

**OMRON**

# **Sysmac Library**

---


## **User's Manual for Servo Press Library SYSMAC-XR013**

## NOTE

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, mechanical, electronic, photocopying, recording, or otherwise, without the prior written permission of OMRON.

No patent liability is assumed with respect to the use of the information contained herein. Moreover, because OMRON is constantly striving to improve its high-quality products, the information contained in this manual is subject to change without notice. Every precaution has been taken in the preparation of this manual. Nevertheless, OMRON assumes no responsibility for errors or omissions. Neither is any liability assumed for damages resulting from the use of the information contained in this publication.

## Trademarks

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

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

## Copyrights

Microsoft product screen shots reprinted with permission from Microsoft Corporation.

# Introduction

---

Thank you for purchasing an NJ/NX-series CPU Unit or an NY-series Industrial PC.

This manual contains information that is necessary to use the function blocks in the Servo Press Library. ("Function block" is sometimes abbreviated as "FB".) Please read this manual and make sure you understand the functionality and performance of the NJ/NX-series CPU Unit before you attempt to use it in a control system.

This manual provides function block specifications. It does not describe application restrictions or combination restrictions for Controllers, Units, and components.

Refer to the user's manuals for all of the products in the application before you use any of the products.

Keep this manual in a safe place where it will be available for reference during operation.

## Features of the Library

The Servo Press Library is used to generate the operation commands and monitor the operations of actuator for servo presses with NJ/NX-series CPU Unit or NY-series Industrial PC. You can use the Servo Press Library to realize the high-speed and high precision servo press control and reduce programming work.

## Intended Audience

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

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

For programming, this manual is intended for personnel who understand the programming language specifications in international standard IEC 61131-3 or Japanese standard JIS B 3503.

## Applicable Products

For the model numbers and versions of an NJ/NX-series CPU Unit, NY-series Industrial PC, and the Sysmac Studio that this library supports, refer to Sysmac Library Version Information in the *SYS-MAC-XR□□□ Sysmac Library Catalog* (Cat. No. P102). This catalog can be downloaded from the OMRON website (<http://www.ia.omron.com/products/family/3459/download/catalog.html>).

# Manual Structure

---

## Special Information

Special information in this manual is classified as follows:



### **Precautions for Safe Use**

---

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



### **Precautions for Correct Use**

---

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



### **Additional Information**

---

Additional information to read as required.

This information is provided to increase understanding or make operation easier.



### **Version Information**

---

Information on differences in specifications and functionality for CPU Units and Industrial PCs with different unit versions and for different versions of the Sysmac Studio are given.

Note References are provided to more detailed or related information.

# CONTENTS

---

<b>Introduction .....</b>	<b>1</b>
Features of the Library.....	1
Intended Audience.....	1
Applicable Products.....	2
<b>Manual Structure .....</b>	<b>3</b>
Special Information.....	3
<b>CONTENTS.....</b>	<b>4</b>
<b>Terms and Conditions Agreement.....</b>	<b>6</b>
Warranty, Limitations of Liability.....	6
Application Considerations.....	7
Disclaimers.....	7
<b>Safety Precautions .....</b>	<b>8</b>
<b>Precautions for Safe Use.....</b>	<b>11</b>
<b>Precautions for Correct Use.....</b>	<b>12</b>
<b>Related Manuals .....</b>	<b>13</b>
<b>Revision History .....</b>	<b>16</b>
<b>Procedure to Use Sysmac Libraries.....</b>	<b>17</b>
Procedure to Use Sysmac Libraries Installed Using the Installer.....	18
Procedure to Use Sysmac Libraries Uploaded from a CPU Unit or an Industrial PC.....	22
<b>Servo Press Library .....</b>	<b>25</b>
What Is the Servo Press Library?.....	26
Servo Press Library Functionality.....	27
<b>Common Specifications of Function Blocks .....</b>	<b>33</b>
Common Variables.....	34
Precautions.....	40
<b>Individual Specifications of Function Blocks .....</b>	<b>41</b>
SP_SingleAxisPrgOpr.....	42
SP_PrgStatusCtrl.....	98
SP_StepCompleteJudge.....	115
SP_StepLoadAlarm.....	127
SP_PrgLoadAlarm.....	135
SingleAxisCtrl.....	143
TorqueToLoad.....	172
LoadToTorque.....	177
PrgOprRsltRec.....	181
PrgOprRsltCSVWrite.....	191
PrgOprTracePut.....	200
PrgOprTracePut2.....	207
PrgOprTraceCSVWrite.....	215
PrgOprTraceCSVWrite2.....	225
XYDataRec.....	235
XYDataRec2.....	245
XYDataToGraph.....	256
XYDataToGraph2.....	265

**Appendix..... 275**  
    Referring to Library Information.....276  
    Referring to Function Block and Function Source Codes .....279

# Terms and Conditions Agreement

---

## Warranty, Limitations of Liability

### Warranties

---

#### ● Exclusive Warranty

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

#### ● Limitations

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

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

#### ● Buyer Remedy

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

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

### Limitation on Liability; Etc

---

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

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



## Application Considerations

### Suitability of Use

---

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

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

### Programmable Products

---

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

## Disclaimers

### Performance Data

---

Data presented in Omron Company websites, catalogs and other materials is provided as a guide for the user in determining suitability and does not constitute a warranty. It may represent the result of Omron's test conditions, and the user must correlate it to actual application requirements. Actual performance is subject to the Omron's Warranty and Limitations of Liability.

### Change in Specifications

---

Product specifications and accessories may be changed at any time based on improvements and other reasons. It is our practice to change part numbers when published ratings or features are changed, or when significant construction changes are made. However, some specifications of the Product may be changed without any notice. When in doubt, special part numbers may be assigned to fix or establish key specifications for your application. Please consult with your Omron's representative at any time to confirm actual specifications of purchased Product.

### Errors and Omissions

---

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



# Safety Precautions

## Definition of Precautionary Information





The following notation is used in this user’s manual to provide precautions required to ensure safe usage of an NJ/NX-series Controller and an NY-series Industrial PC.

The safety precautions that are provided are extremely important to safety. Always read and heed the information provided in all safety precautions.

The following notation is used.

 <b>WARNING</b>	Indicates a potentially hazardous situation which, if not avoided, could result in death or serious injury. Additionally, there may be severe property damage.
 <b>Caution</b>	Indicates a potentially hazardous situation which, if not avoided, may result in minor or moderate injury, or property damage.

## Symbols

	The circle and slash symbol indicates operations that you must not do. The specific operation is shown in the circle and explained in text. This example indicates prohibiting disassembly.
	The triangle symbol indicates precautions (including warnings). The specific operation is shown in the triangle and explained in text. This example indicates a precaution for electric shock.
	The triangle symbol indicates precautions (including warnings). The specific operation is shown in the triangle and explained in text. This example indicates a general precaution.
	The filled circle symbol indicates operations that you must do. The specific operation is shown in the circle and explained in text. This example shows a general precaution for something that you must do.

## Warnings

### **WARNING**

---

Emergency stop circuits, interlock circuits, limit circuits, and similar safety measures must be provided in external control circuits. Not doing so may result in serious accidents due to incorrect operation.



---

Using this function block (FB) in a device, confirm that the program and FB operates properly. Design a program so that safety measures such as fail-safe circuits are implemented outside of the FB.

---



## Cautions

### **Caution**

---

Read all related manuals carefully before you use this library.



---

Check the user program, data, and parameter settings for proper execution before you use them for actual operation.



---

Keep the emergency stop switch in hand to prevent a sudden operation of the motor when you perform testing operation.



---

The sample programming shows only the portion of a program that uses the function or function block from the library.



---

When using actual devices, also program safety circuits, device interlocks, I/O with other devices, and other control procedures.



---

Understand the contents of sample programming before you use the sample programming and create the user program.



---

Create a user program so that the actual device operates as intended.



---

Check the user program, data, and parameter settings for proper execution so that the actual device operates as intended before you use them for actual operation.



# Precautions for Safe Use

## Operation

- The Sysmac Library and manuals are assumed to be used by personnel that is given in Intended Audience in this manual. Otherwise, do not use them.
- Execute the following four function blocks which consist of single-axis program operation in the same task. If these function blocks are executed in the different task, the completion for each step or the completion for the program operation may not be correctly determined.
  - SP\_PrgStatusCtrl (Program Status Control)
  - SP\_StepCompleteJudge (Step Completion Determination)
  - SP\_StepLoadAlarm (Step Load Alarm Determination)
  - SP\_PrgLoadAlarm (Program Load Alarm Determination)
- When you use the LoadToTorque (Load-to-Torque Conversion) function to calculate the *MeasuringTorque* (Measured Torque) input variable in the SingleAxisCtrl (Single-axis Control) function block, execute the LoadToTorque function and the SingleAxisCtrl function block in the same task.
- If you use *TorqueLimitPositiveVal* (Positive Torque Limit) or *TorqueLimitNegativeVal* (Negative Torque Limit) in the SP\_SingleAxisPrgOpr (Single-axis Program Operation) function block, make sure to connect the *TorqueLimitParam* (Torque Limit Settings) output variable in the SingleAxisCtrl (Single-axis Control) function block that is combined with the SP\_SingleAxisPrgOpr (Single-axis Program Operation) function block to the MC\_SetTorqueLimit (Set Torque Limit) instruction.
- If execution of the SP\_SingleAxisPrgOpr (Single-axis Program Operation) function block is aborted, function blocks used in combination with it, such as the SingleAxisCtrl (Single-axis Control) function block, will not stop. Create a user program that stops axis processing when the SP\_SingleAxisPrgOpr (Single-axis Program Operation) function block is aborted.
- If the SP\_SingleAxisPrgOpr (Single-axis Program Operation) function block ends in error, function blocks used in combination with it, such as a SingleAxisCtrl (Single-axis Control) function block, will not stop. Create a user program that stops axis processing when the SP\_SingleAxisPrgOpr (Single-axis Program Operation) function block ends in an error.
- When you change the velocity override during execution of SingleAxisCtrl (Single-axis Control) function block, use *VelOverrideEnable* (Velocity Override Enable) and *InputVelFactor* (Velocity Override Input Value) in the SingleAxisCtrl (Single-axis Control) function block. If you use the MC\_SetOverride (Set Override Factors) instruction to change velocity override, the last set value that is specified is valid.
- If another motion control instruction is executed for the axis for which the SingleAxisCtrl (Single-axis Control) function block is being executed, write the user program so that multi-execution of the other instruction is not started until after execution of the SingleAxisCtrl (Single-axis Control) function block is started.
- If *UnachievedNextStepNo* is set to 0 and the step completion conditions are not met, *Done* changes to TRUE and the SP\_SingleAxisPrgOpr (Single-axis Program Operation) function block is completed. At this time, the MC\_Stop (Stop) instruction will not be executed for the next SingleAxisCtrl (Single-axis Control) function block. Set *UnachievedNextStepNo* to a value other than 0 and specify the operation if the step completion conditions are not met.
- If *TimeOutNextStepNo* is set to 0 and the step timeout occurs, *Done* changes to TRUE and the SP\_SingleAxisPrgOpr (Single-axis Program Operation) function is completed. At this time, the MC\_Stop (Stop) instruction will not be executed for the next SingleAxisCtrl (Single-axis Control) function block. Set *TimeOutNextStepNo* to a value other than 0 and specify the operation if the step timeout occurs.

## Changing the Control Mode

- When the Control Mode is changed, the current position may change suddenly.

# Precautions for Correct Use

---

## Using the Library

---

- You cannot change the source code of the functions or function blocks that are provided in the Sysmac Library.

## Operation

---

- Specify the input parameter values within the valid range.
- In the function or function block with an Enabled output variable, if the value of Enabled is FALSE, do not use the processing result of the function or function block as a command value to the control target.
- The axis velocity is higher for torque control. Make sure that you set *Velocity* (Velocity Set Value) for safety.
- When you use an OMRON G5-series Servo Drive, set the Velocity Limit Selection (3317 hex) of the Servo Drive to 1 (speed limit value via EtherCAT communications). Otherwise, the velocity limit is not affected. Also, the axis does not stop even if the limit input signal turns ON.
- Process data 607F hex is used for the velocity limit. When you use an OMRON G5-series Servo Drive, set the advanced settings in the Axis Parameter Settings of the Sysmac Studio to use the Velocity Limit Value (607F hex). To use a velocity limit with a Servo Drive from another manufacturer, refer to the manual for the Servo Drive.
- If *MeasuringTorque* (Measured Torque) is not connected for torque feedback control, the axis torque command value is greater and the axis velocity is higher. Make sure that you set *TorqueLowLmt* (Torque Lower Limit), *TorqueUpLmt* (Torque Upper Limit), and *Velocity* (Velocity Set Value) for safety.
- If you change the Control Mode of the Servo Drive during axis operation, an error may occur depends on the Servo Drive.
- Execute the *SingleAxisCtrl* (Single-axis Control) function block and the *SP\_SingleAxisPrgOpr* (Single-axis Program Operation) function block in the same task when you connect these two function blocks. If these two function blocks are executed in the different task, the motion control may not be correctly executed.
- *MeasuringTorque* (Measured Torque):

This variable gives the actual axis torque that is measured. It is used as a feedback input of the actual torque monitor value in the torque feedback control. Convert the rated torque for the Servomotor to the percentage that is assumed as 100% and input the percentage in increments of %.

Use the *LoadToTorque* (Torque-to-Load Conversion) function when you convert the force that is measured by an externally-mounted load cell in newtons into a percentage in increments of % to the rated torque. Refer to *LoadToTorque* on page 177 for the details.

# Related Manuals

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

Manual name	Cat. No.	Model numbers	Application	Description
NX-series CPU Unit Hardware User's Manual	W535	NX701-□□□□	Learning the basic specifications of the NX-series NX701 CPU Units, including introductory information, designing, installation, and maintenance. Mainly hardware information is provided	An introduction to the entire NX701 CPU Unit system is provided along with the following information on the CPU Unit. Features and system configuration Overview Part names and functions General specifications Installation and wiring Maintenance and inspection
NX-series NX102 CPU Unit Hardware User's Manual	W593	NX102-□□□□	Learning the basic specifications of the NX102 CPU Units, including introductory information, designing, installation, and maintenance. Mainly hardware information is provided.	An introduction to the entire NX102 system is provided along with the following information on the CPU Unit. Features and system configuration Overview Part names and functions General specifications Installation and wiring Maintenance and Inspection
NX-series NX1P2 CPU Unit Hardware User's Manual	W578	NX1P2-□□□□	Learning the basic specifications of the NX-series NX1P2 CPU Units, including introductory information, designing, installation, and maintenance. Mainly hardware information is provided	An introduction to the entire NX1P2 CPU Unit system is provided along with the following information on the CPU Unit. Features and system configuration Overview Part names and functions General specifications Installation and wiring Maintenance and Inspection
NJ-series CPU Unit Hardware User's Manual	W500	NJ501-□□□□ NJ301-□□□□ NJ101-□□□□	Learning the basic specifications of the NJ-series CPU Units, including introductory information, designing, installation, and maintenance. Mainly hardware information is provided	An introduction to the entire NJ-series system is provided along with the following information on the CPU Unit. Features and system configuration Overview Part names and functions General specifications Installation and wiring Maintenance and inspection
NY-series IPC Machine Controller Industrial Panel PC Hardware User's Manual	W557	NY532-□□□□	Learning the basic specifications of the NY-series Industrial Panel PCs, including introductory information, designing, installation, and maintenance. Mainly hardware information is provided	An introduction to the entire NY-series system is provided along with the following information on the Industrial Panel PC. Features and system configuration Introduction Part names and functions General specifications Installation and wiring Maintenance and inspection

Manual name	Cat. No.	Model numbers	Application	Description
NY-series IPC Machine Controller Industrial Box PC Hardware User's Manual	W556	NY512-□□□□	Learning the basic specifications of the NY-series Industrial Box PCs, including introductory information, designing, installation, and maintenance. Mainly hardware information is provided	An introduction to the entire NY-series system is provided along with the following information on the Industrial Box PC. Features and system configuration Introduction Part names and functions General specifications Installation and wiring Maintenance and inspection
NJ/NX-series CPU Unit Software User's Manual	W501	NX701-□□□□ NX102-□□□□ NX1P2-□□□□ NJ501-□□□□ NJ301-□□□□ NJ101-□□□□	Learning how to program and set up an NJ/NX-series CPU Unit. Mainly software information is provided	The following information is provided on a Controller built with an NJ/NX-series CPU Unit. CPU Unit operation CPU Unit features Initial settings Programming based on IEC 61131-3 language specifications
NY-series IPC Machine Controller Industrial Panel PC / Industrial Box PC Software User's Manual	W558	NY532-□□□□ NY512-□□□□	Learning how to program and set up the Controller functions of an NY-series Industrial PC	The following information is provided on NY-series Machine Automation Control Software. Controller operation Controller features Controller settings Programming based on IEC 61131-3 language specifications
NJ/NX-series Instructions Reference Manual	W502	NX701-□□□□ NX102-□□□□ NX1P2-□□□□ NJ501-□□□□ NJ301-□□□□ NJ101-□□□□	Learning detailed specifications on the basic instructions of an NJ/NX-series CPU Unit	The instructions in the instruction set (IEC 61131-3 specifications) are described.
NY-series Instructions Reference Manual	W560	NY532-□□□□ NY512-□□□□	Learning detailed specifications on the basic instructions of an NY-series Industrial PC	The instructions in the instruction set (IEC 61131-3 specifications) are described.
NJ/NX-series CPU Unit Motion Control User's Manual	W507	NX701-□□□□ NX102-□□□□ NX1P2-□□□□ NJ501-□□□□ NJ301-□□□□ NJ101-□□□□	Learning about motion control settings and programming concepts of an NJ/NX-series CPU Unit.	The settings and operation of the CPU Unit and programming concepts for motion control are described.
NY-series IPC Machine Controller Industrial Panel PC / Industrial Box PC Motion Control User's Manual	W559	NY532-□□□□ NY512-□□□□	Learning about motion control settings and programming concepts of an NY-series Industrial PC.	The settings and operation of the Controller and programming concepts for motion control are described.
NJ/NX-series Motion Control Instructions Reference Manual	W508	NX701-□□□□ NX102-□□□□ NX1P2-□□□□ NJ501-□□□□ NJ301-□□□□ NJ101-□□□□	Learning about the specifications of the motion control instructions of an NJ/NX-series CPU Unit.	The motion control instructions are described.
NY-series Motion Control Instructions Reference Manual	W561	NY532-□□□□ NY512-□□□□	Learning about the specifications of the motion control instructions of an NY-series Industrial PC.	The motion control instructions are described.
NJ/NY-series NC Integrated Controller User's Manual	O030	NJ501-5300 NY532-5400	Performing numerical control with NJ/NY-series Controllers.	Describes the functionality to perform the numerical control. Use this manual together with the <i>NJ/NY-series G code Instructions Reference Manual</i> (Cat. No. O031) when programming.



Manual name	Cat. No.	Model numbers	Application	Description
NJ/NY-series G code Instructions Reference Manual	O031	NJ501-5300 NY532-5400	Learning about the specifications of the G code/M code instructions.	The G code/M code instructions are described. Use this manual together with the <i>NJ/NY-series NC Integrated Controller User's Manual</i> (Cat. No. O030) when programming.
Sysmac Studio Version 1 Operation Manual	W504	SYSMAC -SE2□□□	Learning about the operating procedures and functions of the Sysmac Studio.	Describes the operating procedures of the Sysmac Studio.
CNC Operator Operation Manual	O032	SYSMAC -RTNC0□□□D	Learning an introduction of the CNC Operator and how to use it.	An introduction of the CNC Operator, installation procedures, basic operations, connection operations, and operating procedures for main functions are described.
ZW-7000 series Confocal Fiber Type Displacement Sensor User's Manual'	Z362	ZW-7000□	Learning how to use the ZW-7000 series Confocal Fiber Type Displacement Sensors.	Describes the hardware, setup methods and functions of the ZW-7000 series Confocal Fiber Type Displacement Sensors.

# Revision History

---

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

**Cat. No. W573-E1-03**

↑  
Revision code

Revision code	Date	Revised content
01	July 2016	Original production
02	November 2016	Changed the manual name.
03	January 2019	<ul style="list-style-type: none"><li>• Added compatible models.</li><li>• Added FBs/FUNs that support variable-length arrays.</li></ul>

# Procedure to Use Sysmac Libraries

# Procedure to Use Sysmac Libraries Installed Using the Installer

This section describes the procedure to use Sysmac Libraries that you installed using the installer.

There are two ways to use libraries.

- Using newly installed Sysmac Libraries
- Using upgraded Sysmac Libraries

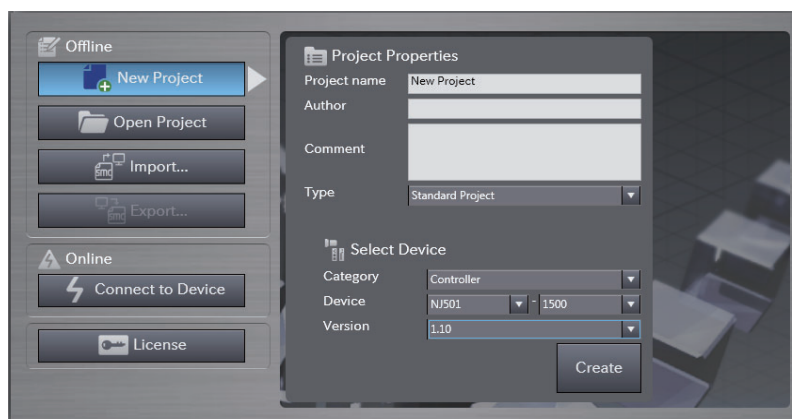


## Version Information

To use Sysmac Libraries, you need the Sysmac Studio version 1.14 or higher.

## Using Newly Installed Libraries

- 1 Start the Sysmac Studio and open or create a new project in which you want to use Sysmac Libraries.

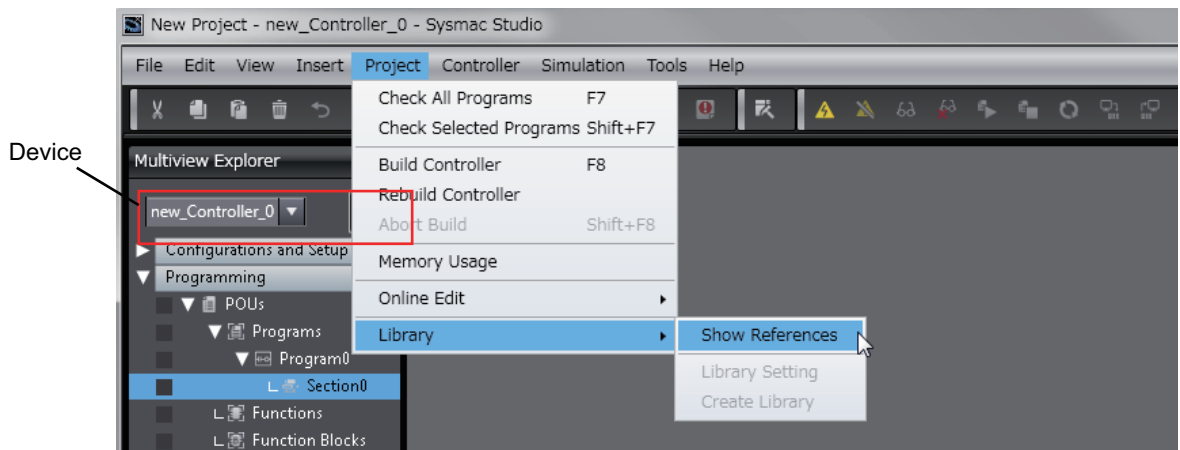


## Precautions for Correct Use


If you create a new project, be sure to configure the settings as follows to enable the use of Sysmac Libraries. If you do not configure the following settings, you cannot proceed to the step 2 and later steps.

- Set the project type to Standard Project or Library Project.
- Set the device category to Controller.
- Set the device version to 1.01 or later.

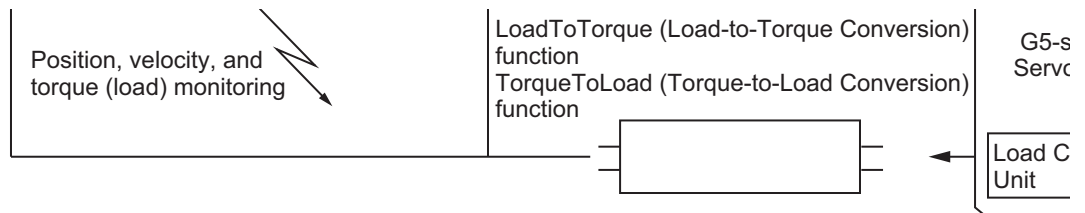
**2 Select Project – Library – Show References.**



**Precautions for Correct Use**

If you have more than one registered device in the project, make sure that the device selected currently is an NJ/NX-series CPU Unit or an NY-series Industrial PC. If you do not select an NJ/NX-series CPU Unit or an NY-series Industrial PC as the device, Library References does not appear in the above menu. When the device selected currently is an NJ/NX-series CPU Unit or an NY-series Industrial PC, the device icon  is displayed in the Multiview Explorer.

**3 Add the desired Sysmac Library to the list and click the OK Button.**



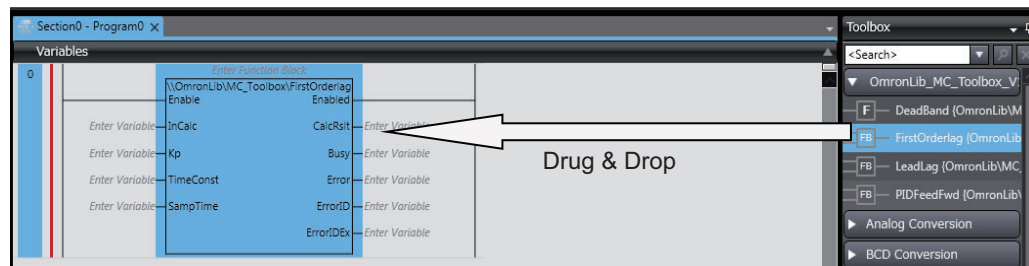
The Sysmac Library file is read into the project.

Now, when you select the Ladder Editor or ST Editor, the function blocks and functions included in a Sysmac Library appear in the Toolbox.

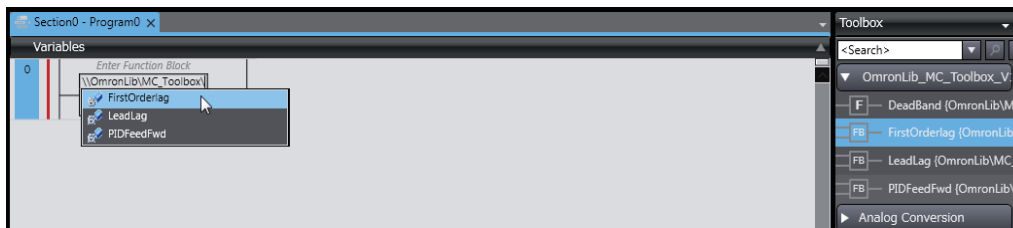
For the procedure for adding and setting libraries in the above screen, refer to the *Sysmac Studio Version 1 Operation Manual (Cat. No. W504)*.

**4 Insert the Sysmac Library’s function blocks and functions into the circuit using one of the following two methods.**

- Select the desired function block or function in the Toolbox and drag and drop it onto the programming editor.



- Right-click the programming editor, select **Insert Function Block** in the menu, and enter the fully qualified name (\\name of namespace\\name of function block).



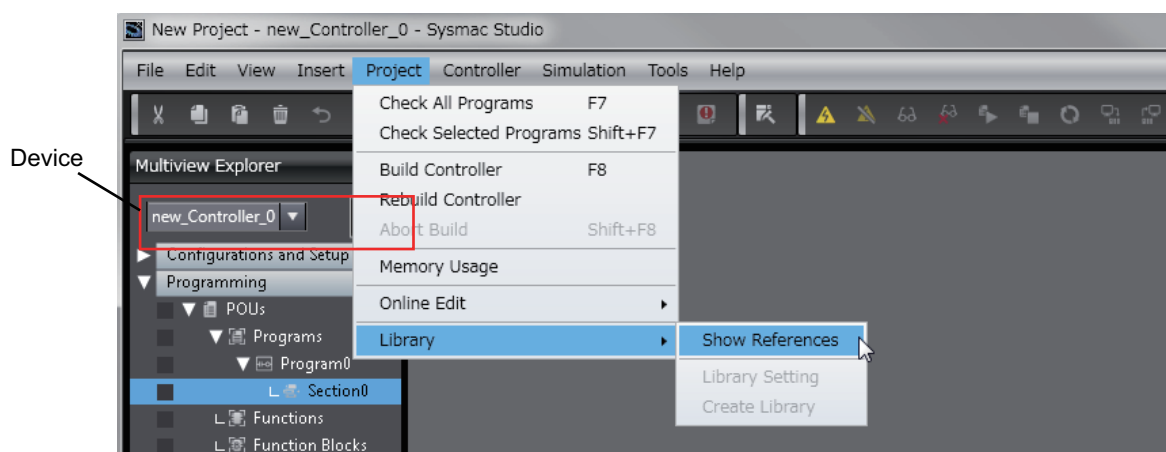
## Precautions for Correct Use

After you upgrade the Sysmac Studio, check all programs and make sure that there is no error of the program check results on the Build Tab Page.


Select **Project – Check All Programs** from the Main Menu.

## Using Upgraded Libraries

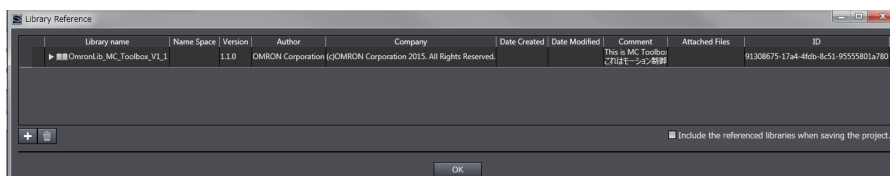
- 1 Start the Sysmac Studio and open a project in which any old-version Sysmac Library is included.
- 2 Select **Project – Library – Show References**.



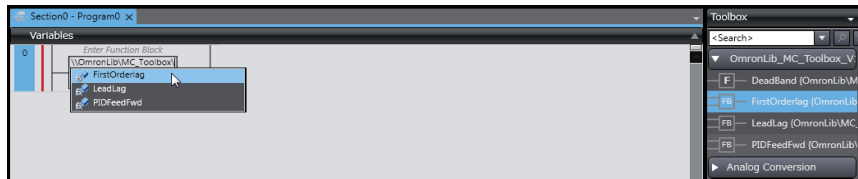
## Precautions for Correct Use

If you have more than one registered device in the project, make sure that the device selected currently is an NJ/NX-series CPU Unit or an NY-series Industrial PC. Otherwise, Library References does not appear in the above menu. When the device selected currently is an NJ/NX-series CPU Unit or an NY-series Industrial PC, the device icon  is displayed in the Multiview Explorer.

- 3 Select an old-version Sysmac Library and click the **Delete Reference Button**.



**4** Add the desired Sysmac Library to the list and click the **OK** Button.



# Procedure to Use Sysmac Libraries Uploaded from a CPU Unit or an Industrial PC

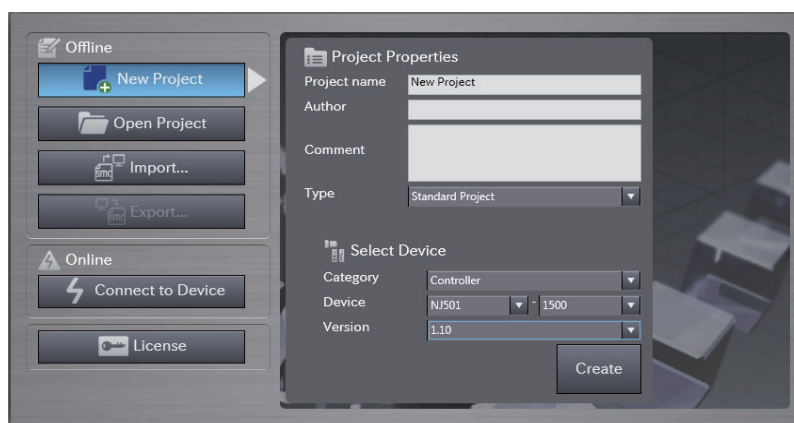
You can use Sysmac Libraries uploaded from a CPU Unit or an Industrial PC to your computer if they are not installed.

The procedure to use uploaded Sysmac Libraries from a CPU Unit or an Industrial PC is as follows.

## ✓ Version Information

To use Sysmac Libraries, you need the Sysmac Studio version 1.14 or higher.

- 1 Start the Sysmac Studio and create a new project in which you want to use Sysmac Libraries.



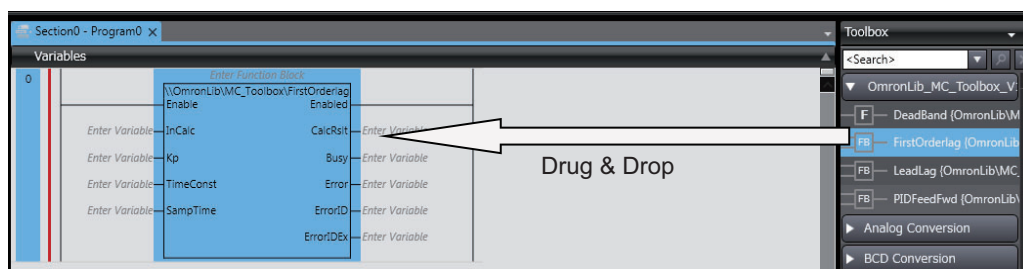
- 2 Connect the computer to the CPU Unit or the Industrial PC and place it online.

- 3 Upload POUs in which any Sysmac Library is used to the computer.

Now, when you select the Ladder Editor or ST Editor, the function blocks and functions included in the Sysmac Library used in the uploaded POUs appear in the Toolbox.

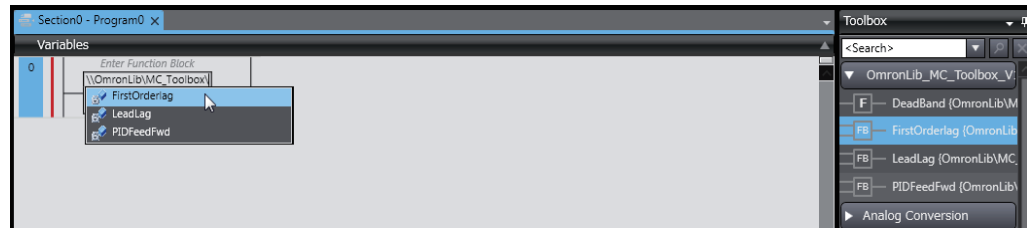
- 4 Insert the Sysmac Library's function blocks and functions into the circuit using one of the following two methods.

- Select the desired function block or function in the Toolbox and drag and drop it onto the Ladder Editor.





- Right-click the programming editor, select **Insert Function Block** in the menu, and enter the fully qualified name (\\name of namespace\name of function block).



### Precautions for Correct Use

- The Sysmac Studio installs library files of the uploaded Sysmac Studio to the specified folder on the computer if they are not present. However, the Sysmac Studio does not install library files to the specified folder on the computer if they are present.  
The specified folder here means the folder in which library files are installed by the installer.
- Note that uploading Sysmac Libraries from a CPU Unit or an Industrial PC does not install the manual and help files for the Sysmac Libraries, unlike the case where you install them using the installer. Please install the manual and help files using the installer if you need them.



# Servo Press Library

# What Is the Servo Press Library?

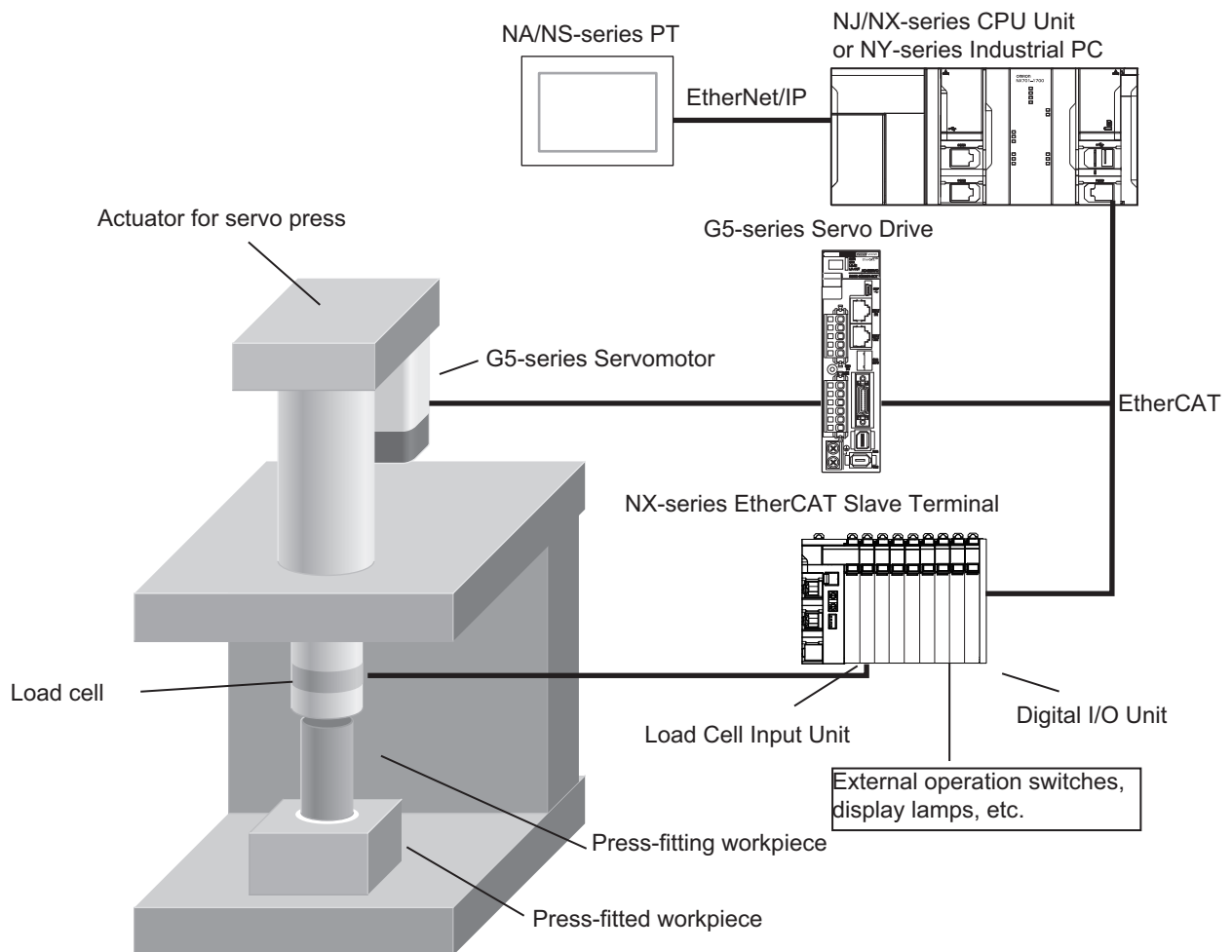
The Servo Press Library is a collection of software functional objects that control actuators for the servo presses that are used in assembling and processing machines. The library is provided as function blocks that are used to build control programming in an NJ/NX-series CPU Unit or an NY-series Industrial PC to generate the Servomotor operation commands that drive the actuator in a servo press, to monitor operation, and to perform other operations.

## System Configuration

The Servo Press Library assumes the following type of system configuration built around an NJ/NX-series CPU Unit or an NY-series Industrial PC.

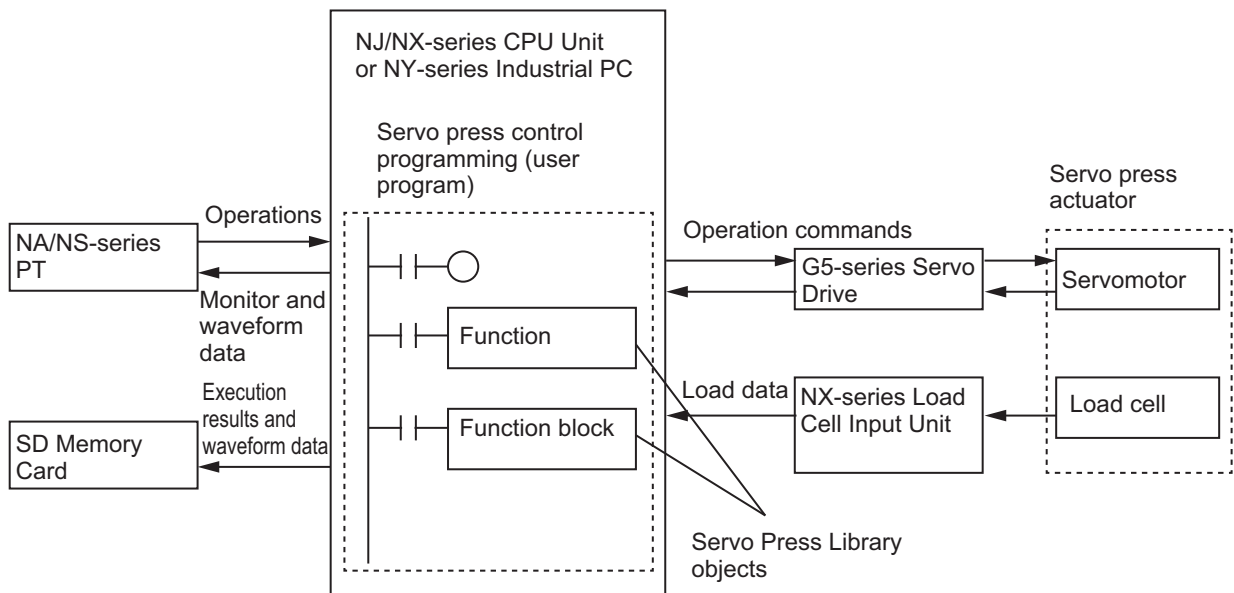
Operation commands are generated for a Servomotor that runs an actuator of a servo press according to an operation pattern prepared in advance. Operation is controlled from the NA/NS-series PT, external operation switches, and machine control programming.

The position and load/torque of the actuator used in the servo press are monitored and displayed as waveforms on an HMI. The operation results and waveform data are saved in memory built into the CPU Unit or in an SD Memory Card.



# Servo Press Library Functionality

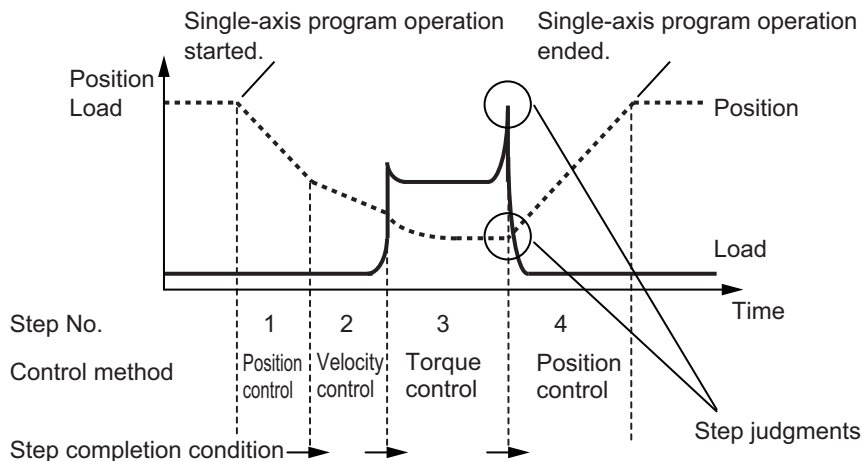
The Servo Press Library contains software objects that are used to build user programs to control actuators used in servo presses. The Servo Press Library contains the following functions and function blocks.

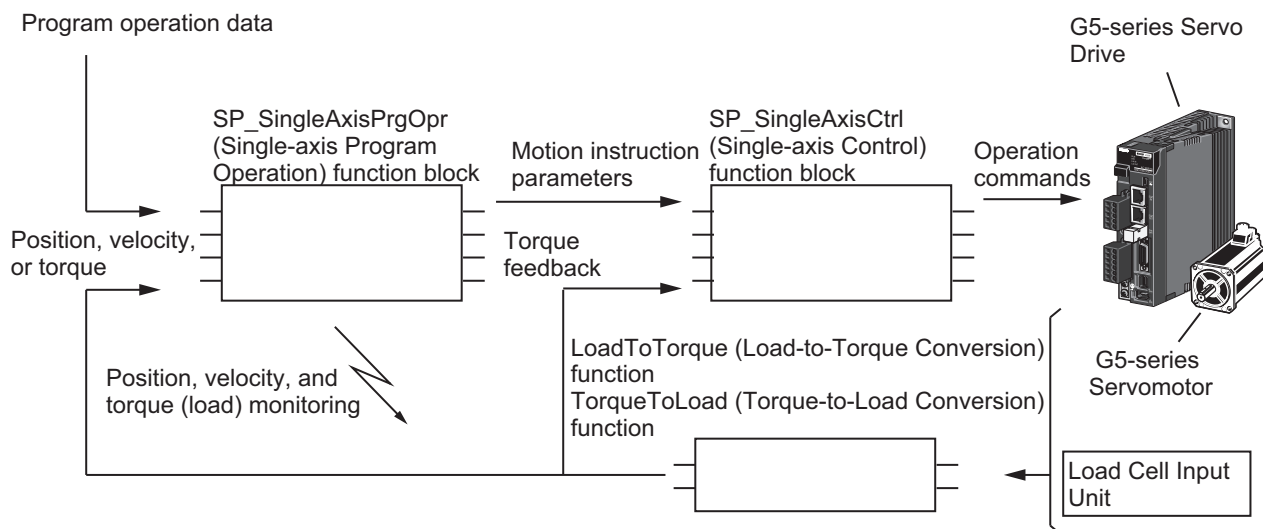


## Single-axis Program Operation for Servo Press Control

The Servomotor is controlled through a series of operations according to an operation pattern that is prepared in advance.

The operation pattern can combine different control methods (selectable from position control, velocity control, torque control, and feedback control), step completion conditions (to set the condition to change from one operation step to the next step, such as reaching a target position or load), and step judgment conditions (e.g., monitoring the position or load while an operation step is in progress or after it ends).





The related functions and function blocks are as follows:

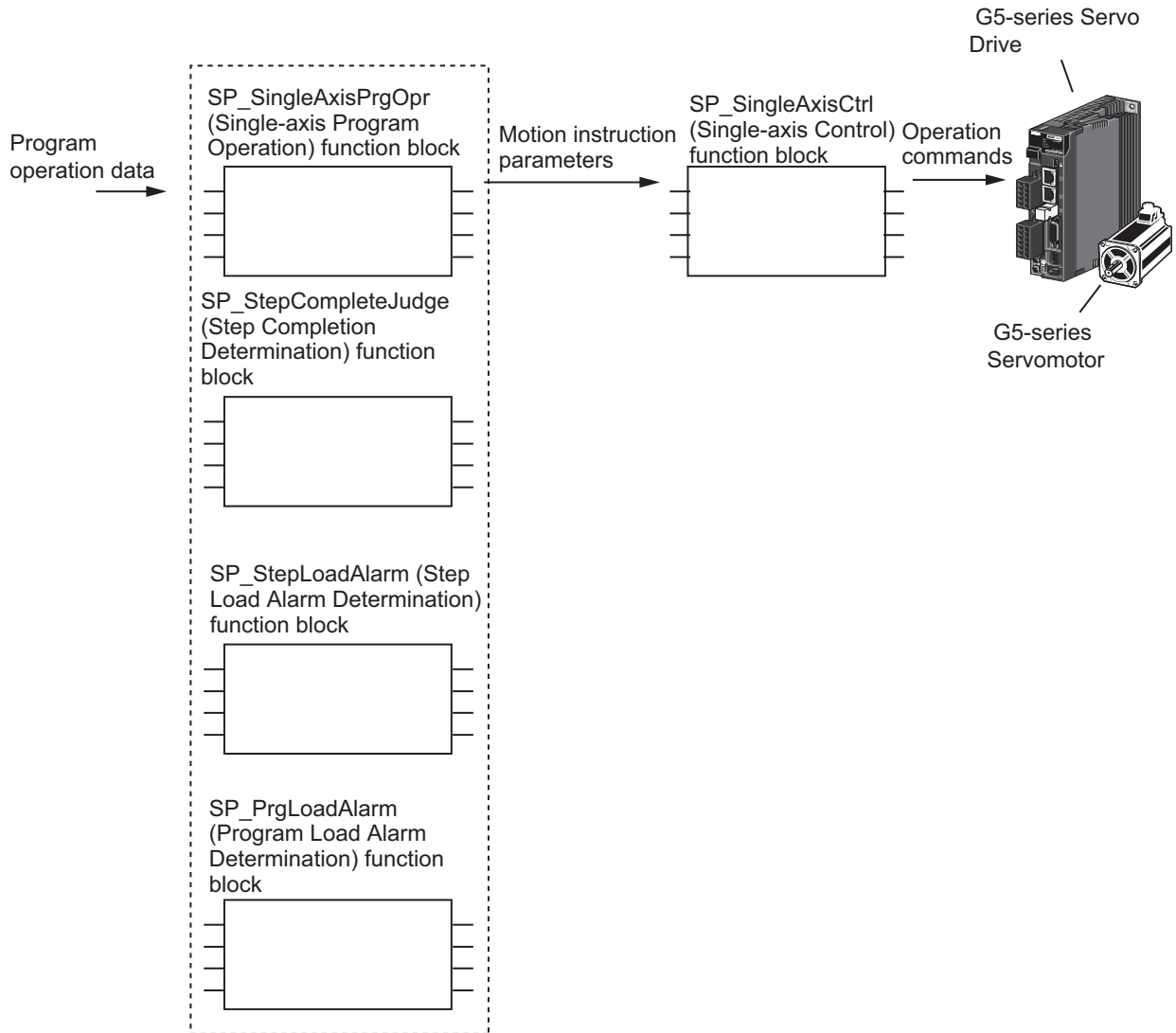
FB/FUN name	Description	Reference
SP_SingleAxisPrgOpr (Single-axis Program Operation) function block	Analyzes single-axis program operation data that combines multiple single-axis motion controls to generate parameters to execute motion instructions.	SP_SingleAxisPrgOpr on page 42
SingleAxisCtrl (Single-axis Control) function block	Executes motion instructions to operate to the Servomotor according to the motion instruction parameters generated by the Single-axis Program Operation function block.	SingleAxisCtrl on page 143
TorqueToLoad (Torque-to-Load Conversion) function	Convert units between the load cell measurement values and Servomotor torque values to monitor the load value during program operation and perform torque feedback control.	TorqueToLoad on page 172
LoadToTorque (Load-to-Torque Conversion) function		LoadToTorque on page 177

The Single-axis Program Operation function block performs the following four operations.

- Program Status Control  
The program operation data is interpreted, and the order of execution is controlled.
- Step Completion Determination  
The transition condition for the program operation step is checked to determine when to end the step.
- Step Load Alarm Determination  
The load data for each step is checked against a preset condition while program operation is in progress to determine if an alarm state has occurred.
- Program Load Alarm Determination  
The load data is checked against a preset condition while program operation is in progress to determine if an alarm state has occurred.

This library also contains function blocks that perform these operations individually.

The user can combine multiple function blocks to achieve user-specific single-axis program operation.

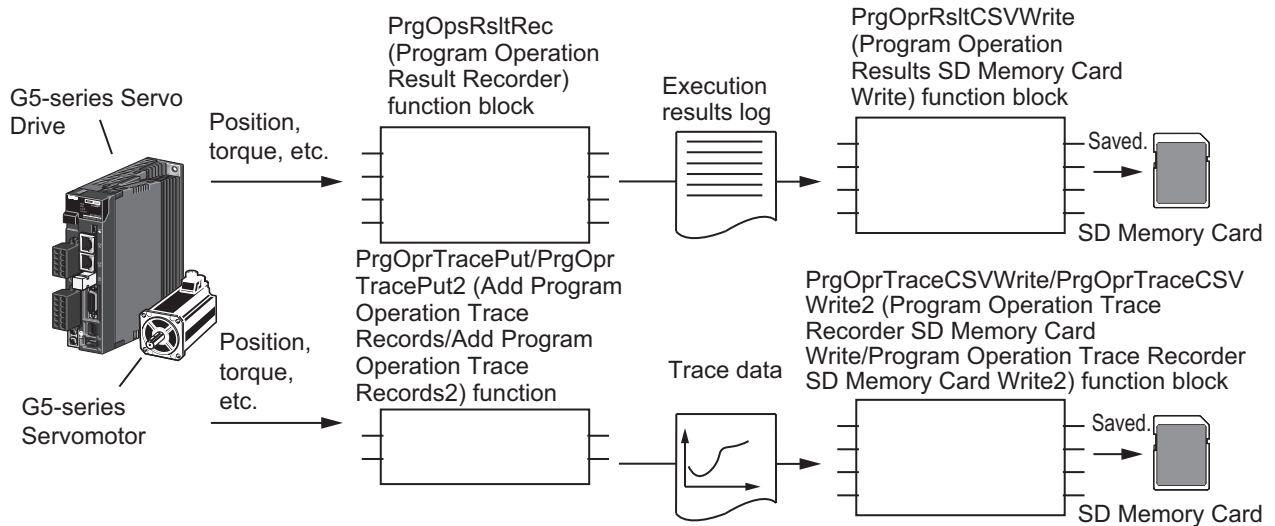


The related functions and function blocks are as follows:

FB/FUN name	Description	Reference
SP_PrgStatusCtrl (Program Status Control) function block	The program operation data is interpreted, and the order of execution is controlled.	<i>SP_PrgStatusCtrl</i> on page 98
SP_StepCompleteJudge (Step Completion Determination) function block	The transition condition for the program operation step is checked to determine when to end the step.	<i>SP_StepCompleteJudge</i> on page 115
SP_StepLoadAlarm (Step Load Alarm Determination) function block	The load data for each step is checked against a preset condition while program operation is in progress to determine if an alarm state has occurred.	<i>SP_StepLoadAlarm</i> on page 127
SP_PrgLoadAlarm (Program Load Alarm Determination) function block	The load data is checked against a preset condition while program operation is in progress to determine if an alarm state has occurred.	<i>SP_PrgLoadAlarm</i> on page 135

## Logging Servo Press Operations, Recording Trace Data, and Saving Data to an SD Memory Card

The result for the operation for each step of single-axis program operation for servo press operation and the trace data for the operation are recorded while operation is in progress. You can also save this data in an SD Memory Card in CSV format.



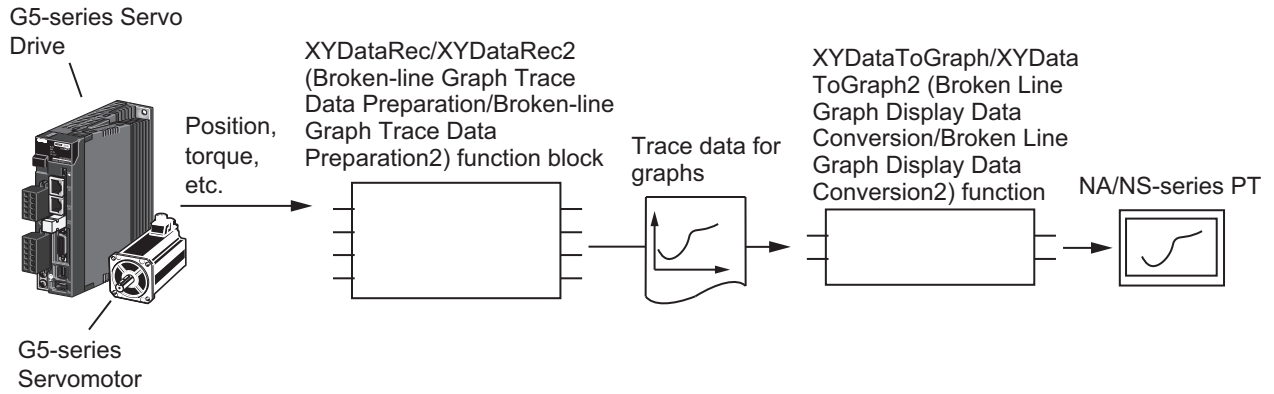
The related functions and function blocks are as follows:

FB/FUN name	Description	Reference
PrgOpsRsltRec (Program Operation Result Recorder) function block	Records the results of the execution of each step of the single-axis program operation as log data.	<i>PrgOprRsltRec</i> on page 181
PrgOprRsltCSVWrite (Program Operation Results SD Memory Card Write) function block	Saves the execution results data recorded by the Program Operation Result Recorder function block in an SD Memory Card in CSV format.	<i>PrgOprRsltCSVWrite</i> on page 191
PrgOprTracePut/ PrgOprTracePut2 (Add Program Operation Trace Records/Add Program Operation Trace Records2) function	Records operation data while single-axis program operation is in progress as sampled data.	<i>PrgOprTracePut</i> on page 200 <i>PrgOprTracePut2</i> on page 207
PrgOprTraceCSVWrite/ PrgOprTraceCSVWrite2 (Program Operation Trace Recorder SD Memory Card Write/Program Operation Trace Recorder SD Memory Card Write2) function block	Saves the trace data recorded with the Add Program Operation Trace Records function in an SD Memory Card in CSV format.	<i>PrgOprTraceCSVWrite</i> on page 215 <i>PrgOprTraceCSVWrite2</i> on page 225



## Servo Press Operation Waveform Data Creation for Graph Display

The data is created to display graphs on an NA/NS-series PT to show the operation waveforms of single-axis program operation for a servo press.



The related functions and function blocks are as follows:

FB/FUN name	Description	Reference
XYDataRec/ XYDataRec2 (Broken-line Graph Trace Data Preparation/Broken-line Graph Trace Data Preparation2) function block	Samples operation data while single-axis operation is in progress and creates trace data for line graph display on the NA/NS-series PT.	XYDataRec on page 235 XYDataRec2 on page 245
XYDataToGraph/ XYDataToGraph2 (Broken Line Graph Display Data Conversion/Broken Line Graph Display Data Conversion2) function	Converts the trace data that was created with the Broken-line Graph Trace Data Preparation function block into a data format that is suitable for displaying as graphs on the NS-series PT.	XYDataToGraph on page 256 XYDataToGraph2 on page 265



# Common Specifications of Function Blocks

# Common Variables

This section describes the specifications of variables (*EN, Execute, Enable, Abort, ENO, Done, CalcRslt, Enabled, Busy, CommandAborted, Error, ErrorID, and ErrorIDEx*) that are used for more than one function or function block. The specifications are described separately for functions, for execute-type function blocks, and for enable-type function blocks.

## Definition of Input Variables and Output Variables

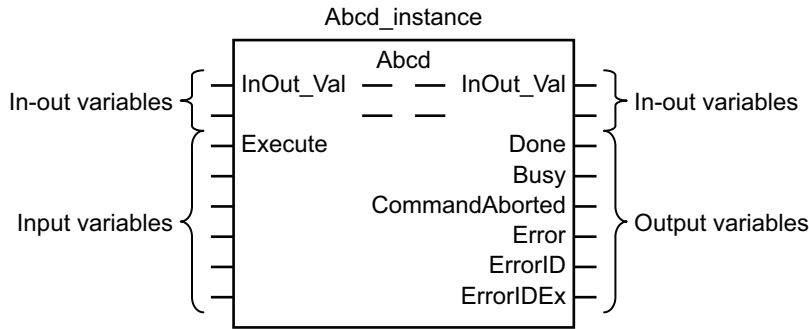
Common input variables and output variables used in functions and function blocks are as follows.

Variable	I/O	Data type	Function/function block type to use			Meaning	Definition
			Function block		Function		
			Execute-type	Enable-type			
EN	Input	BOOL			OK	Execute	The processing is executed while the variable is TRUE.
Execute			OK			Execute	The processing is executed when the variable changes to TRUE.
Enable				OK		Run	The processing is executed while the variable is TRUE.
Abort		BOOL	OK			Abort	The processing is aborted. You can select the aborting method.

Variable	I/O	Data type	Function/function block type to use			Meaning	Definition
			Function block		Function		
			Execute-type	Enable-type			
ENO	Output	BOOL			OK	Done	The variable changes to TRUE when the processing ends normally. It is FALSE when the processing ends in an error, the processing is in progress, or the execution condition is not met.
Done		BOOL	OK			Done	The variable changes to TRUE when the processing ends normally. It is FALSE when the processing ends in an error, the processing is in progress, or the execution condition is not met.
Busy		BOOL	OK	OK		Executing	The variable is TRUE when the processing is in progress. It is FALSE when the processing is not in progress.
CalcRslt		LREAL		OK		Calculation Result	The calculation result is output.
Enabled		BOOL		OK		Enabled	The variable is TRUE when the output is enabled. It is used to calculate the control amount for motion control, temperature control, etc.
Command Aborted		BOOL	OK			Command Aborted	The variable changes to TRUE when the processing is aborted. It changes to FALSE when the processing is re-executed the next time.
Error		BOOL	OK	OK		Error	This variable is TRUE while there is an error. It is FALSE when the processing ends normally, the processing is in progress, or the execution condition is not met.
ErrorID		WORD	OK	OK		Error Code	An error code is output.
ErrorIDEx		DWORD	OK	OK		Expansion Error Code	An expansion error code is output.

## Execute-type Function Blocks

- Processing starts when *Execute* changes to TRUE.
- When *Execute* changes to TRUE, *Busy* also changes to TRUE. When processing is completed normally, *Busy* changes to FALSE and *Done* changes to TRUE.
- When continuously executes the function blocks of the same instance, change the next *Execute* to TRUE for at least one task period after *Done* changes to FALSE in the previous execution.
- If the function block has a *CommandAborted* (Instruction Aborted) output variable and processing is aborted, *CommandAborted* changes to TRUE and *Busy* changes to FALSE.
- If an error occurs in the function block, *Error* changes to TRUE and *Busy* changes to FALSE.
- For function blocks that output the result of calculation for motion control and temperature control, you can use the BOOL input variable *Abort* to abort the processing of a function block. When *Abort* changes to TRUE, *CommandAborted* changes to TRUE and the execution of the function block is aborted.

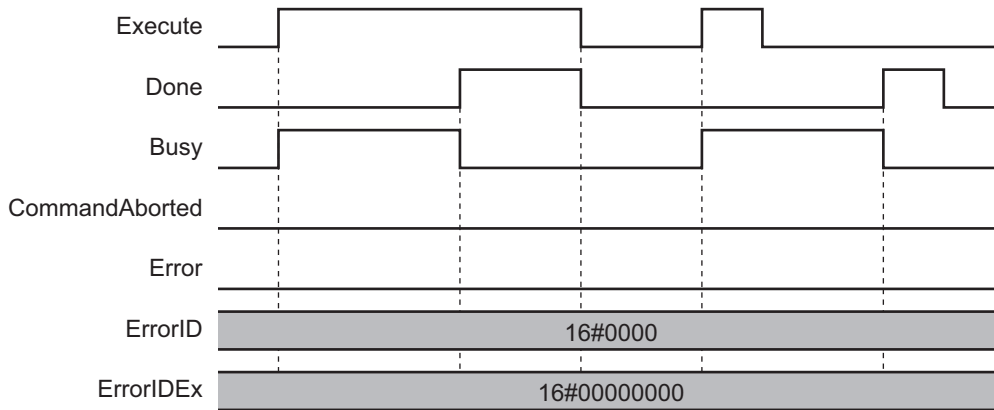


- If *Execute* is TRUE and *Done*, *CommandAborted*, or *Error* changes to TRUE, *Done*, *CommandAborted*, and *Error* changes to FALSE when *Execute* is changed to FALSE.
- If *Execute* is FALSE and *Done*, *CommandAborted*, or *Error* changes to TRUE, *Done*, *CommandAborted*, and *Error* changes to TRUE for only one task period.
- If an error occurs, the relevant error code and expansion error code are set in *ErrorID* (Error Code) and *ErrorIDEx* (Expansion Error Code). The error codes are retained even after *Error* changes to FALSE, but *ErrorID* is set to 16#0000 and *ErrorIDEx* is set to 16#0000 0000 when *Execute* changes to TRUE.

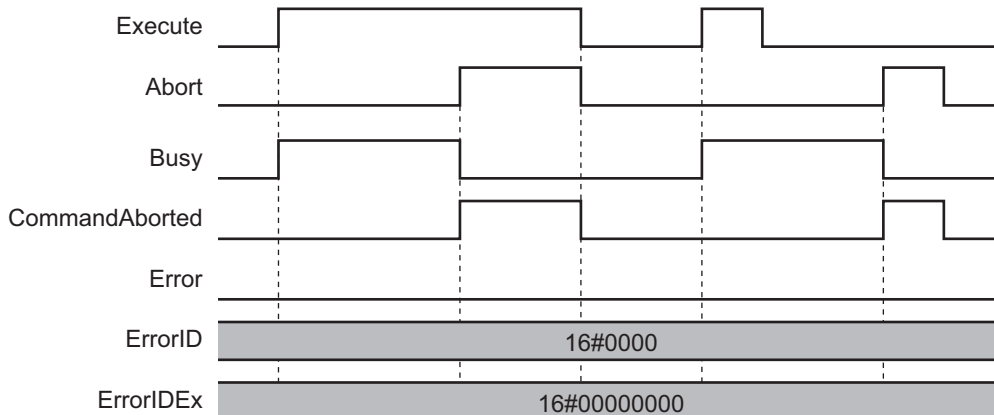
## Timing Charts

This section provides timing charts for a normal end, aborted execution, and errors.

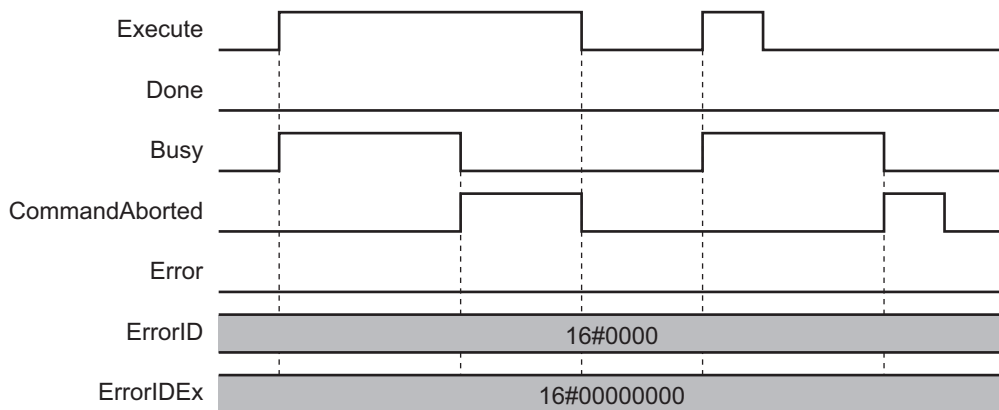
### ● Normal End



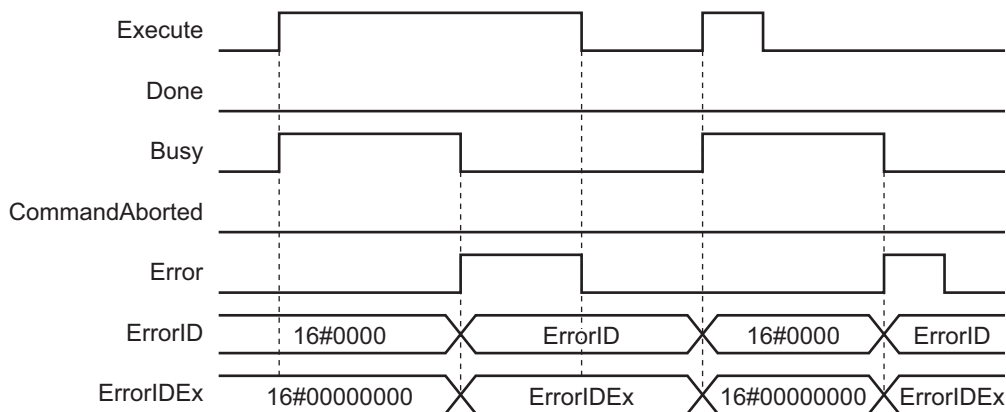
### ● Canceled Execution



● **Aborted Execution**

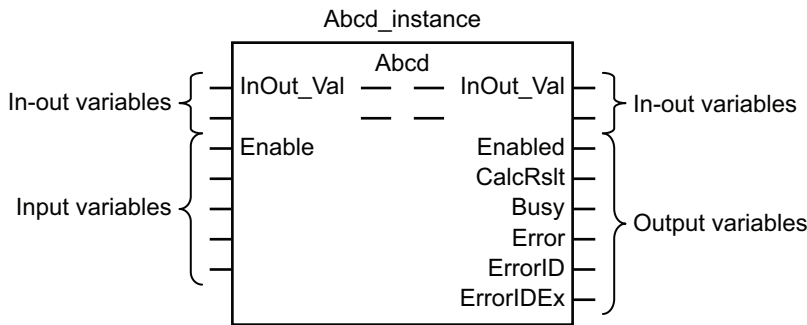


● **Errors**



## Enable-type Function Blocks

- Processing is executed while *Enable* is TRUE.
- When *Enable* changes to TRUE, *Busy* also changes to TRUE. *Enabled* is TRUE during calculation of the output value.
- If an error occurs in the function block, *Error* changes to TRUE and *Busy* and *Enabled* change to FALSE. When *Enable* changes to FALSE, *Enabled*, *Busy*, and *Error* change to FALSE.

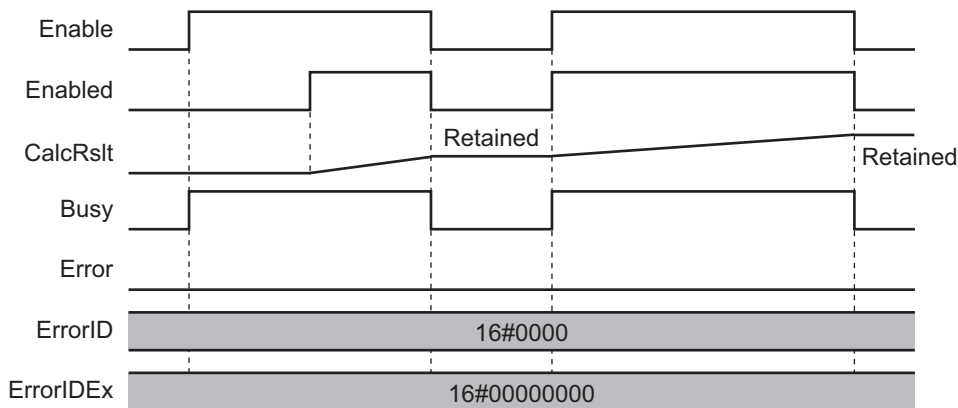


- If an error occurs, the relevant error code and expansion error code are set in *ErrorID* (Error Code) and *ErrorIDEx* (Expansion Error Code). The error codes are retained even after *Error* changes to FALSE, but *ErrorID* is set to 16#0000 and *ErrorIDEx* is set to 16#0000 0000 when *Enable* changes to TRUE.
- For function blocks that calculate the control amount for motion control, temperature control, etc., *Enabled* is FALSE when the value of *CalcRslt* (Calculation Result) is incorrect. In such a case, do not use *CalcRslt*. In addition, after the function block ends normally or after an error occurs, the value of *CalcRslt* is retained until *Enable* changes to TRUE. The control amount will be calculated based on the retained *CalcRslt* value, if it is the same instance of the function block that changed *Enable* to TRUE. If it is a different instance of the function block, the control amount will be calculated based on the initial value.

## Timing Charts

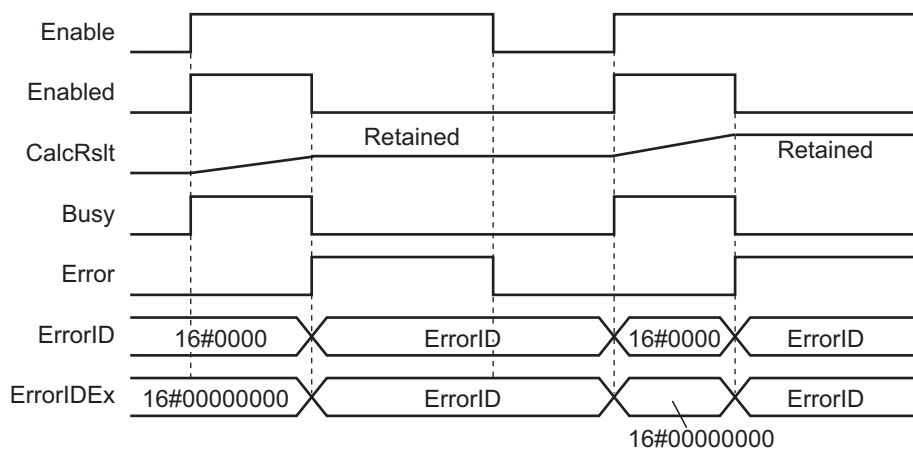
This section provides timing charts for a normal end and errors.

### ● Normal End





● Errors



# Precautions

---

This section provides precautions for the use of this function block.

## Nesting

You can nest calls to this function block for up to four levels.

For details on nesting, refer to the software user's manual.

## Instruction Options

You cannot use the upward differentiation option for this function block.

## Re-execution of Function Blocks

Execute-type function blocks cannot be re-executed by the same instance.

If you do so, the output value will be the initial value.

For details on re-execution, refer to the motion control user's manual.

# Individual Specifications of Function Blocks

Function block name	Name	Page
SP_SingleAxisPrgOpr	Single-axis Program Operation	P.42
SP_PrgStatusCtrl	Program Status Control	P.98
SP_StepCompleteJudge	Step Completion Determination	P.115
SP_StepLoadAlarm	Step Load Alarm Determination	P.127
SP_PrgLoadAlarm	Program Load Alarm Determination	P.135
SingleAxisCtrl	Single-axis Control	P.143
TorqueToLoad	Torque-to-Load Conversion	P.172
LoadToTorque	Load-to-Torque Conversion	P.177
PrgOprRsltRec	Program Operation Results Recorder	P.181
PrgOprRsltCSVWrite	Write Program Operation Results to SD Memory Card	P.191
PrgOprTracePut	Add Program Operation Trace Records	P.200
PrgOprTracePut2	Add Program Operation Trace Records 2	P.207
PrgOprTraceCSVWrite	Write from Program Operation Trace Recorder to SD Memory Card	P.215
PrgOprTraceCSVWrite2	Write from Program Operation Trace Recorder to SD Memory Card 2	P.225
XYDataRec	Broken Line Graph Trace Data Preparation	P.235
XYDataRec2	Broken Line Graph Trace Data Preparation 2	P.245
XYDataToGraph	Broken Line Graph Display Data Conversion	P.256
XYDataToGraph2	Broken Line Graph Display Data Conversion 2	P.265

# SP\_SingleAxisPrgOpr

The SP\_SingleAxisPrgOpr function block executes single-axis program operation that combines multiple single-axis motion controls.

