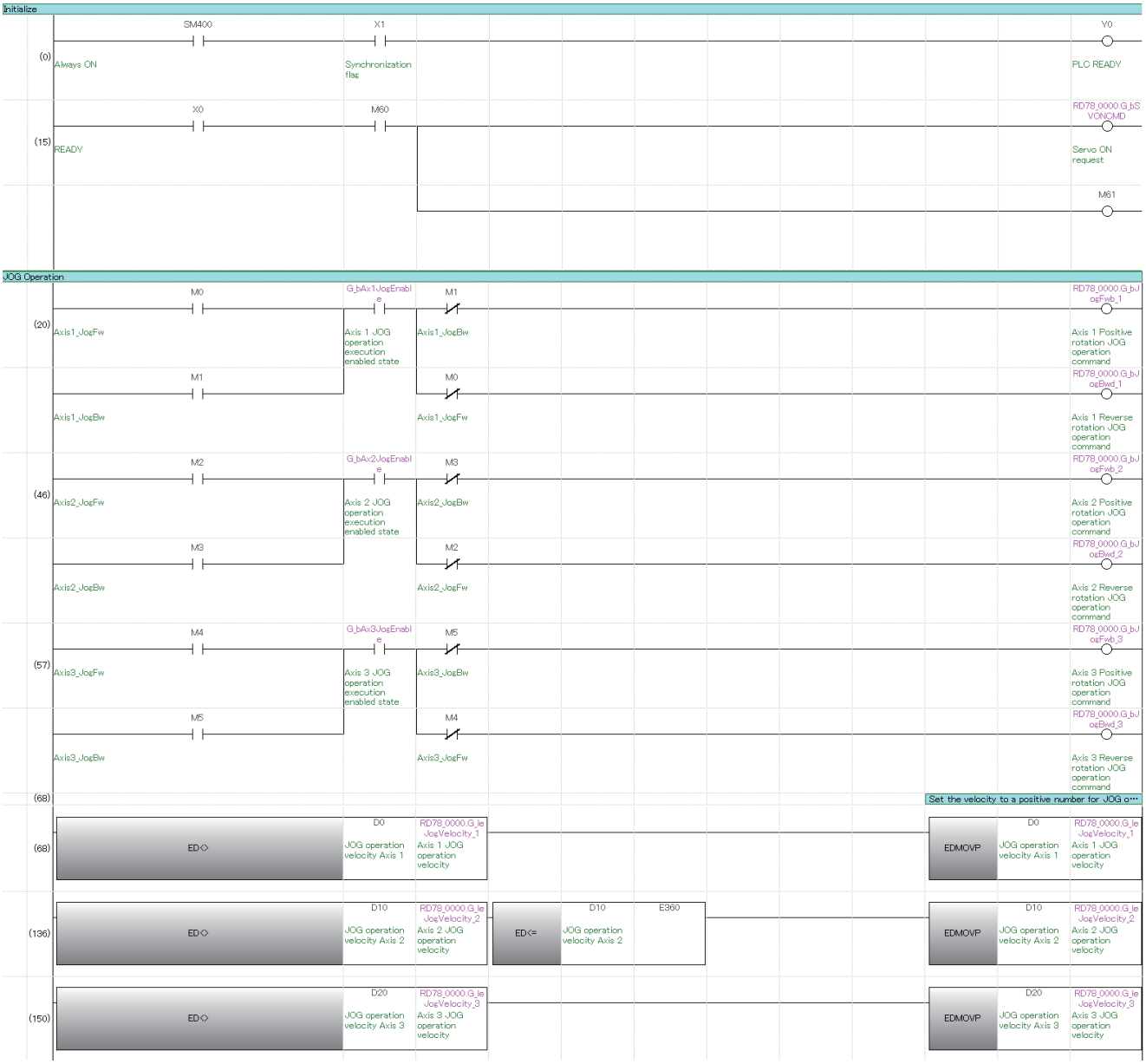


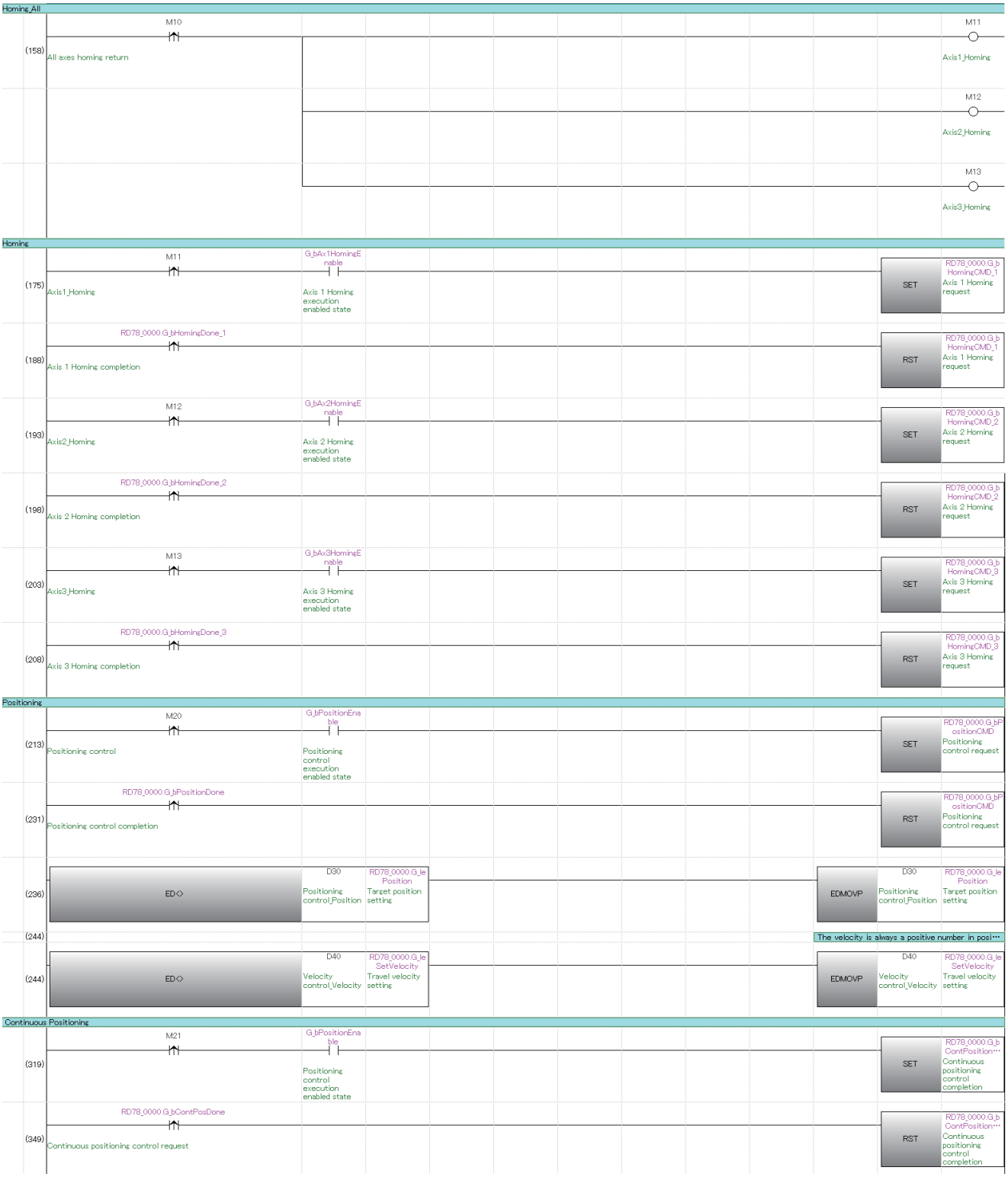
Appendix 4 Sequence Programs

