OMRON

Sysmac Library

User's Manual for MC Tool Box Library

SYSMAC-XR003

SYSMAC always in control

W547-E1-05

NOTE

Trademarks

Copyrights

Introduction

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

This manual provides information required to use the function blocks in the MC Tool Box 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 MC Tool Box Library is used to implement a program to perform motor control in the user program.

The processings such as PID processing and filter processing are provided in this library.

You can use this library to reduce manpower of programming when creating a program to perform motor control.

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.
- Personnel with knowledge of control logic.

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 *SYSMAC-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

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 CPU Unit 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.	
When you perform a test run, hold an emergency stop switch in your hand or otherwise prepare for rapid motor operation.	

Precautions for Safe Use

Designing

- Understand the library specifications completely before you use the MC Tool Box Library.

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.

Using Sample Programming

- 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.
- 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

- When you use a function block that changes an *Enabled* output variable to TRUE while the processing result is output normally, confirm that *Enabled* is TRUE before you use the processing result.
- Use the Real Number Check (CheckReal) instruction to make sure that an input parameter is not infinity or nonnumeric data.

Related Manuals

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

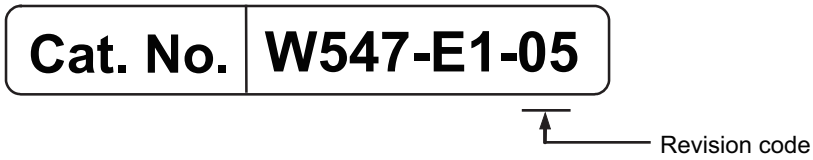
Manual name	Cat. No.	Model numbers	Application	Description
NX-series CPU Unit Hardware User's Manual	W535	NX701-□□□□	Learning the basic specifications of the NX-series NX701 CPU Units, including introductory information, designing, installation, and maintenance. Mainly hardware information is provided	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 *NJ/NY-series G code Instructions Reference Manual* (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 *NJ/NY-series NC Integrated Controller User's Manual* (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.



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

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.

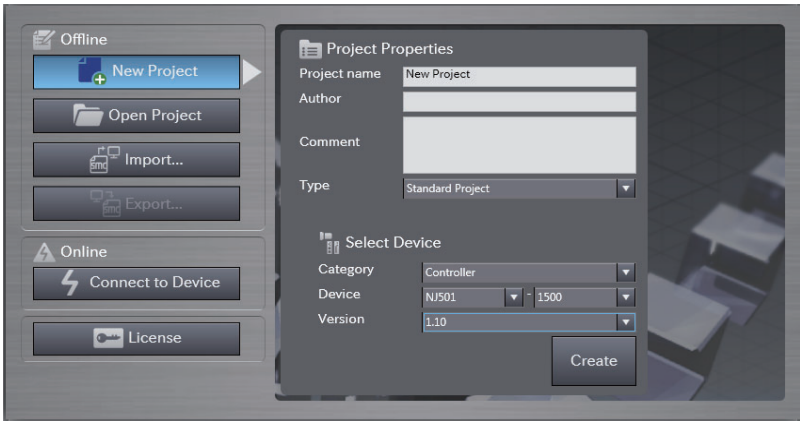
- Using newly installed Sysmac Libraries
- Using upgraded Sysmac Libraries

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.

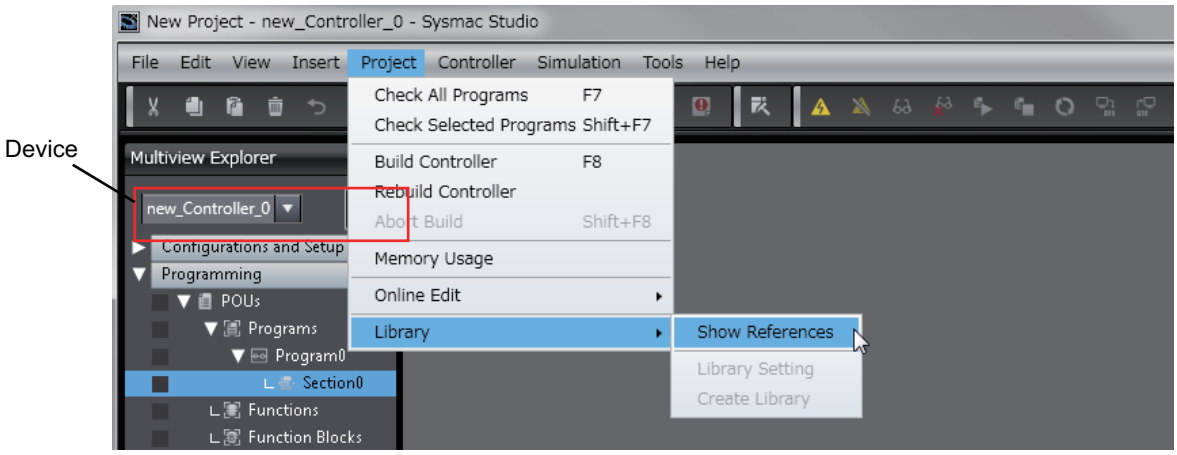


Precautions for Correct Use

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.

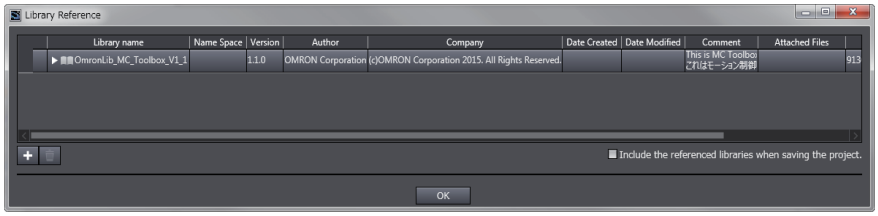
2 Select **Project** – **Library** – **Show References**.



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.



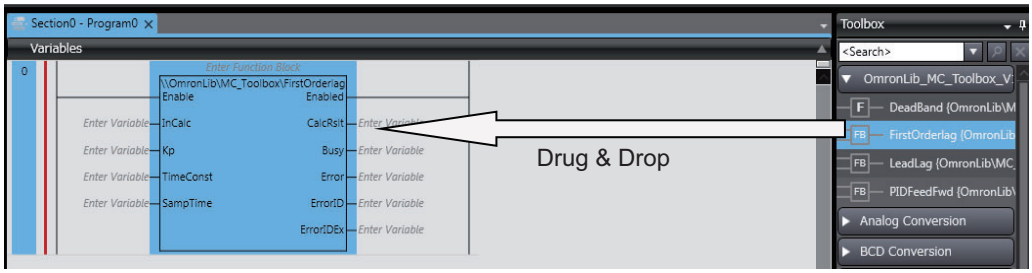
The Sysmac Library file is read into the project.

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

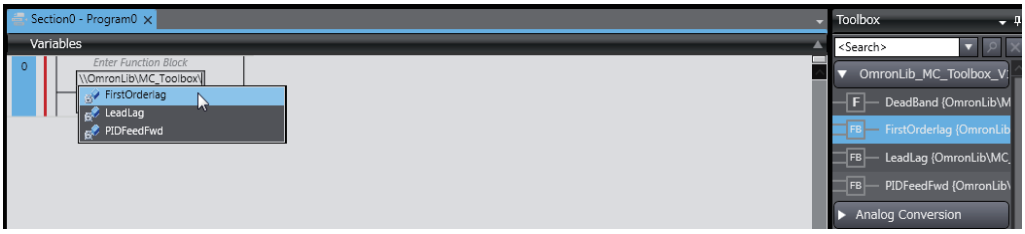
For the procedure for adding and setting libraries in the above screen, refer to the *Sysmac Studio Version 1 Operation Manual* (Cat. No. W504).

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

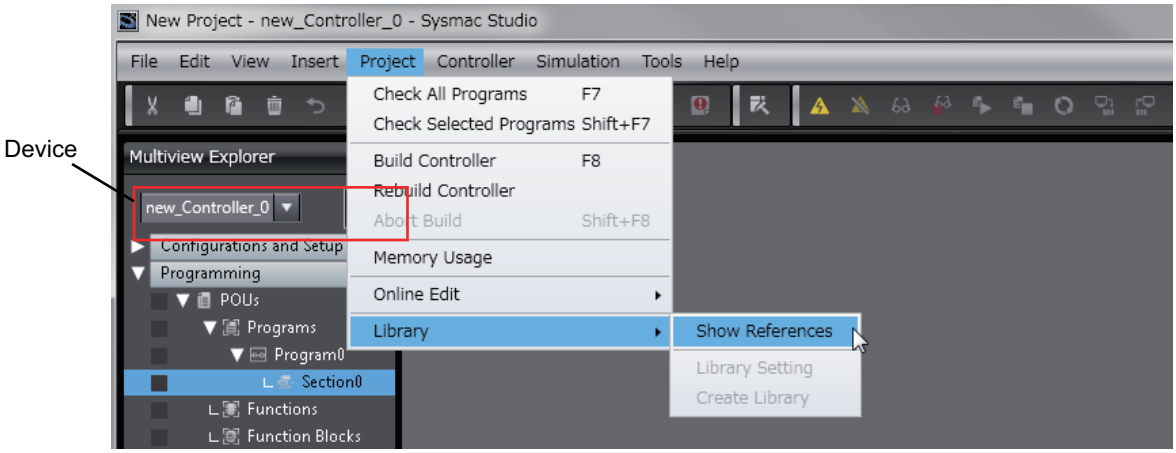
After you upgrade the Sysmac Studio, check all programs and make sure that there is no error of the program check results on the Build Tab Page.

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.

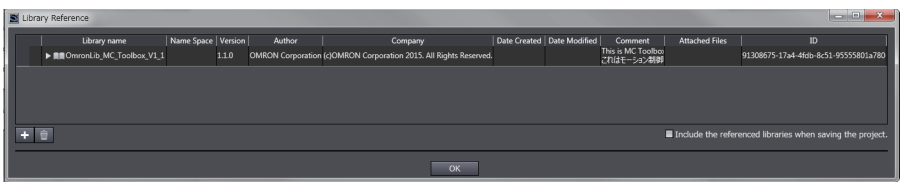
2 Select **Project** – **Library** – **Show References**.



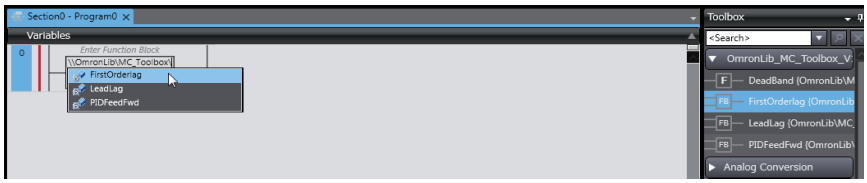
Precautions for Correct Use

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

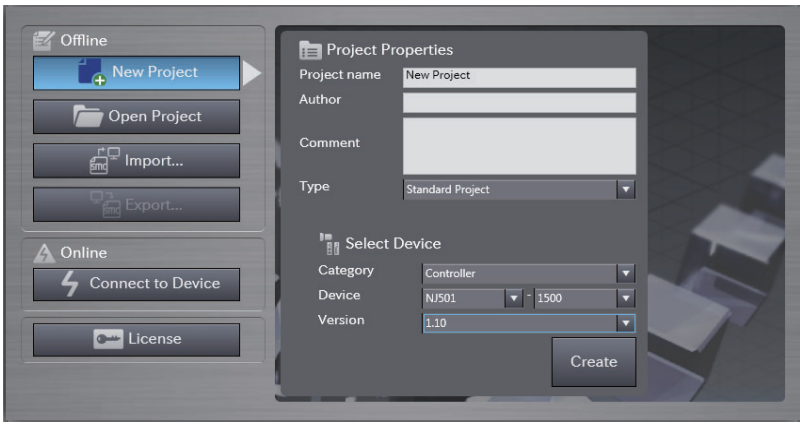
You can use Sysmac Libraries uploaded from a CPU Unit or an Industrial PC to your computer if they are not installed.

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.



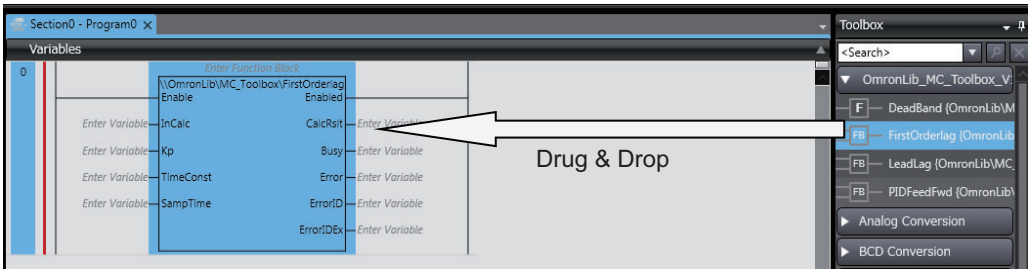
2 Connect the computer to the CPU Unit or the Industrial PC and place it online.

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

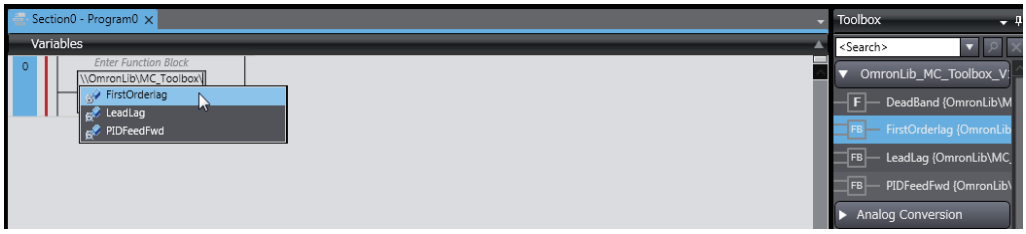
Now, when you select the Ladder Editor or ST Editor, the function blocks and functions included in the Sysmac Library used in the uploaded POUs appear in the Toolbox.

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 Stutio 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 then using the installer. Please install the manual and help files using the installer if you need them.

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

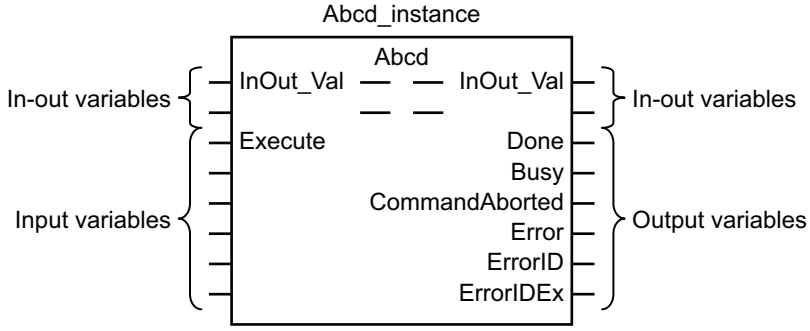
Common input variables and output variables used in functions and function blocks are as follows.

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 continously 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.

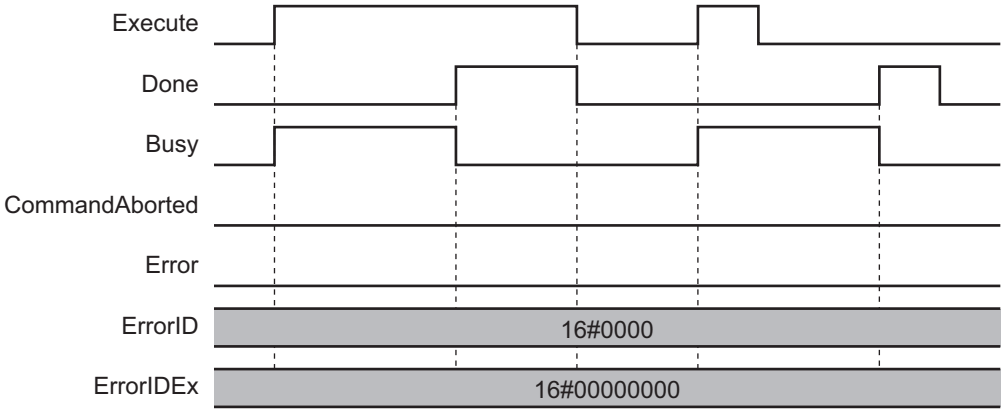


- If *Execute* is TRUE and *Done*, *CommandAborted*, or *Error* changes to TRUE, *Done*, *Command-Aborted*, and *Error* changes to FALSE when *Execute* is changed to FALSE.
- If *Execute* is FALSE and *Done*, *CommandAborted*, or *Error* changes to TRUE, *Done*,*Command-Aborted*, and *Error* changes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