Function block name	Name	FB/FUN	Graphic expression	ST expression
SP_SingleAxisPrgOpr	Single-axis Program Operation	FB		<pre>SP_SingleAxisPrgOpr_instance (   PrgTable,   PrgLoadAlarm,   Execute,   Abort,   StartStepNo,   SingleMode,   Position,   Velocity,   Load,   MCCmdDone,   StepCompleteCode,   SingleCmdProfileNo,   Done,   MCCmdExec,   SingleCmdProfile,   CurrentStepNo,   StepLoadAlarmOut,   PrgLoadAlarmOut,   ExtrOutputCode,   StepExec,   StepCompleted,   Busy,   CommandAborted,   Error,   ErrorID,   ErrorIDEx);</pre>

## Function Block and Function Information

Item	Description
Library file name	OmronLib_ServoPress_V1_0.slr OmronLib_ServoPress_V2_0.slr
Namespace	OmronLib\ServoPress
Function block and function number	00071
Publish/Do not publish source code	Not published.
Function block and function version	1.00

## Variables

### Input Variables

Name	Meaning	Data type	Default	Valid range	Unit	Description
Execute	Execute	BOOL	FALSE	Depends on data type.	---	Execute trigger for this function block Executes the function block when it changes to TRUE.
Abort	Abort	BOOL	FALSE	Depends on data type.	---	Abort trigger for this function block Aborts the function block when it changes to TRUE.
StartStepNo	Execution Start Step Number	USINT	0	0 to 50	---	Step number from which to start execution
SingleMode	Single Mode	BOOL	FALSE	Depends on data type.	---	This variable indicates whether to execute operation in Single Mode. TRUE: Execute operation in Single Mode. FALSE: Do not execute operation in Single Mode.
Position	Current Position	LREAL	0	Depends on data type.	Command units	The current position monitor input for the controlled system.
Velocity	Current Velocity	LREAL	0	Depends on data type.	Command units/s	The current velocity monitor input for the controlled system.
Load	Current Load	LREAL	0	Depends on data type.	Load units <sup>*1</sup>	The current load monitor input for the controlled system.
MCCmd Done	Motion Instruction Completion	BOOL	FALSE	Depends on data type.	---	Motion instruction completion TRUE: Completed FALSE: Not completed.
Step Complete Code	Step Complete Code	USINT	0	0 to 255	---	This code is matched to the step completion condition.
SingleCmd ProfileNo	Single-axis Command Profile Number	USINT	0	0 to 10	---	When you consecutively execute multiple single-axis motion control instructions, this number indicates the ordinal number of the current single-axis motion control instruction.

\*1. The unit of load is not specified in this function block. Uniquely set newtons (N) or % as a torque conversion value in the user program that uses this function block.

## Output Variables

Name	Meaning	Data type	Valid range	Unit	Description
Done	Done	BOOL	Depends on data type.	---	Completion of execution TRUE: Normal end FALSE: Error end, execution in progress, or execution condition not met
MCCmdExec	Motion Function Block Execution Trigger	ARRAY[0..9] OF BOOL	Depends on data type.	---	Execution trigger for next motion function block
SingleCmdProfile	Single-axis Command Profile	Omron-Lib\Servo Press\sINGLE_CMD_PROFILE	---	---	Single-axis control instruction table for each step
CurrentStepNo	Current Step Number	USINT	0 to 50	---	The number of the step being executed.
StepLoadAlarm Out	Step Load Alarm	BOOL	Depends on data type.	---	This variable indicates whether a step load alarm occurred in the current step. TRUE: A step load alarm occurred. FALSE: A step load alarm did not occur.
PrgLoadAlarm Out	Program Load Alarm	BOOL	Depends on data type.	---	This variable indicates whether a program load alarm occurred in the current single-axis program operation. TRUE: A program load alarm occurred. FALSE: A program load alarm did not occur.
ExtrOutputCode	External Output Code	USINT	0 to 255	---	Code that is output for each step that is completed.
StepExec	Step Start Trigger	BOOL	Depends on data type.	---	The start trigger for the step.
StepCompleted	Step Completed Trigger	BOOL	Depends on data type.	---	The completion trigger for the step.
Busy	Executing	BOOL	Depends on data type.	---	Executing TRUE: Executing FALSE: Not executing
Command Aborted	Instruction Aborted	BOOL	Depends on data type.	---	Execution aborted This variable changes to TRUE if the instruction is aborted.

Name	Meaning	Data type	Valid range	Unit	Description
Error	Error	BOOL	Depends on data type.	---	Error end TRUE: Error end FALSE: Normal end, executing, or execution conditions not met
ErrorID	Error Code	WORD	*1	---	An error code is output if an error occurs. The value is WORD#16#0 for a normal end.
ErrorIDEx	Expansion Error Code	DWORD	*1	---	An expansion error code is set if an error occurs. The value is DWORD#16#0 for a normal end.

\*1. For details, refer to *Troubleshooting* on page 81.

### ● Structure

The data type of the *SingleCmdProfile* output variable is the structure `OmronLib\ServoPress\SINGLE_CMD_PROFILE`. The specifications are as follows:

Name	Meaning	Description	Data type	Valid range	Unit	Default
SingleCmdProfile	Single-axis Command Profile	Single-axis control instruction tables	OmronLib\ServoPress\sSINGLE_CMD_PROFILE	---	---	---
SingleCmdData	Single-axis Command Data	Single-axis control instruction profile datas	ARRAY[1..10] OF OmronLib\ServoPress\sSTEP_CMD_DATA			
CtrlCode	Control Method	The control method. 0: Deceleration stop 1: Absolute positioning 2: Relative positioning 3: CSV mode Velocity control 4: Torque control 5: Torque feedback control	USINT	0 to 5	---	0
Position	Position Set Value	The position set value.	LREAL	*1	Command units	0
Velocity	Velocity Set Value	The velocity set value.	LREAL	*1	Command units/s	0
Acceleration	Acceleration Rate Set Value	The acceleration rate set value.	LREAL	*1	Command units/s <sup>2</sup>	0
Deceleration	Deceleration Rate Set Value	The deceleration rate set value.	LREAL	*1	Command units/s <sup>2</sup>	0
Jerk	Jerk Set Value	The jerk set value.	LREAL	*1	Command units/s <sup>3</sup>	0

Name	Meaning	Description	Data type	Valid range	Unit	Default
Direction	Direction	Direction _mcPositiveDirection: Positive direction _mcShortestWay: Shortest way _mcNegativeDirection: Negative direction _mcCurrentDirection: Current direction _mcNoDireciton: No direction specified	_eMC _DIRECTION	*1	---	_mcPositiveDirection
BufferMode	Buffer Mode Selection	The buffer mode selection. _mcAborting: Aborting _mcBuffered: Buffered _mcBlendingLow: Blending low _mcBlendingPrevious: Blending previous _mcBlendingNext: Blending next _mcBlendingHigh: Blending high	_eM- C_BUFFER_ MODE	*1	---	_mcAborting
Torque	Torque Set Value	The torque set value.	LREAL	*1	%	0
Torque Ramp	Torque Ramp Set Value	The torque ramp set value.	LREAL	*1	%/s	0
Torque Limit Positive Enable	Enable Positive Torque Limit	This variable indicates whether to enable the positive torque limit. TRUE: Enabled. FALSE: Disabled	BOOL	Depends on data type.	---	FALSE
Torque Limit Negative Enable	Enable Negative Torque Limit	This variable indicates whether to enable the negative torque limit. TRUE: Enabled. FALSE: Disabled	BOOL	Depends on data type.	---	FALSE
Torque Limit Positive Val	Positive Torque Limit	The positive torque limit.	LREAL	0.0 or 0.1 to 1000.0	%	300.0
Torque Limit Negative Val	Negative Torque Limit	The negative torque limit.	LREAL	0.0 or 0.1 to 1000.0	%	300.0
TorqueFbk Params	Torque Feedback Parameters	Torque feedback control parameters	OmronLib\ ServoPress\ sTORQUE_F BK_PARAMS	*1	---	---
Kp	Proportional Gain	The proportional gain.	LREAL	0.0 to 3000.0	---	1.0



Name	Meaning	Description	Data type	Valid range	Unit	Default
Ki	Integral Gain	The integral gain.*2	LREAL	0.0 to 3000.0	---	1.0
Kd	Derivative Gain	The derivative gain.*3	LREAL	0.0 to 3000.0	---	1.0
Torque LowLmt	Torque Lower Limit	The output torque lower limit.	LREAL	-1,000.0 to 0.0*4	0.1%	-300.0
TorqueUp Lmt	Torque Upper Limit	The output torque upper limit.	LREAL	0.0 to 1000.0*4	0.1%	300.0
InTorque Width	In Torque Width	The width for determining if the target torque is reached.	LREAL	0.0 to 100.00*4	0.1%	0.1

\*1. The valid range depends on the value of *CtrlCode* (Control Method). For details on the valid range, refer to *Valid Ranges of SingleCmdProfile (Single-axis Command Profile) Members* on page 67.

\*2. The integration time is 1 s.

\*3. The derivative time is 1 s.

\*4. The value is rounded to the second decimal place.

## In-Out Variables

Name	Meaning	Data type	Valid range	Unit	Description
PrgTable	Program Tables	ARRAY[1..50] OF OmronLib\ ServoPress\ sPRG_TABLE	---	---	Program tables that contains settings for each step.
PrgLoadAlarm	Program Load Alarm Conditions	OmronLib\ ServoPress\ sPRG_LOAD_ALARM	---	---	Conditions necessary for program load alarm to occur

● **Structure**

The data type of the *PrgTable* in-out variable is the structure OmronLib\ServoPress\sPRG\_TABLE.  
The specifications are as follows:

Name	Meaning	Description	Data type	Valid range	Unit	Default
PrgTable	Program Tables	Program tables that contains settings for each step.	ARRAY[1..50] OF OmronLib\ServoPress\sPRG_TABLE	---	---	---
StepCmd Data	Step Command Data	Instruction data for each step	OmronLib\ServoPress\sSTEP_CMD_DATA	---	---	---
CtrlCode	Control Method	Control method 0: Single-axis program operation completion 1: Absolute positioning 2: Relative positioning 3: CSV mode velocity control 4: Torque control 5: Torque feedback control 6: Operation hold 7: Skip 8 to 30: Reserved 31 to 39: User defined	USINT	0 to 7 or 31 to 39	---	0
Position	Position Set Value	The position set value.	LREAL	*1	Command units*2	0
Velocity	Velocity Set Value	The velocity set value.	LREAL	*1	Command units/s	0
Acceleration	Acceleration Rate Set Value	The acceleration rate set value.	LREAL	*1	Command units/s <sup>2</sup>	0
Deceleration	Deceleration Rate Set Value	The deceleration rate set value.	LREAL	*1	Command units/s <sup>2</sup>	0
Jerk	Jerk Set Value	The jerk set value.	LREAL	*1	Command units/s <sup>3</sup>	0
Direction	Direction	Direction _mcPositiveDirection: Positive direction _mcShortestWay: Shortest way _mcNegativeDirection: Negative direction _mcCurrentDirection: Current direction _mcNoDirrection: No direction specified	_eMC_DIRECTION	*1	---	_mcPositiveDirrection

Name	Meaning	Description	Data type	Valid range	Unit	Default
Buffer Mode	Buffer Mode Selection	The buffer mode selection. _mcAborting: Aborting _mcBuffered: Buffered _mcBlendingLow: Blending low _mcBlendingPrevious: Blending previous _mcBlendingNext: Blending next _mcBlendingHigh: Blending high	_eMC_BUFFER_MODE	*1	---	_mcAborting
Torque	Torque Set Value	The torque set value.	LREAL	*1	%	0
Torque Ramp	Torque Ramp Set Value	The torque ramp set value.	LREAL	*1	%/s	0
Torque Limit Positive Enable	Enable Positive Torque Limit	This variable indicates whether to enable the positive torque limit. TRUE: Enabled FALSE: Disabled	BOOL	Depends on data type.	---	FALSE
Torque Limit Negative Enable	Enable Negative Torque Limit	This variable indicates whether to enable the negative torque limit. TRUE: Enabled FALSE: Disabled	BOOL	Depends on data type.	---	FALSE
Torque Limit Positive Val	Positive Torque Limit	The positive torque limit.	LREAL	0.0 or 0.1 to 1000.0	%	300
Torque Limit Negative Val	Negative Torque Limit	The negative torque limit.	LREAL	0.0 or 0.1 to 1000.0	%	300
Torque Fbk Params	Torque Feedback Parameters	Torque feedback control parameters	OmronLib\Servo-Press\TORQUE_FBK_PARAMS	*1	---	---
Kp	Proportional Gain	Proportional gain	LREAL	0.0 to 3,000.0	---	1.0
Ki	Integral Gain	Integral gain <sup>*3</sup>	LREAL	0.0 to 3,000.0	---	1.0
Kd	Derivative Gain	Derivative gain <sup>*4</sup>	LREAL	0.0 to 3,000.0	---	1.0
Torque LowLmt	Torque Lower Limit	The output torque lower limit.	LREAL	-1,000.0 to 0.0 <sup>*5</sup>	0.1%	-300.0
TorqueUp Lmt	Torque Upper Limit	The output torque upper limit.	LREAL	0.0 to 1,000.00 <sup>*5</sup>	0.1%	300.0
InTorque Width	In Torque Width	The width for determining if the target torque is reached.	LREAL	0.0 to 100.00 <sup>*5</sup>	0.1%	0.1

Name	Meaning	Description	Data type	Valid range	Unit	Default
Extr Output Code	External Output Code	Code that is output for each step that is completed.	USINT	0 to 255	---	0
Step Complete Data	Step Completion Condition	The completion condition for step.	Omron-Lib\ServoPress\sSTEP_COMPLETE_DATA	---	---	---
Complete Type	Step Completion Type	Completion type for step 0: Wait for completion of motion instruction 1: Target absolute position 2: Target relative position 3: Target velocity 4: Target load or higher 5: Target load or lower 6: Step complete code match 7: Continuous detection of position load gradient 8: Continuous load decrease 9: Continuous load increase 10: Continuous position control target 11: Wait 12 to 30: Reserved 31 to 39: User defined	USINT	0 to 11 or 31 to 39	---	0
TimeOut	Step Timeout Time	The timeout time for the step. 0: Do not monitor for timeouts.	TIME	Positive number or 0	ms	0
TimeOut NextStep No	Step Timeout Next Step Number	The step number to execute after step timeout. 0: End single-axis program operation.	USINT	0 to 50	---	0
InPos Width	Positioning In-position Width	The positioning in-position width when value of <i>CompleteType</i> is 1, 2, 3, 4, 5, or 7.	LREAL	0.0 to 1000.0	Command units for monitoring	0
Absolute Position	Target Absolute Position	The target absolute position when value of <i>CompleteType</i> is 1.	LREAL	Depends on data type.	Command units for monitoring	0
Relative Position	Target Relative Position	The target relative position when value of <i>CompleteType</i> is 2.	LREAL	Depends on data type.	Command units for monitoring	0
Velocity	Target Velocity	The target velocity when value of <i>CompleteType</i> is 3.	LREAL	Depends on data type.	Command units per second for monitoring	0

Name	Meaning	Description	Data type	Valid range	Unit	Default
Load	Target Load	The target load when value of <i>CompleteType</i> is 4 or 5.	LREAL	Depends on data type.	Load units* <sup>6</sup>	0
Step Complete Code	Step Complete Code Set Value	The step complete code set value when value of <i>CompleteType</i> is 6.	USINT	Depends on data type.	---	0
InflPoint Gradient	Position Load Gradient	The gradient of position load that determines that step is completed when value of <i>CompleteType</i> is 7.	LREAL	Depends on data type.	Load units* <sup>6</sup> / Command units	---
Monitor MinPos	Minimum Monitoring Position	The absolute position to start monitoring velocity or load when value of <i>CompleteType</i> is 3, 4, 5, or 7.	LREAL	Depends on data type.	Command units	0
Monitor MaxPos	Maximum Monitoring Position	The absolute position to stop monitoring velocity or load when value of <i>CompleteType</i> is 3, 4, 5, or 7.	LREAL	Depends on data type.	Command units	0
Monitor StartTime	Monitoring Start Time	The elapsed time from start of step until load monitoring starts when value of <i>CompleteType</i> is 8 or 9.	TIME	Positive number or 0	ms	0
Monitor EndTime	Monitoring End Time	The elapsed time from start of step until load monitoring ends when value of <i>CompleteType</i> is 8 or 9.	TIME	Positive number or 0	ms	0
InflPoint Gradient Count	Number of Consecutive Position Load Gradients	The threshold of number of consecutive position load gradients that determines that step is completed when value of <i>CompleteType</i> is 7.	USINT	1 to 255	---	1
Load Decrease Count	Number of Consecutive Load Decreases Threshold	The threshold of number of consecutive load decreases that determines that step is completed when value of <i>CompleteType</i> is 8.	USINT	1 to 255	---	1
Load Increase Count	Number of Consecutive Load Increases Threshold	The threshold of number of consecutive load increases that determines that step is completed when value of <i>CompleteType</i> is 9.	USINT	1 to 255	---	1
UnachievedNext StepNo	Step Number When Step Not Completed	The next step number to execute when value of <i>CompleteType</i> is 3, 4, 5, 7, 8, or 9 and step was not completed. 0: End single-axis program operation.	USINT	0 to 50	---	0
WaitTime	Wait Time	The wait time when value of <i>CompleteType</i> is 11.	TIME	Positive number or 0	ms	0

Name	Meaning	Description	Data type	Valid range	Unit	Default
StepLoad Alarm	Step Load Alarm Conditions	The conditions for step load alarm.	Omron-Lib\ServoPress\sSTEP_LOAD_ALARM	---	---	---
StepAlarm Type	Step Load Alarm Determination Type	The type of step load alarm determination. 0: Do not perform determination. 1: Perform determination while step execution is in progress. 2: Perform determination when step is completed. 3 to 30: Reserved 31 to 39: User defined	USINT	0 to 2 or 31 to 39	---	0
Monitor MinPos	Monitoring Range Minimum Position	The minimum position of monitoring range.	LREAL	Depends on data type.	Reference units for monitoring <sup>*7</sup>	0
Monitor MaxPos	Monitoring Range Maximum Position	The maximum position of monitoring range.	LREAL	Depends on data type.	Reference unit for monitoring. <sup>*7</sup>	0
InPos Width	Upper/Lower Monitoring Limit In-position Width	The upper/lower monitoring limit in-position width.	LREAL	0.0 to 1000.0	Reference units for monitoring <sup>*7</sup>	0
Monitor MinLoad	Monitoring Load Range Lower Limit	The lower limit of load that determines that no step load alarm occurred.	LREAL	Depends on data type.	Load units <sup>*6</sup>	0
Monitor MaxLoad	Monitoring Load Range Upper Limit	The upper limit of load that determines that no step load alarm occurred.	LREAL	Depends on data type.	Load units <sup>*6</sup>	0

\*1. The valid range depends on the value of *CtrlCode* (Control Method). For details on the valid range, refer to *Valid Ranges of SingleCmdProfile (Single-axis Command Profile) Members* on page 67.

\*2. For details, refer to Unit Conversion Settings in the motion control user's manual.

\*3. The integration time is 1 s.

\*4. The derivative time is 1 s.

\*5. The value is rounded to the second decimal place.

\*6. The unit of load is not specified in this function block. Uniquely set newtons (N) or % as a torque conversion value in the user program that uses this function block.

\*7. This reference unit depends on the target axis parameter unit setting. It is specified as the absolute position from the axis home.

The data type of the *PrgLoadAlarm* in-out variable is the structure OmronLib\ServoPress\sPRG\_LOAD\_ALARM. The specifications are as follows:

Name	Meaning	Description	Data type	Valid range	Unit	Default
PrgLoad Alarm	Program Load Alarm Conditions	The program load alarm conditions.	Omron-Lib\ServoPress\sPRG_LOAD_ALARM	---	---	---
PrgAlarm Type	Program Load Alarm Determination Type	The type of program load alarm determination. 0: Do not perform determination. 1: Trapezoid area determination 2: Rectangle area determination 3 to 30: Reserved 31 to 39: User defined	USINT	0 to 2 or 31 to 39	---	0
UnitType	Standard Type	The standard type for determination. 0: Elapsed time from start of execution of the function block 1: Current position of axis	USINT	0 or 1	---	0
Trapezoid Data	Trapezoid Area Data	The trapezoid area data.	ARRAY[0..4] OF sPLA_TRAPEZ_DATA	---	---	---
Alarm Type	Alarm Type	The type of determination condition for trapezoid area data. 0: Do not perform determination. 1: Outside area 2: Inside area	USINT	0 to 2	---	0
Monitor Lower Time	Monitoring Lower Limit Elapsed Time	The elapsed time <sup>*1</sup> until monitoring starts when value of <i>UnitType</i> is 0.	TIME	Positive number or 0	ms	0
Monitor Upper Time	Monitoring Upper Limit Elapsed Time	The elapsed time <sup>*1</sup> until monitoring ends when value of <i>UnitType</i> is 0.	TIME	Positive number or 0	ms	0
Monitor Lower Pos	Minimum Monitoring Position	The lower limit position of monitoring range when value of <i>UnitType</i> is 1.	LREAL	Depends on data type.	Reference units for monitoring	0
Monitor Upper Pos	Maximum Monitoring Position	The upper limit position of the monitoring range when the value of <i>UnitType</i> is 1.	LREAL	Depends on data type.	Reference units for monitoring	0
LoadMin Lower	Monitoring Start Point Minimum Load	The lower limit of the load at the monitoring start point.	LREAL	Depends on data type.	Load units <sup>*2</sup>	0

Name	Meaning	Description	Data type	Valid range	Unit	Default
LoadMax Lower	Monitoring Start Point Maximum Load	The upper limit of the load at the monitoring start point.	LREAL	Depends on data type.	Load units <sup>*2</sup>	0
LoadMin Upper	Monitoring End Point Minimum Load	The lower limit of the load at the monitoring end point.	LREAL	Depends on data type.	Load units <sup>*2</sup>	0
LoadMax Upper	Monitoring End Point Maximum Load	The upper limit of the load at the monitoring end point.	LREAL	Depends on data type.	Load units <sup>*2</sup>	0
Rectangle Data	Rectangle Area Data	The rectangle area data.	ARRAY[0..9] OF sPLA_RECTA_DATA	---	---	---
Alarm Type	Alarm Type	The type of determination condition for trapezoid area data. 0: Do not perform determination. 1: Outside area 2: Inside area	USINT	0 to 2	---	0
Monitor Lower Time	Monitoring Lower Limit Elapsed Time	The elapsed time <sup>*1</sup> until monitoring starts when value of <i>UnitType</i> is 0.	TIME	Positive number or 0	ms	0
Monitor Upper Time	Monitoring Upper Limit Elapsed Time	The elapsed time <sup>*1</sup> until monitoring ends when value of <i>UnitType</i> is 0.	TIME	Positive number or 0	ms	0
Monitor Lower Pos	Minimum Monitoring Position	The lower limit position of monitoring range when value of <i>UnitType</i> is 1.	LREAL	Depends on data type.	Reference units for monitoring	0
Monitor Upper Pos	Maximum Monitoring Position	The upper limit position of the monitoring range when the value of <i>UnitType</i> is 1.	LREAL	Depends on data type.	Reference units for monitoring	0
LoadMin Limit	Load Lower Limit	The lower limit of the load.	LREAL	Depends on data type.	Load units <sup>*2</sup>	0
LoadMax Limit	Load Upper Limit	The upper limit of the load.	LREAL	Depends on data type.	Load units <sup>*2</sup>	0
InPos Width	Upper/Lower Monitoring Limit In-position Width	The upper/lower limit in-position range when the value of <i>UnitType</i> is 1.	LREAL	0.0 to 1000.0	Reference units for monitoring	0



- \*1. This parameter gives the elapsed time from when execution of the function block starts.
- \*2. The unit of load is not specified in this function block. Uniquely set newtons (N) or % as a torque conversion value in the user program that uses this function block.

## Function

This function block executes single-axis program operation that combines multiple motion controls, including position control, velocity control, torque control, and torque feedback control in combination with the SingleAxisCtrl (Single-axis Control) function block.

This function block also performs the following.

- Performs step load alarm determination for each step of the single-axis program operation.
- Performs program load alarm determination through the entire single-axis program operation.

This section describes the following items.

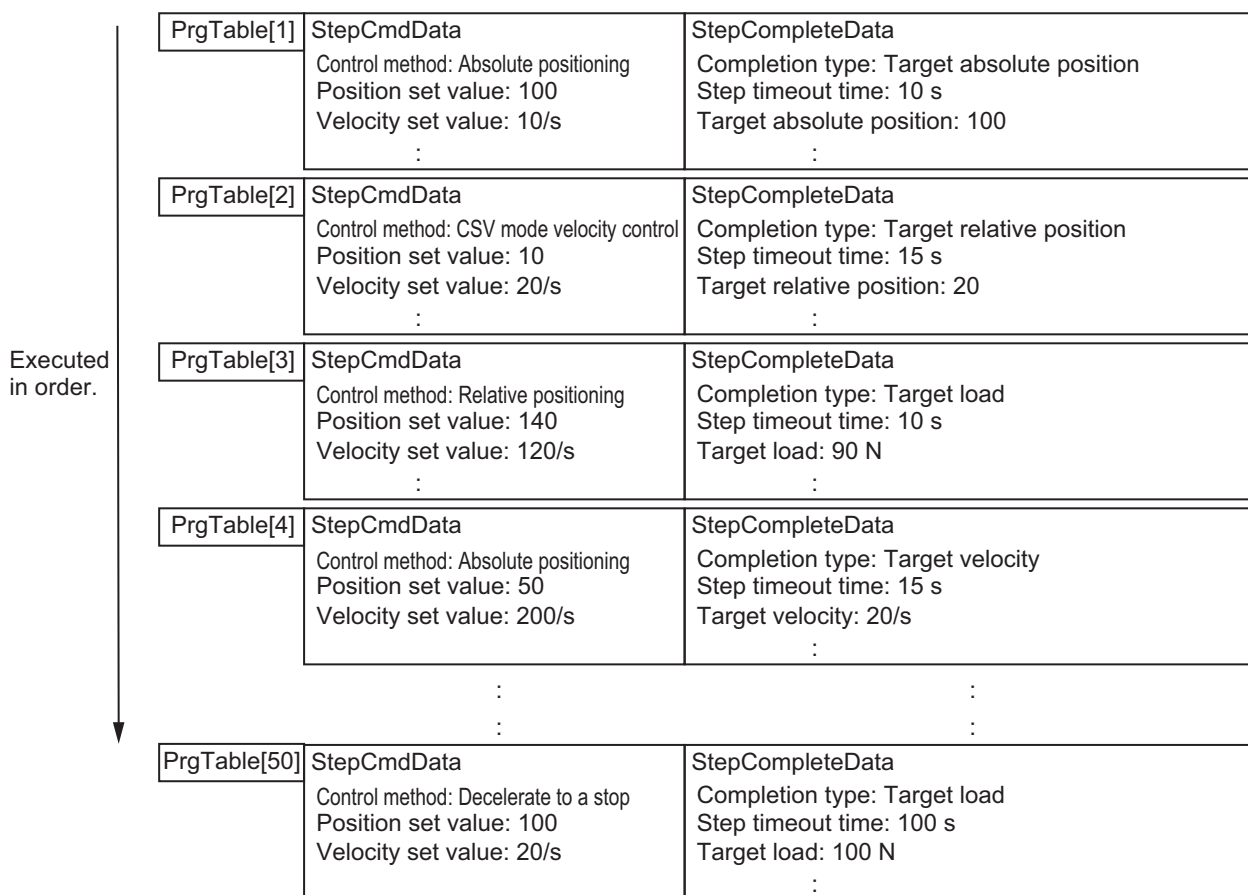
- Defining single-axis program operation that combines multiple single-axis motion controls
- Step command data
- Step completion condition
- Connections with the SingleAxisCtrl (Single-axis Control) function block
- Step load alarm determination
- Program load alarm determination

## Defining Single-axis Program Operation That Combines Multiple Single-axis Motion Controls

Use the *PrgTable* (Program Tables) in-out variable to define the single-axis program operation that combines multiple single-axis motion controls.

### ● *PrgTable* (Program Tables)

*PrgTable* is a structure array that gives the settings for a series of single-axis motion controls in order of execution. The following conceptual diagram illustrates *PrgTable*.



The members of *PrgTable* include *StepCmdData* (Step Command Data), *StepCompleteData* (Step Completion Condition), and *StepLoadAlarm* (Step Load Alarm Conditions). Of these, step load alarm determination is described below for *StepLoadAlarm*.

*StepCmdData* is a data set that describes aspects of individual motion controls, such as the control method, position set value, and velocity set value.

*StepCompleteData* is a data set that describes the completion conditions for the motion controls that are given in *StepCmdData*.

When single-axis program control is started, the motion control given in *PrgTable[1].StepCmdData* is executed first. When the completion condition given in *PrgTable[1].StepCompleteData* is met, the motion control given in *PrgTable[2].StepCmdData* is executed next. This function block is completed (i.e., *Done* is TRUE) if one of the following conditions is met.

- The condition satisfied *PrgTable.StepCompleteData* (Step Completion Condition) in the step for which *CtrlCode* is set to 0 (single-axis program operation completion).  
 (Example 1) In the step for which *CtrlCode* is set to 0 (single-axis program operation completion) and *CompleteType* is set to 0 (wait for completion of motion instruction), after the completion of the MC\_Stop instruction (*CtrlCode* is set to 0) that is executed in the SingleAxisCtrl function block, *Done* changes to TRUE and this function block is completed.

(Example 2) In the step for which *CtrlCode* is set to 0 (single-axis program operation completion) and *CompleteType* is set to 1 (target absolute position), if the *Position* (Current Position) input variable in this function block becomes to *AbsolutePosition* (i.e., a condition of end) according the MC\_Stop instruction executed in the SingleAxisCtrl function block, *Done* changes to TRUE and this function block is completed.

- Controls up to *PrgTable[50]* are executed.
- Execution of the step specified with *StartStepNo* (Execution Start Step Number) is completed when *SingleMode* is TRUE.
- A timeout occurred during a step for which *TimeOutNextStepNo* is set to USINT#0.
- A step for which *UnachievedNextStepNo* is set to USINT#0 was not completed.

The stages of single-axis program operation given in the array elements of *PrgTable* are called steps. Also, the element numbers in *PrgTable* are called the step numbers. The flow of a series of steps is called the step sequence.

The *PrgTable* array size is 50. Therefore, you can set a maximum of 50 single-axis program operation steps.



#### Precautions for Safe Use

---

If *UnachievedNextStepNo* is set to 0 and the step completion conditions are not met, *Done* changes to TRUE and this function block is completed. At this time, the MC\_Stop (Stop) instruction will not be executed for the next SingleAxisCtrl (Single-axis Control) function block. Set *UnachievedNextStepNo* to a value other than 0 and specify the operation if the step completion conditions are not met.

---

## Step Command Data

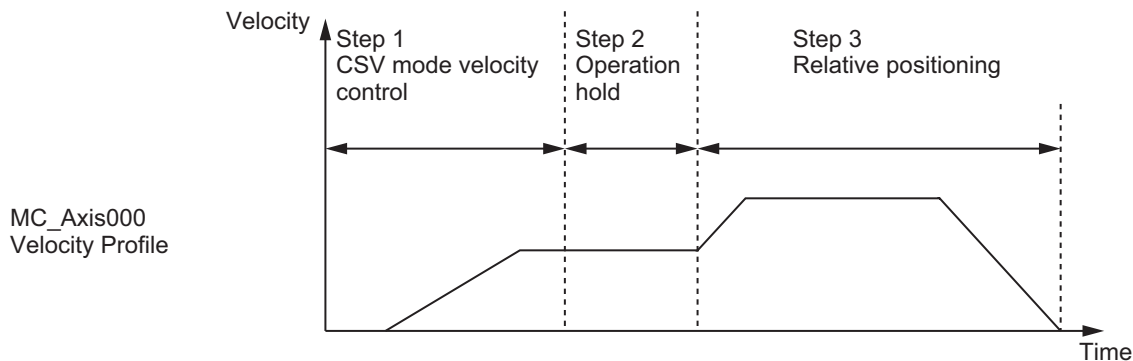
Specify the motion controls to execute for the steps with *PrgTable.StepCmdData* (Step Command Data).

Specify the control methods for motion controls with *CtrlCode* (Control Method). If the value of *CtrlCode* is between 0 to 5, the motion control instructions or motion control in the SingleAxisCtrl (Single-axis Control) function block that is connected after this function block are executed. The following table gives the relation between the value of *CtrlCode*, the control method, and the motion control instruction that is executed in the SingleAxisCtrl (Single-axis Control) function block.

Value of <i>CtrlCode</i>	Control method	Motion control instruction or motion control function
0	Single-axis program operation completion	MC_Stop Instruction
1	Absolute positioning	MC_MoveAbsolute (Absolute Positioning) instruction
2	Relative positioning	MC_MoveRelative (Relative Positioning) instruction
3	CSV mode velocity control	CSV mode velocity control
4	Torque control	MC_TorqueControl (Torque Control) instruction
5	Torque feedback control	Torque feedback control
6	Operation hold	---
7	Skip	---

### ● Operation Hold

When *CtrlCode* is set to USINT#6, motion control instructions are not executed in the SingleAxisCtrl (Single-axis Control) function block. Therefore, the motion control instruction operation that was executed in the previous step is held.



### ● Skip

If *CtrlCode* is set to USINT#7, the function block immediately moves to the next step. This control method is called skipping.

## Step Completion Condition

Specify the completion conditions for the steps with *PrgTable.StepCompleteData* (Step Completion Condition).

Specify the step completion type with *CompleteType* (Step Completion Type). The following table shows the relation between the value of *CompleteType* and the step completion type.

Value of <i>CompleteType</i>	Step completion type
0	Wait for completion of motion instruction
1	Target absolute position
2	Target relative position
3	Target velocity
4	Target load or higher
5	Target load or lower
6	Step complete code match
7	Continuous detection of position load gradient
8	Continuous load decrease
9	Continuous load increase
10	Continuous position control target
11	Wait

The meanings of the step completion types and the meaning of timeout are explained below.

### ● Wait for Completion of Motion Instruction (*CompleteType* = USINT#0)

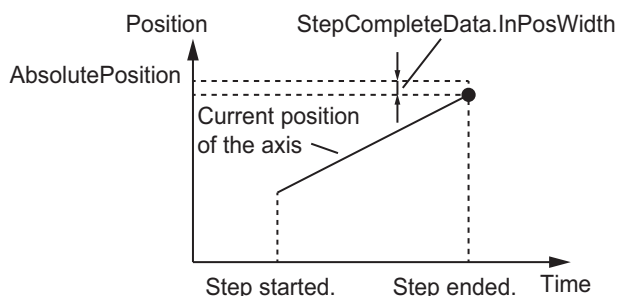
When the *MCCmdDone* (Motion Instruction Completion) input variable changes to TRUE, the step is ended.

This completion condition is selected when the completion of the motion control instruction (i.e., *Done* is TRUE) is used as a step completion condition.

### ● Target Absolute Position (*CompleteType* = USINT#1)

It is determined that the step is completed when the difference between the axis current position and the value of *AbsolutePosition* (Target Absolute Position) is equal to or lower than the value of *StepCompleteData.InPosWidth* (Positioning In-position Width).

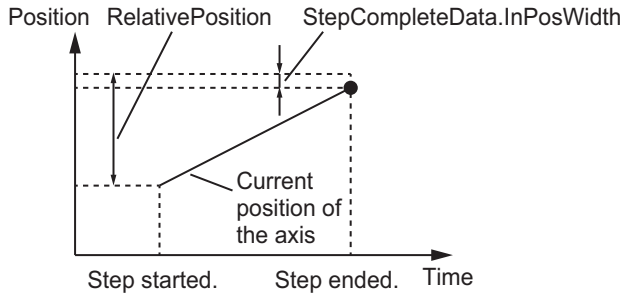
The following conceptual diagram illustrates reaching the target absolute position.



● **Target Relative Position (*CompleteType* = USINT#2)**

It is determined that the step is completed when the difference between the axis current position and the value of (Current position of the axis at the start of the step + *RelativePosition* (Target Relative Position)) is less than or equal to the value of *StepCompleteData.InPosWidth* (Positioning In-position Width).

The following conceptual diagram illustrates reaching the target relative position.



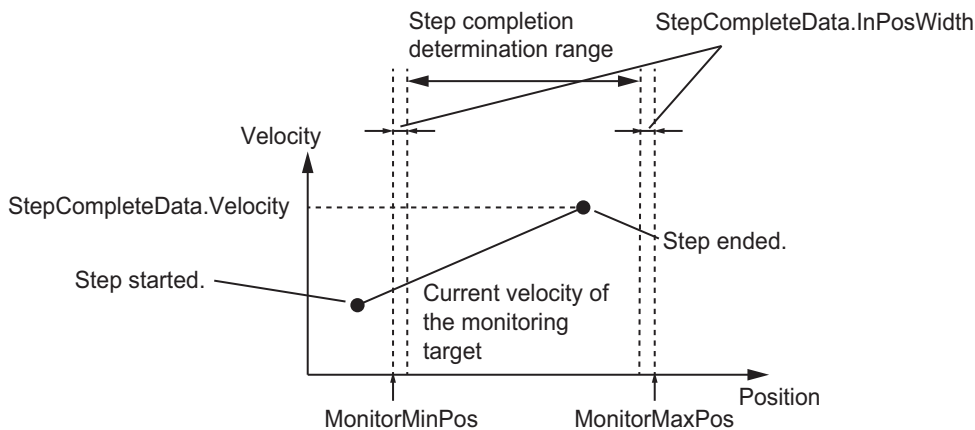
● **Target Velocity (*CompleteType* = USINT#3)**

If the current velocity of the axis at the start of the step is lower than the value of *StepCompleteData.Velocity* (Target Velocity), it is determined that the step is completed when the current velocity of the monitoring target is equal to or higher than the value of *StepCompleteData.Velocity*.

If the current velocity of the monitoring target at the start of the step is higher than the value of *StepCompleteData.Velocity*, it is determined that the step is completed when the current velocity of the monitoring target is equal to or lower than the value of *StepCompleteData.Velocity*.

However, step completion is only determined when the current position of the monitoring target is between (*MonitorMinPos* (Minimum Monitoring Position) + *StepCompleteData.InPosWidth* (Positioning In-position Width)) and (*MonitorMaxPos* (Maximum Monitoring Position) - *StepCompleteData.InPosWidth*). If the step is not completed even when the current position of the monitoring target exceeds *MonitorMinPos* - *StepCompleteData.InPosWidth*, the step is ended immediately, and processing moves to the step specified with *UnachievedNextStepNo* (Step Number When Step Not Completed). The single-axis program operation ends if *UnachievedNextStepNo* is USINT#0.

The following conceptual diagram illustrates reaching the target velocity.



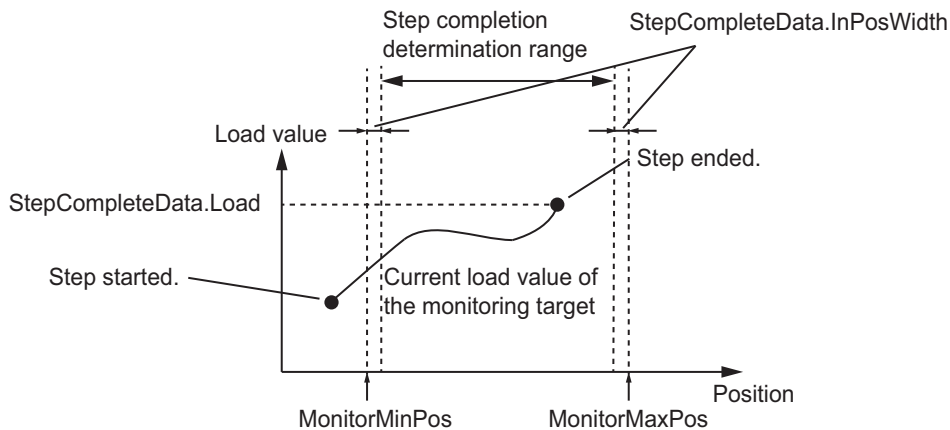
It is determined that the step is completed when *MonitorMinPos* = *MonitorMaxPos*, regardless of the current position of the monitoring target.

### ● Target Load or Higher (*CompleteType* = USINT#4)

If the current load of the monitoring target is equal to or higher than the value of *StepCompleteData.Load* (Target Load), it is determined that the step is completed.

However, step completion is only determined when the current position of the monitoring target is between (*MonitorMinPos* (Minimum Monitoring Position) + *StepCompleteData.InPosWidth* (Positioning In-position Width)) and (*MonitorMaxPos* (Maximum Monitoring Position) - *StepCompleteData.InPosWidth*). If the step is not completed even when the current position of the monitoring target exceeds *MonitorMinPos* - *StepCompleteData.InPosWidth*, the step is ended immediately, and processing moves to the step specified with *UnachievedNextStepNo* (Step Number When Step Not Completed). The single-axis program operation ends if *UnachievedNextStepNo* is USINT#0.

The following conceptual diagram illustrates being equal to or higher than the target load.



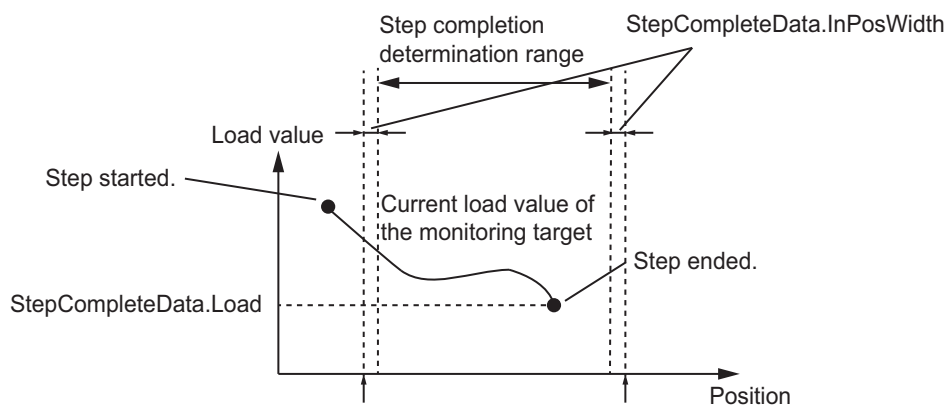
It is determined that the step is completed when  $MonitorMinPos = MonitorMaxPos$ , regardless of the current position of the monitoring target.

### ● Target Load or Lower (*CompleteType* = USINT#5)

If the current load of the monitoring target is equal to or lower than the value of *StepCompleteData.Load* (Target Load), it is determined that the step is completed.

However, step completion is only determined when the current position of the monitoring target is between (*MonitorMinPos* (Minimum Monitoring Position) + *StepCompleteData.InPosWidth* (Positioning In-position Width)) and (*MonitorMaxPos* (Maximum Monitoring Position) - *StepCompleteData.InPosWidth*). If the step is not completed even when the current position of the monitoring target exceeds *MonitorMinPos* - *StepCompleteData.InPosWidth*, the step is ended immediately, and processing moves to the step specified with *UnachievedNextStepNo* (Step Number When Step Not Completed). The single-axis program operation ends if *UnachievedNextStepNo* is USINT#0.

The following conceptual diagram illustrates being equal to or lower than the target load.



It is determined that the step is completed when  $MonitorMinPos = MonitorMaxPos$ , regardless of the current position of the monitoring target.

### ● Step Complete Code Match (*CompleteType* = USINT#6)

It is determined that the step is completed when the value of the *StepCompleteCode* (Step Complete Code) input variable in this function block matches the value of *StepCompleteData.StepCompleteCode* (Step Complete Code Set Value).

Use this when you want to end the step under any conditions.



## ● Continuous Detection of Position Load Gradient (*CompleteType* = USINT#7)

The position load gradient is defined with the following formula.

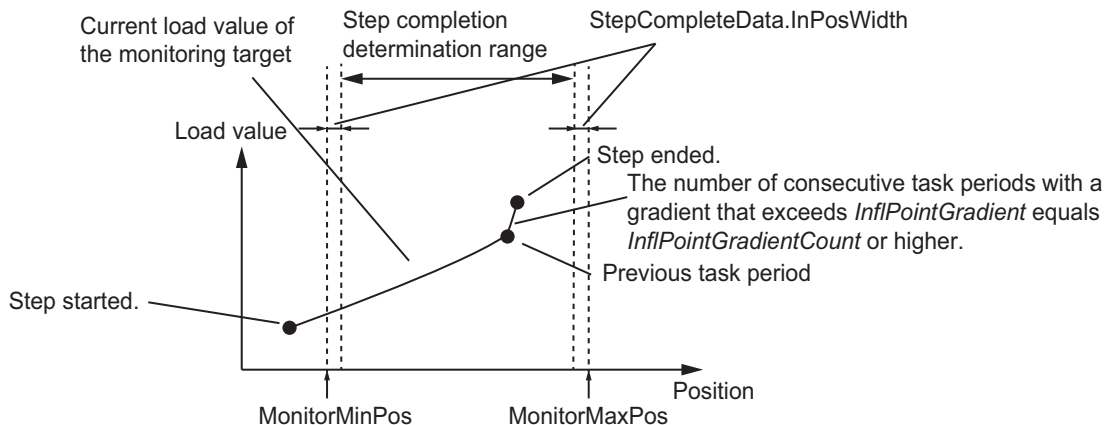
$$\text{Position load gradient} = \frac{\text{Load for the current task period} - \text{Load for the previous task period}}{\text{Current position in the current task period} - \text{Current position in the previous task period}}$$

It is determined that the step is completed when the relation between the current load gradient and the value of *InflPointGradient* (Position Load Gradient) satisfies the following formula in consecutive task periods for more than *InflPointGradientCount* (Number of Consecutive Position Load Gradients).

- *InflPointGradient* > 0  
Position load gradient > *InflPointGradient*
  
- *InflPointGradient* < 0  
Position load gradient < *InflPointGradient*

However, step completion is only determined when the current position of the monitoring target is between (*MonitorMinPos* (Minimum Monitoring Position) + *StepCompleteData.InPosWidth* (Positioning In-position Width)) and (*MonitorMaxPos* (Maximum Monitoring Position) - *StepCompleteData.InPosWidth*). If the step is not completed even when the current position of the monitoring target exceeds *MonitorMinPos* - *StepCompleteData.InPosWidth*, the step is ended immediately, and processing moves to the step specified with *UnachievedNextStepNo* (Step Number When Step Not Completed). The single-axis program operation ends if *UnachievedNextStepNo* is USINT#0.

The following conceptual diagram illustrates a continuous detection of position load gradient.



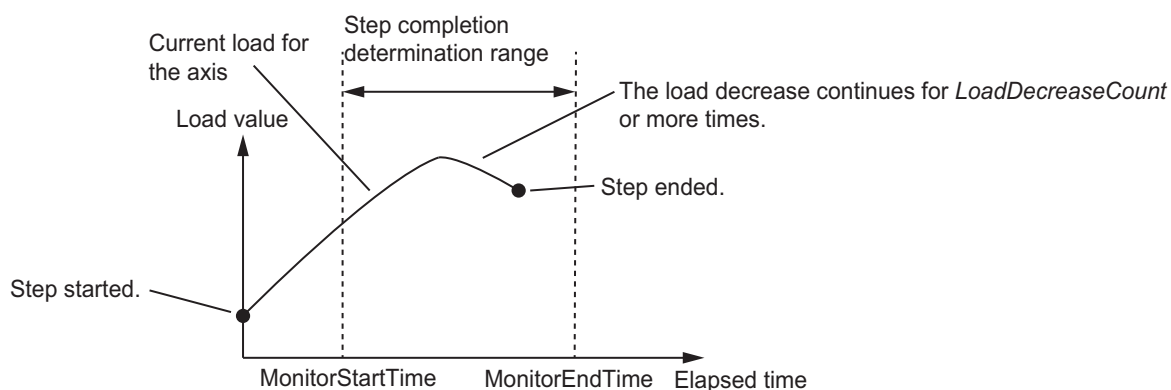
It is determined that the step is completed when *MonitorMinPos* = *MonitorMaxPos*, regardless of the current position of the monitoring target.

● **Continuous Load Decrease (*CompleteType* = USINT#8)**

It is determined that the step is completed when the load decreases to less than the load of the previous task period consecutively for the number of times or more than *LoadDecreaseCount* (Number of Consecutive Load Decreases Threshold).

However, it is determined that the step is completed only when the elapsed time from the start of the step is between *MonitorStartTime* (Monitoring Start Time) and *MonitorEndTime* (Monitoring End Time). If the step is not completed even when the elapsed time exceeds *MonitorEndTime*, the step is ended immediately, and processing moves to the step specified with *UnachievedNextStepNo* (Step Number When Step Not Completed). The single-axis program operation ends if *UnachievedNextStepNo* is USINT#0.

The following conceptual diagram illustrates a continuous load decrease.



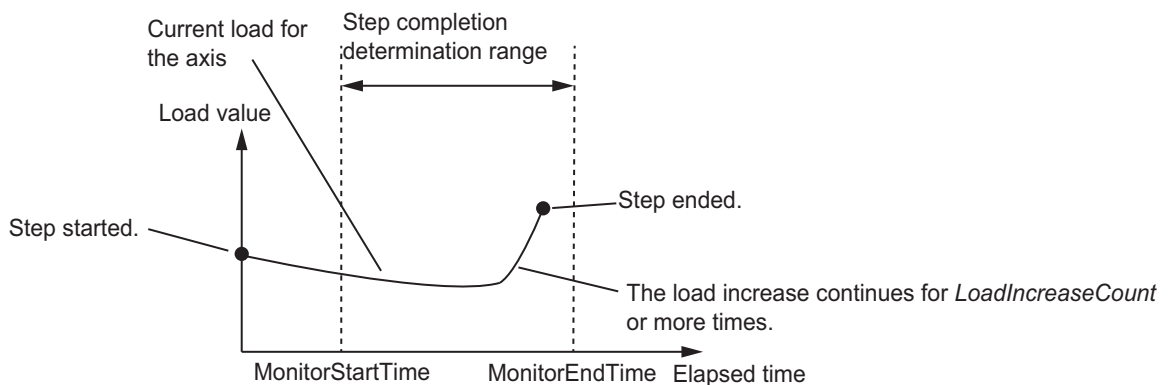
It is determined that the step is completed when *MonitorStartTime* = *MonitorEndTime*, regardless of the elapsed time from the start of the step.

### ● Continuous Load Increase (*CompleteType* = USINT#9)

It is determined that the step is completed when the load increases to higher than the load of the previous task period consecutively for the number of times or more than *LoadIncreaseCount* (Number of Consecutive Load Increases Threshold).

However, it is determined that the step is completed only when the elapsed time from the start of the step is between *MonitorStartTime* (Monitoring Start Time) and *MonitorEndTime* (Monitoring End Time). If the step is not completed even when the elapsed time exceeds *MonitorEndTime*, the step is ended immediately, and processing moves to the step specified in *UnachievedNextStepNo* (Step Number When Step Not Completed). The single-axis program operation ends if *UnachievedNextStepNo* is USINT#0.

The following conceptual diagram illustrates a continuous load increase.



It is determined that the step is completed when *MonitorStartTime* = *MonitorEndTime*, regardless of the elapsed time from the start of the step.

### ● Continuous Position Control Target (*CompleteType* = USINT#10).

When *CtrlCode* is set to USINT#1 (absolute positioning) or *CtrlCode* is set to USINT#2 (relative positioning), you can perform continuous position control by setting the value of *StepCmdData.BufferMode* (Buffer Mode Selection) to one of the following: *\_mcBlendingLow* (Blending Low), *\_mcBlendingPrevious* (Blending Previous), *\_mcBlendingNext* (Blending Next), or *\_mcBlendingHigh* (Blending High).

To perform continuous position control, set the step complete condition to continuous position control target.

### ● Wait (*CompleteType* = USINT#11)

After the step starts, it is determined that the step is completed when the time set in *WaitTime* (Wait Time) has elapsed. If the value of *WaitTime* is higher than the value set in *TimeOut* (Step Timeout Time), the step will time out when the time set in *TimeOut* has elapsed from the start of the step.

### ● Timeout

If the step is not completed even if the time set in *TimeOut* (Step Timeout Time) has elapsed from the start of the step, it will be determined that the step timed out. If a timeout occurs, the process moves to the step specified in *TimeOutNextStepNo* (Step Timeout Next Step Number). The single-axis program operation ends when *TimeOutNextStepNo* is USINT#0.



#### Precautions for Safe Use

If *TimeOutNextStepNo* is set to 0 and the step timeout occurs, *Done* changes to TRUE and this function is completed. At this time, the MC\_Stop (Stop) instruction will not be executed for the next SingleAxisCtrl (Single-axis Control) function block. Set *TimeOutNextStepNo* to a value other than 0 and specify the operation if the step timeout occurs.

## Connections with the SingleAxisCtrl (Single-axis Control) Function Block

When this function block is executed in a user program, the SingleAxisCtrl (Single-axis Control) function block is connected as the next function block.

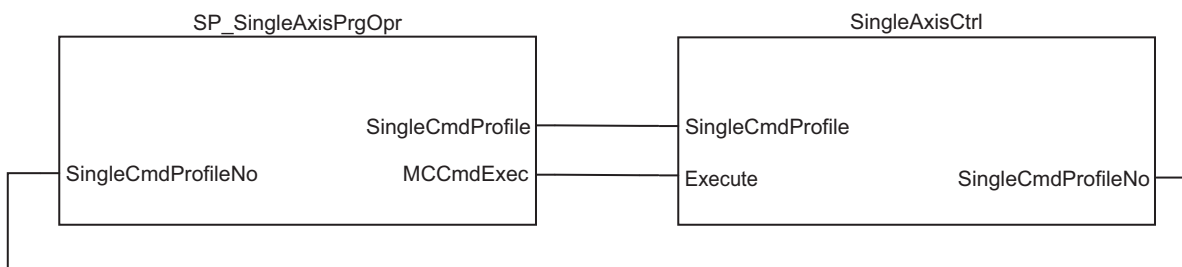
The role of this function block is to output *MCCmdExec* (Motion Function Block Execution Trigger) and *SingleCmdProfile* (Single-axis Command Profile) to the SingleAxisCtrl (Single-axis Control) function block at the appropriate timing for each step.

*MCCmdExec* is the execution trigger for the SingleAxisCtrl (Single-axis Control) function block.

*SingleCmdProfile* is a data set that describes the single-axis motion control for each step. The SingleAxisCtrl (Single-axis Control) function block interprets *SingleCmdProfile*, and executes single-axis motion control for each step.

A simplified connection between this function block and the SingleAxisCtrl (Single-axis Control) function block is given in the following figure. The following table gives the three signal lines to connect.

Signal line in this function block	Signal line in the connected SingleAxisCtrl (Single-Axis Control) function block
MCCmdExec[0]	Execute
SingleCmdProfile	SingleCmdProfile
SingleCmdProfileNo	SingleCmdProfileNo



## ● Valid Ranges of *SingleCmdProfile* (Single-axis Command Profile) Members

The valid ranges for *SingleCmdProfile* (Single Command Profile) depend on the value of *CtrlCode* (Control Method). The following table shows the relationship between the value of *CtrlCode* and the valid ranges of the members.

Also, the values listed in the following table are the same as those for the valid ranges of *StepCmd-Data* (Step Command Data). Three hyphens (---) in the table mean that any setting will be disabled.

Member	Value of <i>CtrlCode</i>				
	0: Deceleration stop	1: Absolute positioning 2: Relative positioning	3: Velocity control	4: Torque control	5: Torque feedback control
Position	---	Positive number, negative number or 0	---	---	---
Velocity	---	Positive number	Positive number, negative number or 0	Positive number or 0	Positive number or 0
Acceleration	---	Positive number or 0	Positive number or 0	---	---
Deceleration	Positive number or 0	Positive number or 0	Positive number or 0	---	---
Jerk	Positive number or 0	Positive number or 0	---	---	---
Direction	---	_mcPositiveDirection _mcShortestWay _mcNegativeDirection _mcCurrentDirection _mcNoDireciton	---	_mcPositiveDirection _mcNegativeDirection	---
BufferMode	_mcAborting	_mcAborting _mcBuffered _mcBlendingLow _mcBlendingPrevious _mcBlendingNext _mcBlendingHigh	---	_mcAborting _mcBuffered	---
Torque	---	---	---	0 to 1000.0	-1,000.0 to 1,000.0
TorqueRamp	---	---	---	Positive number or 0	Positive number or 0
TorqueLimit Positive Enable	TRUE or FALSE	TRUE or FALSE	TRUE or FALSE	TRUE or FALSE	TRUE or FALSE
TorqueLimit Negative Enable	TRUE or FALSE	TRUE or FALSE	TRUE or FALSE	TRUE or FALSE	TRUE or FALSE
TorqueLimit PositiveVal	0.0 or 0.1 to 1000.0	0.0 or 0.1 to 1000.0	0.0 or 0.1 to 1000.0	0.0 or 0.1 to 1000.0	0.0 or 0.1 to 1000.0
TorqueLimit NegativeVal	0.0 or 0.1 to 1000.0	0.0 or 0.1 to 1000.0	0.0 or 0.1 to 1000.0	0.0 or 0.1 to 1000.0	0.0 or 0.1 to 1000.0
TorqueFbk Params	---	---	---	---	*1

\*1. For details, refer to *Torque Feedback Control Function* on page 162.

## Step Load Alarm Determination

Error determination is performed for the axis current position and load for each step in a single-axis program operation. This function is called step load alarm determination. Also, an error that occurs is known as a step load alarm. When a step load alarm occurs, the *StepLoadAlarmOut* (Step Load Alarm) output variable changes to TRUE.

### ● Settings for Step Load Alarm Conditions

Step load alarm determination conditions are set for each step with *StepLoadAlarm* (Step Load Alarm Conditions).

### ● *StepAlarmType* (Step Load Alarm Determination Type)

*StepAlarmType* (Step Load Alarm Determination Type) indicates the type of step load alarm determination for the step. The following table shows the relation between the value of *StepAlarmType* and the step load alarm determination type.

Value of <i>StepAlarmType</i>	Step load alarm determination type
0	Do not perform step load alarm determination.*1
1	Perform determination while step execution is in progress.
2	Perform determination when step is completed.

\*1. The value of *StepLoadAlarmOut* is always FALSE.

### ● Perform Determination While Step Execution Is in Progress (*StepAlarmType* = USINT#1)

If *StepAlarmType* is set to USINT#1, determination is performed while the step is in progress.

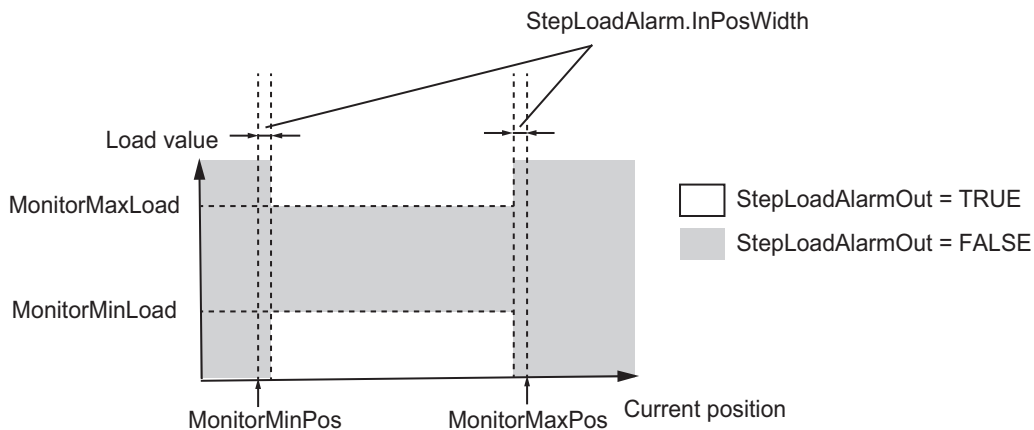
Load error determination is performed for the step in progress when the current position of the axis is between (*MonitorMinPos* (Monitoring Range Minimum Position) + *StepLoadAlarm.InPosWidth* (Upper/Lower Monitoring Limit In-position Width)) and (*MonitorMaxPos* (Monitoring Range Maximum Position) – *StepLoadAlarm.InPosWidth*).

Error determination is not performed when the current position of the axis is not between (*MonitorMinPos* + *StepLoadAlarm.InPosWidth*) and (*MonitorMaxPos* – *StepLoadAlarm.InPosWidth*). If that is the case, the value of *StepLoadAlarmOut* changes to FALSE.

The following table shows the relation between the current position of the axis, the load, and the value of *StepLoadAlarmOut*.

Current position of the axis	Load value	Value of <i>StepLoadAlarmOut</i>
Less than ( <i>MonitorMinPos</i> + <i>StepLoadAlarm.InPosWidth</i> )	---	FALSE
Between ( <i>MonitorMinPos</i> + <i>StepLoadAlarm.InPosWidth</i> ) and ( <i>MonitorMaxPos</i> - <i>StepLoadAlarm.InPosWidth</i> )	Less than <i>MonitorMinLoad</i>	TRUE
	Between <i>MonitorMinLoad</i> and <i>MonitorMaxLoad</i> , inclusive	FALSE
	Greater than <i>MonitorMaxLoad</i> .	TRUE
Greater than ( <i>MonitorMaxPos</i> – <i>StepLoadAlarm.InPosWidth</i> )	---	FALSE

The relations given in the above table are illustrated in the following diagram.



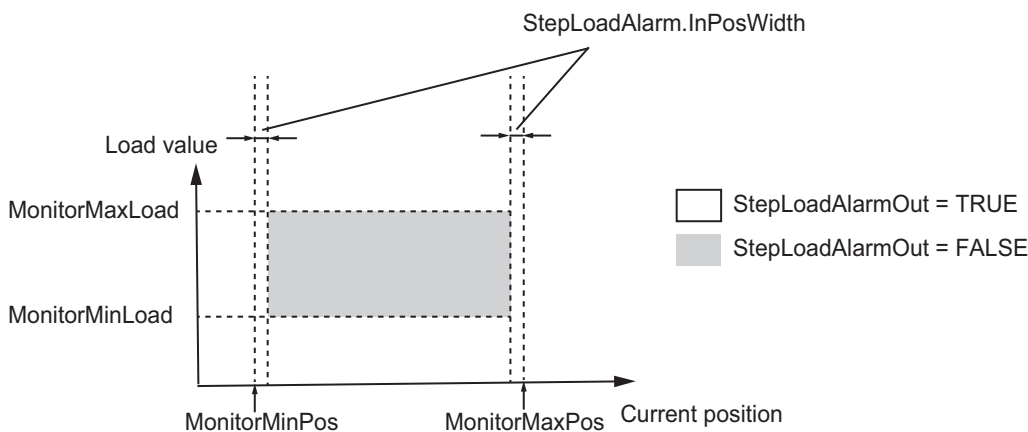
● **Perform Determination When Step Is Completed (*StepAlarmType* = USINT#2)**

If *StepAlarmType* is set to USINT#2, determination is performed when the step is completed.

Error determination is performed for the current position of the axis and the load when the step is completed. The value of *StepLoadAlarmOut* changes to FALSE until the step is completed. The following table shows the relation between the current position of the axis, the load, and the value of *StepLoadAlarmOut* when the step is completed.

Current position of the axis	Load value	Value of <i>StepLoadAlarmOut</i>
Less than ( <i>MonitorMinPos</i> + <i>StepLoadAlarm.InPosWidth</i> )	---	TRUE
Between ( <i>MonitorMinPos</i> + <i>StepLoadAlarm.InPosWidth</i> ) and ( <i>MonitorMaxPos</i> - <i>StepLoadAlarm.InPosWidth</i> )	Less than <i>MonitorMinLoad</i>	TRUE
	Between <i>MonitorMinLoad</i> and <i>MonitorMaxLoad</i> , inclusive	FALSE
	Greater than <i>MonitorMaxLoad</i> .	TRUE
Greater than ( <i>MonitorMaxPos</i> - <i>StepLoadAlarm.InPosWidth</i> )	---	TRUE

The relations given in the above table are illustrated in the following diagram.



**Program Load Alarm Determination**

The function block determines load and current position errors throughout an entire single-axis program operation. This function is called program load alarm determination. Also, an error that occurs is known as a program load alarm. When a program load alarm occurs, the *PrgLoadAlarmOut* (Program Load Alarm) output variable changes to TRUE.

● **Setting Program Load Alarm Conditions**

Set the conditions for program load alarm determination with the *PrgLoadAlarm* (Program Load Alarm Conditions) in-out variable.



## ● *PrgAlarmType* (Program Load Alarm Determination Type)

*PrgAlarmType* (Program Load Alarm Determination Type) indicates the type of program load alarm determination. The following table shows the relation between the value of *PrgAlarmType* and the program load alarm determination type.

Value of <i>PrgAlarmType</i>	Program load alarm determination type
0	Do not perform program load alarm determination.*1
1	Trapezoid area determination
2	Rectangle area determination

\*1. The value of *PrgLoadAlarmOut* is always FALSE.

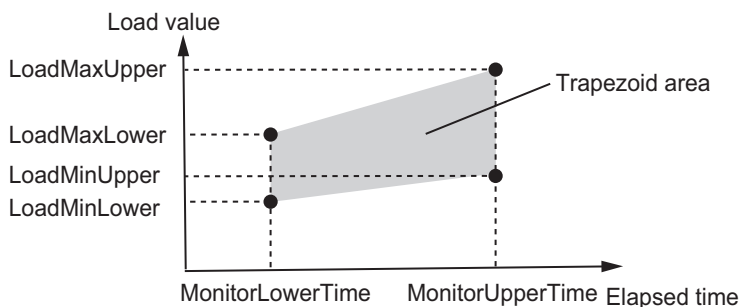
## ● Trapezoid Area Determination

Trapezoid area determination is performed when *PrgAlarmType* is set to USINT#1.

The trapezoid area is evaluated for errors that result from the relation between the trapezoid area and the load. Trapezoid area determination is performed when the time that has elapsed from the start of execution of this function block is between *MonitorUpperTime* (Monitoring Upper Limit Elapsed Time) and *MonitorLowerTime* (Monitoring Lower Limit Elapsed Time), inclusive.

The trapezoid area must be enclosed within the following four points: *LoadMaxUpper* (Monitoring End Point Maximum Load) and *LoadMinUpper* (Monitoring End Point Minimum Load) at the maximum elapsed time, and *LoadMaxLower* (Monitoring Start Point Maximum Load) and *LoadMinLower* (Monitoring Start Point Minimum Load) at the minimum elapsed time.

The following conceptual diagram illustrates the trapezoid area.



You can set the current position of the axis as the reference type to evaluate outside the time elapsed from the execution of this function block. To do so, read the above figure as if *MonitorUpperTime* were replaced with  $(MonitorUpperPos + PrgLoadAlarm.InPosWidth)$  and *MonitorLowerTime* as if it were replaced with  $(MonitorLowerPos - PrgLoadAlarm.InPosWidth)$ .

Set the reference type to evaluate with *UnitType* (Standard Type), as given in the following table.

Value of <i>UnitType</i>	Standard type for determination
0	Elapsed time from start of execution of the function block
1	Current position of the axis

You can select whether to perform error determination when the load is within the trapezoid area or when the load is outside the trapezoid area. The following table shows how to set *AlarmType* (Alarm Type) to specify whether error determination is performed inside or outside the area.

Value of <i>AlarmType</i>	Area for error determination
0	Do not perform program load alarm determination.*1
1	Outside area
2	Inside area

\*1. The value of *PrgLoadAlarmOut* is always FALSE.

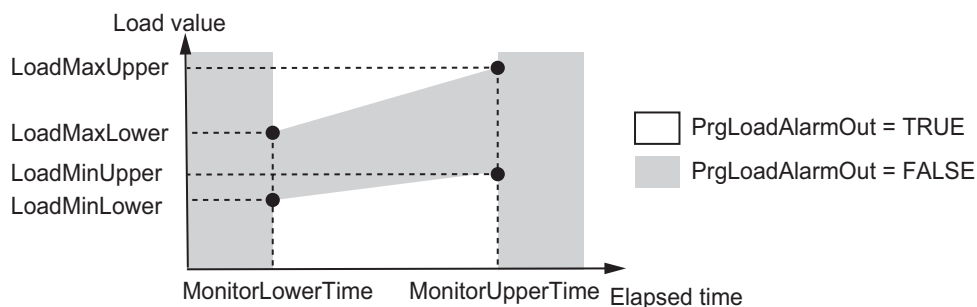
Load error determination is not performed when the elapsed time is not between *MonitorUpperTime* and *MonitorLowerTime*, inclusive. If that is the case, the value of *PrgLoadAlarmOut* changes to FALSE.

The following table shows the relation between the elapsed time, the value of *AlarmType*, the load, and the value of *PrgLoadAlarmOut*.

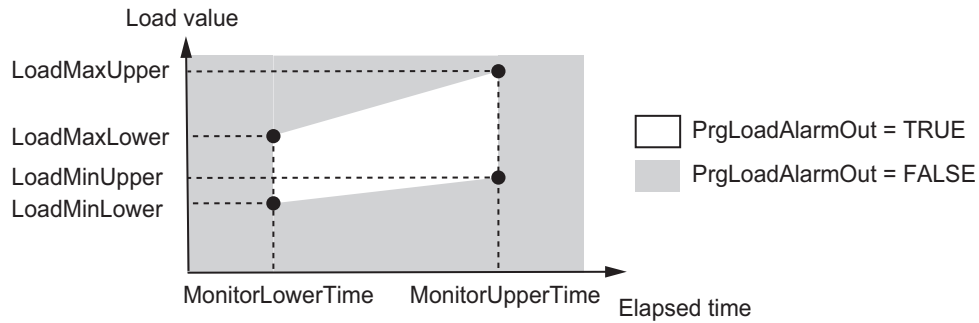
Elapsed time	Value of <i>AlarmType</i>	Load value	Value of <i>PrgLoadAlarmOut</i>
Less than <i>MonitorLowerTime</i>	---	---	FALSE
Between <i>MonitorLowerTime</i> and <i>MonitorUpperTime</i> , inclusive	1	Outside area	TRUE
		Inside area	FALSE
	2	Outside area	FALSE
		Inside area	TRUE
Greater than <i>MonitorUpperTime</i>	---	---	FALSE

The relations given in the above table are illustrated in the following diagram.

- *AlarmType* = USINT#1



- *AlarmType* = USINT#2



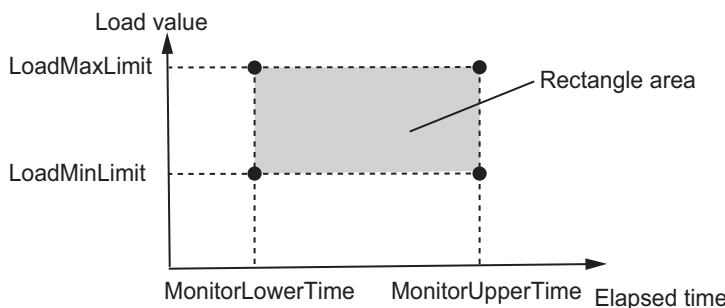
*TrapezoidData* (Trapezoid Area) is a size 5 array. Therefore, you can set a maximum of five trapezoid areas.

### ● Rectangle Area Determination

Rectangle area determination is performed when *PrgAlarmType* is USINT#2.

Rectangle area determination is evaluated for errors that result from the relation between the rectangle area and the load. Rectangle area determination is performed when the time that has elapsed from the start of execution of this function block is between *MonitorUpperTime* (Monitoring Upper Limit Elapsed Time) and *MonitorLowerTime* (Monitoring Lower Limit Elapsed Time).

The rectangle area must be enclosed within the following four straight lines: *MonitorUpperTime* (Monitoring Upper Limit Elapsed Time) and *MonitorLowerTime* (Monitoring Lower Limit Elapsed Time) of the elapsed time, and *LoadMaxLimit* (Load Upper Limit) and *LoadMinLimit* (Load Lower Limit) of the elapsed time. The following conceptual diagram illustrates the rectangle area.



You can set the current position of the axis as the reference type to evaluate outside the time elapsed from the execution of this function block. To do so, read the above figure as if *MonitorUpperTime* were replaced with  $(MonitorUpperPos + PrgLoadAlarm.InPosWidth)$  and *MonitorLowerTime* as if it were replaced with  $(MonitorLowerPos - PrgLoadAlarm.InPosWidth)$ .

Set the reference type to evaluate with *UnitType* (Standard Type), as given in the following table.

Value of <i>UnitType</i>	Standard type for determination
0	Elapsed time from start of execution of the function block
1	Current position of the axis

You can select whether to perform error determination when the load is within the rectangle area or when the load is outside the rectangle area. The following table shows how to set *AlarmType* (Alarm Type) to specify whether error determination is performed inside or outside the area.

Value of <i>AlarmType</i>	Area for error determination
0	Do not perform program load alarm determination.*1
1	Outside area
2	Inside area

\*1. The value of *PrgLoadAlarmOut* is always FALSE.

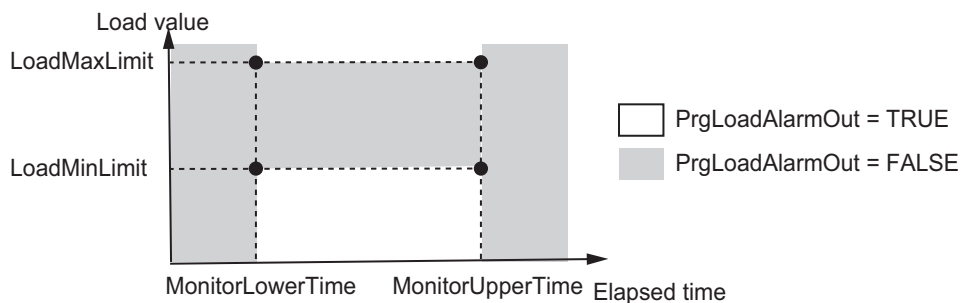
Load error determination is not performed when the elapsed time is not between *MonitorUpperTime* and *MonitorLowerTime*, inclusive. If that is the case, the value of *PrgLoadAlarmOut* changes to FALSE.

The following table shows the relation between the elapsed time, the value of *AlarmType*, the load, and the value of *PrgLoadAlarmOut*.

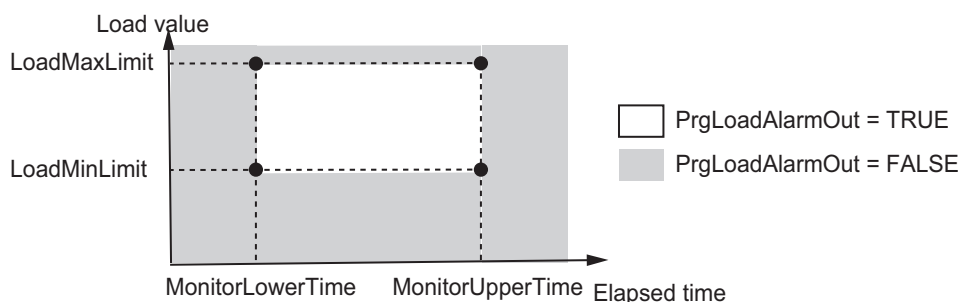
Elapsed time	Value of <i>AlarmType</i>	Load value	Value of <i>PrgLoadAlarmOut</i>
Less than <i>MonitorLowerTime</i>	---	---	FALSE
Between <i>MonitorLowerTime</i> and <i>MonitorUpperTime</i> , inclusive	1	Outside area	TRUE
		Inside area	FALSE
	2	Outside area	FALSE
		Inside area	TRUE
Greater than <i>MonitorUpperTime</i>	---	---	FALSE

The relations given in the above table are illustrated in the following diagram.