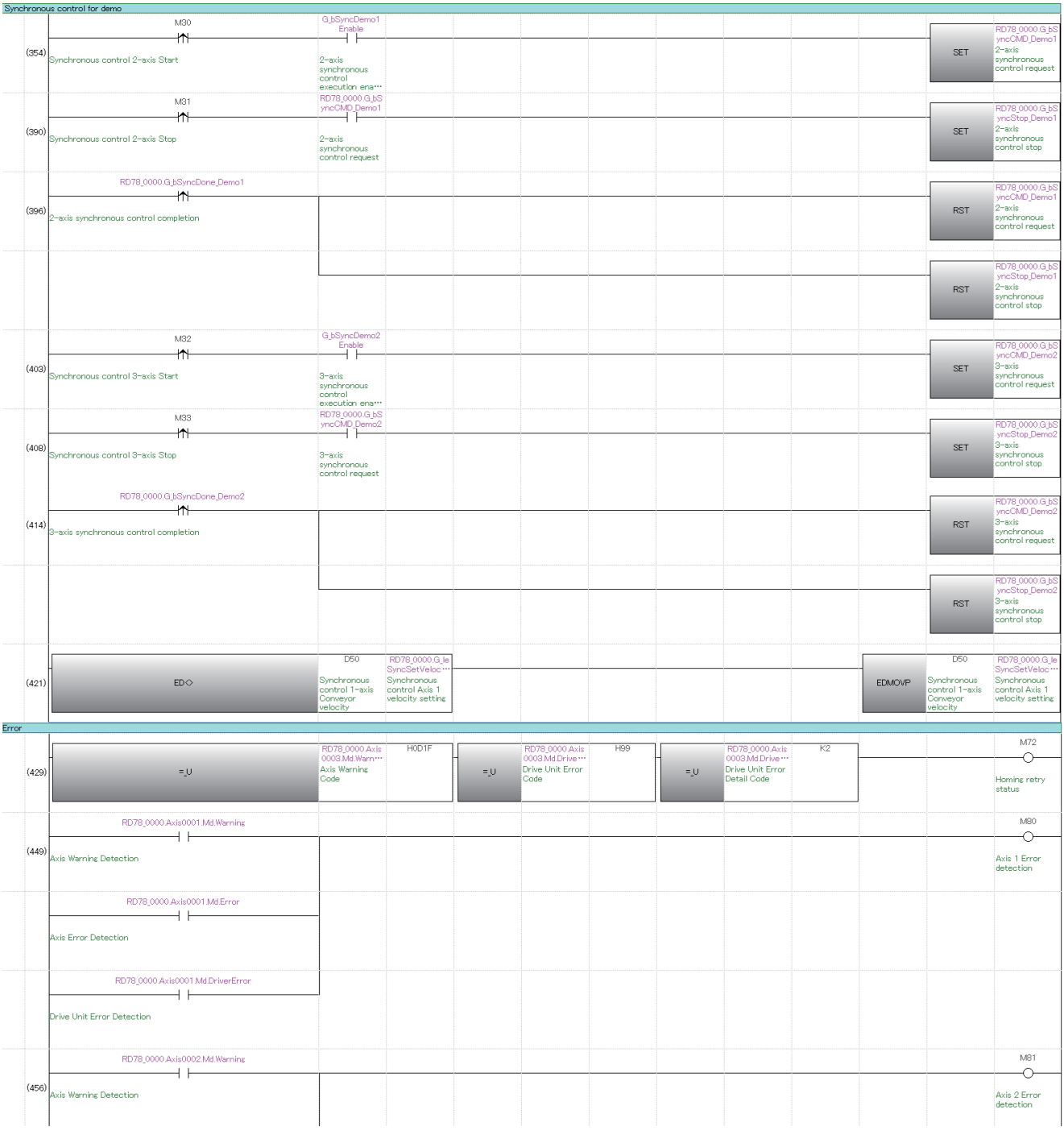
The following shows the sequence programs for the PLC CPU in the project "school_Motion.gx3".