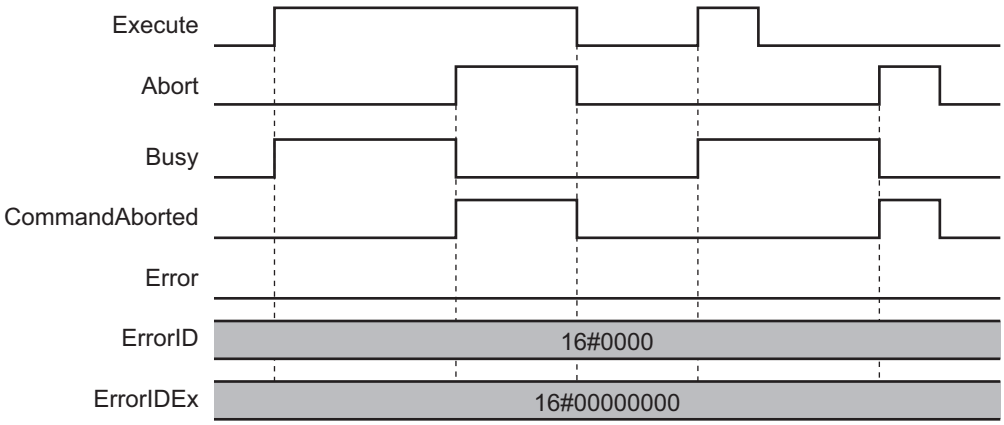
Timing Charts

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

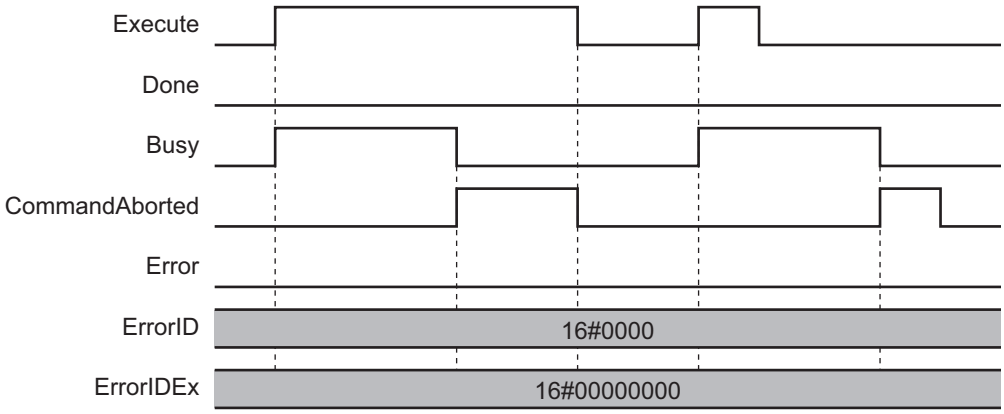
● Normal End



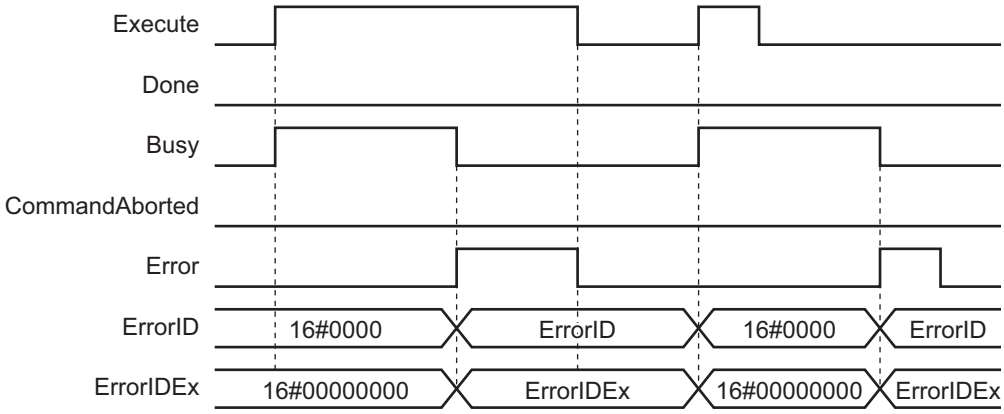
● Canceled Execution



Aborted Execution

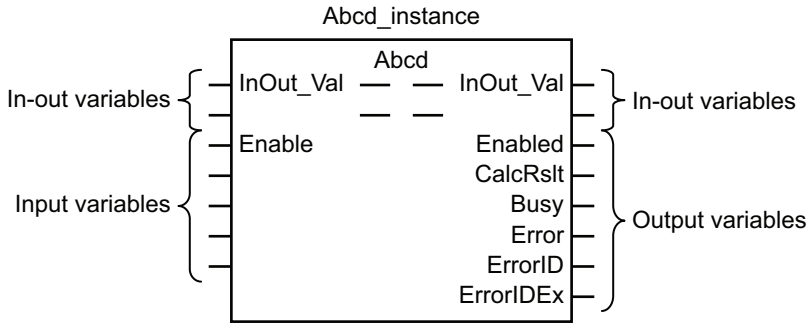


Errors



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.

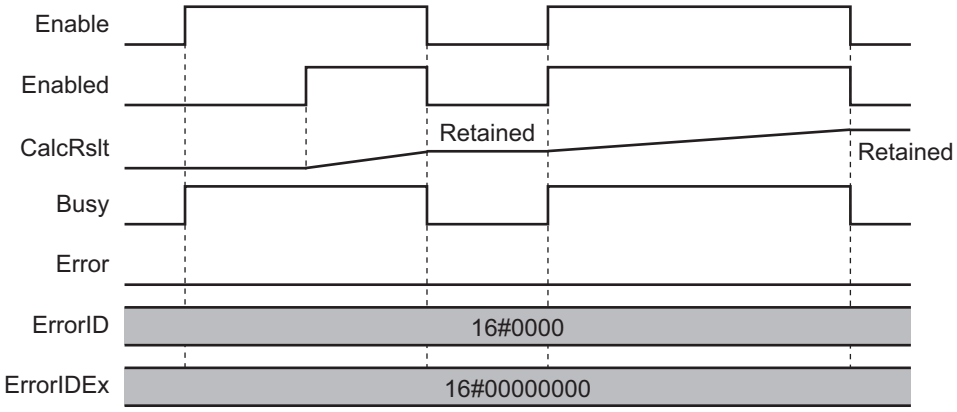


- If an error occurs, the relevant error code and expansion error code are set in *ErrorID* (Error Code) and *ErrorIDEx* (Expansion Error Code). The error codes are retained even after *Error* changes to FALSE, but *ErrorID* is set to 16#0000 and *ErrorIDEx* is set to 16#0000 0000 when *Enable* changes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

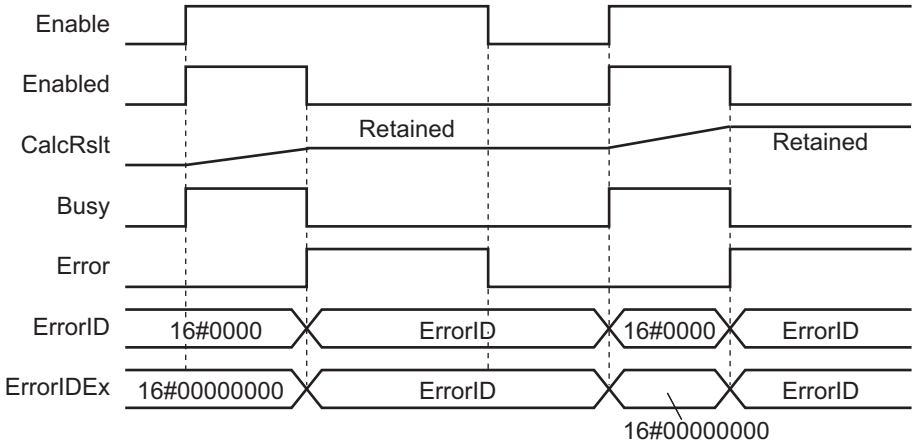
Timing Charts

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

● Normal End



● Errors


Enable
Enabled
CalcRslt
Busy
Error
ErrorID
ErrorIDEx
Retained
Retained
16#0000
ErrorID
16#0000
ErrorID
16#00000000
ErrorID
ErrorID
16#00000000

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.

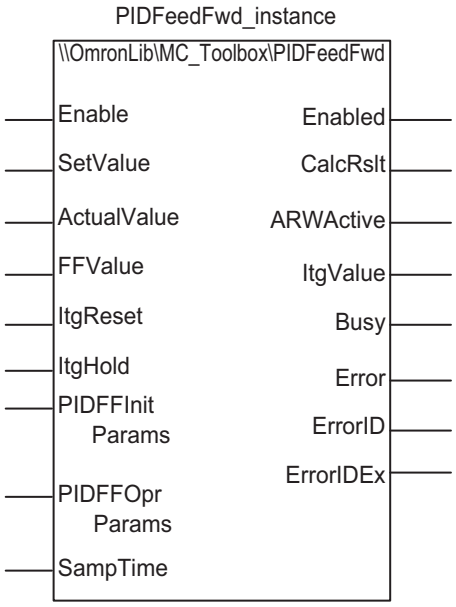
For details on re-execution, refer to the motion control user's manual.

Specifications of Individual Function Blocks

Function block name	Name	Page
PIDFeedFwd	PID Feedforward	P. 34
FirstOrderLag	First Order Lag	P. 58
LeadLag	Phase Lead Lag	P. 67
DeadBand	Deadband Control without Output Offset	P. 77

PIDFeedFwd

The PIDFeedFwd function block performs PID feedforward processing according to a specified parameter table.

Function block name	Name	FB/ FUN	Graphic expression	ST expression
PIDFeed-Fwd	PID Feedforward	FB		PIDFeedFwd_instance (Enable, SetValue, ActualValue, FFValue, ItgReset, ItgHold, PIDFFInitParams, PIDFFOprParams, SampTime, Enabled, CalcRslt, ARWActive, ItgValue, Busy, Error, ErrorID, ErrorIDEx);

Function Block and Function Information

Item	Description
Library file name	OmronLib_MC_Toolbox_V1_1.slr
Namespace	OmronLib\MC_Toolbox
Function block and function number	00003
Source code published/not published	Not published
Function block and function version	1.01

Variables

	Meaning	I/O	Description	Valid range	Unit	Initial value
Enable	Enable	Input*1	TRUE: Execute FALSE: Stop	TRUE or FALSE	---	FALSE
SetValue	Set Point	Input*1	Input the set point.	Depends on data type.	---	0.0
ActualValue	Process Value	Input*1	Input the process value.	Depends on data type.	---	0.0
FFValue	Feedforward Value	Input*1	Input the feedforward value.	Depends on data type.	---	0.0
ItgReset	Integral Processing Reset	Input*1	Use to stop the integral processing and output zero to the integral process value. TRUE: Resets the integral processing. FALSE: Continues the integral processing.	TRUE or FALSE	---	FALSE
ItgHold	Integral Processing Hold	Input	Reserved	TRUE or FALSE	---	FALSE
PIDFFInitParams	Initial Setting Parameters	Input*2	Initial setting parameters	---	---	---
PIDFFOprParams	Operation Setting Parameters	Input*1	Operation parameters	---	---	---
SampTime	Sampling Period	Input*2	The period for performing the PID feedforward processing.	0.001 to 100000.0*3	ms	1.0
Enabled	Enabled	Output	Outputs TRUE in any period in which *CalcRslt* (Processing Result) is updated.	TRUE or FALSE	---	---
CalcRslt	Processing Result	Output	Outputs the processing result.	Depends on data type.	---	---
ARWActive	ARW Executing	Output	Outputs the ARW (anti-reset windup) status. TRUE: Executing. FALSE: Not executing.	Depends on data type.	---	---
ItgValue	Calculated Integral Variable	Output	Outputs the current calculated integral variable.	Depends on data type.	---	---
Busy	Busy	Output	Indicates when processing is in progress.	TRUE or FALSE	---	---
Error	Error End	Output	Outputs TRUE while there is an error.	TRUE or FALSE	---	---
ErrorID	Error Code	Output	Contains the error code when an error occurs.	*4	---	---
ErrorIDEx	Expansion Error Code	Output	Contains the expansion error code when an error occurs.	*4	---	---

*1. Any changes made during execution of this function block are applied to the output results in the same control period.

*2. The set values in the task period in which *Enable* to this function block changes to TRUE are used in processing. Values that change while *Enabled* is TRUE are not applied to processing.

*3. Any settings below 0.001 (ms) are truncated.

*4. Refer to *Troubleshooting* on page 48 for details.

	Boolean	Bit strings				Integers								Real numbers		Times, durations, dates, and text strings				
	BOOL	BYTE	WORD	DWOR	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Enable	OK																			
SetValue															OK					
ActualValue															OK					
FFValue															OK					
ItgReset	OK																			
ItgHold	OK																			
PIDFFInitParams	Refer to *Function* for details on the structure OmronLib\MC_Toolbox/sPIDFF_INIT_PARAMS.																			
PIDFFOprParams	Refer to *Function* for details on the structure OmronLib\MC_Toolbox/sPIDFF_OPR_PARAMS.																			
Enable	OK																			
SampTime															OK					
CalcRslt															OK					
ARWActive	OK																			
ItgValue															OK					
Busy	OK																			
Error	OK																			
ErrorID			OK																	
ErrorIDEx				OK																

Precautions for Correct Use

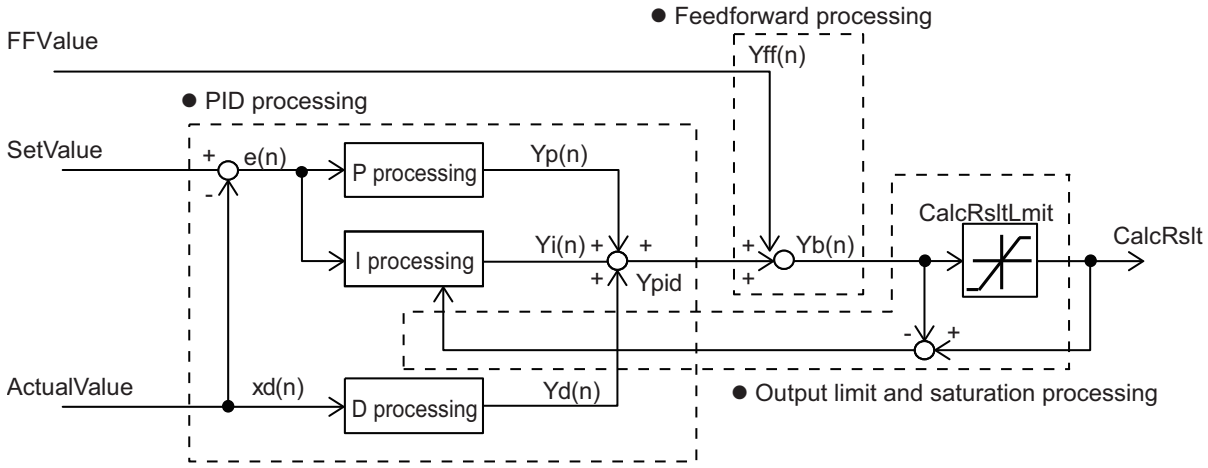
- If you use this function block to calculate control values, confirm that *Enabled* is TRUE before you use the Processing Result (*CalcRslt*) as a control value.
- Until this function block is executed the first time, 0.0 is output for the Processing Result (*CalcRslt*). After this function block is executed, the most recent valid output value is always output to the Processing Result (*CalcRslt*).

  For example, if *Enable* to this function block is changed between TRUE and FALSE, the most recent valid output value for TRUE is output to the Processing Result (*CalcRslt*) while *Enable* is FALSE. Even if this function block ends in an error during execution, the most recent valid output value is output to the Processing Result (*CalcRslt*).
- If *SetValue* or *ActualValue* is nonnumeric or infinity, *CalcRslt* may become nonnumeric or infinity. In this case, *Enabled* will change to FALSE. If *Enabled* changes to FALSE, do not use *CalcRslt* as a control value.

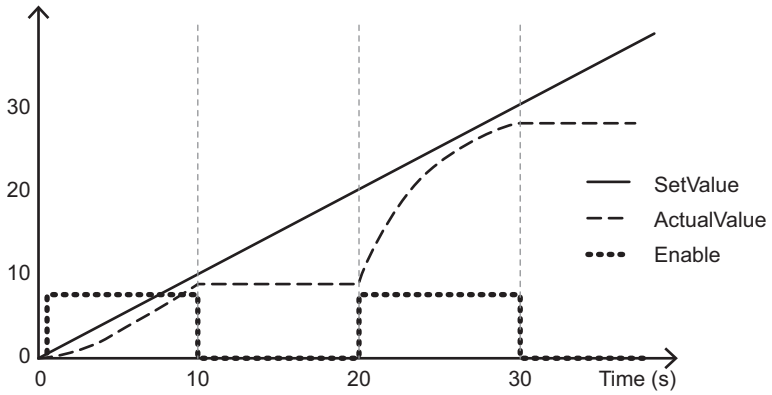
Function

PID Feedforward Processing

While *Enable* to this function block is TRUE, PID feedforward processing is repeated each fixed period. This processing is shown in the following block diagram.



In the example in the following figure, the output values are shown when *Enable* changes to FALSE during PID feedforward processing while Set Point (*SetValue*) increases at a specific rate. Here, PID feedforward processing stops from the period in which *Enable* changes to FALSE and the most recent value of the Processing Result (*CalcRslt*) is held. When *Enable* changes back to TRUE, PID feedforward processing for the current input value is started again.



P (Proportional) Processing, I (Integral) Processing, and D (Derivative) Processing

The PID processor performs the following processing according to the specified parameter table.

$$Ypid(n) = Yp(n) + Yi(n) + Yd(n)$$

Ypid(n): PID processing results at time n

Yp(n): P processing result at time n

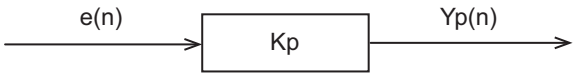
Yi(n): I processing result at time n

Yd(n): D processing results at time n

The processing is described in the following sections.

● P Processing

The following calculations are performed for P processing.



This is expressed as follows with a difference equation:

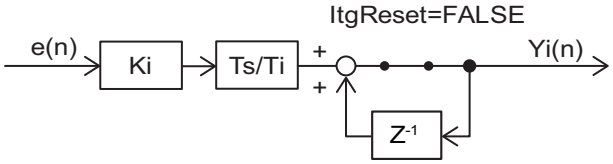
$$Yp(n) = Kp \times e(n)$$

Kp: Proportional gain (Corresponds to the *Kp* input variable.)

e(n): Deviation at time n

● I Processing

The following calculations are performed for the I action.



Z^{-1}: Lag operator

This is expressed as follows with a difference equation:

$$Yi(n) = Ki \times \frac{Ts}{Ti} \times e(n) + Yi(n-1)$$

However, the initial value of the I processing output value, Yi(0), will be 0.

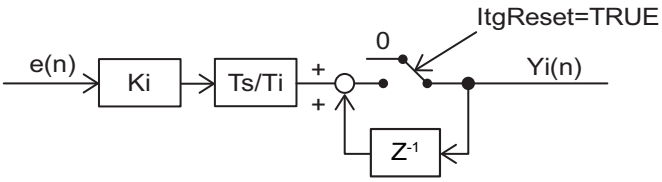
Ki: Integral gain (Corresponds to the *Ki* input variable.)

Ts: PID processing execution period

Ti: Integration time (Corresponds to the *Ti* input variable.)

Refer to *Execution Timing of PID Feedforward Processing* on page 41 for details on Ts.

