Servo System Controller

MELSEC iQ-R Series Motion Module Application (RD78G(H) Interpolation/Synchronous Control)

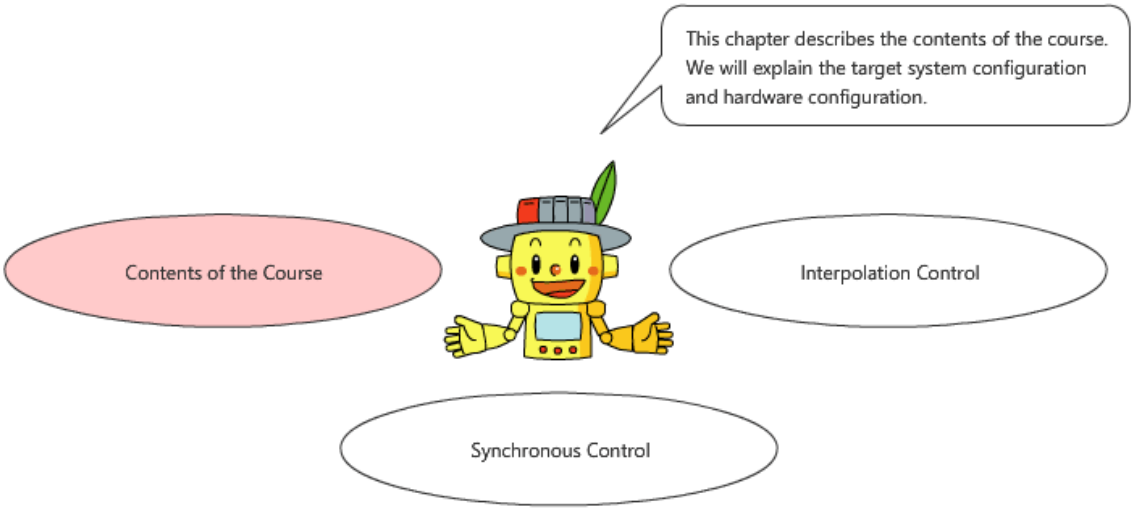
This training course is designed for anyone new to interpolation control and synchronous control using the motion control system of the MELSEC iQ-R Series Motion module.

Click the Next button at the upper right corner to go to the next page.

L(NA)00244ENG

Introduction Course Objective

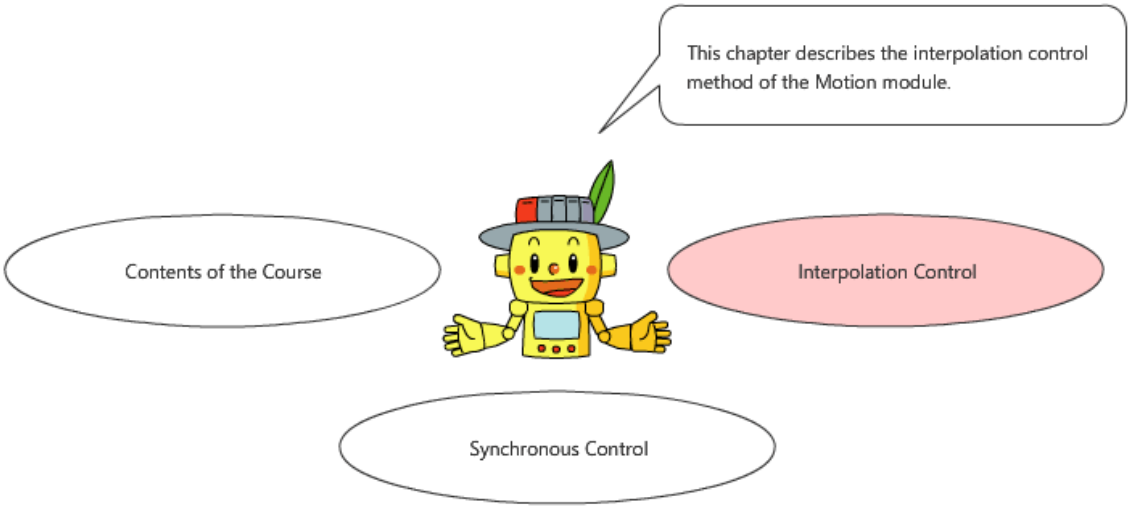
This course is designed for anyone new to interpolation control and synchronous control using the motion control system of the MELSEC iQ-R Series Motion module, to learn about system design, installation, wiring, setting, and programming.



This course is a sequel to MELSEC iQ-R Series Motion Module Basics (RD78G(H) Startup) and MELSEC iQ-R Series Motion Module Basics (RD78G(H) Position Control).
Make sure to finish the above courses before taking this course.

Introduction Course Objective

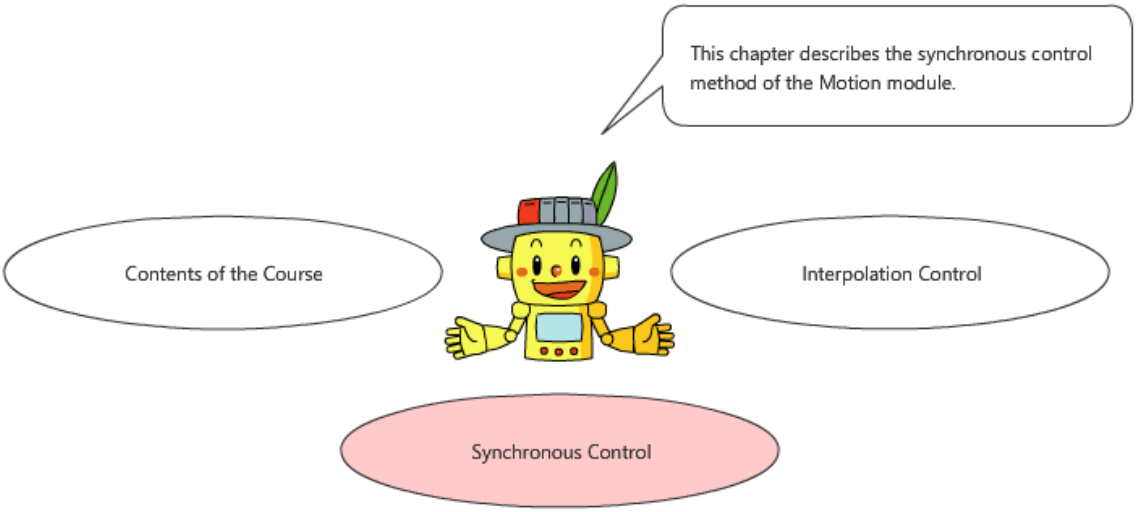
This course is designed for anyone new to interpolation control and synchronous control using the motion control system of the MELSEC iQ-R Series Motion module, to learn about system design, installation, wiring, setting, and programming.



This course is a sequel to MELSEC iQ-R Series Motion Module Basics (RD78G(H) Startup) and MELSEC iQ-R Series Motion Module Basics (RD78G(H) Position Control).
Make sure to finish the above courses before taking this course.

Introduction Course Objective

This course is designed for anyone new to interpolation control and synchronous control using the motion control system of the MELSEC iQ-R Series Motion module, to learn about system design, installation, wiring, setting, and programming.



This course is a sequel to MELSEC iQ-R Series Motion Module Basics (RD78G(H) Startup) and MELSEC iQ-R Series Motion Module Basics (RD78G(H) Position Control).
Make sure to finish the above courses before taking this course.

Introduction Course Structure

This course consists of the following chapters.
We recommend that you start from Chapter 1.

Chapter 1 Contents of the Course

This chapter describes the contents of the course. We will explain the target system configuration and hardware configuration.

Chapter 2 Interpolation Control

This chapter describes the interpolation control method of the Motion module.

Chapter 3 Synchronous Control

This chapter describes the synchronous control method of the Motion module.

Final Test

4 sections (22 questions) Passing score: 60%

Introduction How to Use This e-Learning Tool

Go to the next page	>	Go to the next page.
Back to the previous page	<	Back to the previous page.
Move to the desired page	TOC	"Table of Contents" will be displayed, enabling you to navigate to the desired page.
Exit the learning	X	Exit the learning. Window such as "Contents" screen and the learning will be closed.

Introduction Precautions for Use

■Safety precautions

When using actual products for learning purposes, please carefully read the "Safety Precautions" described in the manual of the product to be used, and pay close attention to safety and proper use.

■Precautions on this course

The screen images shown in the course may differ from your actual software depending on the version.
The following software versions are used in the course.
For the latest version of each software, check the MITSUBISHI ELECTRIC FA Global Website.

MELSOFT GX Works3	Ver.1.072A	Motion control setting function	Ver.1.015R
GX LogViewer	Ver.1.106K		
MELSOFT MR Configurator2	Ver.1.115V or later		

The firmware version 44 or later is required for the PLC CPU (version 46 or later for RD78GH).
The firmware version 14 or later is required for the Motion module.
For how to update the firmware version, refer to MITSUBISHI ELECTRIC FA Global Website or the module configuration manual.

indicates the reference manual.
This course refers to the manuals as of the following versions.
Note that the descriptions and contents may slightly differ depending on the version.

Manual name	Manual No.	Version
MELSEC iQ-R Motion Module User's Manual (Startup)	IB-0300406	D
MELSEC iQ-R Motion Module User's Manual (Application)	IB-0300411	E
MELSEC iQ-R Motion Module User's Manual (Network)	IB-0300426	D
MELSEC iQ-R Programming Manual (Motion Module Instructions, Standard Functions/Function Blocks)	IB-0300431	D
MELSEC iQ-R Programming Manual (Motion Control Function Blocks)	IB-0300533	B
MELSEC iQ-R Structured Text (ST) Programming Guide Book	SH-081483	E
MELSEC iQ-R Programming Manual (CPU Module Instructions, Standard Functions/Function Blocks)	SH-081266	X
MELSEC iQ-R CPU Module User's Manual (Application)	SH-081264	AH

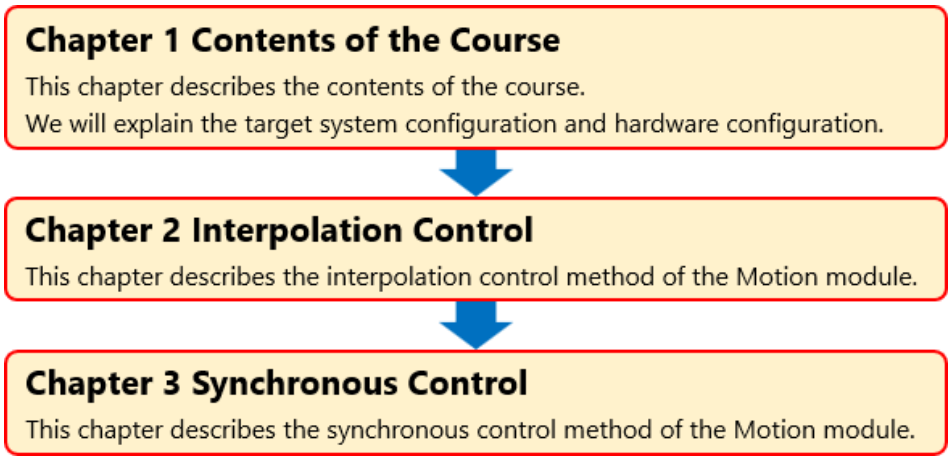
Chapter 1 Contents of the Course

Download the sample program file to be used in this course from the following.

RD78GAdvanced_Sample.zip (1.50MB)
Refer to “download” folder.

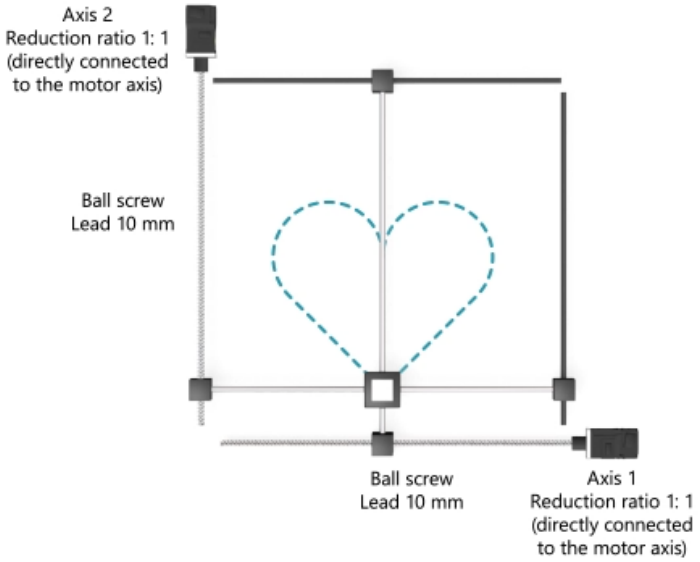
1.1 Subject of the Course

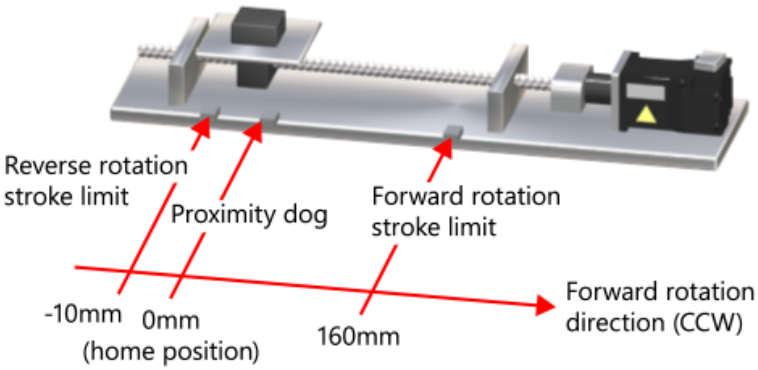
The following describes the contents of the course.



1.2 Target Machine Configuration

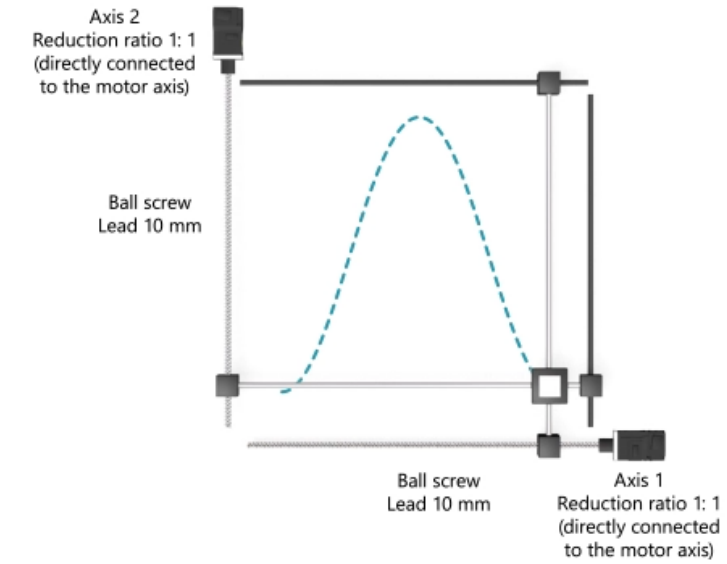
In this course, we use the machine configuration of the XY table shown below.
X-axis is represented as axis 1 (Axis0001) and Y-axis is represented as axis 2 (Axis0002).
The location of the limit switch is assumed to be the same for X-axis and Y-axis.

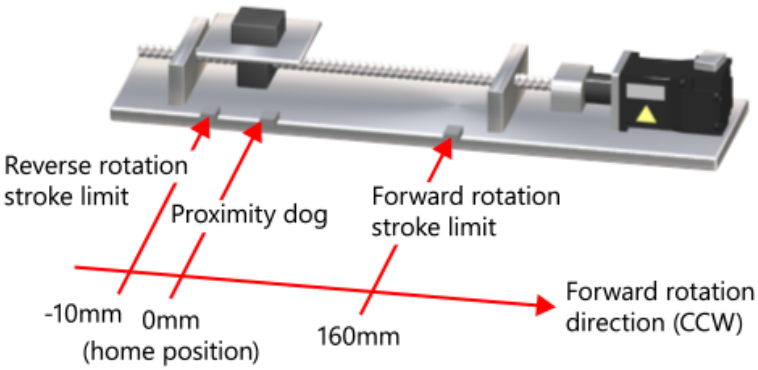




1.2 Target Machine Configuration

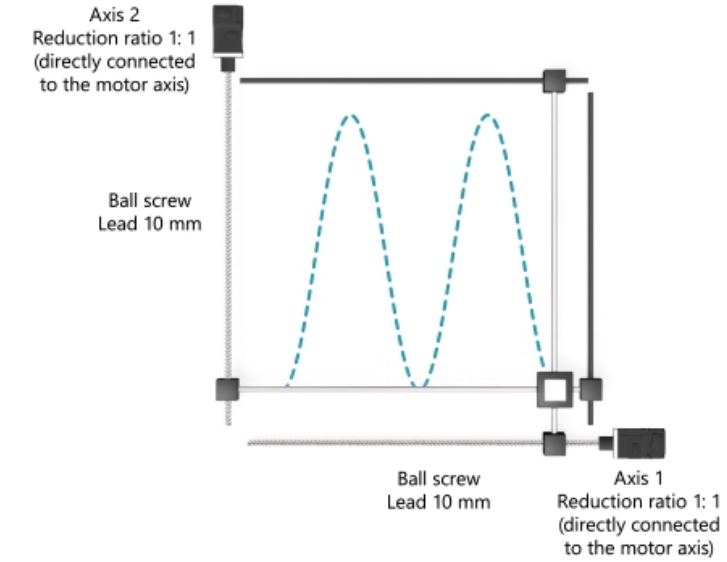
In this course, we use the machine configuration of the XY table shown below.
X-axis is represented as axis 1 (Axis0001) and Y-axis is represented as axis 2 (Axis0002).
The location of the limit switch is assumed to be the same for X-axis and Y-axis.

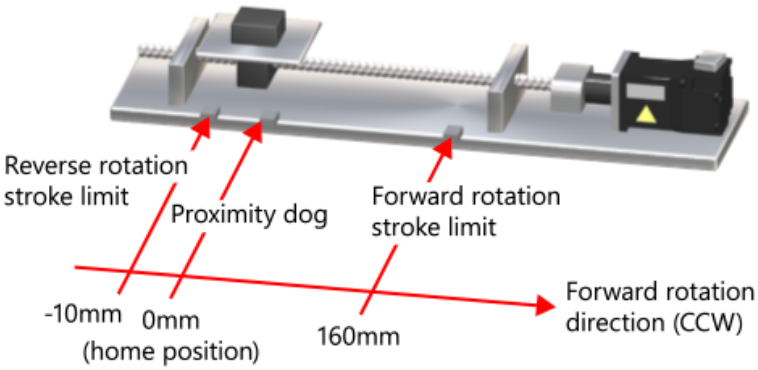




1.2 Target Machine Configuration

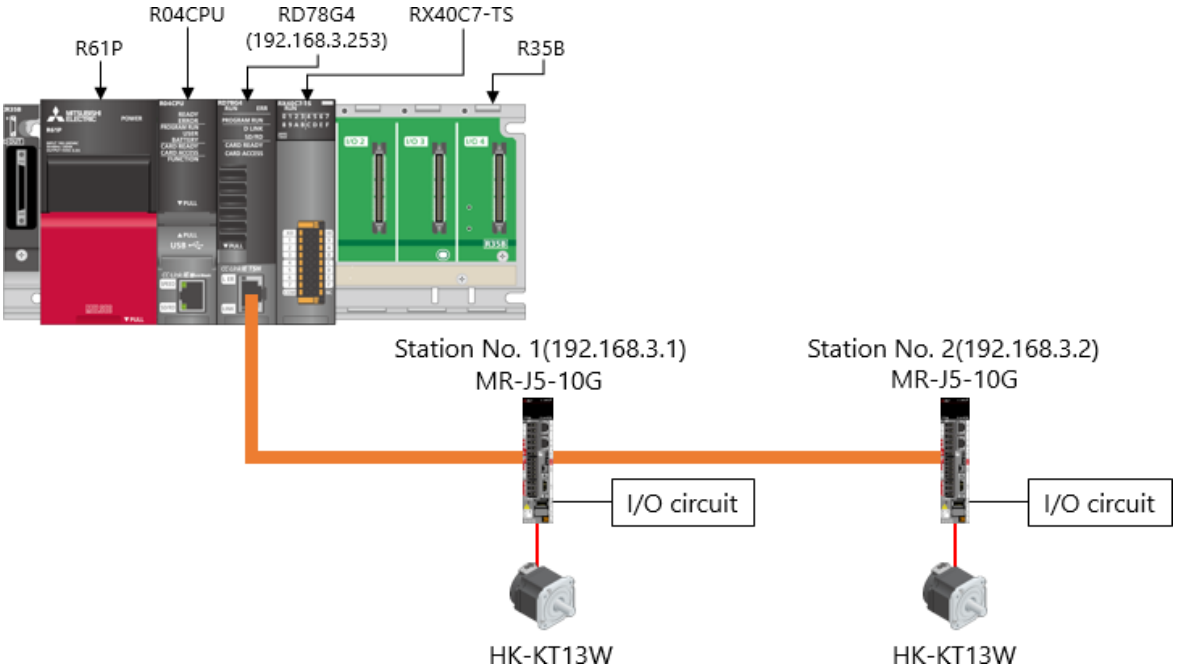
In this course, we use the machine configuration of the XY table shown below.
X-axis is represented as axis 1 (Axis0001) and Y-axis is represented as axis 2 (Axis0002).
The location of the limit switch is assumed to be the same for X-axis and Y-axis.





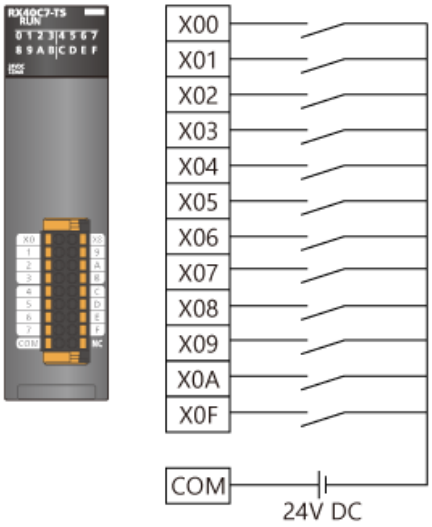
1.3 Target System Configuration

In this course, we use the following system configuration.
Add the second servo amplifier as station No. 2 to the system configuration of the Basics course (Positioning Control).



1.4 Wiring of External Circuit

Wiring of power supply for the programmable controller and servo amplifier, and the connection method of the servo motor are the same as described in the Basics course.
The following shows the external circuit wiring of the input module.



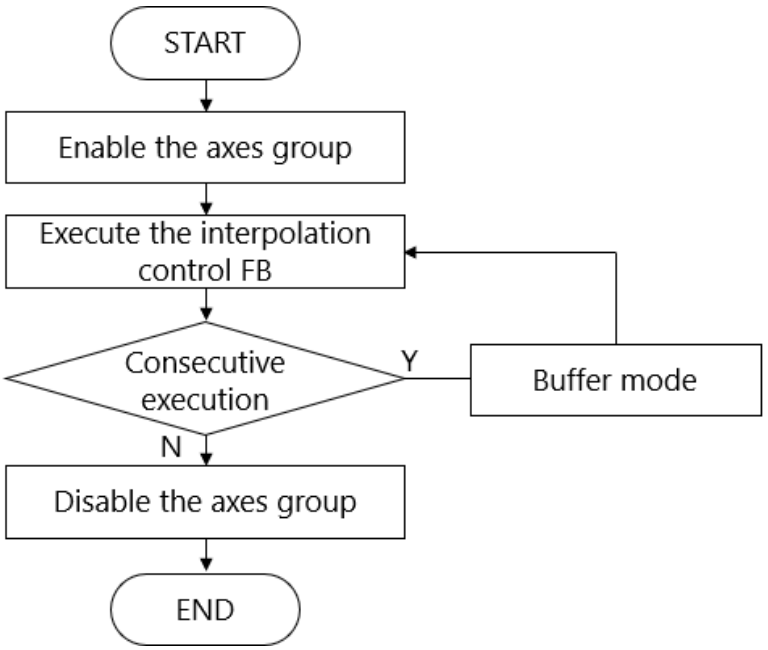
(Note 1)
X00: Servo OFF
X01: Axis 1 forward rotation JOG
X02: Axis 1 reverse rotation JOG
X03: Axis 2 forward rotation JOG
X04: Axis 2 reverse rotation JOG
X05: Axis 1 homing (proximity dog type)
X06: Axis 2 homing (proximity dog type)
X07: Virtual drive axis 1 homing (Note 2)
X08: Interpolation control start (for Chapter 2)
X09: Synchronous control 1 start (for Chapter 3)
X0A: Synchronous control 2 start (for Chapter 3)
X0F: Error reset

(Note)
1. Since the I/O No. of RX40C7-TS is 0020H, programs will use X20 to X2A and X2F.
2. The details of the virtual drive axis are described in Chapter 3.

Chapter 2 Interpolation Control

2.1 Steps of Interpolation Control

The following shows the steps of interpolation control using multiple axes.



… MC_GroupEnable is executed to enable the axes group.

… The interpolation control FB is executed.

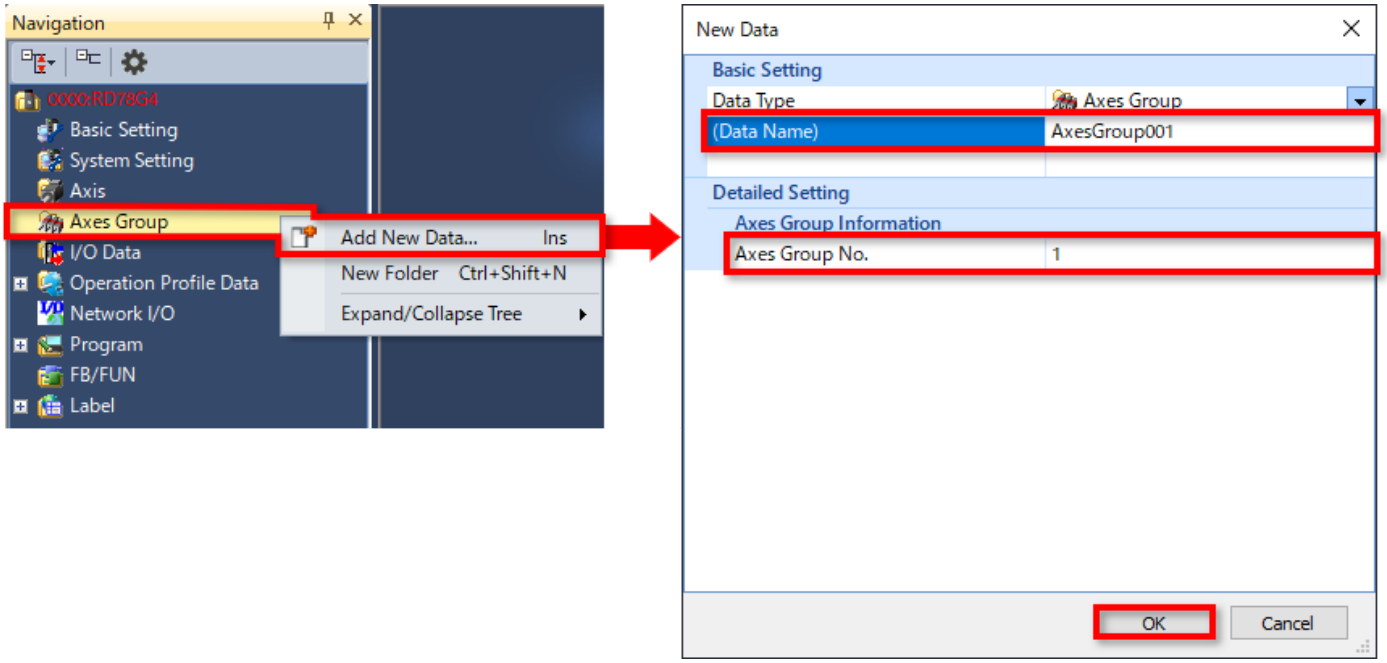
… MC_GroupDisable is executed to disable the axes group.

2.2 Axes Group

Interpolation control requires the target axes to be defined as axes groups.
For example, interpolation control for the XY table requires X-axis and Y-axis to be registered in a single axes group.

(1) Creating a new axes group

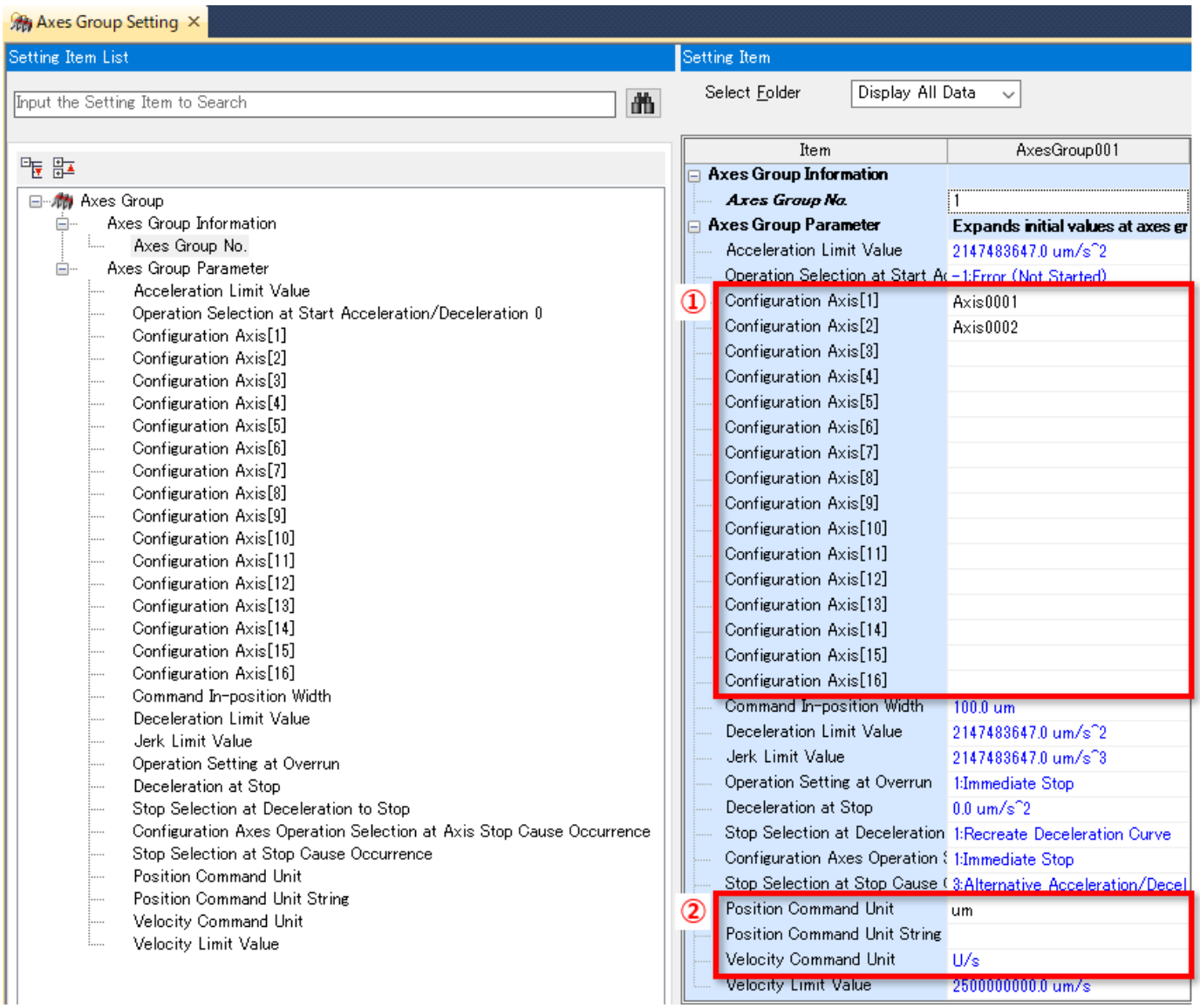
With the motion control setting function, right click "Axes group" in the Navigation window and select [Add New Data].
Enter the data name and axes group No. in the New Data window.
We use the default settings here: data name "AxesGroup001" and axes group No. "1".
Click the [OK] button after entering the information.