Motion

"Motion" is a program that requests various types of motion control such as running the CPU module and servo amplifier, JOG operation, and positioning control, displays the error codes of each axis on the GOT screen, and resets the error codes.

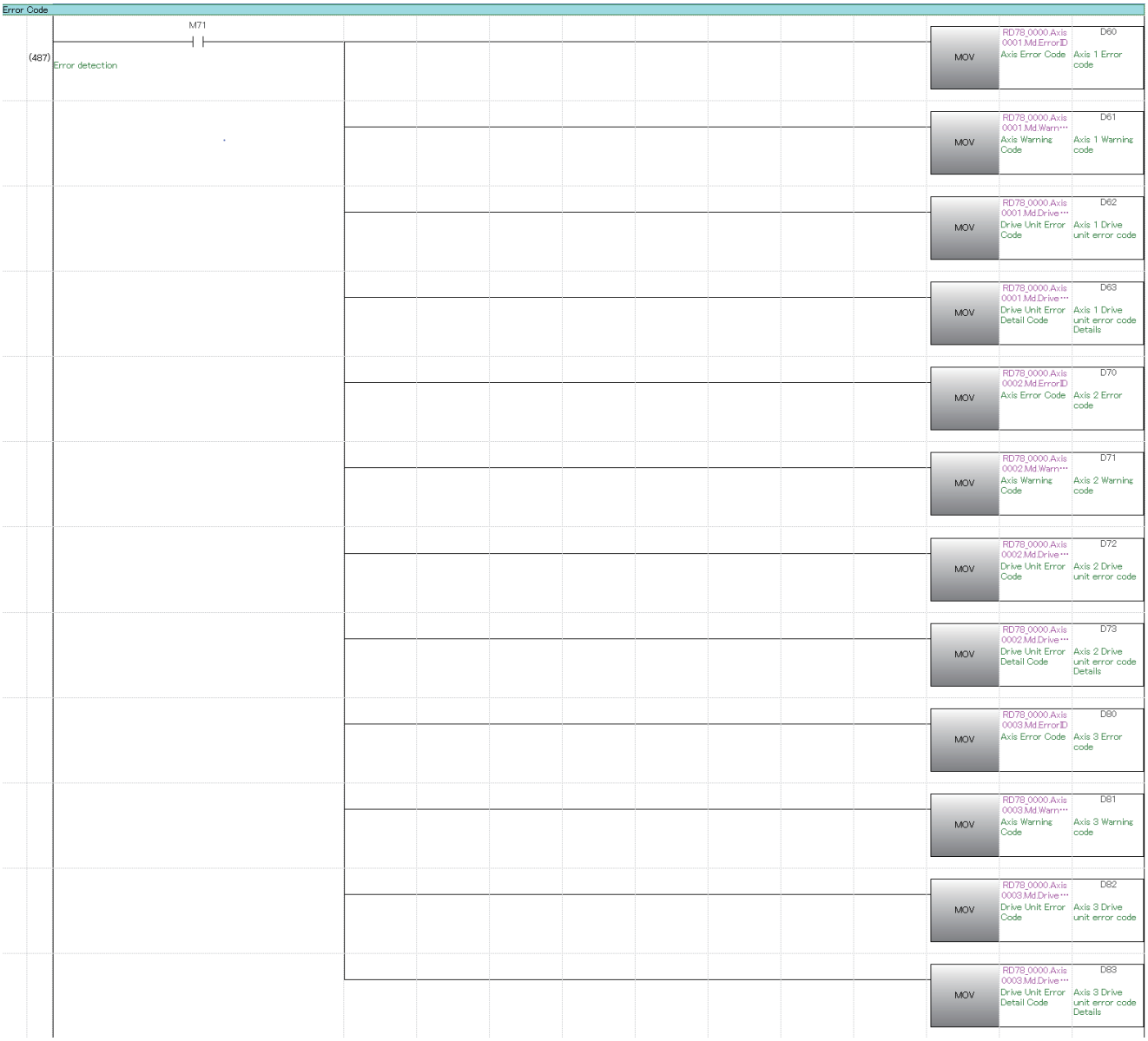



Homing_All
M10
(158)
All axes homing return
M11
Axis1_Homing
M12
Axis2_Homing
M13
Axis3_Homing
Homing
M11
G_bAx1HomingEnable
(175)
Axis1_Homing
Axis 1 Homing execution enabled state
SET
RD78_0000.G_bHomingCMD_1
Axis 1 Homing request
RD78_0000.G_bHomingDone_1
(188)
Axis 1 Homing completion
RST
RD78_0000.G_bHomingCMD_1
Axis 1 Homing request
M12
G_bAx2HomingEnable
(193)
Axis2_Homing
Axis 2 Homing execution enabled state
SET
RD78_0000.G_bHomingCMD_2
Axis 2 Homing request
RD78_0000.G_bHomingDone_2
(198)
Axis 2 Homing completion
RST
RD78_0000.G_bHomingCMD_2
Axis 2 Homing request
M13
G_bAx3HomingEnable
(203)
Axis3_Homing
Axis 3 Homing execution enabled state
SET
RD78_0000.G_bHomingCMD_3
Axis 3 Homing request
RD78_0000.G_bHomingDone_3
(208)
Axis 3 Homing completion
RST
RD78_0000.G_bHomingCMD_3
Axis 3 Homing request
Positioning
M20
G_bPositionEnable
(213)
Positioning control
Positioning control execution enabled state
SET
RD78_0000.G_bPositionCMD
Positioning control request
RD78_0000.G_bPositionDone
(231)
Positioning control completion
RST
RD78_0000.G_bPositionCMD
Positioning control request
(236)
ED◇
D30
Positioning control_Position
RD78_0000.G_lePosition
Target position setting
EDMOVP
D30
Positioning control_Position
RD78_0000.G_lePosition
Target position setting
(244)
The velocity is always a positive number in posi…
(244)
ED◇
D40
Velocity control_Velocity
RD78_0000.G_leSetVelocity
Travel velocity setting
EDMOVP
D40
Velocity control_Velocity
RD78_0000.G_leSetVelocity
Travel velocity setting
Continuous Positioning
M21
G_bPositionEnable
(319)
Positioning control execution enabled state
SET
RD78_0000.G_bContPosition…
Continuous positioning control completion
RD78_0000.G_bContPosDone
(349)
Continuous positioning control request
RST
RD78_0000.G_bContPosition…
Continuous positioning control completion