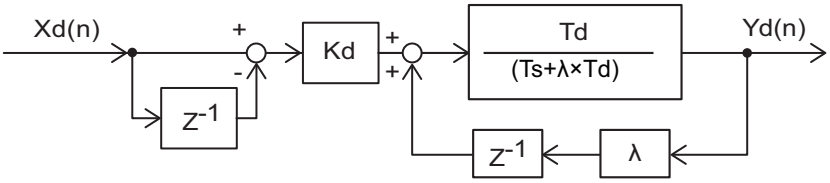
For the integral action, you can change the Integral Reset (*ItgReset)* input variable to TRUE to reset the calculated integral variable. At the same time, the value of Yi(n) also goes to 0.



The calculated I variable is output to Calculated Integral Value (*ItgValue*).

● D Processing

The derivative action processes the incomplete derivative to prevent increases in high-frequency noise.



Xd(n): Input value to D processing

λ: Incomplete derivative coefficient (Always 0.3.)

Kd: Derivative gain (Corresponds to the *Kd* input variable.)

Td: Derivative time (Corresponds to the *Ti* input variable.)

This is expressed as follows with a difference equation:

$$Yd(n) = \frac{Td}{Ts + \lambda \times Td}\{Kd \times (Xd(n) - Xd(n-1)) + \lambda \times Yd(n-1)\}$$

However, the initial value of the D processing output value, Yd(0), will be 0.

Feedforward Processing

The feedforward processor performs the following processing. The feedforward processor adds the value set for the *FFValue* (Feedforward Value) input variable to the value calculated by the PID processor.

$$Yb(n) = Yff(n) + Ypid(n)$$

Yb(n): Previously calculated result of the processing output limit at time n

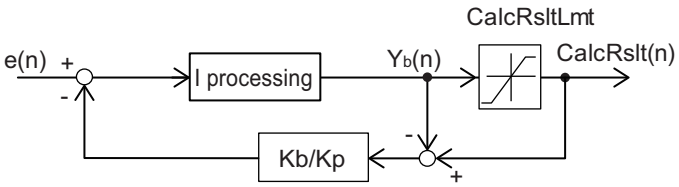
Yff(n): Feedforward value (Corresponds to the *FFValue* input variable.)

Output Limit and Saturation Processing

The output value is limited by Output Limit (*CalcRsltLmt*).

$$CalcRslt(n) = \begin{cases} Yb(n) & : (CalcRsltLowLmt < Yb(n) < CalcRsltUpLmt) \\ CalcRsltLowLmt & : (Yb \leq CalcRsltLowLmt) \\ CalcRsltUpLmt & : (CalcRsltUpLmt \leq Yb) \end{cases}$$

Saturation processing is performed to suppress overshooting and undershooting when excessive deviation occurs. For this processing, any processing result that exceeds Output Limit (*CalcRsltLmt*) is subtracted from the I processing deviation. This saturation processing is enabled when the value set for the output limit by Yb(n) is exceeded (i.e., when CalcRslt(n) ≠ Yb(n)). While this processing is enabled, the *ARWActive* (ARW Executing) output variable is TRUE.



Structures

The data type of initial setting parameters *PIDFFInitParams* is structure sPIDFF_INIT_PARAMS. The specifications are as follows:

Variable or member	Name	Description	Data type	Valid range	Unit	Initial value
PIDFFInit-Params	Initial Setting Parameters		OmronLib\MC_Toolbox/sPIDFF_INIT_PARAMS	---	---	---
Ti	Integration Time	Set the integration time for PID control.	LREAL	0.001 to 100000.0*1	ms	1.0
Td	Derivative Time	Set the derivative time for PID control.	LREAL	0.001 to 100,000.0*1	ms	1.0
Kb	Integral Subtraction Gain	Set the value to subtract from the integral variable when the processing output limit is saturated.	LREAL	0.0 to 1.0		1.0

*1. Any settings below 0.001 (ms) are truncated.

The data type of the Operation Setting Parameters (*PIDFFOprParams*) in PID feedforward processing is the structure sPIDFF_OPR_PARAMS.

The specifications are as follows:

Variable or member	Name	Description	Data type	Valid range	Unit	Initial value
PIDFFOpr-Params	Operation Setting Parameters		OmronLib\MC_Tool-box/sPIDFF_SET_PARAMS	---	---	---
Kp	Propor-tional Gain	Set the propor-tional gain for PID control.	LREAL	0.0 to 3000.0	---	1.0
Ki	Integral Gain	Set the integral gain for PID con-trol.	LREAL	0.0 to 3000.0	---	1.0
Kd	Derivative Gain	Set the derivative gain for PID con-trol.	LREAL	0.0 to 3000.0	---	1.0
CalcRslt-LowLmt	Output Lower Limit	Set the lower limit to the calculated output.*1	LREAL	Depends on data type.*2	---	0.0
CalcRsltU-pLmt	Output Upper Limit	Set the upper limit to the calculated output. *1	LREAL	Depends on data type.*2	---	0.0

*1. Set the Output Upper Limit (*CalcRsltUpLmt*) to a value equal to or greater than the value of the Output Lower Limit (*CalcRsltLowLmt*).

*2. If either *CalcRsltLowLmt* or *CalcRsltUpLmt* is nonnumeric data, a PIDFeedFwd Output Limit Out of Range error (error code: 16#3C0A, expansion error code: 16#00000008) will occur.

Precautions for Correct Use

The initial values of Output Upper Limit (*CalcRsltUpLmt*) and Output Lower Limit (*CalcRsltLow-Lmt*) limit the output to 0.0. For this reason, the Process Output (*CalcRslt*) will always be 0.0. Set values that are suitable for the application.

Execution Timing of PID Feedforward Processing

The execution timing for PID feedforward processing is determined by the Processing Period (*Samp-Time*) and the task period in which this function block is executed. The relationship is as follows:

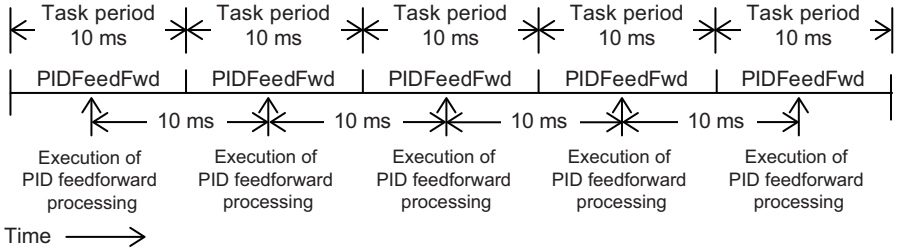
- Processing Period (SampTime) ≤ Task period

  PID feedforward processing is executed in each task period.
- Processing Period (SampTime) > Task period

  PID feedforward processing is executed in the integer multiple of the task period that is longer than the Processing Period (*SampTime*).

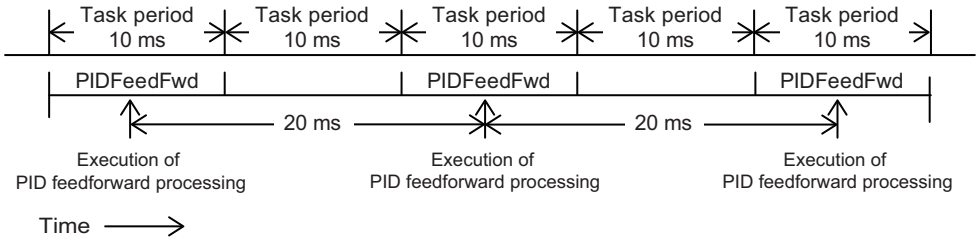
● Task Period = 10 ms and *SampTime* ≤ 10 ms

SampTime is less than or equal to the task period, so PID feedforward processing is executed once every task period.



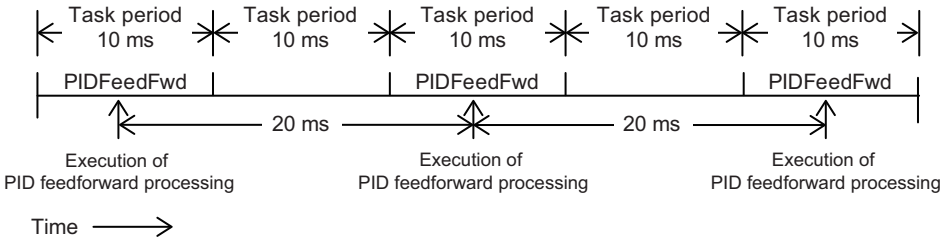
● Task Period = 10 ms and *SampTime* = 11 ms

PID feedforward processing is executed every 20 ms, i.e., the integer multiple of the task period that is longer than the Processing Period (*SampTime*).



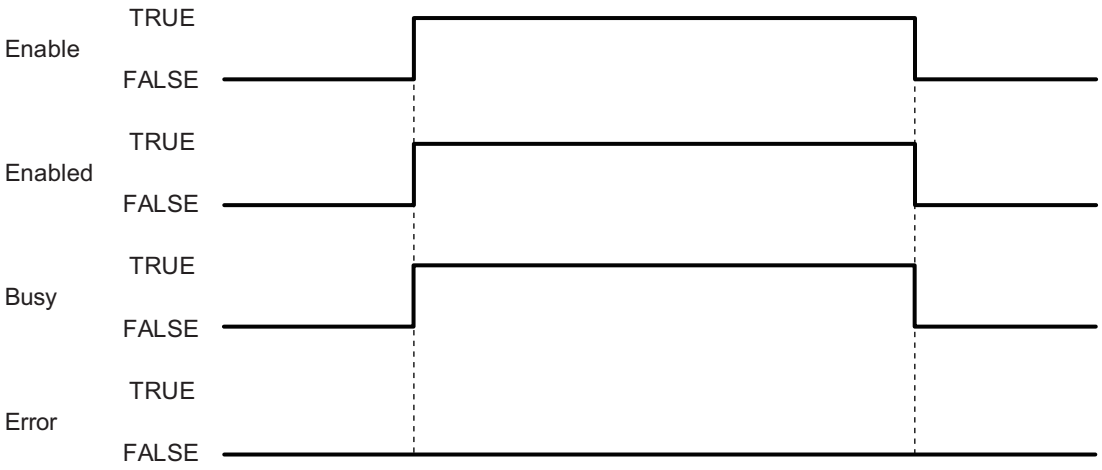
● Task Period = 10 ms and *SampTime* = 19 ms

PID feedforward processing is executed every 20 ms, i.e., the integer multiple of the task period that is longer than the Processing Period (*SampTime*).

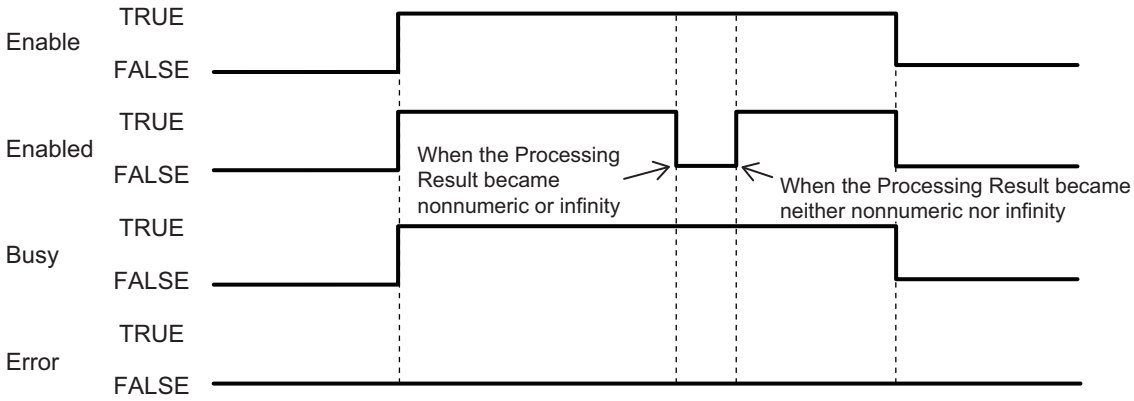


Timing Charts

- When *Enable* changes to TRUE, *Busy* (Executing) changes to TRUE and PID feedforward processing is performed according to the input values.
- When *Enable* changes to FALSE, *Busy* (Executing) changes to FALSE in the same period, PID feedforward processing stops, and the value of *CalcRstl* (the processing result) is no longer updated.

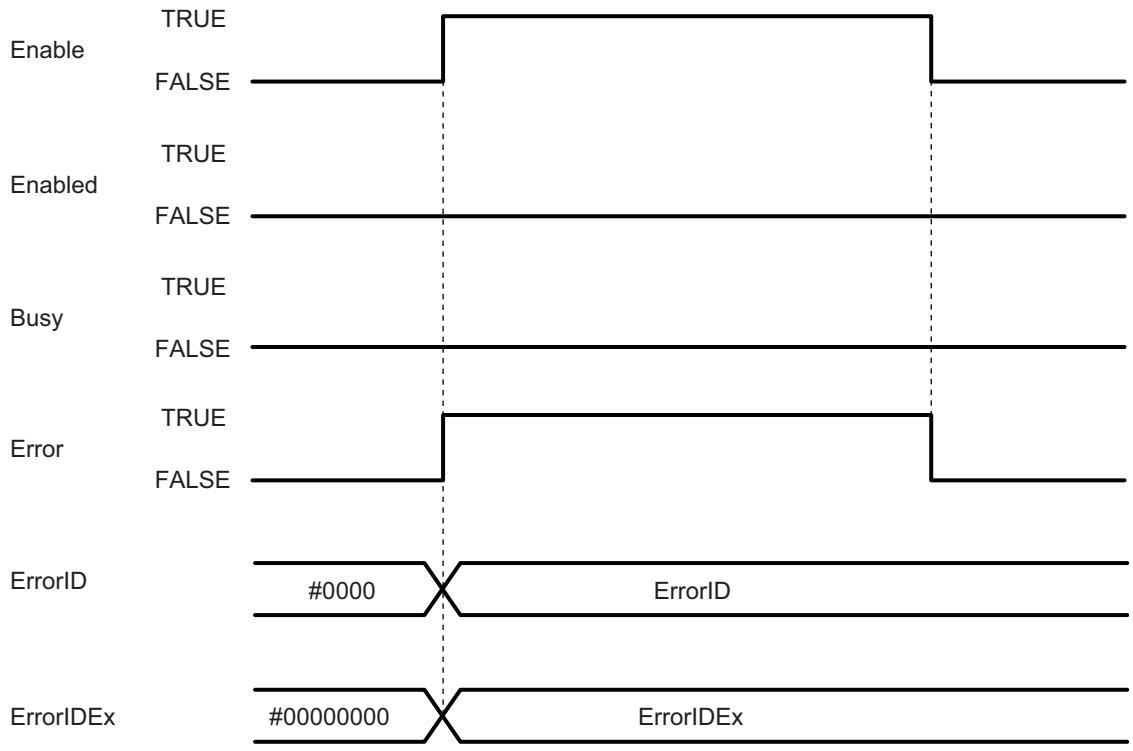


- If the Processing Result (*CalcRslt*) of this function block is nonnumeric or infinity, *Enabled* will change to FALSE. When the Processing Result becomes neither nonnumeric nor infinity, *Enabled* will change to TRUE.



- If an error occurs during function block execution, *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).

  When *Enable* to this function block changes to TRUE, *ErrorID* (Error Code) and *ErrorIDEx* (Expansion Error Code) are cleared.

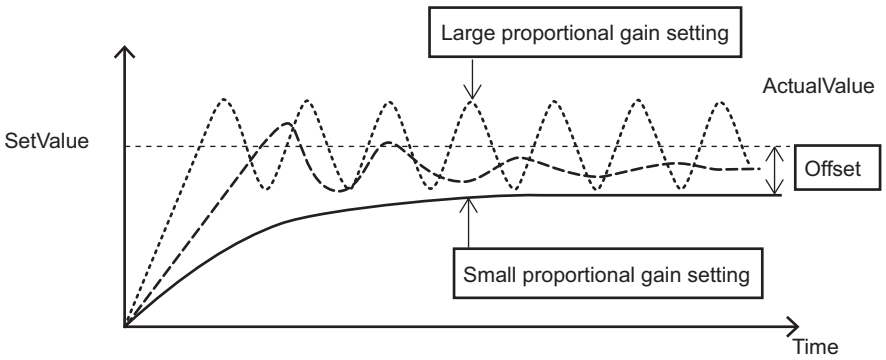


Additional Information

This section describes concepts related to the parameter settings for PID feedforward processing.

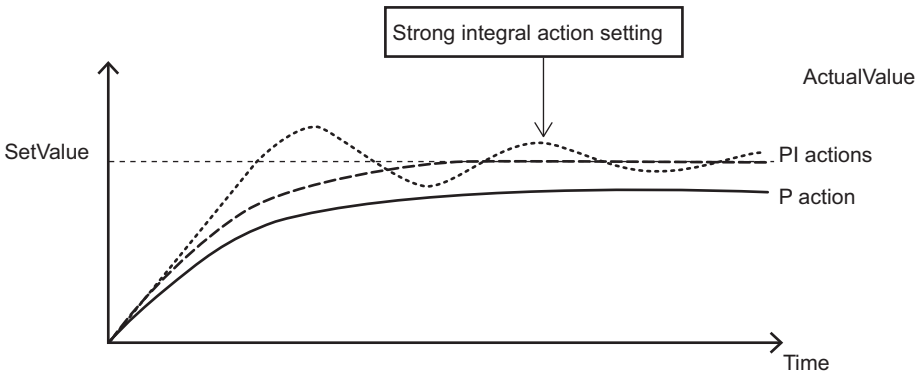
Proportional Action (P Action)

- The strength of the control action is set by setting the value of the proportional gain (*Kp*).
- If only proportional action control is performed, an offset results for the steady state.
- If the proportional gain is increased, the offset is reduced but hunting becomes larger.



