Servo System Controller

MELSEC iQ-R Series Motion Module Basics (RD78G(H) Positioning Control)

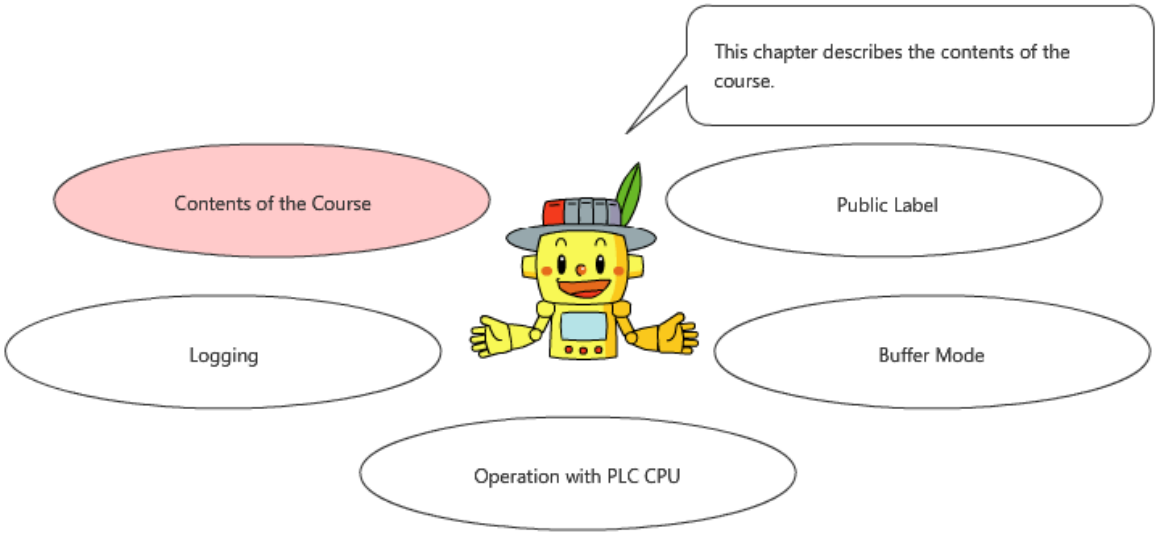
This training course is intended for those who will construct a motion control system using the MELSEC iQ-R Series Motion module for the first time.

Click the Forward button at the upper right corner to proceed to the next page.

L(NA)00260ENG

Introduction Purpose of the Course

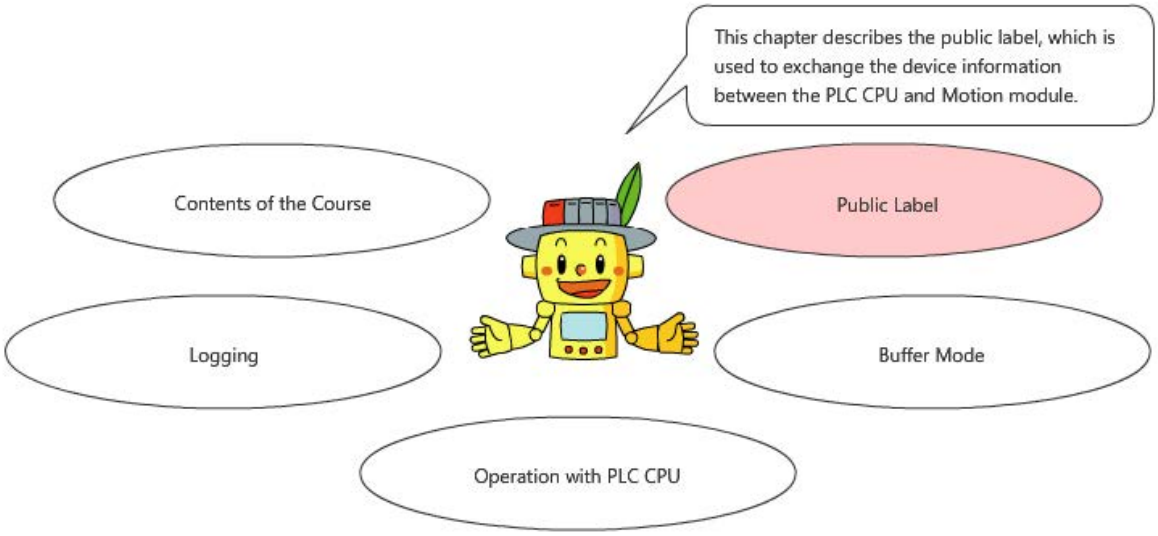
This course aims to develop knowledge and understanding about positioning control of the Motion control system using the MELSEC iQ-R Series Motion module.



This course is a continuation of MELSEC iQ-R Series Motion Module Basics (RD78G(H) Startup).
Make sure you finish the Startup course before taking this course.

Introduction Purpose of the Course

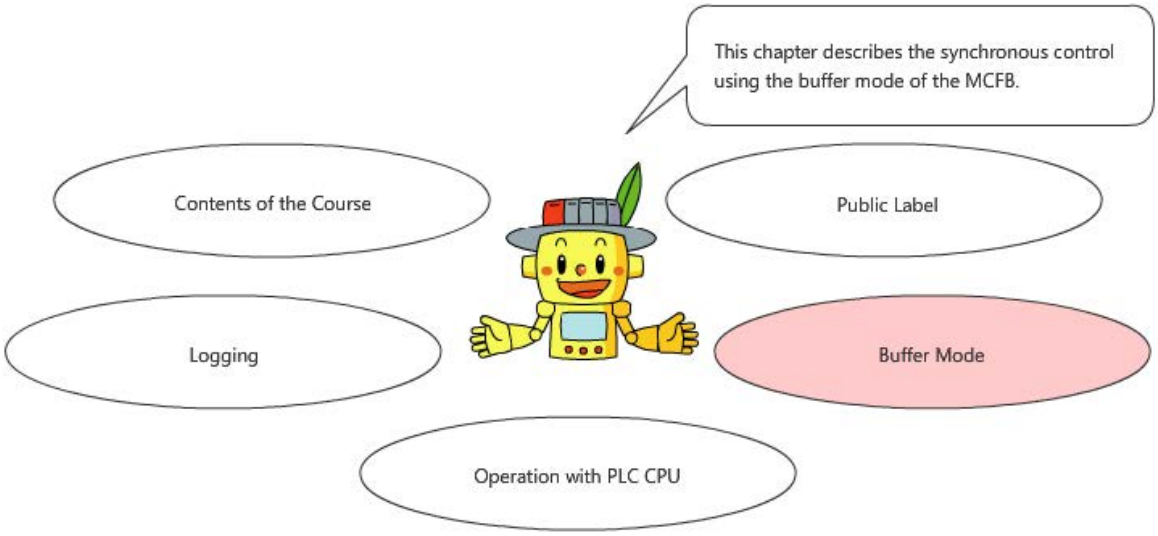
This course aims to develop knowledge and understanding about positioning control of the Motion control system using the MELSEC iQ-R Series Motion module.



This course is a continuation of MELSEC iQ-R Series Motion Module Basics (RD78G(H) Startup).
Make sure you finish the Startup course before taking this course.

Introduction Purpose of the Course

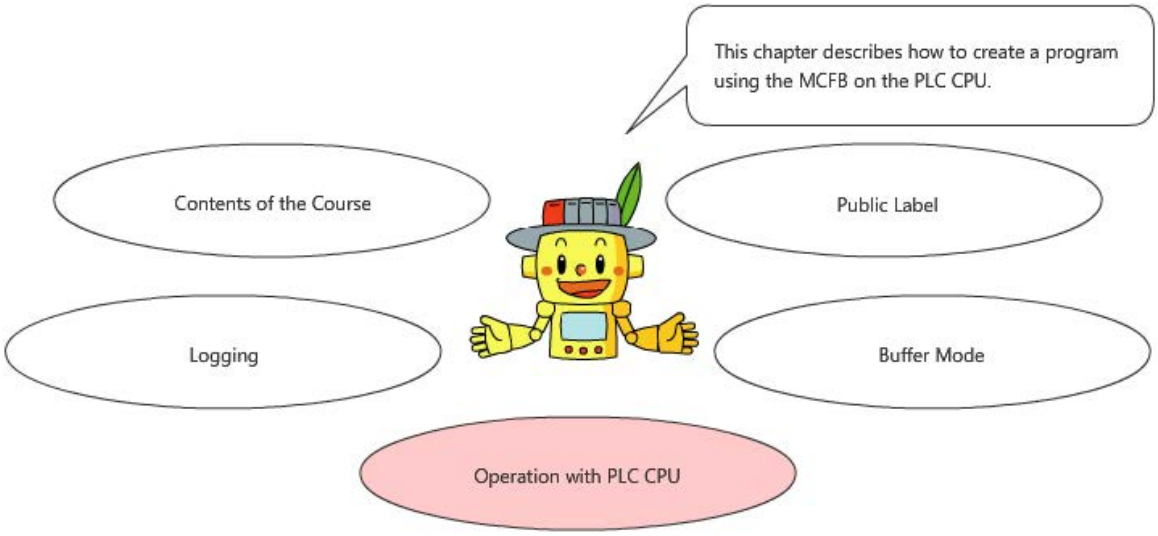
This course aims to develop knowledge and understanding about positioning control of the Motion control system using the MELSEC iQ-R Series Motion module.



This course is a continuation of MELSEC iQ-R Series Motion Module Basics (RD78G(H) Startup).
Make sure you finish the Startup course before taking this course.

Introduction Purpose of the Course

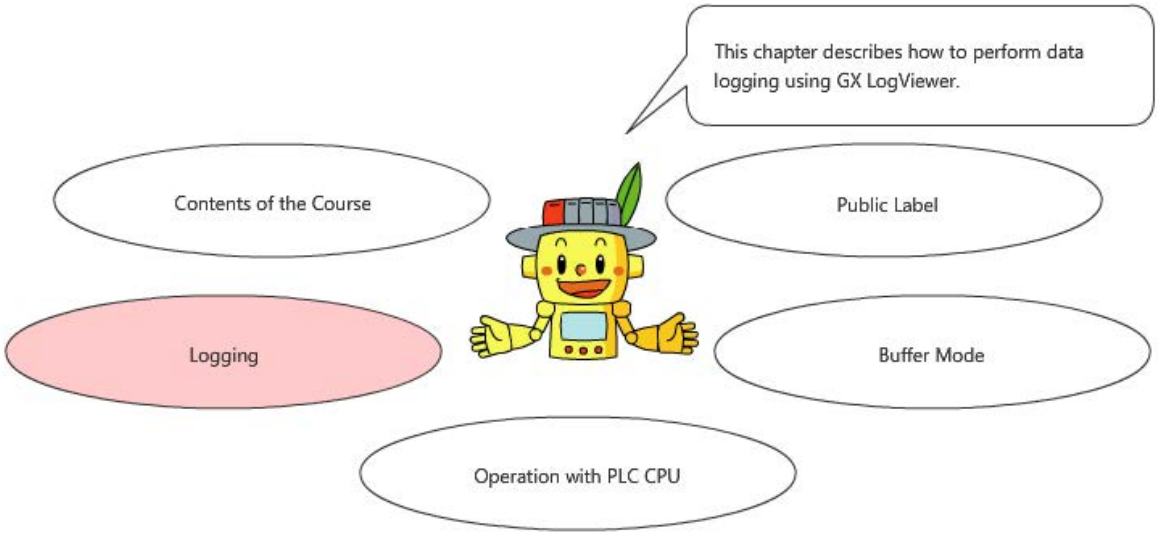
This course aims to develop knowledge and understanding about positioning control of the Motion control system using the MELSEC iQ-R Series Motion module.



This course is a continuation of MELSEC iQ-R Series Motion Module Basics (RD78G(H) Startup).
Make sure you finish the Startup course before taking this course.

Introduction Purpose of the Course

This course aims to develop knowledge and understanding about positioning control of the Motion control system using the MELSEC iQ-R Series Motion module.



This course is a continuation of MELSEC iQ-R Series Motion Module Basics (RD78G(H) Startup).
Make sure you finish the Startup course before taking this course.

Introduction Course Structure

The contents of this course are as follows.
We recommend that you start from Chapter 1.

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 in 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 Website.

MELSOFT GX Works3	Ver.1.066U	Motion Control Setting function	Ver.1.012N
GX LogViewer	Ver.1.106K		
MELSOFT MR Configurator2	Ver.1.110Q or later		

The firmware version of the PLC CPU must be 44 or later (46 or later for RD78GH).
The firmware version of the motion module must be 10 or later.
For how to update the firmware version, refer to MITSUBISHI ELECTRIC FA Website or the module configuration manual.

The icon indicates the reference manual.
The contents of the manuals described in this course are those of the following versions.
If the versions differ, the location of description and contents may be slightly different.

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

Chapter 1 Contents of the Course

1.1 Course Overview

The following shows the course overview.

Chapter 1 Contents of the Course

This chapter describes the contents of the course.

Chapter 2 Public Label

This chapter describes the public label, which is used to exchange the device information between the PLC CPU and Motion module.

Chapter 3 Buffer Mode

This chapter describes the synchronous control using the buffer mode of the Motion control FB.

Chapter 4 Operation with PLC CPU

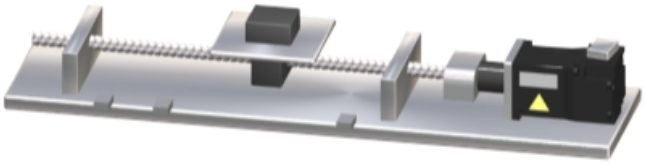
This chapter describes how to create a program using the Motion control FB on the PLC CPU.

Chapter 5 Logging

This chapter describes how to perform data logging using GX LogViewer to check the operation of the Motion module.

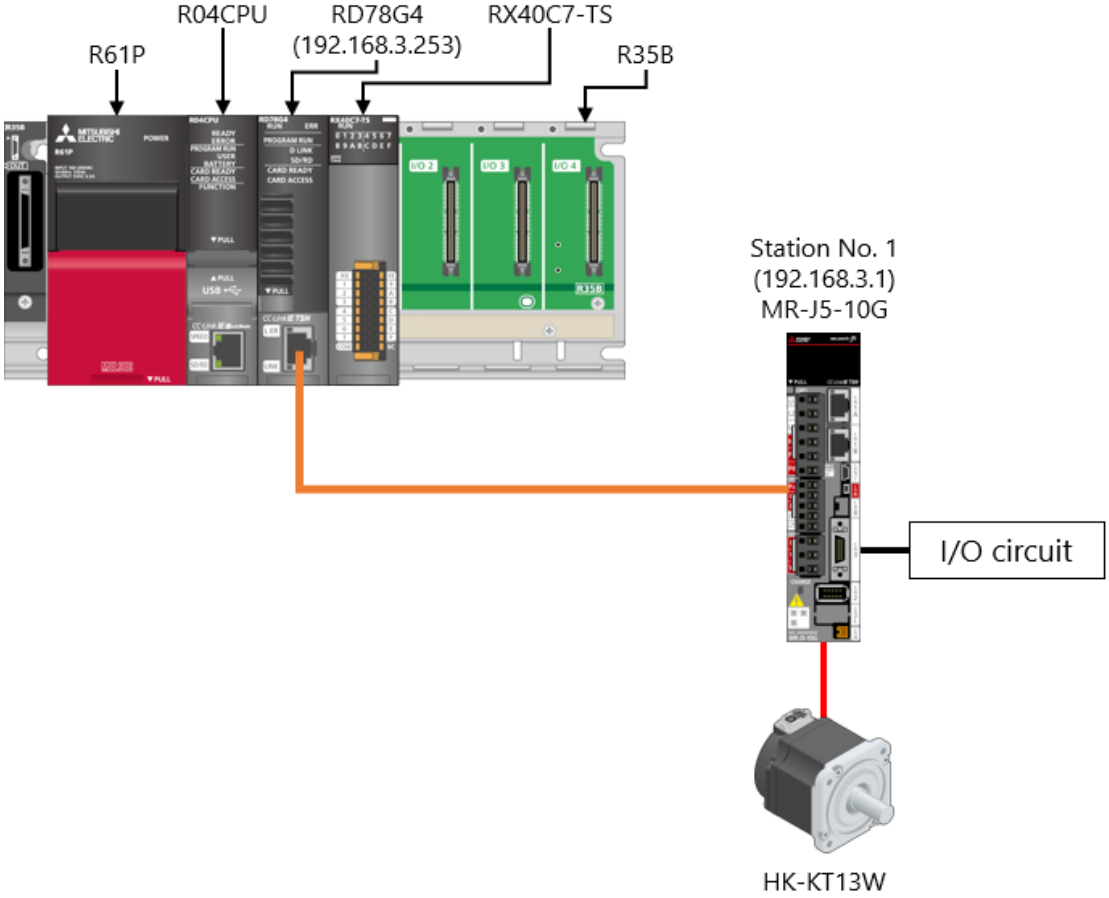
1.2 Target Machine Configuration

This course uses the same single-shaft ball screw mechanism as used in the Startup course.



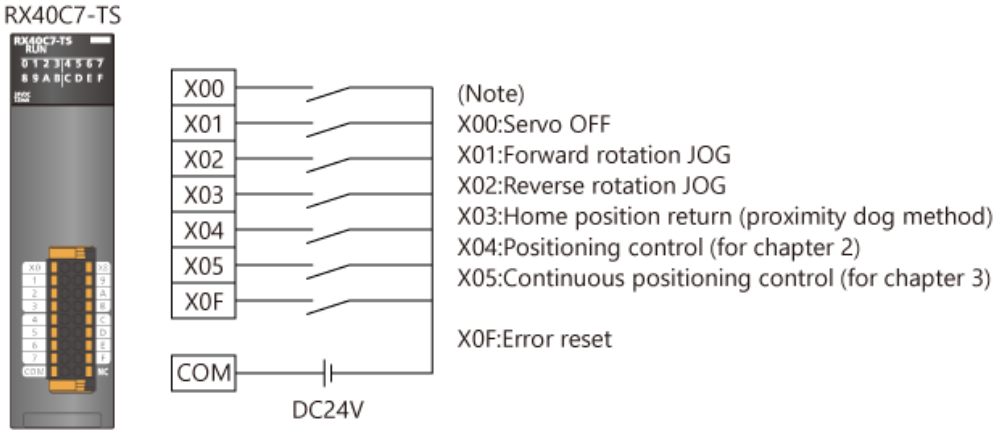
1.3 Target System Configuration

The configuration of the target system is as follows.
Remove the remote input module from the system used in the Startup course, and add the input module RX40C7-TS to slot 1 of the base unit of the programmable controller.
The station No. of the servo amplifier MR-J5-10G has been changed to 1, and the IP address has been changed to 192.168.3.1.



1.4 Wiring of External Circuit

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



(Note) Since the I/O No. of RX40C7-TS is 0020H, X20 to X25 and X2F are used in the program.

Chapter 2 Public Label

When the Motion module is controlled by the input module of the programmable controller, as in the system used in this course described in chapter 1, the PLC CPU and Motion module must exchange the device information.

There are following two methods.

1. Use public labels.
2. Use buffer memory of the Motion module.

This chapter describes how to exchange data by using public labels.

Download the sample program to be used in this chapter and chapter 3 by clicking the link below.

RD78GBasic2_sample1.zip (1.34MB)

[Point]
When using the buffer memory, copy the data to be exchanged to the user area (U□\G11478000 to G11997999).

(Program example)

<PLC CPU>

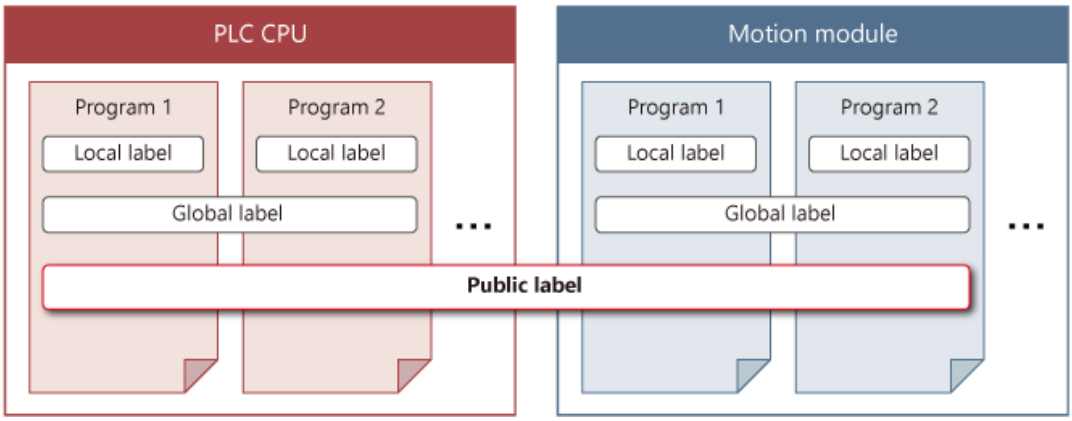
Positioning start

<Motion module>

```
MC_MoveAboslute_1(
    Execute:= G11478000.0 ,
      :
      :
);
```

2.1 What is Public Label?

A public label is a shared label that can be used in both the Motion module and PLC CPU.
The following shows the applicable areas of the local label, global label, and public label.



2.2 Public Label Setting

(1) How to register public labels

Register the public labels from the global labels of the Motion module.

Make sure that the "Public Label" column is visible in the global label editor of the Motion Module Setting Function screen.

Set the labels to be registered as the public label to "Enabled".

This activates the "Motion Control Attribute" column.

Select whether each label is to be read or written from/to the PLC CPU.

	Label Name	Data Type	Class	Initial	Constant	Japanese	English(Display Target)	Chinese	Remark	Public Label	Motion Control Attribute
1	G_bSVONCMD	Bit	VAR_GLOBAL				Servo ON			Enabled	WRITE (=> Motion)
2	G_leJogVelocity	FLOAT [Double Precision]	VAR_GLOBAL				JOG Velocity			Enabled	WRITE (=> Motion)
3	G_bJogFwd	Bit	VAR_GLOBAL				JOG Forward			Enabled	WRITE (=> Motion)
4	G_bJogBwd	Bit	VAR_GLOBAL				JOG Backward			Enabled	WRITE (=> Motion)
5	G_bJogBusy	Bit	VAR_GLOBAL				JOG Busy			Enabled	READ (Motion =>)
6	G_lePosition0	FLOAT [Double Precision]	VAR_GLOBAL				Position0 Address			Disabled	-
7	G_bHomingCMD	Bit	VAR_GLOBAL				Homing Command			Enabled	WRITE (=> Motion)
8	G_bHomingDone	Bit	VAR_GLOBAL				Homing Done			Enabled	READ (Motion =>)
9	G_bHomingReq	Bit	VAR_GLOBAL				Homing Request			Enabled	READ (Motion =>)
10	G_bPosCMD	Bit	VAR_GLOBAL				Positioning Command			Enabled	WRITE (=> Motion)
11	G_bPosDone	Bit	VAR_GLOBAL				Positioning Done			Enabled	READ (Motion =>)
12	G_bPosReq	Bit	VAR_GLOBAL				Positioning Start Request			Enabled	READ (Motion =>)
13	G_bErrorReset	Bit	VAR_GLOBAL				Error Reset			Enabled	WRITE (=> Motion)
14	G_bContPosCMD	Bit	VAR_GLOBAL				Continuous Positioning Command			Enabled	WRITE (=> Motion)
15	G_bContPosReq	Bit	VAR_GLOBAL				Continuous Positioning Start Request			Enabled	WRITE (=> Motion)
16	G_bContPosDone	Bit	VAR_GLOBAL				Continuous Positioning Done			Enabled	READ (Motion =>)
17											

[Point]
If the column of the public label is not visible, scroll the table to the right.

(2) Data types that can be registered as the public label
The following table shows the data types that can be registered as the public label.

Variable type	Type	Array selection	Public label setting	Remarks
Global label	Simple type	No	○	Settings are not possible for the following labels and class. ■Label • String type label • Timer type label • Counter type label • Long counter type label • Retentive timer label • Long retentive time type label • Long timer type label ■Class • VAR_GLOBAL_CONSTANT class
		Yes	△(Note 1,2)	
	Structured data type	No	△(Note 3)	
		Yes	△(Note 1,2,4,5)	
	FB (Including Motion control FBs)	No	×	
		Yes	×	
Program	-	-	×	
Program block local label	-	-	×	
Structured data type	-	-	△(Note 3,5)	
Motion control FB structured data	-	-	△(Note 6,7)	

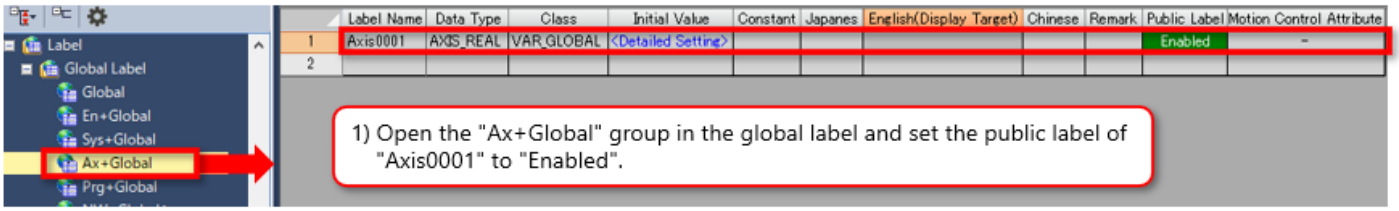
(Note)
1. The public label setting cannot be configured for each element of an array.
2. When a bit type array is used, the public label cannot be set to "Enabled". (In a structured data, only the corresponding member cannot be set to "Enabled".)
3. When the string type is used as a member of the structured data type, the member cannot be set to "Enabled".
4. Structured data with a maximum of four layers can be made public.
5. When an array of the structured data is used as a member of the structured data, the member cannot be set to "Enabled".
6. It may be used in the program of the PLCopen Motion control FB by the CPU module.
7. When the string type is used in a Motion control FB structured data, the Motion control FB structured data type itself cannot be set.

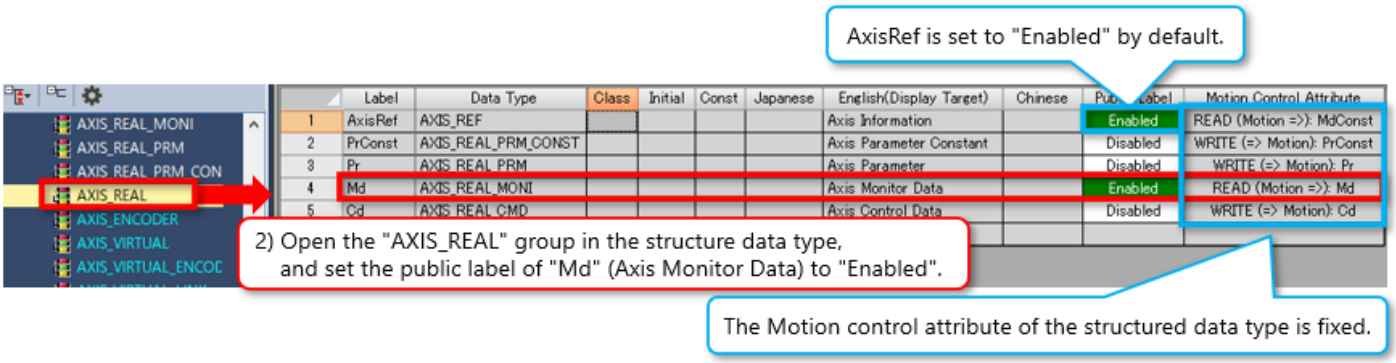
(3) How to register structured data as the public label

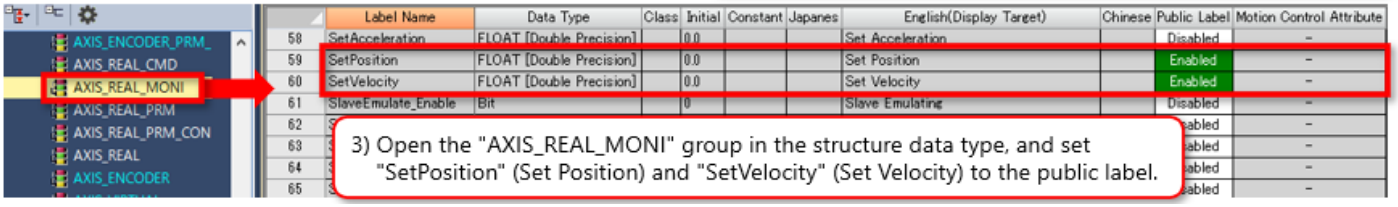
To set the members of a structured data type prepared in the system such as axis monitor data to the public label, register the public labels by layer of the structured data as shown below.

This course describes how to register Set Position (SetPosition) and Set Velocity (SetVelocity), monitor data (Md) of the actual drive axis (Axis_Real), as public labels.

[How to set AxisName.Md.SetPosition (Command current position) and AxisName.Md.SetVelocity (Command current velocity) to the public label]



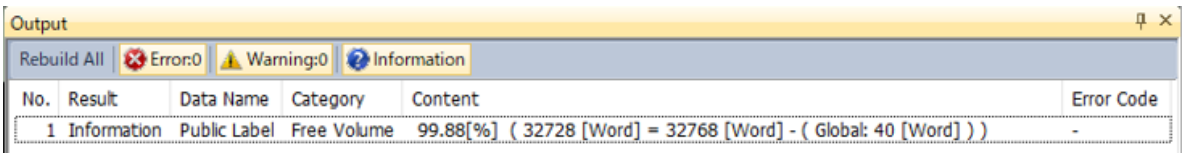




(4) Reflecting the public labels

Select [Convert] → [Rebuild All] in the menu.

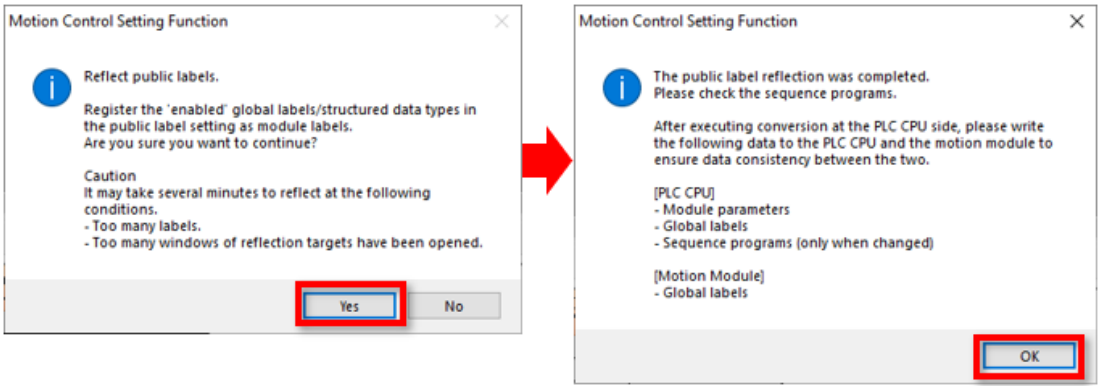
The free capacity of the public label is displayed as Information in the output window.



When the rebuild all process is successfully completed, select [Convert] → [Reflect Public Labels] in the menu.

Click [Yes] in the following pop up window.

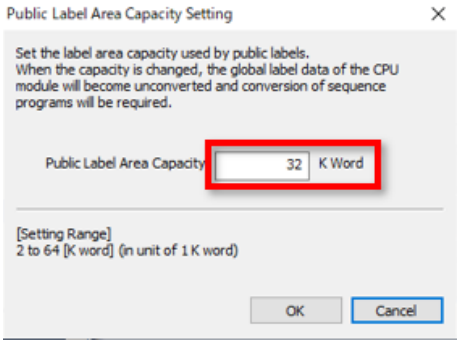
When a message indicating that the public labels are successfully reflected appears, click the [OK] button.



(Note) The memory capacity that can be used to register public labels is 32K words by default.

The capacity can be increased up to 64K words.

To change the capacity, set the memory size from [Convert] → [Public Label Capacity Setting] in the menu.



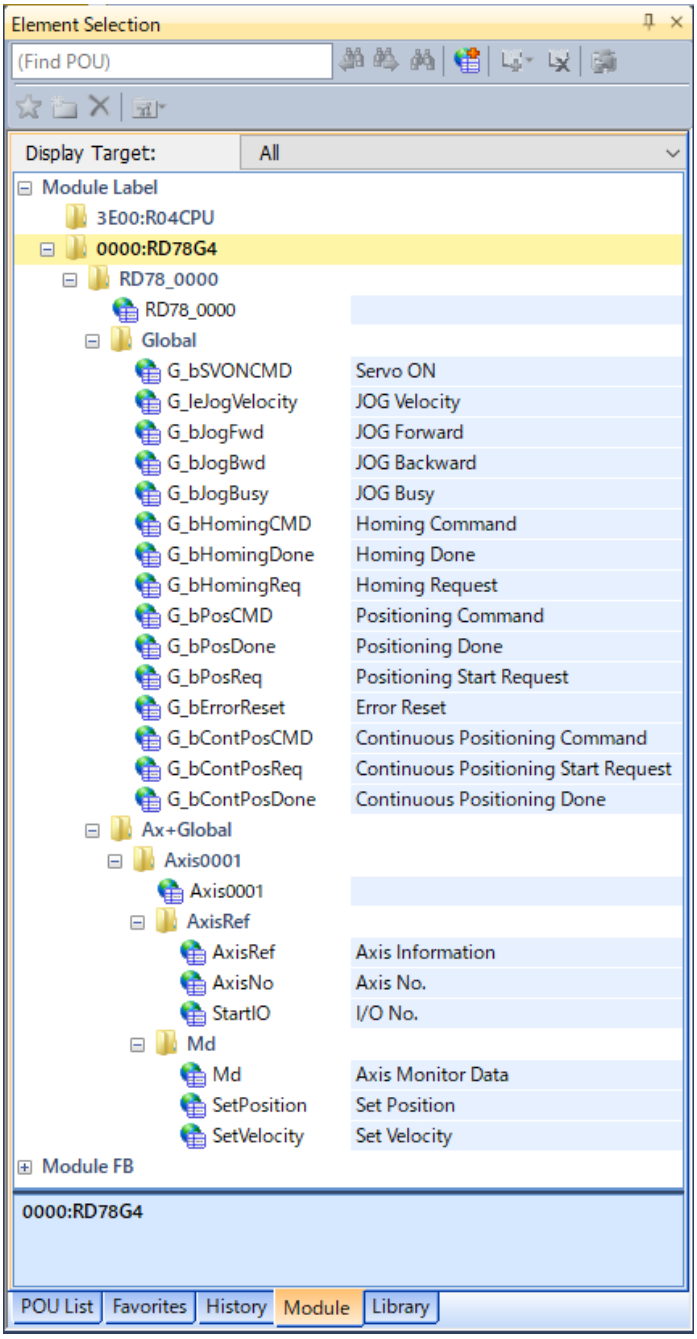
(5) Checking the labels from the PLC CPU side

The reflected public labels are registered to the module label at the PLC CPU side.

Select the module label from the Element Selection window of GX Works3, and check that the public labels have been registered under [0000:RD78G4] in [Module Label].

After changing the public label setting, always execute "Reflect Public Labels" again.

When using the public labels in the PLC CPU, rebuild all the programs.



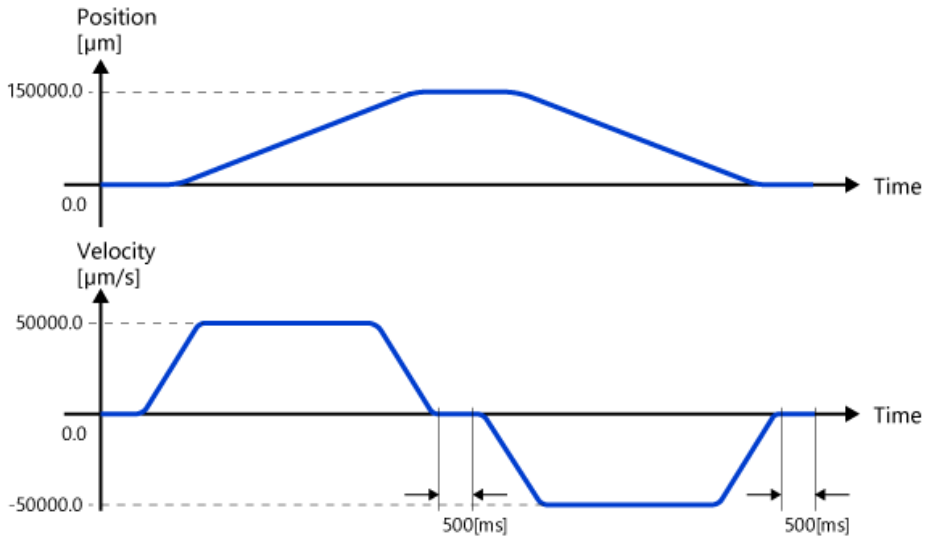
2.3 Program Example

(1) Sample program operation

The input signals of the sample program used in this chapter are assigned as follows.

Input	Operation
X20	Servo off (Note)
X21	Forward rotation JOG operation
X22	Reverse rotation JOG operation
X23	Home position return
X24	Positioning control
X25	Continuous positioning control (Chapter 3)

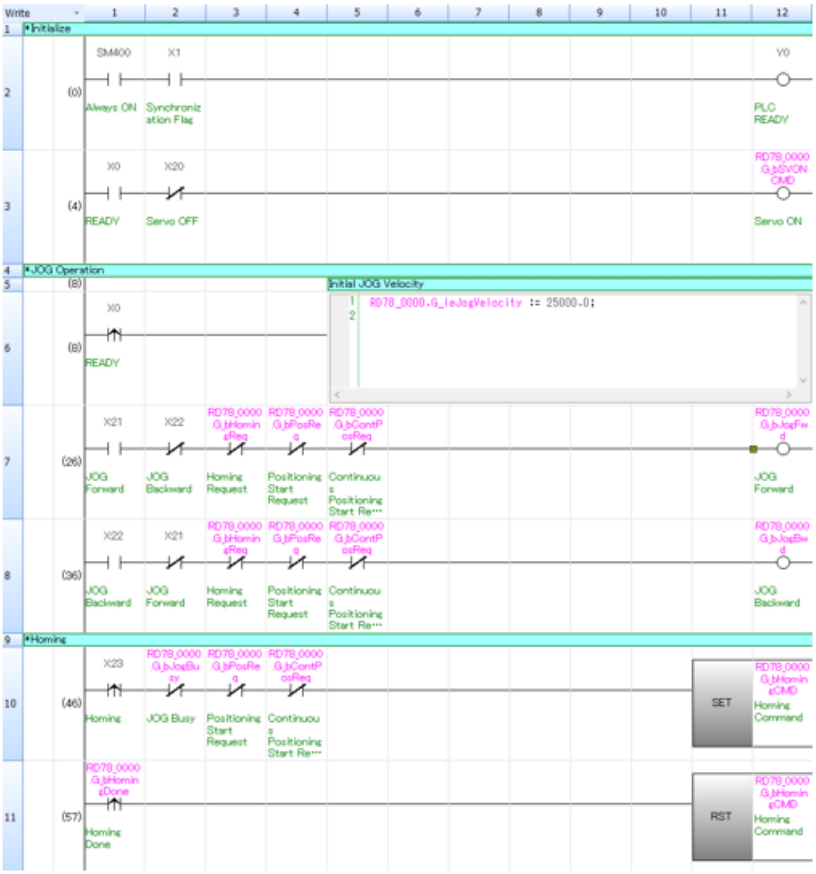
The following shows the operation pattern of X24: positioning control.



(Note) This sample program executes servo ON automatically when the PLC CPU is set to RUN.
When the power is turned on with the start signals ON, the servo motor may be activated.

(2) Program of the PLC CPU

1) MAIN (ladder, scan program)



Y0 is turned on first.

When X0 is turned on, servo ON is executed.
Turn on X20 to execute servo OFF.

Set the initial value of the JOG velocity.
This program used the inline ST.
Since the motion control attribute of the global label "G_JogVelocity" that stores the JOG velocity is set to "WRITE (→Motion)", the numerical value must be set in the PLC CPU.

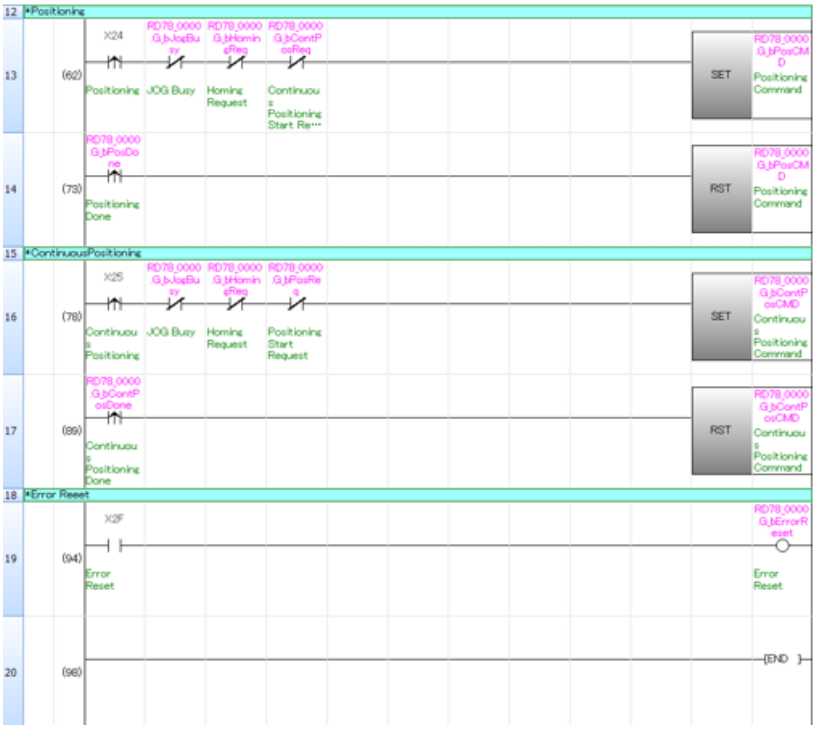
Turns on the start signal of the JOG operation.
It prevents the forward rotation and reverse rotation from being started at the same time.
An interlock is set to prevent JOG operation from being started while another program is being executed.

The startup of the home position return (X23) is retained in the public label G_bHomingCMD, and sent to the Motion module as the start condition of the home position return.
An interlock is set to prevent home position return from being started while another program is running.
Upon receiving that the Motion module turned on the home position return completion signal, G_bHomingCMD is reset at the rising edge of that signal.

(2) Program of the PLC CPU

1) Continued part of MAIN (ladder, scan program)

(Continued from previous page)



The rising edge of the positioning control start (X24) is retained in G_bPosCMD, and sent to the Motion module as the start condition of the positioning control.
An interlock is set to prevent positioning control from being started while another program is running.
Upon receiving that the Motion module turned on the home position return completion signal, G_bPosCMD is reset at the rising edge of that signal.

The start program for continuous positioning described in chapter 3.
For details, refer to 3.4.

Errors are reset when X2F is turned on.

(2) Program of the PLC CPU

2) MONITOR (ST, scan program)

SetPosition (Set Position) and SetVelocity (Set Velocity) of the axis monitor set as the public label are stored in the word devices D0 and D2.

Since SetPosition and SetVelocity are the double precision real number type, they are converted to the signed double word type so that they can be easily handled by the PLC CPU. (Note)

Although these word devices are not used in this course, they are used to display data on other sequence programs and GOT, and for other purposes.

```
1  D0:D := LREAL_TO_DINT(RD78_0000.Axis0001.Md.SetPosition);
2  D2:D := LREAL_TO_DINT(RD78_0000.Axis0001.Md.SetVelocity);
3
```

Specify the signed double word type with "D0:D".

(Note) When the double precision real number type is converted to signed double word type, if the value to be converted is outside the range from -2147483648 to 2147483647, a calculation error occurs.

(3) Program of the Motion module

1) ServoON_JOG (normal execution type)

```
1  //-----Servo ON-----
2  MC_Power_1(
3      Axis     := Axis0001.AxisRef,
4      Enable   := TRUE ,
5      ServoON  := G_bSVONCMD,
6      Busy     => bPowerBusy
7  );
8
9  //-----JOG Operation-----
10 //Initial Value Setting
11 IF (bPowerBusy) THEN
12     leJogAcceleration := 50000.0;
13     leJogDeceleration := 50000.0;
14     leJogJerk         := 0.0;
15 END_IF;
16
17 //JOG
18 MCv_Jog_1(
19     Axis         := Axis0001.AxisRef ,
20     JogForward   := G_bJogFwd ,
21     JogBackward  := G_bJogBwd ,
22     Velocity     := G_leJogVelocity ,
23     Acceleration:= leJogAcceleration ,
24     Deceleration:= leJogDeceleration ,
25     Jerk         := leJogJerk ,
26     Busy         => G_bJogBusy
27 );
28
```

Receives the servo ON signal (G_bSVONCMD) from the PLC CPU and executes servo ON.

Stores the values of JOG acceleration, JOG deceleration, and JOG jerk to the labels when the Busy output of MC_Power_1 is turned on.

Receives the JOG start signals and JOG velocity from the PLC CPU.

Returns the Busy output to the PLC CPU.

(Note) In this sample program, I/O signals of FBs that are not used or that have not been changed from the initial values are omitted.

(3) Program of the Motion module

2) Homing (normal execution type)

```
1  //-----Homing Operation-----
2  //Initial Value Setting, Operation Start Request
3  IF G_bHomingCMD THEN
4      G_lePositionD := 0.0 ;
5      G_bHomingReq  := TRUE ;
6  ELSE
7      G_bHomingReq  := FALSE ;
8  END_IF;
9
10 //Homing
11 MC_Home_1(
12     Axis            := Axis0001.AxisRef ,
13     Execute         := G_bHomingReq,
14     Position        := G_lePositionD ,
15     Done            => bHomingDone ,
16     Busy            => G_bHomingBusy ,
17     CommandAborted  => bHomingAborted ,
18     Error           => bHomingError
19 );
20
21 //Done Signal => PLC CPU
22 G_bHomingDone := bHomingDone OR bHomingAborted OR bHomingError;
23
```

Receives the home position return command signal (G_bHomingCMD) from the PLC CPU.
Stores the home position address to the label and turns on the home position return request (G_bHomingReq).

Turns off G_bHomingReq when G_bHomingCMD is turned off.

Returns the execution completion signal to the PLC CPU after the home position return is successfully completed (Done output ON),
execution is interrupted (CommandAborted output ON), or an error occurred (Error output ON).

(3) Program of the Motion module

3) Positioning (normal execution type)

```
1  //-----Positioning Operation-----
2  //Initial Value Setting, Operation Start Request
3  IF G_bPosCMD THEN
4      lePosition1        := 150000.0;
5      lePosVelocity      := 50000.0;
6      lePosAcceleration  := 100000.0;
7      lePosDeceleration  := 100000.0;
8      lePosJerk          := 200000.0;
9      G_bPosReq := TRUE;
10 ELSE
11     G_bPosReq := FALSE;
12 END_IF;
13
14 //Positioning1
15 MC_MoveAbsolute_1(
16     Axis            := Axis0001.AxisRef ,
17     Execute         := G_bPosReq ,
18     Position        := lePosition1 ,
19     Velocity        := lePosVelocity ,
20     Acceleration    := lePosAcceleration ,
21     Deceleration    := lePosDeceleration ,
22     Jerk            := lePosJerk ,
23     Direction       := MC_DIRECTION__mcShortestWay ,
24     BufferMode      := MC_BUFFER_MODE__mcAborting ,
25     Done            => bMoveAbs1Done ,
26     CommandAborted  => bMoveAbs1Aborted ,
27     Error           => bMoveAbs1Error
28 );
29 //Dwell
30 TON_1(IN:= bMoveAbs1Done ,PT:= T#500ms ,Q=> bDwell1_out );
31 //Positioning2
32 MC_MoveAbsolute_2(
33     Axis            := Axis0001.AxisRef ,
34     Execute         := bDwell1_out ,
35     Position        := G_lePosition0 ,
36     Velocity        := lePosVelocity ,
37     Acceleration    := lePosAcceleration ,
38     Deceleration    := lePosDeceleration ,
39     Jerk            := lePosJerk ,
40     Direction       := MC_DIRECTION__mcShortestWay ,
41     BufferMode      := MC_BUFFER_MODE__mcAborting ,
42     Done            => bMoveAbs2Done ,
43     CommandAborted  => bMoveAbs2Aborted ,
44     Error           => bMoveAbs2Error
45 );
46 //Dwell
47 TON_2(IN:= bMoveAbs2Done ,PT:= T#500ms ,Q=> bDwell2_out );
48
49 //Error Signal, Aborted Signal
50 bPosError   := bMoveAbs1Error OR bMoveAbs2Error;
51 bPosAborted := bMoveAbs1Aborted OR bMoveAbs2Aborted;
52
53 //Done Signal => PLC CPU
54 G_bPosDone := bDwell2_out OR bPosError OR bPosAborted;
55
```

Receives the positioning control start signal (G_bPosCMD) from the PLC CPU.
The data required for the positioning is stored to the label, and the positioning control start request (G_bPositioningReq) is turned on.

Turns off G_bPositioningReq after G_bPosCMD is turned off by the PLC CPU.

MC_MoveAbsolute executes positioning.

Starts the on-delay timer that triggers dwell when the Done output of MC_MoveAbsolute_1 is turned on.

After the dwell time is elapsed, MC_MoveAbsolute executes positioning again.

Starts the on-delay timer that triggers dwell when the Done output of MC_MoveAbsolute_2 is turned on.

Returns the execution completion signal to the PLC CPU after the dwell time is elapsed or either Error output or CommanndAborted output of MC_MoveAbsolute is turned on.

(3) Program of the Motion module

4) ErrorReset (normal execution type)

```
1  //Axis Error Reset
2  MC_Reset_1(
3      Axis    := Axis0001.AxisRef ,
4      Execute := G_bErrorReset
5  );
6
7  //System Error Reset
8  MCv_MotionErrorReset_1(
9      Execute := G_bErrorReset
10 );
11
```

Receives the error reset signal (G_bErrorReset) from the PLC CPU and executes the axis error reset and system error reset.

2.4 Writing Program

Write the program and parameters to the PLC CPU and Motion module.

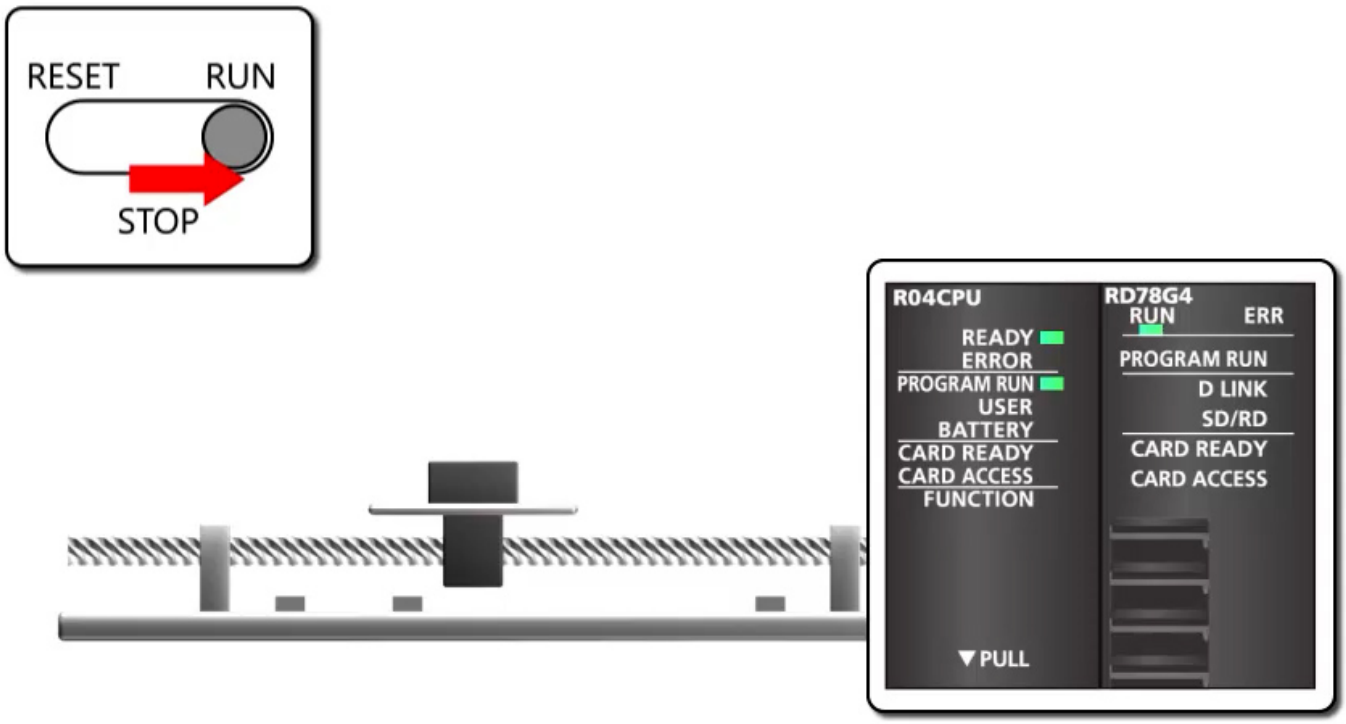
1) After all the programs in the PLC CPU are rebuilt, select [Online] → [Write to PLC] in the tool bar of GX Works3 to write all data to the PLC CPU.

