OMRON

Sysmac Library

User's Manual for RFID Communications Library

SYSMAC-XR019

SYSMAC always in control

W609-E1-02

NOTE

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

Trademarks

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

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

Copyrights

Microsoft product screen shots reprinted with permission from Microsoft Corporation.

Introduction

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

This manual provides information required to use the function blocks in the NX-series RFID Unit NX-V680C□. (“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 combinationrestrictions 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 RFID Communications Library provides a function for reading/writing the memory of RF Tag when performing production management based on individual recognition by using an NJ/NX-series CPU Unit, an NY-series Industrial PC, and an NX-series RFID Unit NX-V680C□. Using the RFID Communications Library allows you reduce the programming works during the implementation of the processing of the RFID Unit.

Intended Audience

This manual is intended for the following personnel,

who must also have knowledge of electrical systems (an electrical engineer or the equivalent).

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

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

Applicable Products

This manual covers the following products.

Item	Product name	Model	Version
Sysmac Library	NX-V680 Library	SYSMAC-XR019	Version 1.0.0 or later
Automation software	Sysmac Studio	SYSMAC-SE□□□□	Version 1.23 or later
Devices	CPU Unit	NX701-□□□□ NJ101-□□□□	Version 1.18 or later
		NJ501-□□□□ NJ301-□□□□	Version 1.18 or later
		NX1P2-□□□□□□□(1)	Version 1.18 or later
		NX102-□□□□	Version 1.30 or later
	Industrial PC	NY5□□-1□□□	Version 1.18 or later
	NX-series RFID Unit	NX-V680C□	Version 1.0 or later

Manual Structure

Icon

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.

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.

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 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 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 circle 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 circle 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.

Caution

⚠ WARNING

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.	
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.	
To ensure that the actual device operates as intended, check the user program, data, and parameter settings for proper execution before you use them for actual operation.	
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.	
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 library.
- Do not turn OFF the power supply to the Controller or the Communications Coupler Unit or stop communications until the processing for the function blocks provided in the library ends normally or ends in an error.

Using Sample Programming

- The sample programming shows only the portion of a program that uses the function or function block from the library.
- Create a user program so that the actual device operates as intended.
- Check the user program for proper execution before you use it for actual operation.

Related Manuals

The following table shows related manuals. Use these manuals for reference.

Manual name	Cat. No.	Model numbers	Application	Description
NX-series RFID Units User's Manual	Z401	NX-V680C□	Learning how to use NX-series RFID Units.	The hardware, setup methods, and functions of the NX-series RFID Units are described.
V680-series RF Tags and Amplifiers (FRAM Type) User's Manual	Z248	V680-HA63B V680-HS□□ V680-H01-V2 V680-D2K□□□□□ V680-D8K□□□□ V680S-D2K□□□□ V680S-D8K□□□□	Learning about the specifications, performance, and installation of the V680-series RF tags and amplifiers (FRAM type).	The general specifications, communications specifications, and installation method of the V680-series RF tags and amplifiers (FRAM type) are described.
V680-series RF Tags and Amplifiers (EEPROM Type) User's Manual	Z262	V680-HA63A V680-HS□□ V680-H01-V2 V680-D1KP□□□□	Learning about the specifications, performance, and installation of the V680-series RF tags and amplifiers (EEPROM type).	The general specifications, communications specifications, and installation method of the V680-series RF tags and amplifiers (EEPROM type) are described.
NX-series Data Reference Manual	W525	NX-□□□□□□	Referencing lists of the data that is required to configure systems with NX-series Units	Lists of the power consumptions, weights, and other NX Unit data that is required to configure systems with NX-series Units are provided.
NX-series System Units User's Manual	W523	NX-PD1□□□ NX-PF0□□□ NX-PC0□□□ NX-TBX01	Learning how to use NX-series System Units	The hardware and functions of the NX-series System Units are described.
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.
NX-IO Configurator Operation Manual	W585	CXONE-AL□□D-V4	Learning about the operating procedures and functions of the NX-IO Configurator.	Describes the operating procedures of the NX-IO Configurator.
NJ/NX-series Troubleshooting Manual	W503	NX701-□□□□ NJ501-□□□□ NJ301-□□□□ NJ101-□□□□ NX1P2-□□□□	Learning about the errors that may be detected in an NJ/NX-series Controller	Concepts on managing errors that may be detected in an NJ/NX-series Controller and information on individual errors are described.
NY-series Troubleshooting Manual	W564	NY532-□□□□ NY512-□□□□	Learning about the errors that may be detected in an NY-series Industrial PC	Concepts on managing errors that may be detected in an NY-series Controller and information on individual errors are described.

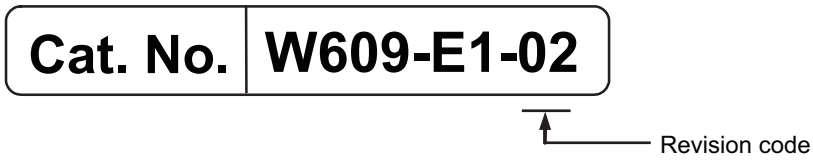
Manual name	Cat. No.	Model numbers	Application	Description
NX-series EtherCAT® Coupler Unit User's Manual	W519	NX-ECC20□	NX-series Learning how to use the EtherCAT Coupler Unit and EtherCAT Slave Terminals.	The following items are described: the overall system and configuration methods of an EtherCAT Slave Terminal (which consists of an NX-series EtherCAT Coupler Unit and NX Units), and information on hardware, setup, and functions to set up, control, and monitor NX Units through EtherCAT.
NX-series EtherNet/IP™ Coupler Unit User's Manual	W536	NX-EIC202	Learning how to use an NX-series EtherNet/IP Coupler Unit and EtherNet/IP Slave Terminals	The following items are described: the overall system and configuration methods of an EtherNet/IP Slave Terminal (which consists of an NX-series EtherNet/IP Coupler Unit and NX Units), and information on hardware, setup, and functions to set up, control, and monitor NX Units.
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 NX102CPU Unit Hardware User's Manual	W593	NX102-□□□□	Learning the basic specifications of the NX-series NX102 CPU Units, including introductory information, designing, installation, and maintenance. Mainly hardware information is provided.	An introduction to the entire NX102 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 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

Manual name	Cat. No.	Model numbers	Application	Description
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 User's Manual NY-series IPC MachineController IndustrialPanel PC HardwareUser'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 • Overview • Part names and functions • General specifications • Installation and wiring • Maintenance and inspection
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 • Overview • Part names and functions • General specifications • Installation and wiring • Maintenance and inspection
NJ/NX-series CPU Unit Software User's Manual	W501	NX701-□□□□ NJ501-□□□□ NJ301-□□□□ NJ101-□□□□ NX1P2-□□□□	Learning how to program and set up an NJ/NX-series CPU Unit. Mainly software information is provided.	The following information is provided on 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

Manual name	Cat. No.	Model numbers	Application	Description
NJ/NX-series CPU Unit Built-in EtherCAT® Port User's Manual	W505	NX701-□□□□ NJ501-□□□□ NJ301-□□□□ NJ101-□□□□ NX1P2-□□□□	Using the built-in EtherCAT port on an NJ/NX-series CPU Unit	Information on the built-in EtherCAT port is provided. This manual provides an introduction and provides information on the configuration, features, and setup.
NY-series IPC Machine Controller Industrial Panel PC / Industrial Box PC Built-in EtherCAT® Port User's Manual	W562	NY532-□□□□ NY512-□□□□	Using the built-in EtherCAT port on an NY-series Industrial PC	Information on the built-in EtherCAT port is provided. This manual provides an introduction and provides information on the configuration, features, and setup.
NJ/NX-series Instructions Reference Manual	W502	NX701-□□□□ NJ501-□□□□ NJ301-□□□□ NJ101-□□□□ NX1P2-□□□□	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.

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	October 2018	Original production
02	May 2021	Corrected mistakes

Procedure to Use Sysmac Libraries

Procedure to Use Sysmac Libraries Installed Using the Installer

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

There are two ways to use libraries.

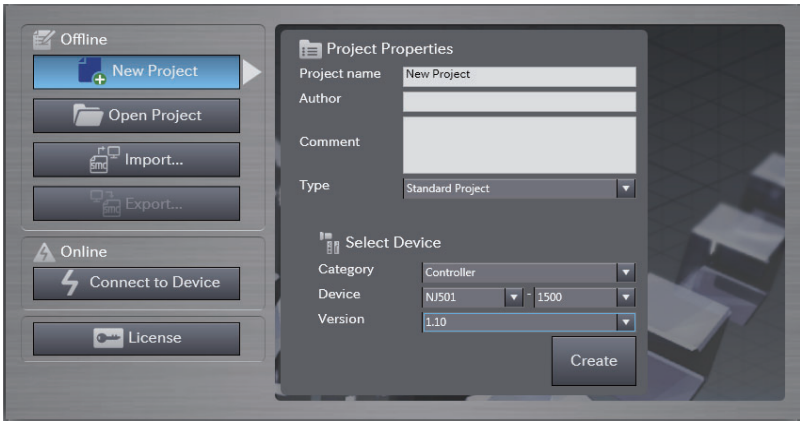
- Using newly installed Sysmac Libraries
- Using upgraded Sysmac Libraries

Version Information

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

Using Newly Installed Libraries

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

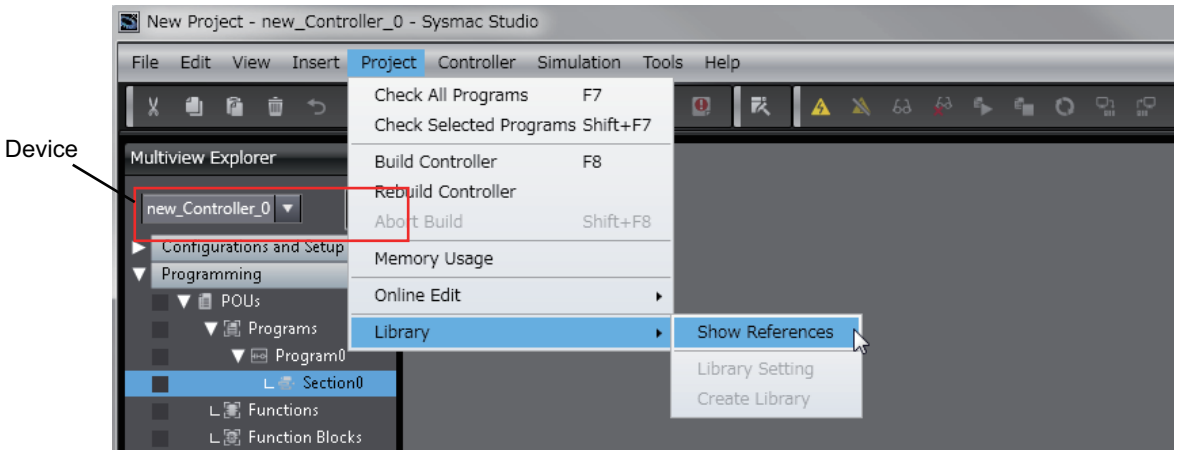


Precautions for Correct Use

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

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

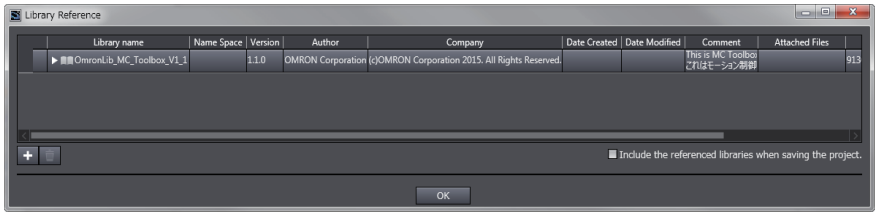
2 Select **Project** – **Library** – **Show References**.