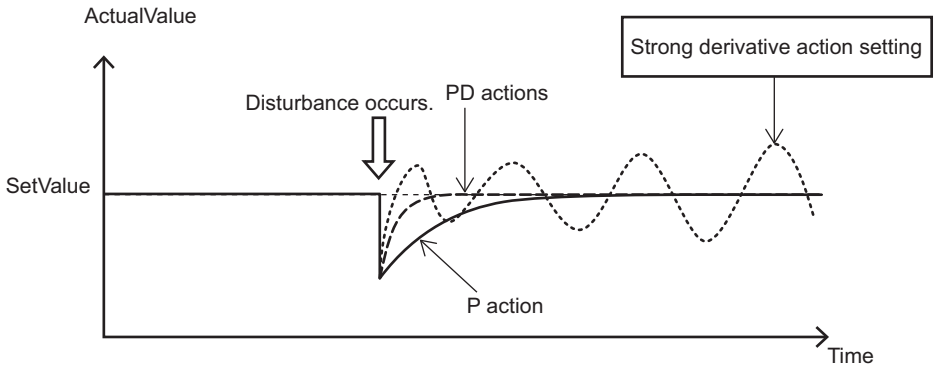
Integral Action (I Action)

- The strength of the integral action is set by setting the values of the integral gain (*Ki*) and integration time (*Ti*).
- The integral action becomes stronger when the integral gain (*Ki*) is increased or the integration time (*Ti*) is decreased.
- If you increase the strength of the integral action, the offset that results from the P action is corrected quickly.
- If the integral action becomes too strong, hunting occurs.



Derivative Action (D Action)

- The strength of the derivative action is set by setting the values of the derivative gain (*Kd*) and derivative time (*Td*).
- The derivative action becomes stronger when the derivative gain (*Kd*) is increased or the derivative time (*Td*) is increased.
- You can make the derivative action stronger to correct for response delays caused by P action or PI action disturbances.
- If the derivative action becomes too strong, hunting occurs.

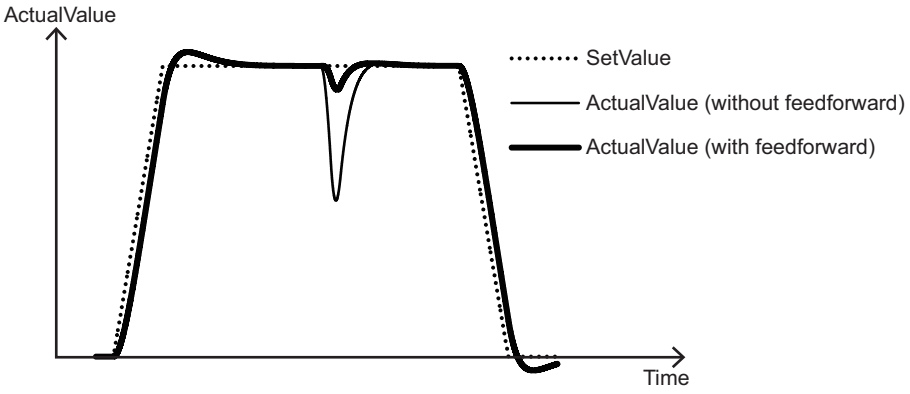


Feedforward Operation

This section provides examples of using feedforward.

● Eliminating Disturbance

To adjust D in the above PID parameters, feedforward is used to suppress the influence of disturbance after the influence of disturbance is applied to the Process Value (*ActualValue*). If the influence and timing of the disturbance are known, you can set the Feedforward Input Value (*FFValue*) in advance to reduce the influence of disturbance and reduce the deviation between the Set Point (*SetValue*) and Process Value (*ActualValue*).

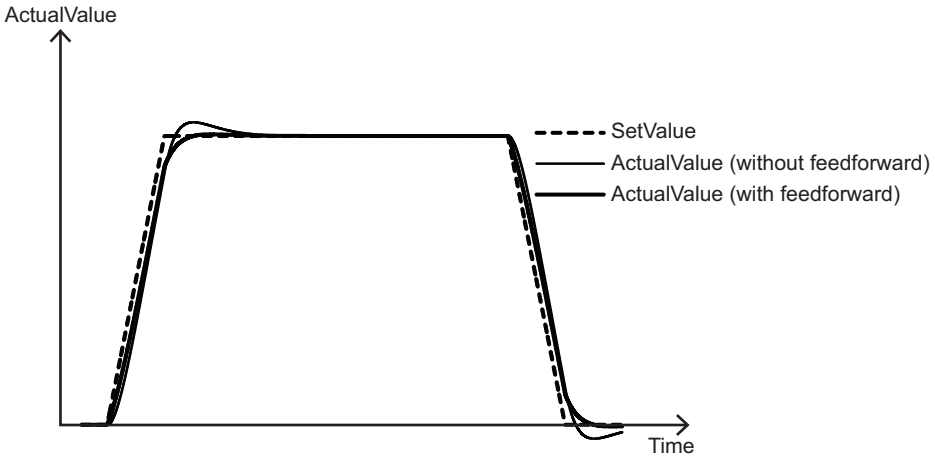


● Increasing Set Point Following Characteristics

The following figure provides an example of adjustments to increase the set point following characteristics. The Feedforward Input Value (*FFValue*) is set to the derivative of the set point.

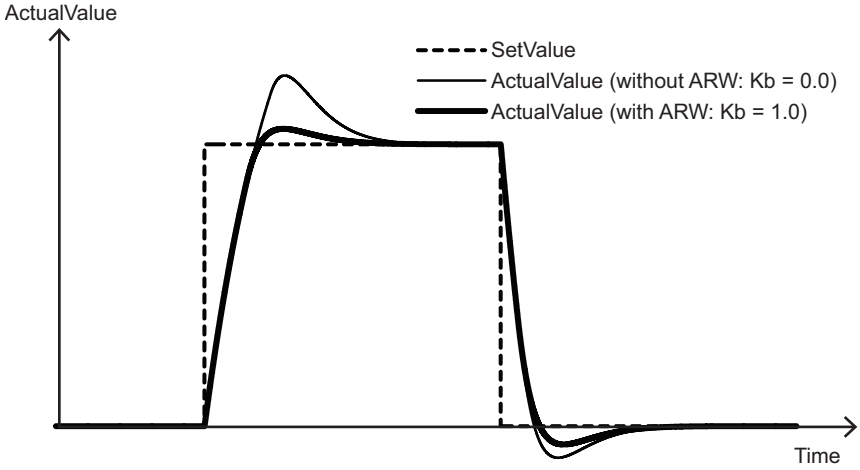
You can see that if you use the Feedforward Input Value (*FFValue*), the deviation between the Set Point (*SetValue*) and the Process Value (*ActualValue*) is smaller even if you use the same PID parameters.

Also, 0.0 is input for the derivative gain (Kd) in the PID processing to increase the set point following characteristics.



Anti-reset Windup Processing

If there is a large deviation between the Set Point (*SetValue*) and the Process Value (*ActualValue*) for an extended period of time, large overshooting or undershooting may occur. This is called reset windup and it is caused by an excessive accumulation of the I (integral) processing output that exceeds the processing output limit. To prevent this, anti-reset windup (ARW) processing is used.



Precautions for Correct Use

- Depending on the parameter settings for this function block, unstable characteristics may occur in which the output value greatly exceeds the input value. Make sure that you understand the characteristics of PID control, perform sufficient theoretical investigations, and take other considerations for the output and safety.
- To limit any unstable characteristics, set the Output Lower Limit (*CalcRsltLowLmt*) and Output Upper Limit (*CalcRsltUpLmt*) correctly.

Troubleshooting

Error code	Expansion error code	Status	Description	Correction
16#0000	16#00000000	Normal End	---	---
16#3C0A	16#00000001	PIDFeedFwd Processing Period Input Value Out of Range	The *SampTime* input parameter for this function block exceeded the valid range for the input variable.	Correct the value set for *SampTime* so that it is within the valid range.
16#3C0A	16#00000002	PIDFeedFwd Integration Time Constant Out of Range	The *PIDFFInitParams.Ti* input parameter for this function block exceeded the valid range for the input variable.	Correct the value set for *PIDFFInitParams.Ti* so that it is within the valid range.
16#3C0A	16#00000003	PIDFeedFwd Derivative Time Constant Out of Range	The *PIDFFInitParams.Td* input parameter for this function block exceeded the valid range for the input variable.	Correct the value set for *PIDFFInitParams.Td* so that it is within the valid range.
16#3C0A	16#00000004	PIDFeedFwd Integral Subtraction Gain Out of Range	The *PIDFFInitParams.Kb* input parameter for this function block exceeded the valid range for the input variable.	Correct the value set for *PIDFFInitParams.Kb* so that it is within the valid range.
16#3C0A	16#00000005	PIDFeedFwd Proportional Gain Out of Range	The *PIDFFOprParams.Kp* input parameter for this function block exceeded the valid range for the input variable.	Correct the value set for *PIDFFOprParams.Kp* so that it is within the valid range.
16#3C0A	16#00000006	PIDFeedFwd Integral Gain Out of Range	The *PIDFFOprParams.Ki* input parameter for this function block exceeded the valid range for the input variable.	Correct the value set for *PIDFFInitParams.Ki* so that it is within the valid range.
16#3C0A	16#00000007	PIDFeedFwd Derivative Gain Out of Range	The *PIDFFOprParams.Kd* input parameter for this function block exceeded the valid range for the input variable.	Correct the value set for *PIDFFInitParams.Kd* so that it is within the valid range.
16#3C0A	16#00000008	PIDFeedFwd Output Limit Value Out of Range	• The relationship between *PIDFFOprParams.AoutLowLm* and *PIDFFOprParams.AoutUpLmt* did not meet required conditions. • Nonnumeric data was input for Output Upper Limit (*CalcRsltUpLmt*) or Output Lower Limit (*CalcRsltLowLmt*).	• Correct the relationship so that the Output Upper Limit (*CalcRsltUpLmt*) is equal to or greater than the Output Lower Limit (*CalcRsltLowLmt*). • Set the Output Upper Limit (*CalcRsltUpLmt*) and Output Lower Limit (*CalcRsltLowLmt*) to correct real numbers.

Sample Programming

This sample programming makes a servo axis (i.e., the real axis) follow the actual current position of an encoder axis.

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 using actual devices, 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.

Conditions

- The encoder axis is *MC_Axis000* and the servo axis is *MC_Axis001*.

Processing

1. Turn ON the servo of the servo axis.
2. Execute the DeadBand function block for the actual current position of the encoder axis.
3. If the servo of the servo axis is ON, execute the FirstOrderLag function block for the result of DeadBand function block execution.
4. Use the result of FirstOrderLag function block execution and the actual current position of the servo axis to execute the PIDFeedFwd function block.
5. If the PIDFeedFwd function block is enabled and the servo axis status is *Standstill*, execute the MC_SyncMoveAbsolute instruction.

   The processing result of the PIDFeedFwd function block is assigned to the MC_SyncMoveAbsolute instruction.
6. If an error occurs in the DeadBand or PIDFeedFwd function block or if the servo turns OFF, execute the MC_ImmediateStop instruction for the servo axis.

Ladder Diagram

● External Variables

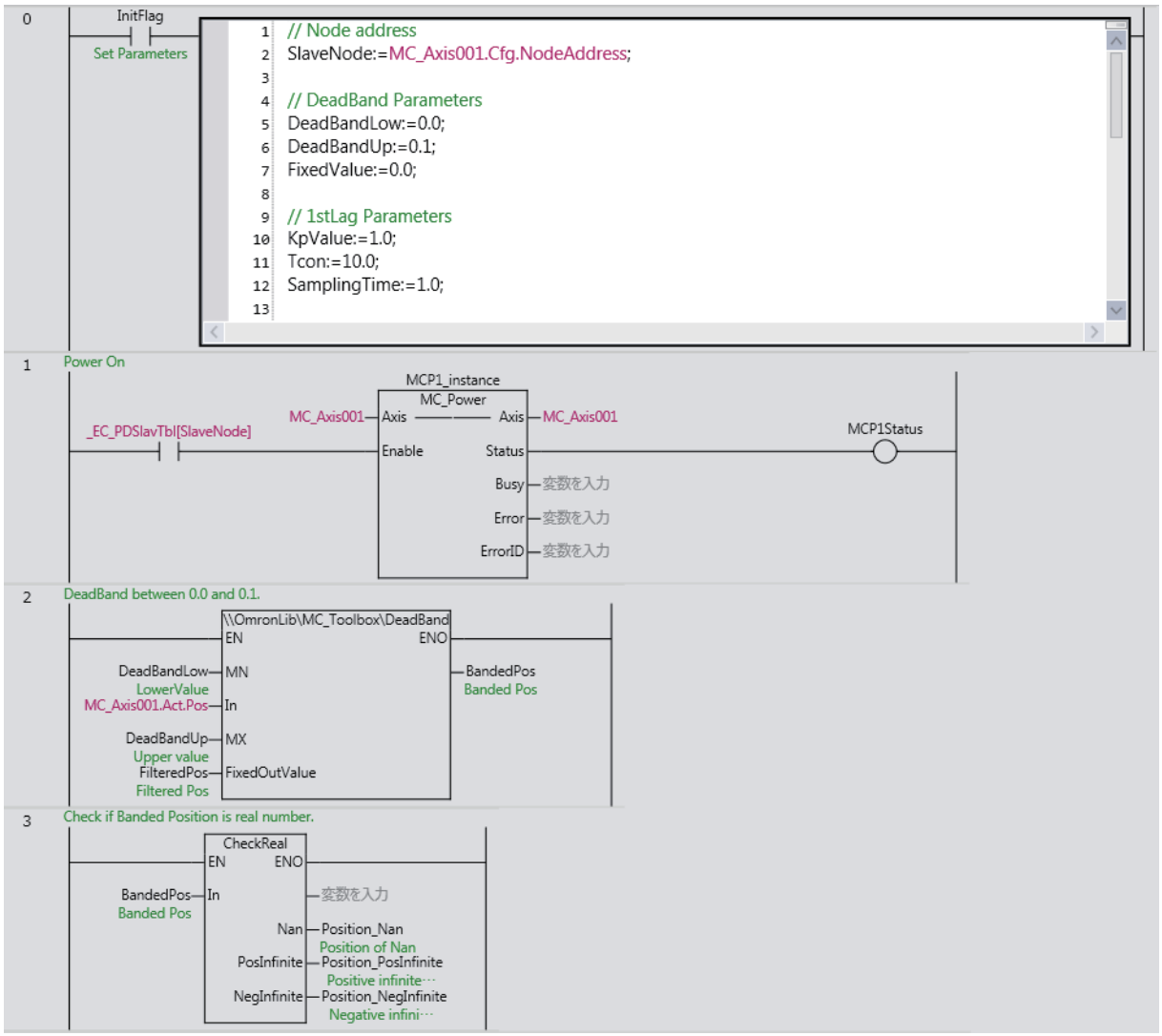
Variable name	Data type	Constant	Comment
MC_Axis000	_sAXIS_REF	✓	Axis 0 (Master Axis)
MC_Axis001	_sAXIS_REF	✓	Axis 1 (Slave Axis)
_EC_PDSlavTbl	ARRAY[1..512] OF BOOL*1	✓	Checking activity of process data communications

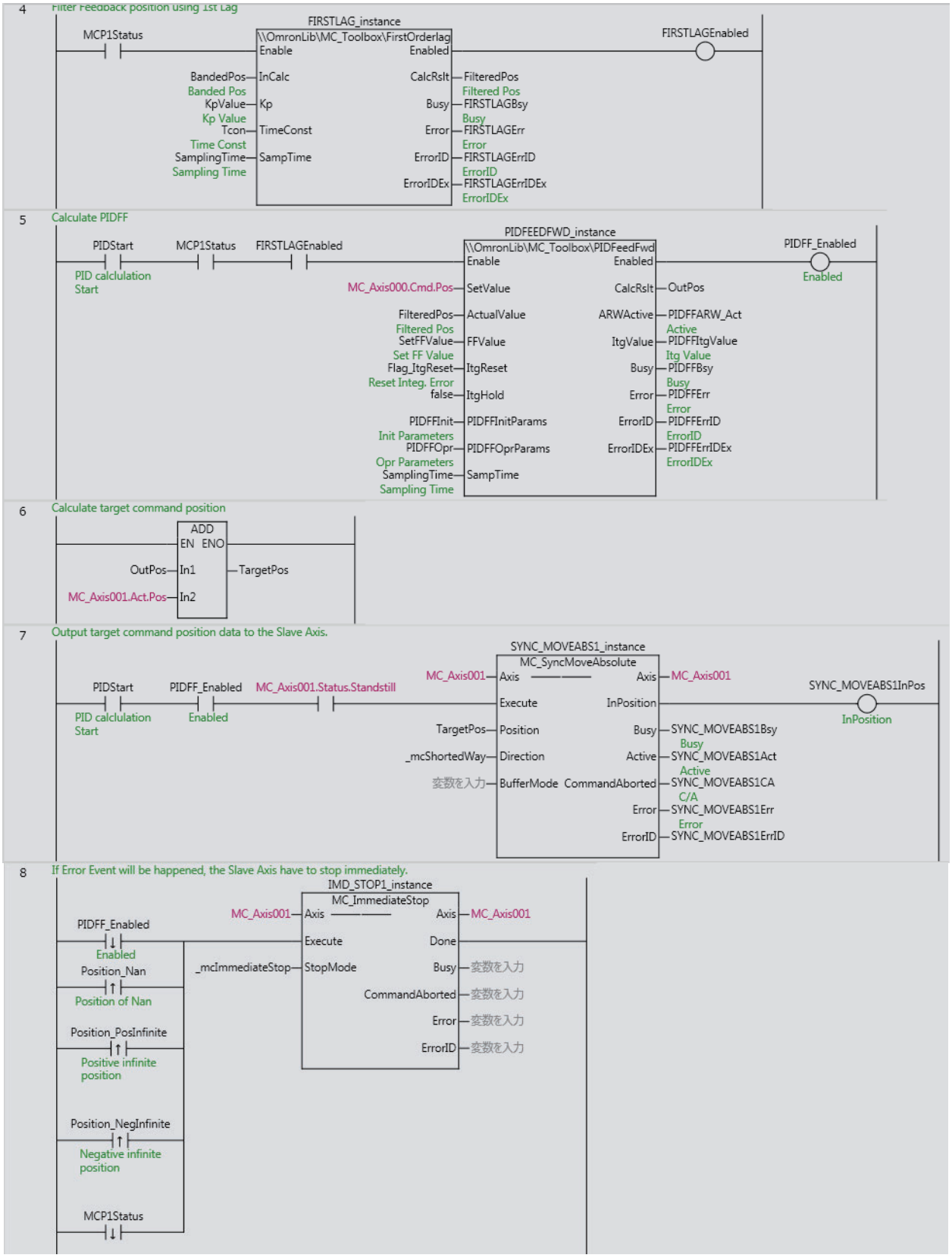
*1. The data type is ARRAY[1..192] OF BOOL for the NJ501-□□□□ or NJ301-□□□□ and ARRAY[1..64] OF BOOL for the NJ101-□□□□.