2) When the parameters are written to the PLC CPU, communication with the Motion module is enabled.
   Select [Online] → [Write to Module] in the tool bar of the Motion Control Setting Function to write all data to the Motion module.

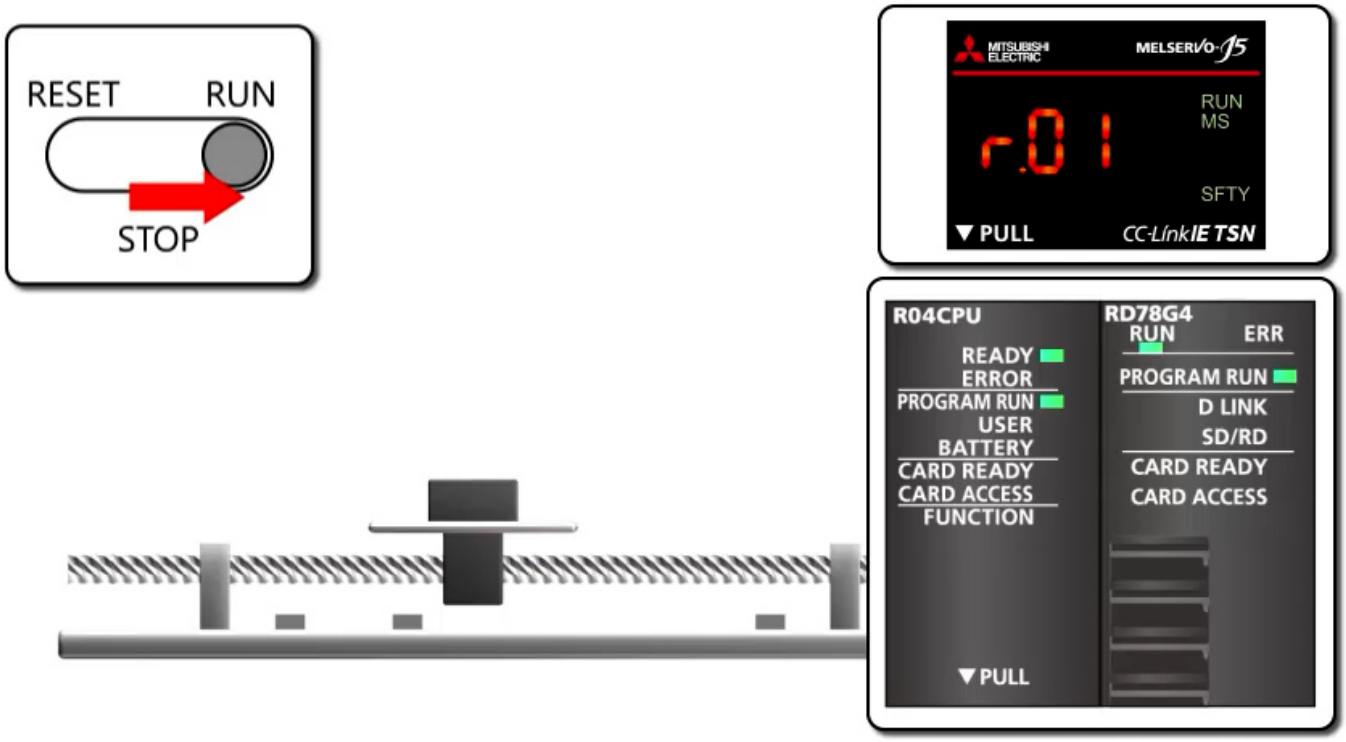
3) Reset the PLC CPU to finish the writing operation.

Click the play button at the lower left of the window.

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

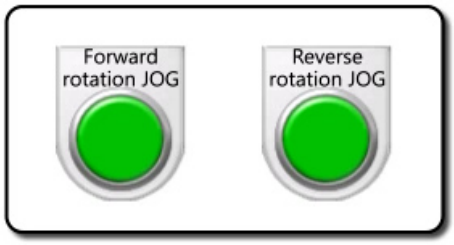


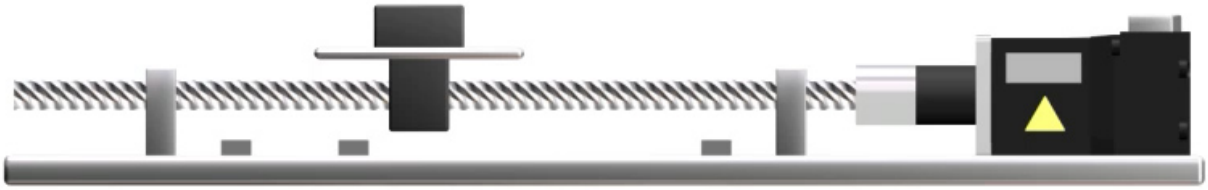
Set the RUN/STOP/RESET switch of the PLC CPU to RUN.
The READY lamp and PROGRAM RUN lamp of the PLC CPU turn on.
The RUN lamp of the Motion module turns on.



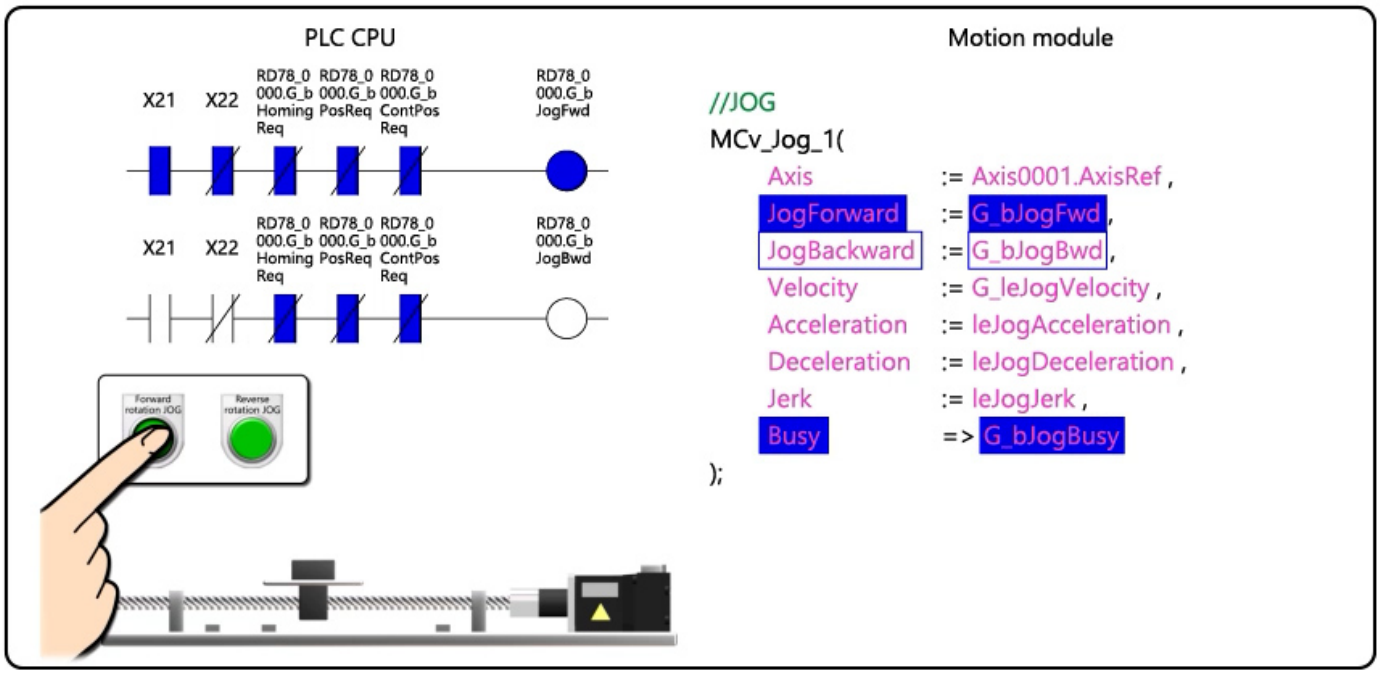
Wait until the PROGRAM RUN lamp of the Motion module turns on.
"r.01" is displayed on the servo amplifier. (The dots are lit.)
The servo motor enters the servo ON state.

Turn on X20 to execute servo OFF.
"r.01" is displayed on the servo amplifier. (The dots blink.)
Turn off X20 to execute the servo ON again.

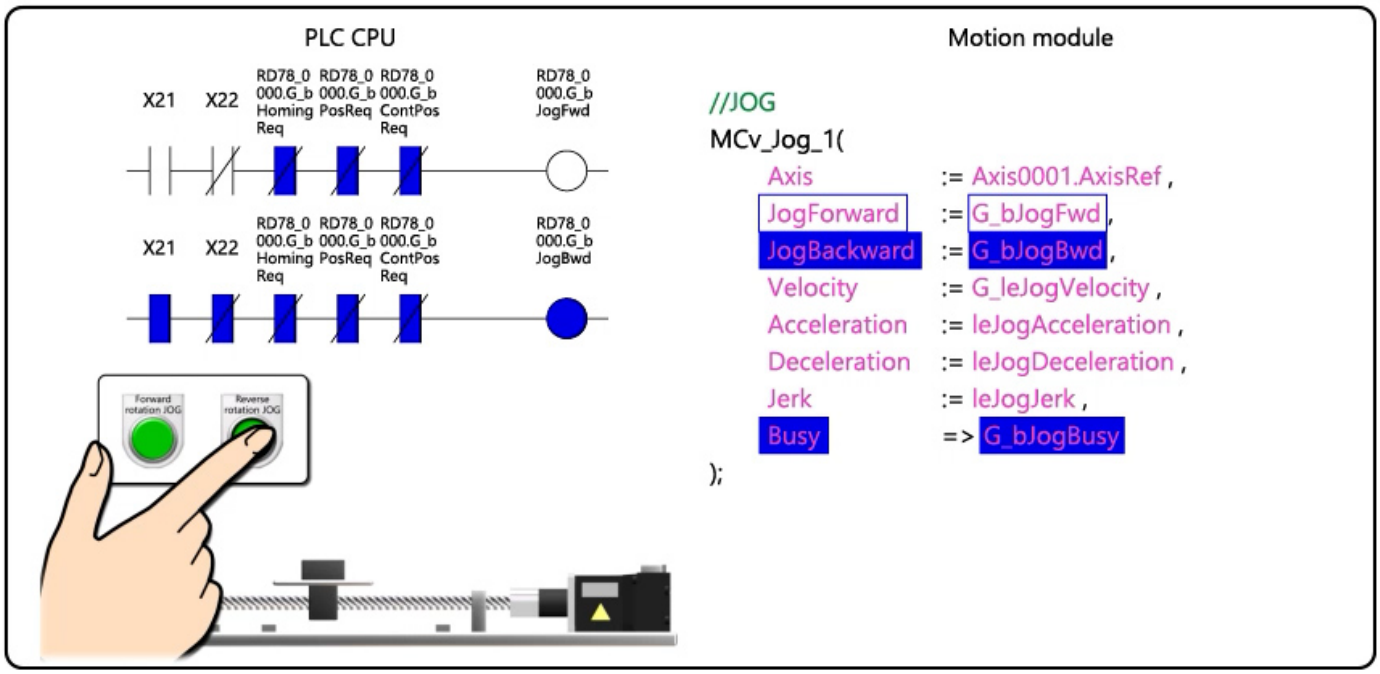




Turn on forward rotation JOG (X21) to move the axis in the address increase direction, and turn off to stop. Turn on reverse rotation JOG (X22) to move the axis to the address decrease direction, and turn off to stop.

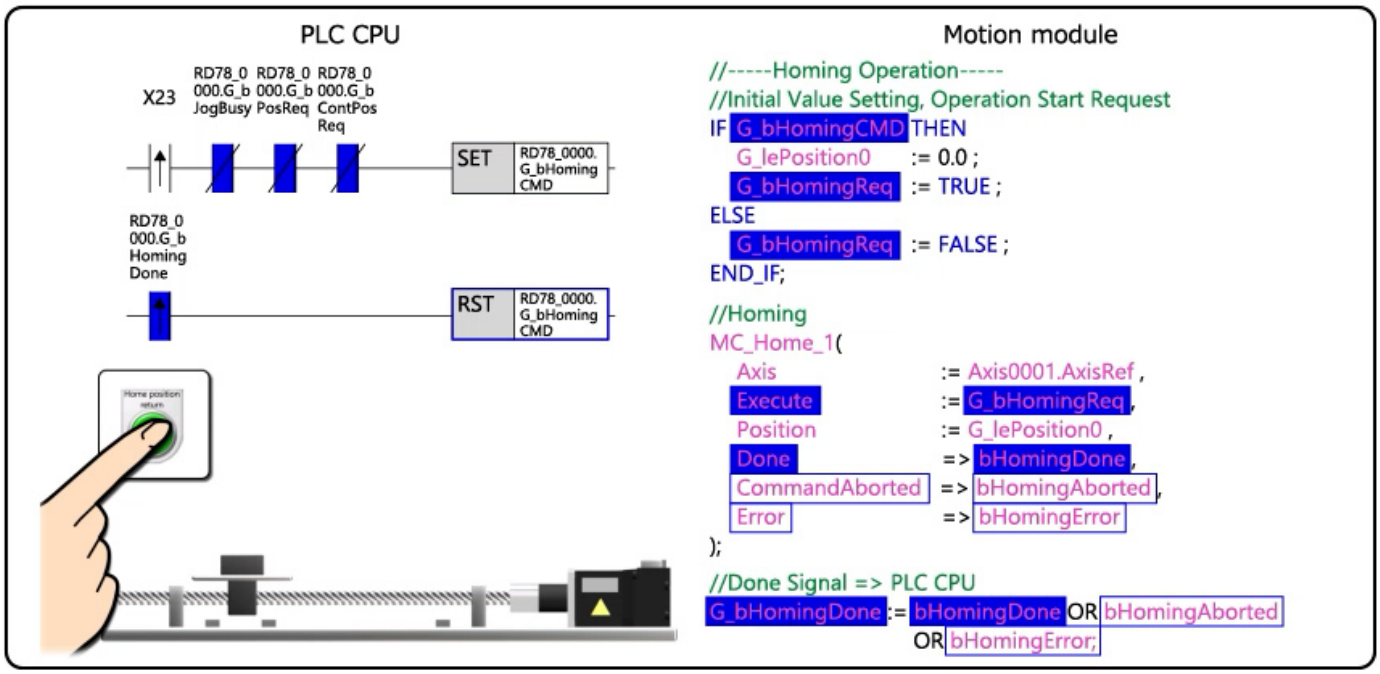


Check the program monitor.
When X21 is turned on, "RD78_0000.G_bJogFwd" turns on and "G_bJogFwd" on the Motion module side turns on.
When the JogForward input of MCv_Jog_1 turns on, the forward rotation JOG starts.



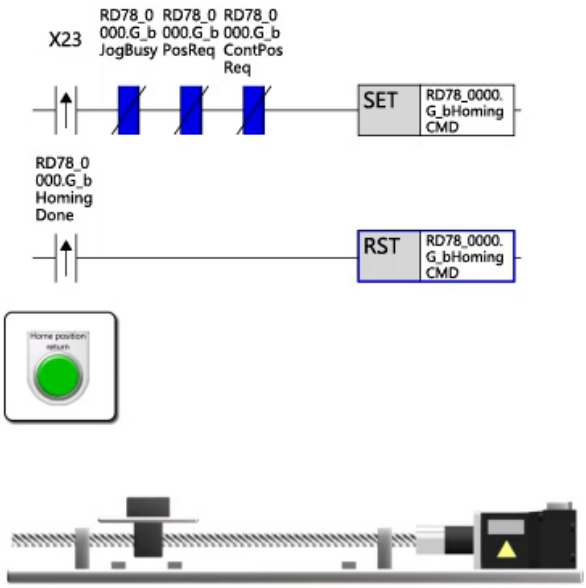
When X22 is turned on, "RD78_0000.G_bJogBwd" turns on and "G_bJogBwd" on the Motion module side turns on.
When the JogBackward input of MCv_Jog_1 turns on, the reverse rotation JOG starts.

Turn on home position return (X23) to start the home position return.
Execute the home position return with the proximity dog method
(33 is subtracted from Pr.PT45)
The axis stops a little further beyond the dog,
and sets that point as the home position.



Check the program monitor.
When X23 is turned on, "RD78_0000.G_bHomingCMD" is set.
"G_bHomingCMD" on the Motion module side turns on, and "G_bHomingReq", which is the execution command of MC_Home_1, turns on.

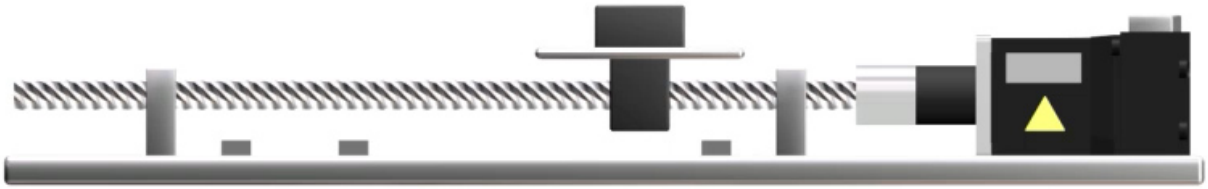
PLC CPU



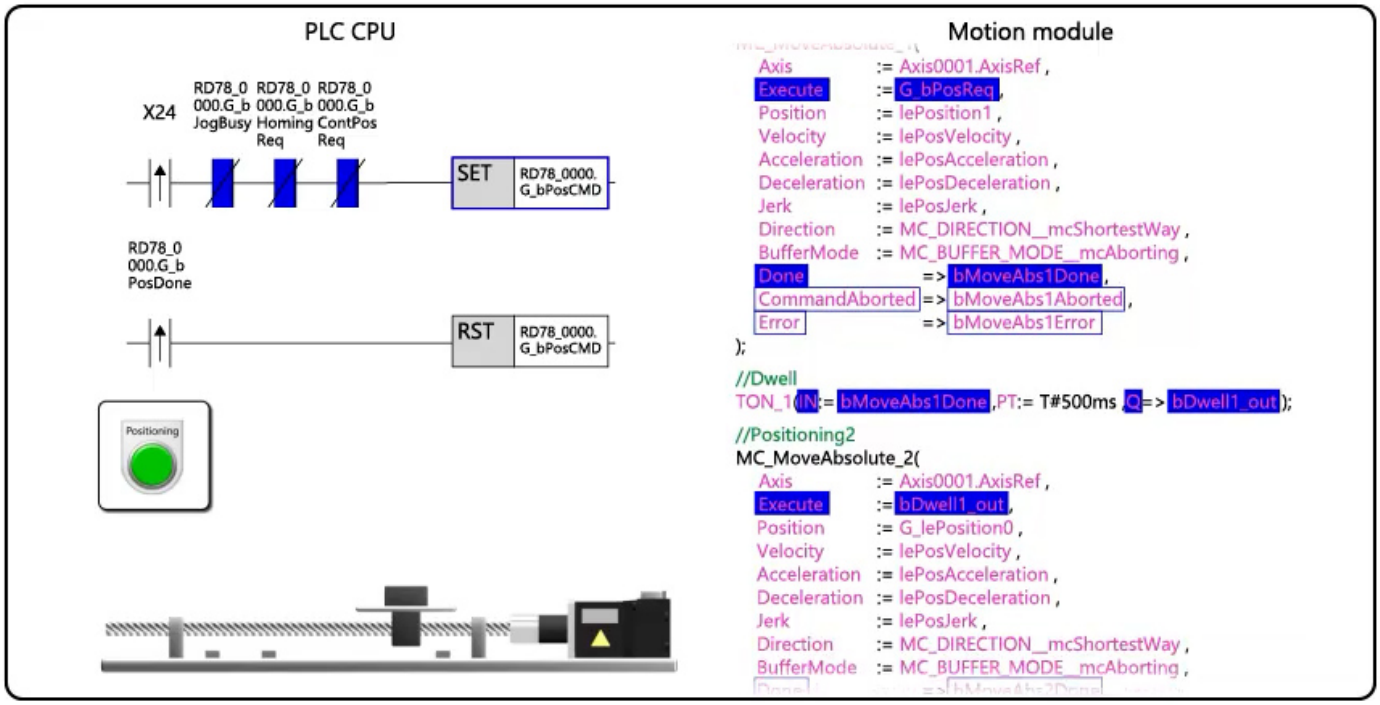
Motion module

```
//-----Homing Operation-----
//Initial Value Setting, Operation Start Request
IF G_bHomingCMD THEN
   G_lePosition0      := 0.0 ;
   G_bHomingReq       := TRUE ;
ELSE
   G_bHomingReq       := FALSE ;
END_IF;

//Homing
MC_Home_1(
   Axis               := Axis0001.AxisRef ,
   Execute            := G_bHomingReq ,
   Position           := G_lePosition0 ,
   Done               => bHomingDone ,
   CommandAborted     => bHomingAborted ,
   Error              => bHomingError
);

//Done Signal => PLC CPU
G_bHomingDone := bHomingDone OR bHomingAborted
                 OR bHomingError;
```

When the home position return is completed, the Done output and "G_bHomingDone" turn on.
"G_bHomingCMD" on the PLC CPU side is reset to the initial state.

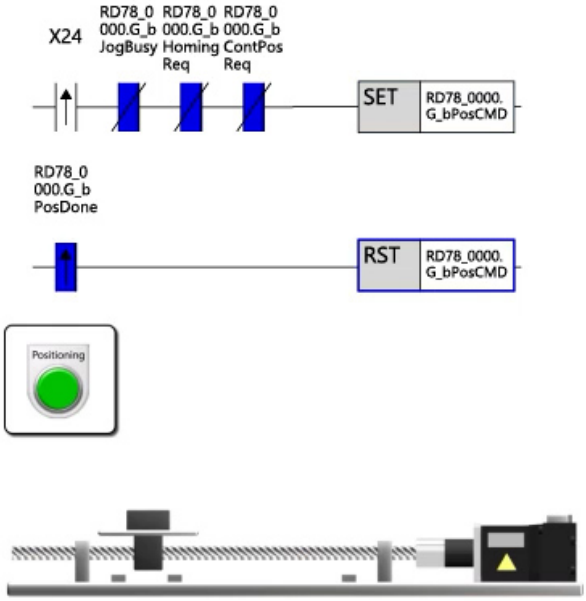


Turning on positioning start (X24) starts reciprocating motion.
The axis moves forward 150 mm and stops for 0.5 seconds,
and moves back 150 mm and stops for 0.5 seconds.



Check the program monitor.
When X24 is turned on, "RD78_0000.G_bPosCMD" is set.
"G_bPosCMD" on the Motion module side turns on, and "G_bPosReq", which is the execution command of MC_MoveAbsolute_1, turns on.

PLC CPU

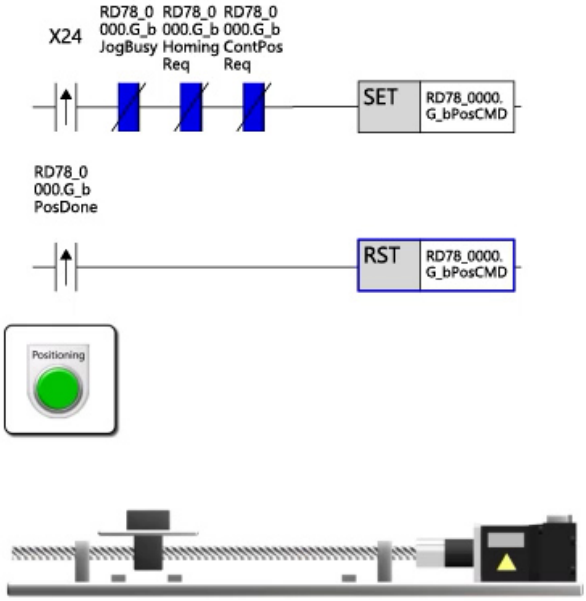


Motion module

```
//Positioning2
MC_MoveAbsolute_2(
   Axis          := Axis0001.AxisRef ,
   Execute       := bDwell1_out,
   Position      := G_lePosition0 ,
   Velocity      := lePosVelocity ,
   Acceleration  := lePosAcceleration ,
   Deceleration  := lePosDeceleration ,
   Jerk          := lePosJerk ,
   Direction     := MC_DIRECTION_mcShortestWay ,
   BufferMode    := MC_BUFFER_MODE_mcAborting ,
   Done                 => bMoveAbs2Done ,
   CommandAborted       => bMoveAbs2Aborted ,
   Error                => bMoveAbs2Error
);
//Dwell
TON_2(IN:= bMoveAbs2Done ,PT:= T#500ms ,Q=> bDwell2_out);
//Error Signal, Aborted Signal
bPosError     := bMoveAbs1Error OR bMoveAbs2Error;
bPosAborted   := bMoveAbs1Aborted OR bMoveAbs2Aborted;
//Done Signal => PLC CPU
G_bPosDone    := bDwell2_out OR bPosError OR bPosAborted;
```

After reciprocating motion is finished and dwell time is elapsed, "G_bPosDone" turns on.
"G_bPosCMD" on the PLC CPU side is reset to the initial state.

PLC CPU



Motion module

```
//Positioning2
MC_MoveAbsolute_2(
   Axis          := Axis0001.AxisRef ,
   Execute       := bDwell1_out,
   Position      := G_lePosition0 ,
   Velocity      := lePosVelocity ,
   Acceleration  := lePosAcceleration ,
   Deceleration  := lePosDeceleration ,
   Jerk          := lePosJerk ,
   Direction     := MC_DIRECTION_mcShortestWay ,
   BufferMode    := MC_BUFFER_MODE_mcAborting ,
   Done                =>  bMoveAbs2Done,
   CommandAborted      =>  bMoveAbs2Aborted,
   Error               =>  bMoveAbs2Error
);
//Dwell
TON_2(IN:= bMoveAbs2Done,PT:= T#500ms ,Q=> bDwell2_out);
//Error Signal, Aborted Signal
bPosError       := bMoveAbs1Error OR bMoveAbs2Error;
bPosAborted     := bMoveAbs1Aborted OR bMoveAbs2Aborted;
//Done Signal => PLC CPU
G_bPosDone      := bDwell2_out OR bPosError OR bPosAborted;
```

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

2.6 Summary of This Chapter

In this chapter, you have learned:

- What is Public Label?
- Public Label Setting
- Program Example
- Writing Program
- Operation Check

Point

What is Public Label?	• A public label is a shared label that can be used in both the Motion module and PLC CPU.
Public Label Setting	• Register the public labels from the global labels of the Motion module. • Select whether each label is to be read or written from/to the PLC CPU. • To set the members of a structured data type prepared in the system to the public label, register the public labels by layer of the structured data type. • After setting the public labels in the Motion module, rebuild all the programs and reflect the public labels. • The public labels are registered to the module label on the PLC CPU side.
Program Example	• This chapter described the following program example: a ladder program of the PLC CPU that uses public labels to exchange the positioning start signal and positioning completion signal.
Writing Program	• Write data to the PLC CPU first, and then the Motion module.
Operation Check	• You have checked the operation of the sample program in the video.

The buffer mode executes operations continuously by starting multiple operation FBs of Motion control FBs.
It can be set with the BufferMode input of the Motion control FB.
Up to two FBs can be started simultaneously for each axis and axis group.

(Example) MC_MoveAbsolute

```
MC_MoveAbsolute_1(
    Axis            := Axis0001.AxisRef ,
    Execute         := G_bPositioningReq ,
    ContinuousUpdate:= FALSE ,
    Position        := lePosition1 ,
    Velocity        := lePosVelocity ,
    Acceleration    := lePosAcceleration ,
    Deceleration    := lePosDeceleration ,
    Jerk            := lePosJerk ,
    Direction       := MC_DIRECTION__mcShortestWay ,
    BufferMode      := MC_BUFFER_MODE__mcAborting ,
    Options         := 0 ,//mcAccDec
    Done            => bMoveAbs1Done ,
    CommandAborted  => bMoveAbs1Aborted ,
    Error           => bMoveAbs1Error
);
```

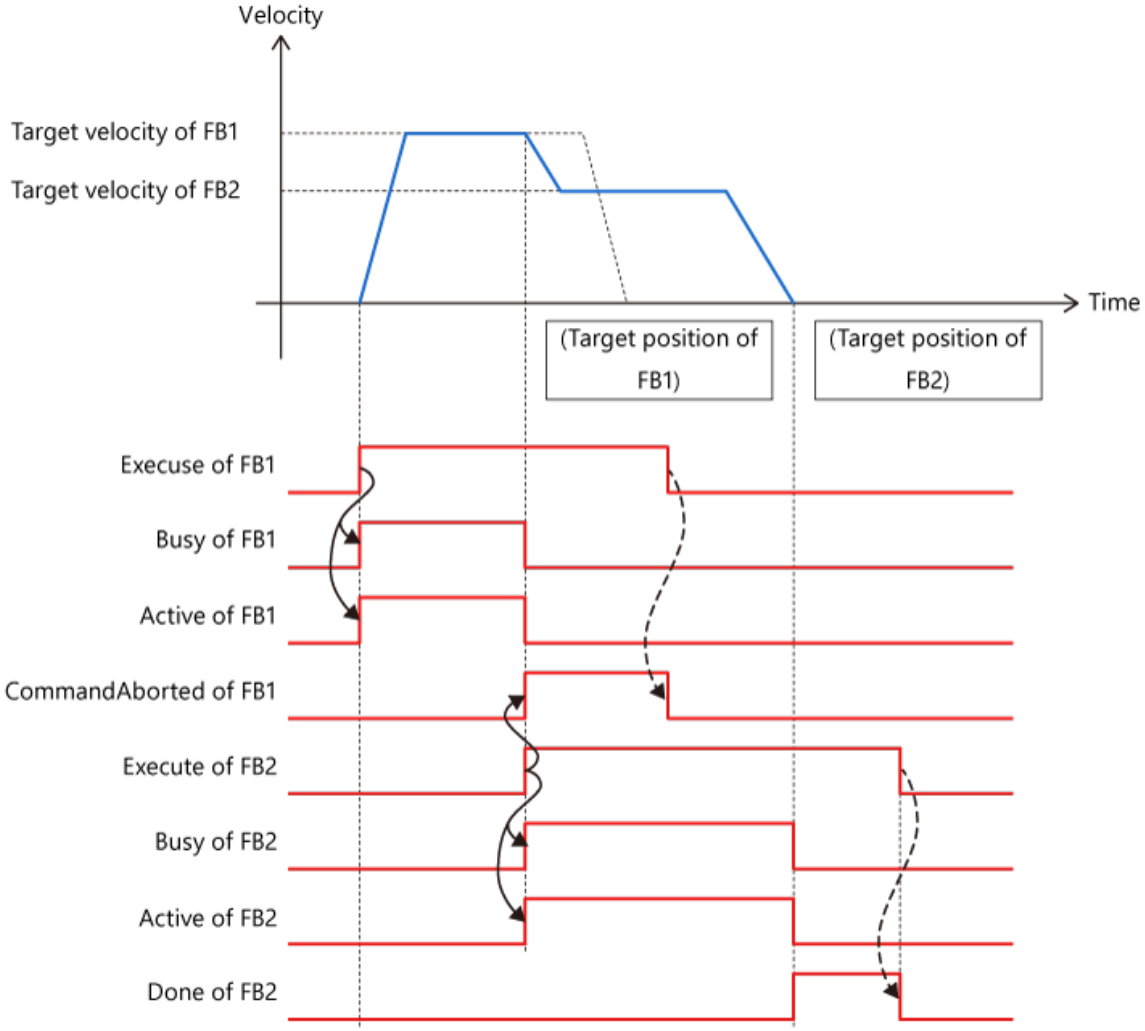
0 or MC_BUFFER_MODE__mcAborting ･･･The FB being executed is interrupted and the next FB is immediately executed.
1 or MC_BUFFER_MODE__mcBuffered ･･･After the operation of the FB being executed is completed, the next FB is executed.
2 or MC_BUFFER_MODE__mcBlendingLow ･･･The lower one of target velocities for the FB being executed and FB to be buffered is set as the switching velocity.
3 or MC_BUFFER_MODE__mcBlendingPrevious ･･･The target velocity of the FB being executed is set as the switching velocity.
4 or MC_BUFFER_MODE__mcBlendingNext ･･･The target velocity of the FB to be buffered is set as the switching velocity.
5 or MC_BUFFER_MODE__mcBlendingHigh ･･･The higher one of the target velocities for the FB being executed and FB to be buffered is set as the switching velocity.

[Point]
For Direction and BufferMode input, specify numbers or ENUM enumerators starting with MC_BUFFER_MODE and MC_DIRECTION.
For details of the ENUM enumerators, refer to the following manual.

MELSEC iQ-R Programming Manual (Motion Control Function Blocks)
2 VARIABLES AND MOTION CONTROL FB
2.2 List of Enumerators

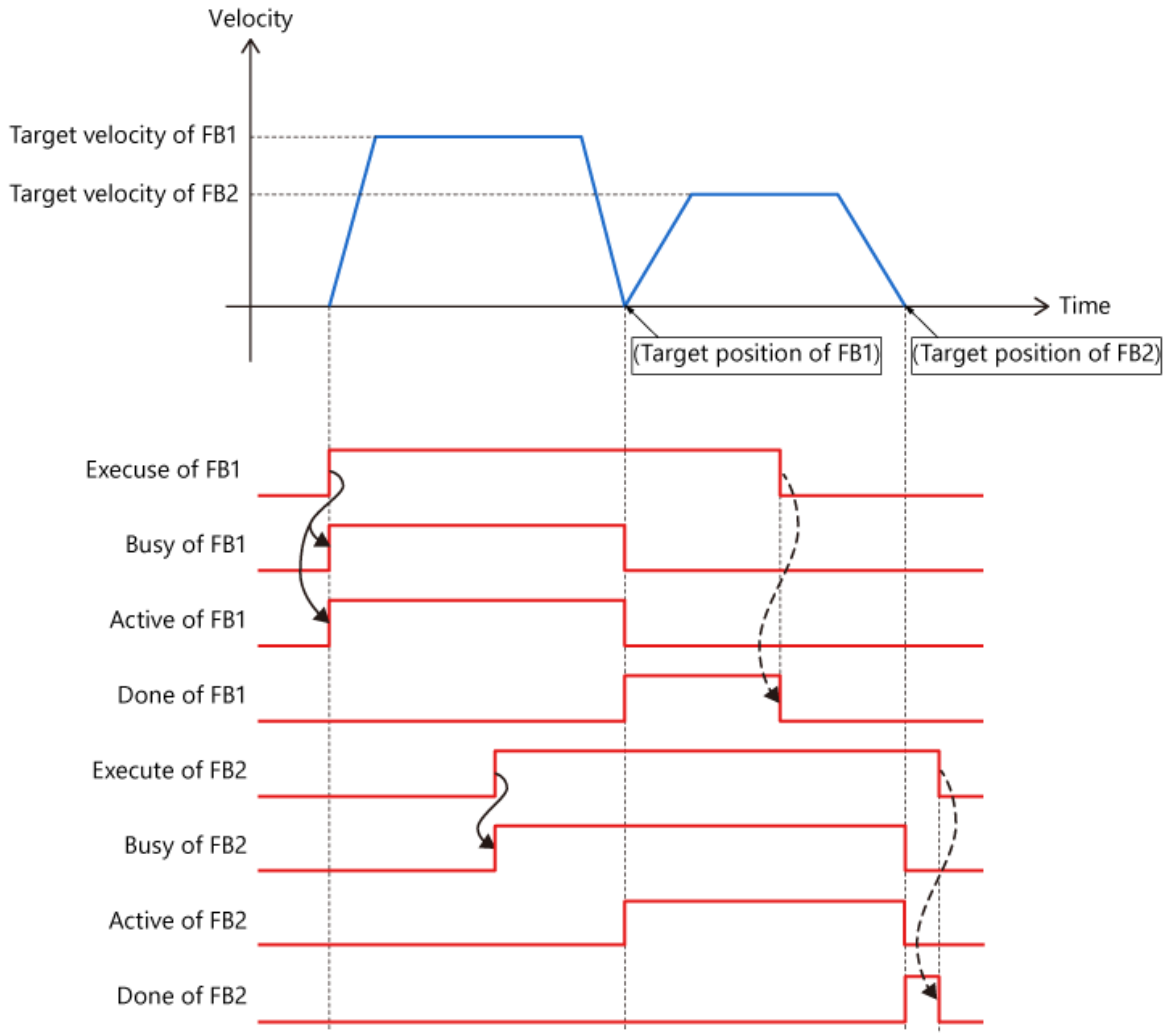
3.1 Aborting

The following diagram shows the operation when BufferMode is set to 0:
mcAborting. The FB being executed is interrupted and the next FB is immediately executed.



3.2 Buffered

The following diagram shows the operation when BufferMode is set to 1:mcBuffered.
When the operation of the FB being executed is completed, the next FB is executed.

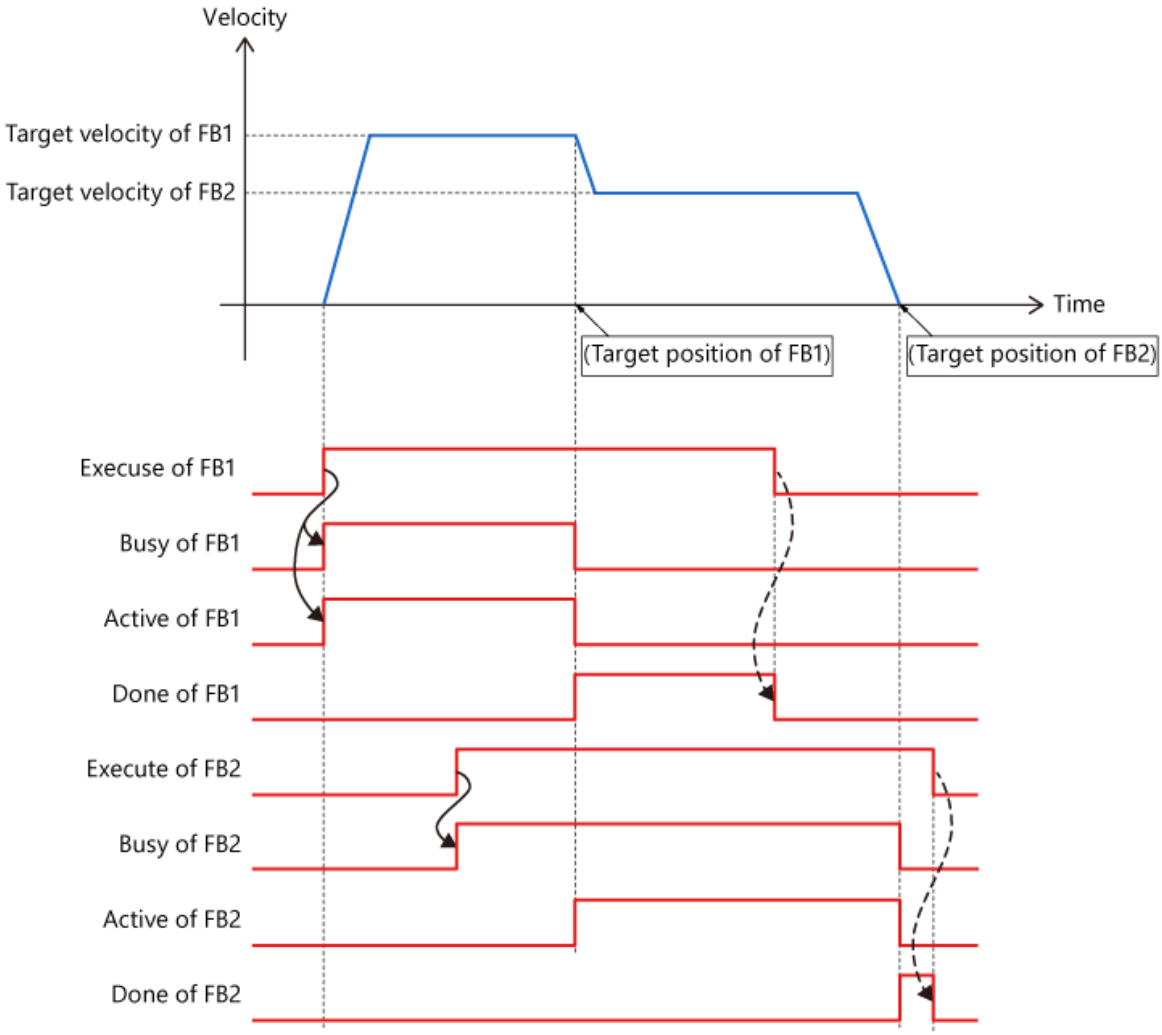


3.3 Blending

When BufferMode is set to mcBlending***, the next FB is continuously executed after the target position of the FB being executed is reached.
In the following description, the FB to be executed first is FB1, and FB to be buffered is FB2.

(1) BlendingPrevious

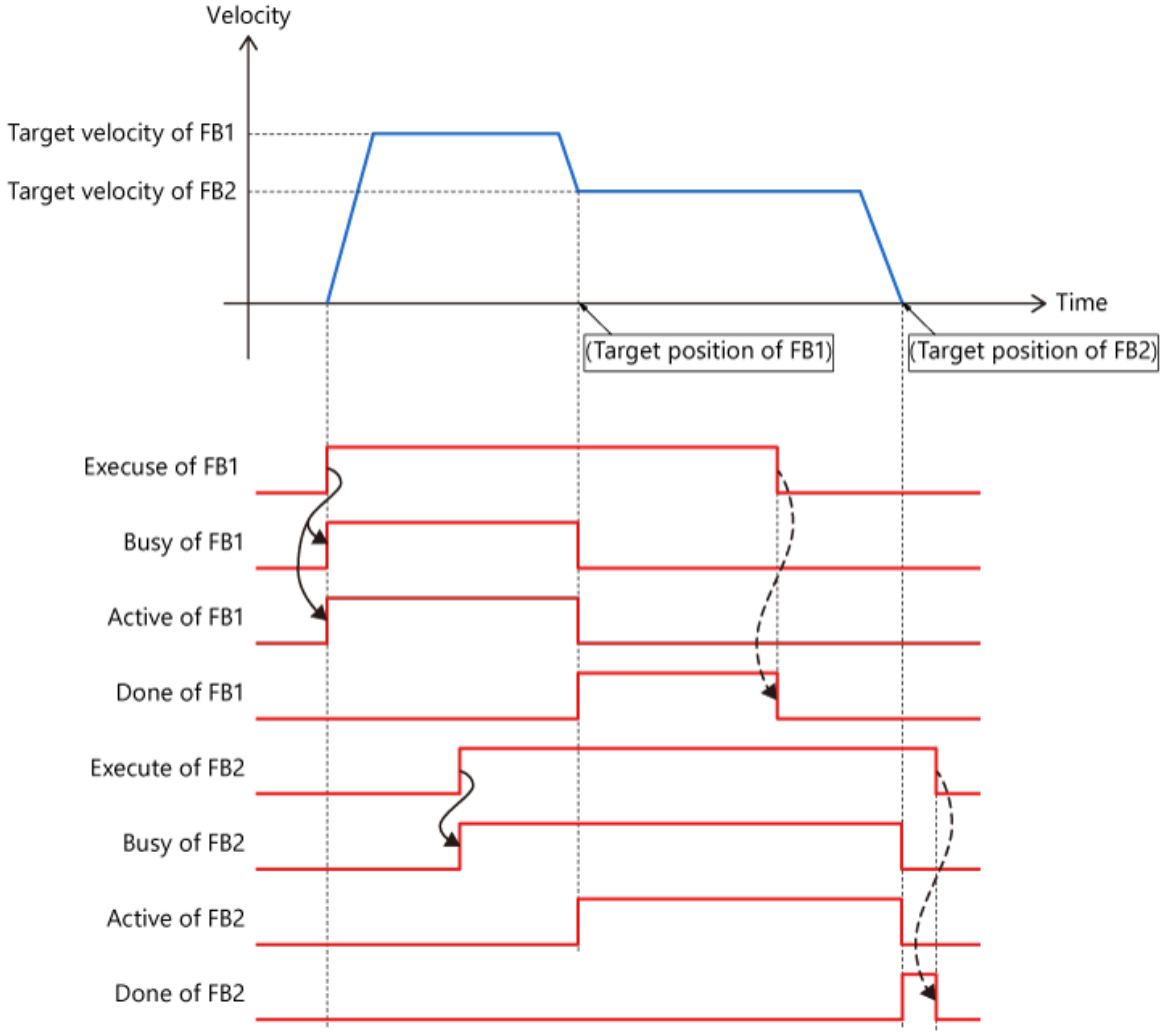
The following diagram shows the operation when BufferMode is set to 3: mcBlendingPrevious.
The operation is performed at the target velocity of FB1 until the target position of FB1.
When the operation is switched to FB2, the velocity is changed to the target velocity of FB2, and move to the target position of FB2.



(2) BlendingNext

The following diagram shows the operation when BufferMode is set to 4: mcBlendingNext.

The velocity changes to the target velocity of FB2 when the operation reaches the target position of FB1.



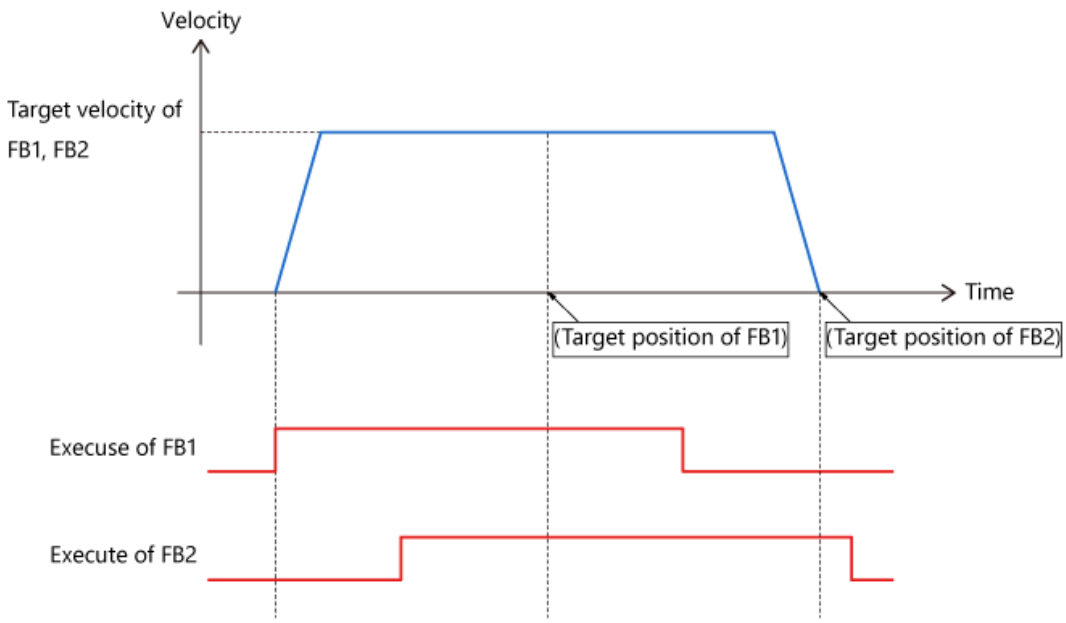
(3) BlendingLow, BlendingHigh

The operation when BufferMode is set to 2: mcBlendingLow or 5: mcBlendingHigh varies depending on which target velocities of FB1 and FB2 is larger.

Setting value	Target velocity of FB1 > Target velocity of FB2	Target velocity of FB1 < Target velocity of MFB2
2 : mcBlendingLow	Same operation as BlendingPrevious	Same operation as BlendingNext
5 : mcBlendingHigh	Same operation as BlendingNext	Same operation as BlendingPrevious

[Point]

The following diagram shows the velocity waveform for BlendingPrevious, BlendingNext, BlendingHigh, and BlendingLow when the target velocity of FB1 and FB2 is the same.



3.4 Program Example

(1) Sample program operation

This chapter uses the sample program used in Chapter 2.

Check the difference in buffer mode operation in a program that starts with X25.

Item	FB1 (MC_MoveAbsolute)	FB2 (MC_MoveAbsolute)
Positioning address	75000.0[μm]	150000.0[μm]
Velocity	50000.0[μm/s]	25000.0[μm/s]
Acceleration, deceleration	100000.0[μm/s^2]	50000.0[μm/s^2]
Jerk	200000.0[μm/s^3]	100000.0[μm/s^3]

(2) Program of the PLC CPU
MAIN (ladder, scan program)

The rising edge of the continuous positioning control start (X25) is retained with G_bContPosCMD, and sent to the Motion module as the start condition of the continuous positioning control.
An interlock is set to prevent positioning control from being started while another program is running.
Upon receiving that the Motion module turned on the continuous home position return completion signal, G_bContPosCMD is reset at the rising edge of that signal.

(3) Motion module program
ContinuousPositioning (normal execution type)

```
1  //-----Continuous Positioning Operation-----
2  //Initial Value Setting, Operation Start Request
3  IF G_bContPosCMD THEN
4      lePosition1         := 75000.0;
5      lePosVelocity1      := 50000.0;
6      lePosAcceleration1  := 100000.0;
7      lePosDeceleration1  := 100000.0;
8      lePosJerk1          := 200000.0;
9      lePosition2         := 150000.0;
10     lePosVelocity2      := 25000.0;
11     lePosAcceleration2  := 50000.0;
12     lePosDeceleration2  := 50000.0;
13     lePosJerk2          := 100000.0;
14     G_bContPosReq := TRUE;
15 ELSE
16     G_bContPosReq := FALSE;
17 END_IF;
18
19 //Positioning1
20 MC_MoveAbsolute_1(
21     Axis        := Axis0001.AxisRef ,
22     Execute     := G_bContPosReq ,
23     Position    := lePosition1 ,
24     Velocity    := lePosVelocity1 ,
25     Acceleration:= lePosAcceleration1 ,
26     Deceleration:= lePosDeceleration1 ,
27     Jerk        := lePosJerk1 ,
28     Direction   := MC_DIRECTION_mcShortestWay ,
29     Done            => bMoveAbs1Done ,
30     Active          => bMoveAbs1Active ,
31     CommandAborted  => bMoveAbs1Aborted ,
32     Error           => bMoveAbs1Error
33 );
34 //Positioning2
35 MC_MoveAbsolute_2(
36     Axis        := Axis0001.AxisRef ,
37     Execute     := bMoveAbs1Active ,
38     Position    := lePosition2 ,
39     Velocity    := lePosVelocity2 ,
40     Acceleration:= lePosAcceleration2 ,
41     Deceleration:= lePosDeceleration2 ,
42     Jerk        := lePosJerk2 ,
43     Direction   := MC_DIRECTION_mcShortestWay ,
44     BufferMode  := MC_BUFFER_MODE_mcBuffered,
45     Done            => bMoveAbs2Done ,
46     CommandAborted  => bMoveAbs2Aborted,
47     Error           => bMoveAbs2Error
48 );
49 //Dwell
50 SET(bMoveAbs2Done,bDwell_In);
51 TON_1(IN:= bDwell_In ,PT:= T#100ms ,Q=> bDwell_out );
52
53 //Error Signal, Aborted Signal
54 bError   := bMoveAbs1Error OR bMoveAbs2Error;
55 bAborted := bMoveAbs1Aborted OR bMoveAbs2Aborted;
56
57 //Done Signal => PLC CPU
58 G_bContPosDone := bDwell_out OR bError OR bAborted;
59 //Reset Dwell_In
60 RST(G_bContPosDone,bDwell_In);
61
```

FB1

FB2

Receives the continuous positioning control start signal (G_bContPosCMD) from the PLC CPU.
The data required for the positioning is stored to the label, and the continuous positioning control start request (G_bContPositioningReq) is turned on.

Turns off G_bContPositioningReq after G_bContPosCMD is turned off.

FB1 (MC_MoveAbsolute_1) executes positioning.

FB2 (MC_MoveAbsolute_2) is started by the Active output of FB1(MC_MoveAbsolute_1). Then, FB2 is buffered while FB1 is running.

Change the BufferMode input of FB2 (MC_MoveAbsolute_2) and check the operation of the buffer mode.

The dwell that triggers the on-delay timer is started by the Done output of FB2 (MC_MoveAbsolute_2).