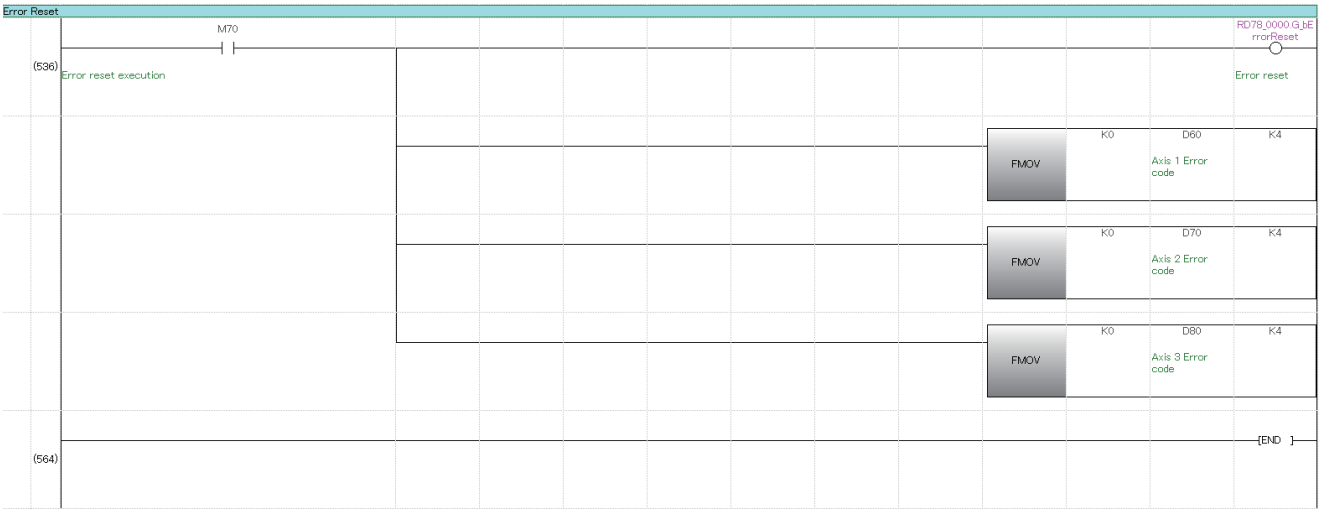

Synchronous control for demo
M30
G_bSyncDemo1 Enable
(354)
Synchronous control 2-axis Start
2-axis synchronous control execution ena…
SET
RD78_0000.G_bSyncCMD_Demo1 2-axis synchronous control request
M31
RD78_0000.G_bSyncCMD_Demo1
(390)
Synchronous control 2-axis Stop
2-axis synchronous control request
SET
RD78_0000.G_bSyncStop_Demo1 2-axis synchronous control stop
RD78_0000.G_bSyncDone_Demo1
(396)
2-axis synchronous control completion
RST
RD78_0000.G_bSyncCMD_Demo1 2-axis synchronous control request
RST
RD78_0000.G_bSyncStop_Demo1 2-axis synchronous control stop
M32
G_bSyncDemo2 Enable
(403)
Synchronous control 3-axis Start
3-axis synchronous control execution ena…
SET
RD78_0000.G_bSyncCMD_Demo2 3-axis synchronous control request
M33
RD78_0000.G_bSyncCMD_Demo2
(408)
Synchronous control 3-axis Stop
3-axis synchronous control request
SET
RD78_0000.G_bSyncStop_Demo2 3-axis synchronous control stop
RD78_0000.G_bSyncDone_Demo2
(414)
3-axis synchronous control completion
RST
RD78_0000.G_bSyncCMD_Demo2 3-axis synchronous control request
RST
RD78_0000.G_bSyncStop_Demo2 3-axis synchronous control stop
(421)
ED◇
D50
Synchronous control 1-axis Conveyor velocity
RD78_0000.G_leSyncSetVeloc…
Synchronous control Axis 1 velocity setting
EDMOVP
D50
Synchronous control 1-axis Conveyor velocity
RD78_0000.G_leSyncSetVeloc…
Synchronous control Axis 1 velocity setting
Error
(429)
=_U
RD78_0000.Axis0003.Md.Warn…
Axis Warning Code
H0D1F
=_U
RD78_0000.Axis0003.Md.Drive…
Drive Unit Error Code
H99
=_U
RD78_0000.Axis0003.Md.Drive…
Drive Unit Error Detail Code
K2
M72
Homing retry status
RD78_0000.Axis0001.Md.Warning
(449)
Axis Warning Detection
M80
Axis 1 Error detection
RD78_0000.Axis0001.Md.Error
Axis Error Detection
RD78_0000.Axis0001.Md.DriverError
Drive Unit Error Detection
RD78_0000.Axis0002.Md.Warning
(456)