(2) Setting the axes group parameter

The axes group setting window appears. In this course, configure the settings as follows.

1) Under [Axes Group Parameter], register the axis names "Axis0001" and "Axis0002" to be controlled by interpolation in [Configuration Axis].
2) Under [Axes Group Parameter], set the control unit of the axes group for [Position Command Unit] and [Velocity Command Unit].
   In this course, we use the same units as the axis parameter: position command unit "um" and velocity command unit "U/s".



[Point]
An axis can be registered to multiple axes groups.
Create as many axes groups as combinations of axes for interpolation control.

2.3 FBs for Interpolation Control

2.3.1 Enabling/disabling the axes group

(1) Overview

MC_GroupEnable/MC_GroupDisable is used to enable/disable the axes group.
Before performing interpolation control, execute MC_GroupEnable to switch the axes group status ((AxesGroupName).Md.GroupStatus) from 0:GroupDisabled to 4:GroupStandby.

Type	FB name	Description
MCFB (Management FBs)	MC_GroupEnable	Enables the specified axes group and switches the group status from 0:GroupDisabled to 4:GroupStandby.
	MC_GroupDisable	Disables the specified axes group and switches the group status to 0:GroupDisabled.

2.3.1 Enabling/disabling the axes group

(2) FB specification (excerpt)

The following table shows the I/O variables of MC_GroupEnable/MC_GroupDisable.

```
MC_GroupEnable_1(
    AxesGroup   := AxesGroup001.AxesGroupRef ,
    Execute     := bGrpENReq ,
    Done    => bGrpEnDone ,
    Busy    => bGrpEnBusy ,
    Error   => bGroEnError ,
    ErrorID => uGrpErrorID
);

MC_GroupDisable_1(
    AxesGroup   := AxesGroup001.AxesGroupRef ,
    Execute     := bGrpDsblReq ,
    Done    => bGrpDsblDone ,
    Busy    => bGrpDsblBusy ,
    Error   => bGrpDsblError ,
    ErrorID => uGrpDsblErrorID
);

```

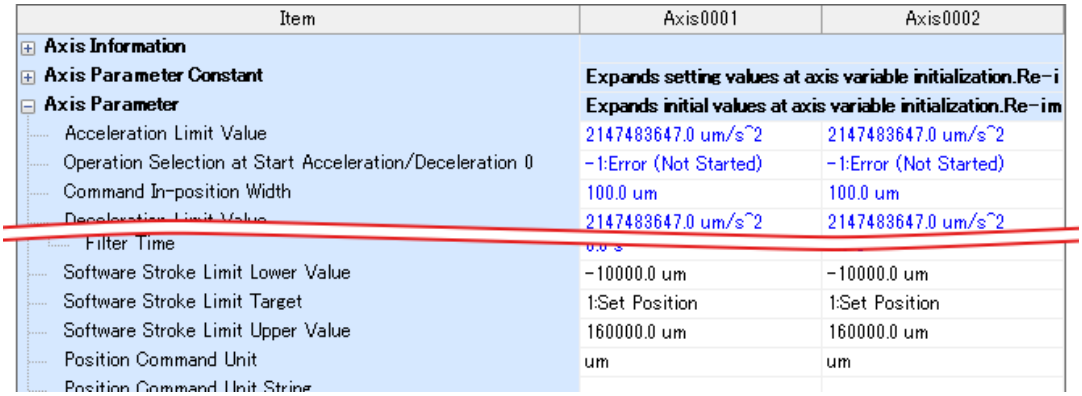
I/O variable name		Variable name	Data type	Description
I/O	Axes group information	AxesGroup	AXES_GROUP_REF	Structure of axes group information
Input	Execution command	Execute	BOOL	Executes the FB when it is TRUE.
Output	Execution completion	Done	BOOL	Indicates that the FB operation is completed.
	Executing	Busy	BOOL	Indicates that the FB operation is in progress.
	Error	Error	BOOL	Indicates that the error has occurred in the FB when it is TRUE.
	Error code	ErrorID	UINT	Returns the error code generated in the FB.

2.3.2 Interpolation operation

The Motion module is provided with the FBs for linear interpolation and circular interpolation.
The details are described in Section 2.4 and 2.5.

Type	FB name	Description
MCFB (Motion)	MCv_MoveLinearInterpolateAbsolute	Absolute value linear interpolation control
	MCv_MoveLinearInterpolateRelative	Relative value linear interpolation control
	MCv_MoveCircularInterpolateAbsolute	Absolute value circular interpolation control (Note)
	MCv_MoveCircularInterpolateRelative	Relative value circular interpolation control (Note)

(Note)
When executing the interpolation control FB in the Motion module, the software stroke limit must be enabled.
The following shows the preset items for the sample program.



Item	Axis0001	Axis0002
⊞ **Axis Information**		
⊞ **Axis Parameter Constant**	**Expands setting values at axis variable initialization.Re-i**	
⊟ **Axis Parameter**	**Expands initial values at axis variable initialization.Re-im**	
Acceleration Limit Value	2147483647.0 um/s^2	2147483647.0 um/s^2
Operation Selection at Start Acceleration/Deceleration 0	-1:Error (Not Started)	-1:Error (Not Started)
Command In-position Width	100.0 um	100.0 um
Deceleration Limit Value	2147483647.0 um/s^2	2147483647.0 um/s^2
Filter Time	0.0 s	
Software Stroke Limit Lower Value	-10000.0 um	-10000.0 um
Software Stroke Limit Target	1:Set Position	1:Set Position
Software Stroke Limit Upper Value	160000.0 um	160000.0 um
Position Command Unit	um	um
Position Command Unit String		

2.4 Linear Interpolation

This section describes MCv_MoveLinearInterpolateAbsolute that performs the linear interpolation of absolute position specification used in the sample program.
The details of the input variables are described in the following pages.

```
1  wLinearAxesNum[0]:= 1;
2  wLinearAxesNum[1]:= 2;
3
4  lePointAddress[0]:= 100000.0;
5  lePointAddress[1]:= 100000.0;
6
7  wDirection[0]:= MC_DIRECTION__mcShortestWay;
8  wDirection[1]:= MC_DIRECTION__mcShortestWay;
9
10 MCv_MoveLinearInterpolateAbsolute_1(
11     AxesGroup        := AxesGroup001.AxesGroupRef,
12     Execute          := bLinearReq ,
13     ContinuousUpdate:= FALSE ,
14     LinearAxes       := wLinearAxesNum ,
15     Position         := lePointAddress ,
16     Velocity         := leVelocity ,
17     Acceleration     := leAcceleration ,
18     Deceleration     := leDeceleration ,
19     Jerk             := leJerk ,
20     VelocityMode     := MC_INTERPOLATE_SPEED_MODE__VectorSpeed ,
21     Direction        := wDirection ,
22     BufferMode       := MC_BUFFER_MODE__mcAborting ,
23     Options          := H0 ,
24     Done             => bLinearDone ,
25     Busy             => bLinearBusy ,
26     Active           => bLinearActive ,
27     CommandAborted   => bLinearAborted ,
28     Error            => bLinearError ,
29     ErrorID          => uLinearErrirID
30 );
```

LinearAxes input

Position input

Direction input

These three input variables will be arrays.

〈FB specification (excerpt)〉

I/O variable name		Variable name	Data type	Description
Input	Axes group information	AxesGroup	AXES_GROUP_REF	Specifies the structure that indicates the axes group.
	Start	Execute	BOOL	Executes the FB when it is TRUE.
	Continuous update	ContinuousUpdate	BOOL	The travel distance, velocity, acceleration, and deceleration can be continuously changed while it is TRUE.
	Linear interpolation axis	LinearAxes	INT[0..15]	Specifies the axis to be used for linear interpolation control from the configuration axes. The index No. (1 to 16) of the configuration axis is specified in the array.
	Target position	Position	LREAL[0..15]	Sets the target absolute position according to the unit of the axes group.
	Velocity	Velocity	LREAL	Sets the velocity according to the unit of the axes group.
	Acceleration	Acceleration	LREAL	Sets the acceleration rate according to the unit of the axes group.
	Deceleration	Deceleration	LREAL	Sets the deceleration rate according to the unit of the axes group.

	Jerk	Jerk	LREAL	Sets the jerk according to the unit of the axes group.
	Velocity mode	VelocityMode	INT (MC_INTERPOLATE_SPEED_MODE)	Specifies the velocity mode of interpolation control.
	Direction selection	Direction	INT (MC_DIRECTION[0..15])	Sets the direction.
	Buffer mode	BufferMode	INT (MC_BUFFER_MODE)	Selects the buffer mode.
	Option	Options	DWORD(HEX) (Note)	Sets the functional option in bits.
Output	Execution completion	Done	BOOL	Indicates that the control is completed.
	Executing	Busy	BOOL	Becomes TRUE during FB execution.
	Controlling	Active	BOOL	Becomes TRUE when the FB is controlling an axis.
	Abortion of execution	CommandAborted	BOOL	Becomes TRUE when the execution is interrupted.
	Error	Error	BOOL	Becomes TRUE when an error occurs in the FB.
	Error code	ErrorID	WORD(UINT)	Returns the error code of the error that has occurred in the FB.

(Note) Hexadecimal values are described as "H□" or "16#□".

2.4 Linear Interpolation

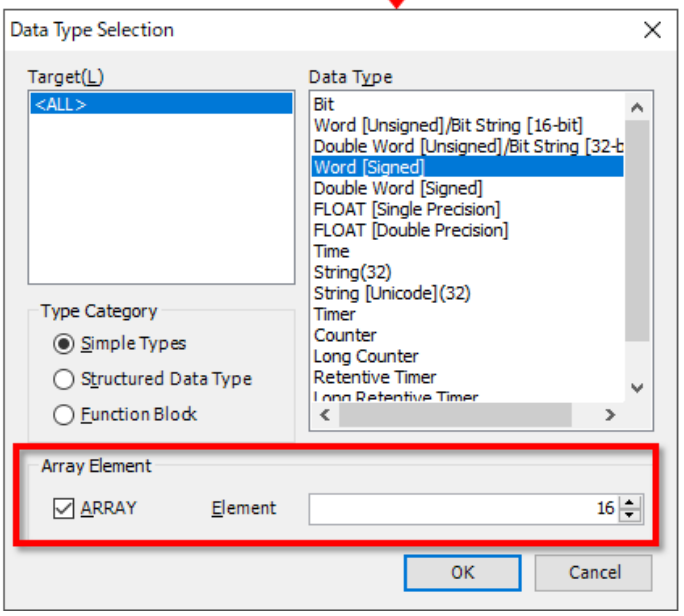
This section provides the detailed information of the input variables of MCv_MoveLinearInterpolateAbsolute.

(1) LinearAxes

Specify the array of the signed word type (INT type) with 16 elements for LinearAxes.
Set wLinearAxesNum(0..15) as an input label.
Specify *N* of the configuration axis [*N*] in the axes group setting for the value of wLinearAxesNum.

〈Label setting screen〉

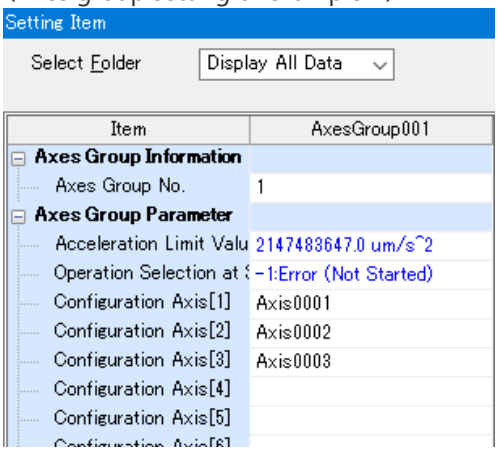
	Label Name	Data Type	Class	Initial	Constant	Comment
1	wLinearAxesNum	Word [Signed](0..15)	VAR			Axis Number
2	lePointAddress	FLOAT [Double Precision](0..15)	VAR			Target Position
3	wDireation	Word [Signed](0..15)	VAR			Travel Direction
4						



(Example1: Sample program)
Since Axis0001 is set for configuration axis [1] and Axis0002 is set for configuration axis [2] in the AxesGroup setting to perform linear interpolation for configuration axis [1] and configuration axis [2], input "1" in wLinearAxesNum[0], and input "2" in wLinearAxesNum[1].
Set 0 (initial value) for wLinearAxesNum[2] to wLinearAxesNum[15].

(Example 2)
If Axis0001 is set for configuration axis [1], Axis0002 is set for configuration axis [2], and Axis0003 is set for configuration axis [3] in the AxesGroup setting to perform linear interpolation for **configuration axis [1](Axis0001)** and **configuration axis [3] (Axis0003)** , input "1" in wLinearAxesNum[0], and input "3" in wLinearAxesNum[1].
Set 0 (initial value) for wLinearAxesNum[2] to wLinearAxesNum[15].

〈Axes group setting of example 2〉



(2) Position

Specify the array of the double-precision real number type (LREAL type) with 16 elements for Position.
Set lePointAddress(0..15) as an input label.
Input the target position of configuration axis [*N*] to lePointAddress[*N-1*].

(Example 1: Sample program)
To perform linear interpolation for configuration axis [1] (Axis0001) and configuration axis [2] (Axis0002), input the target position of configuration axis 1 to lePointAddress[0] and target position of configuration axis 2 to lePointAddress[1].

(Example 2)
To perform linear interpolation for **configuration axis [1] (Axis0001)** and **configuration axis [3] (Axis0003)**, input the target position of configuration axis 1 to lePointAddress[0] and target position of configuration axis 3 to lePointAddress[**2**].

(3) VelocityMode

Input the ENUM enumerator or numerical value in the following table to VelocityMode.

Value	ENUM enumerator	Description
0	MC_INTERPOLATE_SPEED_MODE__VectorSpeed	Vector velocity
1	MC_INTERPOLATE_SPEED_MODE__LongAxisSpeed	Long axis velocity
2	MC_INTERPOLATE_SPEED_MODE__ReferenceAxisSpeed	Reference axis velocity

(4) Direction

Specify the array of the signed word type (INT type) with 16 elements for Direction.
Set wDirection(0..15) as an input label.
Input the travel direction of configuration axis [*N*] to wDirection[*N-1*].
Set the travel direction with the ENUM enumerator or numerical value in the following table.

Value	ENUM enumerator	Description
1	MC_DIRECTION__mcPositiveDirection	Positive direction
2	MC_DIRECTION__mcNegativeDirection	Negative direction
3	MC_DIRECTION__mcShortestWay	Shortest path

(Example 1: Sample program)
Since the linear interpolation is performed for configuration axis [1] (Axis0001) and configuration axis [2] (Axis0002), input the travel distance of configuration axis 1 to wDirection[0] and travel distance of configuration 2 to wDirection[1].

(Example 2)
To perform linear interpolation for **configuration axis [1] (Axis0001)** and **configuration axis [3] (Axis0003)**, input the travel direction of configuration axis [1] to wDirection[0],and input the travel direction of configuration axis [3] to wDirection[**2**].

2.5 Circular Interpolation

This section describes MCv_MoveCircularInterpolateAbsolute that performs the circular interpolation of absolute position specification used in the sample program.
The details of the input variables are described in the following pages.

```
wCircAxesNum[0]:=1 ;
wCircAxesNum[1]:=2 ;

leAuxPoint[0]:= 10000.0;
leAuxPoint[1]:= 0.0;

leEndPoint[0]:= 10000.0;
leEndPoint[1]:= 10000.0;

MCv_MoveCircularInterpolateAbsolute_1(
    AxesGroup          := AxesGroup001.AxesGroupRef,
    Execute            := bCircReq ,
    ContinuousUpdate:= FALSE ,
    CircAxes           := wCircAxesNum ,
    CircMode           := MC_CIRC_MODE__mcCenter ,
    AuxPoint           := leAuxPoint ,
    EndPoint           := leEndPoint ,
    PathChoice         := MC_CIRC_PATHCHOICE__mcCW ,
    Velocity           := leVelocity ,
    Acceleration       := leAcceleration ,
    Deceleration       := leDeceleration ,
    Jerk               := leJerk ,
    CircularErrorTolerance  := leTolerance ,
    BufferMode         := MC_BUFFER_MODE__mcAborting ,
    Options            := H0 ,
    Done               => bCircDone ,
    Busy               => bCircBusy ,
    Active             => bCircActive ,
    CommandAborted     => bCircAborted ,
    Error              => bCircError ,
    ErrorID            => uCircErrorID
);
```

CircAxes input

AuxPoint input

EndPoint input

These three input variables will be arrays.

〈FB specification (excerpt)〉

I/O variable name		Variable name	Data type	Description
Input	Axes group information	AxesGroup	AXES_GROUP_REF	Specifies the structure that indicates the axes group.
	Start	Execute	BOOL	Executes the FB when it is TRUE.
	Continuous update	ContinuousUpdate	BOOL	The travel distance, velocity, acceleration, and deceleration can be continuously changed while it is TRUE.
	Circular interpolation axis	CircAxes	INT[0..1]	Specifies the axis to be used for the circular interpolation control from the configuration axis. The index No. (1 to 16) of the configuration axis is specified in the array.
	Circular interpolation mode	CircMode	INT(MC_CIRC_MODE)	Sets the circular interpolation mode.
	Sub point	AuxPoint	LREAL[0..15]	Sets the absolute position of the sub point according to the unit of the axes group. The content varies by CircMode.
	End point	EndPoint	LREAL[0..15]	Sets the absolute position of the end point according to the unit of the axes group.

	Path selection	PathChoice	INT (MC_CIRC_PATHCHOICE)	Sets the rotation direction of the circular interpolation.
	Velocity	Velocity	LREAL	Sets the velocity according to the unit of the axes group.
	Acceleration	Acceleration	LREAL	Sets the acceleration rate according to the unit of the axes group.
	Deceleration	Deceleration	LREAL	Sets the deceleration rate according to the unit of the axes group.
	Jerk	Jerk	LREAL	Sets the jerk according to the unit of the axes group.
	Circular interpolation error tolerance value	CircularErrorTolerance	LREAL	Sets the tolerance range of the circular interpolation error.
	Buffer mode	BufferMode	INT (MC_BUFFER_MODE)	Selects the buffer mode.
	Option	Options	DWORD(HEX)	Sets the functional option in bits.
Output	Execution completion	Done	BOOL	Indicates that the control is completed.
	Executing	Busy	BOOL	Becomes TRUE during FB execution.
	Controlling	Active	BOOL	Becomes TRUE when the FB is controlling an axis.
	Abortion of execution	CommandAborted	BOOL	Becomes TRUE when the execution is interrupted.
	Error	Error	BOOL	Becomes TRUE when an error occurs in the FB.
	Error code	ErrorID	WORD(UINT)	Returns the error code of the error that has occurred in the FB.

2.5 Circular Interpolation

(1) CircAxes

Specify the array of the signed word type (INT type) with **2** elements in CircAxes.
Set wCircAxesNum(0..1) as an input label.
For the value of wCircAxesNum, specify *N* of the configuration axis [*N*] in the axes group setting.

(Example 1: Sample program)
Since Axis0001 is set for configuration axis [1] and Axis0002 is set for configuration axis [2] in the AxesGroup setting to perform circular interpolation for configuration axis 1 (Axis0001) and configuration axis 2 (Axis0002),
input "1" to wCircAxesNum[1] and "2" to wCircAxesNum[1].

(Example 2)
If Axis0001 is set for configuration axis [1], Axis0002 is set for configuration axis [2],
and Axis0003 is set for configuration axis [3] in the AxesGroup setting to perform circular interpolation for **configuration axis 1 (Axis0001)** and **configuration axis 3 (Axis0003)**,
input "1" to wCircAxesNum[0] and "3" to wCircAxesNum[1].

(2) CircMode

Input the ENUM enumerator or numerical value in the following table to CircMode.

Value	ENUM enumerator	Description
0	MC_CIRC_MODE__mcBorder	Border point specification
1	MC_CIRC_MODE__mcCenter	Center point specification
2	MC_CIRC_MODE__mcRadius	Radius specification

(3) EndPoint

Input the end point coordinates of the circular arc to EndPoint.

Set leEndPoint(0..15) as the input label of EndPoint.
Input the end point coordinates of the configuration axis [*N*] to leEndPoint[*N-1*].

(Example 1: Sample program)
If Axis0001 is set for configuration axis [1] and Axis0002 is set for configuration axis [2] in the AxesGroup setting to perform circular interpolation for configuration axis [1] (Axis0001) and configuration axis [2] (Axis0002), input the coordinates of the end point of Axis0001 to leEndPoint[0] and the coordinates of the end point of Axis0002 to leEndPoint[1].

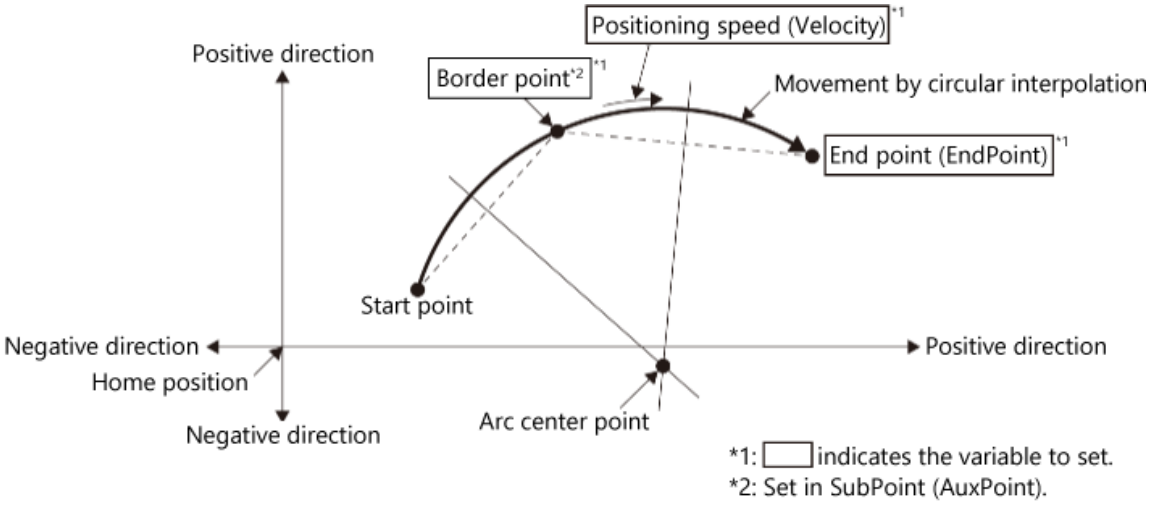
(Example 2)
If Axis0001 is set for configuration axis [1],
Axis0002 is set for configuration axis [2], and Axus0003 is set for configuration axis [3] in the AxesGroup setting to perform circular interpolation with border point specification for **configuration axis [1] (Axis0001)** and **configuration axis [3] (Axis0003)**, input the coordinates of the end point of Axis0001 to leEndPoint[0] and the coordinates of the end point of Axis0003 to leEndPoint[2].
leEndPoint[1] and leEndPoint[3] to [15] will ignore inputs.

(4) AuxPoint, PathChoice

The contents of the input variables AuxPoint and PathChoice vary depending on the type of circular interpolation.

(a) For circular interpolation with border point specification

When CircMode is 0:mcBorder, input the coordinates of the sub point on the arc to AuxPoint.
No setting is required for PathChoice input. (It ignores the input.)



Set leBorderPoint(0..15) as the input label of AuxPoint.
Input the coordinates of the sub point of configuration axis [*N*] to leBorderPoint[*N-1*].

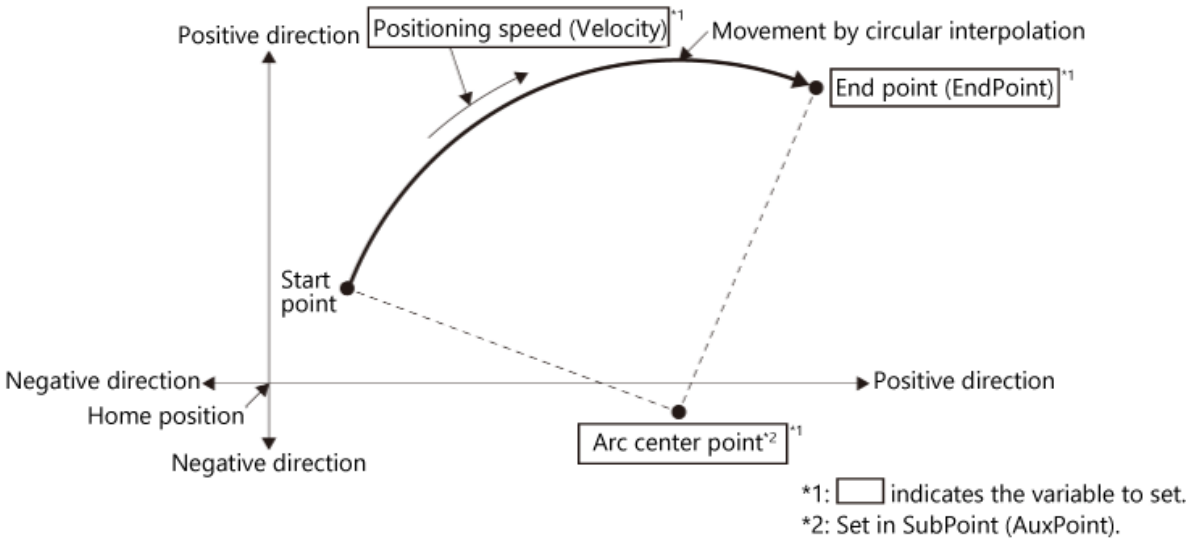
(Example 1: Sample program)
If Axis0001 is set for configuration axis [1] and Axis0002 is set for configuration axis [2] in the AxesGroup setting to perform circular interpolation with border point specification for configuration axis [1] (Axis0001) and configuration axis [2] (Axis0002), input the coordinates of the sub point of Axis0001 to leBorderPoint[0] and the coordinates of the sub point of Axis0002 to leBorderPoint[1].

(Example 2)
If Axis0001 is set for configuration axis [1], Axis0002 is set for configuration axis [2], and Axis0003 is set for configuration axis [3] in the AxesGroup setting to perform circular interpolation with border point specification for **configuration axis [1] (Axis0001)** and **configuration axis [3] (Axis0003)**, input the coordinates of the sub point of Axis0001 to leBorderPoint[0] and the coordinates of the sub point of Axis0003 to leBorderPoint[2].
leBorderPoint[1] and leBorderPoint[3] to [15] will ignore inputs.

(b) For circular interpolation with center point specification

When CircMode is 1:mcCenter, input the coordinates of the center point of the arc to AuxPoint.

Input the ENUM enumerator or numerical value in the following table to PathChoice input.



<AuxPoint input>

Set leCenterPoint(0..15) as the input label as AuxPoint.

Input the coordinates of the sub point of configuration axis [*N*] to leCenterPoint[*N-1*].

(Example 1: Sample program)

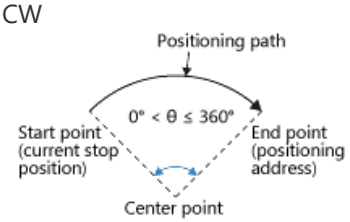
If Axis0001 is set for configuration axis [1] and Axis0002 is set for configuration axis [2] in the AxesGroup setting to perform circular interpolation with center point specification for configuration axis [1] (Axis0001) and configuration axis [2] (Axis0002), input the coordinates of the center point of Axis0001 to leCenterPoint[0] and the coordinates of the center point of Axis0002 to leCenterPoint[1].

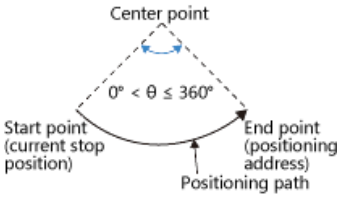
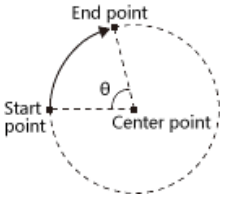
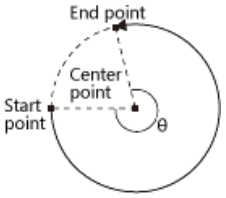
(Example 2)

If Axis0001 is set for configuration axis [1], Axis0002 is set for configuration axis [2], and Axis0003 is set for configuration axis [3] in the AxesGroup setting to perform the circular interpolation with center point specification for **configuration axis [1] (Axis0001)** and **configuration axis [3] (Axis0003)**, input the coordinates of the center point of Axis0001 to leCenterPoint[0] and the coordinates of the center point of Axis0003 to leCenterPoint[2].

leCenterPoint[1] and leCenterPoint[3] to [15] will ignore inputs.