Returns the execution completion signal to the PLC CPU after the dwell time is elapsed or either Error output or CommanndAborted output of MC_MoveAbsolute is turned on.
At the same time, the input of the on-delay time is reset.

Click the play button at the lower left of the window.

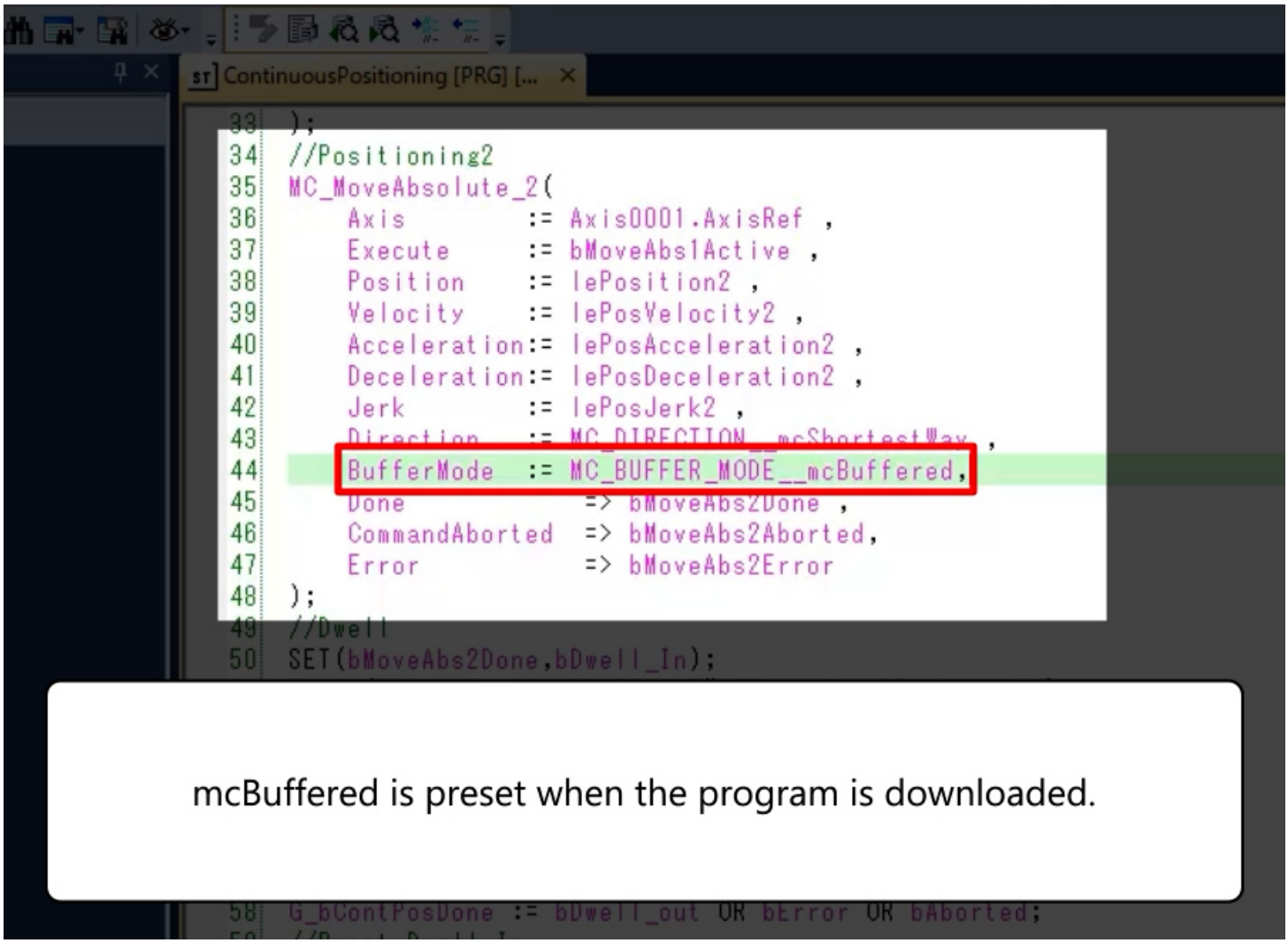
3.5 Operation Check

ST ContinuousPositioning [PRG] [...

```
33  );
34  //Positioning2
35  MC_MoveAbsolute_2(
36      Axis         := Axis0001.AxisRef ,
37      Execute      := bMoveAbs1Active ,
38      Position     := lePosition2 ,
39      Velocity     := lePosVelocity2 ,
40      Acceleration:= lePosAcceleration2 ,
41      Deceleration:= lePosDeceleration2 ,
42      Jerk         := lePosJerk2 ,
43      Direction    := MC_DIRECTION__mcShortestWay ,
44      BufferMode   := MC_BUFFER_MODE__mcBuffered,
45      Done             => bMoveAbs2Done ,
46      CommandAborted   => bMoveAbs2Aborted,
47      Error            => bMoveAbs2Error
48  );
49  //Dwell
50  SET(bMoveAbs2Done,bDwell_In);
```

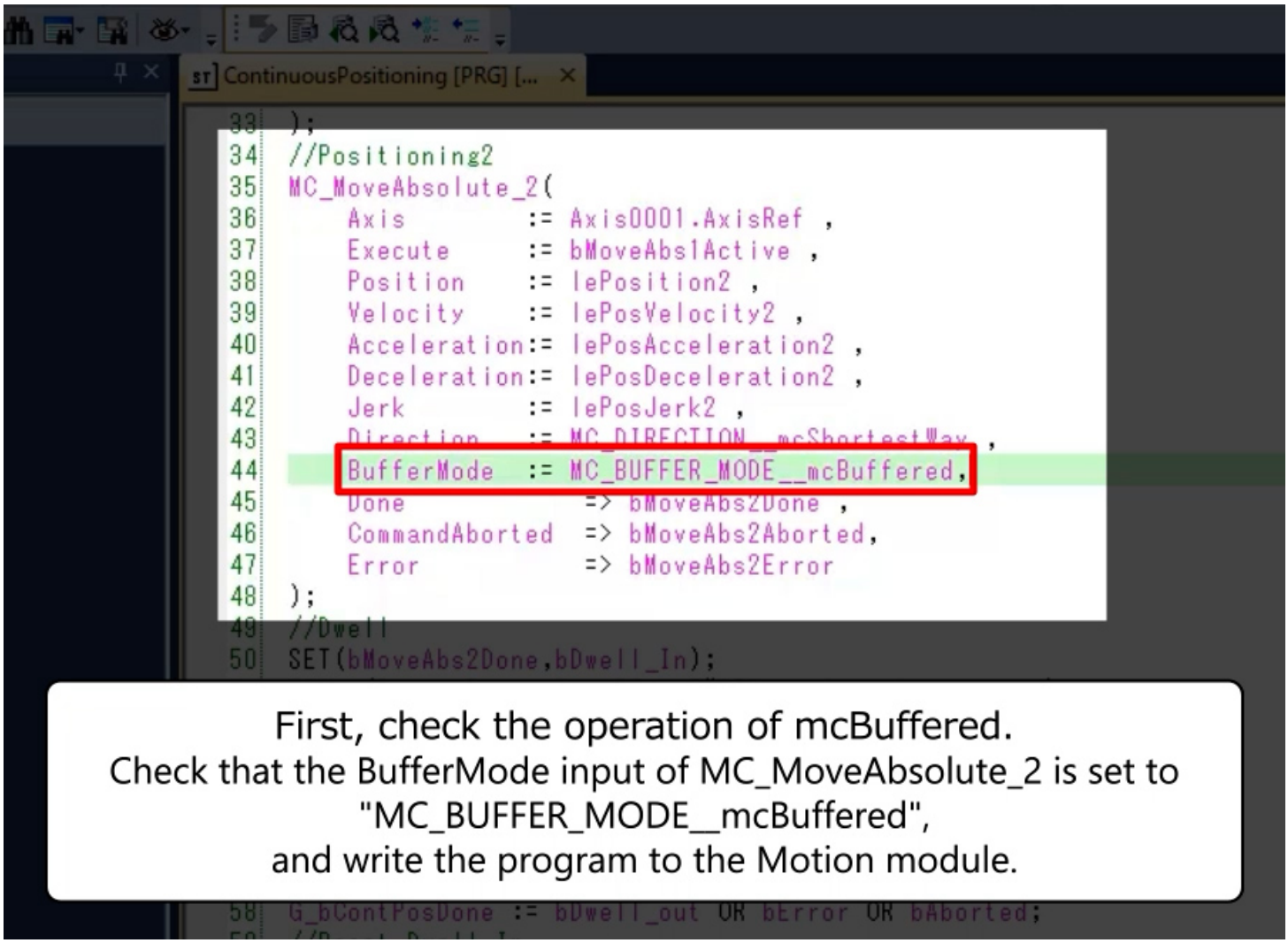
Check the buffer mode operation.
Open "LinearInterpolation" in the sample program and change the BufferMode input of MC_MoveAbsolute_2 to check the operation of the buffer mode.

```
58  G_bContPosDone := bDwell_out OR bError OR bAborted;
```

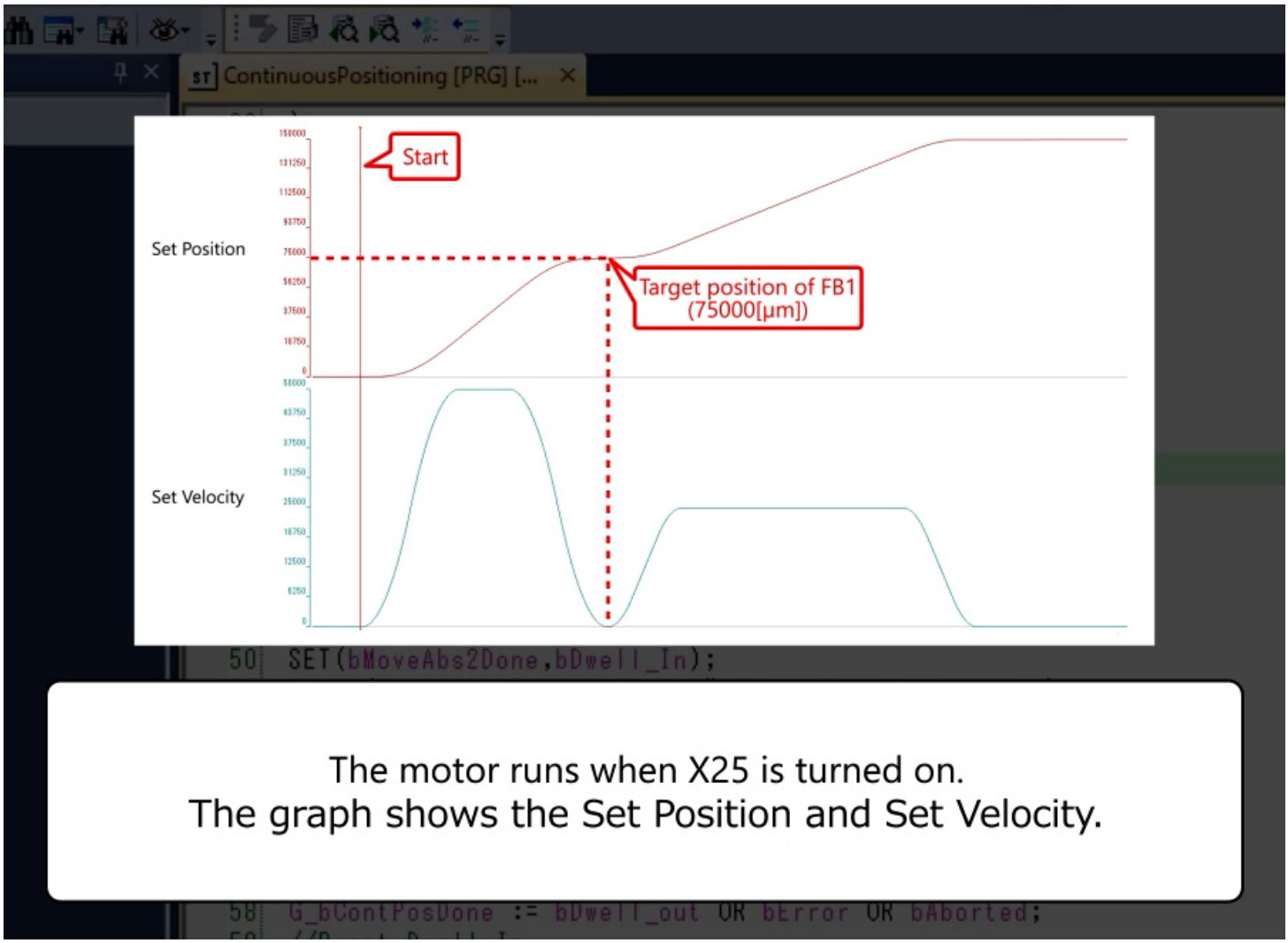


mcBuffered is preset when the program is downloaded.

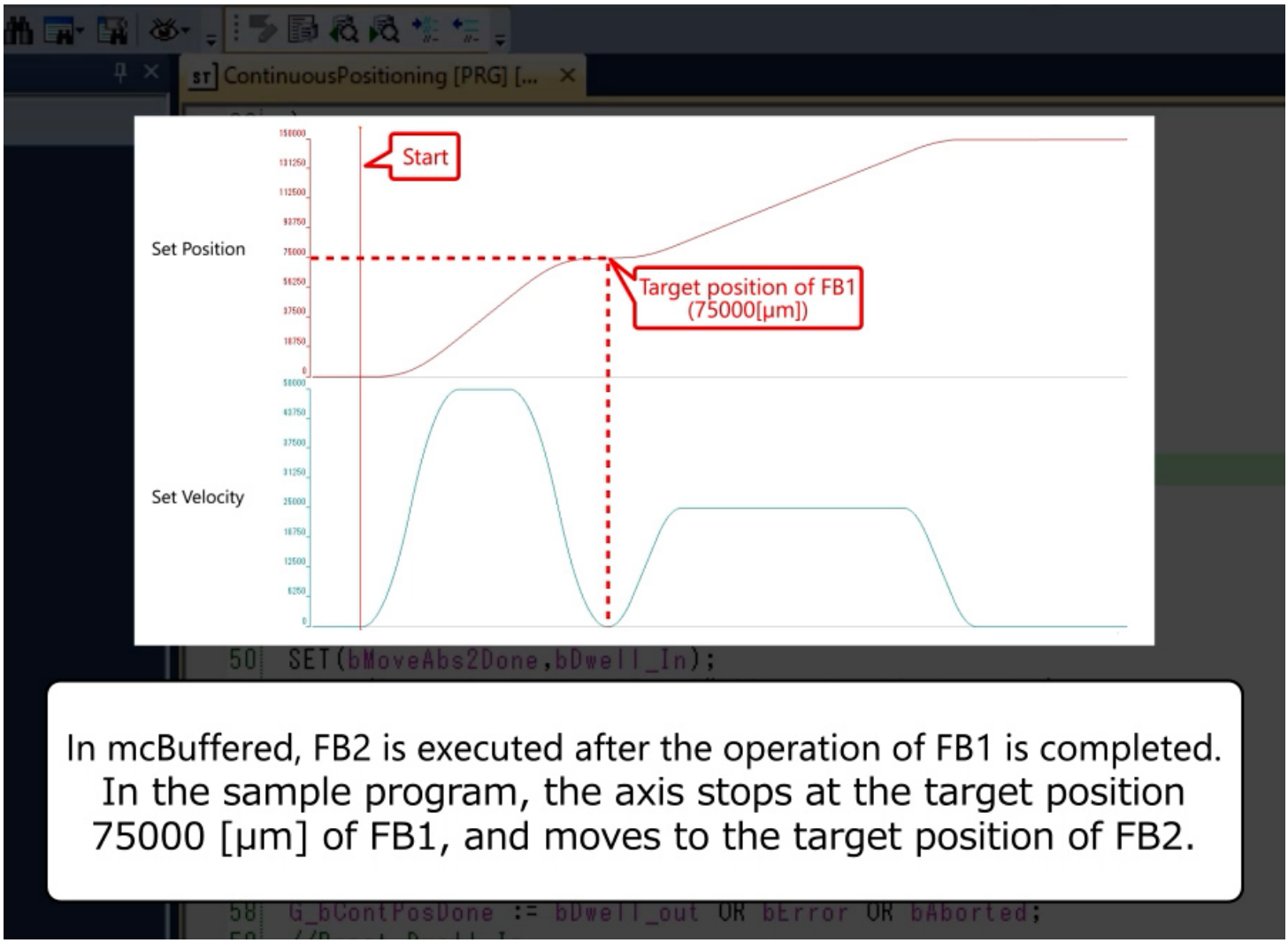
3.5 Operation Check



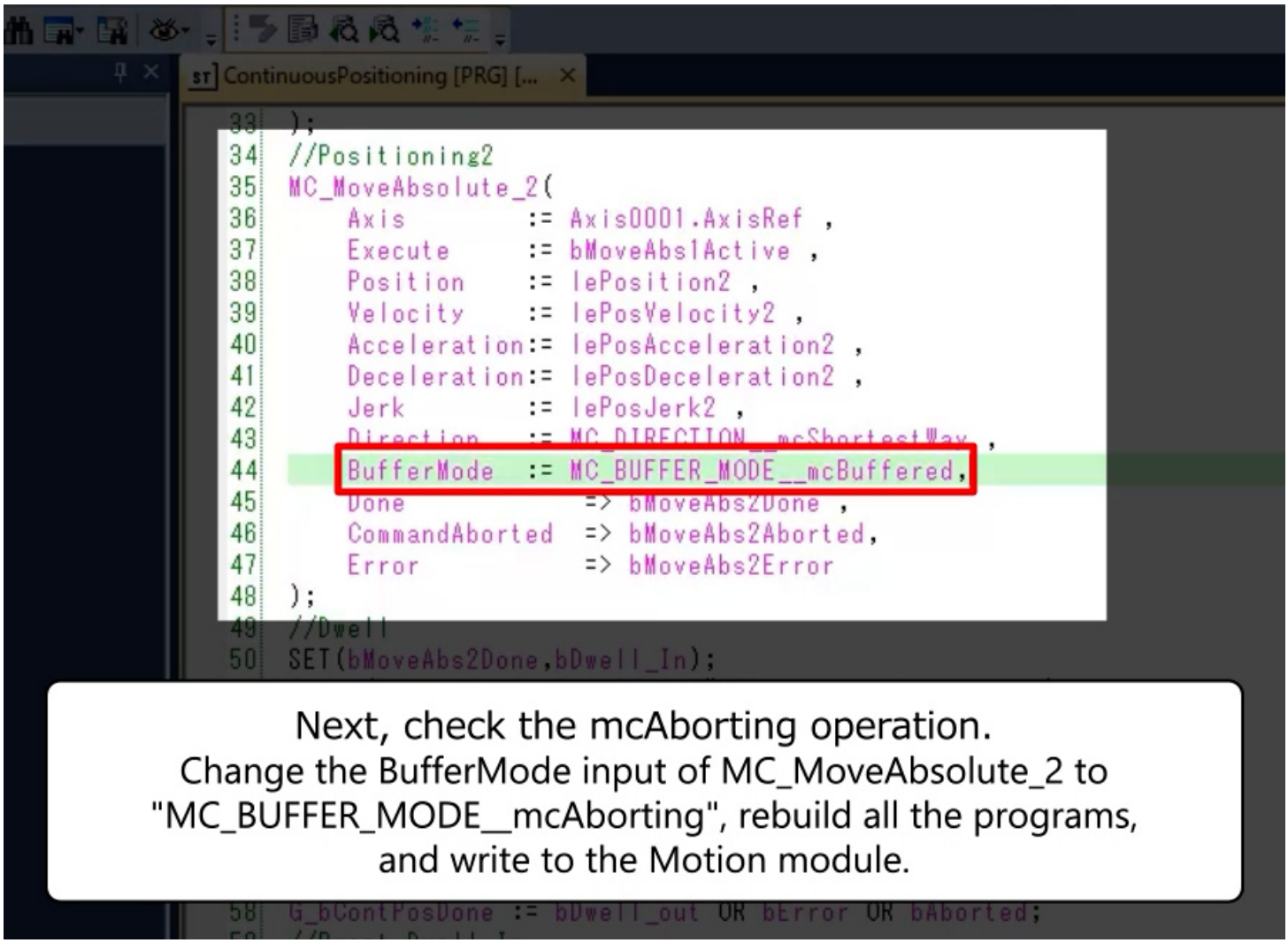
First, check the operation of mcBuffered.
Check that the BufferMode input of MC_MoveAbsolute_2 is set to "MC_BUFFER_MODE__mcBuffered",
and write the program to the Motion module.



The motor runs when X25 is turned on.
The graph shows the Set Position and Set Velocity.

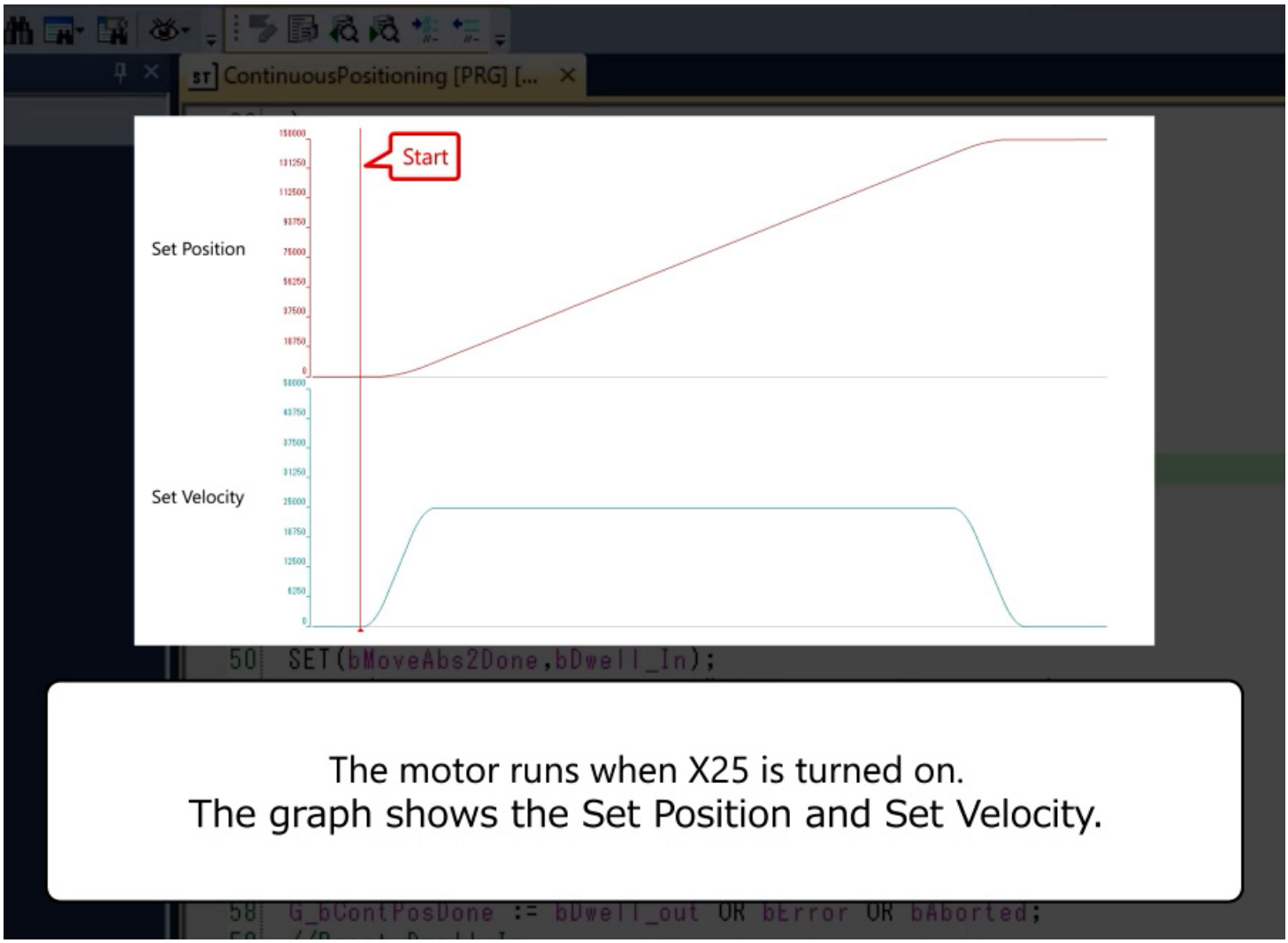


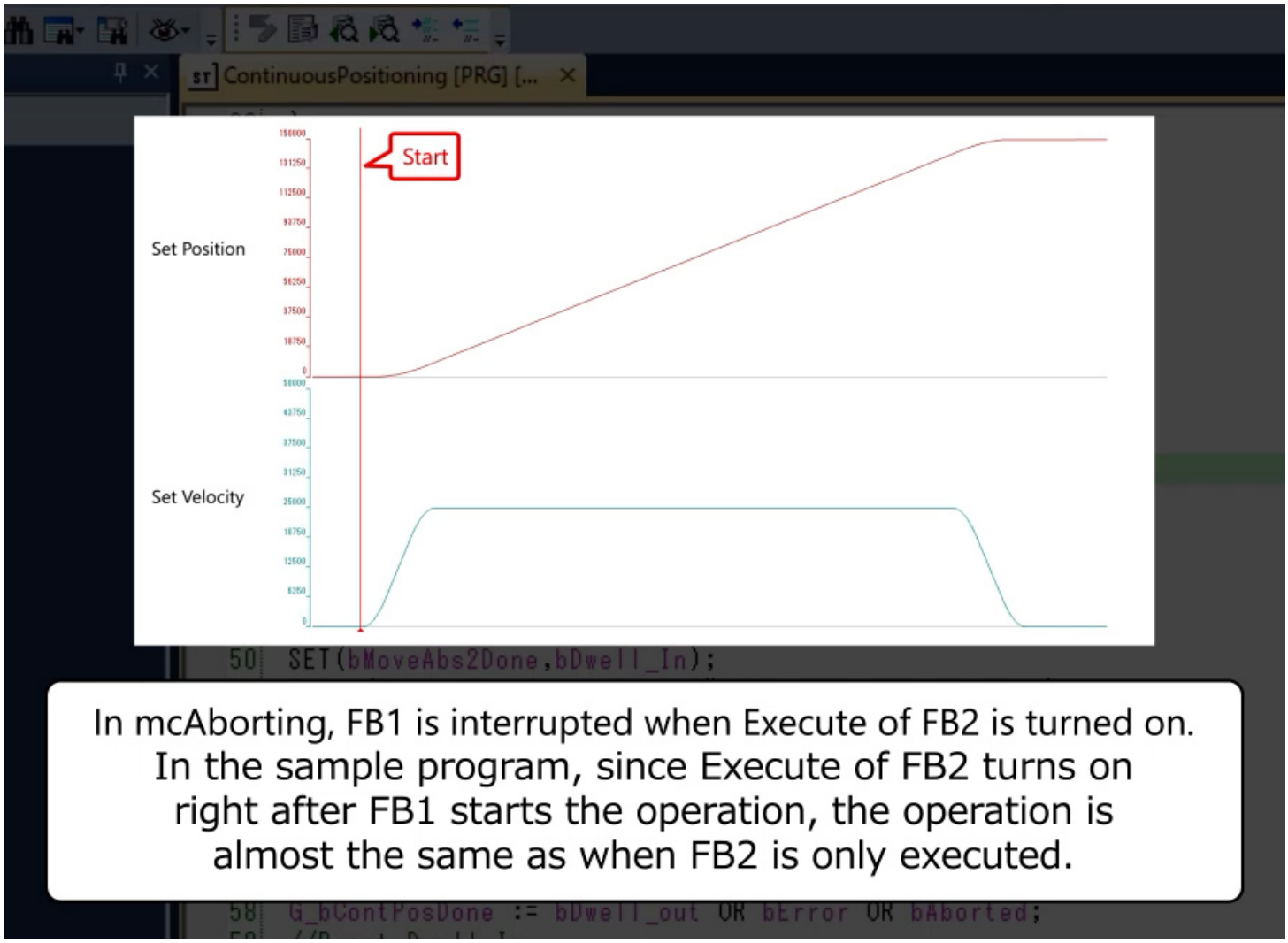
In mcBuffered, FB2 is executed after the operation of FB1 is completed. In the sample program, the axis stops at the target position 75000 [μm] of FB1, and moves to the target position of FB2.



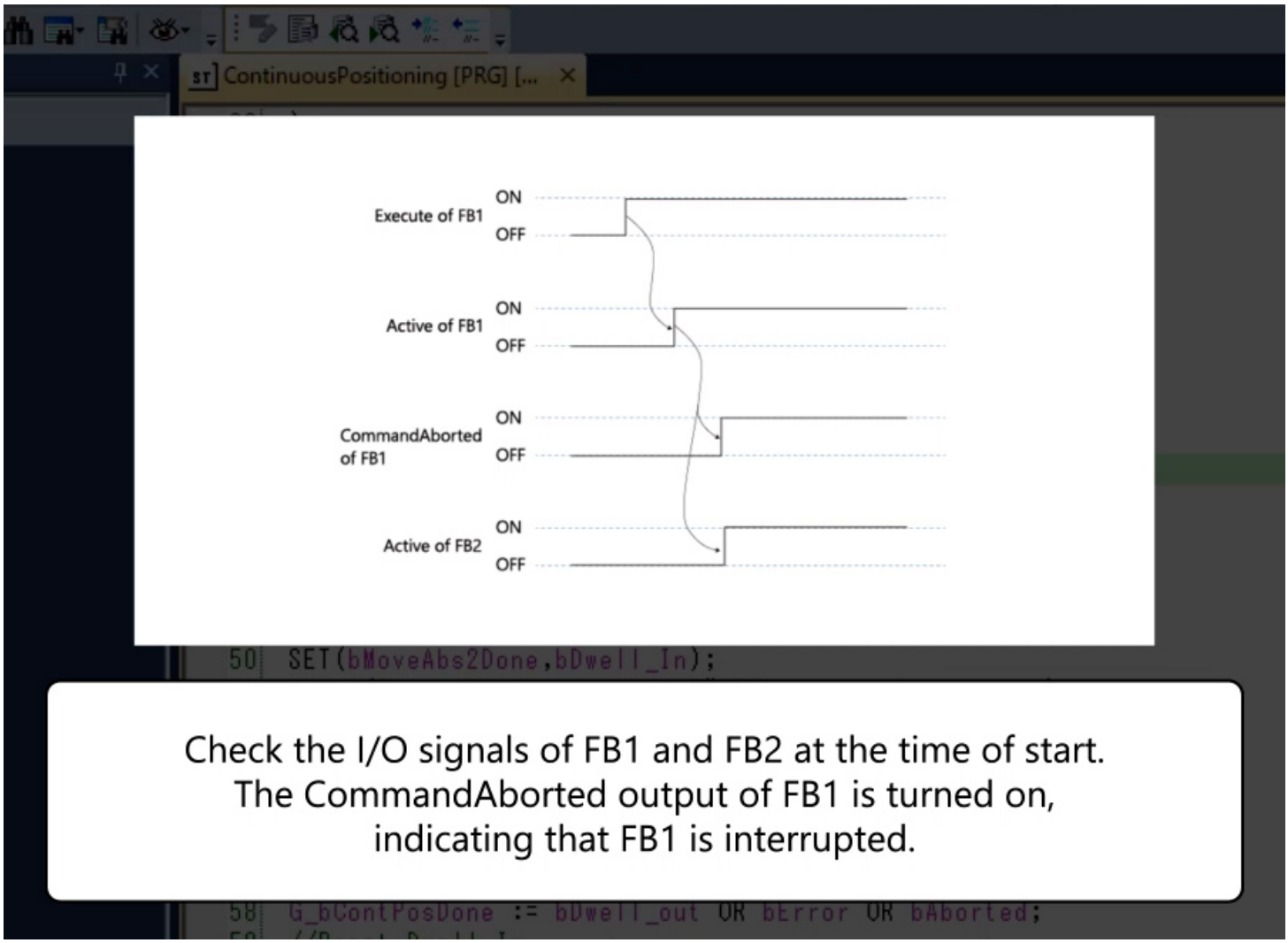
Next, check the mcAborting operation.
Change the BufferMode input of MC_MoveAbsolute_2 to "MC_BUFFER_MODE__mcAborting", rebuild all the programs, and write to the Motion module.

3.5 Operation Check





In mcAborting, FB1 is interrupted when Execute of FB2 is turned on. In the sample program, since Execute of FB2 turns on right after FB1 starts the operation, the operation is almost the same as when FB2 is only executed.

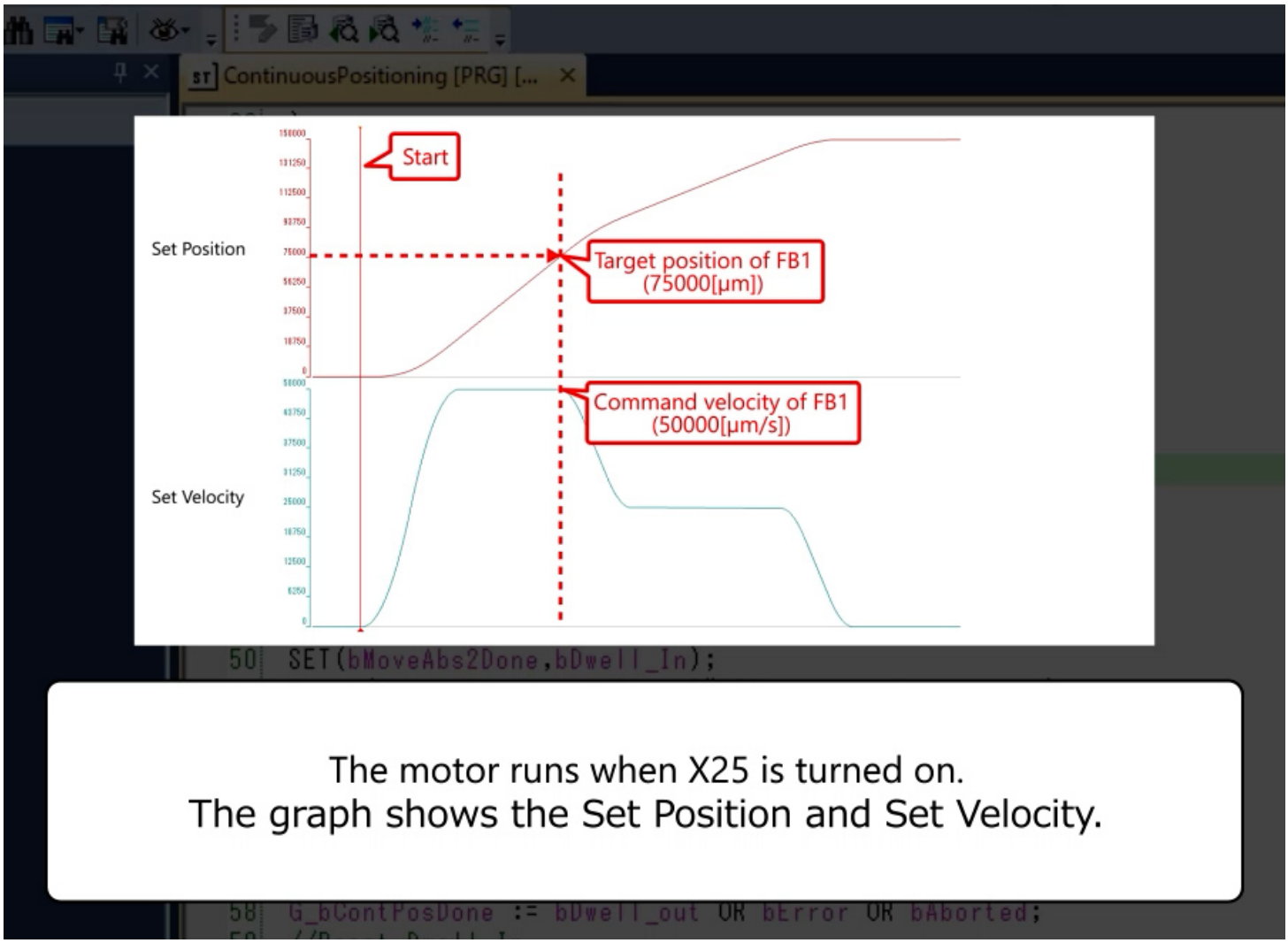

ST ContinuousPositioning [PRG] [...
Execute of FB1
ON
OFF
Active of FB1
ON
OFF
CommandAborted of FB1
ON
OFF
Active of FB2
ON
OFF
50 SET(bMoveAbs2Done,bDwell_In);
Check the I/O signals of FB1 and FB2 at the time of start.
The CommandAborted output of FB1 is turned on,
indicating that FB1 is interrupted.
58 G_bContPosDone := bDwell_out OR bError OR bAborted;

ST ContinuousPositioning [PRG] [...

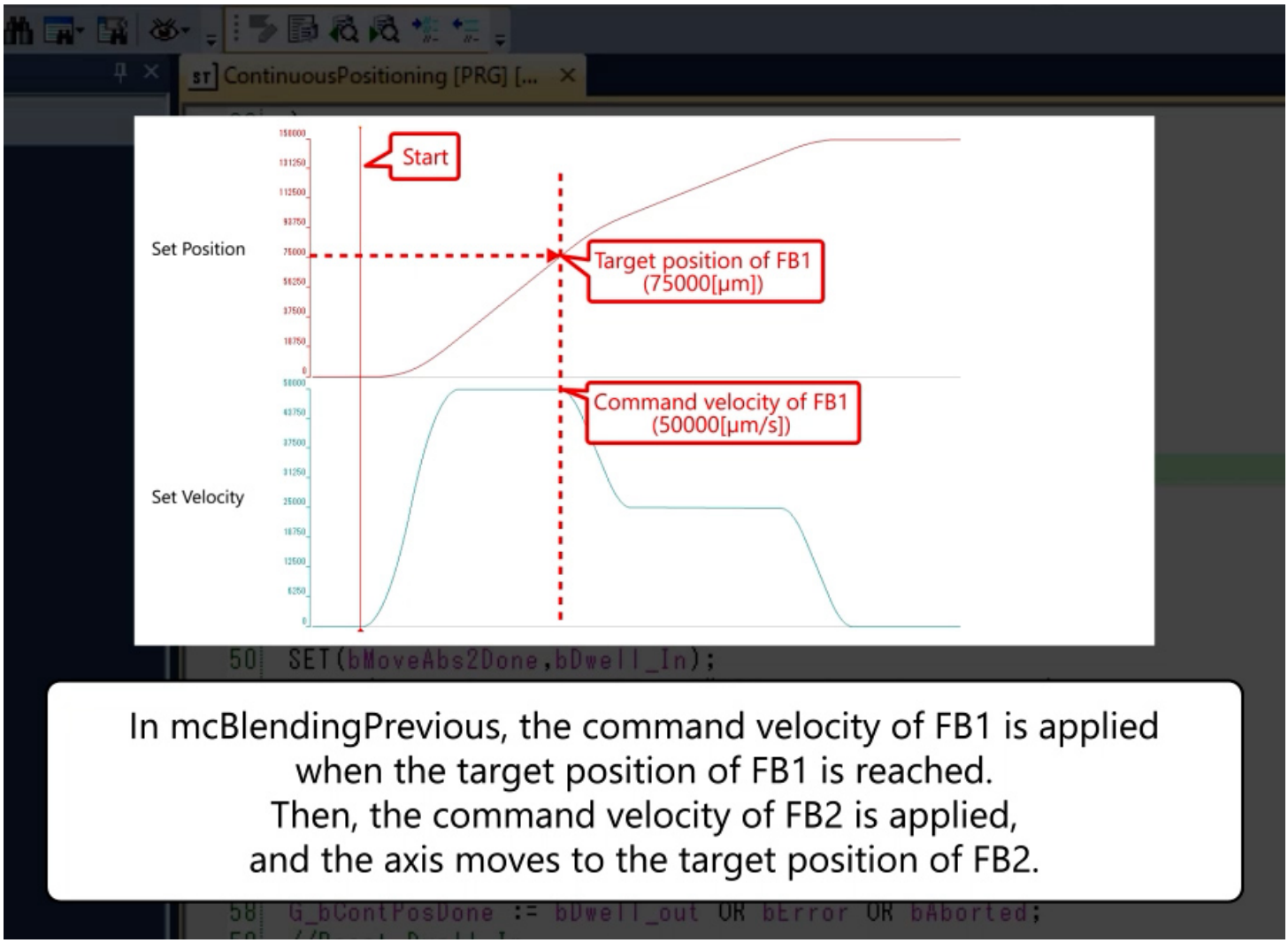
```
33  );
34  //Positioning2
35  MC_MoveAbsolute_2(
36      Axis         := Axis0001.AxisRef ,
37      Execute      := bMoveAbs1Active ,
38      Position     := lePosition2 ,
39      Velocity     := lePosVelocity2 ,
40      Acceleration:= lePosAcceleration2 ,
41      Deceleration:= lePosDeceleration2 ,
42      Jerk         := lePosJerk2 ,
43      Direction    := MC_DIRECTION__mcShortestWay ,
44      BufferMode   := MC_BUFFER_MODE__mcBlendingPrevious,
45      Done              => bMoveAbs2Done ,
46      CommandAborted    => bMoveAbs2Aborted,
47      Error             => bMoveAbs2Error
48  );
49  //Dwell
50  SET(bMoveAbs2Done,bDwell_In);
```

Next, check the mcBlendingPrevious operation.
Change the BufferMode input of MC_MoveAbsolute_2 to "MC_BUFFER_MODE__mcBlendingPrevious", rebuild all the programs, and write to the Motion module.

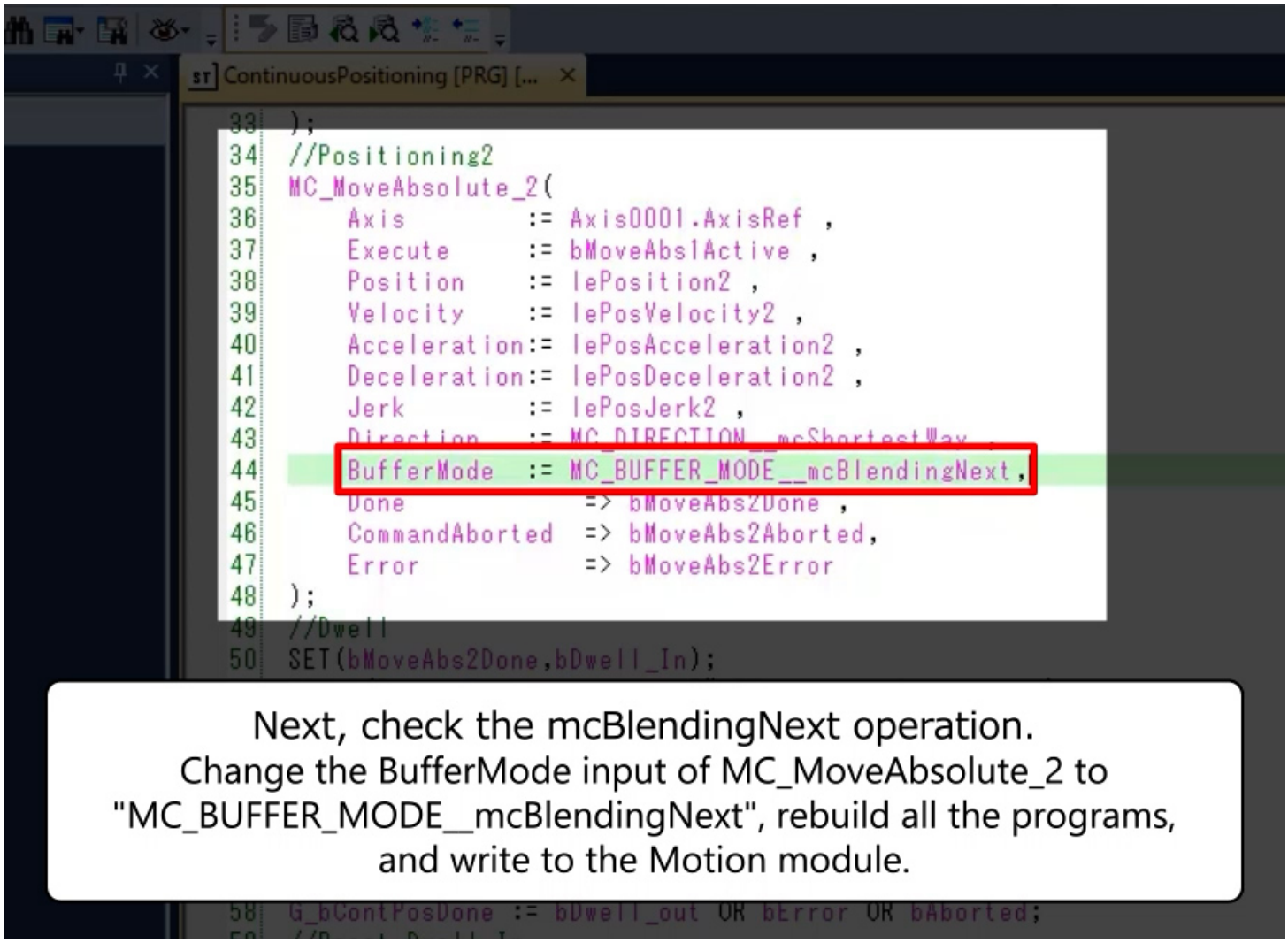
```
58  G_bContPosDone := bDwell_out OR bError OR bAborted;
```



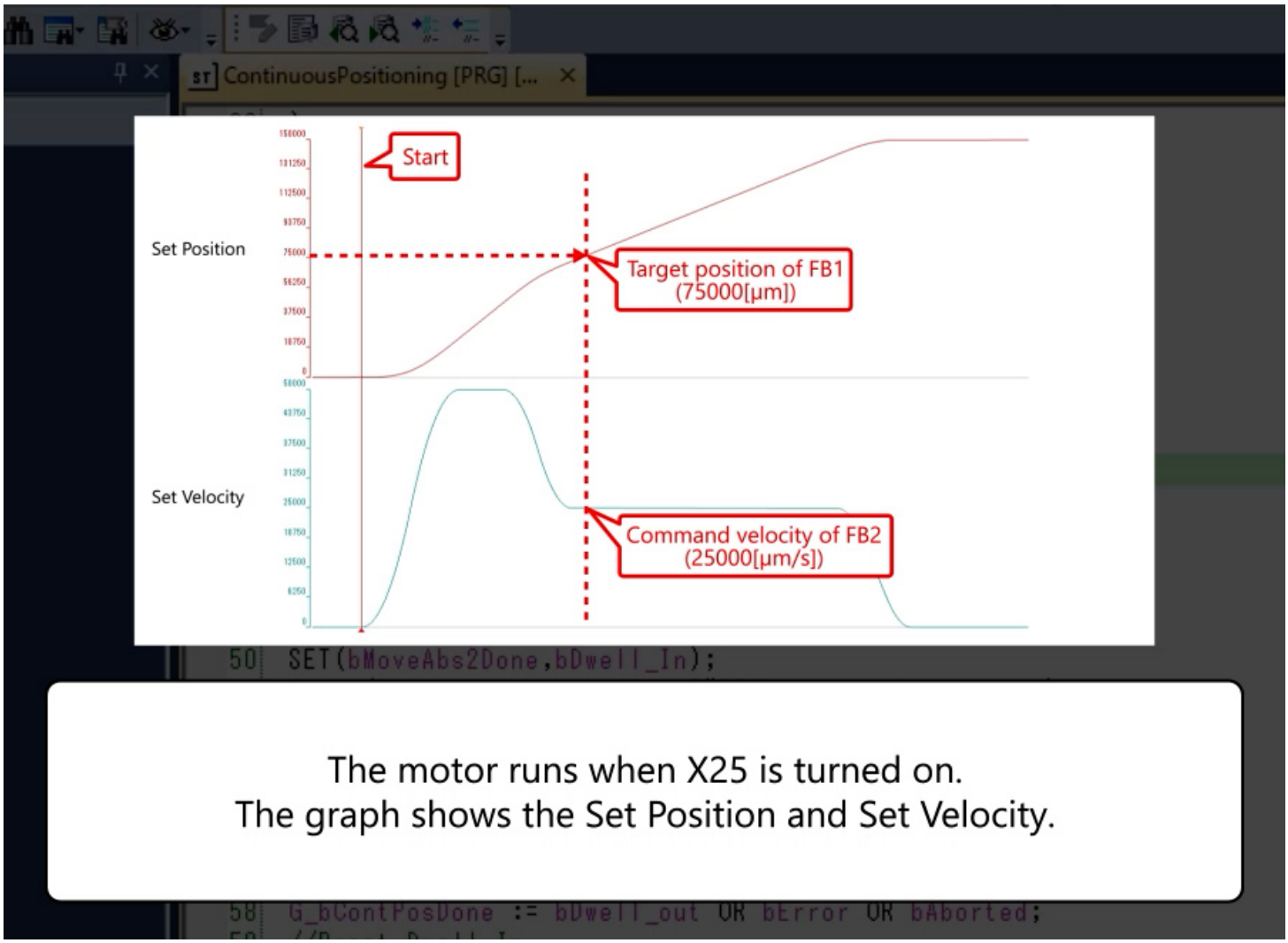
The motor runs when X25 is turned on.
The graph shows the Set Position and Set Velocity.



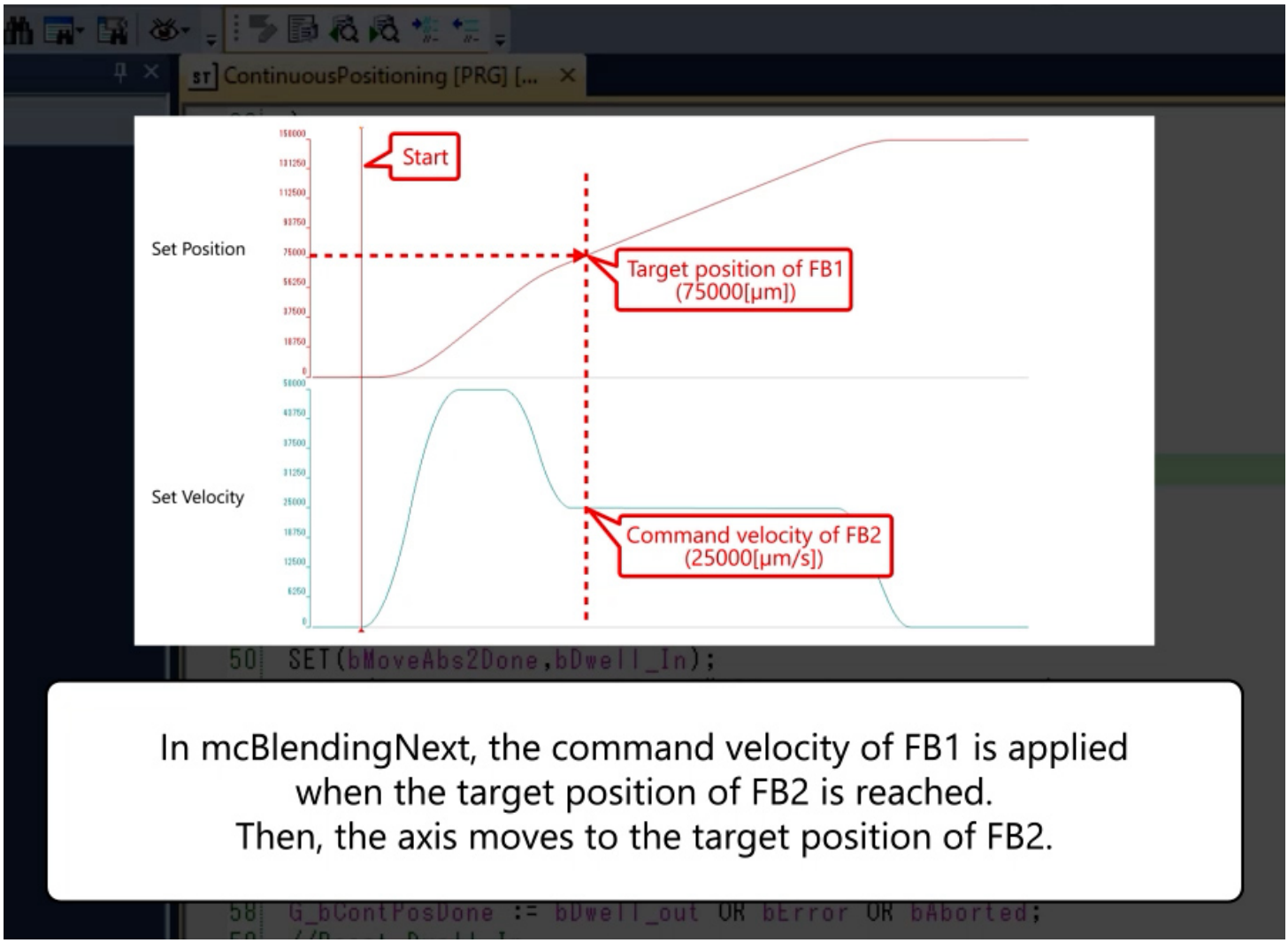
In mcBlendingPrevious, the command velocity of FB1 is applied when the target position of FB1 is reached.
Then, the command velocity of FB2 is applied, and the axis moves to the target position of FB2.