Internal Variables

Variable name	Data type	Initial value	Comment
PIDStart	BOOL	FALSE	Start of processing
InitFlag	BOOL	FALSE	Parameters initialization
DeadBandLow	LREAL	0.0	Minimum value of deadband
DeadBandUp	LREAL	0.0	Maximum value of deadband
FixedValue	LREAL	0.0	Fixed output value
SetFFValue	LREAL	0.0	Feedforward value
PIDFFInit	OmronLib\MC_Tool-box\sPIDFF_INIT_PARAMS		Initial setting parameters
PIDFFOpr	OmronLib\MC_Tool-box\sPIDFF_O-PR_PARAMS		Operation setting parameters
SamplingTime	LREAL	0.0	Processing period
PIDFF_Enabled	BOOL	FALSE	Enabled
KpValue	LREAL	0.0	Proportional gain
Tcon	LREAL	0.0	First order lag time constant
FilteredPos	LREAL	FALSE	FirstOrderLag processing result
BandedPos	LREAL	0.0	Output value of deadband
OutPos	LREAL	FALSE	PIDFeedFwd processing result
TargetPos	LREAL	0.0	Command target position of servo axis
Flag_ItgReset	BOOL	FALSE	Integral processing reset
MCP1Status	BOOL	FALSE	Servo status

● Programming




4
Filter Feedback position using 1st Lag
FIRSTLAG_instance
\\OmronLib\MC_Toolbox\FirstOrderlag
MCP1Status
Enable
Enabled
FIRSTLAGEnabled
BandedPos
Banded Pos
InCalc
CalcRslt
FilteredPos
Filtered Pos
KpValue
Kp Value
Kp
Busy
FIRSTLAGBsy
Busy
Tcon
Time Const
TimeConst
Error
FIRSTLAGErr
Error
SamplingTime
Sampling Time
SampTime
ErrorID
FIRSTLAGErrID
ErrorID
ErrorIDEx
FIRSTLAGErrIDEx
ErrorIDEx
5
Calculate PIDFF
PIDFEEDFWD_instance
\\OmronLib\MC_Toolbox\PIDFeedFwd
PIDStart
PID calclulation Start
MCP1Status
FIRSTLAGEnabled
Enable
Enabled
PIDFF_Enabled
Enabled
MC_Axis000.Cmd.Pos
SetValue
CalcRslt
OutPos
FilteredPos
Filtered Pos
ActualValue
ARWActive
PIDFFARW_Act
Active
SetFFValue
Set FF Value
FFValue
ItgValue
PIDFFItgValue
Itg Value
Flag_ItgReset
Reset Integ. Error
ItgReset
Busy
PIDFFBsy
Busy
false
ItgHold
Error
PIDFFErr
Error
PIDFFInit
Init Parameters
PIDFFInitParams
ErrorID
PIDFFErrID
ErrorID
PIDFFOpr
Opr Parameters
PIDFFOprParams
ErrorIDEx
PIDFFErrIDEx
ErrorIDEx
SamplingTime
Sampling Time
SampTime
6
Calculate target command position
ADD
EN
ENO
OutPos
In1
TargetPos
MC_Axis001.Act.Pos
In2
7
Output target command position data to the Slave Axis.
SYNC_MOVEABS1_instance
MC_SyncMoveAbsolute
PIDStart
PID calclulation Start
PIDFF_Enabled
Enabled
MC_Axis001.Status.Standstill
MC_Axis001
Axis
Axis
MC_Axis001
Execute
InPosition
SYNC_MOVEABS1InPos
InPosition
TargetPos
Position
Busy
SYNC_MOVEABS1Bsy
Busy
_mcShortedWay
Direction
Active
SYNC_MOVEABS1Act
Active
変数を入力
BufferMode
CommandAborted
SYNC_MOVEABS1CA
C/A
Error
SYNC_MOVEABS1Err
Error
ErrorID
SYNC_MOVEABS1ErrID
8
If Error Event will be happened, the Slave Axis have to stop immediately.
IMD_STOP1_instance
MC_ImmediateStop
PIDFF_Enabled
Enabled
MC_Axis001
Axis
Axis
MC_Axis001
Execute
Done
Position_Nan
Position of Nan
_mcImmediateStop
StopMode
Busy
変数を入力
CommandAborted
変数を入力
Position_PosInfinite
Positive infinite position
Error
変数を入力
ErrorID
変数を入力
Position_NegInfinite
Negative infinite position
MCP1Status

The contents of the inline ST are given below.

```
// Node address
SlaveNode:=MC_Axis001.Cfg.NodeAddress;

// DeadBand Parameters
DeadBandLow:=0.0;
DeadBandUp:=0.1;
FixedValue:=0.0;

// 1stLag Parameters
KpValue:=1.0;
Tcon:=10.0;
SamplingTime:=1.0;

// PIDFF Parameters
SetFFValue:=0.0;
PIDFFInit.Ti:=1000;
PIDFFInit.Td:=1000;
PIDFFInit.Kb:=1.0;
PIDFFOpr.CalcRsltUpLmt:=100000.0;
PIDFFOpr.CalcRsltLowLmt:=-100000.0;
PIDFFOpr.Kd:=0.05;
PIDFFOpr.Kp:=1;
PIDFFOpr.Ki:=2;
```

Structured Text (ST)

External Variables

Variable name	Data type	Constant	Comment
MC_Axis000	_sAXIS_REF	✓	Axis 0 (Master Axis)
MC_Axis001	_sAXIS_REF	✓	Axis 1 (Slave Axis)
_EC_PDSlavTbl	ARRAY[1..512] OF BOOL*1	✓	Checking activity of process data communications

*1. The data type is ARRAY[1..192] OF BOOL for the NJ501-□□□□ or NJ301-□□□□ and ARRAY[1..64] OF BOOL for the NJ101-□□□□.

Internal Variables

Variable name	Data type	Initial value	Comment
PIDStart	BOOL	FALSE	Start of processing
InitFlag	BOOL	FALSE	Parameters initialization
DeadBandLow	LREAL	0.0	Minimum value of deadband
DeadBandUp	LREAL	0.0	Maximum value of deadband
FixedValue	LREAL	0.0	Fixed output value
SetFFValue	LREAL	0.0	Feedforward value
PIDFFInit	OmronLib\MC_Toolbox\sPIDFF_INIT_PARAMS		Initial setting parameters
PIDFFOpr	OmronLib\MC_Toolbox\sPIDFF_OPR_PARAMS		Operation setting parameters
SamplingTime	LREAL	0.0	Processing period
PIDFF_Enabled	BOOL	FALSE	Enabled
KpValue	LREAL	0.0	Proportional gain
Tcon	LREAL	0.0	First order lag time constant
FilteredPos	LREAL	FALSE	FirstOrderLag processing result
BandedPos	LREAL	0.0	Output value of deadband
OutPos	LREAL	FALSE	PIDFeedFwd processing result
TargetPos	LREAL	0.0	Command target position of servo axis
Flag_ItgReset	BOOL	FALSE	Integral processing reset
MCP1Status	BOOL	FALSE	Servo status

Programming

```
// Set Parameters
IF InitFlag THEN
  // Node address
  SlaveNode:=MC_Axis001.Cfg.NodeAddress;

  // DeadBand Parameters
  DeadBandLow:=0.0;
  DeadBandUp:=0.1;
  FixedValue:=0.0;

  // 1stLag Parameters
  KpValue:=1.0;
  Tcon:=10.0;
  SamplingTime:=1.0;
```

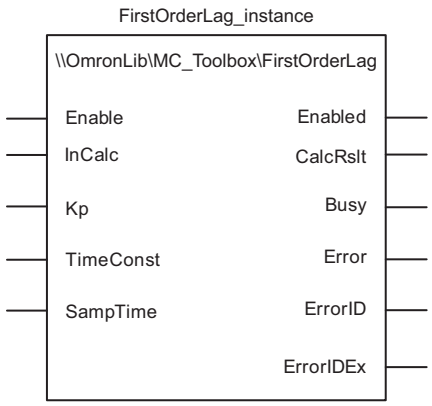
```
  // PIDFF Parameters
  SetFFValue:=0.0;
  PIDFFInit.Ti:=1000;
  PIDFFInit.Td:=1000;
  PIDFFInit.Kb:=1.0;
  PIDFFOpr.CalcRsltUpLmt:=100000.0;
  PIDFFOpr.CalcRsltLowLmt:=-100000.0;
  PIDFFOpr.Kd:=0.05;
  PIDFFOpr.Kp:=1;
  PIDFFOpr.Ki:=2;
END_IF;

// Power ON
IF _EC_PDSlavTbl[SlaveNode] THEN
  MCP1Enable:=TRUE;
ELSE
  MCP1Enable:=FALSE;
END_IF;

//DeadBand between 0.0 and 0.1.
BandedPos:=\\OmronLib\MC_Toolbox\DeadBand(MN:=DeadBandLow,In:=MC_Ax-
is001.Act.Pos,MX:=DeadBandUp,FixedOutValue:=FilteredPos);

//Check if Banded Position is real number.
CheckReal(In:=BandedPos,Nan=>Position_Nan,PosInfinite=>Position_PosInfinite,NegIn-
finite=>Position_PosInfinite);

//Filter Feedback position using 1st Lag,
IF MCP1Status THEN
  FIRSTLAGEnable:=TRUE;
ELSE
  FIRSTLAGEnable:=FALSE;
END_IF;

//Calculate PIDFF
IF PIDStart AND MCP1Status AND FIRSTLAGEnabled THEN
  PIDFFEnable:=TRUE;
ELSE
  PIDFFEnable:=FALSE;
END_IF;

// Calculate Target Postion
TargetPos:=MC_Axis001.Act.Pos+OutPos;

//Output position data to the Slave Axis.
IF PIDStart AND PIDFF_Enabled AND MC_Axis001.Status.Standstill THEN
  SYNC_MOVEABS1Execute:=TRUE;
ELSE
  SYNC_MOVEABS1Execute:=FALSE;
END_IF;

// If Error Event will be happened, the Slave Axis have to stop immediately.
IF (LastPIDFF_Enabled=TRUE) AND  (PIDFF_Enabled=FALSE) THEN
  IMD_STOP1Execute:=TRUE;
ELSIF (LastPosition_Nan=FALSE) AND (Position_Nan=TRUE) THEN
  IMD_STOP1Execute:=TRUE;
ELSIF  (LastPosition_PosInfinite=FALSE) AND (Position_PosInfinite=TRUE) THEN
  IMD_STOP1Execute:=TRUE;
ELSIF (LastPosition_NegInfinite=FALSE) AND (Position_NegInfinite=TRUE) THEN
  IMD_STOP1Execute:=TRUE;
ELSIF (LastMCP1Status=TRUE) AND (MCP1Status=FALSE) THEN
  IMD_STOP1Execute:=TRUE;
ELSE
  IMD_STOP1Execute:=FALSE;
END_IF;
```

```
// Update Last Value
LastPIDFF_Enabled:=PIDFF_Enabled;
LastPosition_Nan:=Position_Nan;
LastPosition_PosInfinite:=Position_PosInfinite;
LastPosition_NegInfinite:=Position_NegInfinite;
LastMCP1Status:=MCP1Status;

// MC_Power
MCP1_instance(
  Axis:=MC_Axis001,
  Enable:=MCP1Enable,
  Status=>MCP1Status
  );

// FirstOrderLag
FIRSTLAG_instance(
  Enable:=FIRSTLAGEnable,
  InCalc:=BandedPos,
  Kp:=KpValue,
  TimeConst:=Tcon,
  SampTime:=SamplingTime,
  Enabled=>FIRSTLAGEnabled,
  CalcRslt=>FilteredPos,
  Busy=>FIRSTLAGBsy,
  Error=>FIRSTLAGErr,
  ErrorID=>FIRSTLAGErrID,
  ErrorIDEx=>FIRSTLAGErrIDEx);

//PIDFeedFwd
PIDFEEDFWD_instance(
  Enable:=PIDFFEnable,
  SetValue:=MC_Axis000.Cmd.Pos,
  ActualValue:=FilteredPos,
  FFValue:=SetFFValue,
  ItgReset:=Flag_ItgReset,
  PIDFFInitParams:=PIDFFInit,
  PIDFFOprParams:=PIDFFOpr,
  SampTime:=SamplingTime,
  Enabled=>PIDFF_Enabled,
  CalcRslt=>OutPos,
  ARWActive=>PIDFFARW_Act,
  ItgValue=>PIDFFItgValue,
  Busy=>PIDFFBsy,
  Error=>PIDFFErr,
  ErrorID=>PIDFFErrID,
  ErrorIDEx=>PIDFFErrIDEx);

// MC_SyncMoveAbsolute
SYNC_MOVEABS1_instance(
  Axis:=MC_Axis001,
  Execute:=SYNC_MOVEABS1Execute,
  Position:=OutPos,
  Direction:=_mcShortedWay,
  Inposition=>SYNC_MOVEABS1InPos,
  Busy=>SYNC_MOVEABS1Bsy,
  Active=>SYNC_MOVEABS1Act,
  CommandAborted=>SYNC_MOVEABS1CA,
  Error=>SYNC_MOVEABS1Err,
  ErrorID=>SYNC_MOVEABS1ErrID);

//MC_ImmediateStop
IMD_STOP1_instance(
  Axis:=MC_Axis001,
  Execute:=IMD_STOP1Execute,
```

```
StopMode:=_mcImmediateStop);
```

FirstOrderLag

The FirstOrderLag function block processes a first order lag according to a specified parameter table.

Function block name	Name	FB/ FUN	Graphic expression	ST expression
FirstOrder-Lag	First Order Lag	FB		FB_FirstOrderLag_instance (Enable, InCalc, Kp, TimeConst, SampTime, Enabled, CalcRslt, Busy, Error, ErrorID, ErrorIDEx);

Function Block and Function Information

Item	Description
Library file name	OmronLib_MC_Toolbox_V1_1.slr
Namespace	OmronLib\MC_Toolbox
Function block and function number	00004
Source code published/not published	Not published
Function block and function version	1.01

Variables

Name	Meaning	I/O	Description	Valid range	Unit	Initial value
Enable	Enable	Input*1	TRUE: Execute FALSE: Stop	TRUE or FALSE	---	FALSE
InCalc	Processing Input	Input*1	Input the value for which to process the first order lag.	Depends on data type.	---	
Kp	Proportional Gain	Input*1	Set the proportional gain.	Depends on data type.	---	1
TimeConst	First Order Lag Time Constant	Input*1	Set the time constant for the first order lag.	0.001 to 100000*2	ms	1.0
SampTime	First Order Lag Processing Period	Input*2	Input the period for which to process the first order lag.	0.001 to 100000.0*2*3	ms	1.0
Enabled	Enabled	Output	Outputs TRUE in any period in which *CalcRslt* (Processing Result) is updated.	TRUE or FALSE	---	---
CalcRslt	Processing Result	Output	Outputs the result of first order lag processing.	Depends on data type.	---	---
Busy	Busy	Output	Indicates when processing is in progress.	Depends on data type.		
Error	Error End	Output	Outputs TRUE while there is an error.	*4		
ErrorID	Error Code	Output	Contains the error code when an error occurs.	*4		
ErrorIDEx	Expansion Error Code	Output	Contains the expansion error code when an error occurs.	*4		

*1. Any changes made during execution of this function block are applied to the output results in the same control period.

*2. The set values in the task period in which *Enable* to this function block changes to TRUE are used in processing. Values that change while *Enabled* is TRUE are not applied to processing.

*3. Any settings below 0.001 (ms) are truncated.

*4. Refer to *Troubleshooting* on page 76 for details.

	Boolean	Bit strings				Integers								Real numbers		Times, durations, dates, and text strings				
	BOOL	BYTE	WORD	DWOR	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Enable	OK																			
InCalc															OK					
Kp															OK					
TimeConst															OK					
SampTime															OK					
Enabled	OK																			
CalcRslt															OK					
Busy	OK																			
Error	OK																			
ErrorID			OK																	

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

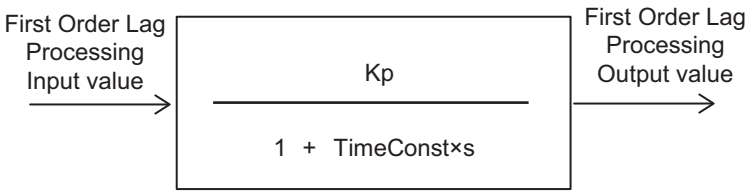
Precautions for Correct Use

- If you use this function block to calculate control values, confirm that *Enabled* is TRUE before you use the Processing Result (*CalcRslt*) as a control value.
- Until this function block is executed the first time, 0.0 is output for the Processing Result (CalcRslt). After this function block is executed, the most recent valid output value is always output to the Processing Result (*CalcRslt*).

  For example, if *Enable* to this function block is changed between TRUE and FALSE, the most recent valid output value for TRUE is output to the Processing Result (*CalcRslt*) while *Enable* is FALSE. Even if this function block ends in an error during execution, the most recent valid output value is output to the Processing Result (*CalcRslt*).