Precautions for Correct Use

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

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



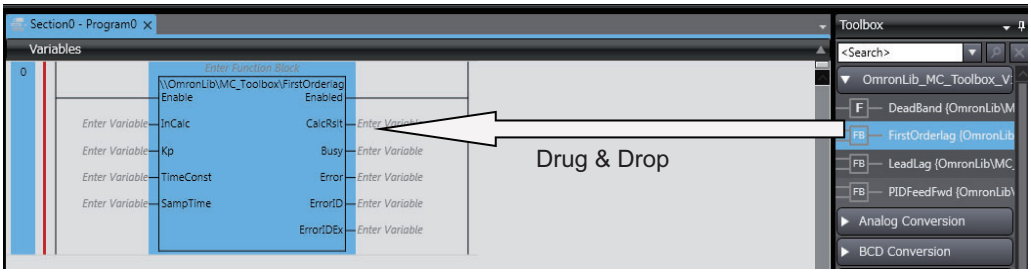
The Sysmac Library file is read into the project.

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

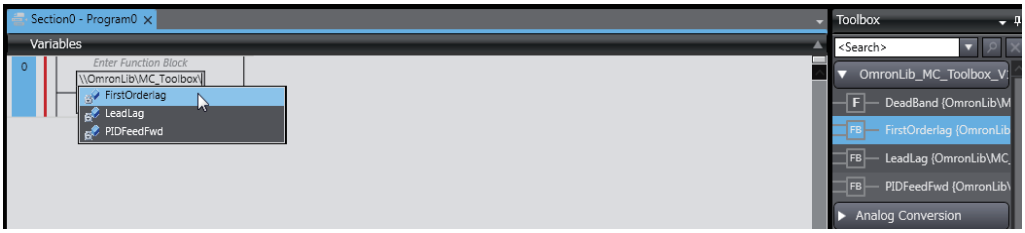
For the procedure for adding and setting libraries in the above screen, refer to the *Sysmac Studio Version 1 Operation Manual* (Cat. No. W504).

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

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



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



Precautions for Correct Use

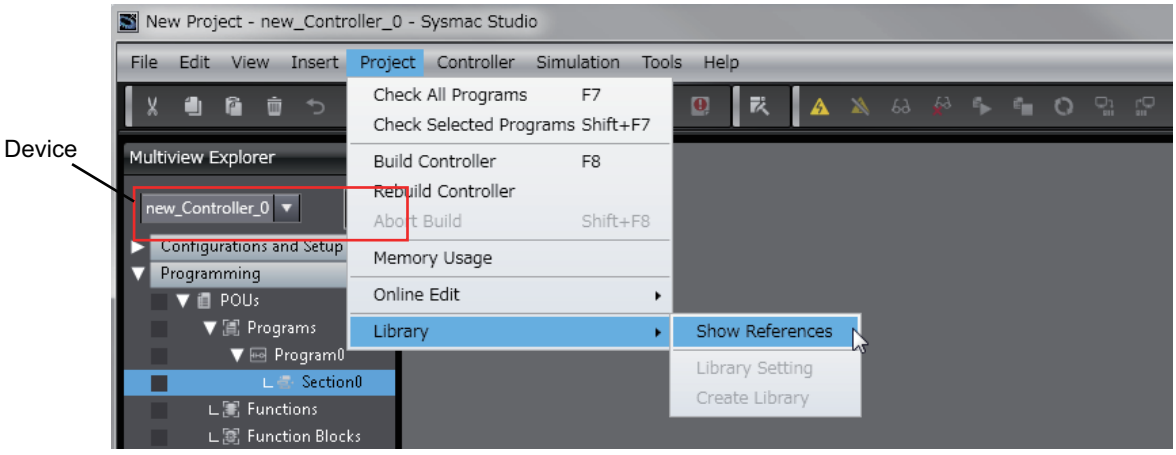
After you upgrade the Sysmac Studio, check all programs and make sure that there is no error of the program check results on the Build Tab Page.

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

Using Upgraded Libraries

1 Start the Sysmac Studio and open a project in which any old-version Sysmac Library is included.

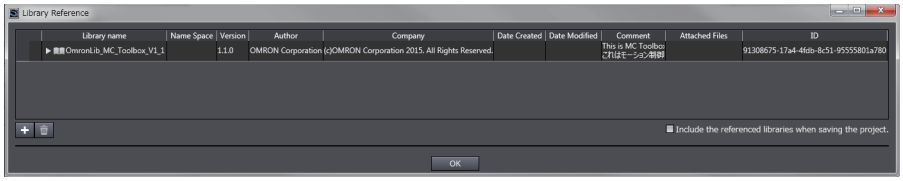
2 Select **Project** – **Library** – **Show References**.



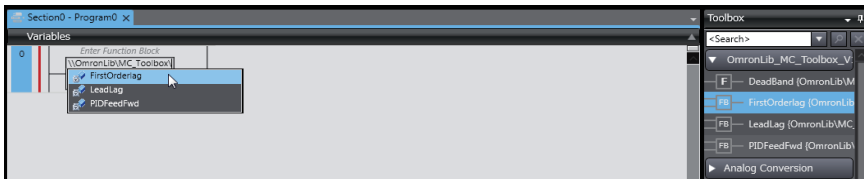
Precautions for Correct Use

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

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



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



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

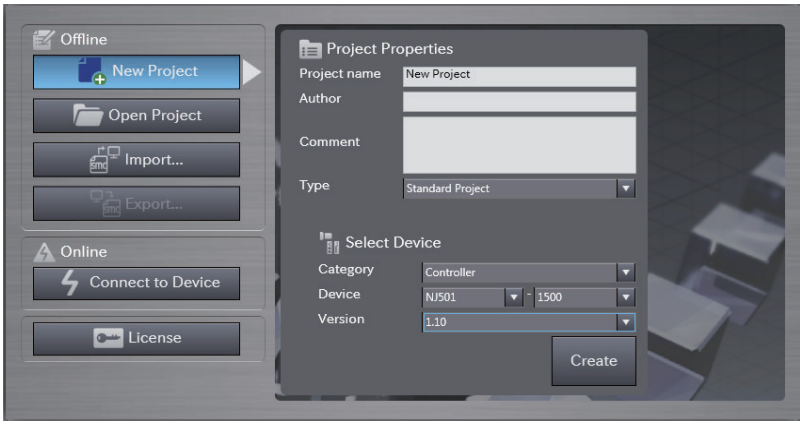
You can use Sysmac Libraries uploaded from a CPU Unit or an Industrial PC to your computer if they are not installed.

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

Version Information

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

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



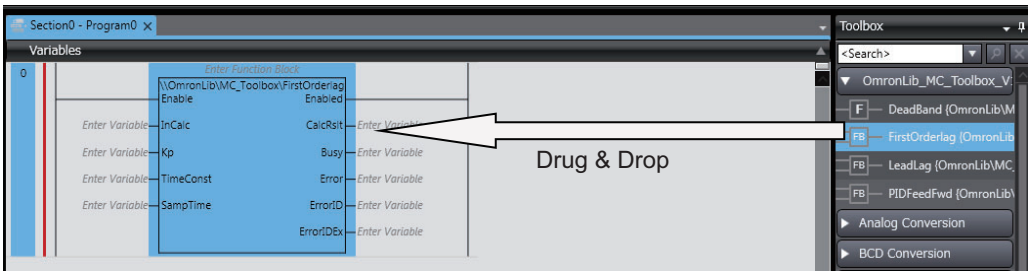
2 Connect the computer to the CPU Unit or the Industrial PC and place it online.

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

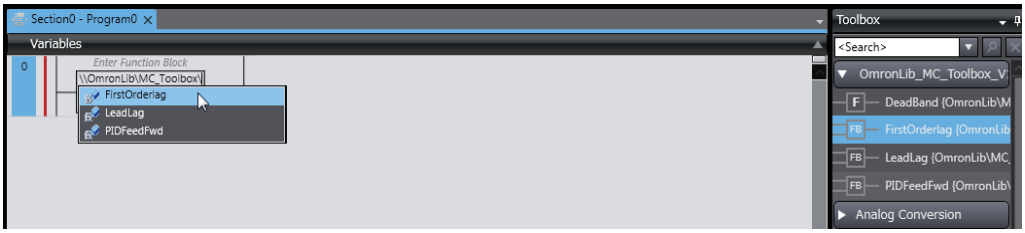
Now, when you select the Ladder Editor or ST Editor, the function blocks and functions included in the Sysmac Library used in the uploaded POUs appear in the Toolbox.

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

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



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



Precautions for Correct Use

- The Sysmac Studio installs library files of the uploaded Sysmac 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.

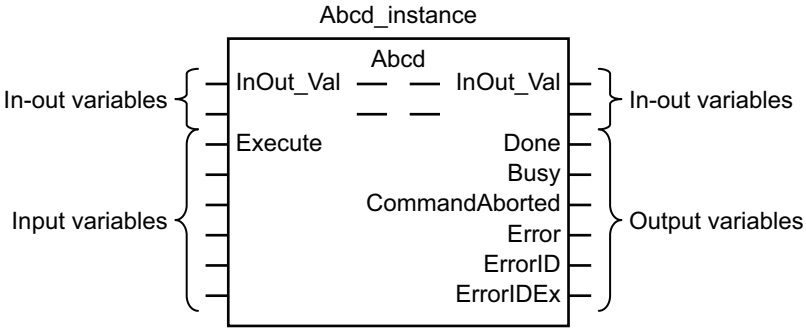
Definition of Input Variables and Output Variables

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.
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.

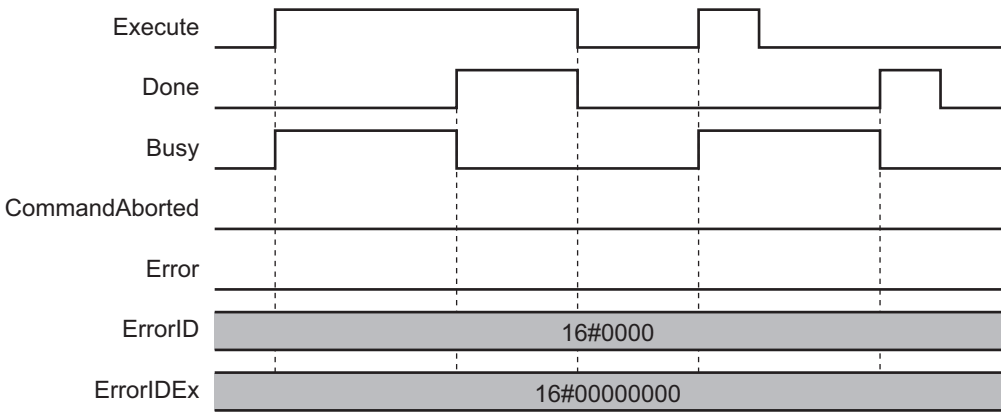


- If *Execute* is TRUE and *Done*, *CommandAborted*, or *Error* changes to TRUE, *Done*, *CommandAborted*, and *Error* changes to FALSE when *Execute* is changed to FALSE.
- If *Execute* is FALSE and *Done*, *CommandAborted*, or *Error* changes to TRUE, *Done*,*CommandAborted*, and *Error* changes to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