ST ContinuousPositioning [PRG] [...
33);
34 //Positioning2
35 MC_MoveAbsolute_2(
36 Axis := Axis0001.AxisRef ,
37 Execute := bMoveAbs1Active ,
38 Position := lePosition2 ,
39 Velocity := lePosVelocity2 ,
40 Acceleration:= lePosAcceleration2 ,
41 Deceleration:= lePosDeceleration2 ,
42 Jerk := lePosJerk2 ,
43 Direction := MC_DIRECTION__mcShortestWay ,
44 BufferMode := MC_BUFFER_MODE__mcBlendingNext,
45 Done => bMoveAbs2Done ,
46 CommandAborted => bMoveAbs2Aborted,
47 Error => bMoveAbs2Error
48);
49 //Dwell
50 SET(bMoveAbs2Done,bDwell_In);
Next, check the mcBlendingNext operation.
Change the BufferMode input of MC_MoveAbsolute_2 to
"MC_BUFFER_MODE__mcBlendingNext", rebuild all the programs,
and write to the Motion module.

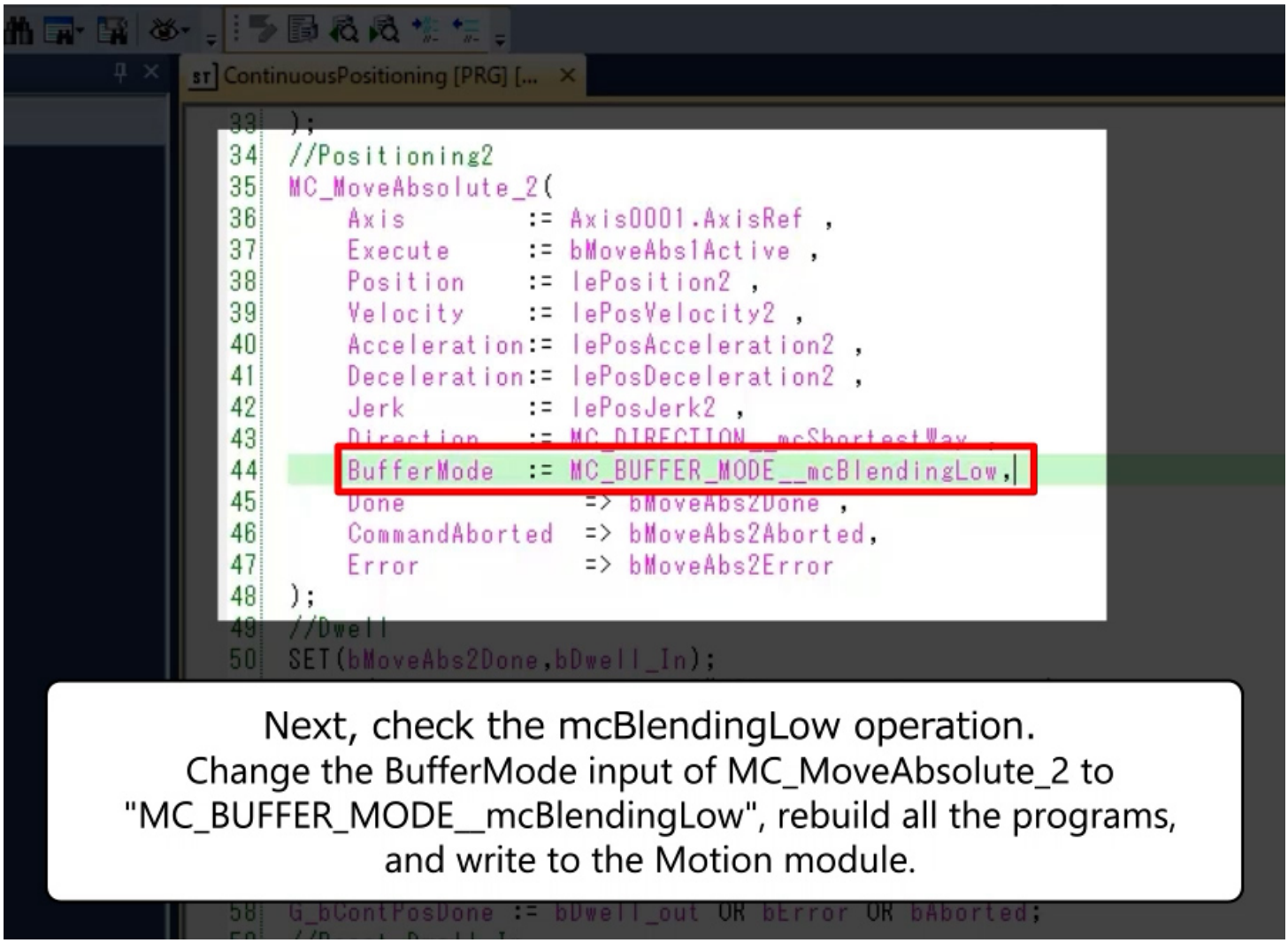
3.5 Operation Check



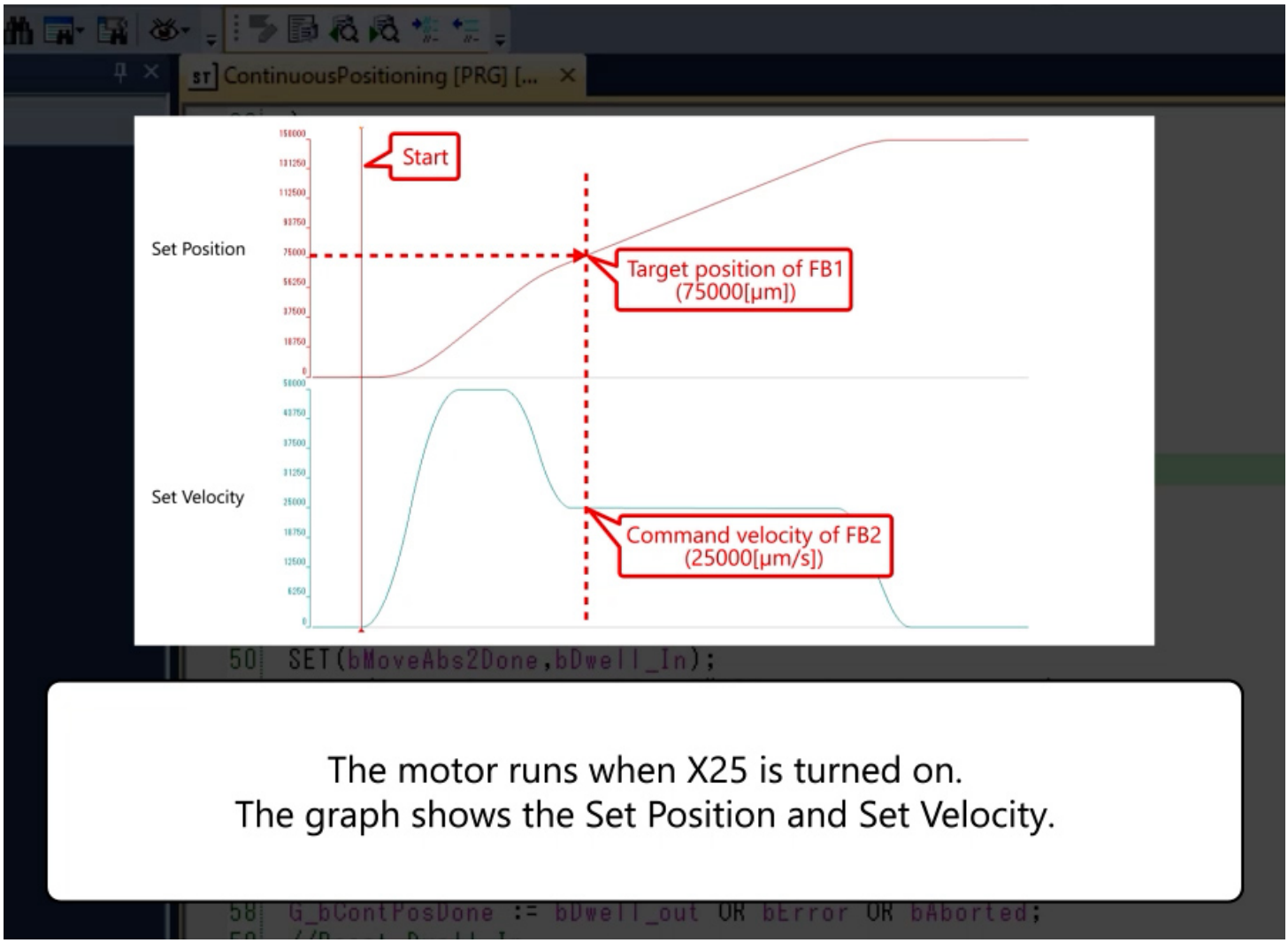
The motor runs when X25 is turned on.
The graph shows the Set Position and Set Velocity.



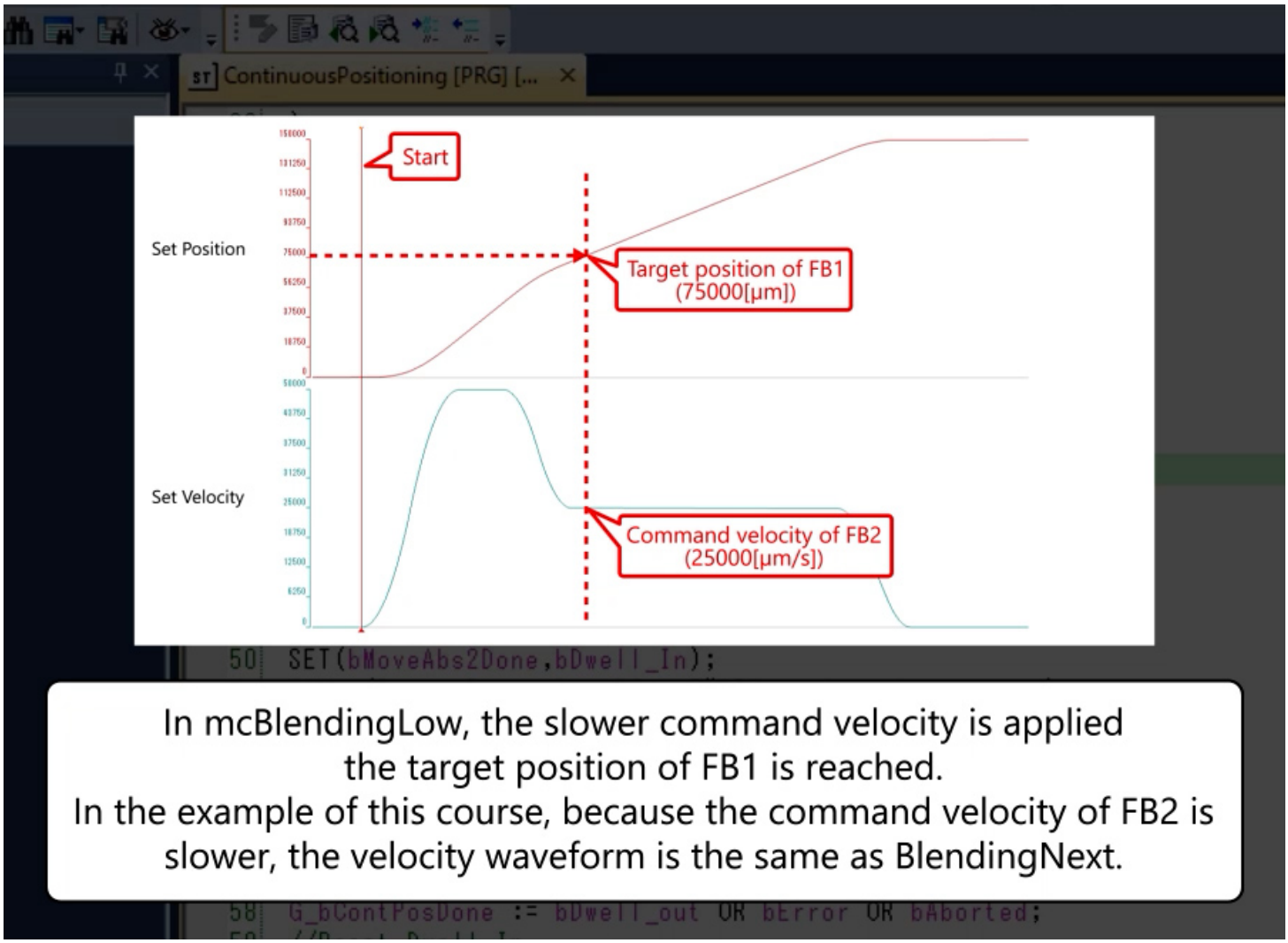
In mcBlendingNext, the command velocity of FB1 is applied when the target position of FB2 is reached.
Then, the axis moves to the target position of FB2.


ST ContinuousPositioning [PRG] [...
33);
34 //Positioning2
35 MC_MoveAbsolute_2(
36 Axis := Axis0001.AxisRef ,
37 Execute := bMoveAbs1Active ,
38 Position := lePosition2 ,
39 Velocity := lePosVelocity2 ,
40 Acceleration:= lePosAcceleration2 ,
41 Deceleration:= lePosDeceleration2 ,
42 Jerk := lePosJerk2 ,
43 Direction := MC_DIRECTION__mcShortestWay ,
44 BufferMode := MC_BUFFER_MODE__mcBlendingLow,
45 Done => bMoveAbs2Done ,
46 CommandAborted => bMoveAbs2Aborted,
47 Error => bMoveAbs2Error
48);
49 //Dwell
50 SET(bMoveAbs2Done,bDwell_In);
Next, check the mcBlendingLow operation.
Change the BufferMode input of MC_MoveAbsolute_2 to
"MC_BUFFER_MODE__mcBlendingLow", rebuild all the programs,
and write to the Motion module.

3.5 Operation Check



The motor runs when X25 is turned on.
The graph shows the Set Position and Set Velocity.



In mcBlendingLow, the slower command velocity is applied the target position of FB1 is reached.
In the example of this course, because the command velocity of FB2 is slower, the velocity waveform is the same as BlendingNext.

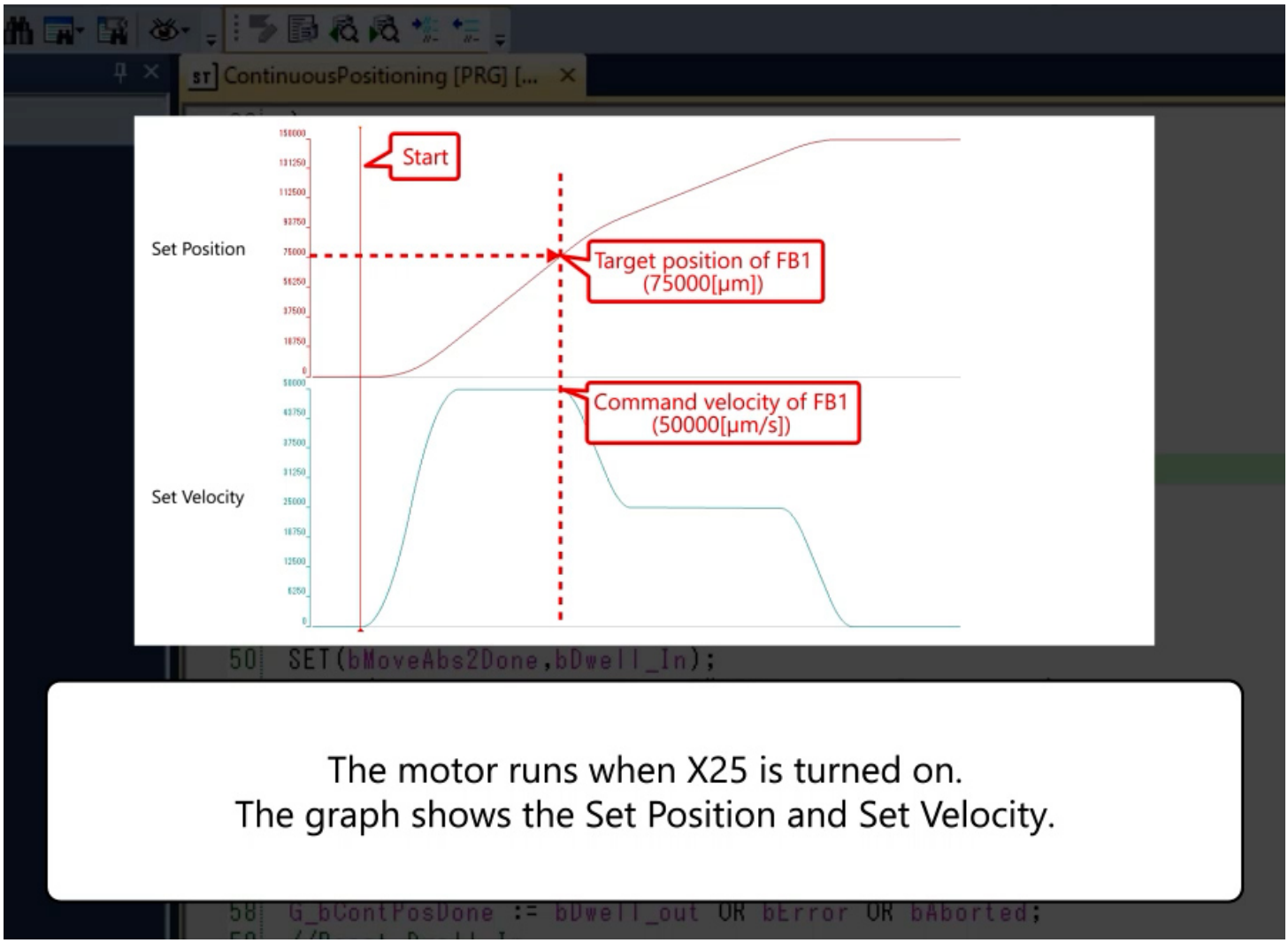
3.5 Operation Check

ST ContinuousPositioning [PRG] [...

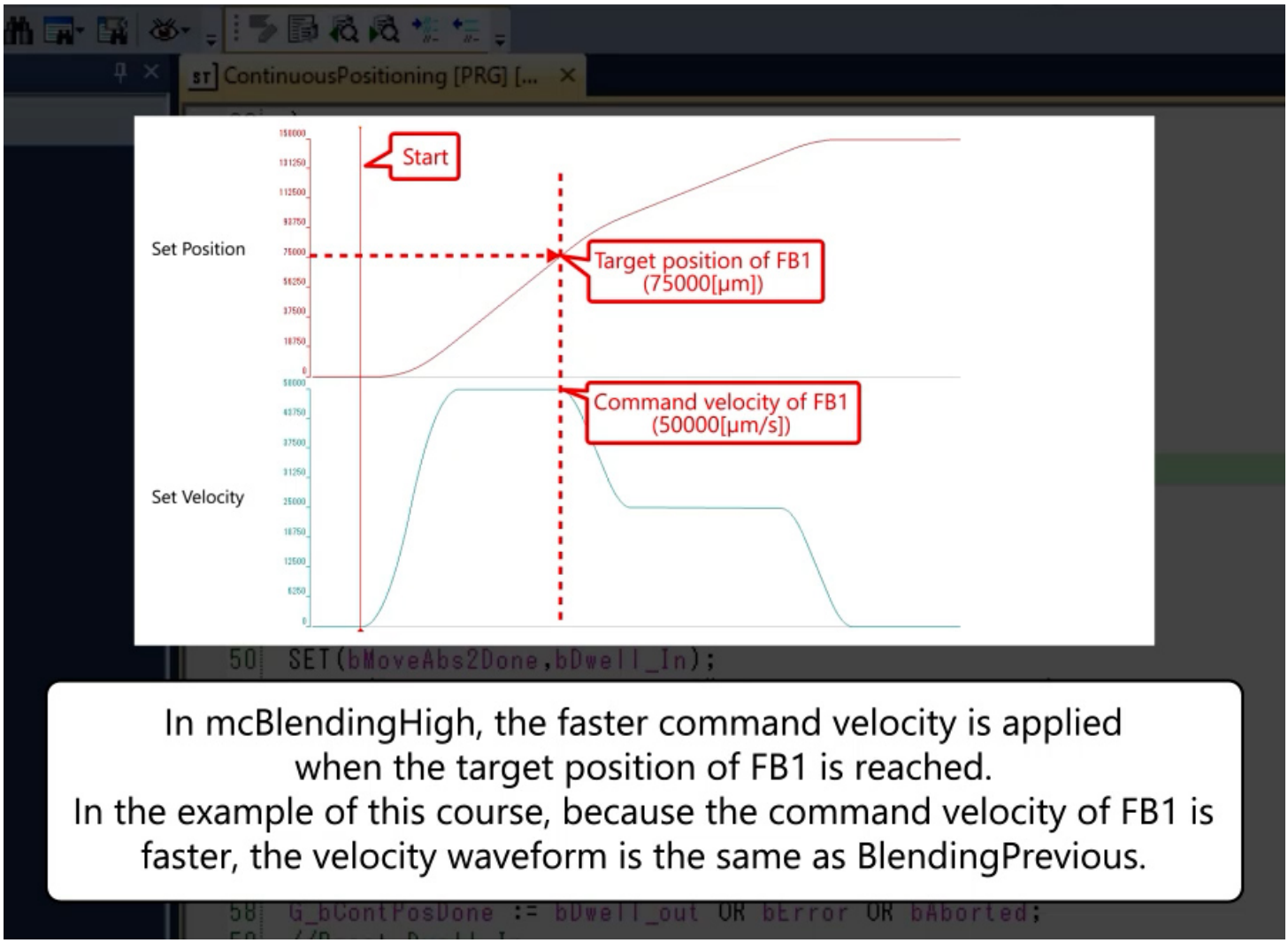
```
33  );
34  //Positioning2
35  MC_MoveAbsolute_2(
36      Axis         := Axis0001.AxisRef ,
37      Execute      := bMoveAbs1Active ,
38      Position     := lePosition2 ,
39      Velocity     := lePosVelocity2 ,
40      Acceleration:= lePosAcceleration2 ,
41      Deceleration:= lePosDeceleration2 ,
42      Jerk         := lePosJerk2 ,
43      Direction    := MC_DIRECTION__mcShortestWay ,
44      BufferMode   := MC_BUFFER_MODE__mcBlendingHigh,
45      Done              => bMoveAbs2Done ,
46      CommandAborted    => bMoveAbs2Aborted,
47      Error             => bMoveAbs2Error
48  );
49  //Dwell
50  SET(bMoveAbs2Done,bDwell_In);
```

Lastly, check the mcBlendingHigh operation.
Change the BufferMode input of MC_MoveAbsolute_2 to "MC_BUFFER_MODE__mcBlendingHigh", rebuild all the programs, and write to the Motion module.

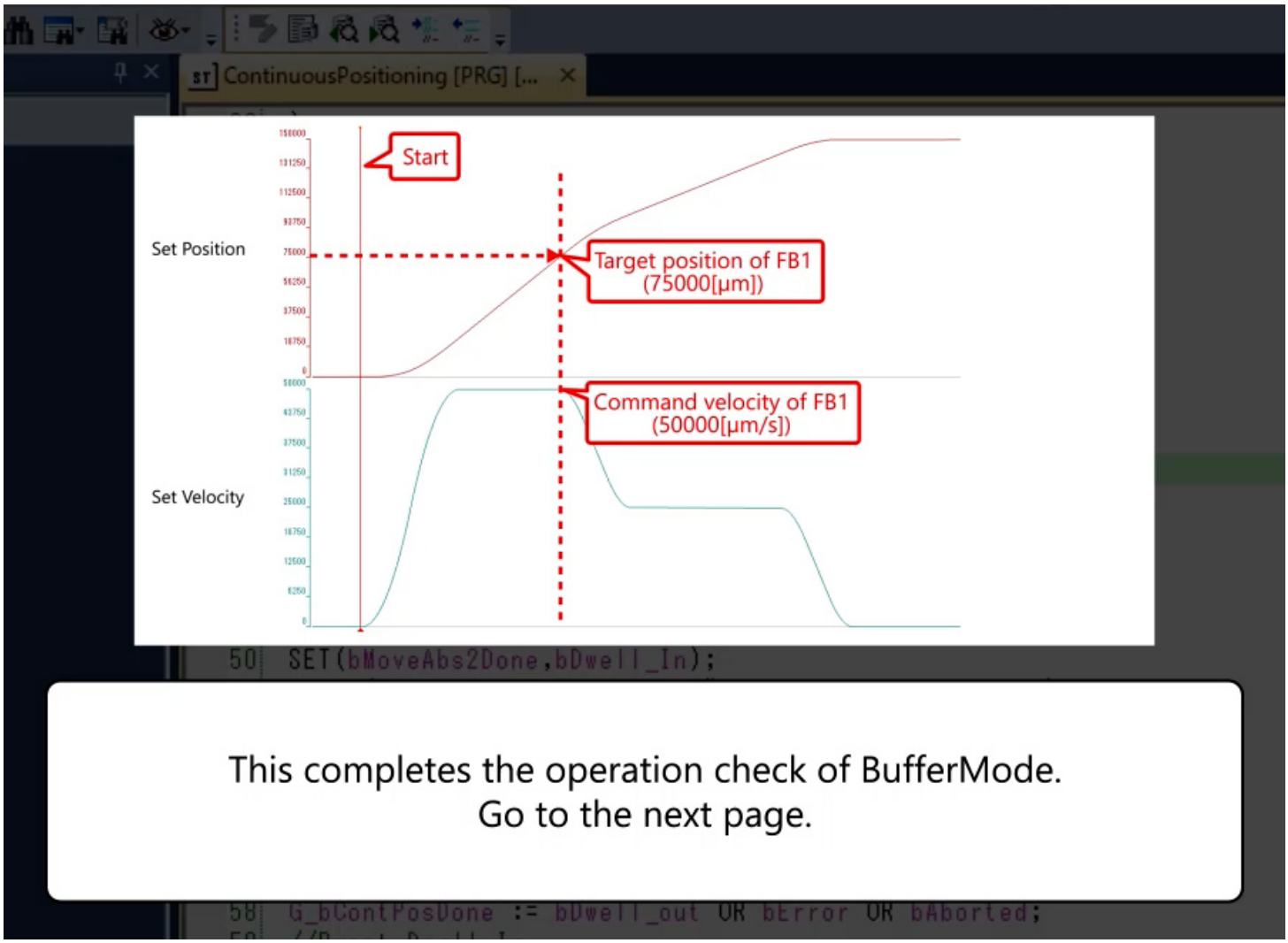
```
58  G_bContPosDone := bDwell_out OR bError OR bAborted;
```



The motor runs when X25 is turned on.
The graph shows the Set Position and Set Velocity.



In mcBlendingHigh, the faster command velocity is applied when the target position of FB1 is reached.
In the example of this course, because the command velocity of FB1 is faster, the velocity waveform is the same as BlendingPrevious.



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

3.6 Summary of This Chapter

In this chapter, you have learned:

- Aborting
- Buffered
- Blending
- Program Example
- Operation Check

Point

Aborting	• When the operation-type FB is running and the next operation-type FB is executed, Aborting interrupts the FB being executed and executes the next FB.
Buffered	• When the operation-type FB is running and the next operation-type FB is executed, Buffered waits until the FB being executed is completed and executes the next FB.
Blending	• When the operation-type FB is running and the next operation-type FB is executed, Blending executes the next FB without stopping the operation of the FB being executed. • In Blending, there are four velocity switching methods: BlendingLow, BlendingHigh, BlendingPrevious, and BlendingNext.
Program Example	• Select the buffer mode with the BufferMode input of the operation FB.
Operation Check	• You have checked the difference in operation of each buffer mode in the video.

Chapter 4 Operation with PLC CPU

Download the sample program to be used in this chapter from the link below.
The program content is the same as the sample program described in chapter 2 and chapter 3.
Only the programming method is different.

RD78GBasic2_sample2.zip (1.39 MB)

4.1 Registering the Motion Module FB Library

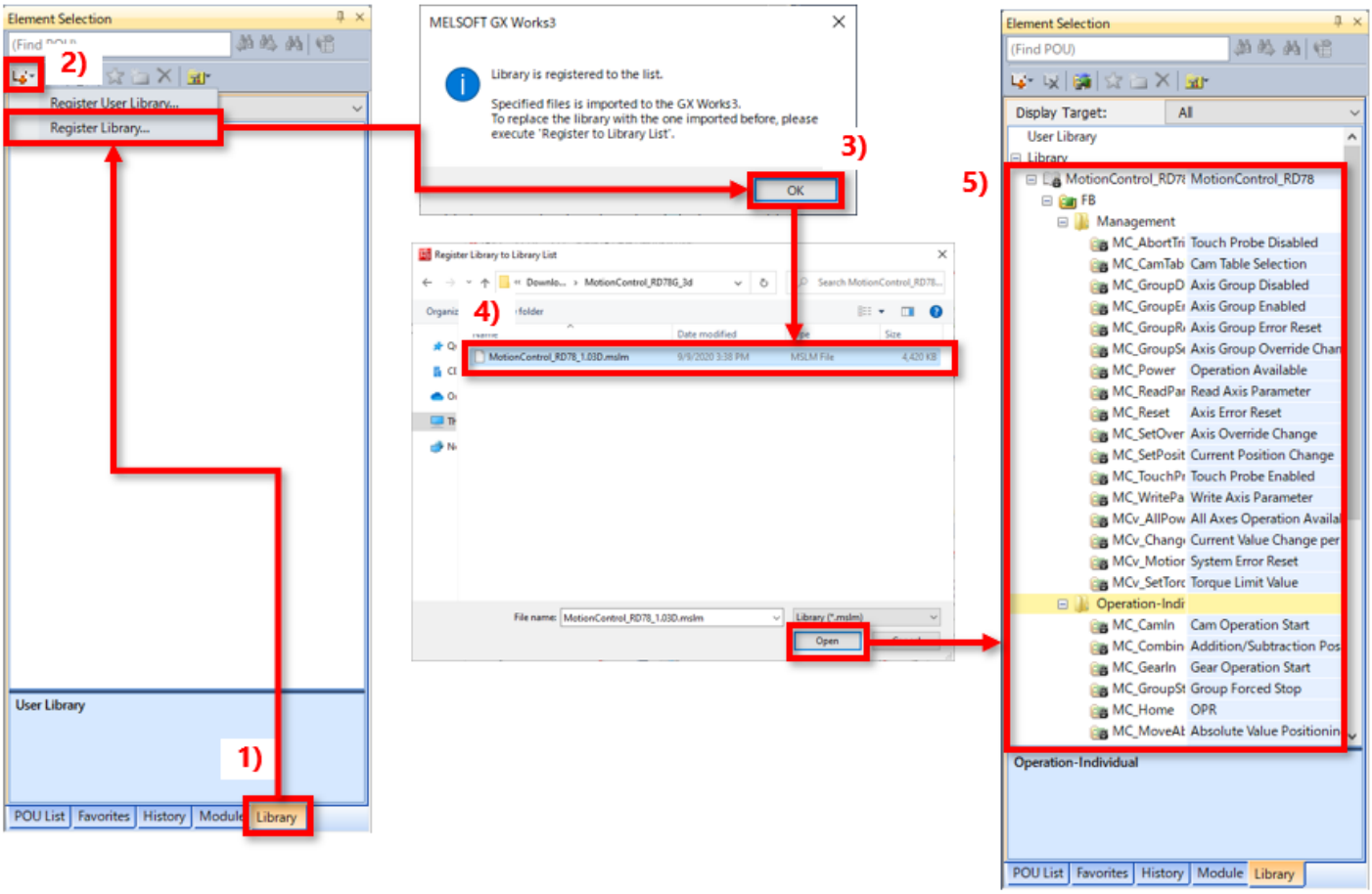
(1) Download the FB library
The Motion control FB can be used in the program of the PLC CPU by registering the FB library for the Motion module to GX Works3.
Download the FB library from the link below, and unzip the ZIP file to the desired destination.

MotionControl_RD78G_3d.zip(4.29 MB)

(Note) The latest version of FB library can be downloaded from the MITSUBISHI ELECTRIC FA Global Website.

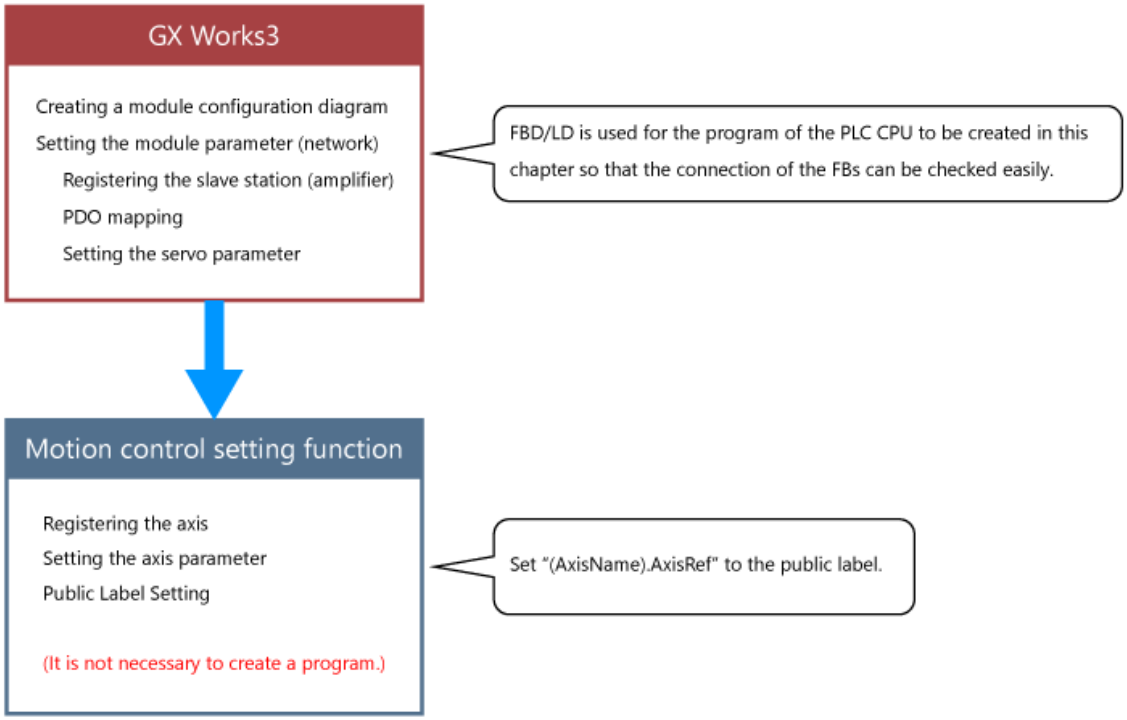
(2) Registering the FB library

1) Open any project in GX Works3, and open the Library tab in the Element Selection window.
2) Click the [Register to Library List] button at the upper part, and select [Register Library].
3) When a message "Library is registered to the list" is displayed, click [OK].
4) Select the FB library file "MotionControl_RD78_****.mslm", and click [Open].
   (**** indicates the version.)
5) The Motion control FB is registered to the library in the Element Selection window.

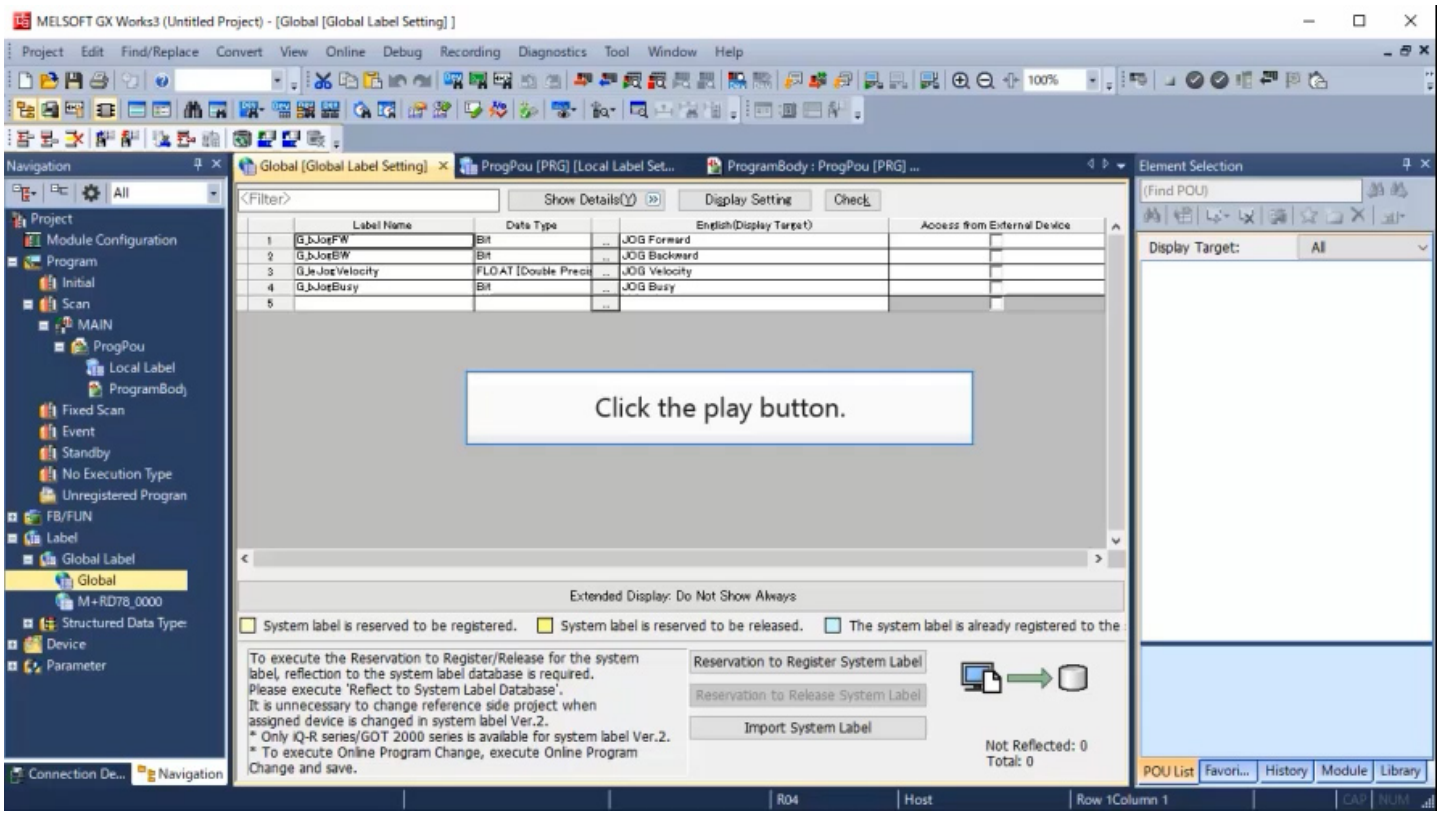


4.2 Creating Projects

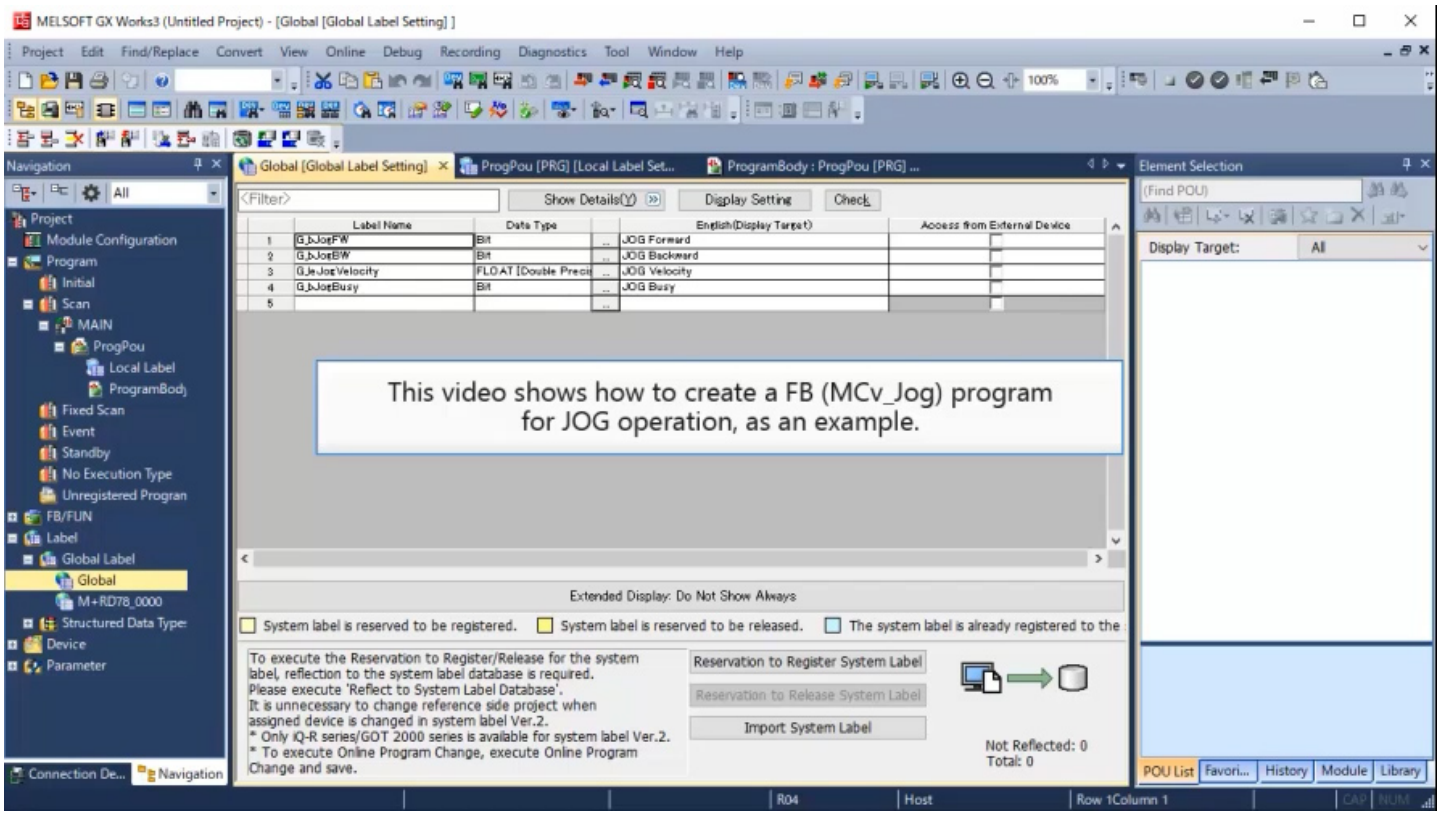
The procedure for creating a project is the same as described in the previous section.