Function

The FirstOrderLag function block processes a first order lag according to a parameter table that is set in advance. The first order lag processing is expressed as follows if a Laplace operator is used:

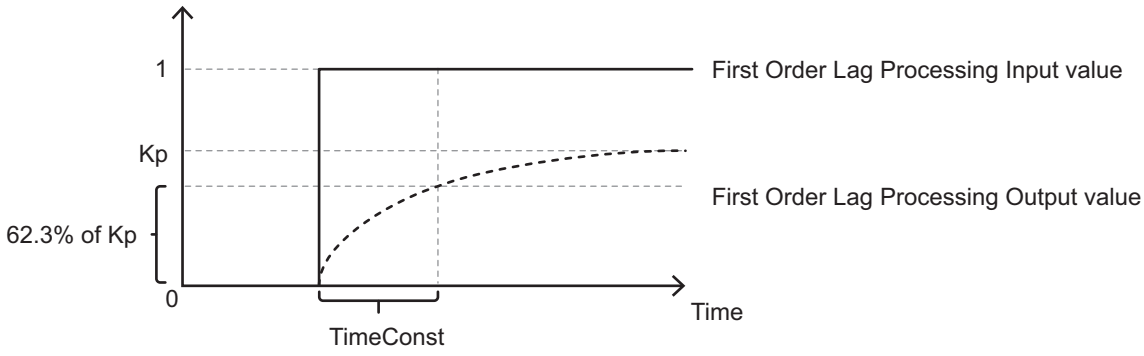


Kp: Proportional gain

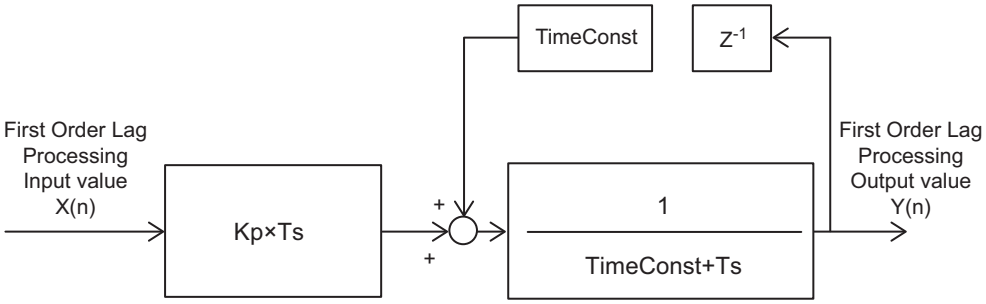
TimeConst: First order lag time constant

s: Laplace operator

With the first order lag processing, when the input value changes, a value that reaches approximately 63.2% of the input value multiplied by the proportional gain is output after the first order lag time constant T (ms).



In the above function block, the Laplace operator s is approximated with a backward difference and the following processing is performed.



However, the initial value of the first order lag processing output value, Y(0), will be Kp × X(0).

Ts: First order lag processing execution period
Refer to *Execution Timing of First Order Lag Processing* on page 63 for details.

Z-1: Lag operator

X(n): First Order Lag Processing Input value at time n

Y(n): First Order Lag Processing Output value at time n

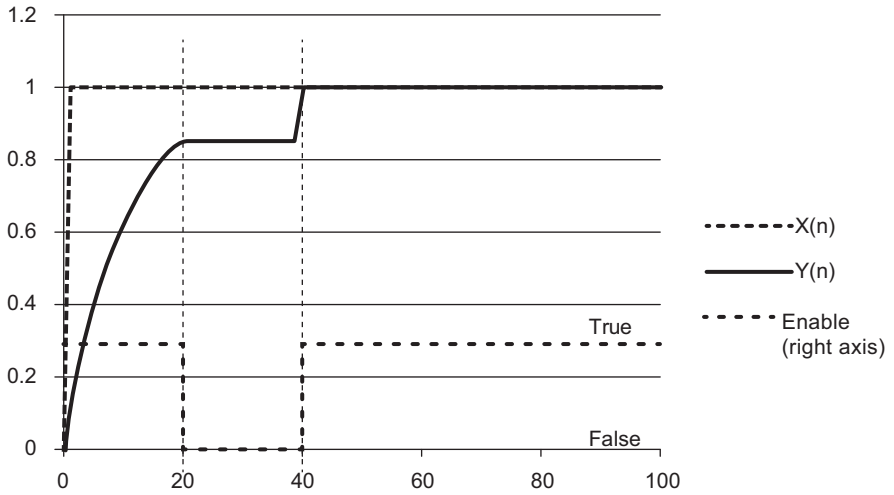
Meanings of Variables

The meanings of the variables that are used in this function block are described below.

● Enable (Execution Condition)

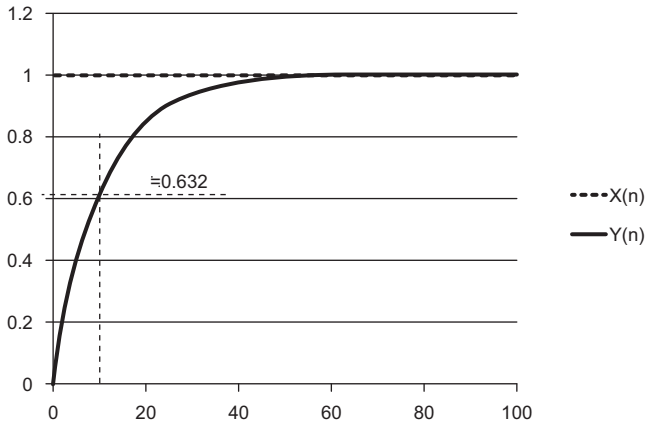
This is the execution condition for the function block. First order lag processing is performed while the value is TRUE. First order lag processing is stopped when the value changes to FALSE.

In the example in the following figure, the output values are shown when the input value changes in stepwise fashion and the value of *Enable* changes to FALSE during first order lag processing. Here, first order lag processing stops from the period in which *Enable* changes to FALSE and the most recent value of Processing Result (*CalcRslt*) is held. When *Enable* changes back to TRUE, first order lag processing for the current input value is started again.



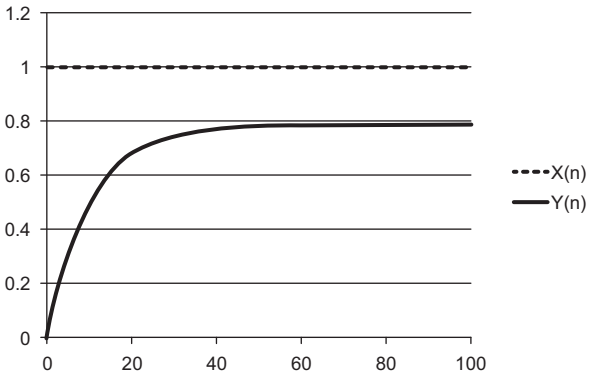
● TimeConst (First Order Lag Time Constant)

The first order lag time constant is the time required for the output value to reach approximately 63.2% of the final value for a stepwise input value. The following figure shows the output results for the following values: *Kp* = 1.0, *SampTime* = 1.0 (ms), and *TimeConst* = 10.0 (ms). The output value reaches approximately 63.2% of the input value 10 ms after the input value changes in stepwise fashion.



Kp (Proportional Gain)

The proportional gain tells the factor to multiply a stepwise input by to find the final output value. The following figure shows the output results for the following values: *Kp* = 0.8, *SampTime* = 1.0 (ms), and *TimeConst* = 10.0 (ms). The final output value will be approximately 0.8 times the input value.



SampTime (Sampling Period)

This is the minimum value of the period for first order lag processing. Refer to *Execution Timing of First Order Lag Processing* on page 63 for details.

Enabled

Enabled will be TRUE in periods in which first order lag processing is executed correctly.

Execution Timing of First Order Lag Processing

The execution timing for first order lag processing is determined by the Processing Period (*SampTime*) and the task period in which this function block is executed. The relationship is as follows:

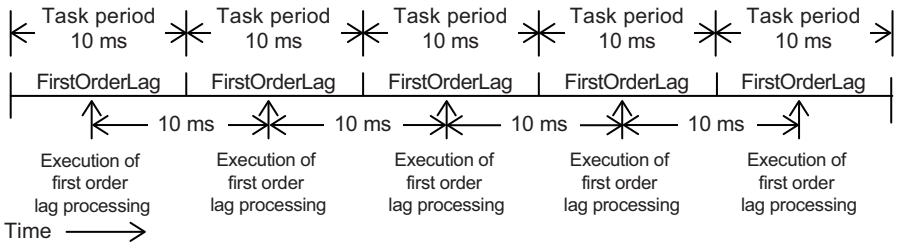
- Processing Period (SampTime) ≤ Task period

  First order lag processing is executed in each task period.
- Processing Period (SampTime) > Task period

  First order lag processing is executed in the integer multiple of the task period that is longer than the Processing Period (*SampTime*).

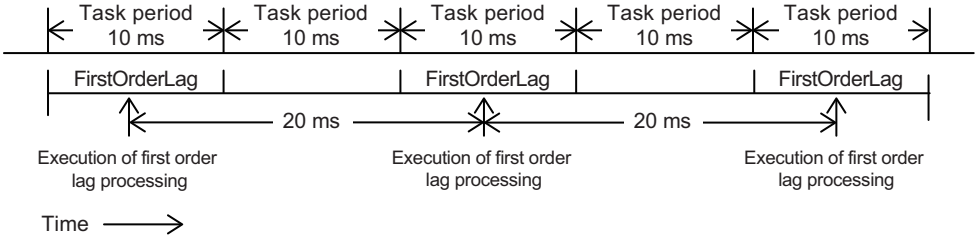
Task Period = 10 ms and *SampTime* ≤ 10 ms

SampTime is less than or equal to the task period, so first order lag processing is executed once every task period.



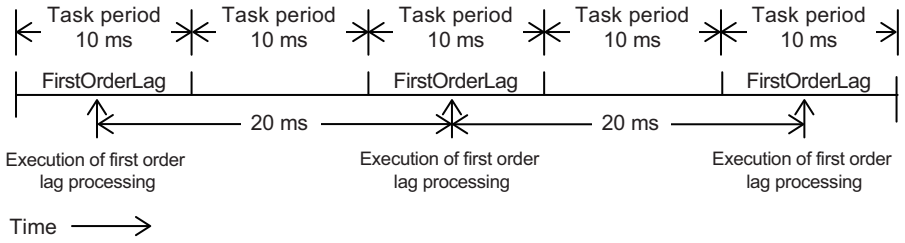
● Task Period = 10 ms and *SampTime* = 11 ms

First order lag processing is executed every 20 ms, i.e., the integer multiple of the task period that is longer than the Processing Period (*SampTime*).



● Task Period = 10 ms and *SampTime* = 19 ms

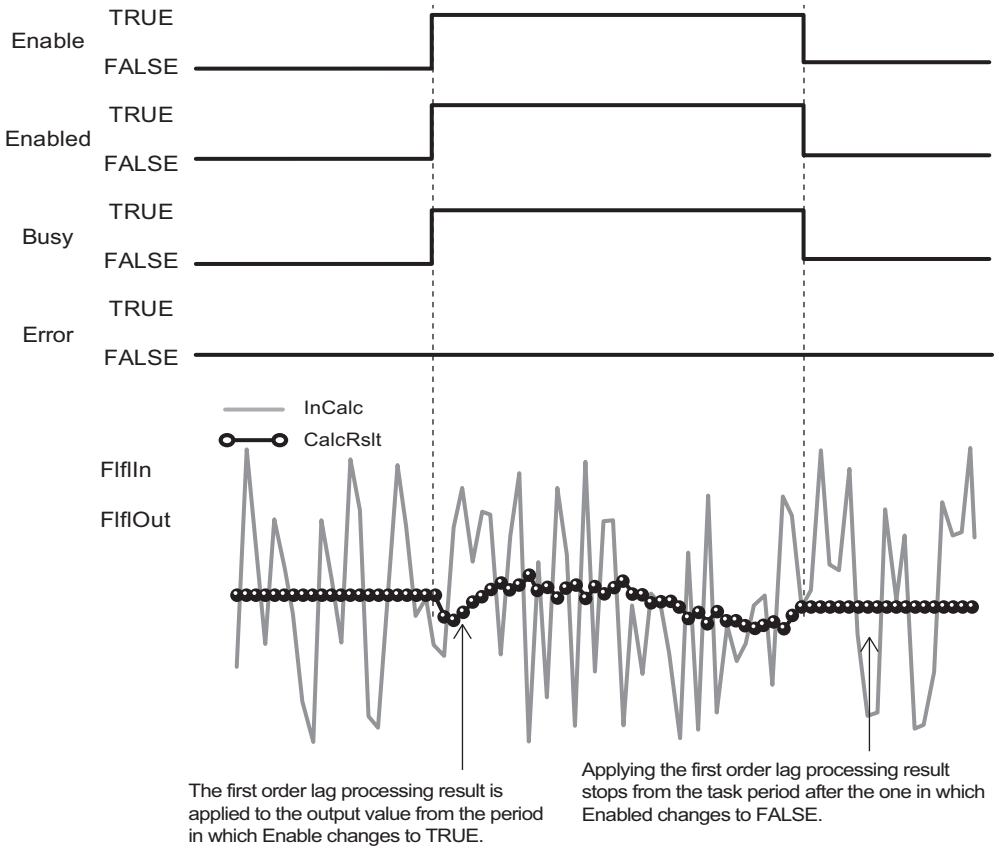
First order lag processing is executed every 20 ms, i.e., the integer multiple of the task period that is longer than the Processing Period (*SampTime*).



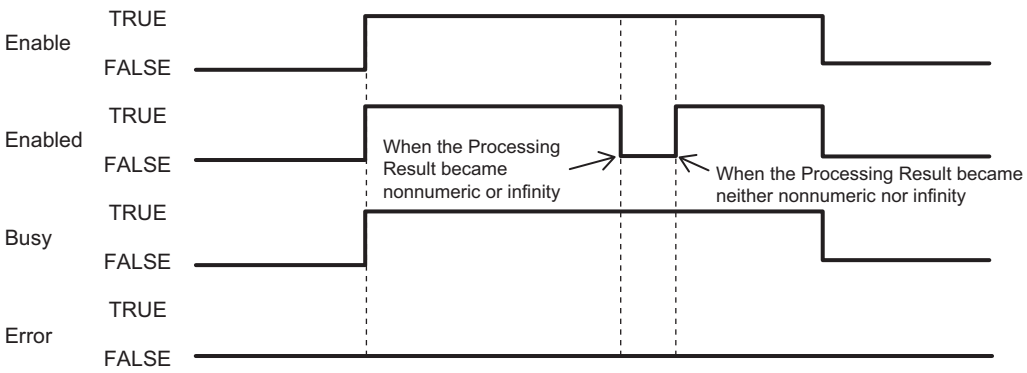
Timing Charts

- When *Enable* changes to TRUE, *Enabled* and *Busy* (Executing) change to TRUE and first order lag processing is performed for the input value.
- When *Enable* changes to FALSE, *Enabled* and *Busy* (Executing) change to FALSE.

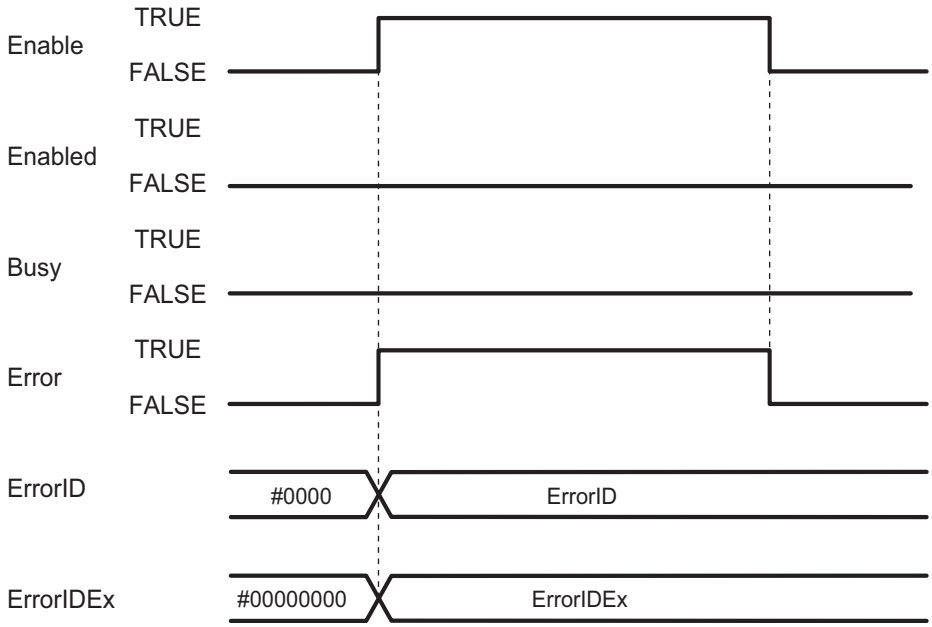
  When the value of *Busy* (Executing) changes to FALSE, first order lag processing is stopped and First Order Lag Processing Result (*CalcRslt*) is no longer updated.
- The following figure shows the output result for the First Order Lag Processing Result (*CalcRslt*) when random values are input to the First Order Lag Processing Input Value (*InCalc*).



- If the Processing Result (*CalcRslt*) of this function block is nonnumeric or infinity, *Enabled* will change to FALSE. When the Processing Result becomes neither nonnumeric nor infinity, *Enabled* will change to TRUE.



- If an error occurs during function block execution, *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). When *Enable* to this function block changes to TRUE, *ErrorID* (Error Code) and *ErrorIDEx* (Expansion Error Code) are cleared.

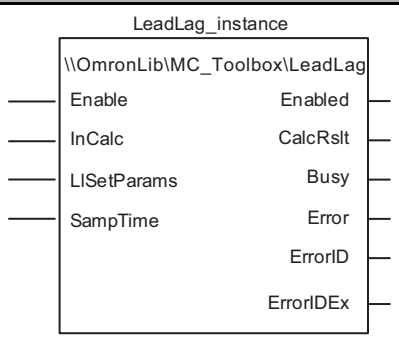


Troubleshooting

Error code	Expansion error code	Status	Description	Correction
16#0000	16#00000000	Normal End	---	---
16#3C0B	16#00000001	First Order Lag Processing Period Input Value Out of Range	The *SampTime* input parameter for this function block exceeded the valid range for the input variable.	Correct the value set for *SampTime* so that it is within the valid range.
16#3C0B	16#00000002	Phase Lead Lag Time Constant Input Value Out of Range	The *TimeConst* input parameter for this function block exceeded the valid range for the input variable.	Correct the value set for *TimeConst* so that it is within the valid range.

LeadLag

The LeadLag function block performs phase lead lag processing according to a specified parameter table.