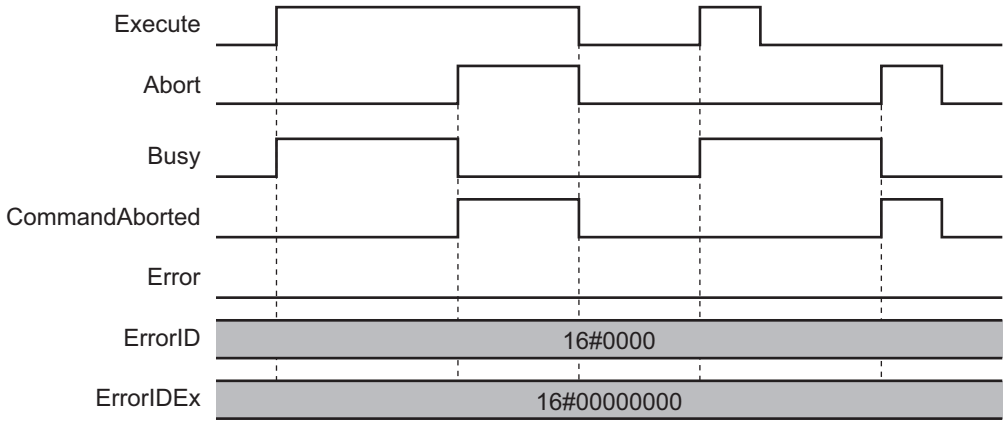
Timing Charts

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

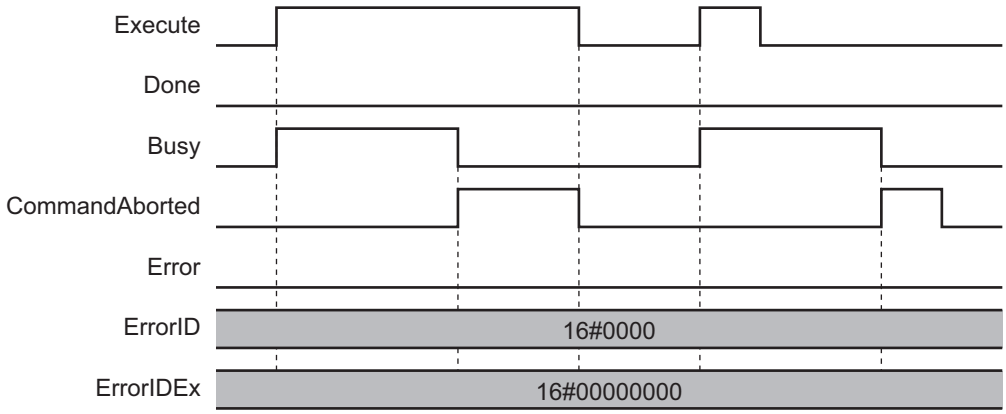
● Normal End



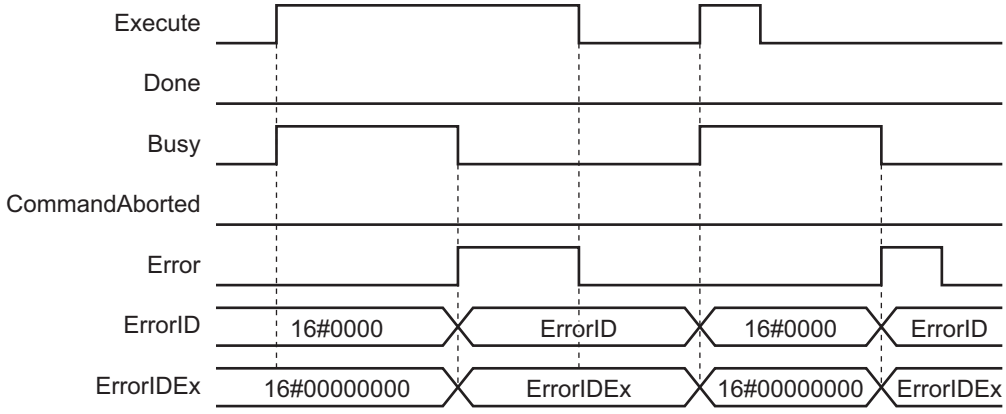
● Canceled Execution



● Aborted Execution

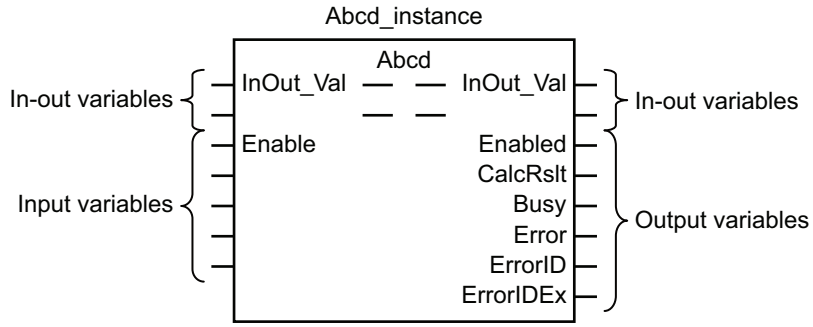


● Errors



Enable-type Function Blocks

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

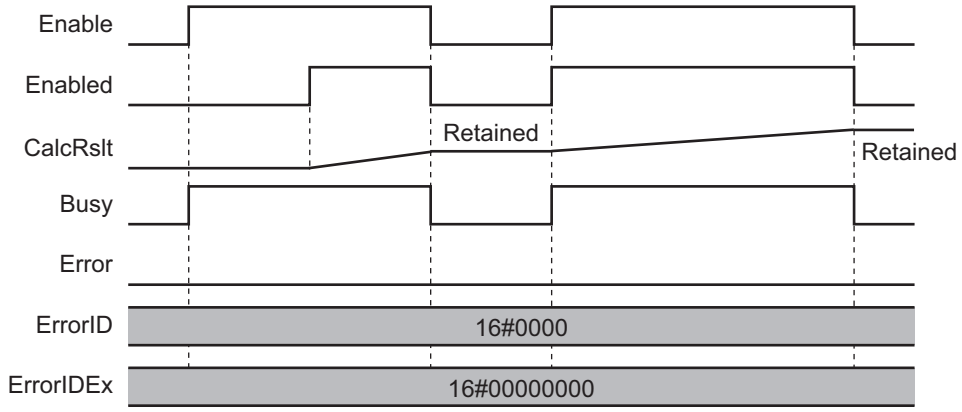


- If an error occurs, the relevant error code and expansion error code are set in *ErrorID* (Error Code) and *ErrorIDEx* (Expansion Error Code). The error codes are retained even after *Error* changes to FALSE, but *ErrorID* is set to 16#0000 and *ErrorIDEx* is set to 16#0000 0000 when *Enable* changes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

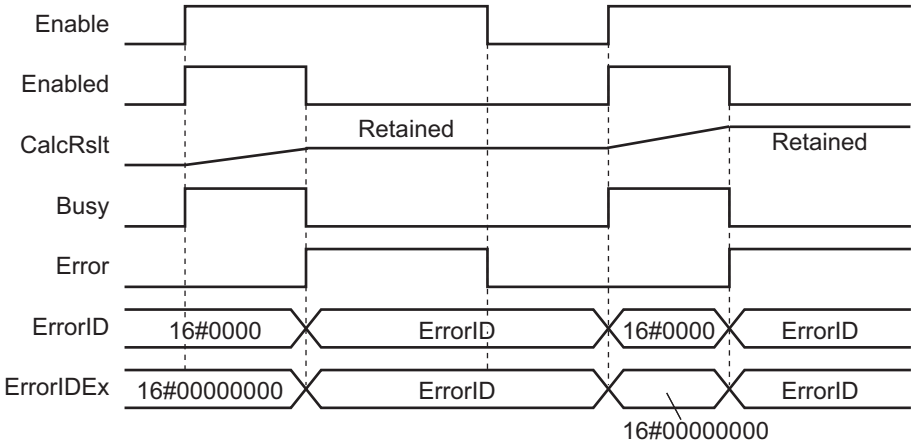
Timing Charts

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

● Normal End



● Errors


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.

Individual Specifications of Function Blocks

Function block name	Name	Page
ReadData_V680	RF Tag data read for V680	P.36
WriteData_V680	RF Tag data write for V680	P.59

Common Data Types for All Function Blocks

This section describes the data types used in common in all function blocks included in the “NX_V680” library.

Structure *sV680DeviceVariables*

Members	Name	Data Type	Valld range	Description
Status	Status	WORD	Depends on data type.	Allocated from the I/O map *Status* in the input area of the RFID Unit.
RefreshCount	Refresh Count	UINT	0 to 65535	Allocated from the I/O map *Refresh Count* in the input area of the RFID Unit.
ResponseCode	Response Code	WORD	Depends on data type.	Allocated from the I/O map *Response Code* in the input area of the RFID Unit.
InputSID	Input SID	UINT	0 to 512	Allocated from the I/O map *Input SID* in the input area of the RFID Unit.
OutputSIDResponse	Output SID Response	UINT	0 to 512	Allocated from the I/O map *Output SID Response* in the input area of the RFID Unit.
InputData1	Input Data 1	ARRAY[0..15] OF BYTE	Depends on data type.	Allocated from the I/O map *Input Data 1* in the input area of the RFID Unit.
UID	UID	ARRAY[0..7] OF BYTE	Depends on data type.	Allocated from the I/O map *UID* in the input area of the RFID Unit.
RFCommunicationsTime	RF Communications Time	UINT	0 to 65535	Allocated from the I/O map *RF Communications Time* in the input area of the RFID Unit.
NoiseLevel	Noise Level	UINT	0 to 100	Allocated from the I/O map *Noise Level* in the input area of the RFID Unit.
OperationCommand	Operation Command	BYTE	Depends on data type.	Allocated to the I/O map *Operation Command* in the output area of the RFID Unit.
RFCommunicationsOption	RF Communications Option	USINT	Depends on data type.	Allocated to the I/O map *RF Communications Option* in the output area of the RFID Unit.
CommandCode	Command Code	WORD	Depends on data type.	Allocated to the I/O map *Command Code* in the output area of the RFID Unit.
MemoryAddress	Memory Address	UINT	Depends on data type.	Allocated to the I/O map *Memory Address* in the output area of the RFID Unit.
DataSize	Data Size	UINT	Depends on data type.	Allocated to the I/O map *Data Size* in the output area of the RFID Unit.
RefreshCountResponse	Refresh Count Response	UINT	0 to 65535	Allocated to the I/O map *Refresh Count Response* in the output area of the RFID Unit.
OutputSID	Output SID	UINT	0 to 512	Allocated to the I/O map *Output SID* in the output area of the RFID Unit.
InputSIDResponse	Input SID Response	UINT	0 to 512	Allocated to the I/O map *Input SID Response* in the output area of the RFID Unit.

Members	Name	Data Type	Valld range	Description
OutputData1	Output Data 1	ARRAY[0..15] OF BYTE	Depends on data type.	Allocated to the I/O map *Output Data 1* in the output area of the RFID Unit.
SelectUID	Select UID	ARRAY[0..7] OF BYTE	Depends on data type.	Allocated to the I/O map *Select UID* in the output area of the RFID Unit.

Precautions for Correct Use

- The I/O allocation settings of the RFID Unit used in the structure *sV680DeviceVariables* are as described below.

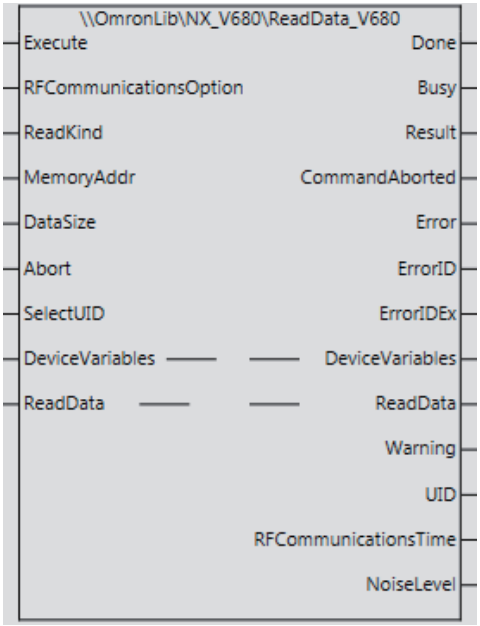
  Refer to Section 6 I/O Data Specifications in the NX-series RFID Units User's Manual (Z401) for details on the I/O entry mapping of the RFID Unit.