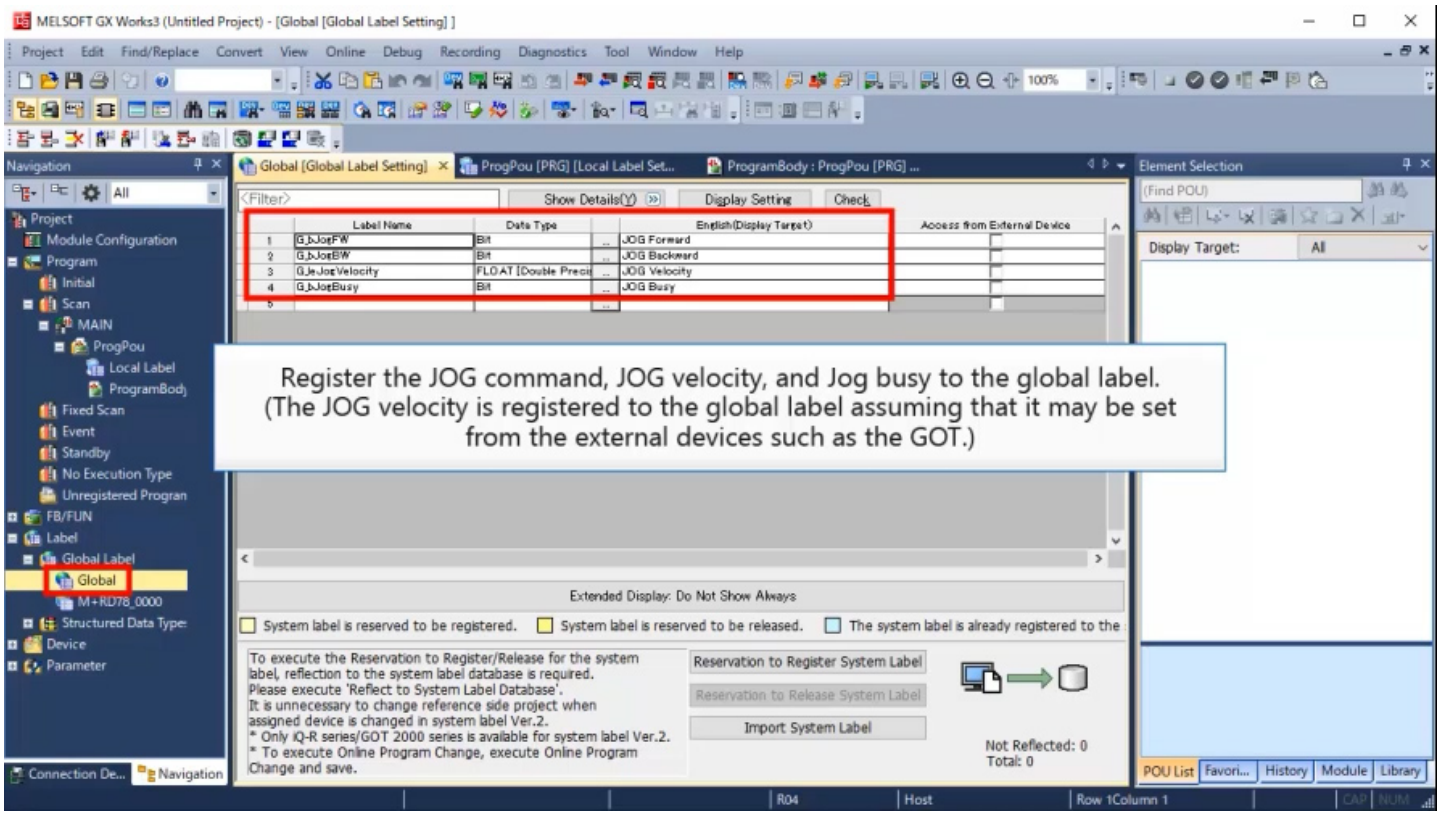
4.3 How to Use the Motion Control FB



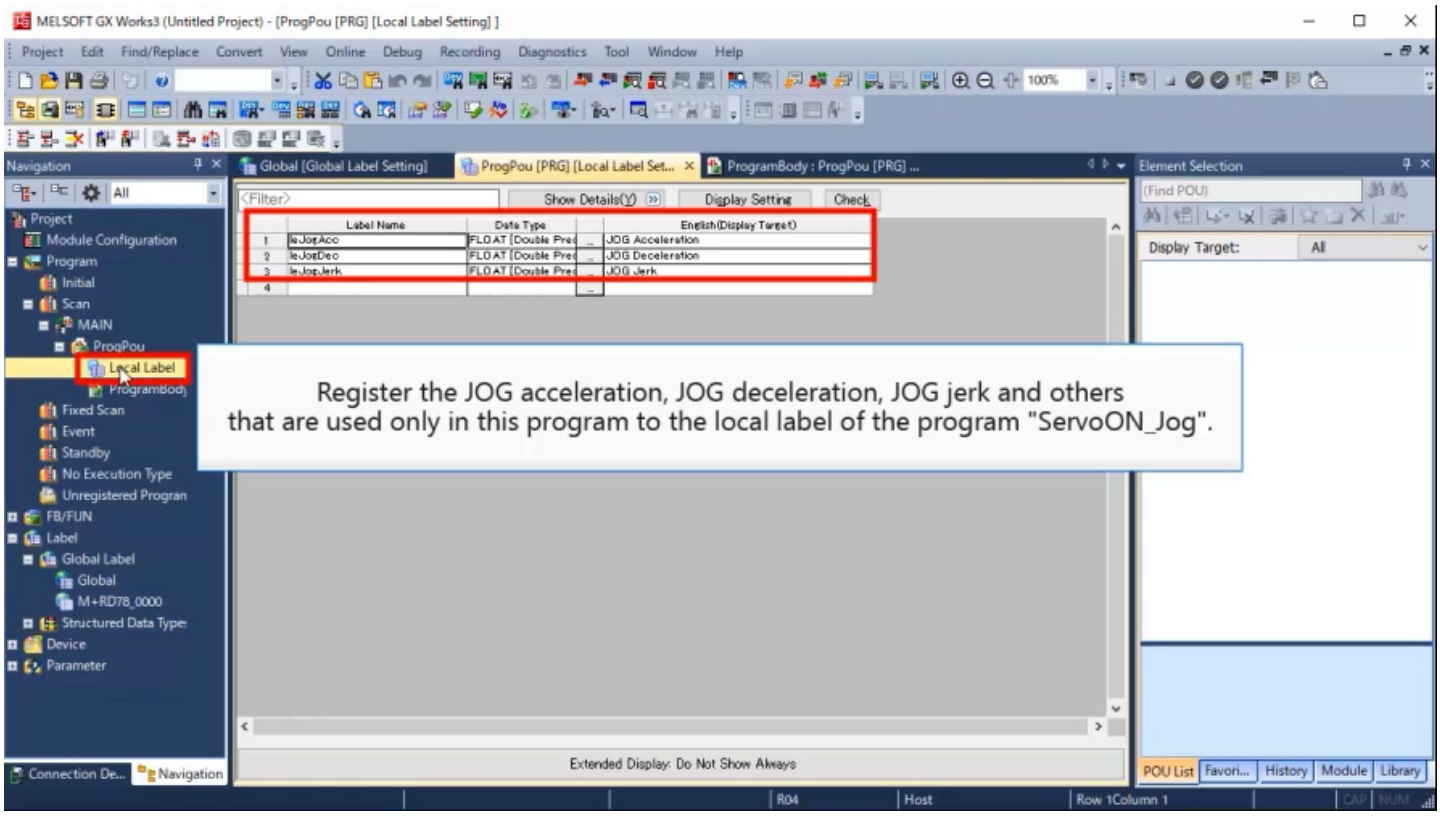
4.3 How to Use the Motion Control FB

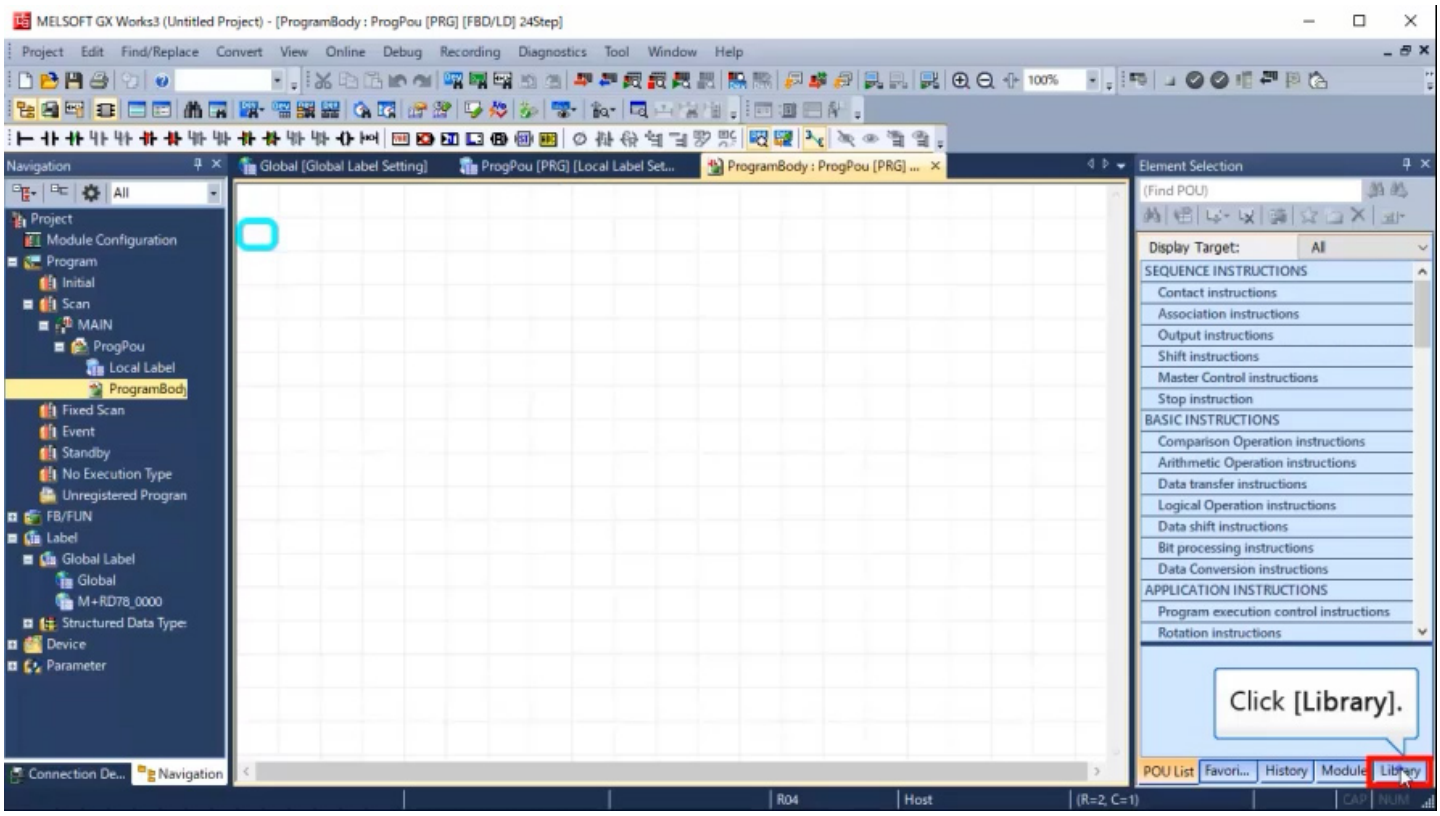


4.3 How to Use the Motion Control FB

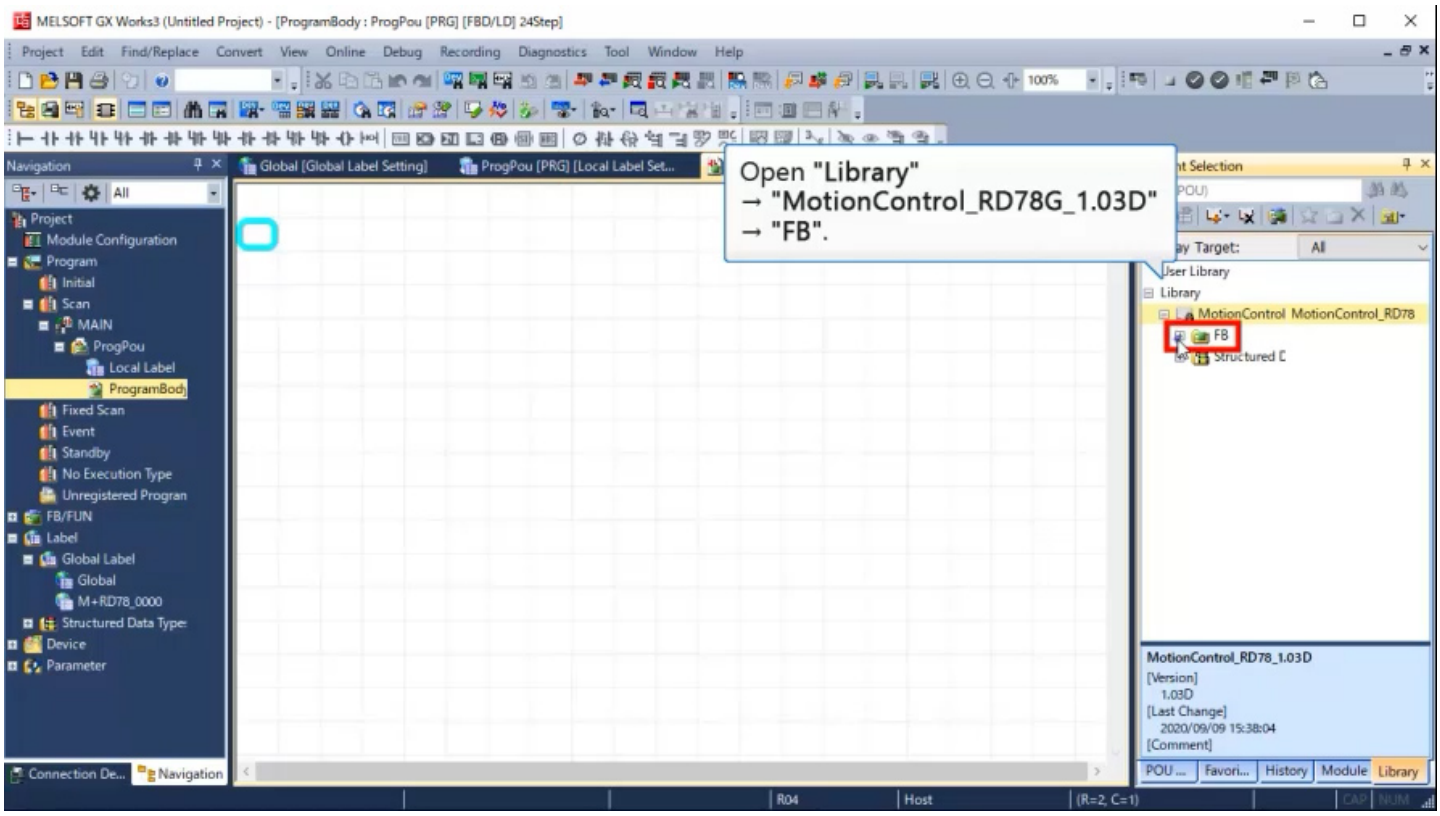


4.3 How to Use the Motion Control FB

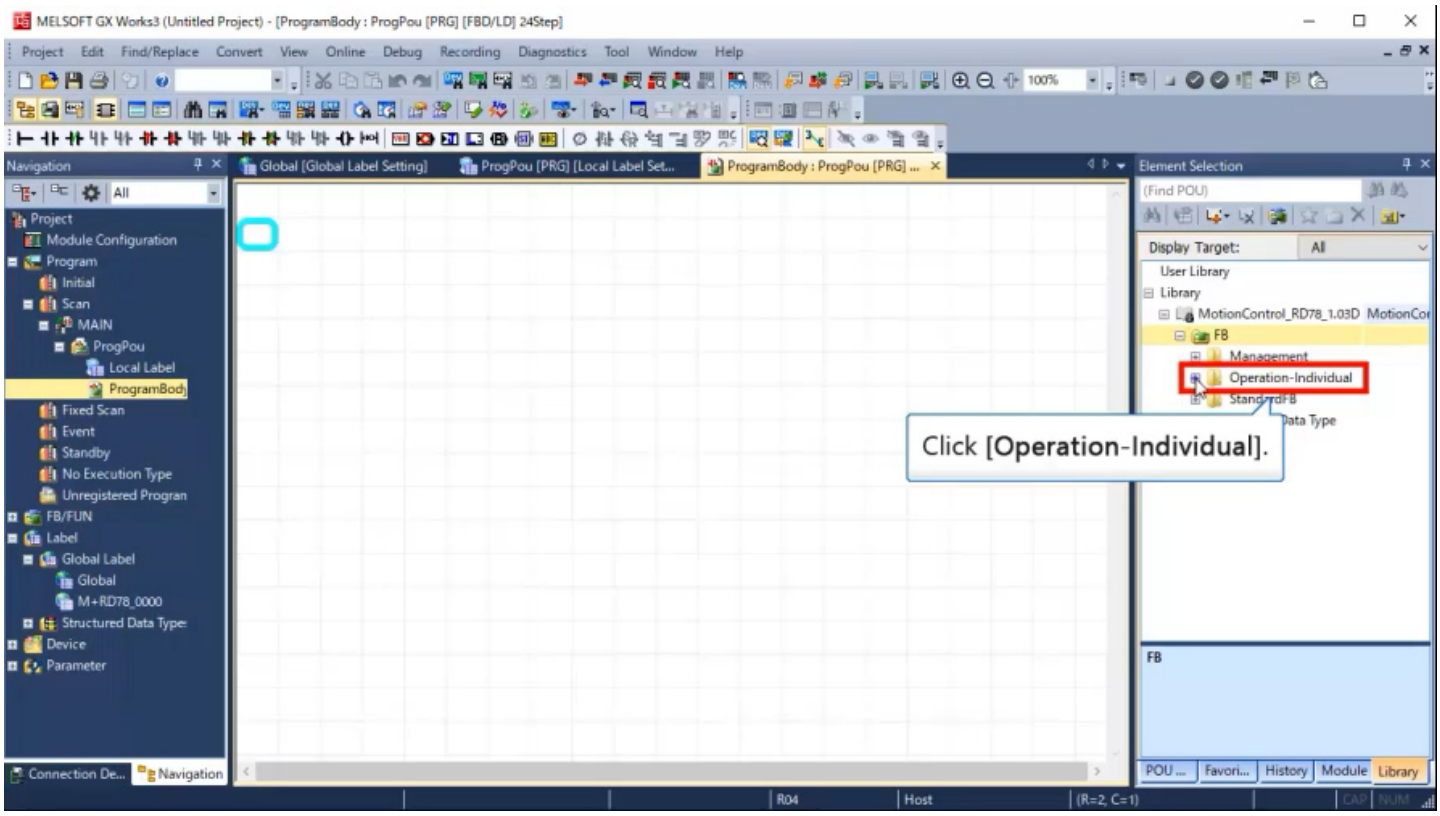



MELSOFT GX Works3 (Untitled Project) - [ProgramBody : ProgPou [PRG] [FBD/LD] 24Step]
Project Edit Find/Replace Convert View Online Debug Recording Diagnostics Tool Window Help
Navigation
Project
Module Configuration
Program
Initial
Scan
MAIN
ProgPou
Local Label
ProgramBody
Fixed Scan
Event
Standby
No Execution Type
Unregistered Progran
FB/FUN
Label
Global Label
Global
M+RD78_0000
Structured Data Type
Device
Parameter
Connection De...
Navigation
Global [Global Label Setting]
ProgPou [PRG] [Local Label Set...
ProgramBody : ProgPou [PRG] ...
Element Selection
(Find POU)
Display Target:
All
SEQUENCE INSTRUCTIONS
Contact instructions
Association instructions
Output instructions
Shift instructions
Master Control instructions
Stop instruction
BASIC INSTRUCTIONS
Comparison Operation instructions
Arithmetic Operation instructions
Data transfer instructions
Logical Operation instructions
Data shift instructions
Bit processing instructions
Data Conversion instructions
APPLICATION INSTRUCTIONS
Program execution control instructions
Rotation instructions
Click [Library].
POU List
Favori...
History
Module
Library
R04
Host
(R=2, C=1)

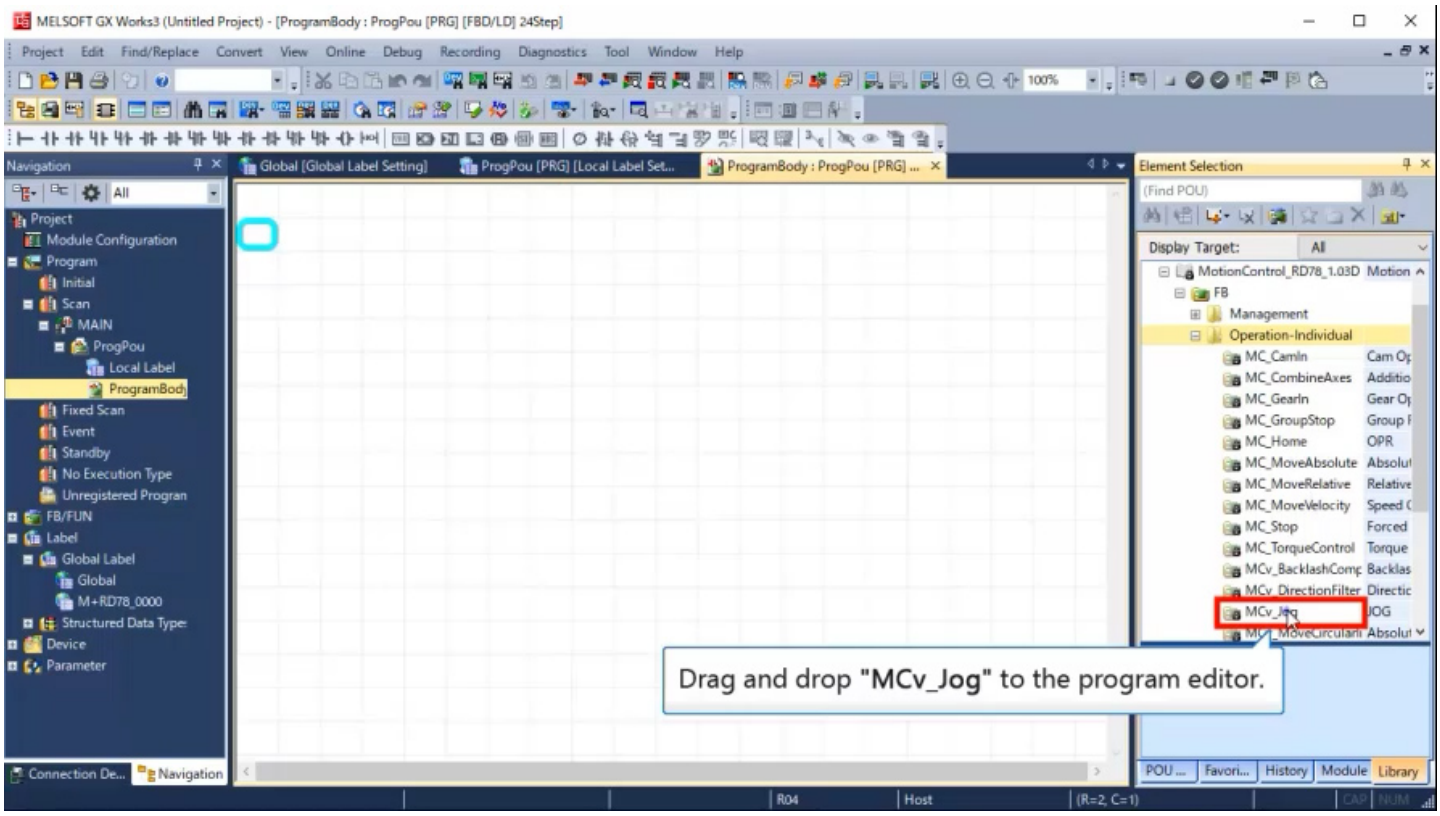
4.3 How to Use the Motion Control FB



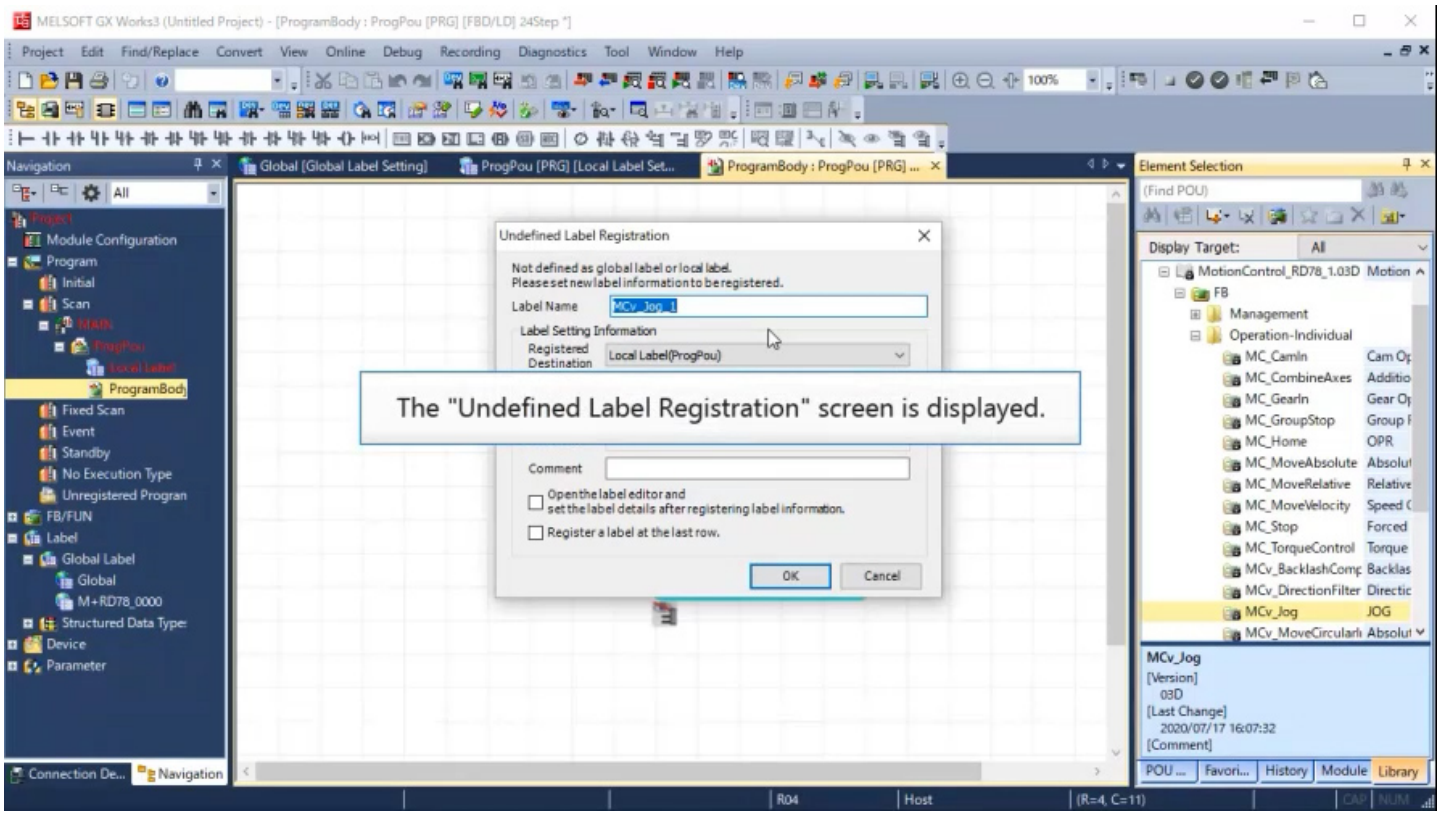
4.3 How to Use the Motion Control FB

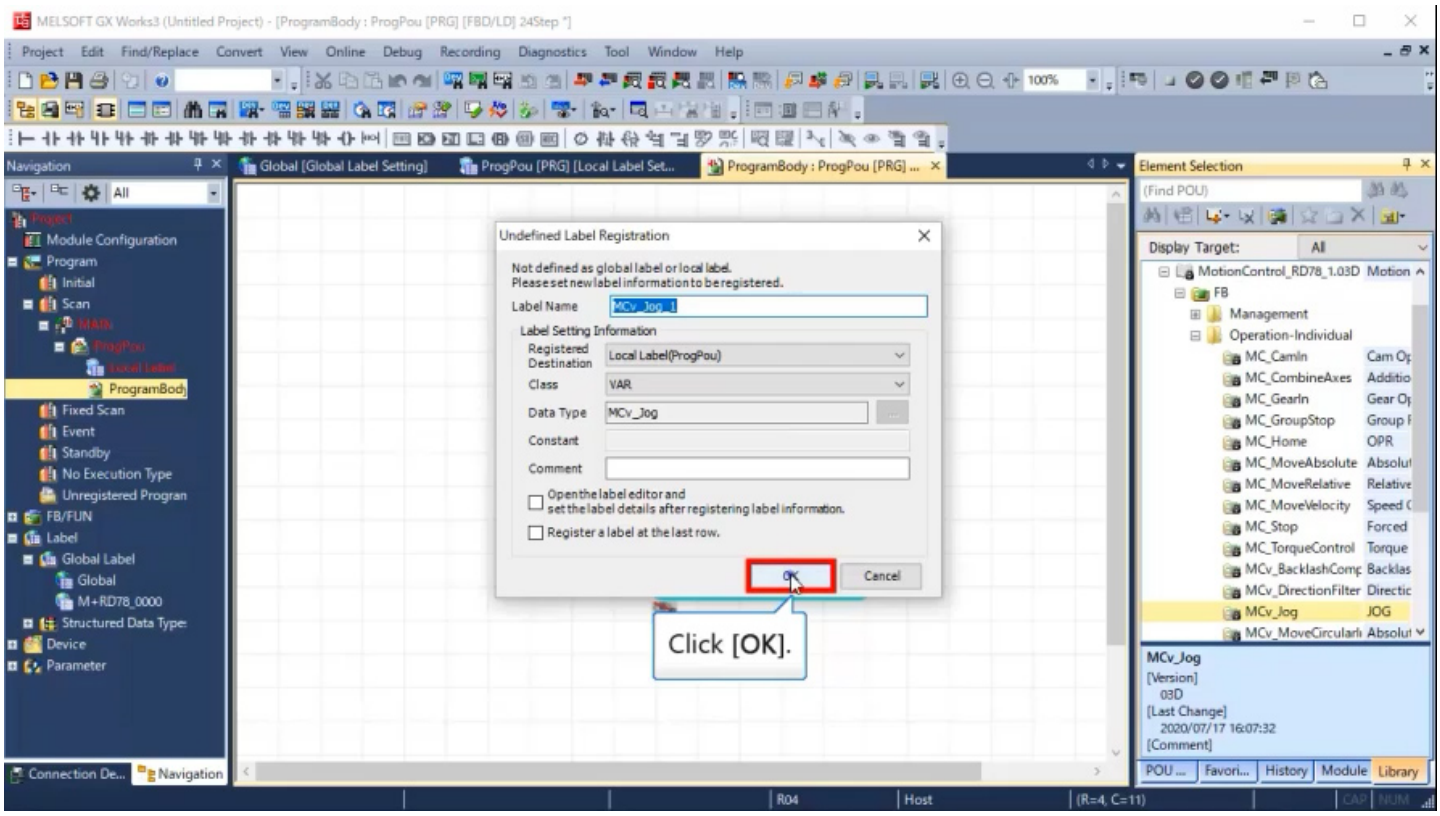


4.3 How to Use the Motion Control FB

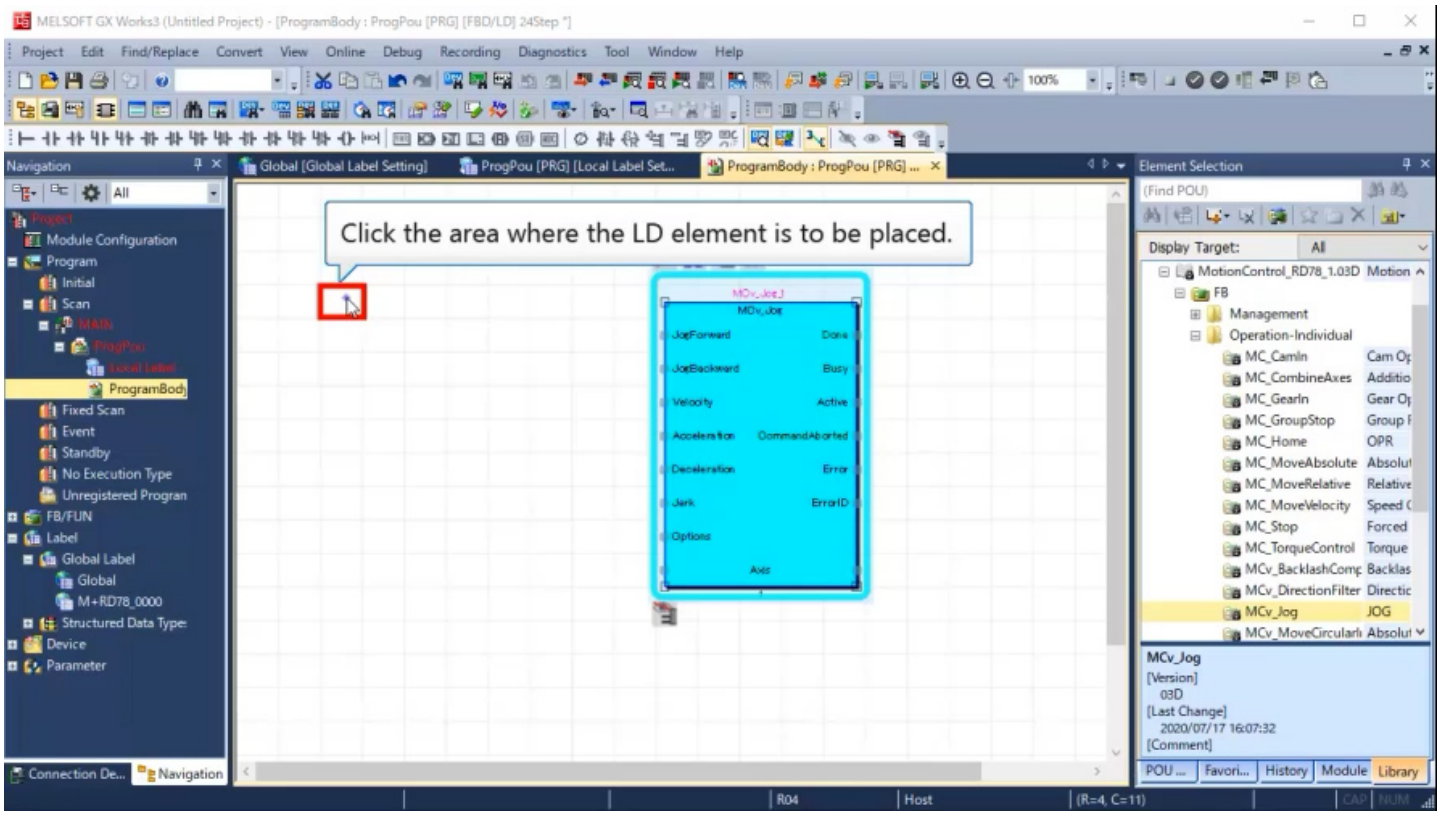


4.3 How to Use the Motion Control FB

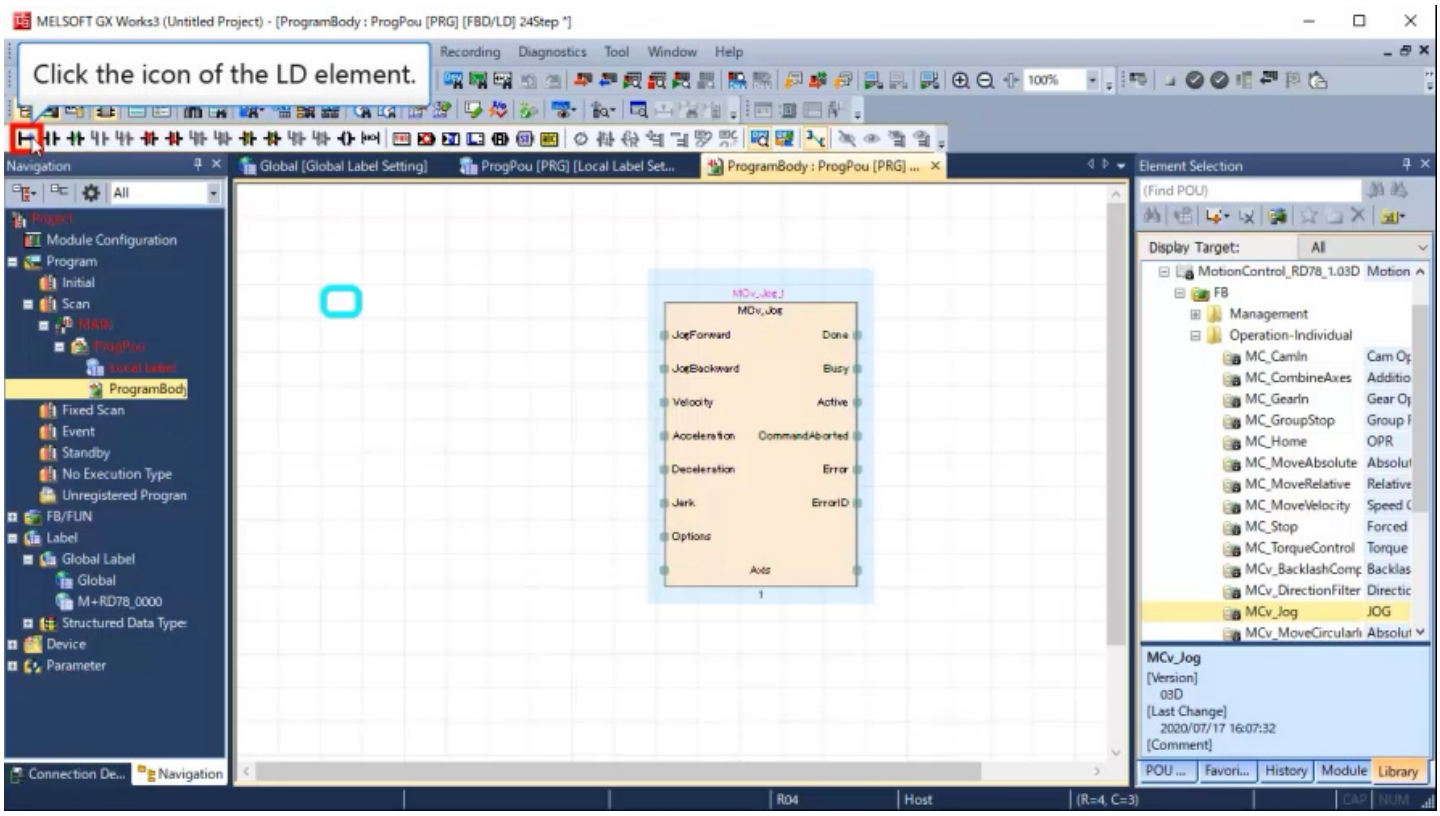



MELSOFT GX Works3 (Untitled Project) - [ProgramBody : ProgPou [PRG] [FBD/LD] 24Step *]
Project Edit Find/Replace Convert View Online Debug Recording Diagnostics Tool Window Help
Navigation
Module Configuration
Program
Initial
Scan
ProgramBody
Fixed Scan
Event
Standby
No Execution Type
Unregistered Progran
FB/FUN
Label
Global Label
Global
M+RD78_0000
Structured Data Type
Device
Parameter
Global [Global Label Setting]
ProgPou [PRG] [Local Label Set...
ProgramBody : ProgPou [PRG] ...
Undefined Label Registration
Not defined as global label or local label.
Please set new label information to be registered.
Label Name
MCv_Jog_1
Label Setting Information
Registered Destination
Local Label(ProgPou)
Class
VAR
Data Type
MCv_Jog
Constant
Comment
Open the label editor and set the label details after registering label information.
Register a label at the last row.
OK
Cancel
Click [OK].
Element Selection
(Find POU)
Display Target:
All
MotionControl_RD78_1.03D
FB
Management
Operation-Individual
MC_CamIn
MC_CombineAxes
MC_GearIn
MC_GroupStop
MC_Home
MC_MoveAbsolute
MC_MoveRelative
MC_MoveVelocity
MC_Stop
MC_TorqueControl
MCv_BacklashComp
MCv_DirectionFilter
MCv_Jog
MCv_MoveCircularIn
MCv_Jog
[Version]
03D
[Last Change]
2020/07/17 16:07:32
[Comment]
POU ... Favori... History Module Library
Connection De... Navigation
R04 Host (R=4, C=11)