- *AlarmType* = USINT#1



- *AlarmType* = USINT#2



*RectangleData* (Rectangle Data) is a size 10 array. Therefore, you can set a maximum of 10 rectangle areas.

## Meanings of Variables

The meanings of the other variables are described below.

- **StartStepNo (Execution Start Step Number)**

This variable is used to specify the step number to start execution of single-axis program operation. For example, if *StartStepNo* is set to USINT#4, the execution of the operation starts from *PrgTable*[4].

However, if *StartStepNo* is set to USINT#0, the function block is ended immediately and single-axis program operation is not executed.

- **SingleMode (Single Mode)**

The execution of only a certain step without execution of a step sequence set with *PrgTable* (Program Tables) is called Single Mode. If you change the value of *SingleMode* (Single Mode) to TRUE and execute this function block, only the step specified with *StartStepNo* (Execution Start Step Number) is executed and execution of this function block is ended.

- **Position (Current Position), Velocity (Current Velocity), and Load (Current Load)**

These variables are used to input the current position monitor, the current velocity monitor, and the current load monitor for the controlled system. These values are used in step completion determination, step load alarm determination, and program load alarm determination.

- **MCCmdDone (Motion Instruction Completion)**

This variable indicates when execution of the current motion control instruction is completed. It is used to set waiting for motion instruction completion as a condition for completion of a step.

It is determined that the step is completed when the value of *MCCmdDone* (Motion Instruction Completion) is TRUE.

- **StepCompleteCode (Step Complete Code)**

This variable is used to set step complete code match as a condition for completion of a step.

It is determined that the step is completed when the value of the *StepCompleteCode* (Step Complete Code) matches the value of *StepCompleteData.StepCompleteCode* (Step Complete Code Set Value). For details on step completion conditions, refer to *Step Completion Condition* on page 59.

- **SingleCmdProfileNo (Single-axis Command Profile Number)**

When you consecutively execute multiple single-axis motion control instructions, this number indicates the ordinal number of the current single-axis motion control instruction. Connect this variable to the *SingleCmdProfileNo* output variable in the *SingleAxisCtrl* (Single-axis Control) function block.

- **MCCmdExec (Motion Function Block Execution Trigger)**

An execution trigger for the *SingleAxisCtrl* (Single-axis Control) function block that connects to the *SP\_SingleAxisPrgOpr* function block is output to *MCCmdExec*[0]. Connect *MCCmdExec*[0] to the *Execute* input variable in the *SingleAxisCtrl* (Single-axis Control) function block.

- **SingleCmdProfile (Single-axis Command Profile)**

*SingleCmdProfile* is a data set that describes the single-axis motion control for each step and it is input to the SingleAxisCtrl (Single-axis Control) function block.

Connect this variable to the *SingleCmdProfile* in-out variable in the SingleAxisCtrl (Single-axis Control) function block.

- **CurrentStepNo (Current Step Number)**

This variable gives the number of the current step. When single-axis program operation is stopped, the final output value is retained.

- **ExtrOutputCode (External Output Code)**

This is the external output code that is set with *PrgTable.ExtrOutputCode* (External Output Code). The code is output in the next task period for each step that is completed. Use it when you want to detect the completion of a specific step from outside the function block.

- **StepExec (Step Start Trigger)**

This variable is used to notify the next function block that a step has started. Every time a step is started, its value changes to TRUE for one task period.

- **StepCompleted (Step Completed Trigger)**

This variable is used to notify the next function block that a step has ended. Every time a step is ended, its value changes to TRUE for one task period.

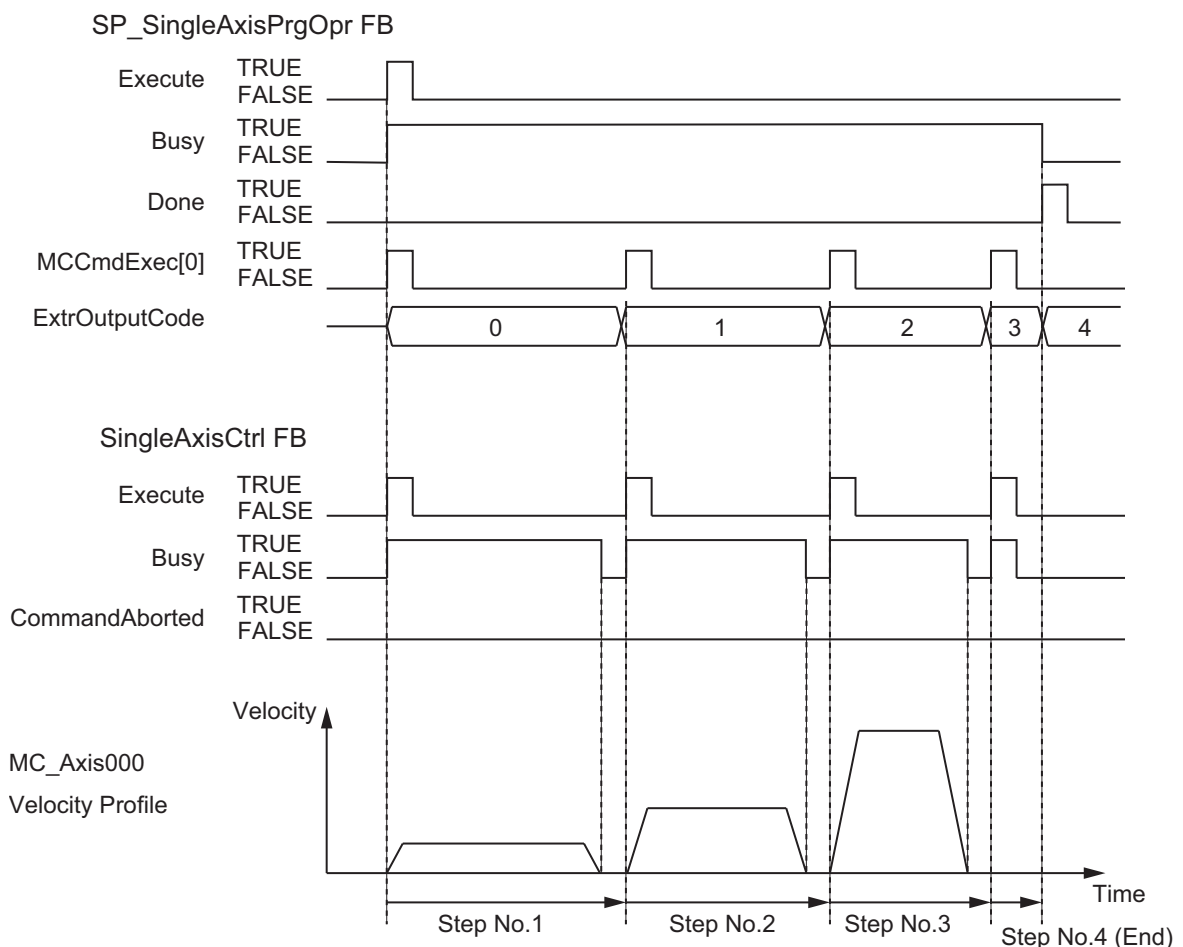
## Timing Charts

The timing charts for the SP\_SingleAxisPrgOpr function block are given together with the timing charts for the SingleAxisCtrl (Single-axis Control) function block that is connected after it.

The following timing chart shows the operation patterns given in the sample programming of this function block. The behavior of output variables in the SingleAxisCtrl (Single-axis Control) function block will change according to the settings of *PrgTable* (Program Tables).

### ● Normal End

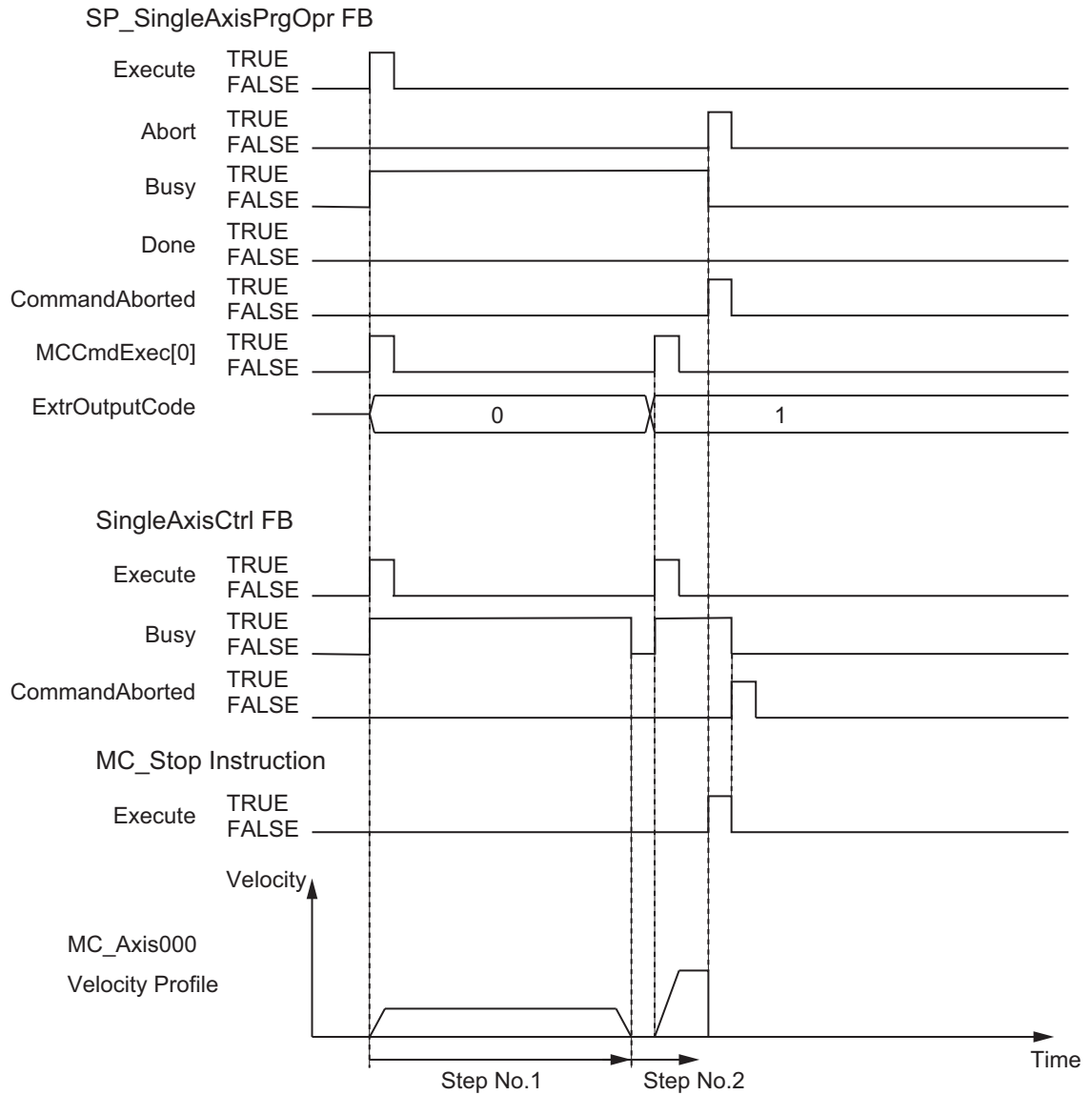
- *Busy* (Executing) changes to TRUE when *Execute* in the function block changes to TRUE. If *MCCmdExec* (Motion Function Block Execution Trigger) changes to TRUE in the same task period, *Busy* (Executing) in the SingleAxisCtrl (Single-axis Control) function block also changes to TRUE.
- When the step sequence is completed, the execution of the function block is ended. When *Busy* changes to FALSE, *Done* changes to TRUE. The example in the following figure uses four steps.
- Step completion determination is performed based on *StepCompleteData* (Step Completion Condition), so the completion of the execution of the SingleAxisCtrl (Single-axis Control) function block may lag behind the completion of the execution of the SP\_SingleAxisPrgOpr function block.
- *ExtrOutputCode* (External Output Code) is output when execution of the next step after the completed step starts.
- The value of *ExtrOutputCode* (External Output Code) in the final step is retained even after *Done* changes to TRUE.



● **Aborting Execution**

The following timing chart gives an example when executing the MC\_Stop (Stop) instruction at the same time as *Abort* in this function block is executed.

- When the value of *Abort* is changed to TRUE, the processing is aborted.
- The value of *ExtrOutputCode* (External Output Code) that is being output when processing is aborted is retained even after processing is aborted.

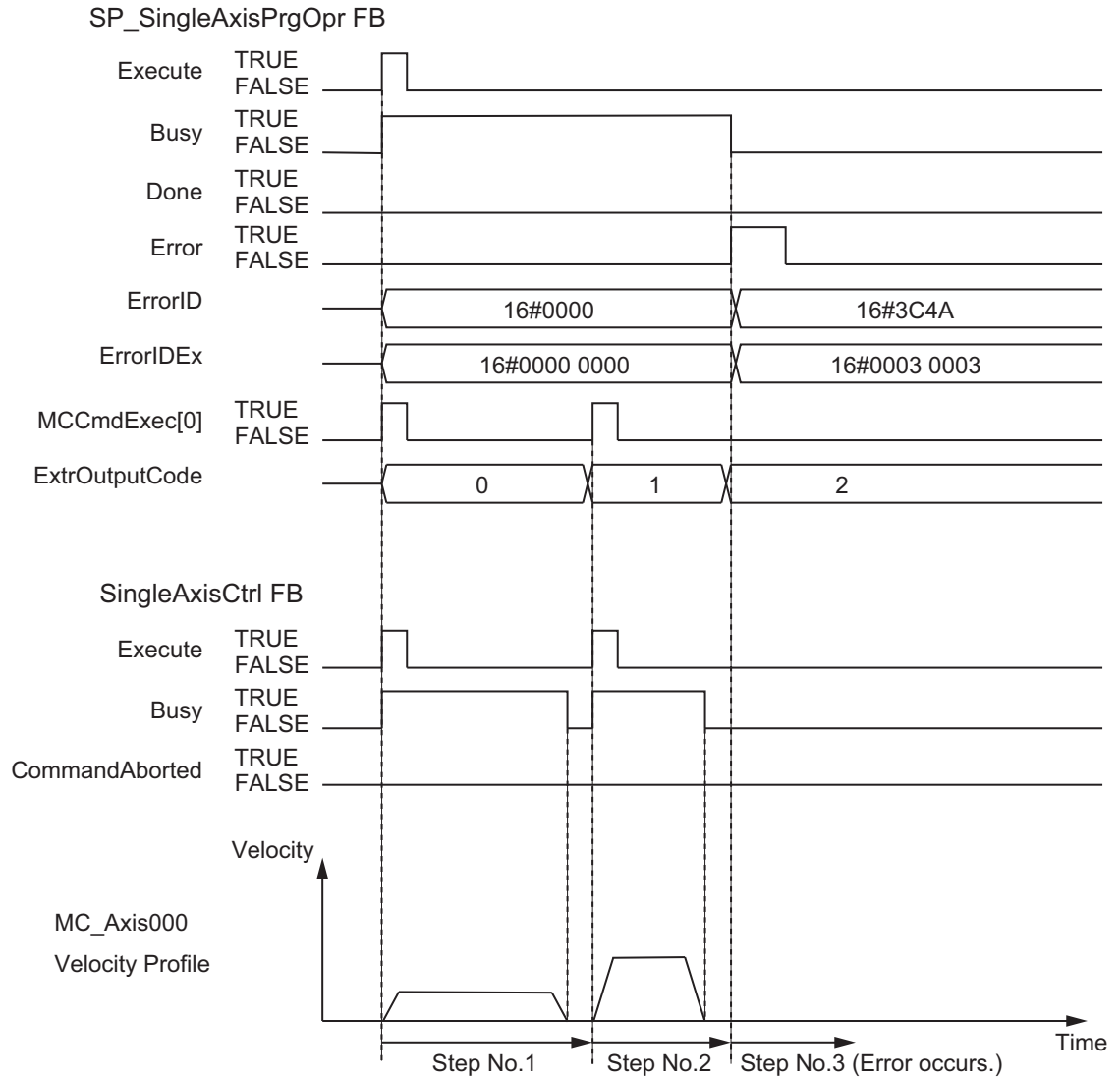




● **Error End**

The following timing chart gives an example when a control method outside the range is specified to *StepCmdData* (Step Command Data) in the third step of *PrgTable* (Program Tables).

- If an error occurs during execution of this function block, *Error* changes to TRUE for two task periods. You can find out the cause of the error by referring to the values output by *ErrorID* (Error Code) and *ErrorIDEx* (Expansion Error Code).
- The value of *ExtrOutputCode* (External Output Code) that is being output when an error occurs is retained even after the function block is ended for the error.



## Precautions for Correct Use

- Connect the *SingleCmdProfileNo* (Single-axis Command Profile Number) output variable in the *SingleAxisCtrl* (Single-axis Control) function block to the *SingleCmdProfileNo* (Single-axis Command Profile Number) input variable in the *SP\_SingleAxisPrgOpr* function block. If you do not connect these variables, step sequence progress management may not be normal when you perform continuous position control (i.e., blending).
- Execute the *SP\_SingleAxisPrgOpr* function block and the *SingleAxisCtrl* (Single-axis Control) function block in the same task when you connect these two function blocks. If these two function blocks are executed in the different task, the motion control may not be correctly executed.
- Even if execution ends before the *PrgTable[50]* step, the *MC\_Stop* (Stop) instruction will not be executed for the next *SingleAxisCtrl* (Single-axis Control) function block.
- If *CtrlCode* is set to *USINT#6* (operation hold) and the motion control instruction for the previous step is not completed, the operation continues at the set value of the motion control instruction for the previous step.
- If *CompleteType* is set to *USINT#1* (target absolute position), measurement positions from an external sensor or other measurement device are input to *Position* (Current Position). If the measurement position cannot follow the target value, it may be determined that the step is completed immediately after the step starts. Make adjustments so that the measurement position can follow the target value.
- If *CompleteType* is set to *USINT#3* (target velocity), input the command current velocity to *Velocity* (Current Velocity).
- If *CompleteType* is set to *USINT#1* (target absolute position) or *USINT#2* (target relative position), measurement positions from an external sensor or other measurement device are input to *Position* (Current Position). If noise or other disturbances are present, it may be determined that the step is completed immediately after the step starts. In cases like this, change the step completion conditions.
- To perform continuous position control (i.e., blending), set *CompleteType* to *USINT#10* (continuous position control target). If the step completion condition is set to any other setting, step sequence progress management may not be normal.
- When *CtrlCode* is set to *USINT#1* (absolute positioning) or *CtrlCode* is set to *USINT#2* (relative positioning), you can perform continuous position control by setting the value of *StepCmdData.BufferMode* (Buffer Mode Selection) to one of the following: *\_mcBlendingLow* (Blending Low), *\_mcBlendingPrevious* (Blending Previous), *\_mcBlendingNext* (Blending Next), or *\_mcBlendingHigh* (Blending High). However, the following restrictions apply:
  - a) All values must be the same in Buffer Mode.
  - b) There must be no more than 10 continuous position control instructions.



### Precautions for Safe Use

- If you use *TorqueLimitPositiveVal* (Positive Torque Limit) or *TorqueLimitNegativeVal* (Negative Torque Limit) in this function block, make sure to connect the *TorqueLimitParam* (Torque Limit Settings) output variable in the *SingleAxisCtrl* (Single-axis Control) function block that is combined with this function block to the *MC\_SetTorqueLimit* (Set Torque Limit) instruction.
- If execution of the *SP\_SingleAxisPrgOpr* function block is aborted, function blocks used in combination with it, such as the *SingleAxisCtrl* (Single-axis Control) function block, will not stop. Create a user program that stops axis processing when the *SP\_SingleAxisPrgOpr* function block is aborted.
- If the *SP\_SingleAxisPrgOpr* function block ends in error, function blocks used in combination with it, such as a *SingleAxisCtrl* (Single-axis Control) function block, will not stop. Create a user program that stops axis processing when the *SP\_SingleAxisPrgOpr* function block ends in an error.

## Troubleshooting

The error codes, expansion error codes, status, descriptions, and corrections are given in the following table.

Error code	Expansion error code	Status	Description	Correction
16#0000	16#00000000	Normal end		
16#3C4A	16#00000001	Execution start step number range exceeded	The value of <i>StartStepNo</i> (Execution Start Step Number) exceeded the valid range.	Check the valid range of the value of <i>StartStepNo</i> (Execution Start Step Number) and set the value within the valid range.
	16#00000002	Next step number range exceeded	The value of <i>NextStepNo</i> (Next Step Number) exceeded the valid range.	Check the valid range of the value of <i>NextStepNo</i> (Next Step Number) and set the value within the valid range.
	16#□□□□0003 <sup>*1</sup>	Control method range exceeded	The value of <i>CtrlCode</i> (Control Method) is outside the valid range.	Check the valid range of the value of <i>CtrlCode</i> (Control Method) and set the value within the valid range.
	16#00000004	Incorrect task setting	An attempt was made to execute this function block in a task period which is not a primary task period or a periodic task.	Execute this function block in a primary periodic task or a periodic task.
16#3C4B	16#□□□□0001 <sup>*1</sup>	Step timeout time out of range	The value of <i>TimeOut</i> (Step Timeout Time) is outside the valid range.	Check the valid range of the value for <i>TimeOut</i> (Step Timeout Time) and set the value within the valid range.
	16#□□□□0002 <sup>*1</sup>	Step timeout next step number out of range	The value of <i>TimeOutNextStepNo</i> (Step Timeout Next Step Number) is outside the valid range.	Check the valid range of the value for <i>TimeOutNextStepNo</i> (Step Timeout Next Step Number) and set the value within the valid range.
	16#□□□□0003 <sup>*1</sup>	Illegal monitoring position	The value of <i>MonitorMinPos</i> (Minimum Monitoring Position) is greater than the value of <i>MonitorMaxPos</i> (Maximum Monitoring Position).	Set <i>MonitorMinPos</i> (Minimum Monitoring Position) to a value that is less than or equal to the value of <i>MonitorMaxPos</i> (Maximum Monitoring Position).
	16#□□□□0004 <sup>*1</sup>	Monitoring start time out of range	The value of <i>MonitorStartTime</i> (Monitoring Start Time) is outside the valid range.	Check the valid range of the value for <i>MonitorStartTime</i> (Monitoring Start Time) and set the value within the valid range.
	16#□□□□0005 <sup>*1</sup>	Monitoring end time out of range	The value of <i>MonitorEndTime</i> (Monitoring End Time) is outside the valid range.	Check the valid range of the value for <i>MonitorEndTime</i> (Monitoring End Time) and set the value within the valid range.
	16#□□□□0006 <sup>*1</sup>	Illegal monitoring time	The value of <i>MonitorStartTime</i> (Monitoring Start Time) is greater than the value of <i>MonitorEndTime</i> (Monitoring End Time).	Set <i>MonitorStartTime</i> (Monitoring Start Time) to a value that is less than or equal to the value of <i>MonitorEndTime</i> (Monitoring End Time).

Error code	Expansion error code	Status	Description	Correction
16#3C4B	16#□□□□0007* <sup>1</sup>	Number of consecutive position load gradients out of range	The value of <i>InflPointGradientCount</i> (Number of Consecutive Position Load Gradients) is outside the valid range.	Check the valid range for the value of <i>InflPointGradientCount</i> (Number of Consecutive Position Load Gradients) and set the value within the valid range.
	16#□□□□0008* <sup>1</sup>	Number of consecutive load decreases threshold out of range	The value of <i>LoadDecreaseCount</i> (Number of Consecutive Load Reductions Threshold) is outside the valid range.	Check the valid range for the value of <i>LoadDecreaseCount</i> (Number of Consecutive Load Reductions Threshold) and set the value within the valid range.
	16#□□□□0009* <sup>1</sup>	Number of consecutive load increases threshold out of range	The value of <i>LoadIncreaseCount</i> (Number of Consecutive Load Increases Threshold) is outside the valid range.	Check the valid range for the value of <i>LoadIncreaseCount</i> (Number of Consecutive Load Increases Threshold) and set the value within the valid range.
	16#□□□□000A* <sup>1</sup>	Step number when step not completed	The value of <i>UnachievedNextStepNo</i> (Step Number When Step Not Completed) is outside the valid range.	Check the valid range of the value for <i>UnachievedNextStepNo</i> (Step Number When Step Not Completed) and set the value within the valid range.
	16#□□□□000B* <sup>1</sup>	Wait time out of range	The value of <i>WaitTime</i> (Wait Time) is outside the valid range.	Check the valid range of the value for <i>WaitTime</i> (Wait Time) and set the value within the valid range.
	16#□□□□000C* <sup>1</sup>	Position load gradient out of range	The value of <i>InflPointGradient</i> (Position Load Gradient) is outside the valid range.	Check the valid range for the value of <i>InflPointGradient</i> (Position Load Gradient) and set the value within the valid range.
	16#□□□□000D* <sup>1</sup>	Step completion type out of range	The value of <i>CompleteType</i> (Step Completion Type) exceeded the valid range.	Check the valid range of the value for <i>CompleteType</i> (Step Completion Type) and set the value within the valid range.
	16#□□□□000E* <sup>1</sup>	Positioning in-position width out of range	The value of <i>InPosWidth</i> (Positioning In-position Width) is outside the valid range.	Check the valid range of the value for <i>InPosWidth</i> (Positioning In-position Width) and set the value within the valid range.
	16#0000000F	Incorrect task setting	An attempt was made to execute this function block in a task period which is not a primary task period or a periodic task.	Execute this function block in a primary periodic task or a periodic task.

Error code	Expansion error code	Status	Description	Correction
16#3C4C	16#□□□□0001* <sup>1</sup>	Illegal monitoring position	The value of <i>MonitorMinPos</i> (Monitoring Range Minimum Position) is greater than the value of <i>MonitorMaxPos</i> (Monitoring Range Maximum Position).	Set <i>MonitorMinPos</i> (Monitoring Range Minimum Position) to a value that is less than or equal to the value of <i>MonitorMaxPos</i> (Monitoring Range Maximum Position).
	16#□□□□0002* <sup>1</sup>	Illegal monitoring load	The value of <i>MonitorMinLoad</i> (Monitoring Load Range Lower Limit) is greater than the value of <i>MonitorMaxLoad</i> (Monitoring Load Range Upper Limit).	Set <i>MonitorMinLoad</i> (Monitoring Load Range Lower Limit) to a value that is less than or equal to the value of <i>MonitorMaxLoad</i> (Monitoring Load Range Upper Limit).
	16#□□□□0003* <sup>1</sup>	Step load Alarm determination type out of range	The value of <i>StepAlarmType</i> (Step Load Alarm Determination Type) is outside the valid range.	Check the valid range of the value for <i>StepAlarmType</i> (Step Load Alarm Determination Type) and set the value within the valid range.
	16#□□□□0004* <sup>1</sup>	Upper/lower monitoring limit in-position width out of range	The value of <i>InPosWidth</i> (Upper/Lower Monitoring Limit In-position Width) is outside the valid range.	Check the valid range of the value for <i>InPosWidth</i> (Upper/Lower Monitoring Limit In-position Width) and set the value within the valid range.
	16#00000005	Incorrect task setting	An attempt was made to execute this function block in a task period which is not a primary task period or a periodic task.	Execute this function block in a primary periodic task or a periodic task.

Error code	Expansion error code	Status	Description	Correction
16#3C4D	16#00000001	Program load alarm determination type out of range	The value of <i>PrgAlarmType</i> (Program Load Alarm Determination Type) is outside the valid range.	Check the valid range of the value for <i>PrgAlarmType</i> (Program Load Alarm Determination Type) and set the value within the valid range.
	16#00000002	Standard type out of range	The value of <i>UnitType</i> (Standard Type) is outside of the valid range.	Check the valid range of the value for <i>UnitType</i> (Standard Type) and set the value within the valid range.
	16#00000003	Trapezoid area data alarm type out of range	The value of <i>TrapezoidData.AlarmType</i> (Alarm Type) is outside of the valid range.	Check the valid range of the value for <i>TrapezoidData.AlarmType</i> (Alarm Type) and set the value within the valid range.
	16#00000004	Trapezoid area data monitoring lower limit elapsed time out of range	The value of <i>TrapezoidData.MonitorLowerTime</i> (Monitoring Lower Limit Elapsed Time) is outside of the valid range.	Check the valid range of the value for <i>TrapezoidData.MonitorLowerTime</i> (Monitoring Lower Limit Elapsed Time) and set the value within the valid range.
	16#00000005	Trapezoid area data monitoring upper limit elapsed time out of range	The value of <i>TrapezoidData.MonitorUpperTime</i> (Monitoring Upper Limit Elapsed Time) is outside of the valid range.	Check the valid range of the value for <i>TrapezoidData.MonitorUpperTime</i> (Monitoring Upper Limit Elapsed Time) and set the value within the valid range.
	16#00000006	Illegal trapezoid area data monitoring lower limit elapsed time	The value of <i>TrapezoidData.MonitorLowerTime</i> (Monitoring Lower Limit Elapsed Time) is greater than the value of <i>TrapezoidData.MonitorUpperTime</i> (Monitoring Upper Limit Elapsed Time).	Set <i>TrapezoidData.MonitorLowerTime</i> (Monitoring Lower Limit Elapsed Time) to a value that is less than or equal to the value of <i>TrapezoidData.MonitorUpperTime</i> (Monitoring Upper Limit Elapsed Time).
	16#00000007	Illegal trapezoid area data monitoring position	The value of <i>TrapezoidData.MonitorLowerPos</i> (Minimum Monitoring Position) is greater than the value of <i>TrapezoidData.MonitorUpperPos</i> (Maximum Monitoring Position).	Set <i>TrapezoidData.MonitorLowerPos</i> (Minimum Monitoring Position) to a value that is less than or equal to the value of <i>TrapezoidData.MonitorUpperPos</i> (Maximum Monitoring Position).

Error code	Expansion error code	Status	Description	Correction
16#3C4D	16#00000008	Illegal trapezoid area data monitoring start point load	The value of <i>TrapezoidData.LoadMinLower</i> (Monitoring Start Point Minimum Load) is greater than the value of <i>TrapezoidData.LoadMaxLower</i> (Monitoring Start Point Maximum Load).	Set the value of <i>TrapezoidData.LoadMinLower</i> (Monitoring Start Point Minimum Load) to a value that is less than or equal to the value of <i>TrapezoidData.LoadMaxLower</i> (Monitoring Start Point Maximum Load).
	16#00000009	Illegal trapezoid area data monitoring end point load	The value of <i>TrapezoidData.LoadMinUpper</i> (Monitoring End Point Minimum Load) is greater than the value of <i>TrapezoidData.LoadMaxUpper</i> (Monitoring End Point Maximum Load).	Set the value of <i>TrapezoidData.LoadMinUpper</i> (Monitoring End Point Minimum Load) to a value that is less than or equal to the value of <i>TrapezoidData.LoadMaxUpper</i> (Monitoring End Point Maximum Load). Set the value of <i>TrapezoidData.LoadMinUpper</i> (Monitoring End Point Minimum Load) to a value that is less than or equal to the value of <i>TrapezoidData.LoadMaxUpper</i> (Monitoring End Point Maximum Load).
	16#0000000A	Rectangle area data alarm type out of range	The value of <i>RectangleData.AlarmType</i> (Alarm Type) is outside of the valid range.	Check the valid range of the value for <i>RectangleData.AlarmType</i> (Alarm Type) and set the value within the valid range.
	16#0000000B	Rectangle area data monitoring lower limit elapsed time out of range	The value of <i>RectangleData.MonitorLowerTime</i> (Monitoring Lower Limit Elapsed Time) is outside of the valid range.	Check the valid range of the value for <i>RectangleData.MonitorLowerTime</i> (Monitoring Lower Limit Elapsed Time) and set the value within the valid range.
	16#0000000C	Rectangle area data monitoring upper limit elapsed time out of range	The value of <i>RectangleData.MonitorUpperTime</i> (Monitoring Upper Limit Elapsed Time) is outside of the valid range.	Check the valid range of the value for <i>RectangleData.MonitorUpperTime</i> (Monitoring Upper Limit Elapsed Time) and set the value within the valid range.
	16#0000000D	Illegal rectangle area data monitoring lower limit elapsed time	The value of <i>RectangleData.MonitorLowerTime</i> (Monitoring Lower Limit Elapsed Time) is greater than the value of <i>RectangleData.MonitorUpperTime</i> (Monitoring Upper Limit Elapsed Time).	Set <i>RectangleData.MonitorLowerTime</i> (Monitoring Lower Limit Elapsed Time) to a value that is less than or equal to the value of <i>RectangleData.MonitorUpperTime</i> (Monitoring Upper Limit Elapsed Time).
	16#0000000E	Illegal rectangle area data monitoring position	The value of <i>RectangleData.MonitorLowerPos</i> (Minimum Monitoring Position) is greater than the value of <i>RectangleData.MonitorUpperPos</i> (Maximum Monitoring Position).	Set <i>RectangleData.MonitorLowerPos</i> (Minimum Monitoring Position) to a value that is less than or equal to the value of <i>RectangleData.MonitorUpperPos</i> (Maximum Monitoring Position).

Error code	Expansion error code	Status	Description	Correction
16#3C4D	16#0000000F	Illegal rectangle area data load	The value of <i>RectangleData.LoadMinLimit</i> (Load Lower Limit) is greater than the value of <i>RectangleData.LoadMaxLimit</i> (Load Upper Limit).	Set the value of <i>RectangleData.LoadMinLimit</i> (Load Lower Limit) to a value that is less than or equal to the value of <i>RectangleData.LoadMaxLimit</i> (Load Upper Limit).
	16#00000010	Illegal trapezoid area data	The positional relationships between the values of <i>TrapezoidData.MonitorLowerTime</i> (Monitoring Lower Limit Elapsed Time), <i>TrapezoidData.MonitorUpperTime</i> (Monitoring Upper Limit Elapsed Time), <i>TrapezoidData.MonitorLowerPos</i> (Minimum Monitoring Position), <i>TrapezoidData.MonitorUpperPos</i> (Maximum Monitoring Position), <i>TrapezoidData.LoadMinLower</i> (Monitoring Start Point Minimum Load), <i>TrapezoidData.LoadMaxLower</i> (Monitoring Start Point Maximum Load), <i>TrapezoidData.LoadMinUpper</i> (Monitoring End Point Minimum Load), and <i>TrapezoidData.LoadMaxUpper</i> (Monitoring End Point Maximum Load) do not form a trapezoid.	Set the values of <i>TrapezoidData.MonitorLowerTime</i> (Monitoring Lower Limit Elapsed Time), <i>TrapezoidData.MonitorUpperTime</i> (Monitoring Upper Limit Elapsed Time), <i>TrapezoidData.MonitorLowerPos</i> (Minimum Monitoring Position), <i>TrapezoidData.MonitorUpperPos</i> (Maximum Monitoring Position), <i>TrapezoidData.LoadMinLower</i> (Monitoring Start Point Minimum Load), <i>TrapezoidData.LoadMaxLower</i> (Monitoring Start Point Maximum Load), <i>TrapezoidData.LoadMinUpper</i> (Monitoring End Point Minimum Load), and <i>TrapezoidData.LoadMaxUpper</i> (Monitoring End Point Maximum Load) so that the enclosed area is a trapezoid.
	16#00000011	Upper/lower monitoring limit in-position width out of range	The value of <i>InPosWidth</i> (Upper/Lower Monitoring Limit In-position Width) is outside the valid range.	Check the valid range of the value for <i>InPosWidth</i> (Upper/Lower Monitoring Limit In-position Width) and set the value within the valid range.
	16#00000012	Incorrect task setting	An attempt was made to execute this function block in a task period which is not a primary task period or a periodic task.	Execute this function block in a primary periodic task or a periodic task.

\*1. The boxes (□□□□) are the BCD values of *CurrentStepNo* (Current Step Number) when the error occurred. For example, if *CurrentStepNo* is USINT#11, *ErrorIDEx* is DWORD#16#00110001.



## Sample Programming

This sample programming shows how to use single-axis program operation for absolute positioning accompanied by torque control. The SP\_SingleAxisPrgOpr (Single-axis Program Operation) and SingleAxisCtrl (Single-axis Control) function blocks are used for single-axis program operation, and the MC\_SetTorqueLimit (Torque Control) instruction is used to perform torque control.

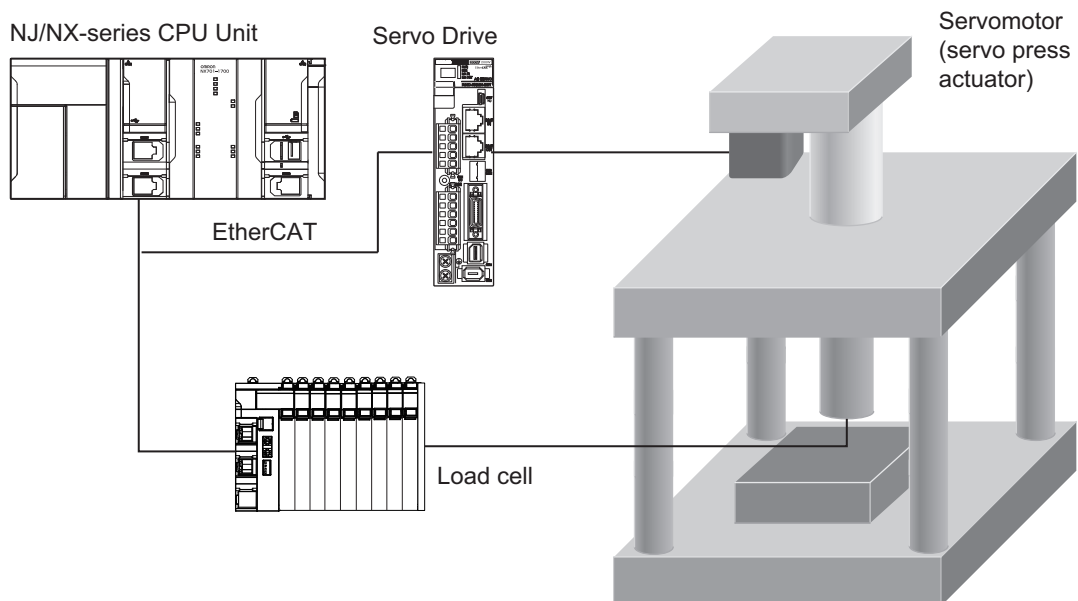


### Precautions for Correct Use

- The sample programming shows only the portion of a program that uses the function or function block from the library.
- When you use the actual device, include user programming for device safety instructions, interlocks, I/O with other devices, and other control procedures.
- Create a user program that will produce the intended device operation.
- Check the user program for proper execution before you use it for actual operation.

## Conditions

The following figure shows a system configuration for the sample programming.

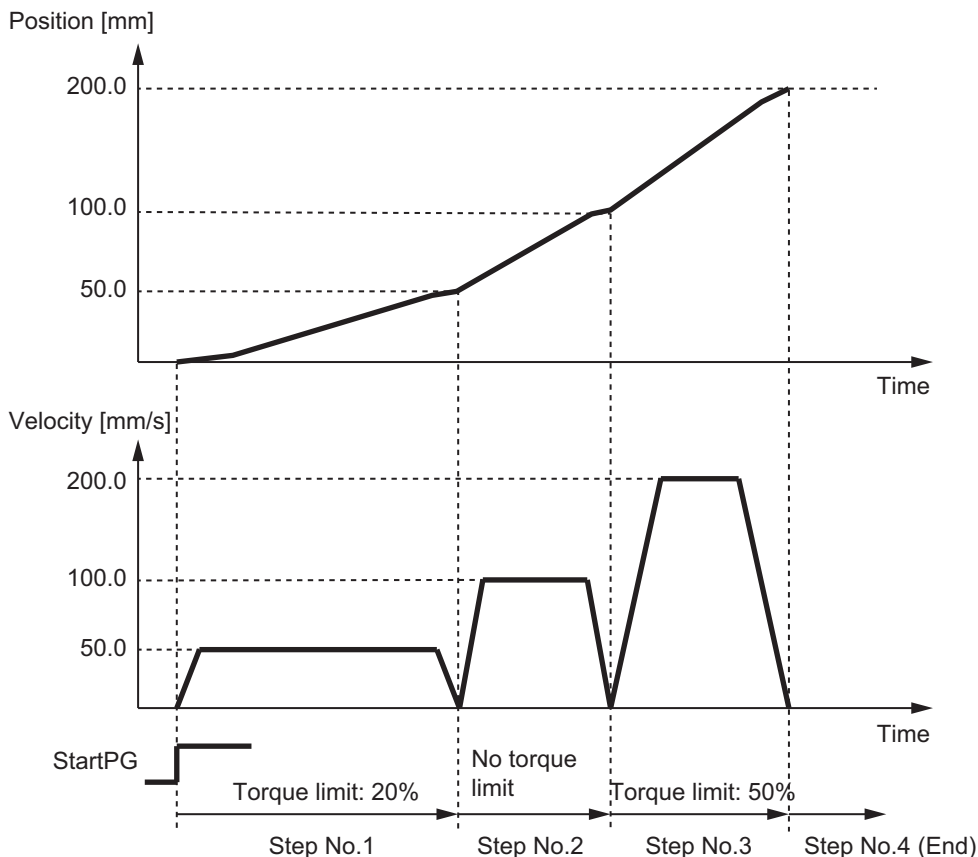


- MC\_Axis000 is allocated to the Servomotor as a axis. The display unit for the axis is millimeters.
- The unit for the load measurement value that is input by the load cell is newtons. In the program operation data for single-axis program operation, the measurement value is converted to a torque and handled as a percentage of the Servomotor's rated torque. The sample programming uses the LoadToTorque (Load-to-Torque Conversion) function to convert the units.

## Processing

- 1** Confirm that the axis can communicate and then turn ON the servo.
- 2** When the axis enters Servo ON state, execute homing.
- 3** Change the *StartPG* variable to TRUE after home has been defined to start single-axis program operation.

Single-axis program operation uses the following type of operation pattern. It is set when the *StartPG* variable changes to TRUE.



## Ladder Diagram

The following gives the main variables.

### ● Internal Variables

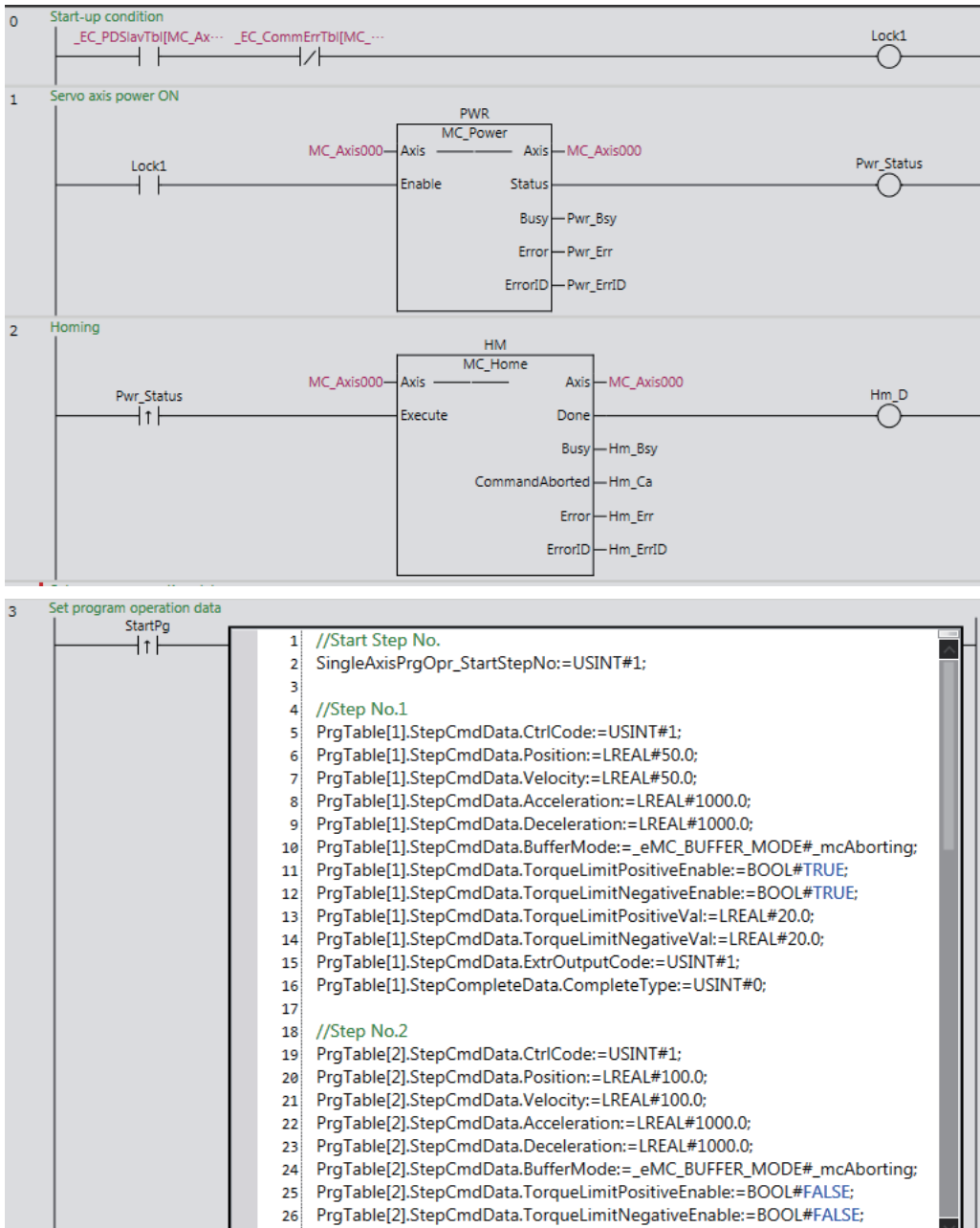
Name	Data Type	Initial Value	Comment
PWR	MC_Power		Instance of the MC_Power (Power Servo) instruction
HM	MC_Home		Instance of the MC_Home (Home) instruction
SET_TRQ_LMT	MC_SetTorqueLimit		Instance of the MC_SetTorqueLimit (Set Torque Limit) instruction
MV	MC_Move		Instance of the MC_Move (Positioning) instruction
STP	MC_Stop		Instance of the MC_Stop (Stop) instruction

Name	Data Type	Initial Value	Comment
SINGLE_AXIS_PRG_OPR	OmronLib\ServoPress\ SP_SingleAxisPrgOpr		Instance of the SP_SingleAxisPrgOpr (Single-axis Program Operation) function block
SINGLE_AXIS_CTRL	OmronLib\ServoPress\ SingleAxisCtrl		Instance of the SingleAxisCtrl (Single-axis Control) function block
PrgTable	ARRAY[1..50] OF Omron- Lib\ServoPress\sPRG_TABLE		Program table
PrgLoadAlarm	OmronLib\ServoPress\ sPRG_LOAD_ALARM		Program load alarm conditions
Lock1	BOOL		Starts Servo ON.
Pwr_Status	BOOL		Servo ON state
Hm_D	BOOL		Ends homing.
StartPg	BOOL		Starts single-axis program operation.
SingleAxisPrgOpr_Abt	BOOL		Abort trigger for single-axis program operation
SingleAxisPrgOpr_Start- StepNo	USINT		Start step number
SingleAxisCtrl_D	BOOL		Ends single-axis control execution.
SingleAxisCtrl_SingleCmd- ProfileNo	USINT		Single-axis command profile number
SingleAxisPrgOpr_Bsy	BOOL		Single-axis program operation busy
SingleAxisPrgOpr_Err	BOOL		Single-axis program operation fault
MCCmdExec	ARRAY[0..9] OF BOOL		Motion function block execution trigger
SingleCmdProfile	OmronLib\ServoPress\ sSINGLE_CMD_PROFILE		Single-axis command profile
StepCompleted	BOOL		Ends step.
StepExec	BOOL		Starts step.
CurrentStepNo	USINT		Current step number
MV_ex	BOOL		Execution trigger for interrupt motion operation
Load	LREAL		Load measurement value
TorqueVal	LREAL		Load-to-torque conversion value
MechaPm	OmronLib\ServoPress\ sMECHA_PARAMS		Servo press actuator machine parameters

### ● External Variables

Name	Data Type	Initial Value	Comment
MC_Axis000	_sAXIS_REF	---	Servo axis
_EC_PDSlavTbl	ARRAY[1..192] OF BOOL	---	Communications enabled status
_EC_CommErrTbl	ARRAY[1..192] OF BOOL	---	Communications error status

● Algorithm



```

27 PrgTable[2].StepCmdData.ExtrOutputCode:=USINT#2;
28 PrgTable[2].StepCompleteData.CompleteType:=USINT#0;
29
30 //Step No.3
31 PrgTable[3].StepCmdData.CtrlCode:=USINT#1;
32 PrgTable[3].StepCmdData.Position:=LREAL#200.0;
33 PrgTable[3].StepCmdData.Velocity:=LREAL#200.0;
34 PrgTable[3].StepCmdData.Acceleration:=LREAL#1000.0;
35 PrgTable[3].StepCmdData.Deceleration:=LREAL#1000.0;
36 PrgTable[3].StepCmdData.BufferMode:=-_eMC_BUFFER_MODE#_mcAborting;
37 PrgTable[3].StepCmdData.TorqueLimitPositiveEnable:=BOOL#TRUE;
38 PrgTable[3].StepCmdData.TorqueLimitNegativeEnable:=BOOL#TRUE;
39 PrgTable[3].StepCmdData.TorqueLimitPositiveVal:=LREAL#50.0;
40 PrgTable[3].StepCmdData.TorqueLimitNegativeVal:=LREAL#50.0;
41 PrgTable[3].StepCmdData.ExtrOutputCode:=USINT#3;
42 PrgTable[3].StepCompleteData.CompleteType:=USINT#0;
43
44 //Step No.4 (Operation end)
45 PrgTable[4].StepCmdData.CtrlCode:=USINT#0;
46 PrgTable[4].StepCmdData.TorqueLimitPositiveEnable:=BOOL#FALSE;
47 PrgTable[4].StepCmdData.TorqueLimitNegativeEnable:=BOOL#FALSE;
48 PrgTable[4].StepCmdData.ExtrOutputCode:=USINT#4;
49 PrgTable[4].StepCompleteData.CompleteType:=USINT#0;

```

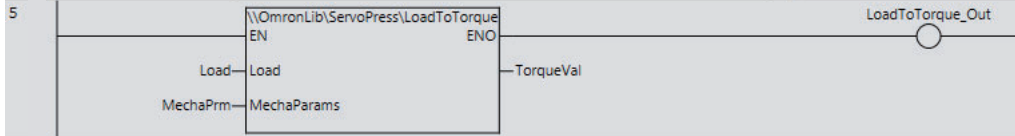
4 \*\*\*\* LoadToTorque Program Sample \*\*\*\*

P\_First\_Run

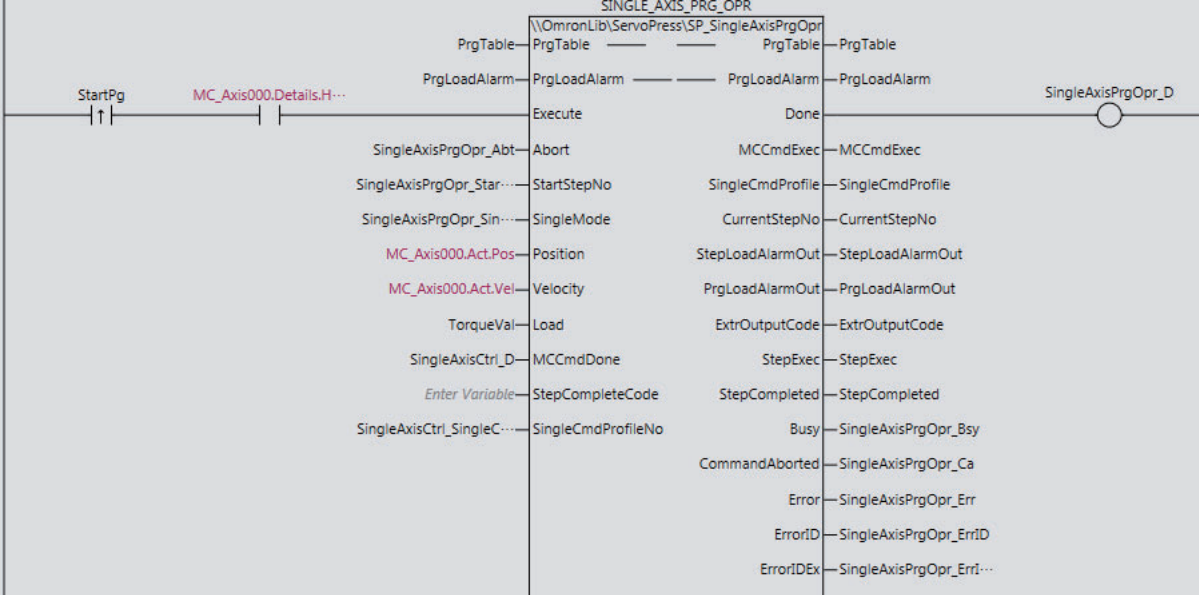
```

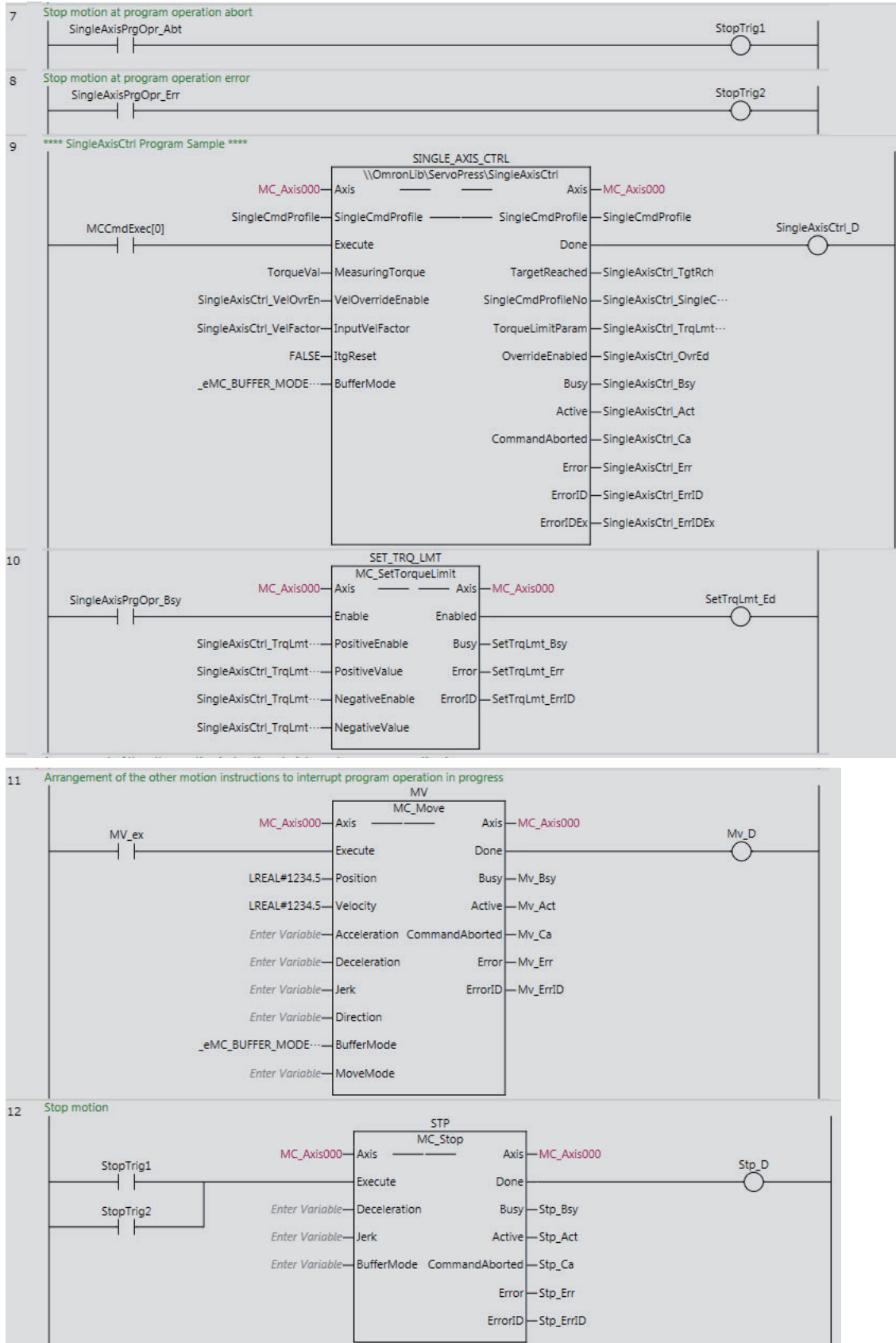
1 //Mechanical parameters
2 MechaPrm.ReductionGearParam.Rn:=1;
3 MechaPrm.ReductionGearParam.Rd:=5;
4 MechaPrm.ReductionGearParam.n2:=0.9;
5 MechaPrm.BallScrewParam.n1:=0.95;
6 MechaPrm.BallScrewParam.R:=10.0;
7 MechaPrm.MotorRatedParam.Tr:=0.64;

```



6 \*\*\*\* SP\_SingleAxisPrgOpr Program Sample \*\*\*\*





## ST

The following gives the main variables.

### ● Internal Variables

Name	Data Type	Initial Value	Comment
PWR	MC_Power		Instance of the MC_Power (Power Servo) instruction
HM	MC_Home		Instance of the MC_Home (Home) instruction
SET_TRQ_LMT	MC_SetTorqueLimit		Instance of the MC_SetTorqueLimit (Set Torque Limit) instruction
MV	MC_Move		Instance of the MC_Move (Positioning) instruction
STP	MC_Stop		Instance of the MC_Stop (Stop) instruction
SINGLE_AXIS_PRG_OPR	OmronLib\ServoPress\ SP_SingleAxisPrgOpr		Instance of the SP_SingleAxisPrgOpr (Single-axis Program Operation) function block
SINGLE_AXIS_CTRL	OmronLib\ServoPress\ SingleAxisCtrl		Instance of the SingleAxisCtrl (Single-axis Control) function block
PrgTable	ARRAY[1..50] OF OmronLib\ ServoPress\sPRG_TABLE		Program table
PrgLoadAlarm	OmronLib\ServoPress\ sPRG_LOAD_ALARM		Program load alarm conditions
Lock1	BOOL		Starts Servo ON.
Pwr_Status	BOOL		Servo ON state
Hm_D	BOOL		Ends homing.
StartPg	BOOL		Starts single-axis program operation.
SingleAxisPrgOpr_Abt	BOOL		Abort trigger for single-axis program operation
SingleAxisPrgOpr_Start-StepNo	USINT		Start step number
SingleAxisCtrl_D	BOOL		Ends single-axis control execution.
SingleAxisCtrl_SingleCmd-ProfileNo	USINT		Single-axis command profile number
SingleAxisPrgOpr_Bsy	BOOL		Single-axis program operation busy
SingleAxisPrgOpr_Err	BOOL		Single-axis program operation fault
MCCmdExec	ARRAY[0..9] OF BOOL		Motion function block execution trigger
SingleCmdProfile	OmronLib\ServoPress\ sSINGLE_CMD_PROFILE		Single-axis command profile
StepCompleted	BOOL		Ends step.
StepExec	BOOL		Starts step.
CurrentStepNo	USINT		Current step number
MV_ex	BOOL		Execution trigger for interrupt motion operation
Load	LREAL		Load measurement value
TorqueVal	LREAL		Load-to-torque conversion value
MechaPrm	OmronLib\ServoPress\ sMECHA_PARAMS		Servo press actuator machine parameters

## ● External Variables

Name	Data Type	Initial Value	Comment
MC_Axis000	_sAXIS_REF	---	Servo axis
_EC_PDSlavTbl	ARRAY[1..192] OF BOOL	---	Communications enabled status
_EC_CommErrTbl	ARRAY[1..192] OF BOOL	---	Communications error status

## ● Algorithm

```
//Start-up condition
IF ( _EC_PDSlavTbl [MC_Axis000.Cfg.NodeAddress]=TRUE)
AND ( _EC_CommErrTbl [MC_Axis000.Cfg.NodeAddress]=FALSE) THEN
  Lock1:=TRUE;
ELSE
  Lock1:=FALSE;
END_IF;

//R_TRIG (Rise edge)
inst1_R_TRIG( Clk := Pwr_Status, Q => UpTrig_Pwr_Status );
inst2_R_TRIG( Clk := StartPg, Q => UpTrig_StartPg );
inst3_R_TRIG( Clk := PrgRsltCSVWrite_StartPg, Q => UpTrig_PrgRsltCSVWrite_StartPg );
inst4_R_TRIG( Clk := PrgTraceCSVWrite_StartPg, Q => UpTrig_PrgTraceCSVWrite_StartPg );
inst5_R_TRIG( Clk := XY_Graph_StartPg, Q => UpTrig_XY_Graph_StartPg );

//Start single axis program operation after homing complete.
IF (MC_Axis000.Details.Homed=TRUE) THEN
  SingleAxisPrgOpr_Ex:=UpTrig_StartPg;
END_IF;

//Stop motion at program operation abort.
IF SingleAxisPrgOpr_Abt=TRUE THEN
  StopTrig1:=TRUE;
ELSE
  StopTrig1:=FALSE;
END_IF;

//Stop motion at program operation error.
IF SingleAxisPrgOpr_Err=TRUE THEN
  StopTrig2:=TRUE;
ELSE
  StopTrig2:=FALSE;
END_IF;

//Set program operation data.
IF SingleAxisPrgOpr_Ex=TRUE THEN
  //Start Step No.
  SingleAxisPrgOpr_StartStepNo:=USINT#1;

  //Step No.1
  PrgTable[1].StepCmdData.CtrlCode:=USINT#1;
  PrgTable[1].StepCmdData.Position:=LREAL#50.0;
  PrgTable[1].StepCmdData.Velocity:=LREAL#50.0;
  PrgTable[1].StepCmdData.Acceleration:=LREAL#1000.0;
  PrgTable[1].StepCmdData.Deceleration:=LREAL#1000.0;
  PrgTable[1].StepCmdData.BufferMode:=_eMC_BUFFER_MODE#_mcAborting;
  PrgTable[1].StepCmdData.TorqueLimitPositiveEnable:=BOOL#TRUE;
  PrgTable[1].StepCmdData.TorqueLimitNegativeEnable:=BOOL#TRUE;
  PrgTable[1].StepCmdData.TorqueLimitPositiveVal:=LREAL#20.0;
  PrgTable[1].StepCmdData.TorqueLimitNegativeVal:=LREAL#20.0;
  PrgTable[1].StepCmdData.ExtrOutputCode:=USINT#1;
  PrgTable[1].StepCompleteData.CompleteType:=USINT#0;
```



```

//Step No.2
PrgTable[2].StepCmdData.CtrlCode:=USINT#1;
PrgTable[2].StepCmdData.Position:=LREAL#100.0;
PrgTable[2].StepCmdData.Velocity:=LREAL#100.0;
PrgTable[2].StepCmdData.Acceleration:=LREAL#1000.0;
PrgTable[2].StepCmdData.Deceleration:=LREAL#1000.0;
PrgTable[2].StepCmdData.BufferMode:=_eMC_BUFFER_MODE#_mcAborting;
PrgTable[2].StepCmdData.TorqueLimitPositiveEnable:=BOOL#FALSE;
PrgTable[2].StepCmdData.TorqueLimitNegativeEnable:=BOOL#FALSE;
PrgTable[2].StepCmdData.ExtrOutputCode:=USINT#2;
PrgTable[2].StepCompleteData.CompleteType:=USINT#0;

//Step No.3
PrgTable[3].StepCmdData.CtrlCode:=USINT#1;
PrgTable[3].StepCmdData.Position:=LREAL#200.0;
PrgTable[3].StepCmdData.Velocity:=LREAL#200.0;
PrgTable[3].StepCmdData.Acceleration:=LREAL#1000.0;
PrgTable[3].StepCmdData.Deceleration:=LREAL#1000.0;
PrgTable[3].StepCmdData.BufferMode:=_eMC_BUFFER_MODE#_mcAborting;
PrgTable[3].StepCmdData.TorqueLimitPositiveEnable:=BOOL#TRUE;
PrgTable[3].StepCmdData.TorqueLimitNegativeEnable:=BOOL#TRUE;
PrgTable[3].StepCmdData.TorqueLimitPositiveVal:=LREAL#50.0;
PrgTable[3].StepCmdData.TorqueLimitNegativeVal:=LREAL#50.0;
PrgTable[3].StepCmdData.ExtrOutputCode:=USINT#3;
PrgTable[3].StepCompleteData.CompleteType:=USINT#0;

//Step No.4 (Operation end)
PrgTable[4].StepCmdData.CtrlCode:=USINT#0;
PrgTable[4].StepCmdData.TorqueLimitPositiveEnable:=BOOL#FALSE;
PrgTable[4].StepCmdData.TorqueLimitNegativeEnable:=BOOL#FALSE;
PrgTable[4].StepCmdData.ExtrOutputCode:=USINT#4;
PrgTable[4].StepCompleteData.CompleteType:=USINT#0;
END_IF;

//Mechanical parameters
MechaPrm.ReductionGearParam.Rn:=1;
MechaPrm.ReductionGearParam.Rd:=5;
MechaPrm.ReductionGearParam.n2:=0.9;
MechaPrm.BallScrewParam.n1:=0.95;
MechaPrm.BallScrewParam.R:=10.0;
MechaPrm.MotorRatedParam.Tr:=0.64;

TorqueVal:=\OmronLib\ServoPress\LoadToTorque(
  EN:=TRUE,
  ENO=>LoadToTorque_Out,
  Load:=Load,
  MechaParams:=MechaPrm
);

//MC_Power
PWR(
  Axis := MC_Axis000,
  Enable := Lock1,
  Status => Pwr_Status,
  Busy => Pwr_Bsy,
  Error => Pwr_Err,
  ErrorID => Pwr_ErrID
);

//MC_Home
HM(
  Axis := MC_Axis000,
  Execute := UpTrig_Pwr_Status,
  Done => Hm_D,

```

```

    Busy => Hm_Bsy,
    CommandAborted=> Hm_Ca,
    Error => Hm_Err,
    ErrorID => Hm_ErrID
);

//SP_SingleAxisPrgOpr
SINGLE_AXIS_PRG_OPR(
    PrgTable := PrgTable,
    PrgLoadAlarm := PrgLoadAlarm,
    Execute := SingleAxisPrgOpr_Ex,
    Abort := SingleAxisPrgOpr_Abt,
    StartStepNo := SingleAxisPrgOpr_StartStepNo,
    SingleMode := SingleAxisPrgOpr_SingleMode,
    Position := MC_Axis000.Act.Pos,
    Velocity := MC_Axis000.Act.Vel,
    Load := TorqueVal,
    MCCmdDone := SingleAxisCtrl_D,
    SingleCmdProfileNo := SingleAxisCtrl_SingleCmdProfileNo,
    Done => SingleAxisPrgOpr_D,
    MCCmdExec => MCCmdExec,
    SingleCmdProfile => SingleCmdProfile,
    CurrentStepNo => CurrentStepNo,
    StepLoadAlarmOut => StepLoadAlarmOut,
    PrgLoadAlarmOut => PrgLoadAlarmOut,
    ExtrOutputCode => ExtrOutputCode,
    StepExec => StepExec,
    StepCompleted => StepCompleted,
    Busy => SingleAxisPrgOpr_Bsy,
    CommandAborted=> SingleAxisPrgOpr_Ca,
    Error => SingleAxisPrgOpr_Err,
    ErrorID => SingleAxisPrgOpr_ErrID,
    ErrorIDEx => SingleAxisPrgOpr_ErrIDEx
);

//SingleAxisCtrl
SINGLE_AXIS_CTRL(
    Axis := MC_Axis000,
    SingleCmdProfile := SingleCmdProfile,
    Execute := MCCmdExec[0],
    MeasuringTorque:=TorqueVal,
    VelOverrideEnable := SingleAxisCtrl_VelOvrEn,
    InputVelFactor := SingleAxisCtrl_VelFactor,
    ItgReset := BOOL#FALSE,
    BufferMode := _eMC_BUFFER_MODE#_mcAborting,
    Done => SingleAxisCtrl_D,
    TargetReached => SingleAxisCtrl_TgtRch,
    SingleCmdProfileNo => SingleAxisCtrl_SingleCmdProfileNo,
    TorqueLimitParam => SingleAxisCtrl_TrqLmtPrm,
    OverrideEnabled => SingleAxisCtrl_OvrEd,
    Busy => SingleAxisCtrl_Bsy,
    Active => SingleAxisCtrl_Act,
    CommandAborted=> SingleAxisCtrl_Ca,
    Error => SingleAxisCtrl_Err,
    ErrorID => SingleAxisCtrl_ErrID,
    ErrorIDEx => SingleAxisCtrl_ErrIDEx
);

//MC_SetTorqueLimit
SET_TRQ_LMT(
    Axis:=MC_Axis000,
    Enable:=SingleAxisPrgOpr_Bsy,
    PositiveEnable := SingleAxisCtrl_TrqLmtPrm.TorqueLimitPositiveEnable,
    PositiveValue := SingleAxisCtrl_TrqLmtPrm.TorqueLimitPositiveVal,
    NegativeEnable := SingleAxisCtrl_TrqLmtPrm.TorqueLimitNegativeEnable,

```

```

    NegativeValue := SingleAxisCtrl_TrqLmtPrm.TorqueLimitNegativeVal,
    Enabled => SetTrqLmt_Ed,
    Busy => SetTrqLmt_Bsy,
    Error => SetTrqLmt_Err,
    ErrorID => SetTrqLmt_ErrID
);

//Arrangement of the other motion instructions to interrupt program operation in
progress.
//MC_Move
MV(
    Axis := MC_Axis000,
    Execute := Mv_Ex,
    Position := LREAL#1234.5,
    Velocity := LREAL#1234.5,
    BufferMode := _eMC_BUFFER_MODE#_mcAborting,
    Done => Mv_D,
    Busy => Mv_Bsy,
    Active => Mv_Act,
    CommandAborted=> Mv_Ca,
    Error => Mv_Err,
    ErrorID => Mv_ErrID
);

//MC_Stop
STP(
    Axis := MC_Axis000,
    Execute := (StopTrig1 OR StopTrig2),
    Done => Stp_D,
    Busy => Stp_Bsy,
    Active => Stp_Act,
    CommandAborted=> Stp_Ca,
    Error => Stp_Err,
    ErrorID => Stp_ErrID
);

```

# SP\_PrgStatusCtrl

The SP\_PrgStatusCtrl function block interprets the program data and controls the next function block step by step.

Function block name	Name	FB/FUN	Graphic expression	ST expression
SP_PrgStatusCtrl	Program Status Control	FB		<pre>SP_PrgStatusCtrl_instance (   PrgTable,   Execute,   SingleStepMode,   StartStepNo,   Abort,   StepCompleted,   StepExit,   NextStepNo,   Done,   MCCmdExec,   SingleCmdProfile,   StepExec,   CurrentStepNo,   ExtrOutputCode,   Busy,   ExtrOutputCode,   Busy,   CommandAborted,   Error,   ErrorID,   ErrorIDEx);</pre>

## Function Block and Function Information

Item	Description
Library file name	OmronLib_ServoPress_V1_0.slr OmronLib_ServoPress_V2_0.slr
Namespace	OmronLib\ServoPress
Function block and function number	00072
Publish/Do not publish source code	Not published.
Function block and function version	1.00

## Variables

### Input Variables

Name	Meaning	Data type	Default	Valid range	Unit	Description
Execute	Execute	BOOL	FALSE	Depends on data type.	---	Execute trigger for this function block Executes the function block when it changes to TRUE.
SingleStep-Mode	Single Mode	BOOL	FALSE	Depends on data type.	---	This variable indicates whether to execute operation in Single Mode. TRUE: Execute operation in Single Mode. FALSE: Do not execute operation in Single Mode.
StartStepNo	Execution Start Step Number	USINT	0	0 to 50	---	Step number from which to start execution
Abort	Abort	BOOL	FALSE	Depends on data type.	---	Abort trigger for this function block The function block is aborted when it changes to TRUE.
Step Completed	Step Completed Trigger	BOOL	FALSE	Depends on data type.	---	The step is completed when the step completion trigger changes to TRUE.
StepExit	Step Abort Trigger	BOOL	FALSE	Depends on data type.	---	The step is aborted when the step abortion trigger changes to TRUE.
NextStepNo	Next Step Number	USINT	0	0 to 50	---	Number of the step that is executed after the step is aborted. 0: End single-axis program operation.

## Output Variables

Name	Meaning	Data type	Valid range	Unit	Description
Done	Done	BOOL	Depends on data type.	---	Completion of execution TRUE: Normal end FALSE: Error end, execution in progress, or execution condition not met.
MCCmdExec	Motion Function Block Execution Trigger	ARRAY[0..9] OF BOOL	Depends on data type.	---	Execution trigger for the motion function block. TRUE: The function block is executed. FALSE: The function block is not executed.
SingleCmdProfile	Single-axis Command Profile	OmronLib\ServoPress\sSINGLE_CMD_PROFILE	---	---	Single-axis control instruction table for each step
StepExec	Step Start Trigger	BOOL	Depends on data type.	---	Step start trigger
CurrentStepNo	Current Step Number	USINT	0 to 50	---	The number of the step being executed.
ExtrOutputCode	External Output Code	USINT	Depends on data type.	---	Code that is output for each step that is completed.
Busy	Executing	BOOL	Depends on data type.	---	Executing TRUE: Executing FALSE: Not executing
Command Aborted	Instruction Aborted	BOOL	Depends on data type.	---	Execution aborted This variable changes to TRUE if the function block is aborted.
Error	Error	BOOL	Depends on data type.	---	Error end TRUE: Error end FALSE: Normal end, executing, or execution conditions not met
ErrorID	Error Code	WORD	*1	---	An error code is output if an error occurs. The value is WORD#16#0 for a normal end.
ErrorIDEx	Expansion Error Code	DWORD	*1	---	An expansion error code is output if an error occurs. The value is DWORD#16#0 for a normal end.

\*1. For details, refer to *Troubleshooting* on page 108.

### ● Structure

The data type of the *SingleCmdProfile* output variable is the structure *OmronLib\ServoPress\sSINGLE\_CMD\_PROFILE*. Refer to *Structure* in *SP\_SingleAxisPrgOpr* on page 42 for details.

