

OMRON

Sysmac Library

User's Manual for Device Operation Monitor Library SYSMAC-XR008

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 Device Operation Monitor 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 Device Operation Monitor Library is used to monitor the operation of devices such as solenoid valves, motors, and other devices. You can use this library to reduce manpower of programming when implementing the processing for each device.

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

Introduction	1
Features of the Library.....	1
Intended Audience	1
Applicable Products.....	1
Manual Structure	2
Special Information.....	2
CONTENTS.....	4
Terms and Conditions Agreement	6
Warranty, Limitations of Liability	6
Application Considerations	7
Disclaimers	7
Safety Precautions	8
Precautions for Correct Use.....	10
Related Manuals	11
Revision History	14
Procedure to Use Sysmac Libraries	15
Procedure to Use Sysmac Libraries Installed Using the Installer	16
Procedure to Use Sysmac Libraries Uploaded from a CPU Unit or an Industrial PC.....	20
Device Operation Monitor Library	23
Purpose of Device Operation Monitor Library.....	24
Applicable Applications	25
Common Specifications of Function Blocks	33
Common Variables	34
Precautions.....	40
Individual Specifications of Function Blocks	41
MonitorCylinder_Measure.....	42
MonitorCylinder_Double	52
MonitorCylinder_Single.....	63
LogCompare	71
LogDataToGraph	87
LogDataCSVWrite	94
LogDataCSVRead	102
MonitorLightSensor.....	109
Stopwatch	116
DataRecorderPut	120
DataRecorderGet.....	124
DataRecorderCSVWrite.....	126
AxisRecorderPut.....	136
AxisRecorderGet	140
AxisRecorderCSVWrite	142
BitRecorderPut	150
BitRecorderGet.....	154
BitRecorderToGraph	156
Appendix	167
Referring to Library Information.....	168

Referring to Function Block and Function Source Codes	171
---	-----

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.

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

Cautions

Caution

Read all related manuals carefully before you use this library.



Emergency stop circuits, interlock circuits, limit circuits, and similar safety measures must be provided in external control circuits.



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



Perform the test run by holding an emergency stop switch in hand or otherwise prepare for rapid motor operation in an application to control the motor.

Also perform the test run by using the parameters for which the motor does not rapidly accelerate or decelerate before you gradually adjust the parameters.



In an application of heating or cooling, perform the test run by using the parameters for which rapid temperature changes will not occur before you gradually adjust the parameters.



You must confirm that the user program and parameter values are appropriate to the specifications and operation methods of the devices.



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.



Precautions for Correct Use

Using the Library

- When you use the library, functions or function blocks that are not described in the library manual may be displayed on the Sysmac Studio. Do not use functions or function blocks that are not described in the manual.
- You cannot change the source code of the functions or function blocks that are provided in the Sysmac Library.
- The multi-execution (buffer mode) cannot be performed in the Sysmac Library.

Using Sample Programming

- 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.

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.
- In the function block with Execute, do not perform re-execution by the same instance. The output value of the function block will return to the default value.

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 Introduction Part names and functions General specifications Installation and wiring Maintenance and Inspection
NX-series NX1P2 CPU Unit Hardware User's Manual	W578	NX1P2-□□□□	Learning the basic specifications of the 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
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.

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. W552-E1-05

↑
Revision code

Revision code	Date	Revised content
01	December 2015	Original production
02	July 2016	Changed the manual name.
03	November 2016	Changed the manual name.
04	January 2019	Added compatible models.
05	May 2020	Corrected mistakes.

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

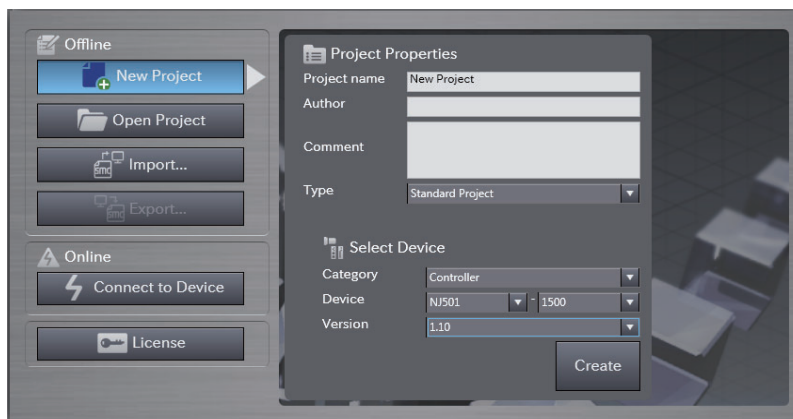


Version Information

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.

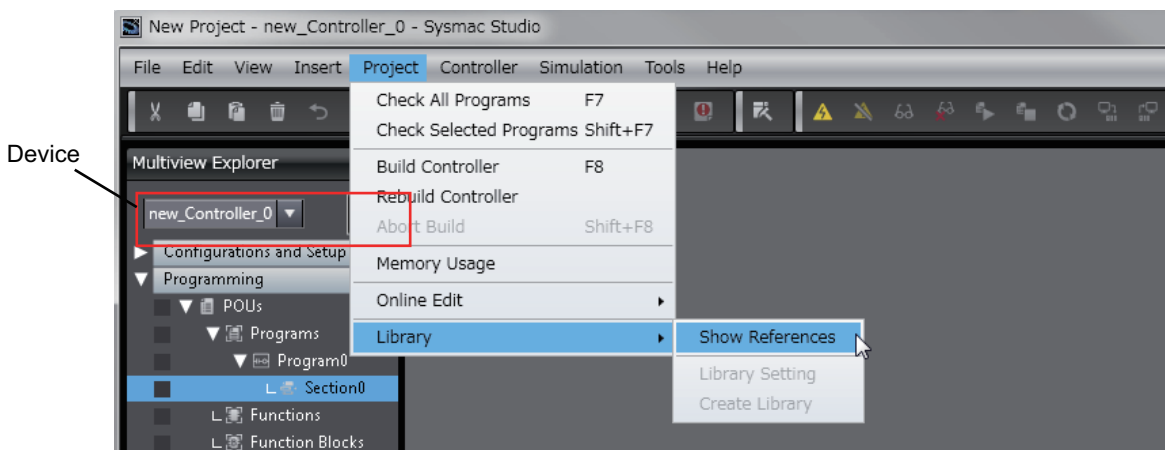


Precautions for Correct Use


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.



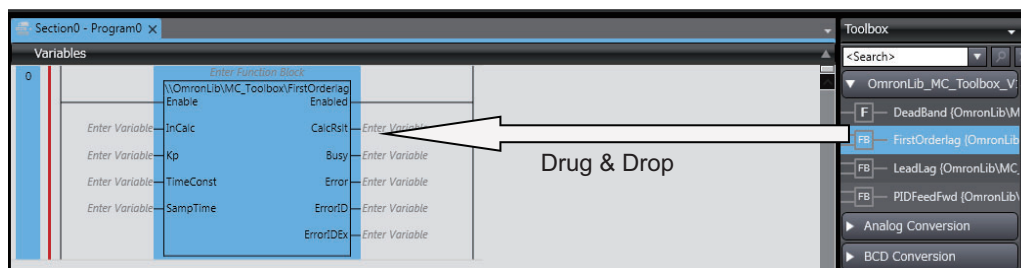
The Sysmac Library file is read into the project.

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

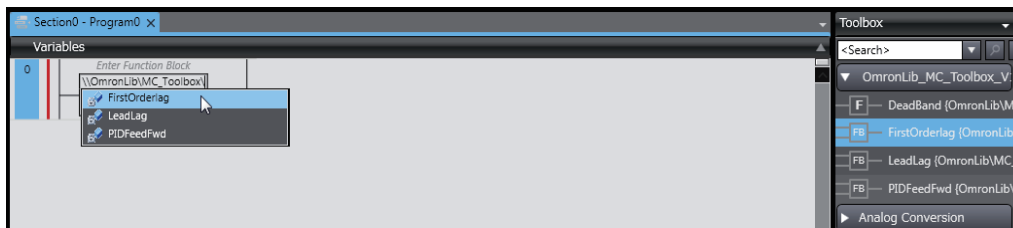
For the procedure for adding and setting libraries in the above screen, refer to the *Sysmac Studio Version 1 Operation Manual (Cat. No. W504)*.

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).



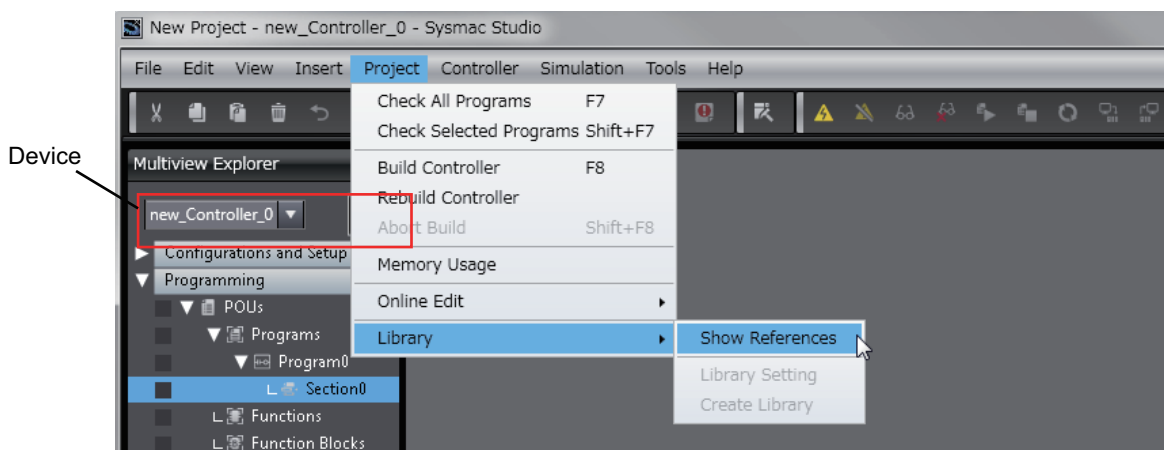
Precautions for Correct Use

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

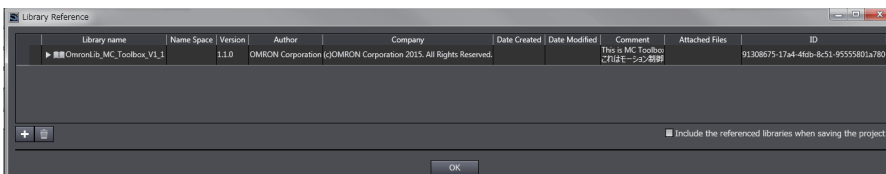
- 1 Start the Sysmac Studio and open a project in which any old-version Sysmac Library is included.
- 2 Select **Project – Library – Show References**.



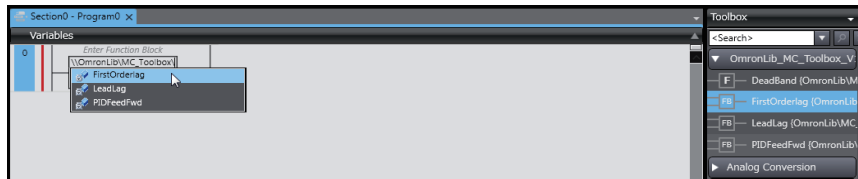
Precautions for Correct Use

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

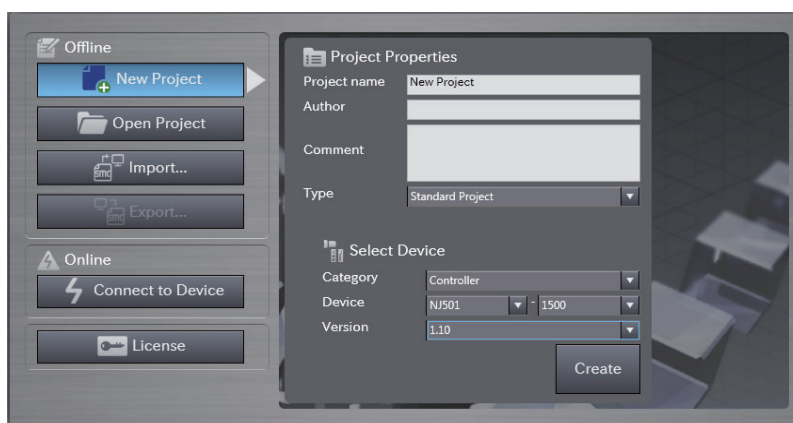
You can use Sysmac Libraries uploaded from a CPU Unit or an Industrial PC to your computer if they are not installed.

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.



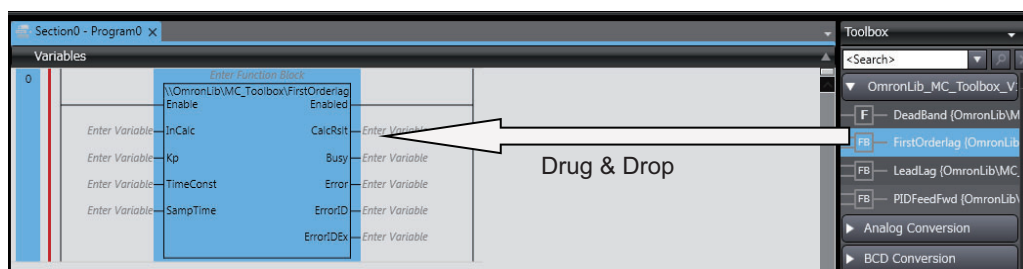
- 2 Connect the computer to the CPU Unit or the Industrial PC and place it online.

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

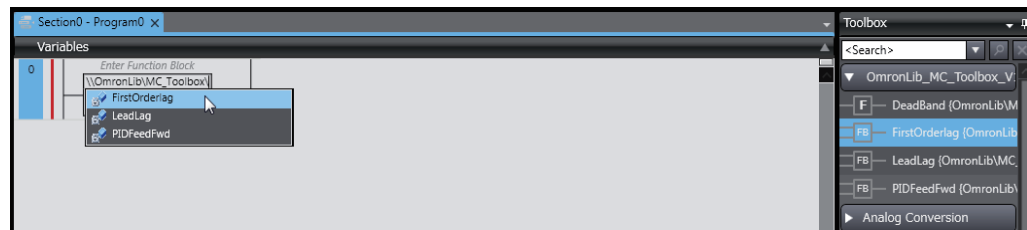
Now, when you select the Ladder Editor or ST Editor, the function blocks and functions included in the Sysmac Library used in the uploaded POUs appear in the Toolbox.

- 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.

Device Operation Monitor Library

Purpose of Device Operation Monitor Library

The purpose of the Device Operation Monitor Library is to monitor the operation of each equipment in the automated facility and implement the self-diagnosis function.

The input information from cylinders and other I/O equipment are compared with the preset reference values to detect machine and equipment errors. An alarm is output when machine and equipment errors are detected. Also, output values in case of an error are recorded and used to help analyze the cause of the error.

The Device Operation Monitor Library consists of multiple function blocks or functions. Use a combination of function blocks and functions that is most suitable for your application.

Applicable Applications

Function blocks and functions designed for the following applications are available for the Device Operation Monitor Library.

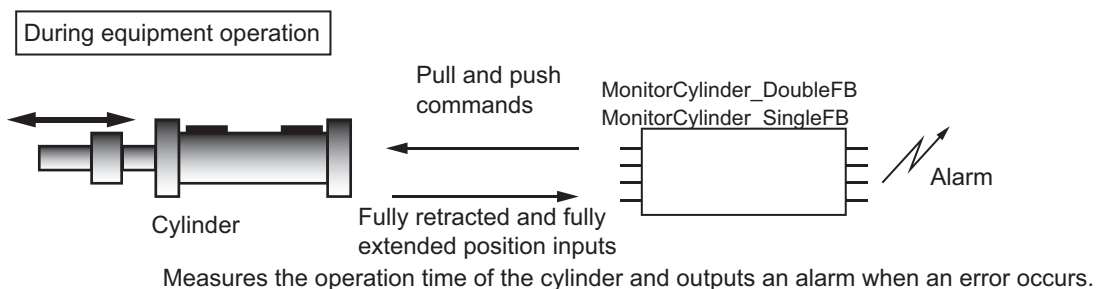
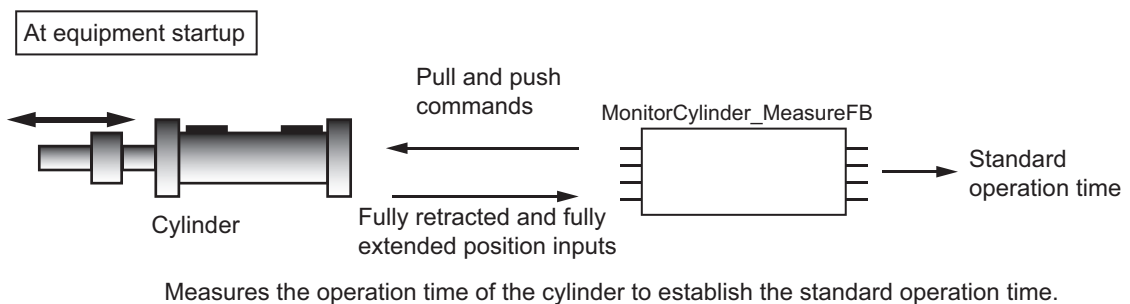
- Monitoring cylinder operations
- Monitoring photoelectric sensor operations
- Monitoring mechanical component operations
- Logging variables
- Displaying as graphs
- Stopwatch

Details of each application are described below.

Monitoring Cylinder Operations

The function block monitors the operation time of the cylinder to detect cylinder operation errors that occur due to deterioration of a cylinder or air pressure system errors.

When the equipment is started, the normal operation time is measured and the standard operation time to determine an error is established. When the operation time of the equipment exceeds the upper and lower limits of the standard operation time, it is determined to be an error and an alarm is output.



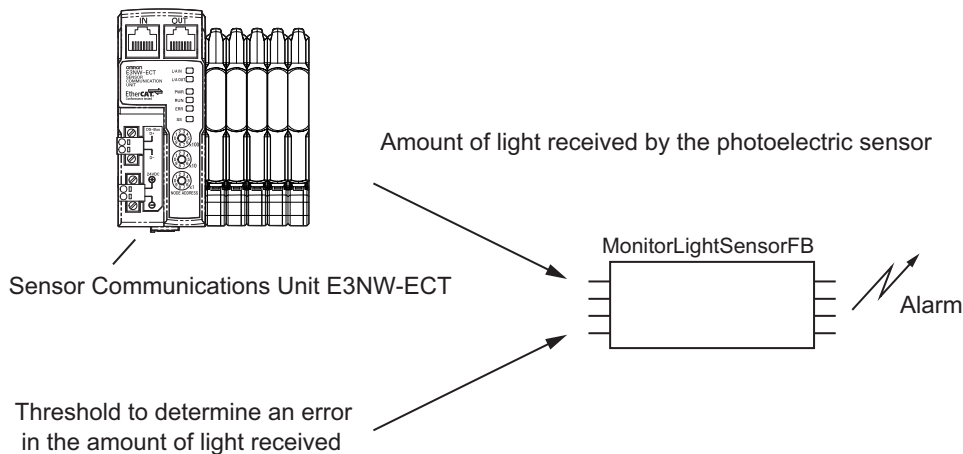
The related function blocks and functions are as follows.

Function block name	Function	Page
MonitorCylinder_Measure	Measures the operation time of the cylinder in normal conditions to establish the standard operation time to determine an error.	<i>MonitorCylinder_Measure</i> on page 42
MonitorCylinder_Double	Monitors the operation time of the cylinder during normal operation and outputs an alarm when an error occurs. It uses push and pull command signals.	<i>MonitorCylinder_Double</i> on page 52
MonitorCylinder_Single	Monitors the operation time of the cylinder during normal operation and outputs an alarm when an error occurs. It only uses the push command.	<i>MonitorCylinder_Single</i> on page 63

Monitoring Photoelectric Sensor Operations

The function block monitors the amount of light received by the photoelectric sensor to detect operation errors of the photoelectric sensor due to a soiled lens or other causes.

The threshold to determine an error in the amount of light received is preset and when the light received is repeatedly at or below the threshold, an alarm is output.



When the light received by the photoelectric sensor is repeatedly at or below the threshold to determine an error, an alarm is output.

The related function blocks and functions are as follows.

Function block name	Function	Page
MonitorLightSensor	Measures the amount of light received by the photoelectric sensor and outputs an alarm when an error occurs.	<i>MonitorLightSensor</i> on page 109

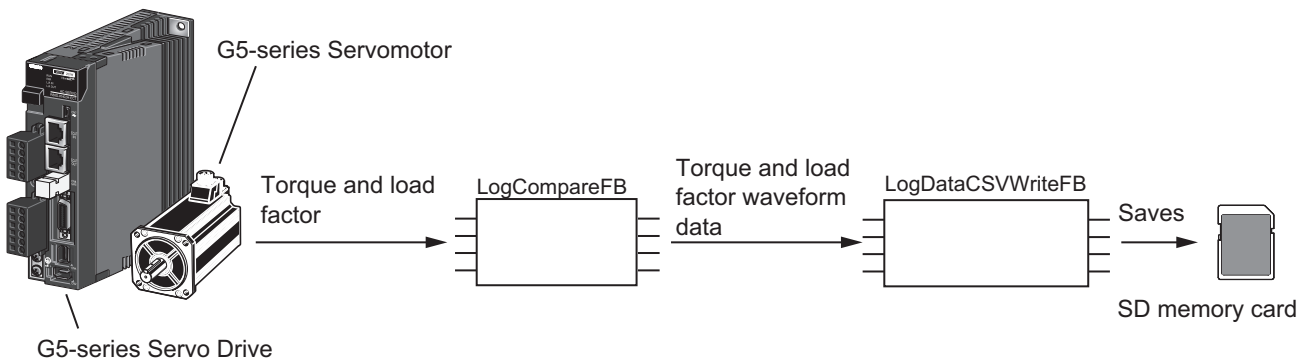
Monitoring Mechanical Component Operations

The function block monitors the torque and load factor of mechanical components to detect operation errors due to damage, wear, and foreign matter on mechanical components.

Torque and load factor waveforms are recorded when equipment is started and compared with torque and load factor waveforms during operation. When the difference between these waveforms exceeds the preset tolerance value, it is determined to be an error and an alarm is output.

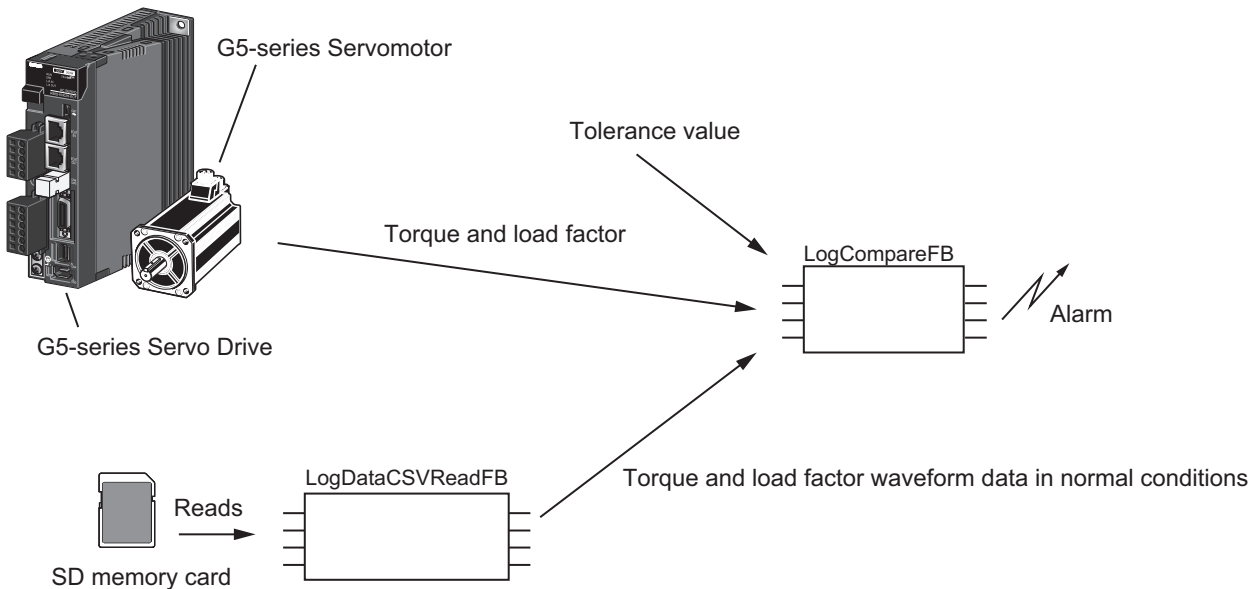
You can save the recorded waveform data in an SD memory card in CSV format.

At equipment startup



Acquires torque and load factor waveform data and saves in an SD memory card.

During equipment operation



Acquires torque and load factor waveform data. When the difference from the torque and load factor waveform data in normal conditions exceeds the tolerance value, an alarm is output.

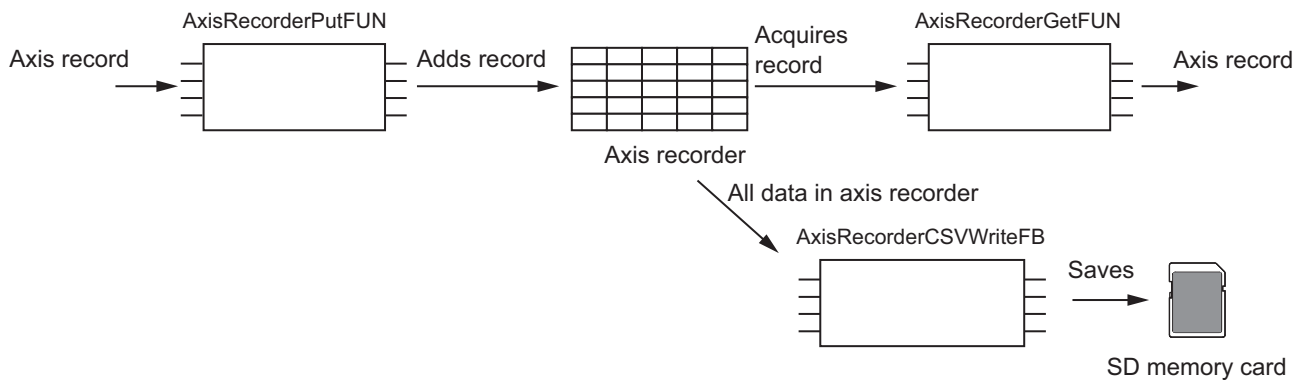
The related function blocks and functions are as follows.

Function block name	Function	Page
LogCompare	Compares the Servomotor torque and load factor waveforms in normal conditions with torque and load factor waveforms during normal operation and outputs an alarm when an error occurs.	<i>LogCompare</i> on page 71
LogDataCSVWrite	Saves the waveform data in an SD memory card in CSV format.	<i>MonitorLightSensor</i> on page 109
LogDataCSVRead	Reads the waveform data from an SD memory card.	<i>LogDataCSVRead</i> on page 102

Logging Variables

The function block logs and adds variables to the recorder and acquires variables from the recorder. You can save the contents of the recorder in an SD memory card in CSV format. You can log axis records, bit records, and general variables according to the type of the variable to log.

Logging axis records

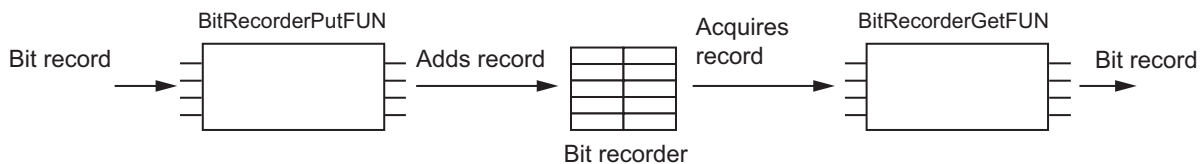


Structure of Axis Recorder

Record Time	Command Current Position	Command Current Velocity	Actual Current Position	Actual Current Velocity	Actual Current Torque
:	:	:	:	:	:
Record Time	Command Current Position	Command Current Velocity	Actual Current Position	Actual Current Velocity	Actual Current Torque

Adds axis records to the axis recorder and acquires axis records from the axis recorder.
Saves the contents of the axis recorder in an SD memory card in CSV format.

Logging bit records

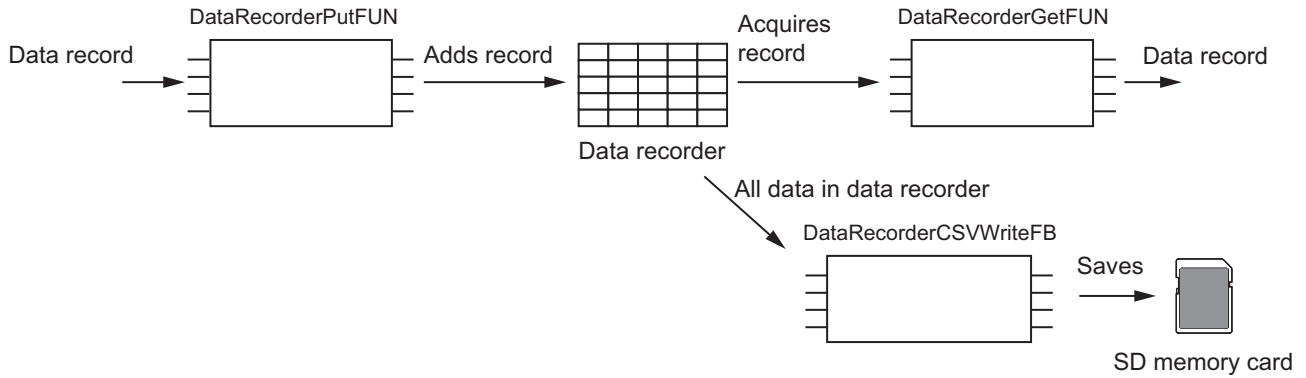


Structure of Bit Recorder

Record Time	32 BOOL data
:	:
Record Time	32 BOOL data

Adds bit records to the bit recorder and acquires bit records from the bit recorder.

Logging general variables



Structure of Data Recorder

Record Time	32 BOOL data	10 INT data	10 DINT data	10 REAL data
:	:	:	:	:
Record Time	32 BOOL data	10 INT data	10 DINT data	10 REAL data

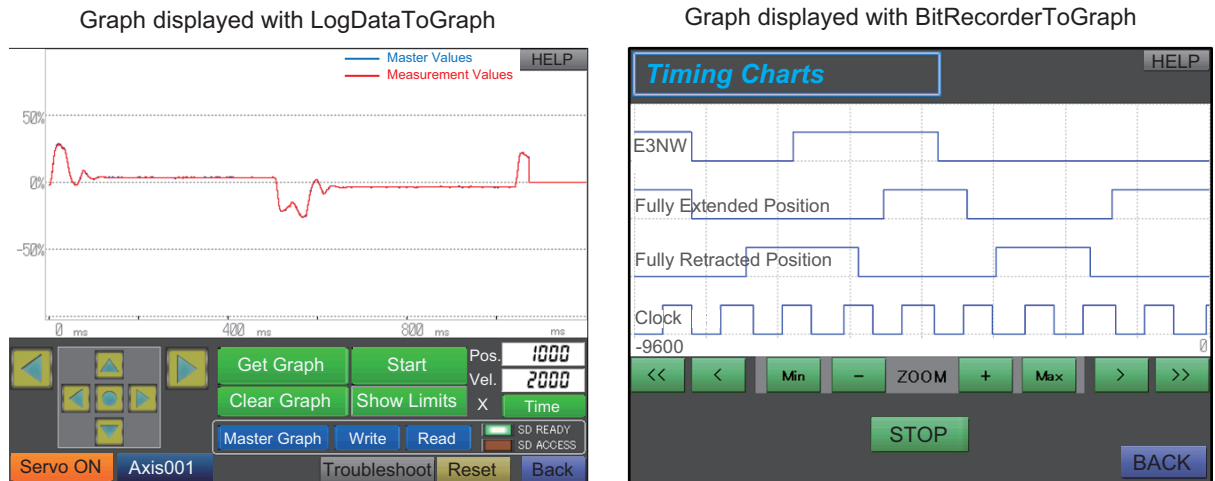
Adds data records to the data recorder and acquires data records from the data recorder. Saves the contents of the data recorder in an SD memory card in CSV format.

The related function blocks and functions are as follows.

Function block name	Function	Page
AxisRecorderPut	Stores axis records in the axis recorder.	<i>AxisRecorderPut</i> on page 136
AxisRecorderGet	Acquires axis records from the axis recorder.	<i>AxisRecorderGet</i> on page 140
AxisRecorderCSVWrite	Saves the contents of the axis recorder in an SD memory card in CSV format.	<i>AxisRecorderCSVWrite</i> on page 142
BitRecorderPut	Stores bit records in the bit recorder.	<i>BitRecorderPut</i> on page 150
BitRecorderGet	Acquires bit records from the bit recorder.	<i>BitRecorderGet</i> on page 154
DataRecorderPut	Stores general data records in the data recorder.	<i>DataRecorderPut</i> on page 120
DataRecorderGet	Acquires general data records from the data recorder.	<i>DataRecorderGet</i> on page 124
DataRecorderCSVWrite	Saves the contents of the general recorder in an SD memory card in CSV format.	<i>DataRecorderCSVWrite</i> on page 126

Displaying as Graphs

The function block converts data that was acquired with Servomotor monitoring function or variable logging function to the data format that is suitable for displaying as a graph on NS-series PT. You can enlarge or reduce the size of waveforms to be displayed.



The related function blocks and functions are as follows.

Function block name	Function	Page
LogDataToGraph	Converts data that was acquired with Servomotor monitoring function to the data format that is suitable for displaying as a graph on NS-series PT.	<i>LogDataToGraph</i> on page 87
BitRecorderToGraph	Converts bit records that were acquired with variable logging function to the data format that is suitable for displaying as a graph on NS-series PT.	<i>BitRecorderToGraph</i> on page 156

Stopwatch

The function block measures the time difference between the rising edges of 2 types of input signals. This function block is used to measure the transit time of the mobile test target or Takt time of manufacturing lines.

The related function blocks and functions are as follows.