<PathChoice input>

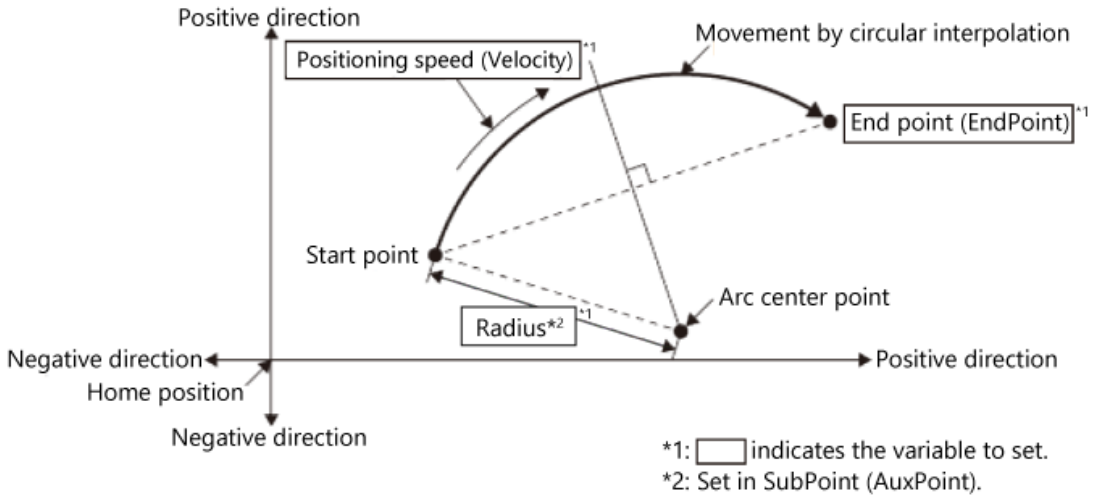
Value	ENUM enumerator	Description
0	MC_CIRC_PATHCHOICE__mcCW	CW 
1	MC_CIRC_PATHCHOICE__mcCCW	CCW

		
2	MC_CIRC_PATHCHOICE__mcShortWay	Shortcut 
3	MC_CIRC_PATHCHOICE__mcLongWay	Detour 

(c) For circular interpolation of radius specification

When CircMode is set to 2:mcRadius, the radius of the arc is input to AuxPoint.

Input the ENUM enumerator or numerical value in the following table to PathChoice input.

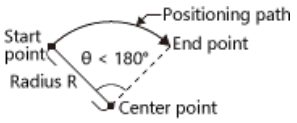
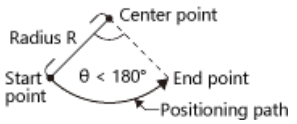
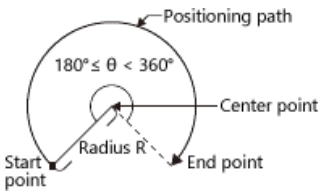
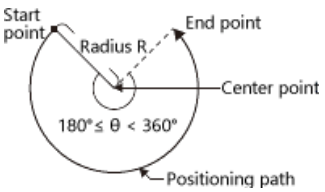


<AuxPoint input>

Set AuxPoint as leRadius(0..15) as the input label of AuxPoint.

Input the radius of the arc to leRadius[0]. Ignore the setting of leRadius[1] to [15].

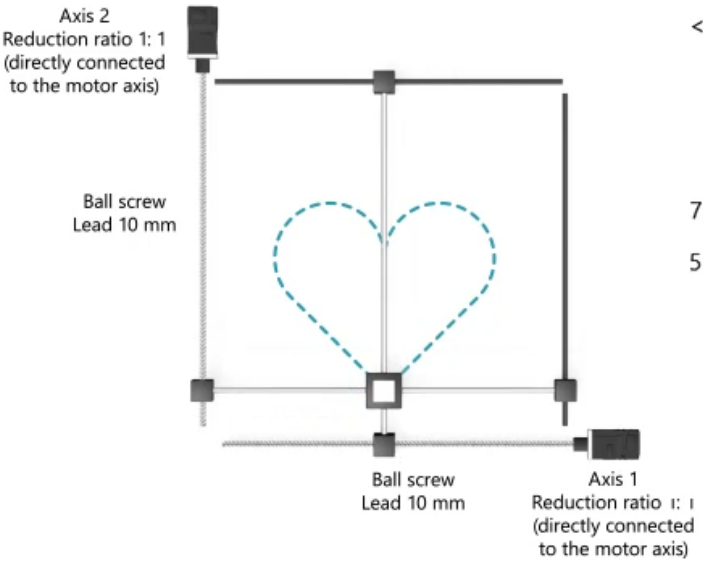
<PathChoice input>

Value	ENUM enumerator	Description
0	MC_CIRC_PATHCHOICE__mcCW	CW shortcut 
1	MC_CIRC_PATHCHOICE__mcCCW	CCW shortcut 
4	MC_CIRC_PATHCHOICE__mcCWLongWay	CW detour 
5	MC_CIRC_PATHCHOICE__mcCCWLongWay	CCW detour 

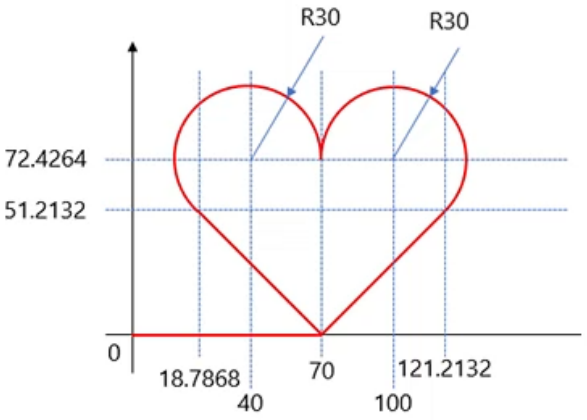
2.6 Program Example

(1) Operation patterns

Create programs that draw the shape combining linear interpolation and circular interpolation as shown below.



< Coordinates of each point >

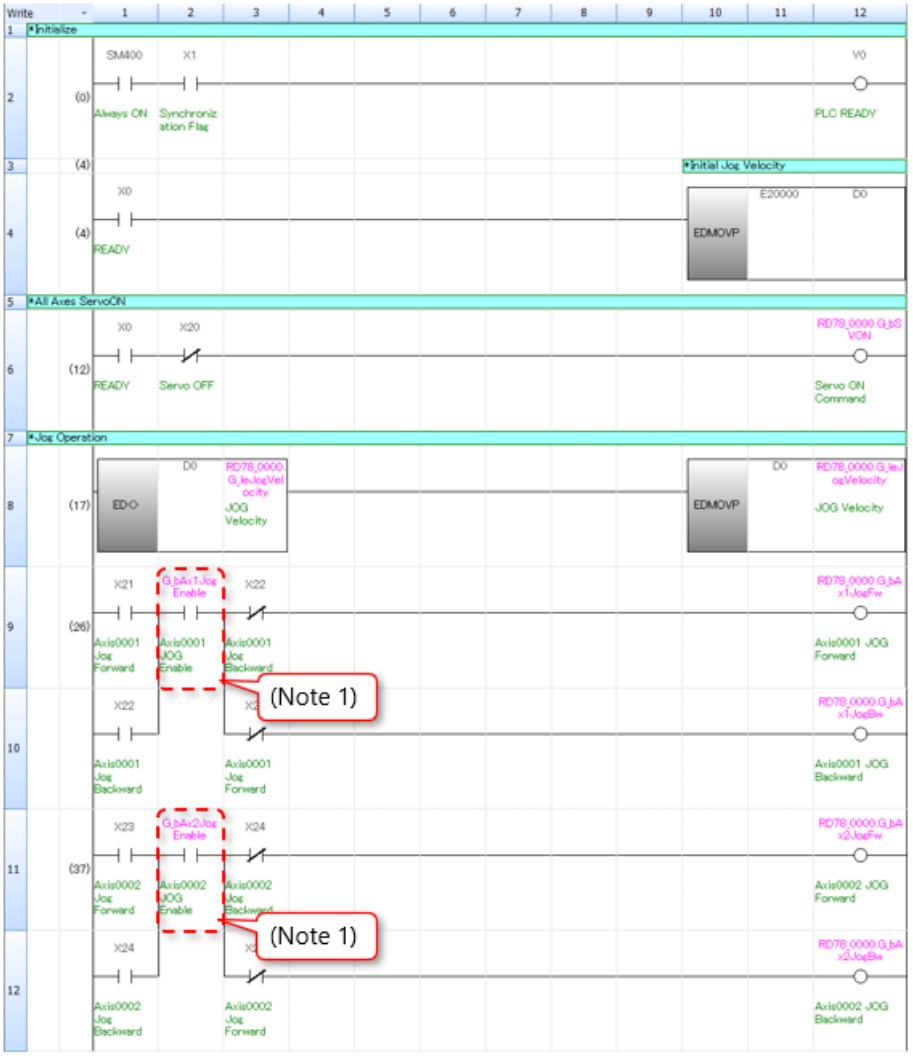


(2) Program of the PLC CPU

The following shows the program of the PLC CPU.

1) Program name [MAIN]

Public labels are used to send the start signal to the Motion module.



[Initialization]
Y0 (PLC ready) is turned on and the initial value of JOG speed is set.

[Servo ON/OFF]
Servo OFF is executed when X20 is turned on.

[JOG operation]
When the value of D0 changes, the JOG speed in the JOG speed (G_leJogVelocity) of the public label is stored. (Note 2)
JOG operation is performed while X21 to X24 are on.
JOG operation ends when they are turned off.
This prevents the simultaneous ON of the forward rotation and reverse rotation.

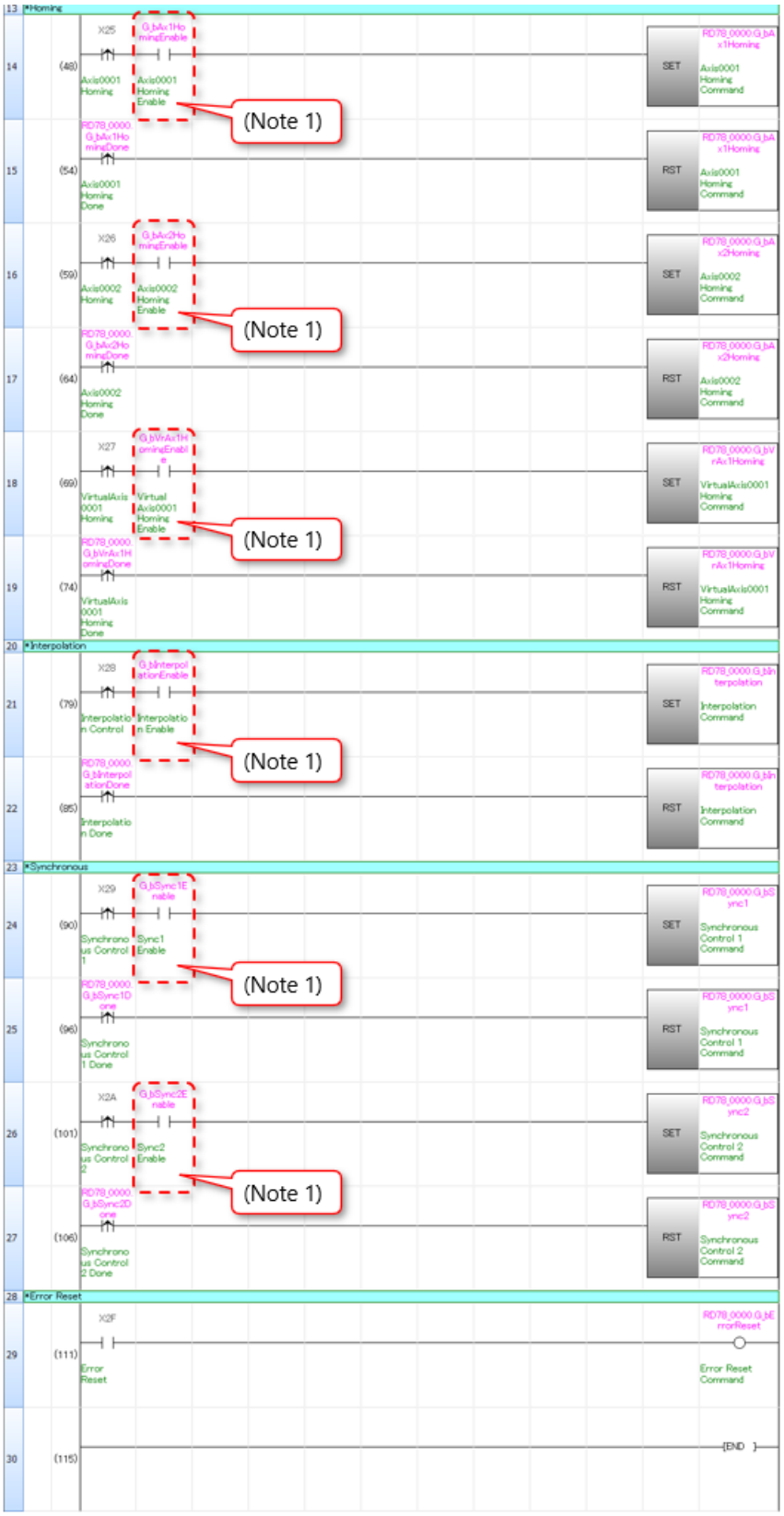
(Note 1) The bit for interlock.
It turns ON when the interlock condition is satisfied.
Refer to 2) program name [interlock].

(Note 2) To change the JOG speed, change the value of D0 in double-precision real number type from the GOT or other programs.

(Continued to the next page)

(2) Program of the PLC CPU

1) Program name [MAIN] (continue)



[Homing]
Axis0001, Axis0002, and VirtualAxis0001 (Note 2) start homing when X25 to X27 are turned on.
When the completion signal is received from the Motion module, the start signal is reset.

(Note 2) The details of the virtual drive axis VirtualAxis0001 are described in Chapter 3.

[Interpolation control]
The interpolation control starts when X28 is turned on.
When the completion signal is received from the Motion module, the start signal is reset.

[Synchronous control] (for Chapter 3)
The synchronous control starts when X29 and X2A are turned on.
When the completion signal is received from the Motion module, the start signal is reset.

[Error reset]
The error reset is performed when X2F is turned on.

(Note 1) The bit for interlock.
It turns ON when the interlock condition is satisfied.
Refer to 2) program name [interlock].

(2) Program of the PLC CPU (continued)

2) Program name [Interlock]

The interlock condition of each program is described below.

(3) Program of the Motion module

The following shows the program of the Motion module.

1) Program name [ServoON_Jog]

This program performs all axes servo ON and JOG operation.

```
1  //-----All Axes Servo ON-----//
2  MCv_AllPower_1(
3      Enable  := TRUE ,
4      ServoON := G_bSVON
5  );
6
7  //-----JOG-----//
8  IF(G_leJogVelocity <> leJogVelocity )THEN
9      leJogVelocity := G_leJogVelocity;
10     leJogAcc := G_leJogVelocity * 2.0;
11     leJogDec := G_leJogVelocity * 2.0;
12     leJogJerk:= G_leJogVelocity * 4.0;
13 END_IF;
14
15 //Axis0001
16 MCv_Jog_1(
17     Axis        := Axis0001.AxisRef ,
18     JogForward  := G_bAx1JogFw ,
19     JogBackward := G_bAx1JogBw ,
20     Velocity    := leJogVelocity ,
21     Acceleration:= leJogAcc ,
22     Deceleration:= leJogDec ,
23     Jerk        := leJogJerk ,
24     Busy        => G_bAx1JogBusy
25 );
26 //Axis0002
27 MCv_Jog_2(
28     Axis        := Axis0002.AxisRef ,
29     JogForward  := G_bAx2JogFw ,
30     JogBackward := G_bAx2JogBw ,
31     Velocity    := leJogVelocity ,
32     Acceleration:= leJogAcc ,
33     Deceleration:= leJogDec ,
34     Jerk        := leJogJerk ,
35     Busy        => G_bAx2JogBusy
36 );
37
```

Servo ON signal (from PLC CPU)

When the JOG speed is changed, acceleration, deceleration, and jerk are calculated again according to the velocity. (Note)

JOG signal (from PLC CPU)

JOG signal (from PLC CPU)

(Note)

When a value larger than 536870911.0 is input to leJogVelocity, the value of leJogJerk exceeds the limit of the input value (2147483647.0) of the FB.

Therefore, be careful when changing the JOG speed.

(3) Program of the Motion module

2) Program name [Homing]

This program performs the Homing of each axis.

```
//-----Homing-----//
//Axis0001
MC_Home_1(
    Axis    := Axis0001.AxisRef ,
    Execute := G_bAx1Homing ,
    Position:= 0.0 ,
    Done            => bAx1HomingDone ,
    CommandAborted  => bAx1HomingAborted ,
    Error           => bAx1HomingError
);

G_bAx1HomingDone := bAx1HomingDone OR bAx1HomingAborted OR bAx1HomingError;

//Axis0002
MC_Home_2(
    Axis    := Axis0002.AxisRef ,
    Execute := G_bAx2Homing ,
    Position:= 0.0 ,
    Done            => bAx2HomingDone ,
    CommandAborted  => bAx2HomingAborted ,
    Error           => bAx2HomingError
);

G_bAx2HomingDone := bAx2HomingDone OR bAx2HomingAborted OR bAx2HomingError;

//VirtualAxis0001
MC_Home_3(
    Axis    := VirtualAxis0001.AxisRef
    Execute := G_bVrAx1Homing ,
    Position:= 0.0 ,
    Done            => bVrAx1HomingDone ,
    CommandAborted  => bVrAx1HomingAborted ,
    Error           => bVrAx1HomingError
);

G_bVrAx1HomingDone := bVrAx1HomingDone OR bVrAx1HomingAborted OR bVrAx1HomingError;
```

Homing start signal (from PLC CPU)

Homing start signal (from PLC CPU)

Homing start signal (from PLC CPU)

(3) Program of the Motion module

3) Program name [Interpolation]

This program performs interpolation control.

```
1  //-----Interpolation-----//
2  //Intial Value Setting
3  IF(G_bInterpolation) THEN
4      //Positioning Velocity,Acceleration,Deceleration,Jerk
5      leVelocity      := 30000.0;
6      leAcceleration  := 60000.0;
7      leDeceleration  := 60000.0;
8      leJerk          := 120000.0;
9      //Request ON
10     G_bInterpolationReq := TRUE;
11 ELSE
12     //Request OFF
13     G_bInterpolationReq := FALSE;
14     //Reset Internal Signal
15     bMoveLinear2Dwell_in := FALSE;
16 END_IF;
17
18 //Move to Point0
19 MC_MoveAbsolute_1(
20     Axis            := Axis0001.AxisRef ,
21     Execute         := G_bInterpolationReq ,
22     Position        := G_lePoint0[0] ,
23     Velocity        := leVelocity ,
24     Acceleration    := leAcceleration ,
25     Deceleration    := leDeceleration ,
26     Jerk            := leJerk ,
27     Done            => bMoveAbs1Done ,
28     Busy            => bMoveAbs1Busy ,
29     Active          => bMoveAbs1Active ,
30     CommandAborted  => bMoveAbs1Aborted ,
31     Error           => bMoveAbs1Error
32 );
33
34 //Dwell
35 TON_1(IN:= bMoveAbs1Done ,PT:= T#1s ,Q=> bMoveAbs1Dwell_out);
36
```

Interpolation control start signal (from PLC CPU)

When the PLC CPU turns on the interpolation control start signal (G_bInterpolation), the positioning speed and other commands are set and the start signal on the Motion module side (G_bInterpolationReq) is turned on.

When the PLC CPU turns off the interpolation control start signal (G_bInterpolation), the internal signal is turned off.

Axis0001 is moved to Point0 by single axis positioning.

Dwell

(Continued to the next page)

(3) Program of the Motion module

3) Program name [Interpolation] (continued)

```
//Group Enable
MC_GroupEnable_1(
    AxesGroup:= AxesGroup001.AxesGroupRef ,
    Execute:= bMoveAbs1Dwell_out ,
    Done=> bGrpEnDone ,
    Error=> bGrpEnError
);

//Move to Point1 Linear Interpolation
MCv_MoveLinearInterpolateAbsolute_1(
    AxesGroup       := AxesGroup001.AxesGroupRef ,
    Execute         := bGrpEnDone ,
    LinearAxes      := G_wAxesNum ,
    Position        := G_lePoint1 ,
    Velocity        := leVelocity ,
    Acceleration    := leAcceleration ,
    Deceleration    := leDeceleration ,
    Jerk            := leJerk ,
    VelocityMode    := MC_INTERPOLATE_SPEED_MODE__VectorSpeed ,
    Direction       := G_wDirection ,
    Busy            => bMoveLinear1Busy ,
    Active          => bMoveLinear1Active ,
    CommandAborted  => bMoveLinear1Aborted ,
    Error           => bMoveLinear1Error
);

//Move to Point2 Circular Interpolation
MCv_MoveCircularInterpolateAbsolute_1(
    AxesGroup       := AxesGroup001.AxesGroupRef ,
    Execute         := bMoveLinear1Active ,
    ContinuousUpdate:= FALSE ,
    CircAxes        := G_wCircAxesNum ,
    CircMode        := MC_CIRC_MODE__mcCenter ,
    AuxPoint        := G_leCenter1 ,
    EndPoint        := G_lePoint2 ,
    PathChoice      := MC_CIRC_PATHCHOICE__mcCW ,
    Velocity        := leVelocity ,
    Acceleration    := leAcceleration ,
    Deceleration    := leDeceleration ,
    Jerk            := leJerk ,
    BufferMode      := MC_BUFFER_MODE__mcBlendingNext ,
    Busy            => bMoveCirc1Busy ,
    Active          => bMoveCirc1Active ,
    CommandAborted  => bMoveCirc1Aborted ,
    Error           => bMoveCirc1Error
);
```

The axes group is enabled.

The axes are controlled to move to Point1 by linear interpolation.

Vector velocity specification

It is set in Program name: Initialize. Refer to 2.6 (3) 5).

The axes are controlled to move to Point2 by circular interpolation with center point specification. BlendingNext of the buffer mode is used for execution.

Center point specification CW

It is set in Program name: Initialize. Refer to 2.6 (3) 5).

(Continued to the next page)

(3) Program of the Motion module

3) Program name [Interpolation] (continued)

```
//Move to Point3 Circular Interpolation
MCv_MoveCircularInterpolateAbsolute_2(
    AxesGroup        := AxesGroup001.AxesGroupRef ,
    Execute          := bMoveCirc1Active ,
    ContinuousUpdate:= FALSE ,
    CircAxes         := G_wCircAxesNum ,
    CircMode         := MC_CIRC_MODE__mcRadius ,
    AuxPoint         := G_leRadius ,
    EndPoint         := G_lePoint3 ,
    PathChoice       := MC_CIRC_PATHCHOICE__mcCWLongWay ,
    Velocity         := leVelocity ,
    Acceleration     := leAcceleration ,
    Deceleration     := leDeceleration ,
    Jerk             := leJerk ,
    BufferMode       := MC_BUFFER_MODE__mcBuffered ,
    Busy             => bMoveCirc2Busy ,
    Active           => bMoveCirc2Active ,
    CommandAborted   => bMoveCirc2Aborted ,
    Error            => bMoveCirc2Error
);

//Move to Point0 Linear Interpolation
MCv_MoveLinearInterpolateAbsolute_2(
    AxesGroup        := AxesGroup001.AxesGroupRef ,
    Execute          := bMoveCirc2Active ,
    ContinuousUpdate:= FALSE ,
    LinearAxes       := G_wAxesNum ,
    Position         := G_lePoint0 ,
    Velocity         := leVelocity ,
    Acceleration     := leAcceleration ,
    Deceleration     := leDeceleration ,
    Jerk             := leJerk ,
    VelocityMode     := MC_INTERPOLATE_SPEED_MODE__VectorSpeed ,
    Direction        := G_wDirection ,
    BufferMode       := MC_BUFFER_MODE__mcBlendingNext ,
    Done             => bMoveLinear2Done ,
    Busy             => bMoveLinear2Busy ,
    Active           => bMoveLinear2Active ,
    CommandAborted   => bMoveLinear2Aborted ,
    Error            => bMoveLinear2Error
);

//Dwell
SET(bMoveLinear2Done, bMoveLinear2Dwell_in);
TON_2(IN:= bMoveLinear2Dwell_in ,PT:= T#1s ,Q=> bMoveLinear2Dwell_out);

//Group Disable
MC_GroupDisable_1(
    AxesGroup    := AxesGroup001.AxesGroupRef ,
    Execute      := bMoveLinear2Dwell_in OR bError OR bAborted,
    Done         => bGrpDsblDone
);
```

The axes are controlled to move to Point3 by circular interpolation with radius specification.
Buffered of the buffer mode is used for execution (Note 1).

Radius specification
CW detour (Note 2)

It is set in Program name: Initialize.
Refer to 2.6 (3) 5).

(Note 1) Buffered is used instead of Blending since the operation direction of Axis0002 is reversed at Point2.

(Note 2) Since the center angle of the arc is 180° or more, detour is specified.

The axes are controlled to move to Point0 by linear interpolation.
BlendingNext of the buffer mode is used for execution.

It is set in Program name: Initialize.
Refer to 2.6 (3) 5).

Dwell

The axes group is disabled.

(Continued to the next page)

(3) Program of the Motion module

3) Program name [Interpolation] (continued)

```
137 //Move to Home Point
138 MC_MoveAbsolute_2(
139     Axis             := Axis0001.AxisRef ,
140     Execute          := bGrpDsblDone & bMoveLinear2Dwell_out ,
141     ContinuousUpdate:= FALSE ,
142     Position         := 0.0 ,
143     Velocity         := leVelocity ,
144     Acceleration     := leAcceleration ,
145     Deceleration     := leDeceleration ,
146     Jerk             := leJerk ,
147     BufferMode       := MC_BUFFER_MODE__mcAborting ,
148     Done             => bMoveAbs2Done ,
149     Busy             => bMoveAbs2Busy ,
150     Active           => bMoveAbs2Active ,
151     CommandAborted   => bMoveAbs2Aborted ,
152     Error            => bMoveAbs2Error
153 );
154
155 //Dwell
156 TON_3(IN:= bMoveAbs2Done ,PT:= T#1s ,Q=> bMove2Dwell_out);
157
158 //Error Signal,Aborted Signal
159 bError := bMoveAbs1Error OR bGrpEnError OR bMoveLinear1Error OR bMoveCirc1Error
160           OR bMoveCirc2Error OR bMoveLinear2Error OR bMoveAbs2Error;
161 bAborted := bMoveAbs1Aborted OR bMoveLinear1Aborted OR bMoveCirc1Aborted
162             OR bMoveCirc2Aborted OR bMoveLinear2Aborted OR bMoveAbs2Aborted;
163
164 //Done Signal
165 G_bInterpolationDone := (bMove2Dwell_out OR bError OR bAborted )
166                         & (AxesGroup001.Md.GroupStatus = 0);
167
```

Axis0001 is moved to the home position by single axis positioning.

Dwell

The Error and Aborted outputs of FB are described in OR conditions.

When the program normal completion or the Error/Aborted output turns on, the Done signal is turned on and sent to the PLC CPU.