## In-Out Variables

Name	Meaning	Data type	Valid range	Unit	Description
PrgTable	Program Tables	ARRAY[1..50] OF OmronLib\ ServoPress\ sPRG_TABLE	---	---	Program tables that contain settings for each step

### ● Structure

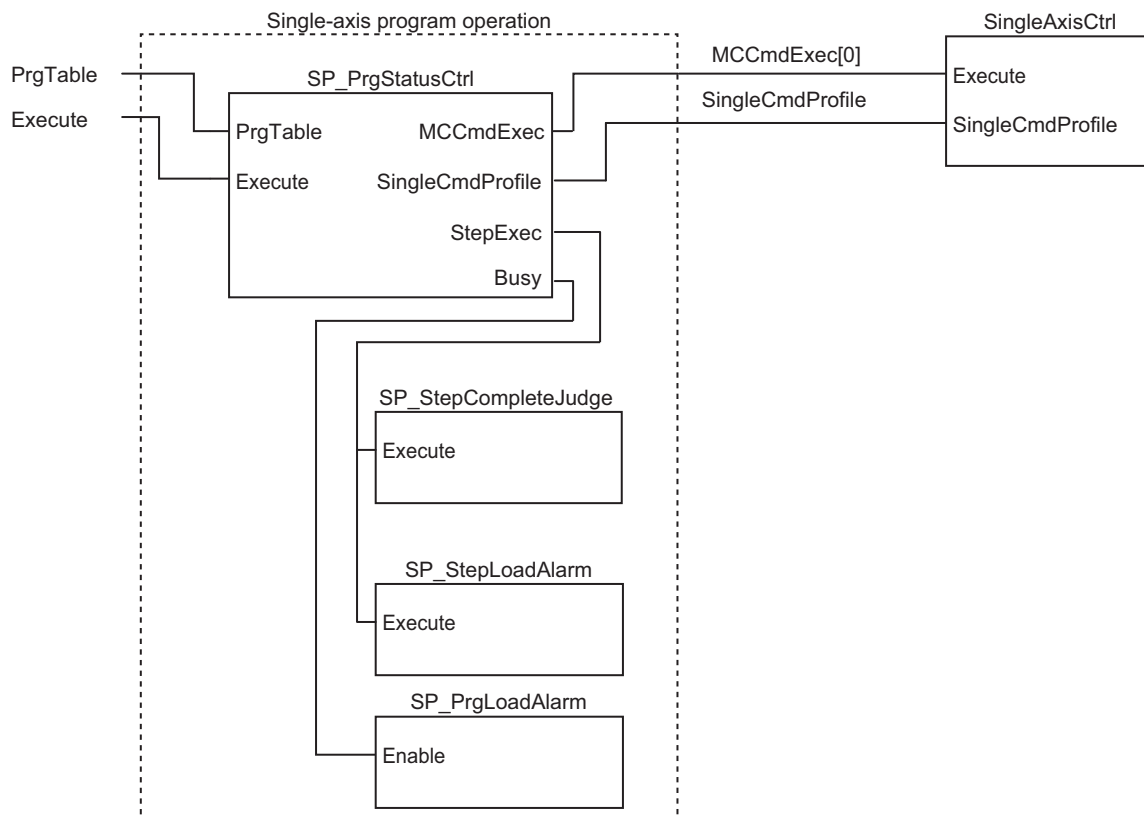
The data type of the *PrgTable* in-out variable is the structure OmronLib\ServoPress\sPRG\_TABLE. Refer to *Structure* in *SP\_SingleAxisPrgOpr* on page 42 for details.

## Function

The SP\_PrgStatusCtrl function block interprets *PrgTable* (Program Tables) and controls the next function block for each step. The following function blocks can come next: SingleAxisCtrl (Single-axis Control), SP\_StepCompleteJudge (Step Completion Determination), and SP\_StepLoadAlarm (Step Load Alarm Determination).

## Relation with Other Function Blocks

This function block provides the single-axis program operation in combination with the SP\_StepCompleteJudge (Step Completion Determination), SP\_StepLoadAlarm (Step Load Alarm Determination), and SP\_PrgLoadAlarm (Program Load Alarm Determination) function blocks. The following conceptual diagram illustrates the relationships between the functions blocks and variables. This figure excludes the variables that are not necessary for this description.



- The *MCCmdExec[0]* (Motion Function Block Execution Trigger) variable that is output from the SP\_PrgStatusCtrl function block is used as the *Execute* variable for the SingleAxisCtrl (Single-axis Control) function block.
- The *SingleCmdProfile* (Single-axis Command Profile) variable that is output from the SP\_PrgStatusCtrl function block is used as the *SingleCmdProfile* (Single-axis Command Profile) variable for the SingleAxisCtrl (Single-axis Control) function block.
- The *StepExec* (Step Start Trigger) variable that is output from the SP\_PrgStatusCtrl function block is used as the *Execute* variable for the SP\_StepCompleteJudge (Step Completion Determination) function block and the SP\_StepLoadAlarm (Step Load Alarm Determination) function block.
- The *Busy* (Executing) variable that is output from the SP\_PrgStatusCtrl function block is used as the *Enable* variable in the SP\_PrgLoadAlarm (Program Load Alarm Determination) function block.



### Precautions for Safe Use

Execute the following four functions block which consist of single-axis program operation in the same task. If these function blocks are executed in the different task, the completion for each step or the completion for the program operation may not be correctly determined.

- SP\_PrgStatusCtrl (Program Status Control)
- SP\_StepCompleteJudge (Step Completion Determination)
- SP\_StepLoadAlarm (Step Load Alarm Determination)
- SP\_PrgLoadAlarm (Program Load Alarm Determination)



## PrgTable (Program Tables)

---

The SP\_PrgStatusCtrl function block interprets *PrgTable* (Program Tables). For detailed on the specifications of *PrgTable*, refer to *Defining Single-axis Program Operation That Combines Multiple Single-axis Motion Controls* on page 56 in the description of the SP\_SingleAxisPrgOpr (Single-axis Program Operation) function block.

## Meanings of Variables

---

The meanings of the other variables are described below.

- **SingleStepMode (Single Mode)**

The execution of only a certain step without execution of a step sequence set with *PrgTable* (Program Tables) is called Single Mode. If you change the value of *SingleStepMode* (Single Mode) to TRUE and execute this function block, only the step specified with *StartStepNo* (Execution Start Step Number) is executed.

- **StartStepNo (Execution Start Step Number)**

This variable is used to specify the step number to start execution of single-axis program operation. For example, if *StartStepNo* is set to USINT#4, the execution of the operation starts from *PrgTable*[4].

- **StepCompleted (Step Completed Trigger)**

This variable is a trigger signal that indicates when a step is completed. Every time a step is ended, its value changes to TRUE for one task period.

- **StepExit (Step Abort Trigger)**

This variable is a trigger signal that indicates when a step is aborted. If the value of *StepExit* (Step Abort Trigger) changes to TRUE, the current step is canceled and processing moves to the step that is specified with *NextStepNo* (Next Step Number).

- **NextStepNo (Next Step Number)**

If the value of *StepExit* (Step Abort Trigger) changes to TRUE, the processing moves to the step that is specified with *NextStepNo* (Next Step Number).

- **MCCmdExec (Motion Function Block Execution Trigger)**

An execution trigger for the SingleAxisCtrl (Single-axis Control) function block that connects to the SP\_SingleAxisPrgOpr function block is output to *MCCmdExec*[0]. Connect *MCCmdExec*[0] to the *Execute* input variable in the SingleAxisCtrl (Single-axis Control) function block.

- **SingleCmdProfile (Single-axis Command Profile)**

*SingleCmdProfile* contains data that describes the single-axis motion control for each step and it is input to the SingleAxisCtrl (Single-axis Control) function block. Connect this variable to the *SingleCmdProfile* in-out variable in the SingleAxisCtrl (Single-axis Control) function block.

- **StepExec (Step Start Trigger)**

This variable is used to notify the next function block that a step has started. Every time a step is started, its value changes to TRUE for one task period.

● **CurrentStepNo (Current Step Number)**

This variable gives the number of the current step. When single-axis program operation is stopped, the final output value is retained.

● **ExtrOutputCode (External Output Code)**

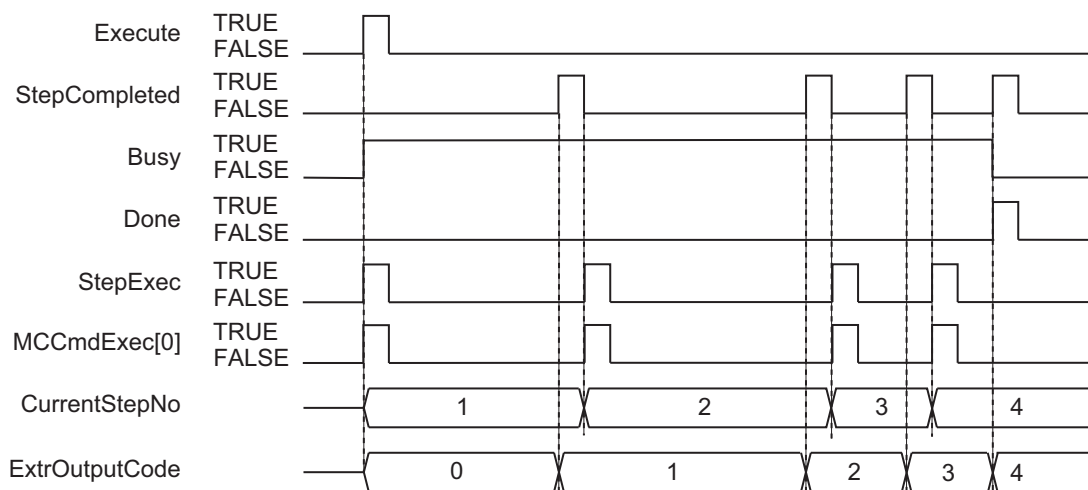
This is the external output code that is set with *PrgTable.ExtrOutputCode* (External Output Code). The code is output to the next task period for each step that is completed. Use it when you want to detect the completion of a specific step from outside the function block.

## Timing Charts

The following timing chart shows a case that the operation patterns shown in the sample programming of the SP\_SingleAxisPrgOpr (Single-axis Program Operation) function block are operated in this function block.

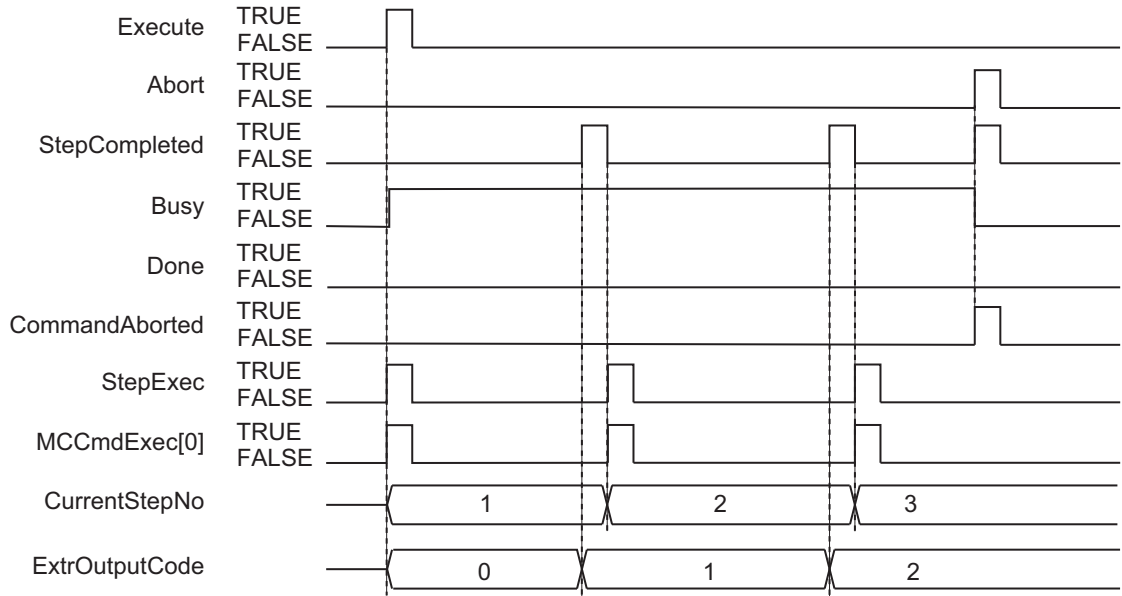
● **Normal End**

- *Busy* (Executing) changes to TRUE when *Execute* in the function block changes to TRUE.
- The value of *CurrentStepNo* (Current Step Number) is updated whenever a step is completed.
- After execution of the step is completed, the value of *CurrentStepNo* (Current Step Number) is retained until the value of *Execute* changes to TRUE again.



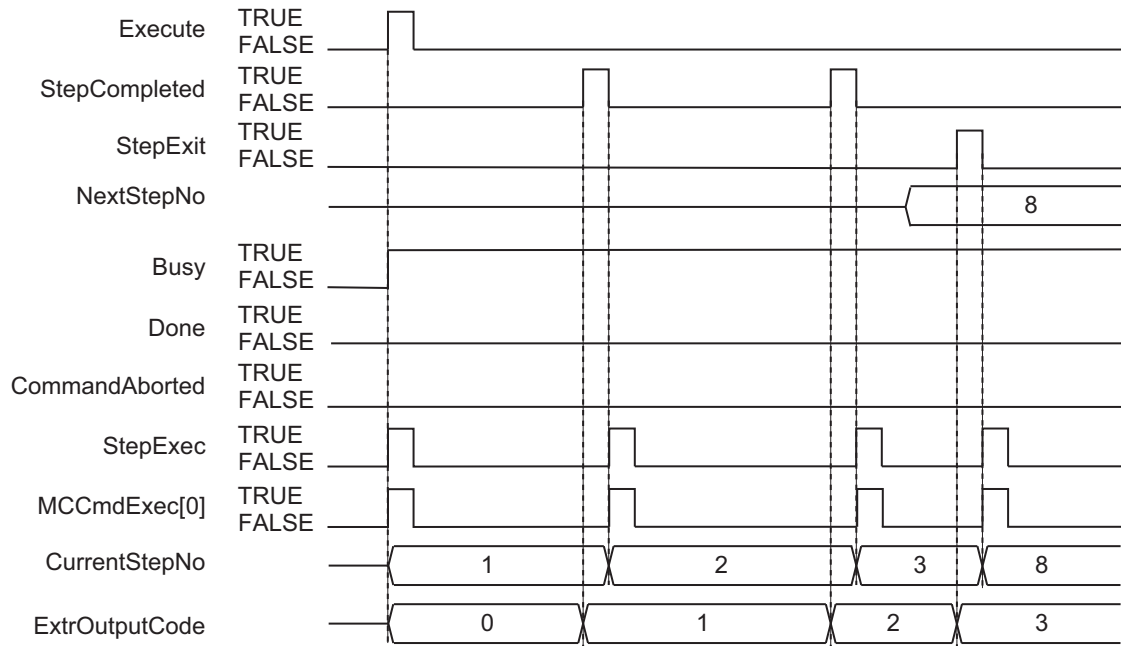
● **Aborting Execution**

- When the value of *Abort* is changed to TRUE, the processing is aborted.
- Even after the step is aborted, the value of *CommandAborted* (Instruction Aborted) is retained until the value of *Execute* changes to FALSE.
- *CurentStepNo* (Current Step Number) is retained until the value of *Execute* changes to TRUE again.
- *Abort* is given priority over *StepCompleted* (Step Completed Trigger).



● **Aborting Steps**

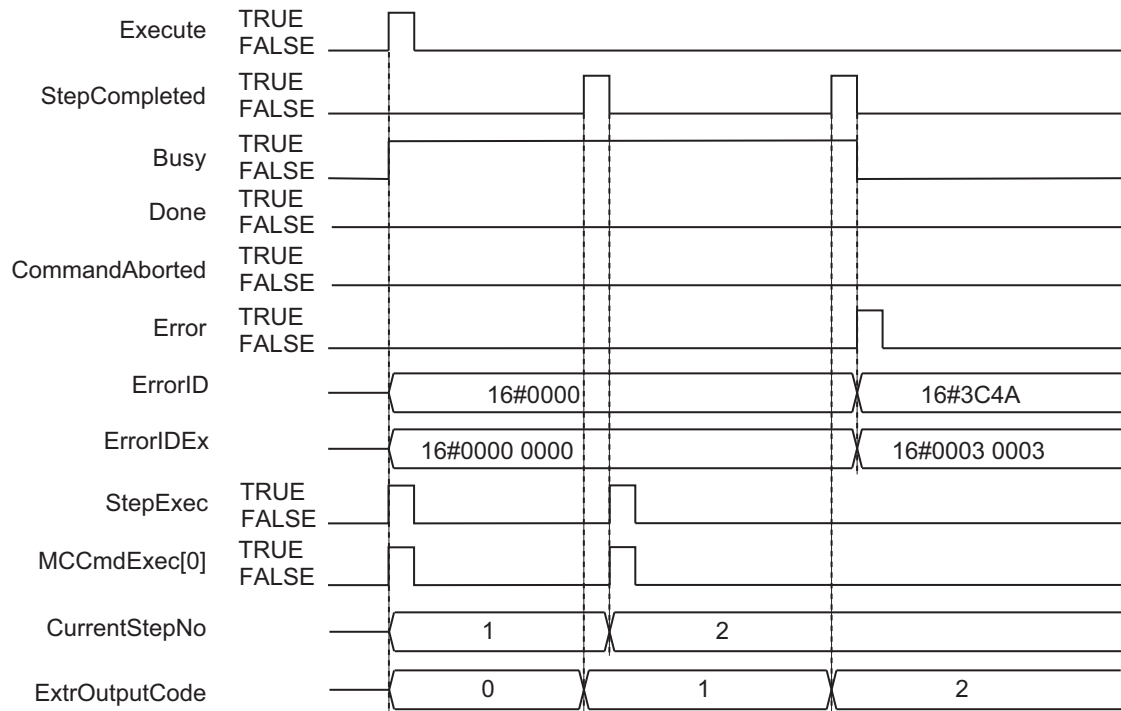
- If the value of *StepExit* (Step Abort Trigger) changes to TRUE, the current step is aborted and the step specified with *NextStepNo* (Next Step Number) is executed. The following shows a case that a value of 8 is specified to *NextStepNo*.
- *CurentStepNo* (Current Step Number) is retained until the value of *Execute* changes to TRUE again.
- If *StepExit* (Step Abort Trigger) and *StepCompleted* (Step Completed Trigger) change to TRUE at the same, *StepCompleted* is given priority.



## ● Error End

The following timing chart gives an example when a control method outside the range is specified to *StepCmdData* (Step Command Data) in the third step of *PrgTable* (Program Tables).

- If an error occurs during execution of this function block, *Error* changes to TRUE. You can find out the cause of the error by referring to the values output by *ErrorID* (Error Code) and *ErrorIDEx* (Expansion Error Code).
- If an error occurs, processing does not move to the next step.
- The value of *Error* is retained while the value of *Execute* is TRUE.



## Precautions for Correct Use

- The values of *SingleStepMode* (Single Mode) and *StartStepNo* (Execution Start Step Number) are valid while *Execute* is TRUE. The value is not refreshed even if it is changed during processing of this function block.
- If the value of *Abort*, *StepCompleted* (Step Completed Trigger), *StepExit* (Step Abort Trigger), *NextStepNo* (Next Step Number), or *PrgTable.StepCmdData* (Step Command Data) is changed during execution of this function block, the value is updated for the processing within the same task period.

## Troubleshooting

The error codes, expansion error codes, status, descriptions, and corrections are given in the following table.

Error code	Expansion error code	Status	Description	Correction
16#0000	16#00000000	Normal end		
16#3C4A	16#00000001	Execution start step number range exceeded	The value of <i>StartStepNo</i> (Execution Start Step Number) exceeded the valid range.	Check the valid range of the value of <i>StartStepNo</i> (Execution Start Step Number) and set the value within the valid range.
	16#00000002	Next step number range exceeded	The value of <i>NextStepNo</i> (Next Step Number) exceeded the valid range.	Check the valid range of the value of <i>NextStepNo</i> (Next Step Number) and set the value within the valid range.
	16#00□□0003* <sup>1</sup>	Control method range exceeded	The value of <i>CtrlCode</i> (Control Method) is outside the valid range.	Check the valid range of the value of <i>CtrlCode</i> (Control Method) and set the value within the valid range.
	16#00000004	Incorrect task setting	An attempt was made to execute this function block in a task period which is not a primary task period or a periodic task.	Execute this function block in a primary periodic task or a periodic task.

\*1. The boxes (□□) are the BCD values of *CurrentStepNo* (Current Step Number) when the error occurred. For example, if *CurrentStepNo* is USINT#11, *ErrorIDEx* is DWORD#16#00110003.

## Sample Programming

Single-axis program operation is implemented by combining the SP\_PrgStatusCtrl (Program Status Control) function block, the SP\_StepCompleteJudge (Step Completion Determination) function block, the SP\_StepLoadAlarm (Step Load Alarm Determination) function block, and the SP\_PrgLoadAlarm (Program Load Alarm Determination) function block.

This sample programming shows only the portion that executes single-axis program operation. It also shows only the minimum rungs required for operation of the sample programming shown for the SP\_SingleAxisPrgOpr (Single-axis Program Operation) function block. Add other rungs as necessary, such as stop rungs when a determination is made.

Refer to the descriptions of the ladder diagram and structured text program for the replacement method to apply this sample programming to the SP\_SingleAxisPrgOpr (Single-axis Program Operation) function block sample programming.



### Precautions for Correct Use

- The sample programming shows only the portion of a program that uses the function or function block from the library.
- When you use the actual device, include user programming for device safety instructions, interlocks, I/O with other devices, and other control procedures.
- Create a user program that will produce the intended device operation.
- Check the user program for proper execution before you use it for actual operation.

## Conditions

The conditions are the same as those for the SP\_SingleAxisPrgOpr (Single-axis Program Operation) function block sample programming.

## Processing

Change the *StartPG* variable to TRUE after home has been defined to start single-axis program operation.

## Ladder Diagram

The following gives the main variables.

### ● Internal Variables

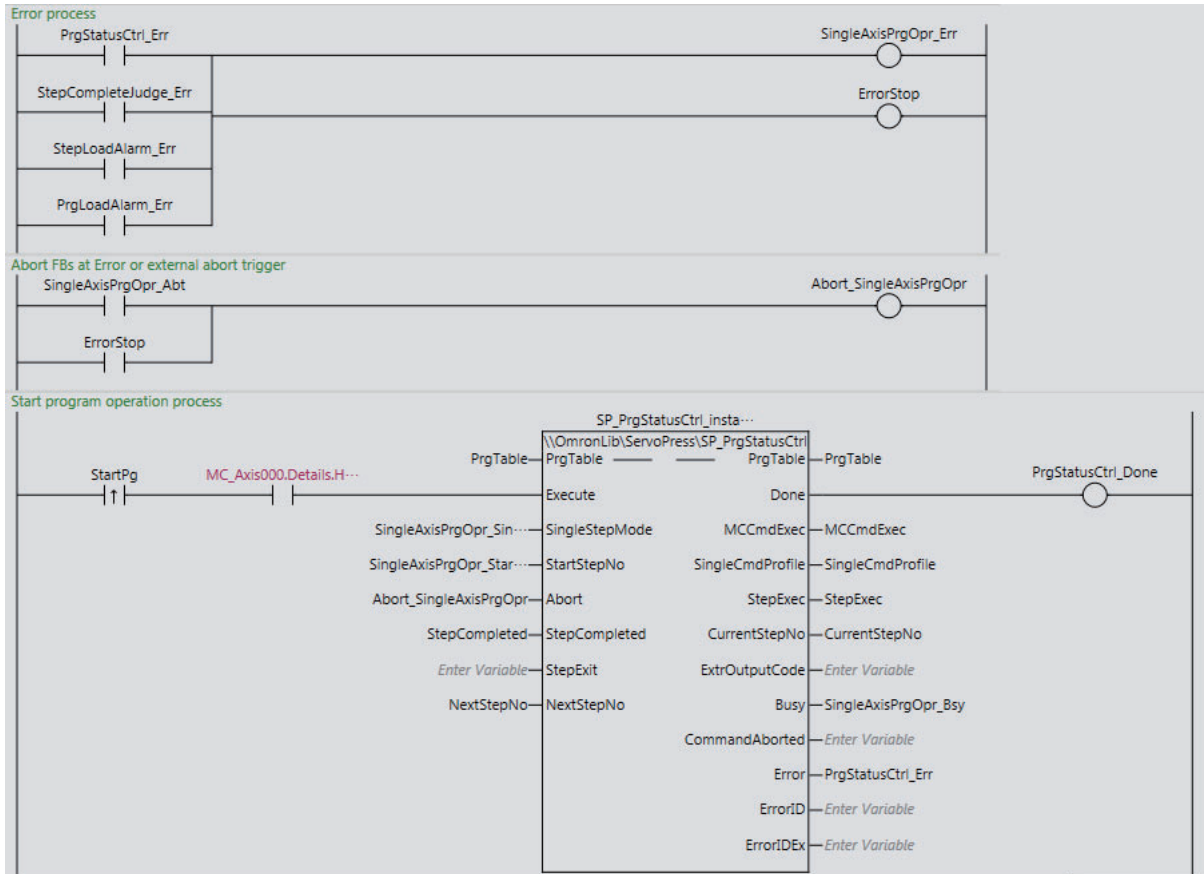
Name	Data Type	Initial Value	Comment
SP_PrgStatusCtrl_instance	OmronLib\ServoPress\ SP_PrgStatusCtrl		Instance of the SP_PrgStatusCtrl (Program Status Control) function block
SP_StepComplete-Judge_instance	OmronLib\ServoPress\ SP_StepCompleteJudge		Instance of the SP_StepCompleteJudge (Step Completion Determination) function block
SP_StepLoadAlarm_instance	OmronLib\ServoPress\ SP_StepLoadAlarm		Instance of the SP_StepLoadAlarm (Step Load Alarm Determination) function block
SP_PrgLoadAlarm_instance	OmronLib\ServoPress\ SP_PrgLoadAlarm		Instance of the SP_PrgLoadAlarm (Program Load Alarm Determination) function block
PrgTable	ARRAY[1..50] OF OmronLib\ ServoPress\sPRG_TABLE		Program table
PrgLoadAlarm	OmronLib\ServoPress\ sPRG_LOAD_ALARM		Program load alarm conditions
StartPg	BOOL		Starts single-axis program operation.
SingleAxisPrgOpr_Abt	BOOL		Abort trigger for single-axis program operation
SingleAxisPrgOpr_Start-StepNo	USINT		Start step number
SingleAxisCtrl_D	BOOL		Ends single-axis control execution.
SingleAxisCtrl_SingleCmd-ProfileNo	USINT		Single-axis command profile number
SingleAxisPrgOpr_Bsy	BOOL		Single-axis program operation busy
SingleAxisPrgOpr_Err	BOOL		Single-axis program operation fault
MCCmdExec	ARRAY[0..9] OF BOOL		Motion function block execution trigger
SingleCmdProfile	OmronLib\ServoPress\ sSINGLE_CMD_PROFILE		Single-axis command profile
StepCompleted	BOOL		Ends step.
StepExec	BOOL		Starts step.
CurrentStepNo	USINT		Current step number

● External Variables

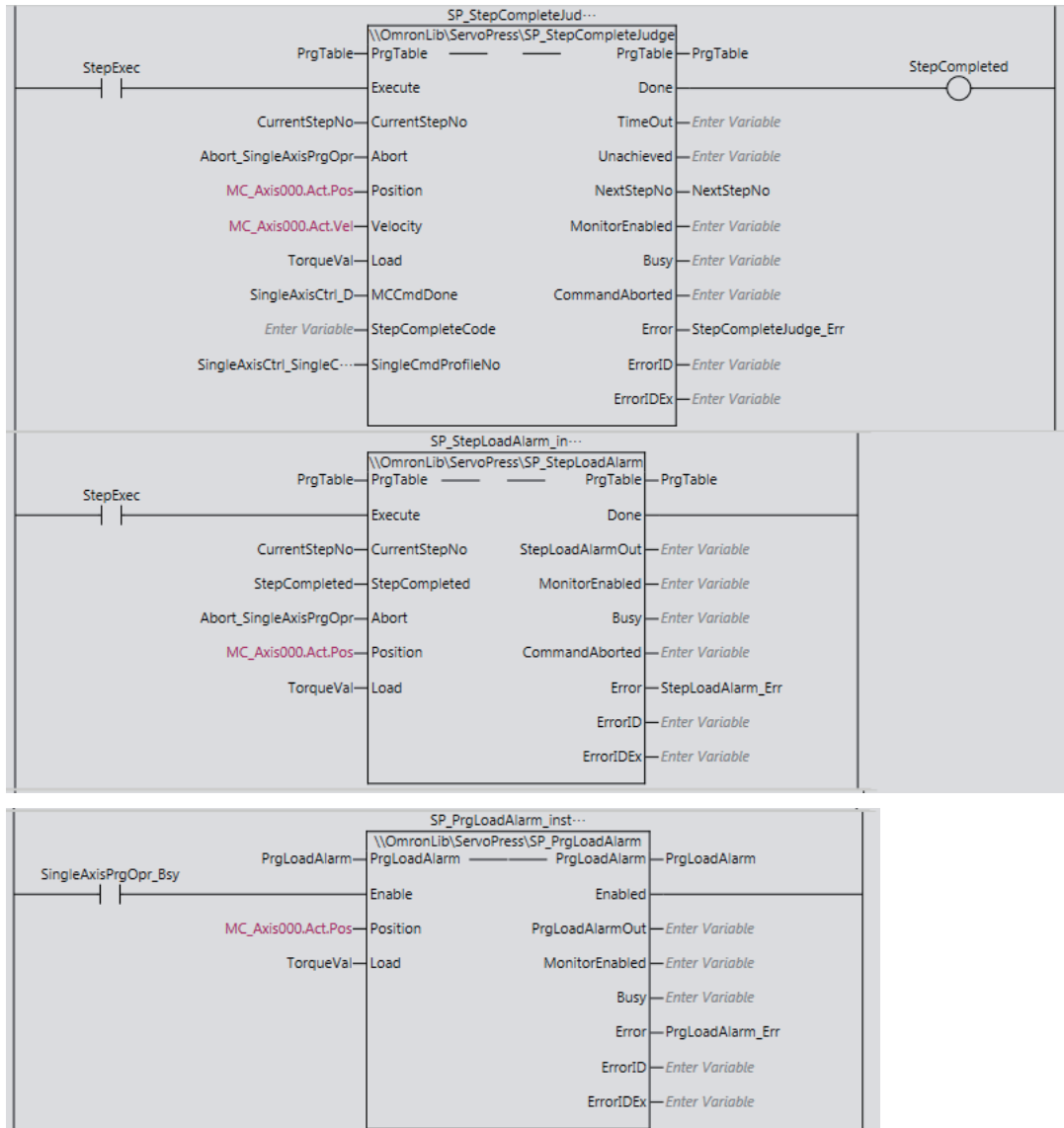
Name	Data Type	Initial Value	Comment
MC_Axis000	_sAXIS_REF	---	Servo axis

● Algorithm

The algorithm can be used in the same manner as in the SP\_SingleAxisPrgOpr (Single-axis Program Operation) function block sample programming by replacing the sixth rung of that sample programming (SP\_SingleAxisPrgOpr (Single-axis Program Operation) function block instance) with the following ladder programming.







● **How To Apply To the SP\_SingleAxisPrgOpr (Single-axis Program Operation) Function Block Sample Programming**

The algorithm can be used in the same manner as in the SP\_SingleAxisPrgOpr (Single-axis Program Operation) function block sample programming by replacing the sixth rung of that sample programming (i.e., instance of the SP\_SingleAxisPrgOpr (Single-axis Program Operation) function block) with the above ladder programming.

## ST

The following gives the main variables.

### ● Internal Variables

Name	Data Type	Initial Value	Comment
SP_PrgStatusCtrl_instance	OmronLib\ServoPress\ SP_PrgStatusCtrl		Instance of the SP_PrgStatusCtrl (Program Status Control) function block
SP_StepComplete-Judge_instance	OmronLib\ServoPress\ SP_StepCompleteJudge		Instance of the SP_StepCompleteJudge (Step Completion Determination) function block
SP_StepLoadAlarm_instance	OmronLib\ServoPress\ SP_StepLoadAlarm		Instance of the SP_StepLoadAlarm (Step Load Alarm Determination) function block
SP_PrgLoadAlarm_instance	OmronLib\ServoPress\ SP_PrgLoadAlarm		Instance of the SP_PrgLoadAlarm (Program Load Alarm Determination) function block
PrgTable	ARRAY[1..50] OF OmronLib\ ServoPress\sPRG_TABLE		Program table
PrgLoadAlarm	OmronLib\ServoPress\ sPRG_LOAD_ALARM		Program load alarm conditions
StartPg	BOOL		Starts single-axis program operation.
SingleAxisPrgOpr_Abt	BOOL		Abort trigger for single-axis program operation
SingleAxisPrgOpr_Start-StepNo	USINT		Start step number
SingleAxisCtrl_D	BOOL		Ends single-axis control execution.
SingleAxisCtrl_SingleCmd-ProfileNo	USINT		Single-axis command profile number
SingleAxisPrgOpr_Bsy	BOOL		Single-axis program operation busy
SingleAxisPrgOpr_Err	BOOL		Single-axis program operation fault
MCCmdExec	ARRAY[0..9] OF BOOL		Motion function block execution trigger
SingleCmdProfile	OmronLib\ServoPress\ sSINGLE_CMD_PROFILE		Single-axis command profile
StepCompleted	BOOL		Ends step.
StepExec	BOOL		Starts step.
CurrentStepNo	USINT		Current step number

### ● External Variables

Name	Data Type	Initial Value	Comment
MC_Axis000	_sAXIS_REF	---	Servo axis

## ● Algorithm

```

//Start trigger
inst2_R_TRIG( Clk := StartPg, Q => UpTrig_StartPg );

//Start single axis program operation after homing complete.
IF (MC_Axis000.Details.Homed=TRUE) THEN
  SingleAxisPrgOpr_Ex:=UpTrig_StartPg;
END_IF;

//Error stop
IF PrgStatusCtrl_Err OR StepCompleteJudge_Err OR StepLoadAlarm_Err OR PrgLoadAlarm_Err THEN
  SingleAxisPrgOpr_Err:=TRUE;
  ErrorStop:=TRUE;
ELSE
  SingleAxisPrgOpr_Err:=FALSE;
  ErrorStop:=FALSE;
END_IF;

//Abort each FBs
Abort_SingleAxisPrgOpr:=SingleAxisPrgOpr_Abt OR ErrorStop;

//SP_PrgOprCtrl
SP_PrgStatusCtrl_instance(
  PrgTable := PrgTable,
  Execute := SingleAxisPrgOpr_Ex,
  SingleStepMode := SingleAxisPrgOpr_SingleMode,
  StartStepNo := SingleAxisPrgOpr_StartStepNo,
  Abort := Abort_SingleAxisPrgOpr,
  StepCompleted := StepCompleted,
  NextStepNo := NextStepNo,
  Done => PrgStatusCtrl_Done,
  MCCmdExec => MCCmdExec,
  SingleCmdProfile => SingleCmdProfile,
  StepExec => StepExec,
  CurrentStepNo => CurrentStepNo,
  Busy => SingleAxisPrgOpr_Bsy,
  Error => PrgStatusCtrl_Err
);

//SP_StepCompleteJudge
SP_StepCompleteJudge_instance(
  PrgTable := PrgTable,
  Execute := StepExec,
  CurrentStepNo := CurrentStepNo,
  Abort := Abort_SingleAxisPrgOpr,
  Position := MC_Axis000.Act.Pos,
  Velocity := MC_Axis000.Act.Vel,
  Load := TorqueVal,
  MCCmdDone := SingleAxisCtrl_D,
  SingleCmdProfileNo := SingleAxisCtrl_SingleCmdProfileNo,
  Done => StepCompleted,
  NextStepNo => NextStepNo,
  Error => StepCompleteJudge_Err
);

//SP_StepLoadAlarm
SP_StepLoadAlarm_instance(
  PrgTable := PrgTable,
  Execute := StepExec,
  CurrentStepNo := CurrentStepNo,
  StepCompleted := StepCompleted,
  Abort := Abort_SingleAxisPrgOpr,
  Position := MC_Axis000.Act.Pos,

```

```

    Load := TorqueVal,
    Error => StepLoadAlarm_Err
);

//SP_PrgLoadAlarm
SP_PrgLoadAlarm_instance(
    PrgLoadAlarm := PrgLoadAlarm,
    Enable := SingleAxisPrgOpr_Bsy,
    Position := MC_Axis000.Act.Pos,
    Load := TorqueVal,
    Error => PrgLoadAlarm_Err
);

```

## ● How To Apply To the SP\_SingleAxisPrgOpr (Single-Axis Program Operation) Function Block Sample Programming

The following code shows the SP\_SingleAxisPrgOpr (Single-Axis Program Operation) function block instance code in the SP\_SingleAxisPrgOpr (Single-Axis Program Operation) function block sample programming.

```

//SP_SingleAxisPrgOpr
SINGLE_AXIS_PRG_OPR(
    PrgTable := PrgTable,
    PrgLoadAlarm := PrgLoadAlarm,
    Execute := SingleAxisPrgOpr_Ex,
    Abort := SingleAxisPrgOpr_Abt,
    StartStepNo := SingleAxisPrgOpr_StartStepNo,
    SingleMode := SingleAxisPrgOpr_SingleMode,
    Position := MC_Axis000.Act.Pos,
    Velocity := MC_Axis000.Act.Vel,
    Load := TorqueVal,
    MCCmdDone := SingleAxisCtrl_D,
    SingleCmdProfileNo := SingleAxisCtrl_SingleCmdProfileNo,
    Done => SingleAxisPrgOpr_D,
    MCCmdExec => MCCmdExec,
    SingleCmdProfile => SingleCmdProfile,
    CurrentStepNo => CurrentStepNo,
    StepLoadAlarmOut => StepLoadAlarmOut,
    PrgLoadAlarmOut => PrgLoadAlarmOut,
    ExtrOutputCode => ExtrOutputCode,
    StepExec => StepExec,
    StepCompleted => StepCompleted,
    Busy => SingleAxisPrgOpr_Bsy,
    CommandAborted=> SingleAxisPrgOpr_Ca,
    Error => SingleAxisPrgOpr_Err,
    ErrorID => SingleAxisPrgOpr_ErrID,
    ErrorIDEx => SingleAxisPrgOpr_ErrIDEx
);

```

The same operations can be executed in this sample programming as those in the SP\_SingleAxisPrgOpr (Single-axis Program Operation) function block sample programming by replacing the above code. The following part in this sample programming is included in the SP\_SingleAxisPrgOpr (Single-axis Program Operation) function block sample programming. If you make the replacement, use the part after the comment of *//Error stop*.

```

//Start trigger
inst2_R_TRIG( Clk := StartPg, Q => UpTrig_StartPg );

//Start single axis program operation after homing complete.
IF (MC_Axis000.Details.Homed=TRUE) THEN
    SingleAxisPrgOpr_Ex:=UpTrig_StartPg;
END_IF;

```

# SP\_StepCompleteJudge

The SP\_StepCompleteJudge function block performs step completion determination based on the step completion conditions.

Function block name	Name	FB/FUN	Graphic expression	ST expression
SP_StepCompleteJudge	Step Completion Determination	FB		<pre>SP_StepCompleteJudge_instance (   PrgTable,   Execute,   CurrentStepNo,   Abort,   Position,   Velocity,   Load,   MCCmdDone,   StepCompleteCode,   SingleCmdProfileNo,   Done,   TimeOut,   Unachieved,   NextStepNo,   MonitorEnabled,   Busy,   CommandAborted,   Error,   MonitorEnabled,   Busy,   CommandAborted,   Error,   ErrorID,   ErrorIDEx);</pre>

## Function Block and Function Information

Item	Description
Library file name	OmronLib_ServoPress_V1_0.slr OmronLib_ServoPress_V2_0.slr
Namespace	OmronLib\ServoPress
Function block and function number	00073
Publish/Do not publish source code	Do not publish
Function block and function version	1.00

## Variables

### Input Variables

Name	Meaning	Data type	Default	Valid range	Unit	Description
Execute	Execute	BOOL	FALSE	Depends on data type.	---	Execute trigger for this function block Executes the function block when it changes to TRUE.
Current StepNo	Current Step Number	USINT	1	1 to 50	---	Step number to evaluate for step completion
Abort	Abort	BOOL	FALSE	Depends on data type.	---	Abort trigger for this function block Aborts the function block when it changes to TRUE.
Position	Current Position	LREAL	0	Depends on data type.	Command units	The current position monitor input for the monitoring target.
Velocity	Current Velocity	LREAL	0	Depends on data type.	Command units/s	The current velocity monitor input for the monitoring target.
Load	Current Load	LREAL	0	Depends on data type.	Load units <sup>*1</sup>	The current load monitor input for the monitoring target.
MCCmd Done	Motion Instruction Completion	BOOL	FALSE	Depends on data type.	---	Motion instruction completion TRUE: Completed FALSE: Not completed.
Step Complete Code	Step Complete Code	USINT	0	0 to 255	---	Code to match to the step completion condition.
SingleCmd ProfileNo	Single-axis Command Profile Number	USINT	0	0 to 10	---	When you consecutively execute multiple single-axis motion control instructions, this number indicates the ordinal number of the current single-axis motion control instruction.

\*1. The unit of load is not specified in this function block. Uniquely set newtons (N) or % as a torque conversion value in the user program that uses this function block.

## Output Variables

Name	Meaning	Data type	Valid range	Unit	Description
Done	Done	BOOL	Depends on data type.	---	Completion of execution TRUE: Normal end FALSE: Error end, execution in progress, or execution condition not met
TimeOut	Timeout Occurrence	BOOL	Depends on data type.	---	Timeout occurrence TRUE: Timeout occurred. FALSE: Timeout did not occur.
Unachieved	Step Incompletion	BOOL	Depends on data type.	---	Whether the step has been completed or not <sup>*1</sup> TRUE: Step not completed. FALSE: Step completed.
NextStepNo	Next Step Number	USINT	0 to 50	---	Number of next step to execute <sup>*2,3</sup>
MonitorEnabled	Load Monitoring	BOOL	Depends on data type.	---	Load monitoring <sup>*4</sup> TRUE: Monitoring in progress. FALSE: Monitoring not in progress.
Busy	Executing	BOOL	Depends on data type.	---	Executing TRUE: Executing FALSE: Not executing
Command Aborted	Instruction Aborted	BOOL	Depends on data type.	---	Execution aborted This variable changes to TRUE if the function block is aborted.
Error	Error	BOOL	Depends on data type.	---	Error end TRUE: Error end FALSE: Normal end, executing, or execution conditions not met
ErrorID	Error Code	WORD	*5	---	An error code is output if an error occurs. The value is WORD#16#0 for a normal end.
ErrorIDEx	Expansion Error Code	DWORD	*5	---	An expansion error code is output if an error occurs. The value is DWORD#16#0 for a normal end.

\*1. *CompleteType* (Step Completion Type) is enabled in the following cases.

Target velocity (*CompleteType* = USINT3)

Equal to or greater than target load (*CompleteType* = USINT#4)

Equal to or less than target load (*CompleteType* = USINT#5)

Continuous detection of position load gradient (*CompleteType* = USINT7)

Continuous load decrease (*CompleteType* = USINT#8)

Continuous load increase (*CompleteType* = USINT#9)

\*2. If the value of *TimeOut* (Timeout Occurrence) is FALSE, the value of *NextStepNo* is 0.

\*3. If the value of *Unachieved* (Step Incompletion) is FALSE, the value of *NextStepNo* is 0.

- \*4. *CompleteType* (Step Completion Type) is enabled in the following cases.
- Equal to or greater than target load (*CompleteType* = USINT#4)
  - Equal to or less than target load (*CompleteType* = USINT#5)
  - Continuous detection of position load gradient (*CompleteType* = USINT7)
  - Continuous load decrease (*CompleteType* = USINT#8)
  - Continuous load increase (*CompleteType* = USINT#9)
- \*5. For details, refer to *Troubleshooting* on page 125.

## In-Out Variables

Name	Meaning	Data type	Valid range	Unit	Description
PrgTable	Program Tables	ARRAY[1..50] OF OmronLib\ ServoPress\ sPRG_TABLE	---	---	Program tables that contains settings for each step

### ● Structure

The data type of the PrgTable in-out variable is the structure OmronLib\ServoPress\sPRG\_TABLE. Refer to *Structure* in *SP\_SingleAxisPrgOpr* on page 42 for details.



## Function

This function block determines whether a step is completed based on the step completion conditions that are set in *PrgTable* (Program Tables).

This function block provides the single-axis program operation in combination with the SP\_PrgStatusCtrl (Program Status Control), SP\_StepLoadAlarm (Step Load Alarm Determination), and SP\_PrgLoadAlarm (Program Load Alarm Determination) function blocks. Refer to *Relation with Other Function Blocks* on page 102 in the description of the SP\_PrgStatusCtrl (Program Status Control) function block on relation with these function blocks.

## Step Completion Conditions

Specify the completion conditions for the steps with *PrgTable.StepCompleteData* (Step Completion Condition).

For details on step completion conditions, refer to *Step Completion Condition* on page 59 in the description of the SP\_SingleAxisPrgOpr (Single-axis Program Operation) function block.

## Meanings of Variables

The meanings of the other variables are described below.

- **CurrentStepNo (Executing Step Number)**

This variable gives the number of the current step. This step is evaluated for step completion.

- **Position (Current Position), Velocity (Current Velocity), and Load (Current Load)**

These variables are used to input the current position monitor, the current velocity monitor, and the current load monitor for the monitoring target.

- **MCCmdDone (Motion Instruction Completion)**

This variable indicates when execution of the current motion control instruction is completed. It is used when waiting for motion instruction completion (*CompleteType* = USINT#0) is set as the condition for completion of the step.

It is determined that the step is completed when the value of *MCCmdDone* (Motion Instruction Completion) is TRUE. For details on step completion conditions, refer to *Step Completion Condition* on page 59 in the description of the SP\_SingleAxisPrgOpr (Single-axis Program Operation) function block.

- **StepCompleteCode (Step Complete Code)**

This variable is used when step complete code match (*CompleteType* = USINT#6) is set as the condition for completion of the step. It is determined that the step is completed when the value of the *StepCompleteCode* (Step Complete Code) matches the value of *StepCompleteData.StepCompleteCode* (Step Complete Code Set Value). For details on step completion conditions, refer to *Step Completion Condition* on page 59 in the description of the SP\_SingleAxisPrgOpr (Single-axis Program Operation) function block.

- **SingleCmdProfileNo (Single-axis Command Profile Number)**

When you consecutively execute multiple single-axis motion control instructions, this number indicates the ordinal number of the current single-axis motion control instruction.

- **TimeOut (Timeout Occurrence)**

The variable indicates whether a timeout occurred in the step.

If the step is not completed even if the time set with *StepCompleteData.TimeOut* (Step Timeout Time) has elapsed from the start of the step, it will be determined that the step timed out. If a timeout occurs, the value of *TimeOut* (Timeout Occurrence) changes to TRUE, and execution of this function block is ended.

- **Unachieved (Step Incompletion)**

This variable indicates when a step is not completed when *CompleteType* (Step Completion Type) is target velocity (*CompleteType* = USINT3), equal to or greater than target load (*CompleteType* = USINT4), equal to or less than target load (*CompleteType* = USINT5), continuous detection of position load gradient (*CompleteType* = USINT7), continuous load decrease (*CompleteType* = USINT8), or continuous load increase (*CompleteType* = USINT9).

If the current position of the axis exceeds *MonitorMinPos* (Maximum Monitoring Position) but does not meet the step completion condition, it is determined that the step was not completed, and the value of *Unachieved* (Step Incompletion) changes to TRUE. If it is determined that the step was not completed, the execution of this function block is ended.

- **NextStepNo (Next Step Number)**

This is the number of the step to execute when a timeout occurs or a step is not completed. If the value of *TimeOut* (Timeout Occurrence) or the value of *Unachieved* (Step Incompletion) is FALSE, the value of *NextStepNo* (Next Step Number) changes to zero.

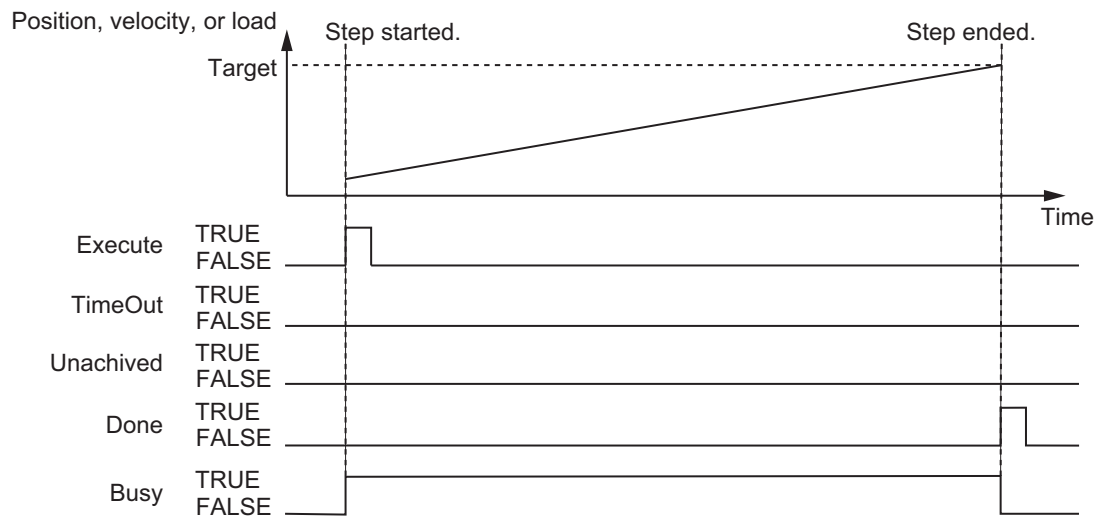
- **MonitorEnabled (Load Monitoring)**

This variable indicates whether load monitoring is in progress. When *CompleteType* (Step Completion Type) is target velocity (*CompleteType* = USINT3), equal to or greater than target load (*CompleteType* = USINT4), equal to or less than target load, (*CompleteType* = USINT5), continuous detection of position load gradient (*CompleteType* = USINT7), continuous load decrease (*CompleteType* = USINT8), or continuous load increase (*CompleteType* = USINT9), and the value of *Position* (Current Position) is between *MonitorMinPos* (Minimum Monitoring Position) and *MonitorMaxPos* (Maximum Monitoring Position), the value of *MonitorEnabled* (Load Monitoring) changes to TRUE.

## Timing Charts

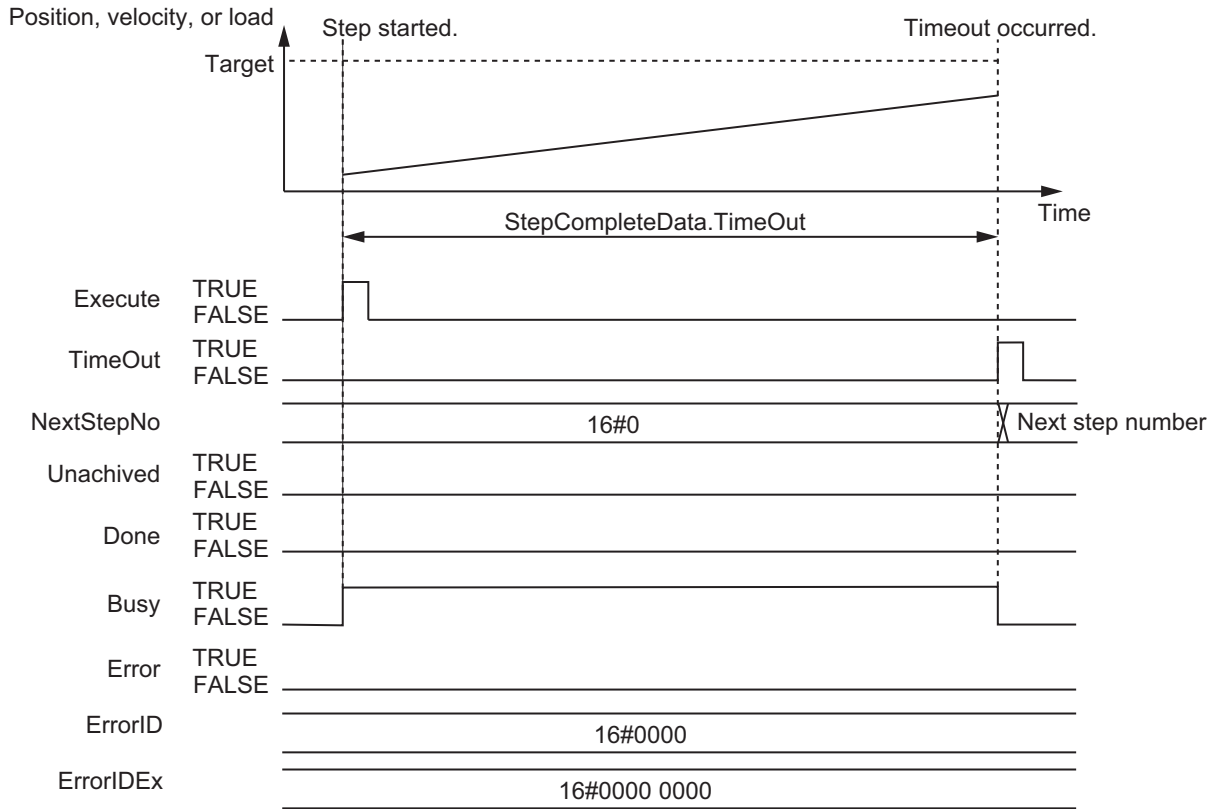
### ● Normal End

- *Busy* (Executing) changes to TRUE when *Execute* in the function block changes to TRUE.
- If the step completion condition is met, the value of *Done* changes to TRUE and execution of this function block is ended.



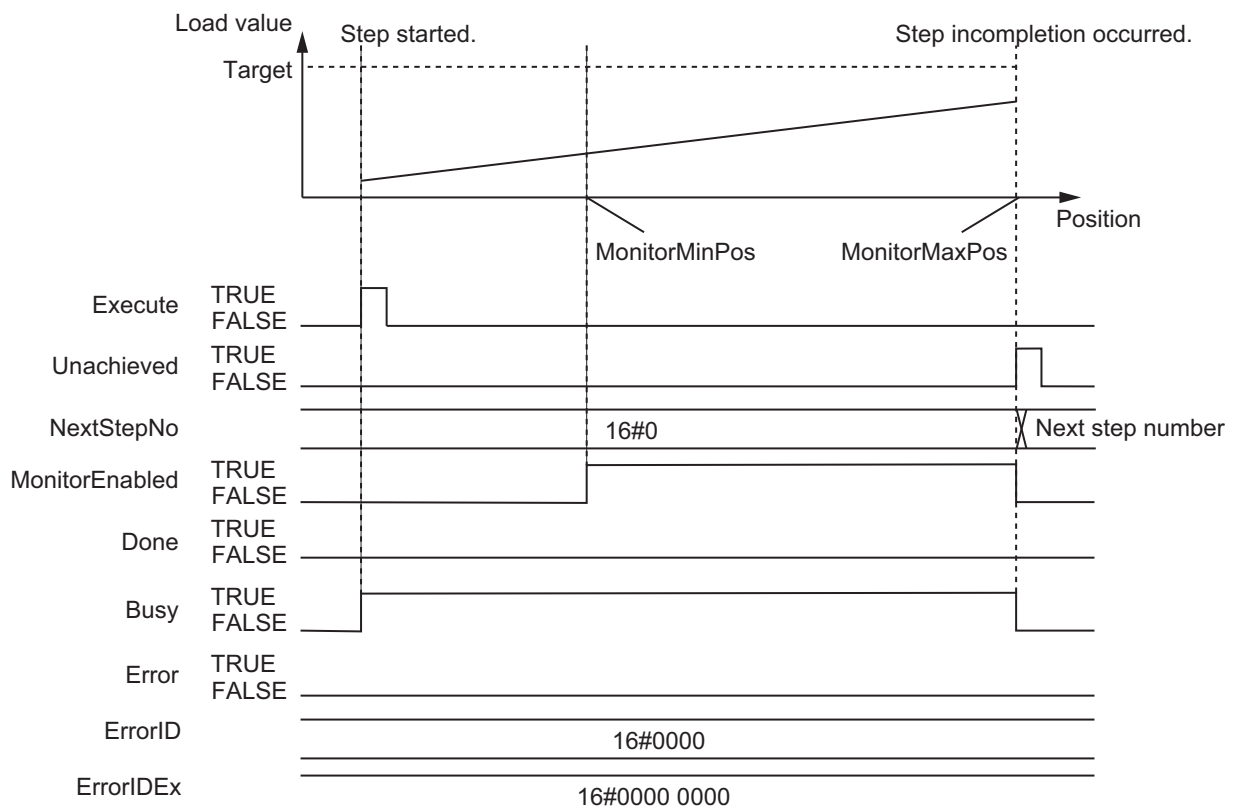
● **Timeout Occurrence**

- If a timeout occurs, the value of *TimeOut* (Timeout Occurrence) changes to TRUE and execution of this function block is ended.
- The value of *NextStepNo* (Next Step Number) is retained even after a timeout occurs until the value of *Execute* changes to TRUE.
- Even if a timeout occurs, an error will not occur.



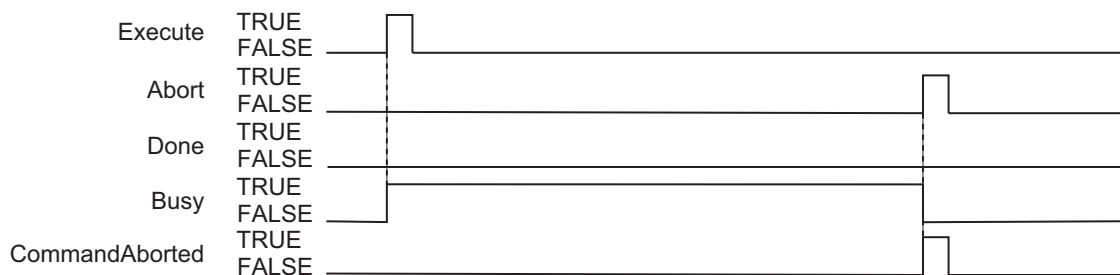
● **Step Incompletion**

- If the value of *Position* (Current Position) is between *MonitorMinPos* (Minimum Monitoring Position) and *MonitorMaxPos* (Maximum Monitoring Position), the value of *MonitorEnabled* (Load Monitoring) changes to TRUE.
- If the step is not completed, the value of *Unachieved* (Step Incompletion) changes to TRUE, and the execution of this function block is ended.
- If both a timeout and step incompletion occur in the same task period, step incompletion takes priority.
- The value of *NextStepNo* (Next Step Number) is retained even after step incompletion occurs until the value of *Execute* changes to TRUE.
- Even if step incompletion occurs, an error will not occur.



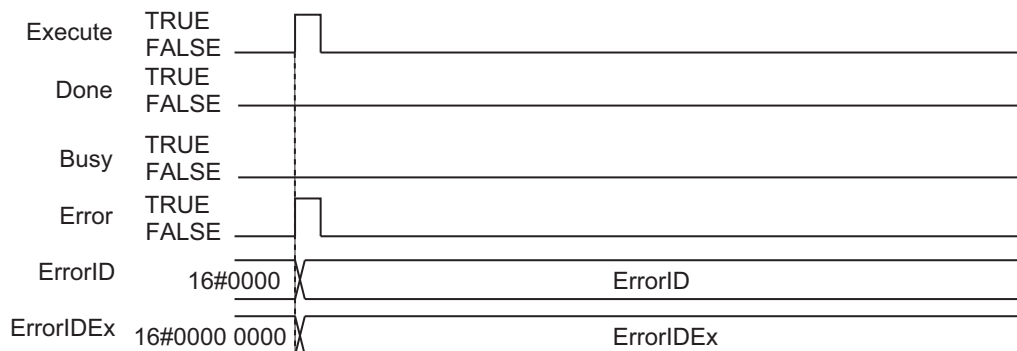
● **Aborting Execution**

- When the value of *Abort* is changed to TRUE, the processing is aborted.



● **Error End**

- If an error occurs during execution of this function block, *Error* changes to TRUE. You can find out the cause of the error by referring to the values output by *ErrorID* (Error Code) and *ErrorIDEx* (Expansion Error Code).
- The value of *Error* is retained while the value of *Execute* is TRUE.



**Precautions for Correct Use**

- The value of *CurrentStepNo* (Current Step Number) and the value of *StepCompleteData* (Step Completion Condition) are enabled when the value of *Execute* changes to TRUE. The value is not refreshed even if it is changed during processing of this function block.
- If the value of *Abort*, *Position* (Current Position), *Velocity* (Current Velocity), *Load* (Current Load), *StepCompleteCode* (Step Complete Code), or *SingleCmdProfileNo* (Single-axis Command Profile Number) is changed during execution of this function block, the value is updated by the processing within the same task period.

## Troubleshooting

The error codes, expansion error codes, status, descriptions, and corrections are given in the following table.

Error code	Expansion error code	Status	Description	Correction
16#0000	16#00000000	Normal end		
16#3C4B	16#□□□□0001 <sup>*1</sup>	Step timeout time out of range	The value of <i>TimeOut</i> (Step Timeout Time) is outside the valid range.	Check the valid range of the value for <i>TimeOut</i> (Step Timeout Time) and set the value within the valid range.
	16#□□□□0002 <sup>*1</sup>	Step timeout next step number out of range	The value of <i>TimeOutNextStepNo</i> (Step Timeout Next Step Number) is outside the valid range.	Check the valid range of the value for <i>TimeOutNextStepNo</i> (Step Timeout Next Step Number) and set the value within the valid range.
	16#□□□□0003 <sup>*1</sup>	Illegal monitoring position	The value of <i>MonitorMinPos</i> (Minimum Monitoring Position) is greater than the value of <i>MonitorMaxPos</i> (Maximum Monitoring Position).	Set <i>MonitorMinPos</i> (Minimum Monitoring Position) to a value that is less than or equal to the value of <i>MonitorMaxPos</i> (Maximum Monitoring Position).
	16#□□□□0004 <sup>*1</sup>	Monitoring start time out of range	The value of <i>MonitorStartTime</i> (Monitoring Start Time) is outside the valid range.	Check the valid range of the value for <i>MonitorStartTime</i> (Monitoring Start Time) and set the value within the valid range.
	16#□□□□0005 <sup>*1</sup>	Monitoring end time out of range	The value of <i>MonitorEndTime</i> (Monitoring End Time) is outside the valid range.	Check the valid range of the value for <i>MonitorEndTime</i> (Monitoring End Time) and set the value within the valid range.
	16#□□□□0006 <sup>*1</sup>	Illegal monitoring time	The value of <i>MonitorStartTime</i> (Monitoring Start Time) is greater than the value of <i>MonitorEndTime</i> (Monitoring End Time).	Set <i>MonitorStartTime</i> (Monitoring Start Time) to a value that is less than or equal to the value of <i>MonitorEndTime</i> (Monitoring End Time).

Error code	Expansion error code	Status	Description	Correction
16#3C4B	16#□□□□0007 <sup>*1</sup>	Number of consecutive position load gradients out of range	The value of <i>InflPointGradientCount</i> (Number of Consecutive Position Load Gradients) is outside the valid range.	Check the valid range for the value of <i>InflPointGradientCount</i> (Number of Consecutive Position Load Gradients) and set the value within the valid range.
	16#□□□□0008 <sup>*1</sup>	Number of consecutive load decreases threshold out of range	The value of <i>LoadDecreaseCount</i> (Number of Consecutive Load Decreases Threshold) is outside the valid range.	Check the valid range for the value of <i>LoadDecreaseCount</i> (Number of Consecutive Load Decreases Threshold) and set the value within the valid range.
	16#□□□□0009 <sup>*1</sup>	Number of consecutive load increases threshold out of range	The value of <i>LoadIncreaseCount</i> (Number of Consecutive Load Increases Threshold) is outside the valid range.	Check the valid range for the value of <i>LoadIncreaseCount</i> (Number of Consecutive Load Increases Threshold) and set the value within the valid range.
	16#□□□□000A <sup>*1</sup>	Step number when step not completed	The value of <i>UnachievedNextStepNo</i> (Step Number When Step Not Completed) is outside the valid range.	Check the valid range of the value for <i>UnachievedNextStepNo</i> (Step Number When Step Not Completed) and set the value within the valid range.
	16#□□□□000B <sup>*1</sup>	Wait time out of range	The value of <i>WaitTime</i> (Wait Time) is outside the valid range.	Check the valid range of the value for <i>WaitTime</i> (Wait Time) and set the value within the valid range.
	16#□□□□000C <sup>*1</sup>	Position load gradient out of range	The value of <i>InflPointGradient</i> (Position Load Gradient) is outside the valid range.	Check the valid range for the value of <i>InflPointGradient</i> (Position Load Gradient) and set the value within the valid range.
	16#□□□□000D <sup>*1</sup>	Step completion type out of range	The value of <i>CompleteType</i> (Step Completion Type) exceeded the valid range.	Check the valid range of the value for <i>CompleteType</i> (Step Completion Type) and set the value within the valid range.
	16#□□□□000E <sup>*1</sup>	Positioning in-position width out of range	The value of <i>InPosWidth</i> (Positioning In-position Width) is outside the valid range.	Check the valid range of the value for <i>InPosWidth</i> (Positioning In-position Width) and set the value within the valid range.
	16#0000000F	Incorrect task setting	An attempt was made to execute this function block in a task period which is not a primary task period or a periodic task.	Execute this function block in a primary periodic task or a periodic task.

\*1. The boxes (□□□□) are the BCD values of *CurrentStepNo* (Current Step Number) when the error occurred. For example, if *CurrentStepNo* is USINT#11, *ErrorIDEx* is DWORD#16#00110001.

## Sample Programming

Refer to *Sample Programming* on page 108 in the description of the SP\_PrgStatusCtrl (Program Status Control) function block.



# SP\_StepLoadAlarm

The SP\_StepLoadAlarm function block performs step load alarm determination based on the step load alarm conditions.

Function block name	Name	FB/FUN	Graphic expression	ST expression
SP_StepLoadAlarm	Step Load Alarm Determination	FB		<pre>SP_StepLoadAlarm_instance ( PrgTable, Execute, CurrentStepNo, StepCompleted, Abort, Position, Load, Done, StepLoadAlarmOut, MonitorEnabled, Busy, CommandAborted, Error, ErrorID, ErrorIDEx);</pre>

## Function Block and Function Information

Item	Description
Library file name	OmronLib_ServoPress_V1_0.slr OmronLib_ServoPress_V2_0.slr
Namespace	OmronLib\ServoPress
Function block and function number	00074
Publish/Do not publish source code	Do not publish
Function block and function version	1.00

## Variables

### Input Variables

Name	Meaning	Data type	Default	Valid range	Unit	Description
Execute	Execute	BOOL	FALSE	Depends on data type.	---	Execute trigger for this function block Executes the function block when it changes to TRUE.
CurrentStep No	Current Step Number	USINT	1	Depends on data type.	---	Step number to evaluate for step load alarm
Step Completed	Step Completed Trigger	BOOL	FALSE	Depends on data type.	---	The step is completed when the step completion trigger changes to TRUE.
Abort	Abort	BOOL	FALSE	Depends on data type.	---	Abort trigger for this function block Aborts the function block when it changes to TRUE.
Position	Current Position	LREAL	0	Depends on data type.	Command units for monitoring	The current position monitor input for the monitoring target.
Load	Current Load	LREAL	0	Depends on data type.	Load units <sup>*1</sup>	The current load monitor input for the monitoring target.

\*1. The unit of load is not specified in this function block. Uniquely set newtons (N) or % as a torque conversion value in the user program that uses this function block.

## Output Variables

Name	Meaning	Data type	Valid range	Unit	Description
Done	Done	BOOL	Depends on data type.	---	Completion of execution TRUE: Normal end FALSE: Error end, execution in progress, or execution condition not met
StepLoadAlarm Out	Step Load Alarm	BOOL	Depends on data type.	---	Step load alarm TRUE: A step load alarm occurred. FALSE: A step load alarm did not occur.
MonitorEnabled	Load Monitoring	BOOL	Depends on data type.	---	Load monitoring TRUE: Monitoring in progress. FALSE: Monitoring not in progress.
Busy	Executing	BOOL	Depends on data type.	---	Executing TRUE: Executing. FALSE: Not executing.
Command Aborted	Instruction Aborted	BOOL	Depends on data type.	---	Execution aborted This variable changes to TRUE if the function block is aborted.
Error	Error	BOOL	Depends on data type.	---	Error end TRUE: Error end FALSE: Normal end, executing, or execution conditions not met
ErrorID	Error Code	WORD	*1	---	An error code is output if an error occurs. The value is WORD#16#0 for a normal end.
ErrorIDEx	Expansion Error Code	DWORD	*1	---	An expansion error code is output if an error occurs. The value is DWORD#16#0 for a normal end.

\*1. For details, refer to *Troubleshooting* on page 134.

## In-Out Variables

Name	Meaning	Data type	Valid range	Unit	Description
PrgTable	Program Tables	ARRAY[1..50] OF OmronLib\ ServoPress\ sPRG_TABLE	---	---	Program tables that contain settings for each step.

### ● Structure

The data type of the *PrgTable* in-out variable is the structure OmronLib\ServoPress\sPRG\_TABLE. Refer to *Structure* in *SP\_SingleAxisPrgOpr* on page 42 for details.

## Function

This function performs step load alarm determination based on the step load alarm conditions that are set in *PrgTable* (Program Tables).

This function block provides the single-axis program operation in combination with the SP\_PrgStatusCtrl (Program Status Control), SP\_StepCompleteJudge (Step Completion Determination), and SP\_PrgLoadAlarm (Program Load Alarm Determination) function blocks. Refer to *Relation with Other Function Blocks* on page 102 in the description of the P\_PrgStatusCtrl (Program Status Control) function block on relation with these function blocks.

## Step Load Alarm Conditions

Step load alarm conditions are set for each step with *PrgTable.StepLoadAlarm* (Step Load Alarm Conditions).

For details on step load alarm conditions, refer to *Step Load Alarm Determination* on page 68 in the description of the SP\_SingleAxisPrgOpr (Single-axis Program Operation) function block.

## Meanings of Variables

The meanings of the other variables are described below.

- **CurrentStepNo (Current Step Number)**

This variable gives the number of the current step. Step load alarm determination is performed for this step.

- **StepCompleted (Step Completed Trigger)**

This variable is a trigger signal that indicates when a step is completed. If the value of *StepCompleted* (Step Completed Trigger) changes to TRUE, the step is recognized as completed and the function block moves to the next step.

- **Position (Current Position) and Load (Current Load)**

These variables are used to input the current position monitor and the current load monitor for the monitoring target.

- **StepLoadAlarmOut (Step Load Alarm)**

This variable indicates whether a step load alarm occurred in the step. When a step load alarm occurs, the value of *StepLoadAlarmOut* (Step Load Alarm) changes to TRUE.

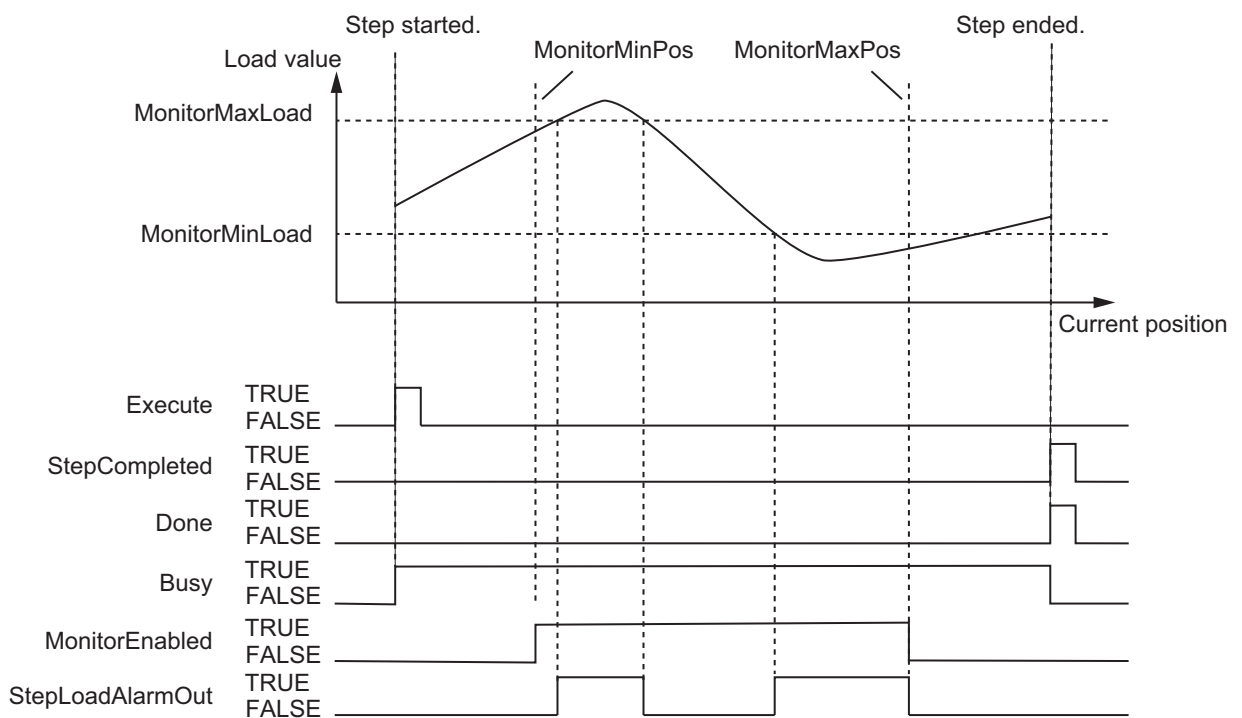
- **MonitorEnabled (Load Monitoring)**

This variable indicates whether load monitoring is in progress. If the value of *Position* (Current Position) is between *MonitorMinPos* (Minimum Monitoring Position) and *MonitorMaxPos* (Maximum Monitoring Position), the value of *MonitorEnabled* (Load Monitoring) changes to TRUE.