Axis Warning Detection
M81
Axis 2 Error detection

A


RD78_0000.Axis0002.Md.Error
Axis Error Detection
RD78_0000.Axis0002.Md.DriverError
Drive Unit Error Detection
RD78_0000.Axis0003.Md.Warning
M72
M82
(463)
Axis Warning Detection
Homing retry status
Axis 3 Error detection
RD78_0000.Axis0003.Md.Error
Axis Error Detection
RD78_0000.Axis0003.Md.DriverError
Drive Unit Error Detection
RD78_0000.G_bJogError
M71
(471)
JOG operation error
Error detection
RD78_0000.G_bHomingError
Homing error
RD78_0000.G_bPositionError
Positioning control error
RD78_0000.G_bContPositionError
Continuous positioning control error
RD78_0000.G_bSynchro1Error
2-axis synchronous control error
RD78_0000.G_bSynchro2Error
3-axis synchronous control error
M80
Axis 1 Error detection
M81
Axis 2 Error detection
M82
Axis 3 Error detection


Error Code
M71
(487)
Error detection
MOV
RD78_0000.Axis 0001.Md.ErrorID
Axis Error Code
D60
Axis 1 Error code
MOV
RD78_0000.Axis 0001.Md.Warn…
Axis Warning Code
D61
Axis 1 Warning code
MOV
RD78_0000.Axis 0001.Md.Drive…
Drive Unit Error Code
D62
Axis 1 Drive unit error code
MOV
RD78_0000.Axis 0001.Md.Drive…
Drive Unit Error Detail Code
D63
Axis 1 Drive unit error code Details
MOV
RD78_0000.Axis 0002.Md.ErrorID
Axis Error Code
D70
Axis 2 Error code
MOV
RD78_0000.Axis 0002.Md.Warn…
Axis Warning Code
D71
Axis 2 Warning code
MOV
RD78_0000.Axis 0002.Md.Drive…
Drive Unit Error Code
D72
Axis 2 Drive unit error code
MOV
RD78_0000.Axis 0002.Md.Drive…
Drive Unit Error Detail Code
D73
Axis 2 Drive unit error code Details
MOV
RD78_0000.Axis 0003.Md.ErrorID
Axis Error Code
D80
Axis 3 Error code
MOV
RD78_0000.Axis 0003.Md.Warn…
Axis Warning Code
D81
Axis 3 Warning code
MOV
RD78_0000.Axis 0003.Md.Drive…
Drive Unit Error Code
D82
Axis 3 Drive unit error code
MOV
RD78_0000.Axis 0003.Md.Drive…
Drive Unit Error Detail Code
D83
Axis 3 Drive unit error code Details