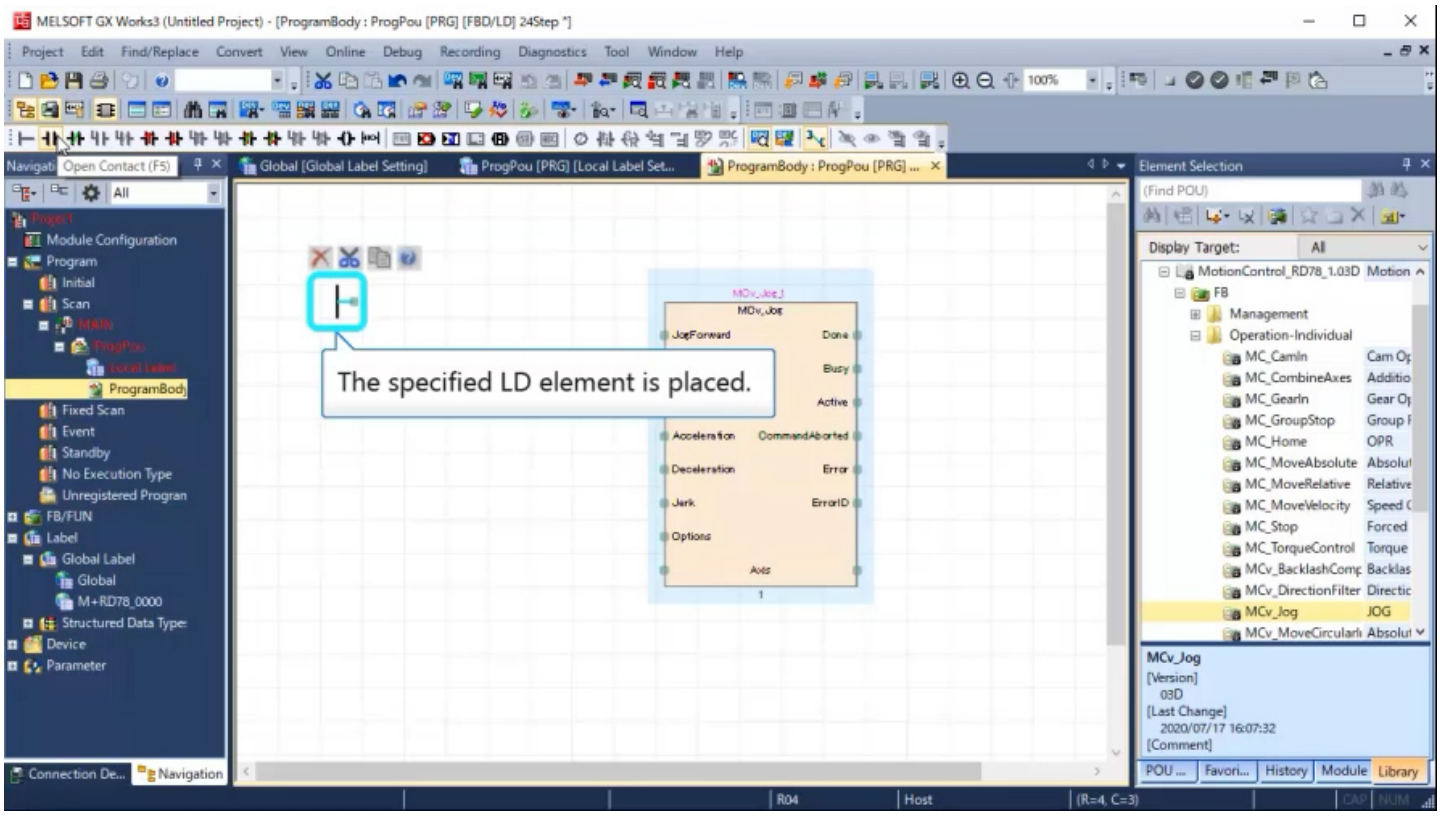
4.3 How to Use the Motion Control FB



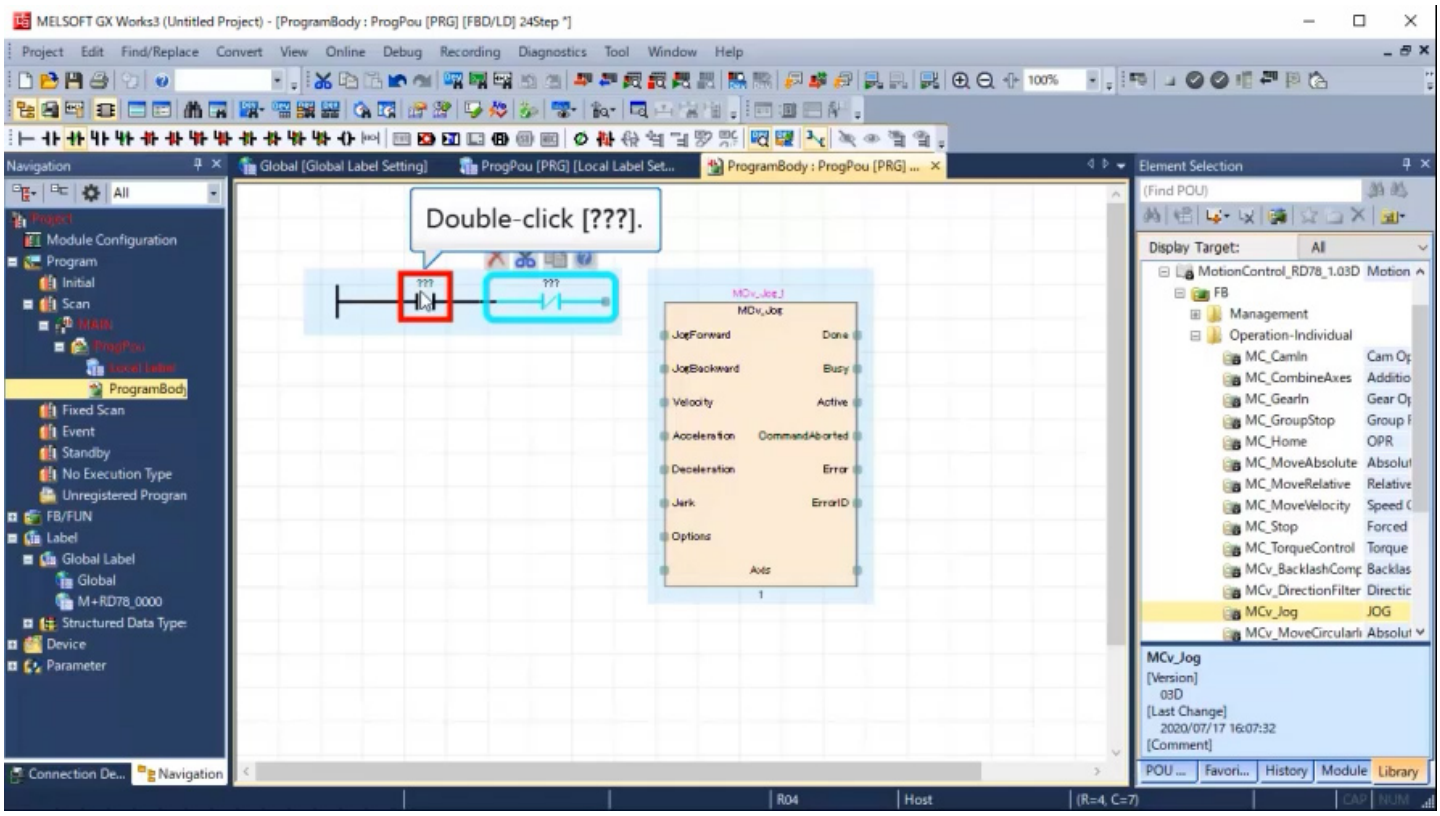
4.3 How to Use the Motion Control FB



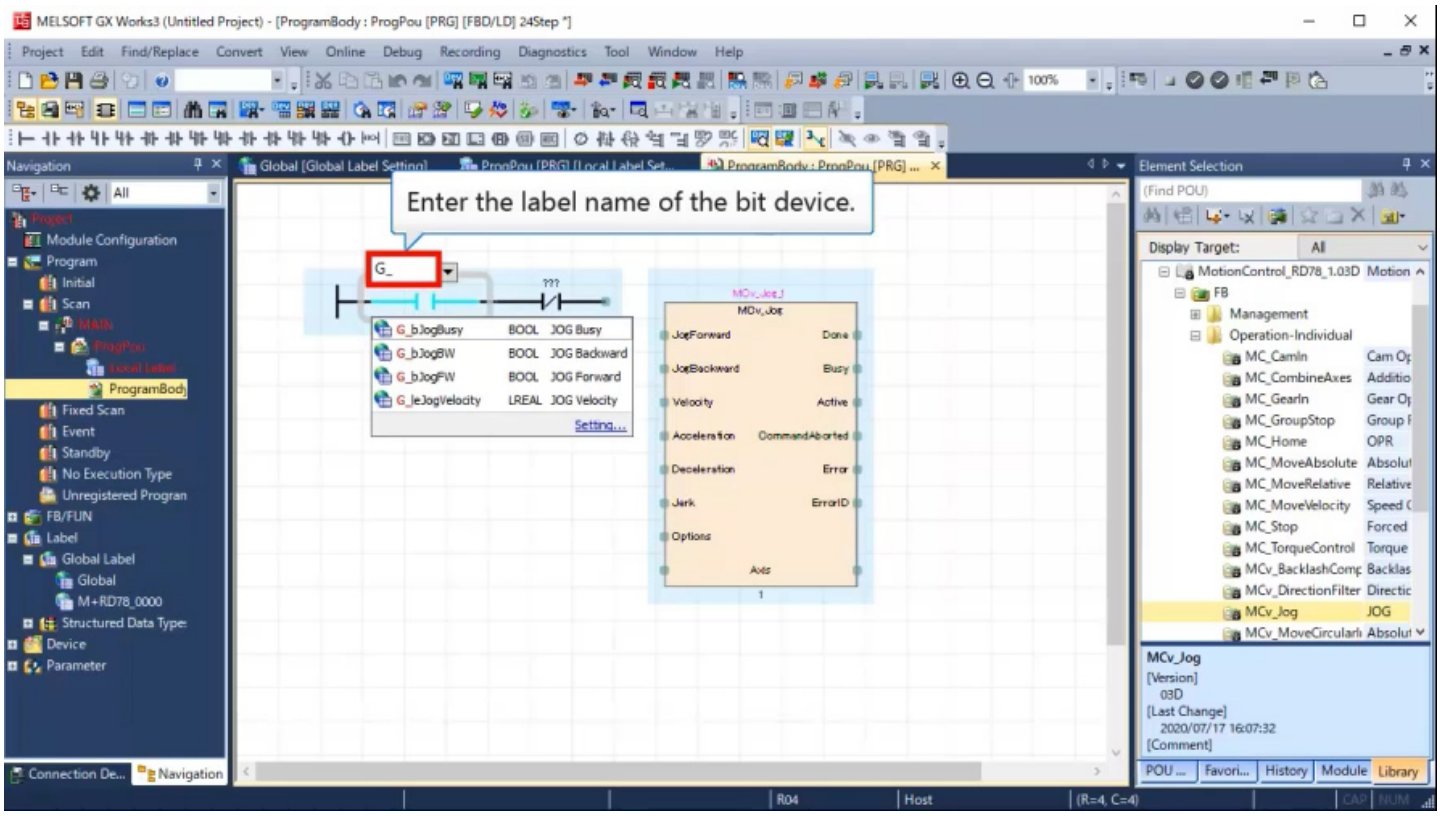
4.3 How to Use the Motion Control FB



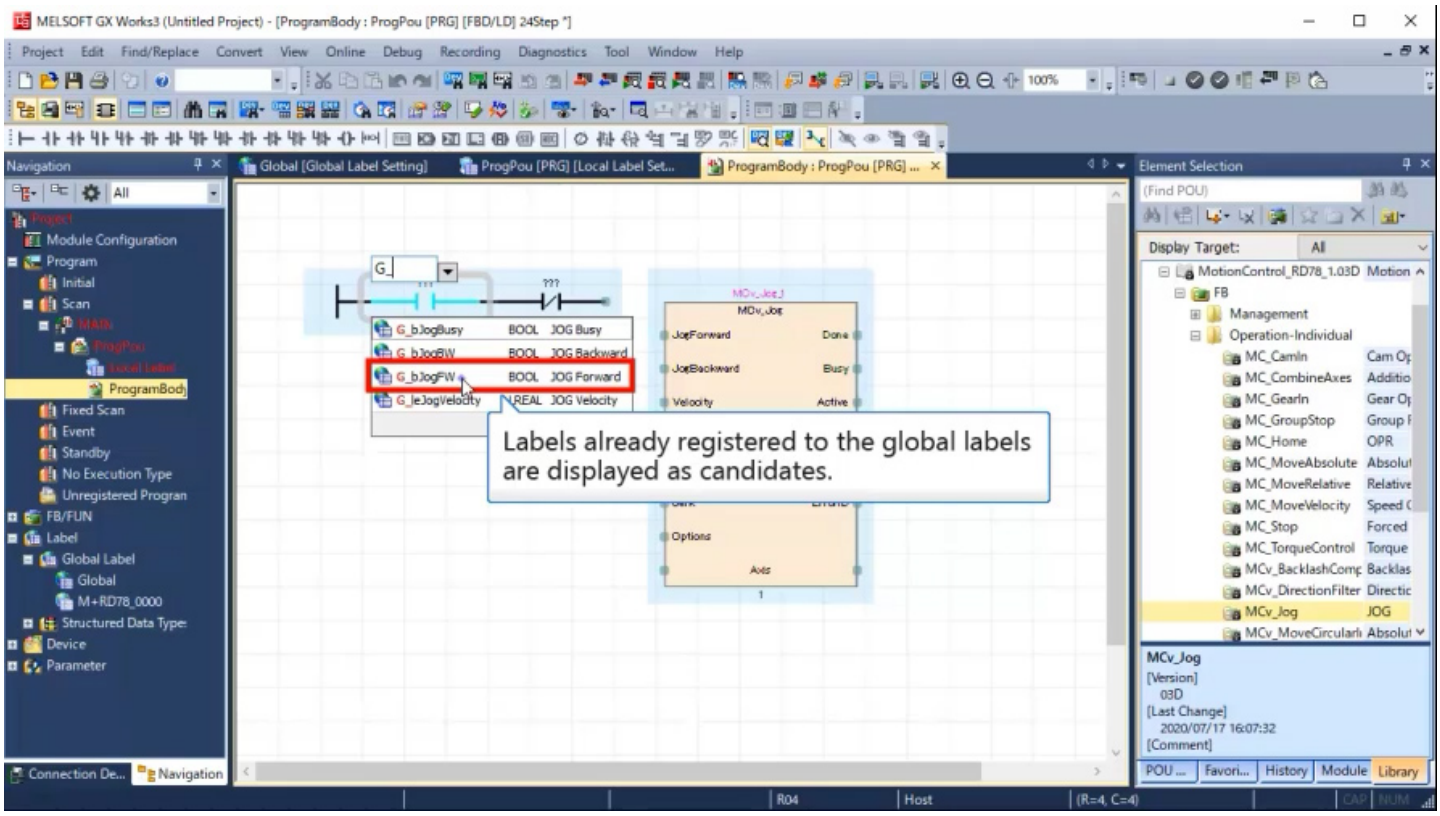
4.3 How to Use the Motion Control FB



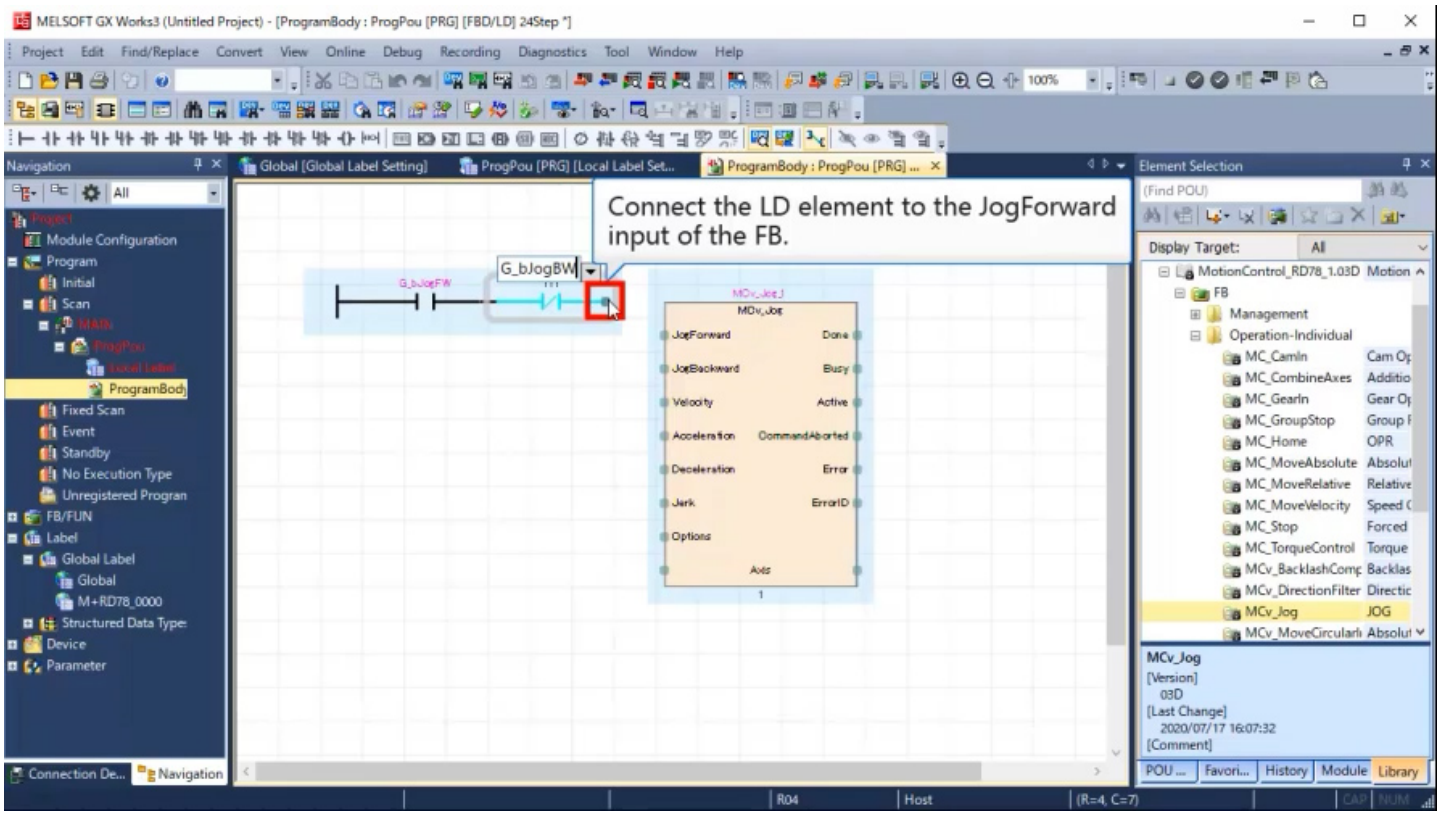
4.3 How to Use the Motion Control FB

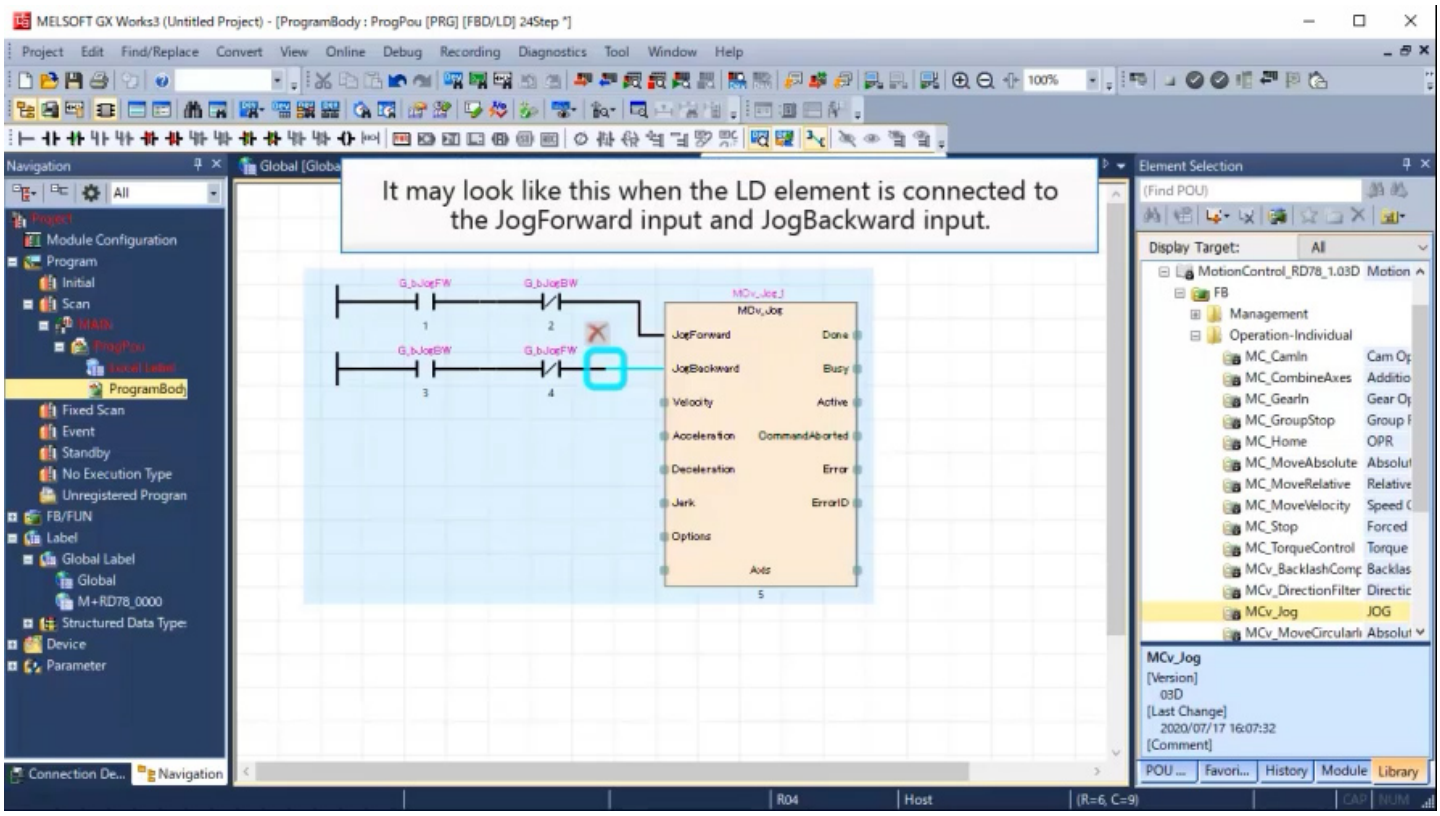


4.3 How to Use the Motion Control FB

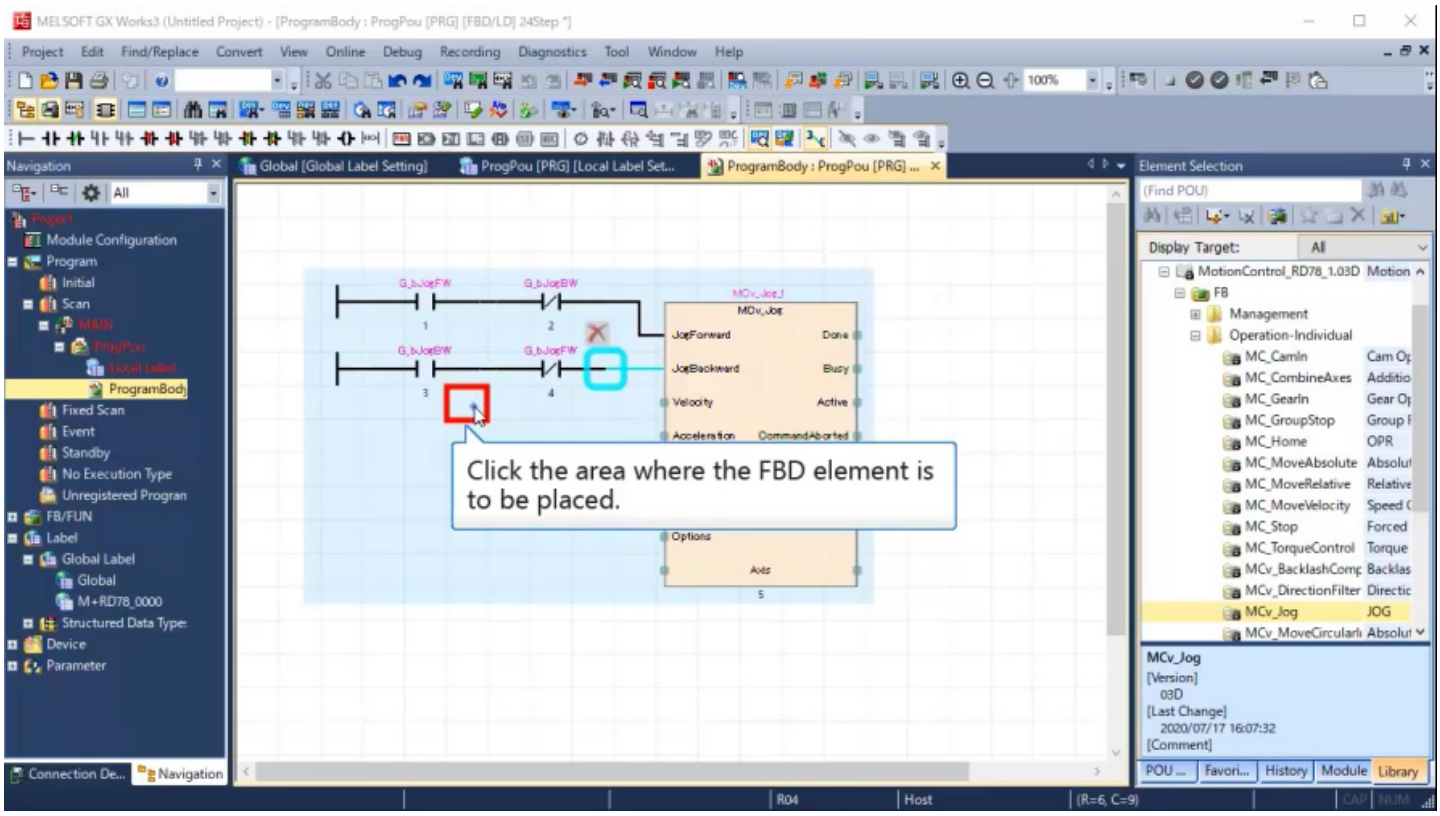


4.3 How to Use the Motion Control FB

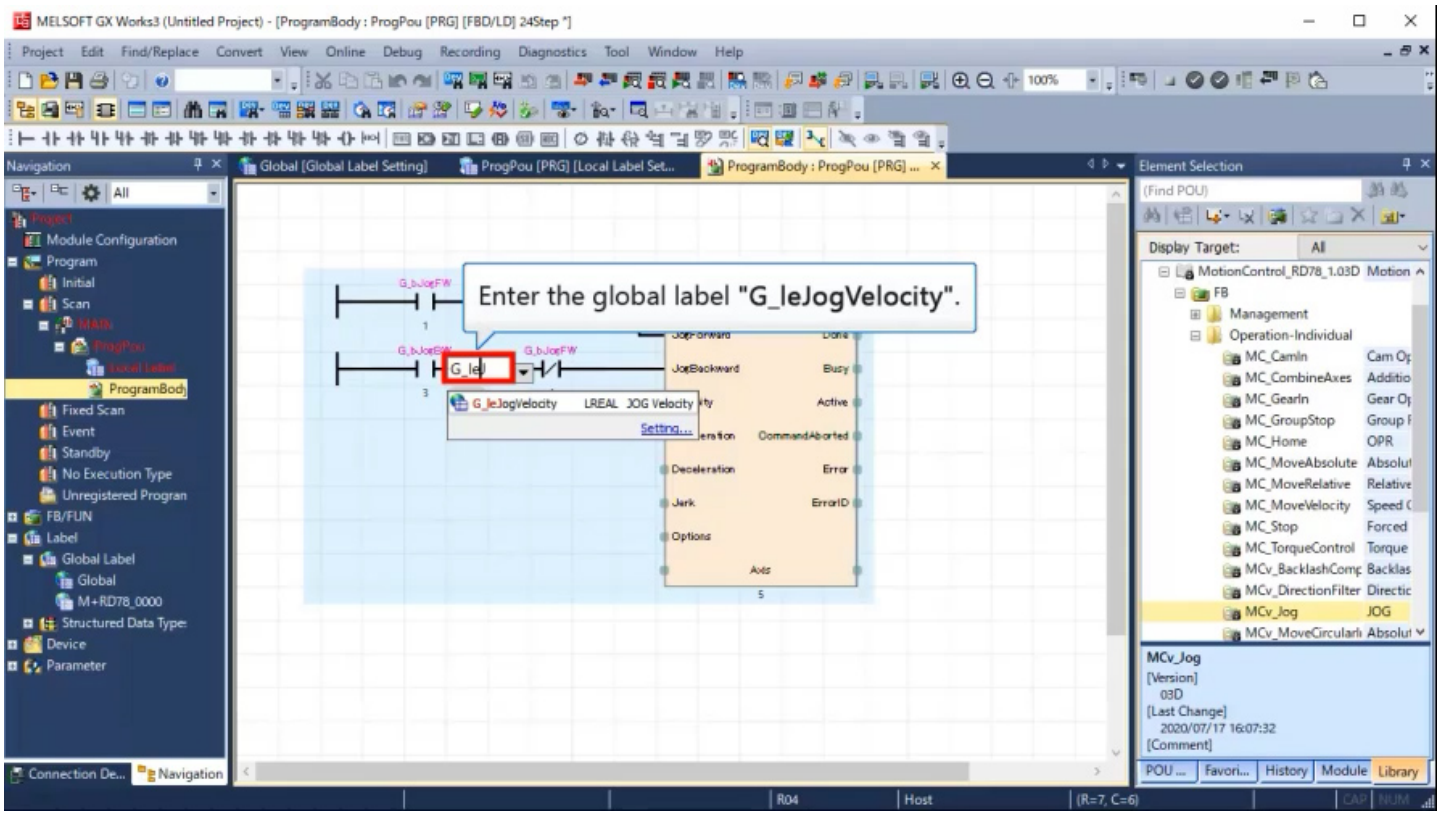



MELSOFT GX Works3 (Untitled Project) - [ProgramBody : ProgPou [PRG] [FBD/LD] 24Step *]
It may look like this when the LD element is connected to the JogForward input and JogBackward input.
G_bJogFW
G_bJogBW
MCv_Jog
JogForward
JogBackward
Velocity
Acceleration
Deceleration
Jerk
Options
Axis
Done
Busy
Active
CommandAborted
Error
ErrorID
Navigation
Element Selection
Display Target:
MotionControl_RD78_1.03D
FB
Management
Operation-Individual
MCv_Jog
[Version]
03D
[Last Change]
2020/07/17 16:07:32
[Comment]

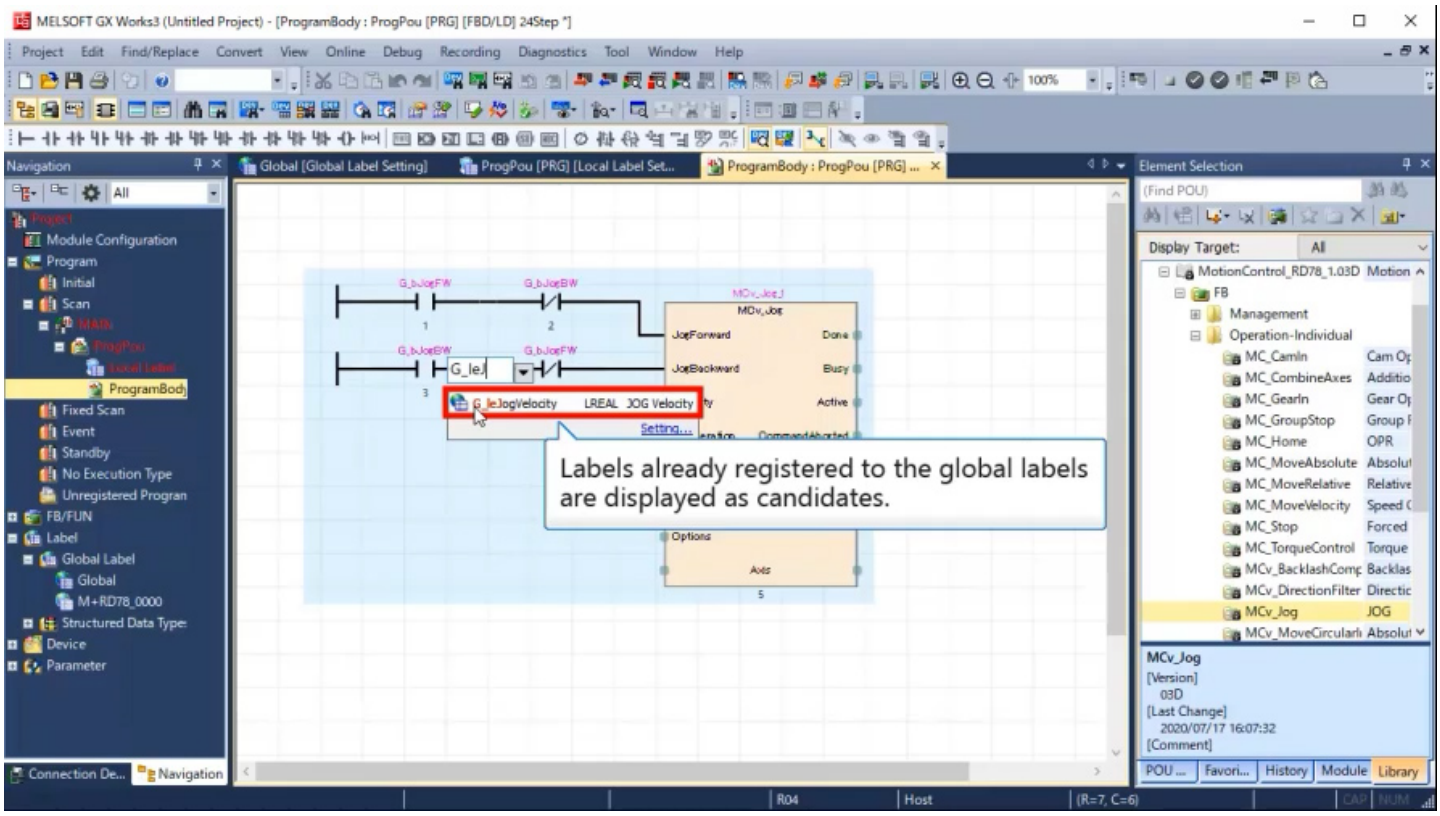
4.3 How to Use the Motion Control FB



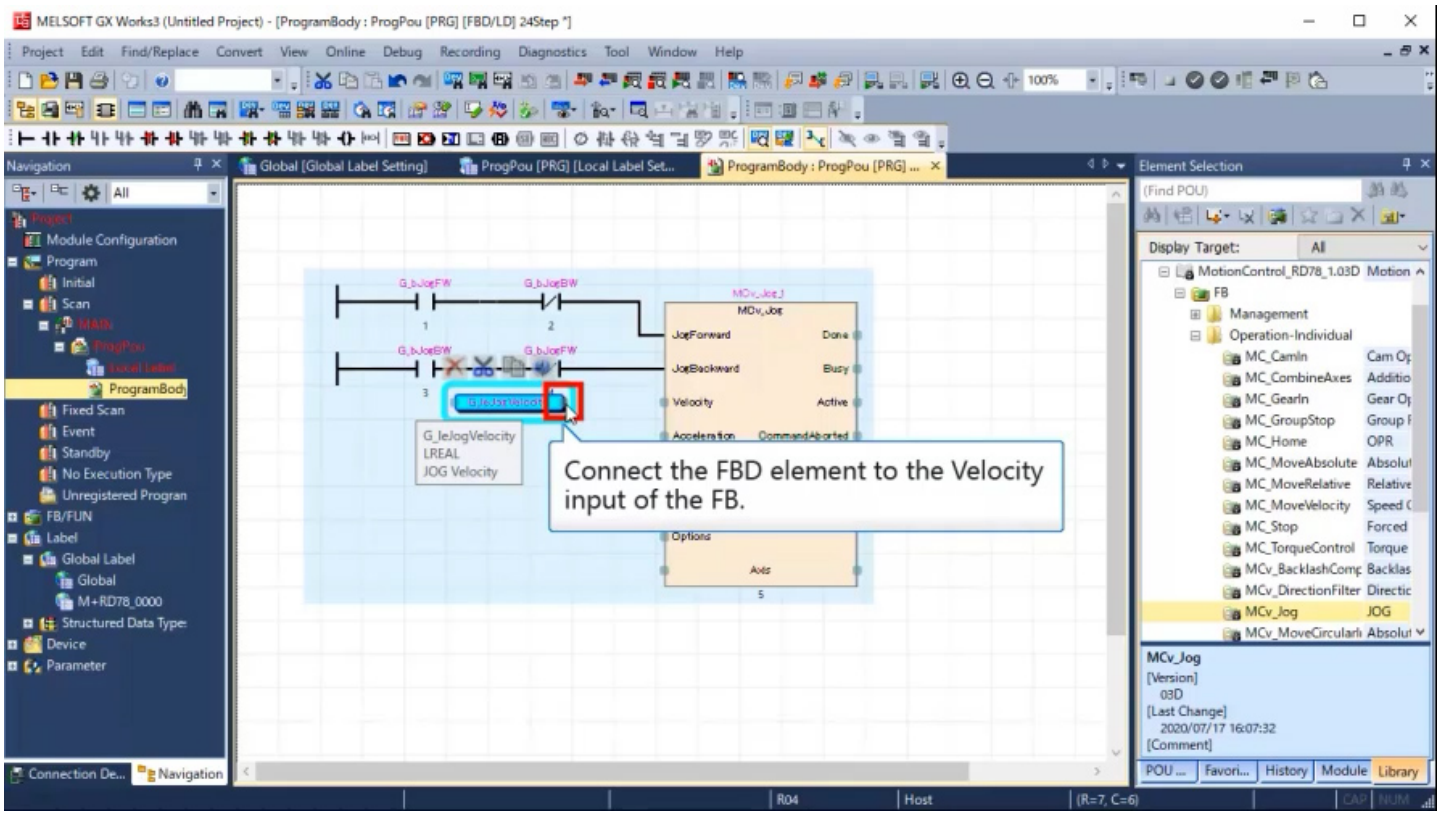
4.3 How to Use the Motion Control FB



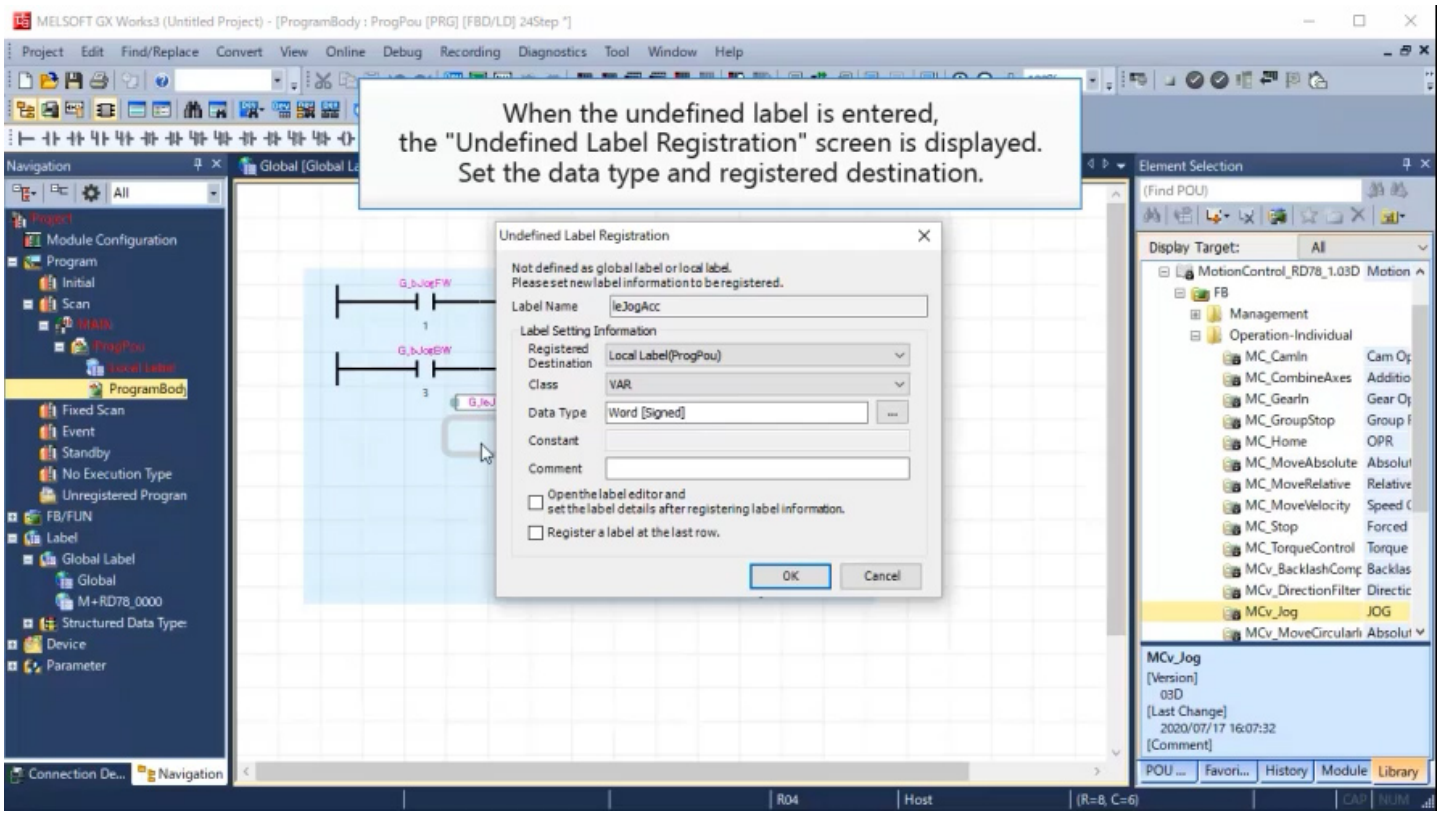
4.3 How to Use the Motion Control FB



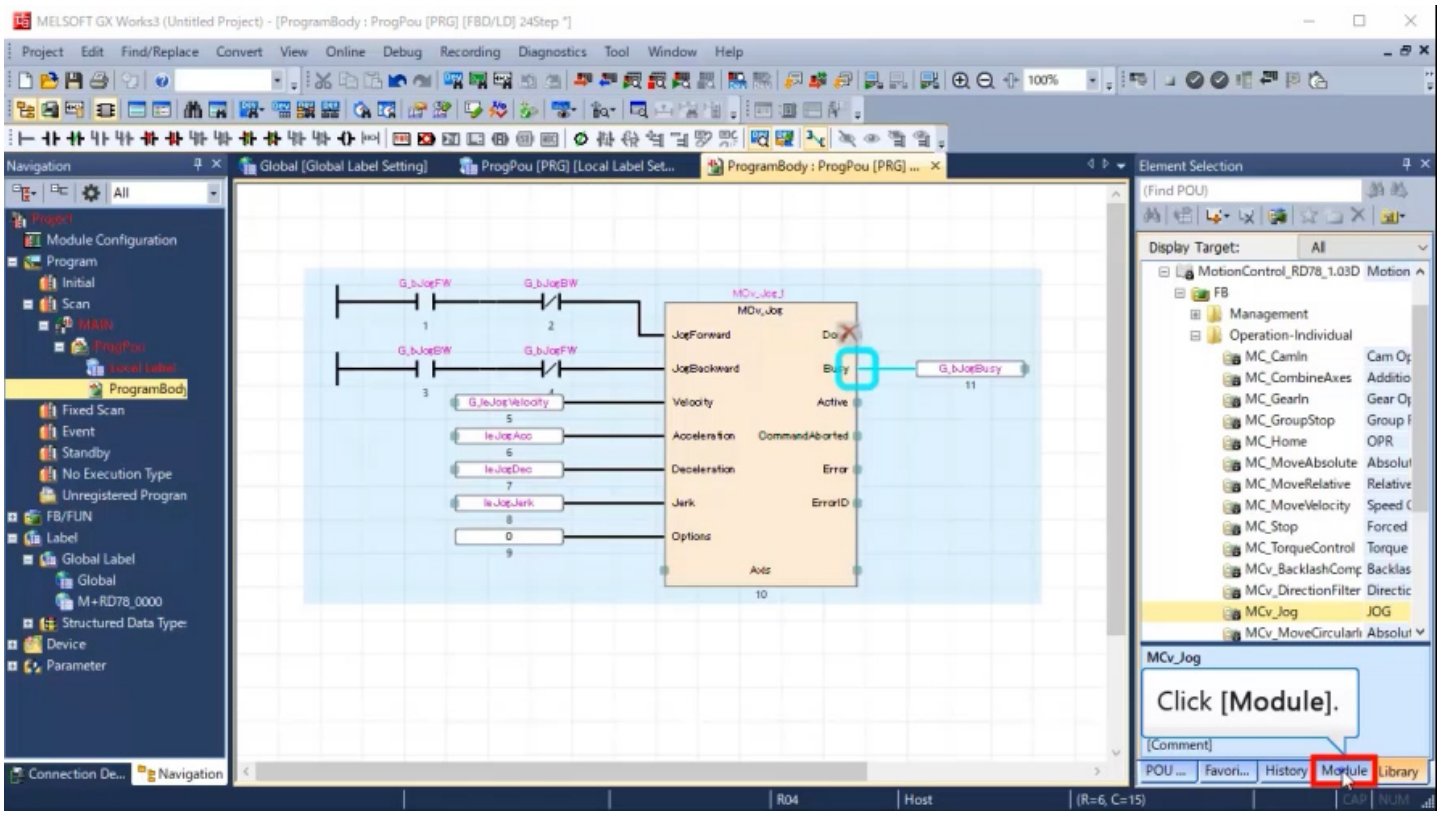
4.3 How to Use the Motion Control FB



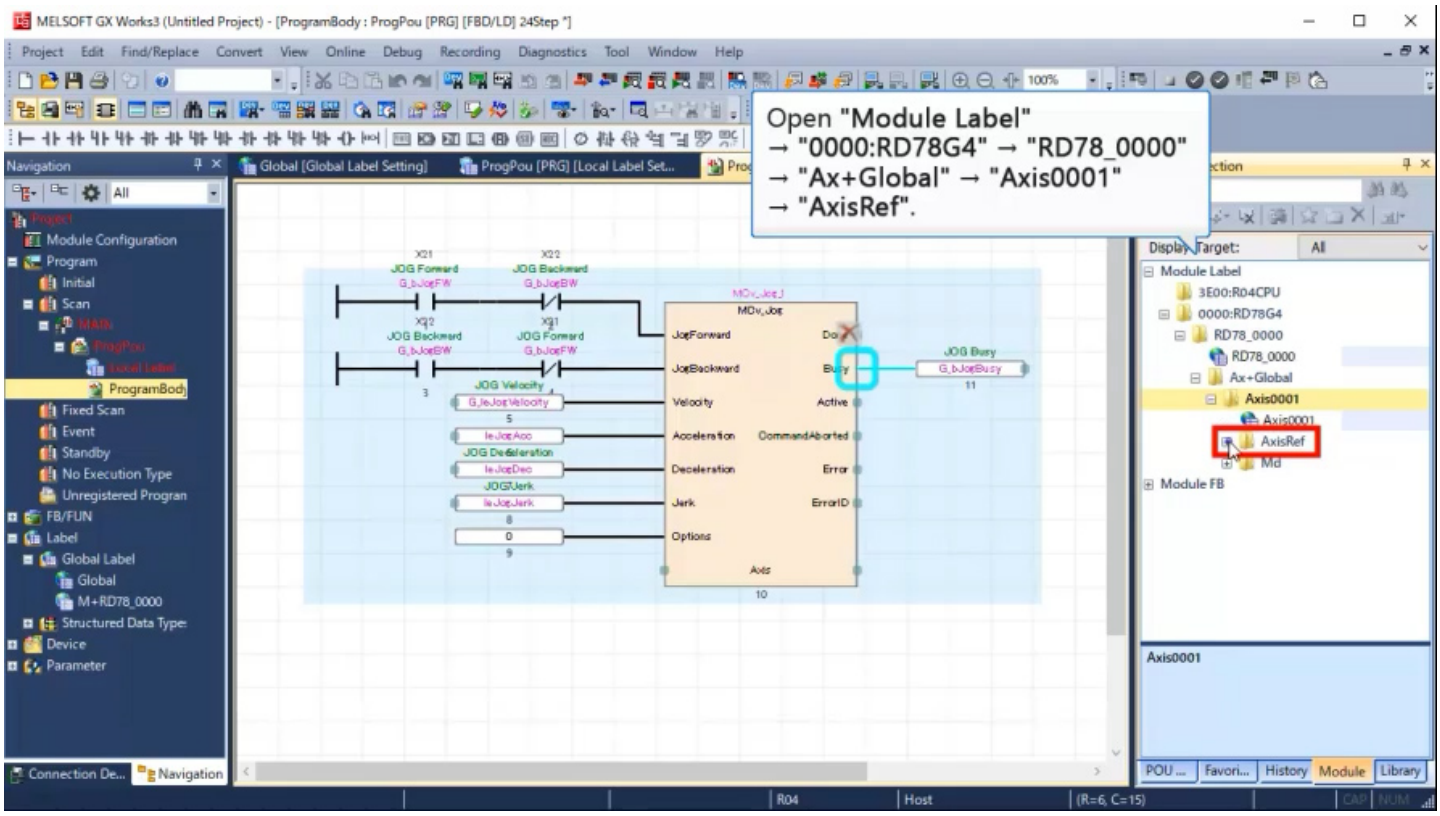
4.3 How to Use the Motion Control FB



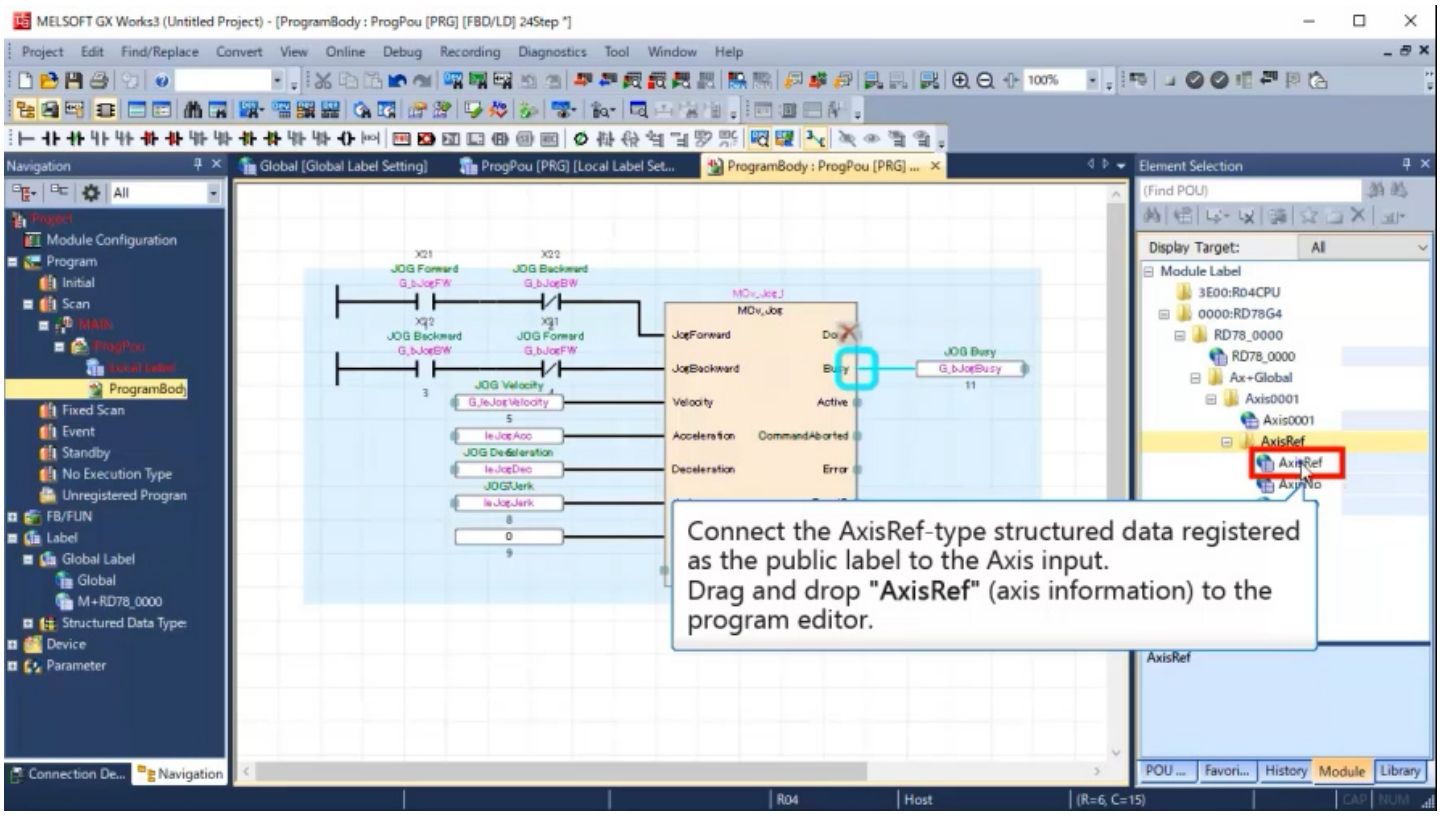
4.3 How to Use the Motion Control FB



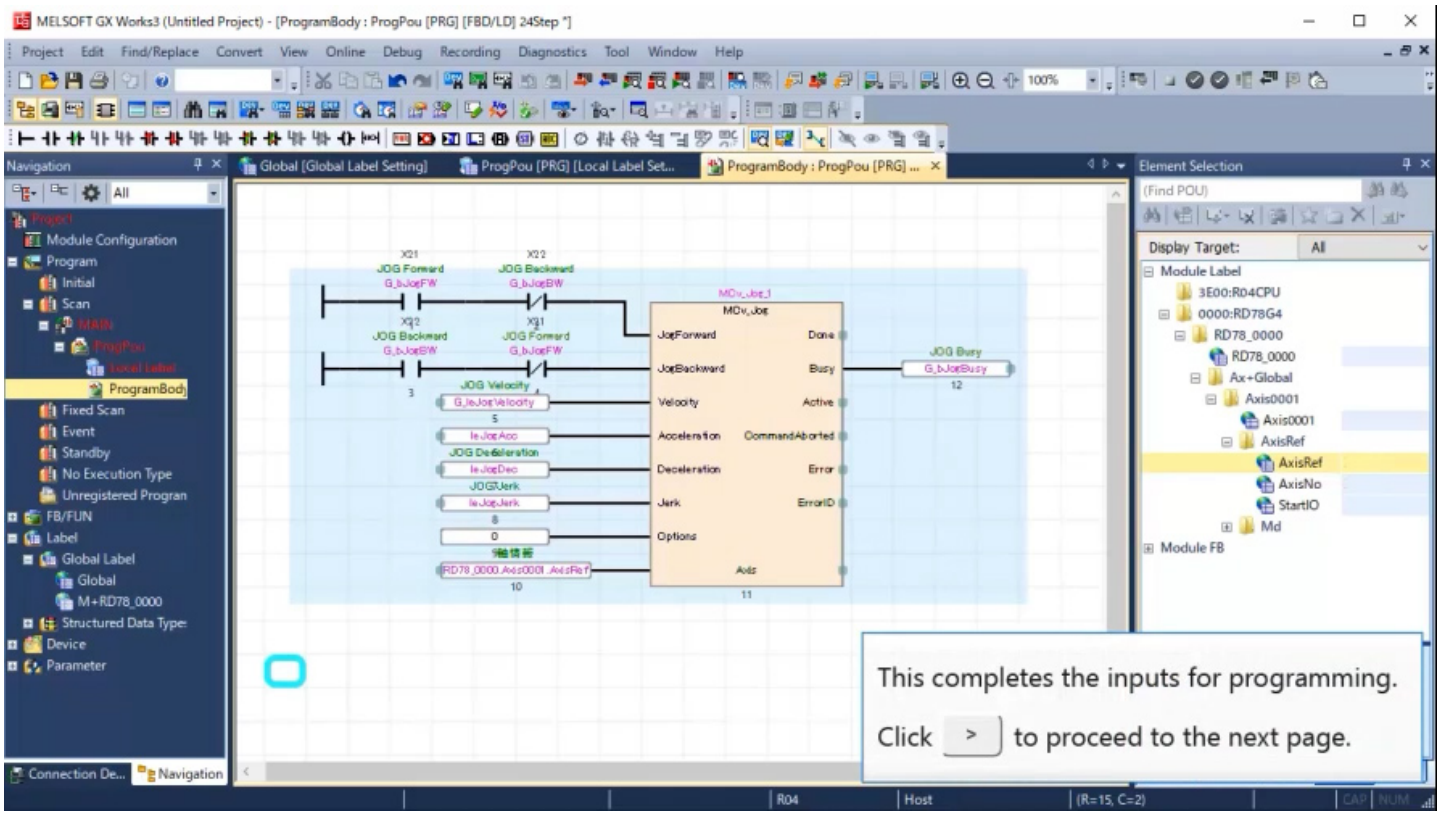
4.3 How to Use the Motion Control FB



4.3 How to Use the Motion Control FB

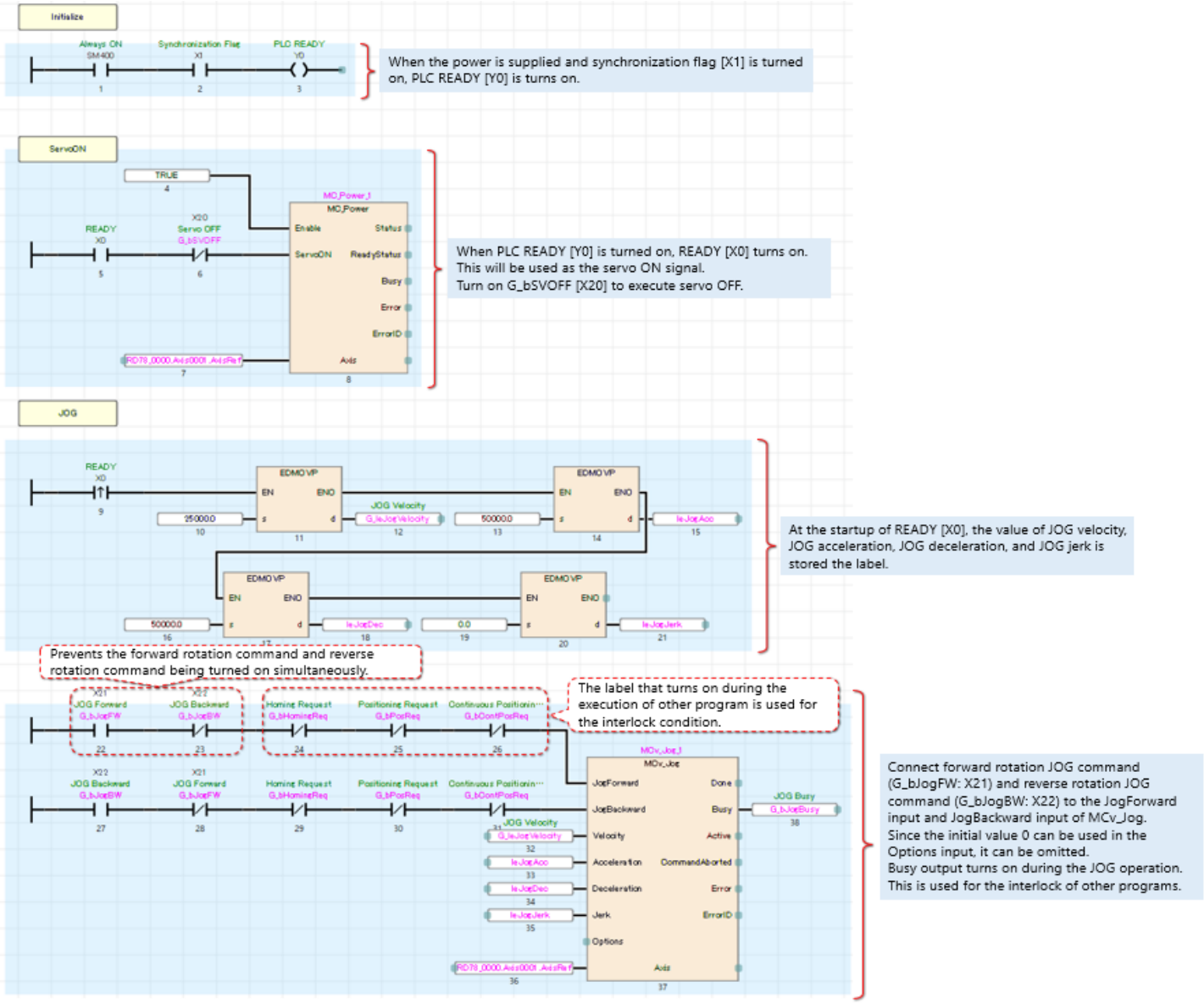


4.3 How to Use the Motion Control FB



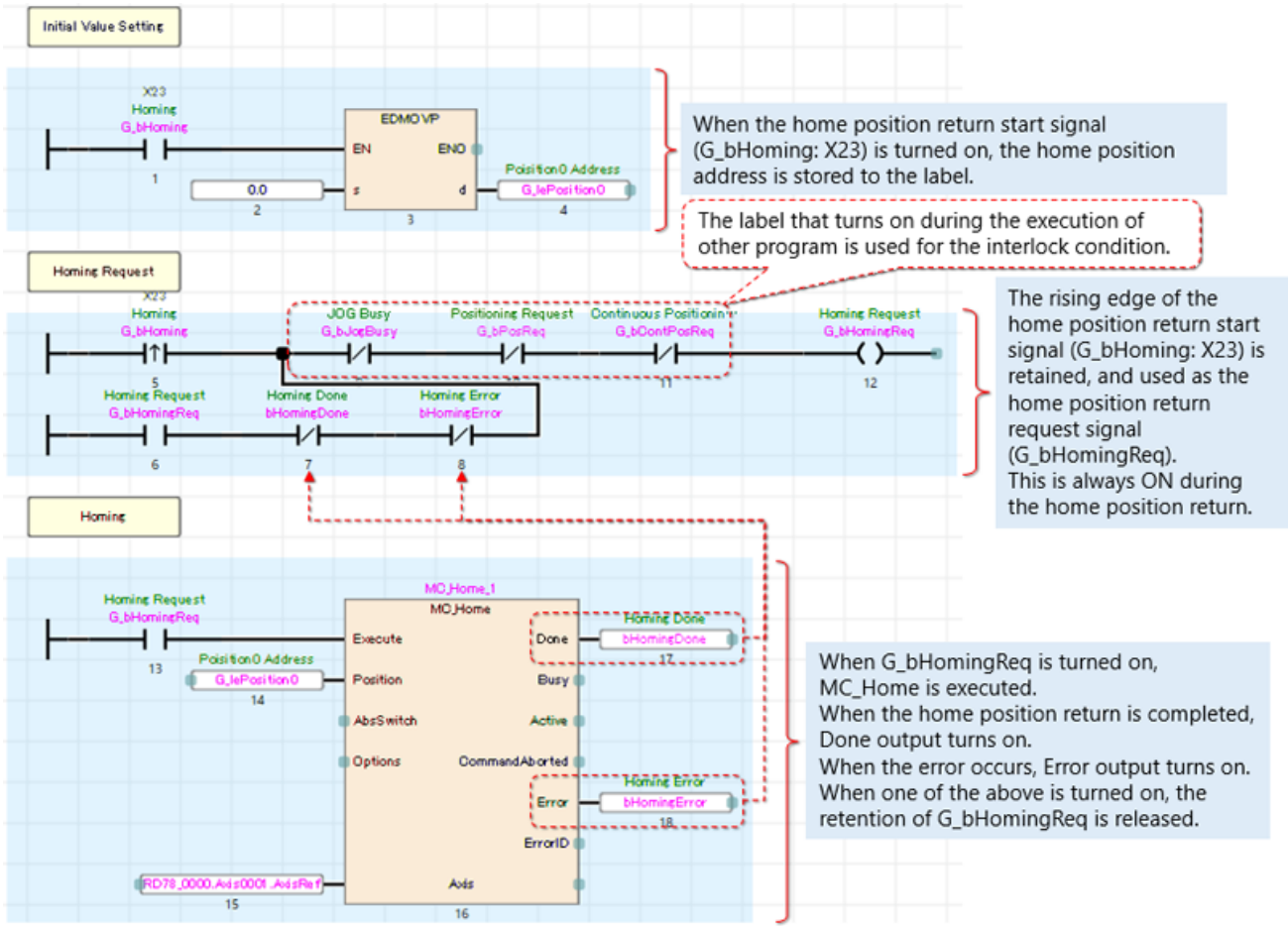
(1) ServoON_Jog

This program executes PLC ready ON, servo ON, and JOG operation.



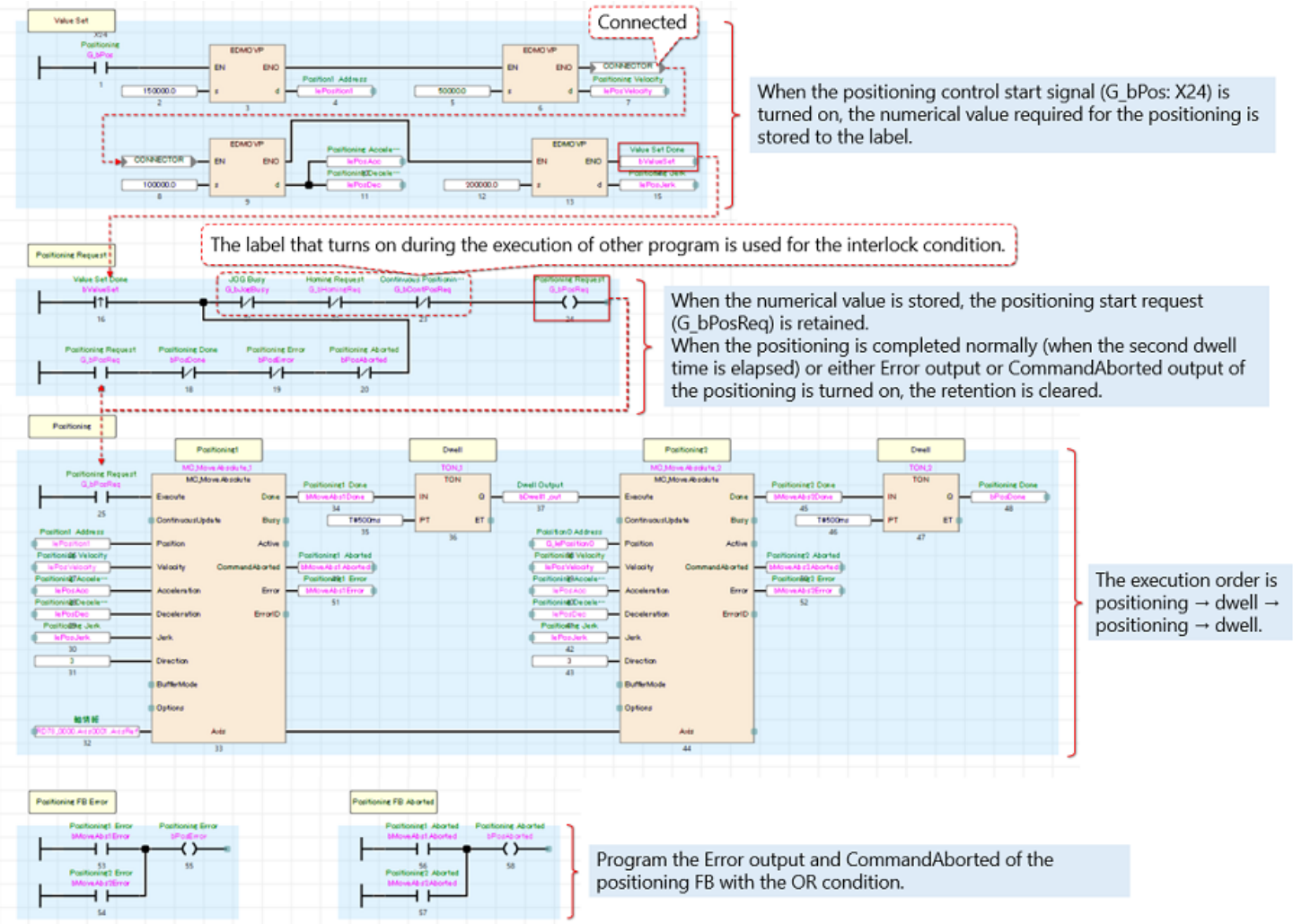
(2) Homing

This program performs the home position return.



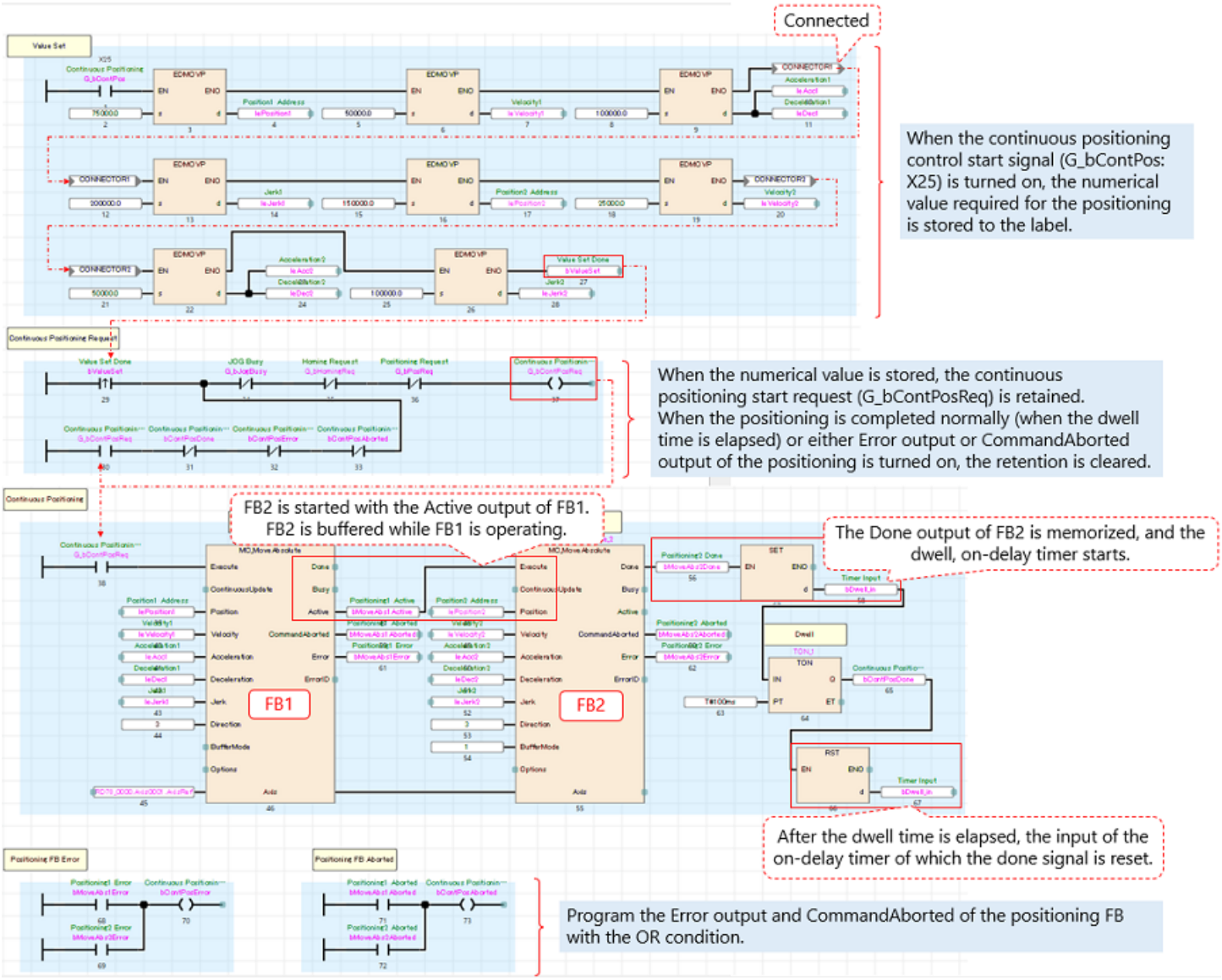
(3) Positioning

This program performs the single axis positioning operation.



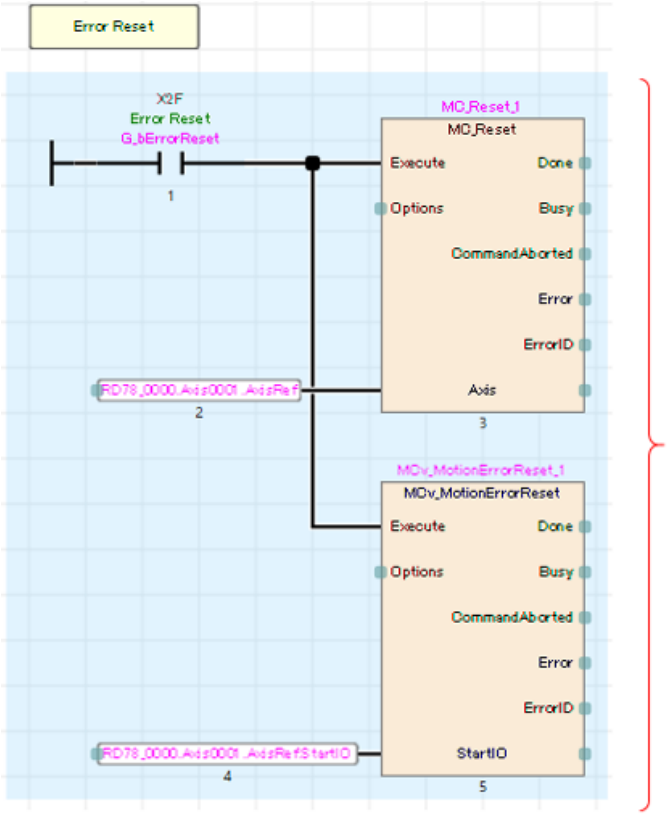
(4) ContinuousPositioning

This program performs continuous positioning by using the buffer mode.



