OMRON

# Sysmac Library

## User's Manual
## for Temperature Control Library
### SYSMAC-XR007

SYSMAC
always in control

W551-E1-04

## 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 Temperature Control 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 contains the specifications of the Function Block. It does not include restrictions on use of the Controller, Units, or components, or restrictions due to combinations. Make sure to read the user's manual for each product before use.

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

## Features of the Library

The Temperature Control Library is used to perform a high-level temperature control. You can use this library together with analog control instructions of the NJ/NX/NY-series Controller. Refer to the instructions reference manual for details on analog control instructions.

## 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 Controller and an NY-series Industrial PC.

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

The following notation is used.

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

## Symbols

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

## Warning

### ⚠Warning

| | |
|---|---|
| Correctly make the settings because the DirectPowerControl function block directly output the manipulated variable without feeding back. Not doing so may result in serious accident or fire due to overheating. Monitor overheating and build an application to safely stop the devices if overheating occurs. |  |
| Correctly make the settings because the DirectPowerControl function block directly output the manipulated variable without feeding back. Not doing so may result in serious accident, steam explosion, or frostbite due to overcooling. Monitor overcooling and build an application to safely stop the devices if overcooling occurs. |  |

## Cautions

### ⚠Caution

| | |
|---|---|
| Read all related manuals carefully before you use this library. |  |
| Emergency stop circuits, interlock circuits, limit circuits, and similar safety measures must be provided in external control circuits. |  |
| Check the user program, data, and parameter settings for proper execution before you use them for actual operation. |  |
| The Sysmac Library and manuals are assumed to be used by personnel that is given in Intended Audience in this manual. Otherwise, do not use them. |  |
| Perform the test run by holding an emergency stop switch in hand or otherwise prepare for rapid motor operation in an application to control the motor.<br>Also perform the test run by using the parameters for which the motor does not rapidly accelerate or decelerate before you gradually adjust the parameters. |  |
| In an application of heating or cooling, perform the test run by using the parameters for which rapid temperature changes will not occur before you gradually adjust the parameters. |  |
| You must confirm that the user program and parameter values are appropriate to the specifications and operation methods of the devices. |  |
| The sample programming shows only the portion of a program that uses the function or function block from the library. |  |
| When using actual devices, also program safety circuits, device interlocks, I/O with other devices, and other control procedures. |  |
| Understand the contents of sample programming before you use the sample programming and create the user program. |  |

# Precautions for Correct Use

## Using the Library

- When you use the library, functions or function blocks that are not described in the library manual may be displayed on the Sysmac Studio. Do not use functions or function blocks that are not described in the manual.
- You cannot change the source code of the functions or function blocks that are provided in the Sysmac Library.
- The multi-execution (buffer mode) cannot be performed in the Sysmac Library.
- Provide safety measures such as monitoring overheating or overcooling, and other measures outside the functions or function blocks when you use the Temperature Control Library.

## Using Sample Programming

- Create a user program that will produce the intended device operation.
- Check the user program for proper execution before you use it for actual operation.

## Operation

- Specify the input parameter values within the valid range.
- In the function or function block with an Enabled output variable, if the value of Enabled is FALSE, do not use the processing result of the function or function block as a command value to the control target.
- In the function block with Execute, do not perform re-execution by the same instance. The output value of the function block will return to the default value.

# Related Manuals

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

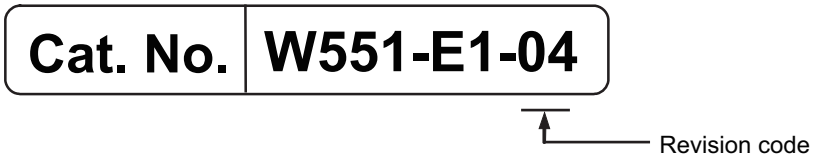
| Manual name | Cat. No. | Model numbers | Application | Description |
|---|---|---|---|---|
| NX-series CPU Unit Hardware User's Manual | W535 | NX701-□□□□ | Learning the basic specifications of the NX-series NX701 CPU Units, including introductory information, designing, installation, and maintenance. Mainly hardware information is provided | An introduction to the entire NX701 CPU Unit system is provided along with the following information on the CPU Unit.<br>Features and system configuration<br>Overview<br>Part names and functions<br>General specifications<br>Installation and wiring<br>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.<br>Features and system configuration<br>Introduction<br>Part names and functions<br>General specifications<br>Installation and wiring<br>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.<br>Features and system configuration<br>Overview<br>Part names and functions<br>General specifications<br>Installation and wiring<br>Maintenance and Inspection |
| NJ-series CPU Unit Hardware User's Manual | W500 | NJ501-□□□□<br>NJ301-□□□□<br>NJ101-□□□□ | Learning the basic specifications of the NJ-series CPU Units, including introductory information, designing, installation, and maintenance.<br>Mainly hardware information is provided | An introduction to the entire NJ-series system is provided along with the following information on the CPU Unit.<br>Features and system configuration<br>Overview<br>Part names and functions<br>General specifications<br>Installation and wiring<br>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.<br>Features and system configuration<br>Introduction<br>Part names and functions<br>General specifications<br>Installation and wiring<br>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.<br>Features and system configuration<br>Introduction<br>Part names and functions<br>General specifications<br>Installation and wiring<br>Maintenance and inspection |
| NJ/NX-series CPU Unit Software User's Manual | W501 | NX701-□□□□<br>NX102-□□□□<br>NX1P2-□□□□<br>NJ501-□□□□<br>NJ301-□□□□<br>NJ101-□□□□ | Learning how to program and set up an NJ/NX-series CPU Unit.<br>Mainly software information is provided | The following information is provided on a Controller built with an NJ/NX-series CPU Unit.<br>CPU Unit operation<br>CPU Unit features<br>Initial settings<br>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-□□□□<br>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.<br>Controller operation<br>Controller features<br>Controller settings<br>Programming based on IEC 61131-3 language specifications |
| NJ/NX-series Instructions Reference Manual | W502 | NX701-□□□□<br>NX102-□□□□<br>NX1P2-□□□□<br>NJ501-□□□□<br>NJ301-□□□□<br>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-□□□□<br>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-□□□□<br>NX102-□□□□<br>NX1P2-□□□□<br>NJ501-□□□□<br>NJ301-□□□□<br>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-□□□□<br>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-□□□□<br>NX102-□□□□<br>NX1P2-□□□□<br>NJ501-□□□□<br>NJ301-□□□□<br>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-□□□□<br>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<br>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<br>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 | December 2015 | Original production |
| 02 | July 2016 | Changed the manual name. |
| 03 | November 2016 | Changed the manual name. |
| 04 | 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.

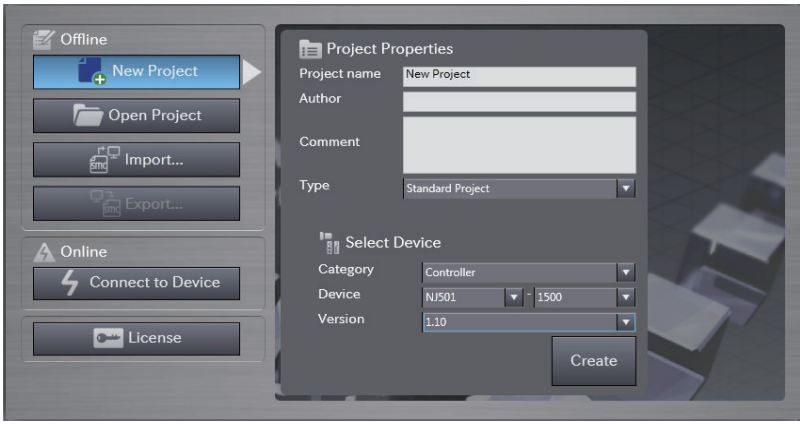
- Using newly installed Sysmac Libraries
- Using upgraded Sysmac Libraries

**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.

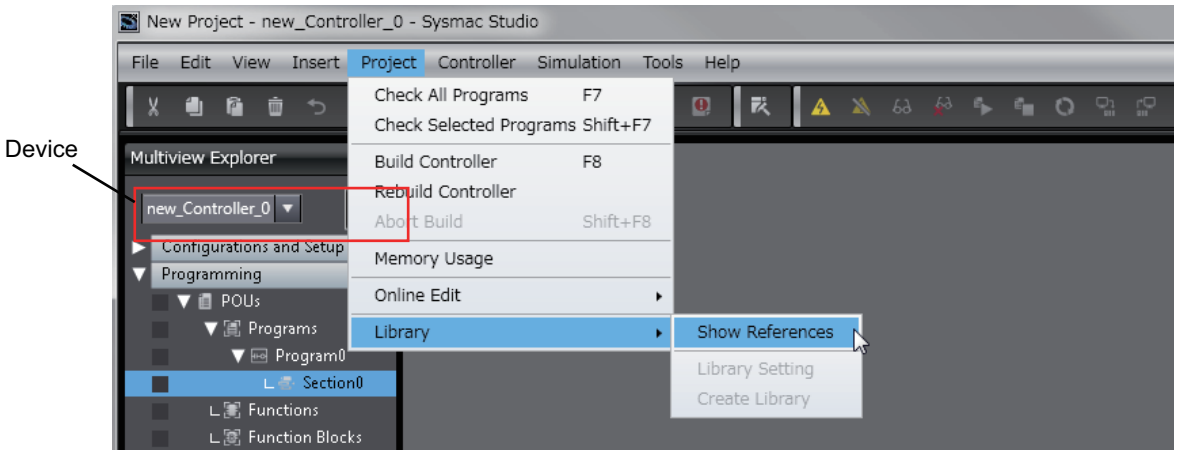



**Precautions for Correct Use**

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.

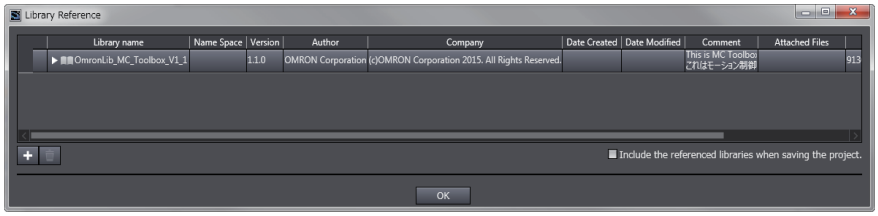
**2** Select **Project** – **Library** – **Show References**.




**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.



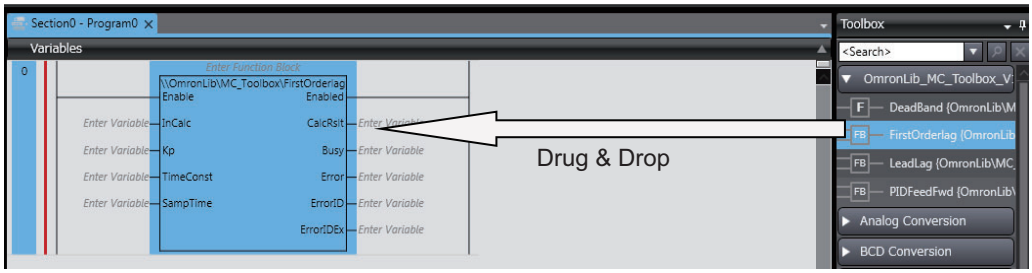

The Sysmac Library file is read into the project.

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

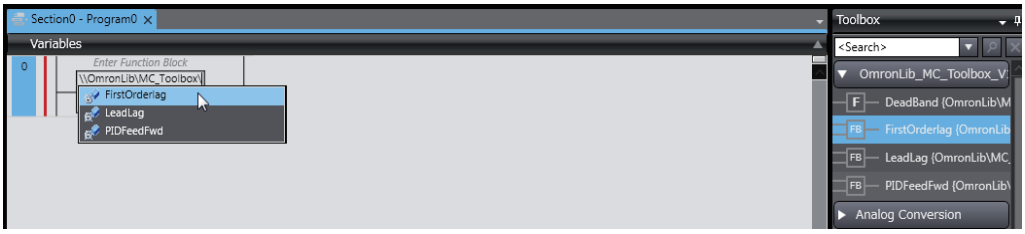
For the procedure for adding and setting libraries in the above screen, refer to the *Sysmac Studio Version 1 Operation Manual* (Cat. No. W504).

**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

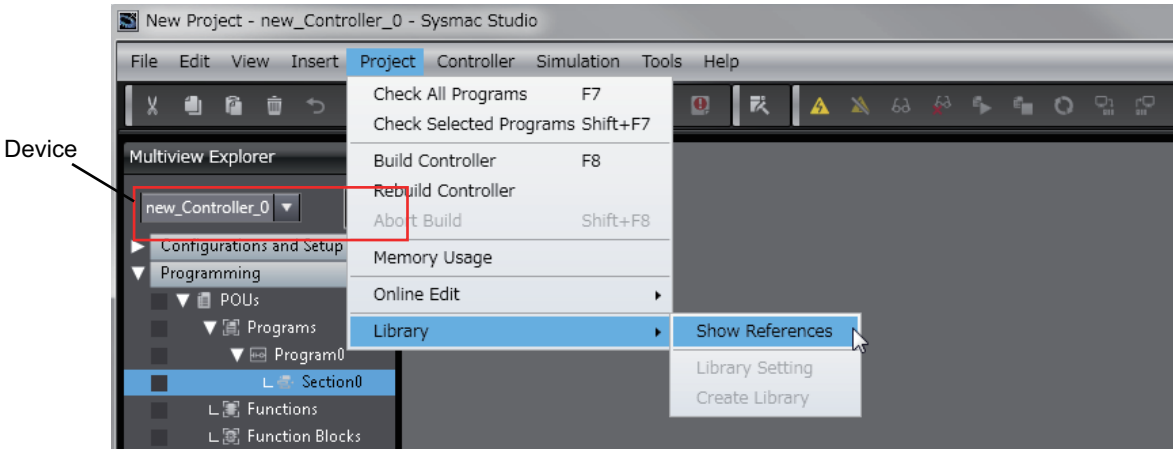
After you upgrade the Sysmac Studio, check all programs and make sure that there is no error of the program check results on the Build Tab Page.

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.

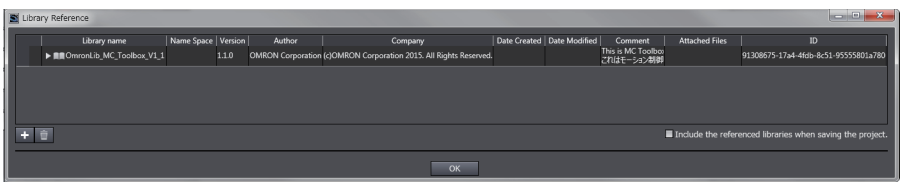
**2** Select **Project** – **Library** – **Show References**.



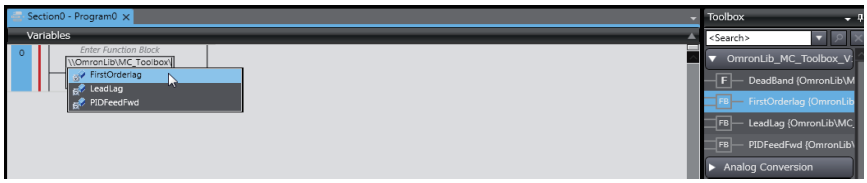

### Precautions for Correct Use

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

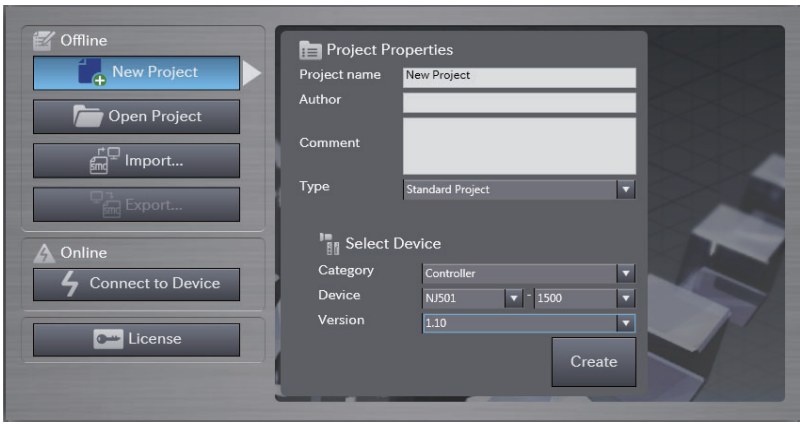
You can use Sysmac Libraries uploaded from a CPU Unit or an Industrial PC to your computer if they are not installed.