A


Error Reset
M70
RD78_0000.G_bErrorReset
(536)
Error reset execution
Error reset
FMOV
K0
D60
K4
Axis 1 Error code
FMOV
K0
D70
K4
Axis 2 Error code
FMOV
K0
D80
K4
Axis 3 Error code
END
(564)

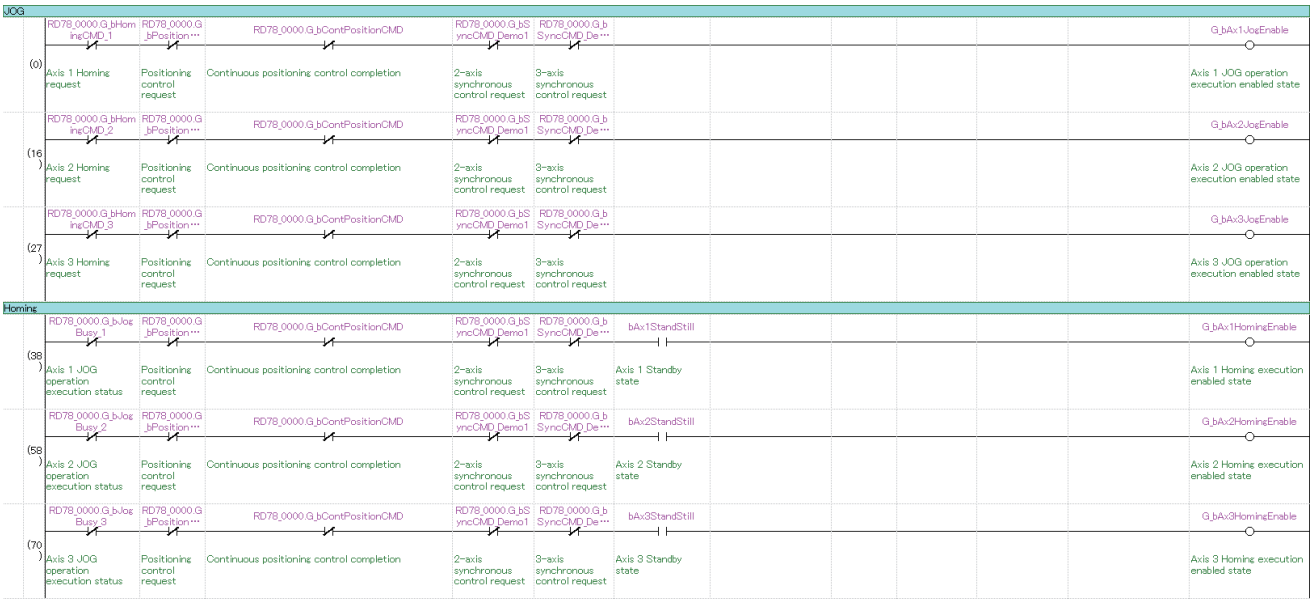
Monitor

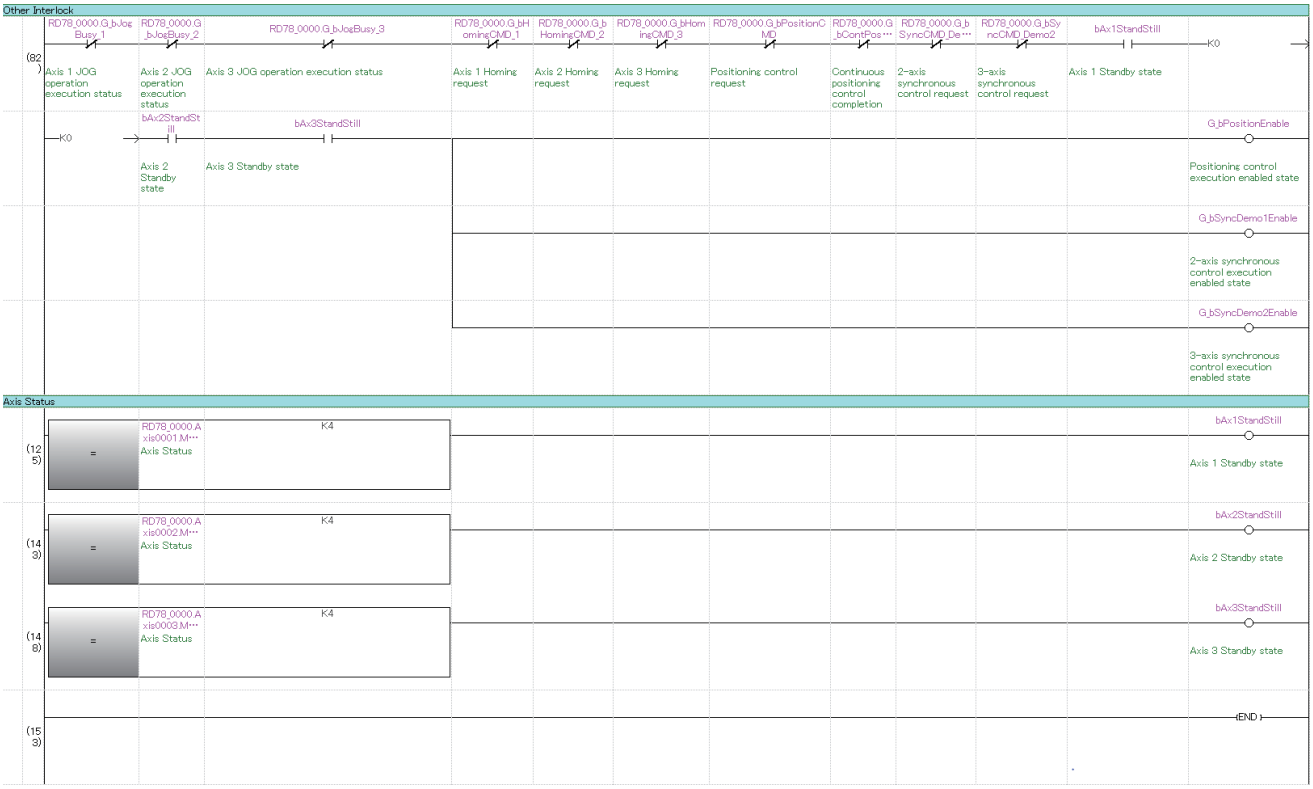
"Monitor" is a program that stores information such as the homing status, current value, and velocity of each axis in devices.

```
1  //Axis Monitor
2  //Axis0001
3  M100 := RD78_0000.Axis0001.Md.Homing_Request;
4  M101 := RD78_0000.Axis0001.Md.Homing_Complete;
5  D100:D := LREAL_TO_DINT(RD78_0000.Axis0001.Md.SetPosition);
6  D102:D := LREAL_TO_DINT(RD78_0000.Axis0001.Md.SetVelocity);
7  D110 := RD78_0000.Axis0001.Md.AxisStatus;
8
9  //Axis0002
10 M200 := RD78_0000.Axis0002.Md.Homing_Request;
11 M201 := RD78_0000.Axis0002.Md.Homing_Complete;
12 D200:D := LREAL_TO_DINT(RD78_0000.Axis0002.Md.SetPosition);
13 D202:D := LREAL_TO_DINT(RD78_0000.Axis0002.Md.SetVelocity);
14 D210 := RD78_0000.Axis0002.Md.AxisStatus;
15
16 //Axis0003
17 M300 := RD78_0000.Axis0003.Md.Homing_Request;
18 M301 := RD78_0000.Axis0003.Md.Homing_Complete;
19 D300:D := LREAL_TO_DINT(RD78_0000.Axis0003.Md.SetPosition);
20 D302:D := LREAL_TO_DINT(RD78_0000.Axis0003.Md.SetVelocity);
21 D310 := RD78_0000.Axis0003.Md.AxisStatus;