Function block name	Function	Page
Stopwatch	Measures the time difference between the rising edges of 2 types of input signals.	<i>Stopwatch</i> on page 116

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

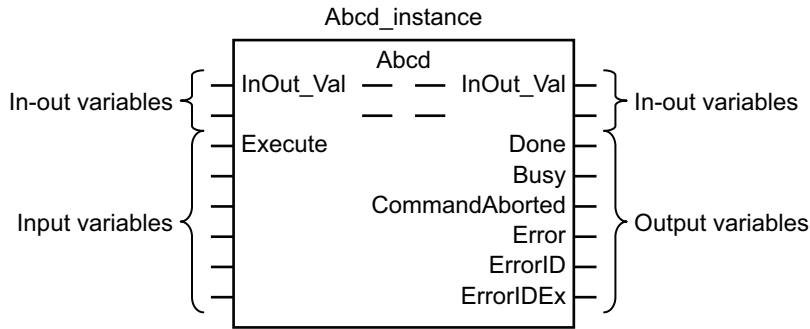
Common input variables and output variables used in functions and function blocks are as follows.

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.

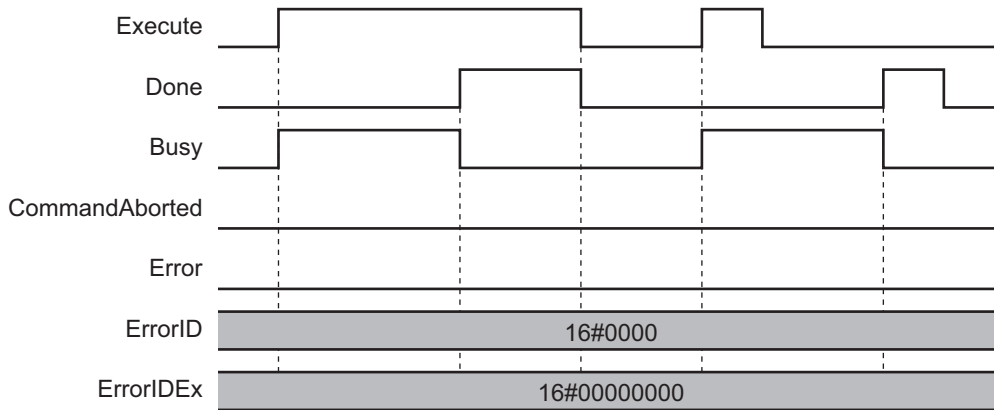


- If *Execute* is TRUE and *Done*, *CommandAborted*, or *Error* changes to TRUE, *Done*, *CommandAborted*, and *Error* changes to FALSE when *Execute* is changed to FALSE.
- If *Execute* is FALSE and *Done*, *CommandAborted*, or *Error* changes to TRUE, *Done*, *CommandAborted*, and *Error* changes to 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.

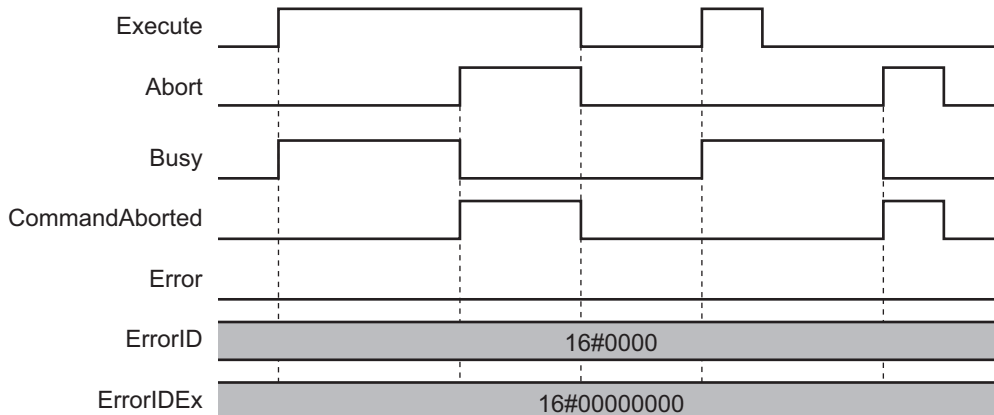
Timing Charts

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

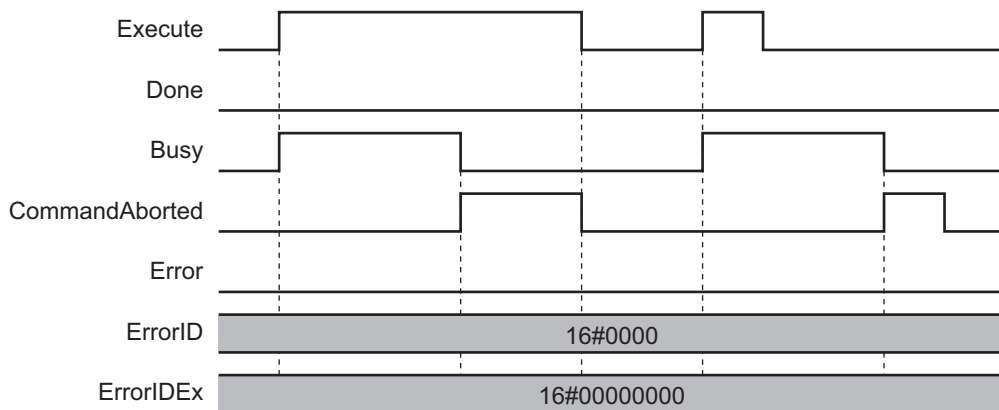
● Normal End



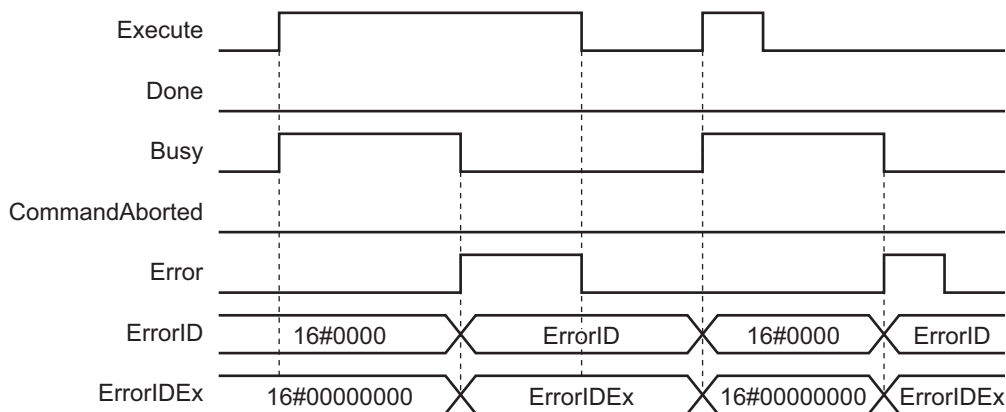
● Canceled Execution



● **Aborted Execution**

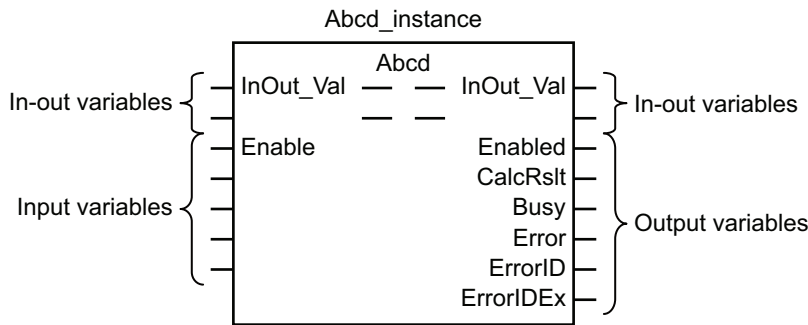


● **Errors**



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.

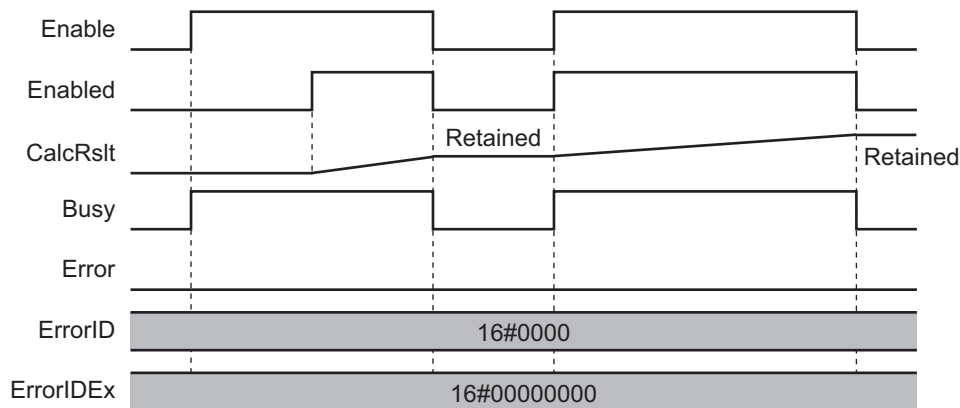


- If an error occurs, the relevant error code and expansion error code are set in *ErrorID* (Error Code) and *ErrorIDEx* (Expansion Error Code). The error codes are retained even after *Error* changes to FALSE, but *ErrorID* is set to 16#0000 and *ErrorIDEx* is set to 16#0000 0000 when *Enable* changes 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.

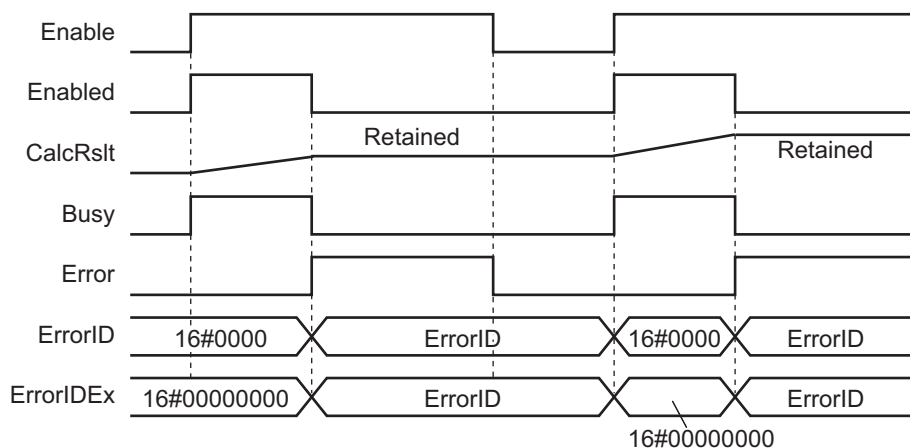
Timing Charts

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
MonitorCylinder_Measure	Monitor Cylinder Device Operation (Measure)	P.42
MonitorCylinder_Double	Monitor Cylinder Device Operation (Double)	P.52
MonitorCylinder_Single	Monitor Cylinder Device Operation (Single)	P.63
LogCompare	Logging Compare	P.71
LogDataToGraph	Display Log Data	P.87
LogDataCSVWrite	Write Log Data to SD Memory Card	P.94
LogDataCSVRead	Read Log Data from SD Memory Card	P.102
MonitorLightSensor	Monitor Photoelectric Sensor Device Operation	P.109
Stopwatch	Measure Cycle Time	P.116
DataRecorderPut	Add Data Record	P.120
DataRecorderGet	Get Data Record	P.124
DataRecorderCSVWrite	Write from Data Recorder to SD Memory Card	P.126
AxisRecorderPut	Add Axis Record	P.136
AxisRecorderGet	Get Axis Record	P.140
AxisRecorderCSVWrite	Write Axis Record to SD Memory Card	P.142
BitRecorderPut	Add Bit Record	P.150
BitRecorderGet	Get Bit Record	P.154
BitRecorderToGraph	Display Bit Record	P.156

MonitorCylinder_Measure

The MonitorCylinder_Measure function block measures the operation time of the cylinder, and outputs the statistics of operation time, etc. of the 10 most recent times.

Function block name	Name	FB/FUN	Graphic expression	ST expression
MonitorCylinder_Measure	Monitor Cylinder Device Operation (Measure)	FB		<pre>MonitorCylinder_Measure_instance(Enable, Pull, Push, MeasureMode, FullyRetractedPos, FullyExtendedPos, Timeout, Enabled, Measuring, MeasuredStatus, Error, ErrorID, ErrorIDEx);</pre>

Function Block and Function Information

Item	Description
Library file name	OmronLib_BC_DeviceMonitor_V1_0.slr
Namespace	OmronLib\BC_DeviceMonitor
Function block and function number	00017
Publish/Do not publish source code	Do not publish
Function block and function version	1.00

Variables

	Meaning	I/O	Description	Valid range	Unit	Default
Enable	Enable	Input	TRUE: Enable FALSE: Disable	TRUE or FALSE	-	FALSE
Pull	Pull Command Flag	Input	TRUE: Pull command TRUE FALSE: Pull command FALSE	TRUE or FALSE		FALSE
Push	Push Command Flag	Input	TRUE: Push command TRUE FALSE: Push command FALSE	TRUE or FALSE		FALSE
MeasureMode	Measurement Mode	Input	TRUE: Single mode FALSE: Double mode	TRUE or FALSE		FALSE
FullyRetractedPos	Fully Retracted Position	Input	TRUE: Fully retracted position reached FALSE: Fully retracted position not reached	TRUE or FALSE		FALSE
FullyExtendedPos	Fully Extended Position	Input	TRUE: Fully extended position reached FALSE: Fully extended position not reached	TRUE or FALSE		FALSE
Timeout	Measurement Timeout	Input	Measurement timeout	Depends on data type		10 s
Enabled	Enabled	Output	TRUE: Enabled FALSE: Disabled	TRUE or FALSE		-
Measuring	Measuring	Output	TRUE: Measurement is in progress. FALSE: Measurement is not in progress.	TRUE or FALSE		-
MeasuredStatus	Measured Cylinder Status	Output	Status of measured cylinder	*1		-
Error	Error	Output	TRUE: Error end FALSE: Normal end, execution in progress, or execution condition not met	TRUE or FALSE	-	
ErrorID	Error Code	Output	This is an error code for an error end. The value is 16#0 for a normal end.	*2	-	-
ErrorIDEx	Expansion Error Code	Output	This is an expansion error code for an error end. The value is 16#0 for a normal end.	*2	-	-

*1. Refer to the structure sMeasuredStatus for details.

*2. Refer to *Troubleshooting* on page 51 for details.

	Boolean	Bit strings				Integers							Real numbers		Times, durations, dates, and text strings					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Enable	OK																			
Pull	OK																			
Push	OK																			
MeasureMode	OK																			
FullyRetractedPos	OK																			
FullyExtendedPos	OK																			
Timeout																OK				
Enabled	OK																			
Measuring	OK																			
MeasuredStatus	Refer to Function for details on the structure OmronLib\BC_DeviceMonitor\sMeasuredStatus.																			
Error	OK																			
ErrorID			OK																	
ErrorIDEx				OK																

Function

This function block measures the operation time of the cylinder, and outputs the statistics of operation time, etc. of the 10 most recent times. The statistics of the 10 most recent operation times determined with this function block are the references when setting error values for the MonitorSylinder_Double function block and the MonitorSylinder_Single function block.

However, if the execution count of this FB is less than ten, the statistics of the measured times are output. For example, if the execution count is five, the statistics of the five times are output.

Operation starts when *Enable* (Enable) is set to TRUE.

Connection with Cylinder

The following table shows the combination between the cylinder I/O used and the connected input variables for this FB.

The two types of the measurement mode are single mode and double mode, and the connected input variables vary for each mode. Set the single mode for a single-acting cylinder and the double mode for a double-acting cylinder.

Cylinder I/O	Connected input variables	
	Single mode	Double mode
Push command	Push (Push Command Flag)	Push (Push Command Flag)
Pull command	—	Pull (Pull Command Flag)
Fully extended position reed switch	FullyExtendedPos (Fully Extended Position)	FullyExtendedPos (Fully Extended Position)
Fully retracted position reed switch	FullyRetractedPos (Fully Retracted Position)	FullyRetractedPos (Fully Retracted Position)

Measuring Cylinder Operation Time

This function block measures the operation time on the push side and the pull side for the most recent 10 times of cylinder operation.

Set the measurement mode with *MeasureMode*. The *MeasureMode* value, measure start timing, and measure end timing for each measurement mode are as follows.

Measurement mode	Value of <i>MeasureMode</i>	Push side		Pull side	
		Measure start timing	Measure end timing	Measure start timing	Measure end timing
Single mode	TRUE	Push (Push Command Flag) change to TRUE	FullyExtendedPos (Fully Extended Position) change to TRUE	Push (Push Command Flag) change to FALSE	FullyRetractedPos (Fully Retracted Position) change to TRUE
Double mode	FALSE	Push (Push Command Flag) change to TRUE	FullyExtendedPos (Fully Extended Position) change to TRUE	Pull (Pull Command Flag) change to TRUE	FullyRetractedPos (Fully Retracted Position) change to TRUE

The measurement result is output to *MeasuredStatus* (Measured Cylinder Status).

During measuring of operation time, the value of *Measuring* is TRUE.

If the value of *Enable* is TRUE, even when the value of *MeasureMode* is changed, it is not reflected.

Measurement Timeout

The cylinder operation time exceeds the value of *Timeout* (Measurement Timeout), it is regarded as measurement timeout and an error occurs.

If the value of *Timeout* (Measurement Timeout) is zero, a measurement timeout error does not occur.

If the value of *Enable* is TRUE, even when the value of *Timeout* (Measurement Timeout) is changed, it is not reflected.

Measured Cylinder Status

You can find the status of the measured cylinder with *MeasuredStatus* (Measured Cylinder Status). The value of *MeasuredStatus* is cleared when *Enable* (Enabled) changes to TRUE.

The data type of *MeasuredStatus* is structure OmronLib\BC_DeviceMonitor\MeasuredStatus. The specifications are as follows:

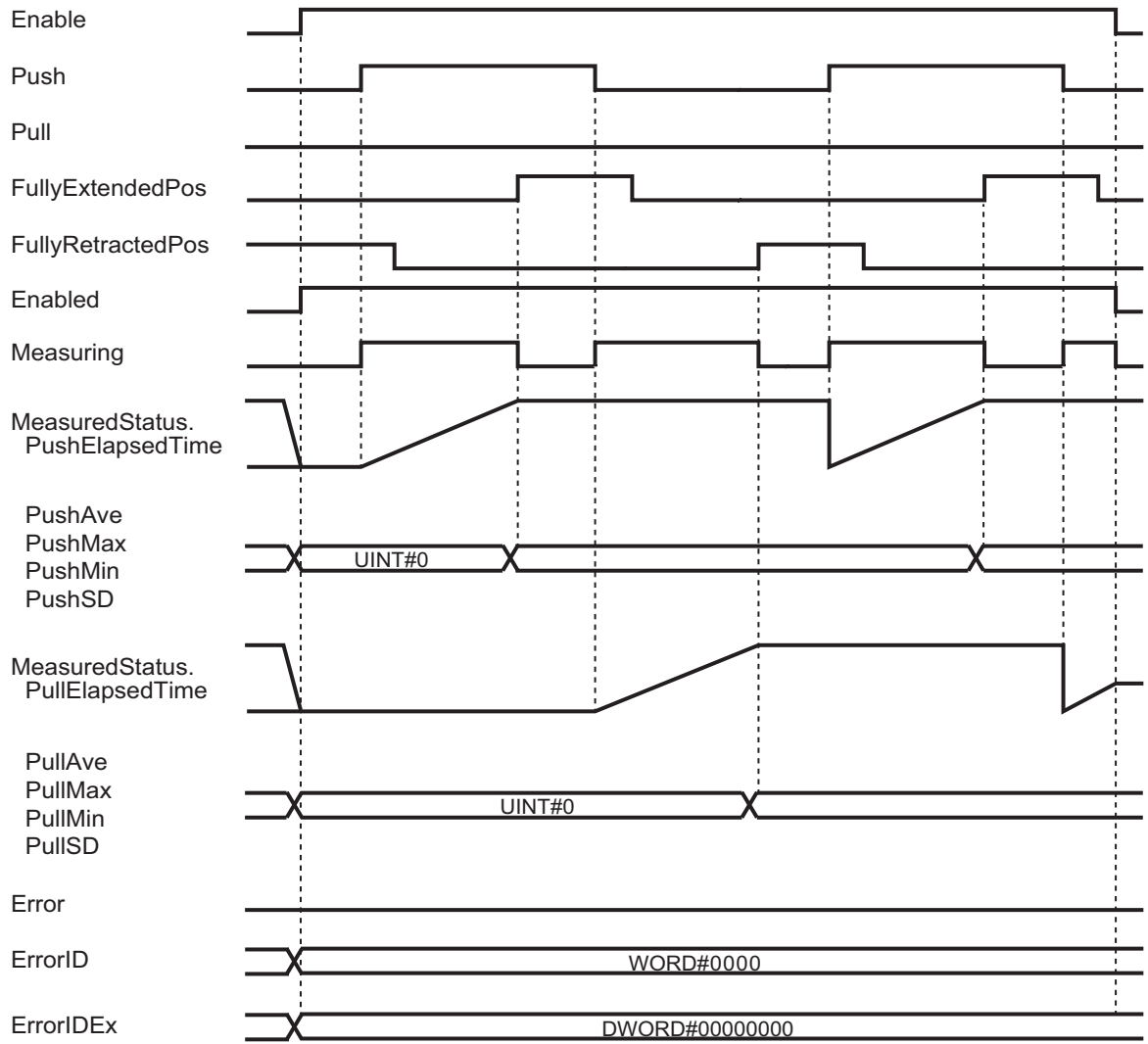
Name	Meaning	Description	Data type	Valid range	Unit	Default
MeasuredStatus	Measured Cylinder Status	Status of measured cylinder	Omron-Lib\BC_Device-Monitor\MeasuredStatus	-	-	-
PushAve	Push Average Operation Time	Average value of the operation time on the push side	UINT	0 to 65535	0.01 s	-
PushMax	Push Maximum Operation Time	Maximum operation time on the push side	UINT	0 to 65535	0.01 s	-
PushMin	Push Minimum Operation Time	Minimum operation time on the push side	UINT	0 to 65535	0.01 s	-
PushSD	Push Operation Standard Deviation Time	Standard deviation of operation time on the push side	UINT	0 to 65535	0.01 s	-
PushElapsedTime	Push Elapsed Operation Time	Elapsed operation time on the push side	UINT	0 to 65535	0.01 s	-
PullAve	Pull Average Operation Time	Average value of the operation time on the pull side	UINT	0 to 65535	0.01 s	-
PullMax	Pull Maximum Operation Time	Maximum operation time on the pull side	UINT	0 to 65535	0.01 s	-
PullMin	Pull Minimum Operation Time	Minimum operation time on the pull side	UINT	0 to 65535	0.01 s	-
PullSD	Pull Operation Standard Deviation Time	Standard deviation of operation time on the pull side	UINT	0 to 65535	0.01 s	-
PullElapsedTime	Pull Elapsed Operation Time	Elapsed operation time on the pull side	UINT	0 to 65535	0.01 s	-

Timing Charts

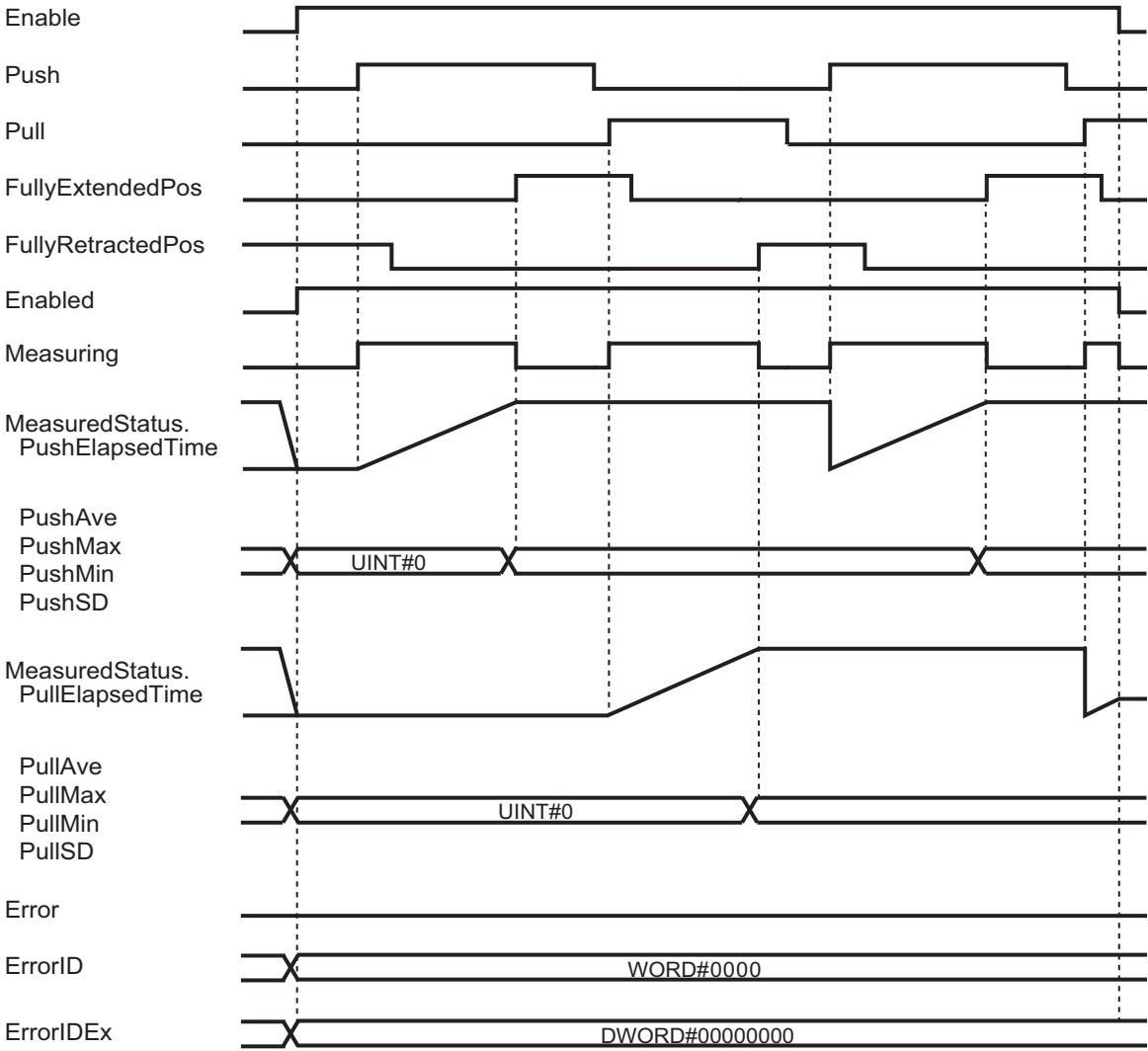
The following figures show the timing charts for the program part.

- *Enabled* (Enabled) changes to TRUE at the same time as *Enable* (Enable) changes to TRUE.
- *Measuring* (Measuring) changes to TRUE at the same time as *Push* (Push Command Flag) changes to TRUE, and push operation time measurement starts.
- The measured operation time is output to *MeasuredStatus.PushElapsedTime* (Push Elapsed Operation Time). The measurement ends when *FullyExtendedPos* (Fully Extended Position) changes to TRUE, and the values of *MeasuredStatus.PushAve* (Push Average Operation Time), *MeasuredStatus.PushMax* (Push Maximum Operation Time), *MeasuredStatus.PushMin* (Push Minimum Operation Time) and *MeasuredStatus.PushSD* (Push Operation Standard Deviation Time) are updated.
- The operation on the pull side varies depending on the measurement mode. In single mode, *Measuring* (Measuring) changes to TRUE at the same time as *Push* (Push Command Flag) changes to FALSE, and pull operation time measurement starts. In double mode, *Measuring* (Measuring) changes to TRUE at the same time as *Pull* (Pull Command Flag) changes to TRUE, and pull operation time measurement starts.
- The measurement ends at the same time as *FullyRetractedPos* (Fully Retracted Position) changes to TRUE in pull operation regardless of the measurement mode.
- If an error occurs when this function block is executed, *Enabled* (Enabled) and *Measuring* (Measuring) will change to FALSE, and *Error* (Error) will change 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).

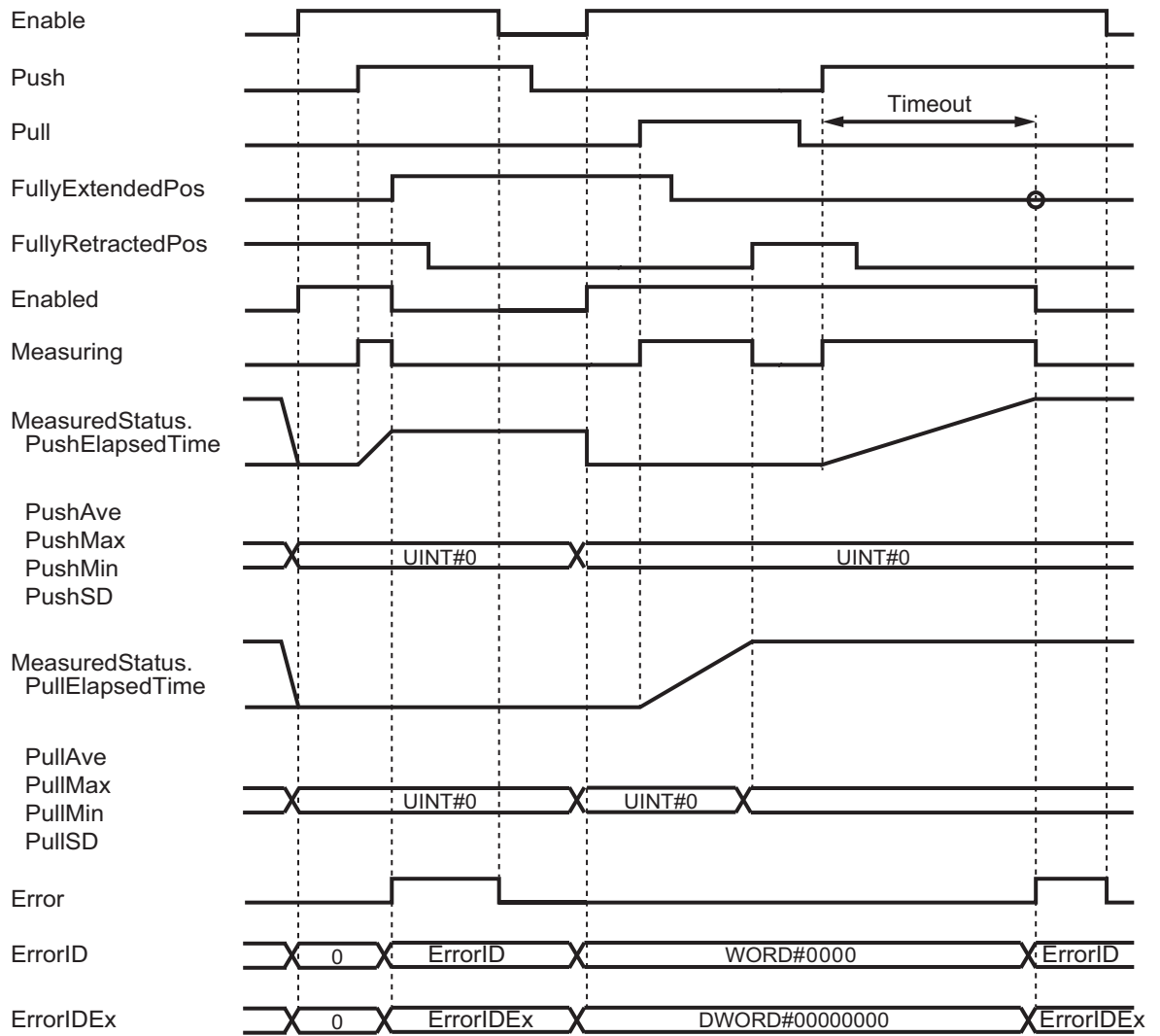
● **Timing Chart for Normal End (in Single mode)**



● Timing Chart for Normal End (in Double mode)



● Timing Chart for Error End



Troubleshooting

Error Code	Expansion error code	Status	Description	Correction
16#0000	16#00000000	Normal End	–	–
16#3C1C	16#00000001	User Software Error	The measurement mode is double mode, and the values of both <i>Pull</i> (Pull Command Flag) and <i>Push</i> (Push Command Flag) are TRUE.	Check the values of <i>Pull</i> (Pull Command Flag) and <i>Push</i> (Push Command Flag), and cylinder operation to confirm whether any error occurred.
16#3C1C	16#00000002	Timeout Error	A measurement time exceeds the value of <i>Timeout</i> (Measurement Timeout).	Check the set value of <i>Timeout</i> (Measurement Timeout) and cylinder operation to confirm whether any error occurred.
16#3C1C	16#00000003	User Software Error	When the values of <i>FullyRetractedPos</i> (Fully Retracted Position) and <i>FullyExtendedPos</i> (Fully Extended Position) are both TRUE.	Check the values of <i>FullyRetractedPos</i> (Fully Retracted Position) and <i>FullyExtendedPos</i> (Fully Extended Position), and cylinder operation to confirm whether any error occurred.

Sample Programming

Refer to the sample programming for *MonitorCylinder_Double* on page 52.



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 programming actual applications, also program safety circuits, device interlocks, I/O with other devices, and other control procedures.
- Create a user program that will produce the intended device operation.
- Check the user program for proper execution before you use it for actual operation.

MonitorCylinder_Double

The MonitorCylinder_Double function block measures the operation time of the cylinder, and outputs an alarm and error if it exceeds the upper or lower limit set by the operation time.

It uses push and pull command signals.