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.



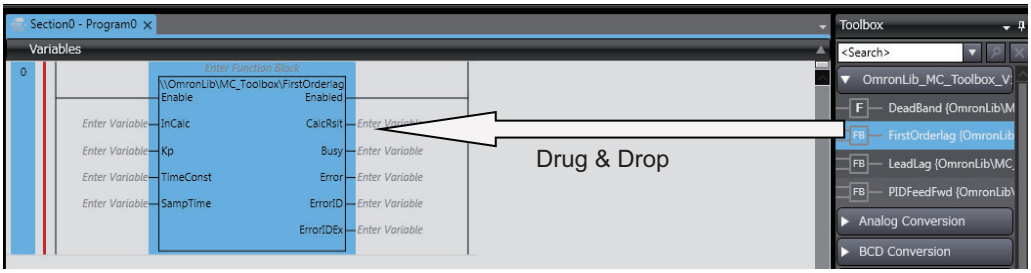

**2** Connect the computer to the CPU Unit or the Industrial PC and place it online.

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

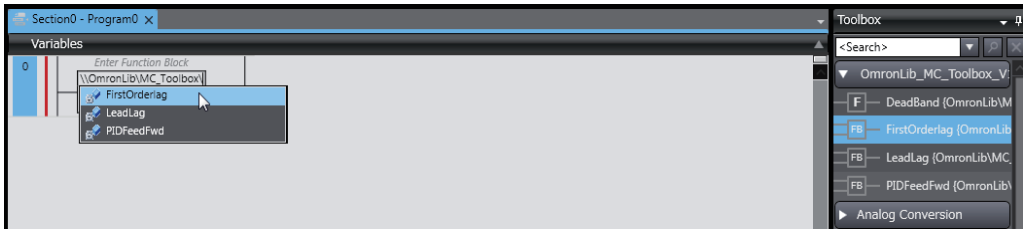
Now, when you select the Ladder Editor or ST Editor, the function blocks and functions included in the Sysmac Library used in the uploaded POUs appear in the Toolbox.

**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

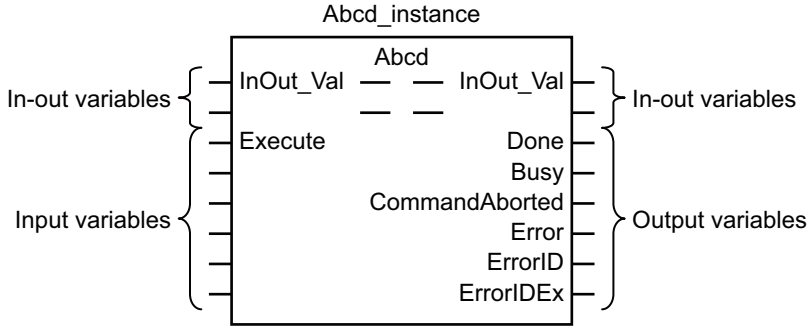
Common input variables and output variables used in functions and function blocks are as follows.

| 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.<br>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.<br>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.<br>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.<br>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.<br>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.<br>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.

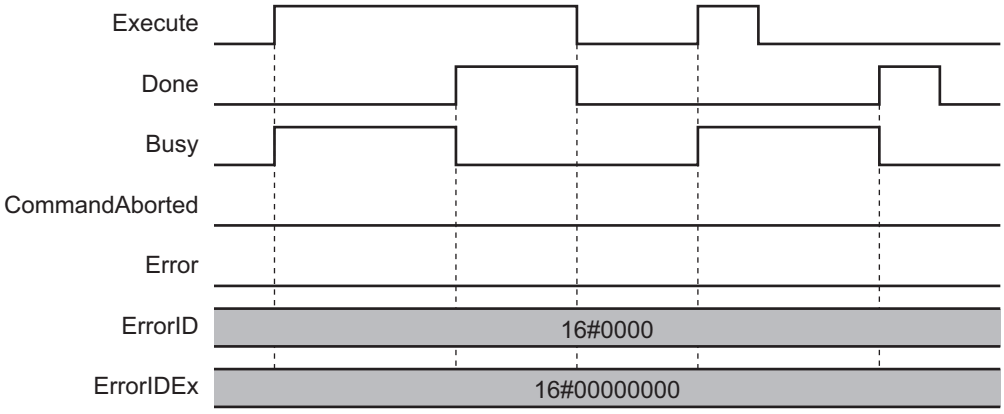



- If *Execute* is TRUE and *Done*, *CommandAborted*, or *Error* changes to TRUE, *Done*, *Command-Aborted*, and *Error* changes to FALSE when *Execute* is changed to FALSE.
- If *Execute* is FALSE and *Done*, *CommandAborted*, or *Error* changes to TRUE, *Done*,*Command-Aborted*, and *Error* changes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

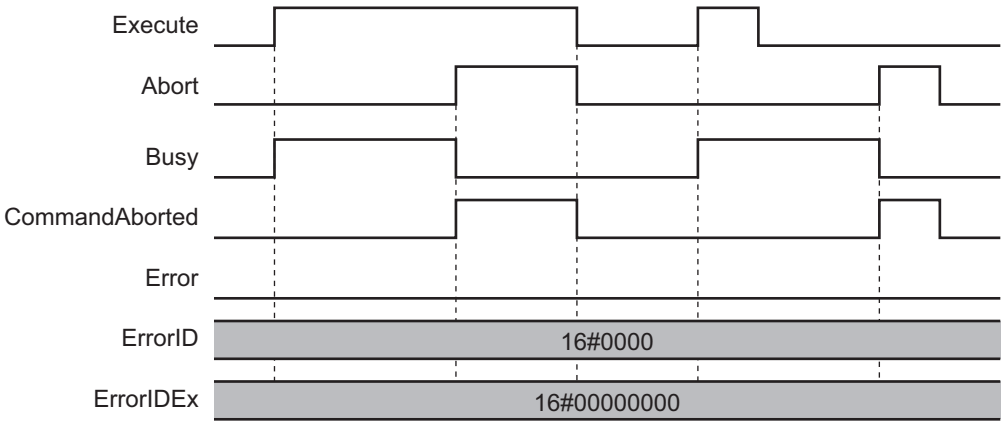
## Timing Charts

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

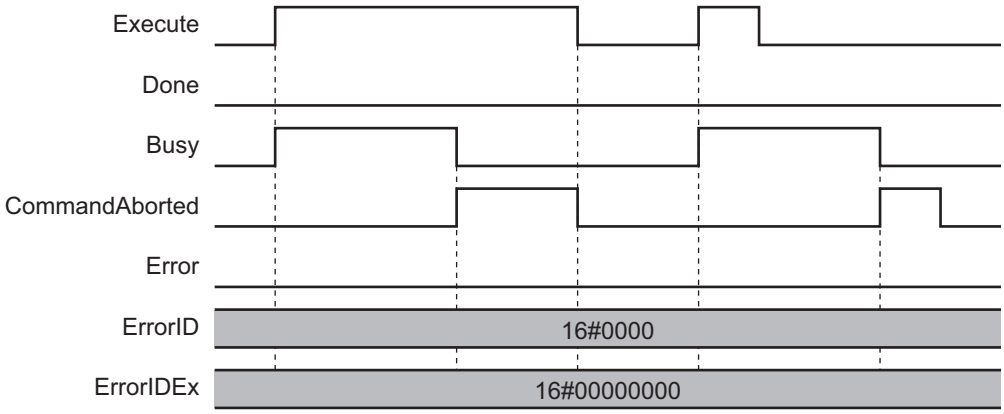
### ● Normal End



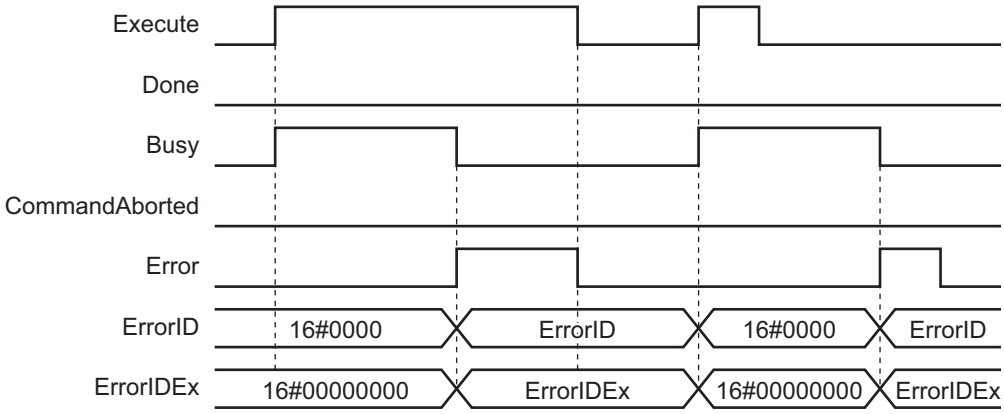

### ● Canceled Execution



## Aborted Execution

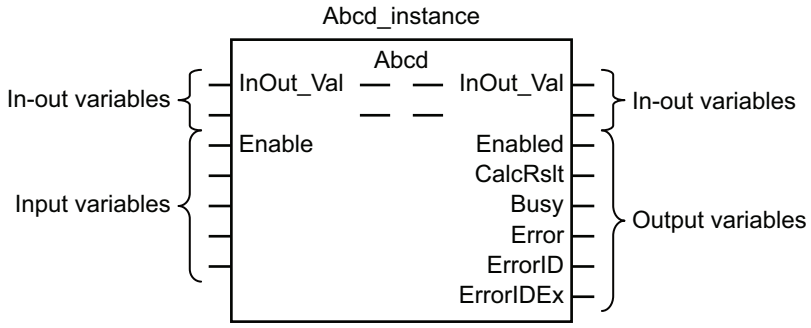



## Errors



## 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.

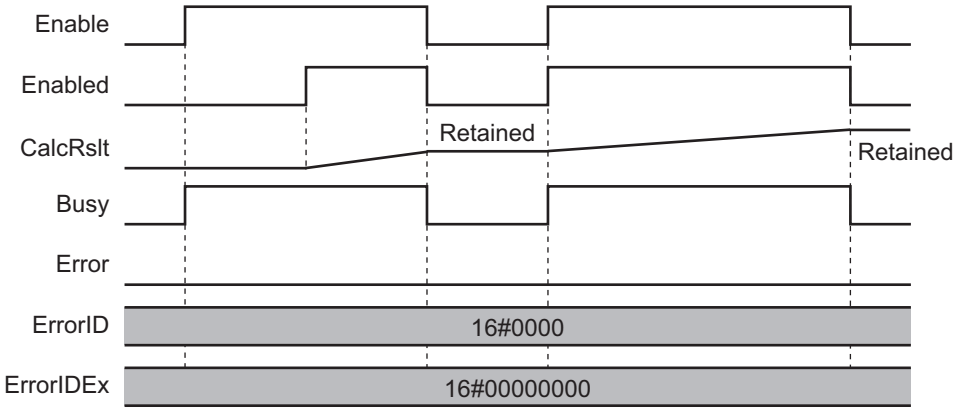



- If an error occurs, the relevant error code and expansion error code are set in *ErrorID* (Error Code) and *ErrorIDEx* (Expansion Error Code). The error codes are retained even after *Error* changes to FALSE, but *ErrorID* is set to 16#0000 and *ErrorIDEx* is set to 16#0000 0000 when *Enable* changes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

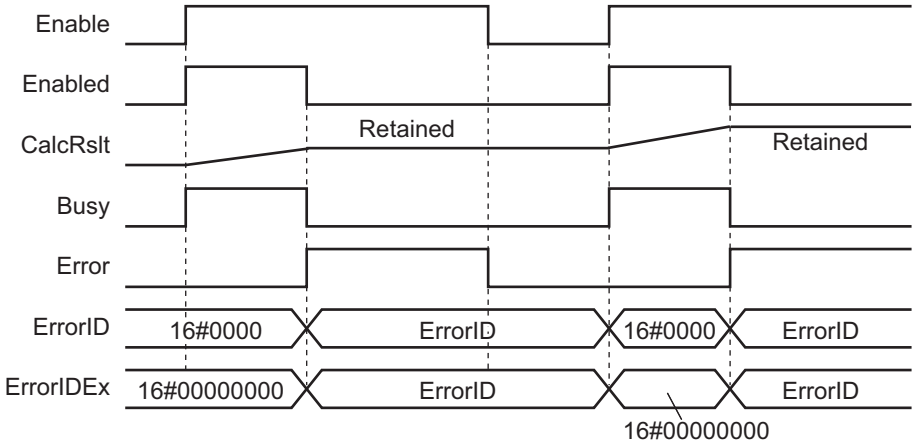
## Timing Charts

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.

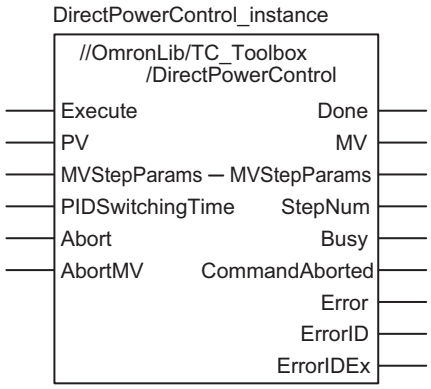
For details on re-execution, refer to the motion control user's manual.

# Specifications of Individual Function Blocks

| Function block name | Name | Page |
|---|---|---|
| DirectPowerControl | Direct Manipulated Variable Control | P. 32 |
| TempUniformityFilter | Temperature Uniformity Filter | P. 46 |

# DirectPowerControl

The DirectPowerControl function block directly manipulates the manipulated variable to follow the set point in temperature control. You can use it to increase the performance of following the set point.

| Function block name | Name | FB/ FUN | Graphic expression | ST expression |
|---|---|---|---|---|
| DirectPowerControl | Direct Manipulated Variable Control | FB |  | DirectPowerControl_instance(<br>Execute:=,<br>PV:=,<br>MVStepParams:=,<br>PIDSwitchingTime:=,<br>Abort:=,<br>AbortMV:=,<br>Done=>,<br>MV=>,<br>MVStepParams:=,<br>StepNum=>,<br>Busy=>,<br>CommandAborted=>,<br>Error=>,<br>ErrorID=>,<br>ErrorIDEx=><br>); |

## Function Block and Function Information

| Item | Description |
|---|---|
| Library file name | OmronLib_TC_Toolbox_V1_0.slr |
| Namespace | OmronLib\TC_Toolbox |
| Function block and function number | 00027 |
| Source code published/not published | Not published |
| Function block and function version | 1.00 |

**Precautions for Correct Use**

When you use this library, implement safety measures, such as monitoring for excessive temperature rise and fall, outside of the function block.