```

Interlock

"Interlock" is a program that prevents another operation command from being executed when each axis is operating.




Other Interlock
RD78_0000.G_bJogBusy_1
RD78_0000.G_bJogBusy_2
RD78_0000.G_bJogBusy_3
RD78_0000.G_bHomingCMD_1
RD78_0000.G_bHomingCMD_2
RD78_0000.G_bHomingCMD_3
RD78_0000.G_bPositionCMD
RD78_0000.G_bContPos…
RD78_0000.G_bSyncCMD_De…
RD78_0000.G_bSyncCMD_Demo2
bAx1StandStill
(82)
Axis 1 JOG operation execution status
Axis 2 JOG operation execution status
Axis 3 JOG operation execution status
Axis 1 Homing request
Axis 2 Homing request
Axis 3 Homing request
Positioning control request
Continuous positioning control completion
2-axis synchronous control request
3-axis synchronous control request
Axis 1 Standby state
K0
bAx2StandStill
bAx3StandStill
K0
Axis 2 Standby state
Axis 3 Standby state
G_bPositionEnable
Positioning control execution enabled state
G_bSyncDemo1Enable
2-axis synchronous control execution enabled state
G_bSyncDemo2Enable
3-axis synchronous control execution enabled state
Axis Status
(125)
RD78_0000.Axis0001.M…
Axis Status
K4
bAx1StandStill
Axis 1 Standby state
(143)
RD78_0000.Axis0002.M…
Axis Status
K4
bAx2StandStill
Axis 2 Standby state
(148)
RD78_0000.Axis0003.M…
Axis Status
K4
bAx3StandStill
Axis 3 Standby state
(153)
END