Function block name	Name	FB/FUN	Graphic expression	ST expression
MonitorCylinder_Double	Monitor Cylinder Device Operation (Double)	FB		<pre>MonitorCylinder_Double_instance(Enable, Pull, Push, FullyRetractedPos, FullyExtendedPos, AlarmSetting, Enabled, Monitoring, CylinderStatus, CylinderAlarm, Error, ErrorID, ErrorIDEx);</pre>

Function Block and Function Information

Item	Description
Library file name	OmronLib_BC_DeviceMonitor_V1_0.slr
Namespace	OmronLib\BC_DeviceMonitor
Function block and function number	00018
Publish/Do not publish source code	Do not publish
Function block and function version	1.00

Variables

	Meaning	I/O	Description	Valid range	Unit	Default
Enable	Enable	Input	TRUE: Enable FALSE: Disable	TRUE or FALSE	-	FALSE
Pull	Pull Command Flag	Input	TRUE: Pull command TRUE FALSE: Pull command FALSE	TRUE or FALSE		FALSE
Push	Push Command Flag	Input	TRUE: Push command TRUE FALSE: Push command FALSE	TRUE or FALSE		FALSE
FullyRetractedPos	Fully Retracted Position	Input	TRUE: Fully retracted position reached FALSE: Fully retracted position not reached	TRUE or FALSE		FALSE
FullyExtendedPos	Fully Extended Position	Input	TRUE: Fully extended position reached FALSE: Fully extended position not reached	TRUE or FALSE		FALSE
AlarmSetting	Error Value Setting	Input	Set value of alarm or error	*1		-
Enabled	Enabled	Output	TRUE: Enabled FALSE: Disabled	TRUE or FALSE		-
Monitoring	Monitoring	Output	TRUE: Monitor is in progress. FALSE: Monitor is not in progress.	TRUE or FALSE	-	
CylinderStatus	Cylinder Status	Output	Status of cylinder	*2	-	
CylinderAlarm	Alarm Output	Output	TRUE: Alarm occurred FALSE: No alarm occurred	TRUE or FALSE	-	
Error	Error	Output	TRUE: Error end FALSE: Normal end, execution in progress, or execution condition not met	TRUE or FALSE	-	
ErrorID	Error Code	Output	This is an error code for an error end. The value is 16#0 for a normal end.	*3	-	-
ErrorIDEx	Expansion Error Code	Output	This is an expansion error code for an error end. The value is 16#0 for a normal end.	*3	-	-

*1. Refer to the structure sCylinderAlarmSetting for details.

*2. Refer to the structure sCylinderStatus for details.

*3. Refer to *Troubleshooting* on page 60 for details.

	Boolean	Bit strings					Integers							Real numbers		Times, durations, dates, and text strings				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Enable	OK																			
Pull	OK																			
Push	OK																			
FullyRetractedPos	OK																			
FullyExtendedPos	OK																			
AlarmSetting	Refer to Function for details on the structure OmronLib\BC_DeviceMonitor'sCylinderAlarmSetting.																			
Enabled	OK																			
Monitoring	OK																			
CylinderStatus	Refer to Function for details on the structure OmronLib\BC_DeviceMonitor'sCylinderStatus.																			
CylinderAlarm	OK																			
Error	OK																			
ErrorID			OK																	
ErrorIDEx				OK																

Function

This function block measures the cylinder operation time and outputs an alarm and error if the operation time exceeds the set threshold for early or delayed operation.

It uses push and pull command signals.

Operation starts when *Enable* (Enable) is set to TRUE.

Connection with Cylinder

The following table shows the connections between the cylinder and the function block input variables.

Cylinder	Connected input variables
Push command	Push (Push Command Flag)
Pull command	Pull (Pull Command Flag)
Fully extended position reed switch	FullyExtendedPos (Fully Extended Position)
Fully retracted position reed switch	FullyRetractedPos (Fully Retracted Position)

Alarm or Error Output

Push operation time is measured when the value of *Push* (Push Command Flag) is changed to TRUE. Pull operation time is measured when the value of *Pull* (Pull Command Flag) is changed to TRUE.

While the operation time is measured, the value of *Monitoring* (Monitoring) changes to TRUE.

When the measurement value exceeds the alarm value set for *AlarmSetting* (Error Value Setting), the value of *CylinderAlarm* (Alarm Output) changes to TRUE. Also, when the error value is exceeded, the value of *Error* (Error) changes to TRUE.

When the values of *Push* (Push Command Flag) and *Pull* (Pull Command Flag) are both changed to TRUE, an error occurs and the value of *Error* changes to TRUE.

When the value of *CylinderAlarm* (Alarm Output) is TRUE, and when *Push* (Push Command Flag) or *Pull* (Pull Command Flag) are changed to TRUE, the value of *CylinderAlarm* (Alarm Output) changes to FALSE.

The data type of *AlarmSetting* is structure `OmronLib\BC_DeviceMonitor\CylinderAlarmSetting`. The specifications are as follows. Change the set value to zero for monitoring items that do not need to issue an alarm or error.

Name	Meaning	Description	Data type	Valid range	Unit	Default
AlarmSetting	Error Value Setting	Set value of alarm or error	Omron-Lib\BC_Device-Monitor\CylinderAlarmSetting	–	–	–
PushLL	Early Push Operation Error Value	Error value for early push operation	UINT	0 to 65535	0.01 s	–
PushL	Early Push Operation Alarm Value	Alarm value for early push operation	UINT	0 to 65535	0.01 s	–
PushH	Delayed Push Operation Alarm Value	Alarm value for delayed push operation	UINT	0 to 65535	0.01 s	–
PushHH	Delayed Push Operation Error Value	Error value for delayed push operation	UINT	0 to 65535	0.01 s	–
PullLL	Early Pull Operation Error Value	Error value for early pull operation	UINT	0 to 65535	0.01 s	–
PullL	Early Pull Operation Alarm Value	Alarm value for early pull operation	UINT	0 to 65535	0.01 s	–
PullH	Delayed Pull Operation Alarm Value	Alarm value for delayed pull operation	UINT	0 to 65535	0.01 s	–
PullHH	Delayed Pull Operation Error Value	Error value for delayed pull operation	UINT	0 to 65535	0.01 s	–

Cylinder Status

You can find out the status of the cylinder with *CylinderStatus* (Cylinder Status).

The data type of *CylinderStatus* is structure OmronLib\BC_DeviceMonitor\CylinderStatus. The specifications are as follows:

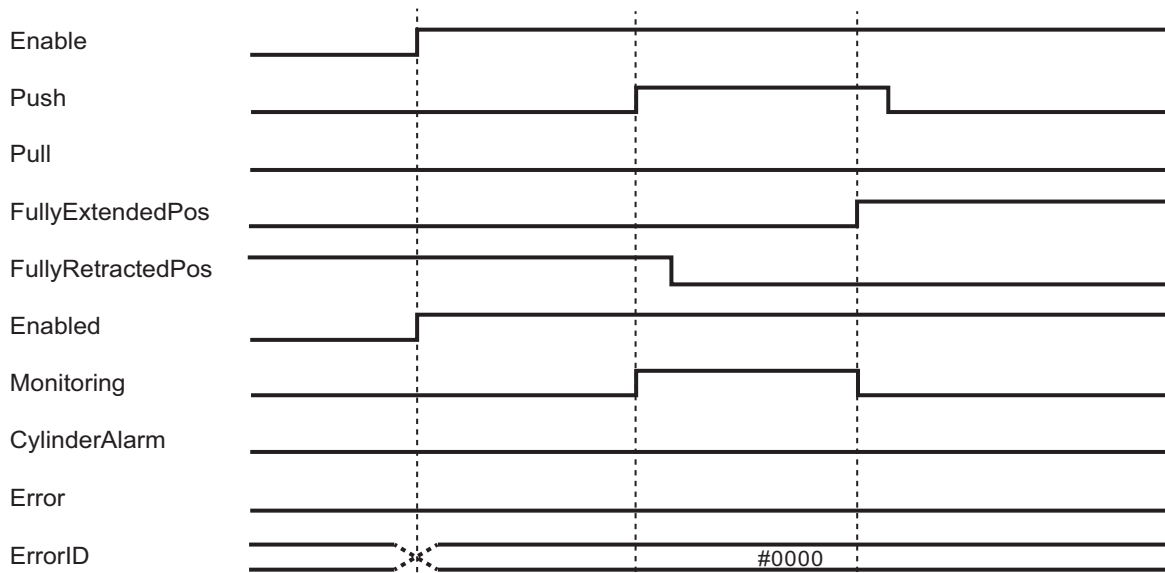
Name	Meaning	Description	Data type	Valid range	Unit	Default
CylinderStatus	Cylinder Status	Status of cylinder	Omron-Lib\BC_Device-Monitor\CylinderStatus	-	-	-
PushOnWay	During Push Operation	TRUE: Push operation is in progress. FALSE: Push operation is not in progress.	BOOL	Depends on data type.	-	-
PushFinished	Fully Extended Position Reached	TRUE: Fully extended position reached FALSE: Fully extended position not reached	BOOL	Depends on data type.	-	-
PushAlarm	Push Alarm	TRUE: Push alarm occurred FALSE: No push alarm occurred	BOOL	Depends on data type.	-	-
PushError	Push Error	TRUE: Push error occurred FALSE: No push error occurred	BOOL	Depends on data type.	-	-
PullOnWay	During Pull Operation	TRUE: Pull operation is in progress. FALSE: Pull operation is not in progress.	BOOL	Depends on data type.	-	-
PullFinished	Fully Retracted Position Reached	TRUE: Fully retracted position reached FALSE: Fully retracted position not reached	BOOL	Depends on data type.	-	-
PullAlarm	Pull Alarm	TRUE: Pull alarm occurred FALSE: No pull alarm occurred	BOOL	Depends on data type.	-	-
PullError	Pull Error	TRUE: Pull error occurred FALSE: No pull error occurred	BOOL	Depends on data type.	-	-

Timing Charts

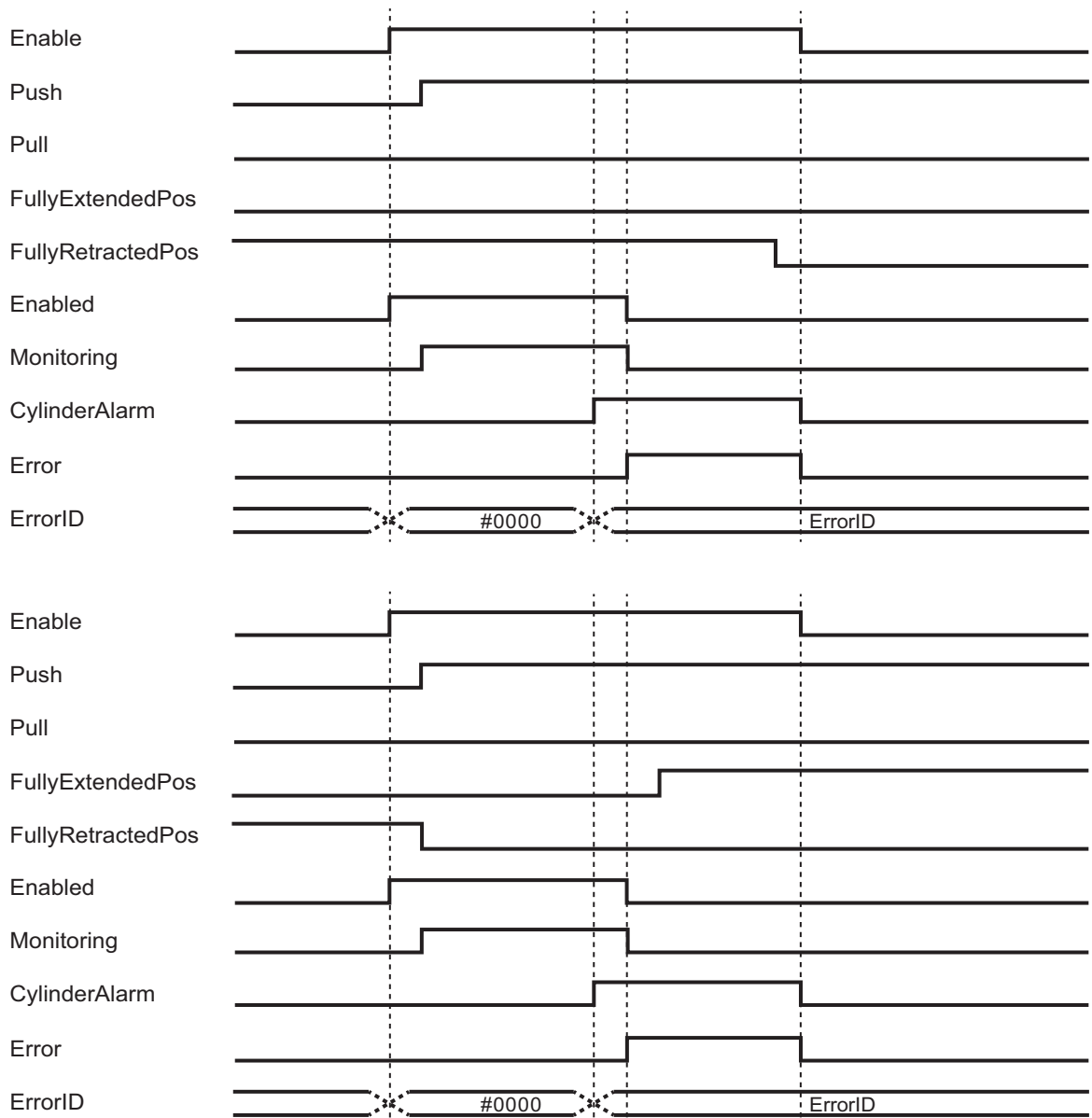
The following figures show the timing charts for the program part.

- *Enabled* (Enabled) changes to TRUE at the same time as *Enable* (Enable) changes to TRUE.
- Push operation time measurement starts when *Push* (Push Command Flag) changes to TRUE. Measurement ends when *FullyExtendedPos* (Fully Extended Position) is changed to TRUE.
- Pull operation time measurement starts when *Pull* (Pull Command Flag) changes to TRUE. Measurement ends when *FullyRetractedPos* (Fully Retracted Position) is changed to TRUE.
- During measurement, *Monitoring* (Monitoring) changes to TRUE.
- If an error occurs during execution of the function block, *Enabled* (Enabled) and *Monitoring* (Monitoring) change to FALSE and *Error* (Error) changes to TRUE. You can find out the cause of the error by referring to the values output to *ErrorID* (Error Code) and *ErrorIDEx* (Expansion Error Code).
- If an alarm occurs during execution of the function block, *CylinderAlarm* (Alarm Output) changes to TRUE.

● Timing Chart for Normal End



● **Timing Chart for Error End**



Precautions for Correct Use



Precautions for Correct Use

Always set *AlarmSetting* (Error Value Setting).

Troubleshooting

Error code	Expansion error code	Status	Description	Correction
16#0000	16#00000000	Normal End	–	–
16#3C1A	16#00000001	Push Delay Error	Elapsed time of push operation exceeds the delayed operation error value	Check the value of <i>PushHH</i> (Delayed Push Operation Error Value), and check that there are no errors in the cylinder operation.
16#3C1A	16#00000002	Push Early Error	Elapsed time of push operation is less than the early operation error value	Check the value of <i>PushLL</i> (Early Push Operation Error Value), and check that there are no errors in the cylinder operation.
16#3C1A	16#00000003	Pull Delay Error	Elapsed time of pull operation exceeds the delayed operation error value	Check the value of <i>PullHH</i> (Delayed Pull Operation Error Value), and check that there are no errors in the cylinder operation.
16#3C1A	16#00000004	Pull Early Error	Elapsed time of pull operation is less than the early operation error value	Check the value of <i>PullLL</i> (Early Pull Operation Error Value), and check that there are no errors in the cylinder operation.
16#3C1A	16#00000005	Push Delay Alarm	Elapsed time of push operation exceeds the delayed operation alarm value	Check the value of <i>PushH</i> (Delayed Push Operation Alarm Value), and check that there are no errors in the cylinder operation.
16#3C1A	16#00000006	Push Early Alarm	Elapsed time of push operation is less than the early operation alarm value	Check the value of <i>PushL</i> (Early Push Operation Alarm Value), and check that there are no errors in the cylinder operation.
16#3C1A	16#00000007	Pull Delay Alarm	Elapsed time of pull operation exceeds the delayed operation alarm value	Check the value of <i>PullH</i> (Delayed Pull Operation Alarm Value), and check that there are no errors in the cylinder operation.
16#3C1A	16#00000008	Pull Early Alarm	Elapsed time of pull operation is less than the early operation alarm value	Check the value of <i>PullL</i> (Early Pull Operation Alarm Value), and check that there are no errors in the cylinder operation.
16#3C1A	16#00000009	User Software Error	The both values of <i>Pull</i> (Pull Command Flag) and <i>Push</i> (Push Command Flag) are TRUE.	Check the values of <i>Pull</i> (Pull Command Flag) and <i>Push</i> (Push Command Flag), and cylinder operation to confirm whether any error occurred.
16#3C1A	16#0000000A	User Software Error	The values of <i>FullyRetractedPos</i> (Fully Retracted Position) and <i>FullyExtendedPos</i> (Fully Extended Position) are both TRUE.	Check the values of <i>FullyRetractedPos</i> (Fully Retracted Position) and <i>FullyExtendedPos</i> (Fully Extended Position), and cylinder operation to confirm whether any error occurred.

Sample Programming

Description of Operation

When *Measure_Enable* is changed to TRUE to operate the cylinder, alarm and error values are set with an inline ST based on the average operation time, maximum operation time, minimum operation time and standard deviation time for both push and pull operations.

The cylinder is operated after measurement completes, and if an error occurs, such as when operation does not end within the set amount of time, the value of *CylinderAlarm* or *Error* changes to TRUE.

Check *CylinderStatus* for details on the Cylinder Status.

Variables

● Internal Variables

Name	Data type	Default	Comment
MonitorCylinder_Double_instance	OmronLib\BC_DeviceMonitor\ MonitorCylinder_Double	–	Instance of Monitor Cylinder Device Operation (Double) FB
MonitorCylinder_Measure_instance	OmronLib\BC_DeviceMonitor\ MonitorCylinder_Measure	–	Instance of Monitor Cylinder Device Operation (Measure) FB
Measure_Enable	BOOL	FALSE	Measurement enable flag
MonCylinder_MeaMode	BOOL	FALSE	Measurement mode
MonCylinder_Timeout	TIME	10 s	Measurement timeout
MonCylinder_MeaStatus	OmronLib\BC_DeviceMonitor\ sMeasuredStatus	–	Status of measured cylinder
MonCylinder_Measuring	BOOL	–	Measurement in progress
MonCylinder_Status	OmronLib\BC_DeviceMonitor\ sCylinderStatus	–	Cylinder status
MonCylinder_AlarmSetting	OmronLib\BC_DeviceMonitor\ sCylinderErrorValue	–	Alarm or error value setting
MonCylinder_Monitoring	BOOL	–	Monitor flag
MonCylinder_Alarm	BOOL	–	Monitor alarm flag
MonCylinder_Err	BOOL	–	Monitor error flag
MonCylinder_ErrID	WORD	–	Monitor error code
MonCylinder_ErrIDEx	DWORD	–	Monitor expansion error code
MonCylinder_Measure_Err	BOOL	–	Measurement error flag
MonCylinder_Measure_ErrID	WORD	–	Measurement error code
MonCylinder_Measure_ErrIDEx	DWORD	–	Measurement expansion error code

● External Variables

Name	Data type	Constant	Comment
MonCylinder_Pull	BOOL	–	Pull command (Assigned to the Digital Output Units of the NX Units)
MonCylinder_Push	BOOL	–	Push command (Assigned to the Digital Output Units of the NX Units)
MonCylinder_FRePos	BOOL	–	Fully retracted position reached flag (Assigned to the Digital Input Units of the NX Units)
MonCylinder_FExPos	BOOL	–	Fully extended position reached flag (Assigned to the Digital Input Units of the NX Units)

MonitorCylinder_Single

The MonitorCylinder_Single function block measures the operation time of the cylinder, and outputs an alarm and error if it exceeds the upper or lower limit set by the operation time.

It only uses the push command.

Function block name	Name	FB/FUN	Graphic expression	ST expression
MonitorCylinder_Single	Monitor Cylinder Device Operation (Single)	FB	<p>MonitorCylinder_Single_instance</p>	<pre>MonitorCylinder_Single_instance(Enable, Push, FullyRetractedPos, FullyExtendedPos, AlarmSetting, Enabled, Monitoring, CylinderStatus, CylinderAlarm, Error ErrorID, ErrorIDEx);</pre>

Function Block and Function Information

Item	Description
Library file name	OmronLib_BC_DeviceMonitor_V1_0.slr
Namespace	OmronLib\BC_DeviceMonitor
Function block and function number	00019
Publish/Do not publish source code	Do not publish
Function block and function version	1.00

Variables

	Meaning	I/O	Description	Valid range	Unit	Default
Enable	Enable	Input	TRUE: Enable FALSE: Disable	TRUE or FALSE	-	FALSE
Push	Push Com- mand Flag	Input	TRUE: Push command TRUE FALSE: Push command FALSE	TRUE or FALSE		FALSE
FullyRetract- edPos	Fully Retracted Position	Input	TRUE: Fully retracted position reached FALSE: Fully retracted position not reached	TRUE or FALSE		FALSE
FullyExtended- Pos	Fully Extended Position	Input	TRUE: Fully extended position reached FALSE: Fully extended position not reached	TRUE or FALSE		FALSE
AlarmSetting	Error Value Setting	Input	Set value of alarm or error	*1		-
Enabled	Enabled	Output	TRUE: Enabled FALSE: Disabled	TRUE or FALSE		-
Monitoring	Monitoring	Output	TRUE: Monitor is in progress. FALSE: Monitor is not in progress.	TRUE or FALSE		-
CylinderStatus	Cylinder Status	Output	Status of cylinder	*2		-
CylinderAlarm	Alarm Output	Output	TRUE: Alarm occurred FALSE: No alarm occurred	TRUE or FALSE		-
Error	Error	Output	TRUE: Error end FALSE: Normal end, execution in progress, or execution condition not met	TRUE or FALSE		-
ErrorID	Error Code	Output	This is an error code for an error end. The value is 16#0 for a normal end.	*3	-	-
ErrorIDEx	Expansion Error Code	Output	This is an expansion error code for an error end. The value is 16#0 for a normal end.	*3	-	-

*1. Refer to the structure sCylinderAlarmSetting for details.

*2. Refer to the structure sCylinderStatus for details.

*3. Refer to *Troubleshooting* on page 69 for details.

	Boolean	Bit strings				Integers								Real numbers		Times, durations, dates, and text strings				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Enable	OK																			
Push	OK																			
FullyRetractedPos	OK																			
FullyExtendedPos	OK																			
AlarmSetting	Refer to Function for details on the structure OmronLib\BC_DeviceMonitor\sCylinderAlarmSetting.																			
Enabled	OK																			
Monitoring	OK																			
CylinderStatus	Refer to Function for details on the structure OmronLib\BC_DeviceMonitor\sCylinderStatus																			
CylinderAlarm	OK																			
Error	OK																			
ErrorID			OK																	
ErrorIDEx				OK																

Function

This function block measures the cylinder operation time and outputs an alarm and error if the operation time exceeds the set threshold for early or delayed operation.

It only uses the push command, and not the pull command.

Operation starts when *Enable* (Enable) is set to TRUE.

Connection with Cylinder

The following table shows the connections between the cylinder and the function block input variables.

Cylinder	Connected input variables
Push command	Push (Push Command Flag)
Fully extended position reed switch ^{*1}	FullyExtendedPos (Fully Extended Position)
Fully retracted position reed switch ^{*2}	FullyRetractedPos (Fully Retracted Position)

*1. Connection not required if only the operation time to the fully retracted position is measured.

*2. Connection not required if only the operation time to the fully extended position is measured.

Alarm or Error Output

Push operation time is measured when the value of *Push* (Push Command Flag) is changed to TRUE.

While the operation time is measured, the value of *Monitoring* (Monitoring) changes to TRUE.

When the measured time exceeds the alarm value set for *AlarmSetting* (Error Value Setting), the value of *CylinderAlarm* (Alarm Output) changes to TRUE. Also, when the error value is exceeded, the value of *Error* (Error) changes to TRUE.

If a pull side alarm value or error value is set for *AlarmSetting*, the operation time from when *Push* changes to FALSE to the fully retracted position is also measured.

When the value of *CylinderAlarm* (Alarm Output) is TRUE, and when *Push* (Push Command Flag) is changed to TRUE, the value of *CylinderAlarm* (Alarm Output) changes to FALSE.

The data type of *AlarmSetting* is structure `OmronLib\BC_DeviceMonitor\CylinderAlarmSetting`. Change the set value to zero for monitoring items that do not need to issue an alarm or error. Refer to *MonitorCylinder_Double* on page 52 for details on `OmronLib\BC_DeviceMonitor\CylinderAlarmSetting`.

Cylinder Status

You can find out the status of the cylinder with *CylinderStatus* (Cylinder Status).

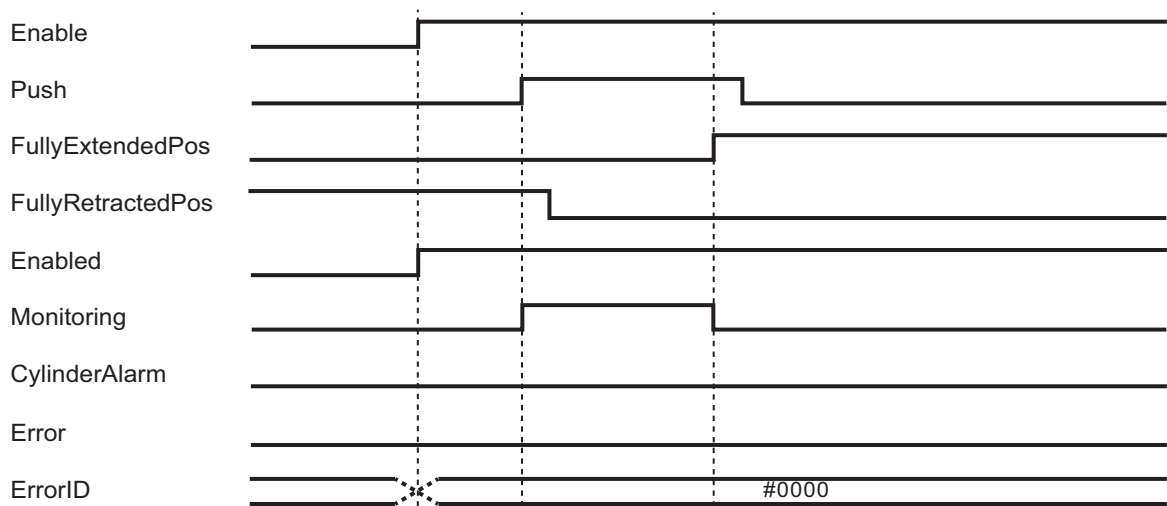
The data type of *CylinderStatus* is structure `OmronLib\BC_DeviceMonitor\CylinderStatus`. Refer to *MonitorCylinder_Double* on page 52 for details.

Timing Charts

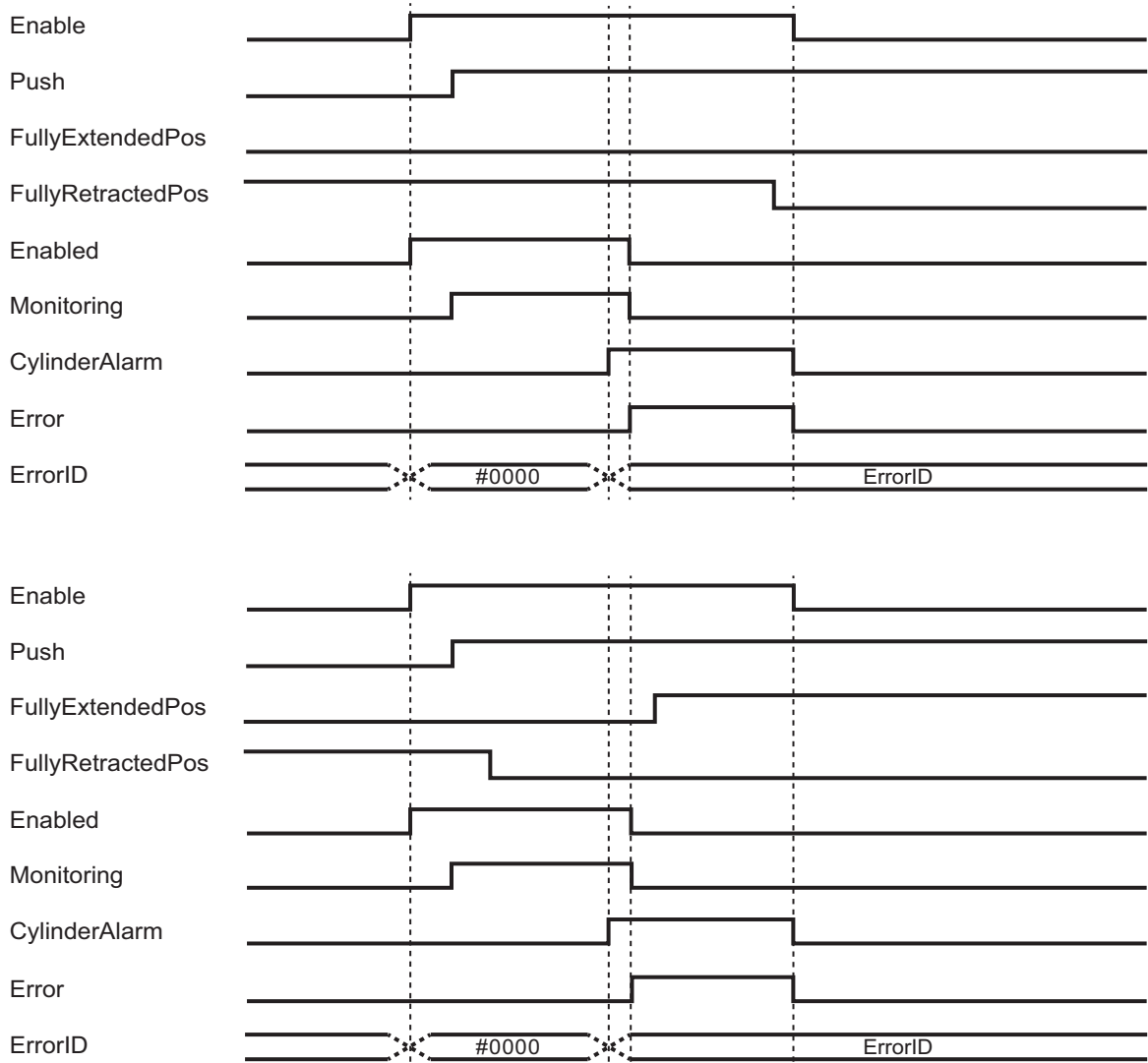
The following figures show the timing charts for the program part.

- *Enabled* (Enabled) changes to TRUE at the same time as *Enable* (Enable) changes to TRUE.
- Push operation time measurement starts when *Push* (Push Command Flag) changes to TRUE. Measurement ends when *FullyExtendedPos* (Fully Extended Position) is changed to TRUE.
- If a pull side alarm value or error value is set for *AlarmSetting*, measurement of the operation time on the pull side starts at the same time when *Push* changes to FALSE. Measurement ends when *FullyRetractedPos* (Fully Retracted Position) is changed to TRUE.
- During measurement, *Monitoring* (Monitoring) changes to TRUE.
- If an error occurs during execution of the function block, *Enabled* (Enabled) and *Monitoring* (Monitoring) change to FALSE and *Error* (Error) changes to TRUE. You can find out the cause of the error by referring to the values output to *ErrorID* (Error Code) and *ErrorIDEx* (Expansion Error Code).
- If an alarm occurs during execution of the function block, *CylinderAlarm* (Alarm Output) changes to TRUE.
- If a pull side alarm value or error value is set for *AlarmSetting*, the operation time from when *Push* changes to FALSE to the fully retracted position is also measured.

● Timing Chart for Normal End



● **Timing Chart for Error End**



Precautions for Correct Use



Precautions for Correct Use

Always set *AlarmSetting* (Error Value Setting).

Troubleshooting

Error Code	Expansion Error Code	Status	Description	Correction
16#0000	16#00000000	Normal End	–	–
16#3C1B	16#00000001	Push Delay Error	Elapsed time of push operation exceeds the delayed operation error value	Check the value of <i>PushHH</i> (Delayed Push Operation Error Value), and check that there are no errors in the cylinder operation.
16#3C1B	16#00000002	Push Early Error	Elapsed time of push operation is less than the early operation error value	Check the value of <i>PushLL</i> (Early Push Operation Error Value), and check that there are no errors in the cylinder operation.
16#3C1B	16#00000003	Pull Delay Error	Elapsed time of pull operation exceeds the delayed operation error value	Check the value of <i>PullHH</i> (Delayed Pull Operation Error Value), and check that there are no errors in the cylinder operation.
16#3C1B	16#00000004	Pull Early Error	Elapsed time of pull operation is less than the early operation error value	Check the value of <i>PullLL</i> (Early Pull Operation Error Value), and check that there are no errors in the cylinder operation.
16#3C1B	16#00000005	Push Delay Alarm	Elapsed time of push operation exceeds the delayed operation alarm value	Check the value of <i>PushH</i> (Delayed Push Operation Alarm Value), and check that there are no errors in the cylinder operation.
16#3C1B	16#00000006	Push Early Alarm	Elapsed time of push operation is less than the early operation alarm value	Check the value of <i>PushL</i> (Early Push Operation Alarm Value), and check that there are no errors in the cylinder operation.
16#3C1B	16#00000007	Pull Delay Alarm	Elapsed time of pull operation exceeds the delayed operation alarm value	Check the value of <i>PullH</i> (Delayed Pull Operation Alarm Value), and check that there are no errors in the cylinder operation.
16#3C1B	16#00000008	Pull Early Alarm	Elapsed time of pull operation is less than the early operation alarm value	Check the value of <i>PullL</i> (Early Pull Operation Alarm Value), and check that there are no errors in the cylinder operation.
16#3C1B	16#0000000A	User Software Error	The values of <i>FullyRetractedPos</i> (Fully Retracted Position) and <i>FullyExtendedPos</i> (Fully Extended Position) are both TRUE.	Check the values of <i>FullyRetractedPos</i> (Fully Retracted Position) and <i>FullyExtendedPos</i> (Fully Extended Position), and cylinder operation to confirm whether any error occurred.

Sample Programming

Refer to the sample programming for *MonitorCylinder_Double* on page 52.



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 programming actual applications, also program safety circuits, device interlocks, I/O with other devices, and other control procedures.
 - Create a user program that will produce the intended device operation.
 - Check the user program for proper execution before you use it for actual operation.
-

LogCompare

The LogCompare function block logs measurement values and compares them with the master values log data.