Area	Data name	Size (Byte)	Data type	Assigned.
Output	Chn Status	2	WORD,BOOL	Fixed
	Chn Refresh Count	2	UINT	Fixed
	Chn Response Code	2	WORD	Fixed
	Chn Measurement Result	2	UINT?	Fixed
	Chn Input SID	2	UINT	Fixed
	Chn Output SID Response	2	UINT	Fixed
	Chn Input Data 1	16	ARRAY[0..15] OF BYTE	Fixed
	Chn UID	8	ARRAY[0..7] OF BYTE	Variable
	Chn RF Communications Time	2	UINT	Variable
	Chn Noise Level	2	UINT	Variable
output	Chn Operation Command	1	BYTE,BOOL	Fixed
	Chn RF Communications Option	1	USINT	Fixed
	Chn Command Code	2	WORD	Fixed
	Chn Memory Address	2	UINT	Fixed
	Chn Data Size	2	UINT	Fixed
	Chn Refresh Count Response	2	UINT	Fixed
	Chn Output SID	2	UINT	Fixed
	Chn Input SID Response	2	UINT	Fixed
	Chn Output Data 1	16	ARRAY[0..15] OF BYTE	Fixed
	Chn Select UID	8	ARRAY[0..7] OF BYTE	Variable

- When the 2CH Unit NX-V680C2 is used, do not set together the I/O ports of 1CH and 2CH in the structure *sV680DeviceVariables*. Doing so may result in the malfunction of the function block. Always arrange the I/O ports of the same CH.

ReadData_V680

This command reads data from the memory of an RF Tag in the communications range of the antenna mounted on the RFID Unit (NX-V680).

Program part names	Series	FB/ FUN	Graphic expression	STexpression
ReadD-ata_V680	RF Tag data read for V680	FB		ReadData_V680_instance(Execute := <Parameter>, RFCommunicationsOption := <Parameter>, ReadKind := <Parameter>, MemoryAddr := <Parameter>, DataSize := <Parameter>, Abort := <Parameter>, SelectUID := <Parameter>, Done => <Parameter>, Busy => <Parameter>, Result => <Parameter>, CommandAborted => <Parameter>, Error => <Parameter>, ErrorID => <Parameter>, ErrorIDEx => <Parameter>, Warning => <Parameter>, UID => <Parameter>, RFCommunicationsTime => <Parameter>, NoiseLevel => <Parameter>, ReadData := <Parameter>, DeviceVariables := <Parameter>,);

Function Block and Function Information

Item	Description
Library file name	OmronLib_NX_V680_Vx_x.slr (x indicates the version)
Namespace	OmronLib\NX_V680
Function block and function number	00213
Source code	Do not publish.

Input variable

Variables	Name	Data Type	Description	Valid range	Unit	Initial value
Execute	Execute	BOOL	The processing is started when the variable changes to TRUE. TRUE: Execute FALSE: Do not execute	TRUE or FALSE	---	FALSE
RFCommunicationsOption	RF Communications Option	USINT	Specify the operation sequence during communications. 0: Trigger 1: Auto 2: Repeat 3: FIFO Trigger 4: FIFO Repeat 5: Multi Trigger 6: Multi Repeat 7: Selective	0 to 7	---	0
ReadKind	Read Type	USINT	Specify the type of the read command.*1 0: Normal 1: With error detection 2: With error correction	0 to 2	---	0
MemoryAddr	Memory Address	UINT	Specify the start address of the memory to which data is read from the RF Tag.*2	0 to 65535	Byte	0
DataSize	Data Size	UINT	Enter the size of the data to be read from the RF Tag.*3	1 to 8192	Byte	0
Abort	Abort	BOOL	The processing is aborted when the variable changes to TRUE.	TRUE or FALSE	---	FALSE
SelectUID	Select UID	ARRAY[0..7]OF BYTE	Specify the UID of the communications target RF Tag when the RF communications option is *Selective*.	Depends on data type.	---	16#00

*1. Refer to the List of Commands in the *NX-series RFID Units User's Manual* (Z401) for details on the types of read command and differences in their operation.

*2. Enter a value in consideration of the memory map of the RF Tag actually used.

*3. Enter a value in consideration of the memory capacity of the RF Tag actually used. When 0 is specified, no operation is performed and the processing ends.

Output Variables

Variables	Name	Data Type	Description	Valid range	Unit	Initial value
Done	Done	BOOL	The variable changes to TRUE when the processing is completed.	TRUE or FALSE	---	FALSE
Busy	Busy	BOOL	The variable changes to TRUE when the processing is acknowledged.	TRUE or FALSE	---	FALSE
Result	Result	BOOL	The value of the variable alternately changes between TRUE and FALSE each time the result of communications with the RF Tag is output when the communications specification is set to *Repeat*, *FIFO repeat*, *Multi trigger*, or *Multi repeat*.	TRUE or FALSE	---	FALSE
Command Aborted	Instruction Aborted	BOOL	The variable changes to TRUE when the processing is aborted.	TRUE or FALSE	---	FALSE
Error	Error	BOOL	This variable is TRUE while there is an error. TRUE: Error end FALSE: Normal end or Executing	TRUE or FALSE	---	FALSE
ErroriD	Error code	WORD	An error code is output. This is the error ID for an error end. The value is 16#0 for a normal end.	*1	---	0
ErrorIDEx	Expansion error code	DWORD	An expansion error code is output. This is the error ID for an error end. The value is 16#0 for a normal end.	*1	---	0
Warning	Warning	BOOL	The variable changes to TRUE when an error correction occurs.	TRUE or FALSE	---	FALSE
UID	UID	ARRAY[0..7] OF BYTE	The UID of the RF Tag with which communications are performed is output.	Depends on data type.	---	16#00
RFCommunicationsTime	RF Communications Time	UINT	The measured RF communications time is output.	0 to 65535	ms	0
NoiseLevel	Noise Level	UINT	The measured noise level is output.	0 to 99	---	0

*1. For details, refer to *Troubleshooting* on page 52.

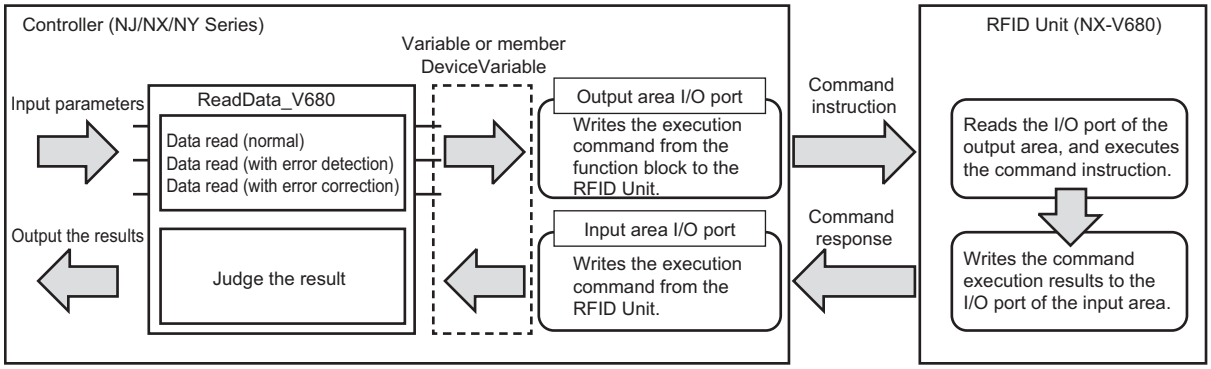
In-Out Variables

Variables	Name	Data Type	Description	Valid range	Unit	Initial value
DeviceVariables	device variable	Omron-Lib\RFID\sNX-V680Device-Variables	Device variables of the RFID Unit (I/O port of the input area / output area)	Depends on data type.	---	---
ReadData	Read Data	ARRAY [Variable length] OF BYTE	Data array read from the RF Tag	Depends on data type.	---	---

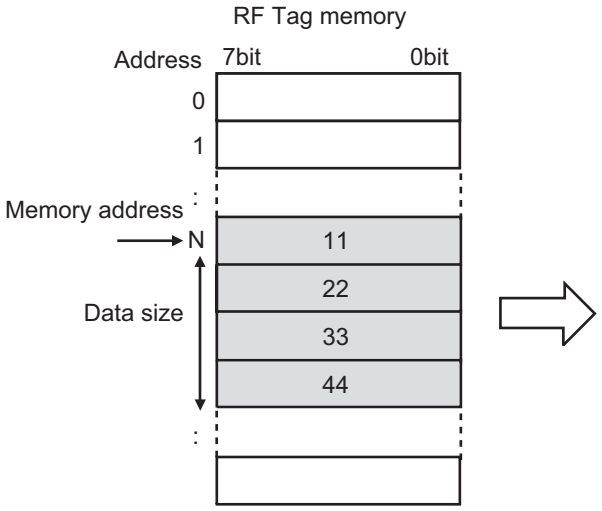
Note A BYTE array of any length can be specified. However, the array length must be equal to or more than the DataSize.
Both 0 and n can be set for the array start number.

Operation

This function block controls the command for reading the memory of the RF Tag for an RFID Unit (NX-V680) that is exchanging data with an NJ/NX/NY-series Controller. Therefore, the device variables (the I/O port of the output area and the I/O port of the input area) of the RFID Unit to be controlled must be set in the input/output variable *DeviceVariables* of the function block.



- When *Execute* changes to TRUE, the data read from the memory of the RF Tag is saved to the byte array specified by *ReadData*.



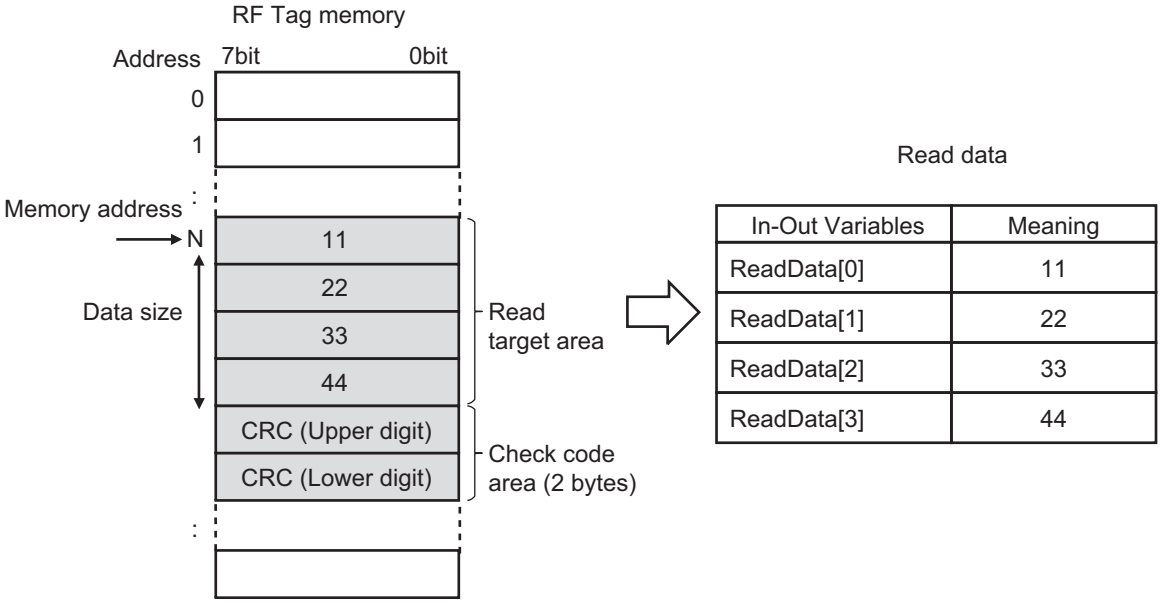
Read data

In-Out Variables	Meaning
ReadData[0]	11
ReadData[1]	22
ReadData[2]	33
ReadData[3]	44