(3) Program of the Motion module

4) Program name [ErrorReset]

This program performs the error reset.

```
1  //Error Reset
2  MC_Reset_1(
3      Axis:= Axis0001.AxisRef ,
4      Execute:= G_bErrorReset
5  );
6
7  MC_Reset_2(
8      Axis:= Axis0002.AxisRef ,
9      Execute:= G_bErrorReset
10 );
11
12 MC_GroupReset_1(
13     AxesGroup:= AxesGroup001.AxesGroupRef,
14     Execute:= G_bErrorReset
15 );
16
17 MCv_MotionErrorReset_1(
18     Execute:= G_bErrorReset
19 );
20
```

Axis error reset

Axes group error reset

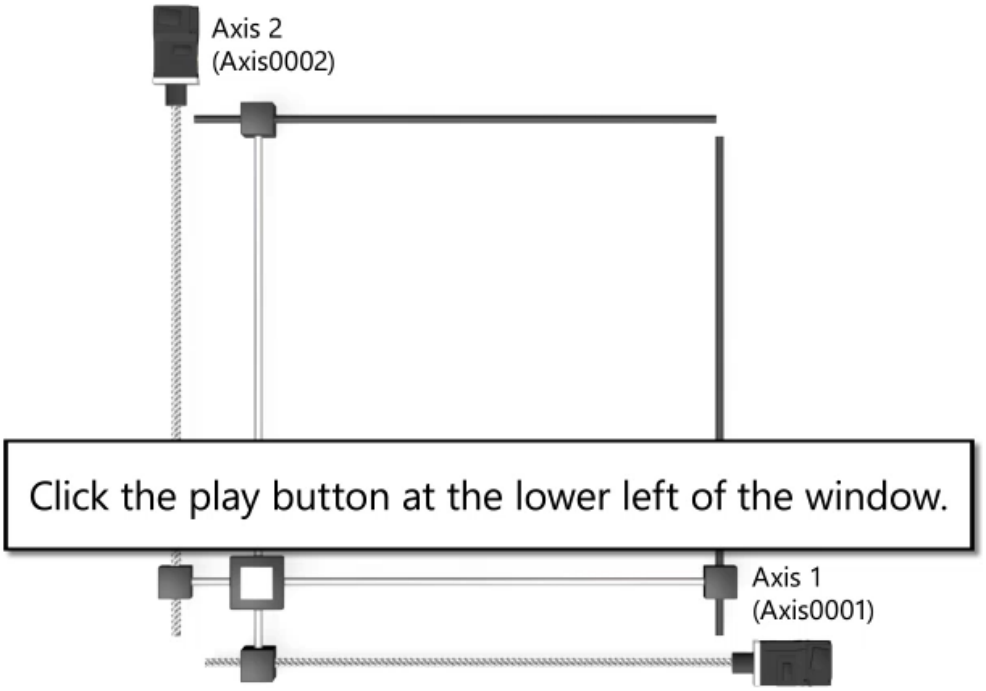
System error reset

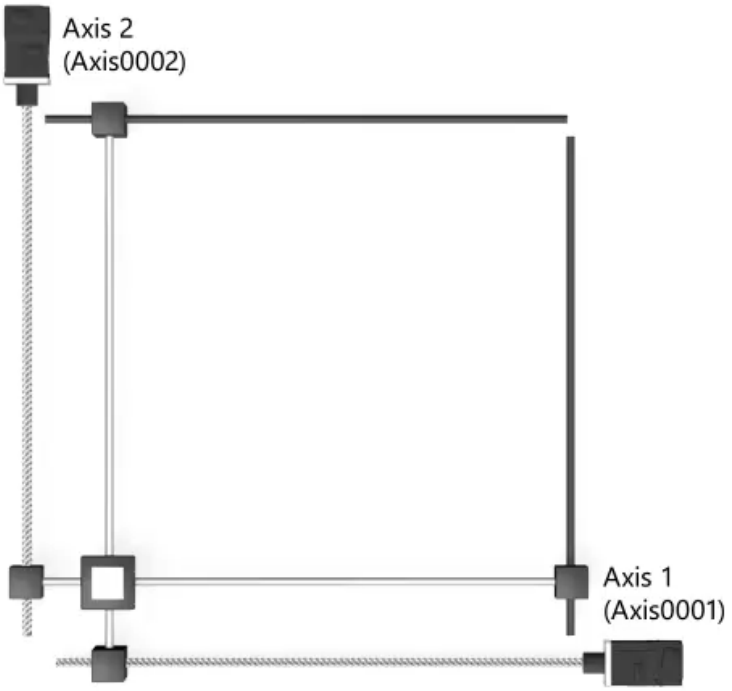
(3) Program of the Motion module

5) Program name [Initialize] (initial execution type)

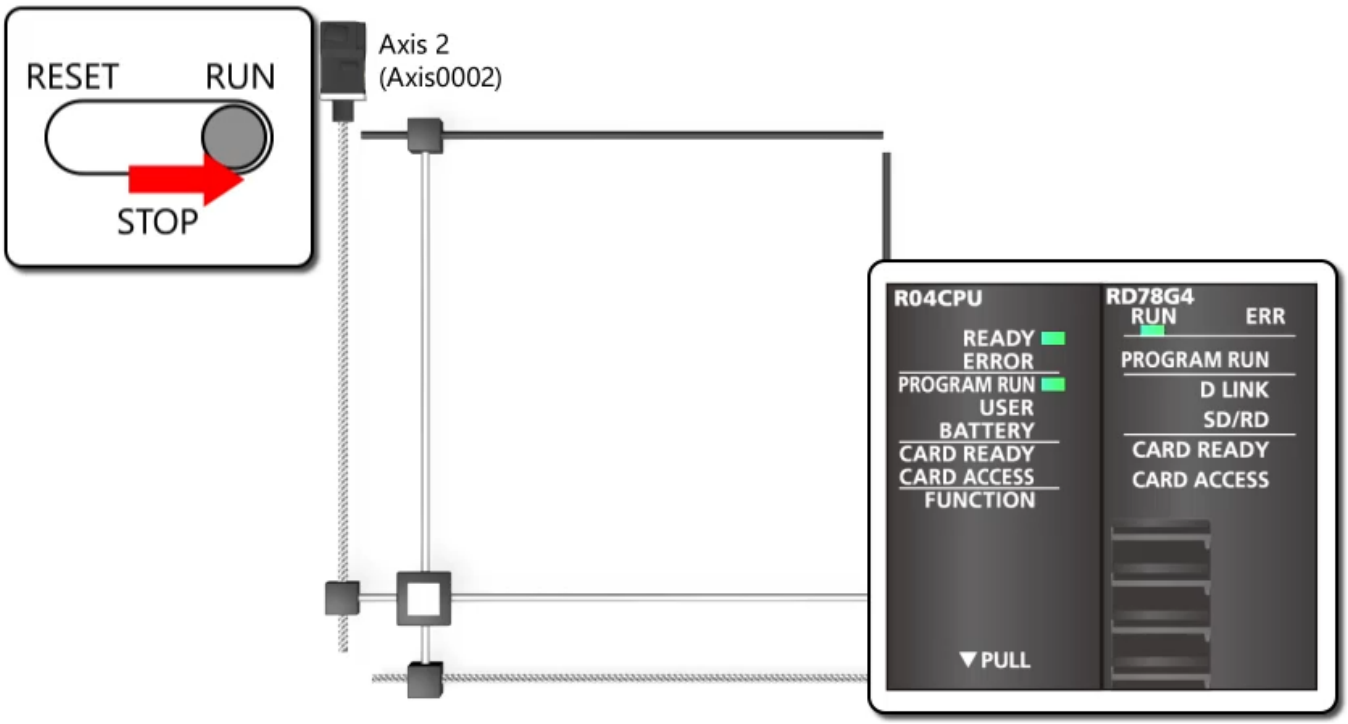
Set the coordinates of each point, axis number, coordinates of the center point, and arc radius to be used for interpolation control and the coordinates of the start point and end point of the spindle for synchronous control.

```
//Initial Value Setting
//Interpolation
G_leePoint0[0] := 70000.0;
G_leePoint0[1] := 0.0;
G_leePoint1[0] := 18786.8;
G_leePoint1[1] := 51213.2;
G_leePoint2[0] := 70000.0;
G_leePoint2[1] := 72426.4;
G_leePoint3[0] := 121213.2;
G_leePoint3[1] := 51213.2;

G_leCenter1[0] := 40000.0;
G_leCenter1[1] := 72426.4;

G_leRadius[0] := 30000.0;

G_wAxesNum[0] := 1;
G_wAxesNum[1] := 2;

G_wCircAxesNum[0] := 1;
G_wCircAxesNum[1] := 2;

G_wDirection[0] := MC_DIRECTION__mcShortestWay;
G_wDirection[1] := MC_DIRECTION__mcShortestWay;

//Synchronous1,Synchronous2
G_leHomePoint := 0.0;
G_leMovePoint := 150000.0;

```

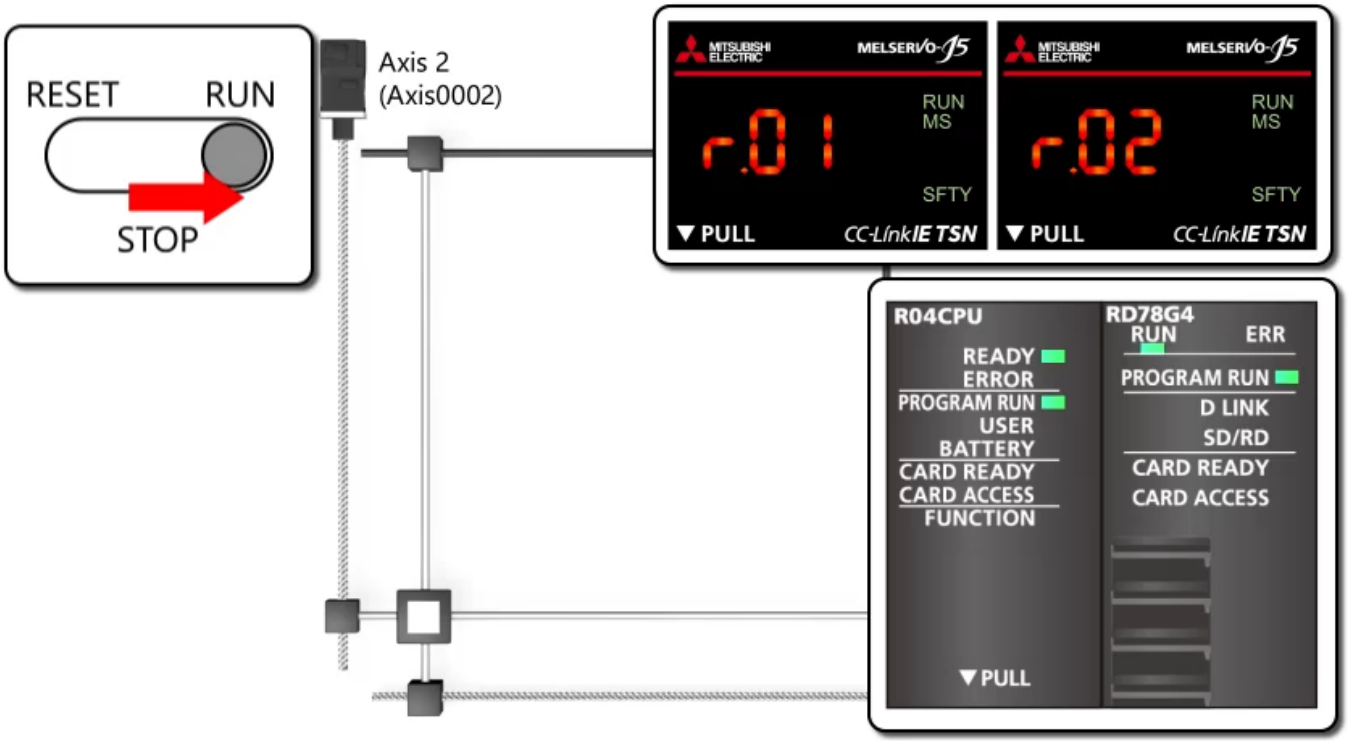

Axis 2
(Axis0002)
Click the play button at the lower left of the window.
Axis 1
(Axis0001)



Check the sample program operation.
Before starting operation, make sure that the programs of the PLC CPU and Motion module are installed.

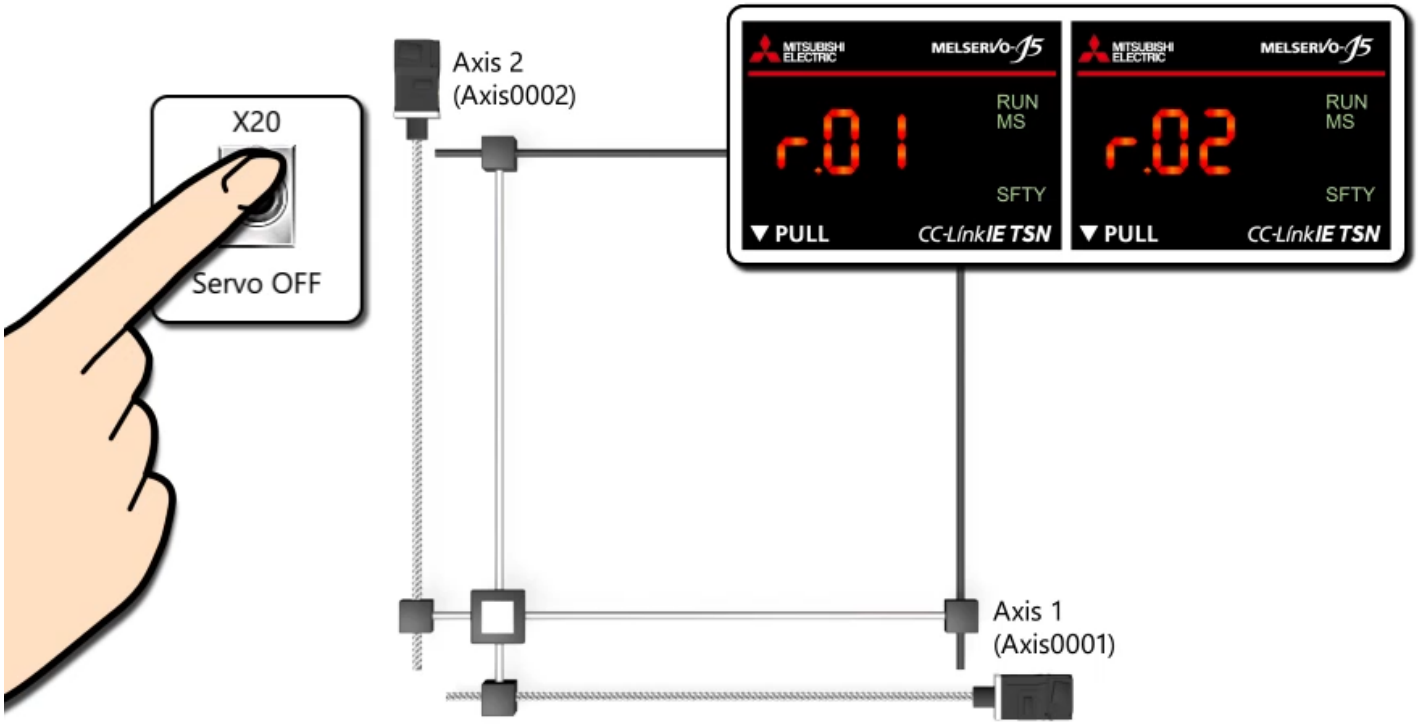


Set the RUN/STOP/RESET switch of the PLC CPU to RUN.
READY lamp and PROGRAM RUN lamp of
the programmable controller turn on.
RUN lamp of the Motion module turns on.

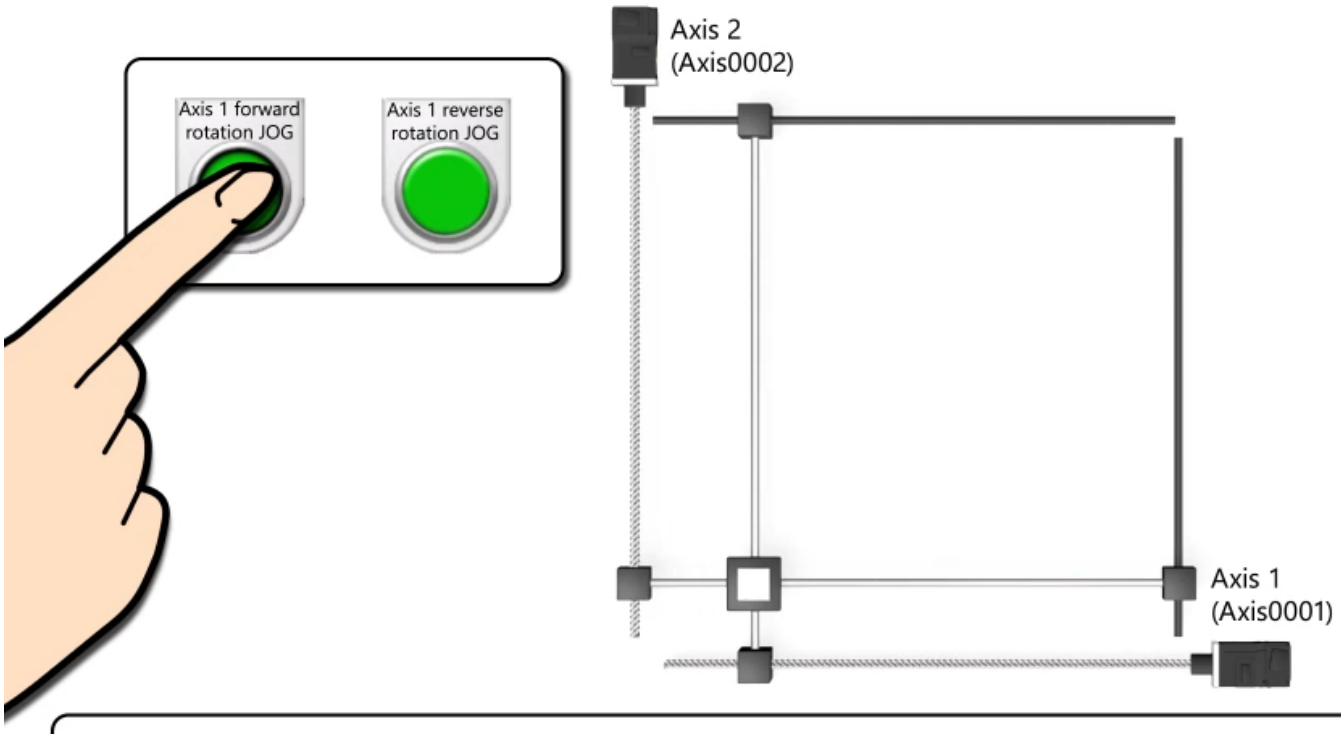


Wait until PROGRAM RUN lamp of the Motion module turns on.
"r.01" and "r.02" are displayed on the servo amplifier.
(The dot turns on.)
The servo motor is set to the servo ON state.

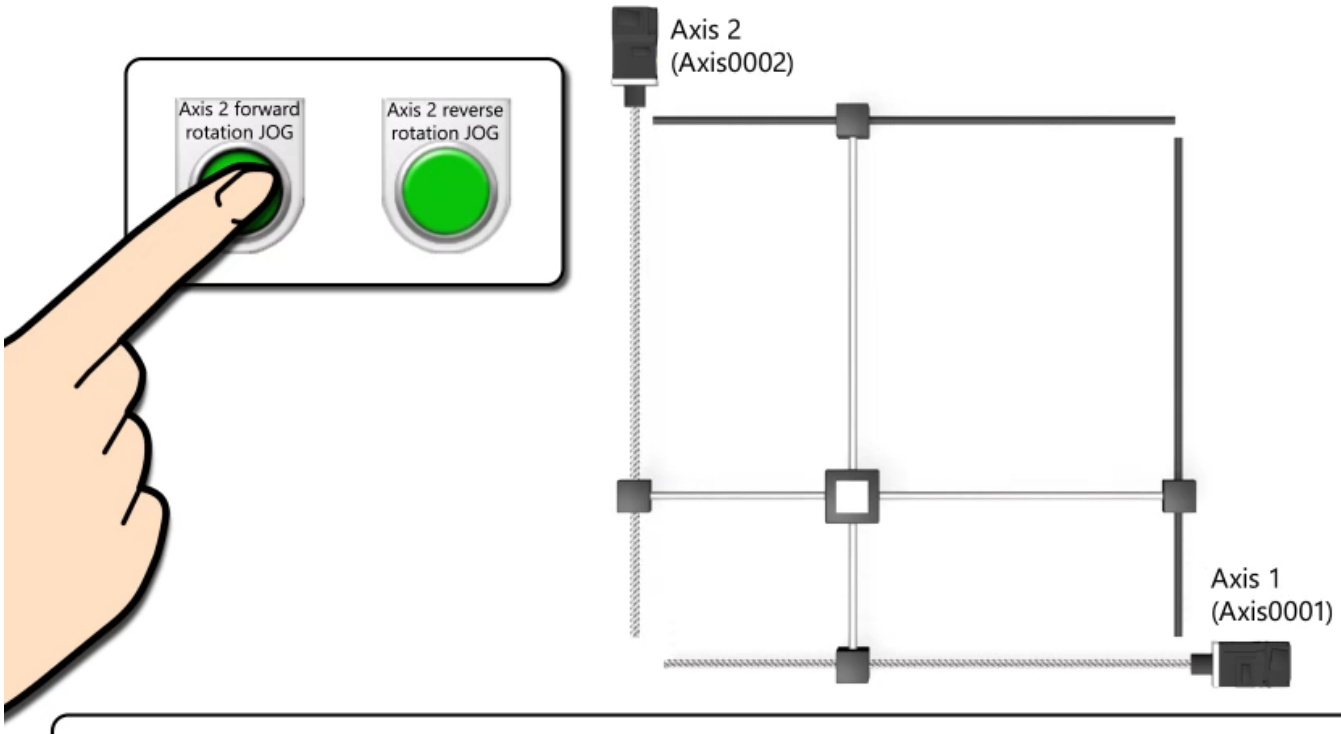
2.7 Operation Check



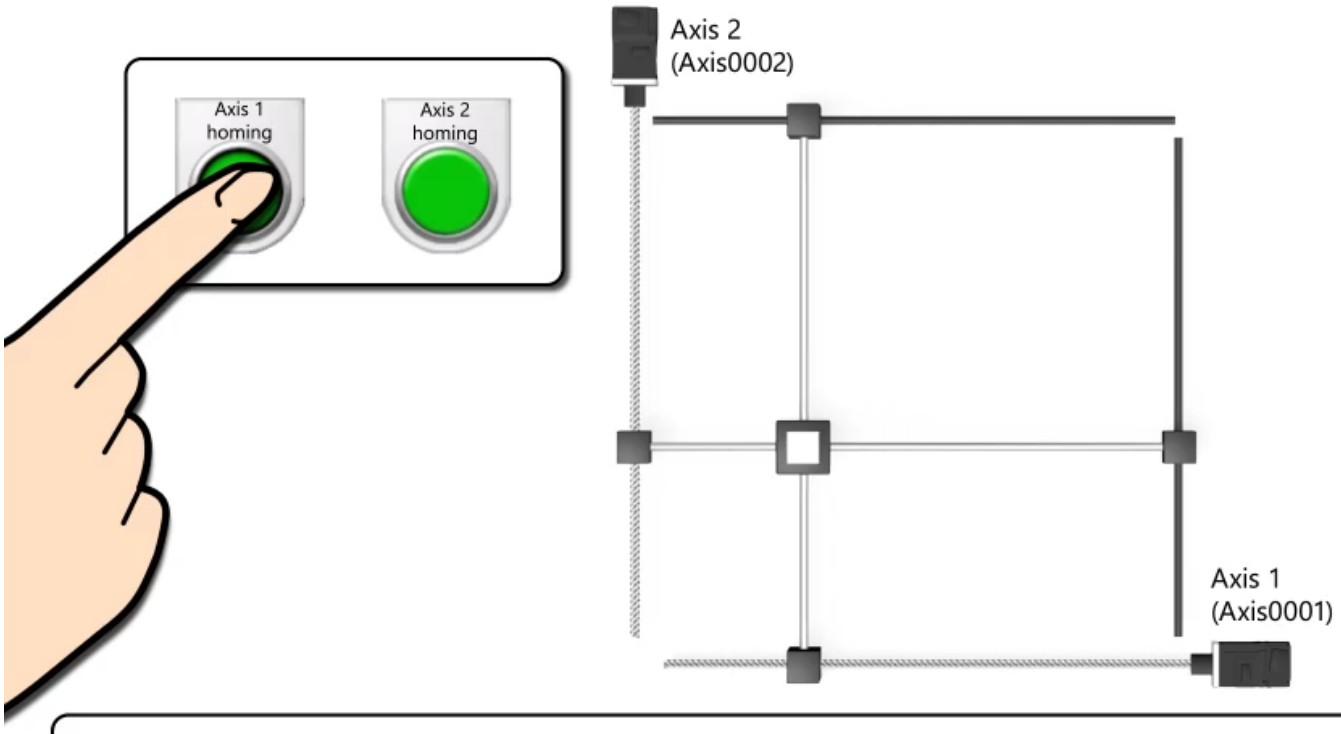
Turn on X20 to execute servo OFF.
"r.01" and "r.02" are displayed on the servo amplifier.
(The dot flashes.)
Turn off X20 to execute the servo ON again.



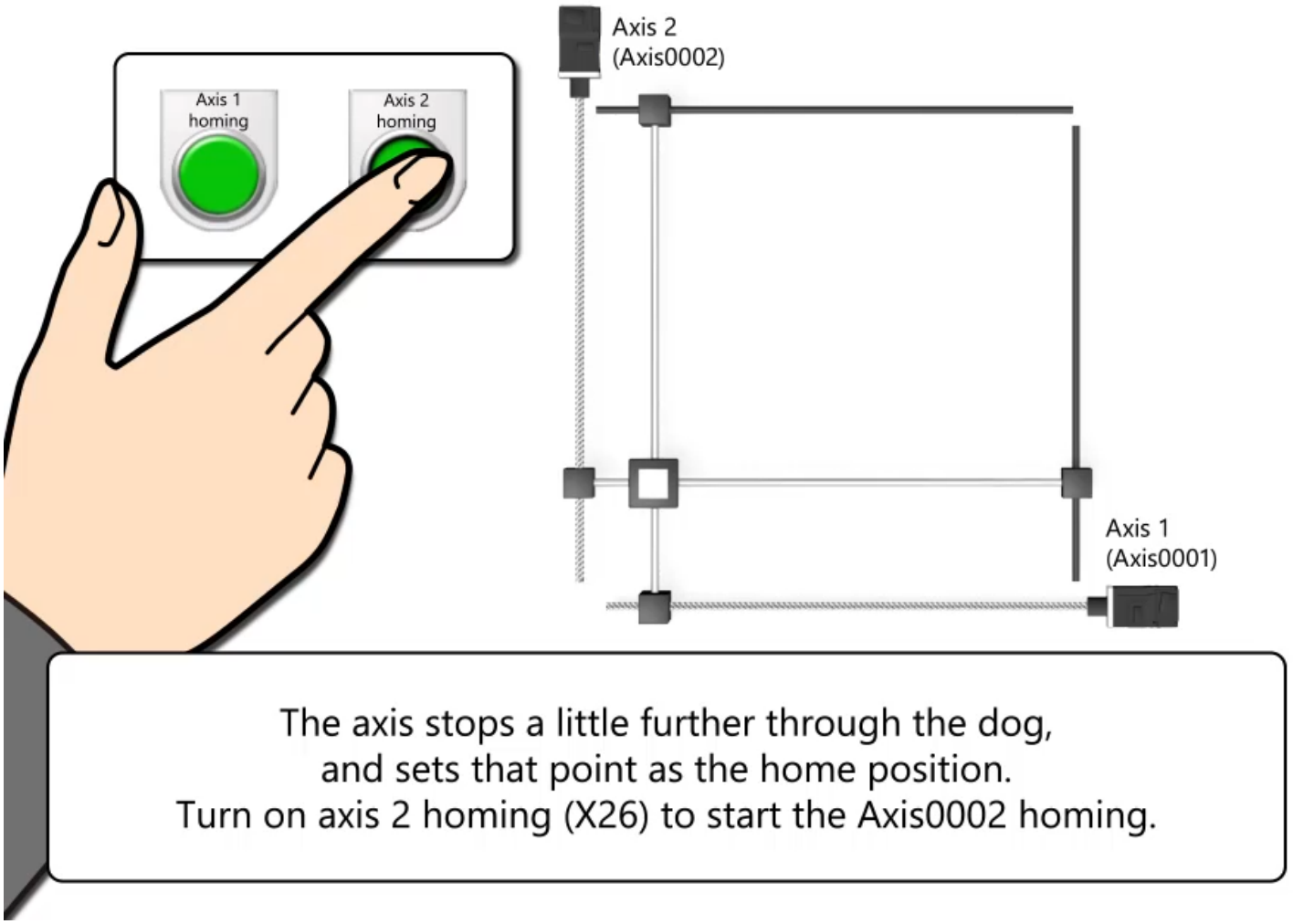
Turn on axis 1 forward rotation JOG (X21) to move to the address increase direction, and turn off to stop.
Turn on axis 1 reverse rotation JOG (X22) to move to the address decrease direction, and turn off to stop.



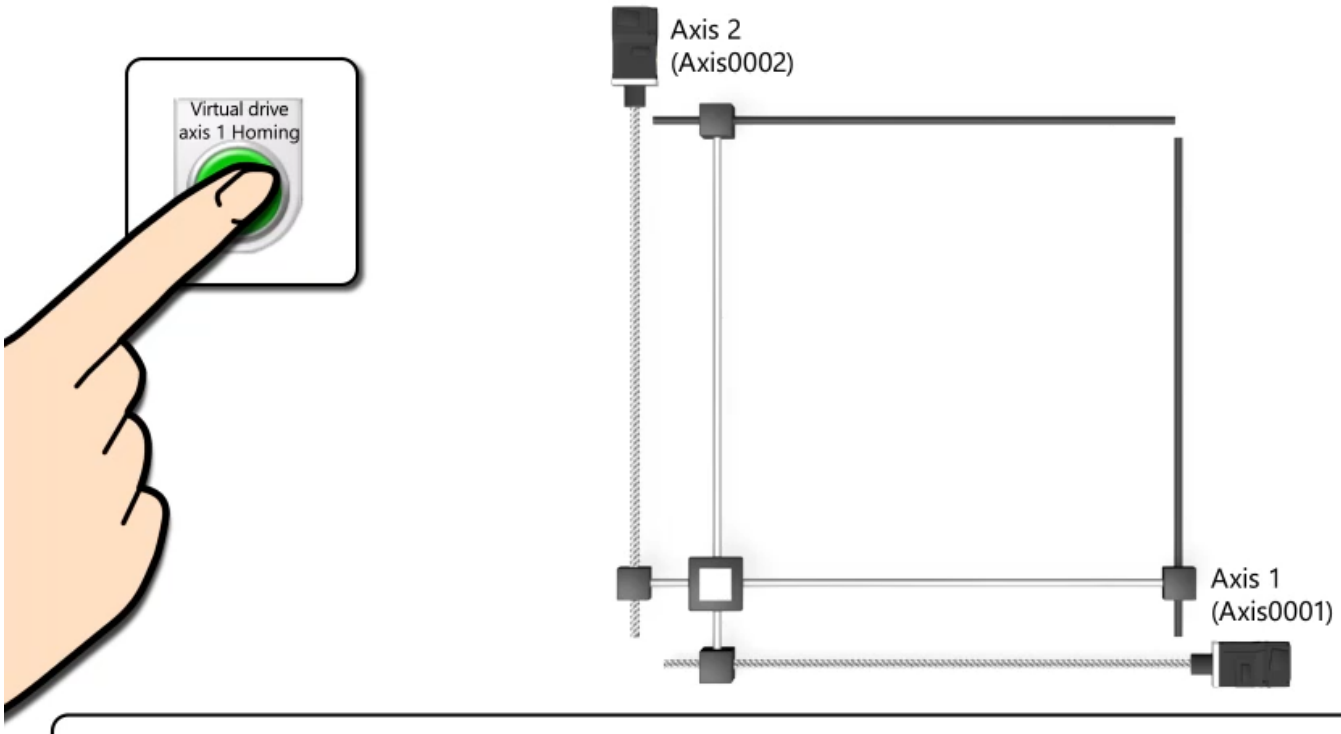
Turn on axis 2 forward rotation JOG (X23) to move to the address increase direction, and turn off to stop.
Turn on axis 2 reverse rotation JOG (X24) to move to the address decrease direction, and turn off to stop.



Turn on axis 1 homing (X25) to start the Axis0001 homing.
Execute homing of proximity dog type
(33 is subtracted from Pr.PT45)

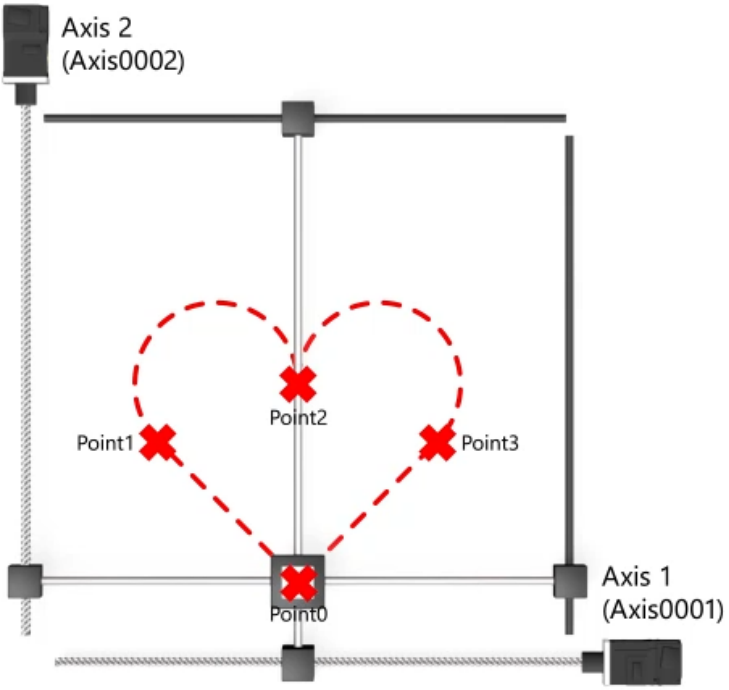


The axis stops a little further through the dog,
and sets that point as the home position.
Turn on axis 2 homing (X26) to start the Axis0002 homing.