Function block name	Name	FB/FUN	Graphic expression	ST expression
LogCompare	Logging Compare	FB		<pre>LogCompare_instance(Enable, Teach, Scan, XType, X, Y, Tolerance, TeachedLog, ScanLog, Enabled, Logging, Alarm, AlarmPos, Error, ErrorID, ErrorIDEx);</pre>

Function Block and Function Information

Item	Description
Library file name	OmronLib_BC_DeviceMonitor_V1_0.slr
Namespace	OmronLib\BC_DeviceMonitor
Function block and function number	00020
Publish/Do not publish source code	Publish
Function block and function version	1.00

Variables

	Meaning	I/O	Description	Valid range	Unit	Default
Enable	Enable	Input	TRUE: Enable FALSE: Disable	TRUE or FALSE		FALSE
Teach	Execute Master Values Logging	Input	TRUE: Execute master values logging FALSE: Do not execute master values logging	TRUE or FALSE		FALSE
Scan	Execute Measurement Values Logging	Input	TRUE: Execute measurement values logging FALSE: Do not execute measurement values logging	TRUE or FALSE		FALSE
XType	X Input Type	Input	TRUE: X input value FALSE: Time	TRUE or FALSE		FALSE
X	X Input Value	Input	X input value	Depends on data type.		0
Y	Y Input Value	Input	Y input value	Depends on data type.		0
Tolerance	Tolerance	Input	Tolerance of master values	Depends on data type.		0
TeachedLog	Master Values	Input/output	Master values	Depends on data type.	–	–
ScanLog	Measurement Values	Input/output	Measurement values	Depends on data type.		–
Enabled	Enabled	Output	TRUE: Enabled FALSE: Disabled	TRUE or FALSE		–
Logging	Logging	Output	TRUE: Logging is in progress. FALSE: Logging is not in progress.	TRUE or FALSE		–
Alarm	Alarm	Output	TRUE: Alarm occurred FALSE: No alarm occurred	TRUE or FALSE		–
AlarmPos	Alarm Position	Output	Position of alarm occurrence	0 to 1999		–
Error	Error	Output	TRUE: Error end FALSE: Normal end, execution in progress, or execution condition not met	TRUE or FALSE		–
ErrorID	Error Code	Output	This is an error code for an error end. The value is 16#0 for a normal end.	*1		–
ErrorIDEx	Expansion Error Code	Output	This is an expansion error code for an error end. The value is 16#0 for a normal end.	*1		–

*1. Refer to *Troubleshooting* on page 77 for details.

	Boolean	Bit strings				Integers							Real numbers		Times, durations, dates, and text strings					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Enable	OK																			
Teach	OK																			
Scan	OK																			
XType	OK																			
X															OK					
Y															OK					
Tolerance															OK					
TeachedLog	Refer to Function for details on the structure OmronLib\BC_DeviceMonitor\sLogData.																			
ScanLog	Refer to Function for details on the structure OmronLib\BC_DeviceMonitor\sLogData																			
Enabled	OK																			
Logging	OK																			
Alarm	OK																			
AlarmPos							OK													
Error	OK																			
ErrorID			OK																	
ErrorIDEx				OK																

Function

This function block logs measurement values and compares them with the master values log data.

The measurement values log data is stored in *ScanLog* (Measurement Values). The master values log data is stored in *TeachedLog* (Master Values).

If the difference between *ScanLog* (Measurement Values) and *TeachedLog* (Master Values) exceeds the tolerance specified with *Tolerance* (Tolerance), *Alarm* (Alarm) and *AlarmPos* (Alarm Position) are output.

Operation starts when the value of *Enable* (Enable) is changed to TRUE. Also, *XType* (X Input Type), *Tolerance* (Tolerance) and *TeachedLog* (Master Values) are input.

Logged Data

The following information is logged for measurement values and master values depending on the value of *XType* (X Input Type).

Value of <i>XType</i> (X Input Type)	Logged Data
TRUE	X (X Input Value), Y (Y Input Value)
FALSE	Time elapsed since logging started, Y (Y Input Value)

The data type of *ScanLog* (Measurement Values) and *TeachedLog* (Master Values) is structure Omron-Lib\BC_DeviceMonitor\sLogData. The specifications are as follows.

Name	Meaning	Description	Data type	Valid range	Unit	Default
LogData	Log Data Recorder	Log Data Recorder	Omron-Lib\BC_Device-Monitor\sLogData			
Count	Number of Log Data	Number of recorded log data	UINT	0 to 2000	–	0
X	X Input Value	X input value or time elapsed since logging started*1	ARRAY[0..1999] OF LREAL	Depends on data type.	–	0
Y	Y Input Value	Y input value	ARRAY[0..1999] OF LREAL	Depends on data type.	–	0

*1. When the time elapsed since logging started is recorded, X (X Input Value) is expressed as an LREAL value in 1 μ s units.

The maximum number of log data is 2000 for both *ScanLog* (Measurement Values) and *TeachedLog* (Master Values).

Master Values Logging

Master values logging is performed as follows.

- The *XType* (X Input Type) is set.
- Master values logging starts when the value of *Teach* (Execute Master Values Logging) is changed to TRUE.
- While master values logging is executed, the value of *Logging* (Logging) changes to TRUE.
- Master values logging is ended in the following situations.
 - a) The value of *Teach* (Execute Master Values Logging) changes to FALSE.
 - b) The number of log data reaches 2000.

Measurement Values Logging

Measurement values logging is performed as follows.

- The *XType* (X Input Type) is set.
- Measurement values logging starts when the value of *Scan* (Execute Measurement Values Logging) is changed to TRUE.
- While measurement values logging is executed, the value of *Logging* (Logging) changes to TRUE.
- Measurement values logging is ended in the following situations.
 - a) The value of *Scan* (Execute Measurement Values Logging) changes to FALSE.
 - b) The number of log data reaches 2000.

Comparison of Measurement Values Log Data with Master Values Log Data

While measurement values logging is executed, the measurement values log data and master values log data are compared. The values with the same element number of arrays *TeachedLog.Y* and *ScanLog.Y* are compared.

Only *Y* (Y Input Value) is compared. *X* (X Input Value) is not compared.

If the comparison result indicates that the difference between the values exceeds the allowable range, *Alarm* (Alarm) and *AlarmPos* (Alarm Position) are output. The values of *Alarm* (Alarm) and *AlarmPos* (Alarm Position) are determined as follows depending on the measurement values, master values and the value of *Tolerance* (Tolerance).

Relationship between measurement values, master values and <i>Tolerance</i> (<i>Tolerance</i>) ^{*1}	Value of <i>Alarm</i> (Alarm)	Value of <i>AlarmPos</i> (Alarm Position)
$ScanLog.Y[n] < TeachedLog.Y[n] - Tolerance$	TRUE	n
$TeachedLog.Y[n] - Tolerance \leq ScanLog.Y[n] \leq TeachedLog.Y[n] + Tolerance$	FALSE	0
$TeachedLog.Y[n] + Tolerance < ScanLog.Y[n]$	TRUE	n

*1. "n" is the array element number from 0 to 1999.

Comparison of measurement values log data and master values log data ends in the following situations.

- The value of *Alarm* (Alarm) changes to TRUE.
- The number of measurement values log data exceeds the number of master values log data.
- The value of *Scan* (Execute Measurement Values Logging) changes to FALSE and measurement values logging ends.

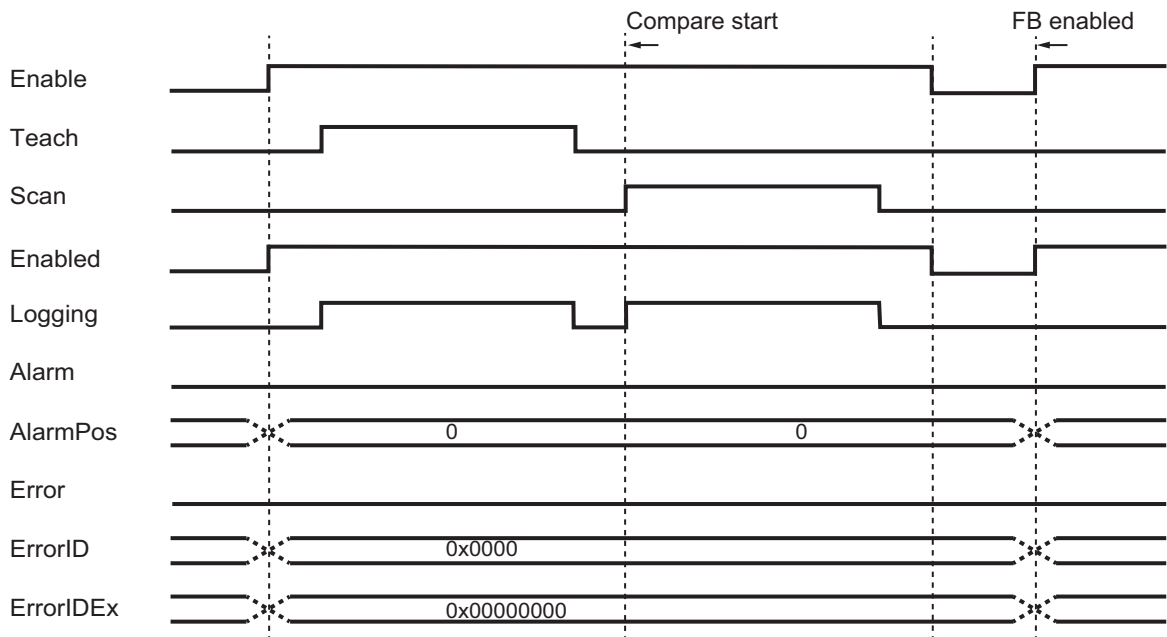
Timing Charts

The following figures show the timing charts for the program part.

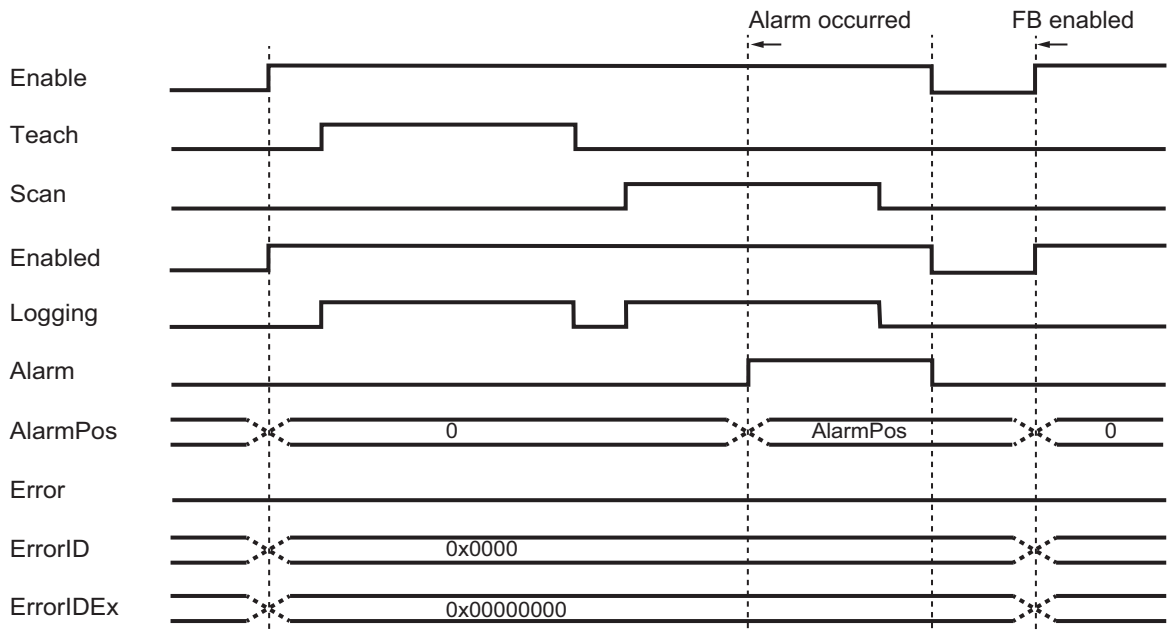
- When *Enable* (Enable) changes to TRUE, *Enabled* (Enabled) changes to TRUE.
- When *Teach* (Execute Master Values Logging) or *Scan* (Execute Measurement Values Logging) is changed to TRUE, *Logging* (Logging) changes to TRUE.
- When master values logging and measurement values logging is ended, *Logging* (Logging) changes to FALSE.
- When an alarm occurs, *Alarm* (Alarm) changes to TRUE. Also, *AlarmPos* (Alarm Position) is output.

● Timing Chart for Normal End

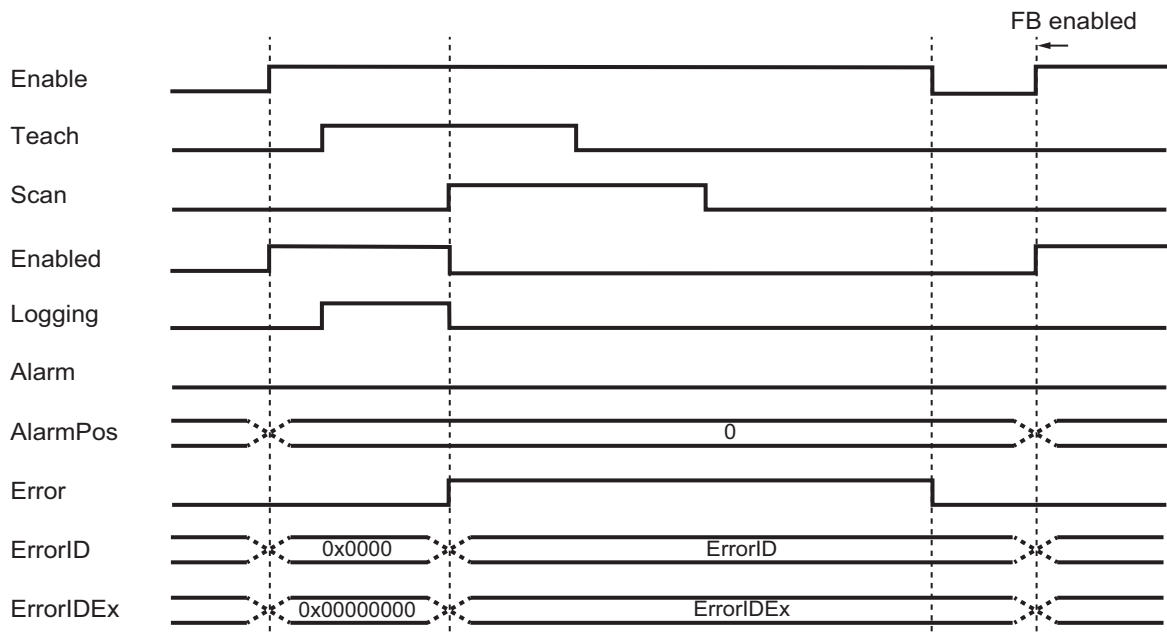
When no alarm occurs



When alarm occurs



● Timing Chart for Error End



Precautions for Correct Use

- An error occurs when the values of *Teach* (Execute Master Values Logging) and *Scan* (Execute Measurement Values Logging) are both changed to TRUE.
- When performing comparison again after comparison of the master values log data and measurement values log data ended, change the value of *Enable* (Enable) to FALSE once, and then change the value of *Enable* (Enable) to TRUE.
- The value of *TeachedLog* (Master Values) is retained even if *Enable* (Enable) changes to FALSE.
- The value of *ScanLog* (Measurement Values) is cleared when *Enable* (Enable) changes to TRUE.
- The values of *Alarm* (Alarm) is cleared when *Enable* (Enable) changes to FALSE.
- The values of *AlarmPos* (AlarmPosition) is cleared when *Enable* (Enable) changes to TRUE.

Troubleshooting

Error code	Expansion error code	Status	Description	Correction
16#0000	16#00000000	Normal End	–	–
16#3C1E	16#00000001	Execute Master Values Logging and Measurement Values Logging	<i>Teach</i> (Execute Master Values Logging) and <i>Scan</i> (Execute Measurement Values Logging) are both changed to TRUE.	Do not execute both <i>Teach</i> (Execute Master Values Logging) and <i>Scan</i> (Execute Measurement Values Logging).
16#0400	16#00000000	Input Value Out of Range	<i>TeachedLog.Count</i> (Number of log data) is set to a value outside the range of 0 to 2000.	Set <i>TeachedLog.Count</i> (Number of log data) to a value from 0 to 2000.
	16#00000000		<i>ScanLog.Count</i> (Number of log data) is set to a value outside the range of 0 to 2000.	Set <i>ScanLog.Count</i> (Number of log data) to a value from 0 to 2000.

Sample Programming

Description of Operation

This sample programming performs the following processes.

- The current position and torque value of *MC_Axis000* in the G5-series Servomotor are logged as measurement values and compared with the master values log data. Master values log data can be acquired by using LogCompare function block or read from an SD memory card.
- The sample programming can write the acquired master values log data to an SD memory card in CSV format.
- The LogCompare function block converts and displays master values log data and measurement values log data to the data format that is suitable for displaying as a broken-line graph on NS-series PT.

● How to Acquire Master Values Log Data Using LogCompare Function Block

- 1** Change the value of *Enable* to TRUE.
LogCompare_instance is executed and the value of *Enabled* changes to TRUE.
- 2** Change the value of *Start_Teach* to TRUE.
The master values log data is acquired. When the data is acquired, the value of *Ready_MasterData* changes to TRUE.
- 3** Change the value of *Start_Scan* to TRUE.
Measurement values are logged and compared with the master values log data.

● How to Read Master Values Log Data from an SD Memory Card

- 1** Change the value of *Start_ReadMasterData* to TRUE.
The master values log data is read from the SD memory card. When the data is read, the value of *Busy_ReadMasterData* changes to FALSE, and the value of *Ready_MasterData* changes to TRUE.
- 2** Change the value of *Enable* to TRUE.
LogCompare_instance is executed and the value of *Enabled* changes to TRUE.
- 3** Change the value of *Start_Scan* to TRUE.
Measurement values are logged and compared with the master values log data.

● How to Write Master Values Log Data to an SD Memory Card in CSV Format

- 1** Change the value of *Start_SaveMasterData* to TRUE.
The master values log data is written to the SD memory card in CSV format. When the data is written, the value of *Busy_SaveMasterData* changes to FALSE.

Variables

● Internal Variables

Name	Data type	Default	Comment
LogCompare_instance	OmronLib\BC_DeviceMonitor\LogCompare		Instance of Logging Compare FB
LC_Teach	BOOL		Acquire master values
LC_Scan	BOOL		Acquire measurement values
LC_Tolerance	LREAL	LREAL#20	Tolerance value
LC_Logging	BOOL		Logging
LC_Alarm	BOOL		Outside of the tolerance range
LC_AlarmPos	UINT		Position outside of the tolerance range
LC_Error	BOOL		Error
LC_ErrorID	WORD		Error code
LC_ErrorIDEx	DWORD		Expansion error code
LC_TeachLog	OmronLib\BC_DeviceMonitor\sLogData		Master values
LC_ScanLog	OmronLib\BC_DeviceMonitor\sLogData		Measurement values
LogDataCSVRead_instance	OmronLib\BC_DeviceMonitor\Log-DataCSVRead		Instance of Read Log Data from SD Memory Card FB
LogDataCSVWrite_instance	OmronLib\BC_DeviceMonitor\Log-DataCSVWrite		Instance of Write Log Data to SD Memory Card FB
LW_Done	BOOL		Write done
LW_Busy	BOOL		Writing
LW_Error	BOOL		Write error
LW_ErrorID	WORD		Write_error code
LW_ErrorIDEx	DWORD		Write_expansion error code
LR_Done	BOOL		Read done
LR_Busy	BOOL		Reading
LR_Error	BOOL		Read error
LR_ErrorID	WORD		Read_error code
LR_ErrorIDEx	DWORD		Read_expansion error code
GraphTeachedLog	OmronLib\BC_DeviceMonitor\sGraphLog-Data		Graph log data_master values
GraphScanLog	OmronLib\BC_DeviceMonitor\sGraphLog-Data		Graph log data_measurement values
i	UINT		
LC_Enable	BOOL		Enable

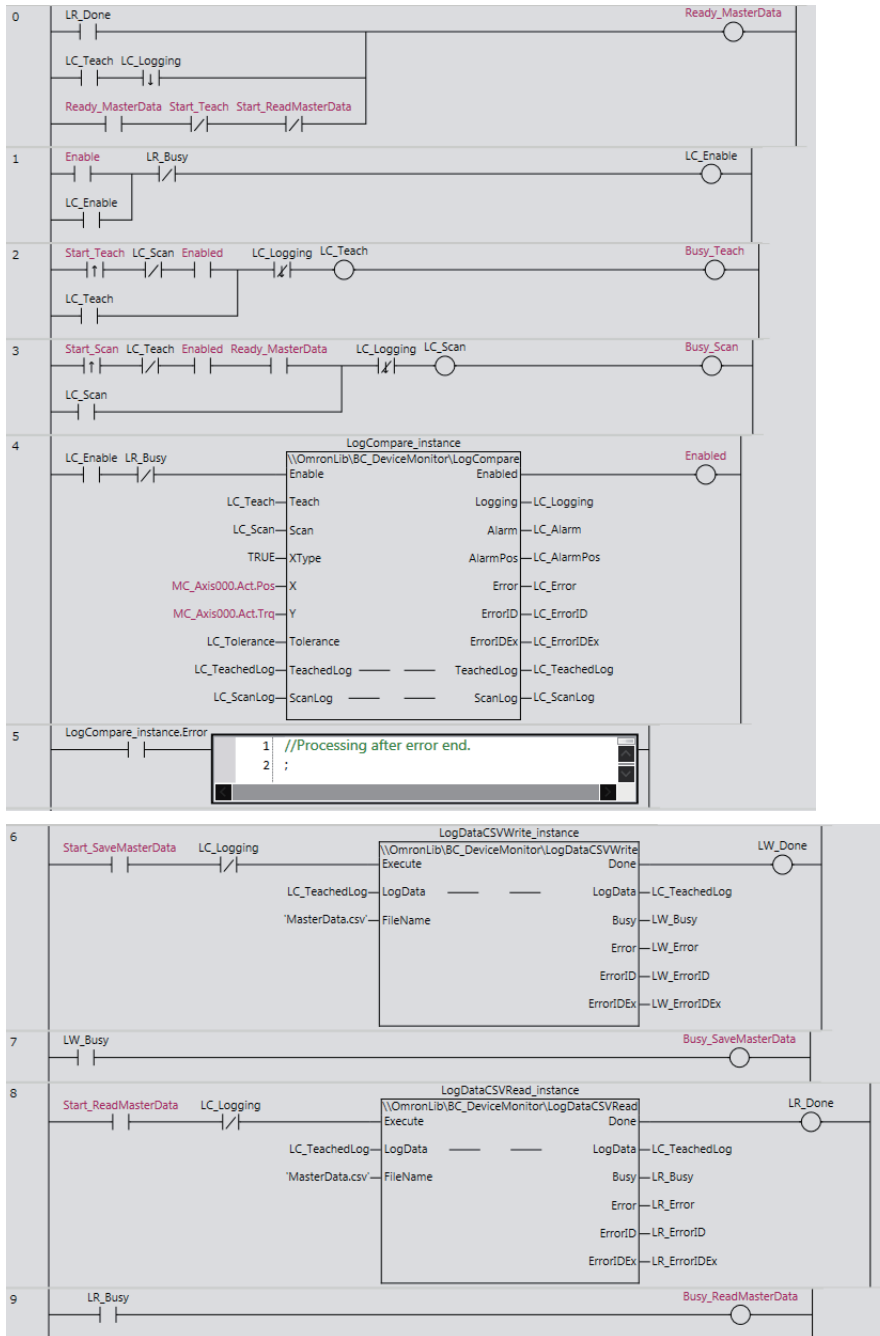
● External Variables

Name	Data type	Constant	Comment
MC_Axis000	_sAxis_Ref	✓	
Start_Scan	BOOL		Acquire measurement values
Start_Teach	BOOL		Acquire master values
Start_ReadMasterData	BOOL		Read master values
Start_SaveMasterData	BOOL		Save master values
Offset	UINT		Offset
X_Magnification	UINT		X axis magnification ratio
Y_Magnification	UINT		Y axis magnification ratio
TeachedLog	sGraphLogData2		Display master values graph
ScanLog	sGraphLogData2		Display measurement values graph
Busy_Teach	BOOL		Acquiring master values
Busy_Scan	BOOL		Acquiring measurement values
Busy_SaveMasterData	BOOL		Saving master values
Busy_ReadMasterData	BOOL		Reading master values
Ready_MasterData	BOOL		Master values are ready
Enable	BOOL		Enable
Enabled	BOOL		Enabled

● Data Type Definitions

Name	Data type	Comment
sGraphLogData2	STRUCT	Graph log data recorder 2
Count	UINT	Number of log data
XLabel	ARRAY[0..20] OF LREAL	X axis scale line label
YLabel	ARRAY[0..20] OF LREAL	Y axis scale line label
X	ARRAY[0..599] OF REAL	REAL X input value
Y	ARRAY[0..599] OF REAL	REAL Y input value
Y_Low	ARRAY[0..599] OF REAL	Allowable REAL Y lower limit
Y_High	ARRAY[0..599] OF REAL	Allowable REAL Y upper limit

Ladder Diagram



10	<pre> \\OmronLib\BC_DeviceMonitor\LogDataToGraph EN Offset—Offset X_Magnification—X_Magnification Y_Magnification—Y_Magnification UINT#3—X_DivisionNum UINT#3—Y_DivisionNum LREAL#110.0—Y_Maximum LC_Tolerance—Tolerance LC_TeachedLog—LogData ———— LogData—LC_TeachedLog GraphTeachedLog—GraphLogData ———— GraphLogData—GraphTeachedLog </pre>
11	<pre> \\OmronLib\BC_DeviceMonitor\LogDataToGraph EN Offset—Offset X_Magnification—X_Magnification Y_Magnification—Y_Magnification UINT#3—X_DivisionNum UINT#3—Y_DivisionNum LREAL#110.0—Y_Maximum LREAL#0.0—Tolerance LC_ScanLog—LogData ———— LogData—LC_ScanLog GraphScanLog—GraphLogData ———— GraphLogData—GraphScanLog </pre>
12	<pre> 1 FOR i:=0 TO 599 BY 1 DO 2 TeachedLog.Y[i]:=LREAL_TO_REAL(GraphTeachedLog.Y[i]); 3 TeachedLog.Y_High[i]:=LREAL_TO_REAL(GraphTeachedLog.Y_High[i]); 4 TeachedLog.Y_Low[i]:=LREAL_TO_REAL(GraphTeachedLog.Y_Low[i]); 5 ScanLog.Y[i]:=LREAL_TO_REAL(GraphScanLog.Y[i]); 6 END_FOR; 7 8 TeachedLog.Count:=GraphTeachedLog.Count; 9 ScanLog.Count:=GraphScanLog.Count; 10 IF(TeachedLog.Count>=ScanLog.Count)THEN 11 ScanLog.Count:=TeachedLog.Count; 12 FOR i:=0 TO 20 BY 1 DO 13 ScanLog.XLabel[i]:=GraphTeachedLog.XLabel[i]; 14 ScanLog.YLabel[i]:=GraphTeachedLog.YLabel[i]; 15 END_FOR; 16 ELSE 17 TeachedLog.Count:=ScanLog.Count; 18 FOR i:=0 TO 20 BY 1 DO 19 ScanLog.XLabel[i]:=GraphScanLog.XLabel[i]; 20 ScanLog.YLabel[i]:=GraphScanLog.YLabel[i]; 21 END_FOR; 22 END_IF; </pre>

CX-Designer Settings

The CX-Designer displays the following four types of graph using the broken-line graph function of NS-series PT.

- Master values log data
- Master values log data plus the tolerance
- Master values log data minus the tolerance
- Measurement values log data

Configure the following settings with CX-Designer. Note that, in the following setting example, the name of the host in the CX-Designer communication setup is set to HOST3.

● **Size Settings in X Axis Direction and Y Axis Direction**

Configure settings in X axis direction and Y axis direction under **Graph** tab in the Broken-line Graph setting window.

- To set the graph size in X axis direction, set *ScanLog.Count* by going to **No. of vertices in each line - Display Points - Indirect Reference**.
- Set 600, which is the maximum value of *GraphLogData.Count* of LogDataToGraph function, by going to **No. of vertices in each line - Monitor Points**.
- To set the graph size in Y axis direction, set **Maximum Limit** for each broken-line to 110, which is the value of the *Y_Maximum* of LogDataToGraph function. Set **Minimum Limit** to 0.

Broken-line Graph - BLG0016

General | Graph | Background | Scroll Bar | Frame | Flicker | Control Flag | Size/Position

Use the graph of a broken-line graph group

No.	Address	Maximum Limit	Minimum Limit	Nor	Out	Line S	Q	S	M	S	Col
1	HOST3:TeachedLog.Y[0]	110	0			Solid	0	J	N		
2	HOST3:TeachedLog.Y_Hieh[0]	110	0			Dotted	0	J	N		
3	HOST3:TeachedLog.Y_Low[0]	110	0			Dotted	0	J	N		
4	HOST3:ScanLog.Y[0]	110	0			Solid	0	J	N		

Draw Value Outside of the Range Storage Type: REAL(Real number)

No. of vertices in each line

Monitor Points:

Display Points:

No. of Points

Indirect Reference Set1

Display start position:

Position

Indirect Reference Set2

Use As Default Display Extension Tabs

Apply OK Cancel Help

Annotations:

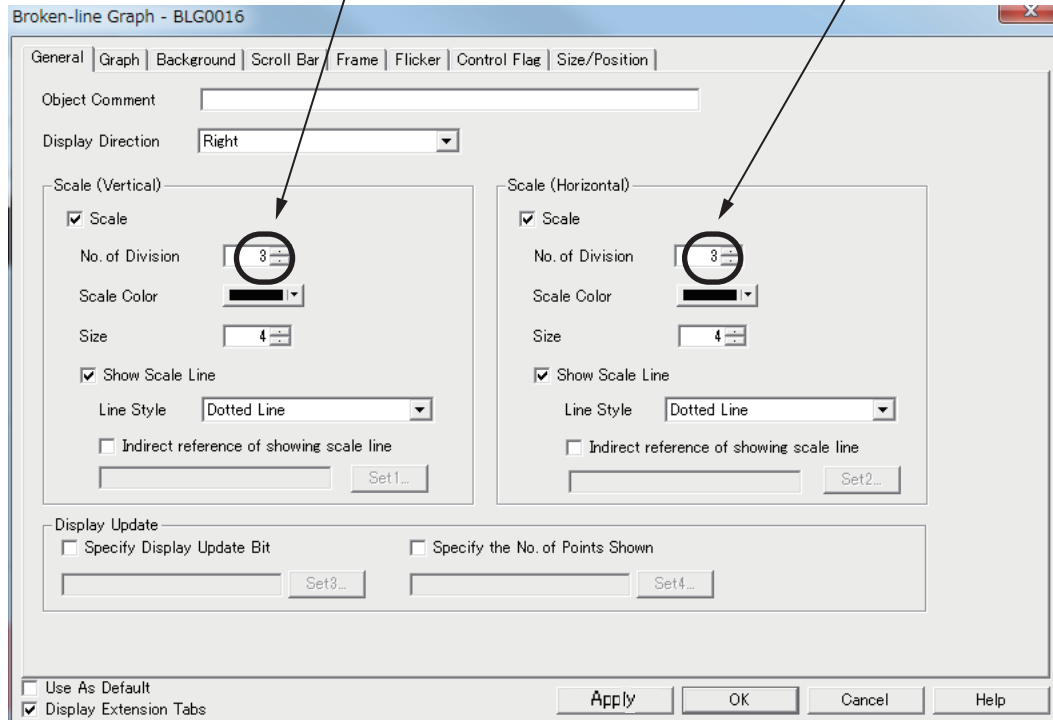
- Set Maximum Limit to 110.
- Master values log data
- Master values log data + tolerance
- Master values log data - tolerance
- Measurement values log data
- Set Monitor Points to 600.
- Set Display Points to ScanLog.Count

● X Axis Scale Line and Y Axis Scale Line Settings

Configure settings for X axis scale line and Y axis scale line under **General** tab in the Broken-line Graph setting window.

- For X axis scale line, set 3, which is the value of $X_DivisionNum$ of LogDataToGraph function, by going to **Scale(Horizontal) - No. of Division**.
- For Y axis scale line, set 3, which is the value of $Y_DivisionNum$ of LogDataToGraph function, by going to **Scale(Vertical) - No. of Division**.