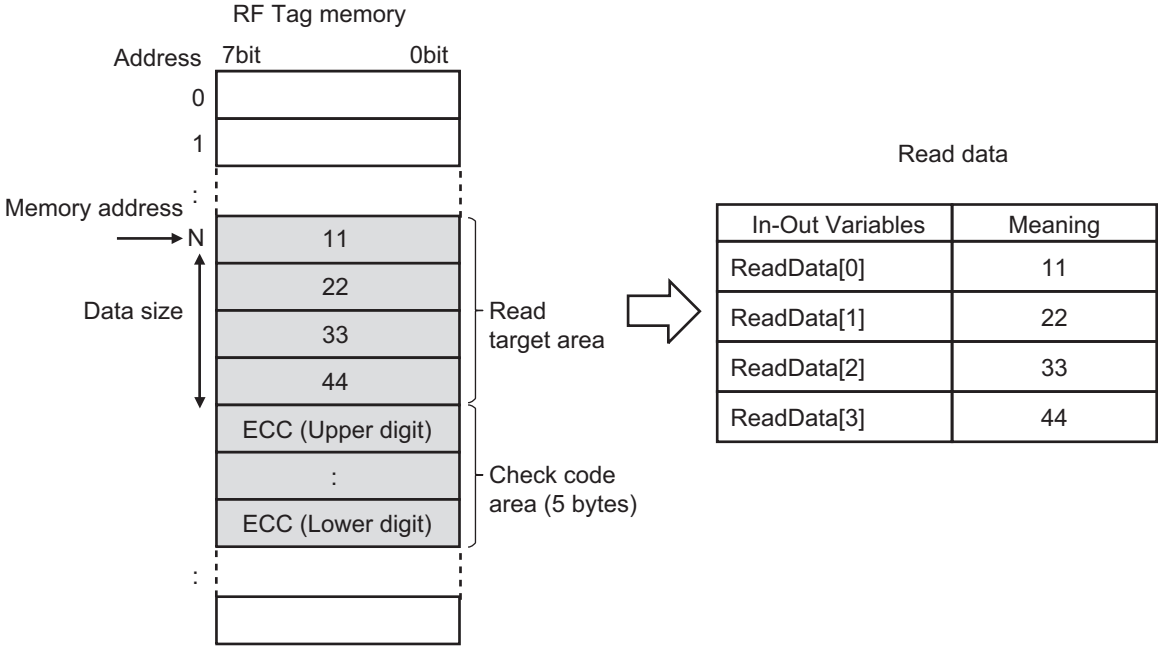
- The memory capacity read from the RF Tag in a single execution varies depending on the specified *ReadKind*. The values that can be specified in *MemoryAddr* and *DataSize* are described below.

ReadKind	MemoryAddr	DataSize
0: Normal	0 to 65535	1 to 8192
1: With error detection	0 to 65533	1 to 8190
2: With error correction	0 to 65530	1 to 510

- If *ReadKind* is executed as *With error detection*, the data and check code are read from the RF Tag, and errors in the data are detected. When using this function, write beforehand the data and check code in the target area as *With error detection* of the function block “WriteData_V680”.



- If *ReadKind* is executed as *With error correction*, the data and check code are read from the RF Tag, and errors in the data are detected, and 1-bit errors are corrected. When using this function, write beforehand the data and check code in the target area as *With error correction* of the function block “WriteData_V680”.



Note If the data and check code are not written beforehand in the target area as *With error detection* and *With error correction* of the function block “WriteData_V680” when *ReadKind* is executed as “With error detection”, a communications error (RF Tag Data Error Detected) will occur.

Timing Chart

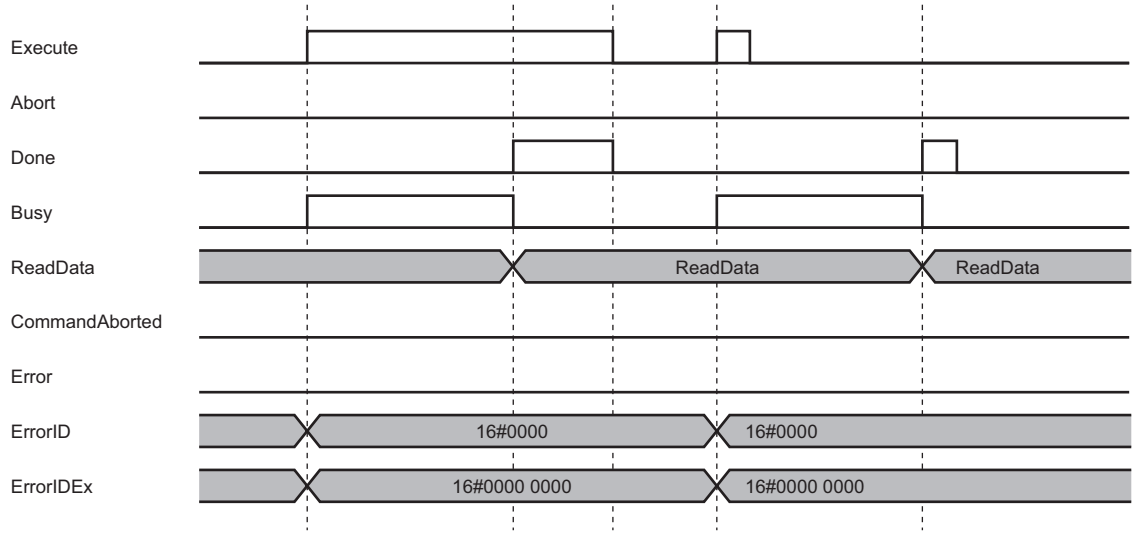
The timing charts are shown below.

Communicating with One RF Tag (RF Communications Option: Trigger, Auto, FIFO trigger or Selective)

If this function block is started by specifying Trigger, Auto, FIFO trigger, or Selective in *RFCommunicationsOption*, the result of communications with one RF Tag present in the communications range of the antenna is output.

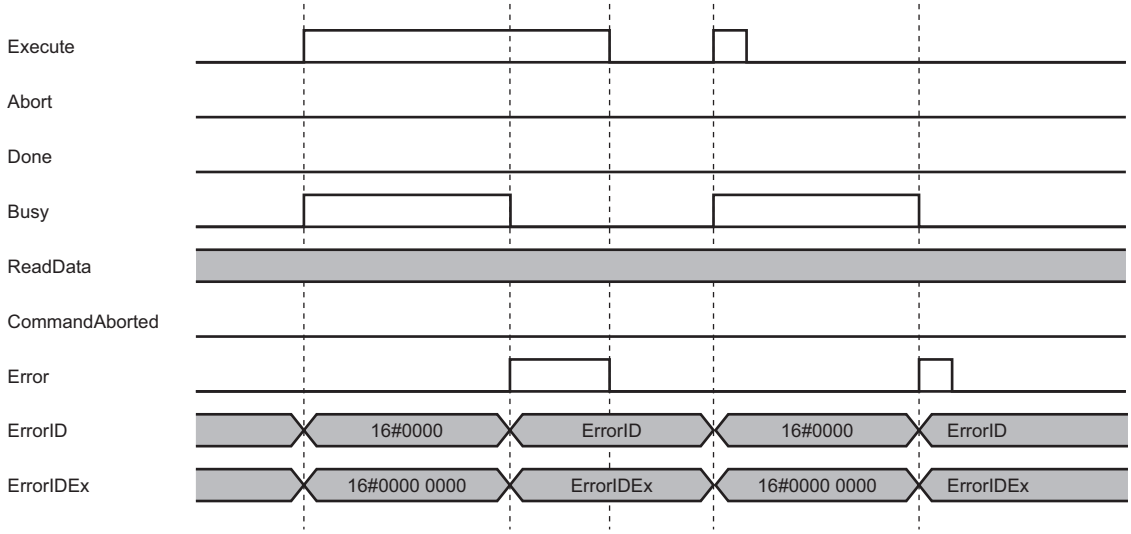
● Timing Chart for Normal End

- Busy (Executing) changes to TRUE when Execute (Execute) changes to TRUE.
- If reading of data from the memory of the RF Tag ends normally, *Done* changes to TRUE, and *Busy* (Executing) changes to FALSE. At the same time, a data string is output in *ReadData* and UID, and a value is output in *RFCommunicationsTime* and *NoiseLevel*.
- If *Execute* remains TRUE even after the execution of this function block is complete, the output value of *Done* is retained.
- If Execute changes to FALSE before the execution of this function block is complete, *Done* changes to TRUE only for one task period after the execution of the function block is complete.



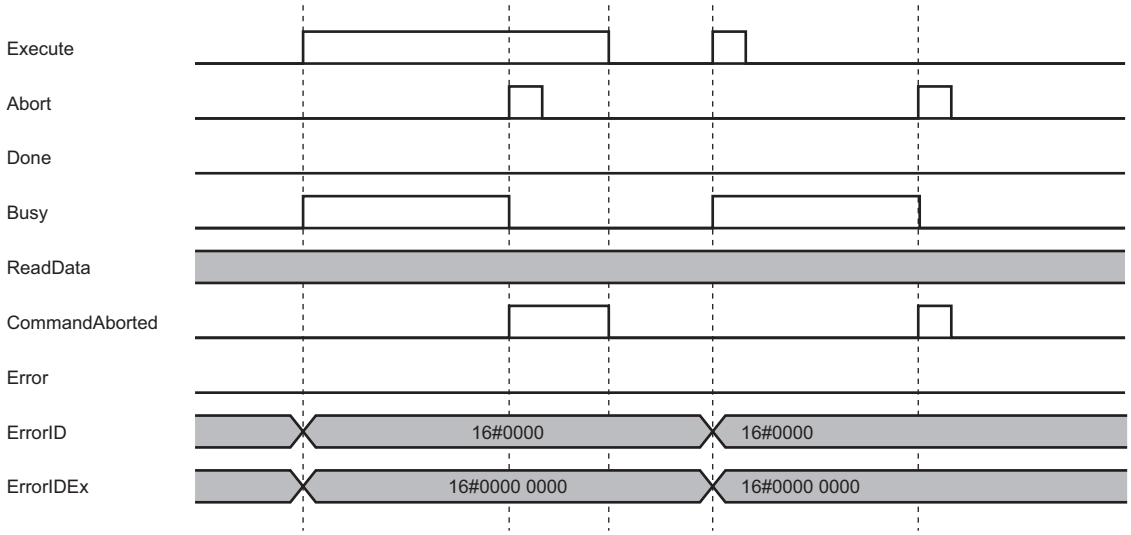
Timing Chart for Error End

- If an error occurs while the execution of this function block is in progress, *Error* changes to TRUE and *Busy* (Executing) changes to FALSE.
- You can find out the cause of the error by referencing the values output to *ErrorID* (Error Code) and *ErrorIDEx* (Expansion Error Code).
- If *Execute* remains TRUE even after the execution of this function block is complete, the output value of *Error* is retained.
- The output values of *ErrorID* (Error Code) and *ErrorIDEx* (Expansion Error Code) are retained until this function block is executed again.
- If *Execute* changes to FALSE before the execution of this function block is complete, *Error* changes to TRUE only for one task period after the execution of the function block is complete.



Timing Chart when the Function Block is Aborted

To abort communications with the RF Tag while the execution of this function block is in progress, set *Abort* to TRUE. If *Abort* changes to TRUE, *Busy* (Executing) changes to FALSE and *Command-Aborted* changes to TRUE.