# Variables

## Input Variables

| | Meaning | Data type | Initial value | Valid range | Unit | Description |
|---|---|---|---|---|---|---|
| Execute | Execute | BOOL | FALSE | TRUE or FALSE | --- | Executes the function block. |
| PV*1 | Process Value | REAL | 0 | ±3200.0 | --- | Inputs the present temperature. |
| PIDSwitchingTime*1 | PID Switching Time | TIME | T#0s | Depends on data type. | --- | Sets the time required to change to PID control.<br>• If the outputs for all of the steps specified with *StepSetParams* are completed, the MV from the last step that was executed is output.<br>• If this function block is aborted with *Abort*, the MV from when *Abort* changed to TRUE is output. |
| Abort*1 | Abort | BOOL | FALSE | TRUE or FALSE | --- | Aborts the function block. The *CommandAborted* output variable changes to TRUE when aborting the function block is completed. When *Abort* changes to TRUE, the MV (manipulated variable) specified with *AbortMV* is output. To disable *Abort*, re-execute the function block. |
| AbortMV*1 | Abort MV | BOOL | FALSE | TRUE or FALSE | --- | Specifies the MV to output when the function block is aborted.<br>TRUE: An MV of 0.0% is output.<br>FALSE: The MV from when *Abort* changed to TRUE is output. |

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

**Precautions for Correct Use**

- Do not re-execute the same instance of a function block that uses *Execute*. The values output by the function block returns to the initial values.
- Specify the values of the input parameters within the valid ranges.

**Additional Information**

The execution period of the PID control instruction depends on the sampling period of the PID control instruction. Therefore, the task period of the NJ/NX/NY-series Controller and the execution period of the PID control instruction are not synchronized. It is therefore possible that the MV from this function block will not be detected by the PID control instruction when changing from direct MV control to PID control, depending on the value of *PIDSwitchingTime*. To ensure that the MV from this function block is detected by the PID control instruction, set *PIDSwitchingTime* as follows:

- *PIDSwitchingTime* > PID control instruction sampling period + Task period of task in which this function block is executed

## Output Variables

| | Meaning | Data type | Valid range | Unit | Description |
|---|---|---|---|---|---|
| Done | Done | BOOL | FALSE or TRUE | --- | Changes to TRUE when function block processing is not aborted and ends normally. When *Execute* changes to FALSE, *Done* returns to FALSE. |
| Busy | Executing | BOOL | FALSE or TRUE | --- | TRUE from when the function block execution conditions are met and execution is started until processing is completed. Processing is considered completed for a normal end, error end, or abortion. |
| Command-Aborted | Instruction Aborted | BOOL | FALSE or TRUE | --- | If the *Abort* input variable changes to TRUE during execution of function block processing, execution is forced to end and *Command-Aborted* changes to TRUE. Also, when *Execute* changes to FALSE, *CommandAborted* returns to FALSE. |
| MV | Manipulated Variable | REAL | −320.0 to +320.0 | % | Outputs the MV for the current step.<br>While changing to PID control, the MV from the last step that was executed is output. |
| StepNum | Step Number | USINT | 0 to 5<br>99 | --- | Outputs the number of the current step.<br>While changing to PID control, 99 is output. |
| Error | Error | BOOL | --- | | TRUE: Error end.<br>FALSE: Normal end, execution in progress, or execution condition not met. |
| ErrorID | Error Code | WORD | *1 | | This is the error ID for an error end.<br>The value is 16#0 for a normal end. |
| ErrorIDEx | Expansion Error Code | DWORD | *1 | | This is the error ID for an Expansion Unit Hardware Error.<br>The value is 16#0 for a normal end. |

*1. Refer to *Troubleshooting* on page 42 for details.

## In-Out Variables

| | Meaning | Data type | Address | Initial value | Valid range | Unit | Description |
|---|---|---|---|---|---|---|---|
| MVStepParams*1 | Step Execution Parameters | ARRAY[1..5] OF sMV_STEP_PARAMS | --- | --- | --- | --- | Sets the step execution parameters as a structure.*2 |

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

*2. Refer to *Structures* on page 35 for the structure definition.

## Structures

The data type of the *MVStepParams* input variable to this function block is structure sMV_-STEP_PARAMS. The specifications are as follows:

| Variable or member | Name | Data type | Valid range | Unit | Description |
|---|---|---|---|---|---|
| MVStepParams | Step Execution Parameters | sMV_-STEP_PARAMS | --- | --- | --- |
| StepEnable | Step Enable Flag | BOOL | FALSE or TRUE | --- | This is one of the step execution conditions. Specify whether to enable or disable each step. Only MV steps that are enabled are executed. |
| StepMV | Step MV | REAL | −320.0 to +320.0 | % | These are the MV output values for each step.<br>An error occurs if the value of *StepMV* is larger than 320.0 or smaller than −320.0. |
| StepTime | Step Time | TIME | Depends on data type. | --- | This is one of the step execution conditions.<br>• The step is ended when the time set for *StepTime* elapses.<br>• If *StepTime* is set to 0, the step is not changed based on time. |
| LowPVCondition | Step End Lower PV Condition | REAL | −3,200.00 to +3,200.00 | --- | This is one of the step execution conditions.<br>If the value of *PV* goes below the value of *LowPVCondition*, the step is ended. |
| UpPVCondition | Step End Upper PV Condition | REAL | −3,200.00 to +3,200.00 | --- | This is one of the step execution conditions.<br>If the value of *PV* goes above the value of *UpPVCondition*, the step is ended. |

The values of the members of the structure can be changed during execution of the function block.

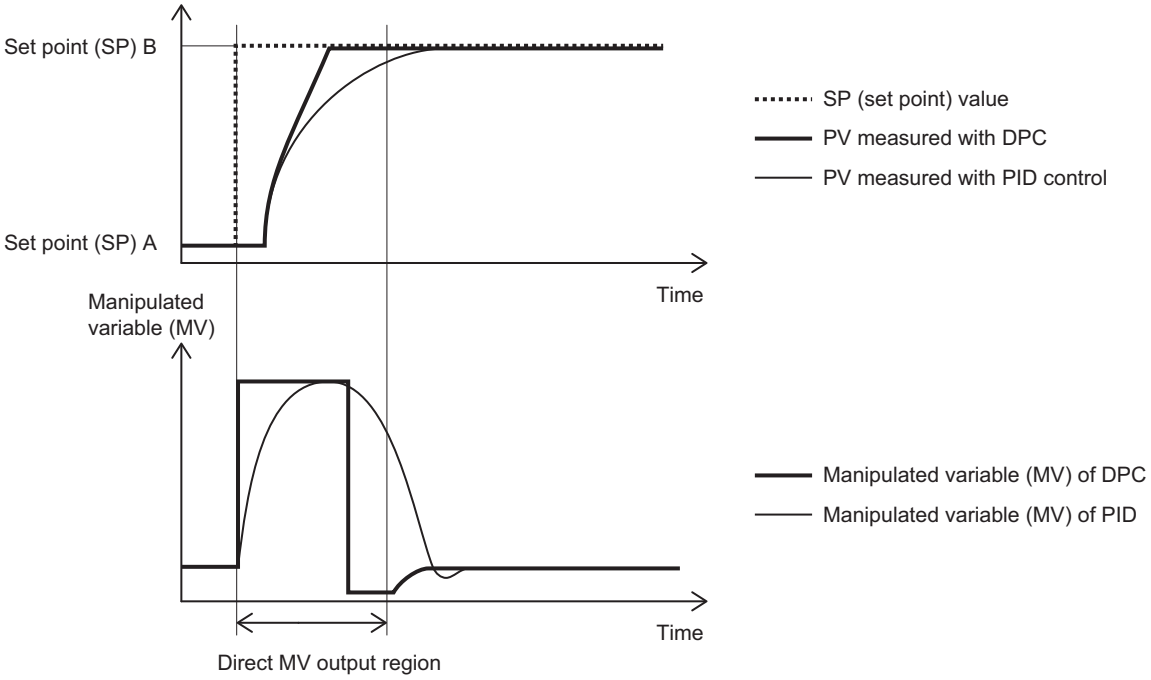
*StepMV* is output when all of the following conditions are met.

- The value of *StepEnable* (Step Enable Flag) is TRUE.
- The elapsed time from the start of the step is equal to or less than the value of *StepTime*.
- *PV* is equal to or greater than the value of *LowPVCondition* (Step End Lower PV Condition) and less than the value of *UpPVCondition* (Step End Upper PV Condition).However, when the value of *LowPVCondition* (Step End Lower PV Condition) and the value of UpPVCondition (Step End Upper PV Condition) are both 0, only the setting of *StepTime* is followed.

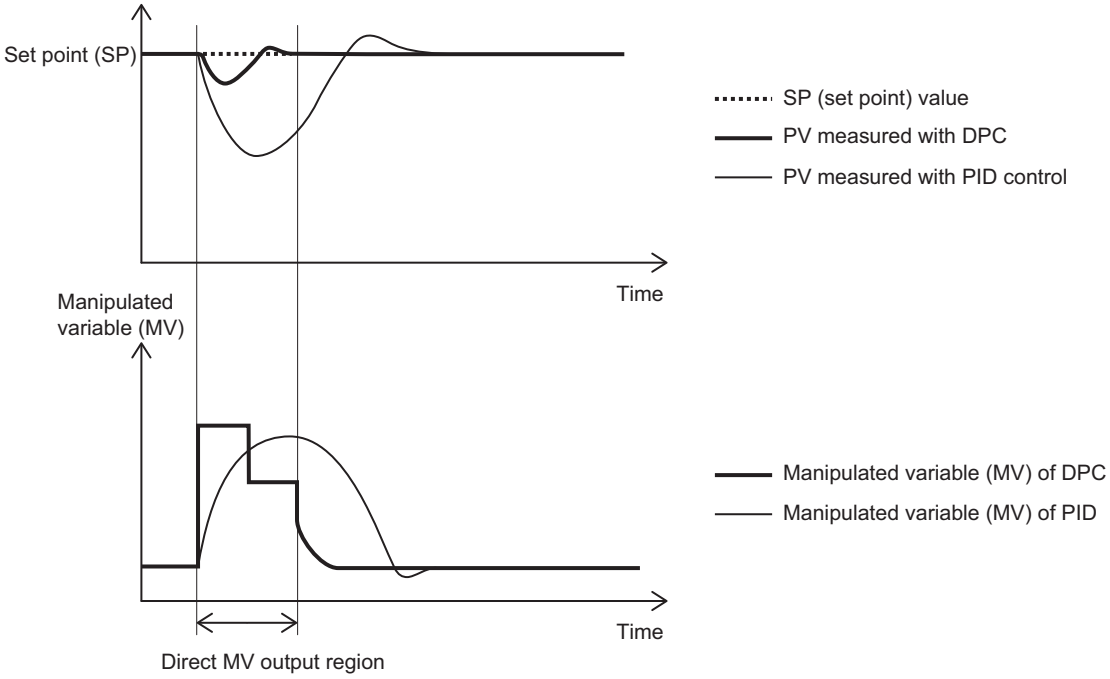
## Function

The DirectPowerControl function block is used with the PIDAT or PIDAT_HeatCool instruction to greatly improve the following performance by directly controlling the MV (manipulated variable) to follow the SP (set point). Directly controlling the MV is called direct manipulated variable control (or DPC: direct power control).

### ● Changing the SP from A to B

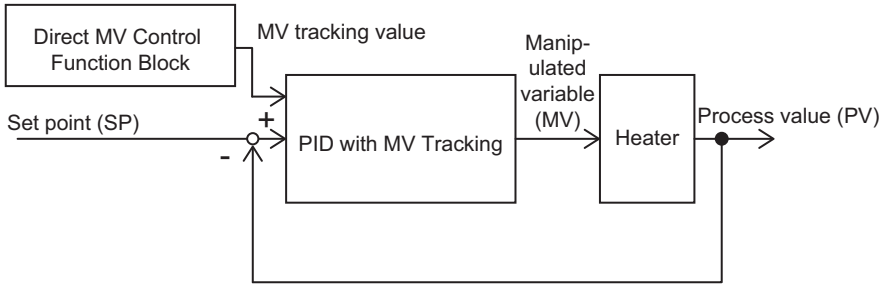



### ● Suppressing Fixed Disturbance



## ● Block Diagram

This function block is used together with a PID control instruction that supports MV tracking. The following control block diagram shows the combination of the PID control instruction and the direct MV control function block.

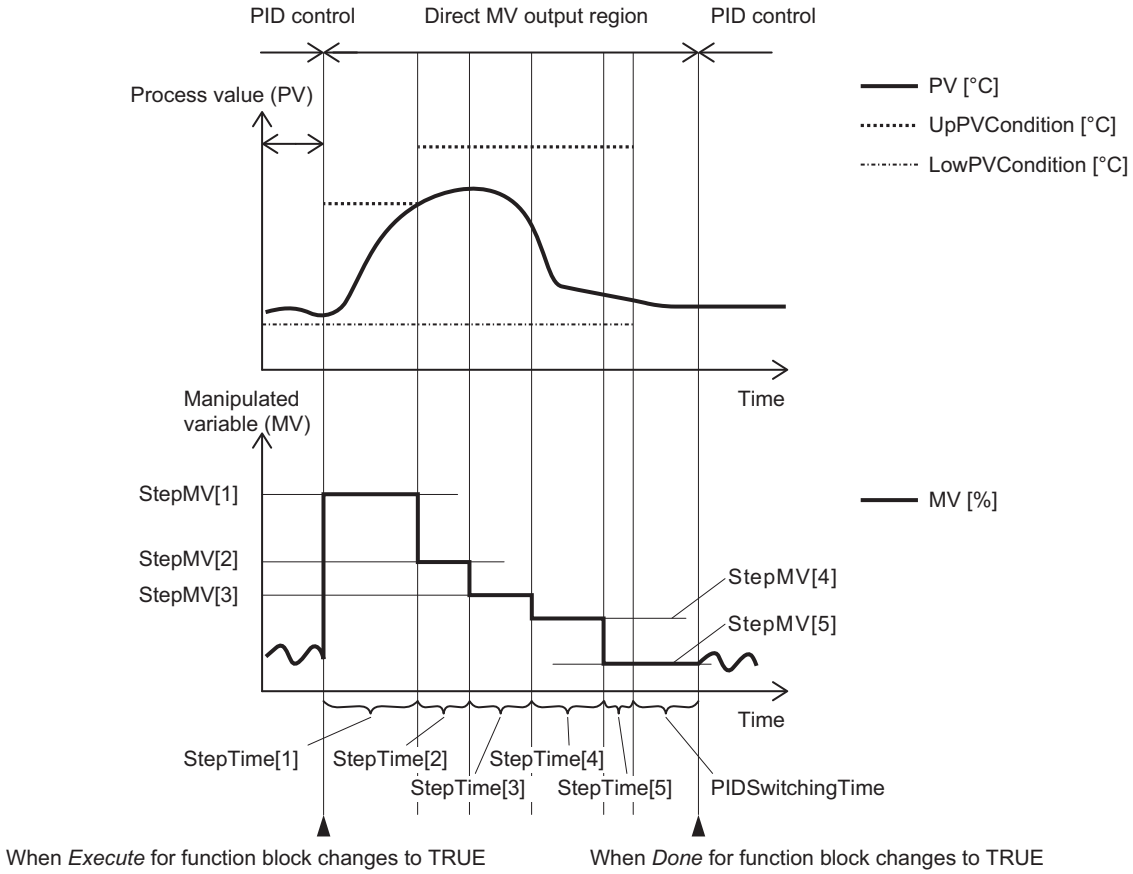



You can use this instruction to generate an MV tracking value. As shown in the following figure, you can set up to five steps of MV tracking values. The *StepMV* is output for each step according to *StepEnable* (Step Enable Flag), *StepTime*, *LowPVCondition* (Step End Lower PV Condition), and *UpPVCondition* (Step End Upper PV Condition), which are described later.