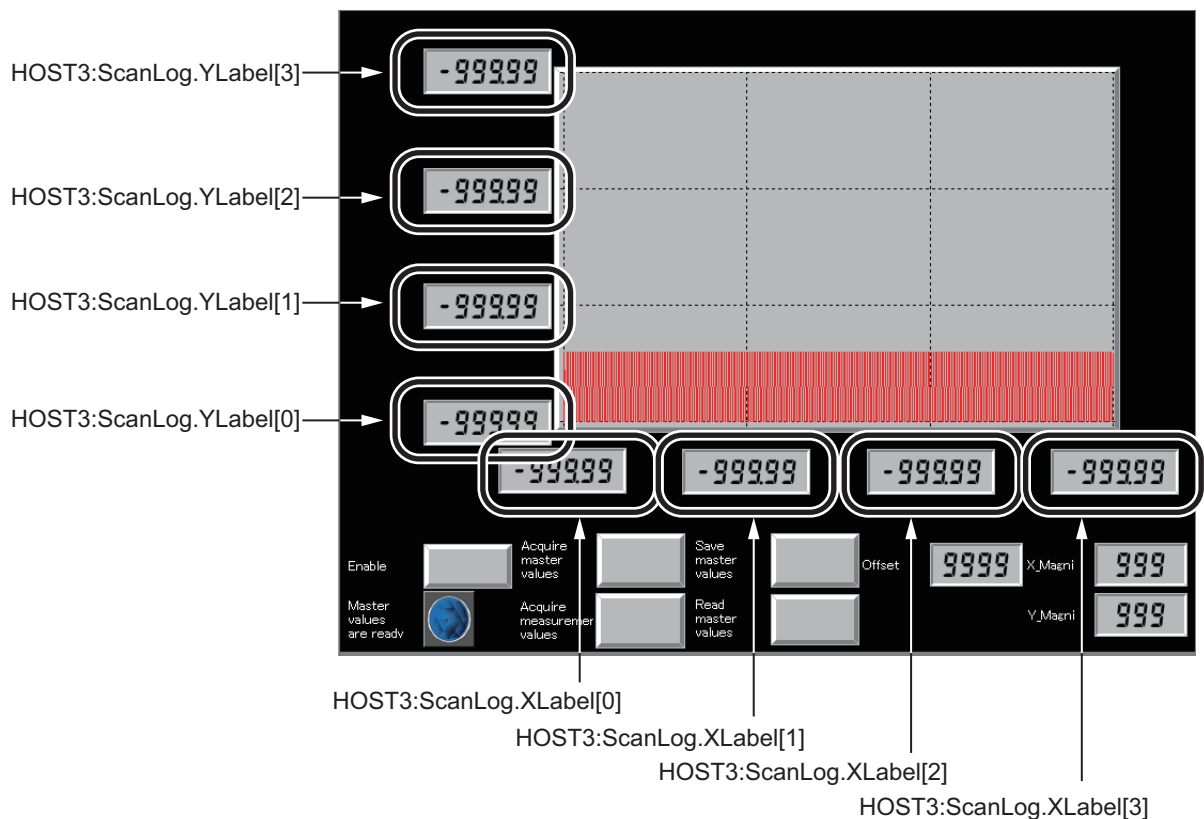
Set **No. of Division** for **Scale(Vertical)** to 3. Set **No. of Division** for **Scale(Horizontal)** to 3.



● **Assignment of Variables to Functional Objects on Screen**

Assume that the broken-line display screen of NS PT is to be configured to the settings listed in the following figure. The variables are assigned to the functional objects on the screen as follows.

Functional Objects	Label	Assigned variable
ON/OFF button	Enable	Write address HOST3:Enable Display address HOST3:Enabled
Bit lamp	Master values are ready	Display address HOST3:Ready_MasterData
ON/OFF button	Acquire master values	Write address HOST3:Start_Teach Display address HOST3:Busy_Teach
ON/OFF button	Acquire measurement values	Write address HOST3:Start_Scan Display address HOST3:Busy_Scan
ON/OFF button	Save master values	Write address HOST3:Start_SaveMasterData Display address HOST3:Busy_SaveMasterData
ON/OFF button	Read master values	Write address HOST3:Start_ReadMasterData Display address HOST3:Busy_ReadMasterData
Numerical Display&Input	Offset	Address HOST3:Offset
Numerical Display&Input	X_Magni	Address HOST3:X_Magnification
Numerical Display&Input	Y_Magni	Address HOST3:Y_Magnification
Numerical Display&Input		Address HOST3:ScanLog.YLabel[0]
Numerical Display&Input		Address HOST3:ScanLog.YLabel[1]
Numerical Display&Input		Address HOST3:ScanLog.YLabel[2]
Numerical Display&Input		Address HOST3:ScanLog.YLabel[3]
Numerical Display&Input		Address HOST3:ScanLog.XLabel[0]
Numerical Display&Input		Address HOST3:ScanLog.XLabel[1]
Numerical Display&Input		Address HOST3:ScanLog.XLabel[2]
Numerical Display&Input		Address HOST3:ScanLog.XLabel[3]



**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 programming actual applications, also program safety circuits, device interlocks, I/O with other devices, and other control procedures.
 - Create a user program that will produce the intended device operation.
 - Check the user program, data, and parameter settings for proper execution before you use them for actual operation.
-

LogDataToGraph

The LogDataToGraph function block converts log data that was acquired with LogCompare function block to the data format that is suitable for displaying as a broken-line graph on NS-series PT.

Function block name	Name	FB/FUN	Graphic expression	ST expression
LogDataTo-Graph	Display Log Data	FUN		<pre>Out := LogDataToGraph(Offset, X_Magnification, Y_Magnification, X_DivisionNum, Y_DivisionNum, Y_Maximum, Tolerance, LogData, GraphLogData);</pre>

Function Block and Function Information

Item	Description
Library file name	OmronLib_BC_DeviceMonitor_V1_0.slr
Namespace	OmronLib\BC_DeviceMonitor
Function block and function number	00039
Publish/Do not publish source code	Publish
Function block and function version	1.00

Compatible Models

Item	Name	Model numbers	Version
Device	NS-series PT	NS□-□□□□□-V2	<ul style="list-style-type: none"> When the CPU Unit is NJ501-□□□□, version 8.5 or later When the CPU Unit is NJ301-□□□□ or NJ101-□□□□, version 8.61 or later When the CPU Unit is NX701-□□□□, version 8.9 or later

Variables

	Meaning	I/O	Description	Valid range	Unit	Default
Offset	Offset	Input	First position of the log data that is displayed on the graph	0 to 1999	–	0
X_Magnification	X Axis Magnification Ratio		X axis magnification ratio for graph display	1 to 100	–	1
Y_Magnification	Y Axis Magnification Ratio		Y axis magnification ratio for graph display	1 to 100	–	1
X_DivisionNum	Number of X Axis Divisions		Number of X axis scale divisions for graph display	1 to 20	–	1
Y_DivisionNum	Number of Y Axis Divisions		Number of Y axis scale divisions for graph display	1 to 20	–	1
Y_Maximum	Maximum Y Axis Value		Maximum Y axis value for graph display	Depends on data type.	–	1000
Tolerance	Tolerance		Tolerance value	Depends on data type.	–	0
Out	Return Value	Output	Function execution results TRUE: Normal end FALSE: Error end	Depends on data type.	–	–
LogData	Log Data Recorder	Input/output	Log data recorder	–	–	–
GraphLogData	Graph Log Data Recorder		Graph log data recorder	–	–	–

	Boo lean	Bit strings					Integers							Real numbers		Times, durations, dates, and text strings				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Offset							OK													
X_Magnification							OK													
Y_Magnification							OK													
X_DivisionNum							OK													
Y_DivisionNum							OK													
Y_Maximum														OK						
Tolerance														OK						
Out	OK																			
LogData	Refer to Function for details on the structure OmronLib\BC_DeviceMonitor\IsLogData.																			
GraphLogData	Refer to Function for details on the structure OmronLib\BC_DeviceMonitor\IsGraphLogData.																			

Function

This function block converts the measurement values that were acquired with the LogCompare function block or master value *LogData* (Log Data Recorder) to the data format that is suitable for displaying as a broken-line graph on NS-series PT and stores in *GraphLogData* (Graph Log Data Recorder).

The structure of the log data recorder is the same as *ScanLog* or *TeacedLog* for the LogCompare function. Refer to *LogCompare* on page 71 for the log data recorder specifications.

GraphLogData Structure

The *GraphLogData* can store 600 log data after the data format conversion.

The data type of *GraphLogData* is the structure `OmronLib\BC_DeviceMonitor\GraphLogData`. The specifications are as follows:

Name	Meaning	Description	Data type	Valid range	Unit	Default
GraphLogData	Graph Log Data Recorder	Structure to store log data after the data format conversion	OmronLib\BC_DeviceMonitor\GraphLogData			
Count	Number of Log Data	Number of log data that is stored in the graph log data recorder	UINT	0 to 600	–	–
XLabel	X Axis Scale Line Label	X axis scale line label with the number of divisions specified for X_DivisionNum	ARRAY[0..20] OF LREAL	Depends on data type.	–	–
YLabel	Y Axis Scale Line Label	Y axis scale line label with the number of divisions specified for Y_DivisionNum	ARRAY[0..20] OF LREAL	Depends on data type.	–	–
X	X Input Value	Array to store X input values of the log data	ARRAY[0..599] OF LREAL	Depends on data type.	–	–
Y	Y Input Value	Array to store Y input values of the log data	ARRAY[0..599] OF LREAL	Depends on data type.	–	–
Y_Low	Allowable Y Lower Limit	Array to store the lower limit of Y input values of the log data that contains allowable errors	ARRAY[0..599] OF LREAL	Depends on data type.	–	–
Y_High	Allowable Y Upper Limit	Array to store the upper limit of Y input values of the log data that contains allowable errors	ARRAY[0..599] OF LREAL	Depends on data type.	–	–

Meanings of Input Parameters

Offset (Offset), *Magnification* (Magnification Ratio), and *DivisionNum* (Number of Divisions) input parameters have the following meanings.

- **Offset (Offset)**

This specifies the place of the log data from the top whose data format is to be converted first. The *Offset* value is the X and Y array element numbers of the log data whose data format is to be converted first.

- **X_Magnification (X Axis Magnification Ratio)**

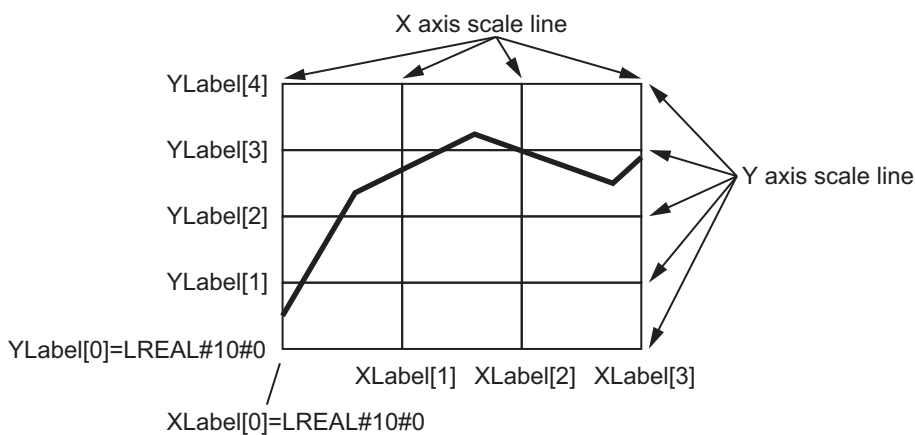
This specifies the X axis magnification ratio for graph display. When the number of log data stored in the graph log data recorder is reduced, the graph display is extended in the X axis direction according to the *X_Magnification* value. The number of log data whose data format is to be converted is $1/X_Magnification$ from the top of the log data recorder. For example, with *X_Magnification* = UINT#10#2, the data format of $600/2 = 300$ log data from the top is converted.

- **Y_Magnification (Y Axis Magnification Ratio)**

This specifies the Y axis magnification ratio for graph display. When the log data format is converted, the Y input value of the log data is multiplied by the *Y_Magnification* value.

- **X_DivisionNum (Number of X Axis Divisions), Y_DivisionNum (Number of Y Axis Divisions)**

These are the numbers of X and Y axis scale divisions that are used for the broken-line graph function of NS-series PT. For example, with *X_DivisionNum* = UINT#10#3 and *Y_DivisionNum* = UINT#10#4, the X axis has 4 scale lines and Y axis has 5 scale lines as shown in the following figure. According to the values of *X_DivisionNum* and *Y_DivisionNum*, the values of X axis scale line label and Y axis scale line label of the graph log data recorder are calculated automatically. The values of *XLabel[0]* and *YLabel[0]* are always UINT#10#0.



- **Y_Maximum (Maximum Y Axis Value)**

This specifies the Y axis maximum value for graph display. The maximum value of the Y axis scale line label *YLabel[Y_DivisionNum]* is calculated with the following equation.

$$YLabel[Y_DivisionNum] = Y_Maximum / Y_Magnification$$

● Tolerance (Tolerance)

This specifies the allowable error values when you want to include allowable errors to display Y input values of the log data as a graph. From the *Tolerance* value, the *Y_Low* (Allowable Y Lower Limit) and *Y_High* (Allowable Y Upper Limit) values are calculated with the following equation.

$$Y_Low[n] = \text{LogData.Y}[n] - \text{Tolerance}$$

$$Y_High[n] = \text{LogData.Y}[n] + \text{Tolerance} \quad n: \text{Array element number}$$

Log Data Whose Data Format is to be Converted

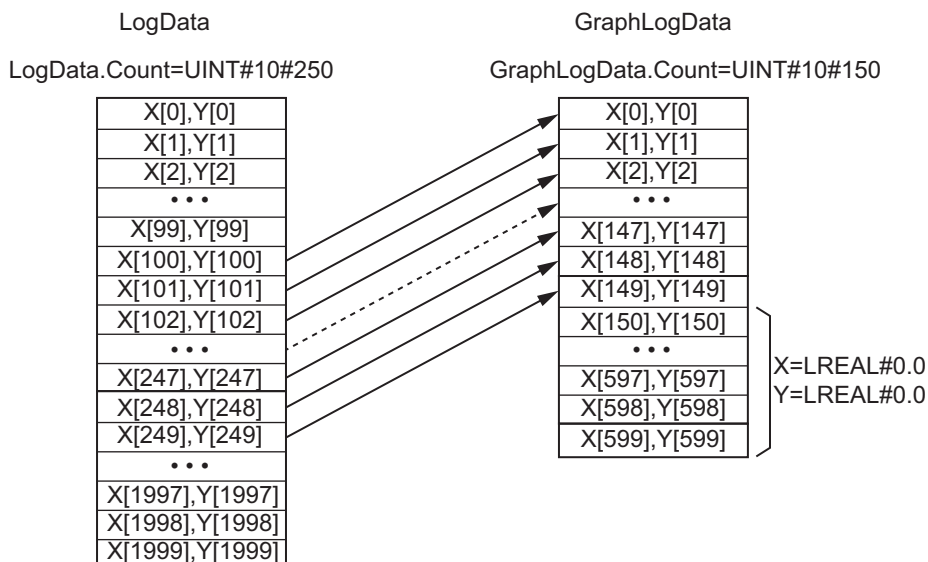
The log data in the log data recorder whose data format is to be converted is determined by the following three elements.

- The number of log data in the log data recorder.
- *Offset* value
- *X_Magnification* value

Out of all log data specified for *Offset*, the data format of 600 log data is to be converted. However, the number of log data whose data format is to be converted varies with *X_Magnification* values.

For example, when the number of log data in the log data recorder is 250, *Offset* = UINT#10#100, and *X_Magnification* = UINT#2, the number of log data whose data format is to be converted is 150 as shown in the following figure. Therefore, the value of the number of log data in the graph log data recorder *GraphLogData.Count* is UINT#10#150.

Also, when the number of log data whose data format is to be converted is less than 600, the remaining array element values are $X = \text{LREAL}\#0.0$ and $Y = \text{LREAL}\#0.0$. In the example shown in the following figure, the number of log data whose data format is to be converted is 150, thus the remaining array element value is 450.



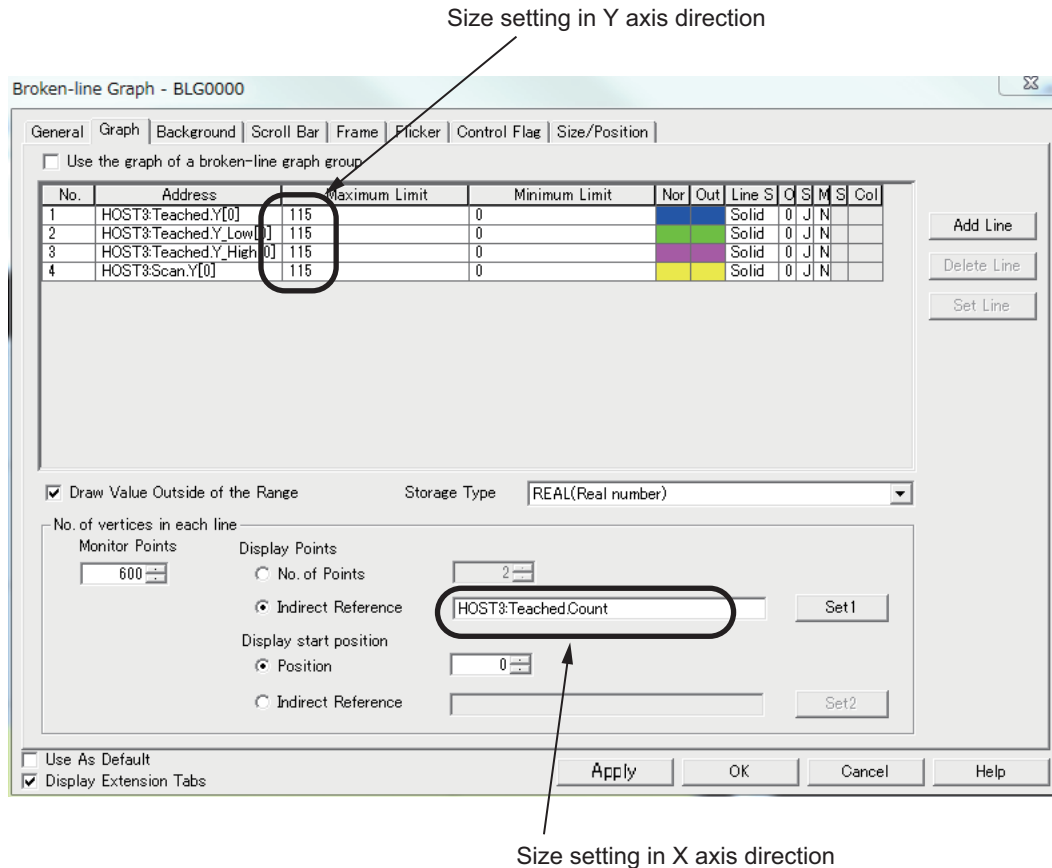
CX-Designer Settings

To display the log data as a graph, you need to configure the following settings with CX-Designer.

● Size settings in X axis direction and Y axis direction

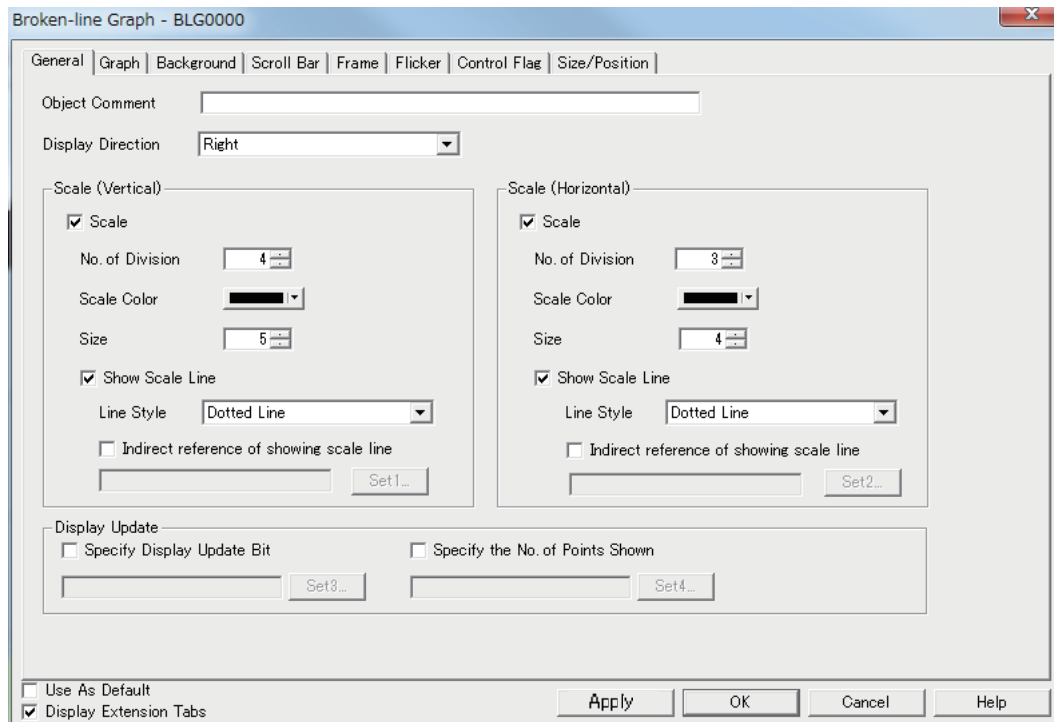
To secure the graph size in X axis direction, in the Broken-line Graph setting window, set *GraphLog-Data.Count* by going to **Graph** tab - **No. of vertices in each line - Display Points - Indirect Reference**.

Also, set the *Y_Maximum* value as the upper limit of each broken line graph. *Y_Maximum* = 115 in the figure below.



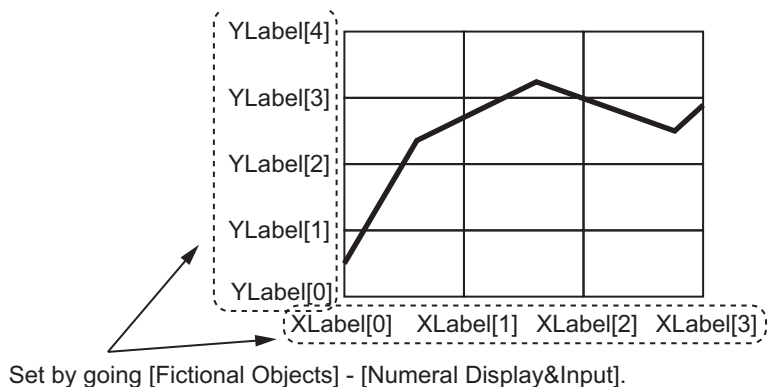
● X axis scale line and Y axis scale line settings

Set **Scale(Vertical)** and **Scale(Horizontal)** under **General** tab in the Broken-line Graph setting window.



● X axis scale line label and Y axis scale line label display

Set the X axis scale line label and Y axis scale line label by going **Fictional Objects - Numeral Display&Input**. Refer to *GraphLogData.Xlabel* for the X axis scale line value, *GraphLogData.Ylabel* for the Y axis scale line value.



Reference

Refer to the *CX-Designer USER'S MANUAL (V099)* for details on how to use the CX-Designer.

Precautions for Correct Use

Set the X input value of the log data recorder so that it is monotonically increased. In other words, set the X[0] value to be the minimum X input value and the X[LogData.Count-1] value to be the maximum X input value. Otherwise the value of the X axis scale line label for the graph log data recorder may become invalid.

LogDataCSVWrite

The LogDataCSVWrite function block writes the log data that is acquired with the LogCompare function block to an SD memory card in CSV format.

Function block name	Name	FB/FUN	Graphic expression	ST expression
LogData CSVWrite	Write Log Data to SD Memory Card	FB		LogDataCSVWrite_instance(Execute, LogData, FileName, Done, Busy, Error, ErrorID, ErrorIDEx);

Function Block and Function Information

Item	Description
Library file name	OmronLib_BC_DeviceMonitor_V1_0.slr
Namespace	OmronLib\BC_DeviceMonitor
Function block and function number	00040
Publish/Do not publish source code	Publish
Function block and function version	1.00

Compatible Models

Item	Name	Model numbers	Version
Device	SD Memory Card	HMC-SD□□□	—

Variables

	Meaning	I/O	Description	Valid range	Unit	Default
Execute	Execute	Input	TRUE: Executes the instruction. FALSE: Does not execute the instruction.	TRUE or FALSE	–	FALSE
LogData	Log Data Recorder	Input/output	Log data recorder	–	–	–
FileName	File Name	Input	File name of CSV file to write	66 bytes max. (65 single-byte alphanumeric characters plus the final NULL character)	–	–
Done	Done	Output	TRUE: Normal end FALSE: Error end, execution in progress, or execution condition not met	TRUE or FALSE	–	–
Busy	Executing	Output	TRUE: Execution processing is in progress. FALSE: Execution processing is not in progress.	TRUE or FALSE	–	–
Error	Error	Output	TRUE: Error end FALSE: Normal end, execution in progress, or execution condition not met	TRUE or FALSE	–	–
ErrorID	Error Code	Output	This is an error code for an error end. The value is 16#0 for a normal end.	*1	–	–
ErrorIDEx	Expansion Error Code	Output	This is an expansion error code for an error end. The value is 16#0 for a normal end.	*1	–	–

*1. Refer to *Troubleshooting* on page 100 for details.

	Boolean	Bit strings					Integers							Real numbers		Times, durations, dates, and text strings				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	OK																			
LogData	Refer to Function for details on the structure OmronLib\BC_DeviceMonitor\sLogData.																			
FileName																				OK
Done	OK																			
Busy	OK																			
Error	OK																			
ErrorID			OK																	
ErrorIDEx				OK																

Function

When *Execute* (Execute) changes to TRUE, this function block writes all the log data that is stored in *LogData* (Log Data 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, create *FileName* in the root of the SD Memory Card.

The structure of the log data recorder is the same as *ScanLog* or *TeacedLog* for the LogCompare function. Refer to *LogCompare* on page 71 for the log data recorder specifications.

CSV File Format

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

LogData.Count	
LogData.X[0]	LogData.Y[0]
LogData.X[1]	LogData.Y[1]
...	...
...	...
LogData.X[LogData.Count-2]	LogData.Y[LogData.Count-2]
LogData.X[LogData.Count-1]	LogData.Y[LogData.Count-1]

LogData.Count is converted to a text string and written with the UINT_TO_STRING instruction. Refer to the instructions reference manual for details on the UINT_TO_STRING instruction.

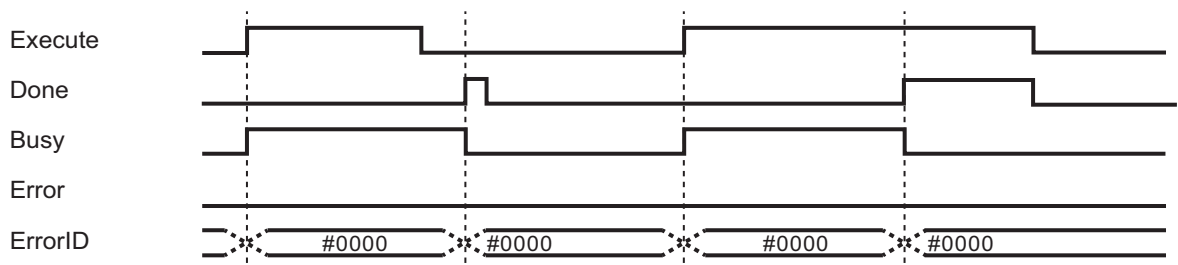
LogData.X and *LogData.Y* are converted to text strings and written with the LrealToFormatString instruction. For the number of digits, the overall is set to eight and the fractional part is to six. Refer to the instructions reference manual for details on the LrealToFormatString instruction.

Timing Charts

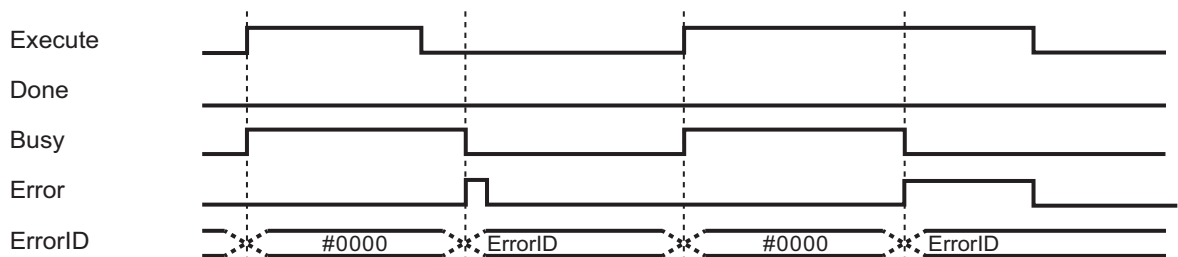
The following figures show the timing charts for the program part.

- *Busy* (Executing) changes to TRUE when *Execute* (Execute) changes to TRUE.
- *Done* changes to TRUE when the data output operation is completed.
- 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* (Execute) changes to FALSE before execution of the function block is ended, *Done* (Done) and *Error* (Error) are TRUE only for one task period after execution of the function block is ended.
- If *Execute* (Execute) remains TRUE even after execution of the function block is ended, the output values of *Done* (Done) and *Error* (Error) are retained.

● Timing Chart for Normal End



● Timing Chart for Error End



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.
- The number of array elements for log data in the log data recorder is defined by the user.
- 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.

**Precautions for Correct Use**

- Do not execute the same instance while an instance is being executed.
 - When you execute the LogDataCSVWrite function block, always stop the LogDataCSVRead functions beforehand. Also, wait until processing the LogCompare function block is completed. If you execute the LogDataCSVWrite 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 content of the log data recorder is 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: Can be used. FALSE: 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: An error occurred. FALSE: No error occurred.
_Card1Access	SD Memory Card Access Flag	BOOL	This flag indicates if the SD Memory Card is currently being accessed. TRUE: Being accessed. FALSE: 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

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.

Error code	Expansion error code	Status	Description	Correction
16#140E	16#00000000	SD Memory Card Access	The SD Memory Card is broken.	Replace the SD Memory Card.
	16#00000000	Failed	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.
16#3C2A	16#00000001	LogData Definition Error	The number of array elements of <i>LogData.X</i> (X Input Value) and the number of array elements of <i>LogData.Y</i> (Y Input Value) do not match.	Check the <i>LogData</i> definition.
	16#00000002	LogData Value Error	The value of <i>LogData.Count</i> (Number of Log Data) exceeds the number of array elements of <i>LogData.X</i> (X Input Value) or the number of array elements of <i>LogData.Y</i> (Y Input Value).	Check the value of <i>LogData.Count</i> (Number of Log Data).
	16#00000003	No Data Stored in Log Data	There is no log data in <i>LogData</i> . (<i>LogData.Count</i> = UINT#10#0)	Check to see if one or more log data are stored in <i>LogData</i> .

LogDataCSVRead

The LogDataCSVRead function block reads the log data that is used with the LogCompare function block from an SD memory card.

Function block name	Name	FB/FUN	Graphic expression	ST expression
LogData CSVRead	Read Log Data from SD Memory Card	FB		LogDataCSVRead_instance(Execute, LogData, FileName, Done, Busy, Error, ErrorID, ErrorIDEx);

Function Block and Function Information

Item	Description
Library file name	OmronLib_BC_DeviceMonitor_V1_0.slr
Namespace	OmronLib\BC_DeviceMonitor
Function block and function number	00041
Publish/Do not publish source code	Publish
Function block and function version	1.00

Compatible Models

Item	Name	Model numbers	Version
Device	SD Memory Card	HMC-SD□□□	—

Variables

	Meaning	I/O	Description	Valid range	Unit	Default
Execute	Execute	Input	TRUE: Executes the instruction. FALSE: Does not execute the instruction.	TRUE or FALSE	–	FALSE
LogData	Log Data Recorder	Input/output	Log data recorder	–	–	–
FileName	File Name	Input	File name of CSV file to read	66 bytes max. (65 single-byte alphanumeric characters plus the final NULL character)	–	–
Done	Done	Output	TRUE: Normal end FALSE: Error end, execution in progress, or execution condition not met	TRUE or FALSE	–	–
Busy	Executing	Output	TRUE: Execution processing is in progress. FALSE: Execution processing is not in progress.	TRUE or FALSE	–	–
Error	Error	Output	TRUE: Error end FALSE: Normal end, execution in progress, or execution condition not met	TRUE or FALSE	–	–
ErrorID	Error Code	Output	This is an error code for an error end. The value is 16#0 for a normal end.	*1	–	–
ErrorIDEx	Expansion Error Code	Output	This is an expansion error code for an error end. The value is 16#0 for a normal end.	*1	–	–

*1. Refer to *Troubleshooting* on page 108 for details.

	Boolean	Bit strings					Integers							Real numbers		Times, durations, dates, and text strings				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	OK																			
LogData	Refer to Function for details on the structure OmronLib\BC_DeviceMonitor\sLogData.																			
FileName																				OK
Done	OK																			
Busy	OK																			
Error	OK																			
ErrorID			OK																	
ErrorIDEx				OK																

Function

When *Execute* (Execute) changes to TRUE, this function block reads the log data that is stored in an SD memory card in CSV format and stores in *LogData* (Log Data Recorder). The name of the file to read is specified with *FileName* (File Name).

With *FileName*, you can specify the name including the folder. If the folder is not specified, read the file that exists in the root of the SD Memory Card.

The structure of the log data recorder is the same as *ScanLog* or *TeacedLog* for the LogCompare function. Refer to *LogCompare* on page 71 for the log data recorder specifications.

CSV File Format

The format of the CSV file to read is as follows.

LogData.Count	
LogData.X[0]	LogData.Y[0]
LogData.X[1]	LogData.Y[1]
...	...
...	...
LogData.X[LogData.Count-2]	LogData.Y[LogData.Count-2]
LogData.X[LogData.Count-1]	LogData.Y[LogData.Count-1]

LogData.Count is converted to a numeric value and read with the STRING_TO_UINT instruction. Refer to the instructions reference manual for details on the STRING_TO_UINT instruction.

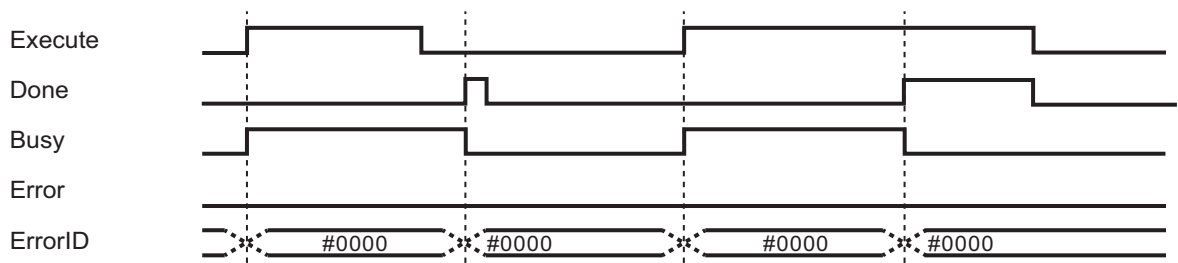
LogData.X and *LogData.Y* are converted to numeric values and read with the STRING_TO_LREAL instruction. Refer to the instructions reference manual for details on the STRING_TO_LREAL instruction.