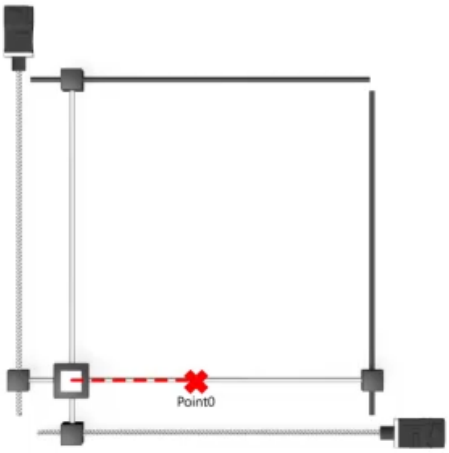


Turn on Virtual drive axis 1 Homing (X27) to start the VirtualAxis0001 homing (data set type). Axis0001 and Axis0002 will not move.

2.7 Operation Check

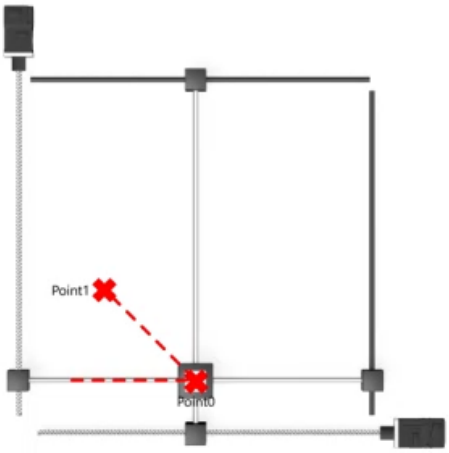


Turn on interpolation control start (X28) to start interpolation control.
The axes move to Point0 and stop for one second.
Then, Axis0001 and Axis0002 draw the shape as shown in Section 2.6.
The axes stop for one second, and then return to the home position.




```
//-----Interpolation-----//
//Intial Value Setting
IF(G_bInterpolation) THEN
    //Positioning Velocity,Acceleration,Deceleration,Jerk
    leVelocity      := 30000.0;
    leAcceleration  := 60000.0;
    leDeceleration  := 60000.0;
    leJerk          := 120000.0;
    //Request ON
    G_bInterpolationReq := TRUE;
ELSE
    //Request OFF
    G_bInterpolationReq := FALSE;
    //Reset Internal Signal
    bMoveLinear2Dwell_in := FALSE;
END_IF;

//Move to Point0
MC_MoveAbsolute_1(
    Axis            := Axis0001.AxisRef ,
    Execute         := G_bInterpolationReq ,
    Position        := G_lePoint0[0] ,
    Velocity        := leVelocity ,
    Acceleration    := leAcceleration ,
    Deceleration    := leDeceleration ,
    Jerk            := leJerk ,
    Done            => bMoveAbs1Done ,
    Busy            => bMoveAbs1Busy ,
    Active          => bMoveAbs1Active ,
    CommandAborted  => bMoveAbs1Aborted ,
```

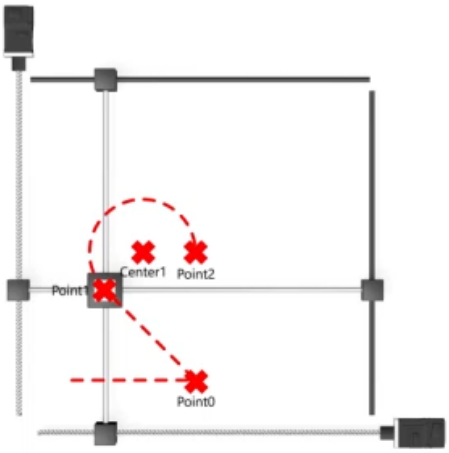
Check the program monitor.
The axes are moved to Point0 by single axis positioning control of Axis0001.
After reaching Point0, the axes stop for one second.




```
TON_1(IN:= bMoveAbs1Done ,PT:= T#1s ,Q=> bMoveAbs1Dwell_out);

//Group Enable
MC_GroupEnable_1(
    AxesGroup:= AxesGroup001.AxesGroupRef ,
    Execute:= bMoveAbs1Dwell_out ,
    Done=> bGrpEnDone ,
    Error=> bGrpEnError
);

//Move to Point1 Linear Interpolation
MCv_MoveLinearInterpolateAbsolute_1(
    AxesGroup       := AxesGroup001.AxesGroupRef ,
    Execute         := bGrpEnDone ,
    LinearAxes      := G_wAxesNum ,
    Position        := G_lePoint1 ,
    Velocity        := leVelocity ,
    Acceleration    := leAcceleration ,
    Deceleration    := leDeceleration ,
    Jerk            := leJerk ,
    VelocityMode    := MC_INTERPOLATE_SPEED_MODE__VectorSpeed ,
    Direction       := G_wDirection ,
    Busy            => bMoveLinear1Busy ,
    Active          => bMoveLinear1Active ,
    CommandAborted  => bMoveLinear1Aborted ,
    Error           => bMoveLinear1Error
);

//Move to Point2 Circular Interpolation
MCv_MoveCircularInterpolateAbsolute_1(
    AxesGroup       := AxesGroup001.AxesGroupRef ,
```

Enabling the axis group

MC_GroupEnable is executed to enable the axis group. After the axis group is enabled, the axes are controlled to move to Point1 by linear interpolation.

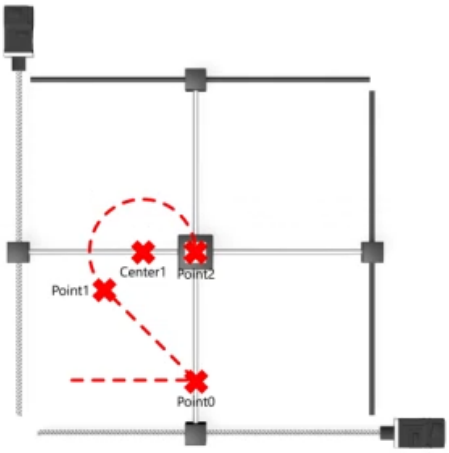



```
56      Direction       := G_wDirection ,
57      Busy            => bMoveLinear1Busy ,
58      Active          => bMoveLinear1Active ,
59      CommandAborted  => bMoveLinear1Aborted ,
60      Error           => bMoveLinear1Error
61  );
62
63  //Move to Point2 Circular Interpolation
64  MCv_MoveCircularInterpolateAbsolute_1(
65      AxesGroup       := AxesGroup001.AxesGroupRef ,
66      Execute         := bMoveLinear1Active ,
67      ContinuousUpdate:= FALSE ,
68      CircAxes        := G_wCircAxesNum ,
69      CircMode        := MC_CIRC_MODE__mcCenter ,
70      AuxPoint        := G_leCenter1 ,
71      EndPoint        := G_lePoint2 ,
72      PathChoice      := MC_CIRC_PATHCHOICE__mcCW ,
73      Velocity        := leVelocity ,
74      Acceleration    := leAcceleration ,
75      Deceleration    := leDeceleration ,
76      Jerk            := leJerk ,
77      BufferMode      := MC_BUFFER_MODE__mcBlendingNext ,
78      Busy            => bMoveCirc1Busy ,
79      Active          => bMoveCirc1Active ,
80      CommandAborted  => bMoveCirc1Aborted ,
81      Error           => bMoveCirc1Error
82  );
83
84  //Move to Point3 Circular Interpolation
85  MCv_MoveCircularInterpolateAbsolute_2(
86      AxesGroup       := AxesGroup001.AxesGroupRef ,
```

Center point specification

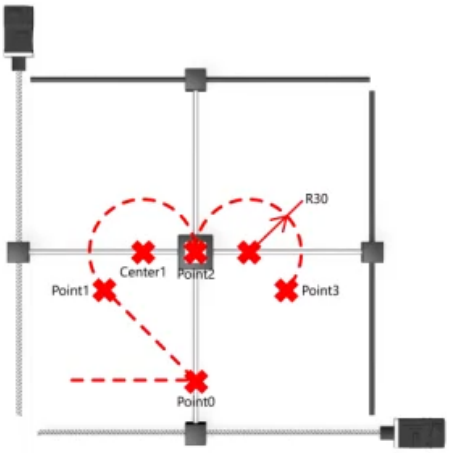
CW

Next, the axes are controlled to move from Point1 to Point2 by circular interpolation.




```
    Direction       := G_wDirection ,
    Busy            => bMoveLinear1Busy ,
    Active          => bMoveLinear1Active ,
    CommandAborted  => bMoveLinear1Aborted ,
    Error           => bMoveLinear1Error
);

//Move to Point2 Circular Interpolation
MCv_MoveCircularInterpolateAbsolute_1(
    AxesGroup       := AxesGroup001.AxesGroupRef ,
    Execute         := bMoveLinear1Active ,
    ContinuousUpdate:= FALSE ,
    CircAxes        := G_wCircAxesNum ,
    CircMode        := MC_CIRC_MODE__mcCenter ,
    AuxPoint        := G_leCenter1 ,
    EndPoint        := G_lePoint2 ,
    PathChoice      := MC_CIRC_PATHCHOICE__mcCW ,
    Velocity        := leVelocity ,
    Acceleration    := leAcceleration ,
    Deceleration    := leDeceleration ,
    Jerk            := leJerk ,
    BufferMode      := MC_BUFFER_MODE__mcBlendingNext ,
    Busy            => bMoveCirc1Busy ,
    Active          => bMoveCirc1Active ,
    CommandAborted  => bMoveCirc1Aborted ,
    Error           => bMoveCirc1Error
);

//Move to Point3 Circular Interpolation
MCv_MoveCircularInterpolateAbsolute_2(
    AxesGroup       := AxesGroup001.AxesGroupRef ,
```

From Point1 to Point2, the center point specification method has been set.
The coordinates of the center point are specified for AuxPoint.

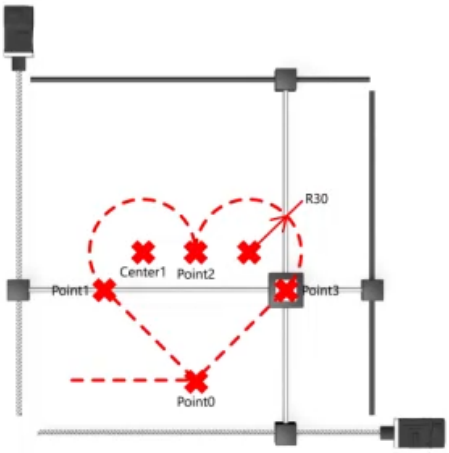



```
84  //Move to Point3 Circular Interpolation
85  MCv_MoveCircularInterpolateAbsolute_2(
86      AxesGroup       := AxesGroup001.AxesGroupRef ,
87      Execute         := bMoveCirc1Active ,
88      ContinuousUpdate:= FALSE ,
89      CircAxes        := G_wCircAxesNum ,
90      CircMode        := MC_CIRC_MODE__mcRadius ,
91      AuxPoint        := G_leRadius ,
92      EndPoint        := G_lePoint3 ,
93      PathChoice      := MC_CIRC_PATHCHOICE__mcCWLongWay ,
94      Velocity        := leVelocity ,
95      Acceleration    := leAcceleration ,
96      Deceleration    := leDeceleration ,
97      Jerk            := leJerk ,
98      BufferMode      := MC_BUFFER_MODE__mcBuffered ,
99      Busy            => bMoveCirc2Busy ,
100     Active          => bMoveCirc2Active ,
101     CommandAborted  => bMoveCirc2Aborted ,
102     Error           => bMoveCirc2Error
103 );
104
105 //Move to Point0 Linear Interpolation
106 MCv_MoveLinearInterpolateAbsolute_2(
107     AxesGroup       := AxesGroup001.AxesGroupRef ,
108     Execute         := bMoveCirc2Active ,
109     ContinuousUpdate:= FALSE ,
110     LinearAxes      := G_wAxesNum ,
111     Position        := G_lePoint0 ,
112     Velocity        := leVelocity ,
113     Acceleration    := leAcceleration ,
```

Radius specification

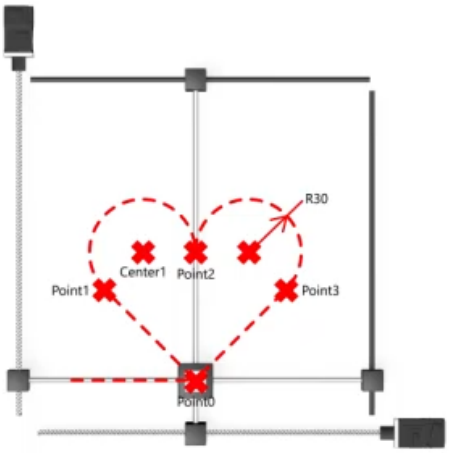
CW Detour

Next, the axes are controlled to move from Point2 to Point3 by circular interpolation.
From Point2 to Point3, the radius specification method has been set.
The radius is specified for AuxPoint.



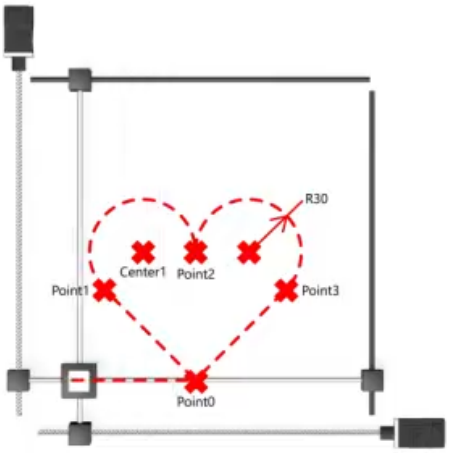

```
100     Active              => bMoveCirc2Active ,
101     CommandAborted      => bMoveCirc2Aborted ,
102     Error               => bMoveCirc2Error
103 );
104
105 //Move to Point0 Linear Interpolation
106 MCv_MoveLinearInterpolateAbsolute_2(
107     AxesGroup           := AxesGroup001.AxesGroupRef ,
108     Execute             := bMoveCirc2Active ,
109     ContinuousUpdate    := FALSE ,
110     LinearAxes          := G_wAxesNum ,
111     Position            := G_lePoint0 ,
112     Velocity            := leVelocity ,
113     Acceleration        := leAcceleration ,
114     Deceleration        := leDeceleration ,
115     Jerk                := leJerk ,
116     VelocityMode        := MC_INTERPOLATE_SPEED_MODE__VectorSpeed ,
117     Direction           := G_wDirection ,
118     BufferMode          := MC_BUFFER_MODE__mcBlendingNext ,
119     Done                => bMoveLinear2Done ,
120     Busy                => bMoveLinear2Busy ,
121     Active              => bMoveLinear2Active ,
122     CommandAborted      => bMoveLinear2Aborted ,
123     Error               => bMoveLinear2Error
124 );
125
126 //Dwell
127 SET(bMoveLinear2Done, bMoveLinear2Dwell_in);
128 TON_2(IN:= bMoveLinear2Dwell_in ,PT:= T#1s ,Q=> bMoveLinear2Dwell_out);
129
130 //Group Disable
```

Then, the axes are controlled to move from Point3 to Point0 by linear interpolation.



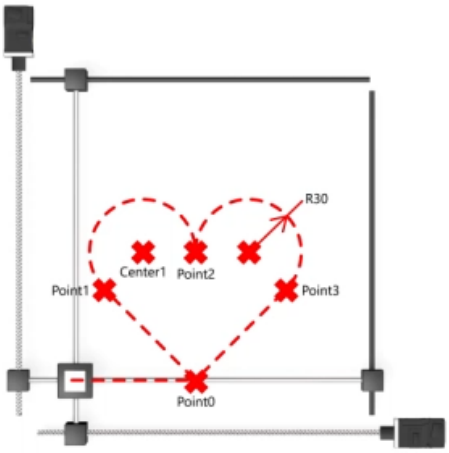

```
100         Active          => bMoveCirc2Active ,
101         CommandAborted  => bMoveCirc2Aborted ,
102         Error           => bMoveCirc2Error
103 );
104
105 //Move to Point0 Linear Interpolation
106 MCv_MoveLinearInterpolateAbsolute_2(
107     AxesGroup       := AxesGroup001.AxesGroupRef ,
108     Execute         := bMoveCirc2Active ,
109     ContinuousUpdate:= FALSE ,
110     LinearAxes      := G_wAxesNum ,
111     Position        := G_lePoint0 ,
112     Velocity        := leVelocity ,
113     Acceleration    := leAcceleration ,
114     Deceleration    := leDeceleration ,
115     Jerk            := leJerk ,
116     VelocityMode    := MC_INTERPOLATE_SPEED_MODE__VectorSpeed ,
117     Direction       := G_wDirection ,
118     BufferMode      := MC_BUFFER_MODE__mcBlendingNext ,
119     Done            => bMoveLinear2Done ,
120     Busy            => bMoveLinear2Busy ,
121     Active          => bMoveLinear2Active ,
122     CommandAborted  => bMoveLinear2Aborted ,
123     Error           => bMoveLinear2Error
124 );
125
126 //Dwell
127 SET(bMoveLinear2Done, bMoveLinear2Dwell_in);
128 TON_2(IN:= bMoveLinear2Dwell_in ,PT:= T#1s ,Q=> bMoveLinear2Dwell_out);
129
130 //Group Disable
```

Dwell is performed and the axis group is disabled after the axes reach Point0.



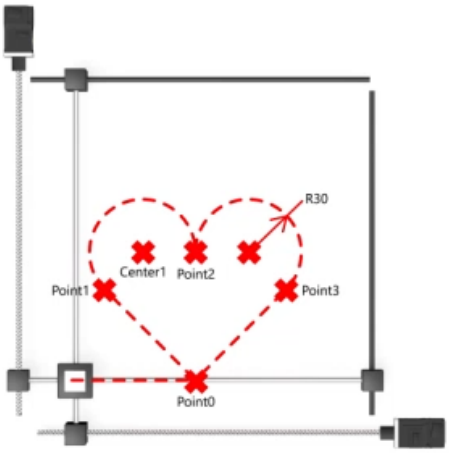

```
//Move to Home Point
MC_MoveAbsolute_2(
    Axis            := Axis0001.AxisRef ,
    Execute         := bGrpDsblDone & bMoveLinear2Dwell_out ,
    ContinuousUpdate:= FALSE ,
    Position        := 0.0 ,
    Velocity        := leVelocity ,
    Acceleration    := leAcceleration ,
    Deceleration    := leDeceleration ,
    Jerk            := leJerk ,
    BufferMode      := MC_BUFFER_MODE__mcAborting ,
    Done            => bMoveAbs2Done ,
    Busy            => bMoveAbs2Busy ,
    Active          => bMoveAbs2Active ,
    CommandAborted  => bMoveAbs2Aborted ,
    Error           => bMoveAbs2Error
);

//Dwell
TON_3(IN:= bMoveAbs2Done ,PT:= T#1s ,Q=> bMoveAbs2Dwell_out);

//Error Signal,Aborted Signal
bError := bMoveAbs1Error OR bGrpEnError OR bMoveLinear1Error OR bMoveCirc1Error
          OR bMoveCirc2Error OR bMoveLinear2Error OR bMoveAbs2Error;
bAborted := bMoveAbs1Aborted OR bMoveLinear1Aborted OR bMoveCirc1Aborted
            OR bMoveCirc2Aborted OR bMoveLinear2Aborted OR bMoveAbs2Aborted;

//Done Signal
G_bInterpolationDone := (bMoveAbs2Dwell_out OR bError OR bAborted )
                        & (AxesGroup001.Md.GroupStatus = 0);
```

Finally, the axes are returned to the home position from Point0 by single axis positioning control.




```
//Move to Home Point
MC_MoveAbsolute_2(
    Axis            := Axis0001.AxisRef ,
    Execute         := bGrpDsblDone & bMoveLinear2Dwell_out ,
    ContinuousUpdate:= FALSE ,
    Position        := 0.0 ,
    Velocity        := leVelocity ,
    Acceleration    := leAcceleration ,
    Deceleration    := leDeceleration ,
    Jerk            := leJerk ,
    BufferMode      := MC_BUFFER_MODE__mcAborting ,
    Done            => bMoveAbs2Done ,
    Busy            => bMoveAbs2Busy ,
    Active          => bMoveAbs2Active ,
    CommandAborted  => bMoveAbs2Aborted ,
    Error           => bMoveAbs2Error
);

//Dwell
TON_3(IN:= bMoveAbs2Done ,PT:= T#1s ,Q=> bMoveAbs2Dwell_out);

//Error Signal,Aborted Signal
bError := bMoveAbs1Error OR bGrpEnError OR bMoveLinear1Error OR bMoveCirc1Error
          OR bMoveCirc2Error OR bMoveLinear2Error OR bMoveAbs2Error;
bAborted := bMoveAbs1Aborted OR bMoveLinear1Aborted OR bMoveCirc1Aborted
            OR bMoveCirc2Aborted OR bMoveLinear2Aborted OR bMoveAbs2Aborted;

//Done Signal
G_bInterpolationDone := (bMoveAbs2Dwell_out OR bError OR bAborted )
                        & (AxesGroup001.Md.GroupStatus = 0);
```

The program ends after the dwell time is elapsed, the internal trigger is reset, completion signal is turned on and transferred to the PLC CPU.




```
//Move to Home Point
MC_MoveAbsolute_2(
    Axis            := AxisOOO1.AxisRef ,
    Execute         := bGrpDsblDone & bMoveLinear2Dwell_out ,
    ContinuousUpdate:= FALSE ,
    Position        := 0.0 ,
    Velocity        := leVelocity ,
    Acceleration    := leAcceleration ,
    Deceleration    := leDeceleration ,
    Jerk            := leJerk ,
    BufferMode      := MC_BUFFER_MODE__mcAborting ,
    Done            => bMoveAbs2Done ,
    Busy            => bMoveAbs2Busy ,
    Active          => bMoveAbs2Active ,
    CommandAborted  => bMoveAbs2Aborted ,
    Error           => bMoveAbs2Error
);

//Dwell
TON_3(IN:= bMoveAbs2Done ,PT:= T#1s ,Q=> bMoveAbs2Dwell_out);

//Error Signal,Aborted Signal
bError := bMoveAbs1Error OR bGrpEnError OR bMoveLinear1Error OR bMoveCirc1Error
          OR bMoveCirc2Error OR bMoveLinear2Error OR bMoveAbs2Error;
bAborted := bMoveAbs1Aborted OR bMoveLinear1Aborted OR bMoveCirc1Aborted
            OR bMoveCirc2Aborted OR bMoveLinear2Aborted OR bMoveAbs2Aborted;

//Done Signal
G_bInterpolationDone := (bMoveAbs2Dwell_out OR bError OR bAborted )
                        & (AxesGroupOO1.Md.GroupStatus = 0);
```

This completes the operation check.
Go to the next page.

2.8 Summary of This Chapter

In this chapter, you have learned:

- Steps of Interpolation Control
- Axes Group
- FBs for Interpolation Control
- Linear Interpolation
- Circular Interpolation
- Program Example
- Operation Check

Important points

Steps of Interpolation Control	• Enable the axes group before executing interpolation control. • Disable the axes group after completing interpolation control.
Axes Group	• Register the target axes for interpolation control to one axes group. • Register the axis name to the configuration axis of the axes group parameter. • The position command unit and velocity command unit can be set.
FBs for Interpolation Control	• MC_GroupEnable/MC_GroupDisable is used to enable/disable the axes group. • MCv_MoveLinearInterpolate*** is used for linear interpolation, and MCv_MoveCircularInterpolate*** is used for circular interpolation.
Linear Interpolation	• You have learned how to set the following four input signals of MCv_MoveLinearInterpolateAbsolute. • LinearAxes, Position, VelocityMode, Direction
Circular Interpolation	• You have learned how to set the following five input signals of MCv_MoveCircularInterpolateAbsolute. • CircAxes, CircMode, AuxPoint, EndPoint, PathChoice
Program Example	• You have learned about the programs that draw the combined path of linear interpolation and circular interpolation.
Operation Check	• You have checked the operation of the sample program in the video.

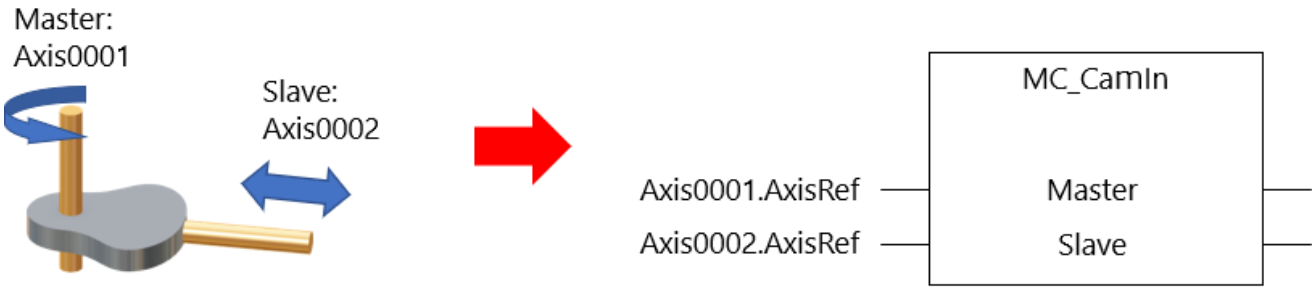
Chapter 3 Synchronous Control

3.1 Concept of Synchronous Control

Synchronous control is a software function that controls mechanical components, such as gears, transmissions, and cams, by transmitting the position information (command) of the slave axis that is synchronized with the master axis.

The following FBs are used to describe the relationship between the master axis and slave axis for the synchronous control.

FB name	Control
MC_CamIn	Executes cam operation.
MC_GearIn	Sets the speed ratio between the master axis and slave axis, and starts gear operation.
MC_CombineAxes	Combines the motion of two axes according to the selected method, and outputs the combined motion to the third axis.



3.1.1 Cam operation MC_CamIn

The following describes MC_CamIn, which is an FB that executes cam operation.

```
MC_CamIn_1(
    Master              := Axis0001.AxisRef ,
    Slave               := Axis0002.AxisRef ,
    Execute             := bCamInExe ,
    ContinuousUpdate    := FALSE ,
    MasterOffset        := 0.0 ,
    SlaveOffset         := 0.0 ,
    MasterScaling       := 1.0 ,
    SlaveScaling        := 1.0 ,
    MasterStartDistance := 0.0 ,
    MasterSyncPosition  := 0.0 ,
    StartMode           := MC_START_MODE__mcImmediate ,
    MasterValueSource   := MC_SOURCE__mcSetValue ,
    CamTableID          := CamTableID ,
    BufferMode          := MC_BUFFER_MODE__mcAborting ,
    Options             := H0 ,
    InSync          => bInSync ,
    Busy            => bCamInBusy ,
    Active          => bCamInActive ,
    CommandAborted  => bCamInAborted ,
    Error           => bCamInError ,
    ErrorID         => uCamInErrorID ,
    EndOfProfile    => bEndOfProfile
);
```

<FB specification (excerpt)>

I/O variable name		Variable name	Data type	Description
I/O	Master axis	Master	AXIS_REF	Specifies the master axis.
	Slave axis	Slave	AXIS_REF	Specifies the slave axis.
Input	Start	Execute	BOOL	Executes the FB when it is TRUE.
	Continuous update	ContinuousUpdate	BOOL	While it is TRUE, the master axis offset, slave axis offset, master axis scaling, slave axis scaling, and cam table ID can be continuously changed.
	Master axis offset	MasterOffset	LREAL	Offsets the phase of the master axis. The initial value is "0.0".
	Slave axis offset	SlaveOffset	LREAL	Offsets the displacement of the slave axis. The initial value is "0.0".
	Master axis scaling	MasterScaling	LREAL	Enlarges or reduces the cam table. The initial value is "1.0".
	Slave axis scaling	SlaveScaling	LREAL	Increases or decreases the stroke amount of the cam table. The initial value is "1.0".
	Master axis follow-up distance	MasterStartDistance	LREAL	Specifies the position of the master axis for the output value (OutputData) to start synchronization.

	Master axis synchronization start position	MasterSyncPosition	LREAL	Specifies the position of the master axis for the current value per cycle (MC_CamIn.InputPerCycle) to start synchronization.	(Note)
	Start mode	StartMode	INT (MC_START_MODE)	Specifies the timing to start cam operation.	
	Master axis data source selection	MasterValueSource	INT (MC_SOURCE)	Specifies the data source of the master axis.	
	Cam table ID	CamTableID	MC_CAM_ID	Specifies the cam ID. For details, refer to 3.4 (3).	
	Buffer mode	BufferMode	INT (MC_BUFFER_MODE)	Selects the buffer mode.	
	Option	Options	DWORD(HEX)	Sets the functional option in bits.	
Output	In synchronization	InSync	BOOL	Becomes TRUE when the output axis starts synchronization.	
	Executing	Busy	BOOL	Becomes TRUE during FB execution.	
	Controlling	Active	BOOL	Becomes TRUE when the FB is controlling an axis.	
	Abortion of execution	CommandAborted	BOOL	Becomes TRUE when the execution is interrupted.	
	Error	Error	BOOL	Becomes TRUE when an error occurs in the FB.	
	Error code	ErrorID	WORD(UINT)	Returns the error code of the error that has occurred in the FB.	
	Cam cycle completion	EndOfProfile	BOOL	Becomes TRUE by 1-cycle movement only for one scan in the program cycle.	

(Note) The details are omitted because this course uses the default values.

3.1.2 Gear operation MC_GearIn

The following describes MC_GearIn, which is an FB that performs gear operation.

```
MC_GearIn_1(
    Master              := Axis0001.AxisRef ,
    Slave               := Axis0002.AxisRef ,
    Execute             := bGearInExe ,
    ContinuousUpdate    := FALSE ,
    RatioNumerator      := dNumerator ,
    RatioDenominator    := udDenominator ,
    MasterValueSource   := MC_SOURCE__mcSetValue ,
    Acceleration        := leAcc ,
    Deceleration        := leDec ,
    Jerk                := leJerk ,
    BufferMode          := MC_BUFFER_MODE__mcAborting ,
    Options             := H0 ,
    InGear          => bInGear ,
    Busy            => bGearInBusy ,
    Active          => bGearInActive ,
    CommandAborted  => bGearInAborted ,
    Error           => bGearInError ,
    ErrorID         => uGearInErrorID
);
```

<FB specification (excerpt)

I/O variable name		Variable name	Data type	Description
I/O	Master Axis	Master	AXIS_REF	Specifies the master axis.
	Slave axis	Slave	AXIS_REF	Specifies the slave axis.
Input	Execution command	Execute	BOOL	Executes the FB when it is TRUE.
	Continuous update	ContinuousUpdate	BOOL	While it is TRUE, the gear ratio numerator, gear ratio denominator, acceleration, and deceleration can be continuously changed.
	Gear ratio numerator	RatioNumerator	DINT	Specifies the gear ratio numerator.
	Gear ratio denominator	RationDenominator	DWORD(UDINT)	Specifies the gear ratio denominator.
	Master axis data source selection	MasterValueSource	INT (MC_SOURCE)	Specifies the data source of the master axis.
	Acceleration	Acceleration	LREAL	Specifies the acceleration rate.
	Deceleration	Deceleration	LREAL	Specifies the deceleration rate.
	Jerk	Jerk	LREAL	Specifies the jerk at the start of acceleration/deceleration.
	Buffer mode	BufferMode	INT (MC_BUFFER_MODE)	Selects the buffer mode.
	Option	Options	DWORD(HEX)	Sets the functional option in bits.
Output	Gear ratio reached	InGear	BOOL	Becomes TRUE when the target speed is reached.