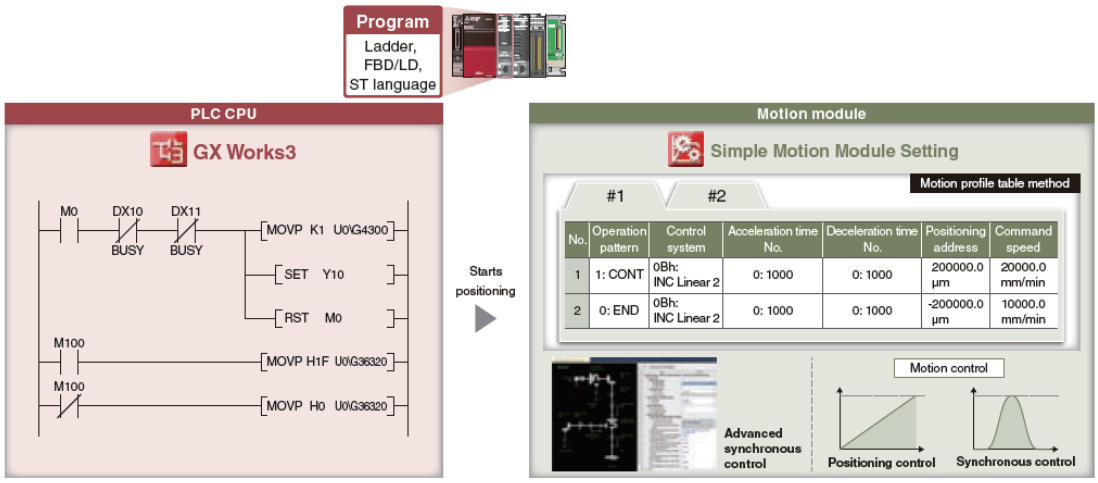
Appendix 5 Simple Motion Mode

The Simple Motion mode is an operation mode that allows the reuse of existing projects to dive servo amplifiers via CC-Link IE TSN. The Simple Motion mode is supported by the Motion module with the firmware version 16 or later.

Features

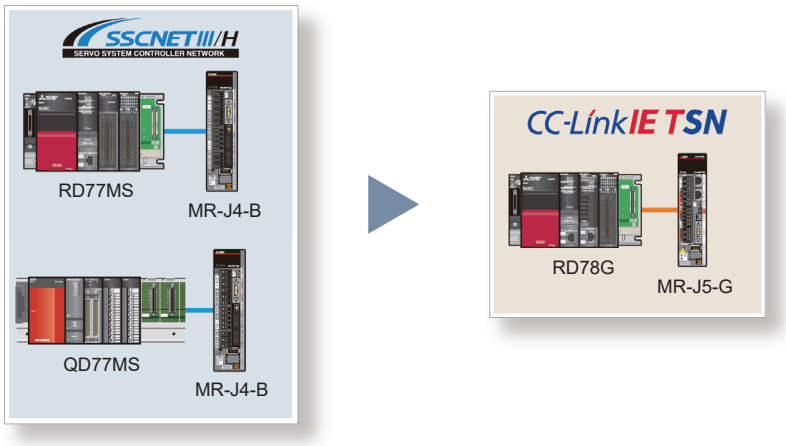
- Positioning control using the point table method and synchronous control by setting synchronization parameters allow the ease of operation.
- The PLC CPU can read/write the data of the remote devices connected via CC-Link IE TSN.
- A digital oscilloscope can be used, which enables data collection and waveform display synchronized with the motion operation cycle to help users to check operation.
- Existing projects can be utilized to reduce programming efforts.

An example of programming in Simple Motion mode



Utilization of program assets

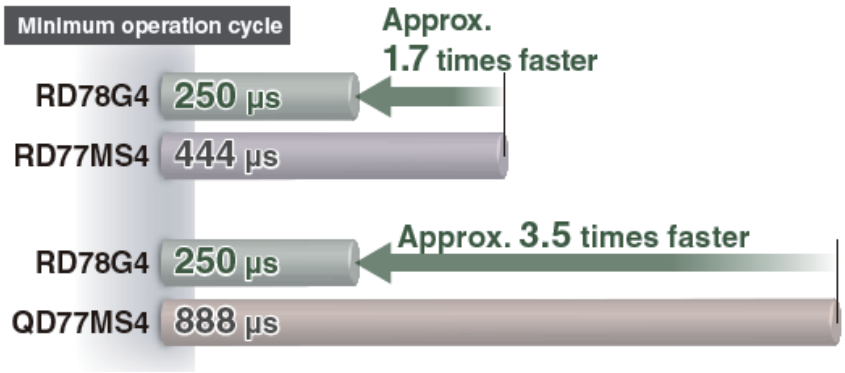
The MELSEC iQ-R/MELSEC-Q series programs and various modules used in existing systems can be utilized for the Motion module RD78G that supports the Simple Motion mode.



A

Performance

The minimum operation cycle of the Motion module in Simple Motion mode is 1.7 to 3.5 times faster than that of previous models, allowing data and I/O signals to be transferred from/to the servo amplifier at high speed. This contributes to reduction of the cycle time.



MEMO

REVISIONS

*The manual number is given on the bottom left of the back cover.

Revision date	*Manual number	Description
June 2023	SH(NA)-030390ENG-A	First edition Listed in the motion control setting function (ver. 1.042U) of the engineering tool GX Works3 Version1 (ver. 1.095Z).

Japanese manual number: SH(NA)-030387-C

Mitsubishi Programmable Controllers Training Manual
MELSEC iQ-R Motion Module

MODEL	SCHOOL-RD78G-E
SH(NA)-030390ENG-A(2307)	

MITSUBISHI ELECTRIC CORPORATION

HEAD OFFICE: TOKYO BLDG., 2-7-3, MARUNOUCHI, CHIYODA-KU, TOKYO 100-8310, JAPAN
NAGOYA WORKS: 1-14, YADA-MINAMI 5-CHOME, HIGASHI-KU, NAGOYA 461-8670, JAPAN