## Timing Charts

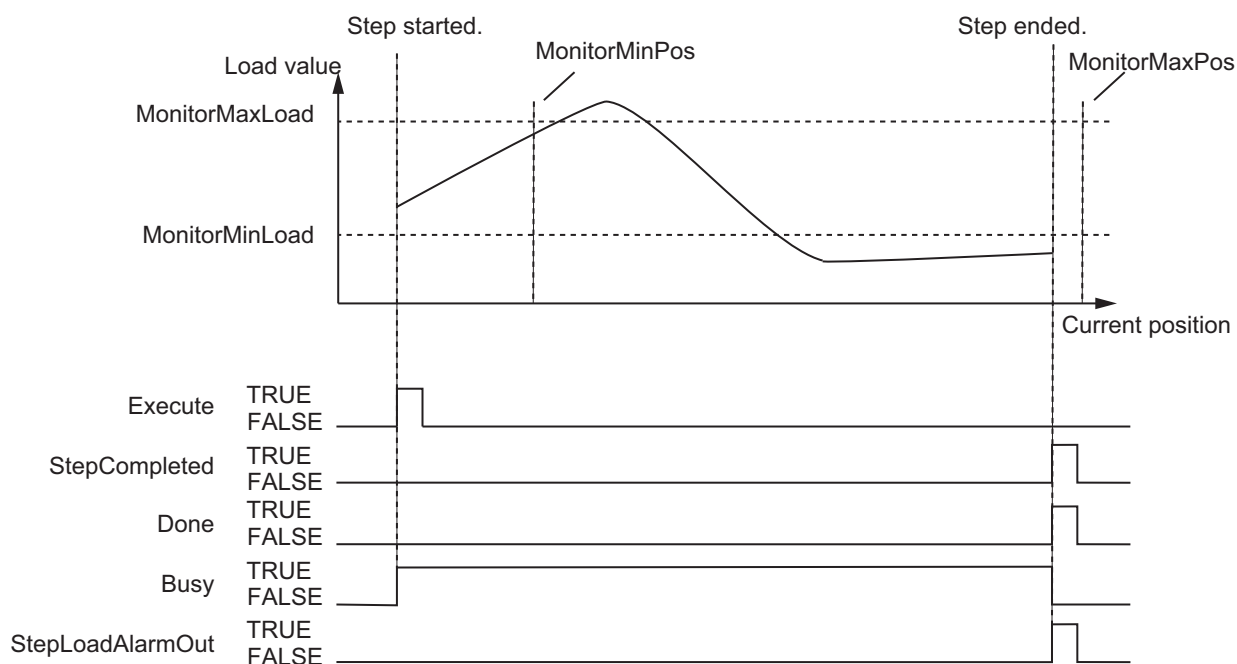
### ● Normal End for Determination While Step Is in Progress

- *Busy* (Executing) changes to TRUE when *Execute* in the function block changes to TRUE.
- The value of *StepLoadAlarmOut* (Step Load Alarm) changes to TRUE if the step load alarm condition is met.
- If the value of *Position* (Current Position) is between *MonitorMinPos* (Minimum Monitoring Position) and *MonitorMaxPos* (Maximum Monitoring Position), the value of *MonitorEnabled* (Load Monitoring) changes to TRUE.
- Step load alarm determination ends when the value of *StepCompleted* (Step Completed Trigger) changes to TRUE. The value of *StepLoadAlarmOut* (Step Load Alarm) changes to FALSE.



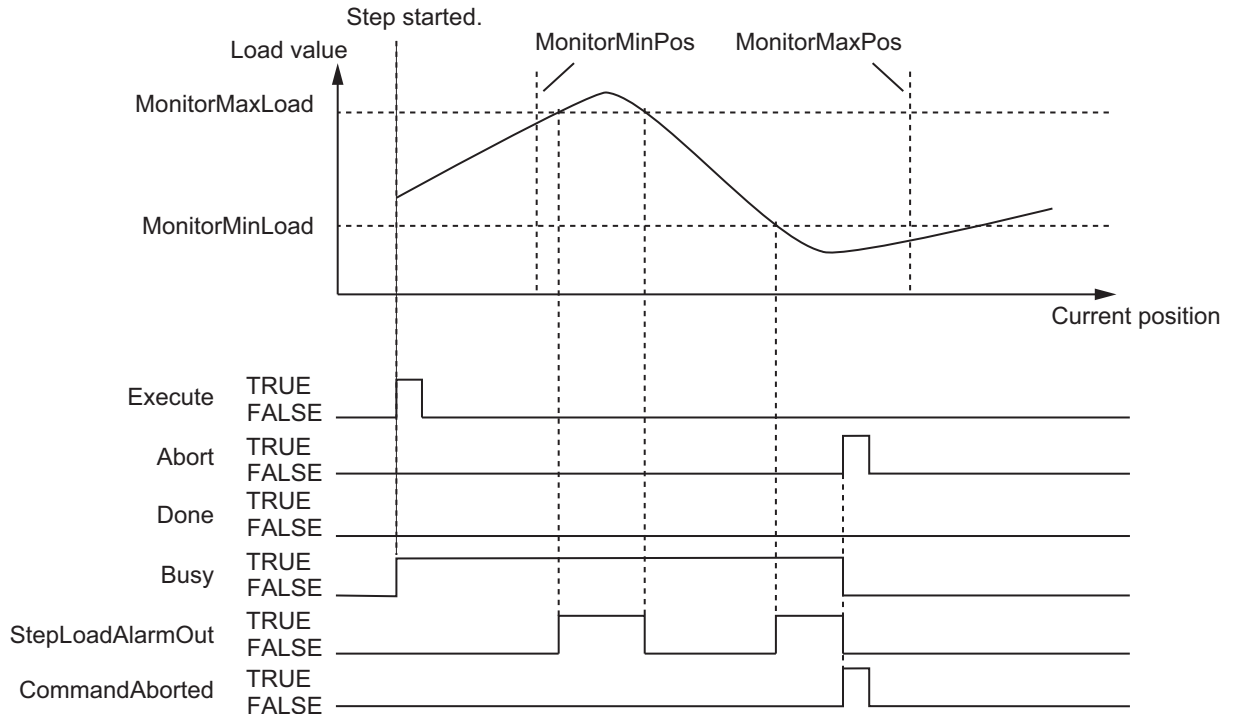
● Normal End for Determination When Step Is Completed

- *Busy* (Executing) changes to TRUE when *Execute* in the function block changes to TRUE.
- Step load alarm determination is performed when the value of *StepCompleted* (Step Completed Trigger) changes to TRUE. The value of *StepLoadAlarmOut* (Step Load Alarm) changes to TRUE if the step load alarm condition is met, but for only one task period.
- The value of *StepLoadAlarmOut* (Step Load Alarm) changes to FALSE while the value of *StepCompleted* (Step Completed Trigger) is FALSE.



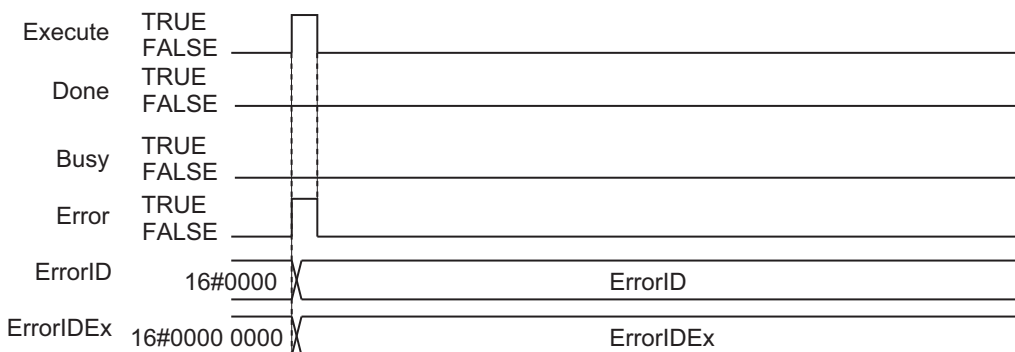
● **Aborting Execution**

- When the value of *Abort* is changed to TRUE, the processing is aborted.
- The value of *StepLoadAlarmOut* (Step Load Alarm) changes to FALSE if execution is aborted.



● **Error End**

- If an error occurs during execution of this function block, *Error* changes to TRUE. You can find out the cause of the error by referring to the values output by *ErrorID* (Error Code) and *ErrorIDEx* (Expansion Error Code).
- The value of *Error* is retained while the value of *Execute* is TRUE.



**Precautions for Correct Use**

If the value of *CurrentStepNo* (Current Step Number), *StepCompleted* (Step Completed Trigger), *Abort*, *Position* (Current Position), or *Load* (Current Load) is changed during execution of this function block, the value is updated for the processing within the same task period.

## Troubleshooting

The error codes, expansion error codes, status, descriptions, and corrections are given in the following table.

Error code	Expansion error code	Status	Description	Correction
16#0000	16#00000000	Normal end		
16#3C4C	16#□□□□0001 <sup>*1</sup>	Illegal monitoring position	The value of <i>MonitorMinPos</i> (Monitoring Range Minimum Position) is greater than the value of <i>MonitorMaxPos</i> (Monitoring Range Maximum Position).	Set <i>MonitorMinPos</i> (Monitoring Range Minimum Position) to a value that is less than or equal to the value of <i>MonitorMaxPos</i> (Monitoring Range Maximum Position).
	16#□□□□0002 <sup>*1</sup>	Illegal monitoring load	The value of <i>MonitorMinLoad</i> (Monitoring Load Range Lower Limit) is greater than the value of <i>MonitorMaxLoad</i> (Monitoring Load Range Upper Limit).	Set <i>MonitorMinLoad</i> (Monitoring Load Range Lower Limit) to a value that is less than or equal to the value of <i>MonitorMaxLoad</i> (Monitoring Load Range Upper Limit).
	16#□□□□0003 <sup>*1</sup>	Step load alarm determination type out of range	The value of <i>StepAlarmType</i> (Step Load Alarm Determination Type) is outside the valid range.	Check the valid range of the value for <i>StepAlarmType</i> (Step Load Alarm Determination Type) and set the value within the valid range.
	16#□□□□0004 <sup>*1</sup>	Upper/lower monitoring limit in-position width out of range	The value of <i>InPosWidth</i> (Upper/Lower Monitoring Limit In-position Width) is outside the valid range.	Check the valid range of the value for <i>InPosWidth</i> (Upper/Lower Monitoring Limit In-position Width) and set the value within the valid range.
	16#00000005	Incorrect task setting	An attempt was made to execute this function block in a task period which is not a primary task period or a periodic task.	Execute this function block in a primary periodic task or a periodic task.

\*1. The boxes (□□□□) are the BCD values of *CurrentStepNo* (Current Step Number) when the error occurred. For example, if *CurrentStepNo* is USINT#11, *ErrorIDEx* is DWORD#16#00110001.

## Sample Programming

Refer to *Sample Programming* on page 108 in the description of the SP\_PrgStatusCtrl (Program Status Control) function block.



# SP\_PrgLoadAlarm

The SP\_PrgLoadAlarm function block performs program load alarm determination based on the program load alarm conditions.

Function block name	Name	FB/FUN	Graphic expression	ST expression
SP_PrgLoadAlarm	Program Load Alarm Determination	FB		<pre>SP_PrgLoadAlarm_instance (   PrgLoadAlarm,   Enable,   Position,   Load,   Enabled,   PrgLoadAlarmOut,   MonitorEnabled,   Busy,   Error,   ErrorID,   ErrorIDEx);</pre>

## Function Block and Function Information

Item	Description
Library file name	OmronLib_ServoPress_V1_0.slr OmronLib_ServoPress_V2_0.slr
Namespace	OmronLib\ServoPress
Function block and function number	00075
Publish/Do not publish source code	Do not publish
Function block and function version	1.00

## Variables

### Input Variables

Name	Meaning	Data type	Default	Valid range	Unit	Description
Enable	Enable	BOOL	FALSE	Depends on data type.	---	Enable TRUE: Enable. FALSE: Do not enable execution.
Position	Current Position	LREAL	0	Depends on data type.	Command units for monitoring	The current position monitor input for the monitoring target.
Load	Current Load	LREAL	0	Depends on data type.	Load units*1	The current load monitor input for the monitoring target.

\*1. The unit of load is not specified in this function block. Uniquely set newtons (N) or % as a torque conversion value in the user program that uses this function block.

### Output Variables

Name	Meaning	Data type	Valid range	Unit	Description
Enabled	Enabled	BOOL	Depends on data type.	---	Enabled TRUE: Enabled. FALSE: Not enabled.
PrgLoadAlarm Out	Program Load Alarm	BOOL	Depends on data type.	---	Program load alarm TRUE: A program load alarm occurred. FALSE: A program load alarm did not occur.
MonitorEnabled	Load Monitoring	BOOL	Depends on data type.	---	Load monitoring TRUE: Monitoring in progress. FALSE: Monitoring not in progress.
Busy	Executing	BOOL	Depends on data type.	---	Executing TRUE: Executing. FALSE: Not executing.
Error	Error	BOOL	Depends on data type.	---	Error end TRUE: Error end FALSE: Normal end, executing, or execution conditions not met
ErrorID	Error Code	WORD	*1	---	An error code is output if an error occurs. The value is WORD#16#0 for a normal end.
ErrorIDEx	Expansion Error Code	DWORD	*1	---	An expansion error code is output if an error occurs. The value is DWORD#16#0 for a normal end.

\*1. For details, refer to *Troubleshooting* on page 140.

## In-Out Variables

Name	Meaning	Data type	Valid range	Unit	Description
PrgLoadAlarm	Program Load Alarm Conditions	OmronLib\ ServoPress\ sPRG_LOAD_ ALARM	---	---	Conditions necessary for program load alarm to occur

### ● Structure

The data type of the *PrgLoadAlarm* in-out variable is the structure OmronLib\ServoPress\sPRG\_LOAD\_ALARM.

Refer to *Structure* in *SP\_SingleAxisPrgOpr* on page 42 for details.

## Function

This function block performs program load alarm determination based on *PrgLoadAlarm* (Program Load Alarm Conditions).

This function block provides the single-axis program operation in combination with the SP\_PrgStatusCtrl (Program Status Control), SP\_StepCompleteJudge (Step Completion Determination), and SP\_PrgLoadAlarm (Program Load Alarm Determination) function blocks. Refer to *Relation with Other Function Blocks* on page 102 in the description of the SP\_PrgStatusCtrl (Program Status Control) function block on relation with these function blocks.

### Program Load Alarm Conditions

Set the program load alarm conditions with *PrgLoadAlarm* (Program Load Alarm Conditions).

For details on program load alarm conditions, refer to *Program Load Alarm Determination* on page 70 in the description of the SP\_SingleAxisPrgOpr (Single-axis Program Operation) function block.

### Meanings of Variables

The meanings of the other variables are described below.

- ***Position* (Current Position) and *Load* (Current Load)**

These variables are used to input the current position monitor and the current load monitor for the monitoring target.

- ***PrgLoadAlarmOut* (Program Load Alarm)**

This variable indicates whether a program load alarm occurred. If a program load alarm occurs, the *PrgLoadAlarmOut* (Program Load Alarm) changes to TRUE.

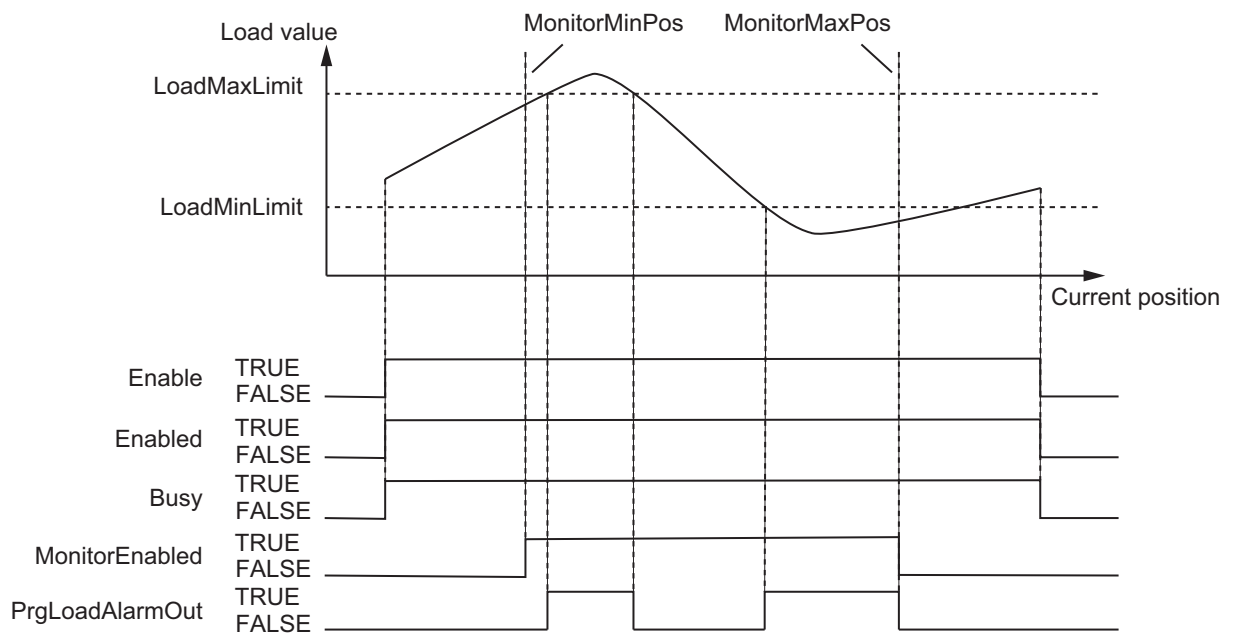
- ***MonitorEnabled* (Load Monitoring)**

This variable indicates whether load monitoring is in progress. If the value of *Position* (Current Position) is between *MonitorMinPos* (Minimum Monitoring Position) and *MonitorMaxPos* (Maximum Monitoring Position), the value of *MonitorEnabled* (Load Monitoring) changes to TRUE.

## Timing Charts

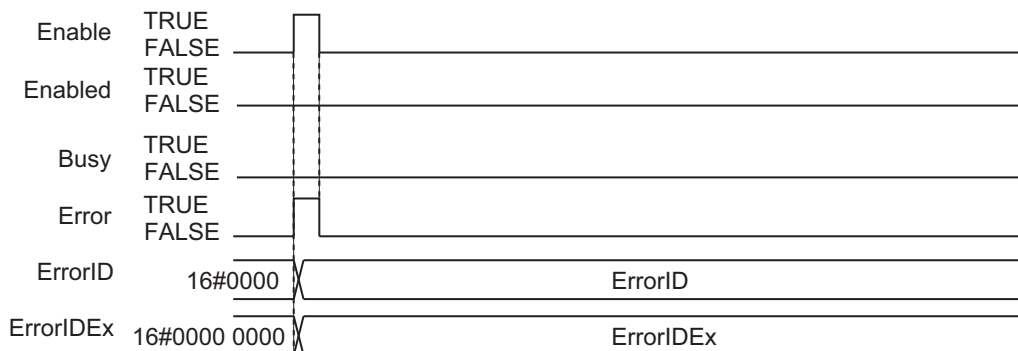
### ● Normal End

- *Busy* (Executing) changes to TRUE when *Enable* in the function block changes to TRUE.
- If the program load alarm condition is met, the value of *PrgLoadAlarmOut* (Program Load Alarm) changes to TRUE.
- If the program load alarm condition is not met, the value of *PrgLoadAlarmOut* (Program Load Alarm) changes to FALSE.
- When the value of *Enable* changes to FALSE, the value of *PrgLoadAlarmOut* (Program Load Alarm) changes to FALSE.



### ● Error End

- If an error occurs during execution of this function block, *Error* changes to TRUE. You can find out the cause of the error by referring to the values output by *ErrorID* (Error Code) and *ErrorIDEx* (Expansion Error Code).
- The value of *Error* is retained while the value of *Enable* is TRUE.



## Precautions for Correct Use

If the value of *Position* (Current Position) or *Load* (Current Load) is changed during execution of this function block, the value is updated for the processing within the same task period.

## Troubleshooting

The error codes, expansion error codes, status, descriptions, and corrections are given in the following table.

Error code	Expansion error code	Status	Description	Correction
16#0000	16#00000000	Normal end		
16#3C4D	16#00000001	Program load alarm determination type out of range	The value of <i>PrgAlarmType</i> (Program Load Alarm Determination Type) is outside the valid range.	Check the valid range of the value for <i>PrgAlarmType</i> (Program Load Alarm Determination Type) and set the value within the valid range.
	16#00000002	Standard type out of range	The value of <i>UnitType</i> (Standard Type) is outside of the valid range.	Check the valid range of the value for <i>UnitType</i> (Standard Type) and set the value within the valid range.
	16#00000003	Trapezoid area data alarm type out of range	The value of <i>TrapezoidData.AlarmType</i> (Alarm Type) is outside of the valid range.	Check the valid range of the value for <i>TrapezoidData.AlarmType</i> (Alarm Type) and set the value within the valid range.
	16#00000004	Trapezoid area data monitoring lower limit elapsed time out of range	The value of <i>TrapezoidData.MonitorLowerTime</i> (Monitoring Lower Limit Elapsed Time) is outside of the valid range.	Check the valid range of the value for <i>TrapezoidData.MonitorLowerTime</i> (Monitoring Lower Limit Elapsed Time) and set the value within the valid range.
	16#00000005	Trapezoid area data monitoring upper limit elapsed time out of range	The value of <i>TrapezoidData.MonitorUpperTime</i> (Monitoring Upper Limit Elapsed Time) is outside of the valid range.	Check the valid range of the value for <i>TrapezoidData.MonitorUpperTime</i> (Monitoring Upper Limit Elapsed Time) and set the value within the valid range.
	16#00000006	Illegal trapezoid area data monitoring lower limit elapsed time	The value of <i>TrapezoidData.MonitorLowerTime</i> (Monitoring Lower Limit Elapsed Time) is greater than the value of <i>TrapezoidData.MonitorUpperTime</i> (Monitoring Upper Limit Elapsed Time).	Set <i>TrapezoidData.MonitorLowerTime</i> (Monitoring Lower Limit Elapsed Time) to a value that is less than or equal to the value of <i>TrapezoidData.MonitorUpperTime</i> (Monitoring Upper Limit Elapsed Time).
	16#00000007	Illegal trapezoid area data monitoring position	The value of <i>TrapezoidData.MonitorLowerPos</i> (Minimum Monitoring Position) is greater than the value of <i>TrapezoidData.MonitorUpperPos</i> (Maximum Monitoring Position).	Set <i>TrapezoidData.MonitorLowerPos</i> (Minimum Monitoring Position) to a value that is less than or equal to the value of <i>TrapezoidData.MonitorUpperPos</i> (Maximum Monitoring Position).

Error code	Expansion error code	Status	Description	Correction
16#3C4D	16#00000008	Illegal trapezoid area data monitoring start point load	The value of <i>TrapezoidData.LoadMinLower</i> (Monitoring Start Point Minimum Load) is greater than the value of <i>TrapezoidData.LoadMaxLower</i> (Monitoring Start Point Maximum Load).	Set the value of <i>TrapezoidData.LoadMinLower</i> (Monitoring Start Point Minimum Load) to a value that is less than or equal to the value of <i>TrapezoidData.LoadMaxLower</i> (Monitoring Start Point Maximum Load).
	16#00000009	Illegal trapezoid area data monitoring end point load	The value of <i>TrapezoidData.LoadMinUpper</i> (Monitoring End Point Minimum Load) is greater than the value of <i>TrapezoidData.LoadMaxUpper</i> (Monitoring End Point Maximum Load).	Set the value of <i>TrapezoidData.LoadMinUpper</i> (Monitoring End Point Minimum Load) to a value that is less than or equal to the value of <i>TrapezoidData.LoadMaxUpper</i> (Monitoring End Point Maximum Load).
	16#0000000A	Rectangle area data alarm type out of range	The value of <i>RectangleData.AlarmType</i> (Alarm Type) is outside of the valid range.	Check the valid range of the value for <i>RectangleData.AlarmType</i> (Alarm Type) and set the value within the valid range.
	16#0000000B	Rectangle area data monitoring lower limit elapsed time out of range	The value of <i>RectangleData.MonitorLowerTime</i> (Monitoring Lower Limit Elapsed Time) is outside of the valid range.	Check the valid range of the value for <i>RectangleData.MonitorLowerTime</i> (Monitoring Lower Limit Elapsed Time) and set the value within the valid range.
	16#0000000C	Rectangle area data monitoring upper limit elapsed time out of range	The value of <i>RectangleData.MonitorUpperTime</i> (Monitoring Upper Limit Elapsed Time) is outside of the valid range.	Check the valid range of the value for <i>RectangleData.MonitorUpperTime</i> (Monitoring Upper Limit Elapsed Time) and set the value within the valid range.
	16#0000000D	Illegal rectangle area data monitoring lower limit elapsed time	The value of <i>RectangleData.MonitorLowerTime</i> (Monitoring Lower Limit Elapsed Time) is greater than the value of <i>RectangleData.MonitorUpperTime</i> (Monitoring Upper Limit Elapsed Time).	Set <i>RectangleData.MonitorLowerTime</i> (Monitoring Lower Limit Elapsed Time) to a value that is less than or equal to the value of <i>RectangleData.MonitorUpperTime</i> (Monitoring Upper Limit Elapsed Time).
	16#0000000E	Illegal rectangle area data monitoring position	The value of <i>RectangleData.MonitorLowerPos</i> (Minimum Monitoring Position) is greater than the value of <i>RectangleData.MonitorUpperPos</i> (Maximum Monitoring Position).	Set <i>RectangleData.MonitorLowerPos</i> (Minimum Monitoring Position) to a value that is less than or equal to the value of <i>RectangleData.MonitorUpperPos</i> (Maximum Monitoring Position).

Error code	Expansion error code	Status	Description	Correction
16#3C4D	16#0000000F	Illegal rectangle area data load	The value of <i>RectangleData.LoadMinLimit</i> (Load Lower Limit) is greater than the value of <i>RectangleData.LoadMaxLimit</i> (Load Upper Limit).	Set the value of <i>RectangleData.LoadMinLimit</i> (Load Lower Limit) to a value that is less than or equal to the value of <i>RectangleData.LoadMaxLimit</i> (Load Upper Limit).
	16#00000010	Illegal trapezoid area data	The positional relationships between the values of <i>TrapezoidData.MonitorLowerTime</i> (Monitoring Lower Limit Elapsed Time), <i>TrapezoidData.MonitorUpperTime</i> (Monitoring Upper Limit Elapsed Time), <i>TrapezoidData.MonitorLowerPos</i> (Minimum Monitoring Position), <i>TrapezoidData.MonitorUpperPos</i> (Maximum Monitoring Position), <i>TrapezoidData.LoadMinLower</i> (Monitoring Start Point Minimum Load), <i>TrapezoidData.LoadMaxLower</i> (Monitoring Start Point Maximum Load), <i>TrapezoidData.LoadMinUpper</i> (Monitoring End Point Minimum Load), and <i>TrapezoidData.LoadMaxUpper</i> (Monitoring End Point Maximum Load) do not form a trapezoid.	Set the values of <i>TrapezoidData.MonitorLowerTime</i> (Monitoring Lower Limit Elapsed Time), <i>TrapezoidData.MonitorUpperTime</i> (Monitoring Upper Limit Elapsed Time), <i>TrapezoidData.MonitorLowerPos</i> (Minimum Monitoring Position), <i>TrapezoidData.MonitorUpperPos</i> (Maximum Monitoring Position), <i>TrapezoidData.LoadMinLower</i> (Monitoring Start Point Minimum Load), <i>TrapezoidData.LoadMaxLower</i> (Monitoring Start Point Maximum Load), <i>TrapezoidData.LoadMinUpper</i> (Monitoring End Point Minimum Load), and <i>TrapezoidData.LoadMaxUpper</i> (Monitoring End Point Maximum Load) so that the enclosed area is a trapezoid.
	16#00000011	Upper/lower monitoring limit in-position width out of range	The value of <i>InPosWidth</i> (Upper/Lower Monitoring Limit In-position Width) is outside the valid range.	Check the valid range of the value for <i>InPosWidth</i> (Upper/Lower Monitoring Limit In-position Width) and set the value within the valid range.
	16#00000012	Incorrect task setting	An attempt was made to execute this function block in a task period which is not a primary task period or a periodic task.	Execute this function block in a primary periodic task or a periodic task.

## Sample Programming

Refer to *Sample Programming* on page 108 in the description of the SP\_PrgStatusCtrl (Program Status Control) function block.



# SingleAxisCtrl

The SingleAxisCtrl function block executes position control, velocity control, torque control, and torque feedback control.

Function block name	Name	FB/FUN	Graphic expression	ST expression
SingleAxisCtrl	Single-axis Control	FB		<pre>SingleAxisCtrl_instance (   Axis,   SingleCmdProfile,   Execute,   MeasuringTorque,   VelOverrideEnable,   InputVelFactor,   ItgReset,   BufferMode,   Done,   TargetReached,   SingleCmdProfileNo,   TorqueLimitParam,   OverrideEnabled,   Busy,   Active,   CommandAborted,   Error,   ErrorID,   ErrorIDEx);</pre>

## Function Block and Function Information

Item	Description
Library file name	OmronLib_ServoPress_V1_0.slr OmronLib_ServoPress_V2_0.slr
Namespace	OmronLib\ServoPress
Function block and function number	00076
Publish/Do not publish source code	Do not publish
Function block and function version	1.00

## Variables

### Input Variables

Name	Meaning	Data type	Default	Valid range	Unit	Description
Execute	Execute	BOOL	FALSE	Depends on data type.	---	Execute trigger for this function block Executes the function block when it changes to TRUE.
Measuring Torque	Measured Torque	LREAL	0	-1000.0 to 1000.0 <sup>*1*2</sup>	% <sup>*3</sup>	Measured actual torque
VelOverride Enable	Velocity Override Enable	BOOL	FALSE	Depends on data type.	---	Whether to enable or disable the velocity override TRUE: Enabled FALSE: Disabled.
InputVel Factor	Velocity Override Input Value	LREAL	100	0 or 0.01 to 500 <sup>*4</sup>	%	Velocity override input value
ItgReset	Integral Processing Reset	BOOL	FALSE	Depends on data type.	---	Integral processing reset TRUE: Resets the processing. FALSE: Does not reset the processing.
BufferMode	Buffer Mode Selection	_eM-C_BUFFER_MODE	_mcAborting	_mcAborting	---	Operation for multi-execution of motion control instructions _mcAborting: Aborting

\*1. A setting less than LREAL#-1000.0 is treated as LREAL#-1000.0. A setting greater than LREAL#1000.0 is treated as LREAL#1000.0.

\*2. The value is rounded to the second decimal place.

\*3. Set the percentage of the rated torque.

\*4. Values of 500 and higher are treated as 500. Values of less than 0.01 are treated as 0.01. However, 0 is treated as 0.

### Output Variables

Name	Meaning	Data type	Valid range	Unit	Description
Done	Done	BOOL	Depends on data type.	---	Completion of execution TRUE: Normal end FALSE: Error end, execution in progress, or execution condition not met
TargetReached	Target Reached	BOOL	Depends on data type.	---	Whether or not the target was reached TRUE: The target velocity was reached. FALSE: The target velocity was not reached.

Name	Meaning	Data type	Valid range	Unit	Description
SingleCmd ProfileNo	Single-axis Command Profile Num- ber	USINT	0 to 10	---	When you consecutively execute multiple single-axis motion control instructions, this number indicates the ordinal number of the current single-axis motion control instruction.
TorqueLimit Param	Torque Limit Settings	OmronLib\ ServoPress\ sTORQUE_ LIMIT_PARAM	---	---	Torque limit settings
OverrideEnabled	Override Enabled	BOOL	Depends on data type.	---	Whether the velocity override is enabled TRUE: Enabled. FALSE: Disabled.
Busy	Executing	BOOL	Depends on data type.	---	Executing TRUE: Executing FALSE: Not executing
Active	Controlling	BOOL	Depends on data type.	---	Controlling TRUE: Controlling FALSE: Not controlling
Command Aborted	Instruction Aborted	BOOL	Depends on data type.	---	Execution aborted This variable changes to TRUE if the function block is aborted.
Error	Error	BOOL	Depends on data type.	---	Error end TRUE: Error end FALSE: Normal end, executing, or execution conditions not met
ErrorID	Error Code	WORD	*1	---	An error code is output if an error occurs. The value is WORD#16#0 for a normal end.
ErrorIDEx	Expansion Error Code	DWORD	*1	---	An expansion error code is output if an error occurs. The value is DWORD#16#0 for a normal end.

\*1. For details, refer to *Troubleshooting* on page 169.

## ● Structure

The data type of the *TorqueLimitParam* output variable is the structure `OmronLib\ServoPress\sTORQUE_LIMIT_PARAM`. The specifications are as follows:

Name	Meaning	Description	Data type	Valid range	Unit
TorqueLimitParam	Torque Limit Settings	Torque limit settings	OmronLib\ServoPress\sTORQUE_LIMIT_PARAM	---	---
TorqueLimitPositiveEnable	Enable Positive Torque Limit	This variable indicates whether to enable the positive torque limit. TRUE: Enabled. FALSE: Disabled.	BOOL	Depends on data type.	---
TorqueLimitNegativeEnable	Enable Negative Torque Limit	This variable indicates whether to enable the negative torque limit. TRUE: Enabled. FALSE: Disabled.	BOOL	Depends on data type.	---
TorqueLimitPositiveVal	Positive Torque Limit	Positive torque limit	LREAL	0.0 or 0.1 to 1,000.0	%
TorqueLimitNegativeVal	Negative Torque Limit	Negative torque limit	LREAL	0.0 or 0.1 to 1,000.0	%

## ● Output Variable Update Timing

Variable	Timing for changing to TRUE	Timing for changing to FALSE
Done	<ul style="list-style-type: none"> <li>When <i>CtrlCode</i> is set to USINT#0 (deceleration stop) and the axis decelerates to a stop and the velocity reaches 0</li> <li>When <i>CtrlCode</i> is set to USINT#1 (absolute positioning) or USINT#2 (relative positioning) and positioning is completed</li> <li>If the value of <i>CtrlCode</i> is not one of the above values, this variable does not change to TRUE.</li> </ul>	<ul style="list-style-type: none"> <li>When <i>Execute</i> is TRUE and changes to FALSE</li> <li>After one task period when <i>Execute</i> is FALSE</li> </ul>
TargetReached	<ul style="list-style-type: none"> <li>When <i>CtrlCode</i> is set to USINT#3 (velocity control) and the target velocity is reached</li> <li>When <i>CtrlCode</i> is set to USINT#4 (torque control) and the target torque is output</li> <li>When <i>CtrlCode</i> is set to USINT#5 (torque feedback control) and the difference between <i>MeasuringTorque</i> (Measured Torque) and <i>TargetTorque</i> (Target Torque) becomes equal to or less than <i>InTorqueWidth</i> (In Torque Width)</li> <li>When <i>CtrlCode</i> is not set to one of the above values, this variable changes to TRUE at the same time as <i>Done</i> but for only one task period.</li> </ul>	<ul style="list-style-type: none"> <li>When <i>Error</i> changes to TRUE</li> <li>When <i>CommandAborted</i> changes to TRUE</li> <li>When this function block is re-executed and the target values are changed</li> </ul>
Busy	When <i>Execute</i> changes to TRUE	<ul style="list-style-type: none"> <li>When <i>Error</i> changes to TRUE</li> <li>When <i>CommandAborted</i> changes to TRUE</li> </ul>
Active	When output is started using the control method set in <i>CtrlCode</i>	<ul style="list-style-type: none"> <li>When <i>Error</i> changes to TRUE</li> <li>When <i>CommandAborted</i> changes to TRUE</li> </ul>
Command Aborted	<ul style="list-style-type: none"> <li>When this function block is aborted by another instruction or function block</li> <li>When this function block is aborted because an error occurred in another instruction or function block</li> <li>When this function block is started during execution of an MC_Stop instruction</li> </ul>	<ul style="list-style-type: none"> <li>When <i>Execute</i> is TRUE and changes to FALSE</li> <li>After one task period when <i>Execute</i> is FALSE</li> </ul>
Error	When the start conditions or input parameters for this function block contain the cause of an error	When the error is cleared

## In-Out Variables

Name	Meaning	Data type	Valid range	Unit	Description
Axis	Axis	_sAXIS_REF	---	---	Axis <sup>*1</sup>
SingleCmdProfile	Single-axis Command Profile	OmronLib\ ServoPress\ sSINGLE_CM D_PROFILE	---	---	Single-axis control instruction tables

\*1. Specify a user-defined Axis Variable that was created in the Axis Basic Settings of the Sysmac Studio (default: *MC\_Axis\*\**) or a system-defined axis variable name (*\_MC\_AX[\*\*]*).

### ● Structure

The data type of the *SingleCmdProfile* in-out variable is the structure *OmronLib\ServoPress\sSINGLE\_CMD\_PROFILE*. Refer to *Structure* in *SP\_SingleAxisPrgOpr* on page 42 for details.

For information on member valid ranges for *SingleCmdProfile*, refer to *Valid Ranges of SingleCmdProfile (Single-axis Command Profile) Members* on page 67.

## Function

You can set the parameters in *SingleCmdProfile* (Single-axis Command Profile) and execute this function block to execute position control, velocity control, torque control, and torque feedback control.

The next motion control instruction or motion control function is executed in this function block depending on the value of *CtrlCode* (Control Method).

Value of <i>CtrlCode</i>	Control method	Motion control instruction or motion control function	Reference
0	Deceleration stop	MC_Stop instruction	motion control instructions reference manual
1	Absolute positioning	MC_MoveAbsolute (Absolute Positioning) instruction	
2	Relative positioning	MC_MoveRelative (Relative Positioning) instruction	
3	CSV mode velocity control	CSV mode velocity control	<i>CSV Mode Velocity Control Function</i> on page 159
4	Torque control	MC_TorqueControl (Torque Control) Instruction	motion control instructions reference manual
5	Torque feedback control	Torque feedback control	<i>Torque Feedback Control Function</i> on page 162

When *CtrlCode* is set to USINT#1 (absolute positioning) or USINT#2 (relative positioning), you can use Blending Mode to continuously execute up to 10 function blocks without stopping.

The next member of *SingleCmdProfile* (Single-axis Command Profile) is output in the same task period as *TorqueLimitParam* (Torque Limit Settings). If you input these members to the MC\_SetTorqueLimit (Set Torque Limit) instruction and use the torque control function of the Servo Drive, the output torque of the Servo Drive will be limited.

- *TorqueLimitPositiveEnable* (Enable Positive Torque Limit)
- *TorqueLimitNegativeEnable* (Enable Negative Torque Limit)
- *TorqueLimitPositiveVal* (Positive Torque Limit)
- *TorqueLimitNegativeVal* (Negative Torque Limit)

## Meanings of Variables

The meanings of the other variables are described below.

### ● **MeasuringTorque (Measured Torque)**

This variable gives the actual axis torque that is measured. It is used as a feedback input of the actual torque monitor value in the torque feedback control. Convert the rated torque for the Servomotor to the percentage that is assumed as 100% and input the percentage in increments of %.

Use the LoadToTorque (Torque-to-Load Conversion) function when you convert the force that is measured by an externally-mounted load cell in newtons into a percentage in increments of % to the rated torque. Refer to *LoadToTorque* on page 177 for the details.

### ● **VelOverrideEnable (Velocity Override Enable)**

This variable is used to enable changes to the target velocity or the velocity limit.

When the value of *VelOverrideEnable* (Velocity Override Enable) is TRUE, you can change the target velocity or the velocity limit. Set the new target velocity or velocity limit with *InputVelFactor* (Velocity Override Input Value).

### ● **InputVelFactor (Velocity Override Input Value)**

This variable gives the input value when you change the target velocity or the velocity limit. The value of *InputVelFactor* (Velocity Override Input Value) is enabled only when the value of *VelOverrideEnable* (Velocity Override Enable) is TRUE.

The parameter to which *InputVelFactor* applies depends on the value of *CtrlCode* (Control Method), as described in the following table.

Value of <i>CtrlCode</i>	Control method	Motion control instruction or motion control function executed in this function block	Parameter for which the value of <i>InputVelFactor</i> is applied
0	Deceleration stop	MC_Stop Instruction	Not applied.
1	Absolute positioning	MC_MoveAbsolute (Absolute Positioning) instruction	Target velocity <sup>*1</sup>
2	Relative positioning	MC_MoveRelative (Relative Positioning) instruction	Target velocity <sup>*1</sup>
3	CSV mode velocity control	CSV Mode velocity control	Target velocity <sup>*1</sup>
4	Torque control	MC_TorqueControl (Torque Control) Instruction	Velocity limit <sup>*2</sup>
5	Torque feedback control	Torque feedback control	Velocity limit <sup>*1</sup>

\*1. If the value of *InputVelFactor* is changed while execution of the function block is in process, the value is refreshed for the processing within the same task period.

\*2. The value of *InputVelFactor* is not refreshed even if it is changed during processing of the function block.

### ● **ItgReset (Integral Processing Reset)**

This variable indicates whether or not to stop integral processing when performing torque feedback control when *CtrlCode* is set to USINT#5. The relationship between the value of *ItgReset* (Integral Processing Reset) and the integral processing is as follows:

Value of <i>ItgReset</i>	Integral processing
TRUE	Stop.
FALSE	Do not stop.

If the value of *CtrlCode* is not 5, the value of *ItgReset* is disabled.



- **BufferMode (Buffer Mode Selection)**

*BufferMode* specifies how operations join when this function block is executed during execution of another motion control instruction. The only valid value is *\_mcAborting* (Aborting). This means that the motion control instruction currently being executed is aborted and this function block is executed.

For details on buffer mode selections, refer to the motion control user's manual.

- **TargetReached (Target Reached)**

This variable indicates whether or not the target was reached. The value of *TargetReached* (Target Reached) changes to TRUE when the target is reached.

- **SingleCmdProfileNo (Single-axis Command Profile Number)**

When you consecutively execute multiple single-axis motion control instructions, this number indicates the ordinal number of the current single-axis motion control instruction.

Connect this variable to the *SingleCmdProfileNo* input variable in the SP\_SingleAxisPrgOpr (Single-axis Program Operation) function block.

- **TorqueLimitParam (Torque Limit Settings)**

This variable indicates whether the positive and negative torque limit values are enabled and gives the torque limit values.

- **OverrideEnabled (Override Enabled)**

This variable indicates whether the velocity override is enabled. If the value of *VelOverrideEnable* (Velocity Override Enable) is TRUE, the value of *OverrideEnabled* is also TRUE.

## Other Specifications

This section describes other specifications.

### ● Stopping Execution of This Function Block

To stop execution of this function block, use the MC\_Stop (Stop) instruction, or set *CtrlCode* to USINT#0 to re-execute the function block.

### ● Mapping Data Objects

For any value of *CtrlCode* (Control Method), map the following object data in the **Detailed Settings** Area of the Axis Basic Settings Display of the Sysmac Studio.

Value of <i>Ctrl-Code</i>	Control method	Object data mapping
0	Deceleration stop	*1
1	Absolute positioning	*1
2	Relative positioning	*1
3	CSV mode velocity control	<ul style="list-style-type: none"> <li>• Target velocity (60FF hex)</li> <li>• Modes of operation (6060 hex)</li> <li>• Modes of operation display (6061 hex)</li> </ul>
4	Torque control	<ul style="list-style-type: none"> <li>• Target velocity (6071 hex)</li> <li>• Modes of operation (6060 hex)</li> <li>• Torque actual value (6077 hex)</li> <li>• Modes of operation display (6061 hex)</li> </ul>
5	Torque feedback control	<ul style="list-style-type: none"> <li>• Target velocity (6071 hex)</li> <li>• Modes of operation (6060 hex)</li> <li>• Torque actual value (6077 hex)</li> <li>• Modes of operation display (6061 hex)</li> </ul>

\*1. For details, refer to information on required objects for PDO mapping in the motion control instructions reference manual.

### ● Continuous Position Control (Blending Mode) Specification

If you perform blending of consecutive position control operations, specify Blending Mode in *SingleCmdProfile* (Single-axis Command Profile).

## Timing Charts

### ● Normal End of Single-axis Control

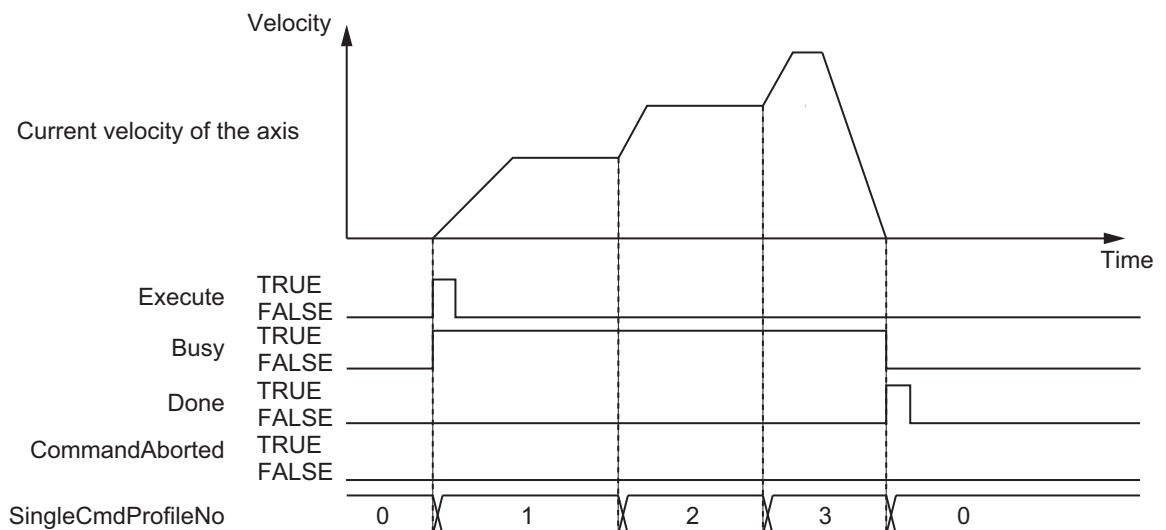
The timing chart when you use this function block for single-axis control depends of the value of *CtrlCode* (Control Method). For details, refer to the manual for instructions for each control method.

### ● Normal End of Continuous Position Control

Continuous position control can be executed when *CtrlCode* is set to USINT#1 (absolute positioning) or USINT#2 (relative positioning).

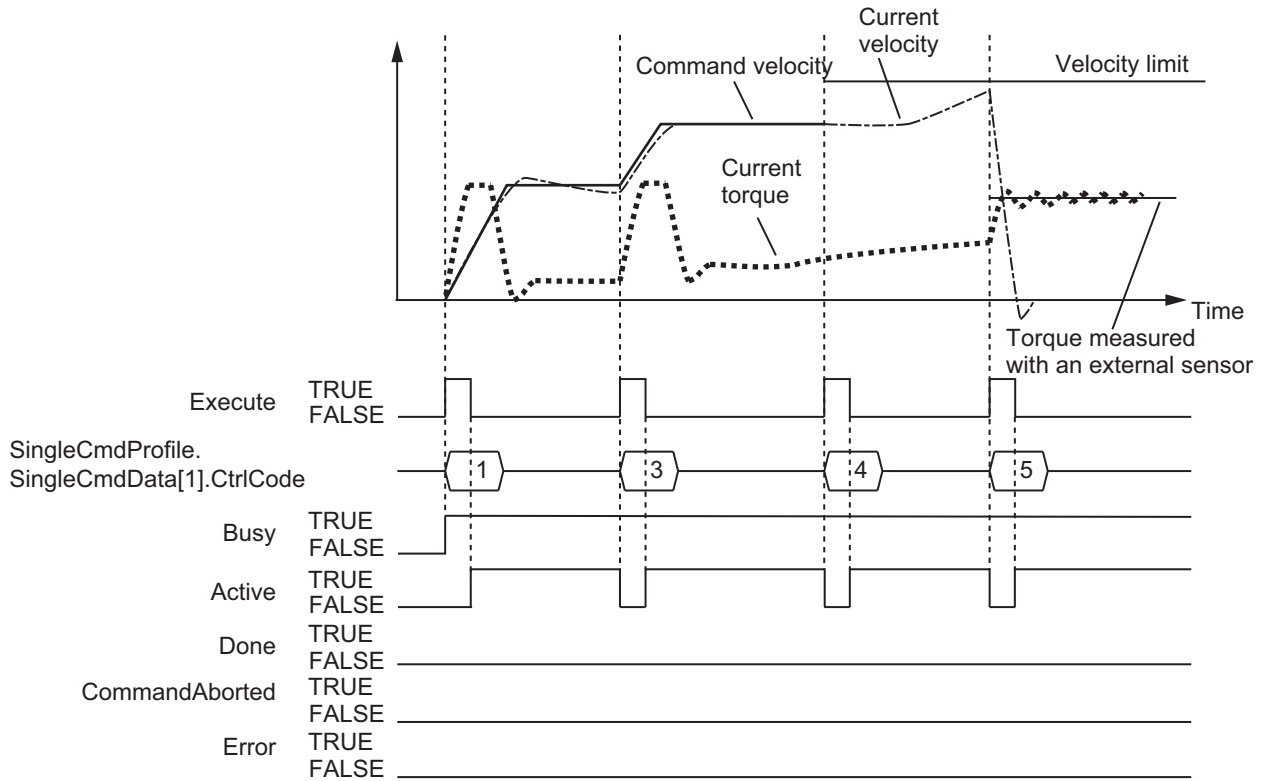
The following timing chart is for continuous operation of three position instructions when *SingleCmdProfile*. *BufferMode* is set to *\_mcBlendingPrevious* (Blending previous).

- *Busy* (Executing) changes to TRUE when *Execute* changes to TRUE.
- When these three position instructions are completed, the value of *Done* changes to TRUE.
- The value of *SingleCmdProfileNo* (Single-axis Command Profile Number) indicates the serial location of the current position instruction from the beginning.



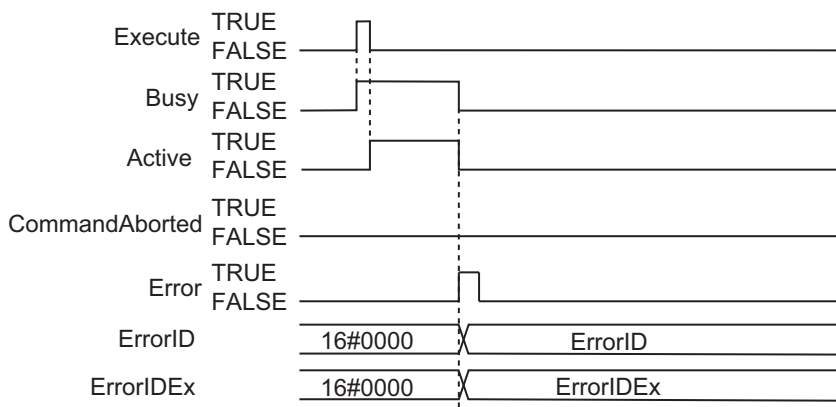
● **When Re-execution of this Function Block Is Performed with *Aborting***

If you change the value of *CtrlCode* (Control Method) during execution of this function block and re-execute the function block with *Aborting*, the operation for each output variable is shown in the following timing chart.



● **Error End**

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



## Re-execution of Motion Instructions

Whether this function block can be re-executed depends on the value of *CtrlCode* (Control Method) of the current instance of the function block. The following table indicates whether this function block can be re-executed for the different values of *CtrlCode*, and gives restrictions. However, re-execution of the function block refers to the execution of this function block when the value of *CtrlCode* is the same as the current instance of the function block.

Value of <i>Ctrl-Code</i>	Control method	Re-execution	Restrictions <sup>*1</sup>
0	Deceleration stop	Yes	When you re-execute the function block, the value of <i>Deceleration</i> (Deceleration Set Value) changes. The value of <i>Jerk</i> (Jerk Set Value) does not change.
1	Absolute positioning	Yes	You can change the following input variables when you re-execute the function block: <i>Position</i> (Position Set Value), <i>Velocity</i> (Velocity Set Value), <i>Acceleration</i> (Acceleration Set Value), and <i>Deceleration</i> (Deceleration Set Value).
2	Relative positioning	Yes	You can change the following input variables when you re-execute the function block: <i>Position</i> (Position Set Value), <i>Velocity</i> (Velocity Set Value), <i>Acceleration</i> (Acceleration Set Value), and <i>Deceleration</i> (Deceleration Set Value).
3	CSV mode velocity control	Yes	A Motion Control Instruction Re-execution Disabled event (error code: 543B hex) will occur.
4	Torque control	Yes	You can change the following input variables when you re-execute the function block: <i>Velocity</i> (Velocity Set Value), <i>Acceleration</i> (Acceleration Set Value), and <i>Deceleration</i> (Deceleration Set Value).
5	Torque feedback control	Yes	Any of the input variables can be changed when you re-execute the function block.

\*1. For details on restrictions, refer to the manual for instructions for each control method.

## Multi-execution of Motion Instructions

For details on multi-execution of motion control instructions, refer to the motion control user's manual.

### ● Execution of Other Instructions during Execution of this Function Block

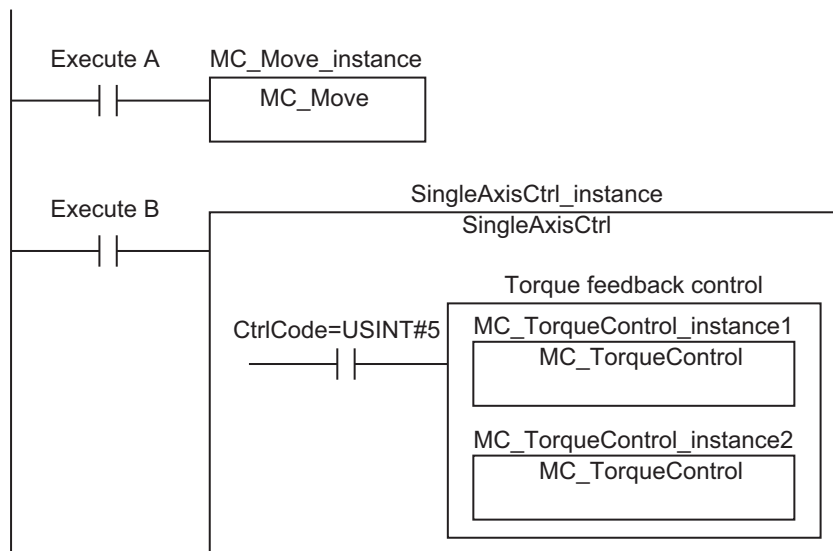
Whether other instructions can be executed during the execution of this function block depends on the value of *CtrlCode* (Control Method) of the current instance of the function block. For details, refer to the manual for instructions for each control method.



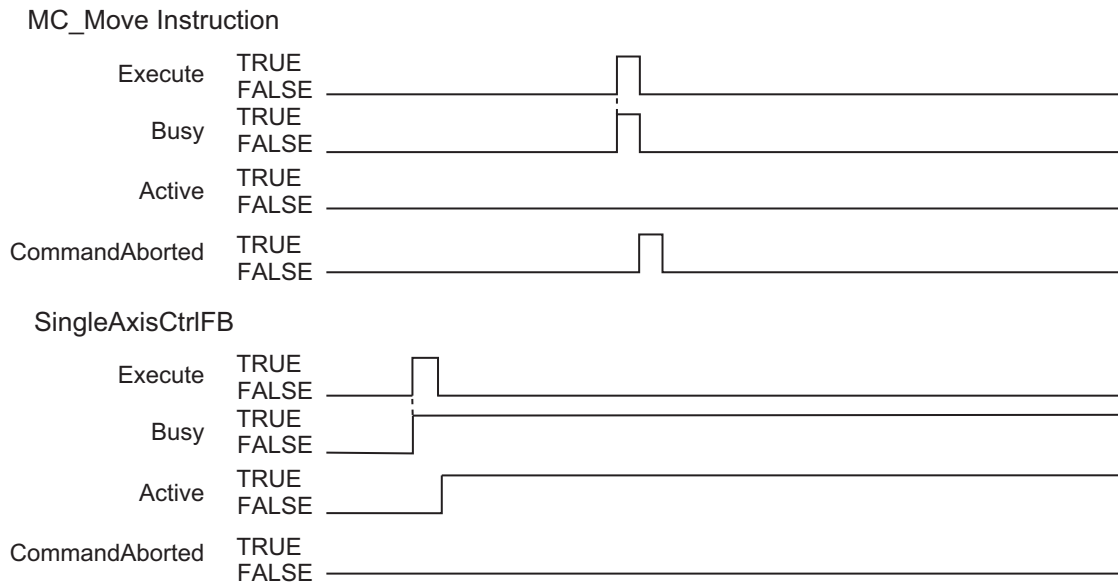
#### Precautions for Safe Use

If another motion control instruction is executed for the axis for which the SingleAxisCtrl (Single-axis Control) function block is being executed, write the user program so that multi-execution of the other instruction is not started until after execution of this function block is started.

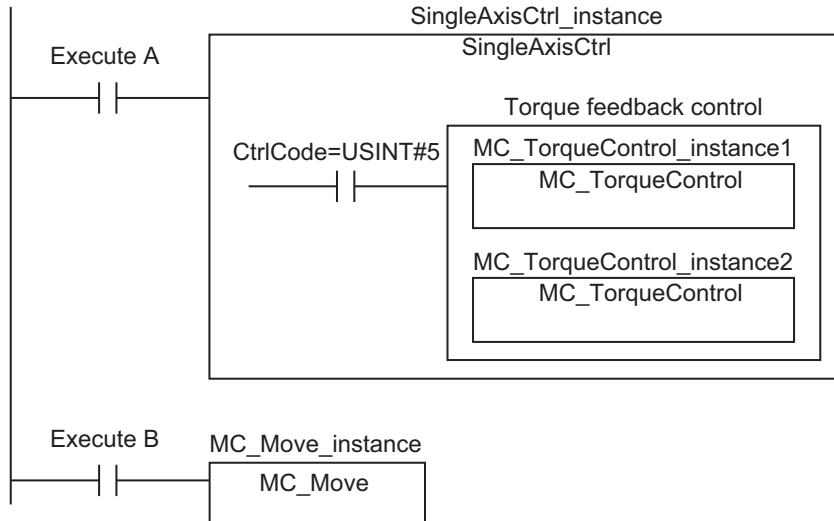
This function block alternately and internally performs multi-execution of two MC\_TorqueControl (Torque Control) instructions each task period. Therefore, in a user program that executes an MC\_Move (Positioning) instruction before this function block, as in the following figure, if multi-execution of the MC\_Move (Positioning) instruction is performed during execution of this function block, the MC\_TorqueControl (Torque Control) instruction that is executed later aborts execution of the MC\_Move (Positioning) instruction.



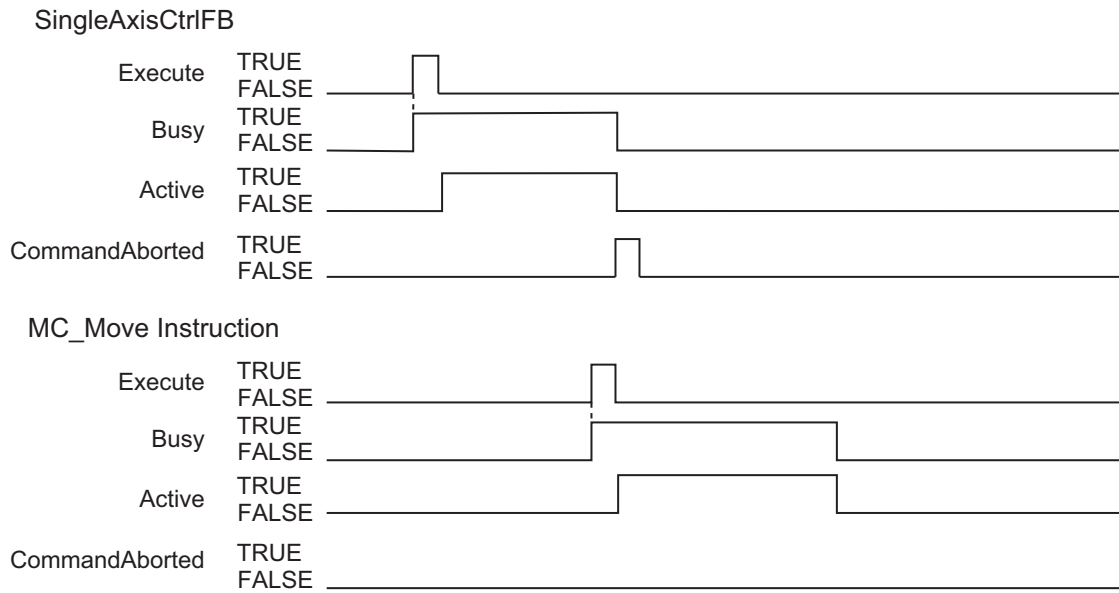
The following figure shows a timing chart for this programming.



On the other hand, in a user program that executes an MC\_Move (Positioning) instruction after this function block, the MC\_Move (Positioning) instruction is not aborted because multi-execution of the MC\_Move (Positioning) instruction is performed after execution of the MC\_TorqueControl (Torque Control) instruction in this function block.



The following figure shows a timing chart for this programming.





## CSV Mode Velocity Control Function

The CSV mode velocity control function is executed when *CtrlCode* is set to USINT#3.

This function uses *Velocity* (Velocity Set Value), *Acceleration* (Acceleration Rate Set Value), *Deceleration* (Deceleration Rate Set Value), and *Jerk* (Jerk Set Value) to calculate the velocity command value and outputs it to the Servo Drive each task period in Cyclic Synchronous Velocity (CSV) Control Mode.

In this function, CSV mode velocity control is implemented by using the velocity command value that is calculated from *Velocity* (Target Velocity), *Acceleration* (Acceleration Rate), and *Deceleration* (Deceleration Rate) to change the *Velocity* (Target Velocity) input variable of the MC\_SyncMoveVelocity (Cyclic Synchronous Velocity Control) instruction each task period. For details on the MC\_SyncMoveVelocity (Cyclic Synchronous Velocity Control) instruction, refer to the motion control instructions reference manual.

### Variables To Use

The following table gives the variables in *SingleCmdProfile* (Single-axis Command Profile) that are used for the CSV mode velocity control function.

Name	Meaning	Data type	Default	Valid range	Unit	Description
Velocity	Velocity Set Value	LREAL	0	Depends on data type.	Command units/s <sup>*1</sup>	The velocity set value.
Acceleration	Acceleration Rate Set Value	LREAL	0	Positive number or 0	Command units/s <sup>2</sup>	The acceleration rate set value.
Deceleration	Deceleration Rate Set Value	LREAL	0	Positive number or 0	Command units/s <sup>2</sup>	The deceleration rate set value.
Jerk	Jerk Set Value	LREAL	---	---	---	The jerk set value. <sup>*2</sup>
BufferMode	Buffer Mode Selection	_eM-C_BUFFER_MODE	_mcAborting	_mcAborting	---	Operation for multi-execution of motion control instructions _mcAborting: Aborting

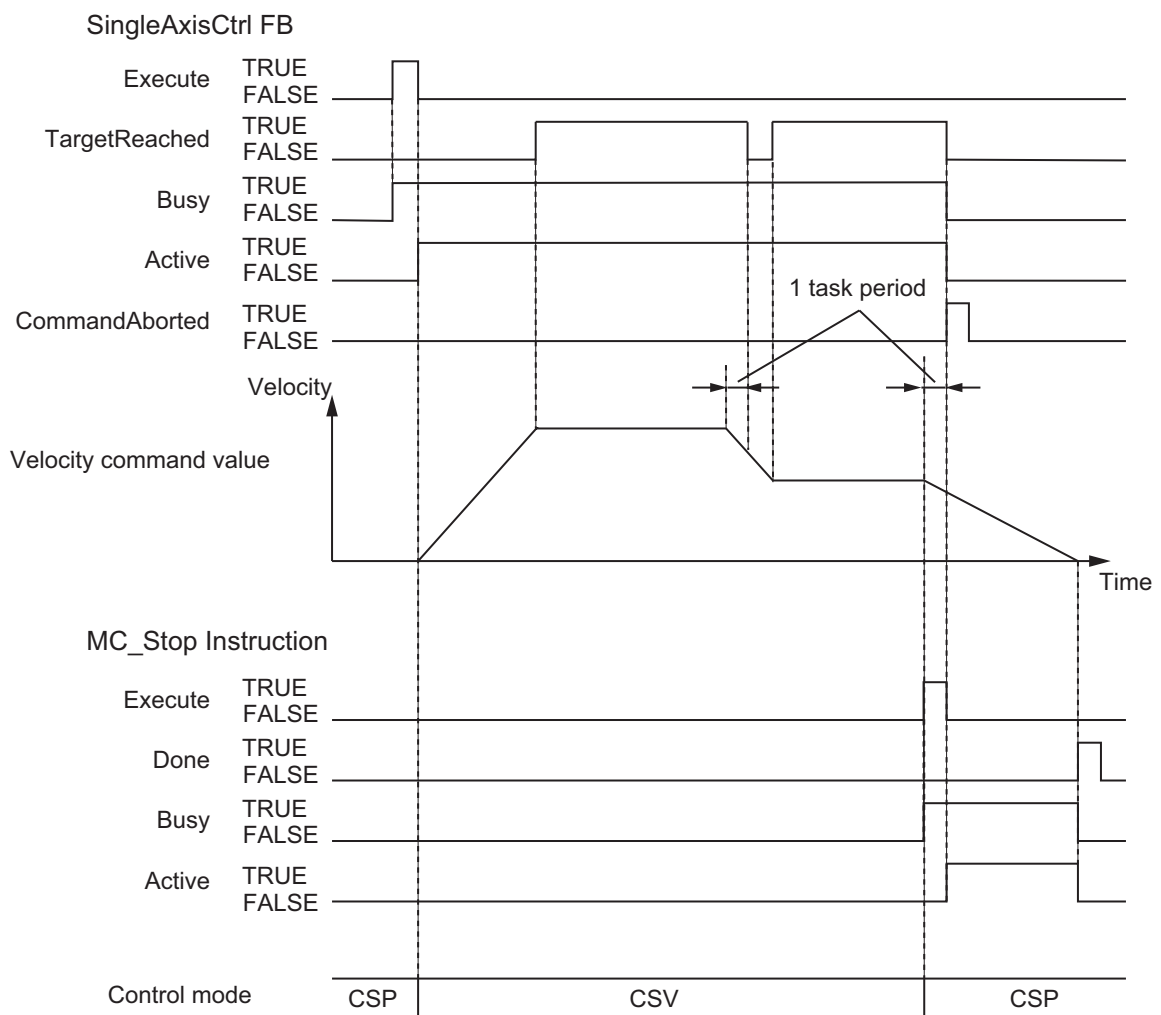
\*1. Refer to Unit Conversion Settings in the motion control user's manual for information on command units.

\*2. This is a reserved variable. The user cannot assign a value to it.

## Timing Charts

### ● Normal End

- *Busy* (Executing) changes to TRUE at the same time as *Execute* changes to TRUE. *Active* (Controlling) changes to TRUE in the next task period.
- *TargetReached* (Target Reached) changes to TRUE when the command velocity in this function block reaches *Velocity* (Velocity Set Value).
- If another instruction aborts execution of this function block, *CommandAborted* changes to TRUE and *Busy* (Executing), *Active* (Controlling), and *TargetReached* (Target Reached) change to FALSE.
- The MC\_Stop instruction is used to stop execution of this function block.

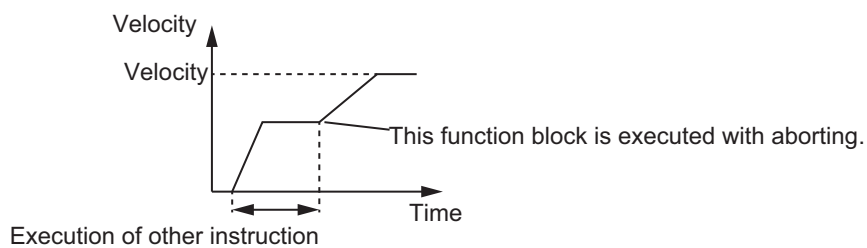


## Multi-execution of Motion Instructions

For details on multi-execution of motion control instructions in the CSV mode velocity control, refer to the motion control user's manual.

### ● Execution of This Function during Execution of Other Instructions

*BufferMode* (Buffer Mode Selection) can be set to Aborting. When this function is executed with *BufferMode* set to Aborting, the velocity of another instruction whose execution is in progress becomes the initial velocity for this function.



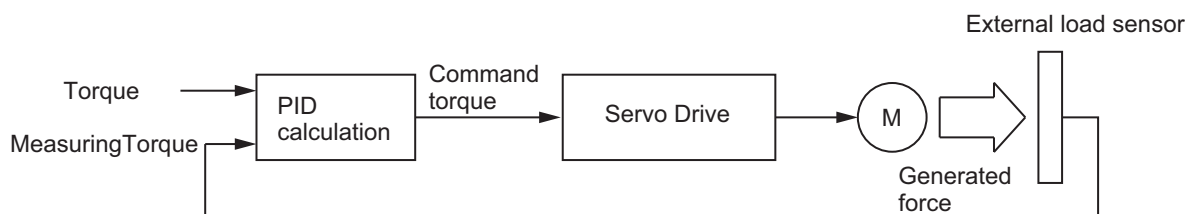
### ● Execution of Other Instructions during Execution of This Function

*BufferMode* (Buffer Mode Selection) can be set to Aborting or Buffered. Even when Buffered is selected, the other instruction is executed immediately, the same as when Aborting is set. The Control Mode is switched when processing for the multi-executed instructions is started.

## Torque Feedback Control Function

The torque feedback control function is executed when *CtrlCode* is set to USINT#5.

This function calculates *Torque* (Torque Set Value) and *MeasuringTorque* (Measured Torque) with PID and outputs the command torque to the Servo Drive each task period.



In this function, torque feedback control is implemented by inputting the command torque calculated to the *Torque* (Target torque) input variable of the MC\_TorqueControl (Torque Control) instruction.

For details on the MC\_TorqueControl (Torque Control) instruction, refer to the motion control instructions reference manual.

## Variables To Use

The following table gives the variables in *SingleCmdProfile* (Single-axis Command Profile) that are used for the torque feedback control function.

Name	Meaning	Data type	Default	Valid range	Unit	Description
Torque Ramp	Torque Ramp Set Value	LREAL	0	Positive number or 0	%/s	The torque set value.
Velocity	Velocity Set Value	LREAL	0	Positive number or 0	Command units/s <sup>*1</sup>	The velocity set value.
Torque	Torque Set Value	LREAL	0	-1000.0 to 1000.0	%	The torque set value.
TrqFbk Params	Torque Feedback Parameters	sTORQUE_FBK_PARAMS	---	---	---	Torque feedback control parameters
Kp	Proportional Gain	LREAL	1.0	0.0 to 3000.0	---	The proportional gain.
Ki	Integral Gain	LREAL	1.0	0.0 to 3000.0	---	The integral gain. <sup>*2</sup>
Kd	Derivative Gain	LREAL	1.0	0.0 to 3000.0	---	The derivative gain. <sup>*3</sup>
Torque LowLmt	Torque Lower Limit	LREAL	-300.0	-1,000.0 to 0.0 <sup>*4</sup>	0.1%	The output torque lower limit.
TorqueUp Lmt	Torque Upper Limit	LREAL	300.0	0.0 to 1,000.00 <sup>*4</sup>	0.1%	The output torque upper limit.
InTorque Width	In Torque Width	LREAL	0.1	0.0 to 100.00 <sup>*4</sup>	0.1%	The width for determining if the target torque is reached.
BufferMode	Buffer Mode Selection	_eM-C_BUFFER_MODE	_mcAborting	_mcAborting	---	Operation for multi-execution of motion control instructions _mcAborting: Aborting

\*1. For details, refer to Unit Conversion Settings in the motion control user's manual.

\*2. The integration time is 1 s.

\*3. The derivative time is 1 s.

\*4. The value is rounded to the second decimal place.

## Meanings of Variables

The meanings of the variable are described below.

### ● **Torque (Torque Set Value)**

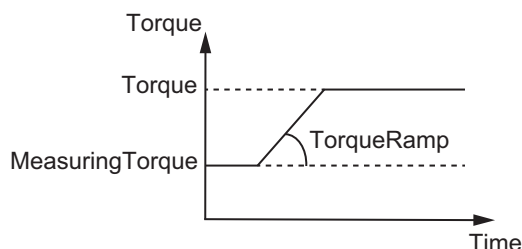
Set *Torque* to the target torque for the axis. Set a percentage in increments of 0.1% to *Torque* when the percentage of the rated torque for the Servomotor is assumed as 100%.

### ● **MeasuringTorque (Measured Torque)**

This is the actual axis torque measured by a sensor. Set a percentage in increments of 0.1% to *MeasuringTorque* (Measured Torque) when the percentage of the rated torque for the Servomotor is assumed as 100%.

- **TorqueRamp (Torque Ramp Set Value)**

Set *TorqueRamp* to the rate of change in the torque from the currently specified command torque until the target torque is output. The meaning of *TorqueRamp* is illustrated in the following figure.



- **Velocity (Velocity Set Value)**

Set *Velocity* to the maximum velocity of the axis during torque control.

When the axis velocity reaches *Velocity* (Velocity Set Value), the torque is limited by the Servo Drive and the axis velocity decreases. The velocity limit is implemented with a Servo Drive function.

For details on the velocity limit, refer to information on torque control in each Servo Drive manual.

- **TrqFbkParams (Torque Feedback Parameters)**

These are the torque feedback control parameters.

*Kp* (Proportional Gain), *Ki* (Integral Gain), *Kd* (Derivative Gain), *TorqueLowLmt* (Torque Lower Limit), and *TorqueUpLmt* (Torque Upper Limit) are set.

*InTorqueWidth* (In Torque Width) is the width that is used to determine when the target torque is reached. When the difference between *Torque* and *MeasuringTorque* (Measured Torque) becomes equal to or less than the value of *InTorqueWidth*, it is assumed that target torque was reached and *TargetReached* (Target Reached) changes to TRUE.

## Other Specifications

This section describes other specifications.

- **Stopping Axes during Torque Feedback Control**

If the MC\_Stop instruction is executed during execution of this function block for an OMRON G5-series Servo Drive, the deceleration rate that is specified for the MC\_Stop instruction is not used and an immediate stop is performed. An immediate stop is also performed even for errors that normally result in deceleration stops.

- **Command Current Position and Actual Current Position during Torque Feedback Control**

The following current positions are given in the system-defined variables for motion control during torque feedback control for this function block.

Type of position	Value
Actual current position	The value returned by the Servo Drive and then multiplied by the gear ratio
Command current position	Actual current position for one task period earlier

## ● Applicable Axes and Execution Condition

- For a servo axis, this function block is ready for execution as soon as *Enable* for the MC\_Power (Power Servo) instruction changes to TRUE (Servo ON state).
- A virtual servo axis will acknowledge this instruction at any time. However, processing to change the Control Mode of the Servo Drive is not performed.
- An error occurs if the instruction is executed for an encoder or virtual encoder axis.

## ● Operation When Servo Turns OFF

Processing to change to CSP Mode is performed by the MC Function Module when the value of the *Status* output variable from the MC\_Power (Power Servo) instruction changes to FALSE.

However, for an OMRON G5-series Servo Drive, commands to change the Control Mode are not acknowledged from the MC Function Module when the Servo is OFF.

## ● Axis Variable Status

*Status.Continuous* (Continuous Motion) in the axis variable status changes to TRUE.

Also, *CST* (Cyclic Synchronous Torque (CST) Control Mode) in the *DrvStatus* (Servo Drive Status) axis variable changes to TRUE.

## ● Home Status

Home remains defined.

## ● Software Limits

The software limits are applied.

They are applied even if one of the following is selected in the axis parameter: Deceleration stopping enabled for command position or Immediate stopping enabled for command position (stop using remaining pulses).

## ● When Count Mode Is Set to Linear Mode

The operation for underflows and overflows is the same as for operations that do not have target positions.

## ● Operation Selection at Reversing

If multi-execution is performed and the torque command value is reversed, the command value is immediately reversed regardless of the setting of the **Operation Selection at Reversing** axis parameter.

The operation for reversing due to multi-execution of instructions is as follows:

- If the command position is reversed by multi-execution of an instruction that uses CSP during execution of this function block, the reversing operation is performed according to the **Operation Selection at Reversing** axis parameter.
- If the torque command value is reversed by multi-execution of this function block during execution of an instruction that uses CSP or CSV, the torque command reverses immediately.
- If the torque command value is reversed by multi-execution of this function block during execution of this function block, the torque command reverses immediately. If the torque command value is reversed by multi-execution of this function block during execution of this function block, the torque command reverses immediately.