	Executing	Busy	BOOL	Becomes TRUE during FB execution.
	Controlling	Active	BOOL	Becomes TRUE when the FB is controlling an axis.
	Abortion of execution	CommandAborted	BOOL	Becomes TRUE when the execution is interrupted.
	Error	Error	BOOL	Becomes TRUE when an error occurs in the FB.
	Error code	ErrorID	WORD(UINT)	Returns the error code of the error that has occurred in the FB.

3.1.3 Addition/subtraction positioning (composite gear) operation

The following describes MC_CombineAxes, which is an FB that performs addition/subtraction positioning (composite gear) operation.

```
MC_CombineAxes_1(
    Master1              := Axis0001.AxisRef ,
    Master2              := Axis0002.AxisRef ,
    Slave                := Axis0003.AxisRef ,
    Execute              := bCombineExe ,
    ContinuousUpdate     := FALSE ,
    CombineMode          := MC_COMBINE_MODE__mcAddAxes ,
    GearRatioNumeratorM1    := dNumerator1 ,
    GearRatioDenominatorM1  := udDenominator1 ,
    GearRatioNumeratorM2    := dNumerator2 ,
    GearRatioDenominatorM2  := udDenominator2 ,
    MasterValueSourceM1 := MC_SOURCE__mcSetValue ,
    MasterValueSourceM2 := MC_SOURCE__mcSetValue ,
    BufferMode           := MC_BUFFER_MODE__mcAborting ,
    Options              := H0 ,
    InSync          => bInSync ,
    Busy            => bCombineAxesBusy ,
    Active          => bCombineAxesActive ,
    CommandAborted  => bCombineAxesAborted ,
    Error           => bCombineAxesError ,
    ErrorID         => uCombineAxesErrorID
);

```

<FB specification (excerpt)>

I/O variable name		Variable name	Data type	Description
I/O	Master axis 1	Master1	AXIS_REF	Specifies the master axis 1.
	Master axis 2	Master2	AXIS_REF	Specifies the master axis 2.
	Slave axis	Slave	AXIS_REF	Specifies the slave axis.
Input	Execution command	Execute	BOOL	Executes the FB when it is TRUE.
	Continuous update	ContinuousUpdate	BOOL	While it is TRUE, the addition/subtraction method, gear ratio numerator, and gear ratio denominator can be continuously changed.
	Addition/subtraction method selection	CombineMode	INT (MC_COMBINE_MODE)	Specifies the method to combine the travel amount of the master axis 1 and master axis 2.
	Master axis 1 gear ratio numerator	RatioNumeratorM1	DINT	Specifies the gear ratio numerator of the master axis 1.
	Master axis 1 gear ratio denominator	RationDenominatorM1	DWORD(UDINT)	Specifies the gear ratio denominator of the master axis 1.
	Master axis 2 gear ratio numerator	RatioNumeratorM2	DINT	Specifies the gear ratio numerator of the master axis 2.
	Master axis 2 gear ratio denominator	RationDenominatorM2	DWORD(UDINT)	Specifies the gear ratio denominator of the master axis 2.
	Master axis 1 data source selection	MasterValueSourceM1	INT (MC_SOURCE)	Specifies the data source of the master axis 1.

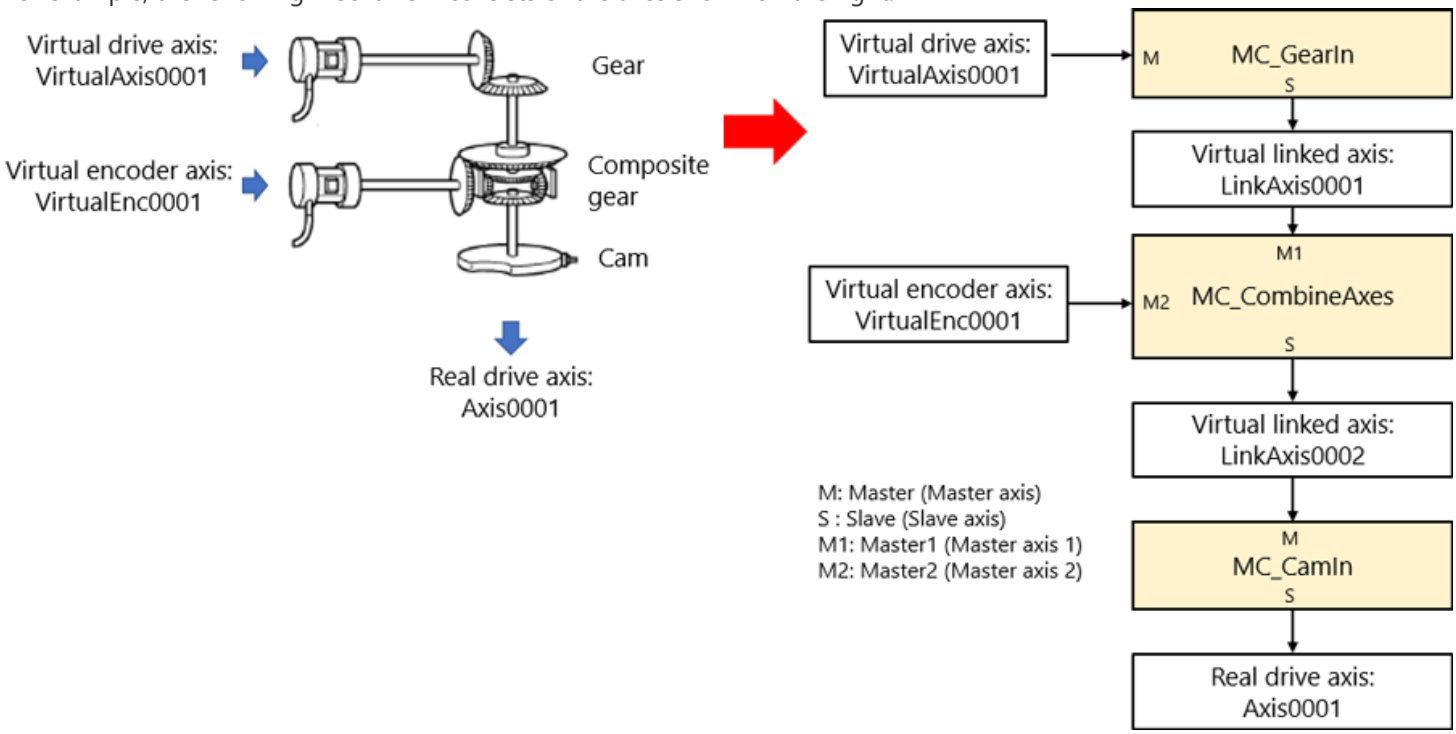
	Master axis 2 data source selection	MasterValueSourceM2	INT (MC_SOURCE)	Specifies the data source of the master axis 2.
	Buffer mode	BufferMode	INT (MC_BUFFER_MODE)	Selects the buffer mode.
	Option	Options	DWORD(HEX)	Sets the functional option in bits.
Output	In synchronization	InSync	BOOL	Becomes TRUE when the slave starts synchronization.
	Executing	Busy	BOOL	Becomes TRUE during FB execution.
	Controlling	Active	BOOL	Becomes TRUE when the FB is controlling an axis.
	Abortion of execution	CommandAborted	BOOL	Becomes TRUE when the execution is interrupted.
	Error	Error	BOOL	Becomes TRUE when an error occurs in the FB.
	Error code	ErrorID	WORD(UINT)	Returns the error code of the error that has occurred in the FB.

3.2 Virtual Axis

Motion control uses virtual axes to generate virtual commands and position data.

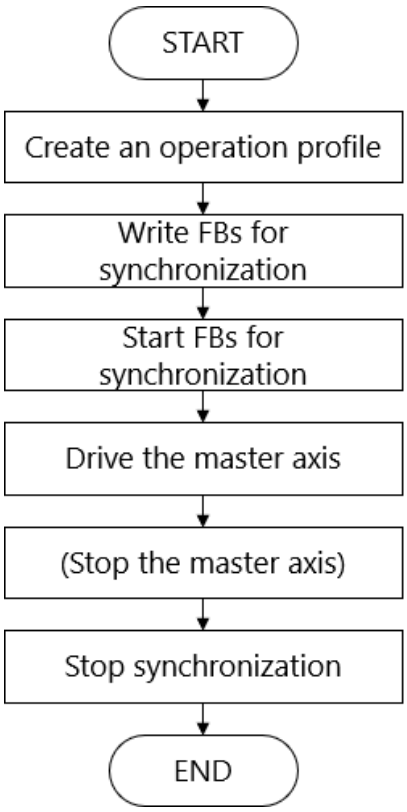
Axis type	Description
Virtual drive axis	Generates virtual commands in the motion system. No actual drive unit is used.
Virtual encoder axis	Generates the current position data from the variables of the motion system. It is used as an input axis for the single axis synchronous control. It cannot be used as a slave axis.
Virtual linked axis	Connects FBs of the single axis synchronous control. Only the data required for the single axis synchronous control is defined.

For example, the following mechanism consists of the axes shown on the right.



3.3 Steps of Synchronous Control

The following shows the steps of the synchronous control process.



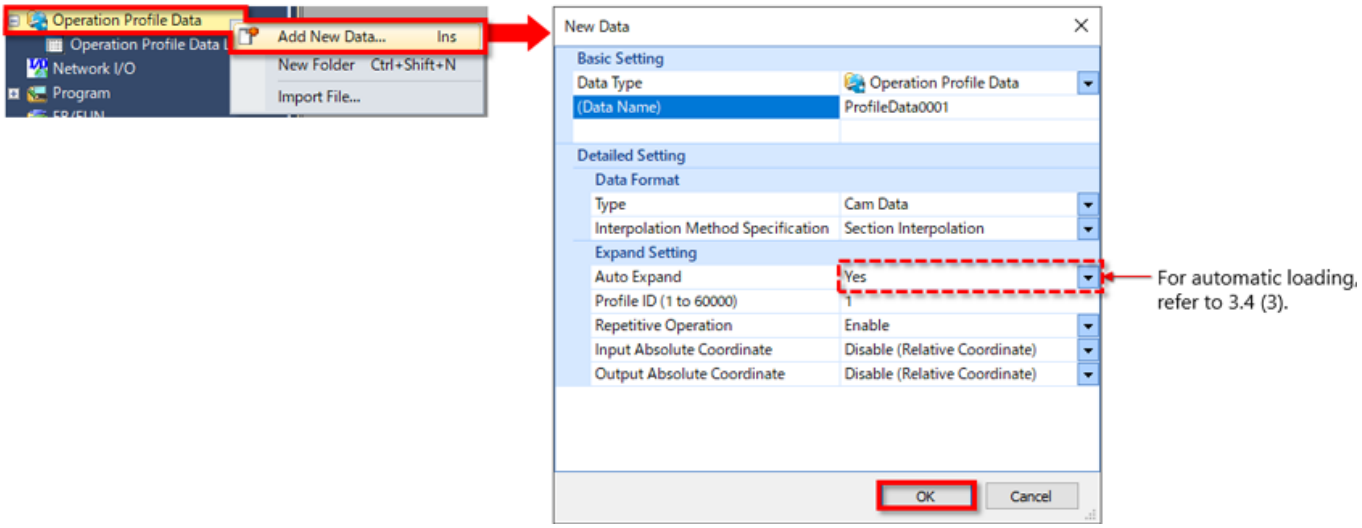
- Before using MC_CamIn, create an operation profile to define the cam pattern.
- Describe the FBs for synchronous control in the program.
- Execute the FBs for synchronous control. The status of the slave axis becomes "SynchronizedMotion".
- When the master axis is driven, the slave axis moves synchronously.
- MC_Stop is executed for the slave axis. The axis status changes from "SynchronizedMotion" to "Standstill", and synchronization is deactivated.

3.4 Operation Profile

Waveform data used for control is collectively called an operation profile.
The following describes how to create cam data.

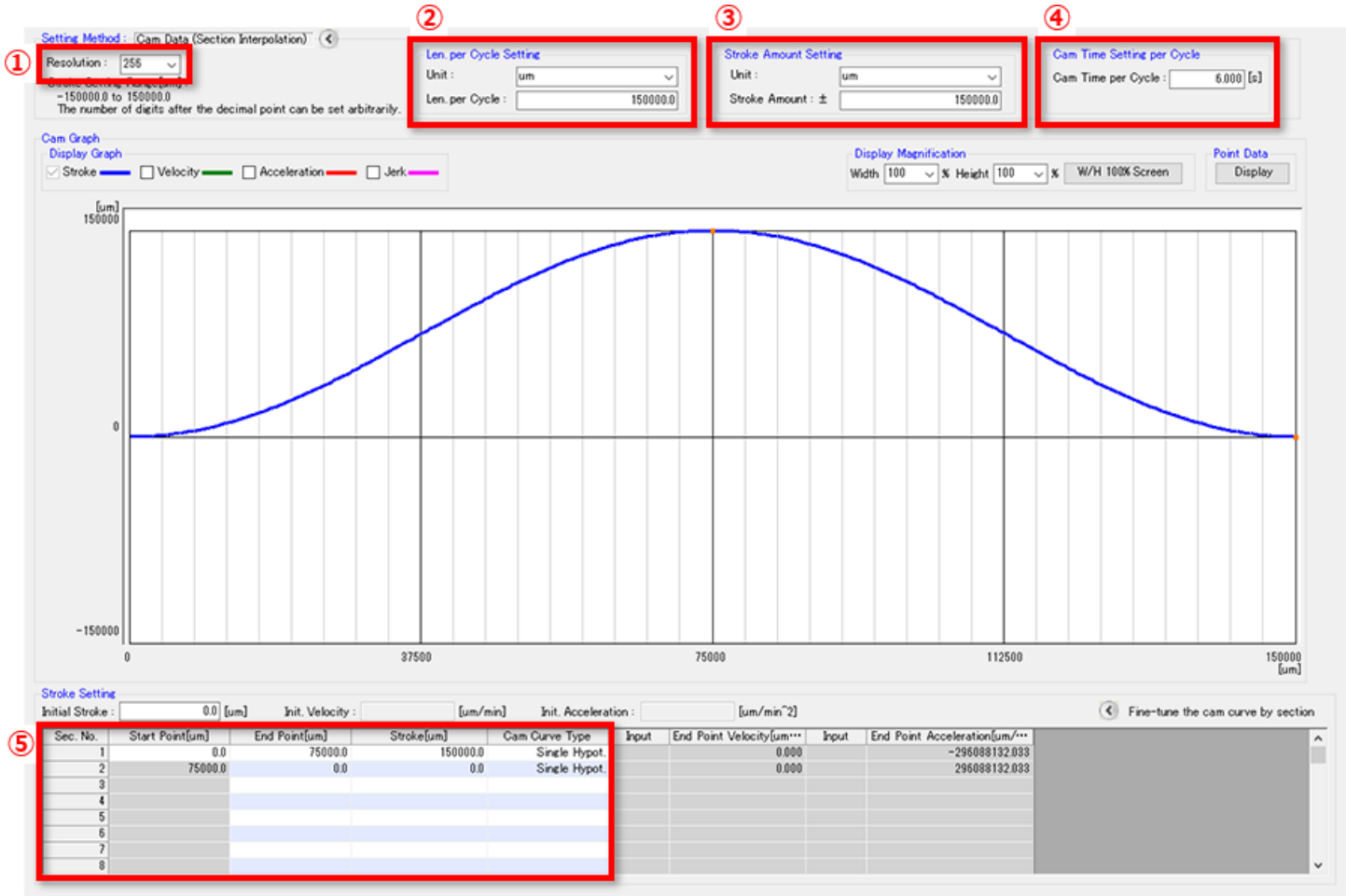
(1) Creating new operation profile data

With the motion control setting function, right-click [Operation Profile Data] in the navigation window and select [Add New Data] to display the "New Data" window.
In this course, we will use the default values for all the basic settings and detail settings.
Click the [OK] button.



(2) Creating cam data

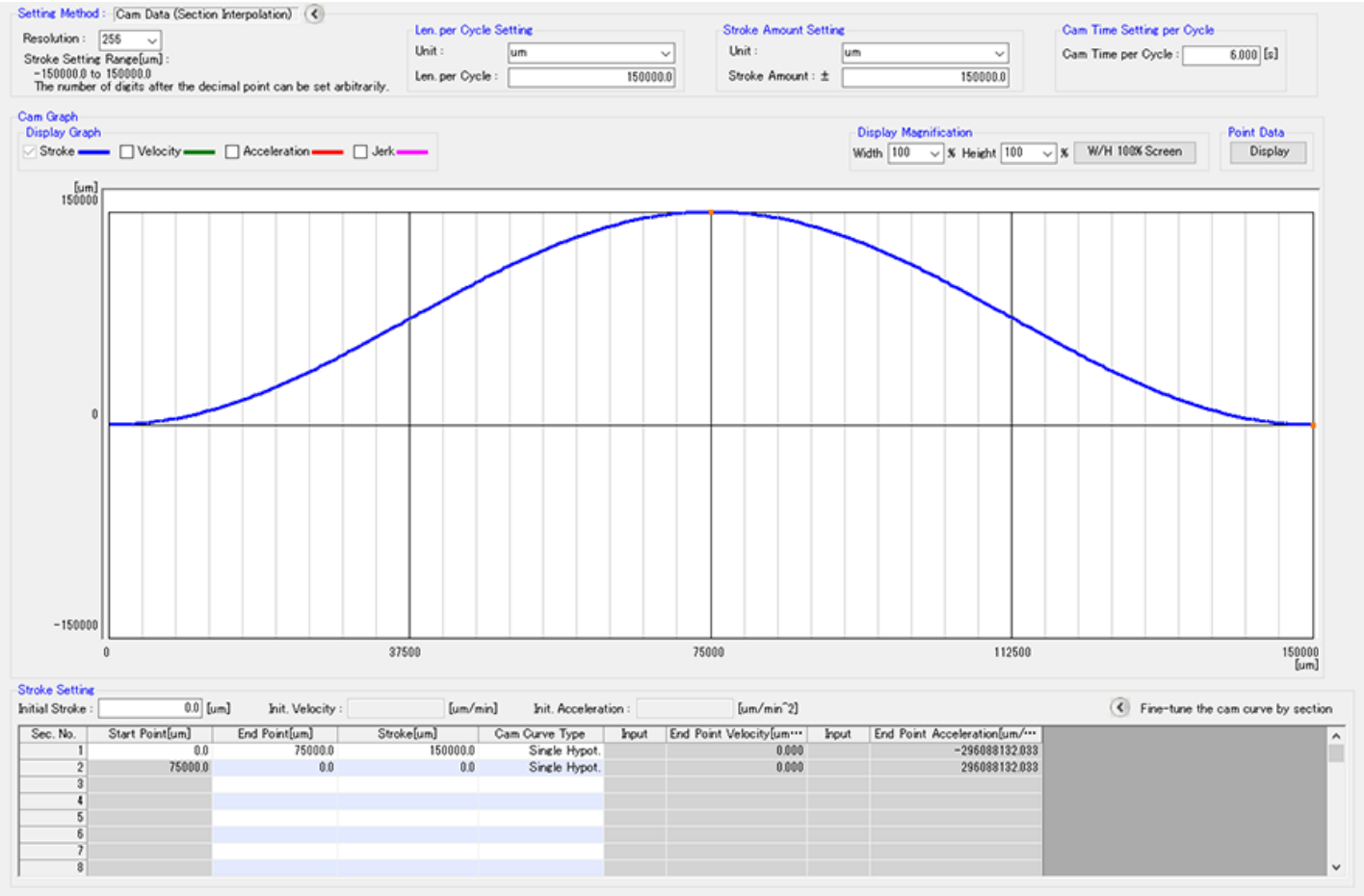
The window as shown below is displayed to create a cam curve.
The following table describes each item in the window.



No.	Item name	Description
1	Resolution	Select the resolution of the cam data.
2	Len. Per Cycle	Set the length and unit of a cycle. (Travel distance of the master axis required for a cam to rotate once)
3	Stroke Amount	Set the stroke amount and its unit. (Maximum travel distance of the slave axis during one rotation of a cam)
4	Cam Time per Cycle	Set the time required for one cycle of a cam. The time is used for calculating the acceleration, deceleration, and jerk values. (It does not affect the run time in the program.)
5	Stroke Setting	Set the stroke positions.

(2) Creating cam data (continued)

In this course, we will create the cam pattern as shown below.



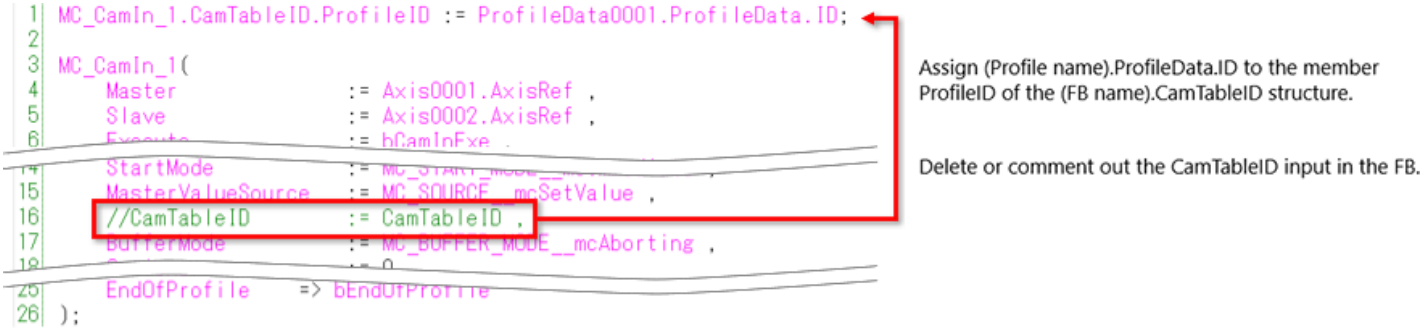
No.	Item name	Setting value
1	Resolution	256
2	Len. Per Cycle	150000.0 Unit: um (Enter manually.)
3	Stroke Amount	150000.0 Unit: um (Enter manually.)
4	Cam Time per Cycle	6[s]
5	Stroke Setting	See the right table.

Sec. No.	Start	End	Stroke	Cam Curve Type
1	0.0	75000.0	150000.0	Single Hypot.
2	75000.0	0.0	0.0	Single Hypot.

(3) How to specify the operation profile data

The cam data created in the operation profile must be stored in the cam storage area in the Motion module.
If "Auto Expand" is set to "Yes" when new operation profile data is created (→ 3.4 (1)),
the cam data is automatically stored in the cam storage area when [Y0] is turned ON.

At this time, set CamTableID specified in MC_CamIn as follows.



If "Auto Expand" is set to "No" when new operation profile data is created, MC_CamTableSelect,
which is an FB for loading cam data, must be executed. For details, refer to the following manual.

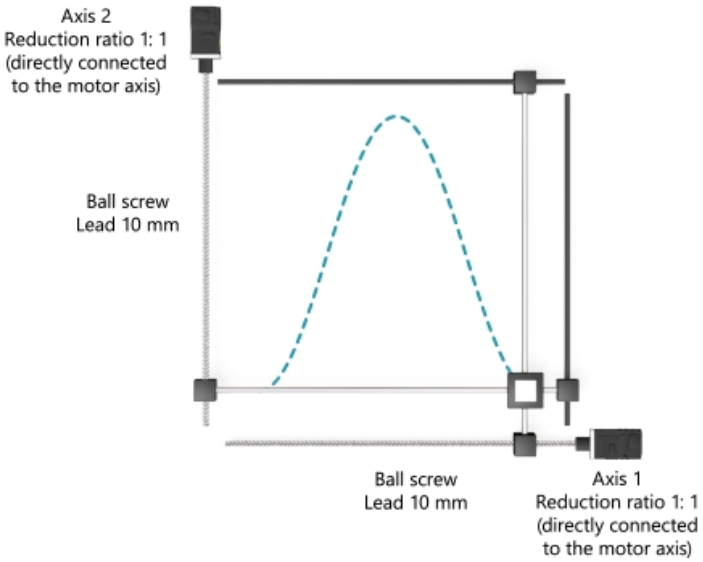
MELSEC iQ-R Programming Manual (Motion Control Function Blocks)
3.1 Management FBs
MC_CamTableSelect

3.5 Program Example

(1) Operation patterns

We will program the cam motion that draws one cycle and two cycles of the sine curve as shown in the following figures. Both patterns provide linear motion to return to the home position.

Motion of one cycle of the sine curve

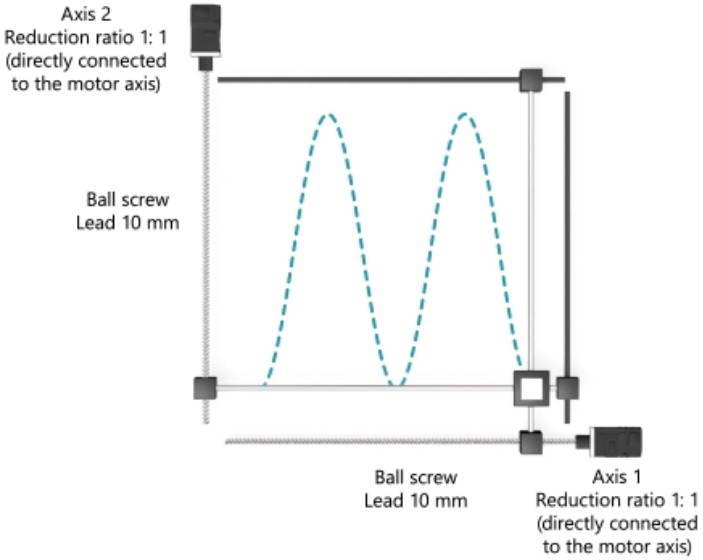


3.5 Program Example

(1) Operation patterns

We will program the cam motion that draws one cycle and two cycles of the sine curve as shown in the following figures. Both patterns provide linear motion to return to the home position.

Motion of two cycles of the sine curve



(2) Program description

For the program of the PLC CPU, refer to Chapter 2.

The following describes the program of the Motion module.

1) Program name [Synchronous1]

A program which draws one cycle of the sine curve.

```
1  //-----Synchronous #1-----//
2  //Initial Value Setting
3  IF(G_bSync1) THEN
4      //Positioning Velocity,Acceleration,Deceleration,Jerk
5      leVelocity       := 30000.0;
6      leAcceleration   := 60000.0;
7      leDeceleration   := 60000.0;
8      leJerk           := 120000.0;
9      //Request ON
10     G_bSync1Req := TRUE;
11 ELSE
12     //Request OFF
13     G_bSync1Req := FALSE;
14     //Reset Internal Signal
15     bGoingReq := FALSE;
16     bReturningReq := FALSE;
17 END_IF;
18
19 //Start Synchronous
20 MC_CamIn_1.CamTableID.ProfileID := ProfileData0001.ProfileData.ID;
21 MC_CamIn_1(
22     Master           := Axis0001.AxisRef ,
23     Slave            := Axis0002.AxisRef ,
24     Execute          := G_bSync1Req ,
25     InSync           => bInSync1,
26     CommandAborted   => bCamInAborted ,
27     Error            => bCamInError
28 );
29
30 //Going Request
31 SET(bInSync1,bGoingReq);
32
33 //Start Axis0001(Going)
34 MC_MoveAbsolute_1(
35     Axis             := Axis0001.AxisRef,
36     Execute          := bGoingReq ,
37     Position         := G_leMovePoint ,
38     Velocity         := leVelocity ,
39     Acceleration     := leAcceleration ,
40     Deceleration     := leDeceleration ,
41     Jerk             := leJerk ,
42     Done             => bMove1Done ,
43     CommandAborted   => bMove1Aborted ,
44     Error            => bMove1Error
45 );
46 //Dwell
47 TON_1(IN:= bMove1Done ,PT:= T#1s ,Q=> bMove1Dwell_out);
48
```

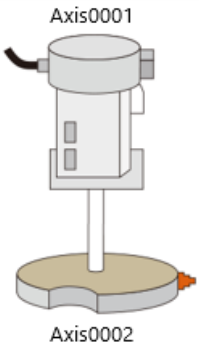
Synchronous control 1 start signal (from PLC CPU)

When the PLC CPU turns on the synchronous control 1 start signal (G_bSync1), the positioning speed and other commands are set and the start signal of the Motion module (G_bSync1Req) is turned on.

When the PLC CPU turns off the synchronous control 1 start signal (G_bSync1), the internal signal is turned off.

When the cam data is automatically loaded, the cam is specified in this way. (→ 3.4 (3))

MC_CamIn is executed to enable the synchronous control of Axis0002.
The relationship between the axes is shown in the figure on the right.



When MC_CamIn is executed and Axis0002 is under the synchronous control, Axis0001 is driven.
Axis0002 follows the specified cam pattern.

Dwell

(Continued to the next page)

(2) Program description

1) Program name [Synchronous1] (continued)

```
49  //Stop Synchronous
50  MC_Stop_1(
51      Axis    := Axis0002.AxisRef ,
52      Execute := bMove1Dwell_out OR bMove1Aborted OR bError,
53      Done    => bStop1Done
54  );
55
56  //Returuning Request
57  SET(bStop1Done & bMove1Dwell_out, bReturningReq);
58
59  //Start Axis0001(Returning)
60  MC_MoveAbsolute_2(
61      Axis            := Axis0001.AxisRef,
62      Execute         := bReturningReq ,
63      Position        := G_leHomePoint ,
64      Velocity        := leVelocity ,
65      Acceleration    := leAcceleration ,
66      Deceleration    := leDeceleration ,
67      Jerk            := leJerk ,
68      Done            => bMove2Done ,
69      CommandAborted  => bMove2Aborted ,
70      Error           => bMove2Error
71  );
72  //Dwell
73  TON_2(IN:= bMove2Done ,PT:= T#1s ,Q=> bMove2Dwell_out);
74
75  //Error Signal,Aborted Signal
76  bError  := bCamInError OR bMove1Error OR bMove2Error;
77  bAborted:= bMove1Aborted OR bMove2Aborted;
78
79  //Done Signal
80  G_bSync1Done:= bMove2Dwell_out OR bError OR bAborted;
81
```

The synchronous control of Axis0002 stops.

Axis0001 is returned to the home position by the single axis positioning control. Because the synchronous control is ended, Axis0002 will not move.