When all of the MVs specified for *StepMV* have been output and the time specified in *PIDSwitching-Time* has elapsed, control is changed to PID control.

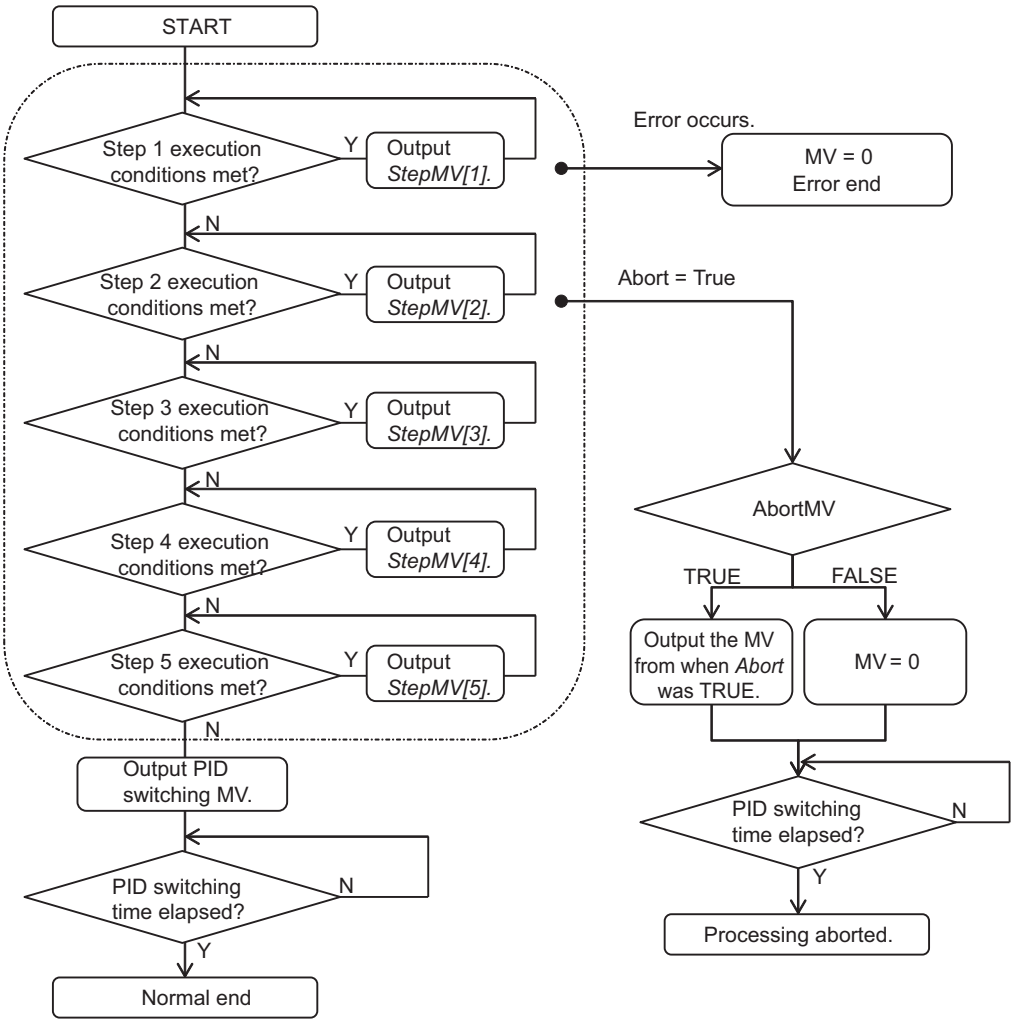
## ● PV (Process Value) and MV (Manipulated Value) When Using the DirectPower-Control Function Block

The following figure shows *PV* (Process Value) and *MV* (Manipulated Variable) if *StepMV[1]* changes to *StepMV[2]* by the value of *LowPVCondition* (Step End Lower PV Condition).



## Processing Flow When Using DirectPowerControl Function Block

- If *Abort* changes to TRUE during *StepMV* output, *AbortMV* is output for the time set with *PID-SwitchingTime* and then function block processing is completed.
- If *Error* changes to TRUE during *StepMV* output, the MV (manipulated variable) output is stopped immediately and the function block ends in an error.

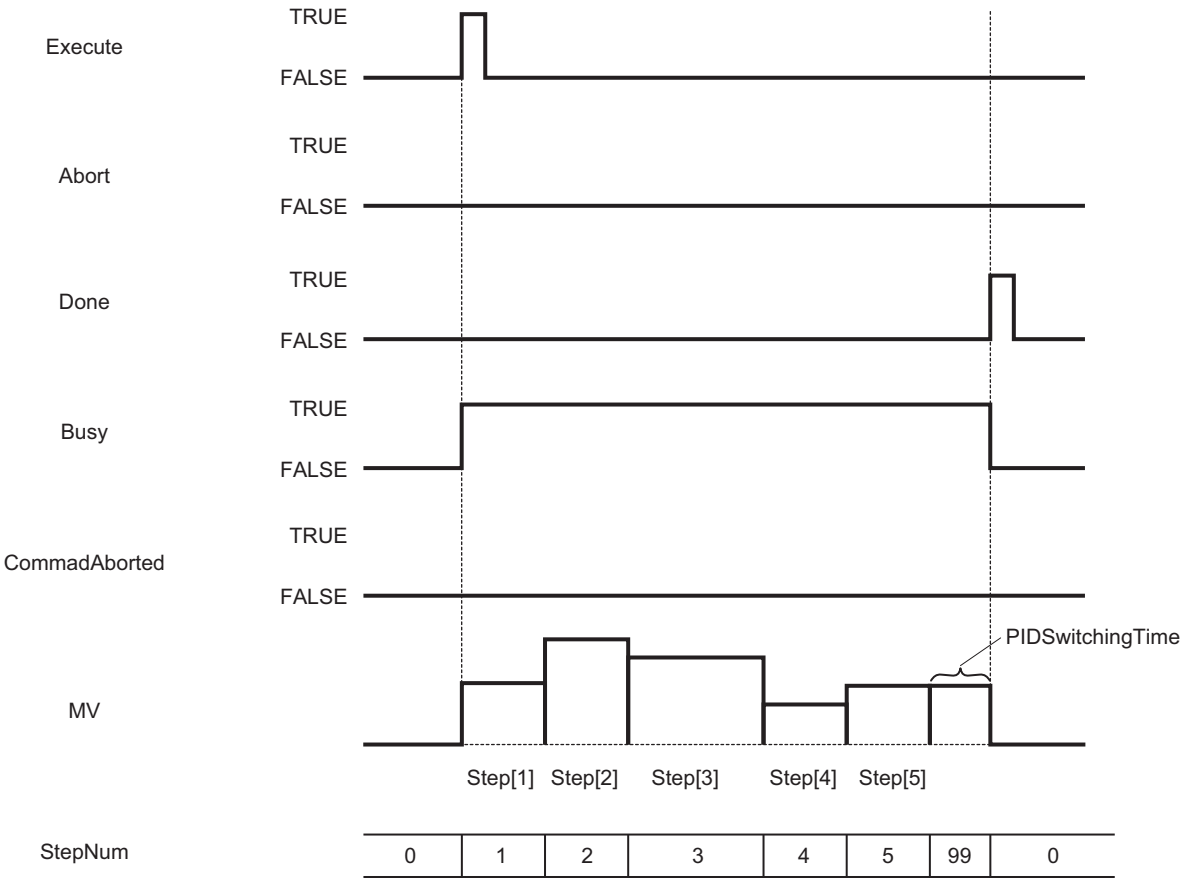


## Timing Charts

This section provides timing charts.

### ● Timing Chart for Normal End

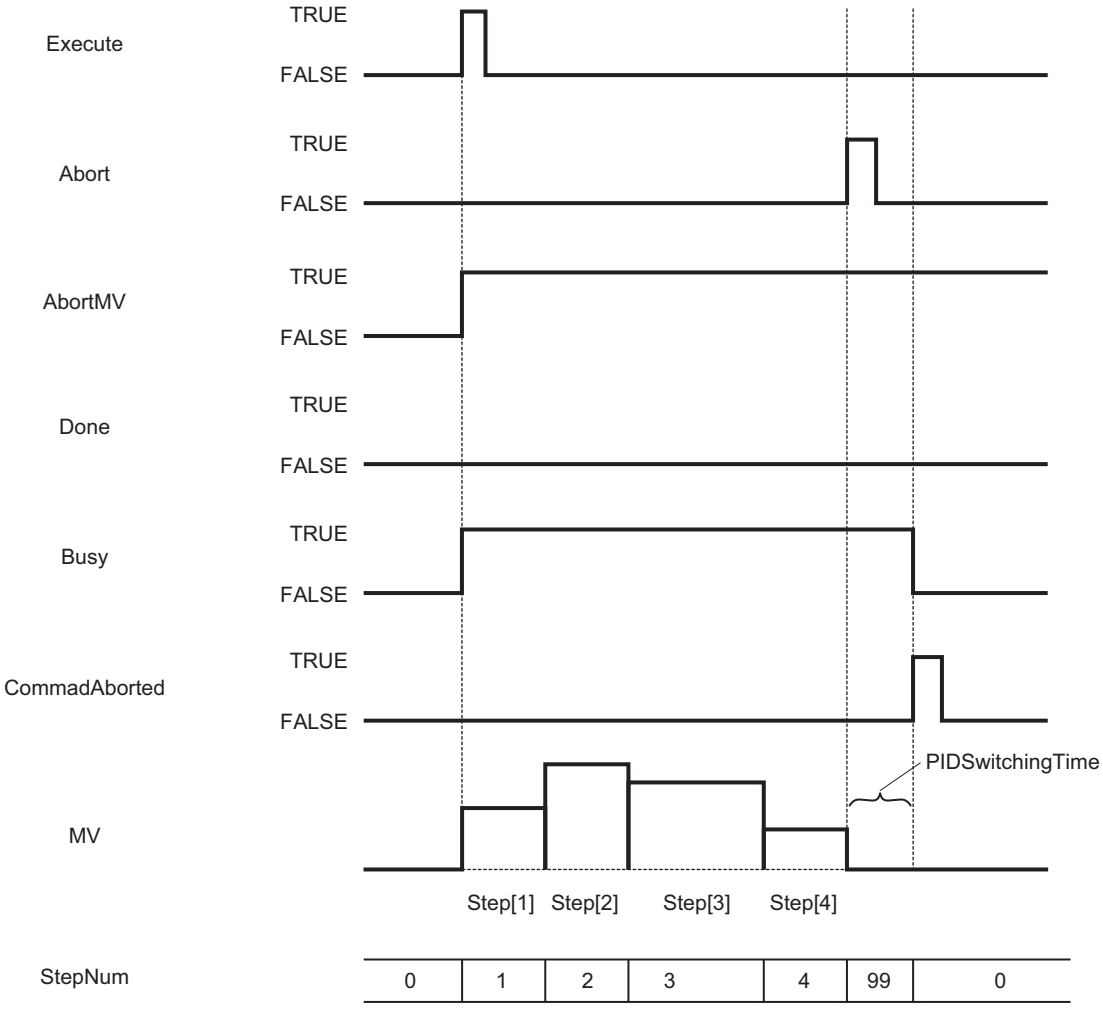
- When *Execute* changes to TRUE, *Busy* (Executing) changes to TRUE and *StepMV* is output. If any of the following switching conditions is met, control changes to the next MV step.
  a) If the current *StepEnable* (Step Enable Flag) changes to FALSE
  b) If the *StepTime* expires
  c) If *PV* goes below the *LowPVCondition* (Step End Lower PV Condition) or above the *UpPVCondition* (Step End Upper PV Condition)
- When output of all of the *StepMV* are completed and *PIDSwitchingTime* has elapsed, *Busy* (Executing) changes to FALSE and *Done* changes to TRUE.
- When output for all steps is completed and *PIDSwitchingTime* has elapsed, *Busy* (Executing) changes to FALSE.
- Steps are always executed in ascending order, Step 1, Step 2, ..., Step 5, and then PID.



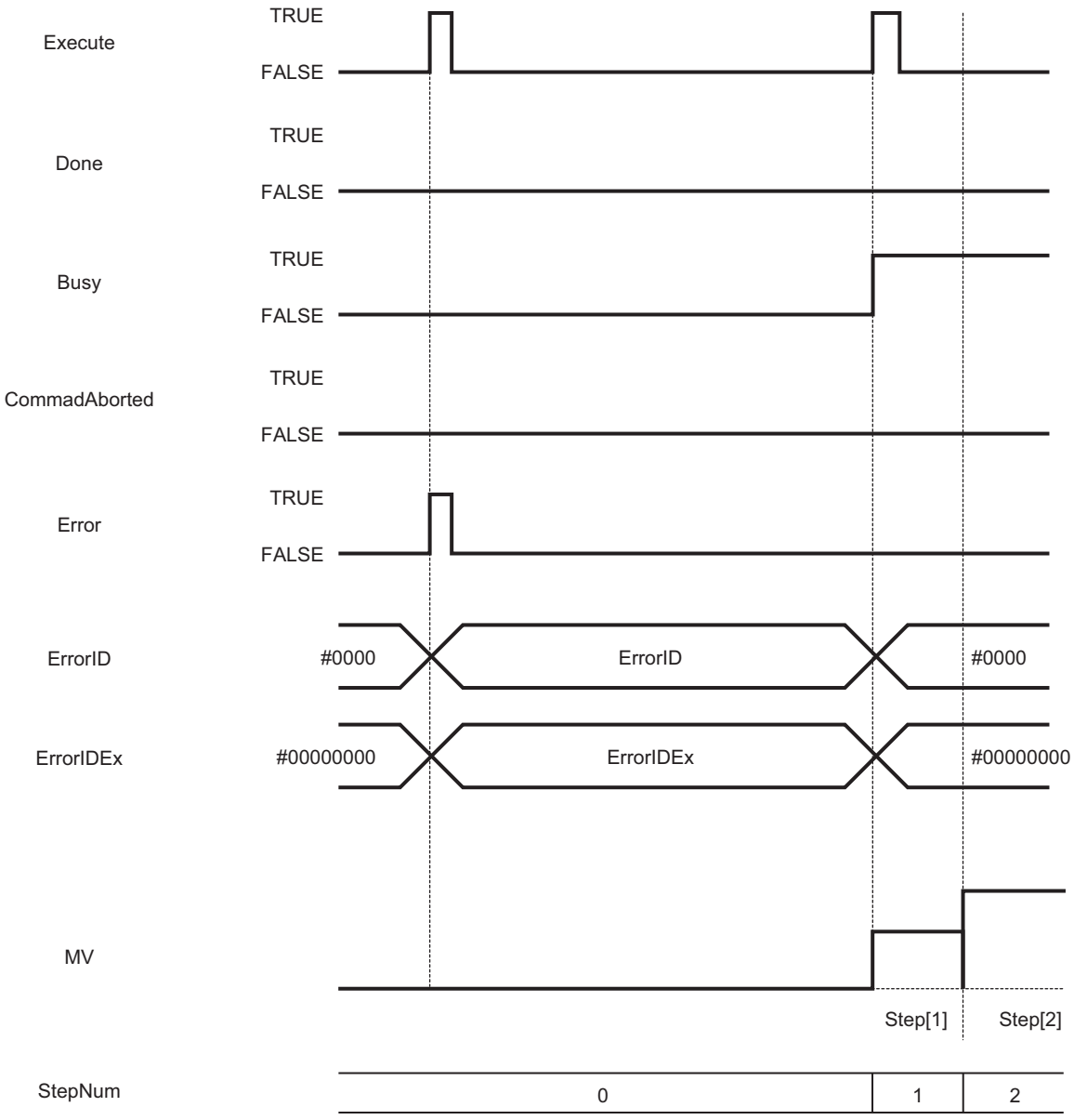

Changes in the values of the input variables and in-out variables are accepted even during function block processing (i.e., while *Busy* (Executing) is TRUE). If the value is changed during function block execution, the new value is used in operation.