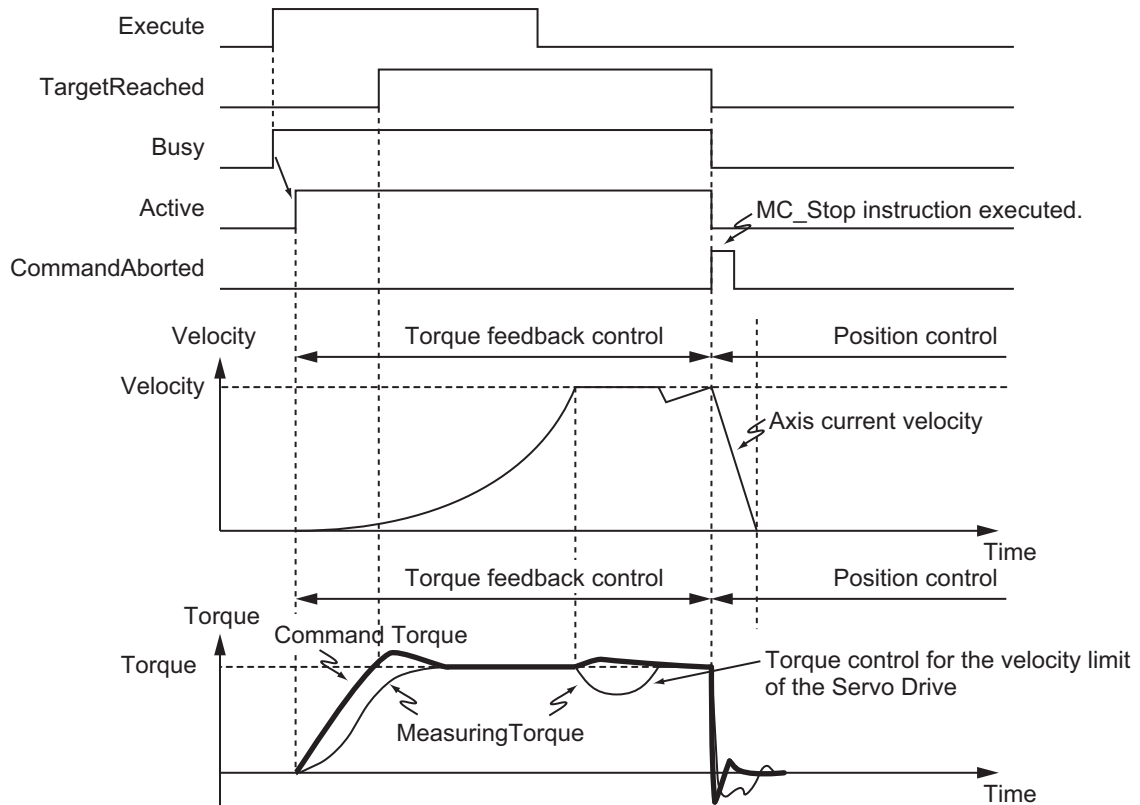
## ● Changing the Control Mode

- If execution of this function block is aborted by another instruction such as MC\_MoveAbsolute (Absolute Positioning) or if an axis error occurs, the Control Mode changes to position control at that point.
- The value of *Active* (Controlling) changes to TRUE when this function block is executed, but it takes several task periods for the Control Mode in the Servo Drive to change. The time that is required for the Control Mode to change depends on the Servo Drive.

## Timing Charts

### ● Normal end

- *Busy* (Executing) changes to TRUE at the same time as *Execute* changes to TRUE. *Active* (Controlling) changes to TRUE in the next task period.
- When the difference between *Torque* (Torque Set Value) and *MeasuringTorque* (Measured Torque) becomes equal to or less than the value of *InTorqueWidth* (In Torque Width), *TargetReached* (Target Reached) changes to TRUE.
- If another instruction aborts this function, *CommandAborted* changes to TRUE and *Busy* (Executing), *Active* (Controlling), and *TargetReached* (Target Reached) change to FALSE.
- The MC\_Stop instruction is used to stop execution of this function block.

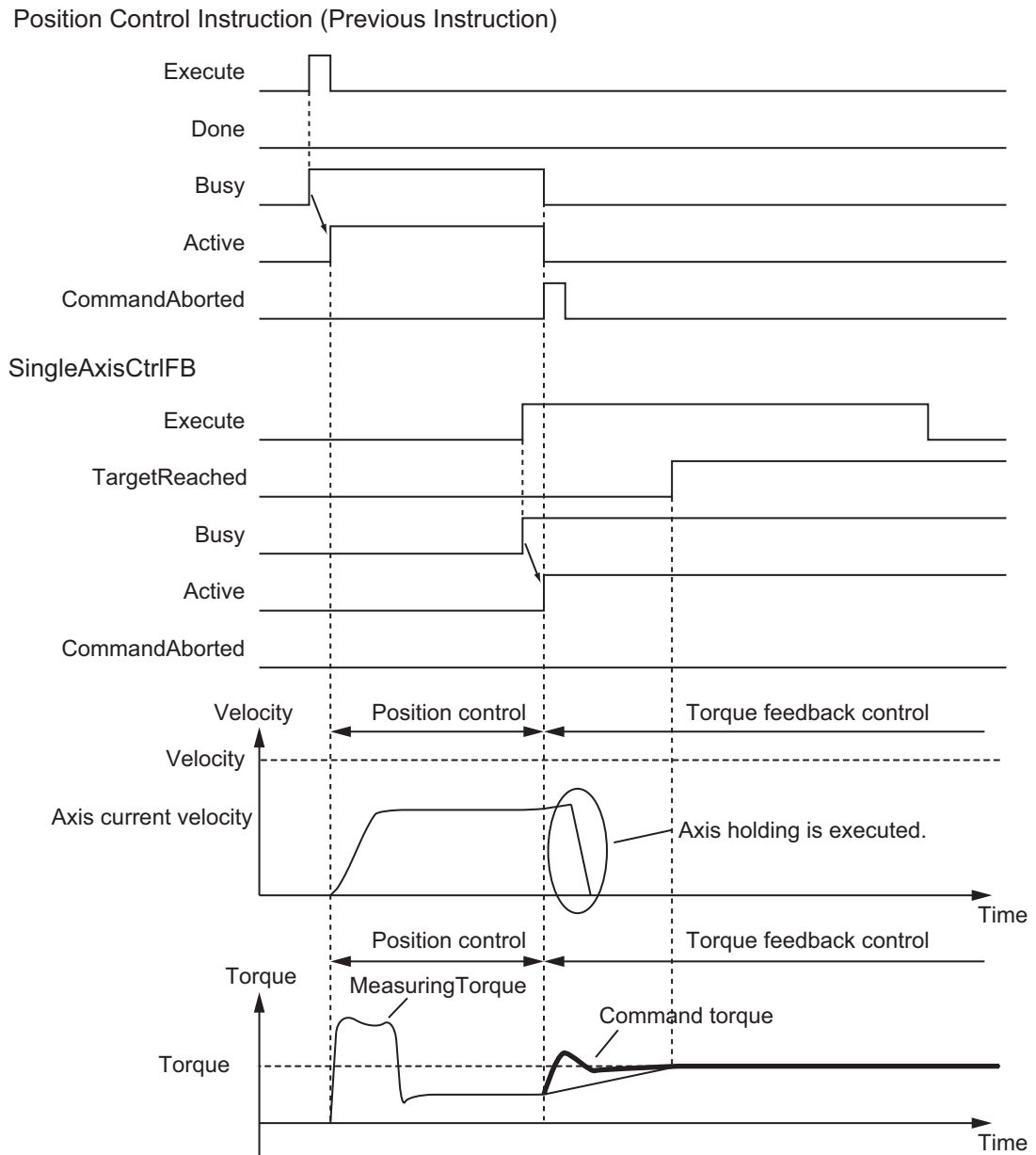




### ● When Multi-execution of This Function Is Performed with Aborting

Control changes to torque control when multi-execution of this function is performed.

The following timing chart shows a case when the axis stops and is held during execution of this function.



## Additional Information

The MC Function Module sends a command to the Servo Drive to change the Control Mode with the timing shown in the timing chart. The timing of implementing the Control Mode change in the Servo Drive depends on Servo Drive specifications.

## Precautions for Correct Use

- If the value of *MeasuringTorque* (Measuring Torque), *VelOverrideEnable* (Velocity Override Enable), *InputVelFactor* (Velocity Override Input Value), or *ItgReset* (Integral Processing Reset) is changed during execution of this function block, the value is updated for the processing within the same task period.
- You can use only *CtrlCode* = USINT#0 (single-axis program operation completion), *CtrlCode* = USINT#1 (absolute positioning), and *CtrlCode* = USINT#2 (relative positioning) for an NX-series NX-PG0□□□ Pulse Output Unit.
- When you set *Velocity* (Velocity Set Value), *Acceleration* (Acceleration Rate Set Value), and *Deceleration* (Deceleration Rate Set Value), make sure that an excessive load is not placed on the mechanical machine of the system for the change in velocity.
- Set *Torque* (Torque Set Value) so that the maximum torque of the motor is not exceeded. The operation that is performed when the maximum torque of the motor is exceeded depends on the Servo Drive.
- The axis velocity is higher for torque control. Make sure that you set *Velocity* (Velocity Set Value) for safety.
- When you use an OMRON G5-series Servo Drive, set the Velocity Limit Selection (3317 hex) of the Servo Drive to 1 (speed limit value via EtherCAT communications). Otherwise, the velocity limit is not affected. Also, the axis does not stop even if the limit input signal turns ON.
- Process data 607F hex is used for the velocity limit. When you use an OMRON G5-series Servo Drive, set the advanced settings in the Axis Parameter Settings of the Sysmac Studio to use the Velocity Limit Value (607F hex). To use a velocity limit with a Servo Drive from another manufacturer, refer to the manual for the Servo Drive.
- If *MeasuringTorque* (Measured Torque) is not connected for torque feedback control, the axis torque command value is greater and the axis velocity is higher. Make sure that you set *TorqueLowLmt* (Torque Lower Limit), *TorqueUpLmt* (Torque Upper Limit), and *Velocity* (Velocity Set Value) for safety.
- If you change the Control Mode of the Servo Drive during axis operation, an error may occur depends on the Servo Drive.
- Execute the SingleAxisCtrl function block and the SP\_SingleAxisPrgOpr (Single-axis Program Operation) function block in the same task when you connect these two function blocks. If these two function blocks are executed in the different task, the motion control may not be correctly executed.



### Precautions for Safe Use

- When you change the velocity override during execution of this function block, use *VelOverrideEnable* (Velocity Override Enable) and *InputVelFactor* (Velocity Override Input Value) in this function block. If you use the MC\_SetOverride (Set Override Factors) instruction to change velocity override, the last set value that is specified is valid.
- When the Control Mode is changed, the current position may change suddenly.
- *When you use the LoadToTorque (Load-to-Torque Conversion) function to calculate the MeasuringTorque (Measured Torque) input variable in the SingleAxisCtrl function block, execute the LoadToTorque function and the SingleAxisCtrl function block in the same task.*

## Troubleshooting

The error codes, expansion error codes, status, descriptions, and corrections are given in the following table.

Error code	Expansion error code	Status	Description	Correction
16#0000	16#00000000	Normal end		
16#3C0A	16#00000005	Torque feedback control proportional gain value out of valid range	The value of <i>Kp</i> (Proportional Gain) is outside the valid range.	Check the valid range of the value of <i>Kp</i> (Proportional Gain) and set the value within the valid range.
	16#00000006	Torque feedback control integral gain value out of valid range	The value of <i>Ki</i> (Integral Gain) is outside the valid range.	Check the valid range of the value of <i>Ki</i> (Integral Gain) and set the value within the valid range.
	16#00000007	Torque feedback control derivative gain value out of valid range	The value of <i>Kd</i> (Derivative Gain) is outside the valid range.	Check the valid range of the value of <i>Kd</i> (Derivative Gain) and set the value within the valid range.
	16#00000008	Torque feedback control output limit value out of valid range	<ul style="list-style-type: none"> <li>The relationship of the values of the <i>TorqueLowLmt</i> and <i>TorqueUpLmt</i> input variables to this function block does not meet the required condition.</li> <li>Nonnumeric data is input for <i>TorqueUpLmt</i> (Torque Upper Limit) or <i>TorqueLowLmt</i> (Torque Lower Limit).</li> </ul>	<ul style="list-style-type: none"> <li>Correct the relationship so that <i>TorqueUpLmt</i> (Torque Upper Limit) is equal to or greater than <i>TorqueLowLmt</i> (Torque Lower Limit).</li> <li>Set <i>TorqueUpLmt</i> (Torque Upper Limit) and <i>TorqueLowLmt</i> (Torque Lower Limit) to correct real numbers.</li> </ul>
	16#0000000A	Torque feedback control in-torque value out of valid range	The value of <i>InTorqueWidth</i> (In Torque Width) is outside the valid range.	Check the valid range of the value of <i>InTorqueWidth</i> (In Torque Width) and set the value within the valid range.
16#3C4E	16#00000001	Axis error detected	An axis error occurred.	*1
	16#00000002	Buffer mode selection out of range	The value of <i>BufferMode</i> (Buffer Mode Selection) is outside the valid range.	Check the valid range of the value of <i>BufferMode</i> (Buffer Mode Selection) and set the value within the valid range.
	16#00□□0001*2	Illegal control method	The value of <i>SingleCmdProfile</i> [□□]. <i>CtrlCode</i> (Control Method) is incorrect.	Correct the value of <i>SingleCmdProfile</i> [□□]. <i>CtrlCode</i> (Control Method).

Error code	Expansion error code	Status	Description	Correction
16#3C4F	16#00000001	Buffer mode selection out of range	The value of <i>BufferMode</i> (Buffer Mode Selection) is outside the valid range.	Check the valid range of the value of <i>BufferMode</i> (Buffer Mode Selection) and set the value within the valid range.
	16#00000002	Axis error detected	An axis error occurred.	*1
	16#00000003	Velocity set value out of range	The value of <i>Velocity</i> (Velocity Set Value) is outside the valid range.	Check the valid range of the value of <i>Velocity</i> (Velocity Set Value) and set the value within the valid range.
	16#00000004	Acceleration rate set value out of range	The value of <i>Acceleration</i> (Acceleration Rate Set Value) is outside the valid range.	Check the valid range of the value of <i>Acceleration</i> (Acceleration Rate Set Value) and set the value within the valid range.
	16#00000005	Deceleration rate set value out of range	The value of <i>Deceleration</i> (Deceleration Rate Set Value) is outside the valid range.	Check the valid range of the value of <i>Deceleration</i> (Deceleration Rate Set Value) and set the value within the valid range.
16#3C5D	16#00000001	Buffer mode selection out of range	The value of <i>BufferMode</i> (Buffer Mode Selection) is outside the valid range.	Check the valid range of the value of <i>BufferMode</i> (Buffer Mode Selection) and set the value within the valid range.
	16#00000002	Axis error detected.	An axis error occurred.	*1
	16#00000003	Torque set value out of range	The value of <i>Torque</i> (Torque Set Value) is outside the valid range.	Check the valid range of the value of <i>Torque</i> (Torque Set Value) and set the value within the valid range.
	16#00000004	Velocity set value out of range	The value of <i>Velocity</i> (Velocity Set Value) is outside the valid range.	Check the valid range of the value of <i>Velocity</i> (Velocity Set Value) and set the value within the valid range.
	16#00000005	Motion control instruction re-execution disabled	An attempt was made to re-execute a motion control instruction that cannot be re-executed.	Stop this function block and then re-execute the motion control instruction.

Error code	Expansion error code	Status	Description	Correction
16#3C5E	16#00000001	Buffer mode selection out of range	The value of <i>BufferMode</i> (Buffer Mode Selection) is outside the valid range.	Check the valid range of the value of <i>BufferMode</i> (Buffer Mode Selection) and set the value within the valid range.
	16#00000002	Axis error detected	An axis error occurred.	*1
	16#00000003	Torque set value out of range	The value of <i>Torque</i> (Torque Set Value) is outside the valid range.	Check the valid range of the value of <i>Torque</i> (Torque Set Value) and set the value within the valid range.
	16#00000004	Measured torque out of range	The value of <i>MeasuringTorque</i> (Measured Torque) is outside the valid range.	Check the valid range of the value of <i>MeasuringTorque</i> (Measured Torque) and set the value within the valid range.
	16#00000005	Torque ramp set value out of range	The value of <i>TorqueRamp</i> (Torque Ramp Set Value) is outside the valid range.	Check the valid range of the value of <i>TorqueRamp</i> (Torque Ramp Set Value) and set the value within the valid range.
	16#00000006	Velocity set value out of range	The value of <i>Velocity</i> (Velocity Set Value) is outside the valid range.	Check the valid range of the value of <i>Velocity</i> (Velocity Set Value) and set the value within the valid range.

\*1. For details, refer to the motion control user's manual.

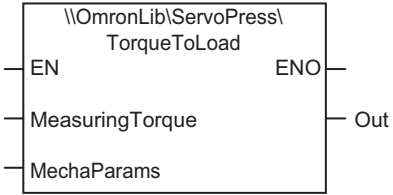
\*2. The boxes (□□) are the array element number of SingleCmdProfile (Single-axis Command Profile).

## Sample Programming

Refer to *Sample Programming* on page 87 in the description of the SP\_SingleAxisPrgOpr (Single-axis Program Operation) function block.

# TorqueToLoad

The TorqueToLoad function converts the measured torque into a load value.

Function name	Name	FB/ FUN	Graphic expression	ST expression
TorqueToLoad	Torque-to-Load Conversion	FUN		<pre>Out:=\\OmronLib\ServoPress\ TorqueToLoad (   MeasuringTorque,   MechaParams);</pre>

## Function Block and Function Information

Item	Description
Library file name	OmronLib_ServoPress_V1_0.slr OmronLib_ServoPress_V2_0.slr
Namespace	OmronLib\ServoPress
Function block and function number	00083
Publish/Do not publish source code	Do not publish.
Function block and function version	1.00

## Variables

### Input Variables

Name	Meaning	Data type	Default	Valid range	Unit	Description
EN	Execute	BOOL	FALSE	Depends on data type.	---	Execute TRUE: The function is executed. FALSE: The function is not executed.
Measuring Torque	Measured Torque	LREAL	0	Depends on data type.	%*1	Measured torque
Mecha Params	Mechanism Parameters	Omron-Lib\Servo-Press\sMECHA_PARAMS	---	---	---	Mechanism parameters

\*1. Set the percentage of the rated torque.

## ● Structure

The data type of the *MechaParams* input variable is structure OmronLib\ServoPress\sMECHA\_PARAMS. The specifications are as follows:

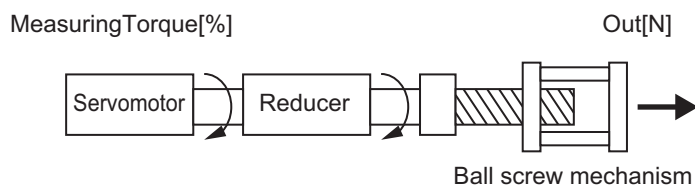
Name	Meaning	Description	Data type	Valid range	Unit	Default
Mecha-Params	Mechanism Parameters	The mechanism parameters.	OmronLib\ServoPress\sMECHA_PARAMS	---	---	---
ReductionGear-Param	Reduction Gear Parameters	The reduction gear parameters.	OmronLib\ServoPress\sREDUCTION_GEAR_PARAM	---	---	---
Rn	Gear Ratio Numerator	The numerator of the gear ratio.	DINT	Depends on data type.	---	0
Rd	Gear Ratio Denominator	The denominator of the gear ratio.	UDINT	Positive number	---	0
n2	Reduction Gear Efficiency	The efficiency of the reduction gear.	REAL	Positive number	%	0
BallScrew-Param	Ball Screw Parameters	The ball screw parameters.	OmronLib\ServoPress\sBALL_SCREW_PARAM	---	---	---
n1	Ball Screw Efficiency	The efficiency of the ball screw.	REAL	Positive number	%	0
R	Ball Screw Pitch	The pitch of the ball screw.	REAL	Positive number	mm	0
MotorRatedParam	Motor Rated Parameter	The motor rated parameter.	OmronLib\ServoPress\sMOTOR_RATING_CAPACITY	---	---	---
Tr	Rated Torque	The rated torque of the motor.	REAL	Positive number	N·m	0

## Output Variables

Name	Meaning	Data type	Valid range	Unit	Description
ENO	Done	BOOL	Depends on data type.	---	Done TRUE: Normal end FALSE: Error end, execution in progress, or execution condition not met.
Out	Load Value	LREAL	Depends on data type.	N	The load value after conversion.

## Function

This function converts *MeasuringTorque* (Measured Torque) of the motor to *Out* (Load Value) in a direct-operation conversion mechanism that uses a motor, reducer, and ball screw, as shown in the following figure.



*MeasuringTorque* is given as a percentage of the rated torque. The unit for *MeasuringTorque* is %. The unit for *Out* is newtons (N).

The conversion formula is as follows:

$$\text{Out[N]} = \frac{2\pi \times n1 \times 1000}{R} \times \frac{Rd}{Rn} \times n2 \times Tr \times \frac{\text{MeasuringTorque[\%]}}{100}$$

For example, if the values of the input variables are as follows, the value of *Out* (Load Value) is 859.539518431971.

Input variables	Value
MeasuringTorque	50
Tr	0.64
Rn	1
Rd	5
n2	0.9
n1	0.95
R	10

## Precautions for Correct Use

If the value of an input variable is out of range, an error occurs and the value of *ENO* changes to FALSE.

## Sample Programming

This sample programming converts the torque monitor value of the Servomotor (unit: percentage of rated Servomotor torque) to the load for the servo press actuator mechanism (unit: newton).



### Precautions for Correct Use

- The sample programming shows only the portion of a program that uses the function or function block from the library.
- When you use the actual device, include user programming for device safety instructions, interlocks, I/O with other devices, and other control procedures.
- Create a user program that will produce the intended device operation.
- Check the user program for proper execution before you use it for actual operation.



## Conditions

The conditions are the same as those for the SP\_SingleAxisPrgOpr (Single-axis Program Operation) function block sample programming.

## Processing

The torque measurement value as a percentage of the rated Servomotor torque is converted to the load in newtons according to the specified machine parameters.

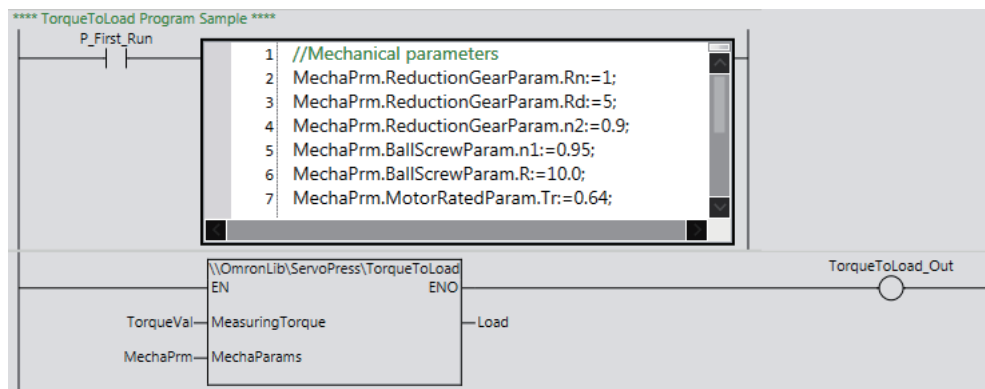
## Ladder Diagram

The following gives the main variables.

### ● Internal Variables

Name	Data Type	Initial Value	Comment
TorqueVal	LREAL		Torque measurement value
Load	LREAL		Torque-to-load conversion value
MechaPrm	OmronLib\ServoPress\ sMECHA_PARAMS		Servo press actuator machine parameters

### ● Algorithm



## ST

The following gives the main variables.

### ● Internal Variables

Name	Data Type	Initial Value	Comment
TorqueVal	LREAL		Torque measurement value
Load	LREAL		Torque-to-load conversion value
MechaPrm	OmronLib\ServoPress\sMECHA_PARAMS		Servo press actuator machine parameters

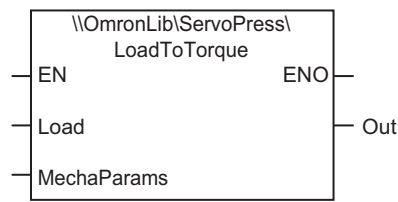
### ● Algorithm

```
//Mechanical parameters
MechaPrm.ReductionGearParam.Rn:=1;
MechaPrm.ReductionGearParam.Rd:=5;
MechaPrm.ReductionGearParam.n2:=0.9;
MechaPrm.BallScrewParam.n1:=0.95;
MechaPrm.BallScrewParam.R:=10.0;
MechaPrm.MotorRatedParam.Tr:=0.64;

Load:=\\OmronLib\ServoPress\TorqueToLoad(
  EN:=TRUE,
  ENO=>TorqueToLoad_Out,
  MeasuringTorque:=TorqueVal,
  MechaParams:=MechaPrm
);
```

# LoadToTorque

The TorqueToLoad function converts a load value into a torque.

Function name	Name	FB/ FUN	Graphic expression	ST expression
LoadToTorque	Load-to-Torque Conversion	FUN		<pre>Out:=\\OmronLib\ServoPress \LoadToTorque (   Load,   MechaParams);</pre>

## Function Block and Function Information

Item	Description
Library file name	OmronLib_ServoPress_V1_0.slr OmronLib_ServoPress_V2_0.slr
Namespace	OmronLib\ServoPress
Function block and function number	00084
Publish/Do not publish source code	Do not publish.
Function block and function version	1.00

## Variables

### Input Variables

Name	Meaning	Data type	Default	Valid range	Unit	Description
EN	Execute	BOOL	FALSE	Depends on data type.	---	Execute TRUE: The function is executed. FALSE: The function is not executed.
Load	Load Value	LREAL	0	Depends on data type.	N	The load value.
Mecha Params	Mechanism Parameters	OmronLib\ ServoPress\ sMECHA_ PARAMS	---	---	---	The mechanism parameters.

### ● Structure

The data type of the *MechaParams* input variable is structure OmronLib\ServoPress\sMECHA\_PARAMS. Refer to *Structure* in *TorqueToLoad* on P.173 for details.

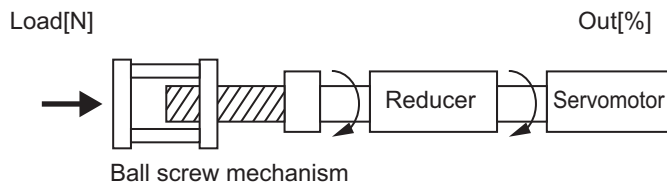
## Output Variables

Name	Meaning	Data type	Valid range	Unit	Description
ENO	Done	BOOL	Depends on data type.	---	Done TRUE: Normal end FALSE: Error end, execution in progress, or execution condition not met.
Out	Torque	LREAL	Depends on data type.	%*1	The torque after conversion.

\*1. Set the percentage of the rated torque.

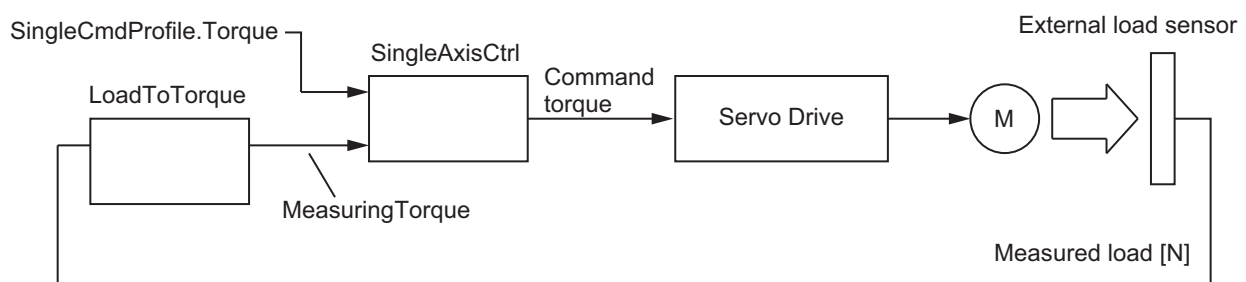
## Function

*Load* is converted to *Out* (Torque) for a press mechanism that uses a motor, reducer, and ball screw, as shown in the following figure.



The unit for *Load* is newtons (N). *Out* is given as a percentage of the rated torque. The unit for *Out* is %.

This function is used when you execute torque feedback control using the SingleAxisCtrl (Single Axis Control) function block. This function converts the load measured by a sensor to a motor torque as shown in the following figure and sets this torque in the *MeasuringTorque* (Measured Torque) input variable for the SingleAxisCtrl (Single Axis Control) function block.



The conversion formula is as follows:

$$\text{Out}[\%] = \text{Load}[\text{N}] \times \frac{R}{2\pi \times n1 \times 1000} \times \frac{Rn}{Rd} \times \frac{100}{n2 \times Tr}$$

For example, if the values of the input variables are as follows, the value of *Out* (Load Value) is 0.581706820079817.

Input variables	Value
Load	10
Tr	0.64
Rn	1
Rd	5
n2	0.9
n1	0.95
R	10

## Precautions for Correct Use

If the value of an input variable is out of range, an error occurs and the value of *ENO* changes to FALSE.



### Precautions for Safe Use

---

When you use the LoadToTorque (Load-to-Torque Conversion) function to calculate the *Measuring Torque* (Measured Torque) input variable in the SingleAxisCtrl function block, execute the LoadToTorque function and the SingleAxisCtrl function block in the same task.

---

## Sample Programming

Refer to *Sample Programming* on page 87 in the description of the SP\_SingleAxisPrgOpr (Single-axis Program Operation) function block.

# PrgOprRsltRec

The PrgOprRsltRec function block records the results of single-axis program operation.

Function block name	Name	FB/FUN	Graphic expression	ST expression
PrgOprRsltRec	Program Operation Results Recorder	FB		<pre>PrgOprRsltRec_instance (   Enable,   StepExec,   StepCompleted,   CurrentStepNo,   Position,   Load,   Enabled,   Busy,   Error,   ErrorID,   ErrorIDEx,   PrgOprRsltRecorder);</pre>

## Function Block and Function Information

Item	Description
Library file name	OmronLib_ServoPress_V1_0.slr OmronLib_ServoPress_V2_0.slr
Namespace	OmronLib\ServoPress
Function block and function number	00080
Publish/Do not publish source code	Published.
Function block and function version	1.00

## Variables

### Input Variables

Name	Meaning	Data type	Default	Valid range	Unit	Description
Enable	Enable	BOOL	FALSE	Depends on data type.	---	Enable TRUE: Enable FALSE: Do not enable execution.
StepExec	Step Start Trigger	BOOL	FALSE	Depends on data type.	---	The start trigger for the step.
StepCompleted	Step Completed Trigger	BOOL	FALSE	Depends on data type.	---	The completion trigger for the step.
Current-StepNo	Current Step Number	USINT	0	Depends on data type.	---	The number of the step being executed.
Position	Current Position	LREAL	0	Depends on data type.	---	The current position of the controlled system.
Load	Current Load	LREAL	0	Depends on data type.	Load units*1	The current load value of the controlled system.

\*1. The unit of load is not specified in this function block. Uniquely set newtons (N) or % as a torque conversion value in the user program that uses this function block.

### Output Variables

Name	Meaning	Data type	Valid range	Unit	Description
Enabled	Enabled	BOOL	Depends on data type.	---	Enabled TRUE: Enabled. FALSE: Not enabled.
Busy	Executing	BOOL	Depends on data type.	---	Executing TRUE: Executing FALSE: Not executing
Error	Error	BOOL	Depends on data type.	---	Error end TRUE: Error end FALSE: Normal end, executing, or execution conditions not met
ErrorID	Error Code	WORD	*1	---	An error code is output if an error occurs. The value is WORD#16#0 for a normal end.
ErrorIDEx	Expansion Error Code	DWORD	*1	---	An expansion error code is output if an error occurs. The value is DWORD#16#0 for a normal end.

\*1. For details, refer to *Troubleshooting* on page 188.



## In-Out Variables

Name	Meaning	Data type	Valid range	Unit	Description
PrgOprRslt Recorder	Program Operation Results Recorder	OmronLib\ ServoPress\ sPRG_ OPR_RSLT_ RECORDER	---	---	The recorder for the program operation results.

### ● Structure

The data type of the *PrgOprRsltRecorder* in-out variable is the structure OmronLib\ServoPress\sPRG\_OPR\_RSLT\_RECORDER. The specifications are as follows:

Name	Meaning	Description	Data type	Valid range	Unit
PrgOprRslt Recorder	Program Operation Results Recorder	The recorder for the program operation results.	OmronLib\ ServPress\sPRG_ OPR_RSLT_ RECORDER	---	---
ExecDate Time	Record Date and Time	The date and time at which recording started.	DATE_AND_TIME	Depends on data type.	Year, month, day, hour, minutes, seconds
PrgMaxPos	Program Operation Maximum Position	The maximum position in the entire operation of a single-axis program.	LREAL	Depends on data type.	---
PrgLoad MaxPos	Load at Program Operation Maximum Position	The load at the maximum position in program operation.	LREAL	Depends on data type.	Load units <sup>*1</sup>
PrgMax Load	Program Operation Maximum Load	The maximum load in the entire operation of a single-axis program.	LREAL	Depends on data type.	Load units <sup>*1</sup>
PrgPosMax Load	Position at Program Operation Maximum Load	The position at the maximum load for program operation.	LREAL	Depends on data type.	---
StepRslt Rec	Step Execution Results	The results of step execution.	ARRAY[0..49] OF OmronLib\Servo- Press\sSTEP_RSLT_ _REC	---	---
No	Step Number	The number of the step.	USINT	Depends on data type.	---
StartPos	Step Start Position	The position at the start of the step.	LREAL	Depends on data type.	---
EndPos	Step End Position	The position at the end of the step.	LREAL	Depends on data type.	---
Start Load	Step Start Load	The load at the start of the step.	LREAL	Depends on data type.	Load units <sup>*1</sup>
EndLoad	Step End Load	The load at the end of the step.	LREAL	Depends on data type.	Load units <sup>*1</sup>
MaxPos	Step Maximum Position	The maximum position in the step.	LREAL	Depends on data type.	---

Name	Meaning	Description	Data type	Valid range	Unit
LoadMax-Pos	Load at Step Maximum Position	The load at the maximum position in the step.	LREAL	Depends on data type.	Load units <sup>*1</sup>
MaxLoad	Step Maximum Load	The maximum load in the step.	LREAL	Depends on data type.	Load units <sup>*1</sup>
PosMax Load	Position of Step Maximum Load	The position of the maximum load in the step.	LREAL	Depends on data type.	---

\*1. The unit of load is not specified in this function block. Uniquely set newtons (N) or % as a torque conversion value in the user program that uses this function block.

## Function

In the operation of a single-axis program that uses the SP\_SingleAxisPrgOpr (Single-Axis Program Operation) function block, the results of execution of each step and the results of execution of the entire single-axis program are recorded in *PrgOprRsltRecorder* (Program Operation Results Recorder).

### Connection with the SP\_SingleAxisPrgOpr (Single-Axis Program Operation) Function Block

This function block is used together with the SP\_SingleAxisPrgOpr (Single-Axis Program Operation) function block. This function block and the SP\_SingleAxisPrgOpr (Single-Axis Program Operation) function block are connected with the following three corresponding signal lines. Refer to NTLPxREFER *SP\_SingleAxisPrgOpr* on page 74 for details on the SP\_SingleAxisPrgOpr (Single-Axis Program Operation) function block.

Signal line in this function block	Signal line in the connected SP_SingleAxisPrgOpr (Single-Axis Program Operation) function block
Enable	Busy
StepExec	StepExec
StepCompleted	StepCompleted

### Procedure for Recording Single-Axis Program Operation Results

The results of single-axis program operation are recorded with the following procedure.

Step No.	Input variable value	Operation of this function block
1	The value of <i>Enable</i> changes from FALSE to TRUE.	<ul style="list-style-type: none"> <li>• <i>PrgOprRsltRecorder</i> (Program Operation Results Recorder) is initialized.</li> <li>• The current date and time are recorded in <i>ExecDateTime</i> (Record Date and Time).</li> <li>• Tracing of the current position and load of the axis is started.</li> <li>• <i>PrgMaxPos</i> (Program Operation Maximum Position), <i>PrgLoadMaxPos</i> (Load at Program Operation Maximum Position), <i>PrgMaxLoad</i> (Program Operation Maximum Load), and <i>PrgPosMaxLoad</i> (Position at Program Operation Maximum Load) are recorded. If there is a change in these values while tracing the values, the recording is updated immediately.</li> </ul>
2 <sup>*1</sup>	The value of <i>StepExec</i> (Step Start Trigger) changes from FALSE to TRUE.	<i>No</i> (Step Number), <i>StartPos</i> (Step Start Position), and <i>StartLoad</i> (Step Start Load) for this step are obtained and then tracing of the current position and load of the axis is started.
3 <sup>*1</sup>	The value of <i>StepCompleted</i> (Step Completed Trigger) changes from FALSE to TRUE.	<i>EndPos</i> (Step End Position), <i>EndLoad</i> (Step End Load), <i>MaxPos</i> (Step Maximum Position), <i>LoadMaxPos</i> (Load at Step Maximum Position), <i>MaxLoad</i> (Step Maximum Load), and <i>PosMaxLoad</i> (Position at Step Maximum Load) for this step are obtained. They are combined with the results obtained in step 2, and then <i>PrgOprRsltRecorder</i> (Program Operation Results Recorder) is recorded. <sup>*2</sup>
4	The value of <i>Enable</i> changes to FALSE.	Execution is ended.

\*1. Steps 2 and 3 are executed each time the step changes.

\*2. The updated variables are members of the *StepRsltRec[CurrentStepNo]* structure array.

## Meanings of Variables

The meanings of the other variables are described below.

- **CurrentStepNo (Current Step Number)**

This variable is used to input the number of the step currently being executed.

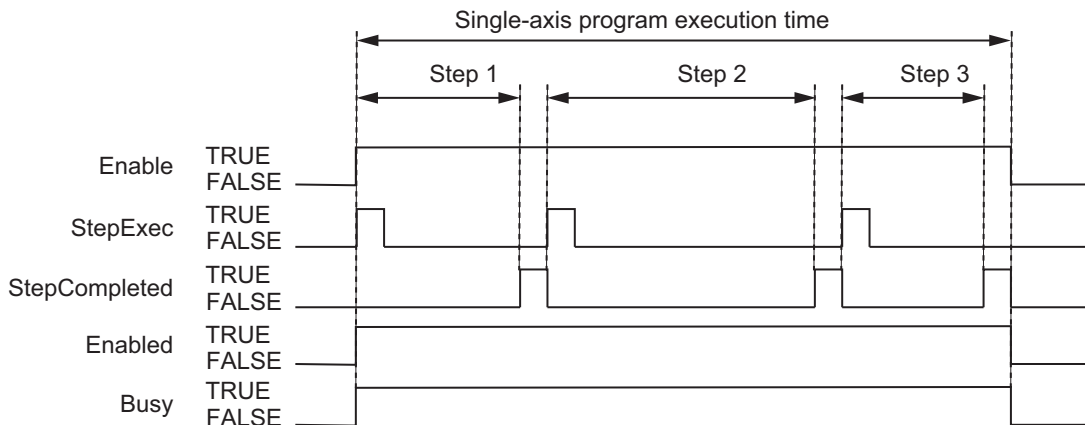
- **Position (Current Position) and Load (Current Load)**

These variables are used to input the current position and the current load for the controlled system.

## Timing Charts

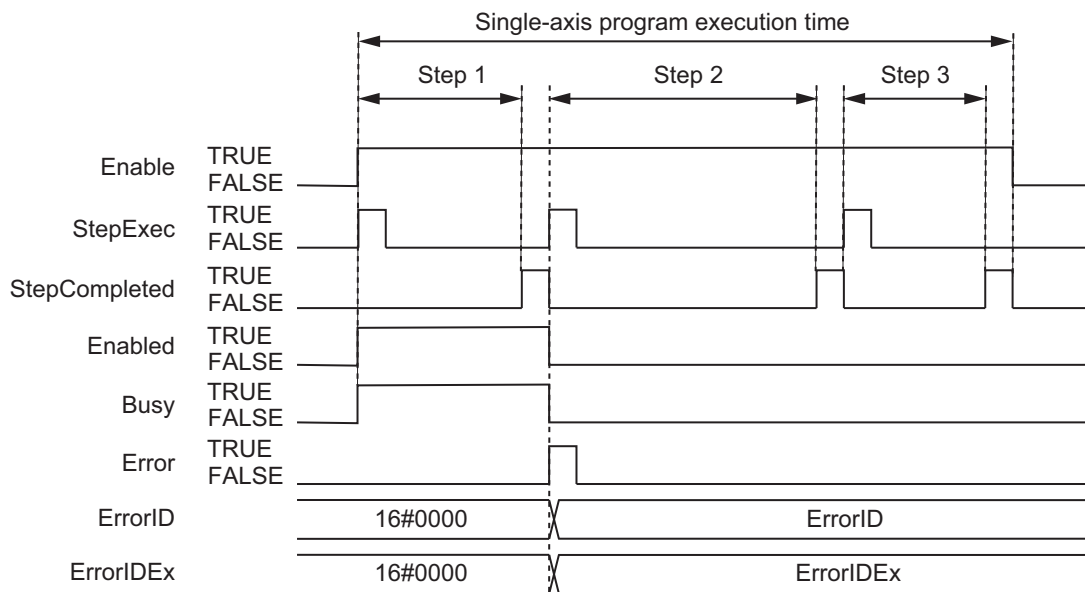
- **Normal End**

- *Enabled* changes to TRUE at the same time as *Enable* in this function block changes to TRUE.
- While *Enabled* is TRUE, the value of *StepRsltRec* (Step Execution Results) is updated at each step in single-axis program operation.
- The start and completion of each step are recognized when *StepExec* (Step Start Trigger) and *StepCompleted* (Step Completed Trigger) change to TRUE.
- The execution results when *Enable* changes to TRUE and when *Enable* changes to FALSE are recorded.
- The execution results when *StepExec* (Step Start Trigger) and *StepCompleted* (Step Completed Trigger) change to TRUE are recorded.



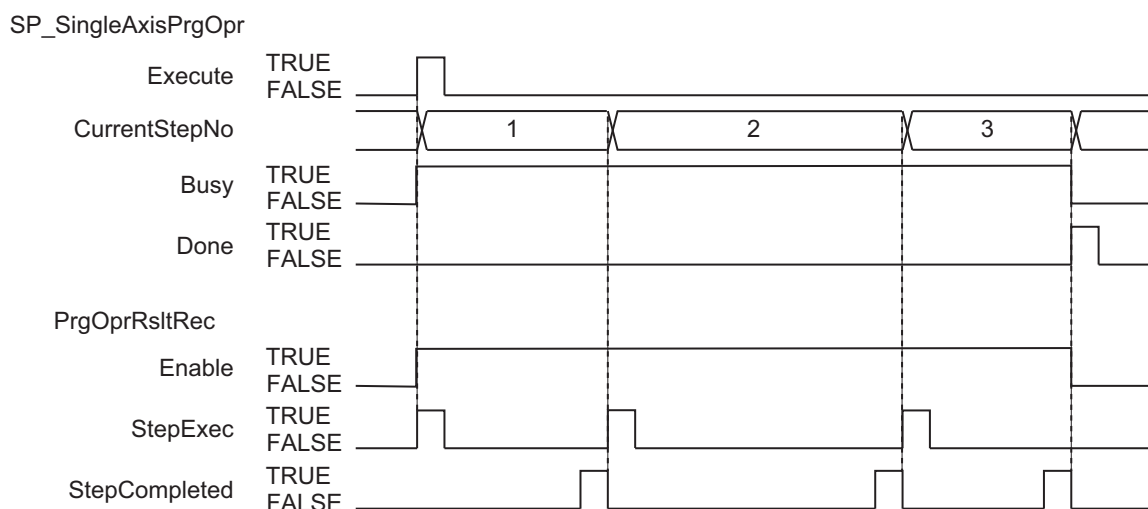
● **Error End**

- If an error occurs during execution of this function block, *Error* changes to TRUE. You can find out the cause of the error by referring to the values output by *ErrorID* (Error Code) and *ErrorIDEx* (Expansion Error Code).
- The execution results when an error occurs are not recorded. *StepRsltRec* (Step Execution Results) is recorded for the last step that is completed before the error. In the following figure, *StepRsltRec* (Step Execution Results) is recorded for step 1, but for step 2 and after, *StepRsltRec* (Step Execution Results), *PrgMaxPos* (Program Operation Maximum Position), *PrgLoadMaxPos* (Load at Program Operation Maximum Position), *PrgMaxLoad* (Program Operation Maximum Load), and *PrgPosMaxLoad* (Position at Program Operation Maximum Load) are not recorded.



● **Relationship with the SP\_SingleAxisPrgOpr (Single-Axis Program Operation) Function Block**

- When the value of *CurrentStepNo* for the SP\_SingleAxisPrgOpr (Single-Axis Program Operation) function block changes, the values of *StepExec* (Step Start Trigger) and *StepCompleted* (Step Completed Trigger) of this function block are changed.



## Additional Information

To save the contents of *PrgOprRsItRecorder* (Program Operation Results Recorder) to an SD Memory Card in CSV format, use the PrgOprRsItCSVWrite function block on P.191.

## Precautions for Correct Use

- Do not execute this function block at the same time as the PrgOprRsItCSVWrite (Write Program Operation Results to SD Memory Card) function block. If you do, the results of executing a single-axis program may not be recorded correctly.
- Do not change the contents of *PrgOprRsItRecorder* (Program Operation Results Recorder) with a user program while execution of a single-axis program is in progress. If you do, the results of executing a single-axis program may not be recorded correctly.
- When the power supply is turned OFF to the Controller, the traced data is discarded.

## Troubleshooting

The error codes, expansion error codes, status, descriptions, and corrections given in the following table.

Error code	Expansion error code	Status	Description	Correction
16#0000	16#00000000	Normal end		
16#3C5F	16#00000001	Step Execution Results Value Error	An attempt was made to record step execution results that exceeded the size of the <i>StepRsItRec</i> (Step Execution Results) array.	Check that the <i>StepRsItRec</i> (Step Execution Results) array is at least as large as the number of steps in the single-axis program operation.

## Sample Programming

This sample programming creates program operation result record data for SP\_SingleAxisPrgOpr (Single-axis Program Operation) function block sample programming execution results.

It is added and executed after the SP\_SingleAxisPrgOpr (Single-axis Program Operation) function block sample programming.



### Precautions for Correct Use

- The sample programming shows only the portion of a program that uses the function or function block from the library.
- When you use the actual device, include user programming for device safety instructions, interlocks, I/O with other devices, and other control procedures.
- Create a user program that will produce the intended device operation.
- Check the user program for proper execution before you use it for actual operation.

## Conditions

The conditions are the same as those for the SP\_SingleAxisPrgOpr (Single-axis Program Operation) function block sample programming.

## Processing

If *PrgRsltRec\_En* (Enable Program Operation Result Recording) is TRUE and data was not saved to the SD Memory Card with the *PrgRsltCSVWrite* (Program Operation Results SD Memory Card Write) function block, this processing records the program operation results while the *SP\_SingleAxisPrgOpr* (Single-axis Program Operation) function block is being executed.

## Ladder Diagram

The following gives the main variables.

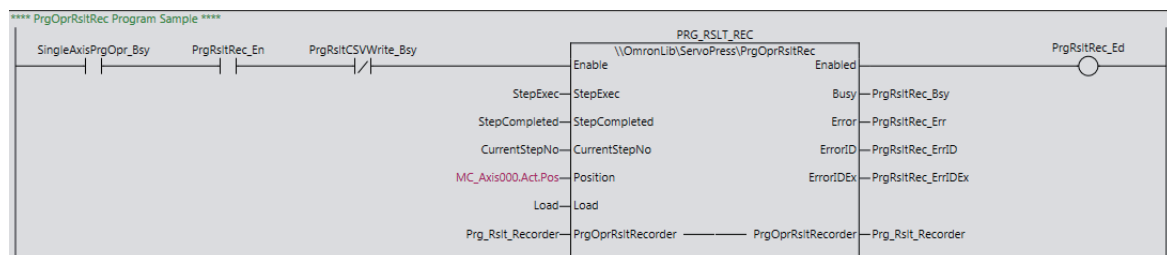
### ● Internal Variables

Name	Data Type	Initial Value	Comment
PRG_RSLT_REC	OmronLib\ServoPress\PrgOprRsltRec		Instance of the PrgOprRsltRec (Program Operation Results Recorder) function block
SingleAxisPrgOpr_Bsy	BOOL		Single-axis program operation busy
PrgRsltRec_En	BOOL		Enables program operation result recording.
PrgRsltCSVWrite_Bsy	BOOL		Saving data to memory card (Used in the PrgOpsRsltCSVWrite (Program Operation Results SD Memory Card Write) function block sample programming.)
StepExec	BOOL		Ends step.
StepCompleted	BOOL		Starts step.
CurrentStepNo	USINT		Current step number
Prg_Rslt_Recorder	OmronLib\ServoPress\sPRG_O-PR_RSLT_RECORDER		Program operation result record data

### ● External Variables

Name	Data Type	Initial Value	Comment
MC_Axis000	_sAXIS_REF	---	Servo axis

### ● Algorithm



## ST

The following gives the main variables.

### ● Internal Variables

Name	Data Type	Initial Value	Comment
PRG_RSLT_REC	OmronLib\ServoPress\ PrgOprRsltRec		Instance of the PrgOprRsltRec (Program Operation Results Recorder) function block
SingleAxisPrgOpr_Bsy	BOOL		Single-axis program operation busy
PrgRsltRec_En	BOOL		Enables program operation result recording.
PrgRsltCSVWrite_Bsy	BOOL		Saving data to memory card (Used in the PrgOpsRsltCSVWrite (Program Operation Results SD Memory Card Write) function block sample programming.)
StepExec	BOOL		Ends step.
StepCompleted	BOOL		Starts step.
CurrentStepNo	USINT		Current step number
Prg_Rslt_Recorder	OmronLib\ServoPress\sPRG_ OPR_RSLT_RECORDER		Program operation result record data

### ● External Variables

Name	Data Type	Initial Value	Comment
MC_Axis000	_sAXIS_REF	---	Servo axis

### ● Algorithm

```
//PrgOprRsltRec
//Start on PrgOprRsltCSVWrite not in progress.
PRG_RSLT_REC(
  Enable := SingleAxisPrgOpr_Bsy AND PrgRsltRec_En AND NOT(PrgRsltCSVWrite_Bsy),
  StepExec := StepExec,
  StepCompleted := StepCompleted,
  CurrentStepNo := CurrentStepNo,
  Position := MC_Axis000.Act.Pos,
  Load := MC_Axis000.Act.Trq,
  Enabled => PrgRsltRec_Ed,
  Busy => PrgRsltRec_Bsy,
  Error => PrgRsltRec_Err,
  ErrorID => PrgRsltRec_ErrID,
  ErrorIDEx => PrgRsltRec_ErrIDEx,
  PrgOprRsltRecorder := Prg_Rslt_Recorder
);
```



# PrgOprRsltCSVWrite

The PrgOprRsltCSVWrite function block writes the results of single-axis program operation to an SD Memory Card in CSV format.

Function block name	Name	FB/FUN	Graphic expression	ST expression
PrgOprRsltCSVWrite	Write Program Operation Results to SD Memory Card	FB		<pre>PrgOprRsltCSVWrite_instance ( Execute, PrgOprRsltRecorder, FileName, Done, Busy, Error, ErrorID, ErrorIDEx);</pre>

## Function Block and Function Information

Item	Description
Library file name	OmronLib_ServoPress_V1_0.slr OmronLib_ServoPress_V2_0.slr
Namespace	OmronLib\ServoPress
Function block and function number	00081
Publish/Do not publish source code	Published.
Function block and function version	1.00

## Variables

### Input Variables

Name	Meaning	Data type	Default	Valid range	Unit	Description
Execute	Execute	BOOL	FALSE	Depends on data type.	---	Execute trigger for this function block Executes the function block when it changes to TRUE.
FileName	File Name	STRING	"	66 bytes max. (65 single-byte alphanumeric characters plus the final NULL character)	---	File name of CSV file to write.

## Output Variables

Name	Meaning	Data type	Valid range	Unit	Description
Done	Done	BOOL	Depends on data type.	---	Completion of execution TRUE: Normal end FALSE: Error end, execution in progress, or execution condition not met
Busy	Executing	BOOL	Depends on data type.	---	Executing TRUE: Executing FALSE: Not executing
Error	Error	BOOL	Depends on data type.	---	Error end TRUE: Error end FALSE: Normal end, executing, or execution conditions not met
ErrorID	Error Code	WORD	*1	---	An error code is output if an error occurs. The value is WORD#16#0 for a normal end.
ErrorIDEx	Expansion Error Code	DWORD	*1	---	An expansion error code is output if an error occurs. The value is DWORD#16#0 for a normal end.

\*1. For details, refer to *Troubleshooting* on page 197.

## In-Out Variables

Name	Meaning	Data type	Valid range	Unit	Description
PrgOprRsltRecorder	Program Operation Results Recorder	OmronLib\ ServoPress\ sPRG_ OPR_RSLT_ RECORDER	---	---	The recorder for the program operation results.

### ● Structure

The data type of the *PrgOprRsltRecorder* in-out variable is the structure `OmronLib\ServoPress\sPRG_OPR_RSLT_RECORDER`. Refer to *Structure* in *PrgOprRsltRec* on P.181 for details.

## Function

This function block writes the *PrgOprRsltRecorder* (Program Operation Results Recorder) results of single-axis program operation to an SD Memory Card in CSV format. The name of the file to write is specified with *FileName*.

With *FileName*, you can specify the name including the folder. If the specified folder does not exist, an error occurs. If the folder is not specified, create *FileName* in the root of the SD Memory Card.

## CSV File Format

The format of the CSV file to write is as follows:

'ExecDateTi me'	ExecDateTim e							
'PrgMaxPos'	PrgMaxPos							
'PrgLoadMax Pos'	PrgLoadMax Pos							
'PrgMaxLoad'	PrgMaxLoad							
'PrgPosMaxL oad'	PrgPosMaxL oad							
'StepRsltRec. No'	'StepRsltRec. StartPos'	'StepRsltRec. EndPos'	'StepRsltRec. StartLoad'	'StepRsltRec. EndLoad'	'StepRsltRec. MaxPos'	'StepRsltRec. LoadMaxPos'	'StepRsltRec. MaxLoad'	'StepRsltRec. PosMaxLoad'
StepRsltRec [0].No	StepRsltRec [0].StartPos	StepRsltRec [0].EndPos	StepRsltRec [0].StartLoad	StepRsltRec [0].EndLoad	StepRsltRec [0].MaxPos	StepRsltRec [0].LoadMax Pos	StepRsltRec [0].MaxLoad	StepRsltRec [0].PosMaxL oad
StepRsltRec [1].No	StepRsltRec [1].StartPos	StepRsltRec [1].EndPos	StepRsltRec [1].StartLoad	StepRsltRec [1].EndLoad	StepRsltRec [1].MaxPos	StepRsltRec [1].LoadMax Pos	StepRsltRec [1].MaxLoad	StepRsltRec [1].PosMaxL oad
:	:	:	:	:	:	:	:	:
StepRsltRec [49].No	StepRsltRec [49].StartPos	StepRsltRec [49].EndPos	StepRsltRec [49].StartLoa d	StepRsltRec [49].EndLoad	StepRsltRec [49].MaxPos	StepRsltRec [49].LoadMa xPos	StepRsltRec [49].MaxLoa d	StepRsltRec [49].PosMax Load

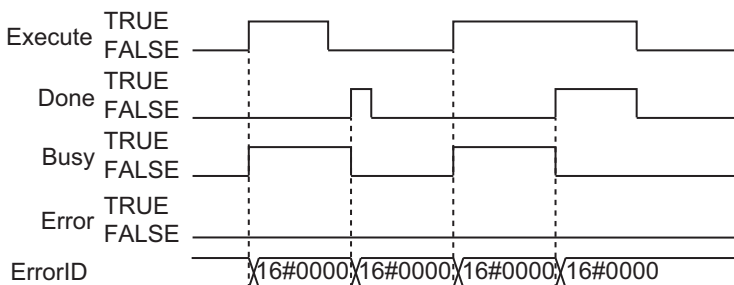
*ExecDateTime* (Record Date and Time) is converted to a text string and written with the DtToString instruction. Refer to the instructions reference manual for details on the DtToString instruction.

Data other than *ExecDateTime* is converted to text strings and written with the LrealToFormatString instruction. The total number of digits is set to eight and the fractional part is set to six digits. Refer to the instructions reference manual for details on the LrealToFormatString instruction.

## Timing Charts

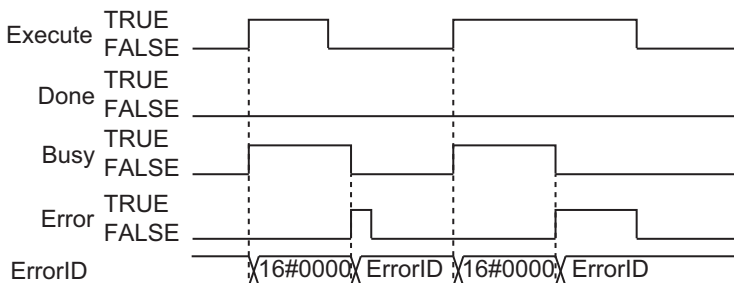
### ● Normal End

- *Busy* (Executing) changes to TRUE when *Execute* changes to TRUE.
- *Done* changes to TRUE when the data output operation is completed.



### ● Error End

- If an error occurs when execution of the function block is in progress, *Error* changes to TRUE and *Busy* (Executing) changes to FALSE.
- You can find out the cause of the error by referring to the value output by *ErrorID* (Error Code).
- If *Execute* changes to FALSE before execution of the function block is ended, *Done* and *Error* are TRUE only for one task period after execution of the function block is ended.
- If *Execute* remains TRUE even after execution of the function block is ended, the output values of *Done* and *Error* are held.



## Additional Information

To store the results of single-axis program operation into *PrgOprRsltRecorder* (Program Operation Results Recorder), use the *PrgOprRsltRec* function block on P.181.

## Precautions for Correct Use

- Execution of this function block will be continued until processing is ended even if the value of *Execute* changes to FALSE or the execution time exceeds the task period. The value of *Done* changes to TRUE when processing is ended. Use this to confirm normal ending of processing.
- Do not simultaneously access the same file. Perform exclusive control of SD Memory Card instructions in the user program.
- An error occurs in the following cases. *Error* will change to TRUE.
  - a) The SD Memory Card is not in a usable condition.
  - b) The SD Memory Card is write protected.
  - c) There is insufficient space available on the SD Memory Card.
  - d) The value of *FileName* is not a valid file name.
  - e) The maximum number of files is exceeded.
  - f) The file specified by *FileName* is being accessed.
  - g) The file specified by *FileName* is write protected.
  - h) The value of *FileName* exceeds the maximum number of characters allowed in a file name.
  - i) An error that prevents access occurs during SD Memory Card access.
- During execution of an instance, do not execute the same instance.
- Always stop the DataRecorderPut (Add Data Record) function and the DataRecorderGet (Get Data Record) function before you execute this function block. If you execute this function block without stopping them, it would take longer to write to the SD Memory Card resulting in missing data or additional errors.
- When the power supply is turned OFF to the Controller, the contents of *PrgOprRsltRecorder* (Program Operation Results Recorder) are discarded.
- Do not turn OFF the power supply to the Controller while data is written to the SD Memory Card.

## Related System-defined Variables

Variable name	Meaning	Data type	Description
_Card1Ready	SD Memory Card Ready Flag	BOOL	TRUE when the SD Memory Card is recognized. It is FALSE when an SD Memory Card is not recognized. TRUE: Card can be used. FALSE: Card cannot be used.
_Card1Protect	SD Memory Card Write Protected Flag	BOOL	This flag indicates if the SD Memory Card is write protected when it is inserted and ready to use. TRUE: Write protected. FALSE: Not write protected.
_Card1Err	SD Memory Card Error Flag	BOOL	This flag indicates if an unspecified SD Memory Card (e.g., an SDHC card) is mounted or if the format is incorrect (i.e., not FAT16 or corrupted). TRUE: There is an error. FALSE: There is no error.
_Card1Access	SD Memory Card Access Flag	BOOL	This flag indicates if the SD Memory Card is currently being accessed. TRUE: Card is being accessed. FALSE: Card is not being accessed.
_Card1PowerFail	SD Memory Card Power Interruption Flag	BOOL	This flag indicates if an error occurred in completing processing when power was interrupted during SD Memory Card access. This flag is not cleared automatically. TRUE: Error FALSE: No error.
_BackupBusy	Backup Function Busy Flag	BOOL	This flag indicates if a backup, restoration, or verification is in progress. TRUE: Backup, restore, or compare operation is in progress. FALSE: Backup, restore, or compare operation is not in progress.

## Troubleshooting

The error codes, expansion error codes, status, descriptions, and corrections given in the following table.

Error code	Expansion error code	Status	Description	Correction
16#0000	16#00000000	Normal end	---	---
16#0400	16#00000000	Input Value Out of Range	The file name specified with <i>FileName</i> contains one or more characters that cannot be used.	Set <i>FileName</i> correctly.
	16#00000000		The directory name specified with <i>FileName</i> is too long.	Check the length of the text strings specified with <i>FileName</i> so that it is within the valid range.
16#1400	16#00000000	SD Memory Card Access Failure	An SD Memory Card is either not inserted or is not inserted properly.	Insert an SD Memory Card correctly.
	16#00000000		The SD Memory Card is broken.	Replace the SD Memory Card with one that operates normally.
	16#00000000		The SD Memory Card slot is broken.	If this error persists even after making the above two corrections, replace the CPU Unit or the Industrial PC.
16#1401	16#00000000	SD Memory Card Write-protected	An attempt was made to write to a write-protected SD Memory Card.	Remove write protection from the SD Memory Card. Slide the small switch on the side of the SD Memory Card from the LOCK position to the writable position.
16#1402	16#00000000	SD Memory Card Insufficient Capacity	The SD Memory Card ran out of free space.	Replace the SD Memory Card for one with sufficient available capacity.
16#1403	16#00000000	File Does Not Exist	The specified directory does not exist.	Specify an existing directory.
16#1404	16#00000000	Too Many Files/Directories	The maximum number of files or directories was exceeded when creating a file or directory for an instruction.	Check that the number of files or directories in the SD Memory Card does not exceed the maximum number.
16#1405	16#00000000	File Already in Use	An instruction attempted to read or write a file already being accessed by another instruction.	Correct the user program so that this function block is executed only when the <i>Busy</i> output variable for all other instructions for the same file is FALSE.
16#140A	16#00000000	Write Access Denied	The file or directory specified for the function block to write is write-protected.	Remove write protection from the file or directory specified for the function block. Or, change the file name of the file to write.
16#140B	16#00000000	Too Many Files Open	The maximum number of open files was exceeded when opening a file for the function block.	Correct the user program to decrease the number of open files.
16#140D	16#00000000	File or Directory Name Is Too Long	The file name or directory name that was specified for an instruction is too long.	Check that the specified file name or directory name does not exceed the maximum length.
16#140E	16#00000000	SD Memory Card Access Failed	The SD Memory Card is broken.	Replace the SD Memory Card.
	16#00000000		The SD Memory Card slot is broken.	If this error occurs even after making the above correction, replace the CPU Unit or the Industrial PC.

## Sample Programming

This sample programming saves the program operation result record data that was created with the PrgOprRsltRec (Program Operation Result) function block sample programming to the SD Memory Card inserted in the CPU Unit in CSV format.

It is added and executed after the SP\_SingleAxisPrgOpr (Single-axis Program Operation) function block and PrgOpsRsltRec (Program Operation Result) function block sample programming.



### Precautions for Correct Use

- The sample programming shows only the portion of a program that uses the function or function block from the library.
- When you use the actual device, include user programming for device safety instructions, interlocks, I/O with other devices, and other control procedures.
- Create a user program that will produce the intended device operation.
- Check the user program for proper execution before you use it for actual operation.

## Conditions

The conditions are the same as those for the SP\_SingleAxisPrgOpr (Single-axis Program Operation) function block sample programming.

## Processing

This sample programming saves the program operation result data to the SD Memory Card with the specified file name when *PrgRsltCSVWrite\_StartPg* (SD Card Save Trigger for Program Operation Result Data) changes to TRUE.

## Ladder Diagram

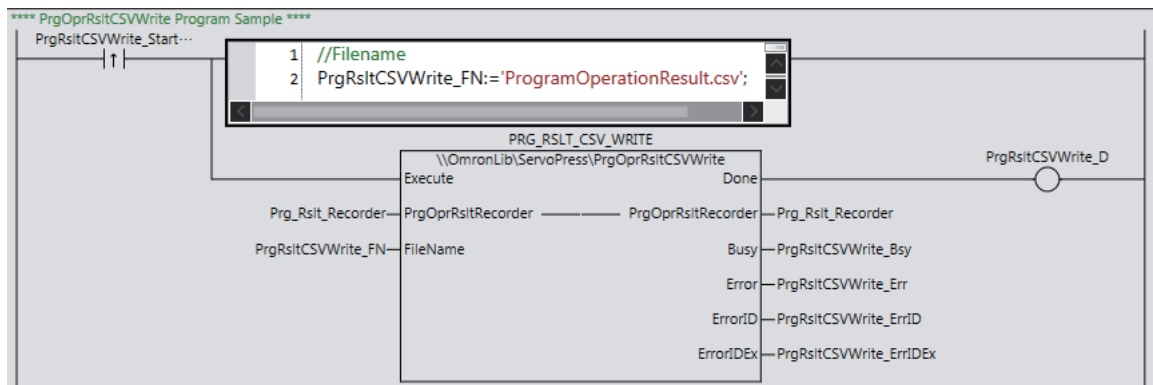
The following gives the main variables.

### ● Internal Variables

Name	Data Type	Initial Value	Comment
PRG_RSLT_CSV_WRITE	OmronLib\ServoPress\PrgOprRsltCSVWrite		Instance of the PrgOprRsltCSVWrite (Write Program Operation Results to SD Memory Card) function block
Prg_Rslt_Recorder	OmronLib\ServoPress\sPRG_OPR_RSLT_RECORDER		Program operation result record data
PrgRsltCSVWrite_StartPg	BOOL		SD card save trigger for program operation result data
PrgRsltCSVWrite_FN	STRING[66]		File name
PrgRsltCSVWrite_Bsy	BOOL		Saving data to memory card



## ● Algorithm



## ST

The following gives the main variables.

## ● Internal Variables

Name	Data Type	Initial Value	Comment
PRG_RSLT_CSV_WRITE	OmronLib\ServoPress\PrgOprRsltCSVWrite		Instance of the PrgOprRsltCSVWrite (Write Program Operation Results to SD Memory Card) function block
Prg_Rslt_Recorder	OmronLib\ServoPress\SPRG_O-PR_RSLT_RECORDER		Program operation result record data
PrgRsltCSVWrite_StartPg	BOOL		SD card save trigger for program operation result data
PrgRsltCSVWrite_FN	STRING[66]		File name
PrgRsltCSVWrite_Bsy	BOOL		Saving data to memory card

## ● Algorithm

```
//Specify the filename and start PrgRsltCSVWrite.
IF UpTrig_PrgRsltCSVWrite_StartPg=TRUE THEN
  //Filename
  PrgRsltCSVWrite_FN:='ProgramOperationResult.csv';
END_IF;

//PrgOprRsltCSVWrite
PRG_RSLT_CSV_WRITE (
  Execute := UpTrig_PrgRsltCSVWrite_StartPg,
  FileName := PrgRsltCSVWrite_FN,
  PrgOprRsltRecorder := Prg_Rslt_Recorder,
  Done => PrgRsltCSVWrite_D,
  Busy => PrgRsltCSVWrite_Bsy,
  Error => PrgRsltCSVWrite_Err,
  ErrorID => PrgRsltCSVWrite_ErrID,
  ErrorIDEx => PrgRsltCSVWrite_ErrIDEx
);
```

# PrgOprTracePut

The PrgOprTracePut function adds trace data to the program operation trace recorder 10 at a time.

Function name	Name	FB/ FUN	Graphic expression	ST expression
PrgOpr TracePut	Add Pro- gram Operation Trace Records	FUN		<pre>Out:=\\OmronLib\ServoPress\ PrgOprTracePut(   PrgOprRecord,   PrgOprTraceRecorder);</pre>

## Function Block and Function Information

Item	Description
Library file name	OmronLib_ServoPress_V1_0.slr
Namespace	OmronLib\ServoPress
Function block and function number	00082
Publish/Do not publish source code	Published.
Function block and function version	1.00

## Variables

### Input Variables

Name	Meaning	Data type	Default	Valid range	Unit	Description
EN	Execute	BOOL	FALSE	Depends on data type.	---	Execute trigger for this function Executes the function when it changes to TRUE.
PrgOpr Record	Program Operation Trace Records	Omron- Lib\Servo- Press\sPRG _OPR_RE- CORD	---	---	---	The trace records that are added to the program operation data recorder.

## ● Structure

The data type of the *PrgOprRecord* input variable is the structure `OmronLib\ServoPress\sPRG_OPR_RECORD`. The specifications are as follows:

Name	Meaning	Description	Data type	Valid range	Unit
PrgOprRecord	Program Operation Trace Records	The trace records that are added to the program operation data recorder.	OmronLib\ServoPress\sPRG_OPR_RECORD	---	---
ExecDate Time	Record Date and Time	The system time at which trace data 0 through 9 were recorded.	DATE_AND_TIME	Depends on data type.	Year, month, day, hour, minutes, seconds
TraceData [0]	Trace Data 0	Trace data 0	LREAL	Depends on data type.	---
TraceData [1]	Trace Data 1	Trace data 1	LREAL	Depends on data type.	---
TraceData [2]	Trace Data 2	Trace data 2	LREAL	Depends on data type.	---
TraceData [3]	Trace Data 3	Trace data 3	LREAL	Depends on data type.	---
TraceData [4]	Trace Data 4	Trace data 4	LREAL	Depends on data type.	---
TraceData [5]	Trace Data 5	Trace data 5	LREAL	Depends on data type.	---
TraceData [6]	Trace Data 6	Trace data 6	LREAL	Depends on data type.	---
TraceData [7]	Trace Data 7	Trace data 7	LREAL	Depends on data type.	---
TraceData [8]	Trace Data 8	Trace data 8	LREAL	Depends on data type.	---
TraceData [9]	Trace Data 9	Trace data 9	LREAL	Depends on data type.	---

## Output Variables

Name	Meaning	Data type	Valid range	Unit	Description
Out	Return Value	BOOL	Depends on data type.	---	Function execution results TRUE: Normal end FALSE: Error end

## In-Out Variables

Name	Meaning	Data type	Valid range	Unit	Description
PrgOprTrace Recorder	Program Operation Trace Recorder	OmronLib\ServoPress\sPRG_OPR_TRACE_RECORDER	---	---	The recorder of the trace of program operation.

## ● Structure

The data type of the *PrgOprTraceRecorder* in-out variable is the structure `OmronLib\ServoPress\sPRG_OPR_TRACE_RECORDER`. The specifications are as follows:

Name	Meaning	Description	Data type	Valid range	Unit	Default
PrgOprTrace Recorder	Program Operation Trace Recorder	The recorder of the trace of program operation.	OmronLib\ ServoPress\ sPRG_ OPR_TRACE_ RECORDER	---	---	---
PrgOprTrc Records	Program Operation Trace Record Array	The array containing the program operation trace records.	ARRAY [0..19999] OF sPRG_ OPR_RECORD	---	---	---
Top	First Trace Record	The index to the first trace record.	UINT	0 to 19,999	---	0
Bottom	Last Trace Record	The index to the last trace record.	UINT	0 to 19,999	---	0
Count	Trace Record Count	The number of trace records stored in the program operation trace recorder.	UINT	0 to 20,000	---	0

## Function

This function adds trace data to *PrgOprTraceRecorder* (Program Operation Trace Recorder) 10 at a time. The 10 trace data records are stored beforehand in *PrgOprRecord* (Program Operation Trace Records).

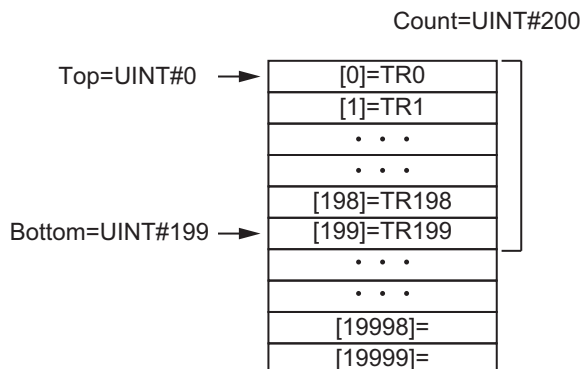
### Structure of *PrgOprTraceRecorder* (Program Operation Trace Recorder)

The 10 trace data records are placed together in *PrgOprRecord* (Program Operation Trace Records). *PrgOprTraceRecorder* (Program Operation Trace Recorder) is a data recorder that can store 20,000 *PrgOprRecord*. Therefore, *PrgOprTraceRecorder* can store a total of 200,000 trace data records.

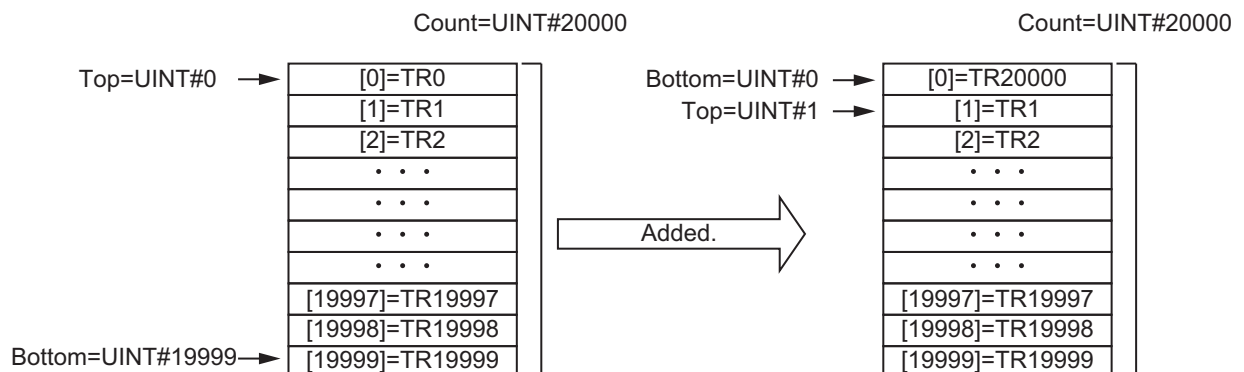
This function block stores trace data in *PrgOprTraceRecorder* in units of *PrgOprRecord*.

*Top* (First Trace Record) is the index to the first trace record stored in *PrgOprTraceRecorder*. *Bottom* (Last Trace Record) is the index to the last trace record stored in *PrgOprTraceRecorder*. *Count* (Trace Record Count) shows the number of trace records that are stored in *PrgOprTraceRecorder*.