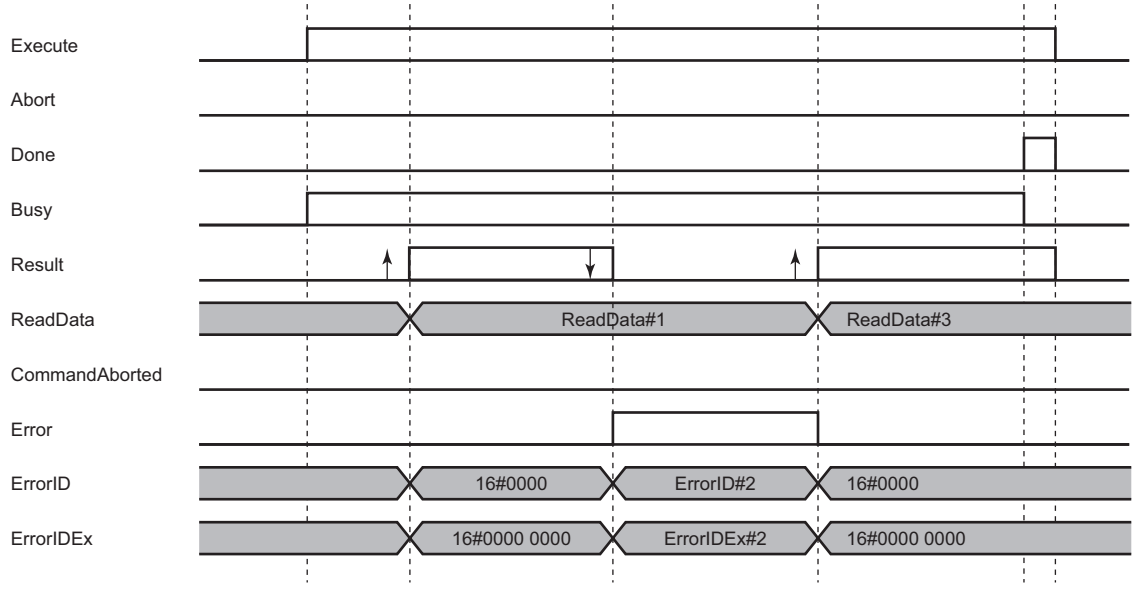


Communicating with Multiple RF Tags (RF Communications Option: Multi Trigger)

If this function block is started by specifying Multi trigger in *RFCommunicationsOption*, the result of communications with the multiple RF Tags present in the communications range of the antenna is output.

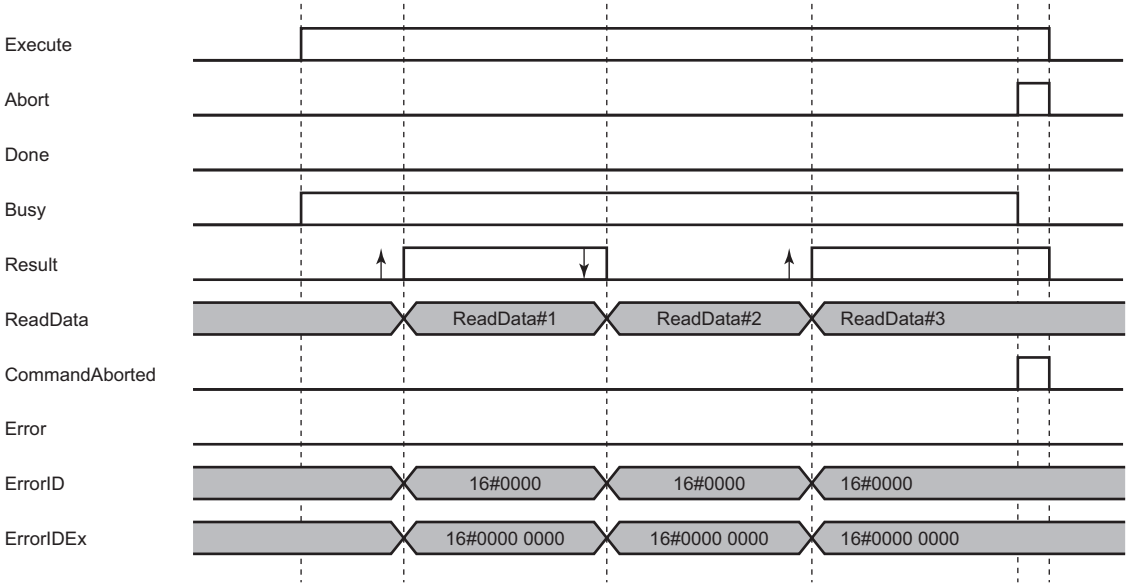
● Timing Chart for Communications with Multiple RF Tags

- Busy (Executing) changes to TRUE when Execute (Execute) changes to TRUE.
- Each time communications are performed with multiple RF Tags, the *Result* toggles between FALSE -> TRUE -> FALSE -> TRUE.
- When communications with all RF Tags in the communications range are complete, *Done* changes to TRUE and *Busy* (Executing) changes to FALSE.
- If reading of data from the memory of the RF Tag ends normally, a data string is output in *ReadData* and UID, and a value is output in *RFCommunicationsTime* and *NoiseLevel* at the same time as a change in *Result*.
- If reading of data from the memory of the RF Tag ends in an error, *Error* changes to TRUE at the same time as a change in *Result*. You can find out the cause of the error by accessing the values output to ErrorID (Error Code) and ErrorIDEx (Expansion Error Code).
- *Result* changes to FALSE at the same time when *Done* changes to FALSE.



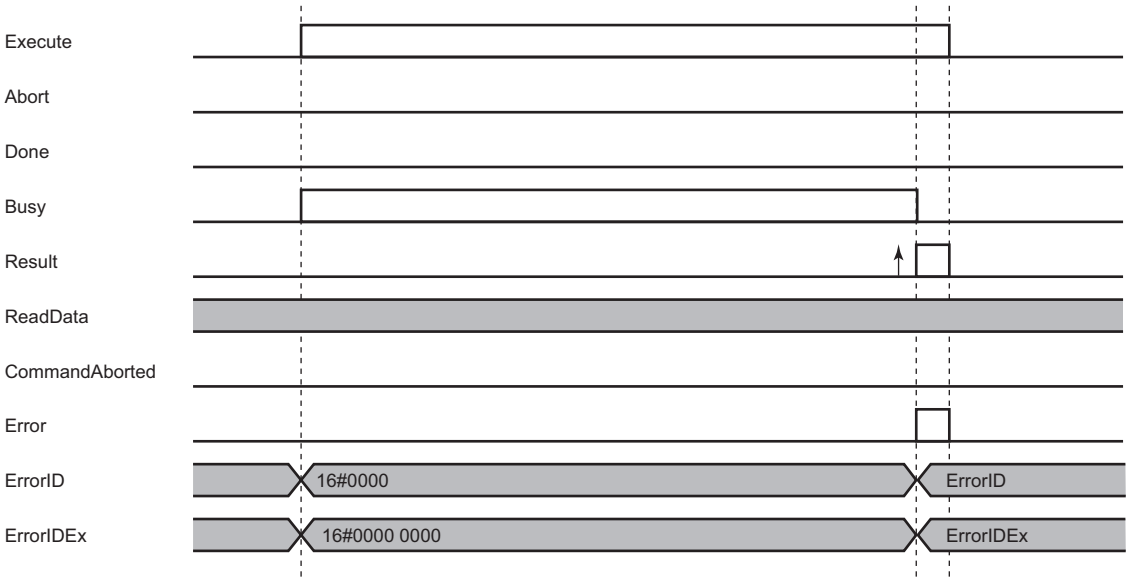
Timing Chart when the Function Block is Aborted

- To abort communications with the RF Tag while the execution of this function block is in progress, set *Abort* to TRUE. If *Abort* changes to TRUE, *Busy* (Executing) changes to FALSE and *CommandAborted* changes to TRUE.
- *Result* changes to FALSE at the same time when *CommandAborted* changes to FALSE.



Timing Chart when the Tag is Missing

- If communications are not performed with the RF Tag even once while the execution of this function block is in progress, *Result* and *Error* change to TRUE, and *Busy* (Executing) changes to FALSE.
- You can find out the cause of the error by accessing the values output to ErrorID (Error Code) and ErrorIDEx (Expansion Error Code).
- If *Execute* remains TRUE even after the execution of this function block is complete, the output values of *Result* and *Error* are retained.
- The output values of *ErrorID* (Error Code) and *ErrorIDEx* (Expansion Error Code) are retained until this function block is executed again.
- If *Execute* changes to FALSE before the execution of this function block is complete, *Result* and *Error* change to TRUE only for one task period after the execution of the function block is complete.

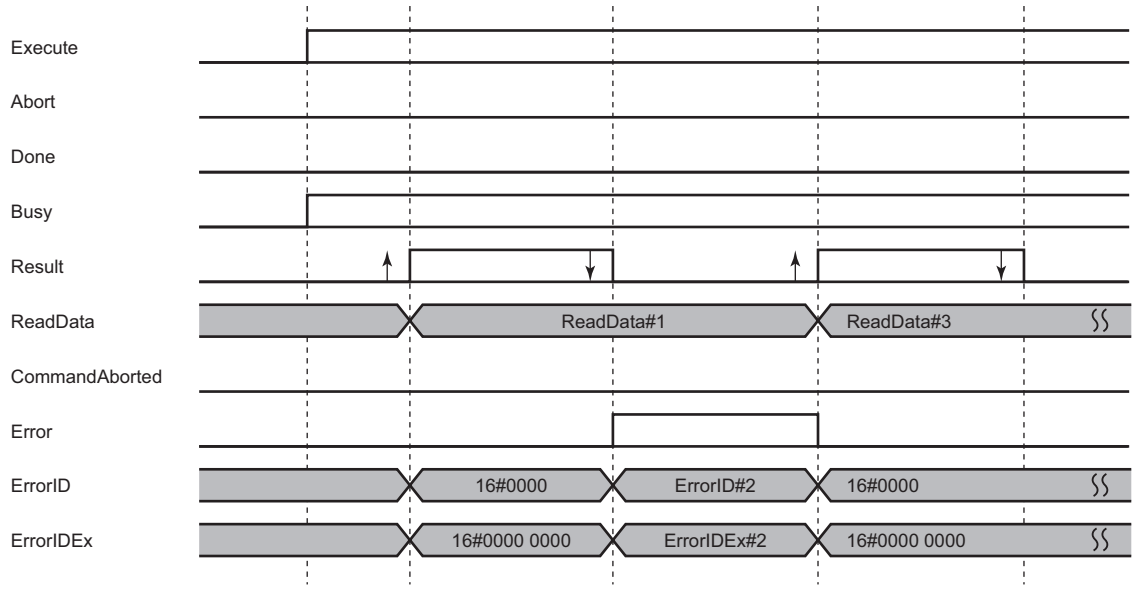


Communicating Repeatedly with an RF Tag (RF Communications Option: Repeat, FIFO Repeat or Multi Repeat)

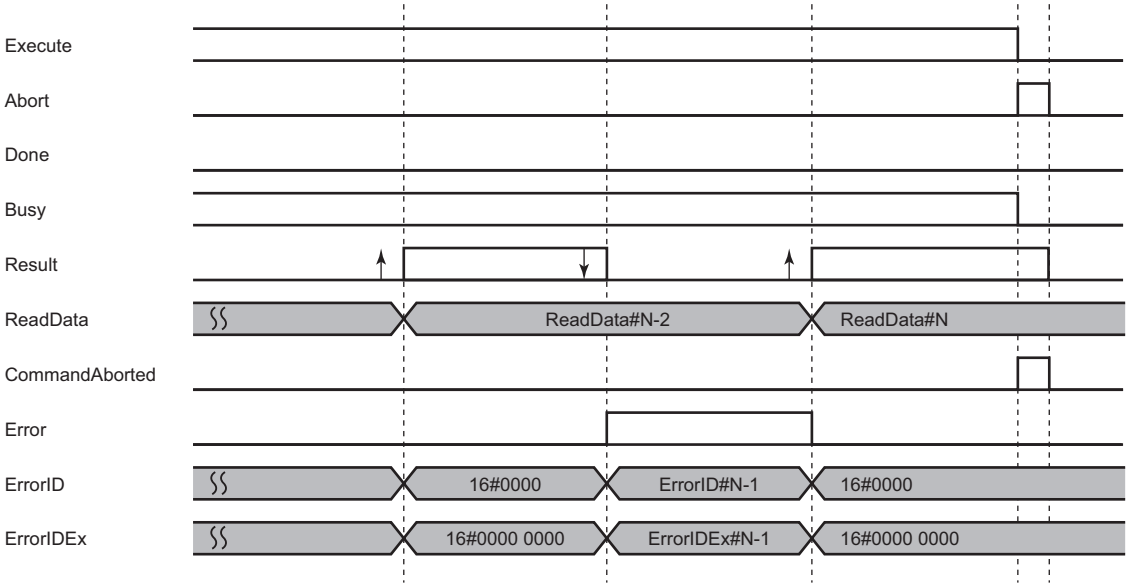
If this function block is started by specifying Repeat, FIFO repeat, or Multi-access repeat in *RFCommunicationsOption* (RF Communications Option), the result of repeated communications with an RF Tag that has moved into the communications range of the antenna is output.