Function block name	Name	FB/ FUN	Graphic expression	ST expression
LeadLag	Phase Lead Lag	FB		LeadLag_instance (Enable, InCalc, LlSetParams, SampTime, Enabled, CalcRslt, Busy, Error, ErrorID, ErrorIDEx);

Function Block and Function Information

Item	Description
Library file name	OmronLib_MC_Toolbox_V1_1.slr
Namespace	OmronLib\MC_Toolbox
Function block and function number	00005
Source code published/not published	Not published
Function block and function version	1.01

Variables

Name	Meaning	I/O	Description	Valid range	Unit	Initial value
Enable	Enable	Input*1	TRUE: Execute FALSE: Stop	TRUE or FALSE	---	FALSE
InCalc	Phase Lead Lag Processing Input	Input*1	Input the value for which to process the phase lead lag.	Depends on data type.	---	0.0
LlSetParams	Phase Lead Lag Setting Parameters	Input*1	The setting parameters for processing the phase lead lag	Depends on data type.		0.0
SampTime	Phase Lead Lag Processing Period	Input*2	Set the period for processing the phase lead lag.	0.001 to 100000.0*3	ms	1.0
Enabled	Enabled	Output	Outputs TRUE in any period in which *CalcRslt* (Processing Result) is updated.	TRUE or FALSE	---	---
CalcRslt	Phase Lead Lag Processing Output	Output	Outputs the result of phase lead lag processing.	Depends on data type.		---
Busy	Busy	Output	Indicates when processing is in progress.	TRUE or FALSE		---
Error	Error End	Output	Outputs TRUE while there is an error.	TRUE or FALSE		
ErrorID	Error Code	Output	Contains the error code when an error occurs.	*4		
ErrorIDEx	Expansion Error Code	Output	Contains the expansion error code when an error occurs.	*4		

*1. Any changes made during execution of this function block are applied to the output results in the same control period.

*2. The set values in the task period in which *Enable* to this function block changes to TRUE are used in processing. Values that change while *Enabled* is TRUE are not applied to processing.

*3. Any settings below 0.001 (ms) are truncated.

*4. Refer to *Troubleshooting* on page 76 for details.

	Boolean	Bit strings				Integers								Real numbers		Times, durations, dates, and text strings				
	BOOL	BYTE	WORD	DWOR	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Enable	OK																			
InCalc															OK					
LlSetParams	Refer to *Function* for details on the structure OmronLib\MC_Toolbox/sLL_SET_PARAMS.																			
SampTime															OK					
Enable	OK																			
CalcRslt															OK					
Busy	OK																			
Error	OK																			

	Boolean	Bit strings				Integers								Real numbers		Times, durations, dates, and text strings				
	BOOL	BYTE	WORD	DWOR	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
ErrorID			OK																	
ErrorIDEx				OK																

Precautions for Correct Use

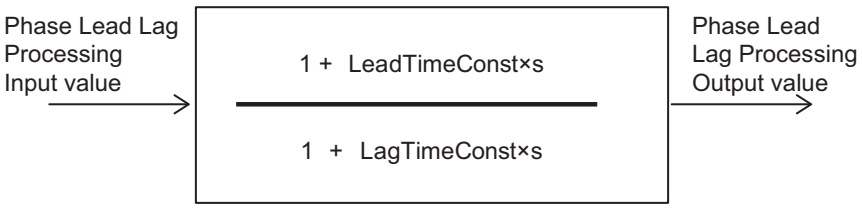
- If you use this function block to calculate control values, confirm that *Enabled* is TRUE before you use the Processing Result (*CalcRslt*) as a control value.
- Until this function block is executed the first time, 0.0 is output for the Processing Result (*CalcRslt*). After this function block is executed, the most recent valid output value is always output to the Processing Result (*CalcRslt*).

  For example, if *Enable* to this function block is changed between TRUE and FALSE, the most recent valid output value for TRUE is output to the Processing Result (*CalcRslt*) while *Enable* is FALSE. Even if this function block ends in an error during execution, the most recent valid output value is output to the Processing Result (*CalcRslt*).

Function

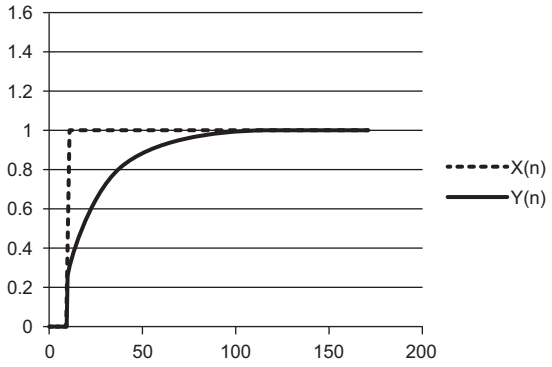
The LeadLag function block processes a phase lead lag according to a parameter table that is set in advance.

The phase lead lag processing is expressed as follows if the Laplace operator s is used:



s: Laplace operator

You can use a phase lead lag to create filters with various characteristics. The following figure shows the Phase Lead Lag Processing Output (*CalcRslt*) value that is set for a phase lead lag characteristic for a stepwise input value on the Phase Lead Lag Processing Input (*InCalc*).



(LeadTimeConst < LagTimeConst)

In the above function block, the Laplace operator s is approximated with a backward difference and the following processing is performed.

$$Y(n) = \frac{1}{A}\{X(n) \times B - X(n-1) \times LagTimeConst + Y(n-1) \times LagTimeConst\}$$

n = 0, 1, 2, …

A = LagTimeConst + Ts

B = LeadTimeConst + Ts

X(n): First Order Lag Processing Input value at time n

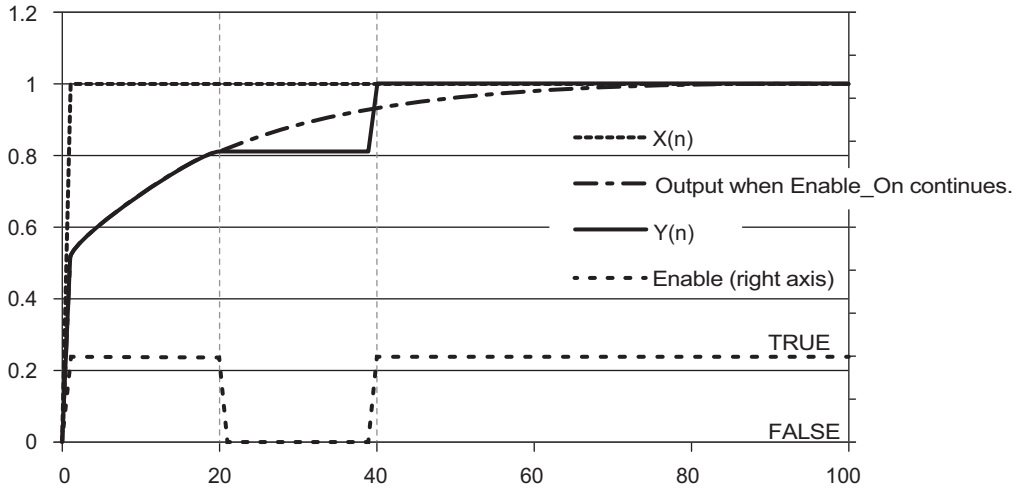
Y(n): First Order Lag Processing Output value at time n

However, the initial value of the phase lead lag processing output value, Y(0), will be X(0).

Ts: Phase lead lag processing execution period
Refer to *Execution Timing of Phase Lead Lag Processing* on page 71 for details.

When *Enable* to this function block changes to TRUE, phase lead lag processing is performed. Phase lead lag processing is stopped when the value changes to FALSE.

In the example in the following figure, the output values are shown when the input value changes in stepwise fashion and *Enable* changes to FALSE during phase lead lag processing. Here, phase lead lag processing stops from the period in which *Enable* changes to FALSE and the most recent value of Processing Result (*CalcRslt*) is held.When *Enable* changes back to TRUE, phase lead lag processing for the current input value is started again.



Structures

The data type of the Phase Lead Lag Setting Parameters (*LlSetParams)* is structure sLL_SET_PARAMS. The specifications are as follows:

Variable or member	Name	Description	Data type	Valid range	Unit	Initial value
LlSetParams	Phase Lead Lag Setting Parameters		Omron-Lib\MC_Toolbox/sLL_SET_PARAMS	---	---	
LeadTime-Const	Phase Lead Time Constant	Specify the phase lead time constant.	LREAL	0.001 to 100000*1	ms	1.0
LagTime-Const	Phase Lag Time Constant	Specify the phase lag time constant.	LREAL	0.001 to 100000*1	ms	1.0

*1. Any settings below 0.001 (ms) are truncated.

Execution Timing of Phase Lead Lag Processing

The execution timing for phase lead lag processing is determined by the Processing Period (*SampTime*) and the task period in which this function block is executed. The relationship is as follows:

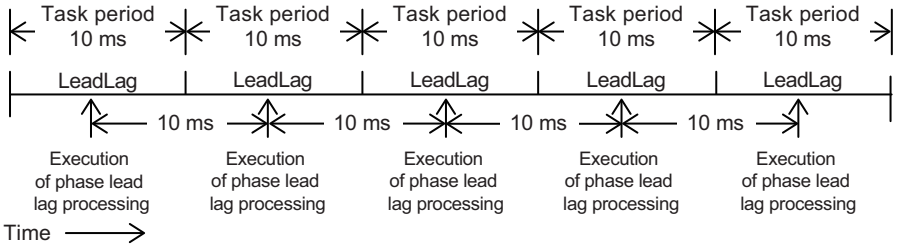
- Processing Period (SampTime) ≤ Task period

  Phase lead lag processing is executed in each task period.
- Processing Period (SampTime) > Task period

  Phase lead lag processing is executed in the integer multiple of the task period that is longer than the Processing Period (*SampTime*).

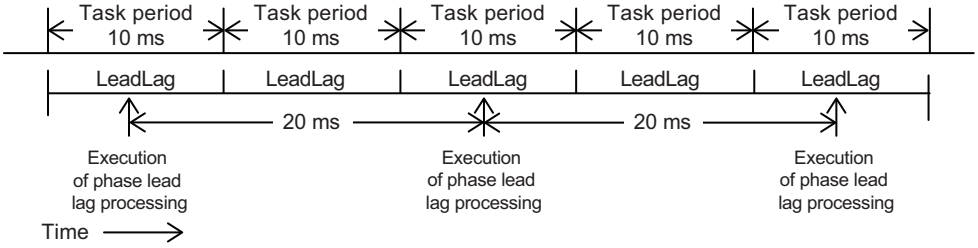
● Task Period = 10 ms and *SampTime* ≤ 10 ms

SampTime is less than or equal to the task period, so PID feedforward processing is executed once every task period.



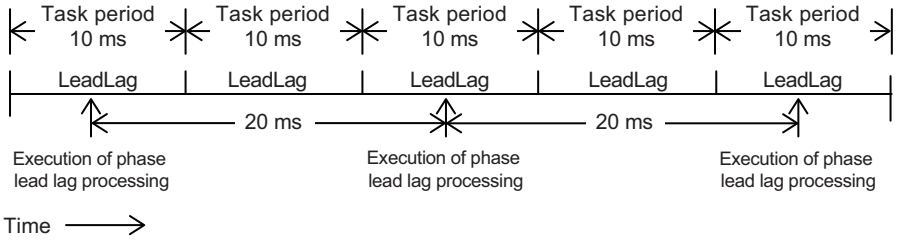
● Task Period = 10 ms and *SampTime* = 11 ms

Phase lead lag processing is executed every 20 ms, i.e., the integer multiple of the task period that is longer than the Processing Period (*SampTime*).



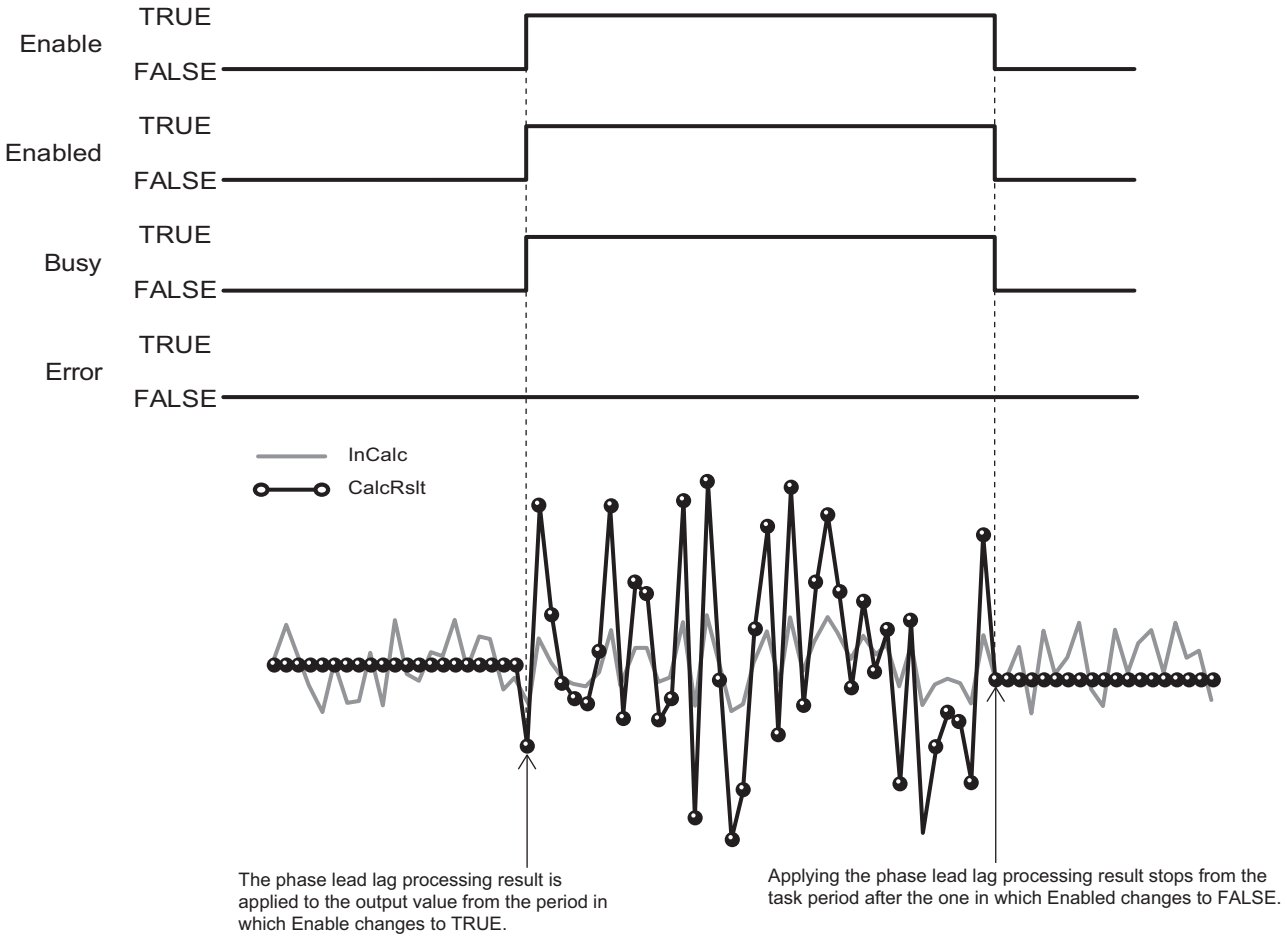
● Task Period = 10 ms and *SampTime* = 19 ms

Phase lead lag processing is executed every 20 ms, i.e., the integer multiple of the task period that is longer than the Processing Period (*SampTime*).

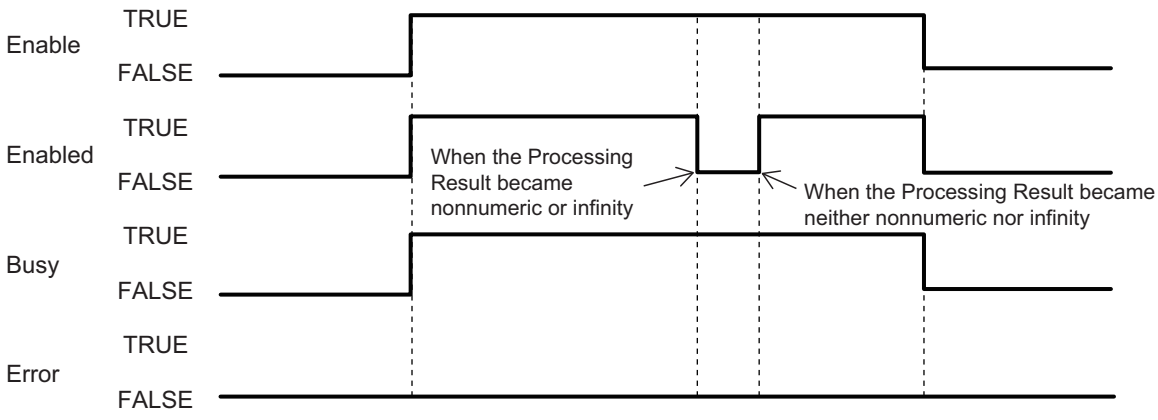


Timing Charts

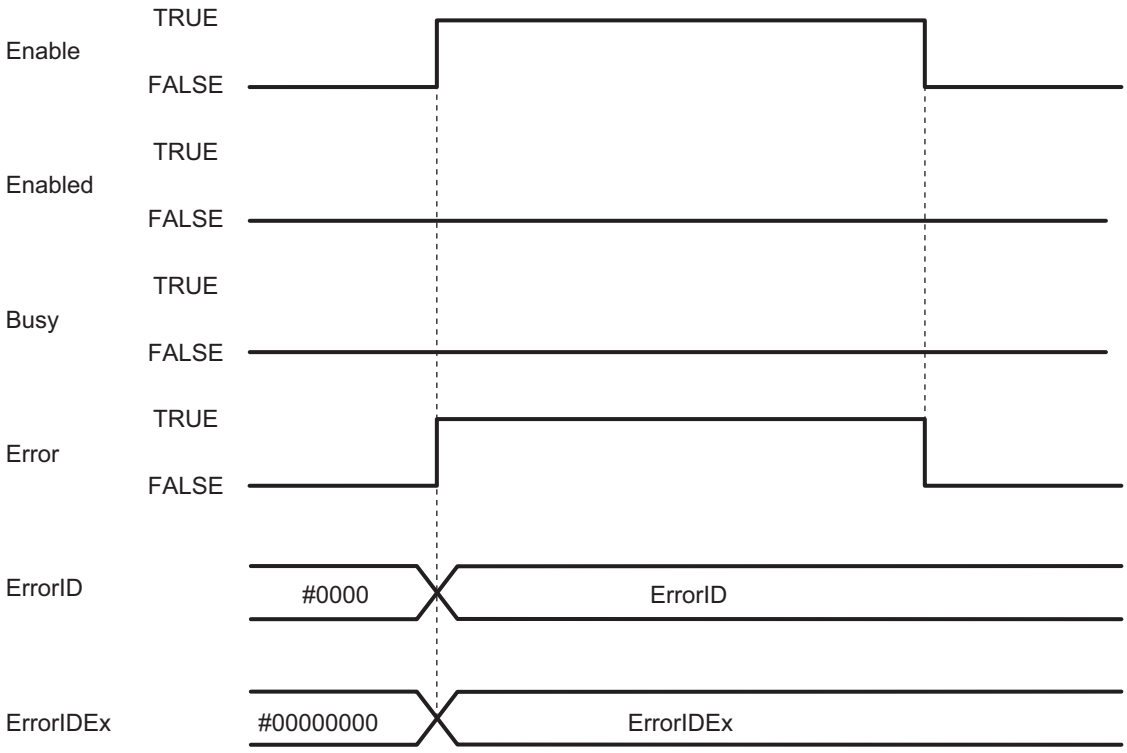
- When *Enable* changes to TRUE, *Enabled* and Busy (Executing) change to TRUE and phase lead lag processing is performed for the input value.
- When *Enable* changes to FALSE, *Enabled* and *Busy* (Executing) change to FALSE. When the value of *Busy* changes to FALSE, phase lead lag processing is stopped and the Phase Lead Lag Processing Result (*CalcRslt*) is no longer updated.
- The following figure shows the output result for the Processing Result (*CalcRslt*) when random values are input to the Processing Input Value (*InCalc*) of this function block (LeadTimeConst = 20.0 and LagTimeConst > 5.0).