Timing Charts

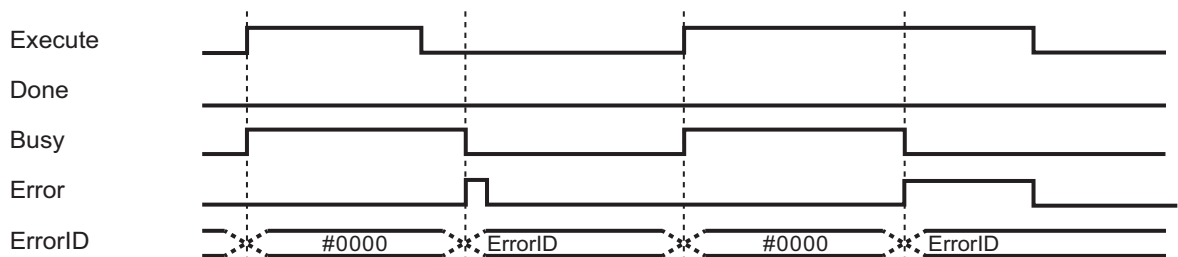
The following figures show the timing charts for the program part.

- *Busy* (Executing) changes to TRUE when *Execute* (Execute) changes to TRUE.
- *Done* changes to TRUE when the data input operation is completed.
- 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* (Execute) changes to FALSE before execution of the function block is ended, *Done* (Done) and *Error* (Error) are TRUE only for one task period after execution of the function block is ended.
- If *Execute* (Execute) remains TRUE even after execution of the function block is ended, the output values of *Done* (Done) and *Error* (Error) are retained.

● Timing Chart for Normal End



● Timing Chart for Error End



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.
- The number of array elements for log data in the log data recorder is defined by the user.
- 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 file specified by *FileName* does not exist.
 - c) The value of *FileName* is not a valid file name.
 - d) The file specified by *FileName* is being accessed.
 - e) An error that prevents access occurs during SD Memory Card access.

**Precautions for Correct Use**

- Do not execute the same instance while an instance is being executed.
 - When you execute the LogDataCSVRead function block, always stop the LogDataCSVWrite function block beforehand. Also, wait until processing the LogCompare function block is completed. If you execute the LogDataCSVRead function block without stopping them, it would take longer to read from the SD Memory Card resulting in an operation error.
 - When the power supply is turned OFF to the Controller, the content of the log data recorder is discarded.
 - Do not turn OFF the power supply to the Controller while data is being read from 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: Can be used. FALSE: 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: An error occurred. FALSE: No error occurred.
_Card1Access	SD Memory Card Access Flag	BOOL	This flag indicates if the SD Memory Card is currently being accessed. TRUE: Being accessed. FALSE: 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

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#1403	16#00000000	File Does Not Exist	The specified directory does not exist.	Specify an existing directory.
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#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.
16#3C2B	16#00000001	LogData Definition Error	The number of array elements of <i>LogData.X</i> (X Input Value) and the number of array elements of <i>LogData.Y</i> (Y Input Value) do not match.	Check the <i>LogData</i> definition.
	16#00000002	Invalid Data Format	The file format specified by <i>FileName</i> (File Name) is not a readable format.	Check the specified file format.

MonitorLightSensor

The MonitorLightSensor function block monitors the amount of light received by the photoelectric sensor and outputs an alarm when the amount of light received is low.

Function block name	Name	FB/FUN	Graphic expression	ST expression
Monitor-LightSensor	Monitor Photoelectric Sensor Device Operation	FB		<pre>MonitorLightSensor_instance(Enable, IncidentLevel, Threshold, AlarmLevel, TolerableCount, Enabled, Alarm, Error, ErrorID, ErrorIDEx);</pre>

Function Block and Function Information

Item	Description
Library file name	OmronLib_BC_DeviceMonitor_V1_0.slr
Namespace	OmronLib\BC_DeviceMonitor
Function block and function number	00021
Publish/Do not publish source code	Do not publish
Function block and function version	1.00

Compatible Models

Item	Name	Model numbers	Version
Device	Sensor	E3NW-□□□□	

Variables

	Meaning	I/O	Description	Valid range	Unit	Default
Enable	Enable	Input	TRUE: Enable FALSE: Disable	TRUE or FALSE	-	FALSE
IncidentLevel	Amount of Light Received	Input	Current value of the amount of light received	-1999 to 9999		FALSE
Threshold	Object Detection Threshold	Input	Threshold for amount of light received to determine the presence/absence of a detectable object	-19999999 to 99999999		FALSE
AlarmLevel	Normal Amount of Light Received	Input	Amount of light received in normal condition	-19999999 to 99999999		FALSE
Tolerable-Count	Alarm Threshold	Input	Threshold for the internal counter for alarm occurrence	2 to 65534		2
Enabled	Enabled	Output	TRUE: Enabled FALSE: Disabled	TRUE or FALSE	-	-
Alarm	Alarm	Output	TRUE: Alarm occurring FALSE: No alarm occurring	TRUE or FALSE		-
Error	Error	Output	TRUE: Error end FALSE: Normal end, execution in progress, or execution condition not met	TRUE or FALSE		-
ErrorID	Error Code	Output	This is an error code for an error end. The value is 16#0 for a normal end.	*1		-
ErrorIDEx	Expansion Error Code	Output	This is an expansion error code for an error end. The value is 16#0 for a normal end.	*1		-

*1. Refer to *Troubleshooting* on page 114 for details.

	Boo lean	Bit strings					Integers							Real num bers		Times, durations, dates, and text strings				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Enable	OK																			
IncidentLevel							OK													
Threshold							OK													
AlarmLevel							OK													
TolerableCount							OK													
Enabled	OK																			
Alarm	OK																			
Error	OK																			
ErrorID			OK																	
ErrorIDEx				OK																

Function

This function block monitors the amount of light received by the photoelectric sensor and outputs an alarm when the amount of light received is low.

Operation starts when *Enable* (Enable) is set to TRUE.

Criterion for Determining Whether Amount of Light Received by Photoelectric Sensor Is Low

When the photoelectric sensor receives light, the amount of light received need to be sufficiently high. This function block determines that the amount of light for the photoelectric sensor is low when the amount of light received does not reach sufficiently high amount.

The following 4 input variables are used to determine whether the amount of light received is low.

Input variables	Name	Meaning
IncidentLevel	Amount of Light Received	Current amount of light received by the photoelectric sensor
AlarmLevel	Normal Amount of Light Received	Amount of light received when the photoelectric sensor is normal
Threshold	Object Detection Threshold	Threshold for amount of light received to determine the presence/absence of a detectable object
TolerableCount	Alarm Threshold	Threshold for the internal counter for alarm occurrence Alarm occurred: The value of the internal counter exceeds <i>TolerableCount</i> (Alarm Threshold) No alarm occurred: The value of the internal counter is less than <i>TolerableCount</i> (Alarm Threshold)

This function block monitors the value of *IncidentLevel* (Amount of Light Received) and determines that the amount of light received by the photoelectric sensor is low when the behavior of *IncidentLevel* (Amount of Light Received) meets all of the following conditions.

- The number of times the value of *IncidentLevel* (Amount of Light Received) changes from larger than *Threshold* (Object Detection Threshold) to smaller than *Threshold* is equal to or greater than *TolerableCount* (Alarm Threshold).
- During that time, the value of *IncidentLevel* (Amount of Light Received) never exceeds the value of *AlarmLevel* (Normal Amount of Light Received).

Processing for This Function Block

The processing for this function block is as follows:

- If the value of *Enable* is changed to TRUE, the operation starts and the internal counter is cleared to zero. The internal counter means the counter which records the time count when *IncidentLevel* (Amount of Light Received) did not reach *AlarmLevel* (Normal Amount of Light Received).
- The value of *IncidentLevel* (Amount of Light Received) is monitored and when the value of *IncidentLevel* (Amount of Light Received) changes from larger than *Threshold* (Object Detection Threshold) to smaller than *Threshold*, the internal counter is incremented.
- When the value of *IncidentLevel* (Amount of Light Received) exceeds the value of *AlarmLevel* (Normal Amount of Light Received), the value of the internal counter is cleared to 0.
- When the value of the internal counter is equal to or greater than *TolerableCount* (Alarm Threshold), the value of *Alarm* (Alarm) changes to TRUE.
- When the value of *IncidentLevel* (Amount of Light Received) exceeds the value of *AlarmLevel* (Normal Amount of Light Received), the value of *Alarm* (Alarm) changes to FALSE. The value of the internal counter is also cleared to 0.
- When *Enable* (Enable) changes to FALSE, the value of *Alarm* (Alarm) changes to FALSE.

Refer to *Timing Charts* on page 113 for details.

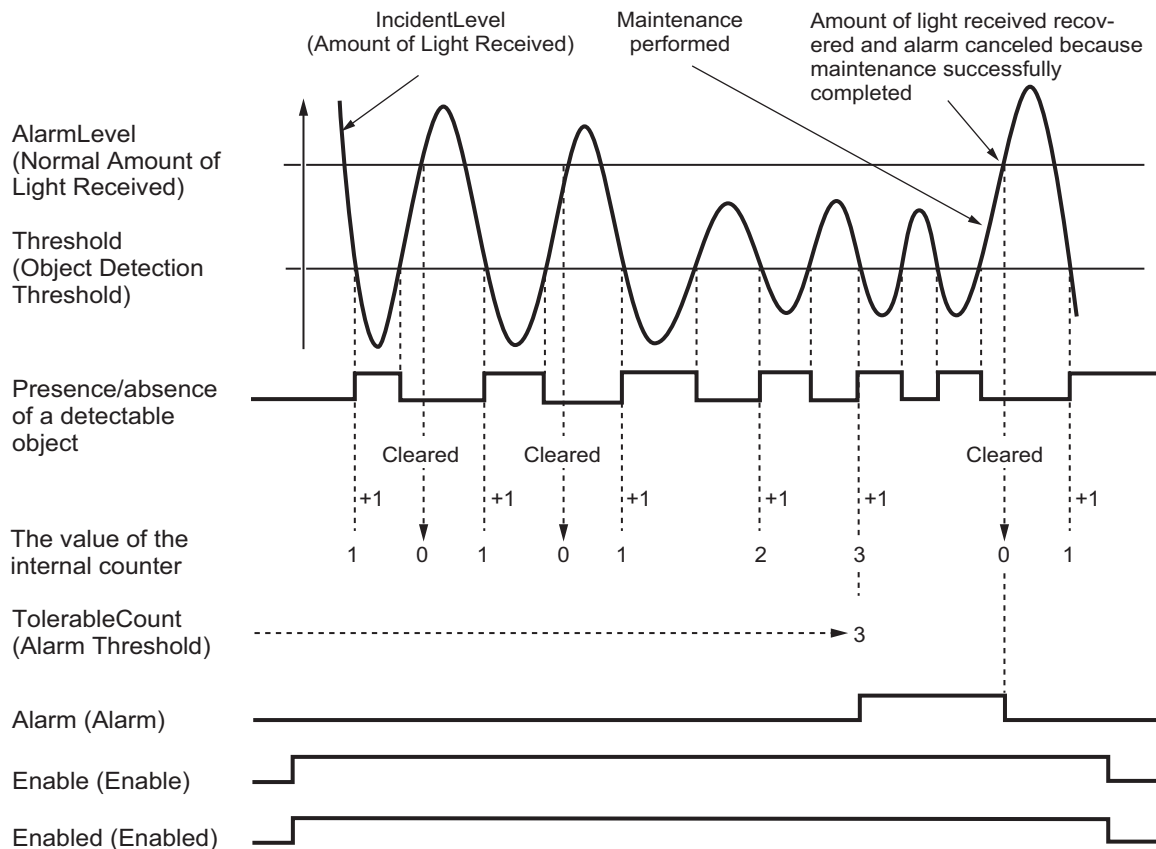
When the value of *Threshold* (Object Detection Threshold) is set equal to or greater than the value of *AlarmLevel* (Normal Amount of Light Received), an error occurs.

Timing Charts

The following figures show the timing charts for the program part.

- *Enabled* (Enabled) changes to TRUE at the same time as *Enable* (Enable) changes to TRUE. *TolerableCount* (Alarm Threshold) is cleared to 0 and *Alarm* (Alarm) changes to FALSE.
- When *Enable* (Enable) changes to FALSE, *Enabled* (Enabled) changes to FALSE. *Alarm* (Alarm) changes to FALSE.

TolerableCount (Alarm Threshold) = UINT#10#3.



Troubleshooting

Error code	Expansion error code	Status	Description	Correction
16#0000	16#00000000	Normal End	–	–
16#0400	16#00000000	Input Value Out of Range	The value of <i>IncidentLevel</i> (Amount of Light Received) setting is out of range.	Set the value of <i>IncidentLevel</i> (Amount of Light Received) from –1999 to 9999.
	16#00000000		The value of <i>Threshold</i> (Object Detection Threshold) setting is out of range	Set the value of <i>Threshold</i> (Object Detection Threshold) from –19999999 to 99999999.
	16#00000000		The value of <i>AlarmLevel</i> (Normal Amount of Light Received) is out of range.	Set the value of <i>AlarmLevel</i> (Normal Amount of Light Received) from –19999999 to 99999999.
	16#00000000		The value of <i>TolerableCount</i> (Alarm Threshold) is 0.	Set the value of <i>TolerableCount</i> (Alarm Threshold) from 2 to 65535.

Sample Programming

Description of Operation

AlarmLevel (Normal Amount of Light Received) is the value when 5000 is added to the value of *Threshold* (Object Detection Threshold).

Input the alarm threshold with *TolerableCount* (Alarm Threshold).

When the value of the internal counter exceeds *TolerableCount* (Alarm Threshold), *Alarm* (Alarm) is generated.

A PDO is not mapped for the amount of light received by the photoelectric sensor (*E001_No_01_Detection_Level_IN1*) as default, map a PDO before using the sensor.

Variables

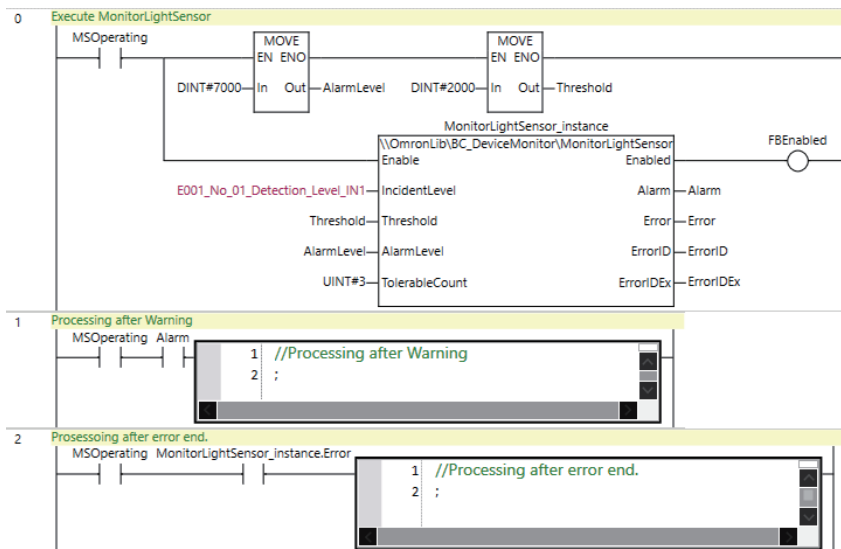
● Internal Variables

Name	Data type	Default	Comment
MonitorLightSensor_instance	OmronLib\BC_DeviceMonitor\MonitorLightSensor	–	Instance of Monitor Photoelectric Sensor Device Operation FB
MSOperating	BOOL	False	Operation monitoring start command
Threshold	DINT	–	Object Detection Threshold
AlarmLevel	DINT	–	Normal Amount of Light Received
FEnabled	BOOL	–	Enable
Alarm	BOOL	–	Alarm
Error	BOOL	–	Error
ErrorID	WORD	–	Error code
ErrorIDEx	DWORD	–	Expansion error code

● External Variables

Name	Data type	Constant	Comment
E001_No_01_Detection_Level_IN1	INT	-	Amount of light received by the photoelectric sensor

Ladder Diagram



Stopwatch

The Stopwatch function block outputs the time from when measurement starts until measurement ends.

Function block name	Name	FB/FUN	Graphic expression	ST expression
Stopwatch	Measure Cycle Time	FB	<p style="text-align: center;">Stopwatch_instance</p>	<pre>Stopwatch_instance(Start, Stop, ClearMeasuredTime, Busy, MeasuredTime);</pre>

Function Block and Function Information

Item	Description
Library file name	OmronLib_BC_DeviceMonitor_V1_0.slr
Namespace	OmronLib\BC_DeviceMonitor
Function block and function number	00022
Publish/Do not publish source code	Do not publish
Function block and function version	1.00

Variables

	Meaning	I/O	Description	Valid range	Unit	Default
Start	Start	Input	TRUE: Start measurement FALSE: Do not start measurement	TRUE or FALSE	-	FALSE
Stop	End	Input	TRUE: End measurement FALSE: Do not end measurement	TRUE or FALSE		FALSE
ClearMeasuredTime	Clear	Input	TRUE: Clear <i>MeasuredTime</i> (Measurement Result) FALSE: Do not clear <i>MeasuredTime</i> (Measurement Result)	TRUE or FALSE	-	FALSE
Busy	Executing	Output	TRUE: Execution processing is in progress. FALSE: Execution processing is not in progress.	TRUE or FALSE		FALSE
MeasuredTime	Measurement Result	Output	Time from when measurement starts until measurement ends	Depends on data type.	-	0

	Boolean	Bit strings				Integers							Real numbers		Times, durations, dates, and text strings						
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING	
Start	OK																				
Stop	OK																				
ClearMeasuredTime	OK																				
Busy	OK																				
MeasuredTime																OK					

Function

This function block outputs the time from when the value of *Start* (Start) changes to TRUE until the value *Stop* (End) changes to TRUE to *MeasuredTime* (Measurement Result).

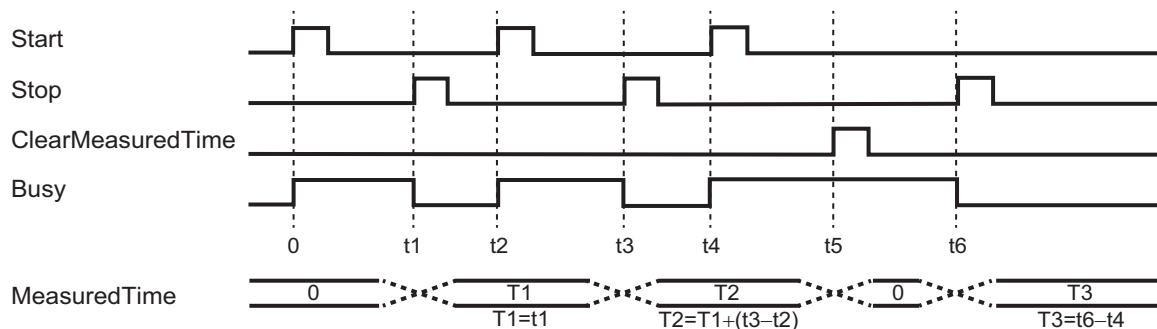
When *ClearMeasuredTime* (Clear) is changed to TRUE, the value of *MeasuredTime* (Measurement Result) is cleared.

The value of *MeasuredTime* (Measurement Result) is updated when the value of *Stop* (End) is changed to TRUE or the value of *ClearMeasuredTime* (Clear) is changed to TRUE. Otherwise, the previous value is retained.

Timing Charts

The following figures show the timing charts for the program part.

- When *Start* (Start) is changed to TRUE, *Busy* (Executing) changes to TRUE.
- When *Stop* (End) is changed to TRUE, *Busy* (Executing) changes to FALSE.



Precautions for Correct Use

- When the values of *Start* (Start) and *Stop* (End) are both changed to TRUE, *Stop* (End) is ignored and time measurement is started.
- *Start* (Start) is ignored when time measurement is in progress, even if it is changed to FALSE and then back to TRUE. The normal time until *Stop* (End) changes to TRUE is measured.
- Measurement ends normally once time measurement is in progress, even if *Stop* (End) and *ClearMeasuredTime* (Clear) are both changed to TRUE. In this case, the value of *MeasuredTime* (Measurement Result) is cleared.
- If *Start* and *ClearMeasuredTime* (Clear) are both changed to TRUE, the value of *MeasuredTime* (Measurement Result) is cleared.
- The measuring error of *MeasuredTime* (Measurement Result) is from -100 ns to (100 ns+1 task period).
- In the Sysmac Studio, the measured time is shown per 0.001 ms, but the timing accuracy is 1 ns.

Sample Programming

Description of Operation

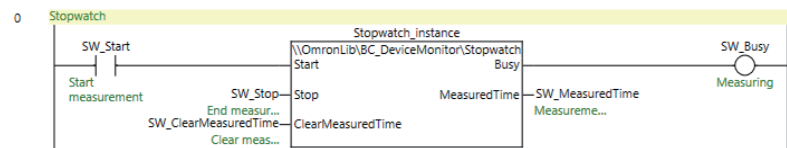
Measures the time from when measurement starts until measurement ends.

Variables

● Internal Variables

Name	Data type	Default	Comment
Stopwatch_instance	OmronLib\BC_DeviceMonitor\Stopwatch	-	Instance of Measure Cycle Time FB
SW_Start	BOOL	False	Start measurement
SW_Stop	BOOL	False	End measurement
SW_ClearMeasuredTime	BOOL	False	Clear measurement result
SW_Busy	BOOL	False	Measuring
SW_MeasuredTime	TIME	-	Measurement result

Ladder Diagram



DataRecorderPut

The DataRecorderPut function block adds data records to the data recorder.

Function block name	Name	FB/ FUN	Graphic expression	ST expression
DataRecorderPut	Add Data Record	FUN		Out := DataRecorderPut(Record, DataRecorder);

Function Block and Function Information

Item	Description
Library file name	OmronLib_BC_DeviceMonitor_V1_0.slr
Namespace	OmronLib\BC_DeviceMonitor
Function block and function number	00023
Publish/Do not publish source code	Publish
Function block and function version	1.00

Variables

	Meaning	I/O	Description	Valid range	Unit	Default
Record	Data Record	Input	A data record that is added to the data recorder	-	-	-
Out	Return Value	Output	Function execution results TRUE: Normal end FALSE: Error end	Depends on data type.	-	-
DataRecorder	Data Recorder	Input/output	Data recorder	-	-	-

	Boolean	Bit strings				Integers						Real numbers		Times, durations, dates, and text strings							
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING	
Record																					Refer to Function for details on the structure OmronLib\BC_DeviceMonitor\sDataRecord.
Out	OK																				
DataRecorder																					Refer to Function for details on the structure OmronLib\BC_DeviceMonitor\sDataRecorder.

Function

This function block adds *Record* (Data Record) to *DataRecorder* (Data Recorder).

DataRecorder Structure

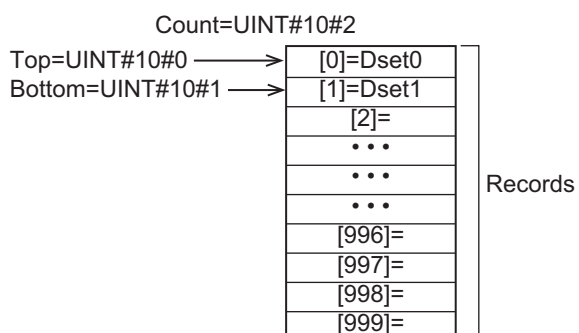
DataRecorder is a data recorder that can store 1000 data records.

The data type of *DataRecorder* is structure *sDataRecorder*. The specifications are as follows:

Name	Meaning	Description	Data type	Valid range	Unit	Default
sDataRecorder	Data Recorder	Structure to store data records	Omron-Lib\BC_Device-Monitor\sDataRecorder			
Records	Data Record Array	Array to store data records	ARRAY[0..999] OfsDataRecord	–	–	–
Top	First Data Record	Index of the first data record in the data recorder	UINT	0 to 999	–	–
Bottom	Last Data Record	Index of the last data record in the data recorder ^{*1}	UINT	0 to 999	–	–
Count	Number of Data Records	Number of data records that are stored in the data recorder	UINT	0 to 1000	–	–

*1. When *Count* = 0, the *Bottom* value is *UINT#10#0*.

The following shows the values of variables for when the number of array elements for *Count* = *UINT#10#2*. Data records that are stored in *Records[0]* are expressed as *Dset0*, and data records in *Records[1]* as *Dset1*.



Record Specifications

Record is a data record that is stored in the data recorder.

The data type of *Record* is structure OmronLib\BC_DeviceMonitor\sDataRecord. The specifications are as follows:

Name	Meaning	Description	Data type	Valid range	Unit	Default
sDataRecord	Data Record	A data record that is stored in the data recorder	Omron-Lib\BC_Device-Monitor\sDataRecord			
RecTime ^{*1}	Record Time	Time when a data record is added to the data recorder	DATE_AND_TIME	Depends on data type.	-	-
BitData	Bit Data	BOOL data	ARRAY[0..31] OF BOOL	Depends on data type.	-	-
IntData	INT Data	INT data	ARRAY[0..9] OF INT	Depends on data type.	-	-
DIntData	DINT Data	DINT data	ARRAY[0..9] OF DINT	Depends on data type.	-	-
RealData	REAL Data	REAL data	ARRAY[0..9] OF REAL	Depends on data type.	-	-

*1. The System Time when a data record is added to the data recorder is stored in *RecTime*.

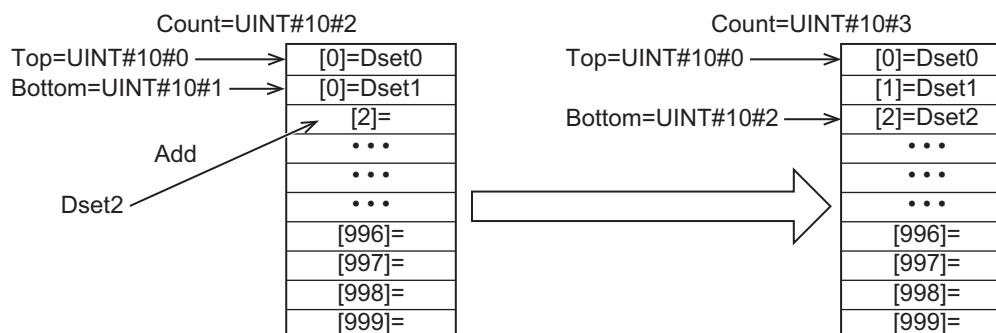
One data record can hold up to 32 BOOL data, 10 INT data, 10 DINT data, and 10 REAL data.

Adding Data Records

When a data record is added to the data recorder, the values of *DataRecorder* members are processed as follows.

Member of <i>DataRecorder</i>	Processing
Records	The value of <i>Record</i> is stored in <i>Records[Bottom+1]</i> .
Top	Does not change.
Bottom	Incremented.
Count	Incremented.

The following shows the values of variables for when a data record is added with the number of array elements for *Count* = UINT#10#2. Data records that are stored in *Records[0]* are expressed as Dset0, data records in *Records[1]* as Dset1, and the value of the data record added as Dset2.

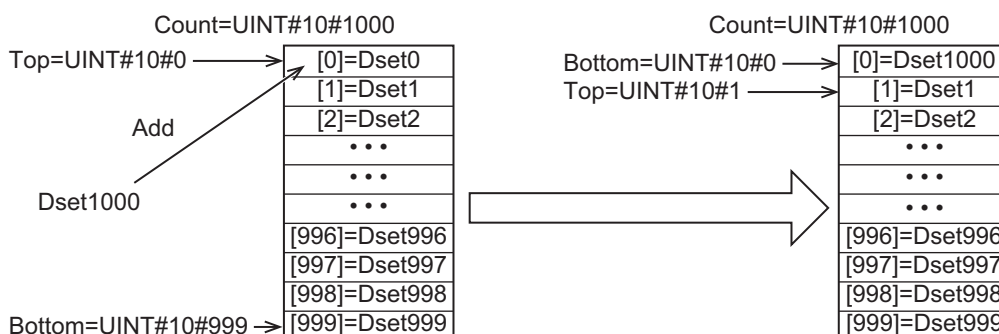


Adding Record When Count = UINT#10#1000

When a data record is added with *Count* = UINT#10#1000, the values of *DataRecorder* members are processed as follows.

Member of <i>DataRecorder</i>	Processing
Records	<ul style="list-style-type: none"> The value of <i>Records[0]</i> is discarded. The value of <i>Record</i> is stored in <i>Records[0]</i>.
Top	Changes to UINT#10#1.
Bottom	Changes to UINT#10#0.
Count	The number of array elements for <i>Records</i> remains.

The following shows the values of variables for when a data record is added with the number of array elements for *Count* = UINT#10#500. Data records that are stored in *Records* are expressed as Dset## and record that is added as Dset1000.



When a data record is added, the oldest data record is discarded and *Top* and *Bottom* are incremented. The value of *Count* does not change.

Sample Programming

Refer to the sample programming for *DataRecorderCSVWrite* on page 126.



Precautions for Correct Use

- Do not execute the *DataRecorderGet* function at the same time in order to add data correctly.
- When the power supply is turned OFF to the Controller, the content of the data recorder is discarded.

DataRecorderGet

The DataRecorderGet function block reads the oldest data record that is stored in the data recorder.

Function block name	Name	FB/FUN	Graphic expression	ST expression
DataRecorderGet	Get Data Record	FUN		Out := DataRecorderGet(DataRecorder, Record);

Function Block and Function Information

Item	Description
Library file name	OmronLib_BC_DeviceMonitor_V1_0.slr
Namespace	OmronLib\BC_DeviceMonitor
Function block and function number	00024
Publish/Do not publish source code	Publish
Function block and function version	1.00

Variables

	Meaning	I/O	Description	Valid range	Unit	Default
Out	Return Value	Output	Function execution results TRUE: Normal end FALSE: Error end	Depends on data type.	-	-
Record	Data Record		Data Record	-	-	-
DataRecorder	Data Recorder	Input/output	Data Recorder	-	-	-

	Boolean	Bit strings				Integers							Real numbers		Times, durations, dates, and text strings					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Out	OK																			
Record	Refer to Function for details on the structure OmronLib\BC_DeviceMonitor\sDataRecord.																			
DataRecorder	Refer to Function for details on the structure OmronLib\BC_DeviceMonitor\sDataRecorder.																			

Function

This function block reads the oldest record (Data Record) that is stored in *DataRecorder* (Data Recorder).

The structure of *Record* is the same as *Record* for the *DataRecorderPut* function. Refer to *DataRecorderPut* on page 120 for *DataRecorder* and *Record* specifications.

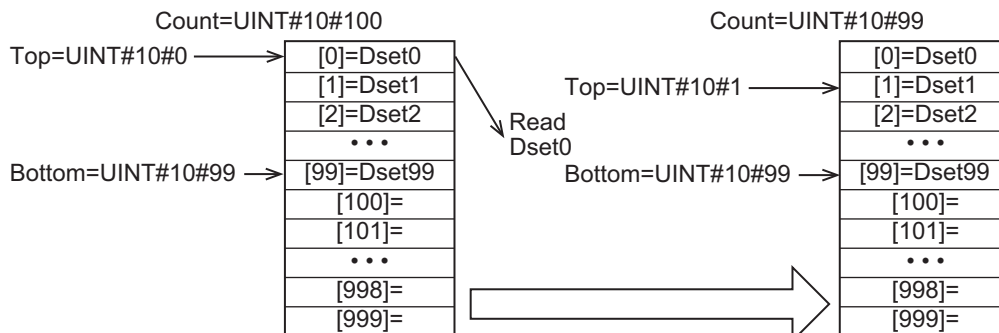
Reading Data Records

When a data record is read from the data recorder, the values of the *DataRecorder* members are processed as follows.

Member of <i>DataRecorder</i>	Processing
Records	Does not change.
Top	Incremented.*1
Bottom	Does not change.
Count	Decrementd.

*1. Does not change when *Count* = UINT#10#0 or *Count* = UINT#10#1.

The following shows the values of variables for when a data record is read with the number of array elements for *Count* = UINT#10#100. Data records that are stored in *Records* are expressed as Dset##.



If this function is executed when *Count*= UINT#10#0, in other words, when no data records are stored in *DataRecorder*, the *Out* (Return Value) value changes to FALSE and the *Record* value becomes indefinite.

Sample Programming

Refer to the sample programming for *DataRecorderCSVWrite* on page 126.



Precautions for Correct Use

- When no data records are stored in the data recorder, the return value changes to FALSE.
- When the power supply is turned OFF to the Controller, the content of the data recorder is discarded.
- Do not execute the *DataRecorderPut* function at the same time.

DataRecorderCSVWrite

The DataRecorderCSVWrite function block writes the data records that are stored in the data recorder to an SD Memory Card in CSV format.

Function block name	Name	FB/ FUN	Graphic expression	ST expression
DataRecorderCSVWrite	Write from Data Recorder to SD Memory Card	FB		<pre>DataRecorderCSVWrite_instance(Execute, DataRecorder, FileName, Done, Busy, Error, ErrorID ErrorIDEx);</pre>

Function Block and Function Information

Item	Description
Library file name	OmronLib_BC_DeviceMonitor_V1_0.slr
Namespace	OmronLib\BC_DeviceMonitor
Function block and function number	00025
Publish/Do not publish source code	Publish
Function block and function version	1.00

Compatible Models

Item	Name	Model numbers	Version
Device	SD Memory Card	HMC-SD□□□	-

Variables

	Meaning	I/O	Description	Valid range	Unit	Default
Execute	Execute	Input	TRUE: Executes the instruction. FALSE: Does not execute the instruction.	TRUE or FALSE	–	FALSE
DataRecorder	Data Recorder	Input/output	Data Recorder	–	–	–
FileName	File Name	Input	File name of CSV file to write	66 bytes max. (65 single-byte alphanumeric characters plus the final NULL character)	–	–
Done	Done	Output	TRUE: Normal end FALSE: Error end, execution in progress, or execution condition not met	TRUE or FALSE	–	–
Busy	Executing	Output	TRUE: Execution processing is in progress. FALSE: Execution processing is not in progress.	TRUE or FALSE	–	–
Error	Error	Output	TRUE: Error end FALSE: Normal end, execution in progress, or execution condition not met	TRUE or FALSE	–	–
ErrorID	Error Code	Output	This is an error code for an error end. The value is 16#0 for a normal end.	*1	–	–
ErrorIDEx	Expansion Error Code	Output	This is an expansion error code for an error end. The value is 16#0 for a normal end.	*1	–	–

*1. Refer to *Troubleshooting* on page 132 for details.