● Timing Chart for Repeated Communications

- Busy (Executing) changes to TRUE when Execute (Execute) changes to TRUE.
- Each time communications are performed with multiple RF Tags, the *Result* toggles between FALSE -> TRUE -> FALSE -> TRUE.
- If reading of data from the memory of the RF Tag ends normally, a data string is output in *ReadData* and UID, and a value is output in *RFCommunicationsTime* and *NoiseLevel* at the same time as a change in *Result*.
- If reading of data from the memory of the RF Tag ends in an error, *Error* changes to TRUE at the same time as a change in *Result*. You can find out the cause of the error by accessing the values output to ErrorID (Error Code) and ErrorIDEx (Expansion Error Code).

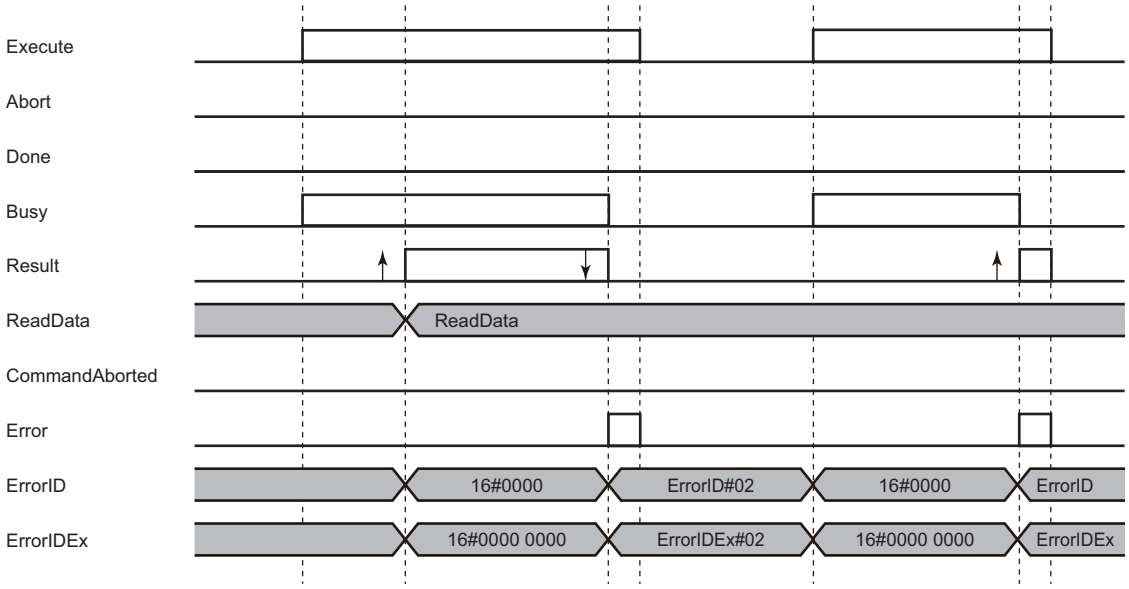


- To end repeated communications with an RF Tag, you must set *Abort* to TRUE. If *Abort* changes to TRUE, *Busy* (Executing) changes to FALSE and *CommandAborted* changes to TRUE.
- *Result* changes to FALSE at the same time when *CommandAborted* changes to FALSE.



Timing Chart for Error End

- If a specific error occurs while the execution of this function block is in progress, *Error* changes to TRUE and *Busy* (Executing) changes to FALSE, and the repetitive process is aborted.
- *Result* changes to FALSE at the same time when *Error* changes to FALSE.
- You can find out the cause of the error by accessing the values output to ErrorID (Error Code) and ErrorIDEx (Expansion Error Code).
- If *Execute* remains TRUE even after the execution of this function block is complete, the output value of *Error* is retained.
- The output values of *ErrorID* (Error Code) and *ErrorIDEx* (Expansion Error Code) are retained until this function block is executed again.
- If *Execute* changes to FALSE before the execution of this function block is complete, *Error* changes to TRUE only for one task period after the execution of the function block is complete.



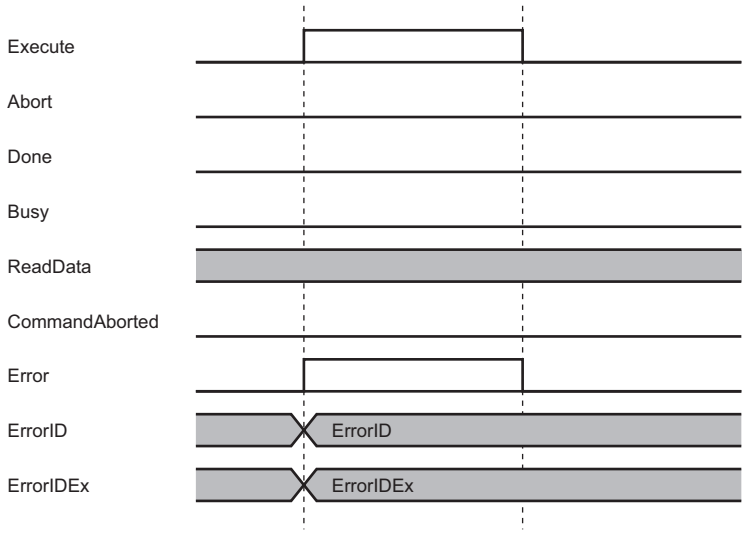
Note The specific errors due to which repeated communications are stopped are as described below.

Expansion error code	Expansion error code	Status
16#3D13	16#6AA00000	Antenna Configuration Error
	16#6AA10000	Amplifier Power Supply Error
	16#6AA20000	Amplifier Disconnection Detection
	16#6A720000	RF Tag Missing Error

Common Behavior

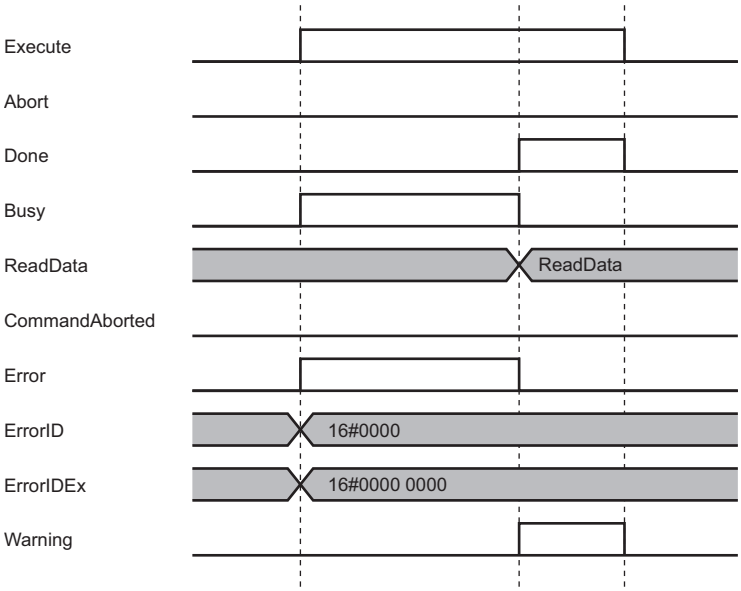
Timing Chart when an Error is Detected during Startup

- If an error is detected when this function block is started, *Error* changes to TRUE. *Busy* (Executing) remains FALSE.
- You can find out the cause of the error by accessing the values output to ErrorID (Error Code) and ErrorIDEx (Expansion Error Code).
- If *Execute* remains TRUE even after the execution of this function block is complete, the output value of *Error* is retained.
- The output values of *ErrorID* (Error Code) and *ErrorIDEx* (Expansion Error Code) are retained until this function block is executed again.

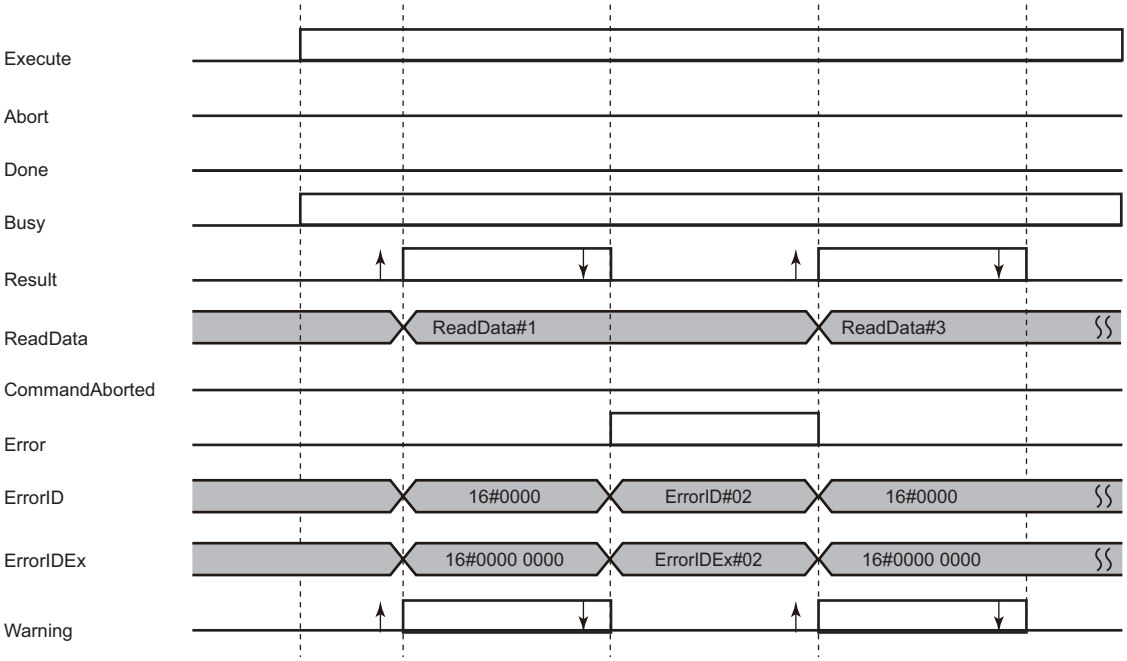


● Timing Chart for a Normal End when 1-bit Error Correction is Performed (Read Type: With Error Correction)

- When a 1-bit error is detected and corrected in the data read from the memory of the RF Tag, *Done* becomes TRUE, and at the same time, *Warning* becomes TRUE.
- If *Execute* remains TRUE even after the execution of this function block is complete, the output value of *Warning* is retained.



- When a 1-bit error is detected and corrected in the data read from the memory of the RF Tag during repeated communications, *Warning* also varies simultaneously when *Result* performs toggle operation.
- If error correction is performed next, *Warning* remains TRUE and does not change.



Precautions for Correct Use

- If the RFID Unit is in the Busy state due to the execution of a command already in progress, *Error* is output as TRUE indicating an error end, and the functions of this function block are not executed.
- When using the RFID Unit (NX-V680C2), integrate the device variables used as the input/output variables for this function block in either channel Ch1 or Ch2. If the device variables of different channels are set together, the operation will not be performed properly.
- Before executing this function block, carefully read the manual of the NX-V680C□ to use and ensure the safety for use.
- If the operation parameter *Data storage order* of the RFID Unit is *Descending*, do not specify the data size of the odd bytes. If you do so, you will not be able to read the correct data.
- Do not add the I/O entry *Input Data 2 to 8* to a channel used in the RFID Unit. If the total size of the input data increases, correct operation will not be performed.