## ● Timing Chart for Abortion End

- If *Abort* changes to TRUE during function block execution (i.e., while *Busy* (Executing) is TRUE), the MV given below is output for the time specified with *PIDSwitchingTime*.
  a) If *AbortMV* is TRUE: 0.0 [%]
  b) If *AbortMV* is FALSE: MV [%] when *Abort* changed to TRUE



- If an error occurs during function block execution (i.e., while *Busy* (Executing) is TRUE), *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).
- If an error occurs, 0.0 is output for *MV* (Manipulated Variable).
- When *Execute* 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 | The service ended normally. | --- | --- |
| 16# 3C14 | 16#00000001 | *PV* Out of Range | The value of *PV* (Process Value) exceeded the valid range. | Correct the value set for *PV* (Process Value) so that it is within the valid range. |
| 16# 3C14 | 16#00000002 | *StepMV* Out of Range | The value of *StepMV* exceeded the valid range. | Correct the value set for *StepMV* so that it is within the valid range. |
| 16# 3C14 | 16#00000003 | *LowPVCondition* Out of Range | The value of *LowPVCondition* (Step End Lower PV Condition) is out of range. | Correct the value set for *LowPVCondition* (Step End Lower PV Condition) so that it is within the valid range. |
| 16# 3C14 | 16#00000004 | *UpPVCondition* Out of Range | The value of *UpPVCondition* (Step End Upper PV Condition) is out of range. | Correct the value set for *UpPVCondition* (Step End Upper PV Condition) so that it is within the valid range. |
| 16# 3C14 | 16#00000005 | Illegal Size Relationship between Limit PV Step Transition Conditions | The size relationship between the limit PV step transition conditions set with *LowPVCondition* (Step End Lower PV Condition) and *UpPVCondition* (Step End Upper PV Condition) is not correct. | Set the step end PV conditions to satisfy the following relationship.<br>LowPVCondition > UpPVCondition |

# Sample Programming

## Description of Operation

Connect the *MV* (Manipulated Variable) output parameter for this function block to the *MVTrackVal* (MV Tracking Value) input parameter of the PIDAT instruction.

This function block outputs the *MV* (Manipulated Variable) in three steps as follows: one second on 100%, two seconds 200%, and one second on 80%.

After the third step is complete, it waits PID switching for five seconds.

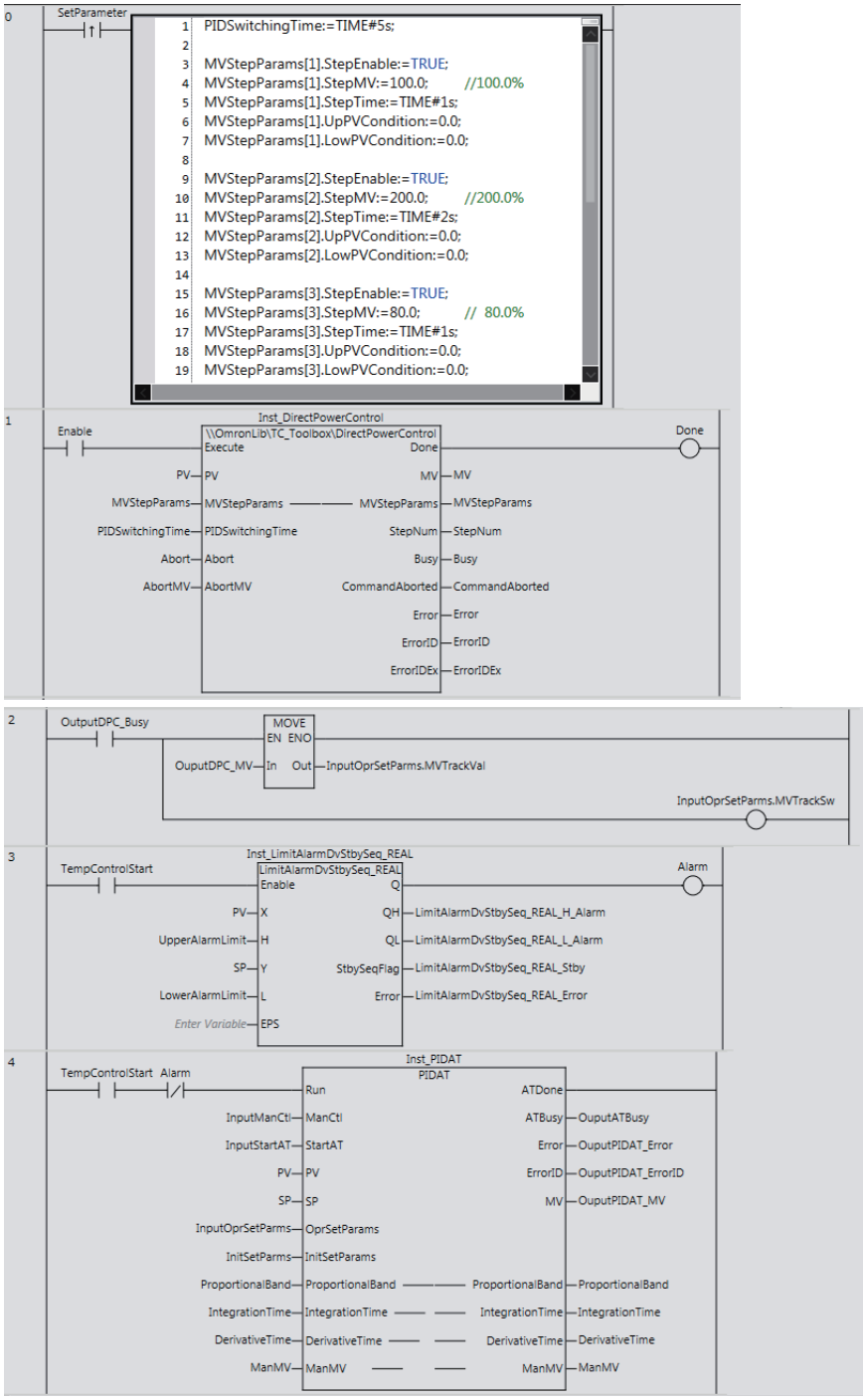
## Variables

### ● Internal Variables

| Name | Data type | Default | Comment |
|---|---|---|---|
| SetParameter | BOOL | | Set Parameter |
| InputPIDSwitchingTime | TIME | | PID Switching Time |
| MVStepParams | ARRAY[1..5] OF OmronLib\TC_Toolbox\sMV_STEP_PARAMS | | MV Step Parameter |
| DPC_start | BOOL | | DirectPowerControl Start |
| Inst_DirectPowerControl | OmronLib\TC_Toolbox\DirectPowerControl | | Instance of DirectPowerControl FB |
| PV | REAL | | Process Value |
| InputAbort | BOOL | | Abort |
| InputAbortMV | BOOL | | |
| DPC_Done | BOOL | | DPC Done |
| OuputDPC_MV | REAL | | DPC MV Output |
| OutputDPC_StepNum | USINT | | |
| OutputDPC_Busy | BOOL | | DPC Busy |
| OutputDPC_CA | BOOL | | |
| OutputDPC_Error | BOOL | | |
| OutputDPC_ErrorID | WORD | | |
| OutputDPC_ErrorIDEx | DWORD | | |
| OuputPIDAT_MV | REAL | | |
| TempControlStart | BOOL | | Temperature Control Start |
| Inst_LimitAlarmDvStbySeq_REAL | LimitAlarmDvStbySeq_REAL | | |
| UpperAlarmLimit | REAL | | Upper Alarm Limit |
| SP | REAL | | Set Point |
| LowerAlarmLimit | REAL | | Lower Alarm Limit |
| AlarmHysteresis | REAL | | Alarm Hysteresis |
| LimitAlarmDvStbySeq_REAL_H_Alarm | BOOL | | |
| LimitAlarmDvStbySeq_REAL_L_Alarm | BOOL | | |
| LimitAlarmDvStbySeq_REAL_Stby | BOOL | | |
| LimitAlarmDvStbySeq_REAL_Error | BOOL | | |
| Alarm | BOOL | | Alarm |
| Inst_PIDAT | PIDAT | | Instance of PIDAT |
| InputManCtl | BOOL | | |

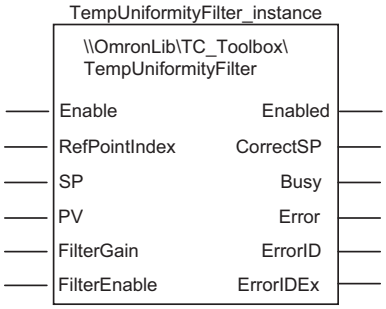
| Name | Data type | Default | Comment |
|---|---|---|---|
| InputStartAT | BOOL | | |
| InputOprSetParms | _sOPR_SET_PARAMS | | |
| InitSetParms | _sINIT_SET_PARAMS | | |
| ProportionalBand | REAL | | |
| IntegrationTime | TIME | | |
| DerivativeTime | TIME | | |
| ManMV | REAL | | |
| OuputATBusy | BOOL | | |
| OuputPIDAT_Error | BOOL | | |
| OuputPIDAT_ErrorID | WORD | | |

## Ladder Diagram



# TempUniformityFilter

The TempUniformityFilter function block unifies the measured temperatures between separate heaters.

| Function block name | Name | FB/ FUN | Graphic expression | ST expression |
|---|---|---|---|---|
| TempUniformityFilter | Temperature Uniformity Filter | FB |  | TempUniformityFilter_instance<br>(Enable:=,<br>RefPointIndex:=,<br>SP:=,<br>PV:=,<br>FilterGain:=,<br>FilterEnable:=,<br>Enabled=>,<br>CorrectSP=>,<br>Busy=>,<br>Error=>,<br>ErrorID=>,<br>ErrorIDEx<br>); |

## Function Block and Function Information

| Item | Description |
|---|---|
| Library file name | OmronLib_TC_Toolbox_V1_0.slr |
| Namespace | OmronLib\TC_Toolbox |
| Function block and function number | 00026 |
| Source code published/not published | Not published |
| Function block and function version | 1.00 |

**Precautions for Correct Use**

When you use this library, implement safety measures, such as monitoring for excessive temperature rise and fall, outside of the function or function block.

# Variables

## Input Variables

| | Meaning | Data type | Initial value | Valid range | Unit | Description |
|---|---|---|---|---|---|---|
| Enable | Enable | BOOL | FALSE | FALSE or TRUE | --- | Processing is performed to keep the temperatures uniform while *Enable* is TRUE. |
| RefPointIndex*1 | Reference Point Index | UINT | 0 | 0 to 9 | --- | Sets a value to determine *RefPoint* (Reference Point). |
| SP*2 | Set Point | ARRAY[1..9] OF REAL | 0 | −3,200.00 to +3,200.00 | --- | Sets the SP for each loop.<br>If temperature uniformity is required when the temperature increases, set the set points in a ramp. |
| PV*2 | Process Value | ARRAY[1..9] OF REAL | 0 | −3,200.00 to +3,200.00 | --- | Sets the PV for each loop. |
| FilterGain*2 | Filter Gain | ARRAY[1..9] OF UINT | 0 | 0 to 200 | % | Sets the correction gain for each loop. The gain is used to adjust *CorrectSP*.<br>Note If the gain is 0%, *SP* is output to *CorrectSP*. |
| FilterEnable*1 | Filter Enable | ARRAY[1..9] OF BOOL | FALSE | FALSE or TRUE | --- | Sets whether or not to enable each loop.<br>• The loop is enabled when the array element is TRUE.<br>• The loop is disabled when the array element is FALSE. |

*1. Processing is started when *Enable* to this function block changes to TRUE.

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

**Precautions for Correct Use**

- Specify the values of the input parameters within the valid ranges.
- If *FilterEnable* changes to FALSE for a loop, the *SP* input to the function block is output as is to *CorrectSP* (Corrected SP).

## Output Variables

| | Meaning | Data type | Valid range | Unit | Description |
|---|---|---|---|---|---|
| Enabled | Enable | BOOL | FALSE or TRUE | --- | |
| CorrectSP | Corrected SP | ARRAY[1..9] OF REAL | --- | --- | Outputs corrected SPs to make the temperatures uniform. |
| Busy | Executing | BOOL | FALSE or TRUE | --- | TRUE while function block execution is in progress.<br>FALSE while function block execution is stopped. |
| Error | Error | BOOL | --- | --- | TRUE: Error end.<br>FALSE: Normal end, execution in progress, or execution condition not met. |
| ErrorID | Error Code | WORD | *1 | --- | This is the error ID for an error end.<br>The value is 16#0 for a normal end. |
| ErrorIDEx | Expansion Error Code | DWORD | *1 | --- | This is the error ID for an Expansion Unit Hardware Error.<br>The value is 16#0 for a normal end. |

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

**Precautions for Correct Use**

Do not use a function or function block with an *Enabled* output variable to output the function or function block processing results to a control target while the value of *Enabled* is FALSE.

## Function

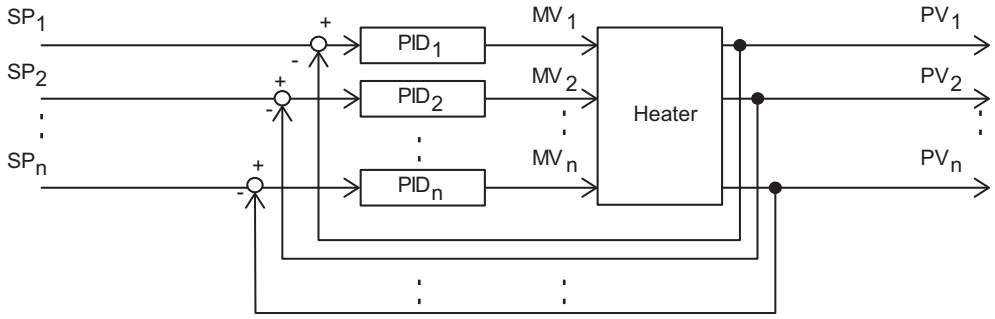
The TempUniformityFilter function block is used to unify the process values between separate heaters.

Normally, separate PID control is used for separate heaters. The TempUniformityFilter function block outputs a corrected SP that is calculated with a temperature uniformity filter, to each PID control loop.

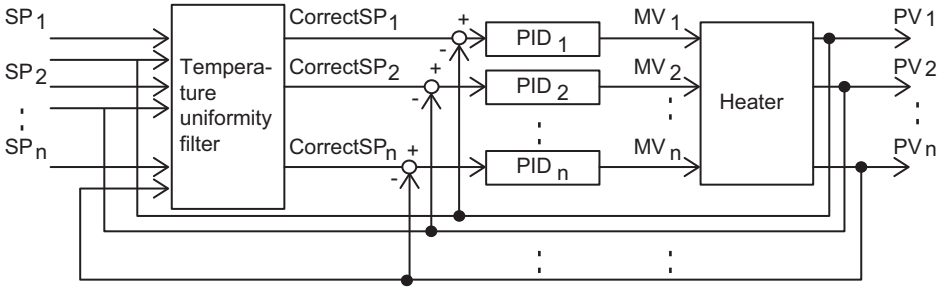
**Additional Information**

Here, “loop” indicates the combination of the temperature sensor, temperature controller (i.e., the PIDAT instruction), and the heater.