(5) ErrorReset

This program performs the error reset.



When the error reset signal (G_bErrorReset: X2F) is turned on, the axis error reset (MC_Reset) and system error reset (MCv_MotionErrorReset) are executed.

(6) Monitor

This program stores SetPosition (Set Position) and SetVelocity (Set Velocity) of the axis monitor global label assigned to D0 and D2 of the PLC CPU.

Since SetPosition and SetVelocity are the double precision real number type, they are converted to the signed double word type so that they can be easily handled by the PLC CPU. (Note)

These word devices are not used in the subject.

They are used to display on other sequence programs and GOT, and for other purposes. This program is described with ST.

```
1  G_dSetPosition := LREAL_TO_DINT(RD78_0000.Axis0001.Md.SetPosition);
2  G_dSetVelocity := LREAL_TO_DINT(RD78_0000.Axis0001.Md.SetVelocity);
3
```

G_dSetPosition → D0
G_dSetVelocity → D2

(Note) When the double precision real number type is converted to signed double word type, if the value to be converted is outside the range from -2147483648 to 2147483647, a calculation error occurs.

4.5 Writing Program

Write the program and parameters to the PLC CPU and Motion module.
The program is only written to the CPU module. The axis parameter and public label settings are required to be written to the Motion module side.

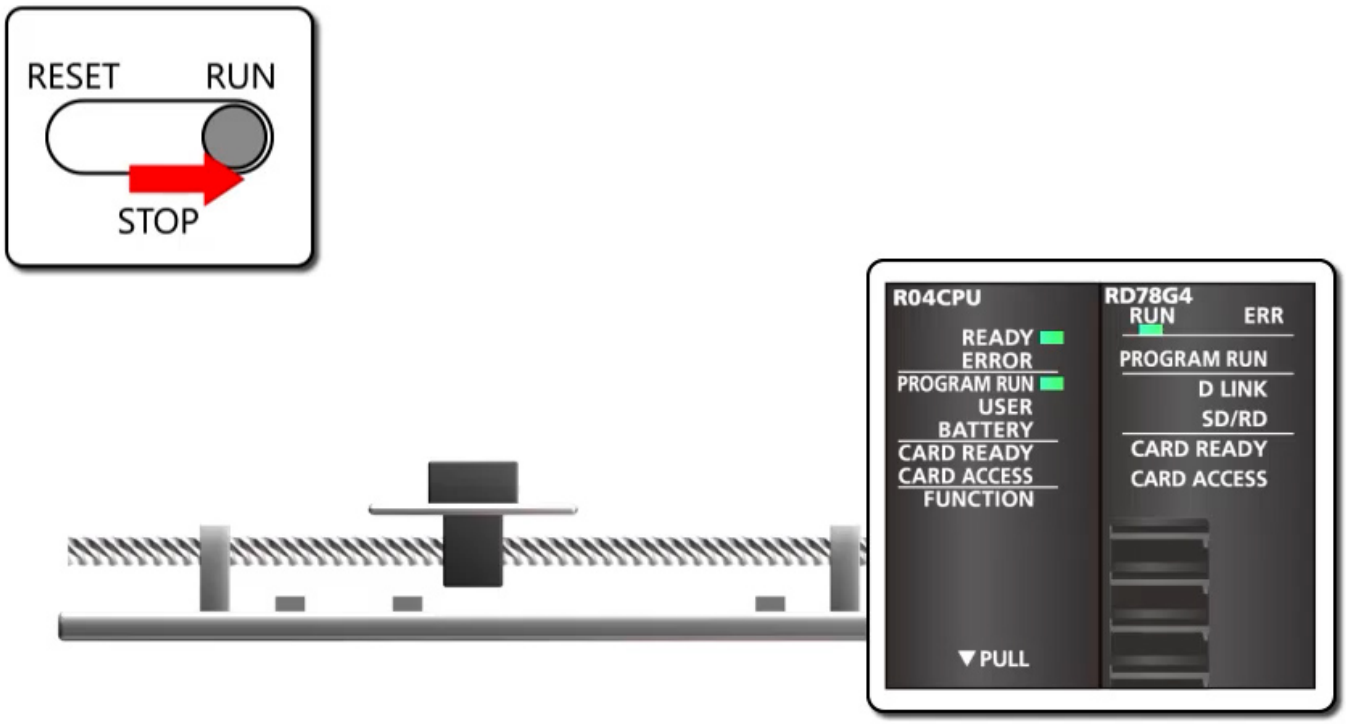
1) After all the programs in the PLC CPU are rebuilt, select [Online] → [Write to PLC] in the tool bar of GX Works3 to write all data to the PLC CPU.

2) When the parameters are written to the PLC CPU, communication with the Motion module is enabled.
   Select [Online]→[Write to Module] in the tool bar of the Motion Control Setting Function to write all data to the Motion module.

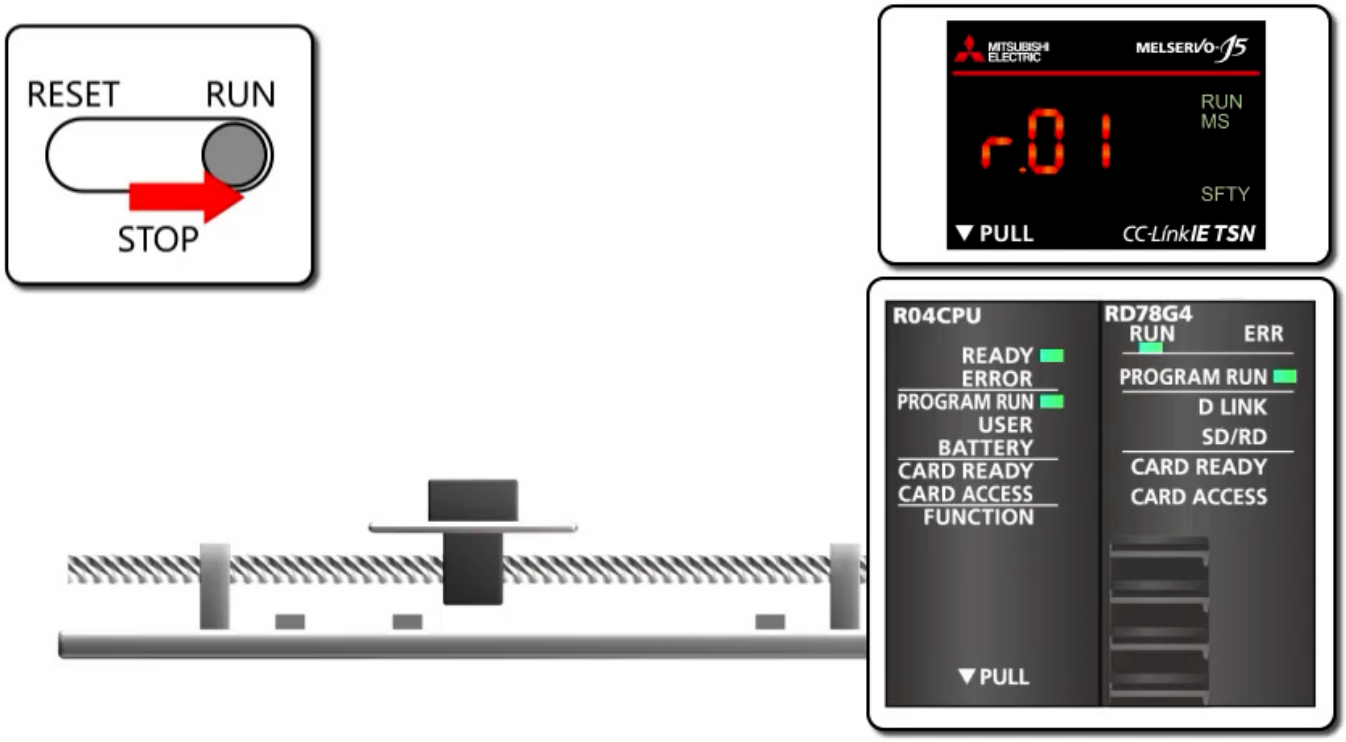
3) Reset the PLC CPU to finish the writing operation.

Click the play button at the lower left of the window.

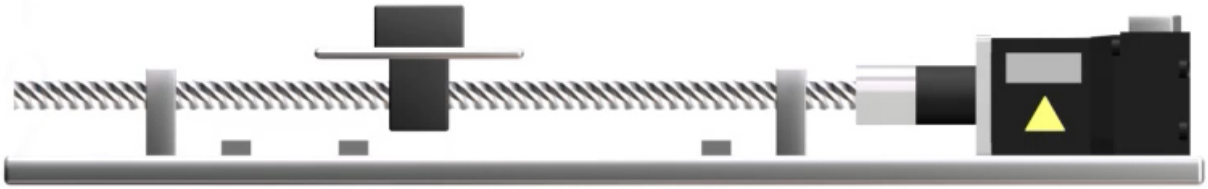
Check the sample program operation.
Before starting operation, make sure that the programs and parameters are written to the PLC CPU and Motion module.



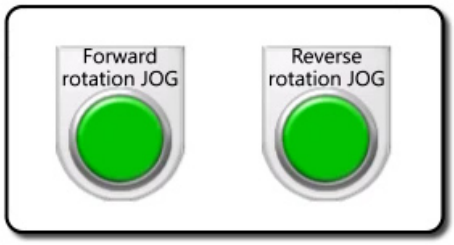
Set the RUN/STOP/RESET switch of the PLC CPU to RUN.
READY lamp and PROGRAM RUN lamp of
the programmable controller turn on.
The RUN lamp of the Motion module turns on.



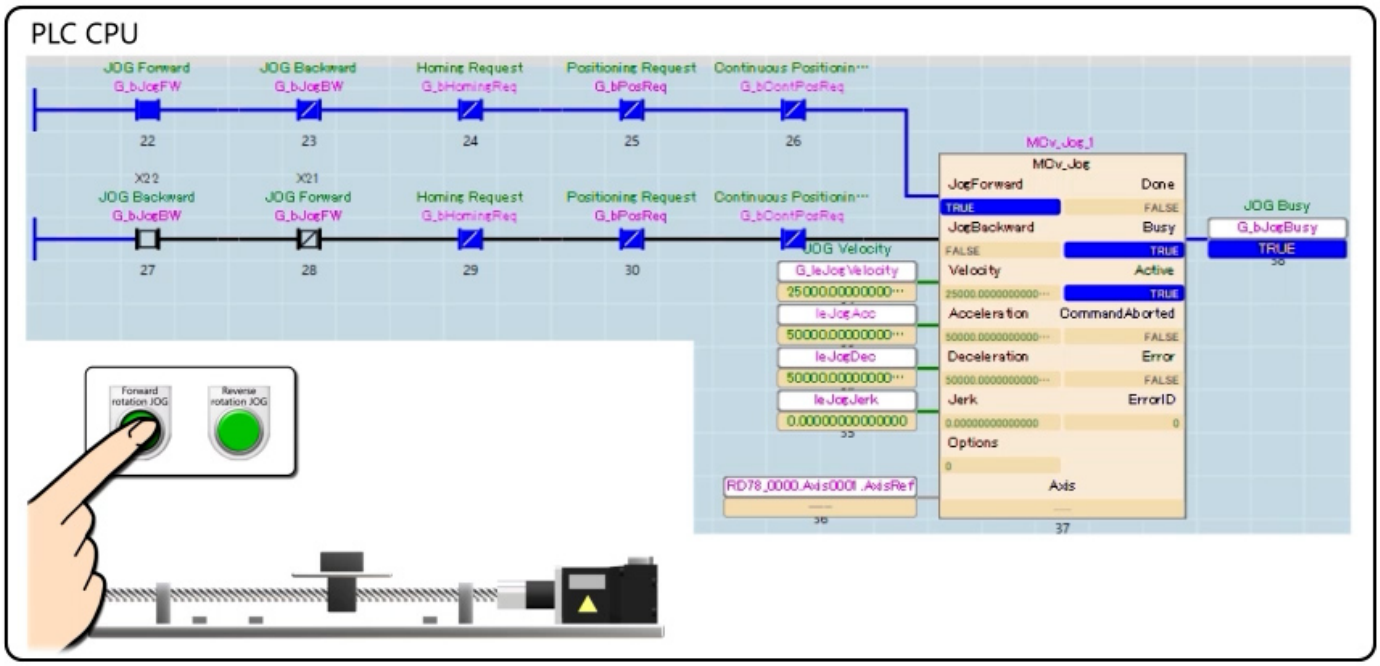
Wait until the PROGRAM RUN lamp of the Motion module turns on.
"r.01" is displayed on the servo amplifier. (The dots are lit.)
The servo motor enters the servo ON state.



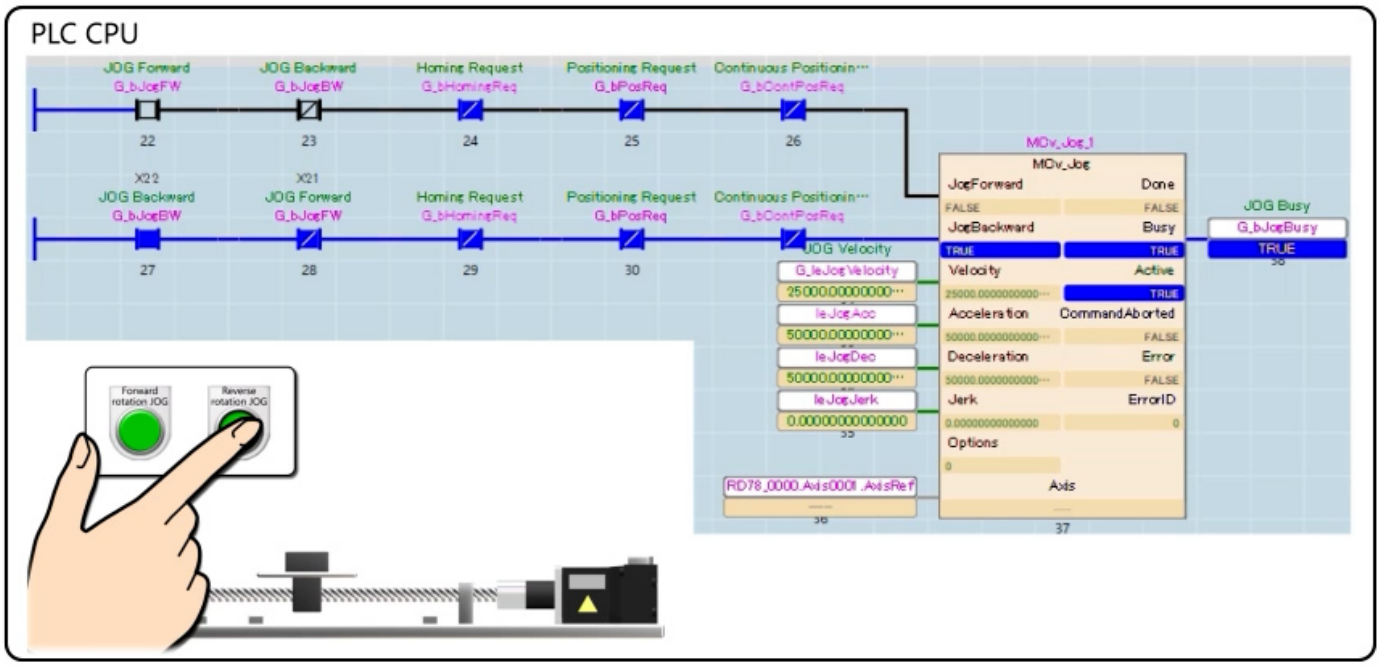
Turn on X20 to execute servo OFF.
"r.01" is displayed on the servo amplifier. (The dots blink.)
Turn off X20 to execute the servo ON again.



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



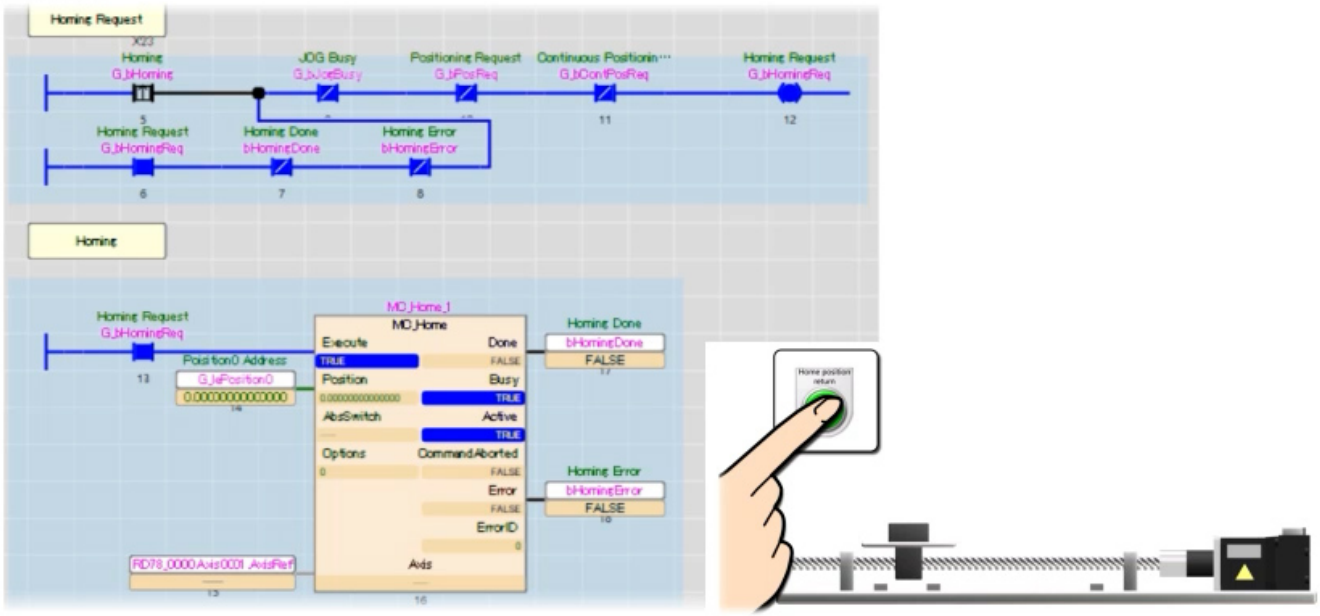
Check the program monitor.
When X21 is turned on, the JogForward input of MCv_Jog_1 turns on.
The normal rotation JOG operation is performed.
Busy output and "G_bJogBusy" turn on during the operation.



When X22 is turned on, the JogBackward input of MCv_Jog_1 turns on.
The reverse rotation JOG operation is performed.
Busy output and "G_bJogBusy" turn on during the operation.

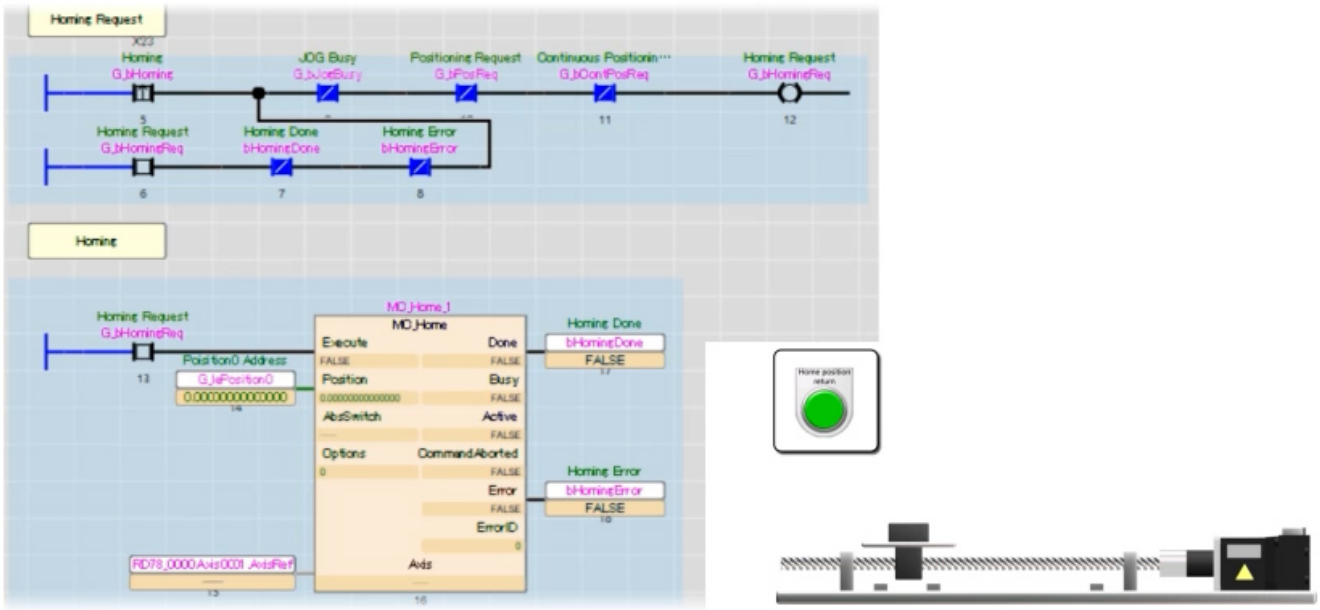
Turn on home position return (X23) to start the home position return.
Execute the home position return with the proximity dog method
(33 is subtracted from Pr.PT45)
The axis stops a little further beyond the dog,
and sets that point as the home position.

PLC CPU



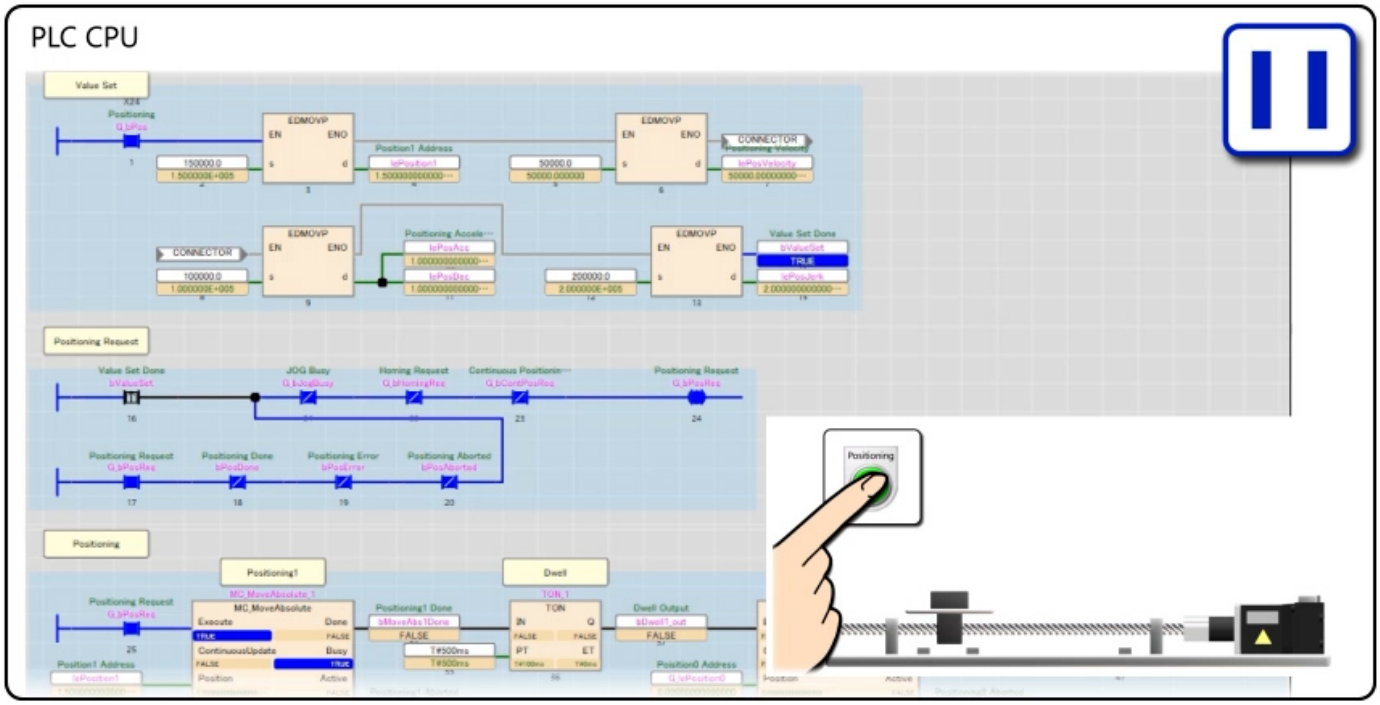
Check the program monitor.
When X23 is turned on, the home position address is stored to the label. "G_bHomingReq", which is the execution command of MC_Home_1, is turned on and retained.

PLC CPU

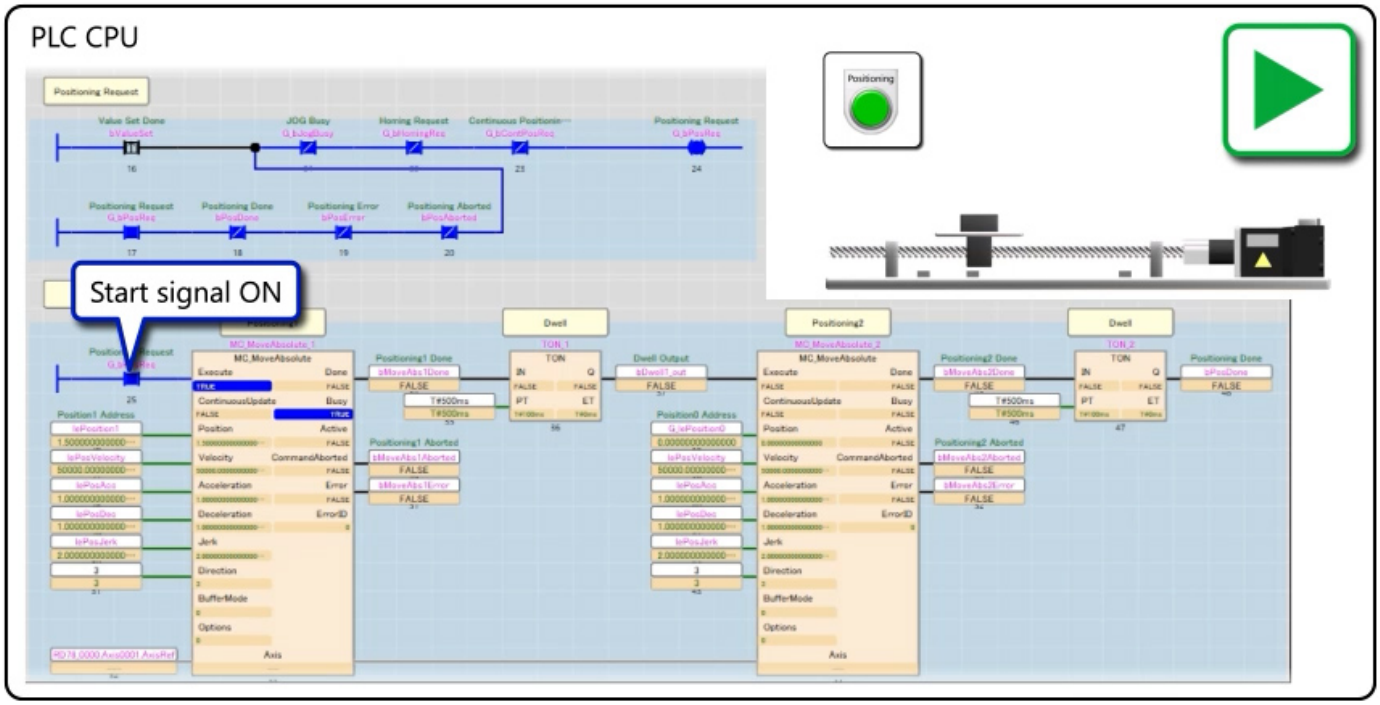


The home position return operation starts.
When the home position return is completed,
the Done output and "bHomingDone" turn on,
and the retention of "G_bHomingReq" is cancelled.

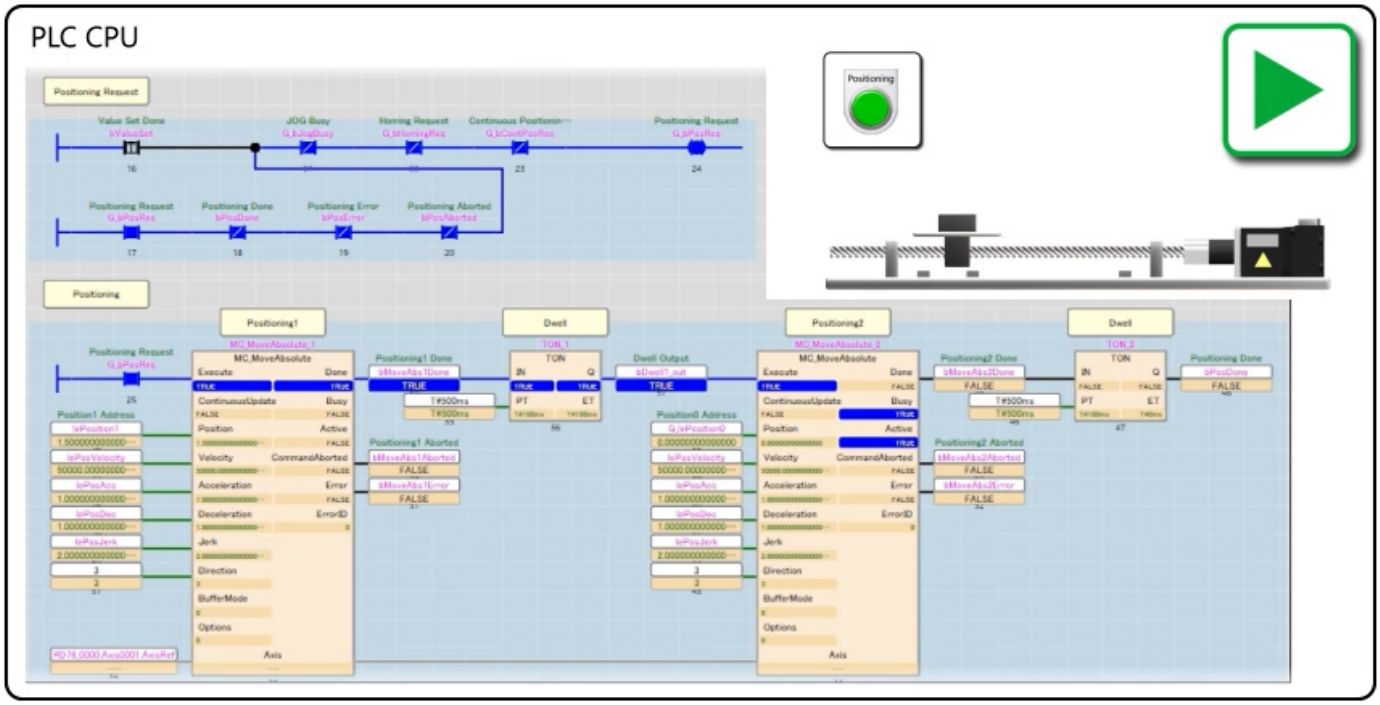
Turning on positioning start (X24) starts reciprocating motion.
The axis moves forward 150 mm and stops for 0.5 seconds,
and moves back 150 mm and stops for 0.5 seconds.



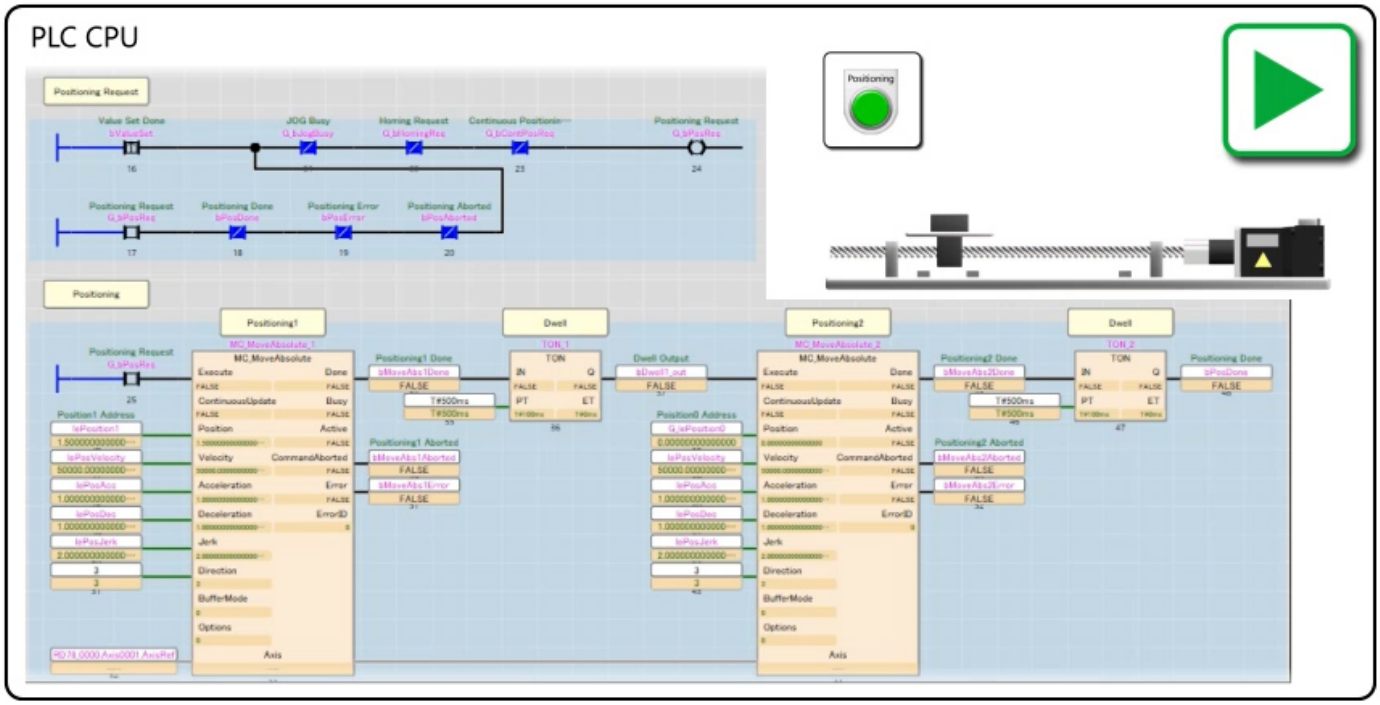
Check the program monitor.
When X24 is turned on, the data for positioning is stored to each label, and "bValueSet" turns on.
"G_bPosReq", which is the execution command of MC_MoveAbsolute_1, is turned on and retained at the rising edge of "bValueSet".



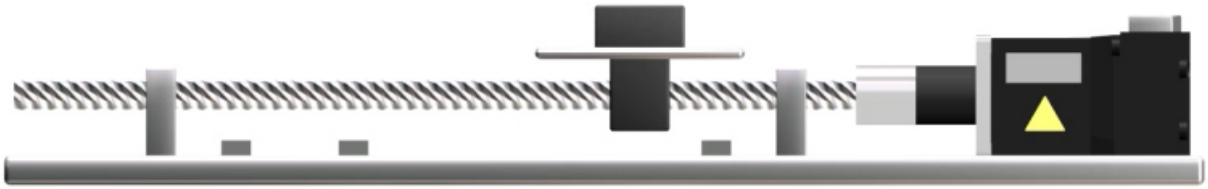
When "G_bPosReq" is turned on, MC_MoveAbsolute_1 starts and the servo motor starts running.



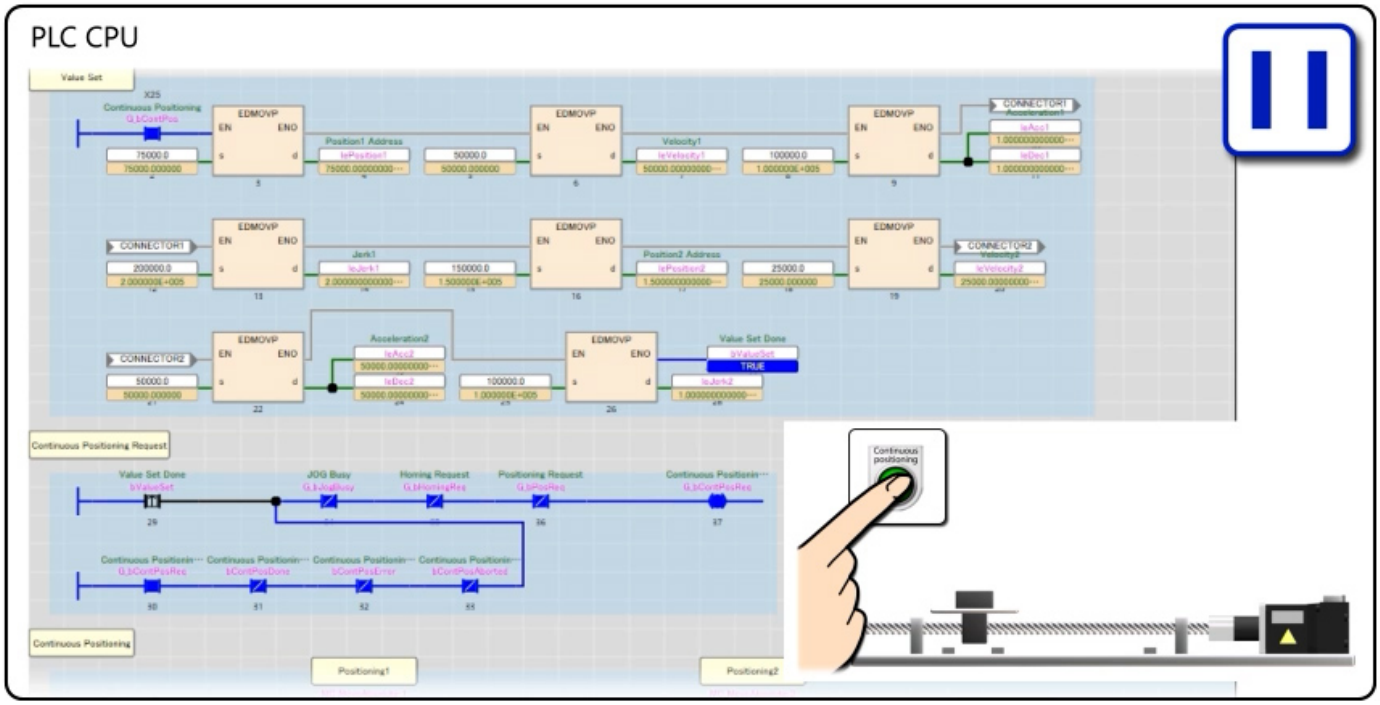
When the positioning by MC_MoveAbsolute_1 is completed, TON_1, which is the dwell, operates.
When 500 ms is elapsed, MC_MoveAbsolute_2 is executed and the servo motor starts running.



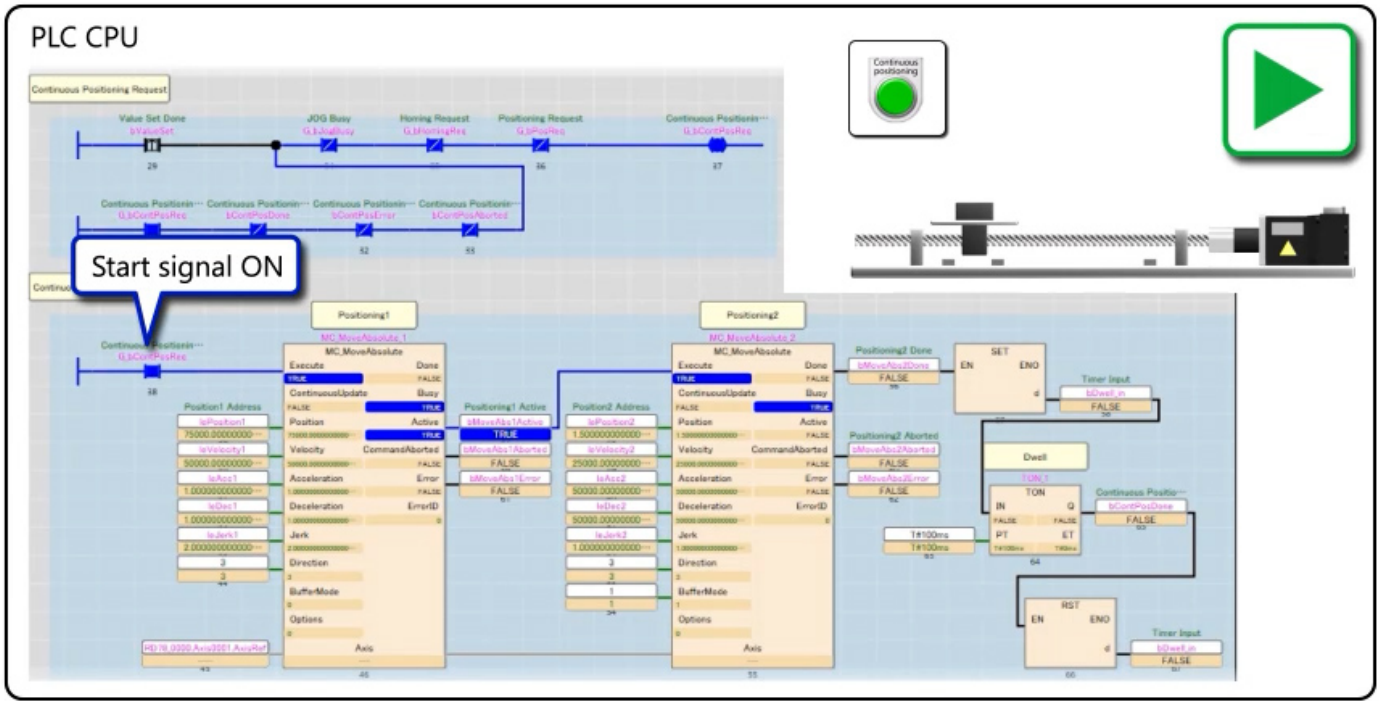
When the positioning by MC_MoveAbsolute_2 is completed, TON_2, which is the dwell, operates.
When 500 ms is elapsed, the retention of "G_bPosReq" is cleared, and reset to the initial state.



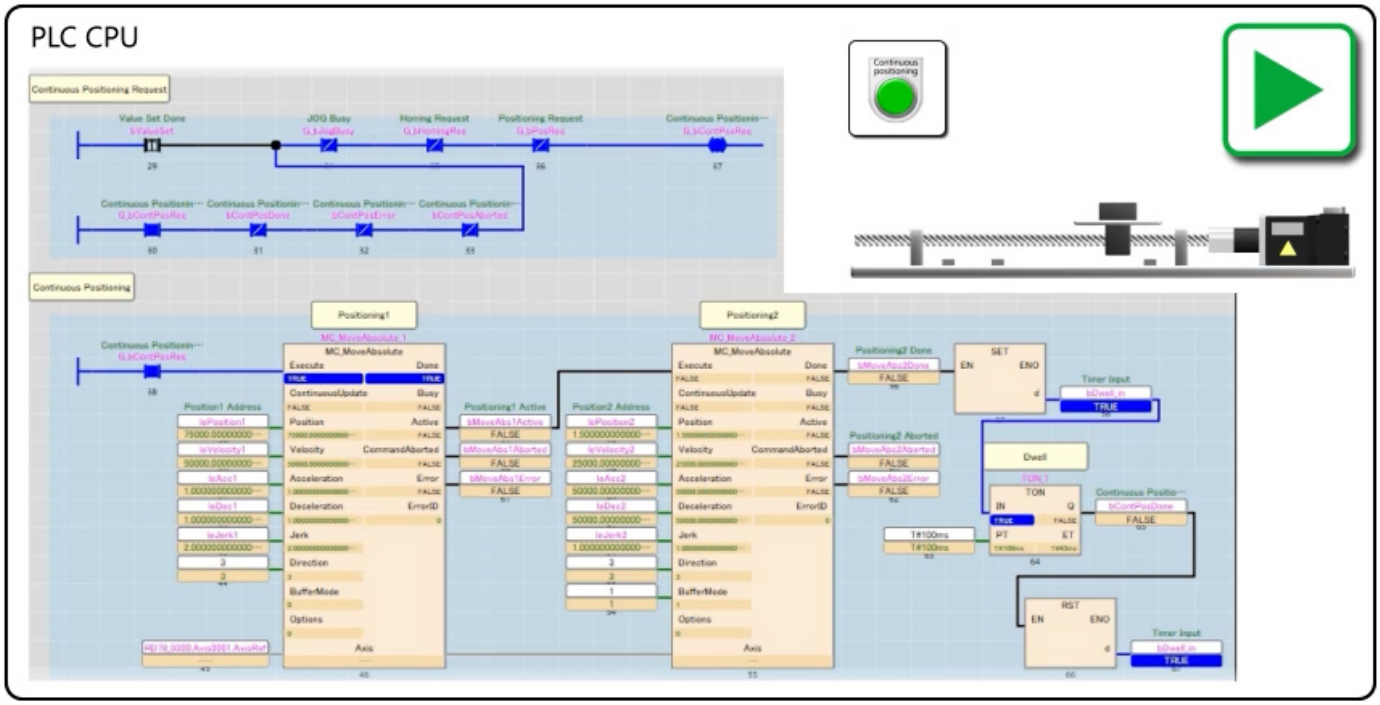
Turn on continuous positioning start (X25) to start the operation of the buffer mode (mc_Buffered).



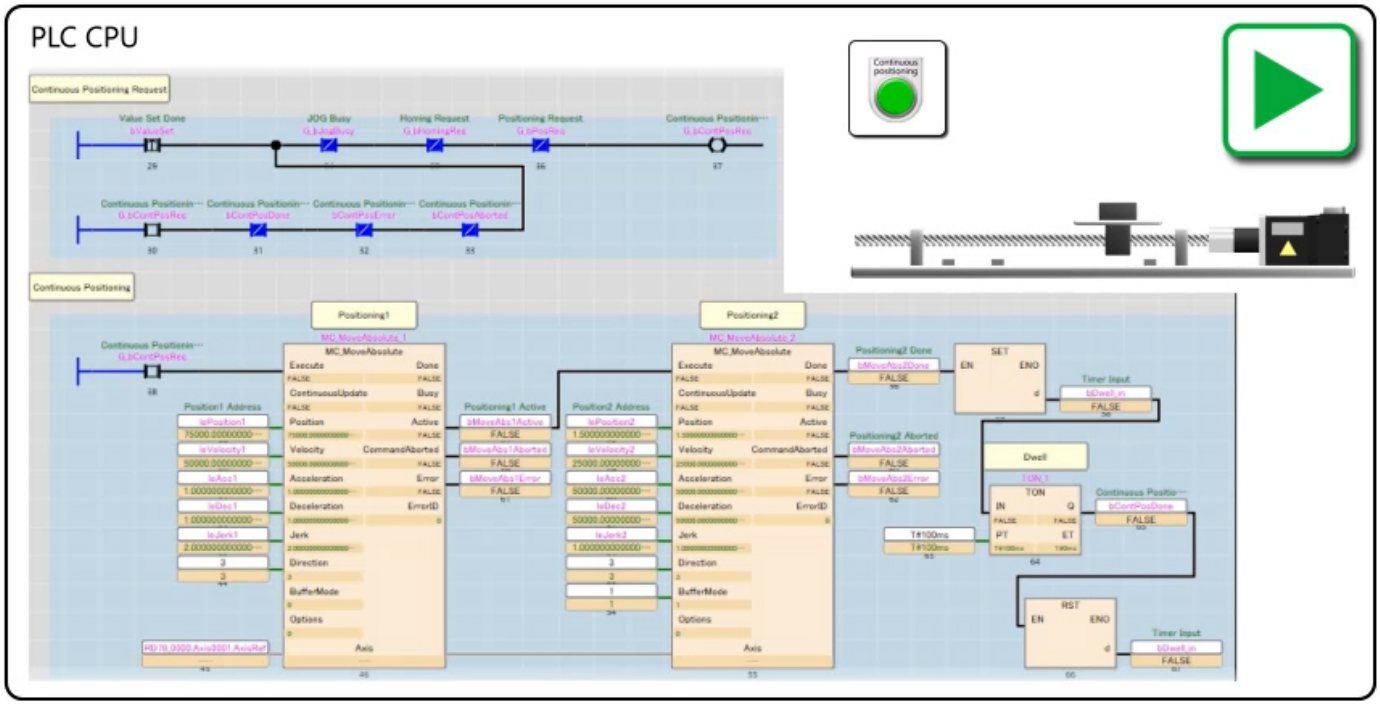
Check the program monitor.
When X25 is turned on, the data for positioning is stored to each label, and "bValueSet" turns on.
"G_bContPosReq", which is the execution command of MC_MoAbsolute_1, is turned on and retained at the rising edge of "bValueSet".



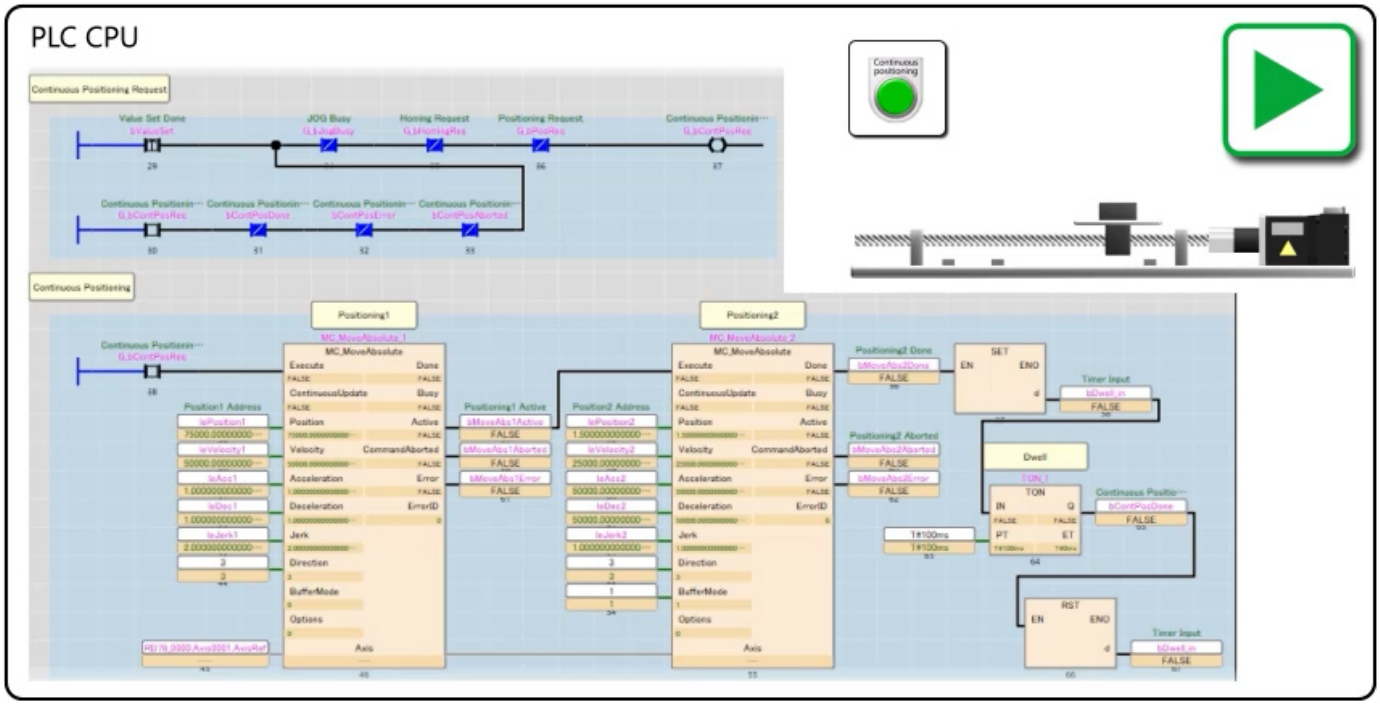
When "G_bContPosReq" is turned on, MC_MoveAbsolute_1 starts and the servo motor starts running.
At this time, since the Active output is the execution command of MC_MoveAbosolute_2, MC_MoveAbsolute_2 is buffered.



When the operation of MC_MoveAbosolute_1 is completed, the buffered C_MoveAbsolute_2 is executed.
When the operation of MC_MoveAbosolute_2 is completed, TON_1, which is the dwell, is executed.



When 100 ms is elapsed, the retention of "G_bContPosReq" is cleared, and reset to the initial state.