For example, when 200 trace records are stored in *PrgOprTraceRecorder*, the values of the variables are as shown in the following figure. The trace record that is stored in *PrgOprTrcRecords[0]* is expressed as TR0 and the trace record that is stored in *PrgOprTrcRecords[1]* is expressed as TR1.



If there are already 20,000 trace records stored in *PrgOprTraceRecorder* and another trace record is added, the oldest trace record is overwritten. The values of the variables when the 20,001st trace record is added are as shown in the following figure.



When *Count* = *UINT#0*, the values of *Top* and *Bottom* are both 0.

## Additional Information

To save the contents of *PrgOprTraceRecorder* (Program Operation Trace Recorder) to an SD Memory Card in CSV format, use the PrgOprTraceCSVWrite function block on P.215.

## Precautions for Correct Use

- Do not execute this function block at the same time as the PrgOprTraceCSVWrite (Write from Program Operation Trace Recorder to SD Memory Card) function block. If you do, the trace data may not be recorded correctly.
- When the power supply is turned OFF to the Controller, the contents of *PrgOprTraceRecorder* (Program Operation Trace Recorder) are discarded.

## Sample Programming

This sample programming creates program operation trace data for the SP\_SingleAxisPrgOpr (Single-axis Program Operation) function block sample programming execution result. The trace data is specified as the *MC\_Axis000.Act.Pos* (Actual Current Position), *MC\_Axis000.Act.Vel* (Actual Current Velocity), and *MC\_Axis000.Act.Trq* (Actual Current Torque) axis variables.

It is added and executed after the SP\_SingleAxisPrgOpr (Single-axis Program Operation) function block sample programming.



### Precautions for Correct Use

- The sample programming shows only the portion of a program that uses the function or function block from the library.
- When you use the actual device, include user programming for device safety instructions, interlocks, I/O with other devices, and other control procedures.
- Create a user program that will produce the intended device operation.
- Check the user program for proper execution before you use it for actual operation.

## Conditions

The conditions are the same as those for the SP\_SingleAxisPrgOpr (Single-axis Program Operation) function block sample programming.

## Processing

If *PrgTracePut\_En* (Enable Program Operation Trace Recording) is TRUE and data was not saved to the SD Memory Card with the PrgTraceCSVWrite (Program Operation Trace Recorder SD Memory Card Write) function block, this processing records a trace of the specified data while the SP\_SingleAxisPrgOpr (Single-axis Program Operation) function block is being executed.

## Ladder Diagram

The following gives the main variables.

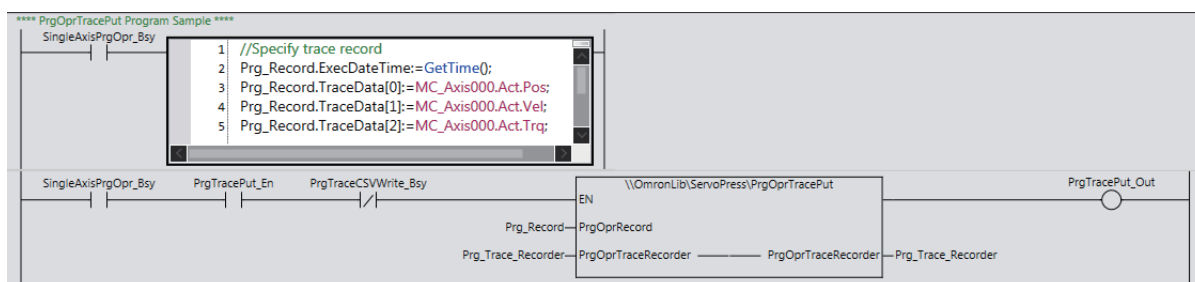
### ● Internal Variables

Name	Data Type	Initial Value	Comment
SingleAxisPrgOpr_Bsy	BOOL		Single-axis program operation busy
PrgTracePut_En	BOOL		Enables program operation trace recording.
PrgTraceCSVWrite_Bsy	BOOL		Saving data to memory card (Used in the PrgOpsTraceCSVWrite (Program Operation Trace Recorder SD Memory Card Write) function block sample programming.)
Prg_Record	OmronLib\ServoPress\PRG_O-PR_RECORD		Program operation trace data (1 record)
Prg_Trace_Recorder	OmronLib\ServoPress\PRG_O-PR_TRACE_RECORDER		Program operation trace data

### ● External Variables

Name	Data Type	Initial Value	Comment
MC_Axis000	_sAXIS_REF	---	Servo axis

### ● Algorithm



## ST

The following gives the main variables.

### ● Internal Variables

Name	Data Type	Initial Value	Comment
SingleAxisPrgOpr_Bsy	BOOL		Single-axis program operation busy
PrgTracePut_En	BOOL		Enables program operation trace recording.
PrgTraceCSVWrite_Bsy	BOOL		Saving data to memory card (Used in the PrgOpsTraceCSVWrite (Program Operation Trace Recorder SD Memory Card Write) function block sample programming.)
Prg_Record	OmronLib\ServoPress\sPRG_OPR_RECORD		Program operation trace data (1 record)
Prg_Trace_Recorder	OmronLib\ServoPress\sPRG_OPR_TRACE_RECORDER		Program operation trace data

### ● External Variables

Name	Data Type	Initial Value	Comment
MC_Axis000	_sAXIS_REF	---	Servo axis

### ● Algorithm

```
//Specify trace record.
IF SingleAxisPrgOpr_Bsy=TRUE THEN
  Prg_Record.ExecDateTime:=GetTime();
  Prg_Record.TraceData[0]:=MC_Axis000.Act.Pos;
  Prg_Record.TraceData[1]:=MC_Axis000.Act.Vel;
  Prg_Record.TraceData[2]:=MC_Axis000.Act.Trq;
END_IF;

//PrgOprTracePut
//Start on PrgOprTraceCSVWrite not in progress.
PrgTracePut_Out:=\OmronLib\ServoPress\PrgOprTracePut(
  EN := SingleAxisPrgOpr_Bsy AND PrgTracePut_En AND NOT(PrgTraceCSVWrite_Bsy),
  PrgOprRecord := Prg_Record,
  PrgOprTraceRecorder := Prg_Trace_Recorder
);
```



# PrgOprTracePut2

The PrgOprTracePut2 function adds trace data to the program operation trace recorder 10 at a time. This function allows you to define the recorder length to any length that you want.

FB/FUN name	Name	FB/FUN	Graphic expression	ST expression
PrgOprTracePut2	Add Program Operation Trace Records 2	FUN		<pre>Out:=\\OmronLib\ServoPress \PrgOprTracePut2( PrgOprRecord, PrgOprTraceRecorderInfo, PrgOprTraceRecorder);</pre>

## Function Block and Function Information

Item	Description
Library file name	OmronLib_ServoPress_V2_0.slr
Namespace	OmronLib\ServoPress
FUN/FB number	00223
Source code	Not published.

## Variables

### Input Variables

Variable	Name	Data type	Initial value	Valid range	Unit	Description
EN	Execute	BOOL	FALSE	Depends on data type.	-	The processing is executed when the variable changes to TRUE.
PrgOpr Record	Program Operation Trace Records	OmronLib\ServoPress\sPRG_OPR_RECORD	-	-	-	The trace records that are added to the program operation data recorder.

## ● Structure

The data type of the *PrgOprRecord* input variable is structure `OmronLib\ServoPress\sPRG_O-PR_RECORD`. The specifications are as follows:

Name	Meaning	Description	Data type	Valid range	Unit
PrgOpr Record	Program Operation Trace Records	The trace records that are added to the program operation data recorder.	OmronLib\ServoPress\sPRG_O-PR_RECORD	-	-
ExecDate Time	Record Date and Time	The system time at which trace data 0 through 9 were recorded.	DATE_AND_TIME	Depends on data type.	Year, month, day, hour, minutes, seconds
TraceData [0]	Trace Data 0	Trace Data 0	LREAL	Depends on data type.	-
TraceData [1]	Trace Data 1	Trace Data 1	LREAL	Depends on data type.	-
TraceData [2]	Trace Data 2	Trace Data 2	LREAL	Depends on data type.	-
TraceData [3]	Trace Data 3	Trace Data 3	LREAL	Depends on data type.	-
TraceData [4]	Trace Data 4	Trace Data 4	LREAL	Depends on data type.	-
TraceData [5]	Trace Data 5	Trace Data 5	LREAL	Depends on data type.	-
TraceData [6]	Trace Data 6	Trace Data 6	LREAL	Depends on data type.	-
TraceData [7]	Trace Data 7	Trace Data 7	LREAL	Depends on data type.	-
TraceData [8]	Trace Data 8	Trace Data 8	LREAL	Depends on data type.	-
TraceData [9]	Trace Data 9	Trace Data 9	LREAL	Depends on data type.	-

## Output Variables

Name	Meaning	Data type	Valid range	Unit	Description
Out	Return Value	BOOL	Depends on data type.	-	Function execution results TRUE: Normal end FALSE: Error end

## In-Out Variables

Name	Meaning	Data type	Valid range	Unit	Description
PrgOprTraceRecorderInfo	Program Operation Trace Recorder Information	OmronLib\ServoPress\sPRG_OPR_TRACE_RECORDERINFO	Depends on data type.	-	Management information on the program operation trace recorder.
PrgOprTraceRecorder[]*1	Program Operation Trace Recorder	ARRAY[*] OF OmronLib\ServoPress\sPRG_OPR_RECORD	Depends on data type.	-	The recorder of the trace of program operation.

\*1. The maximum number of array elements is 20,000. The first number of array element should be 0.

### ● Structure

The data type of the *PrgOprTraceRecorderInfo* in-out variable is the structure OmronLib\ServoPress\sPRG\_OPR\_TRACE\_RECORDERINFO. The specifications are as follows:

Name	Meaning	Description	Data type	Valid range	Unit
PrgOprTraceRecorderInfo	Program Operation Trace Recorder Information	Management information on the program operation data recorder	OmronLib\ServoPress\sPRG_OPR_TRACE_RECORDERINFO	-	-
Top	First Trace Record	The index to the first trace record	UINT	0 to 19,999	-
Bottom	Last Trace Record	The index to the last trace record	UINT	0 to 19,999	-
Count	Trace Record Count	The number of trace records stored in the program operation trace recorder	UINT	0 to 20,000	-

## Function

This function adds trace data to *PrgOprTraceRecorder* (Program Operation Trace Recorder) 10 at a time. The 10 trace data records are stored beforehand in *PrgOprRecord* (Program Operation Trace Records).

### Structure of *PrgOprTraceRecorder* (Program Operation Trace Recorder)

The 10 trace data records are placed together in *PrgOprRecord* (Program Operation Trace Records). *PrgOprTraceRecorder[]* (Program Operation Trace Recorder) is a data recorder that can store up to 20,000 *PrgOprRecord*. Therefore, *PrgOprTraceRecorder[]* can store a total of 200,000 trace data records.

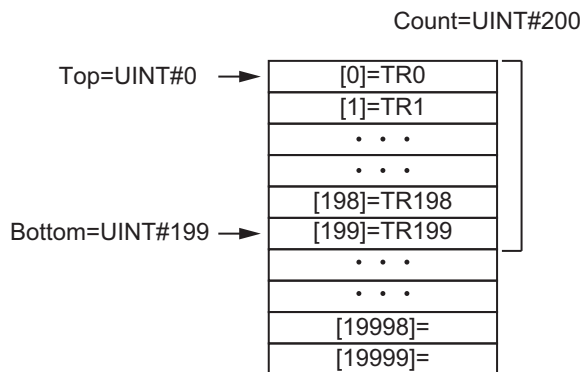
This function stores trace data in *PrgOprTraceRecorder[]* in units of *PrgOprRecord*.

The maximum recorder length can be adjusted by defining the number of *PrgOprTraceRecorder[]* array elements.

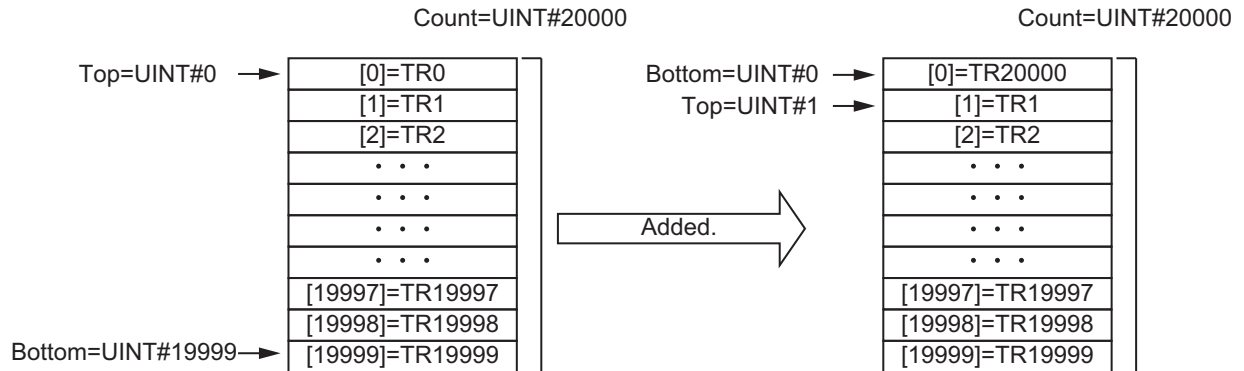
*PrgOprTraceRecorderInfo.Top* (First Trace Record) is the index to the first trace record stored in *PrgOprTraceRecorder[]*. *PrgOprTraceRecorderInfo.Bottom* (Last Trace Record) is the index to the last trace record stored in *PrgOprTraceRecorder[]*. *PrgOprTraceRecorderInfo.Count* (Trace Record Count) shows the number of trace records that are stored in *PrgOprTraceRecorder[]*.

For example, when 200 trace records are stored in *PrgOprTraceRecorder[]* with 20,000 array elements, the values of the variables are as shown in the following figure.

The trace record that is stored in *PrgOprTrcRecords[0]* is expressed as TR0 and the trace record that is stored in *PrgOprTrcRecords[1]* is expressed as TR1.



If the trace records stored in *PrgOprTraceRecorder[]* are recorded to the end of its array elements, and another trace record is added, the oldest trace record is overwritten. The values of the variables when a 20,001st trace record is added to *PrgOprTraceRecorder[]* with 20,000 array elements are as shown in the following figure.



When *Count* = *UINT#0*, the values of *Top* and *Bottom* are both 0.

## Additional Information

To save the contents of *PrgOprTraceRecorder2* (Program Operation Trace Recorder 2) to an SD Memory Card in CSV format, use the function block of *PrgOprTraceCSVWrite2* on page 225.

## Precautions for Correct Use

- Do not execute this function at the same time as the *PrgOprTraceCSVWrite2* (Write from Program Operation Trace Recorder to SD Memory Card 2) function block. If you do, the trace data may not be recorded correctly.
- When the power supply is turned OFF to the Controller, the contents of *PrgOprTraceRecorder* (Program Operation Trace Recorder) are discarded.

## Sample Programming

This sample programming creates program operation trace data for the *SP\_SingleAxisPrgOpr* (Single-axis Program Operation) function block sample programming execution result. The trace data is specified as the *MC\_Axis000.Act.Pos* (Actual Current Position), *MC\_Axis000.Act.Vel* (Actual Current Velocity), and *MC\_Axis000.Act.Trq* (Actual Current Torque) axis variables.

It is added and executed after the *SP\_SingleAxisPrgOpr* (Single-axis Program Operation) function block sample programming.



### Precautions for Correct Use

- The sample programming shows only the portion of a program that uses the function or function block from the library.
- When you use the actual device, include user programming for device safety instructions, interlocks, I/O with other devices, and other control procedures.
- Create a user program so that the actual device operates as intended.
- Check the user program for proper execution before you use it for actual operation.

## Condition

---

The conditions are the same as those for the SP\_SingleAxisPrgOpr (Single-axis Program Operation) function block sample programming.

## Processing

---

If *PrgTracePut\_En* (Enable Program Operation Trace Recording) is TRUE and data was not saved to the SD Memory Card with the PrgTraceCSVWrite2 (Program Operation Trace Recorder SD Memory Card Write 2) function block, this processing records a trace of the data specified while the SP\_SingleAxisPrgOpr (Single-axis Program Operation) function block is being executed.

## Ladder Diagram

The following gives the main variables.

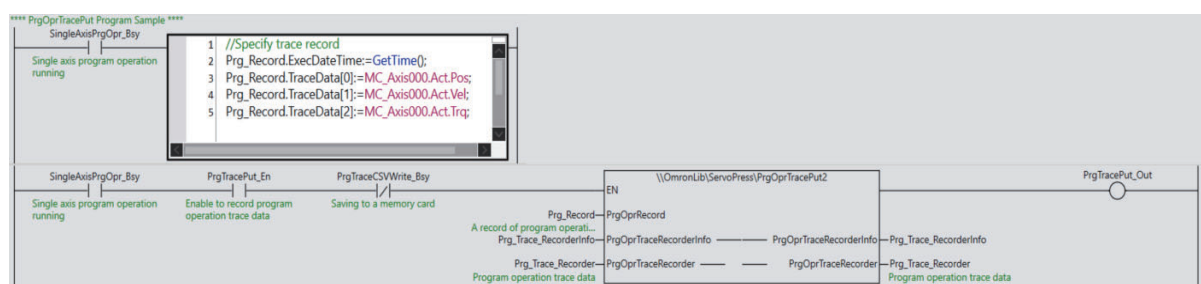
### ● Internal Variables

Name	Data type	Initial value	Comment
SingleAxisPrgOpr_Bsy	BOOL		Single-axis program operation busy
PrgTracePut_En	BOOL		Enables program operation trace recording.
PrgTraceCSVWrite_Bsy	BOOL		Saving data to memory card (Used in PrgOpr-TraceCSVWrite2 (Program Operation Trace Recorder SD Memory Card Write 2) function block sample programming)
Prg_Record	OmronLib\ServoPress \sPRG_OPR_RECORD		Program operation trace data (1 record)
Prg_Trace_Recorder	ARRAY[0..19999] OF Omron- Lib\ServoPress \sPRG_OPR_RECORD		Program operation trace data
Prg_Trace_RecorderInfo	OmronLib\ServoPress \sPRG_OPR_TRACE_RECORDER- ERINFO		Management information on trace recorder

### ● External Variables

Name	Data type	Initial value	Comment
MC_Axis000	_sAXIS_REF	-	Servo axis

### ● Algorithm



## ST

The following gives the main variables.

### ● Internal Variables

Name	Data type	Initial value	Comment
SingleAxisPrgOpr_Bsy	BOOL		Single-axis program operation busy
PrgTracePut_En	BOOL		Enables program operation trace recording.
PrgTraceCSVWrite_Bsy	BOOL		Saving data to memory card (Used in PrgOpr-TraceCSVWrite2 (Program Operation Trace Recorder SD Memory Card Write 2) function block sample programming)
Prg_Record	OmronLib\ServoPress \sPRG_OPR_RECORD		Program operation trace data (1 record)
Prg_Trace_Recorder	ARRAY[0..19999] OF Omron-Lib\ServoPress \sPRG_OPR_RECORD		Program operation trace data
Prg_Trace_RecorderInfo	OmronLib\ServoPress \sPRG_OPR_TRACE_RECORDERINFO		Management information on trace recorder

### ● External Variables

Name	Data type	Initial value	Comment
MC_Axis000	_sAXIS_REF	-	Servo axis

### ● Algorithm

```
//Specify trace record.
IF SingleAxisPrgOpr_Bsy=TRUE THEN
  Prg_Record.ExecDateTime:=GetTime();
  Prg_Record.TraceData[0]:=MC_Axis000.Act.Pos;
  Prg_Record.TraceData[1]:=MC_Axis000.Act.Vel;
  Prg_Record.TraceData[2]:=MC_Axis000.Act.Trq;
END_IF;

//PrgOprTracePut
//Start on PrgOprTraceCSVWrite not in progress.
PrgTracePut_Out:=\\OmronLib\ServoPress\PrgOprTracePut2(
  EN := SingleAxisPrgOpr_Bsy AND PrgTracePut_En AND NOT(PrgTraceCSVWrite_Bsy),
  PrgOprRecord := Prg_Record,
  PrgOprTraceRecorderInfo:=Prg_Trace_RecorderInfo,
  PrgOprTraceRecorder := Prg_Trace_Recorder
);
```



# PrgOprTraceCSVWrite

The PrgOprTraceCSVWrite function block writes the contents of the program operation trace recorder to an SD Memory Card in CSV format.

Function block name	Name	FB/FUN	Graphic expression	ST expression
PrgOprTraceCSVWrite	Write from Program Operation Trace Recorder to SD Memory Card	FB		PrgOprTraceCSVWrite_instance ( Execute, PrgOprTraceRecorder, FileName, WriteLineNum, Done, Busy, Error, ErrorID, ErrorIDEx);

## Function Block and Function Information

Item	Description
Library file name	OmronLib_ServoPress_V1_0.slr
Namespace	OmronLib\ServoPress
Function block and function number	00098
Publish/Do not publish source code	Published.
Function block and function version	1.00

## Variables

### Input Variables

Name	Meaning	Data type	Default	Valid range	Unit	Description
Execute	Execute	BOOL	FALSE	Depends on data type.	---	Execute trigger for this function block Executes the function block when it changes to TRUE.
FileName	File Name	STRING	"	66 bytes max. (65 single-byte alphanumeric characters plus the final NULL character)	---	File name of CSV file to write.
WriteLine Num	Number of Lines to Write	USINT	1	1 to 255	---	The number of lines that are written to the CSV file in one processing.

## Output Variables

Name	Meaning	Data type	Valid range	Unit	Description
Done	Done	BOOL	Depends on data type.	---	Completion of execution TRUE: Normal end FALSE: Error end, execution in progress, or execution condition not met.
Busy	Executing	BOOL	Depends on data type.	---	Executing TRUE: Executing FALSE: Not executing
Error	Error	BOOL	Depends on data type.	---	Error end TRUE: Error end FALSE: Normal end, executing, or execution conditions not met
ErrorID	Error Code	WORD	*1	---	An error code is output if an error occurs. The value is WORD#16#0 for a normal end.
ErrorIDEx	Expansion Error Code	DWORD	*1	---	An expansion error code is output if an error occurs. The value is DWORD#16#0 for a normal end.

\*1. For details, refer to *Troubleshooting* on page 221.

## In-Out Variables

Name	Meaning	Data type	Valid range	Unit	Description
PrgOprTraceRecorder	Program Operation Trace Recorder	OmronLib\ ServoPress\ sPRG_ OPR_TRACE_ RECORDER	---	---	The recorder of the trace of program operation.

### ● Structure

The data type of the *PrgOprTraceRecorder* in-out variable is the structure `OmronLib\ServoPress\sPRG_OPR_TRACE_RECORDER`. Refer to *Structure* in *PrgOprTracePut* on P.200 for details.

## Function

This function block writes all the contents of *PrgOprTraceRecorder* (Program Operation Trace Recorder) to an SD Memory Card in CSV format. The name of the file to write is specified with *FileName*.

With *FileName*, you can specify the name including the folder. If the specified folder does not exist, an error occurs. If the folder is not specified, create *FileName* in the root of the SD Memory Card.

Refer to *Structure* in *PrgOprTracePut* on page 200 for further details on the *PrgOprTraceRecorder* structure.

## CSV File Format

The format of the CSV file to write is as follows:

'ExecDateTime'	'TraceData[0]'	...	'TraceData[9]'
ProgOprTrcRecords[0]. ExecDateTime	ProgOprTrcRecords[0]. TraceData[0]	...	ProgOprTrcRecords[0]. TraceData[9]
:	:	...	:
ProgOprTrcRecords[19999]. ExecDateTime	ProgOprTrcRecords[19999]. TraceData[0]	...	ProgOprTrcRecords[19999]. TraceData[9]

*ExecDateTime* (Record Date and Time) is converted to a text string and written with the *DtToString* instruction. Refer to the instructions reference manual for details on the *DtToString* instruction.

*TraceData* is converted to text strings and written with the *LrealToFormatString* instruction. The total number of digits is set to eight and the fractional part is set to six digits. Refer to the instructions reference manual for details on the *LrealToFormatString* instruction.

## Setting *WriteLineNum* (Number of Lines to Write)

The value of *WriteLineNum* (Number of Lines to Write) is used to specify the number of lines to write to the SD Memory Card.

This function block uses the *FileWrite* (Write File) instruction to write the CSV file.

*WriteLineNum* (Number of Lines to Write) is used to specify the number of CSV file lines to write in the *FileWrite* (Write File) instruction that will be created in one task period.

The smaller the value of *WriteLineNum*, the shorter the execution time for this function block in one task period. However, because this results in the function block being executed over more task periods, the time from the start of execution until the end of execution increases.

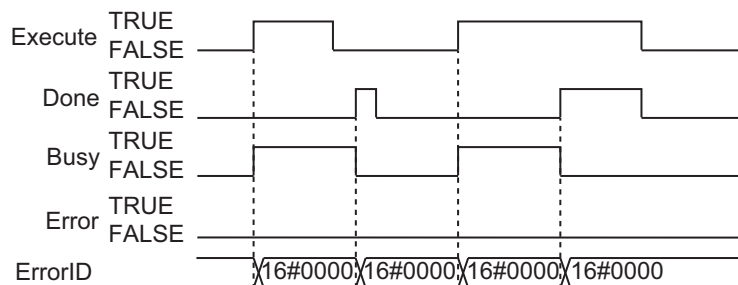
The larger the value of *WriteLineNum*, the longer the execution time for this function block in one task period. However, because this results in the function block being executed over fewer task periods, the time from the start of execution until the end of execution decreases.

Set the value of *WriteLineNum* to match the task period for the task in which this function block is executed.

## Timing Charts

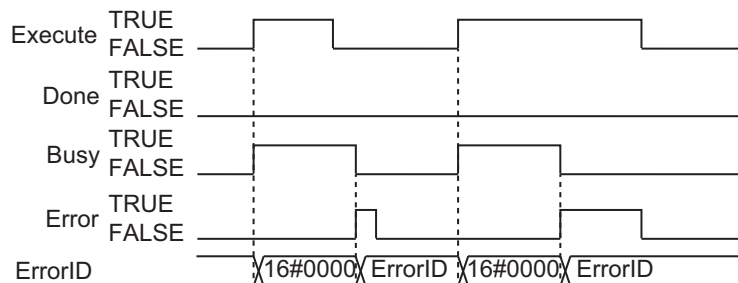
### ● Normal End

- *Busy* (Executing) changes to TRUE when *Execute* changes to TRUE.
- *Done* changes to TRUE when the data output operation is completed.



### ● Error End

- If an error occurs when execution of the function block is in progress, *Error* changes to TRUE and *Busy* (Executing) changes to FALSE.
- You can find out the cause of the error by referring to the value output by *ErrorID* (Error Code).
- If *Execute* changes to FALSE before execution of the function block is ended, *Done* and *Error* are TRUE only for one task period after execution of the function block is ended.
- If *Execute* remains TRUE even after execution of the function block is ended, the output values of *Done* and *Error* are held.



## Additional Information

To add trace data to *PrgOprTraceRecorder* (Program Operation Trace Recorder), use the *PrgOprTracePut* function block on P.200.

## Precautions for Correct Use

- Execution of this function block will be continued until processing is ended even if the value of *Execute* changes to FALSE or the execution time exceeds the task period. The value of *Done* changes to TRUE when processing is ended. Use this to confirm normal ending of processing.
- Do not simultaneously access the same file. Perform exclusive control of SD Memory Card instructions in the user program.
- An error occurs in the following cases. *Error* will change to TRUE.
  - a) The SD Memory Card is not in a usable condition.
  - b) The SD Memory Card is write protected.
  - c) There is insufficient space available on the SD Memory Card.
  - d) The value of *FileName* is not a valid file name.
  - e) The maximum number of files is exceeded.
  - f) The file specified by *FileName* is being accessed.
  - g) The file specified by *FileName* is write protected.
  - h) The value of *FileName* exceeds the maximum number of characters allowed in a file name.
  - i) An error that prevents access occurs during SD Memory Card access.
- During execution of an instance, do not execute the same instance.
- Always stop the DataRecorderPut (Add Data Record) function and the DataRecorderGet (Get Data Record) function before you execute this function block. If you execute this function block without stopping them, it would take longer to write to the SD Memory Card resulting in missing data or additional errors.
- When the power supply is turned OFF to the Controller, the contents of *PrgOprTraceRecorder* (Program Operation Trace Recorder) are discarded.
- Do not turn OFF the power supply to the Controller while data is written to the SD Memory Card.

## Related System-defined Variables

Variable name	Meaning	Data type	Description
_Card1Ready	SD Memory Card Ready Flag	BOOL	TRUE when the SD Memory Card is recognized. It is FALSE when an SD Memory Card is not recognized. TRUE: Card can be used. FALSE: Card cannot be used.
_Card1Protect	SD Memory Card Write Protected Flag	BOOL	This flag indicates if the SD Memory Card is write protected when it is inserted and ready to use. TRUE: Write protected. FALSE: Not write protected.
_Card1Err	SD Memory Card Error Flag	BOOL	This flag indicates if an unspecified SD Memory Card (e.g., an SDHC card) is mounted or if the format is incorrect (i.e., not FAT16 or corrupted). TRUE: There is an error. FALSE: There is no error.
_Card1Access	SD Memory Card Access Flag	BOOL	This flag indicates if the SD Memory Card is currently being accessed. TRUE: Card is being accessed. FALSE: Card is not being accessed.
_Card1PowerFail	SD Memory Card Power Interruption Flag	BOOL	This flag indicates if an error occurred in completing processing when power was interrupted during SD Memory Card access. This flag is not cleared automatically. TRUE: Error FALSE: No error.
_BackupBusy	Backup Function Busy Flag	BOOL	This flag indicates if a backup, restoration, or verification is in progress. TRUE: Backup, restore, or compare operation is in progress. FALSE: Backup, restore, or compare operation is not in progress.

## Troubleshooting

The error codes, expansion error codes, status, descriptions, and corrections given in the following table.

Error code	Expansion error code	Status	Description	Correction
16#0000	16#00000000	Normal end	---	---
16#0400	16#00000000	Input Value Out of Range	The file name specified with <i>FileName</i> contains one or more characters that cannot be used.	Set <i>FileName</i> correctly.
	16#00000000		The directory name specified with <i>FileName</i> is too long.	Check the length of the text strings specified with <i>FileName</i> so that it is within the valid range.
16#1400	16#00000000	SD Memory Card Access Failure	An SD Memory Card is either not inserted or is not inserted properly.	Insert an SD Memory Card correctly.
	16#00000000		The SD Memory Card is broken.	Replace the SD Memory Card with one that operates normally.
	16#00000000		The SD Memory Card slot is broken.	If this error persists even after making the above two corrections, replace the CPU Unit or the Industrial PC.
16#1401	16#00000000	SD Memory Card Write-protected	An attempt was made to write to a write-protected SD Memory Card.	Remove write protection from the SD Memory Card. Slide the small switch on the side of the SD Memory Card from the LOCK position to the writable position.
16#1402	16#00000000	SD Memory Card Insufficient Capacity	The SD Memory Card ran out of free space.	Replace the SD Memory Card for one with sufficient available capacity.
16#1403	16#00000000	File Does Not Exist	The specified directory does not exist.	Specify an existing directory.
16#1404	16#00000000	Too Many Files/Directories	The maximum number of files or directories was exceeded when creating a file or directory for an instruction.	Check that the number of files or directories in the SD Memory Card does not exceed the maximum number.
16#1405	16#00000000	File Already in Use	An instruction attempted to read or write a file already being accessed by another instruction.	Correct the user program so that this function block is executed only when the <i>Busy</i> output variable for all other instructions for the same file is FALSE.
16#140A	16#00000000	Write Access Denied	The file or directory specified for the function block to write is write-protected.	Remove write protection from the file or directory specified for the function block. Or, change the file name of the file to write.
16#140B	16#00000000	Too Many Files Open	The maximum number of open files was exceeded when opening a file for the function block.	Correct the user program to decrease the number of open files.
16#140D	16#00000000	File or Directory Name Is Too Long	The file name or directory name that was specified for an instruction is too long.	Check that the specified file name or directory name does not exceed the maximum length.
16#140E	16#00000000	SD Memory Card Access Failed	The SD Memory Card is broken.	Replace the SD Memory Card.
	16#00000000		The SD Memory Card slot is broken.	If this error occurs even after making the above correction, replace the CPU Unit or the Industrial PC.

Error code	Expansion error code	Status	Description	Correction
16#3CA5	16#00000001	First Data Position Specification Error	The value of <i>Top</i> (First Trace Record) is outside the valid range for the array.	Set the value of <i>Top</i> (First Trace Record) within the valid range for the array.
	16#00000002	Last Data Position Specification Error	The value of <i>Bottom</i> (Last Trace Record) is outside the valid range for the array.	Set the value of <i>Bottom</i> (Last Trace Record) within the valid range for the array.
	16#00000003	No Data Stored in Data Recorder	There are no program operation trace records ( <i>Count</i> = UINT#0) in <i>PrgOprTraceRecorder</i> (Program Operation Trace Recorder).	Make sure that program operation trace records are stored in <i>PrgOprTraceRecorder</i> (Program Operation Trace Recorder).
	16#00000004	Data Recorder Storage Information Error	There is a conflict among the values of <i>Top</i> (First Trace Record), <i>Bottom</i> (Last Trace Record), and <i>Count</i> (Trace Record Count).	Check the values of <i>Top</i> (First Trace Record), <i>Bottom</i> (Last Trace Record), and <i>Count</i> (Trace Record Count).
	16#00000005	Number of Lines to Write Out of Range	The value of <i>WriteLineNum</i> (Number of Lines to Write) is outside the valid range.	Check the valid range of the value for <i>WriteLineNum</i> (Number of Lines to Write) and set the value within the valid range.

## Sample Programming

This sample programming saves the program operation trace data that was created with the PrgOprTracePut (Add Program Operation Trace Records) function sample programming to the SD Memory Card inserted in the CPU Unit in CSV format.

It is added and executed after the SP\_SingleAxisPrgOpr (Single-axis Program Operation) function block and PrgOprTracePut (Add Program Operation Trace Records) function sample programming.



### Precautions for Correct Use

- The sample programming shows only the portion of a program that uses the function or function block from the library.
- When you use the actual device, include user programming for device safety instructions, interlocks, I/O with other devices, and other control procedures.
- Create a user program that will produce the intended device operation.
- Check the user program for proper execution before you use it for actual operation.

## Conditions

The conditions are the same as those for the SP\_SingleAxisPrgOpr (Single-axis Program Operation) function block sample programming.

## Processing

The program operation trace data is saved to the SD Memory Card with the specified file name when *PrgTraceCSVWrite\_StartPg* (SD Card Save Trigger for Program Operation Trace Data) changes to TRUE.



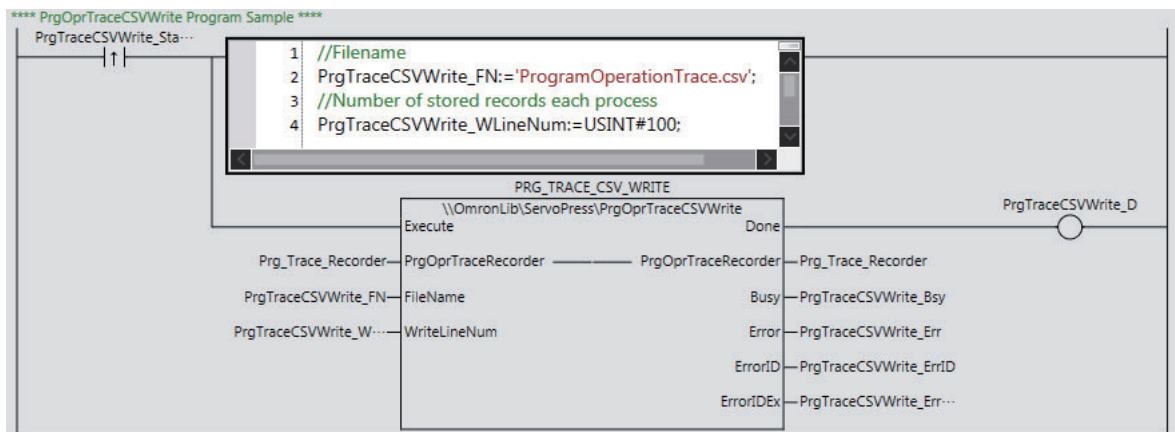
## Ladder Diagram

The following gives the main variables.

### ● Internal Variables

Name	Data Type	Initial Value	Comment
PRG_TRACE_CSV_WRITE	OmronLib\ServoPress\ PrgOprTraceCSVWrite		Instance of the PrgOpr- TraceCSVWrite (Write from Program Operation Trace Recorder to SD Memory Card) function block
Prg_Trace_Recorder	OmronLib\ServoPress\sPRG_ OPR_TRACE_RECORDER		Program operation trace data
PrgTraceCSVWrite_StartPg	BOOL		SD card save trigger for pro- gram operation trace data
PrgTraceCSVWrite_FN	STRING[66]		File name
PrgTraceCSVWrite_Bsy	BOOL		Saving data to memory card

### ● Algorithm



## ST

The following gives the main variables.

### ● Internal Variables

Name	Data Type	Initial Value	Comment
PRG_TRACE_CSV_WRITE	OmronLib\ServoPress\ PrgOprTraceCSVWrite		Instance of the PrgOpr- TraceCSVWrite (Write from Program Operation Trace Recorder to SD Memory Card) function block
Prg_Trace_Recorder	OmronLib\ServoPress\sPRG_ OPR_TRACE_RECORDER		Program operation trace data
PrgTraceCSVWrite_StartPg	BOOL		SD card save trigger for pro- gram operation trace data
PrgTraceCSVWrite_FN	STRING[66]		File name
PrgTraceCSVWrite_Bsy	BOOL		Saving data to memory card

### ● Algorithm

```
//Specify the filename and start PrgTraceCSVWrite.
IF UpTrig_PrgTraceCSVWrite_StartPg=TRUE THEN
  //Filename
  PrgTraceCSVWrite_FN:='ProgramOperationTrace.csv';
END_IF;

//PrgOprTraceCSVWrite
PRG_TRACE_CSV_WRITE(
  Execute := UpTrig_PrgTraceCSVWrite_StartPg,
  FileName := PrgTraceCSVWrite_FN,
  WriteLineNum := PrgTraceCSVWrite_WLineNum,
  PrgOprTraceRecorder := Prg_Trace_Recorder,
  Done => PrgTraceCSVWrite_D,
  Busy => PrgTraceCSVWrite_Bsy,
  Error => PrgTraceCSVWrite_Err,
  ErrorID => PrgTraceCSVWrite_ErrID,
  ErrorIDEx => PrgTraceCSVWrite_ErrIDEx
);
```

# PrgOprTraceCSVWrite2

The PrgOprTraceCSVWrite2 function block writes the contents of the program operation trace recorder to an SD Memory Card in CSV format. This function block writes to recorders of the variable-length array data type created by PrgOprTracePut2.

FB/FUN name	Name	FB/FUN	Graphic expression	ST expression
PrgOpr-TraceCSVWrite2	Write from Program Operation Trace Recorder to SD Memory Card 2	FB		PrgOprTraceCSVWrite2_instance(Execute, PrgOprTraceRecorderInfo, PrgOprTraceRecorder, FileName, WriteLineNum, Done, Busy, Error, ErrorID, ErrorIDEx);

## Function Block and Function Information

Item	Description
Library file name	OmronLib_ServoPress_V2_0.slr
Namespace	OmronLib\ServoPress
FUN/FB number	00224
Source code	Not published.

## Variables

### Input Variables

Name	Meaning	Data type	Initial value	Valid range	Unit	Description
Execute	Execute	BOOL	FALSE	Depends on data type.	-	Execution trigger for this function block Executes the function block when it changes to TRUE.
FileName	File Name	STRING	"	66 bytes max. (65 single-byte alphanumeric characters plus the final NULL character)	-	File name of CSV file to write
WriteLineNum	Number of Lines to Write	USINT	1	1 to 255	-	The number of lines that are written to the CSV file in one processing

## Output Variables

Name	Meaning	Data type	Valid range	Unit	Description
Done	Done	BOOL	Depends on data type.	-	Completion of execution TRUE: Normal end FALSE: Error end, execution in progress, or execution condition not met
Busy	Executing	BOOL	Depends on data type.	-	Executing TRUE: Executing FALSE: Not executing.
Error	Error	BOOL	Depends on data type.	-	Error end TRUE: Error end FALSE: Normal end, execution in progress, or execution condition not met
ErrorID	Error Code	WORD	*1	-	An error code is output if an error occurs. The value is WORD#16#0 for a normal end.
ErrorIDEx	Expansion Error Code	DWORD	*1	-	An expansion error code is output if an error occurs. The value is DWORD#16#0 for a normal end.

\*1. Refer to *Troubleshooting* on page 231 for details.

## In-Out Variables

Name	Meaning	Data type	Valid range	Unit	Description
PrgOprTraceRecorderInfo	Program Operation Trace Recorder Information	OmronLib\ServoPress\sPRG_OPR_TRACE_RECORDERINFO	Depends on data type.	-	Management information on the program operation trace recorder.
PrgOprTraceRecorder[]*1	Program Operation Trace Recorder	ARRAY[*] OF OmronLib\ServoPress\sPRG_OPR_RECORD	Depends on data type.	-	The recorder of the trace of program operation.

\*1. The maximum number of array elements is 20,000. The first number of array element should be 0.

### ● Structure

The data type of the *PrgOprTraceRecorder[]* in-out variable is the structure *OmronLib\ServoPress\sPRG\_OPR\_TRACE\_RECORD*. Refer to *Structure* in *PrgOprTracePut2* on page 207 for details.

The data type of the *PrgOprTraceRecorderInfo* in-out variable is the structure *OmronLib\ServoPress\sPRG\_OPR\_TRACE\_RECORDERINFO*. Refer to *Structure* in *PrgOprTracePut2* on page 207 for details.

## Function

This function block writes all the contents of *PrgOprTraceRecorder[]* (Program Operation Trace Recorder) to an SD Memory Card in CSV format. The name of the file to write is specified with *FileName* (File Name).

With *FileName*, you can specify the name including the folder. If the specified folder does not exist, an error occurs.

If the folder is not specified, the file is created in the root directory of the SD Memory Card.

Refer to Structure in *PrgOprTracePut2* on page 207 for further details on the *PrgOprTraceRecorder[]* structure.

## CSV File Format

For *PrgOprTraceRecorder[]* with 20,000 (0 to 19,999) array elements, the format of the CSV file to write is as follows.

'ExecDateTime'	'TraceData[0]'	...	'TraceData[9]'
PrgOprTraceRecorder[0]. ExecDateTime	PrgOprTraceRecorder[0]. TraceData[0]	...	PrgOprTraceRecorder[0]. TraceData[9]
:	:	...	:
PrgOprTraceRecorder[19999]. ExecDateTime	PrgOprTraceRecorder[19999]. TraceData[0]	...	PrgOprTraceRecorder[19999]. TraceData[9]

*ExecDateTime* (Record Date and Time) is converted to a text string and written with the DtToString instruction. Refer to the instructions reference manual for details on the DtToString instruction.

*TraceData[]* (Trace Data) is converted to text strings and written with the LrealToFormatString instruction. The total number of digits is set to eight and the fractional part is set to six digits. Refer to the instructions reference manual for details on the LrealToFormatString instruction.

## Setting WriteLineNum (Number of Lines to Write)

The value of *WriteLineNum* (Number of Lines to Write) is used to specify the number of lines to write to the SD Memory Card.

This function block uses the FileWrite (Write File) instruction to write the CSV file.

*WriteLineNum* (Number of Lines to Write) is used to specify the number of CSV file lines to write in the FileWrite (Write File) instruction that will be created in one task period.

Refer to the instructions reference manual for details on the FileWrite (Write File) instruction.

The smaller the value of *WriteLineNum*, the shorter the execution time for this function block in one task period. However, because this results in the function block being executed over more task periods, the time from the start of execution until the end of execution increases.

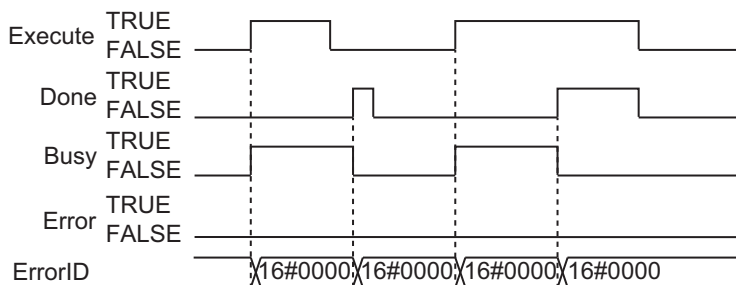
The larger the value of *WriteLineNum*, the longer the execution time for this function block in one task period. However, because this results in the function block being executed over fewer task periods, the time from the start of execution until the end of execution decreases.

Set the value of *WriteLineNum* to match the task period for the task in which this function block is executed.

## Timing Charts

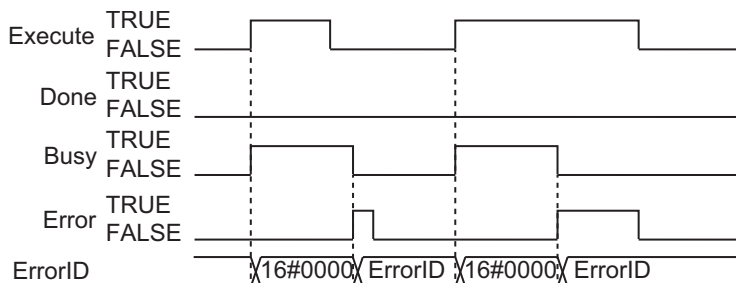
### ● Normal End

- *Busy* (Executing) changes to TRUE when *Execute* changes to TRUE.
- *Done* changes to TRUE when the data output operation is completed.



### ● Error End

- If an error occurs when execution of the function block is in progress, *Error* changes to TRUE and *Busy* (Executing) changes to FALSE.
- You can find out the cause of the error by referring to the value output to *ErrorID* (Error Code).
- If *Execute* changes to FALSE before execution of the function block is ended, *Done* and *Error* are TRUE only for one task period after execution of the function block is ended.
- If *Execute* remains TRUE even after execution of the function block is ended, the output values of *Done* and *Error* are retained.



## Additional Information

To add trace data to *PrgOprTraceRecorder* (Program Operation Trace Recorder), use the function block of *PrgOprTracePut2* on page 207.

## Precautions for Correct Use

- Execution of this function block will be continued until processing is ended even if the value of *Execute* changes to FALSE or the execution time exceeds the task period. The value of *Done* changes to TRUE when processing is ended. Use this to confirm normal ending of processing.
- Do not simultaneously access the same file. Perform exclusive control of SD Memory Card instructions in the user program.
- An error will occur in the following cases. *Error* will change to TRUE.
  - a) The SD Memory Card is not in a usable condition.
  - b) The SD Memory Card is write protected.
  - c) There is insufficient space available on the SD Memory Card.
  - d) The value of *FileName* is not a valid file name.
  - e) The maximum number of files is exceeded.
  - f) The file specified by *FileName* is being accessed.
  - g) The file specified by *FileName* is write protected.
  - h) The value of *FileName* exceeds the maximum number of characters allowed in a file name.
  - i) An error that prevents access occurs during SD Memory Card access.
- Do not execute the same instance while an instance is being executed.
- When you execute this function block, always stop the PrgOprTracePut2 (Add Program Operation Trace Records 2) function beforehand. If you execute this function block without stopping them, it would take longer to write to the SD Memory Card resulting in missing data or additional errors.
- When the power supply is turned OFF to the Controller, the contents of *PrgOprTraceRecorder[]* (Program Operation Trace Recorder) are discarded.
- Do not turn OFF the power supply to the Controller while data is written to the SD Memory Card.

## Related System-defined Variables

Variable name	Meaning	Data type	Description
_Card1Ready	SD Memory Card Ready Flag	BOOL	TRUE when the SD Memory Card is recognized. It is FALSE when an SD Memory Card is not recognized. TRUE: Card can be used. FALSE: Card cannot be used.
_Card1Protect	SD Memory Card Write Protected Flag	BOOL	This flag indicates if the SD Memory Card is write protected when it is inserted and ready to use. TRUE: Write protected. FALSE: Not write protected.
_Card1Err	SD Memory Card Error Flag	BOOL	This flag indicates whether an unspecified SD Memory Card is mounted or if the format is incorrect (i.e. not FAT16 or corrupted). TRUE: There is an error. FALSE: There is no error.
_Card1Access	SD Memory Card Access Flag	BOOL	This flag indicates if the SD Memory Card is currently being accessed. TRUE: Card is being accessed. FALSE: Card is not being accessed.
_Card1PowerFail	SD Memory Card Power Interruption Flag	BOOL	This flag indicates if an error occurred in completing processing when power was interrupted during SD Memory Card access. This flag is not cleared automatically. TRUE: Error. FALSE: No error.
_BackupBusy	Backup Function Busy Flag	BOOL	This flag indicates if a backup, restoration, or verification is in progress. TRUE: Backup, restore, or compare operation is in progress. FALSE: Backup, restore, or compare operation is not in progress.



## Troubleshooting

The error codes, expansion error codes, statuses, descriptions, and corrections are given in the following table.

Error code	Expansion error code	Status	Description	Correction
16#0000	16#00000000	Normal End	-	-
16#0400	16#00000000	Input Value Out of Range	The file name specified with <i>FileName</i> contains one or more characters that cannot be used.	Set <i>FileName</i> correctly.
			The directory name specified with <i>FileName</i> is too long.	Check the length of the text strings specified with <i>FileName</i> so that it is within the valid range.
16#1400	16#00000000	SD Memory Card Access Failure	This error occurs when using this function block in NJ/NX-series CPU Units.	
16#1401	16#00000000	SD Memory Card Write-protected	Refer to the <i>NJ/NX-series Troubleshooting Manual</i> (Cat. No. W503).	
16#1402	16#00000000	SD Memory Card Insufficient Capacity		
16#1404	16#00000000	Too Many Files/Directories		
16#1405	16#00000000	File Already in Use		
16#140A	16#00000000	Write Access Denied		
16#140B	16#00000000	Too Many Files Open		
16#140D	16#00000000	File or Directory Name Is Too Long		
16#140E	16#00000000	SD Memory Card Access Failed		
16#4400	16#00000000	Shared Folder Access Failure	This error occurs when using this function block on NY-series Industrial PCs.	
16#4402	16#00000000	Shared Folder Insufficient Capacity	Refer to the <i>NY-series Troubleshooting Manual</i> (Cat. No. W564).	
16#4404	16#00000000	Too Many Files/Directories		
16#440D	16#00000000	File or Directory Name Is Too Long		
16#440E	16#00000000	Shared Folder Access Failed		

Error code	Expansion error code	Status	Description	Correction
16#3CA5	16#00000001	First Data Position Specification Error	The value of <i>PrgOprTraceRecorderInfo.Top</i> is outside the range for <i>PrgOprTraceRecorder[]</i> .	Set the value of <i>PrgOprTraceRecorderInfo.Top</i> within the range for <i>PrgOprTraceRecorder[]</i> .
	16#00000002	Last Data Position Specification Error	The value of <i>PrgOprTraceRecorder.Bottom</i> is outside the range for <i>PrgOprTraceRecorder[]</i> .	Set the value of <i>PrgOprTraceRecorderInfo.Bottom</i> within the range for <i>PrgOprTraceRecorder[]</i> .
	16#00000003	No Data Stored in Data Recorder	No valid data in <i>PrgOprTraceRecorder[]</i> ( <i>PrgOprTraceRecorderInfo.Count</i> = 0).	Make sure that program operation trace records are stored in <i>PrgOprTraceRecorder[]</i> (Program Operation Trace Recorder).
	16#00000004	Data Recorder Storage Information Error	There are inconsistencies in <i>PrgOprTraceRecorderInfo.Top</i> , <i>PrgOprTraceRecorderInfo.Bottom</i> , and <i>PrgOprTraceRecorderInfo.Count</i> values.	Check <i>PrgOprTraceRecorderInfo.Top</i> , <i>PrgOprTraceRecorderInfo.Bottom</i> , and <i>PrgOprTraceRecorderInfo.Count</i> values.
	16#00000005	Number of Lines to Write Out of Range	The value of <i>WriteLineNum</i> (Number of Lines to Write) is outside the valid range.	Check the valid range of the value for <i>WriteLineNum</i> (Number of Lines to Write) and set the value within the valid range.
	16#00000006	Invalid Data Record First Number	The first element number of <i>PrgOprTraceRecorder[]</i> is not 0.	Set the first number of array element to 0.
	16#00000007	Invalid Data Record Last Number	The last element number of <i>PrgOprTraceRecorder[]</i> is not 19,999 or less.	Set the last number of array element to 19,999 or less.

## Sample Programming

This sample programming saves the program operation trace data that was created with the PrgOprTracePut2 (Add Program Operation Trace Records 2) function sample programming to the SD Memory Card inserted in the CPU Unit in CSV format.

It is added and executed after the SP\_SingleAxisPrgOpr (Single-axis Program Operation) function block and PrgOprTracePut2 (Add Program Operation Trace Records 2) function sample programming.



### Precautions for Correct Use

- The sample programming shows only the portion of a program that uses the function or function block from the library.
- When you use the actual device, include user programming for device safety instructions, interlocks, I/O with other devices, and other control procedures.
- Create a user program so that the actual device operates as intended.
- Check the user program for proper execution before you use it for actual operation.

## Condition

The conditions are the same as those for the SP\_SingleAxisPrgOpr (Single-axis Program Operation) function block sample programming.

## Processing

The program operation trace data is saved to the SD Memory Card with the specified file name when *PrgTraceCSVWrite\_StartPg* (SD Card Save Trigger for Program Operation Trace Data) changes to TRUE.

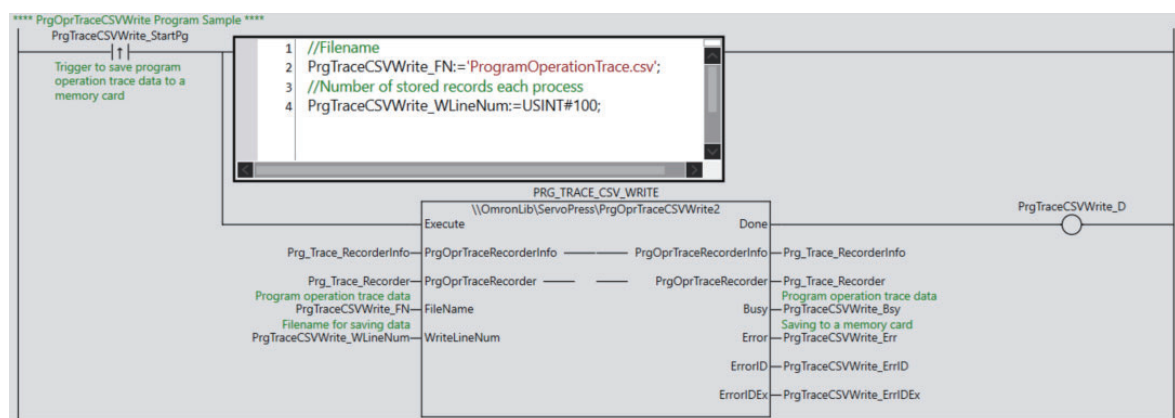
## Ladder Diagram

The following gives the main variables.

### ● Internal Variables

Name	Data type	Initial value	Comment
PRG_TRACE_CSV_WRITE	OmronLib\ServoPress \PrgOprTraceCSVWrite2		Instance of the PrgOpr-TraceCSVWrite2 (Write from Program Operation Trace Recorder to SD Memory Card 2) function block
PrgTraceCSVWrite_StartPg	BOOL		SD card save trigger for program operation trace data
PrgTraceCSVWrite_FN	STRING[66]		File name
PrgTraceCSVWrite_Bsy	BOOL		Saving data to memory card
Prg_Trace_Recorder	ARRAY[0..19999] OF OmronLib \ServoPress \sPRG_OPR_RECORD		Program operation trace data
Prg_Trace_RecorderInfo	OmronLib\ServoPress \sPRG_OPR_TRACE _RECORDERINFO		Management information on trace recorder

### ● Algorithm



## ST

The following gives the main variables.

### ● Internal Variables

Name	Data type	Initial value	Comment
PRG_TRACE_CSV_WRITE	OmronLib\ServoPress \PrgOprTraceCSVWrite2		Instance of the PrgOpr-TraceCSVWrite2 (Write from Program Operation Trace Recorder to SD Memory Card 2) function block
PrgTraceCSVWrite_StartPg	BOOL		SD card save trigger for program operation trace data
PrgTraceCSVWrite_FN	STRING[66]		File name
PrgTraceCSVWrite_Bsy	BOOL		Saving data to memory card
Prg_Trace_Recorder	ARRAY[0..19999] OF OmronLib \ServoPress \sPRG_OPR_RECORD		Program operation trace data
Prg_Trace_RecorderInfo	OmronLib\ServoPress \sPRG_OPR_TRACE _RECORDERINFO		Management information on trace recorder

### ● Algorithm

```
//Specify the filename and start PrgTraceCSVWrite.
IF UpTrig_PrgTraceCSVWrite_StartPg=TRUE THEN
  //Filename
  PrgTraceCSVWrite_FN:='ProgramOperationTrace.csv';
END_IF;

//PrgOprTraceCSVWrite
PRG_TRACE_CSV_WRITE(
  Execute := UpTrig_PrgTraceCSVWrite_StartPg,
  PrgOprTraceRecorderInfo:=Prg_Trace_RecorderInfo,
  PrgOprTraceRecorder := Prg_Trace_Recorder,
  FileName := PrgTraceCSVWrite_FN,
  WriteLineNum := PrgTraceCSVWrite_WLineNum,
  Done => PrgTraceCSVWrite_D,
  Busy => PrgTraceCSVWrite_Bsy,
  Error => PrgTraceCSVWrite_Err,
  ErrorID => PrgTraceCSVWrite_ErrID,
  ErrorIDEx => PrgTraceCSVWrite_ErrIDEx
);
```

# XYDataRec

The XYDataRec function block traces two different input values and prepares trace data for displaying a broken line graph on an NS/NA-series HMI.

Function block name	Name	FB/FUN	Graphic expression	ST expression
XYDataRec	Broken Line Graph Trace Data Preparation	FB		<pre>XYDataRec_instance (   XYTraceData,   Enable,   X_Type,   X,   Y,   X_Orig,   X_Width,   X_Direction,   Enabled,   Busy,   Error,   ErrorID,   ErrorIDEx);</pre>

## Function Block and Function Information

Item	Description
Library file name	OmronLib_ServoPress_V1_0.slr
Namespace	OmronLib\ServoPress
Function block and function number	00099
Publish/Do not publish source code	Published.
Function block and function version	1.00

## Variables

### Input Variables

Name	Meaning	Data type	Default	Valid range	Unit	Description
Enable	Enable	BOOL	FALSE	Depends on data type.	---	Enable TRUE: Enable FALSE: Do not enable execution.
X_Type	X Axis Type	BOOL	FALSE	Depends on data type.	---	The meaning of the X axis on the graph. TRUE: X input value FALSE: Elapsed time
X	X Input Value <sup>*1</sup>	LREAL	0.0	Depends on data type.	---	X input value
Y	Y Input Value	LREAL	0.0	Depends on data type.	---	Y input value
X_Orig	X Origin Value <sup>*1</sup>	LREAL	0.0	Depends on data type.	---	The value of the X axis origin on the graph.
X_Width	X Value Width <sup>*1</sup>	LREAL	0.1	0.001 to 1,000.0	---	The scale width for the X axis values on the graph.
X_Direciton	X Increase/Decrease Direction <sup>*1</sup>	BOOL	TRUE	Depends on data type.	---	Used to specify whether values increase or decrease along the X axis. TRUE: Increase FALSE: Decrease

\*1. These variables are only valid when X\_Type is TRUE.

## Output Variables

Name	Meaning	Data type	Valid range	Unit	Description
Enabled	Enabled	BOOL	Depends on data type.	---	Enabled TRUE: Enabled. FALSE: Not enabled.
Busy	Executing	BOOL	Depends on data type.	---	Executing TRUE: Executing FALSE: Not executing
Error	Error	BOOL	Depends on data type.	---	Error end TRUE: Error end FALSE: Normal end, executing, or execution conditions not met
ErrorID	Error Code	WORD	*1	---	An error code is output if an error occurs. The value is WORD#16#0 for a normal end.
ErrorIDEx	Expansion Error Code	DWORD	*1	---	An expansion error code is output if an error occurs. The value is DWORD#16#0 for a normal end.

\*1. For details, refer to *Troubleshooting* on page 242.

## In-Out Variables

Name	Meaning	Data type	Valid range	Unit	Description
XYTraceData	XY Trace Data	ARRAY [0..19999] OF LREAL	Depends on data type.	---	The trace data for broken line graph display.

## Function

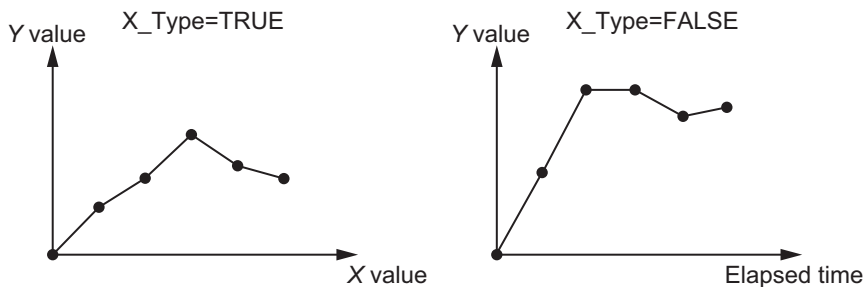
This function block prepares trace data for displaying a broken line graph on an NS/NA-series HMI from X (X Input Value) and Y (Y Input Value).

### X\_Type (X Axis Type)

The meaning of *XYTraceData* (XY Trace Data) depends on the value of *X\_Type* (X Axis Type).

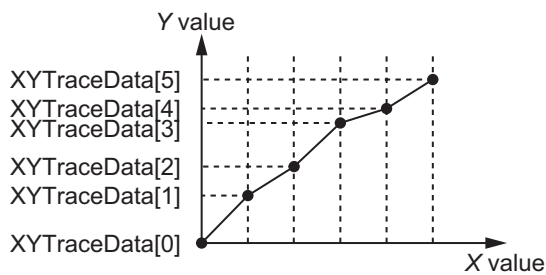
When *X\_Type* is TRUE, the values of X are plotted on the X axis and the values of Y are plotted on the Y axis as shown in the graph on the left side of the following figure.

When *X\_Type* is FALSE, the elapsed time since this function block was started is plotted on the X axis and the values of Y are plotted on the Y axis as shown in the graph on the right side of the following figure. In this case, the values of X (X Input Value), *X\_Orig* (X Origin Value), *X\_Width* (X Value Width), and *X\_Direction* (X Increase/Decrease Direction) are ignored.



### Expression of *XYTraceData* When *X\_Type* Is TRUE

When *X\_Type* is TRUE, the array element numbers in *XYTraceData* express X values at the scale positions on the X axis and the values of the array elements express Y values at the scale positions on the X axis. The following figure shows the concept of this graph.





## ● Correspondence between *XYTraceData* Element Numbers and X Values

The element numbers for *XYTraceData* array show the X value on the X axis scale position. The following equations are used to find the X value that corresponds to element number *n* from the values of *X\_Orig* (X Origin Value), *X\_Width* (X Value Width), and *X\_Direction* (X Increase/Decrease Direction).

When *X\_Direction* Is TRUE

$$\text{X value corresponding to element number } n = X\_Orig + n \times X\_Width$$

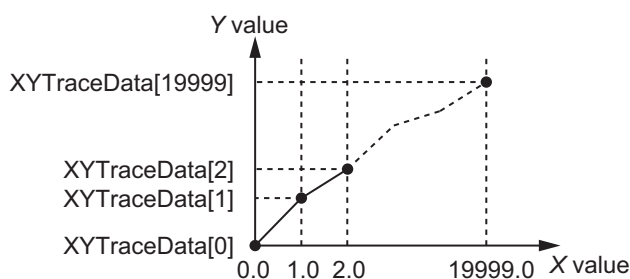
When *X\_Direction* Is FALSE

$$\text{X value corresponding to element number } n = X\_Orig - n \times X\_Width$$

For example, when *X\_Orig* = LREAL#0.0, *X\_Width* = LREAL#1.0, and *X\_Direction* = TRUE, the X values corresponding to the element numbers are as shown in the following table.

Element No.	X value
0	0.0
1	1.0
2	2.0
:	:
19999	19999.0

Therefore, the X values at the X axis scale positions on the graph are as shown in the following graph.



● **Values of XYTraceData Array Elements**

The values of the XYTraceData array elements show the Y values at the X axis scale positions.

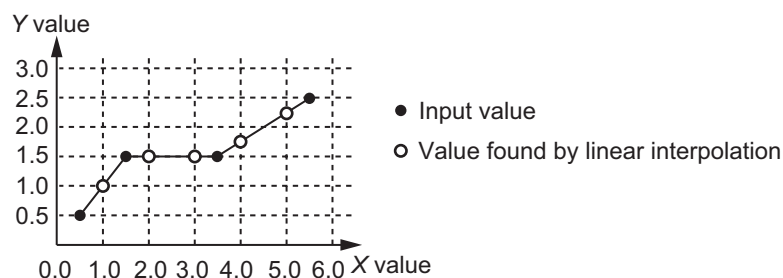
When X (X Input Value) and the scale position on the X axis match, the value of Y (Y Input Value) input in the same task period as X (X Input Value) becomes the value of the XYTraceData element as is.

On the other hand, when X (X Input Value) and the scale position on the X axis do not match, the Y value at the scale position on the X axis is found by linearly interpolating from the values of Y (Y Input Value) adjacent to the scale position on the X axis.

For example, assume that X\_Orig = LREAL#0.0, X\_Width = LREAL#1.0, and X\_Direction = TRUE, and that the following four sets of X (X Input Value) and Y (Y Input Value) have been input.

Task period	Value of X	Value of Y
1	0.5	0.5
2	1.5	1.5
3	3.5	1.5
4	5.5	2.5

These values and the Y values at the scale position on the X axis that are found by linearly interpolating these values result in the following graph.



As a result, the values of the XYTraceData array elements are as shown in the following table.

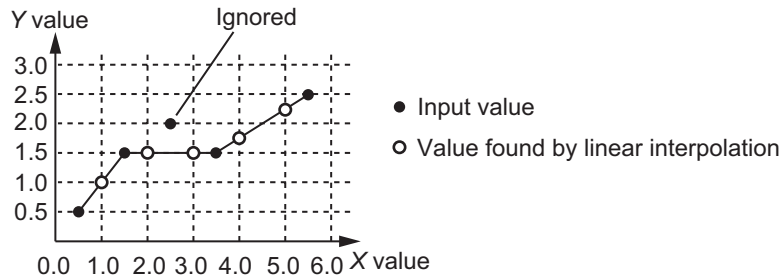
Element No.	Value of XYTraceData Array Element
1	1.0
2	1.5
3	1.5
4	1.75
5	2.25

However, when a value conflicting with the X\_Direction (X Increase/Decrease Direction) setting is input as X (X Input Value), the Y (Y Input Value) input value in the same task period is ignored.

For example, assume that X\_Orig = LREAL#0.0, X\_Width = LREAL#1.0, and X\_Direction = TRUE, and that the following five sets of X (X Input Value) and Y (Y Input Value) have been input.

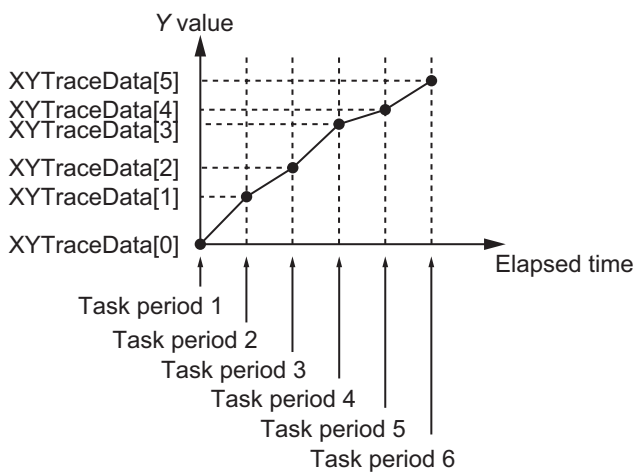
Task period	Value of X	Value of Y
1	0.5	0.5
2	1.5	1.5
3	3.5	1.5
4	2.5	2.0
5	5.5	2.5

The value of X (X Input Value) input in task period 4 is 2.5. This is less than the previous value of X (X Input Value), which was 3.5. This conflicts with the setting *X\_Direction* = TRUE. Therefore, the value 2.0 that was input for Y (Y Input Value) in the same task period is ignored as shown in the following table.



## Expression of XYTraceData When X\_Type Is FALSE

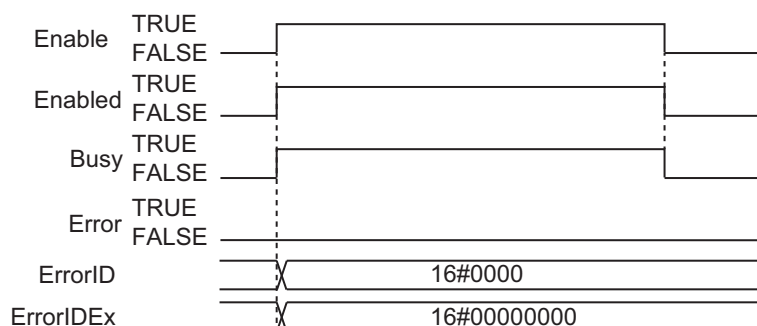
When *X\_Type* is FALSE, the element numbers for the *XYTraceData* array show the elapsed time since this function block was started and the X axis scale positions correspond to the task period since this function block was started. The values of the *XYTraceData* array elements express Y (Y Input Value) that was input in each task period. The following figure shows the concept of this graph.



## Timing Charts

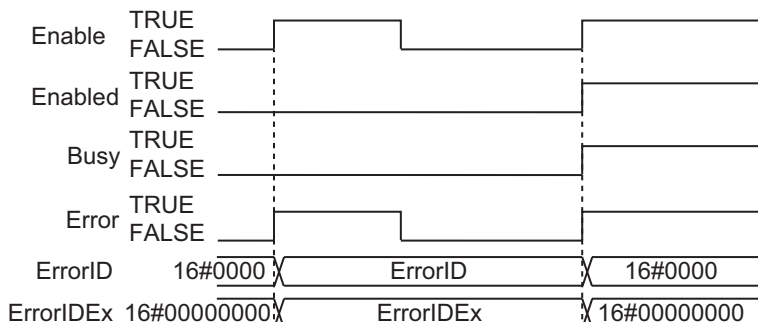
### ● Normal End

- *Busy* (Executing) changes to TRUE at the same time as *Enable* in this function block changes to TRUE.
- While the value of *Enable* is TRUE, X (X Input Value) and Y (Y Input Value) inputs are accepted.



● **Error End**

- If an error occurs during execution of this function block, *Error* changes to TRUE. You can find out the cause of the error by referring to the values output by *ErrorID* (Error Code) and *ErrorIDEx* (Expansion Error Code).



**Additional Information**

The XYDataToGraph function on P.256 is used to convert *XYTraceData* to data for displaying a broken line graph on an NS/NA-series HMI.

**Precautions for Correct Use**

- The values of *X\_Type* (X Axis Type), *X\_Orig* (X Origin Value), *X\_Width* (X Value Width), and *X\_Direction* (X Increase/Decrease Direction) at the point when *Enable* changes to TRUE are valid. The value is not refreshed even if it is changed during processing of this function block.
- If the value of *X* (X Input Value) or *Y* (Y Input Value) is changed while this function block is in process, the values are refreshed for the processing within the same task period.
- Execute this function block in a primary periodic task or in a periodic task.
- If the value of *X* (X Input Value) or *Y* (Y Input Value) is positive infinity, negative infinity, or nonnumeric data, the value of *XYTraceData* (XY Trace Data) will be undefined. Use the CheckReal instruction to determine whether the value of *X* (X Input Value) or *Y* (Y Input Value) is positive infinity, negative infinity, or nonnumeric data. Refer to the instructions reference manual for details on the CheckReal instruction.

**Troubleshooting**

The error codes, expansion error codes, status, descriptions, and corrections given in the following table.

Error code	Expansion error code	Status	Description	Correction
16#0000	16#00000000	Normal end		
16#3CA6	16#00000001	Incorrect Task Setting	An attempt was made to execute this function block in a task period which is not a primary task period or a periodic task.	Execute this function block in a primary periodic task or a periodic task.
	16#00000002	X Origin Value Out of Range	The value of <i>X_Orig</i> (X Origin Value) is outside the valid range.	Check the valid range for the value of <i>X_Orig</i> (X Origin Value) and set the value within the valid range.
	16#00000003	X Value Width Out of Range	The value of <i>X_Width</i> (X Value Width) is outside the valid range.	Check the valid range for the value of <i>X_Width</i> (X Value Width) and set the value within the valid range.

## Sample Programming

This sample programming creates trace data for the SP\_SingleAxisPrgOpr (Single-axis Program Operation) function block sample programming execution results that are used to display a broken-line graph on an NS/NA-series PT. The trace data is specified as the *MC\_Axis000.Act.Pos* (Actual Current Position) axis variable for the horizontal axis and the *MC\_Axis000.Act.Trq* (Actual Current Torque) axis variable for the vertical axis.

It is added and executed after the SP\_SingleAxisPrgOpr (Single-axis Program Operation) function block sample programming.



### Precautions for Correct Use

- The sample programming shows only the portion of a program that uses the function or function block from the library.
- When you use the actual device, include user programming for device safety instructions, interlocks, I/O with other devices, and other control procedures.
- Create a user program that will produce the intended device operation.
- Check the user program for proper execution before you use it for actual operation.

## Conditions

The conditions are the same as those for the SP\_SingleAxisPrgOpr (Single-axis Program Operation) function block sample programming.

## Processing

If *XY\_Rec\_En* (Enable Graph Trace Data Recording) is TRUE, this processing records a trace of the specified data while the SP\_SingleAxisPrgOpr (Single-axis Program Operation) function block is being executed.

## Ladder Diagram

The following gives the main variables.