- If the Processing Result (*CalcRslt*) of this function block is nonnumeric or infinity, *Enabled* will change to FALSE. When the Processing Result becomes neither nonnumeric nor infinity, *Enabled* will change to TRUE.



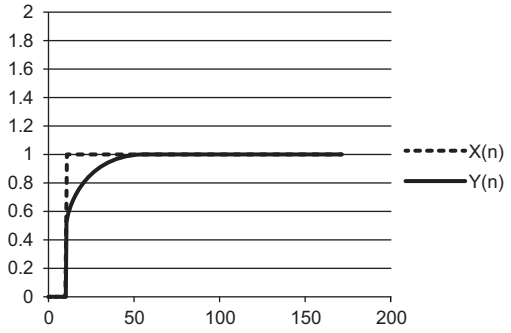
- If an error occurs during function block execution, *Error* will change to TRUE. You can find out the cause of the error by referring to the values output by *ErrorID* (Error Code) and *ErrorIDEx* (Expansion Error Code). When *Enable* to this function block changes to TRUE, *ErrorID* (Error Code) and *ErrorIDEx* (Expansion Error Code) are cleared.



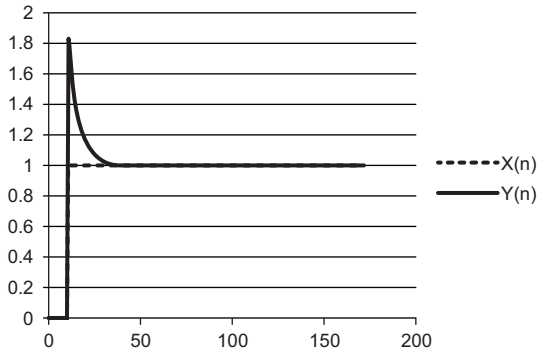
Additional Information

You can change various characteristics by suitably setting the Phase Lead Lag Setting Parameters (*LISetParams*) of this function block. This section provides some diagrams of typical output results for stepwise input values in combination with different Phase Lead Lag Setting Parameters (*LISetParams*).

- **LeadTimeConst < LagTimeConst**

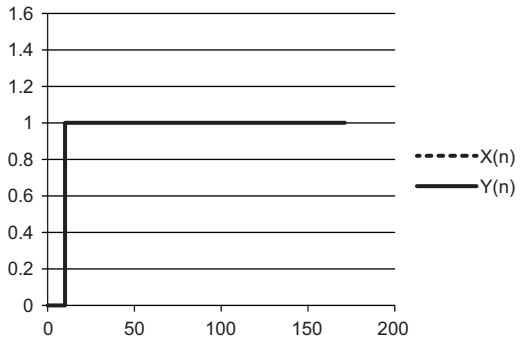


- **LeadTimeConst > LagTimeConst**



- **LeadTimeConst = LagTimeConst**

In this case, the numerator and denominator expressions are the same. Even if you use this function block, the input value and output value will be the same.



Precautions for Correct Use

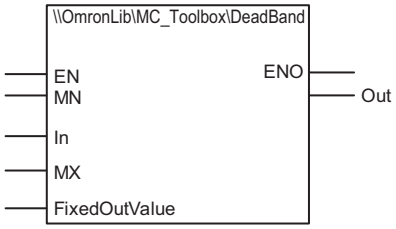
- Depending on the parameter settings for this function block, unstable characteristics may occur in which the output value greatly exceeds the input value. Make sure that you understand the characteristics of a phase lead lag, perform sufficient theoretical investigations, and take other considerations for the output and safety.
- To prevent the calculated output from becoming too large and thereby producing an unstable system, use the Limiter (LIMIT) instruction. Refer to *information on the Limiter (LIMIT) instruction* in the instructions reference manual for details.

Troubleshooting

Error code	Expansion error code	Status	Description	Correction
16#0000	16#00000000	Normal End	---	---
16#3C0C	16#00000001	Phase Lead Lag Processing Period Input Value Out of Range	The *SampTime* input parameter for this function block exceeded the valid range for the input variable.	Correct the value set for *SampTime* so that it is within the valid range.
16#3C0C	16#00000002	Phase Lead Time Constant Input Value Out of Range	The *LISetParams.LeadTimeConst* input parameter for this function block exceeded the valid range for the input variable.	Correct the value set for *LISetParams.LeadTimeConst* so that it is within the valid range.
16#3C0C	16#00000003	Phase Lag Time Constant Input Value Out of Range	The *LISetParams.LagTimeConst* input parameter for this function block exceeded the valid range for the input variable.	Correct the value set for *LISetParams.LagTimeConst* so that it is within the valid range.

DeadBand

The DeadBand function block controls a deadband that does not create an offset with the processing result.

Function block name	Name	FB/ FUN	Graphic expression	ST expression
DeadBand	Deadband Control without Output Offset	FUN		Out:=DeadBand(MN,In,MX,FixedOutValue);

Function Block and Function Information

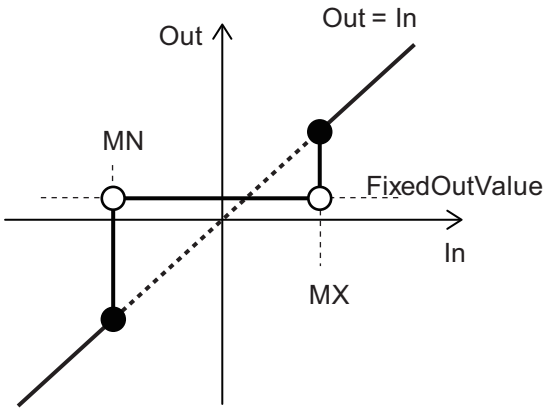
Item	Description
Library file name	OmronLib_MC_Toolbox_V1_1.slr
Namespace	OmronLib\MC_Toolbox
Function block and function number	00006
Source code published/not published	Not published
Function block and function version	1.01

Variables

	Meaning	I/O	Description	Valid range	Unit	Initial value
MN	Minimum Value	Input	Minimum value of deadband	Depends on data type.	---	0.0
In	Input Value		Input value to convert			0.0
MX	Maximum Value		Maximum value of deadband			0.0
FixedOutValue	Fixed Output Value		The value that is output as a fixed value			0.0
Out	Output Value	Output	Processing result	Depends on data type.	---	

	Boolean	Bit strings				Integers								Real numbers		Times, durations, dates, and text strings				
	BOOL	BYTE	WORD	DWOR	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
MN															OK					
In															OK					
MX															OK					
FixedOutValue															OK					
Out															OK					

Function



The input value (*In*) is converted as given in the following table for the Upper Limit (*MX*), the Lower Limit (*MN*), and the Fixed Output Value (*FixedOutValue*).

Value of *In*	Value of *Out*
MX ≤ In	In
MN < In < MX	FixedOutValue
In ≤ MN	In

Precautions for Correct Use

- If the value of *In* is nonnumeric data, the value of *Out* is nonnumeric data. Use the CheckReal (Real Number Check) instruction to make sure that *In* is not infinity or nonnumeric data. Refer to information on the *Real Number Check (CheckReal) instruction* in the instructions reference manual for details.
- If the value of *MN*, *In*, *MX*, or *FixedOutValue* is positive infinity or negative infinity, the value of *Out* is as shown below.

Value of *In*	Value of *MN*	Value of *MX*	Value of *Out*
Positive infinity	Positive infinity	Positive infinity	Positive infinity
		Negative infinity	Error
	Negative infinity	Positive infinity	Positive infinity
		Negative infinity	Positive infinity
Negative infinity	Positive infinity	Positive infinity	Negative infinity
		Negative infinity	Error
	Negative infinity	Positive infinity	Negative infinity
		Negative infinity	Negative infinity

- An error occurs in the following cases. *ENO* will be FALSE, and *Out* will not change.
  a) The value of *MX* is smaller than the value of *MN*.
  b) *MX*, *MN*, or *FixedOutValue* contains nonnumeric data.

Appendix

Referring to Library Information

When you make an inquiry to OMRON about the library, you can refer to the library information to identify the library to ask about.

The library information is useful in identifying the target library among the libraries provided by OMRON or created by the user.

The library information consists of the attributes of the library and the attributes of function blocks and functions contained in the library.

- Attributes of libraries
  Information for identifying the library itself
- Attributes of function blocks and functions
  Information for identifying the function block and function contained in the library

Use the Sysmac Studio to access the library information.

Attributes of Libraries, Function Blocks and Functions

The following attributes of libraries, function blocks and functions are provided as the library information.

● Attributes of Libraries

No.*1	Attribute	Description
(1)	Library file name	The name of the library file
(2)	Library version	The version of the library
(3)	Author	The name of creator of the library
(4)	Comment	The description of the library*2

*1. These numbers correspond to the numbers shown on the screen images in the next section, *Referring to Attributes of Libraries, Function Blocks and Functions* on page 83.

*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 83.

*2. It is provided in English and Japanese.

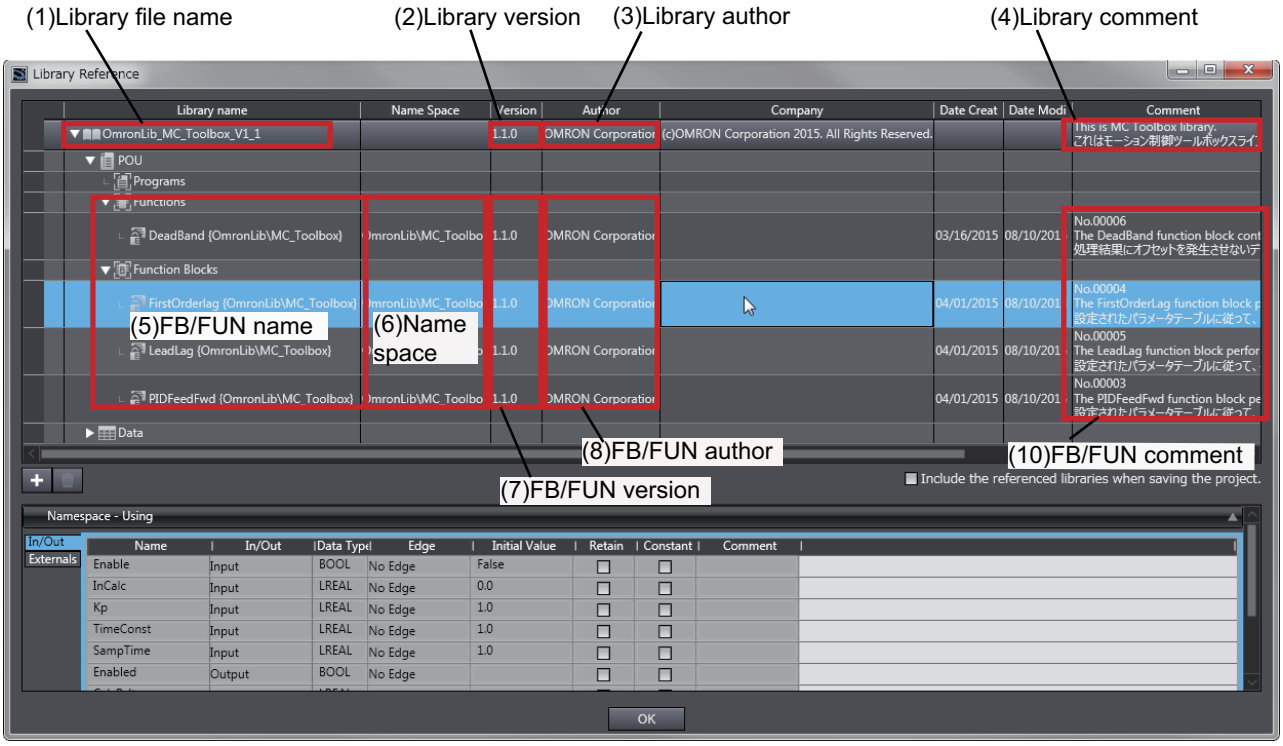
Referring to Attributes of Libraries, Function Blocks and Functions

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

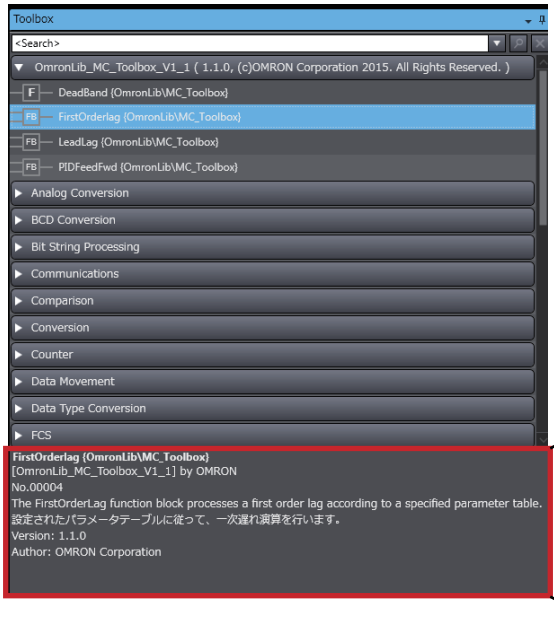
When you refer to the libraries, the library information is displayed at the locations shown below.



(b) Toolbox Pane

Select a function block and function to display its library information at the bottom of the Toolbox Pane.

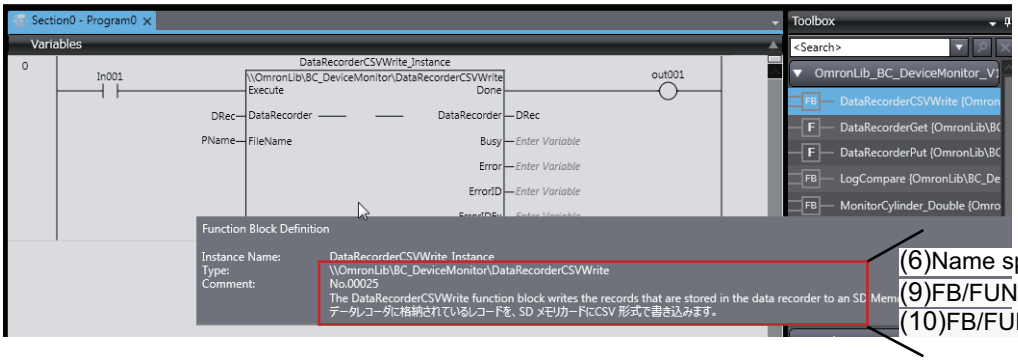
The text “by OMRON” which is shown on the right of the library name (1) indicates that this library was provided by OMRON.



(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.

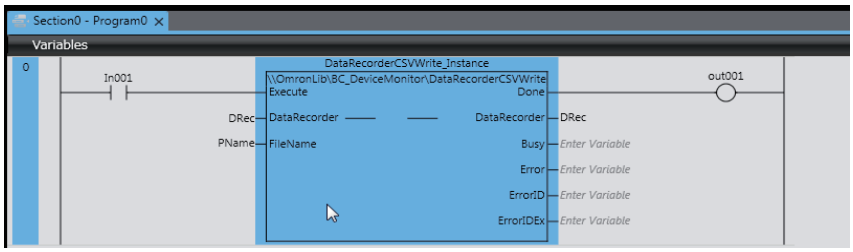
The following are the examples of items that you may need to customize.

- 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.

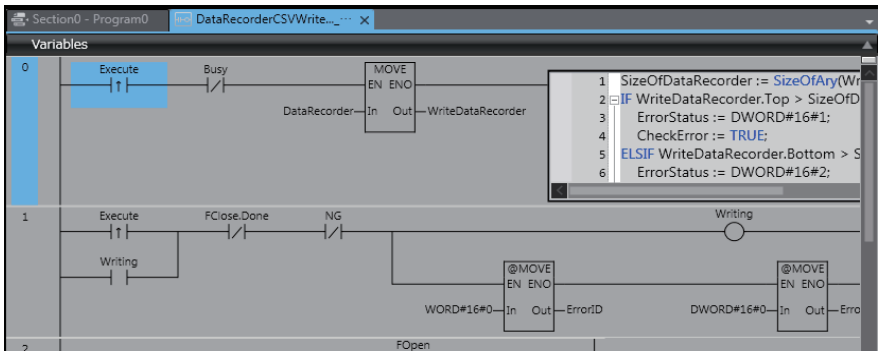
Use the following procedure to refer to the source codes of function blocks and functions.

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:

Cat. No. W547-E1-05 0119