	Boolean	Bit strings				Integers							Real numbers		Times, durations, dates, and text strings					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	OK																			
DataRecorder	Refer to Function for details on the structure OmronLib\BC_DeviceMonitor'sDataRecorder.																			
FileName																				OK
Done	OK																			
Busy	OK																			
Error	OK																			
ErrorID			OK																	
ErrorIDEx				OK																

Function

When *Execute* (Execute) changes to TRUE, this function block writes all the data records that are stored in *DataRecorder* (Data 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, create *FileName* in the root of the SD Memory Card.

The structure of the record is the same as *Record* for the *DataRecorderPut* function. Refer to *DataRecorderPut* on page 120 for *DataRecorder* and *Record* specifications.

CSV File Format

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

'RecTime'	'BitData[0]'	...	'BitData[31]'	'IntData[0]'	...	'IntData[9]'	'DIntData[0]'	...	'DIntData[9]'	'RealData[0]'	...	'RealData[9]'
Records[Top].	Records[Top].		Records[Top].	Records[Top].		Records[Top].	Records[Top].		Records[Top].	Records[Top].		Records[Top].
RecTime	BitData[0]	...	BitData[31]	IntData[0]	...	IntData[9]	DIntData[0]	...	DIntData[9]	RealData[0]	...	RealData[9]
Records[Top+1].	Records[Top+1].		Records[Top+1].	Records[Top+1].		Records[Top+1].	Records[Top+1].		Records[Top+1].	Records[Top+1].		Data[Top+1].
RecTime	BitData[0]	...	BitData[31]	IntData[0]	...	IntData[9]	DIntData[0]	...	DIntData[9]	RealData[0]	...	RealData[9]
...
...
...
Records[Bottom-1].	Records[Bottom-1].		Records[Bottom-1].	Records[Bottom-1].		Records[Bottom-1].	Records[Bottom-1].		Records[Bottom-1].	Records[Bottom-1].		Records[Bottom-1].
RecTime	BitData[0]	...	BitData[31]	IntData[0]	...	IntData[9]	DIntData[0]	...	DIntData[9]	RealData[0]	...	RealData[9]
Records[Bottom].	Records[Bottom].		Records[Bottom].	Records[Bottom].		Records[Bottom].	Records[Bottom].		Records[Bottom].	Records[Bottom].		Records[Bottom].
RecTime	BitData[0]	...	BitData[31]	IntData[0]	...	IntData[9]	DIntData[0]	...	DIntData[9]	RealData[0]	...	RealData[9]

RecTime is converted to a text string and written with the *DtToString* instruction. Refer to the instructions reference manual for details on the *DtToString* instruction.

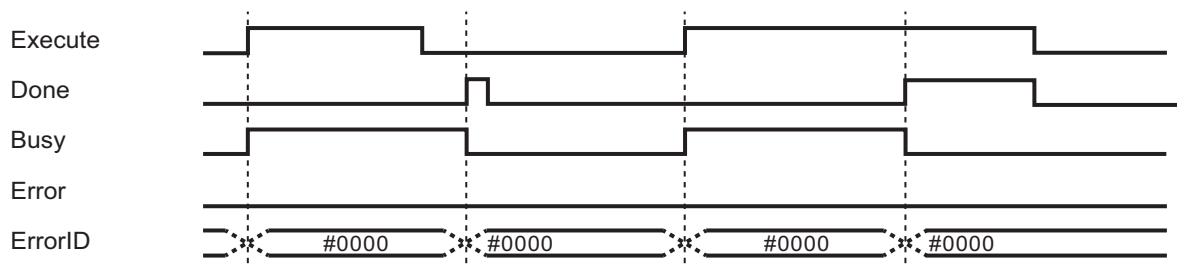
RealData is converted to a text string and written with the *RealToFormatString* instruction. For the number of digits, the overall is set to four and the fractional part is to three. Refer to the instructions reference manual for details on the *RealToFormatString* instruction.

Timing Charts

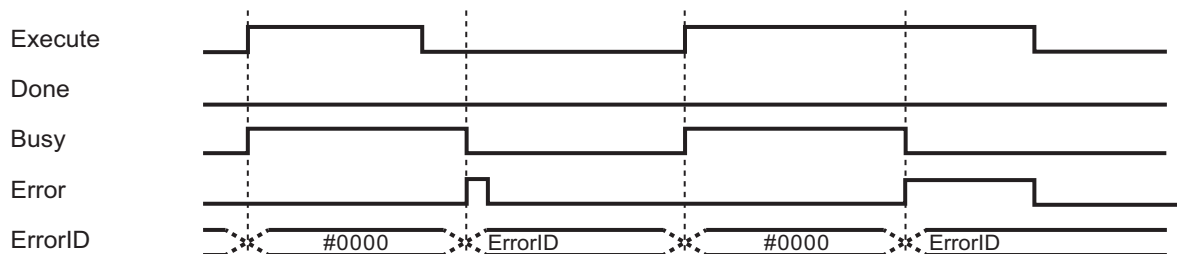
The following figures show the timing charts for the program part.

- *Busy* (Executing) changes to TRUE when *Execute* (Execute) changes to TRUE.
- *Done* changes to TRUE when the data output operation is completed.
- 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* (Execute) changes to FALSE before execution of the function block is ended, *Done* (Done) and *Error* (Error) are TRUE only for one task period after execution of the function block is ended.
- If *Execute* (Execute) remains TRUE even after execution of the function block is ended, the output values of *Done* (Done) and *Error* (Error) are retained.

● Timing Chart for Normal End



● Timing Chart for Error End



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.
- The number of array elements for data records in the data recorder is defined by the user.
- 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.



Precautions for Correct Use

- Do not execute the same instance while an instance is being executed.
 - When you execute the DataRecorderCSVWrite function block, always stop the DataRecorderPut and DataRecorderGet functions beforehand. If you execute the DataRecorderCSVWrite 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 content of the data recorder is 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: Can be used. FALSE: 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: An error occurred. FALSE: No error occurred.
_Card1Access	SD Memory Card Access Flag	BOOL	This flag indicates if the SD Memory Card is currently being accessed. TRUE: Being accessed. FALSE: 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

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.

Error code	Expansion error code	Status	Description	Correction
16#140E	16#00000000	SD Memory Card Access	The SD Memory Card is broken.	Replace the SD Memory Card.
	16#00000000	Failed	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.
16#3C22	16#00000001	First Data Position Specification Error	The value of <i>DataRecorder.Top</i> is outside of the array range.	Set the value of <i>DataRecorder.Top</i> within the array range.
16#3C22	16#00000002	Last Data Position Specification Error	The value of <i>DataRecorder.Bottom</i> is outside of the array range.	Set the value of <i>DataRecorder.Bottom</i> within the array range.
16#3C22	16#00000003	No Data Stored in Data Recorder	There are no data records in <i>DataRecorder</i> . (<i>DataRecorder.Count</i> = UINT#10#0)	Check to see if one or more data records are stored in <i>DataRecorder</i> .
16#3C22	16#00000004	Data Recorder Storage Information Error	There are inconsistencies in <i>DataRecorder.Top</i> , <i>DataRecorder.Bottom</i> , and <i>DataRecorder.Count</i> values.	Check the <i>DataRecorder.Top</i> , <i>DataRecorder.Bottom</i> , and <i>DataRecorder.Count</i> values.

Sample Programming

Description of Operation

This sample programming operates as follows.

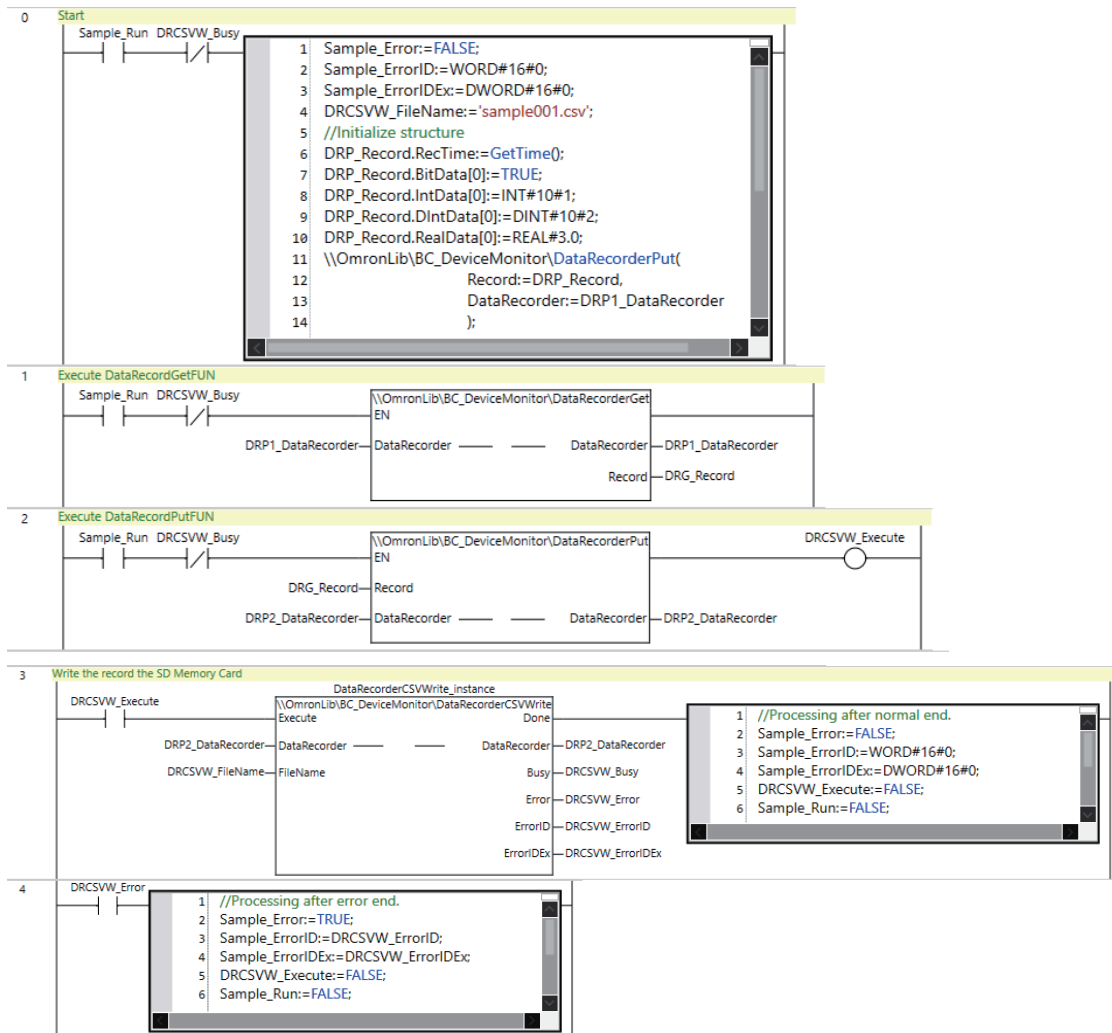
- 1** One data record is stored in the DRP1_DataRecorder structure.
- 2** With the DataRecorderGet function, one data record is read from the DRP1_DataRecorder structure.
- 3** With the DataRecorderPut function, the read data record is stored in the DRP2_DataRecorder structure.
- 4** With the DataRecorderCSVWrite function block, data records in the DRP2_DataRecorder structure are written to the SD Memory Card.

Variables

● Internal Variables

Name	Data type	Default	Comment
DataRecorderCSVWrite_instance	OmronLib\BC_DeviceMonitor\DataRecorderCSVWrite	-	-
Sample_Run	BOOL	-	-
Sample_Error	BOOL	-	-
Sample_ErrorID	WORD	-	-
Sample_ErrorIDEx	DWORD	-	-
DRP_Record	OmronLib\BC_DeviceMonitor\DataRecord	-	-
DRP1_DataRecorder	OmronLib\BC_DeviceMonitor\DataRecorder	-	-
DRP2_DataRecorder	OmronLib\BC_DeviceMonitor\DataRecorder	-	-
DRG_Record	OmronLib\BC_DeviceMonitor\DataRecord	-	-
DRCSVW_Execute	BOOL	-	-
DRCSVW_Busy	BOOL	-	-
DRCSVW_Done	BOOL	-	-
DRCSVW_Error	BOOL	-	-
DRCSVW_ErrorID	WORD	-	-
DRCSVW_ErrorIDEx	DWORD	-	-
DRCSVW_FileName	STRING[66]	-	-

Ladder Diagram



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 programming actual applications, also program safety circuits, device interlocks, I/O with other devices, and other control procedures.
- Create a user program that will produce the intended device operation.
- Check the user program for proper execution before you use it for actual operation.

AxisRecorderPut

The AxisRecorderPut function block adds axis records to the axis recorder.

Function block name	Name	FB/FUN	Graphic expression	ST expression
AxisRecorderPut	Add Axis Record	FUN		<pre>Out := AxisRecorderPut(Record, AxisRecorder);</pre>

Function Block and Function Information

Item	Description
Library file name	OmronLib_BC_DeviceMonitor_V1_0.slr
Namespace	OmronLib\BC_DeviceMonitor
Function block and function number	00033
Publish/Do not publish source code	Publish
Function block and function version	1.00

Variables

	Meaning	I/O	Description	Valid range	Unit	Default
Record	Axis Record	Input	An axis record that is added to the axis recorder	-	-	-
Out	Return Value	Output	Function execution results TRUE: Normal end FALSE: Error end	Depends on data type.	-	-
AxisRecorder	Axis Recorder	Input/output	Axis recorder	-	-	-

	Bool-ean	Bit strings				Integers							Real num-bers		Times, durations, dates, and text strings					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Record	Refer to Function for details on the structure OmronLib\BC_DeviceMonitor\AxisRecord.																			
Out	OK																			
AxisRecorder	Refer to Function for details on the structure OmronLib\BC_DeviceMonitor\AxisRecorder.																			

Function

This function block adds *Record* (Axis Record) to *AxisRecorder* (Axis Recorder).

AxisRecorder Structure

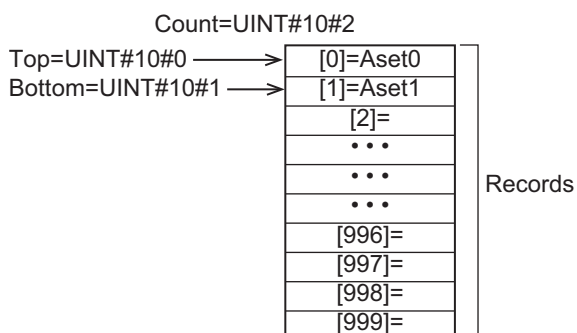
AxisRecorder is an axis recorder that can store 1000 axis records.

The data type of *AxisRecorder* is structure `OmronLib\BC_DeviceMonitor\AxisRecorder`. The specifications are as follows:

Name	Meaning	Description	Data type	Valid range	Unit	Default
AxisRecorder	Axis Recorder	Structure to store axis records	OmronLib\BC_DeviceMonitor\AxisRecorder			
Records	Axis Record Array	Array to store axis records	ARRAY[0..999] OF sAxisRecord	–	–	–
Top	First Axis Record	Index of the first axis record in the axis recorder	UINT	0 to 999	–	–
Bottom	Last Axis Record	Index of the last axis record in the axis recorder *1	UINT	0 to 999	–	–
Count	Number of Axis Records	Number of axis records that are stored in the axis recorder	UINT	0 to 1000	–	–

*1. When *Count*= 0, the *Bottom* value is `UINT#10#0`.

The following shows the values of variables for when *Count* = `UINT#10#2`. Axis records that are stored in *Records[0]* are expressed as *Aset0*, and axis records in *Records[1]* as *Aset1*.



Record Specifications

Record is an axis record that is stored in the axis recorder.

The data type of *Record* is structure `OmronLib\BC_DeviceMonitor\AxisRecord`. The specifications are as follows:

Name	Meaning	Description	Data type	Valid range	Unit	Default
AxisRecord	Axis Record	An axis record that is stored in the axis recorder	OmronLib\BC_DeviceMonitor\AxisRecord			
RecTime*1	Record Time	Time when an axis record is added to the axis recorder	DATE_AND_TIME	Depends on data type.	-	-
CmdPos	Command Current Position	Command current position	LREAL	Depends on data type.	-	-
CmdVel	Command Current Velocity	Command current velocity	LREAL	Depends on data type.	-	-
ActPos	Actual Current Position	Actual current position	LREAL	Depends on data type.	-	-
ActVel	Actual Current Velocity	Actual current velocity	LREAL	Depends on data type.	-	-
ActTrq	Actual Current Torque	Actual current torque	LREAL	Depends on data type.	-	-

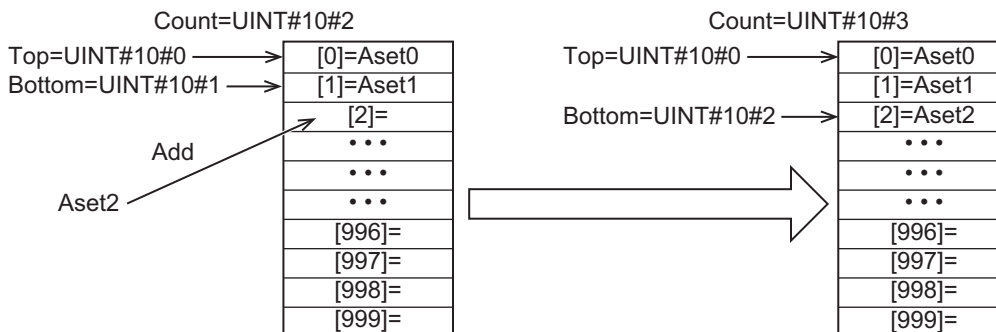
*1. The System Time when an axis record is added to the axis recorder is stored in *RecTime*.

Adding Axis Records

When an axis record is added to the axis recorder, the values of *AxisRecorder* members are processed as follows.

Member of <i>AxisRecorder</i>	Processing
Records	The value of <i>Record</i> is stored in <i>Records[Bottom+1]</i>
Top	Does not change.
Bottom	Incremented.
Count	Incremented.

The following shows the values of variables for when an axis record is added with *Count* = `UINT#10#2`. Axis records that are stored in *Records[0]* are expressed as *Aset0*, axis records in *Records[1]* as *Aset1*, and the value of the axis record added as *Aset2*.

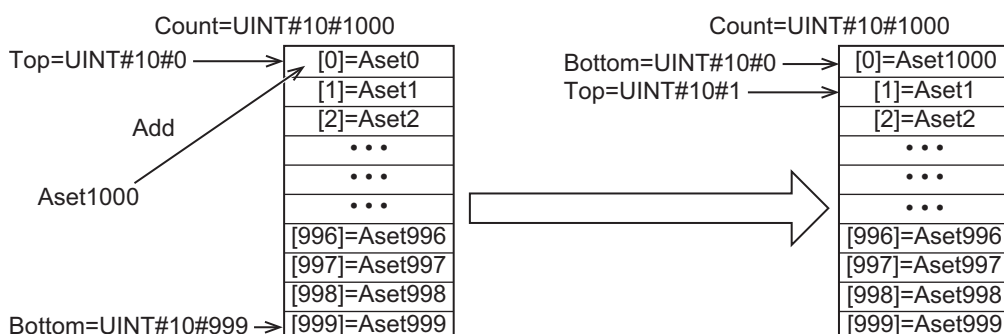


Adding Axis Record When Count = UINT#10#1000

When an axis record is added with *Count* = UINT#10#1000, the values of *AxisRecorder* members are processed as follows.

Member of <i>AxisRecorder</i>	Processing
Records	<ul style="list-style-type: none"> The value of <i>Records[0]</i> is discarded. The value of <i>Record</i> is stored in <i>Records[0]</i>.
Top	Changes to UINT#10#1.
Bottom	Changes to UINT#10#0.
Count	The number of array elements for <i>Records</i> remains.

The following shows the values of variables for when an axis record is added with *Count* = UINT#10#1000. Axis records that are stored in *Records* are expressed as Aset## and axis record that is added as Aset1000.



If this function is executed when *Count* = UINT#10#0, in other words, when no axis records are stored in *AxisRecorder*, the *Out* (Return Value) value changes to FALSE and the *AxisRecorder* value becomes indefinite.

Sample Programming

The use of this function is equivalent to *DataRecorderPut* function. Therefore, for the sample programming of this function, refer to the sample programming for *BitRecorderToGraph* on page 156.



Precautions for Correct Use

- Do not execute the *AxisRecorderGet* function at the same time in order to add data correctly.
- When the power supply is turned OFF to the Controller, the content of the axis recorder is discarded.

AxisRecorderGet

The AxisRecorderGet function block reads the oldest axis record that is stored in the axis recorder.

Function block name	Name	FB/FUN	Graphic expression	ST expression
AxisRecorderGet	Get Axis Record	FUN		Out := AxisRecorderGet(AxisRecorder, Record);

Function Block and Function Information

Item	Description
Library file name	OmronLib_BC_DeviceMonitor_V1_0.slr
Namespace	OmronLib\BC_DeviceMonitor
Function block and function number	00034
Publish/Do not publish source code	Publish
Function block and function version	1.00

Variables

	Meaning	I/O	Description	Valid range	Unit	Default
Out	Return Value	Output	Function execution results TRUE: Normal end FALSE: Error end	Depends on data type.	-	-
Record	Axis Record		Axis Record	-	-	-
AxisRecorder	Axis Recorder	Input/output	Axis recorder	-	-	-

	Boolean	Bit strings				Integers						Real numbers		Times, durations, dates, and text strings						
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Out	OK																			
Record	Refer to Function for details on the structure OmronLib\BC_DeviceMonitor\AxisRecord.																			
AxisRecorder	Refer to Function for details on the structure OmronLib\BC_DeviceMonitor\AxisRecorder.																			

Function

This function block reads the oldest *Record* (Axis Record) that is stored in *AxisRecorder* (Axis Recorder).

The structure of *Record* is the same as *Record* for the *AxisRecorderPut* function. Refer to *AxisRecorderPut* on page 136 for *AxisRecorder* and *Record* specifications.

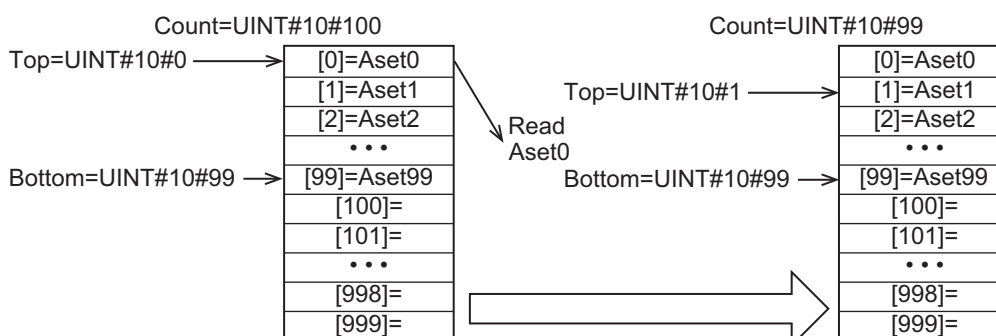
Reading Axis Records

When an axis record is read from the axis recorder, the values of the *AxisRecorder* members are processed as follows.

Member of <i>AxisRecorder</i>	Processing
Records	Does not change.
Top	Incremented.*1
Bottom	Does not change.
Count	Decrementd.

*1. Does not change when $Count = UINT\#10\#0$ or $Count = UINT\#10\#1$.

The following shows the values of variables for when an axis record is read with $Count = UINT\#10\#100$. Axis records that are stored in *Records* are expressed as *Aset##*.



If this function is executed when $Count = UINT\#10\#0$, in other words, when no axis records are stored in *AxisRecorder*, the *Out* (Return Value) value changes to FALSE and the *Record* value becomes indefinite.

Sample Programming

The use of this function is equivalent to *DataRecorderGet* function. Therefore, for the sample programming of this function, refer to the sample programming for *DataRecorderCSVWrite* on page 126.



Precautions for Correct Use

- When no axis records are stored in the axis recorder, the return value changes to FALSE.
- When the power supply is turned OFF to the Controller, the content of the axis recorder is discarded.
- Do not execute the *AxisRecorderPut* function at the same time.

AxisRecorderCSVWrite

The AxisRecorderCSVWrite function block writes the axis records that are stored in the axis recorder to an SD Memory Card in CSV format.

Function block name	Name	FB/ FUN	Graphic expression	ST expression
Axis Recorder CSVWrite	Write Axis Record to SD Memory Card	FB		<pre>AxisRecorderCSVWrite_instance(Execute, AxisRecorder, FileName, Done, Busy, Error, ErrorID, ErrorIDEx);</pre>

Function Block and Function Information

Item	Description
Library file name	OmronLib_BC_DeviceMonitor_V1_0.slr
Namespace	OmronLib\BC_DeviceMonitor
Function block and function number	00035
Publish/Do not publish source code	Publish
Function block and function version	1.00

Compatible Models

Item	Name	Model numbers	Version
Device	SD Memory Card	HMC-SD□□□	–

Variables

	Meaning	I/O	Description	Valid range	Unit	Default
Execute	Execute	Input	TRUE: Executes the instruction. FALSE: Does not execute the instruction.	TRUE or FALSE	–	FALSE
AxisRecorder	Axis Recorder	Input/output	Axis recorder	–	–	–
FileName	File Name	Input	File name of CSV file to write	66 bytes max. (65 single-byte alphanumeric characters plus the final NULL character)	–	–
Done	Done	Output	TRUE: Normal end FALSE: Error end, execution in progress, or execution condition not met	TRUE or FALSE	–	–
Busy	Executing	Output	TRUE: Execution processing is in progress. FALSE: Execution processing is not in progress.	TRUE or FALSE	–	–
Error	Error	Output	TRUE: Error end FALSE: Normal end, execution in progress, or execution condition not met	TRUE or FALSE	–	–
ErrorID	Error Code	Output	This is an error code for an error end. The value is 16#0 for a normal end.	*1	–	–
ErrorIDEx	Expansion Error Code	Output	This is an expansion error code for an error end. The value is 16#0 for a normal end.	*1	–	–

*1. Refer to *Troubleshooting* on page 148 for details.

	Boolean	Bit strings					Integers							Real numbers		Times, durations, dates, and text strings				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Execute	OK																			
AxisRecorder	Refer to Function for details on the structure OmronLib\BC_DeviceMonitor\AxisRecorder.																			
FileName																				OK
Done	OK																			
Busy	OK																			
Error	OK																			
ErrorID			OK																	
ErrorIDEx				OK																

Function

When *Execute* (Execute) changes to TRUE, this function block writes all the axis records that are stored in *AxisRecorder* (Axis 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, create *FileName* in the root of the SD Memory Card.

The structure of the record is the same as *Record* for the *AxisRecorderPut* function. Refer to *AxisRecorderPut* on page 136 for *AxisRecorder* and *Record* specifications.

CSV File Format

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

'RecTime'	'CmdPos'	'CmdVel'	'ActPos'	'ActVel'	'ActTrq'
Records[Top].	Records[Top].	Records[Top].	Records[Top].	Records[Top].	Records[Top].
RecTime	CmdPos	CmdVel	ActPos	AvtVel	ActTrq
Records[Top+1].	Records[Top+1].	Records[Top+1].	Records[Top+1].	Records[Top+1].	Records[Top+1].
RecTime	CmdPos	CmdVel	ActPos	AvtVel	ActTrq
...
...
Records[Bottom-1].	Records[Bottom-1].	Records[Bottom-1].	Records[Bottom-1].	Records[Bottom-1].	Records[Bottom-1].
RecTime	CmdPos	CmdVel	ActPos	AvtVel	ActTrq
Records[Bottom].	Records[Bottom].	Records[Bottom].	Records[Bottom].	Records[Bottom].	Records[Bottom].
RecTime	CmdPos	CmdVel	ActPos	AvtVel	ActTrq

RecTime is converted to a text string and written with the *DtToString* instruction. Refer to the instructions reference manual for details on the *DtToString* instruction.

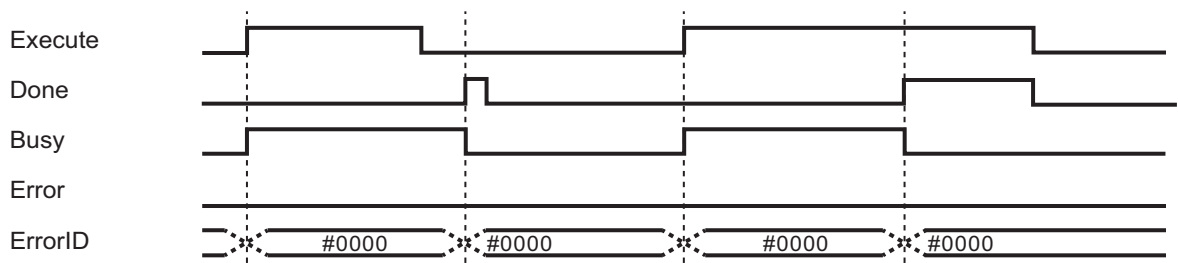
CmdPos, *CmdVel*, *ActPos*, *ActVel* and *ActTrq* are converted to text strings and written with the *LrealToFormatString* instruction. For the number of digits, the overall is set to eight and the fractional part is to six. Refer to the instructions reference manual for details on the *LrealToFormatString* instruction.

Timing Charts

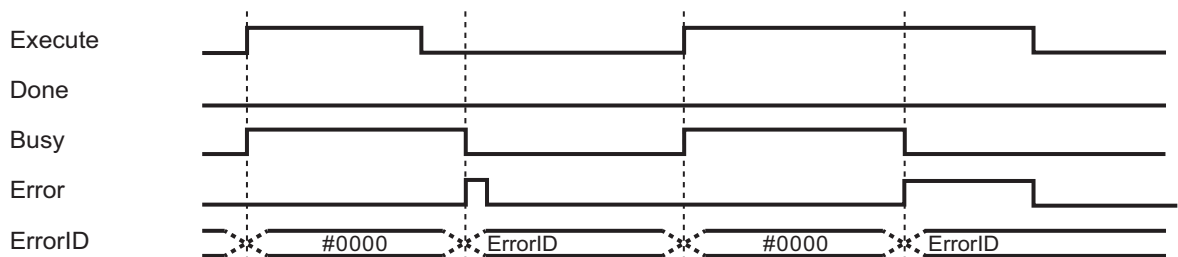
The following figures show the timing charts for the program part.

- *Busy* (Executing) changes to TRUE when *Execute* (Execute) changes to TRUE.
- *Done* changes to TRUE when the data output operation is completed.
- 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* (Execute) changes to FALSE before execution of the function block is ended, *Done* (Done) and *Error* (Error) are TRUE only for one task period after execution of the function block is ended.
- If *Execute* (Execute) remains TRUE even after execution of the function block is ended, the output values of *Done* (Done) and *Error* (Error) are retained.

● Timing Chart for Normal End



● Timing Chart for Error End



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.
- The number of array elements for records in the axis recorder is defined by the user.
- 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.

**Precautions for Correct Use**

- Do not execute the same instance while an instance is being executed.
 - When you execute the AxisRecorderCSVWrite function block, always stop the AxisRecorderPut and AxisRecorderGet functions beforehand. If you execute the AxisRecorderCSVWrite 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 content of the axis recorder is 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: Can be used. FALSE: 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: An error occurred. FALSE: No error occurred.
_Card1Access	SD Memory Card Access Flag	BOOL	This flag indicates if the SD Memory Card is currently being accessed. TRUE: Being accessed. FALSE: 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

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.

Error Code	Expansion error code	Status	Description	Correction
16#140E	16#00000000	SD Memory Card Access	The SD Memory Card is broken.	Replace the SD Memory Card.
	16#00000000	Failed	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.
16#3C25	16#00000001	First Data Position Specification Error	The value of <i>AxisRecorder.Top</i> is outside of the array range.	Set the value of <i>AxisRecorder.Top</i> within the array range.
	16#00000002	Last Data Position Specification Error	The value of <i>AxisRecorder.Bottom</i> is outside of the array range.	Set the value of <i>AxisRecorder.Bottom</i> within the array range.
	16#00000003	No Data Stored in Axis Recorder	There are no axis records in <i>AxisRecorder</i> . (<i>AxisRecorder.Count</i> = UINT#10#0)	Check to see if one or more axis records are stored in <i>AxisRecorder</i> .
	16#00000004	Axis Recorder Storage Information Error	There are inconsistencies in <i>AxisRecorder.Top</i> , <i>AxisRecorder.Bottom</i> , and <i>AxisRecorder.Count</i> values.	Check the <i>AxisRecorder.Top</i> , <i>AxisRecorder.Bottom</i> , and <i>AxisRecorder.Count</i> values.

Sample Programming

The use of this function is equivalent to *DataRecorderCSVWrite* function block. Therefore, for the sample programming of this function, refer to the sample programming for *DataRecorderCSVWrite* on page 126.

BitRecorderPut

The BitRecorderPut function block adds bit records to the bit recorder.

Function name	Name	FB/ FUN	Graphic expression	ST expression
BitRecorderPut	Add Bit Record	FUN		<pre>Out := BitRecorderPut(Record, BitRecorder);</pre>

Function Block and Function Information

Item	Description
Library file name	OmronLib_BC_DeviceMonitor_V1_0.slr
Namespace	OmronLib\BC_DeviceMonitor
Function block and function number	00036
Publish/Do not publish source code	Publish
Function block and function version	1.00

Variables

	Meaning	I/O	Description	Valid range	Unit	Default
Record	Bit Record	Input	A bit record that is added to the bit recorder	-	-	-
Out	Return Value	Output	Function execution results TRUE: Normal end FALSE: Error end	Depends on data type.	-	-
BitRecorder	Bit Recorder	Input/out-put	Bit recorder	-	-	-

	Bool-ean	Bit strings				Integers							Real num-bers		Times, durations, dates, and text strings					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Record	Refer to Function for details on the structure OmronLib\BC_DeviceMonitor\BitRecord.																			
Out	OK																			
BitRecorder	Refer to Function for details on the structure OmronLib\BC_DeviceMonitor\BitRecorder.																			

Function

This function block adds *Record* (Bit Record) to *BitRecorder* (Bit Recorder).