### ● Normal PID Control



## Temperature Uniformity Control

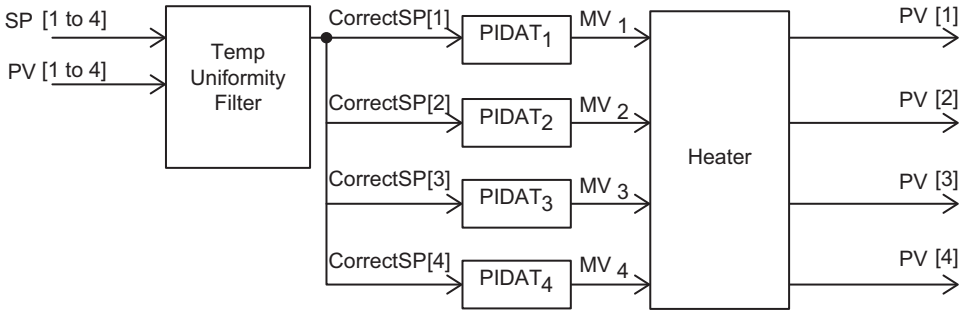



**Additional Information**

If MV (manipulated variable) saturation occurs with separate PID control, sufficient temperature uniformity may not be achieved by using this function block. Particularly when the MV (manipulated variable) is normally saturated and the PV does not settle at the SP, temperature uniformity cannot be achieved.

## Control Blocks

The TempUniformityFilter function block is combined with the PID Control with Autotuning (PIDAT) instruction, which functions as a PID controller, to achieve temperature uniformity. You can use the *FilterEnable* input variable to this function block to specify the loops to which to apply temperature uniformity control.




**Additional Information**

Refer to *Analog Control Instruction* in the instructions reference manual for details on the PIDAT instruction.

**Precautions for Correct Use**

Implement measures to detect heater element burnout or failure and temperature sensor failure. If a heater element burnout is detected, change *Enable* to this function block and *RUN* to the PIDAT or PIDAT_HeatCool instruction to FALSE.

## Meanings of Variables

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

### ● Enable

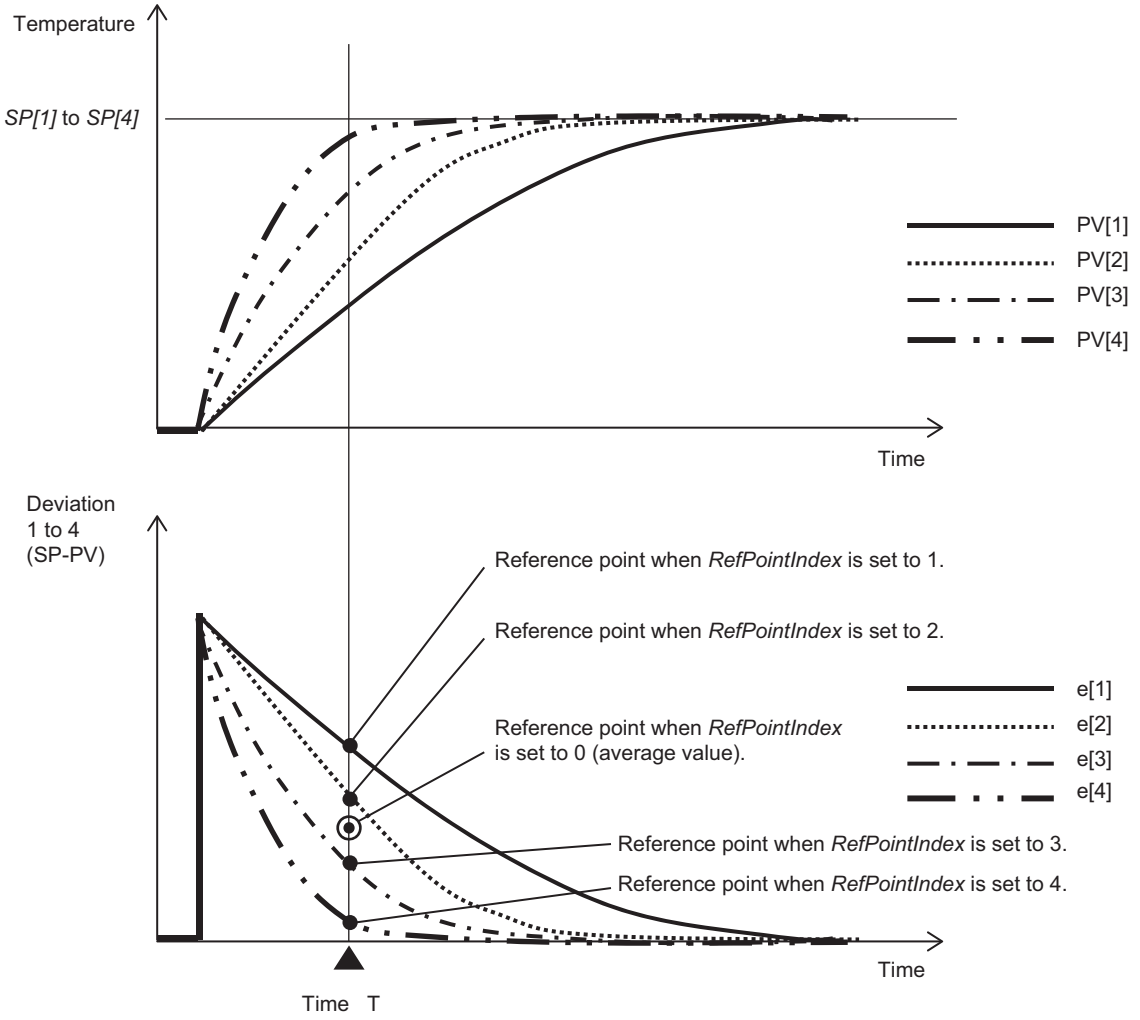
This is the execution condition for the function block. Temperature uniformity control processing is performed while the value is TRUE. If the value changes to FALSE, processing is stopped and the input *SP* is output to *CorrectSP* (Corrected SP).

### ● RefPointIndex (Reference Point Index)

Set a value to determine *RefPoint* (Reference Point). *RefPoint* (Reference Point) is a deviation (i.e., the difference between the SP and PV) in the loop that is used as a reference for temperature uniformity. *RefPoint* (Reference Point) is determined by the value of *RefPointIndex* (Reference Point Index) as shown in the following table.

| Value | Resulting *RefPoint* (Reference Point) |
|---|---|
| *RefPointIndex* = 0 | *RefPoint* (Reference Point) is the average deviation of all loops for which the corresponding element in *FilterEnable* is TRUE. |
| *RefPointIndex* = N (1 to 9) | *RefPoint* (Reference Point) is the Nth largest deviation of all of the enabled loops.<br>Examples:<br>• If *RefPointIndex* is set to 1, *RefPoint* (Reference Point) is the highest deviation of all the loops.<br>• If *RefPointIndex* is set to 4, *RefPoint* (Reference Point) is the fourth highest deviation of all the loops. |

The following figure shows the relationship between *RefPointIndex* (Reference Point Index) and *RefPoint* (Reference Point).

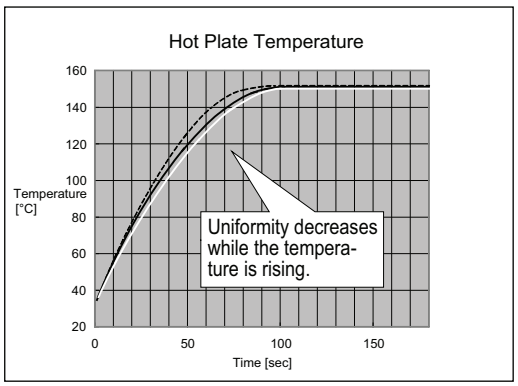


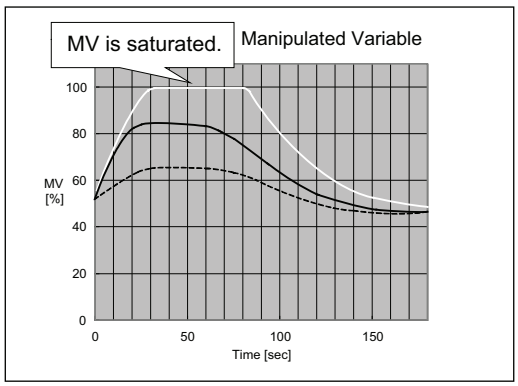

**Additional Information**

- Normally set *RefPointIndex* (Reference Point Index) to 0.
- If temperature uniformity control is being performed for loops with rapid temperature changes and slow temperature changes, set *RefPointIndex* (Reference Point Index) to a loop with slow temperature changes. The speed of temperature changes of loops is indicated by the time constant of the PV when the SP is changed.

**Additional Information**

If temperature uniformity control is performed for loops for which the *MV* (Manipulated Variable) from the PID control instruction (PIDAT) becomes saturated, sufficient temperature uniformity may not be achieved by using this function block.




If the MVs (manipulated variables) become saturated during a transitional state when the temperature is rising or falling, perform the following adjustments.

- When the temperature is rising, set *RefPointIndex* (Reference Point Index) to the loop with the largest deviation (i.e., the lowest temperature). For cooling control, set *RefPointIndex* to the loop with the smallest deviation (i.e., the highest temperature).
- Set the SP in a ramp to achieve smooth temperature changes.

## ● SP

Input the set point. If the control target is a heater, input the target temperature.

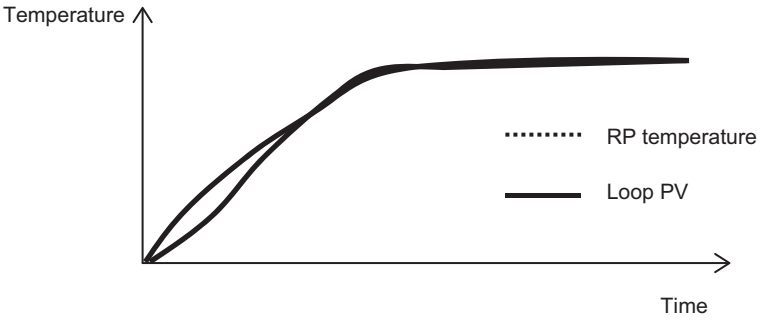
## ● PV

Input the process value. If the control target is a heater, input the measured temperature.

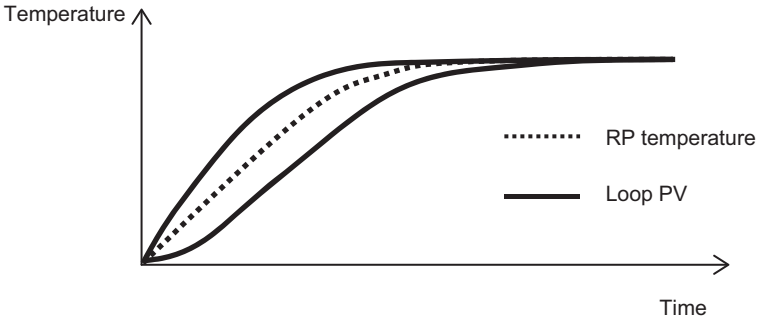
## FilterGain

This coefficient is used to adjust the correction strength of temperature uniformity control. The following figure shows an example for two loops.

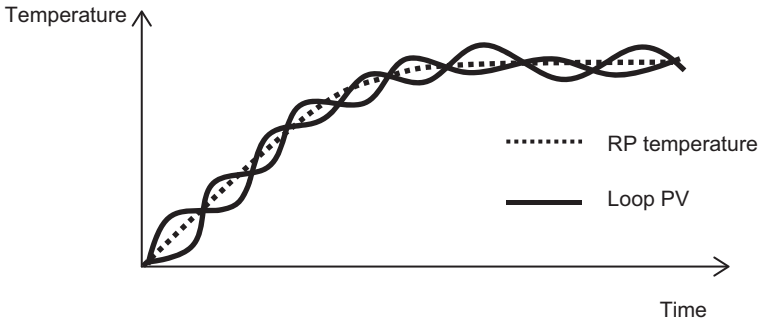
Suitable Value for *FilterGain*




Excessively Small Value for *FilterGain*




Excessively Large Value for *FilterGain*




## FilterEnable

Specify the loops for which to perform temperature uniformity control. To enable temperature uniformity control for a loop, change *FilterEnable[i]* to TRUE, where i is the array element number for the loop. To disable temperature uniformity control for a loop, change *FilterEnable[i]* to FALSE. For example, to enable using this function block for four separate heaters, set *FilterEnable[1]* to *FilterEnable[4]* to TRUE and set *FilterEnable[5]* to *FilterEnable[9]* to FALSE

If *FilterEnable* changes to FALSE for a loop, the *SP* input to the function block is output as is to *CorrectSP* (Corrected SP).

## ● CorrectSP (Corrected SP)

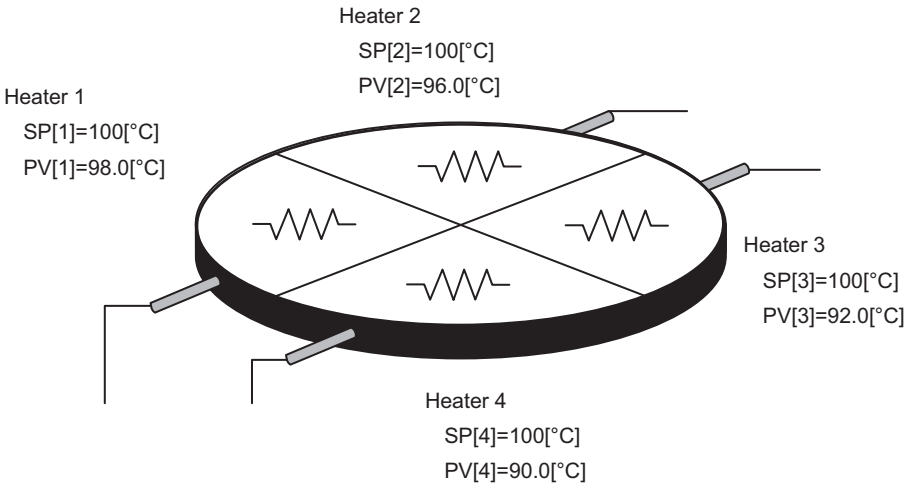
This section describes the processing performed for *CorrectSP* (Corrected SP).

The value of *CorrectSP* (Corrected SP) is calculated for loops for which *FilterEnable* is TRUE based on the *SP, PV, FilterGain*, and *RefPoint* (Reference Point) for each loop. *RefPoint* is determined inside the function block according to *RefPointIndex* (Reference Point Index).

A processing example for *CorrectSP* (Corrected SP) when this function block is used for four-loop heater temperature control is provided below.

Here, we assume that the heater temperatures are set as shown in the following table before the function block is executed.

| Heater | SP | PV |
|---|---|---|
| Heater 1 | 100.0 | 98.0 |
| Heater 2 | 100.0 | 96.0 |
| Heater 3 | 100.0 | 92.0 |
| Heater 4 | 100.0 | 90.0 |




- Performing Temperature Uniformity Control for the Average of the Heater PVs (*PV[1]* to *PV[4]*)

  The following settings are made.

(a) *RefPointIndex* = 0

(b) *FilterGain[1]* to *FilterGain[4]* = 100 [%]

(c) *FilterEnable[1]* to *FilterEnable[4]* = TRUE

Process 1: Calculating *RefPoint* (Reference Point)

*RefPointIndex* is set to 0 (average deviation), so the result of the following formula is calculated as 6.0 [°C].

$$\text{RefPoint} = \frac{\sum_{i=1}^{4} (\text{SP[i]-PV[i]})}{4}$$

Process 2: Calculating *CorrectSP*

After the reference point is determined, the values of *CorrectSP* (Corrected SP) are calculated with the following formula.

CorrectSP[i] = {(SP[i] - PV[i] - RefPoint) * FilterGain} + SP[i]

i : Array element number for loop

The following results are calculated and output from the function block: *CorrectSP[1]* = 96.0 [°C], *CorrectSP[2]* = 98.0 [°C], *CorrectSP[3]* = 102.0 [°C], and *CorrectSP[4]* = 104.0 [°C].