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

4.7 Summary of This Chapter

In this chapter, you have learned:

- Registering the Motion Module FB Library
- Creating Projects
- How to Use the Motion Control FB
- Sample Program Description
- Sample Program Operation Check

Point

Registering the Motion Module FB Library	• The FB library must be registered to GX Works3 to use the Motion control FB in the PLC CPU.
Creating Projects	• Configure axis parameters and other settings as when programming to the Motion module.
How to Use the Motion Control FB	• The Motion control FB can be placed to the program editor by dragging and dropping it from the Library tab of Element Selection window of GX Works3. • Connect the contact and label to the input/output of the FB.
Sample Program Description	• You have created a program similar to the sample programs in chapter 2 and chapter 3 by using only the PLC CPU.
Sample Program Operation Check	• You have checked the operation of the sample program in the video.

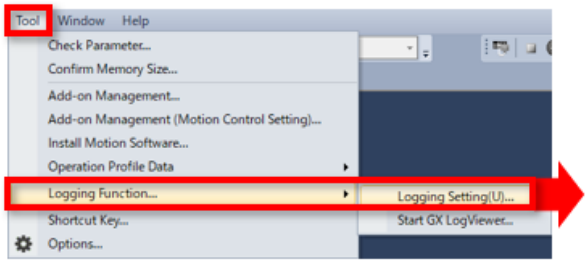
Chapter 5 Logging

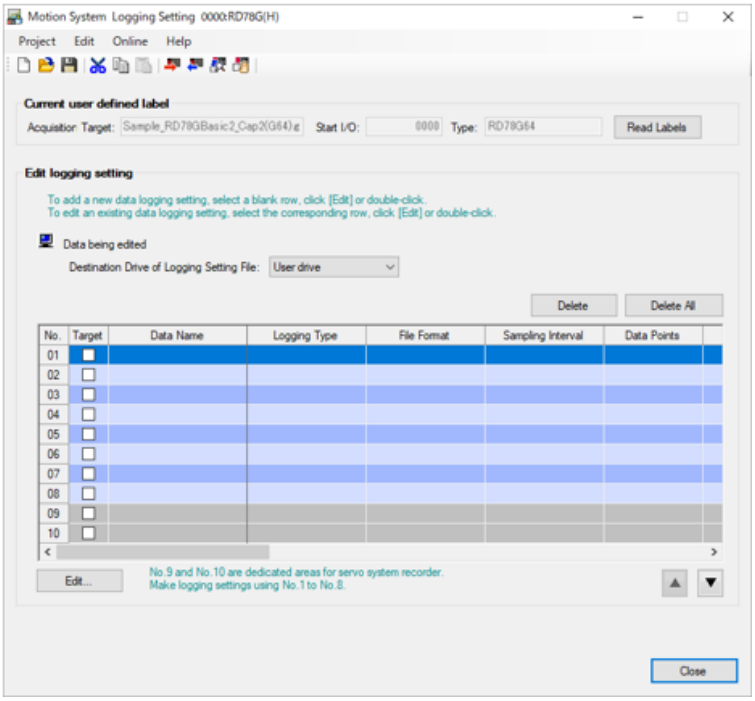
This chapter describes how to log data of the Motion module and display it in the graph.
In this course, the positioning start program of the sample program in chapter 2 and chapter 3 will be logged as an example.

(Note) The program in chapter 4 cannot be logged with the procedure described in this chapter.
"CPU Module Logging Configuration Tool" must be used.

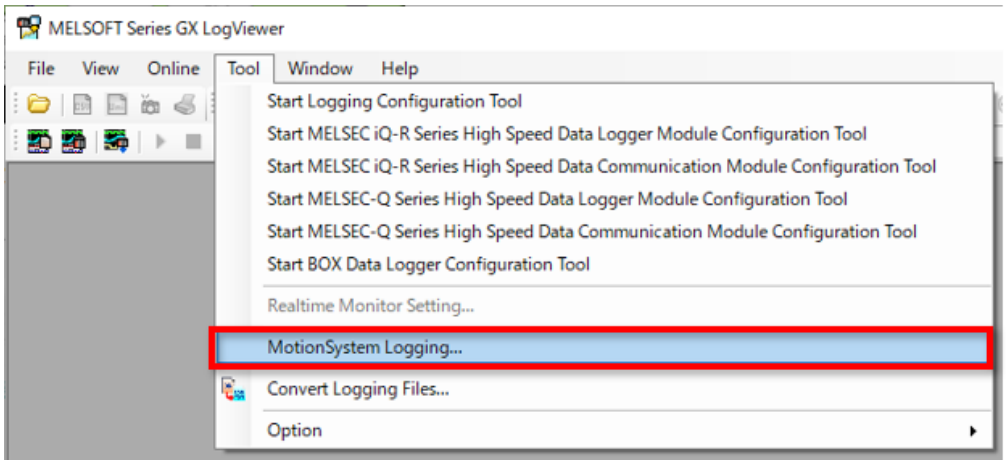
5.1 Starting the Logging Configuration Tool

Select [Tool] → [Logging Function] → [Logging Setting] from the tool bar of the Motion Control Setting Function screen.
The motion system logging setting tool starts.



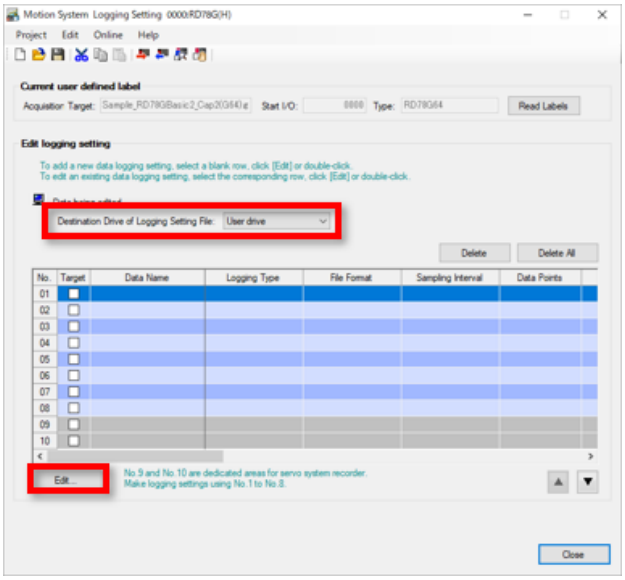


[Point]
The motion system logging setting tool can be started from [Tool] → [MotionSystem Logging] in GX LogViewer.

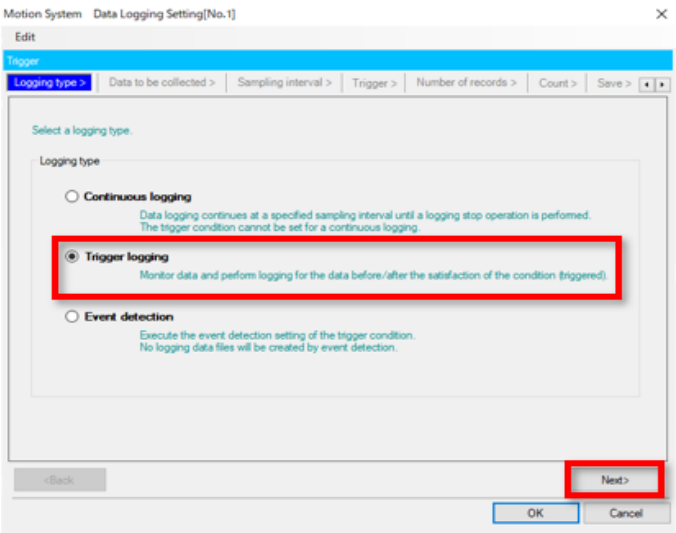


5.2 Setting the Data to be Logged

(1) In the edit logging setting field of the motion system logging setting tool, set the save destination of the logging data. Then, click the [Edit] button.
The Data Logging Setting screen is displayed.



(2) Select [Logging type] from continuous logging, trigger logging, and event detection.
In this course, the details of the trigger logging is described.
Select trigger logging, and click the [Next] button.

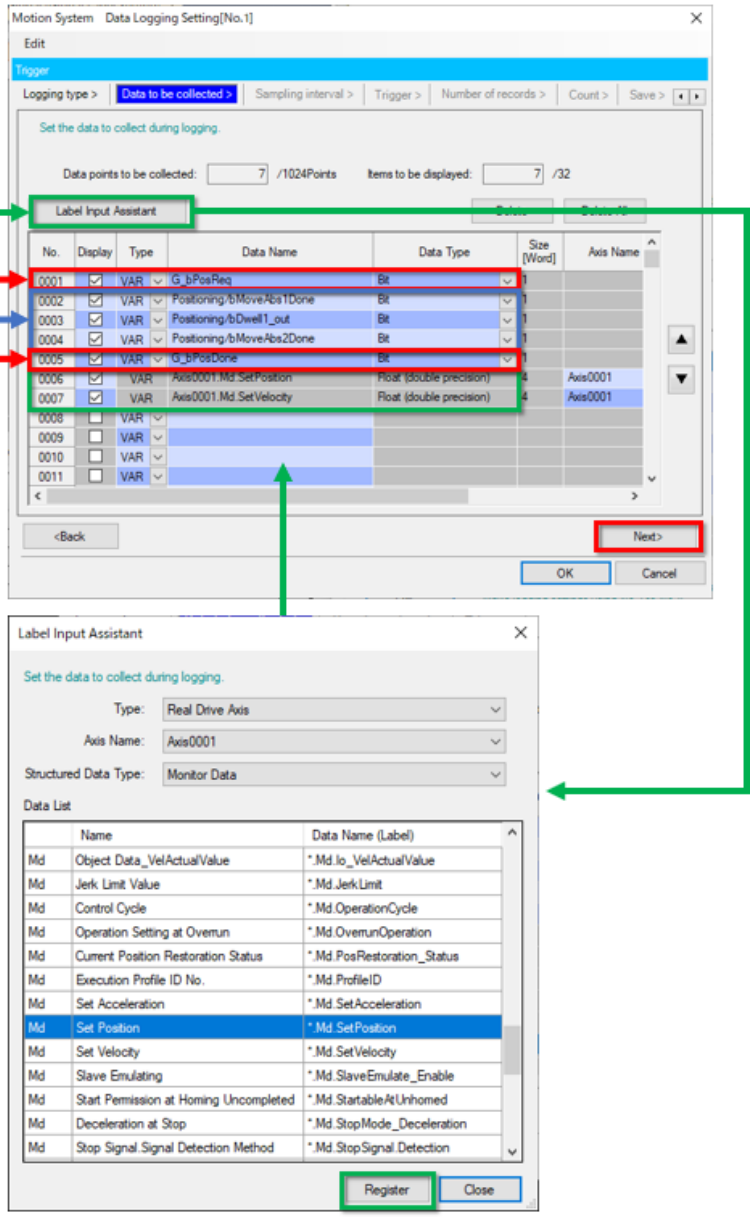


(3) The label of the data to be logged is registered in [Data to be collected].

1) Global label
Enter the global label name to the data name field.
In the data type field, select the data type of the label.

2) Local label
Enter the data name in "program name/local label name" format.
In the data type field, select the data type of the label.

3) Structured data type
Click the [Label Input Assistant] button and select the member of the structured data type from the list.
Select it from the list and click the [Register] button to reflect to the data to be collected.



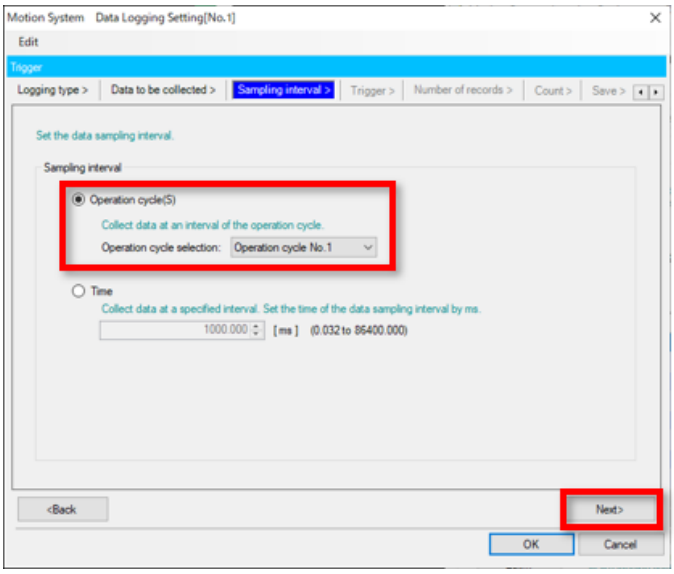
In this course, the following data is logged as example.

Data name
G_bPosReq
Positioning/bMoveAbs1Done
Positioning/bDwell1_out
Positioning/bMoveAbs2Done
G_bPosDone
Axis0001.Md.SetPosition
Axis0001.Md.SetVelocity

Click the [Next] button when the registration is completed.

(4) Set the sampling interval in [Sampling Interval].
In this course, use the operation cycle No. 1 for sampling.

After selecting the sampling interval, click the [Next] button.



(5) The condition to start the logging is set in [Trigger].
In this course, the startup the bit that is the positioning start signal is used as the trigger.

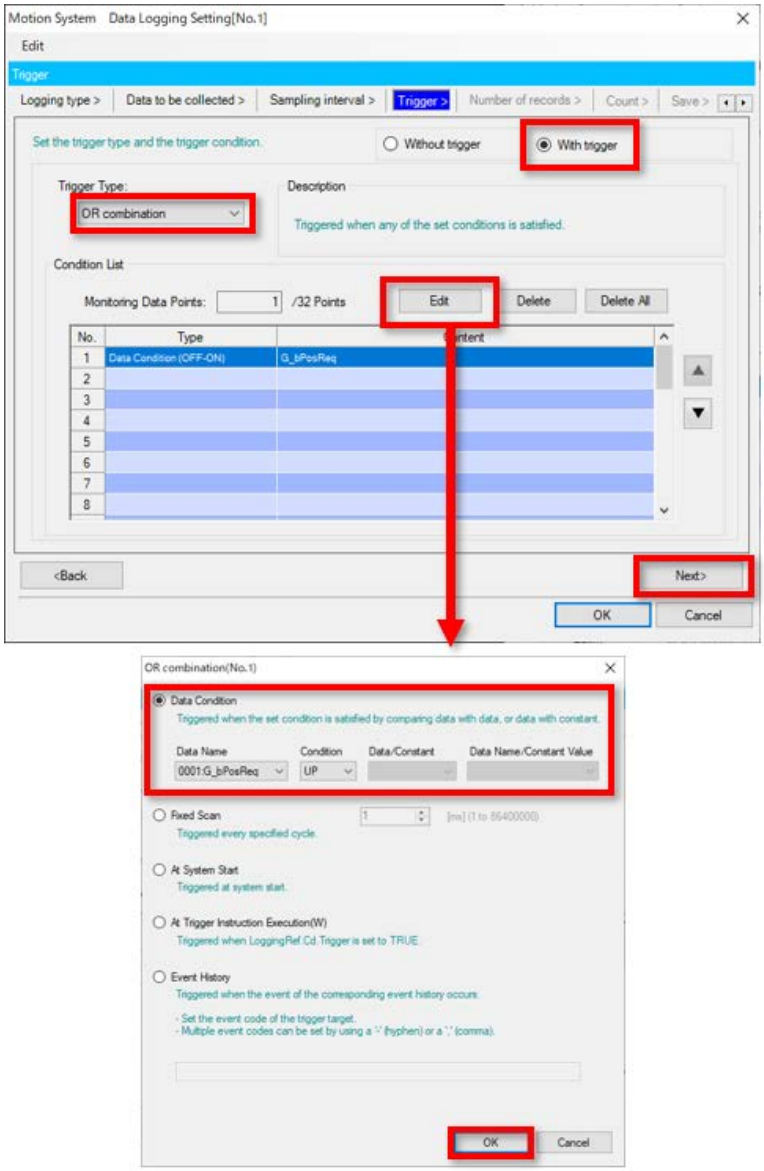
1) Select [With trigger].

2) Select "OR combination" for the trigger type.

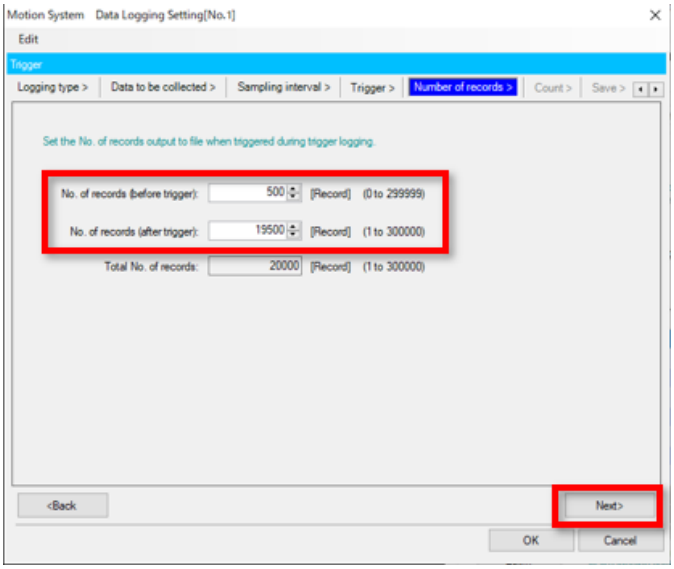
3) Select No. 1 of the condition list, and click the edit button. The sub window appears.

4) Select "Data Condition", and select "0001:G_bPosReq" for the data name.
Select "UP" for the condition.
When the selection is completed, click the [OK] button.

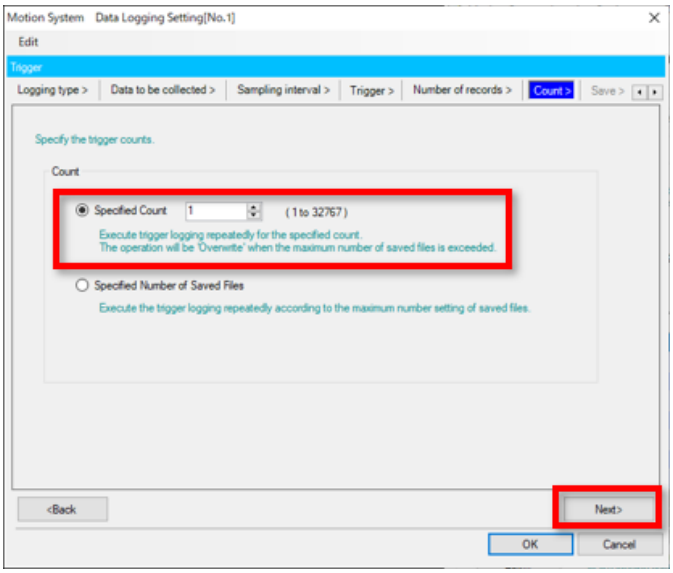
5) After you return to the original screen, click the [Next] button.



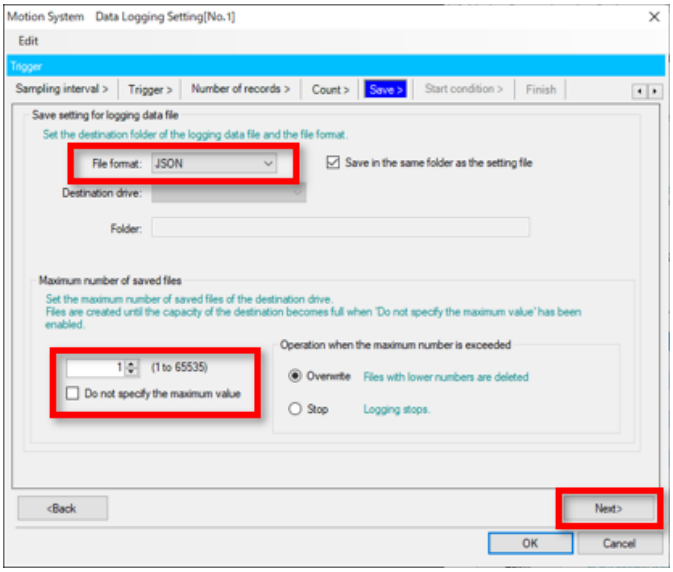
(6) The number of sampling points is set in [Number of records].
In this course, No. of records (before trigger) is set to "500", and No. of records (after trigger) to "19500".
When the setting is completed, click the [Next] button.



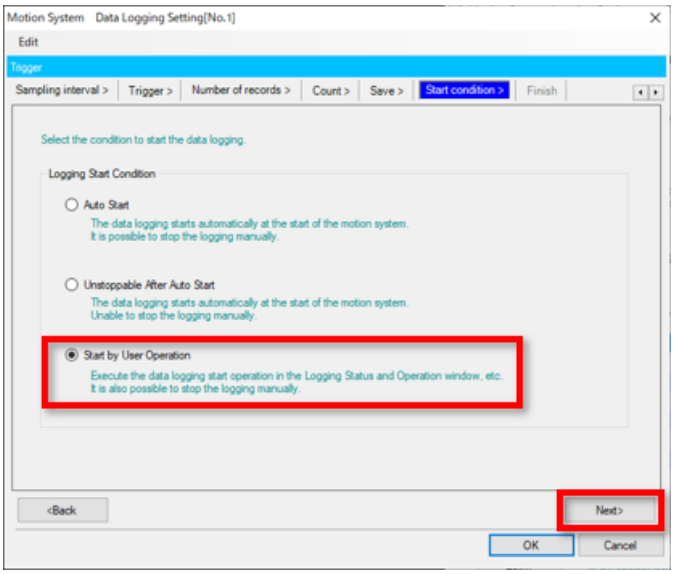
(7) The logging count is set in [Count]. In this course, the count is set to 1.
When the setting is completed, click the [Next] button.



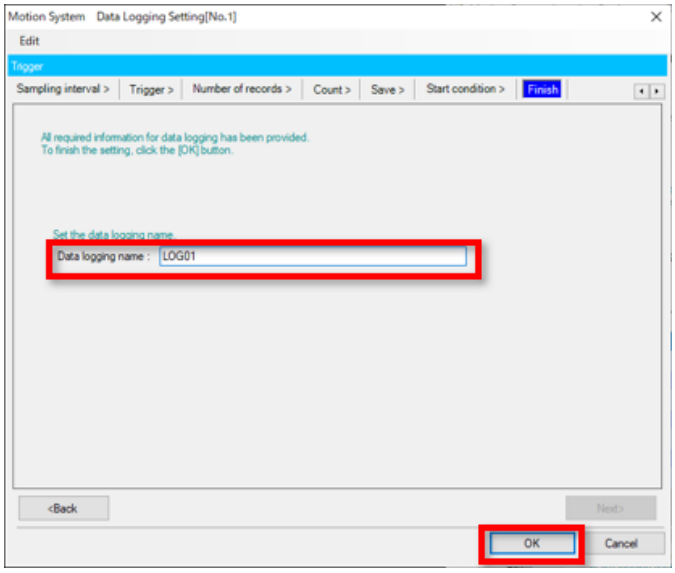
(8) The file format and number of saved files of the logging data is set in [Save].
In this course, the default value (format: JSON, number of saved files: 1) is set.
When the setting is completed, click the [Next] button.



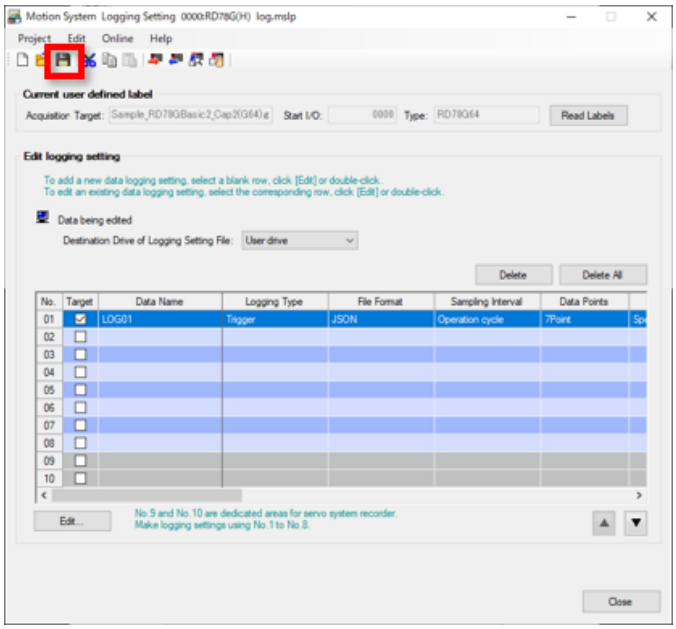
(9) The condition to start the logging is set in [Start condition].
In this course, "Start by User Operation" is set.
When the setting is completed, click the [Next] button.



(10) The data logging name is set in [Finish].
In this course, the default value (LOG01) is set.
When the setting is completed, click the [Next] button.

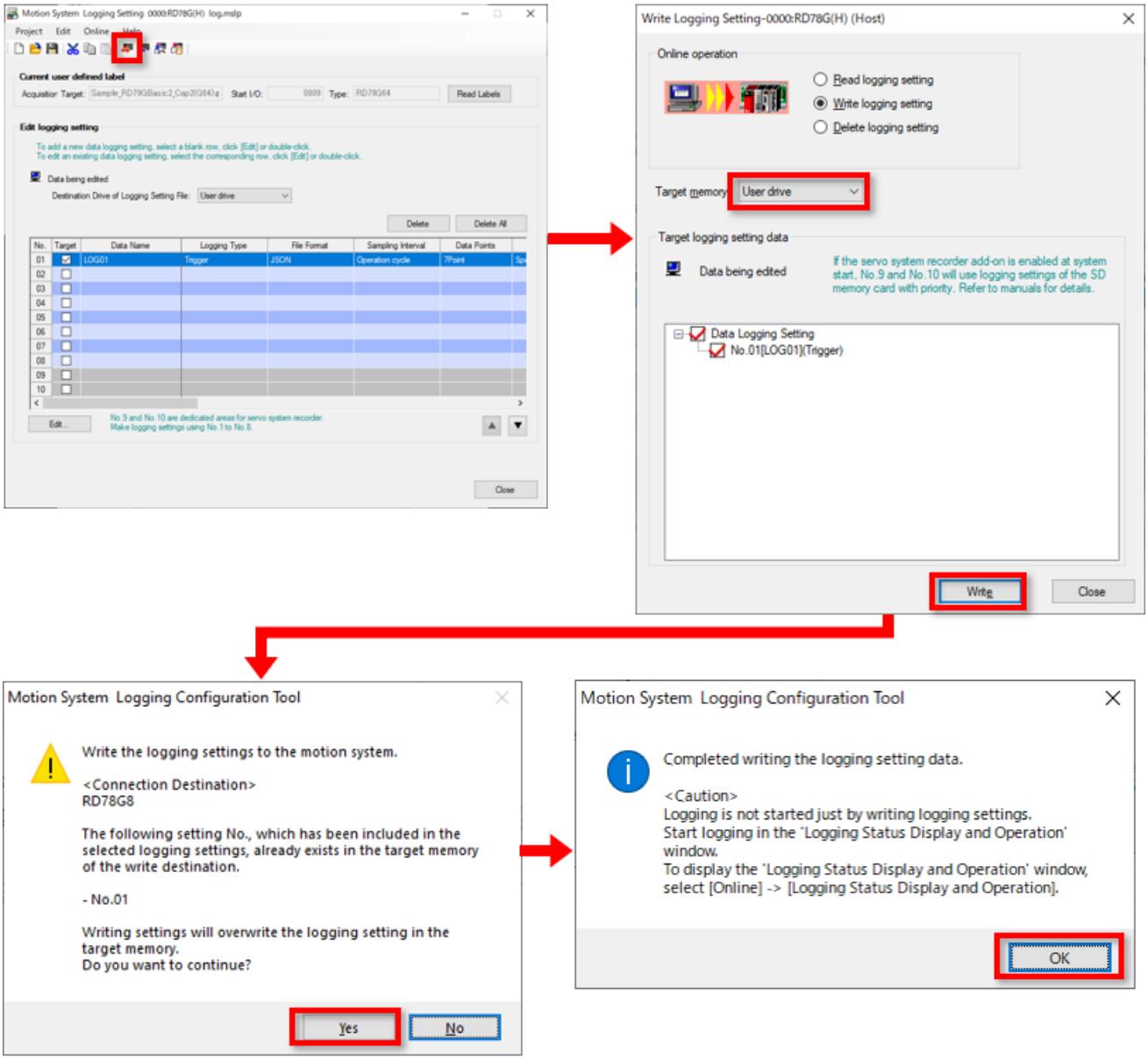


(11) Return to the motion system logging setting tool.
The settings that have been configured can be saved.
Click the save icon and save to the destination of your choice.



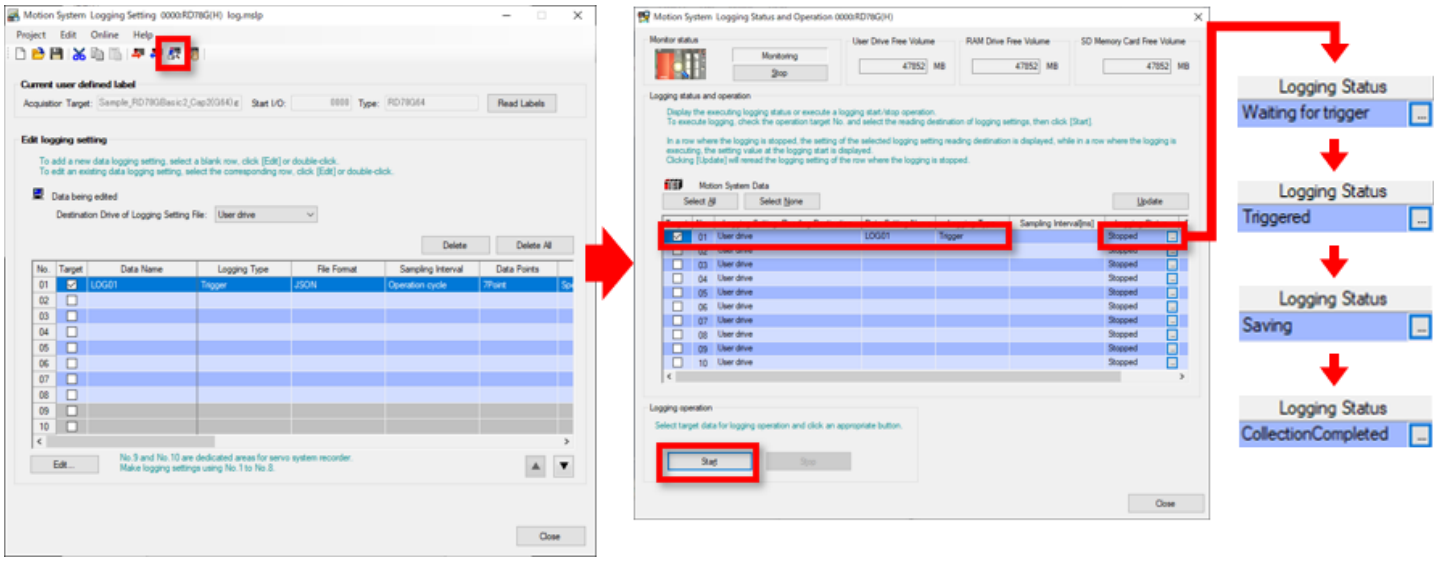
5.3 Writing the Logging Setting

The setting information of the logging is written.
Click the write logging setting icon, select the target memory, and click the [Write] button.
A confirmation window appears.
Click the [Yes] button and continue. When the writing is completed, click the [OK] button and close the screen.



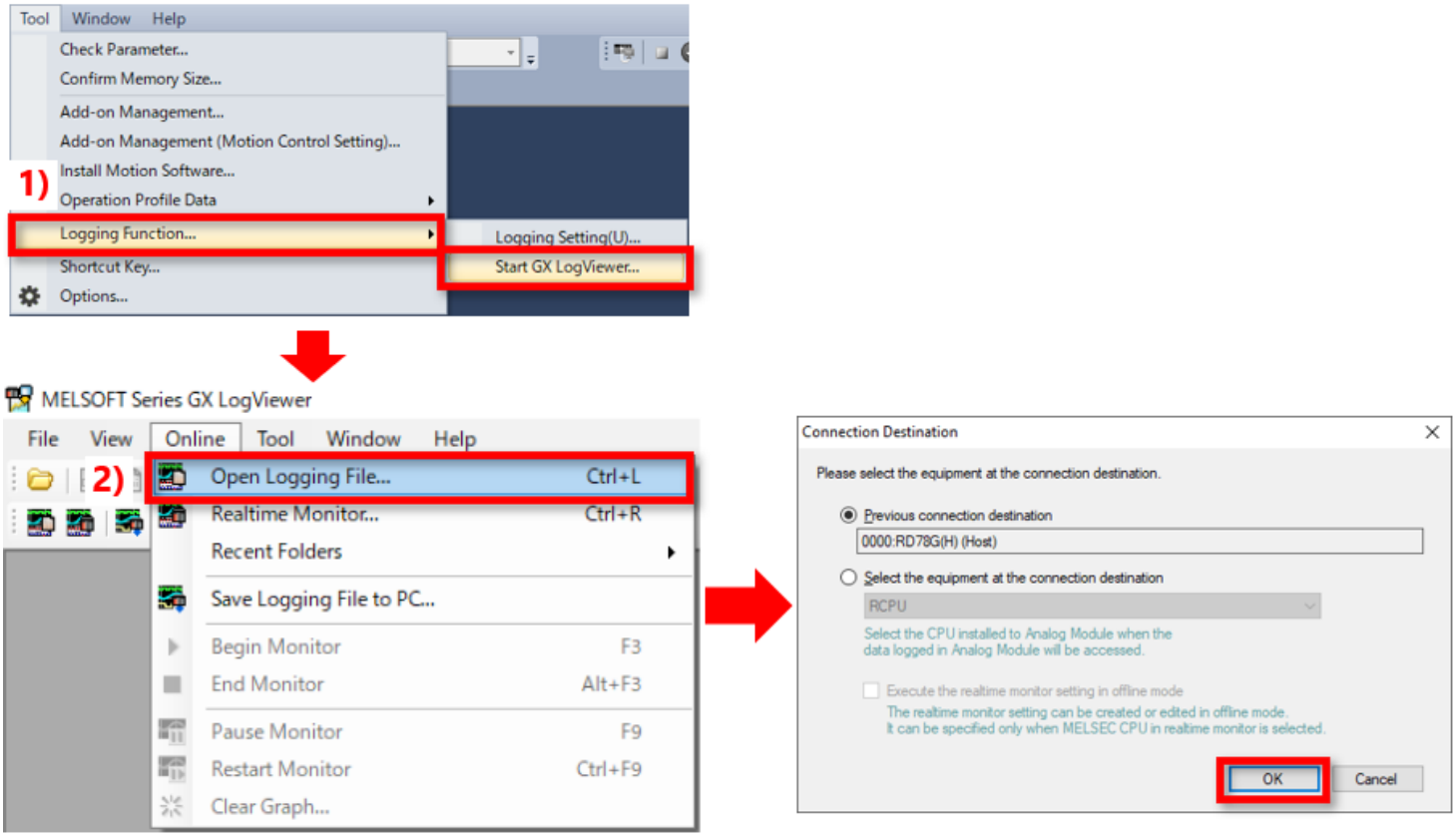
5.4 Starting the Logging

When "Start by User Operation" is set in 5.2 (9), click the [Logging Status and Operation] icon to display the [Logging Status and Operation] screen and to start the logging.
When the logging data setting name to be executed is selected the [Start] button is clicked, the LoggingStatus switches to “Waiting for trigger".
When the program is executed in this state and the trigger condition (when X24 is turned ON in this example of this course) is satisfied, the status switches to “Triggered".
When the logging is completed, the status switches to “CollectionCompleted" from “Saving".



5.5 Reading the Logging Data

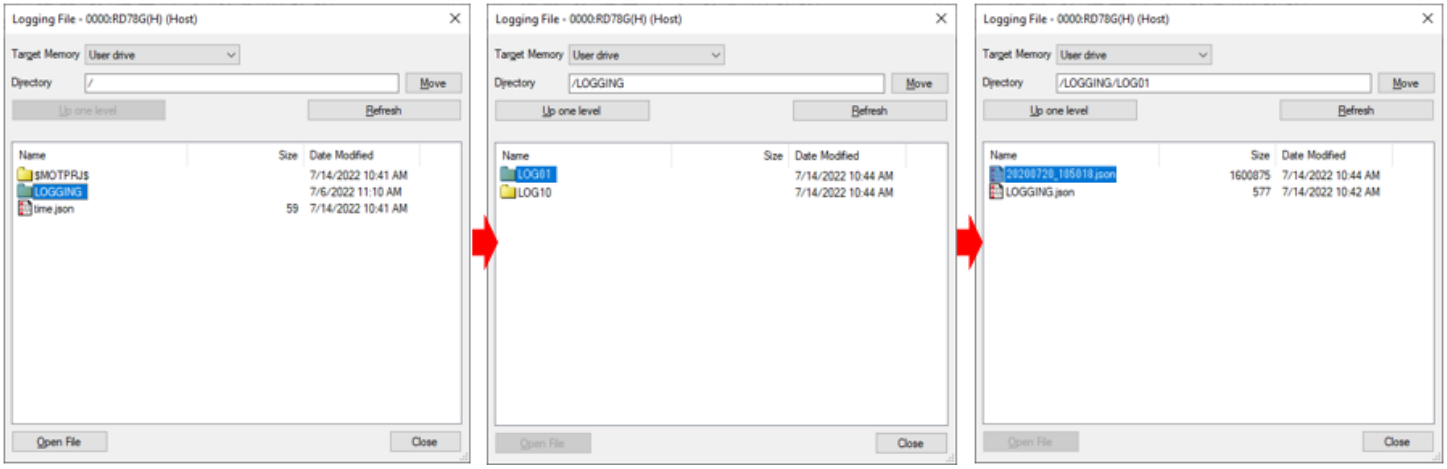
GX LogViewer is used to read the logging data.
Select [Tool] → [Logging Function] → [Start GX LogViewer] from the tool bar of the Motion Control Setting Function screen.
When GX LogViewer is started, select [Online] → [Open Logging File].
Select "0000:RD78G(H) (Host)" in the Connection Destination screen. (Note)



(Note) If GX LogViewer is already started, and the communication with the Motion module is already set, this screen is not displayed.

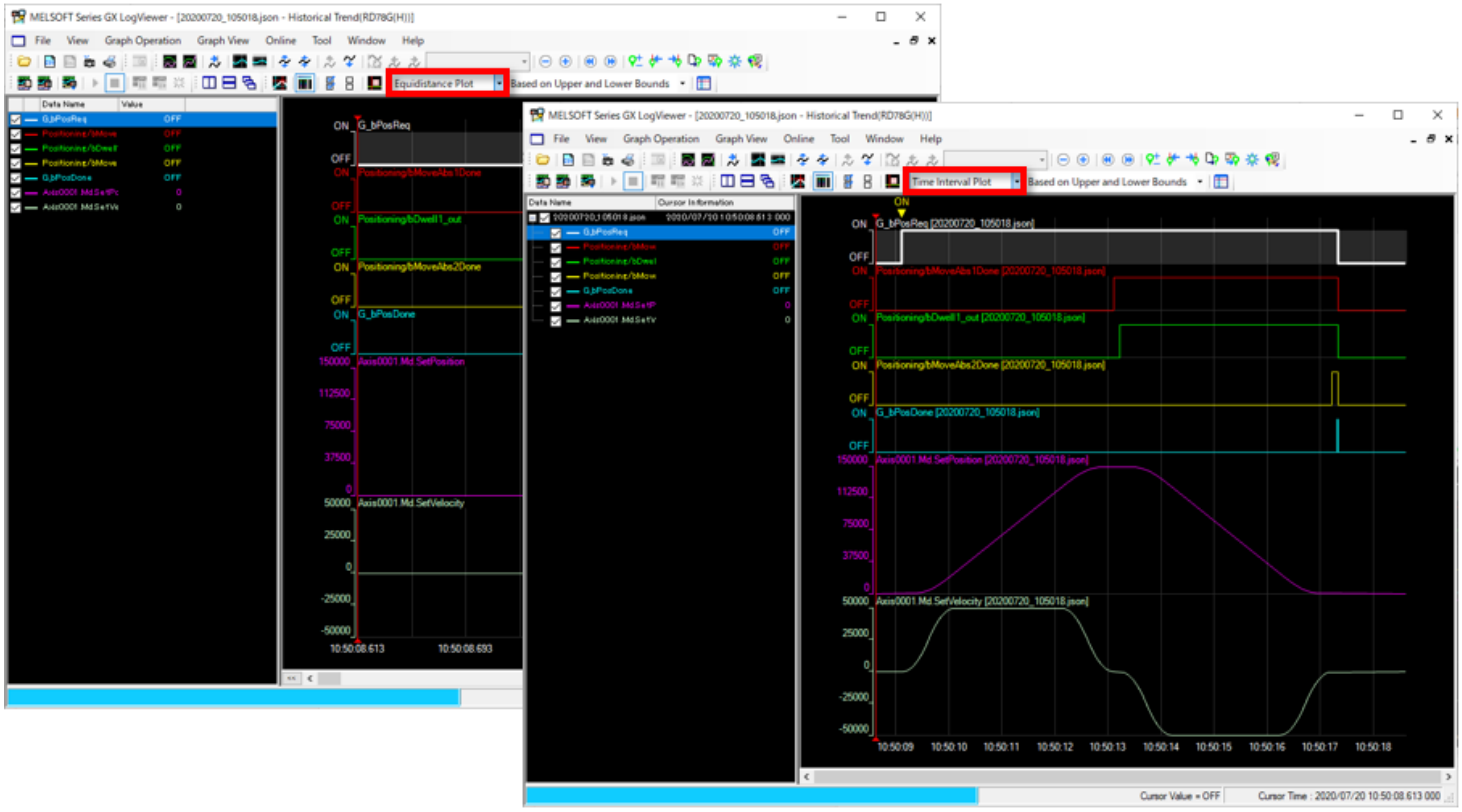
5.5 Reading the Logging Data

Select the logging file to be read.
In the example of this chapter, the user drive in "LOGGING" → "LOG1" → "(Logged date and time).json" is selected.
Select the file name and click the [Open File] button.



5.5 Reading the Logging Data

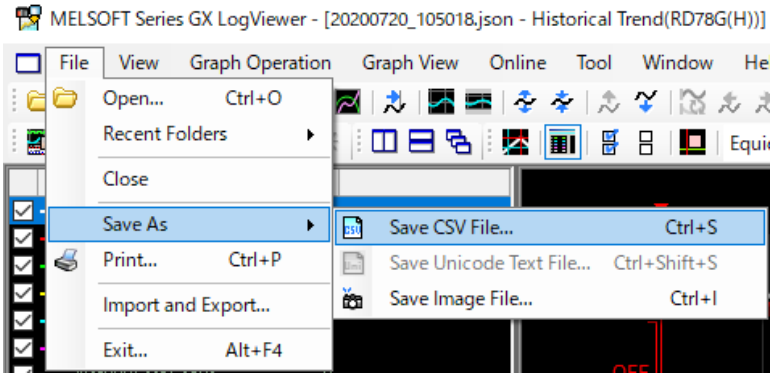
The waveform data logged in GX LogViewer is displayed.
When the plot format is changed from "Equidistance Plot" to "Time Interval Plot", the entire logged waveform can be displayed.



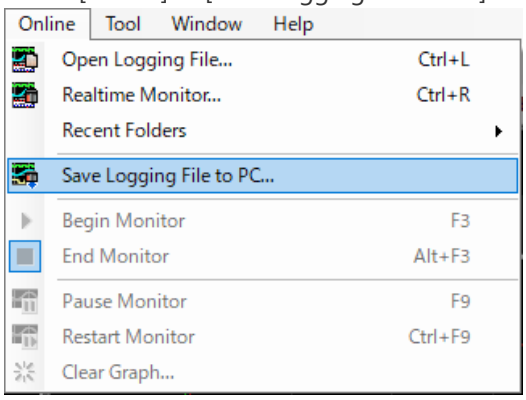
5.6 Saving the Logging Data

The logged waveform data can be saved as a csv file or json file.
(When it is logged in CSV format, it can be saved as CSV file.)

1) When saving as csv file
   Select [File] → [Save As] → [Save CSV File] from the tool bar of GX LogViewer.



2) When saving as json file
   Select [Online] → [Save Logging File to PC] from the tool bar of GX LogViewer.



5.7 Summary of This Chapter

In this chapter, you have learned:

- Starting the Logging Configuration Tool
- Setting the Data to be Logged
- Writing the Logging Setting
- Starting the Logging
- Reading the Logging Data
- Saving the Logging Data

Point

Starting the Logging Configuration Tool	• Start the motion system logging setting tool from the motion control setting function.
Setting the Data to be Logged	• Set the data to be logged, trigger conditions, and others by following the procedure displayed in the motion system logging setting tool.
Writing the Logging Setting	• Write the logging setting data to the Motion module before logging.
Starting the Logging	• When the logging start condition is set to "Start by User Operation", click the start button in the "Logging Status and Operation" screen to start logging.
Reading the Logging Data	• GX LogViewer is used to read the logging data.
Saving the Logging Data	• The logged waveform data can be saved as a csv file or json file.

Test | Final Test

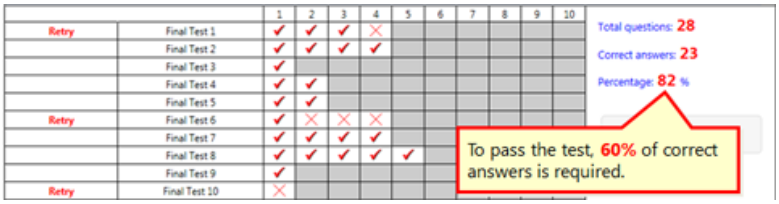
Now that you have completed all of the lessons of the **MELSEC iQ-R Series Motion Module Basics (RD78G(H) Positioning Control)** Course, you are ready to take the final test. If you are unclear on any of the topics covered, please take this opportunity to review those topics.

There are a total of 4 questions (7 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**

Select the correct description(s) of the public label. (You may select multiple answers.)

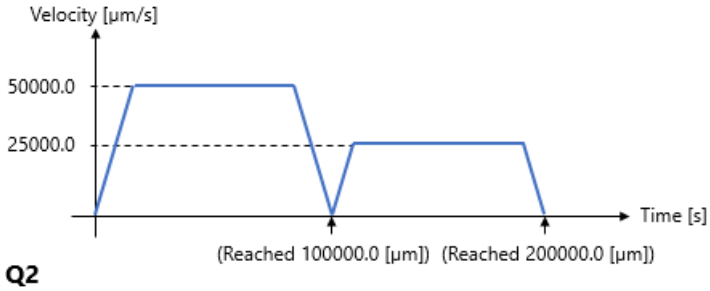
- [ ] **A public label is a shared label that can be used in both the Motion module and PLC CPU.**
- [ ] **Public label is registered from the global label of the PLC CPU.**
- [ ] **When the global label is set to the public label, select whether the label is read or written from/to the PLC CPU.**

Test **Final Test 2**

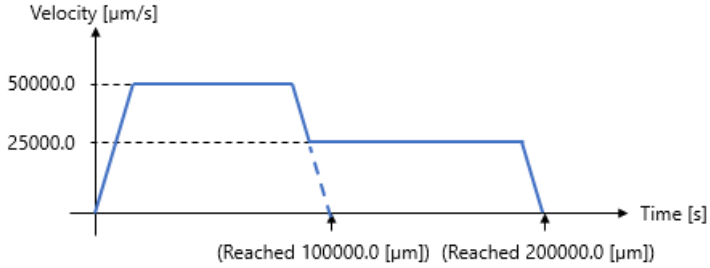
FB1 is executed first, and then FB2 is executed.
When the target position and target velocity of FB1 and FB2 are as shown in the following table, select the buffer mode that is performed next.

	Target position	Target velocity
FB1	100000.0[μm]	50000.0[μm/s]
FB2	200000.0[μm]	25000.0[μm/s]

Q1



Q2



Q1 Select the appropriate answer.

Q2 Select the appropriate answer.

Q1:
- 1：mcAborting
- 2：mcBuffered
- 3：mcBlendingNext
- 4：mcBlendingPrevious

Q2:
- 1：mcBlendingNext and mcBlendingHigh
- 2：mcBlendingPrevious and mcBlendingHigh
- 3：mcBlendingNext and mcBlendingLow
- 4：mcBlendingPrevious and mcBlendingLow

Test

Final Test 3

Select the correct sentence(s) from the following for programming with the PLC CPU. (You may select multiple answers.)

- [ ] **The FB library must be registered to GX Works3 to use the Motion control FB for Motion module in the PLC CPU.**
- [ ] **Place the motion control FB in the program editor from the project tree of GX Works3.**
- [ ] **There are no parameters to be set for the Motion module.**

Test **Final Test 4**

Select the appropriate answers to fill in the blanks.

- Start (Q1) to set the data to be logged.
- Write the logging data to (Q2) to perform logging.
- (Q3) is used to read the logging data and check the waveform.

Q1 Select the appropriate answer.

Q2 Select the appropriate answer.

Q3 Select the appropriate answer.

Q1:
- 1：CPU module logging configuration tool
- 2：Motion system logging setting tool

Q2:
- 1：CPU module
- 2：Motion module
- 3：Servo amplifier

Q3:
- 1：MR Configurator2
- 2：GX LogViewer

Test **Final Test 1**

Select the correct description(s) of the public label. (You may select multiple answers.)

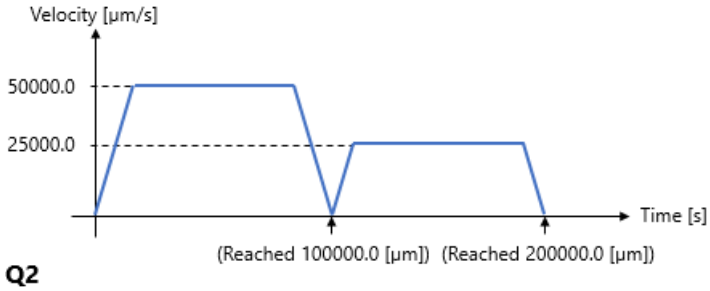
- [x] **A public label is a shared label that can be used in both the Motion module and PLC CPU.**
- [ ] **Public label is registered from the global label of the PLC CPU.**
- [x] **When the global label is set to the public label, select whether the label is read or written from/to the PLC CPU.**

Test **Final Test 2**

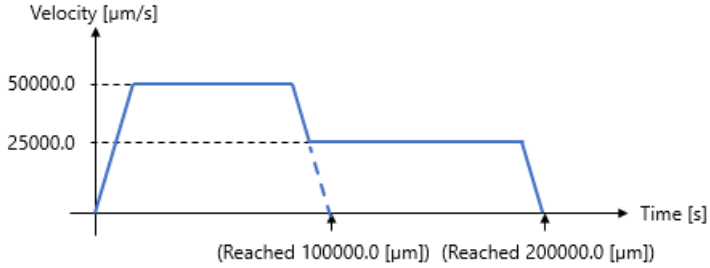
FB1 is executed first, and then FB2 is executed.
When the target position and target velocity of FB1 and FB2 are as shown in the following table, select the buffer mode that is performed next.

	Target position	Target velocity
FB1	100000.0[μm]	50000.0[μm/s]
FB2	200000.0[μm]	25000.0[μm/s]

Q1



Q2



Q1 2：mcBuffered

Q2 3：mcBlendingNext and mcBlendingLow

Q1:
- 1：mcAborting
- 2：mcBuffered
- 3：mcBlendingNext
- 4：mcBlendingPrevious

Q2:
- 1：mcBlendingNext and mcBlendingHigh
- 2：mcBlendingPrevious and mcBlendingHigh
- 3：mcBlendingNext and mcBlendingLow
- 4：mcBlendingPrevious and mcBlendingLow

Test

Final Test 3

Select the correct sentence(s) from the following for programming with the PLC CPU. (You may select multiple answers.)

- [x] **The FB library must be registered to GX Works3 to use the Motion control FB for Motion module in the PLC CPU.**
- [ ] **Place the motion control FB in the program editor from the project tree of GX Works3.**
- [ ] **There are no parameters to be set for the Motion module.**

Test **Final Test 4**

Select the appropriate answers to fill in the blanks.

- Start (Q1) to set the data to be logged.
- Write the logging data to (Q2) to perform logging.
- (Q3) is used to read the logging data and check the waveform.

Q1 2：Motion system logging setting tool

Q2 2：Motion module

Q3 2：GX LogViewer

Q1:
- 1：CPU module logging configuration tool
- 2：Motion system logging setting tool

Q2:
- 1：CPU module
- 2：Motion module
- 3：Servo amplifier

Q3:
- 1：MR Configurator2
- 2：GX LogViewer

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: **7**

Correct answers: **7**

Percentage: **100** %

Clear

You have completed the "MELSEC iQ-R Series Motion Module Application (RD78G(H) Positioning 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