Dwell

The Error and Aborted outputs of the motion control FB are described in OR conditions. (Note)

When the operation is normally completed or the Error output or Aborted output turns on, the completion signal is turned on and the PLC CPU is notified about it.

(Note) The Aborted output of MC_CamIn is excluded from the ON conditions of G_bSync1Done which indicates the completion of the operation because the output is turned on when MC_STOP is executed.

(2) Program description

2) Program name [Synchronous2]

A program which draws two cycle of the sine curve using a virtual drive axis and composite gear.

```
1  //-----Synchronous #2-----//
2  //Initial Value Setting
3  IF G_bSync2 THEN
4      //Positioning Velocity,Acceleration,Deceleration,Jerk
5      leVelocity       := 30000.0;
6      leAcceleration   := 60000.0;
7      leDeceleration   := 60000.0;
8      leJerk           := 120000.0;
9      //Request ON
10     G_bSync2Req := TRUE;
11 ELSE
12     //Request OFF
13     G_bSync2Req := FALSE;
14     //Reset Internal Signal
15     bGoingReq := FALSE;
16     bReturningReq := FALSE;
17 END_IF;
18
19 //Start Synchronous
20 MC_CombineAxes_1(
21     Master1          := Axis0001.AxisRef ,
22     Master2          := VirtualAxis0001.AxisRef ,
23     Slave            := LinkAxis0001.AxisRef ,
24     Execute          := G_bSync2Req ,
25     CombineMode      := MC_COMBINE_MODE__mcAddAxes ,
26     GearRatioNumeratorM1    := 1 ,
27     GearRatioDenominatorM1  := 1 ,
28     GearRatioNumeratorM2    := 1 ,
29     GearRatioDenominatorM2  := 1 ,
30     InSync           => bInSync1 ,
31     CommandAborted   => bCombineAxes1Aborted ,
32     Error            => bCombineAxes1Error
33 );
34
35 MC_CamIn_1.CamTableID.ProfileID := ProfileData0001.ProfileData.ID;
36 MC_CamIn_1(
37     Master           := LinkAxis0001.AxisRef ,
38     Slave            := Axis0002.AxisRef ,
39     Execute          := G_bSync2Req ,
40     InSync           => bInSync2,
41     CommandAborted   => bCamInAborted ,
42     Error            => bCamInError
43 );
44
```

Synchronous control 2 start signal (from PLC CPU)

When the PLC CPU turns on the synchronous control 2 start signal (G_bSync2), the positioning speed and other commands are set and the start signal of the Motion module (G_bSync2Req) is turned on.

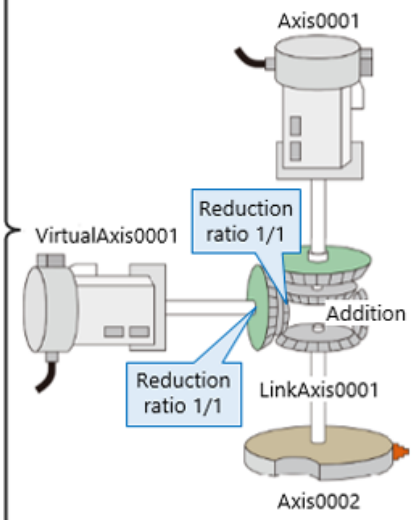
When the PLC CPU turns off the synchronous control 2 start signal (G_bSync2), the internal signal is turned off.

MC_ConbineAxes is executed to enable the synchronous control of the virtual linked axis LinkAxis0001.

Addition
Master1 reduction ratio 1/1
Master2 reduction ratio 1/1

MC_CamIn is executed to enable the synchronous control of the real drive axis Axis0002.

The relationship between the axes is shown in the figure below.



(Continued to the next page)

(2) Program description

2) Program name [Synchronous2] (continued)

```
45  //Set Going Request
46  SET(bInSync1 AND bInSync2,bGoingReq);
47
48  //Start Axis0001(Going)
49  MC_MoveAbsolute_1(
50      Axis            := Axis0001.AxisRef ,
51      Execute         := bGoingReq ,
52      Position        := G_leMovePoint ,
53      Velocity        := leVelocity ,
54      Acceleration    := leAcceleration ,
55      Deceleration    := leDeceleration ,
56      Jerk            := leJerk ,
57      Done            => bMove1Done ,
58      CommandAborted  => bMove1Aborted ,
59      Error           => bMove1Error
60  );
61
62  //Start VirtualAxis0001
63  MC_MoveRelative_1(
64      Axis            := VirtualAxis0001.AxisRef ,
65      Execute         := bGoingReq ,
66      Distance        := G_leMovePoint ,
67      Velocity        := leVelocity ,
68      Acceleration    := leAcceleration,
69      Deceleration    := leDeceleration ,
70      Jerk            := leJerk ,
71      Done            => bMove2Done ,
72      CommandAborted  => bMove2Aborted ,
73      Error           => bMove2Error
74  );
75
76  //Dwell
77  TON_1(IN:= bMove1Done & bMove2Done ,PT:= T#1s ,Q=> bDwell1_out);
78
79  //Stop Synchronous
80  MC_Stop_1(
81      Axis    := LinkAxis0001.AxisRef ,
82      Execute := bDwell1_out OR bMove1Aborted OR bMove2Aborted OR bError,
83      Done    => bStop1Done
84  );
85
86  MC_Stop_2(
87      Axis    := Axis0002.AxisRef ,
88      Execute := bDwell1_out OR bMove1Aborted OR bMove2Aborted OR bError,
89      Done    => bStop2Done
90  );
91
```

The real drive axis and virtual drive axis are started at the same time.
Both the axes move 150000.0 µm. LinkAxis0001, which is the output of MC_CombineAxes, moves 300000.0 µm.

Because LinkAxis0001, which is the input axis of MC_CamIn, moves 300000.0 µm while Axis0001 moves 150000.0 µm, the cam rotate twice, drawing two cycles of the sine curve.

Dwell

The synchronous control with the virtual linked axis LinkAxis0001 and real drive axis Axis0002 is deactivated.

(Continued to the next page)

(2) Program description

2) Program name [Synchronous2] (continued)

```
92  //Returning Request
93  SET(bStop1Done & bStop2Done & bDwell1_out, bReturningReq);
94
95  //Start Axis0001(Returning)
96  MC_MoveAbsolute_2(
97      Axis            := Axis0001.AxisRef ,
98      Execute         := bReturningReq ,
99      Position        := G_leHomePoint ,
100     Velocity        := leVelocity ,
101     Acceleration    := leAcceleration ,
102     Deceleration    := leDeceleration ,
103     Jerk            := leJerk ,
104     Done            => bMove3Done ,
105     CommandAborted  => bMove3Aborted ,
106     Error           => bMove3Error
107 );
108 //Dwell
109 TON_2(IN:= bMove3Done ,PT:= T#1s ,Q=> bDwell2_out);
110
111 //Error Signal,Aborted Signal
112 bError  := bCombineAxes1Error OR bCamInError
113            OR bMove1Error OR bMove2Error OR bMove3Error;
114 bAborted:= bMove1Aborted OR bMove2Aborted OR bMove3Aborted;
115
116 //Done Signal
117 G_bSync2Done:= bDwell2_out OR bError OR bAborted;
118
```

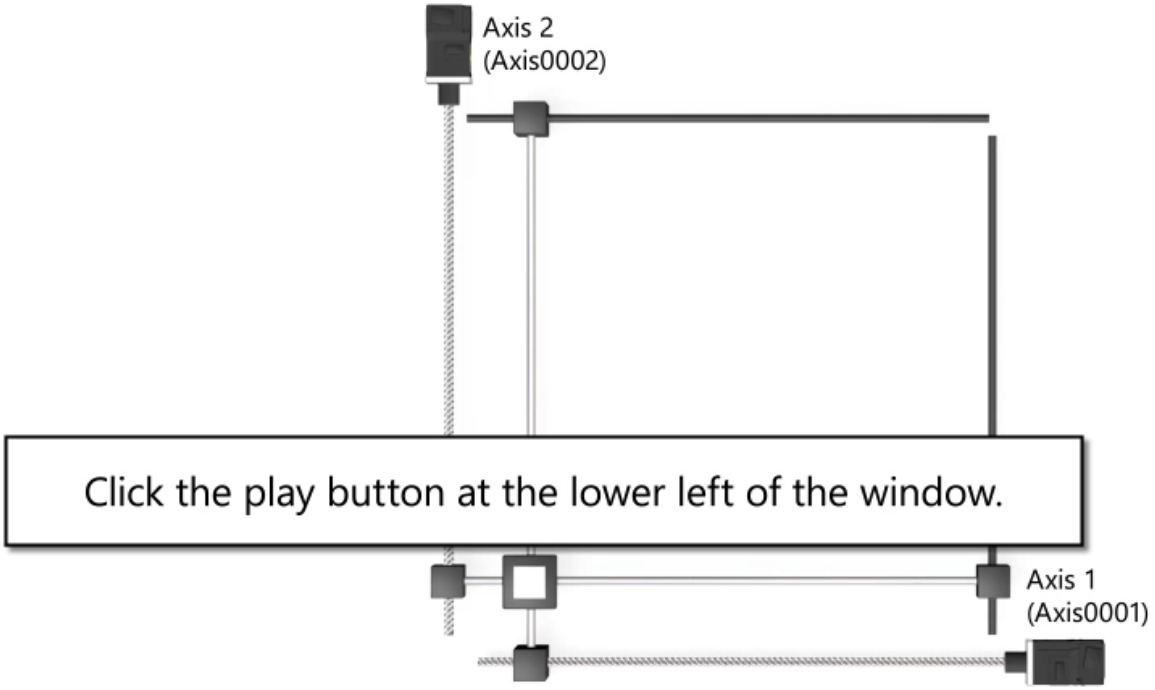
Axis0001 is returned to the home position by the single axis positioning control. Because the synchronous control is ended, LinkAxis0001 and Axis0002 will not move.

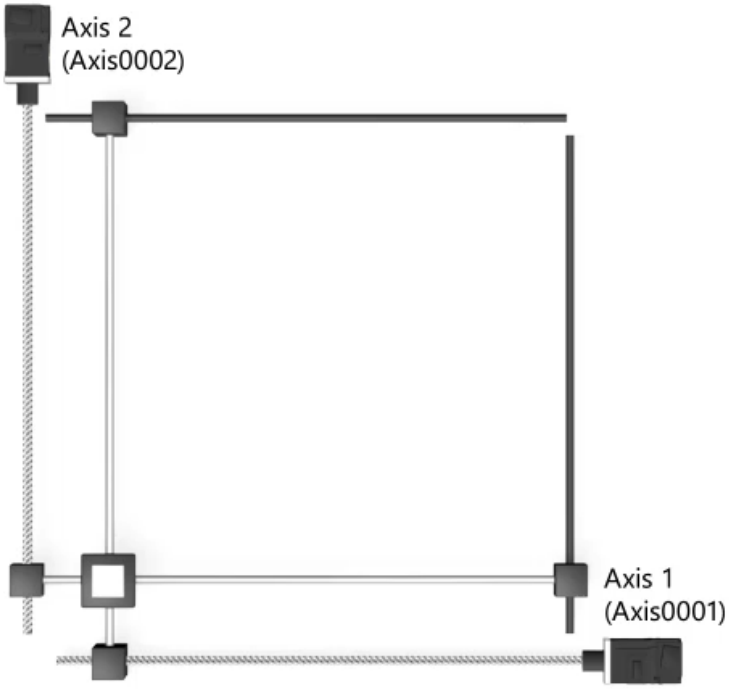
Dwell

The Error and Aborted outputs of the motion control FB are described in OR conditions. (Note)

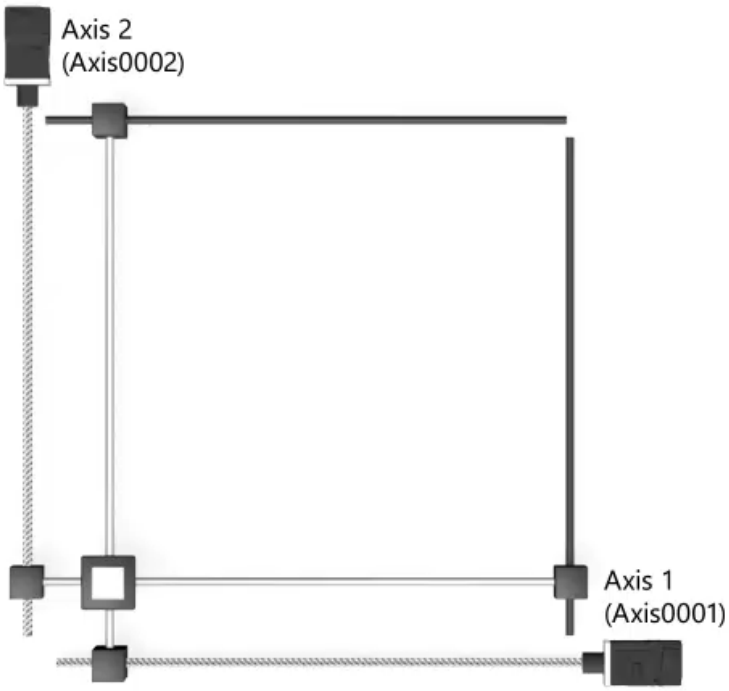
When the operation is normally completed or the Error output or Aborted output turns on, the completion signal is turned on and the PLC CPU is notified about it.

(Note) The Aborted outputs of MC_CombineAxes and MC_CamIn are excluded from the ON conditions of G_bSync2Done which indicates the completion of the operation because these outputs are turned on when MC_STOP is executed.

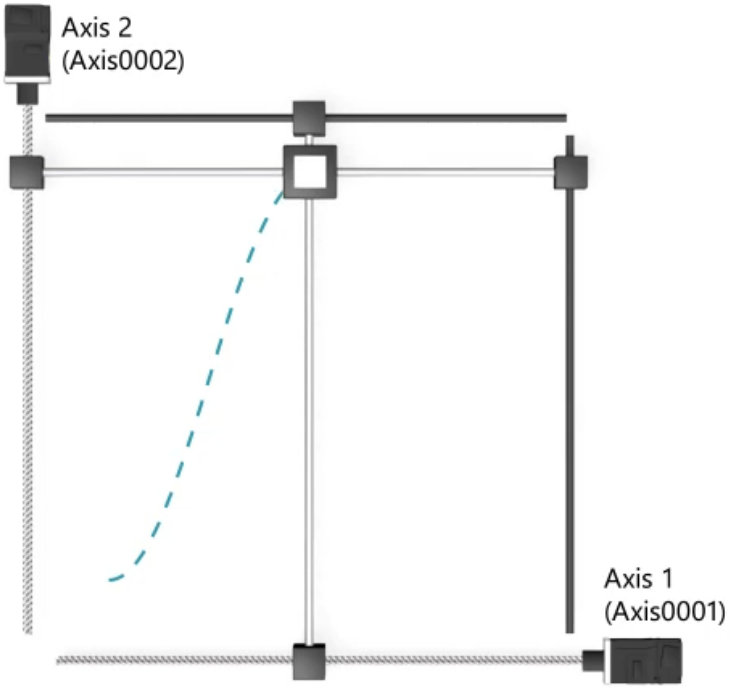

Axis 2
(Axis0002)
Click the play button at the lower left of the window.
Axis 1
(Axis0001)



Check the sample program operation.
Install the programs of the PLC CPU and Motion module and set the RUN/STOP/RESET switch of the PLC CPU to RUN.

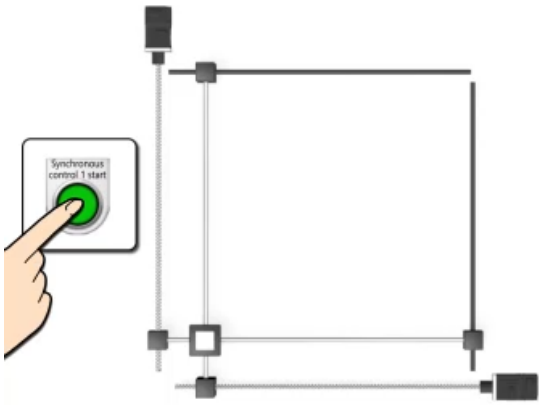


The JOG operation and homing are described in Chapter 2.
Before starting operation, make sure that Axis0001, Axis0002, and VirtualAxis0001 have returned to the home position.



Turn on the synchronous control 1 start (X29).
The X-Y table draws one cycle of a sine curve.
The axes return to the home position in linear motion.

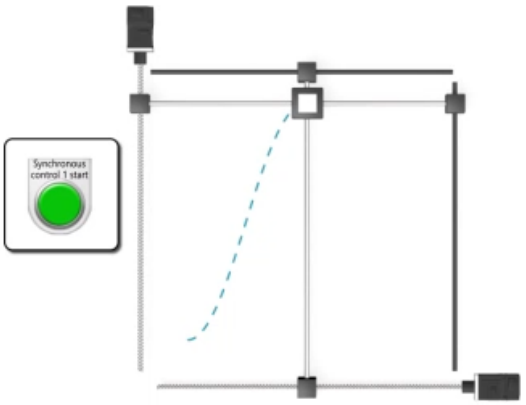
Axis status
Axis0001: Standstill
Axis0002: SynchronizedMotion




```
    leVelocity      := 30000.0;
    leAcceleration  := 60000.0;
    leDeceleration  := 60000.0;
    leJerk          := 120000.0;
    //Request ON
    G_bSync1Req := TRUE;
ELSE
    //Request OFF
    G_bSync1Req := FALSE;
    //Reset Internal Signal
    bGoingReq := FALSE;
    bReturningReq := FALSE;
END_IF;

//Start Synchronous
MC_CamIn_1.CamTableID.ProfileID := ProfileData0001.ProfileData.ID;
MC_CamIn_1(
    Master          := Axis0001.AxisRef ,
    Slave           := Axis0002.AxisRef ,
    Execute         := G_bSync1Req ,
    InSync          => bInSync1,
    CommandAborted  => bCamInAborted ,
    Error           => bCamInError
);

//Going Request
SET(bInSync1,bGoingReq);

//Start Axis0001(Going)
MC_MoveAbsolute_1(
    Axis            := Axis0001.AxisRef,
```

Check the program monitor and axis status.
When MC_CamIn is executed, the axis status of Axis0002 becomes "SynchronizedMotion".

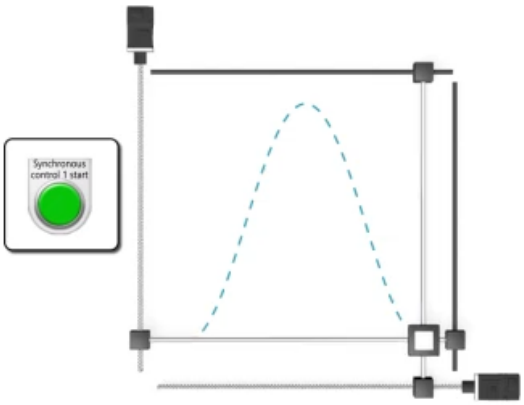
Axis status
Axis0001: DiscreteMotion
Axis0002: SynchronizedMotion




```
    InSync          => bInSync1,
    CommandAborted  => bCamInAborted,
    Error           => bCamInError
);

//Going Request
SET(bInSync1,bGoingReq);

//Start Axis0001(Going)
MC_MoveAbsolute_1(
    Axis            := Axis0001.AxisRef,
    Execute         := bGoingReq,
    Position        := G_leMovePoint,
    Velocity        := leVelocity,
    Acceleration    := leAcceleration,
    Deceleration    := leDeceleration,
    Jerk            := leJerk,
    Done            => bMove1Done,
    CommandAborted  => bMove1Aborted,
    Error           => bMove1Error
);
//Dwell
TON_1(IN:=bMove1Done,PT:=T#1s,Q=>bMove1Dwell_out);

//Stop Synchronous
MC_Stop_1(
    Axis    := Axis0002.AxisRef,
    Execute := bMove1Dwell_out OR bMove1Aborted OR bError,
    Done    => bStop1Done
);
```

In this state, the input axis Axis0001 is positioned
by the single axis positioning control.
Then, Axis0002 follows the specified cam pattern.

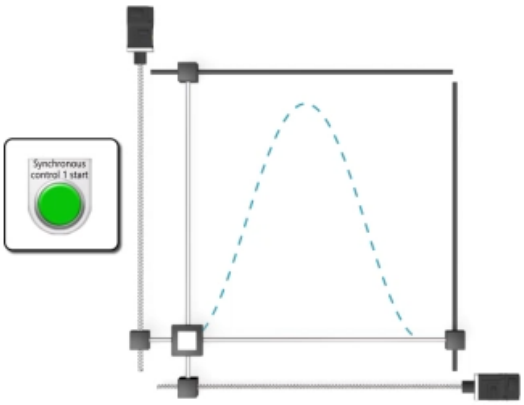
Axis status
Axis0001: DiscreteMotion
Axis0002: SynchronizedMotion




```
    InSync              => bInSync1,
    CommandAborted      => bCamInAborted,
    Error               => bCamInError
);

//Going Request
SET(bInSync1,bGoingReq);

//Start Axis0001(Going)
MC_MoveAbsolute_1(
    Axis                := Axis0001.AxisRef,
    Execute             := bGoingReq,
    Position            := G_leMovePoint,
    Velocity            := leVelocity,
    Acceleration        := leAcceleration,
    Deceleration        := leDeceleration,
    Jerk                := leJerk,
    Done                => bMove1Done,
    CommandAborted      => bMove1Aborted,
    Error               => bMove1Error
);
//Dwell
TON_1(IN:= bMove1Done ,PT:= T#1s ,Q=> bMove1Dwell_out);

//Stop Synchronous
MC_Stop_1(
    Axis    := Axis0002.AxisRef,
    Execute := bMove1Dwell_out OR bMove1Aborted OR bError,
    Done    => bStop1Done
);
```

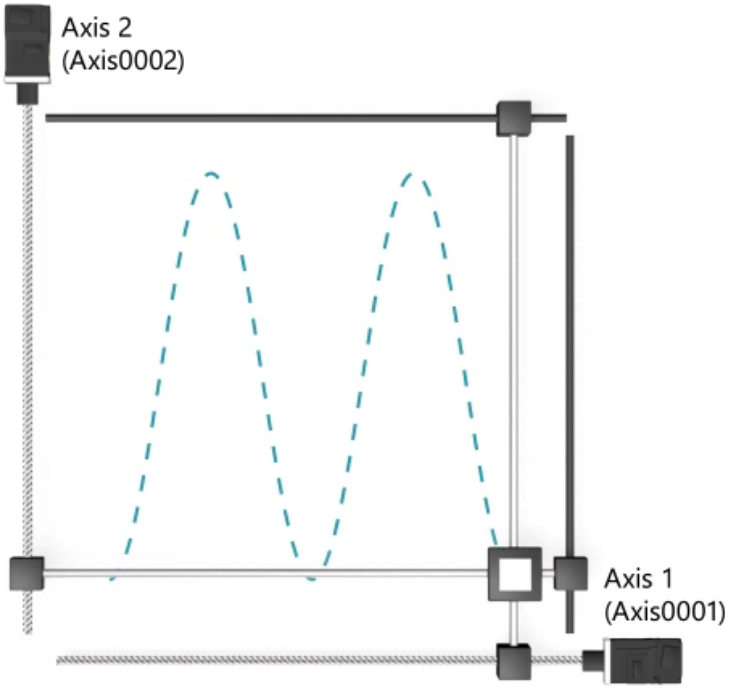
When the axis reaches the target position is reached and the dwell time has elapsed, the synchronous control of Axis0002 is deactivated.
After the synchronous control is deactivated,
Axis0001 is returned to the home position by the single axis positioning control.
Because the synchronization is deactivated, Axis0002 will not move.

Axis status
Axis0001: Standstill
Axis0002: Standstill




```
    Done     => bStop1Done
);

//Returuning Request
SET(bStop1Done & bMove1Dwell_out, bReturningReq);

//Start Axis0001(Returning)
MC_MoveAbsolute_2(
    Axis            := Axis0001.AxisRef,
    Execute         := bReturningReq,
    Position        := G_leHomePoint,
    Velocity        := leVelocity,
    Acceleration    := leAcceleration,
    Deceleration    := leDeceleration,
    Jerk            := leJerk,
    Done            => bMove2Done,
    CommandAborted  => bMove2Aborted,
    Error           => bMove2Error
);
//Dwell
TON_2(IN:= bMove2Done ,PT:= T#1s ,Q=> bMove2Dwell_out);

//Error Signal,Aborted Signal
bError   := bCamInError OR bMove1Error OR bMove2Error;
bAborted:= bMove1Aborted OR bMove2Aborted;

//Done Signal
G_bSync1Done:= bMove2Dwell_out OR bError OR bAborted;
```

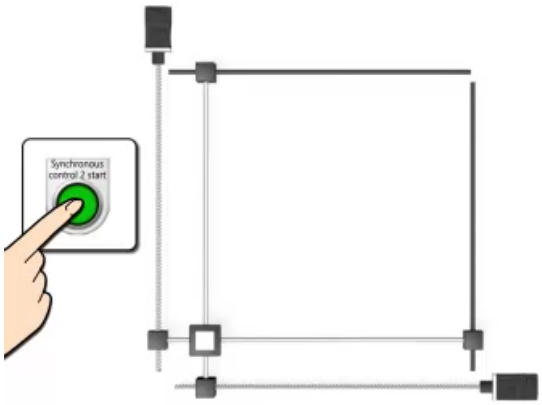
When the axis reaches the home position and the dwell time has elapsed, the completion signal turns on.
This completes the synchronous control 1 operation.



Next, turn on the synchronous control 2 start (X2A).
The X-Y table draws two cycles of the sine curve.
The axes return to the home position in linear motion.

Axis status

Axis0001	: Standstill
VirtualAxis0001	: Standstill
LinkAxis0001	: SynchronizedMotion
Axis0002	: SynchronizedMotion




```
    //Reset Internal Signal
    bGoingReq := FALSE;
    bReturningReq := FALSE;
END_IF;

//Start Synchronous
MC_CombineAxes_1(
    Master1          := Axis0001.AxisRef ,
    Master2          := VirtualAxis0001.AxisRef ,
    Slave            := LinkAxis0001.AxisRef ,
    Execute          := G_bSync2Req ,
    CombineMode      := MC_COMBINE_MODE__mcAddAxes ,
    GearRatioNumeratorM1    := 1 ,
    GearRatioDenominatorM1  := 1 ,
    GearRatioNumeratorM2    := 1 ,
    GearRatioDenominatorM2  := 1 ,
    InSync           => bInSync1 ,
    CommandAborted   => bCombineAxes1Aborted ,
    Error            => bCombineAxes1Error
);

MC_CamIn_1.CamTableID.ProfileID := ProfileData0001_ProfileData_ID;
MC_CamIn_1(
    Master           := LinkAxis0001.AxisRef
    Slave            := Axis0002.AxisRef ,
    Execute          := G_bSync2Req ,
    InSync           => bInSync2 ,
    CommandAborted   => bCamInAborted ,
    Error            => bCamInError
);
```

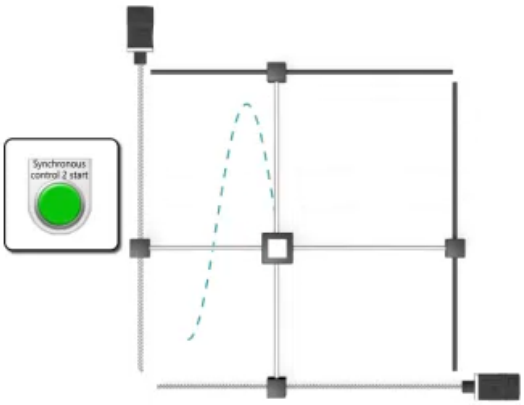
Set position [mm]

Axis0001	:	0.0
VirtualAxis0001	:	0.0
LinkAxis0001	:	0.0
Axis0002	:	0.0

Check the program monitor and axis status and set position of each axis. When MC_CombineAxes and MC_CamIn are executed, the axis statuses of LinkAxis0001 and Axis0002 become "SynchronizedMotion".

Axis status

Axis0001	: DiscreteMotion
VirtualAxis0001	: DiscreteMotion
LinkAxis0001	: SynchronizedMotion
Axis0002	: SynchronizedMotion




```
SET(                AND                );

//Start Axis0001(Going)
MC_MoveAbsolute_1(
    Axis            := Axis0001.AxisRef ,
    Execute         := bGoingReq ,
    Position        := G_leMovePoint ,
    Velocity        := leVelocity ,
    Acceleration    := leAcceleration ,
    Deceleration    := leDeceleration ,
    Jerk            := leJerk ,
    Done            => bMove1Done ,
    CommandAborted  => bMove1Aborted ,
    Error           => bMove1Error
);

//Start VirtualAxis0001
MC_MoveRelative_1(
    Axis            := VirtualAxis0001.AxisRef ,
    Execute         := bGoingReq ,
    Distance        := G_leMovePoint ,
    Velocity        := leVelocity ,
    Acceleration    := leAcceleration,
    Deceleration    := leDeceleration ,
    Jerk            := leJerk ,
    Done            => bMove2Done ,
    CommandAborted  => bMove2Aborted ,
    Error           => bMove2Error
);

//Dwell
```

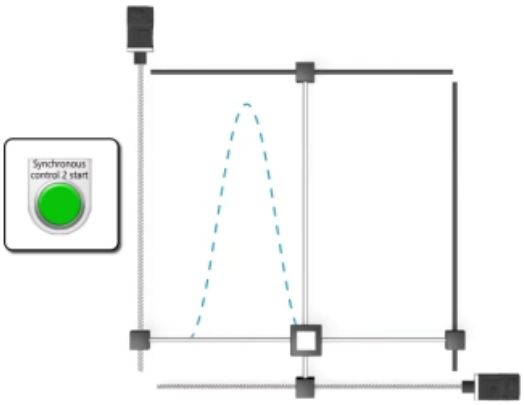
Set position [mm]

Axis0001	:	51.5
VirtualAxis0001	:	51.5
LinkAxis0001	:	102.9
Axis0002	:	63.8

Once the synchronous control starts,
Axis0001 and VirtualAxis0001 will start at the same time.
The set position of LinkAxis0001 is the sum of the values
of the set positions of Axis0001 and VirtualAxis0001.