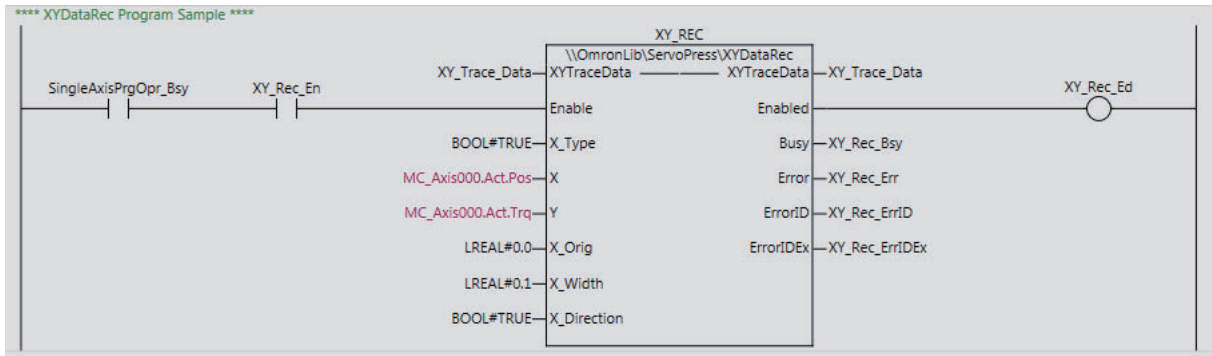
### ● Internal Variables

Name	Data Type	Initial Value	Comment
XY_REC	OmronLib\Servo-Press\XYDataRec		Instance of the XYDataRec (Broken Line Graph Trace Data Preparation) function block
SingleAxisPrgOpr_Bsy	BOOL		Single-axis program operation busy
XY_Rec_En	BOOL		Enables graph trace data recording.
XY_Trace_Data	ARRAY[0..19999] OF LREAL		Trace data for graphs

### ● External Variables

Name	Data Type	Initial Value	Comment
MC_Axis000	_sAXIS_REF	---	Servo axis

● Algorithm



**ST**

The following gives the main variables.

● Internal Variables

Name	Data Type	Initial Value	Comment
XY_REC	OmronLib\Servo-Press\XYDataRec		Instance of the XYDataRec (Broken Line Graph Trace Data Preparation) function block
SingleAxisPrgOpr_Bsy	BOOL		Single-axis program operation busy
XY_Rec_En	BOOL		Enables graph trace data recording.
XY_Trace_Data	ARRAY[0..19999] OF LREAL		Trace data for graphs

● External Variables

Name	Data Type	Initial Value	Comment
MC_Axis000	_sAXIS_REF	---	Servo axis

● Algorithm

```
//XYDataRec
XY_REC (
  XYTraceData :=XY_Trace_Data,
  Enable := SingleAxisPrgOpr_Bsy AND XY_Rec_En,
  X_Type := BOOL#TRUE,
  X := MC_Axis000.Act.Pos,
  Y := MC_Axis000.Act.Trq,
  X_Orig := LREAL#0.0,
  X_Width := LREAL#0.1,
  X_Direction := BOOL#TRUE,
  Enabled => XY_Rec_Ed,
  Busy => XY_Rec_Bsy,
  Error => XY_Rec_Err,
  ErrorID => XY_Rec_ErrID,
  ErrorIDEx => XY_Rec_ErrIDEx
);
```

# XYDataRec2

The XYDataRec2 function block traces two different input values and prepares trace data for displaying a broken line graph on an NS/NA-series HMI. This function block allows you to define the trace data length to any length that you want.

FB/FUN name	Name	FB/FUN	Graphic expression	ST expression
XYDataRec2	Broken Line Graph Trace Data Preparation 2	FB		<pre>XYDataRec2_instance(   XYTraceData,   Enable,   X_Type,   X,   Y,   X_Orig,   X_Width,   X_Direction,   Enabled,   Busy,   Error,   ErrorID,   ErrorIDEx);</pre>

## Function Block and Function Information

Item	Description
Library file name	OmronLib_ServoPress_V2_0.slr
Namespace	OmronLib\ServoPress
FUN/FB number	00225
Source code	Not published.

## Variables

### Input Variables

Name	Meaning	Data type	Initial value	Valid range	Unit	Description
Enable	Enable	BOOL	FALSE	Depends on data type.	-	Enable TRUE: Enable FALSE: Do not enable execution
X_Type	X Axis Type	BOOL	FALSE	Depends on data type.	-	The meaning of the X axis on the graph TRUE: X input value FALSE: Elapsed time
X	X Input Value <sup>*1</sup>	LREAL	0.0	Depends on data type.	-	X input value
Y	Y Input Value	LREAL	0.0	Depends on data type.	-	Y input value
X_Orig	X Origin Value <sup>*1</sup>	LREAL	0.0	Depends on data type.	-	The value of the X axis origin on the graph
X_Width	X Value Width <sup>*1</sup>	LREAL	0.1	0.001 to 1000.0	-	The scale width for the X axis values on the graph
X_Direciton	X Increase/Decrease Direction <sup>*1</sup>	BOOL	TRUE	Depends on data type.	-	Used to specify whether values increase or decrease along the X axis TRUE: Increase FALSE: Decrease

\*1. These variables are only valid when X\_Type is TRUE.



## Output Variables

Name	Meaning	Data type	Valid range	Unit	Description
Enabled	Enabled	BOOL	Depends on data type.	-	Enabled TRUE: Enabled. FALSE: Not enabled.
Busy	Executing	BOOL	Depends on data type.	-	Executing TRUE: Execution processing is in progress. FALSE: Execution processing is not in progress.
Error	Error	BOOL	Depends on data type.	-	Error end TRUE: Error end FALSE: Normal end, execution in progress, or execution condition not met
ErrorID	Error Code	WORD	*1	-	An error code is output if an error occurs. The value is WORD#16#0 for a normal end.
ErrorIDEx	Expansion Error Code	DWORD	*1	-	An expansion error code is output if an error occurs. The value is DWORD#16#0 for a normal end.

\*1. Refer to *Troubleshooting* on page 253 for the details.

## In-Out Variables

Name	Meaning	Data type	Valid range	Unit	Description
XYTrace Data[]*1	XY Trace Data	ARRAY[*] OF LREAL	Depends on data type.	-	The trace data for broken line graph display.

\*1. The maximum number of array elements is 20,000. The first number of array element should be 0.

## Function

This function block prepares trace data for displaying a broken line graph on an NS/NA-series HMI from X (X Input Value) and Y (Y Input Value).

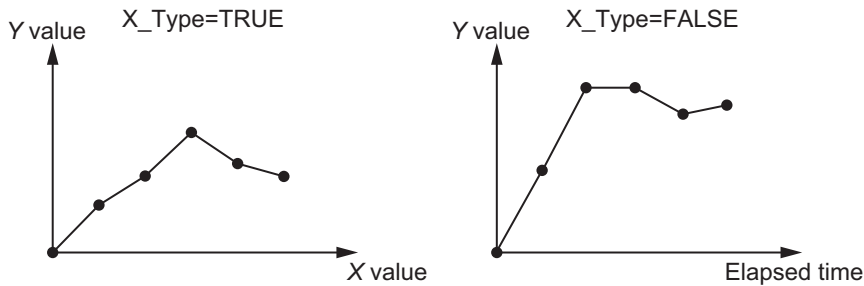
### X\_Type (X Axis Type)

The meaning of *XYTraceData[]* (XY Trace Data) depends on the value of *X\_Type* (X Axis Type).

When *X\_Type* is TRUE, the values of X are plotted on the X axis and the values of Y are plotted on the Y axis as shown in the graph on the left side of the following figure.

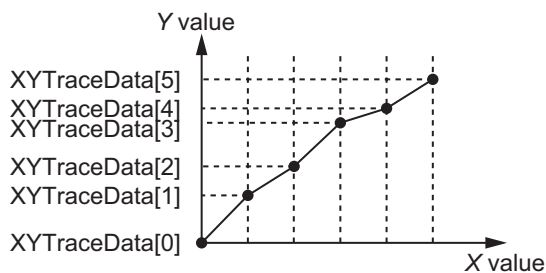
When *X\_Type* is FALSE, the elapsed time since this function block was started is plotted on the X axis and the values of Y are plotted on the Y axis as shown in the graph on the right side of the following figure.

In this case, the values of X (X Input Value), *X\_Orig* (X Origin Value), *X\_Width* (X Value Width), and *X\_Direction* (X Increase/Decrease Direction) are ignored.



### Expression of *XYTraceData* When *X\_Type* Is TRUE

When *X\_Type* is TRUE, the array element numbers in *XYTraceData[]* express X values at the scale positions on the X axis and the values of the array elements express Y values at the scale positions on the X axis. The following figure shows the concept of this graph.



#### ● Correspondence between *XYTraceData[]* Element Numbers and X Values

The element numbers for *XYTraceData[]* array show the X value on the X axis scale position. The following equations are used to find the X value that corresponds to element number n from the values of *X\_Orig* (X Origin Value), *X\_Width* (X Value Width), and *X\_Direction* (X Increase/Decrease Direction).

When *X\_Direction* Is TRUE

$$\text{X value corresponding to element number } n = X\_Orig + n \times X\_Width$$

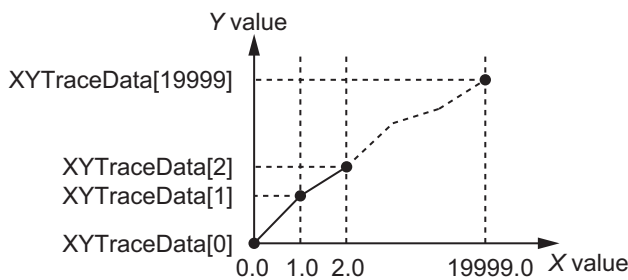
When *X\_Direction* Is FALSE

$$\text{X value corresponding to element number } n = X\_Orig - n \times X\_Width$$

For example, when  $X\_Orig = LREAL\#0.0$ ,  $X\_Width = LREAL\#1.0$ , and  $X\_Direction = TRUE$ , the X values corresponding to the element numbers are as shown in the following table.

Element No.	X value
0	0.0
1	1.0
2	2.0
:	:
19999	19999.0

Therefore, the X values at the X axis scale positions on the graph are as shown in the following graph.



● **Values of *XYTraceData[]* Array Elements**

The values of the *XYTraceData[]* array elements show the Y values at the X axis scale positions.

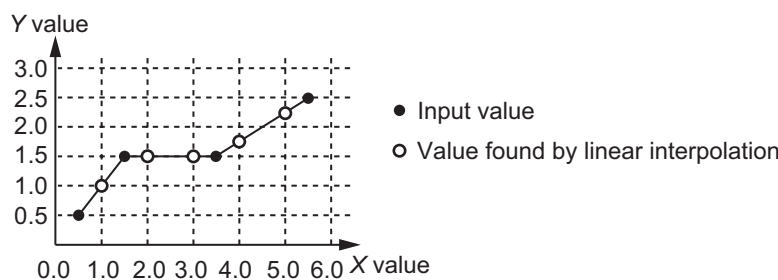
When X (X Input Value) and the scale position on the X axis match, the value of Y (Y Input Value) input in the same task period as X (X Input Value) becomes the value of the *XYTraceData[]* element as is.

On the other hand, when X (X Input Value) and the scale position on the X axis do not match, the Y value at the scale position on the X axis is found by linearly interpolating from the values of Y (Y Input Value) adjacent to the scale position on the X axis.

For example, assume that  $X\_Orig = LREAL\#0.0$ ,  $X\_Width = LREAL\#1.0$ , and  $X\_Direction = TRUE$ , and that the following four sets of X (X Input Value) and Y (Y Input Value) have been input.

Task period	X value	Y value
1	0.5	0.5
2	1.5	1.5
3	3.5	1.5
4	5.5	2.5

These values and the Y values at the scale position on the X axis that are found by linearly interpolating these values result in the following graph.



As a result, the values of the *XYTraceData[]* array elements are as shown in the following table.

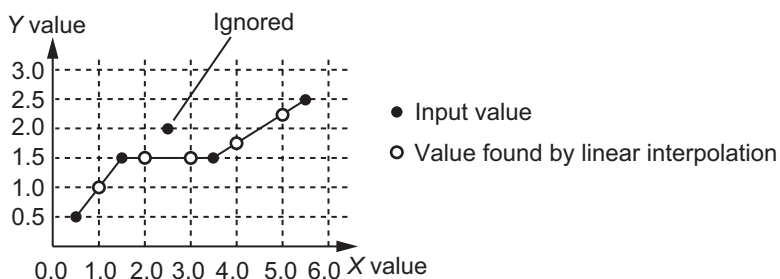
Element No.	Values of <i>XYTraceData[]</i> Array Elements
1	1.0
2	1.5
3	1.5
4	1.75
5	2.25

However, when a value conflicting with the *X\_Direction* (X Increase/Decrease Direction) setting is input as X (X Input Value), the Y (Y Input Value) input value in the same task period is ignored.

For example, assume that *X\_Orig* = LREAL#0.0, *X\_Width* = LREAL#1.0, and *X\_Direction* = TRUE, and that the following five sets of X (X Input Value) and Y (Y Input Value) have been input.

Task period	X value	Y value
1	0.5	0.5
2	1.5	1.5
3	3.5	1.5
4	2.5	2.0
5	5.5	2.5

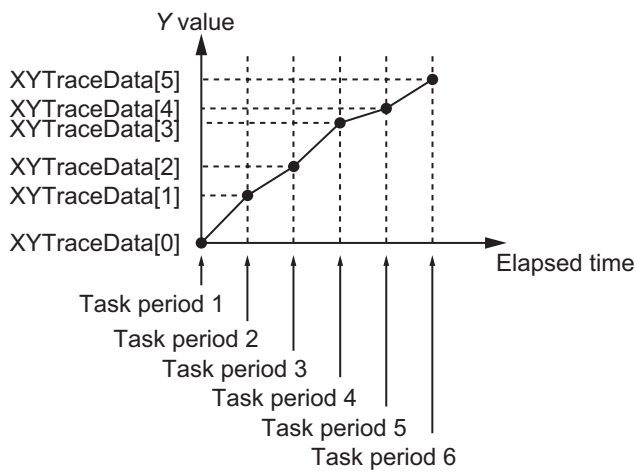
The value of X (X Input Value) input in task period 4 is 2.5. This is less than the previous value of X (X Input Value), which was 3.5. This conflicts with the setting *X\_Direction* = TRUE. Therefore, the value 2.0 that was input for Y (Y Input Value) in the same task period is ignored as shown in the following table.



## Expression of XYTraceData When X\_Type Is FALSE

When *X\_Type* is FALSE, the element numbers for the *XYTraceData[]* array show the elapsed time since this function block was started and the X axis scale positions correspond to the task period since this function block was started. The values of the *XYTraceData[]* array elements express Y (Y Input Value) that was input in each task period.

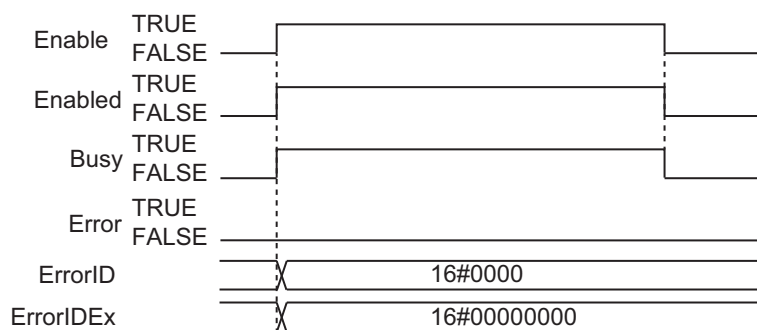
The following figure shows the concept of this graph.



## Timing Charts

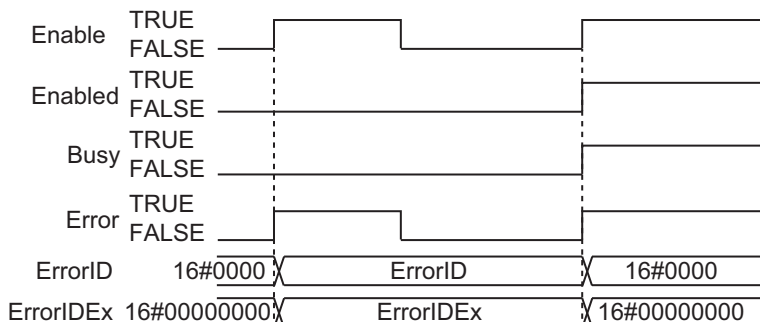
### ● Normal End

- *Busy* (Executing) changes to TRUE when *Enable* in the function block changes to TRUE.
- While the value of *Enable* is TRUE, X (X Input Value) and Y (Y Input Value) inputs are accepted.



● **Error End**

- If an error occurs during execution of this function block, *Error* changes to TRUE. You can find out the cause of the error by referring to the values output by *ErrorID* (Error Code) and *ErrorIDEx* (Expansion Error Code).



**Additional Information**

The *XYDataToGraph2* on page 265 function is used to convert *XYTraceData[]* to data for displaying a broken line graph on an NS/NA-series HMI.

**Precautions for Correct Use**

- The values of *X\_Type* (X Axis Type), *X\_Orig* (X Origin Value), *X\_Width* (X Value Width), and *X\_Direction* (X Increase/Decrease Direction) at the point when *Enable* changes to TRUE are valid. The value is not refreshed even if it is changed during processing of this function block.
- If the value of *X* (X Input Value) or *Y* (Y Input Value) is changed while this function block is in process, the values are refreshed for the processing within the same task period.
- Execute this function block in a primary periodic task or a periodic task.
- If the value of *X* (X Input Value) or *Y* (Y Input Value) is positive infinity, negative infinity, or nonnumeric data, the value of *XYTraceData[]* (XY Trace Data) will be undefined. Use the CheckReal instruction to determine whether the value of *X* (X Input Value) or *Y* (Y Input Value) is positive infinity, negative infinity, or nonnumeric data.

Refer to the instructions reference manual for details on the CheckReal instruction.

## Troubleshooting

The error codes, expansion error codes, statuses, descriptions, and corrections are given in the following table.

Error code	Expansion error code	Status	Description	Correction
16#0000	16#00000000	Normal end	-	-
16#3CA6	16#00000001	Incorrect Task Setting	An attempt was made to execute this function block in a task period which is not a primary periodic task or a periodic task.	Execute this function block in a primary periodic task or a periodic task.
	16#00000002	X Origin Value Out of Range	The value of <i>X_Orig</i> (X Origin Value) is outside the valid range.	Check the valid range of the value of <i>X_Orig</i> (X Origin Value) and set the value within the valid range.
	16#00000003	X Value Width Out of Range	The value of <i>X_Width</i> (X Value Width) is outside the valid range.	Check the valid range for the value of <i>X_Width</i> (X Value Width) and set the value within the valid range.
	16#00000004	Invalid Trace Data First Number	The first element number of <i>XYTraceData[]</i> is not 0.	The first element number of <i>XYTraceData[]</i> should be 0.
	16#00000005	Invalid Trace Data Last Number	The last element number of <i>XYTraceData[]</i> is 20000 or more.	The last element number of <i>XYTraceData[]</i> should be 19999 or less.

## Sample Programming

This sample programming creates trace data for the SP\_SingleAxisPrgOpr (Single-axis Program Operation) function block sample programming execution results that are used to display a broken-line graph on an NS/NA-series PT. The trace data is specified as the *MC\_Axis000.Act.Pos* (Actual Current Position) axis variable for the horizontal axis and the *MC\_Axis000.Act.Trq* (Actual Current Torque) axis variable for the vertical axis.

It is added and executed after the SP\_SingleAxisPrgOpr (Single-axis Program Operation) function block sample programming.



### Precautions for Correct Use

- The sample programming shows only the portion of a program that uses the function or function block from the library.
- When you use the actual device, include user programming for device safety instructions, interlocks, I/O with other devices, and other control procedures.
- Create a user program so that the actual device operates as intended.
- Check the user program for proper execution before you use it for actual operation.

## Conditions

The conditions are the same as those for the SP\_SingleAxisPrgOpr (Single-axis Program Operation) function block sample programming.

## Processing

If *XY\_Rec\_En* (Enable Graph Trace Data Recording) is TRUE, this processing records a trace of the specified data while the SP\_SingleAxisPrgOpr (Single-axis Program Operation) function block is being executed.

## Ladder Diagram

The following gives the main variables.

### ● Internal Variables

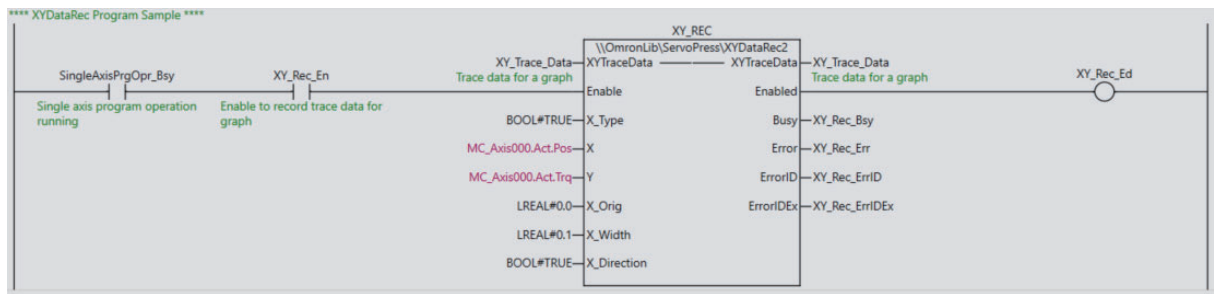
Name	Data type	Initial value	Comment
XY_REC	OmronLib\Servo-Press\XYDataRec2		Instance of the XYDataRec2 (Broken Line Graph Trace Data Preparation 2) function block
SingleAxisPrgOpr_Bsy	BOOL		Single-axis program operation busy
XY_Rec_En	BOOL		Enables graph trace data recording.
XY_Trace_Data	ARRAY[0..19999] OF LREAL		Trace data for graphs

### ● External Variables

Name	Data type	Initial value	Comment
MC_Axis000	_sAXIS_REF	-	Servo axis



● Algorithm



**ST**

The following gives the main variables.

● Internal Variables

Name	Data type	Initial value	Comment
XY_REC	OmronLib\Servo-Press\XYDataRec2		Instance of the XYDataRec2 (Broken Line Graph Trace Data Preparation 2) function block
SingleAxisPrgOpr_Bsy	BOOL		Single-axis program operation busy
XY_Rec_En	BOOL		Enables graph trace data recording.
XY_Trace_Data	ARRAY[0..19999] OF LREAL		Trace data for graphs

● External Variables

Name	Data type	Initial value	Comment
MC_Axis000	_sAXIS_REF	-	Servo axis

● Algorithm

```
//XYDataRec
XY_REC(
  XYTraceData :=XY_Trace_Data,
  Enable := SingleAxisPrgOpr_Bsy AND XY_Rec_En,
  X_Type := BOOL#TRUE,
  X := MC_Axis000.Act.Pos,
  Y := MC_Axis000.Act.Trq,
  X_Orig := LREAL#0.0,
  X_Width := LREAL#0.1,
  X_Direction := BOOL#TRUE,
  Enabled => XY_Rec_Ed,
  Busy => XY_Rec_Bsy,
  Error => XY_Rec_Err,
  ErrorID => XY_Rec_ErrID,
  ErrorIDEx => XY_Rec_ErrIDEx
);
```

# XYDataToGraph

The XYDataToGraph function converts the trace data to NA/NS-series HMI broken line graph display data.

Function name	Name	FB/ FUN	Graphic expression	ST expression
XYDataToGraph	Broken Line Graph Display Data Conversion	FUN		<pre>Out:=\\OmronLib\ServoPress\XYDataToGraph (   XYTraceData,   X_Init,   X_Zoom,   X_ZoomNum,   Y_Zoom,   Y_Zoomnum,   XYGraphData);</pre>

## Function Block and Function Information

Item	Description
Library file name	OmronLib_ServoPress_V1_0.slr
Namespace	OmronLib\ServoPress
Function block and function number	00100
Publish/Do not publish source code	Published.
Function block and function version	1.00

## Variables

### Input Variables

Name	Meaning	Data type	Default	Valid range	Unit	Description
EN	Execute	BOOL	FALSE	Depends on data type.	---	Execute trigger for this function Executes the function when it changes to TRUE.
X_Init	X Axis Initial Position	UINT	0	0 to 19,999	---	The starting element number in <i>XYTrace-Data</i> to be graphed.
X_Zoom	X Axis Zoom	BOOL	FALSE	Depends on data type.	---	Zooms in or zooms out in the X axis direction. TRUE: Zoom in FALSE: Zoom out
X_Zoom Num	X Axis Zoom Coefficient	USINT	1	1 to 100	---	The zoom-in ratio or zoom-out ratio for the X axis direction.
Y_Zoom	Y Axis Zoom	BOOL	FALSE	Depends on data type.	---	Zooms in or zooms out in the Y axis direction. TRUE: Zoom in FALSE: Zoom out
Y_Zoom Num	Y Axis Zoom Coefficient	USINT	1	1 to 100	---	The zoom-in ratio or zoom-out ratio for the Y axis direction.

### Output Variables

Name	Meaning	Data type	Valid range	Unit	Description
Out	Return Value	BOOL	Depends on data type.	---	Function execution results TRUE: Normal end FALSE: Error end

### In-Out Variables

Name	Meaning	Data type	Valid range	Unit	Description
XYTraceData	XY Trace Data	ARRAY [0..19999] OF LREAL	Depends on data type.	---	The trace data for broken line graph display.
XYGraphData	XY Graph Display Data	ARRAY [0..599] OF LREAL	Depends on data type.	---	Broken line graph display data.

## Function

This function converts *XYTraceData* to *XYGraphData* (XY Graph Display Data) NA/NS-series HMI broken line graph display data.

When converting, you can specify the X axis direction conversion start position for the contents of *XYTraceData*. You can also zoom in on or zoom out from the X axis direction and Y axis direction.

### *XYTraceData* Structure

*XYTraceData* is prepared using the XYDataRec (Broken Line Graph Trace Data Preparation) function block. Refer to *XYDataRec* on page 235 for details on the *XYTraceData* structure.

### *XYGraphData* Structure

The *XYGraphData* structure is the same as the *XYTraceData* structure. Specifically, the array element numbers express the X value for each data record and the array element values express the Y value for each data record.

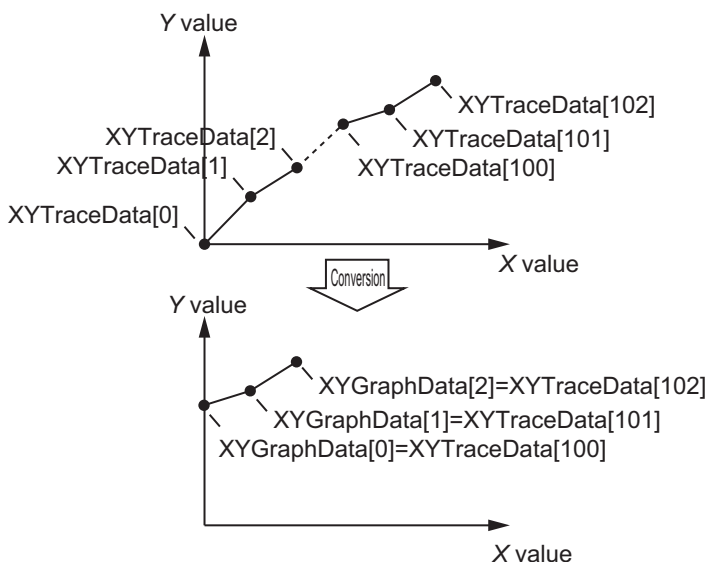
## Data Conversion

When you convert the data, you specify the X axis direction conversion start position and the X axis and Y axis direction zoom as described below.

### ● X Axis Direction Conversion Start Position Specification

The X axis direction conversion start position is specified with *X\_Init* (X Axis Initial Position). Specify the *XYTraceData* array element number at which to start conversion in *X\_Init*.

For example, when *X\_Init*=UINT#100, *XYTraceData*[100] and higher are converted. The following graphs show the concept.



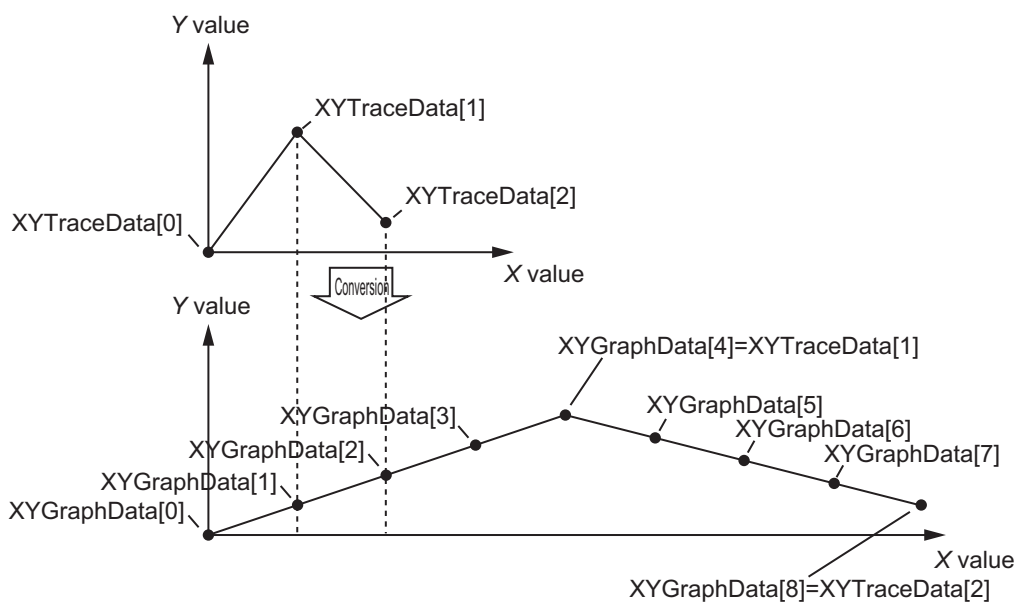
## ● X Axis Direction Zoom In

By changing the value of *X\_Zoom* (X Axis Zoom) to TRUE, you can convert *XYTraceData* so that it is zoomed in along the X axis direction. Specify the zoom-in ratio with *X\_ZoomNum* (X Axis Zoom Coefficient).

Y values are interpolated between two adjacent points in *XYTraceData* and *X\_ZoomNum* minus 1 data records are added. The X values corresponding to the array element numbers in *XYTraceData* and *XYGraphData* are the same, so the *XYTraceData* graph is changed to an *XYGraphData* graph multiplied by *X\_ZoomNum* in the X axis direction.

The number of *XYGraphData* data records is changed to the following value: (Number of *XYTraceData* data records - 1) x *X\_ZoomNum* + 1.

The following figure shows the relationship between *XYTraceData* and *XYGraphData* when there are three *XYTraceData* data records and *X\_ZoomNum* = UINT#4. The graph is multiplied by four in the X axis direction and the number of data records increases to nine.



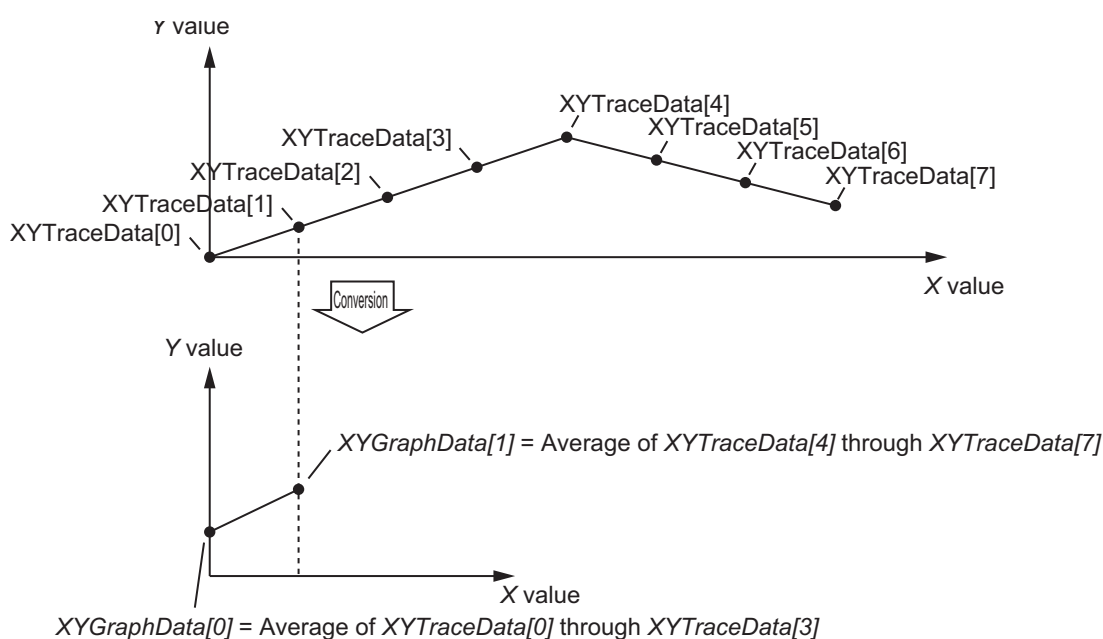
● X Axis Direction Zoom Out

By changing the value of *X\_Zoom* (X Axis Zoom) to FALSE, you can convert *XYTraceData* so that it is zoomed out along the X axis direction. Specify the zoom-out ratio with *X\_ZoomNum* (X Axis Zoom Coefficient). The zoom-out ratio is  $1/X\_ZoomNum$ .

The Y values of *X\_ZoomNum* of adjacent *XYTraceData* data records are averaged and merged into one data record. The X values corresponding to the array element numbers in *XYTraceData* and *XYGraphData* are the same, so the *XYTraceData* graph is changed to an *XYGraphData* graph divided by *X\_ZoomNum* in the X axis direction.

The number of *XYGraphData* data records is the number of *XYTraceData* data records divided by *X\_ZoomNum*.

The following figure shows the relationship between *XYTraceData* and *XYGraphData* when there are eight *XYTraceData* data records and *X\_ZoomNum* = UINT#4. The graph is divided by four in the X axis direction and the number of data records decreases to two.



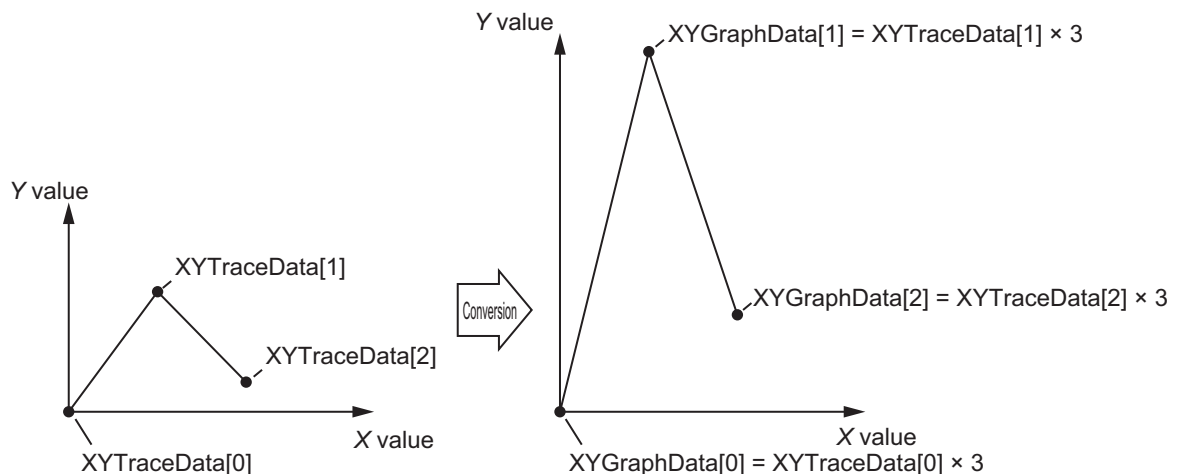
## ● Y Axis Direction Zoom In

By changing the value of *Y\_Zoom* (Y Axis Zoom) to TRUE, you can convert *XYTraceData* so that it is zoomed in along the Y axis direction. Specify the zoom-in ratio with *Y\_ZoomNum* (Y Axis Zoom Coefficient).

The Y value of *XYTraceData* multiplied by *Y\_ZoomNum* becomes the *XYGraphData* Y value.

The number of *XYGraphData* data records and *XYTraceData* data records is the same.

The following figure shows the relationship between *XYTraceData* and *XYGraphData* when there are three *XYTraceData* data records and *Y\_ZoomNum* = UINT#3. The graph is multiplied by three in the Y axis direction and the number of data records remains three.



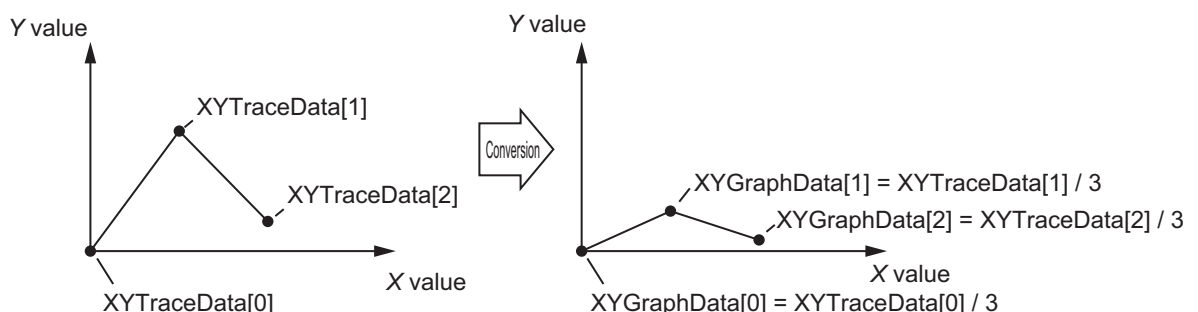
## ● Y Axis Direction Zoom Out

By changing the value of *Y\_Zoom* (Y Axis Zoom) to FALSE, you can convert *XYTraceData* so that it is zoomed out along the Y axis direction. Specify the zoom-out ratio with *Y\_ZoomNum* (Y Axis Zoom Coefficient). The zoom-out ratio is  $1/X\_ZoomNum$ .

The *XYTraceData* Y value divided by *Y\_ZoomNum* becomes the *XYGraphData* Y value.

The number of *XYGraphData* data records and *XYTraceData* data records is the same.

The following figure shows the relationship between *XYTraceData* and *XYGraphData* when there are three *XYTraceData* data records and *Y\_ZoomNum* = UINT#3. The graph is divided by three in the Y axis direction and the number of data records remains three.



## Additional Information

To prepare *XYTraceData* data, use the *XYDataRec* function block on P.235.

## Precautions for Correct Use

If the value of an input variable is out of range, an error occurs and the value of *Out* changes to FALSE. The values of *XYGraphData* are not updated.

## Sample Programming

This sample programming extracts 5,000 records of data from the 1,000th record of the 20,000 records of broken-line graph display trace data created with the *XYDataRec* (Broken-line Graph Trace Data Preparation) function block, and it creates 600 records of display data suitable for displaying on an NS/NA-series PT.

It is added and executed after the *SP\_SingleAxisPrgOpr* (Single-axis Program Operation) function block and *XYDataRec* (Broken-line Graph Trace Data Preparation) function block sample programming.



### Precautions for Correct Use

- The sample programming shows only the portion of a program that uses the function or function block from the library.
- When you use the actual device, include user programming for device safety instructions, interlocks, I/O with other devices, and other control procedures.
- Create a user program that will produce the intended device operation.
- Check the user program for proper execution before you use it for actual operation.

## Conditions

The conditions are the same as those for the *SP\_SingleAxisPrgOpr* (Single-axis Program Operation) function block sample programming.

## Processing

The graph drawing data is created with the specified start point (array index) and scale ratio when *XY\_Graph\_StartPG* (Creation Trigger for Graph Drawing Data) changes to TRUE.



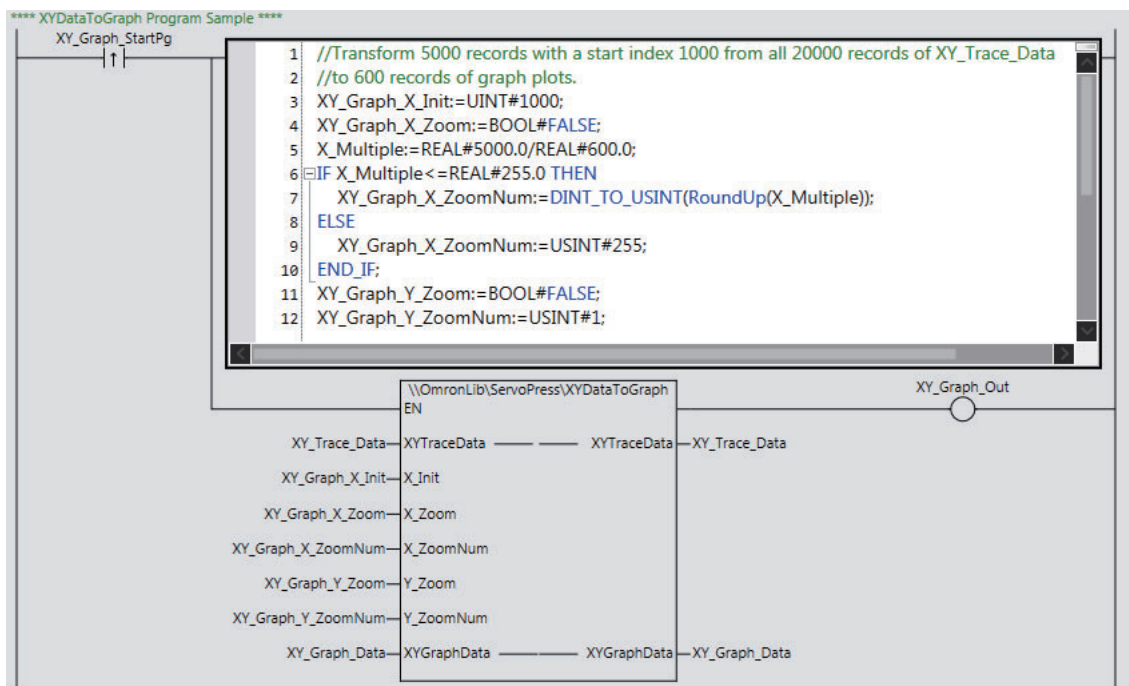
## Ladder Diagram

The following gives the main variables.

### ● Internal Variables

Name	Data Type	Initial Value	Comment
XY_Trace_Data	ARRAY[0..19999] OF LREAL		Trace data for graphs
XY_Graph_StartPg	BOOL		Creation trigger for graph drawing data
XY_Graph_Data	ARRAY[0..599] OF LREAL		Graph drawing data

### ● Algorithm



## ST

The following gives the main variables.

### ● Internal Variables

Name	Data Type	Initial Value	Comment
XY_Trace_Data	ARRAY[0..19999] OF LREAL		Trace data for graphs
XY_Graph_StartPg	BOOL		Creation trigger for graph drawing data
XY_Graph_Data	ARRAY[0..599] OF LREAL		Graph drawing data

### ● Algorithm

```
//Generate graph plots data.
IF UpTrig_XY_Graph_StartPg=TRUE THEN
  //Transform 5000 records with a start index 1000 from all 20000 records of
  XY_Trace_Data to 600 records of graph plots.
  XY_Graph_X_Init:=UINT#1000;
  XY_Graph_X_Zoom:=BOOL#FALSE;
  X_Multiple:=REAL#5000.0/REAL#600.0;
  IF X_Multiple<=REAL#255.0 THEN
    XY_Graph_X_ZoomNum:=DINT_TO_USINT(RoundUp(X_Multiple));
  ELSE
    XY_Graph_X_ZoomNum:=USINT#255;
  END_IF;
  XY_Graph_Y_Zoom:=BOOL#FALSE;
  XY_Graph_Y_ZoomNum:=USINT#1;
END_IF;

//XYGraphToData
XY_Graph_Out:=\\OmronLib\ServoPress\XYDataToGraph(
  EN := UpTrig_XY_Graph_StartPg,
  XYTraceData := XY_Trace_Data,
  X_Init := XY_Graph_X_Init,
  X_Zoom := XY_Graph_X_Zoom,
  X_ZoomNum := XY_Graph_X_ZoomNum,
  Y_Zoom := XY_Graph_Y_Zoom,
  Y_ZoomNum := XY_Graph_Y_ZoomNum,
  XYGraphData := XY_Graph_Data
);
```

# XYDataToGraph2

The XYDataToGraph2 function converts the trace data to NS/NA-series HMI broken line graph display data (600 points). This function allows you to define the trace data length to any length that you want.

FB/FUN name	Name	FB/FUN	Graphic expression	ST expression
XYDataToGraph2	Broken Line Graph Display Data Conversion 2	FUN		<pre>Out:=\OmronLib\ServoPress\XYDataToGraph2(   XYTraceData,   X_Init,   X_Zoom,   X_ZoomNum,   Y_Zoom,   Y_ZoomNum,   XYGraphData);</pre>

## Function Block and Function Information

Item	Description
Library file name	OmronLib_ServoPress_V2_0.slr
Namespace	OmronLib\ServoPress
FUN/FB number	00226
Source code	Not published.

## Variable

### Input variables

Variable	Name	Data type	Initial value	Valid range	Unit	Description
EN	Execute	BOOL	FALSE	Depends on data type.	-	The processing is executed while the variable changes to TRUE.
X_Init	X Axis Initial Position	UINT	0	0 to 19,999	-	The first element number of <i>XYTraceData[]</i> to be graphed.
X_Zoom	X Axis Zoom	BOOL	FALSE	Depends on data type.	-	Zooms in or zooms out in the X axis direction. TRUE: Zoom in FALSE: Zoom out
X_ZoomNum	X Axis Zoom Coefficient	USINT	1	1 to 100	-	The zoom-in ratio or zoom-out ratio for the X axis direction.
Y_Zoom	Y Axis Zoom	BOOL	FALSE	Depends on data type.	-	Zooms in or zooms out in the Y axis direction. TRUE: Zoom in FALSE: Zoom out
Y_ZoomNum	Y Axis Zoom Coefficient	USINT	1	1 to 100	-	The zoom-in ratio or zoom-out ratio for the Y axis direction.

### Output Variables

Variable	Name	Data type	Valid range	Unit	Description
Out	Return Value	BOOL	Depends on data type.	-	Function execution results TRUE: Normal end FALSE: Error end

### In-Out Variables

Variable	Name	Data type	Valid range	Unit	Description
XYTraceData[]*1	XY Trace Data	ARRAY[*] OF LREAL	Depends on data type.	-	The trace data for broken line graph display.
XYGraphData[]	XY Graph Display Data	ARRAY[0..599] OF LREAL	Depends on data type.	-	Broken line graph display data.

\*1. The maximum number of array elements is 20,000. The first number of array element should be 0.

## Function

This function converts *XYTraceData[]* to *XYGraphData[]* (XY Graph Display Data), NS/NA-series HMI broken line graph display data.

When converting, you can specify the X axis direction conversion start position for the contents of *XYTraceData[]*. You can also zoom in on or zoom out from the X axis direction and Y axis direction.

### *XYTraceData[]* Structure

*XYTraceData[]* is prepared using the XYDataRec2 (Broken Line Graph Trace Data Preparation 2) function block. Refer to *XYDataRec2* on page 245 for details on the *XYTraceData[]* structure.

### *XYGraphData* Structure

The *XYGraphData[]* structure is the same as the *XYTraceData[]* structure. Specifically, the array element numbers express the X value for each data record and the array element values express the Y value for each data record.

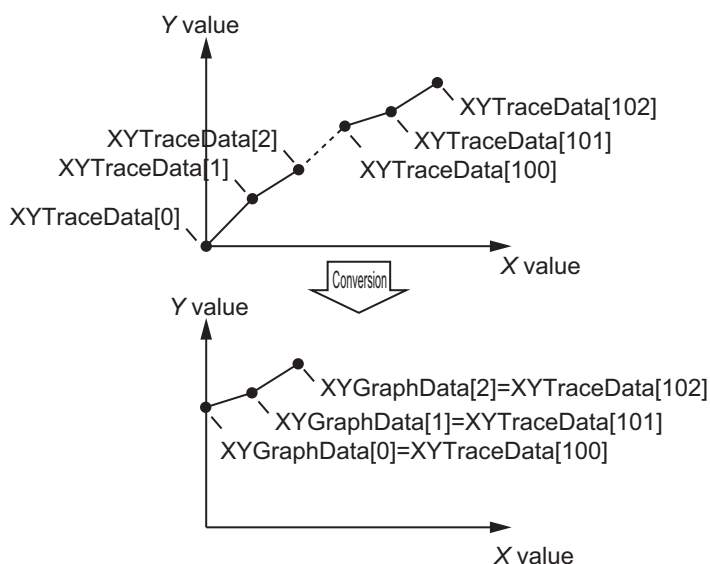
### Data Conversion

When you convert the data, you specify the X axis direction conversion start position and the X axis and Y axis direction zoom as described below.

#### ● X Axis Direction Conversion Start Position Specification

The X axis direction conversion start position is specified with *X\_Init* (X Axis Initial Position). For *X\_Init*, specify the *XYTraceData[]* array element number at which to start conversion.

For example, when *X\_Init* = UINT#100, *XYTraceData[100]* and higher are converted. The following graphs show the concept.



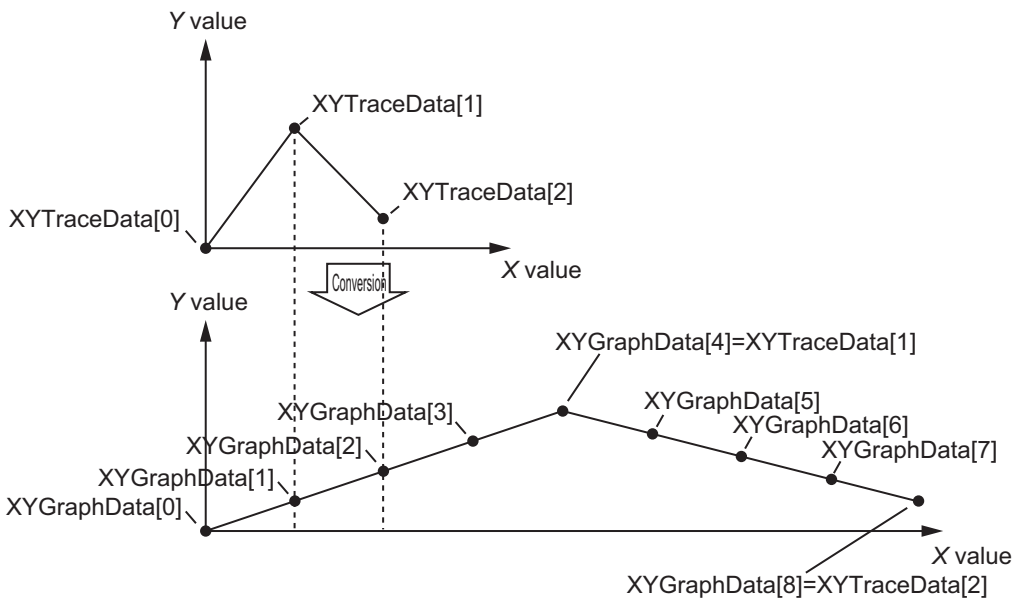
● X Axis Direction Zoom In

By changing the value of *X\_Zoom* (X Axis Zoom) to TRUE, you can convert *XYTraceData[]* so that it is zoomed in along the X axis direction. Specify the zoom-in ratio with *X\_ZoomNum* (X Axis Zoom Coefficient).

Y values are interpolated between two adjacent points in *XYTraceData[]*, and  $(X\_ZoomNum - 1)$  data records are added. The X values corresponding to the array element numbers in *XYTraceData[]* and *XYGraphData[]* are the same, so the *XYGraphData[]* graph is the *XYTraceData[]* graph multiplied by *X\_ZoomNum* in the X axis direction.

The number of data records in *XYGraphData[]* is  $(\text{the number of data records in } XYTraceData - 1) \times (X\_ZoomNum + 1)$ .

The following figure shows the relationship between *XYTraceData[]* and *XYGraphData[]* when there are three *XYTraceData[]* data records and *X\_ZoomNum* = UINT#4. The graph is multiplied by four in the X axis direction and the number of data records increases to nine.



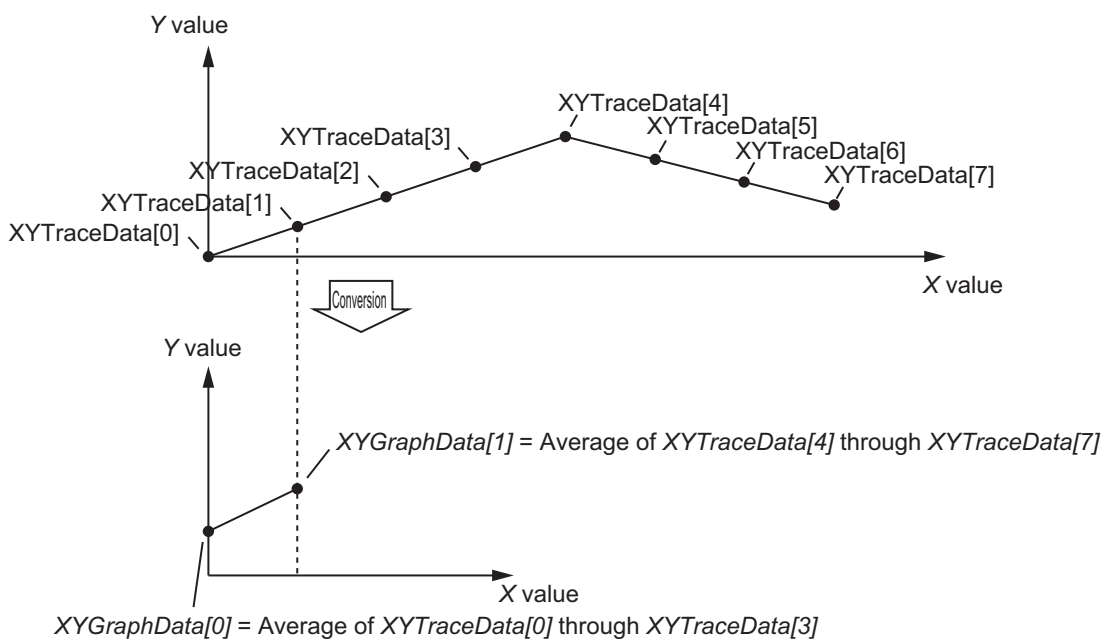
## ● X Axis Direction Zoom Out

By changing the value of  $X\_Zoom$  (X Axis Zoom) to FALSE, you can convert  $XYTraceData[]$  so that it is zoomed out along the X axis direction. Specify the zoom-out ratio with  $X\_ZoomNum$  (X Axis Zoom Coefficient). The zoom-out ratio is  $1/X\_ZoomNum$ .

The Y values of  $X\_ZoomNum$  of adjacent  $XYTraceData[]$  data records are averaged and merged into one data. The X values corresponding to the array element numbers in  $XYTraceData[]$  and  $XYGraphData[]$  are the same, so the  $XYGraphData[]$  graph is the  $XYTraceData[]$  graph divided by  $X\_ZoomNum$  in the X axis direction.

The number of data records in  $XYGraphData[]$  is (the number of data records in  $XYTraceData[]$  divided by  $X\_ZoomNum$ ).

The following figure shows the relationship between  $XYTraceData[]$  and  $XYGraphData[]$  when there are eight  $XYTraceData[]$  data records and  $X\_ZoomNum = UINT\#4$ . The graph is divided by four in the X axis direction and the number of data records decreases to two.



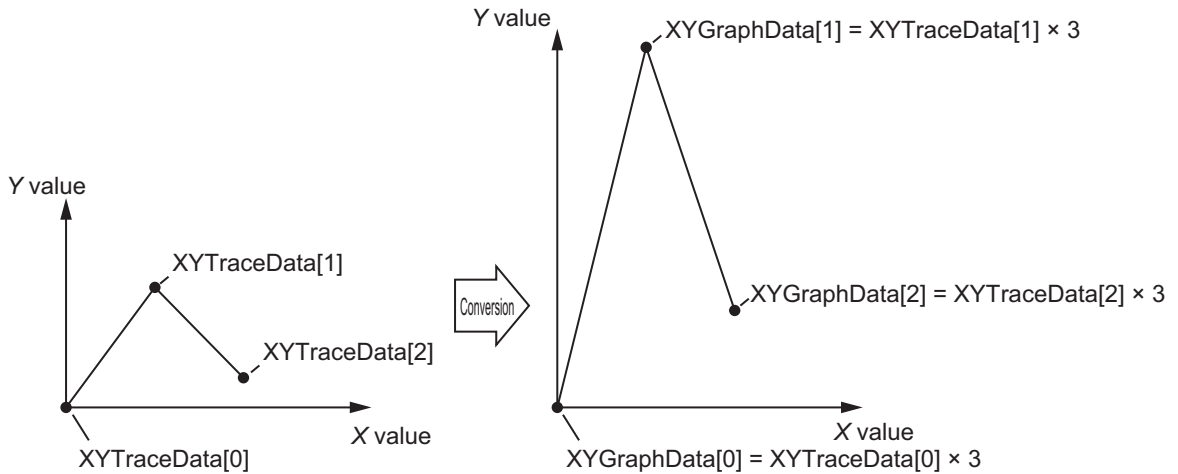
● **Y Axis Direction Zoom In**

By changing the value of *Y\_Zoom* (Y Axis Zoom) to TRUE, you can convert *XYTraceData[]* so that it is zoomed in along the Y axis direction. Specify the zoom-in ratio with *Y\_ZoomNum* (Y Axis Zoom Coefficient).

The Y value of *XYTraceData[]* multiplied by *Y\_ZoomNum* becomes the *XYGraphData[]* Y value.

The number of *XYGraphData[]* data records is the same as the number of *XYTraceData[]* data records.

The following figure shows the relationship between *XYTraceData[]* and *XYGraphData[]* when there are three *XYTraceData[]* data records and *Y\_ZoomNum* = UINT#3. The graph is multiplied by three in the Y axis direction and the number of data records remains three.



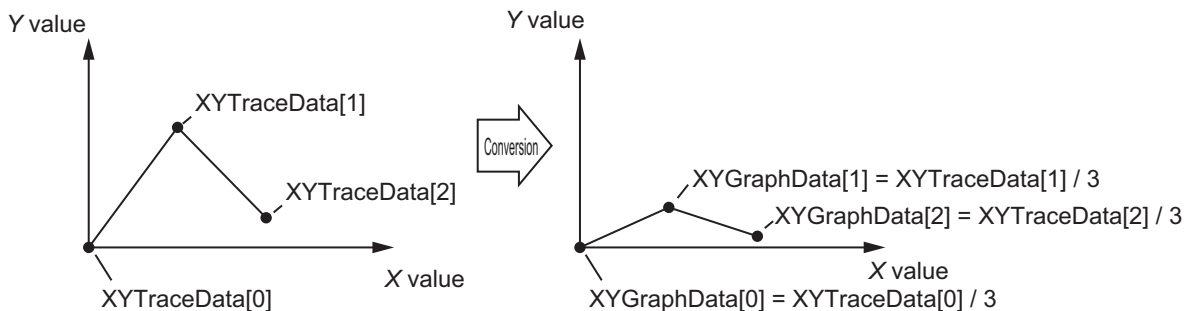
● **Y Axis Direction Zoom Out**

By changing the value of *Y\_Zoom* (Y Axis Zoom) to FALSE, you can convert *XYTraceData[]* so that it is zoomed out along the Y axis direction. Specify the zoom-out ratio with *Y\_ZoomNum* (Y Axis Zoom Coefficient). The zoom-out ratio is  $1/Y\_ZoomNum$ .

The Y value of *XYTraceData[]* divided by *Y\_ZoomNum* becomes the *XYGraphData[]* Y value.

The number of *XYGraphData[]* data records is the same as the number of *XYTraceData[]* data records.

The following figure shows the relationship between *XYTraceData[]* and *XYGraphData[]* when there are three *XYTraceData[]* data records and *Y\_ZoomNum* = UINT#3. The graph is divided by three in the Y axis direction and the number of data records remains three.





## Additional Information

To prepare *XYTraceData[]* data, use the function block of *XYDataRec2* on page 245.

## Precautions for Correct Use

If the value of an input variable is out of range, an error occurs and the value of *Out* changes to FALSE. The values of *XYGraphData[]* (XY Graph Display Data) are not updated.

## Sample Programming

This sample programming extracts 5,000 records of data from the 1,000th record of the 20,000 records of broken-line graph display trace data created with the *XYDataRec2* (Broken Line Graph Trace Data Preparation 2) function block, to create 600 records of display data suitable for displaying on an NS/NA-series PT.

It is added and executed after the *SP\_SingleAxisPrgOpr* (Single-axis Program Operation) function block sample programming and *XYDataRec2* (Broken Line Graph Trace Data Preparation 2) function block sample programming.



### Precautions for Correct Use

- The sample programming shows only the portion of a program that uses the function or function block from the library.
- When you use the actual device, include user programming for device safety instructions, interlocks, I/O with other devices, and other control procedures.
- Create a user program so that the actual device operates as intended.
- Check the user program for proper execution before you use it for actual operation.

## Condition

The conditions are the same as those for the *SP\_SingleAxisPrgOpr* (Single-axis Program Operation) function block sample programming.

## Processing

The graph drawing data is created with the specified start point (array index) and scale ratio when *XY\_Graph\_StartPg* (Creation Trigger for Graph Drawing Data) changes to TRUE.

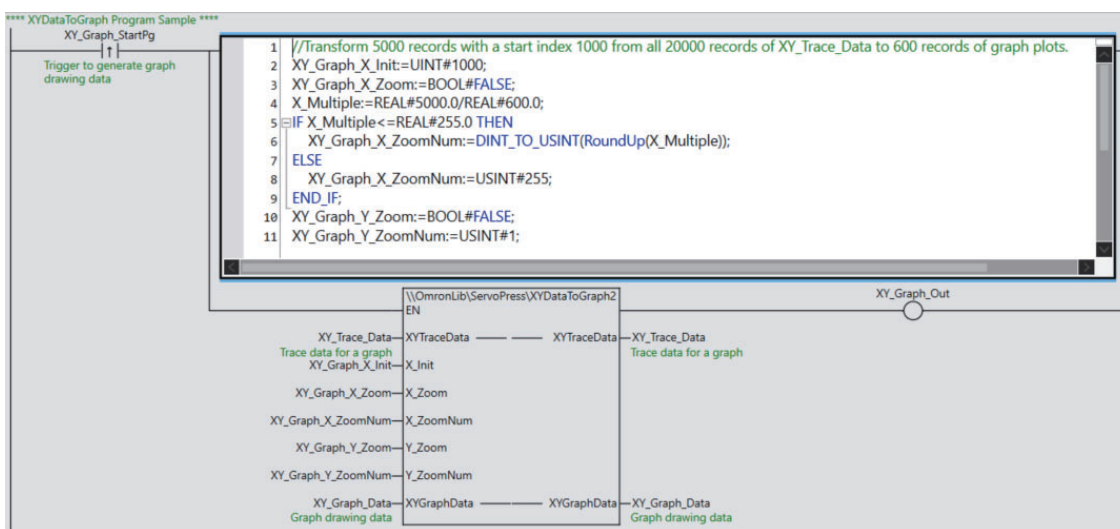
## Ladder Diagram

The following gives the main variables.

### ● Internal Variables

Name	Data type	Initial value	Comment
XY_Trace_Data	ARRAY[0..19999] OF LREAL		Trace data for graphs
XY_Graph_StartPg	BOOL		Creation trigger for graph drawing data
XY_Graph_Data	ARRAY[0..599] OF LREAL		Graph drawing data

### ● Algorithm



## ST

The following gives the main variables.

### ● Internal Variables

Name	Data type	Initial value	Comment
XY_Trace_Data	ARRAY[0..19999] OF LREAL		Trace data for graphs
XY_Graph_StartPg	BOOL		Creation trigger for graph drawing data
XY_Graph_Data	ARRAY[0..599] OF LREAL		Graph drawing data

### ● Algorithm

```
//Generate graph plots data.
IF UpTrig_XY_Graph_StartPg=TRUE THEN
  //Transform 5000 records with a start index 1000 from all 20000 records of
  XY_Trace_Data to 600 records of graph plots.
  XY_Graph_X_Init:=UINT#1000;
  XY_Graph_X_Zoom:=BOOL#FALSE;
  X_Multiple:=REAL#5000.0/REAL#600.0;
  IF X_Multiple<=REAL#255.0 THEN
    XY_Graph_X_ZoomNum:=DINT_TO_USINT(RoundUp(X_Multiple));
  ELSE
    XY_Graph_X_ZoomNum:=USINT#255;
  END_IF;
  XY_Graph_Y_Zoom:=BOOL#FALSE;
  XY_Graph_Y_ZoomNum:=USINT#1;
  END_IF;
  //XYGraphToData
  XY_Graph_Out:=\\OmronLib\ServoPress\XYDataToGraph2(
    EN := UpTrig_XY_Graph_StartPg,
    XYTraceData := XY_Trace_Data,
    X_Init := XY_Graph_X_Init,
    X_Zoom := XY_Graph_X_Zoom,
    X_ZoomNum := XY_Graph_X_ZoomNum,
    Y_Zoom := XY_Graph_Y_Zoom,
    Y_ZoomNum := XY_Graph_Y_ZoomNum,
    XYGraphData := XY_Graph_Data
  );
```



# Appendix

# Referring to Library Information

When you make an inquiry to OMRON about the library, you can refer to the library information to identify the library to ask about.

The library information is useful in identifying the target library among the libraries provided by OMRON or created by the user.

The library information consists of the attributes of the library and the attributes of function blocks and functions contained in the library.

- Attributes of libraries  
Information for identifying the library itself
- Attributes of function blocks and functions  
Information for identifying the function block and function contained in the library

Use the Sysmac Studio to access the library information.

## Attributes of Libraries, Function Blocks and Functions

The following attributes of libraries, function blocks and functions are provided as the library information.

### ● Attributes of Libraries

No.*1	Attribute	Description
(1)	Library file name	The name of the library file
(2)	Library version	The version of the library
(3)	Author	The name of creator of the library
(4)	Comment	The description of the library*2

\*1. These numbers correspond to the numbers shown on the screen images in the next section, *Referring to Attributes of Libraries, Function Blocks and Functions* on page 277.

\*2. It is provided in English and Japanese.

### ● Attributes of Function Blocks and Functions

No.*1	Attribute	Description
(5)	FB/FUN name	The name of the function block or function
(6)	Name space	The name of name space for the function block or function
(7)	FB/FUN version	The version of the function block or function
(8)	Author	The name of creator of the function block or function
(9)	FB/FUN number	The function block number or function number
(10)	Comment	The description of the function block or function*2

\*1. These numbers correspond to the numbers shown on the screen images in the next section, *Referring to Attributes of Libraries, Function Blocks and Functions* on page 277.

\*2. It is provided in English and Japanese.

## Referring to Attributes of Libraries, Function Blocks and Functions

You can refer to the attributes of libraries, function blocks and functions of the library information at the following locations on the Sysmac Studio.

- Library Reference Dialog Box
- Toolbox Pane
- Ladder Editor

### (a) Library Reference Dialog Box

When you refer to the libraries, the library information is displayed at the locations shown below.

(1)Library file name      (2)Library version      (3)Library author      (4)Library comment

Library name	Name Space	Version	Author	Company	Date Creat	Date Modi	Comment
OmronLib_MC_Toolbox_V1_1		1.1.0	OMRON Corporation	(c)OMRON Corporation 2015. All Rights Reserved.			This is MC Toolbox library. これはモーション制御ツールボックスライ
POU							
Programs							
Functions							
DeadBand (OmronLib_MC_Toolbox)	OmronLib_MC_Toolbo	1.1.0	OMRON Corporation		03/16/2015	08/10/201	No.00006 The DeadBand function block cont 処理結果にオフセットが発生させないデ
FirstOrderlag (OmronLib_MC_Toolbox)	OmronLib_MC_Toolbo	1.1.0	OMRON Corporation		04/01/2015	08/10/201	No.00004 The FirstOrderLag function block p 設定されたパラメータテーブルに従って、
LeadLag (OmronLib_MC_Toolbox)	OmronLib_MC_Toolbo	1.1.0	OMRON Corporation		04/01/2015	08/10/201	No.00005 The LeadLag function block perfor 設定されたパラメータテーブルに従って、
PIDFeedFwd (OmronLib_MC_Toolbox)	OmronLib_MC_Toolbo	1.1.0	OMRON Corporation		04/01/2015	08/10/201	No.00003 The PIDFeedFwd function block pe 設定されたパラメータテーブルに従って、

(5)FB/FUN name      (6)Name space      (7)FB/FUN version      (8)FB/FUN author      (10)FB/FUN comment

Namespace - Using

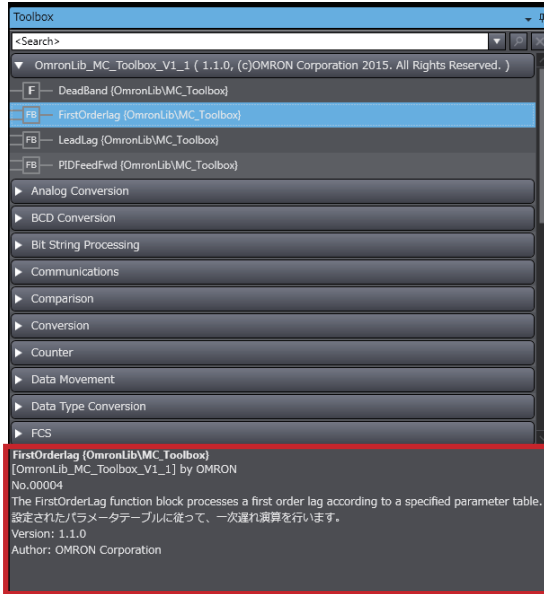
In/Out	Name	In/Out	Data Typel	Edge	Initial Value	Retain	Constant	Comment
Externals	Enable	Input	BOOL	No Edge	False	<input type="checkbox"/>	<input type="checkbox"/>	
	InCalc	Input	LREAL	No Edge	0.0	<input type="checkbox"/>	<input type="checkbox"/>	
	Kp	Input	LREAL	No Edge	1.0	<input type="checkbox"/>	<input type="checkbox"/>	
	TimeConst	Input	LREAL	No Edge	1.0	<input type="checkbox"/>	<input type="checkbox"/>	
	SampTime	Input	LREAL	No Edge	1.0	<input type="checkbox"/>	<input type="checkbox"/>	
	Enabled	Output	BOOL	No Edge		<input type="checkbox"/>	<input type="checkbox"/>	

OK

(b) Toolbox Pane

Select a function block and function to display its library information at the bottom of the Toolbox Pane.

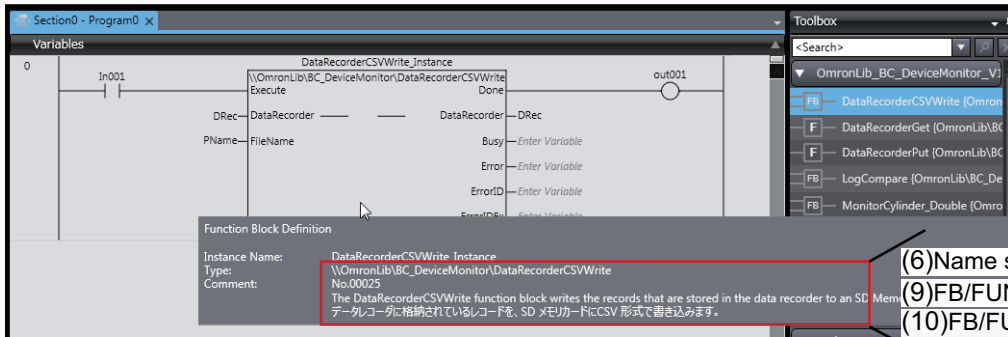
The text “by OMRON” which is shown on the right of the library name (1) indicates that this library was provided by OMRON.



- (5)FB/FUN name (6)Name space
- (1)Library file name
- (9)FB/FUN number
- (10)FB/FUN comment
- (7)FB/FUN version
- (8)FB/FUN author

(c) Ladder Editor

Place the mouse on a function block and function to display the library information in a tooltip.



- (6)Name space (5)FB/FUN name
- (9)FB/FUN number
- (10)FB/FUN comment



# Referring to Function Block and Function Source Codes

You can refer to the source codes of function blocks and functions provided by OMRON to customize them to suit the user's environment.

User function blocks and user functions can be created based on the copies of these source codes.

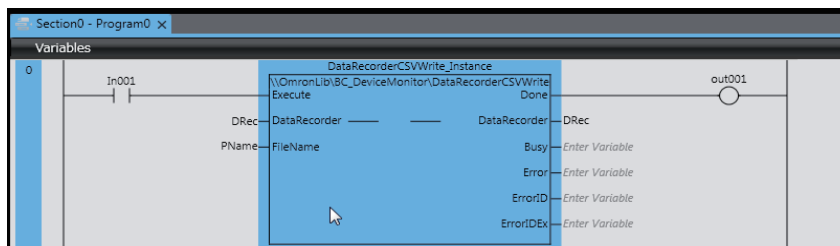
The following are the examples of items that you may need to customize.

- Customizing the size of arrays to suit the memory capacity of the user's Controller
- Customizing the data types to suit the user-defined data types

Note that you can access only function blocks and functions whose Source code published/not published is set to Published in the library information shown in their individual specifications.

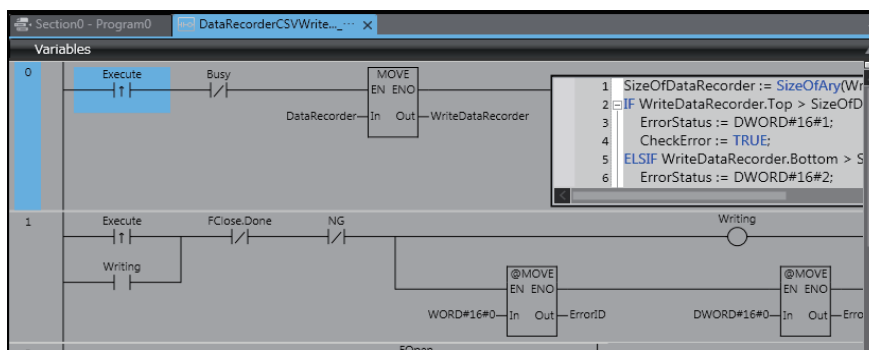
Use the following procedure to refer to the source codes of function blocks and functions.

- 1 Select a function block or function in the program.



- 2 Double-click or right-click and select **To Lower Layer** from the menu.

The source code is displayed.



## Precautions for Correct Use

For function blocks and functions whose source codes are not published, the following dialog box is displayed in the above step 2. Click the **Cancel** button.







**OMRON Corporation** Industrial Automation Company  
Kyoto, JAPAN

Contact: [www.ia.omron.com](http://www.ia.omron.com)

**Regional Headquarters**

**OMRON EUROPE B.V.**

Wegalaan 67-69, 2132 JD Hoofddorp  
The Netherlands  
Tel: (31)2356-81-300/Fax: (31)2356-81-388

**OMRON ELECTRONICS LLC**

2895 Greenspoint Parkway, Suite 200  
Hoffman Estates, IL 60169 U.S.A.  
Tel: (1) 847-843-7900/Fax: (1) 847-843-7787

**OMRON ASIA PACIFIC PTE. LTD.**

No. 438A Alexandra Road # 05-05/08 (Lobby 2),  
Alexandra Technopark,  
Singapore 119967  
Tel: (65) 6835-3011/Fax: (65) 6835-2711

**OMRON (CHINA) CO., LTD.**

Room 2211, Bank of China Tower,  
200 Yin Cheng Zhong Road,  
PuDong New Area, Shanghai, 200120, China  
Tel: (86) 21-5037-2222/Fax: (86) 21-5037-2200

**Authorized Distributor:**

© OMRON Corporation 2016-2019 All Rights Reserved.  
In the interest of product improvement,  
specifications are subject to change without notice.

Cat. No. **W573-E1-03**

0119