- Adjusting to the PV of the Heater with the Nth Largest Deviation

  The following example shows the processing for temperature uniformity control when the PV with the largest deviation is used as the reference point.

  a) *RefPointIndex* = 1
  b) *FilterGain[1]* to *FilterGain[4]* = 100 [%]
  c) *FilterEnable[1]* to *FilterEnable[4]* = TRUE

  Process 1: Calculating *RefPoint*

  If *RefPointIndex* is set to 1 (largest deviation), the deviations of all loops are calculated, and the deviation of the loop with the largest deviation (10.0 [°C]) is used for *RefPoint* (Reference Point).

  a) Loop 1 deviation: 2.0 [°C] (*SP[1] - PV[1]*)
  b) Loop 2 deviation: 4.0 [°C] (*SP[2] - PV[2]*)
  c) Loop 3 deviation: 8.0 [°C] (*SP[3] - PV[3]*)
  d) Loop 4 deviation: 10.0 [°C] (*SP[4] - PV[4]*)

  Process 2: Calculating *CorrectSP*

  After the reference point is determined, the same formula is used to calculate *CorrectSP* as in *Process 2: Calculating CorrectSP* on page 54 of *Performing Temperature Uniformity Control for the Average of the Heater PVs (PV[1] to PV[4])* on page 54.

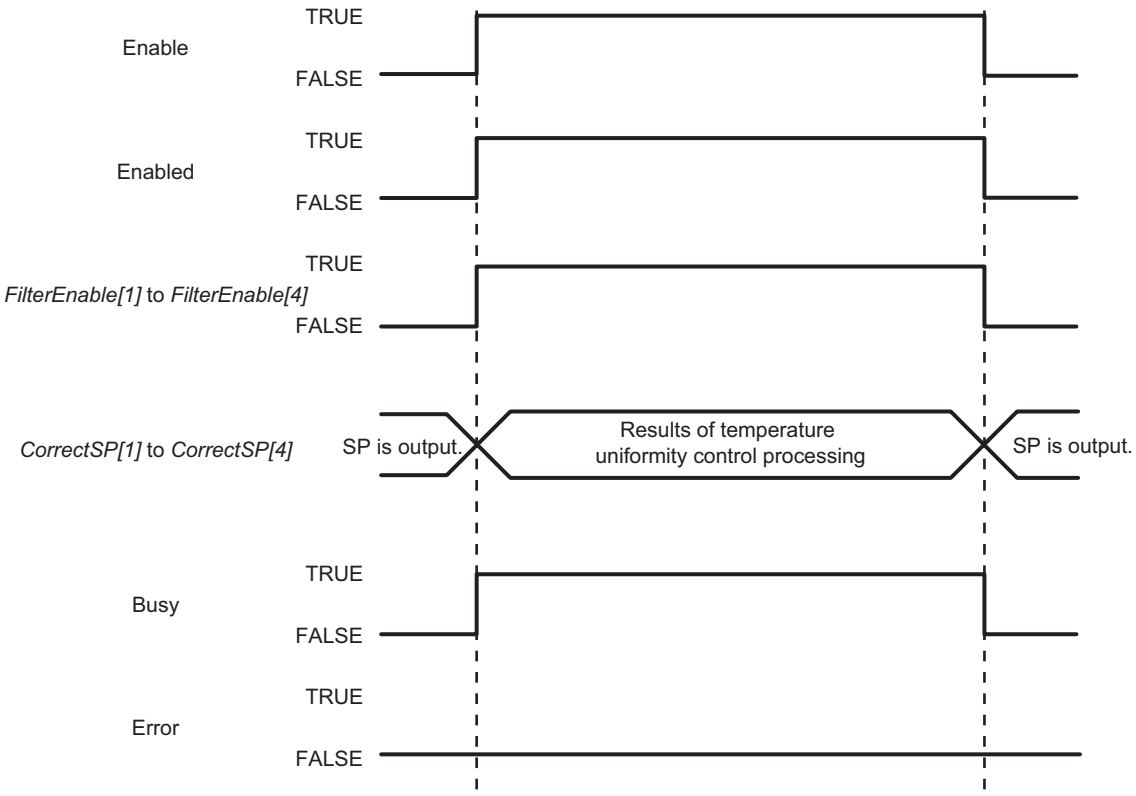
  The following results are calculated and output from the function block: *CorrectSP[1]* = 92.0 [°C], *CorrectSP[2]* = 94.0 [°C], *CorrectSP[3]* = 98.0 [°C], and *CorrectSP[4]* = 100.0 [°C].

## Timing Charts

This section provides timing charts.

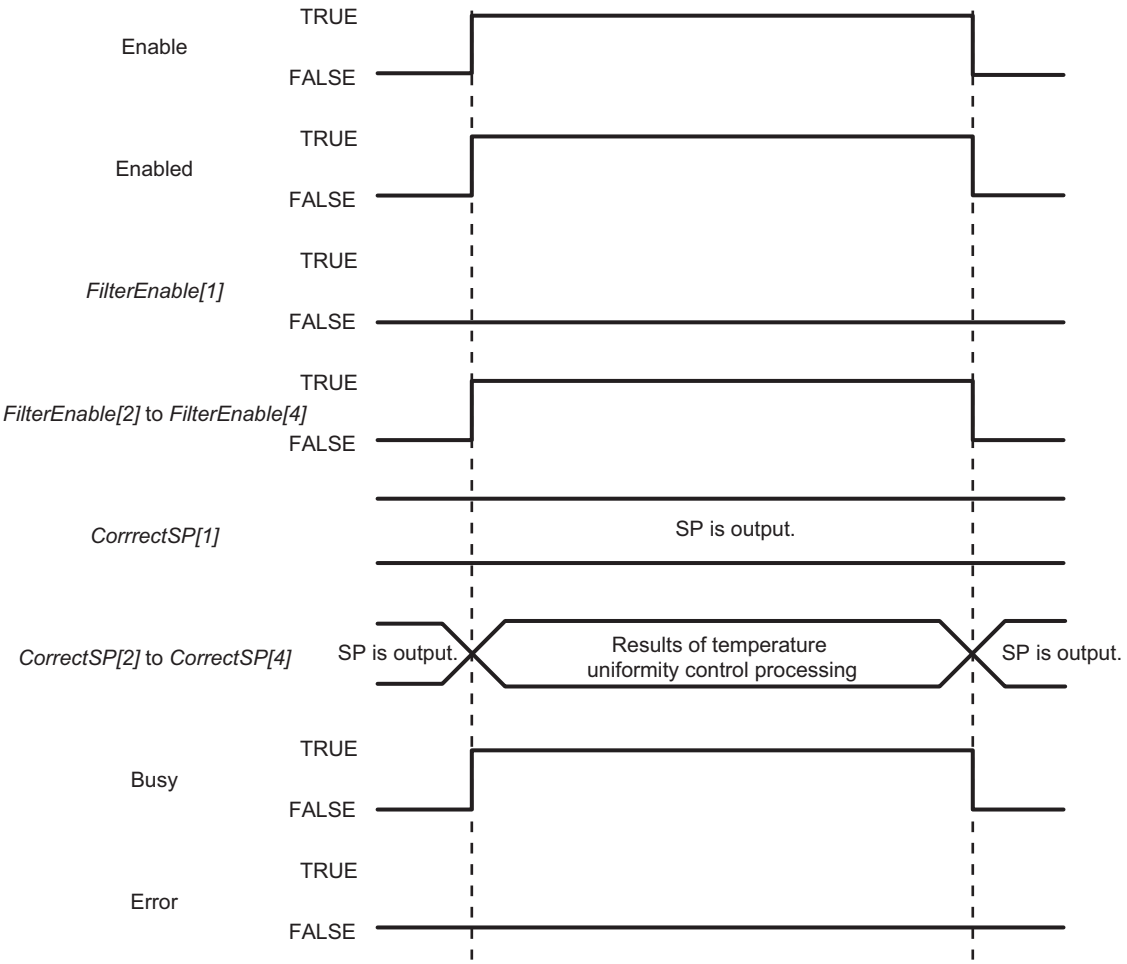
- When *Enable* changes to TRUE, *Enabled* and *Busy* (Executing) change to TRUE and temperature uniformity control processing is performed.
- When *Enable* changes to FALSE, *Enabled* and *Busy* (Executing) change to FALSE and temperature uniformity control processing is stopped. While *Enable* is FALSE, the input *SP* is output as is to *CorrectSP* (Corrected SP). While *Enable* is FALSE, the input *SP* is output as is to *CorrectSP* (Corrected SP).

### ● Timing Chart for Normal End (for Four Loops)

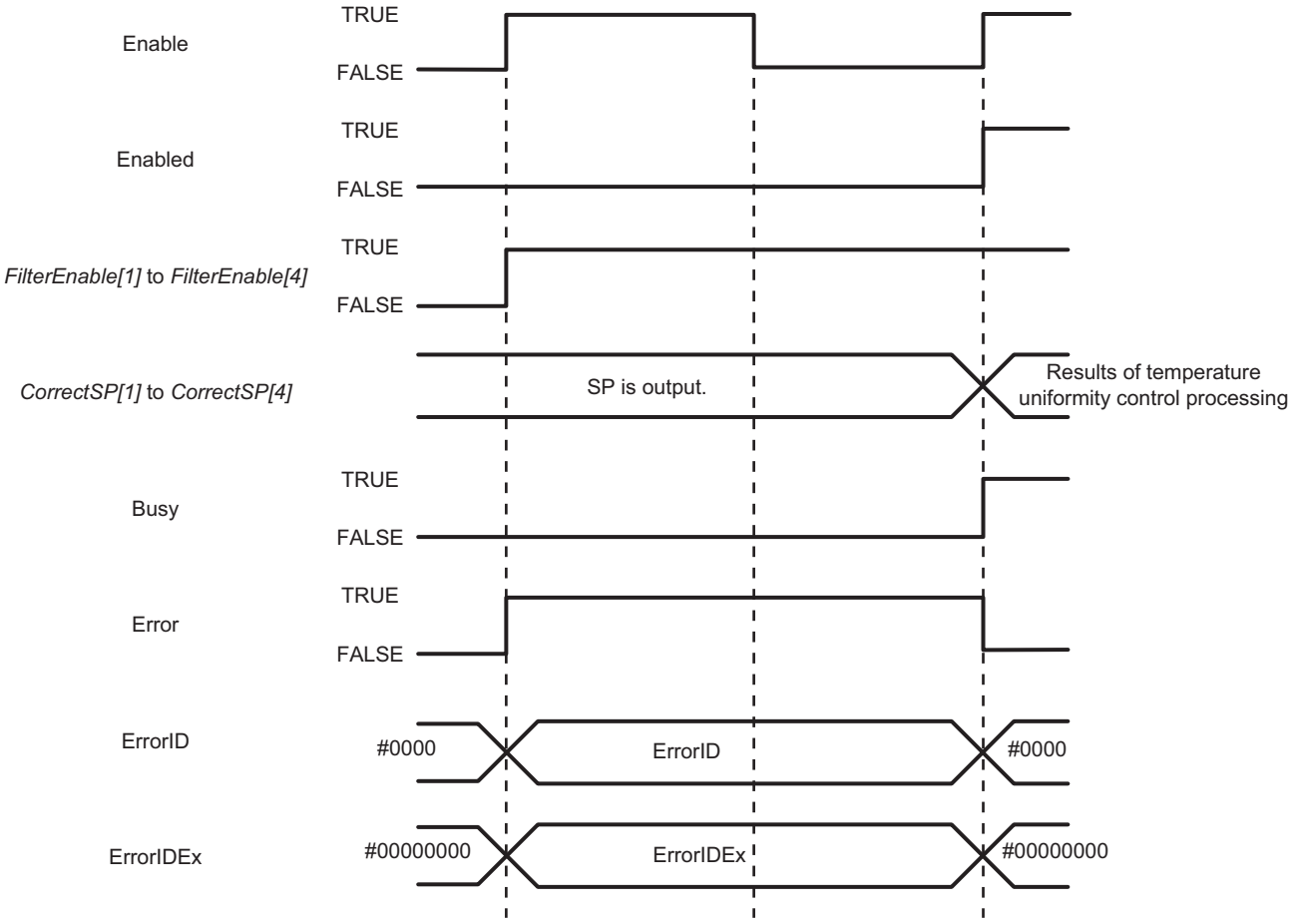



- Temperature uniformity control is performed only for loops for which the corresponding element in *FilterEnable* is set to TRUE and the results of temperature uniformity control processing are output to *CorrectSP* (Corrected SP). If an element in *FilterEnable* changes to FALSE for a loop, the *SP* input to the function block is output as is to *CorrectSP* (Corrected SP) for that loop.

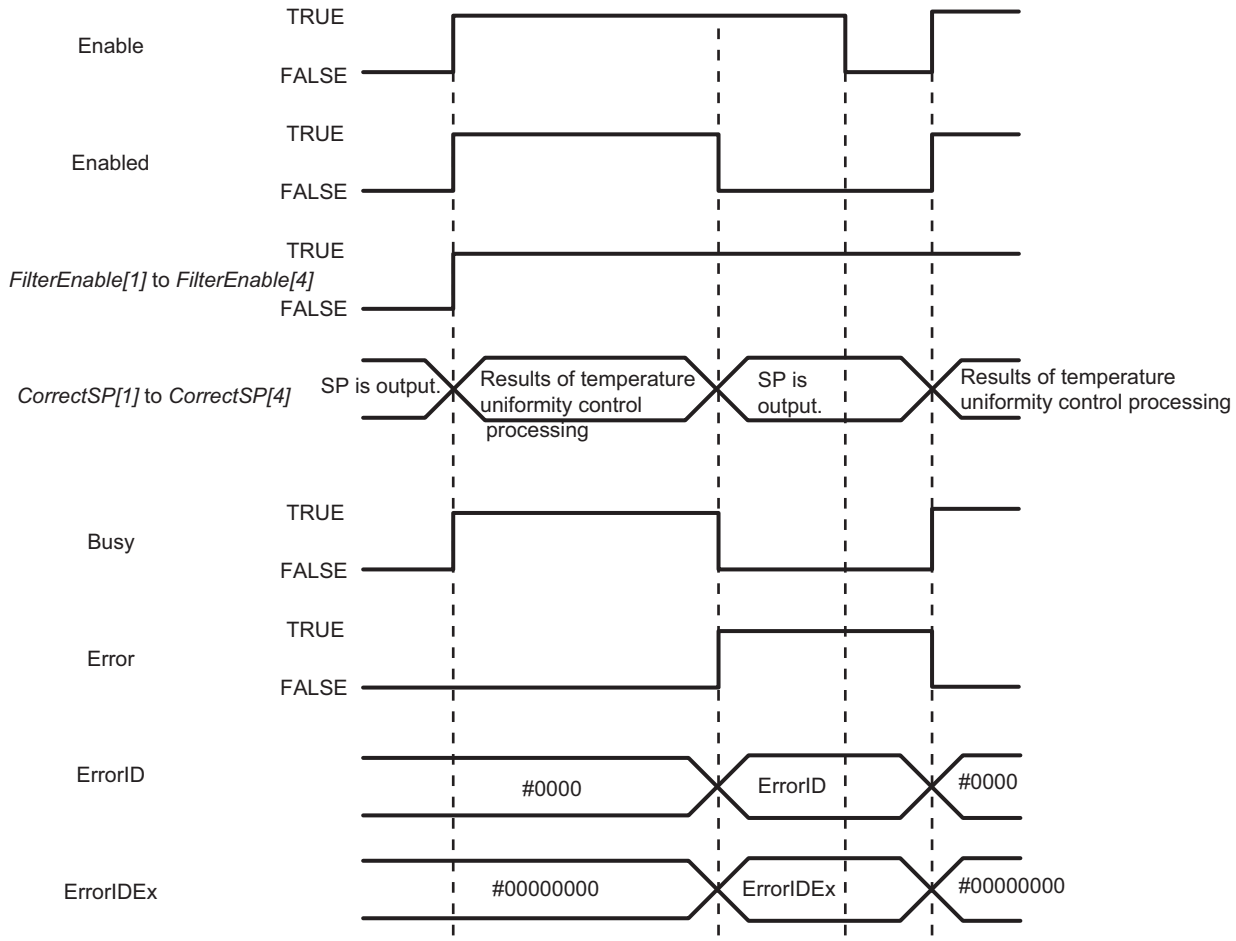
- **Timing Chart for Normal End (*FilterEnable[1]* = FALSE and *FilterEnable[2]* to *FilterEnable[4]* = TRUE)**



- 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).
- If an error occurs, *SP* is output to *CorrectSP* (Corrected SP).
- When *Enable* to this function block changes to TRUE, *ErrorID* (Error Code) and *ErrorIDEx* (Expansion Error Code) are cleared.