Axis status

Axis0001	: DiscreteMotion
VirtualAxis0001	: DiscreteMotion
LinkAxis0001	: SynchronizedMotion
Axis0002	: SynchronizedMotion




```
46  SET(bInSync1 AND bInSync2.bGoingRef);
47
48  //Start Axis0001(Going)
49  MC_MoveAbsolute_1(
50      Axis            := Axis0001.AxisRef ,
51      Execute         := bGoingReq ,
52      Position        := G_leMovePoint ,
53      Velocity        := leVelocity ,
54      Acceleration    := leAcceleration ,
55      Deceleration    := leDeceleration ,
56      Jerk            := leJerk ,
57      Done            => bMove1Done ,
58      CommandAborted  => bMove1Aborted ,
59      Error           => bMove1Error
60  );
61
62  //Start VirtualAxis0001
63  MC_MoveRelative_1(
64      Axis            := VirtualAxis0001.AxisRef ,
65      Execute         := bGoingReq ,
66      Distance        := G_leMovePoint ,
67      Velocity        := leVelocity ,
68      Acceleration    := leAcceleration,
69      Deceleration    := leDeceleration ,
70      Jerk            := leJerk ,
71      Done            => bMove2Done ,
72      CommandAborted  => bMove2Aborted ,
73      Error           => bMove2Error
74  );
75
76  //Dwell
```

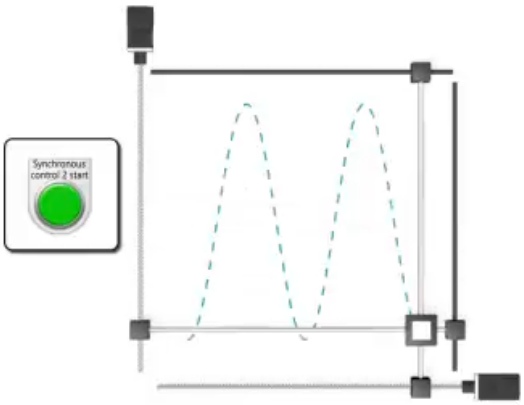
Set position [mm]

Axis0001	:	75.0
VirtualAxis0001	:	75.0
LinkAxis0001	:	150.0
Axis0002	:	0.0

When the set position of Axis0001 reaches 75 mm,
the set position of LinkAxis0001 reaches 150 mm.
Therefore, at this point, Axis0002 has moved around the cam once.

Axis status

Axis0001	: DiscreteMotion
VirtualAxis0001	: DiscreteMotion
LinkAxis0001	: SynchronizedMotion
Axis0002	: SynchronizedMotion




```
    Jerk            := leJerk ,
    Done            => bMove2Done ,
    CommandAborted  => bMove2Aborted ,
    Error           => bMove2Error
);

//Dwell
TON_1( IN:= bMove1Done & bMove2Done ,PT:= T#1s ,Q=> bDwell1_out);

//Stop Synchronous
MC_Stop_1(
    Axis    := LinkAxis0001.AxisRef ,
    Execute := bDwell1_out OR bMove1Aborted OR bMove2Aborted OR bError,
    Done    => bStop1Done
);

MC_Stop_2(
    Axis    := Axis0002.AxisRef ,
    Execute := bDwell1_out OR bMove1Aborted OR bMove2Aborted OR bError,
    Done    => bStop2Done
);

//Returning Request
SET(bStop1Done & bStop2Done & bDwell1_out,

//Start Axis0001(Returning)
MC_MoveAbsolute_2(
    Axis            := Axis0001.AxisRef ,
    Execute         := bReturningReq ,
    Position        := G_leHomePoint ,
    Velocity        := leVelocity ,
```

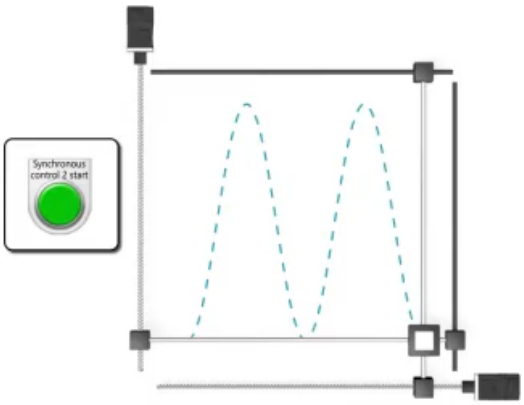
Set position [mm]

Axis0001	: 149.6
VirtualAxis0001	: 149.6
LinkAxis0001	: 299.1
Axis0002	: 18.8

By the time the set position of Axis0001 reaches 150 mm,
Axis0002 will have moved around the cam twice.

Axis status

Axis0001	: Standstill
VirtualAxis0001	: Standstill
LinkAxis0001	: SynchronizedMotion
Axis0002	: SynchronizedMotion




```
    Jerk            := leJerk ,
    Done            => bMove2Done ,
    CommandAborted  => bMove2Aborted ,
    Error           => bMove2Error
);

//Dwell
TON_1(IN:= bMove1Done & bMove2Done ,PT:= T#1s ,Q=> bDwell1_out);

//Stop Synchronous
MC_Stop_1(
    Axis    := LinkAxis0001.AxisRef ,
    Execute := bDwell1_out OR bMove1Aborted OR bMove2Aborted OR bError,
    Done    => bStop1Done
);

MC_Stop_2(
    Axis    := Axis0002.AxisRef ,
    Execute := bDwell1_out OR bMove1Aborted OR bMove2Aborted OR bError,
    Done    => bStop2Done
);

//Returning Request
SET(bStop1Done & bStop2Done & bDwell1_out,

//Start Axis0001(Returning)
MC_MoveAbsolute_2(
    Axis            := Axis0001.AxisRef ,
    Execute         := bReturningReq ,
    Position        := G_leHomePoint ,
    Velocity        := leVelocity ,
```

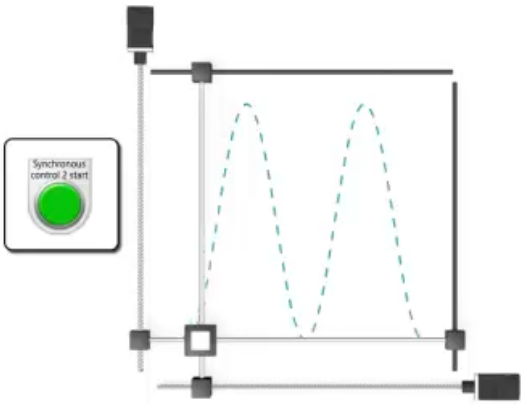
Set position [mm]

Axis0001	:	150.0
VirtualAxis0001	:	150.0
LinkAxis0001	:	300.0
Axis0002	:	0.0

When Axis0001 reaches 150 mm and the dwell time has elapsed,
the synchronous control of LinkAxis0001
and Axis0002 is deactivated.

Axis status

Axis0001	: DiscreteMotion
VirtualAxis0001	: Standstill
LinkAxis0001	: Stopping
Axis0002	: Stopping




```
94
95  //Start Axis0001(Returning)
96  MC_MoveAbsolute_2(
97      Axis            := Axis0001.AxisRef ,
98      Execute         := bReturningRen ,
99      Position        := G_leHomePoint ,
100     Velocity        := leVelocity ,
101     Acceleration    := leAcceleration ,
102     Deceleration    := leDeceleration ,
103     Jerk            := leJerk ,
104     Done            => bMove3Done ,
105     CommandAborted  => bMove3Aborted ,
106     Error           => bMove3Error
107 );
108 //Dwell
109 TON_2(IN:= bMove3Done ,PT:= T#1s ,Q=> bDwell2_out);
110
111 //Error Signal,Aborted Signal
112 bError   := bCombineAxes1Error OR bCamInError
113         OR bMove1Error OR bMove2Error OR bMove3Error;
114 bAborted:= bMove1Aborted OR bMove2Aborted OR bMove3Aborted;
115
116 //Done Signal
117 G_bSync2Done:= bDwell2_out OR bError OR bAb
118
```

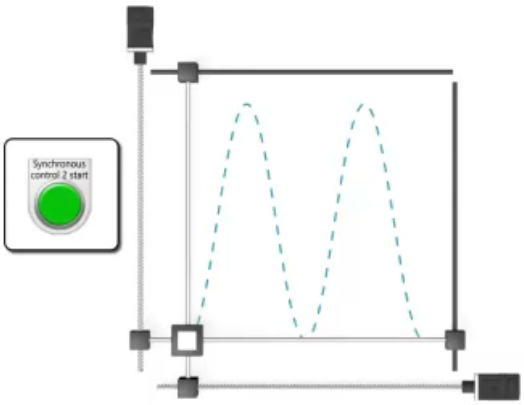
Set position [mm]

Axis0001	:	13.7
VirtualAxis0001	:	150.0
LinkAxis0001	:	300.0
Axis0002	:	0.0

After the synchronous control of LinkAxis0001 and Axis0002 is deactivated, Axis0001 is returned to the home position by the single axis positioning control. Because the synchronous control is deactivated, LinkAxis0001 and Axis0002 will not move.

Axis status

Axis0001	: DiscreteMotion
VirtualAxis0001	: Standstill
LinkAxis0001	: Stopping
Axis0002	: Stopping




```
    Axis            := Axis0001.AxisRef ,
    Execute         := bReturningReq ,
    Position        := G_leHomePoint ,
    Velocity        := leVelocity ,
    Acceleration    := leAcceleration ,
    Deceleration    := leDeceleration ,
    Jerk            := leJerk ,
    Done            => bMove3Done ,
    CommandAborted  => bMove3Aborted ,
    Error           => bMove3Error
);
//Dwell
TON_2(IN:= bMove3Done ,PT:= T#1s ,Q=> bDwell2_out);

//Error Signal,Aborted Signal
bError   := bCombineAxes1Error OR bCamInError
            OR bMove1Error OR bMove2Error OR bMove3Error;
bAborted:= bMove1Aborted OR bMove2Aborted OR bMove3Aborted;

//Done Signal
G_bSync2Done:= bDwell2_out OR bError OR bAborted;
```

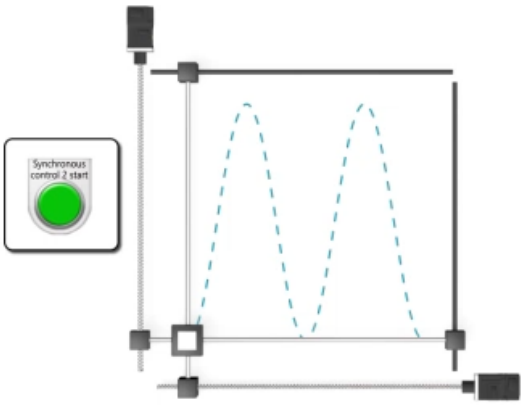
Set position [mm]

Axis0001	:	0.0
VirtualAxis0001	:	150.0
LinkAxis0001	:	300.0
Axis0002	:	0.0

When the axis reaches the home position and
the dwell time has elapsed, the completion signal turns on.
This completes the synchronous control 2 operation.

Axis status

Axis0001	: Standstill
VirtualAxis0001	: Standstill
LinkAxis0001	: Standstill
Axis0002	: Standstill




```
    Axis                := Axis0001.AxisRef ,
    Execute             := bReturningReq ,
    Position            := G_leHomePoint ,
    Velocity            := leVelocity ,
    Acceleration        := leAcceleration ,
    Deceleration        := leDeceleration ,
    Jerk                := leJerk ,
    Done                => bMove3Done ,
    CommandAborted      => bMove3Aborted ,
    Error               => bMove3Error
);
//Dwell
TON_2(IN:= bMove3Done ,PT:= T#1s ,Q=> bDwell2_out);

//Error Signal,Aborted Signal
bError   := bCombineAxes1Error OR bCamInError
            OR bMove1Error OR bMove2Error OR bMove3Error;
bAborted:= bMove1Aborted OR bMove2Aborted OR bMove3Aborted;

//Done Signal
G_bSync2Done:= bDwell2_out OR bError OR bAborted;
```

Set position [mm]

Axis0001	:	0.0
VirtualAxis0001	:	150.0
LinkAxis0001	:	300.0
Axis0002	:	0.0

This completes the operation check of the synchronous control.
Go to the next page.

3.7 Summary of This Chapter

In this chapter, you have learned:

- Concept of Synchronous Control
- Virtual Axis
- Steps of Synchronous Control
- Operation Profile
- Program Example
- Operation Check

Important points

Concept of Synchronous Control	• Synchronous control is a software function that controls mechanical components, such as gears, transmissions, and cams. • FBs are used to describe the relationship between the master axis and slave axis of the gears, composite gears, and cams.
Virtual Axis	• A virtual axis is used generate virtual commands and position data. • There are three types of virtual axes: Virtual drive axis, virtual encoder axis, and virtual linked axis.
Steps of Synchronous Control	• Start the FBs for synchronous control, and set the axis status of the slave to "SynchronizedMotion". When the master axis is moved in this state, the slave axis moves synchronously. • To end the synchronous control, execute MC_Stop.
Operation Profile	• Cam patterns can be created as operation profile data. • Enter the Resolution, Len. Per Cycle, Stroke Amount, and Stroke Setting to create a cam pattern.
Program Example	• You have learned about the program to draw the path of the cam pattern. • You have learned about the program example using a virtual drive axis and composite gear.
Operation Check	• You have checked the operation of the sample program in the video.

Test Final Test

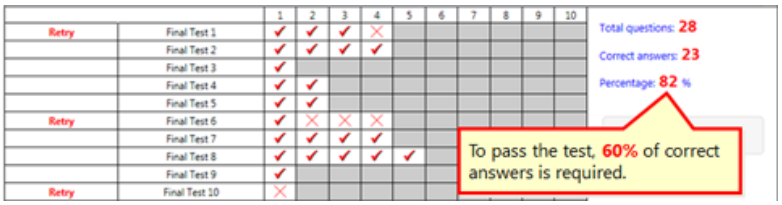
Now that you have completed all of the lessons of the **MELSEC iQ-R Series Motion Module Application (RD78G(H) Interpolation Control/Synchronous Control)** Course, you are ready to take the final test. If you are unclear on any of the topics covered, please take this opportunity to review those topics.

There are a total of 4 questions (22 items) in this Final Test.

You can take the final test as many times as you like.

Score results

The number of correct answers, the number of questions, the percentage of correct answers, and the pass/fail result will appear on the score page.



Test Final Test 1

Regarding the interpolation control and synchronous control, select the correct answer(s). (You may select multiple answers.)

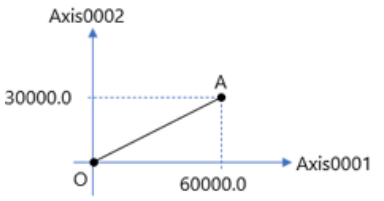
Q1

- [ ] **Before starting the interpolation control, set the operation profile and enable it with the program.**
- [ ] **After the interpolation control is ended, disable the axes group.**
- [ ] **To perform cam operation in the synchronous control, use operation profile data and MC_GearIn.**
- [ ] **To end the synchronous control, execute MC_STOP.**

Test **Final Test 2**

Regarding the linear interpolation control program, select the appropriate values to fill in the blanks below.

The linear interpolation is performed from the home position O to Point A.



* AxesGroup001 consists of the following.
Configuration axis [1]: Axis0001
Configuration axis [2]: Axis0002

```
wAxesNum[0]: = (Q1);
wAxesNum[1]: = (Q2);

lePointA[0]: = (Q3);
lePointA[1]: = (Q4);

wDirection[0]: = MC_DIRECTION_mcShortestWay;
wDirection[1]: = MC_DIRECTION_mcShortestWay;

...
(Omitted)
...

MCv_MoveLinearInterpolateAbsolute_1(
  AxesGroup       : = AxesGroup001.AxesGroupRef,
  Execute         : = bGrpEnDone,
  LinearAxes      : = wAxesNum,
  Position        : = lePointA,
  Velocity        : = leVelocity,
  Acceleration    : = leAcceleration,
  Deceleratoion   : = leDeceleration,
  Jerk            : = leJerk,
  VelocityMode    : =
    MC_INTERPOLATE_SPEED_MODE_VectorSpeed,
  Direction       : = wDirection,
  Active          => bMoveLinear1Active,
  Done            => bMoveLinear1Done
);

...
```

Specify the array of (Q5) elements in wAxesNum, lePointA, and wDirection.

Q1 Select the appropriate answer.

Q2 Select the appropriate answer.

Q3 Select the appropriate answer.

Q4 Select the appropriate answer.

Q5 Select the appropriate answer.

Q1:
- 0
- 1
- 2

Q2:
- 0
- 1
- 2

Q3:
- 0.0
- 30000.0
- 60000.0

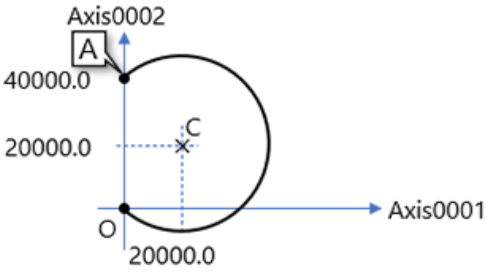
Q4:
- 0.0
- 30000.0
- 60000.0

Q5:
- 0
- 2
- 4
- 16

Test **Final Test 3**

Regarding the linear interpolation control program, select the appropriate values to fill in the blanks below.

The circular interpolation of center point specification is performed starting from the home position O to Point A with Point C used as the center.



* AxesGroup001 consists of the following.
Configuration axis [1]: Axis0001
Configuration axis [2]: Axis0002

```
wCircAxesNum[0]: = (Q1);
wCircAxesNum[1]: = (Q2);

leAuxPoint[0]: = (Q3);
leAuxPoint[1]: = (Q4);

lePointA[0]: = (Q5);
lePointA[1]: = (Q6);

...
(Omitted)
...

MCv_MoveCircularInterpolateAbsolute_1(
  AxesGroup      : = AxesGroup001.AxesGroupRef,
  Execute        : = bGroupEnDone,
  CircAxes       : = wCircAxesNum,
  CircMode       : = (Q7),
  AuxPoint       : = leAuxPoint
  EndPoint       : = lePointA
  PathChoice     : = (Q8),
  Velocity       : = leVelocity,
  Acceleration   : = leAcceleration,
  Deceleratoion  : = leDeceleration,
  Jerk           : = leJerk,
  Active         => bMoveCirc1Active,
  Done           => bMoveCirc1Done
);

...
```

Specify the array of (Q9) elements in wCircAxesNum.
Specify the array of (Q10) elements in leAuxPoint and lePointA.

Q1	Select the appropriate answer.	**Q2**	Select the appropriate answer.
Q3	Select the appropriate answer.	**Q4**	Select the appropriate answer.
Q5	Select the appropriate answer.	**Q6**	Select the appropriate answer.
Q7	Select the appropriate answer.	**Q8**	Select the appropriate answer.
Q9	Select the appropriate answer.	**Q10**	Select the appropriate answer.

Q1:
- 0
- 1
- 2

Q2:
- 0
- 1
- 2

Q3:
- 0.0
- 20000.0
- 40000.0

Q4:
- 0.0
- 20000.0
- 40000.0

Q5:
- 0.0
- 20000.0
- 40000.0

Q6:
- 0.0
- 20000.0
- 40000.0

Q7:
- MC_CIRC_MODE__mcBorder
- MC_CIRC_MODE__mcCenter
- MC_CIRC_MODE__mcRadius

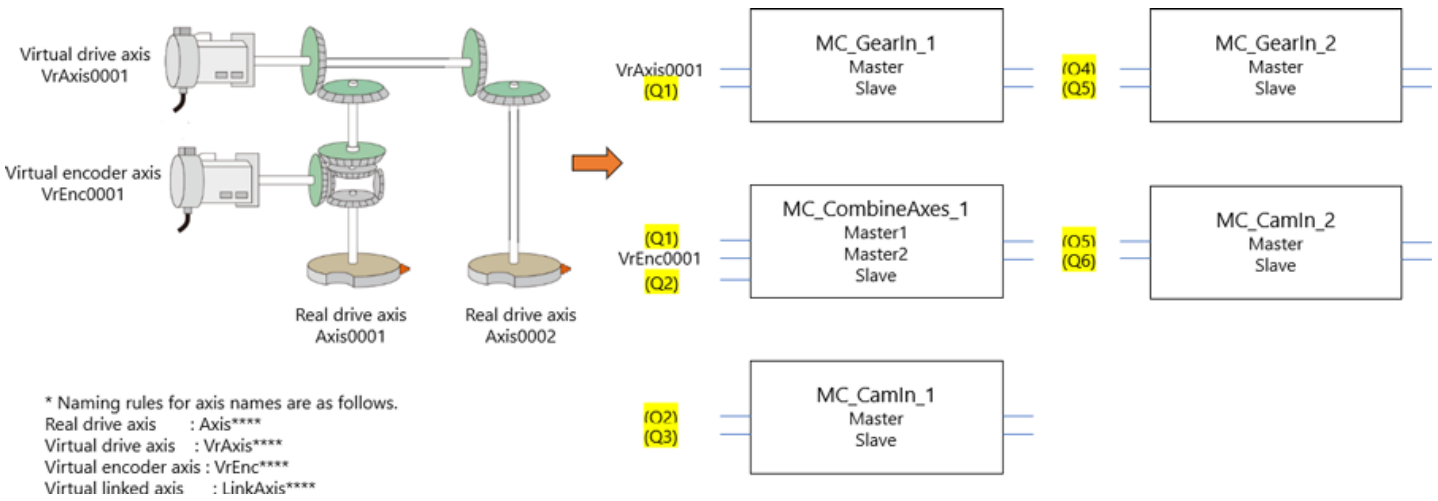
Q8:
- MC_CIRC_PATHCHOICE__mcCW
- MC_CIRC_PATHCHOICE__mcCCW

Q9:
- 2
- 4
- 16

Q10:
- 2
- 4
- 16

Test **Final Test 4**

Regarding the linear interpolation control program, select the appropriate values to fill in the blanks below.



Q1 Select the appropriate answer.

Q2 Select the appropriate answer.

Q3 Select the appropriate answer.

Q4 Select the appropriate answer.

Q5 Select the appropriate answer.

Q6 Select the appropriate answer.

Q1:
- LinkAxis0001
- Ax0001
- VrAxis0002

Q2:
- LinkAxis0001
- LinkAxis0002
- Axis0001

Q3:
- LinkAxis0003
- VrAx0002
- Axis0001

Q4:
- VrAxis0001
- VrAxis0002
- Axis0002

Q5:
- LinkAxis0002
- LinkAxis0003
- VrEnc0001

Q6:
- LinkAxis0003
- VrAx0002
- Axis0002

Test Final Test 1

Regarding the interpolation control and synchronous control, select the correct answer(s). (You may select multiple answers.)

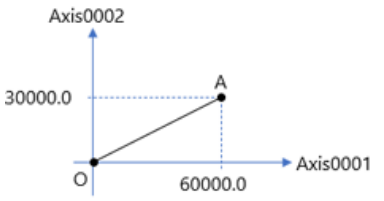
Q1

- [ ] **Before starting the interpolation control, set the operation profile and enable it with the program.**
- [x] **After the interpolation control is ended, disable the axes group.**
- [ ] **To perform cam operation in the synchronous control, use operation profile data and MC_GearIn.**
- [x] **To end the synchronous control, execute MC_STOP.**

Test **Final Test 2**

Regarding the linear interpolation control program, select the appropriate values to fill in the blanks below.

The linear interpolation is performed from the home position O to Point A.



* AxesGroup001 consists of the following.
Configuration axis [1]: Axis0001
Configuration axis [2]: Axis0002

```
wAxesNum[0]: = (Q1);
wAxesNum[1]: = (Q2);

lePointA[0]: = (Q3);
lePointA[1]: = (Q4);

wDirection[0]: = MC_DIRECTION_mcShortestWay;
wDirection[1]: = MC_DIRECTION_mcShortestWay;

...
(Omitted)
...

MCv_MoveLinearInterpolateAbsolute_1(
  AxesGroup        : = AxesGroup001.AxesGroupRef,
  Execute          : = bGrpEnDone,
  LinearAxes       : = wAxesNum,
  Position         : = lePointA,
  Velocity         : = leVelocity,
  Acceleration     : = leAcceleration,
  Deceleratoion    : = leDeceleration,
  Jerk             : = leJerk,
  VelocityMode     : =
    MC_INTERPOLATE_SPEED_MODE_VectorSpeed,
  Direction        : = wDirection,
  Active           => bMoveLinear1Active,
  Done             => bMoveLinear1Done
);

...
```

Specify the array of (Q5) elements in wAxesNum, lePointA, and wDirection.

Q1 1 **Q2** 2

Q3 60000.0 **Q4** 30000.0

Q5 16

Q1: • 0
• 1
• 2

Q2: • 0
• 1
• 2

Q3: • 0.0
• 30000.0
• 60000.0

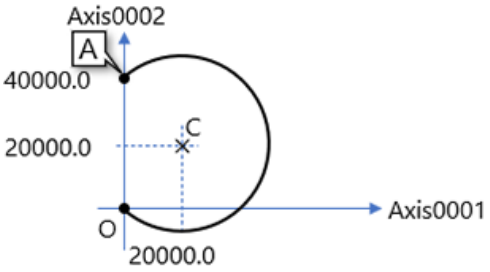
Q4: • 0.0
• 30000.0
• 60000.0

Q5: • 0
• 2
• 4
• 16

Test **Final Test 3**

Regarding the linear interpolation control program, select the appropriate values to fill in the blanks below.

The circular interpolation of center point specification is performed starting from the home position O to Point A with Point C used as the center.



* AxesGroup001 consists of the following.
Configuration axis [1]: Axis0001
Configuration axis [2]: Axis0002

```
wCircAxesNum[0]: = (Q1);
wCircAxesNum[1]: = (Q2);

leAuxPoint[0]: = (Q3);
leAuxPoint[1]: = (Q4);

lePointA[0]: = (Q5);
lePointA[1]: = (Q6);

...
(Omitted)
...

MCv_MoveCircularInterpolateAbsolute_1(
  AxesGroup      : = AxesGroup001.AxesGroupRef,
  Execute        : = bGroupEnDone,
  CircAxes       : = wCircAxesNum,
  CircMode       : = (Q7),
  AuxPoint       : = leAuxPoint
  EndPoint       : = lePointA
  PathChoice     : = (Q8),
  Velocity       : = leVelocity,
  Acceleration   : = leAcceleration,
  Deceleratoion  : = leDeceleration,
  Jerk           : = leJerk,
  Active         => bMoveCirc1Active,
  Done           => bMoveCirc1Done
);

...
```

Specify the array of (Q9) elements in wCircAxesNum.
Specify the array of (Q10) elements in leAuxPoint and lePointA.

Q1	1	**Q2**	2
Q3	20000.0	**Q4**	20000.0
Q5	0.0	**Q6**	40000.0
Q7	MC_CIRC_MODE__mcCenter	**Q8**	MC_CIRC_PATHCHOICE__mcCCW
Q9	2	**Q10**	16

Q1:
- 0
- 1
- 2

Q2:
- 0
- 1
- 2

Q3:
- 0.0
- 20000.0
- 40000.0

Q4:
- 0.0
- 20000.0
- 40000.0

Q5:
- 0.0
- 20000.0
- 40000.0

Q6:
- 0.0
- 20000.0
- 40000.0

Q7:
- MC_CIRC_MODE__mcBorder
- MC_CIRC_MODE__mcCenter
- MC_CIRC_MODE__mcRadius

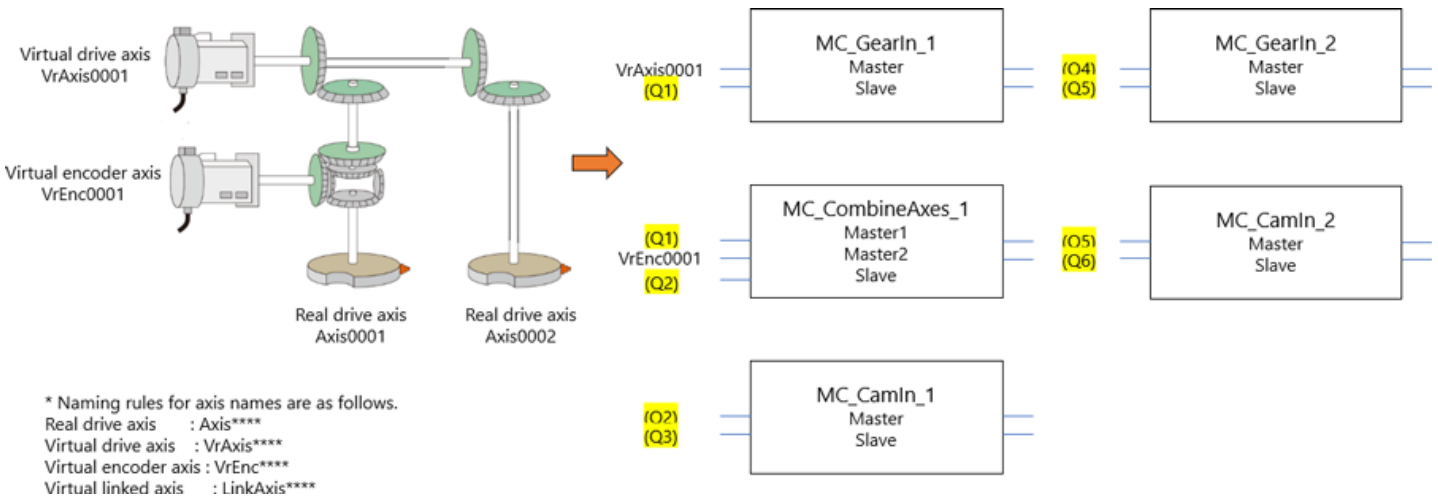
Q8:
- MC_CIRC_PATHCHOICE__mcCW
- MC_CIRC_PATHCHOICE__mcCCW

Q9:
- 2
- 4
- 16

Q10:
- 2
- 4
- 16

Test **Final Test 4**

Regarding the linear interpolation control program, select the appropriate values to fill in the blanks below.



Q1 LinkAxis0001 **Q2** LinkAxis0002

Q3 Axis0001 **Q4** VrAxis0001

Q5 LinkAxis0003 **Q6** Axis0002

Q1:
- LinkAxis0001
- Ax0001
- VrAxis0002

Q2:
- LinkAxis0001
- LinkAxis0002
- Axis0001

Q3:
- LinkAxis0003
- VrAx0002
- Axis0001

Q4:
- VrAxis0001
- VrAxis0002
- Axis0002

Q5:
- LinkAxis0002
- LinkAxis0003
- VrEnc0001

Q6:
- LinkAxis0003
- VrAx0002
- Axis0002

Test **Test Score**

You have completed the Final Test. You results area as follows.
To end the Final Test, proceed to the next page.

		1	2	3	4	5	6	7	8	9	10
	Final Test 1	✓									
	Final Test 2	✓	✓	✓	✓	✓					
	Final Test 3	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Final Test 4	✓	✓	✓	✓	✓	✓				

Total questions: **22**

Correct answers: **22**

Percentage: **100** %

Clear

You have completed the MELSEC iQ-R Series Motion Module Application (RD78G(H) Interpolation/Synchronous Control) Course.

Thank you for taking this course.

We hope you enjoyed the lessons and the information you acquired in this course is useful for configuring systems in the future.

You can review the course as many times as you want.

Review

Close