BitRecorder Structure

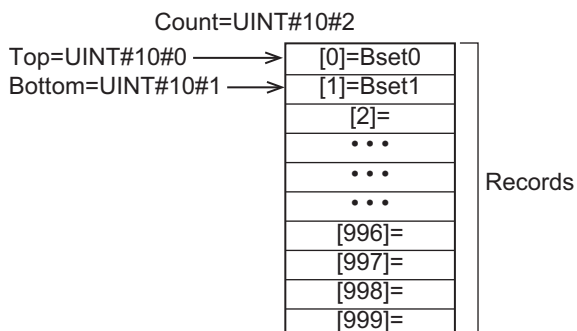
BitRecorder is a bit recorder that can store 1000 bit records.

The data type of *BitRecorder* is structure OmronLib\BC_DeviceMonitor\BitRecorder. The specifications are as follows:

Name	Meaning	Description	Data type	Valid range	Unit	Default
BitRecorder	Bit Recorder	Structure to store bit records	OmronLib\BC_DeviceMonitor\BitRecorder			
Records	Bit Record Array	Array to store bit records	ARRAY[0..999] OF sBitRecord	–	–	–
Top	First Bit Record	Index of the first bit record in the bit recorder	UINT	0 to 999	–	–
Bottom	Last Bit Record	Index of the last bit record in the bit recorder ^{*1}	UINT	0 to 999	–	–
Count	Number of Bit Records	Number of bit records that are stored in the bit recorder	UINT	0 to 1000	–	–

*1. When *Count*= 0, the *Bottom* value is UINT#10#0.

The following shows the values of variables for when *Count* = UINT#10#2. Bit records that are stored in *Records[0]* are expressed as Bset0, and records in *Records[1]* as Bset1.



Record Specifications

Record is a bit record that is stored in the bit recorder.

The data type of *Record* is structure OmronLib\BC_DeviceMonitor\BitRecord. The specifications are as follows:

Name	Meaning	Description	Data type	Valid range	Unit	Default
BitRecord	Bit Record	A bit record that is stored in the bit recorder	OmronLib\BC_DeviceMonitor\BitRecord			
RecTime*1	Record Time	Time when a bit record is added to the bit recorder	DATE_AND_TIME	Depends on data type.	-	-
BitData	Bit Data	BOOL data	ARRAY[0..31] OF BOOL	Depends on data type.	-	-

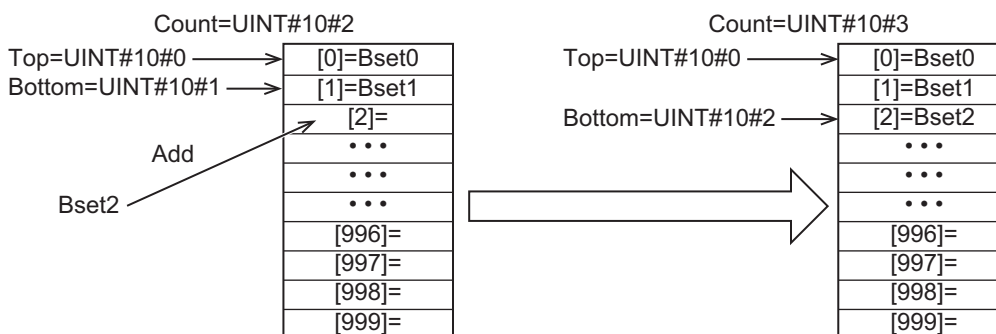
*1. The System Time when a bit record is added to the bit recorder is stored in *RecTime*.

Adding Bit Records

When a bit record is added to the bit recorder, the values of *BitRecorder* members are processed as follows.

Member of <i>BitRecorder</i>	Processing
Records	The value of <i>Record</i> is stored in <i>Records[Bottom+1]</i>
Top	Does not change.
Bottom	Incremented.
Count	Incremented.

The following shows the values of variables for when a bit record is added with *Count* = UINT#10#2. Bit records that are stored in *Records[0]* are expressed as Bset0, bit records in *Records[1]* as Bset1, and the value of the bit record added as Bset2.

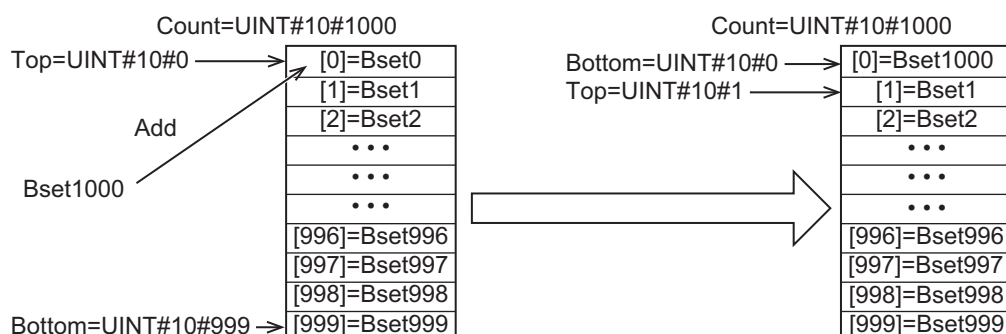


Adding Bit Record When Count = UINT#10#1000

When a bit record is added with *Count* = UINT#10#1000, the values of *BitRecorder* members are processed as follows.

Member of <i>BitRecorder</i>	Processing
Records	<ul style="list-style-type: none"> The value of <i>Records[0]</i> is discarded. The value of <i>Record</i> is stored in <i>Records[0]</i>.
Top	Changes to UINT#10#1.
Bottom	Changes to UINT#10#0.
Count	The number of array elements for <i>Records</i> remains.

The following shows the values of variables for when a bit record is added with *Count* = UINT#10#1000. Bit records that are stored in *Records* are expressed as Bset## and bit record that is added as Bset1000..



When a bit record is added, the oldest bit record is discarded and *Top* and *Bottom* are incremented. The value of *Count* does not change.

Sample Programming

Refer to the sample programming for *BitRecorderToGraph* on page 156.



Precautions for Correct Use

- Do not execute the *BitRecorderGet* function at the same time in order to add data correctly.
- When the power supply is turned OFF to the Controller, the content of the bit recorder is discarded.

BitRecorderGet

The BitRecorderGet function block reads the oldest bit record that is stored in the bit recorder.

Function name	Name	FB/ FUN	Graphic expression	ST expression
BitRecorderGet	Get Bit Record	FUN		<pre>Out := BitRecorderGet(BitRecorder, Record);</pre>

Function Block and Function Information

Item	Description
Library file name	OmronLib_BC_DeviceMonitor_V1_0.slr
Namespace	OmronLib\BC_DeviceMonitor
Function block and function number	00037
Publish/Do not publish source code	Publish
Function block and function version	1.00

Variables

	Meaning	I/O	Description	Valid range	Unit	Default
Out	Return Value	Output	Function execution results TRUE: Normal end FALSE: Error end	Depends on data type.	-	-
Record	Bit Record		Bit record			
BitRecorder	Bit Recorder	Input/output	Bit recorder	-	-	-

	Boolean	Bit strings				Integers						Real numbers		Times, durations, dates, and text strings						
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Out	OK																			
Record	Refer to Function for details on the structure OmronLib\BC_DeviceMonitor\BitRecord.																			
BitRecorder	Refer to Function for details on the structure OmronLib\BC_DeviceMonitor\BitRecorder.																			

Function

This function block reads the oldest *Record* (Bit Record) that is stored in *BitRecorder* (Bit Recorder).

The structure of *Record* is the same as *Record* for the *BitRecorderPut* function. Refer to *BitRecorderPut* on page 150 for *BitRecorder* and *Record* specifications.

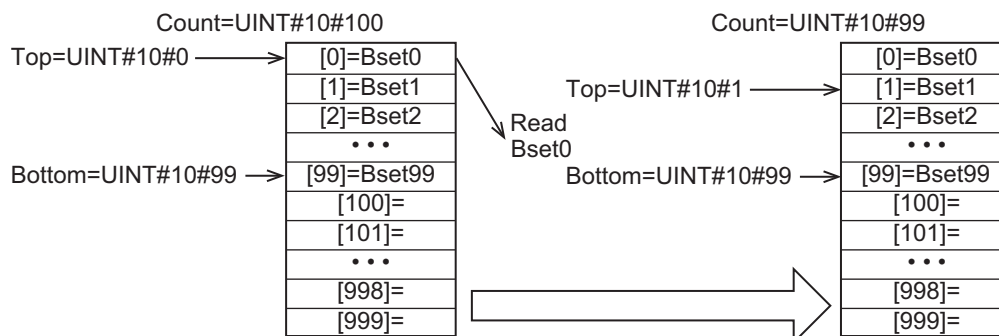
Reading Bit Records

When a bit record is read from the bit recorder, the values of *BitRecorder* members are processed as follows.

Member of <i>BitRecorder</i>	Processing
Records	Does not change.
Top	Incremented.* ¹
Bottom	Does not change.
Count	Decrementd.

*1. Does not change when $Count = UINT\#10\#0$ or $Count = UINT\#10\#1$.

The following shows the values of variables for when a bit record is read with $Count = UINT\#10\#100$. Bit records that are stored in *Records* are expressed as *Bset##*.



If this function is executed when $Count = UINT\#10\#0$, in other words, when no bit records are stored in *BitRecorder*, the *Out* (Return Value) value changes to FALSE and the *Record* value becomes indefinite.

Sample Programming

The use of this function is equivalent to *DataRecorderGet* function. Therefore, for the sample programming of this function, refer to the sample programming for *DataRecorderCSVWrite* on page 126.



Precautions for Correct Use

- When no bit records are stored in the bit recorder, the return value changes to FALSE.
- When the power supply is turned OFF to the Controller, the content of the bit recorder is discarded.
- Do not execute the *BitRecorderPut* function at the same time.

BitRecorderToGraph

The BitRecorderToGraph function converts bit records that are stored in the bit recorder to the data format that is suitable for time chart displays that use the broken-line graph function of NS-series PT.

Function name	Name	FB/ FUN	Graphic expression	ST expression
BitRecorderToGraph	Display Bit Record	FUN		<pre>Out := BitRecorderToGraph(Offset, Magnification, DivisionNum, BitRecorder, GraphBitRecorder);</pre>

Function Block and Function Information

Item	Description
Library file name	OmronLib_BC_DeviceMonitor_V1_0.slr
Namespace	OmronLib\BC_DeviceMonitor
Function block and function number	00038
Publish/Do not publish source code	Publish
Function block and function version	1.00

Compatible Models

Item	Name	Model numbers	Version
Device	NS-series PT	NS□-□□□□-V2	<ul style="list-style-type: none"> When the CPU Unit is NJ501-□□□□, version 8.5 or later When the CPU Unit is NJ301-□□□□ or NJ101-□□□□, version 8.61 or later When the CPU Unit is NX701-□□□□, version 8.9 or later

Variables

	Meaning	I/O	Description	Valid range	Unit	Default
Offset	Offset	Input	First position of the bit data that is displayed on the graph	0 to 999	–	0
Magnification	Magnification Ratio		Magnification ratio for graph display	1 to 20	–	1
DivisionNum	Number of Divisions		Number of scale divisions for graph display	1 to 20	–	1
Out	Return Value	Output	Function execution results TRUE: Normal end FALSE: Error end	Depends on data type.	–	–
BitRecorder	Bit Recorder	Input/output	Bit recorder	–	–	–
GraphBit Recorder	Graph Bit Recorder		Graph bit recorder	–	–	–

	Boolean	Bit strings				Integers							Real numbers		Times, durations, dates, and text strings					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Offset							OK													
Magnification							OK													
DivisionNum							OK													
Out	OK																			
BitRecorder	Refer to Function for details on the structure OmronLib\BC_DeviceMonitor\BitRecorder.																			
GraphBit Recorder	Refer to Function for details on the structure OmronLib\BC_DeviceMonitor\GraphBitRecorder.																			

Function

This function converts bit records that are stored in *BitRecorder* (Bit Recorder) to the data format that is suitable for time chart displays that use the broken-line graph function of NS-series PT and stores in *GraphBitRecorder* (Graph Bit Recorder).

The bit data for data format conversion are the four sequences of the structure `OmronLib\BC_DeviceMonitor\sBitRecorder.OmronLib\BC_DeviceMonitor\sBitRecord[n].BitData[i]` (n: 0-999, i: 0-3).

The structure of the bit record is the same as *Record* for the *BitRecorderPut* function. Refer to *BitRecorderPut* on page 150 for the *BitRecorder* and bit record specifications.

GraphBitRecorder Structure

GraphBitRecorder can store 250 bit records after the data format conversion.

The data type of *GraphBitRecorder* is the structure `OmronLib\BC_DeviceMonitor\sGraphBitRecorder`. The specifications are as follows:

Name	Meaning	Description	Data type	Valid range	Unit	Default
GraphBitRecorder	Graph Bit Recorder	Structure to store bit records after the data format conversion	OmronLib\BC_DeviceMonitor\sGraphBitRecorder			
Count	Number of Bit Records	Number of bit data that is stored in the bit data array	UINT	0 to 250	–	–
Label	Scale Line Label	Array to store the element number of bit data corresponding to the scale line	ARRAY[0..20] OF UINT	Depends on data type.	–	–
BitData	Bit Data	Array to store bit data	ARRAY[0..3,0..249] OF BOOL	–	–	–

Meanings of Input Parameters

Offset (Offset), *Magnification* (Magnification Ratio), and *DivisionNum* (Number of Divisions) input parameters have the following meanings.

● Offset (Offset)

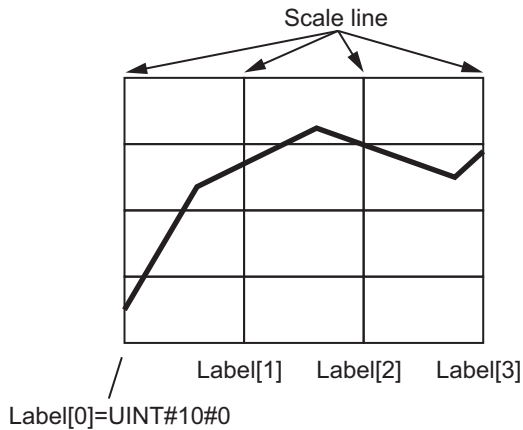
This specifies the place of the bit record from the top whose data format is to be converted first. The top of the bit record for data format to be converted becomes the position acquired by adding the *Offset* value to *BitRecorder.Top*.

● Magnification (Magnification Ratio)

This specifies the magnification ratio for graph display. When the number of bit data stored in the graph bit data recorder is reduced, the graph display is extended in the X axis direction according to the *Magnification* value. The number of bit data whose data format is to be converted is $250/\text{Magnification}$ from the top of the bit data recorder. For example, with *Magnification* = `UINT#10#2`, the data format of $250/2 = 125$ bit data from the top is converted.

● DivisionNum (Number of Divisions)

These are the numbers of X axis scale divisions that are used for the broken-line graph function of NS-series PT. For example, with *DivisionNum#* = `UINT#10#3`, the X axis has 4 scale lines as shown in the following figure. According to the values of *DivisionNum*, the value of the scale line label of the graph bit recorder are calculated automatically. A value of *Label[0]* is always `UINT#10#0`.



Bit Record Whose Data Format is to be Converted

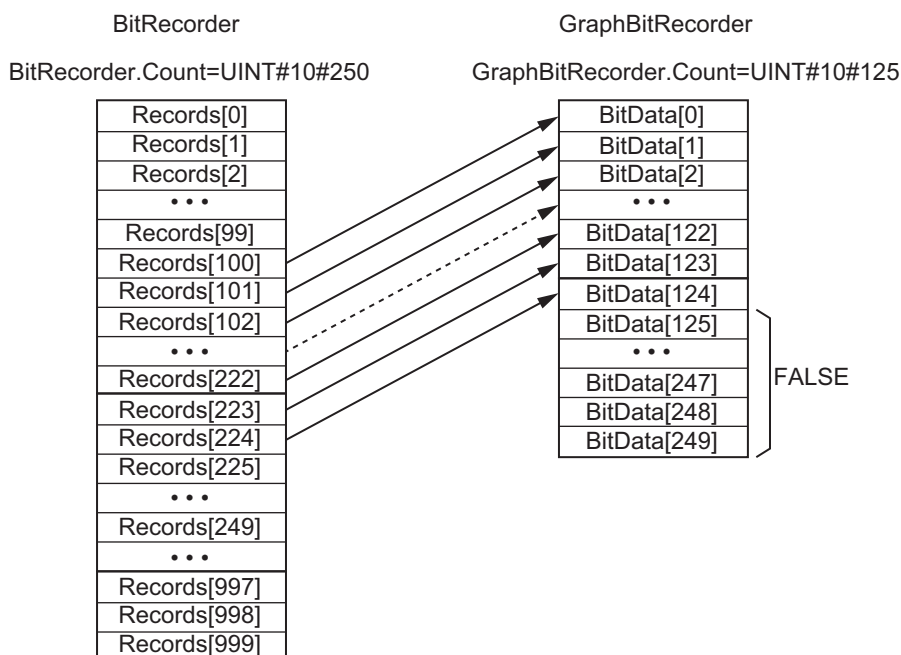
The bit record in the bit record recorder whose data format is to be converted is determined by the following three elements.

- Number of bit records in a bit recorder
- *Offset* value
- *Magnification* value

Out of all log data specified for *Offset*, the data format of 250 bit records is to be converted. However, the number of bit data whose data format is to be converted varies with *Magnification* values.

For example, when the number of bit records in the bit recorder is 250, *Offset* = `UINT#10#100`, and *Magnification* = `UINT#10#2`, the number of bit records whose data format is to be converted is 125 as shown in the following figure. Therefore, the value of the number of bit data in the graph bit data recorder *GraphBitRecord.Count* is `UINT#10#125`.

Also, when the number of bit records whose data format is to be converted is less than 250, the remaining array element values are FALSE. In the example shown in the following figure, the number of log data whose data format is to be converted is 125, thus the remaining array element value is 125.



Sample Programming

Description of Operation

This sample programming performs the following processes by changing the value of *StoreStop*.

StoreStop	Processing
TRUE	Adds bit records that are input to the bit recorder every 1 ms.
FALSE	Discards the oldest bit record in the bit recorder every 10 ms.

Additionally, the sample programming constantly converts and displays bit records in the bit recorder to the data format that is suitable for displaying as a broken-line graph on NS-series PT.

Variables

● Internal Variables

Name	Data type	Default	Comment
Inst_TON1	TON		Instruction instance of timer
BitRecord	OmronLib\BC_DeviceMonitor\BitRecord		Workpiece for data storing
BitRecorder	OmronLib\BC_DeviceMonitor\BitRecorder		Bit recorder
Tmp1	BOOL		Workpiece
Tmp2	BOOL		Workpiece

● External Variables

Name	Data type	Constant	Comment
GraphBitRecorder	OmronLib\BC_DeviceMonitor\GraphBitRecorder		Graph bit recorder
Offset	UINT		Offset
Magnification	UINT		Magnification ratio
N1_Input_Bit_00	BOOL		Input bit 00
N1_Input_Bit_01	BOOL		Input bit01
N1_Input_Bit_02	BOOL		Input bit02
N1_Input_Bit_03	BOOL		Input bit03
Store_Stop	BOOL		Data storing pause
BitRecorderCount	UINT		Number of bit records in a bit recorder

Structured Text (ST)

```

IF Store_Stop AND Get10msClk() THEN
  //Discard data every 10 ms
  IF NOT(Tmp1) THEN
    \\OmronLib\BC_DeviceMonitor\BitRecorderGet(BitRecorder:=BitRecorder);
  END_IF;
  Tmp1:=TRUE;
ELSE
  Tmp1:=FALSE;
END_IF;

IF NOT(Store_Stop) AND Get1msClk() THEN
  IF NOT(Tmp2) THEN
    //Store data every 1 ms
    BitRecord.BitData[0]:=N1_Input_Bit_00;
    BitRecord.BitData[1]:=N1_Input_Bit_01;
    BitRecord.BitData[2]:=N1_Input_Bit_02;
    BitRecord.BitData[3]:=N1_Input_Bit_03;

    //Record data
    \\OmronLib\BC_DeviceMonitor\BitRecorderPut(Record:=BitRecord, BitRe-
corder:=BitRecorder);
  END_IF;
  Tmp2:=TRUE;
ELSE
  Tmp2:=FALSE;
END_IF;

//Data processing for graph display
\\OmronLib\BC_DeviceMonitor\BitRecorderToGraph(
  Offset:=Offset,
  Magnification:=Magnification,
  DivisionNum:=UINT#3,
  BitRecorder:=BitRecorder,
  GraphBitRecorder:=GraphBitRecorder);

BitRecorderCount:=BitRecorder.Count;

```

CX-Designer Settings

The CX-Designer displays four types of graph of input bit data from N1_Input_Bit_00 to N1_Input_Bit_03 using the broken-line graph function of NS-series PT.

Configure the following settings with CX-Designer. Note that, in the following setting example, the name of the host in the CX-Designer communication setup is set to HOST3.

● Size Settings in X Axis Direction and Y Axis Direction

Configure settings in X axis direction and Y axis direction under **Graph** tab in the Broken-line Graph setting window.

- To set the graph size in X axis direction, set *GraphBitRecorder.Count* by going to **No. of vertices in each line - Display Points - Indirect Reference**.
- Set 250, which is the maximum value of *GraphBitRecorder.Count* of BitDataToGraph function, by going to **No. of vertices in each line - Monitor Points**.
- To set the graph size in Y axis direction, set **Maximum Limit** for each broken-line to 5 and **Minimum Limit** to 0.
- Set **INT (signed 1 word)** in **Storage Type**.

Set Maximum Limit to 5.

Set INT (signed 1 word) in Storage Type.

N1_Input_Bit_00
N1_Input_Bit_01
N1_Input_Bit_02
N1_Input_Bit_03

No.	Address	Maximum Limit	Minimum Limit	Nor	Out	Line S	C	S	k
1	HOST3:GraphBitRecorder.BitData[0,0]	5	0			Solid	1	S	↑
2	HOST3:GraphBitRecorder.BitData[1,0]	5	0			Solid	1	S	↑
3	HOST3:GraphBitRecorder.BitData[2,0]	5	0			Solid	1	S	↑
4	HOST3:GraphBitRecorder.BitData[3,0]	5	0			Solid	2	S	↑

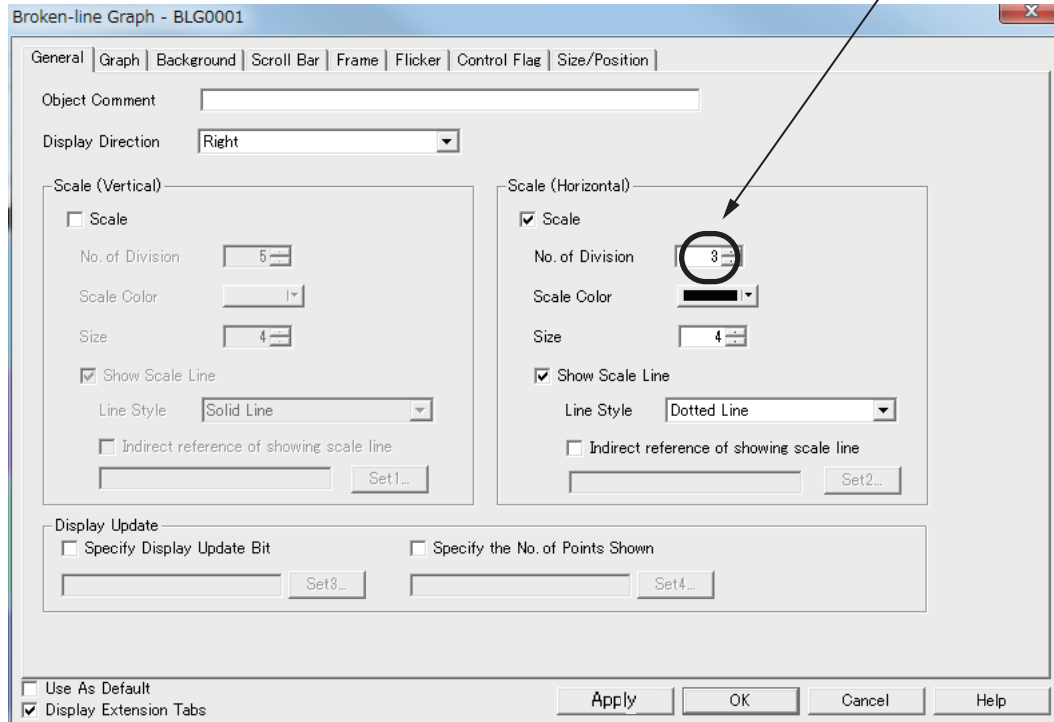
Set Monitor Points to 250.

Set Display Points to *GraphBitRecorder.Count*

● X Axis Scale Line and Y Axis Scale Line Settings

For X axis scale line, set 3, which is the value of *DivisionNum* of BitRecorderToGraph function, by going to **General** tab - **Scale(Horizontal)** - **No. of Divisions** in the Broken-line Graph setting window.

Set No.of Division for Scale(Horizontal) to 3.



● **Settings for Broken-line Graph Line**

Configure settings for each line in the Line Setting (Broken-line Graph) window.

- Set [First Address] to a setting from GraphBitRecorder.BitData[0,0] to GraphBitRecorder.BitData[0,3].
- To display four graphs vertically on top of one another, set **Maximum** to 5, and **Minimum** to 0. Set **Display Offset** as follows.

Bit Data	Display Offset (dot)
N1_Input_Bit_00	10
N1_Input_Bit_01	100
N1_Input_Bit_02	190
N1_Input_Bit_03	280

- To display BOOL data (type), select the **Step Display** check box.

Set GraphBitRecorder.BitData[0,0].

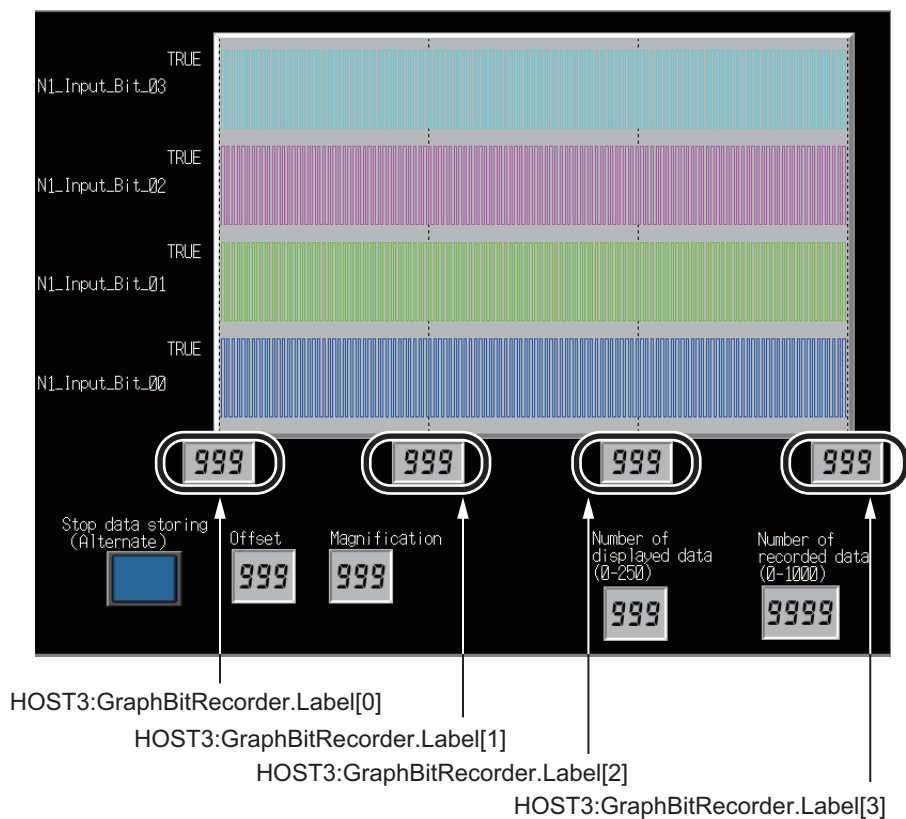
The screenshot shows the 'Line Setting (Broken-line Graph)' dialog box with the following settings and annotations:

- Start Address:** HOST3:GraphBitRecorder.BitData[0,0] (Annotated: Set GraphBitRecorder.BitData[0,0].)
- Maximum/Minimum Value:**
 - Range: -32768 - 32767
 - Maximum: 5 (Annotated: Set Maximum to 5.)
 - Minimum: 0
- Line Color:** Within Min/Max: Blue, Outside of Range: Red
- Line Style:** Solid Line
- Display Offset:** 10 dots (Annotated: Set Display Offset to 10.)
- Step Display:** Checked (Annotated: Select the Step Display check box.)
- Marker:** None selected, Size: Small
- Line:** Display when address changes to ON (unchecked)

● Assignment of Variables to Functional Objects on Screen

Assume that the broken-line display screen of NS PT is to be configured to the settings listed in the following figure. The variables are assigned to the functional objects on the screen as follows.

Functional Objects	Label	Assigned variable
ON/OFF button	Stop data storing (Alternate)	Write address HOST3:Store_Stop
Numerical Display&Input	Offset	Address HOST3:Offset
Numerical Display&Input	Magnification	Address HOST3:Magnification
Numerical Display&Input	Number of displayed data (0-250)i	Address HOST3:GraphBitRecorder.Count
Numerical Display&Input	Number of recorded data (0-1000)i	Address HOST3:BitRecorder.Count
Numerical Display&Input		Address HOST3:GraphBitRecorder.Label[0]
Numerical Display&Input		Address HOST3:GraphBitRecorder.Label[1]
Numerical Display&Input		Address HOST3:GraphBitRecorder.Label[2]
Numerical Display&Input		Address HOST3:GraphBitRecorder.Label[3]



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 169.

*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 169.

*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

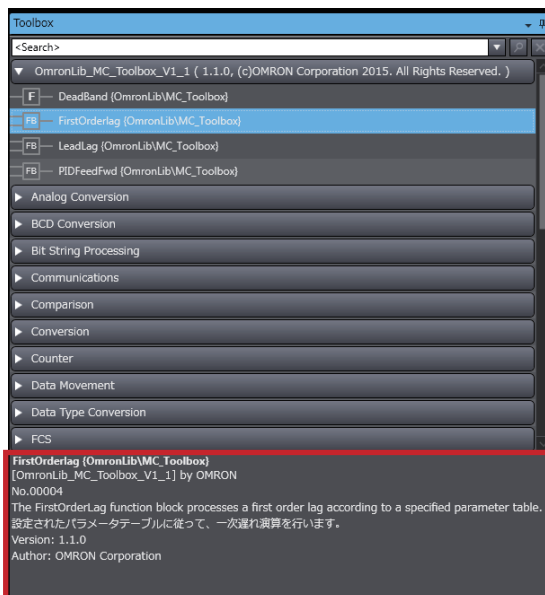
In/Out	Name	In/Out	Data Typel	Edge	Initial Value	Retain	Constant	Comment
Externals	Enable	Input	BOOL	No Edge	False	<input type="checkbox"/>	<input type="checkbox"/>	
	InCalc	Input	LREAL	No Edge	0.0	<input type="checkbox"/>	<input type="checkbox"/>	
	Kp	Input	LREAL	No Edge	1.0	<input type="checkbox"/>	<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.

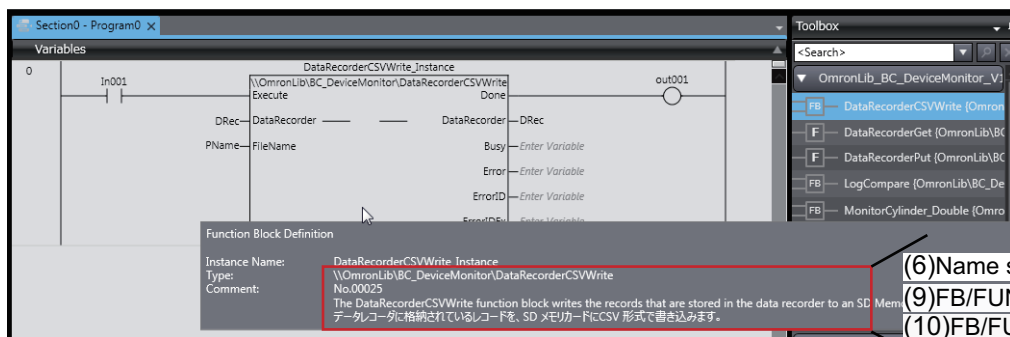
The text “by OMRON” which is shown on the right of the library name (1) indicates that this library was provided by OMRON.



- (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.

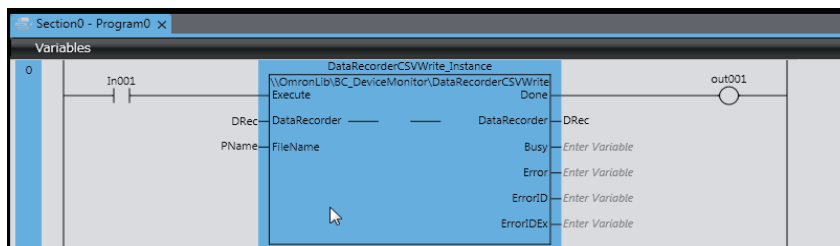
The following are the examples of items that you may need to customize.

- 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.

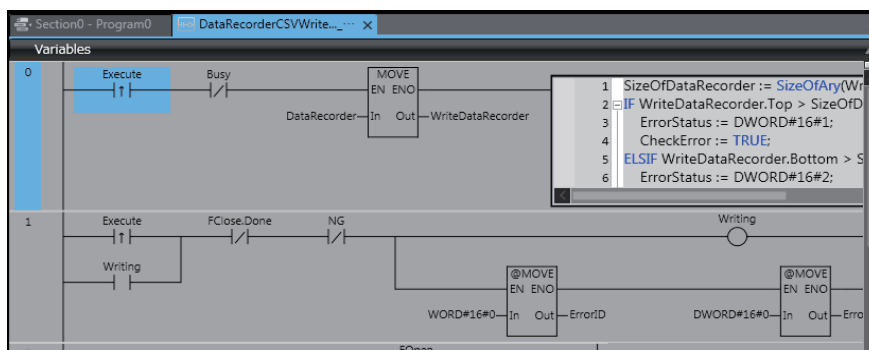
Use the following procedure to refer to the source codes of function blocks and functions.

- 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

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 2015-2020 All Rights Reserved.
In the interest of product improvement,
specifications are subject to change without notice.

Cat. No. **W552-E1-05**

0520