- 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).
- If an error occurs, *SP* is output to *CorrectSP* (Corrected SP).
- 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 | The service ended normally. | --- | --- |
| 16#3C13 | 16#00000001 | Reference Point Input Value Out of Range | The value of  Reference Point is out of the valid range. | Correct the value set for Reference Point so that it is within the valid range. |
| 16#3C13 | 16#00000002 | Reference Point Index Method Setting Out of Range | The value set for *RefPointIndex* (Reference Point Index) exceeds the number of loops enabled (TRUE) in *FilterEnable*. | Set *RefPointIndex* (Reference Point Index) so that it does not exceed the number of loops enabled (TRUE) in *FilterEnable*. |
| 16#3C13 | 16#00000003 | Set Point Input Value Out of Range | The value of *SP* (Set Point) is out of the valid range. | Set the value of *SP* (Set Point) within the range of -3200,00 to 3200,00. |
| 16#3C13 | 16#00000004 | Process Value Input Value Out of Range | The value of *PV* (Process Value) is out of the valid range. | Set the value of *PV* (Process Value) within the range of -3200,00 to 3200,00. |
| 16#3C13 | 16#00000005 | Filter Gain Input Value Out of Range | The value of *FilterGain* (Filter Gain) is out of the valid range. | Set the value of *FilterGain* (Filter Gain) within the range of 0 to 200. |

# Sample Programming

## Description of Operation

Connect the *CorrectSP (Corrected SP)* output parameter for this function block to the *SP* (Set Point) input parameter of the PIDAT instruction to uniform the temperatures of nine loops.

The set point is 1,000 for all loops. The gain is 100% for all loops.

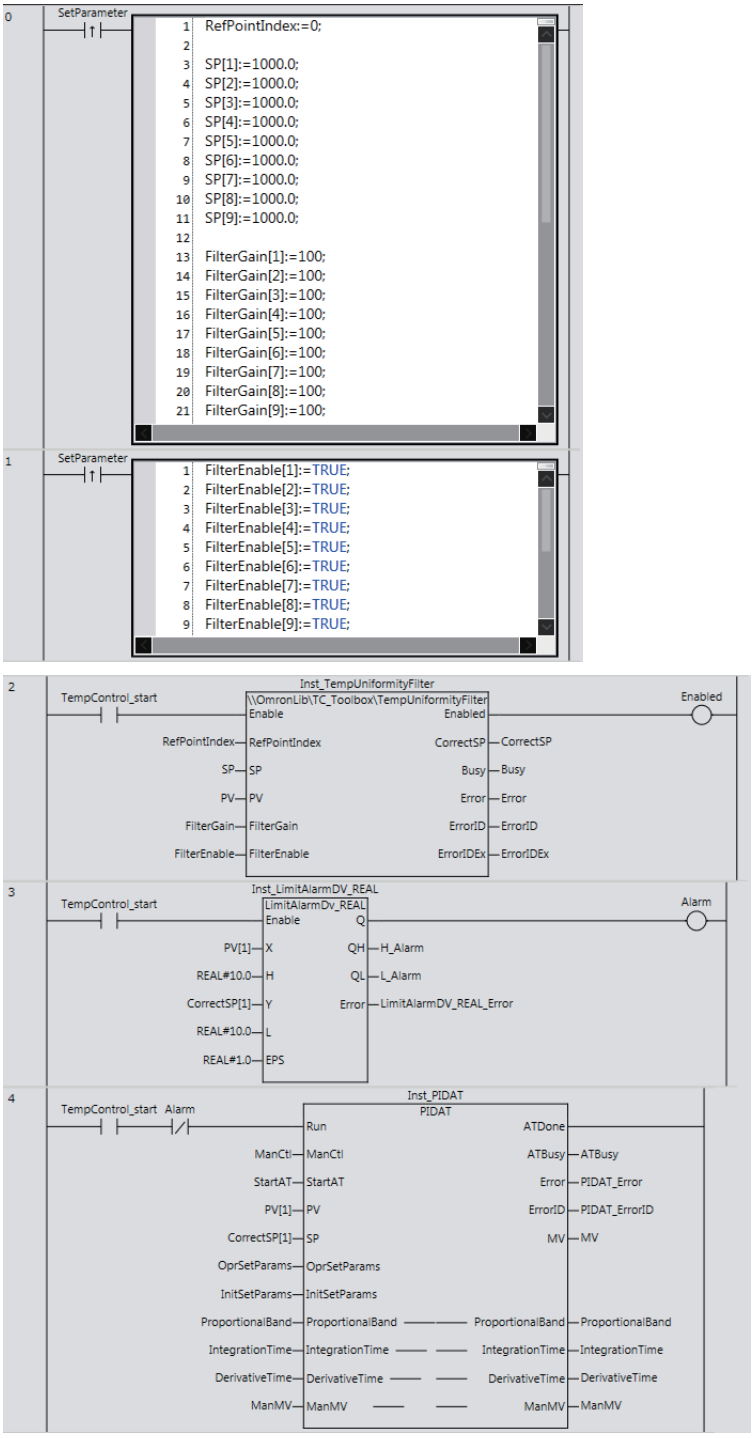
This sample program shows only the PIDAT instruction for one loop. Nine PIDAT instructions are required in practice.

## Variables

### ● Internal Variables

| Name | Data type | Default | Comment |
|---|---|---|---|
| SetParameter | BOOL | | Set Parameter |
| Inst_TempUniformityFilter | OmronLib\TC_Toolbox\TempUniformity-Filter | | Instance of TempUniformityFilter FB |
| Inst_LimitAlarmDV_REAL | LimitAlarmDv_REAL | | |
| LimitAlarmDV_REAL_Error | BOOL | | |
| H_Alarm | BOOL | | |
| L_Alarm | BOOL | | |
| Alarm | BOOL | | Alarm |
| RefPointIndex | UINT | | Reference Point Index |
| SP | ARRAY[1..9] OF REAL | | Set Point |
| PV | ARRAY[1..9] OF REAL | | Process Value |
| FilterGain | ARRAY[1..9] OF UINT | | Filter Gain |
| FilterEnable | ARRAY[1..9] OF BOOL | | Filter Enable |
| TempControl_start | BOOL | | Temperature Control Start |
| CorrectSP | ARRAY[1..9] OF REAL | | Correct SP |
| Enabled | BOOL | | Enabled |
| Busy | BOOL | | Busy |
| Error | BOOL | | Error |
| ErrorID | WORD | | ErrorID |
| ErrorIDEx | DWORD | | ErrorIDEx |
| Inst_PIDAT | PIDAT | | |
| OprSetParams | _sOPR_SET_PARAMS | | |
| InitSetParams | _sINIT_SET_PARAMS | | |
| ManCtl | BOOL | | |
| StartAT | BOOL | | |
| ProportionalBand | REAL | | |
| IntegrationTime | TIME | | |
| ManMV | REAL | | |
| DerivativeTime | TIME | | |
| ATBusy | BOOL | | |
| PIDAT_Error | BOOL | | |
| PIDAT_ErrorID | WORD | | |
| MV | REAL | | |

## Ladder Diagram



# 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 65.

*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 65.

*2. It is provided in English and Japanese.

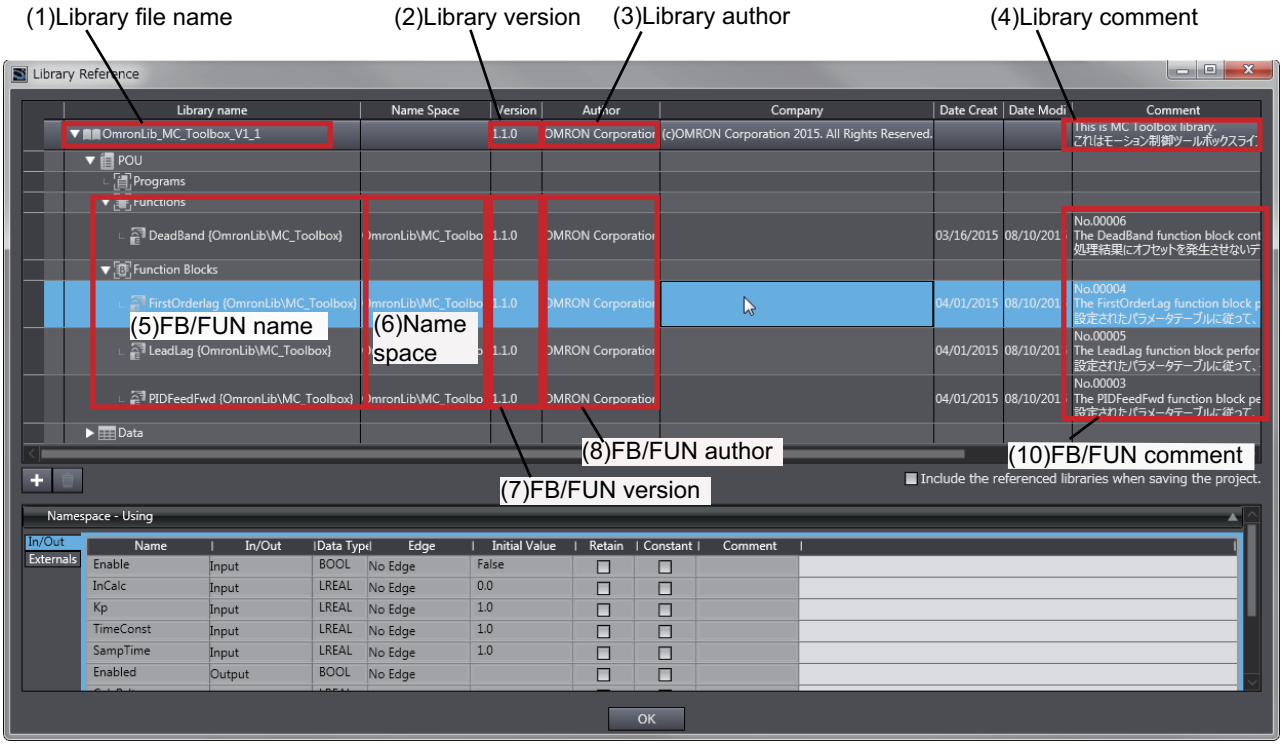
## Referring to Attributes of Libraries, Function Blocks and Functions

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

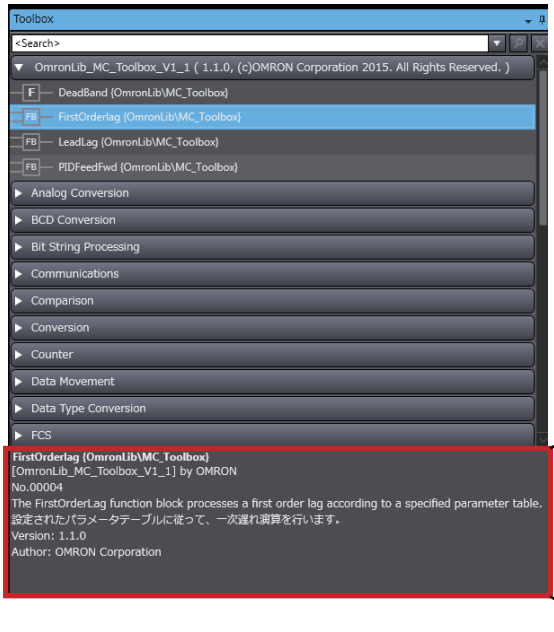
When you refer to the libraries, the library information is displayed at the locations shown below.



(b) Toolbox Pane

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

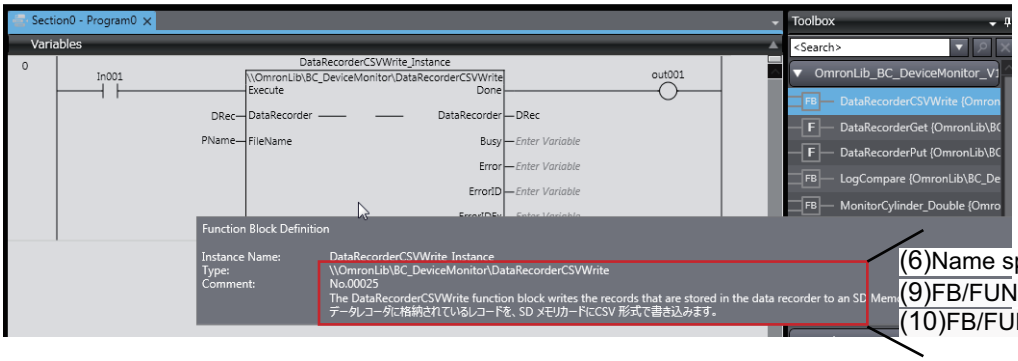
The text “by OMRON” which is shown on the right of the library name (1) indicates that this library was provided by OMRON.




(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.

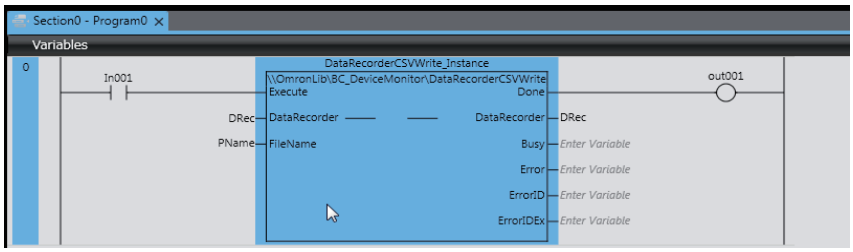
The following are the examples of items that you may need to customize.

- 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.

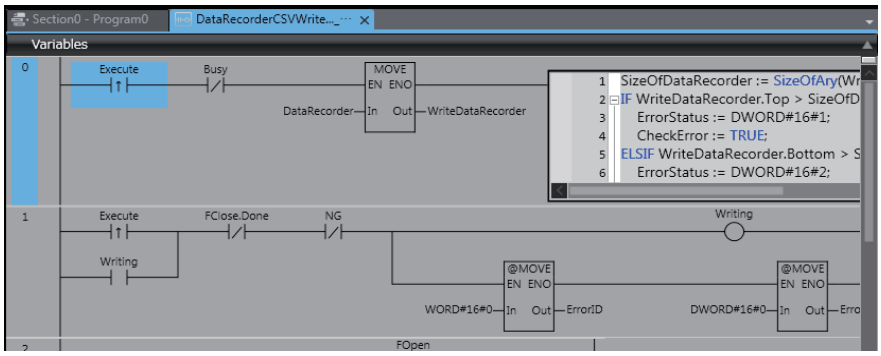
Use the following procedure to refer to the source codes of function blocks and functions.

***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. W551-E1-04** 0119