Troubleshooting

The list of error codes output when this function block ends in an error is shown below.

Expansion error code	Expansion error code	Status	Description	Corrective action
16#0000	16#00000000	Normal End	---	---
16#3D13	16#00000001	Invalid RF Communications Option	The value of RFCommunicationsOption is outside the valid range.	Set the correct value in RFCommunicationsOption.
	16#00000002	Invalid Read Type	The value of ReadKind is outside the valid range.	Set the correct value in ReadKind.
	16#00000003	Invalid Memory Address	The value of MemoryAddr is outside the valid range.	Set the correct value in MemoryAddr.
	16#00000004	Invalid Data Size	The value of DataSize is outside the valid range.	Set the correct value in DataSize.
	16#00000005	Invalid Data Array	The array length of ReadData is below DataSize.	Set the array length of ReadData to DataSize or above.
	16#00000006	Execution Disabled	This function block cannot be started because the command is not ready for execution in the RFID Unit.	Check the status of the RFID Unit to confirm that the command is ready for execution, and then execute the function block again.
	16#6AA00000	Antenna Configuration Error	An unsupported antenna is connected. * Combination of the NX-V680C2 Unit and the V680-H01-V2 Unit	Connect an antenna other than V680-H01-V2 with NX-V680C2. Alternatively, use NX-V680C1.
	16#6AA10000	Amplifier Power Supply Error	No power is supplied to drive the amplifier.	Check the input of the I/O power supply.
	16#6AA20000	Amplifier Disconnection Detection	An amplifier disconnection is detected. The amplifier could not be recognized.	Connect the amplifier. Alternatively, replace the amplifier.
	16#6A720000	RF Tag Missing Error	There is no RF Tag in the communications range.	Adjust the equipment so that the RF Tag enters inside the communications range.
	16#6A700000	RF Tag Communications Failure	An error occurred during communications with the RF Tag, preventing a normal end.	Change the movement speed of the RF Tag and the distance between RF Tags so that they are within the specified range. Also, implement noise countermeasures if there is excessive ambient noise.
	16#6A760000	RF Tag Data Error	An RF Tag data error has been detected.	Check the data within the data check target range, and make sure no unexpected data has been written.
	16#6A7A0000	RF Tag Address Error	The address of the RF Tag is incorrect.	Check the memory capacity of the RF Tag being used, and correct the specified address so that it is within the range of the memory capacity.

Expansion error code	Expansion error code	Status	Description	Corrective action
16#3D13	16#6A790000	RF Tag Response Error	The RF Tag returned an error response, preventing a normal end.	Implement noise countermeasures if there is excessive ambient noise. [1]
	16#6A7F0000	RF Tag Customer Code Error	Communications were performed with an RF Tag that cannot be used.	Change the RF Tag.
	16#6AC00000	Undefined Command	It cannot be executed because it is an undefined command.	Set the correct variables in DeviceVariables.
	16#6AC10000	Invalid Command Parameter	The command cannot be executed because the command parameter is erroneous.	Set the correct variables in DeviceVariables.
	16#6AC20000	Command Execution Failure	The command cannot be executed because the command execution conditions have not been established.	Correct to a RF communications option that can be executed for the command.

*1. Change the RF Tag if the problem is not solved with noise countermeasures.

Sample programming

Program Description

Read the 16-byte data from memory address 8 of the RF Tag.

Preconditions

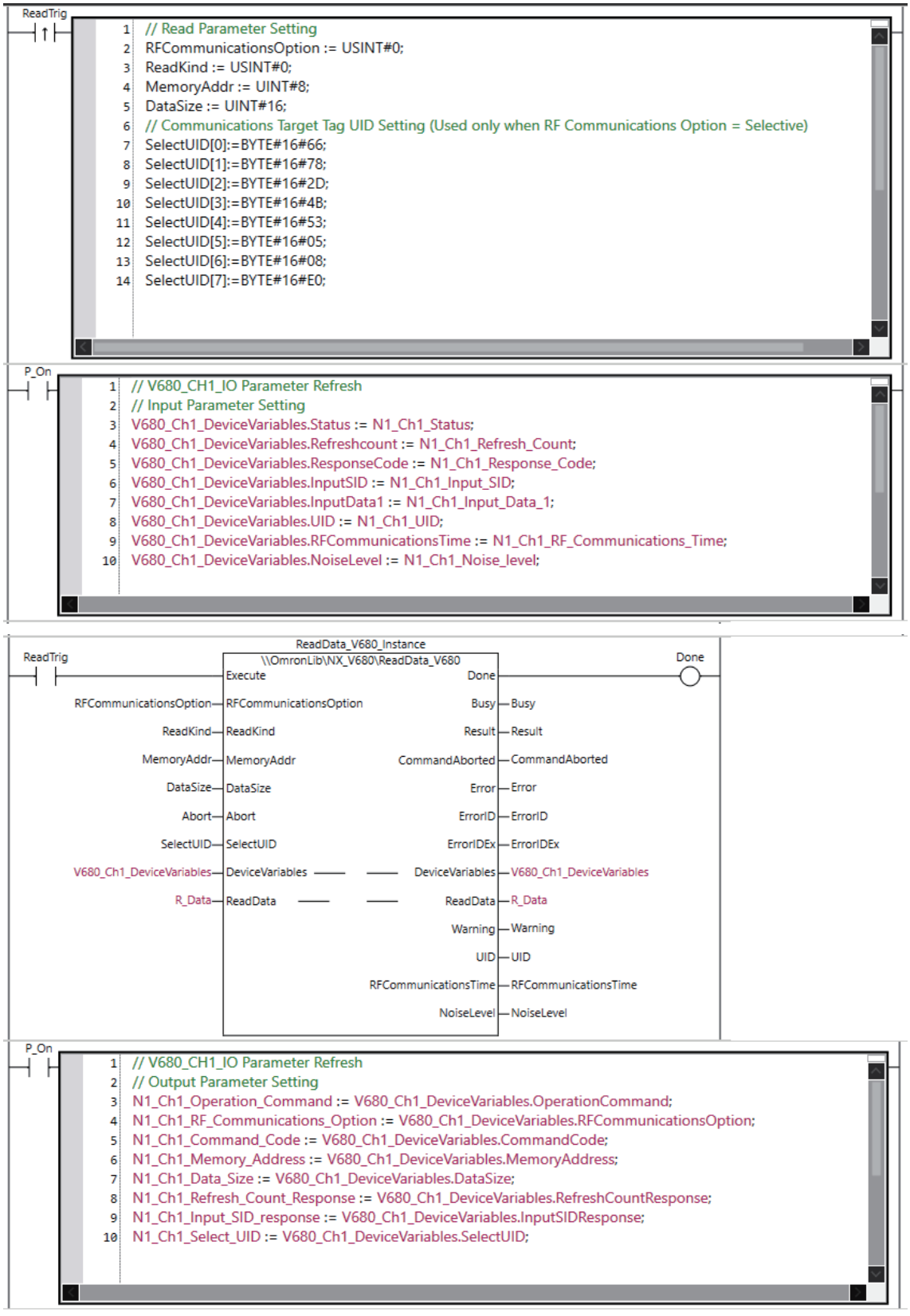
Create device variables for the RFID Unit to be operated, and use external references to the variables in the user program. Refer to the *Sysmac Studio Version 1 Operation Manual* (W504) for details on how to create device variables.

Main Variables

Variable or member	Name	Data Type	Default value	Description
ReadData_V680_Instance	Read Data V680 FB	OmronLib\NX_V680\ReadData_V680	---	Function block for implementing read for the NX-V680
V680_Ch1_DeviceVariables	Ch1 Device Variable	OmronLib\NX_V680\sV680DeviceVariables	---	Structure of Ch1 Device Variable
R_Data	Buffer for Read Data	ARRAY[0..8191] OF BYTE	16#00	Output variable for *ReadData_V680_Instance*
ReadTrig	Read Execution Flag	BOOL	FALSE	The variable is TRUE when read is executed
RFCommunicationsOption	RF Communications Option	USINT	0	Set the RF communications option
ReadKind	Read Type	USINT	0	Set the read type
MemoryAddr	Memory Address	UINT	0	Set the communications start address
DataSize	Data Size	UINT	0	Set the communications data size

Variable or member	Name	Data Type	Default value	Description
Abort	Execution Stop Flag	BOOL	FALSE	The variable is TRUE when read execution has stopped
SelectUID	Select UID	ARRAY[0..7] OF BYTE	16#00	Set the UID of the communications target tag
Command-Stage	Command Busy	UINT	0	Execution status of the ST program
Execute	Execution Flag	BOOL	FALSE	Execution status flag in the ST program
Done	Execution Completion Flag	BOOL	---	Output variable for *ReadData_V680_Instance*
Busy	Enable	BOOL	---	Output variable for *ReadData_V680_Instance*
Result	Result Flag	BOOL	---	Output variable for *ReadData_V680_Instance*
Command-Aborted	Abort Flag	BOOL	---	Output variable for *ReadData_V680_Instance*
Error	Error Flag	BOOL	---	Output variable for *ReadData_V680_Instance*
Warning	Warning Flag	BOOL	---	Output variable for *ReadData_V680_Instance*
ErroriD	Error code	WORD	---	Output variable for *ReadData_V680_Instance*
ErrorIDEx	Expansion error code	DWORD	---	Output variable for *ReadData_V680_Instance*
RFCommunicationsTime	RF Communications Time	UINT	---	Output variable for *ReadData_V680_Instance*
NoiseLevel	Noise Level	UINT	---	Output variable for *ReadData_V680_Instance*
UID	UID	ARRAY[0..7] OF BYTE	---	Output variable for *ReadData_V680_Instance*

Ladder Diagram



● Code of Inline ST (Zeroth Line of Ladder Diagram):

```
// Read Parameter Setting
RFCommunicationsOption := USINT#0;
ReadKind := USINT#0;
MemoryAddr := UINT#8;
DataSize := UINT#16;
// Communications Target Tag UID Setting (Used only when RF Communications Option =
Selective)
SelectUID[0]:=BYTE#16#66;
SelectUID[1]:=BYTE#16#78;
SelectUID[2]:=BYTE#16#2D;
SelectUID[3]:=BYTE#16#4B;
SelectUID[4]:=BYTE#16#53;
SelectUID[5]:=BYTE#16#05;
SelectUID[6]:=BYTE#16#08;
SelectUID[7]:=BYTE#16#E0;
```

● Code of Inline ST (First Line of Ladder Diagram):

```
// V680_CH1_IO Parameter Refresh
// Input Parameter Setting
V680_Ch1_DeviceVariables.Status := N1_Ch1_Status;
V680_Ch1_DeviceVariables.Refreshcount := N1_Ch1_Refresh_Count;
V680_Ch1_DeviceVariables.ResponseCode := N1_Ch1_Response_Code;
V680_Ch1_DeviceVariables.InputSID := N1_Ch1_Input_SID;
V680_Ch1_DeviceVariables.InputData1 := N1_Ch1_Input_Data_1;
V680_Ch1_DeviceVariables.UID := N1_Ch1_UID;
V680_Ch1_DeviceVariables.RFCommunicationsTime := N1_Ch1_RF_Communications_Time;
V680_Ch1_DeviceVariables.NoiseLevel := N1_Ch1_Noise_level;
```

● Code of Inline ST (Third Line of Ladder Diagram):

```
// V680_CH1_IO Parameter Refresh
// Output Parameter Setting
N1_Ch1_Operation_Command := V680_Ch1_DeviceVariables.OperationCommand;
N1_Ch1_RF_Communications_Option := V680_Ch1_DeviceVariables.RFCommunicationsOption;
N1_Ch1_Command_Code := V680_Ch1_DeviceVariables.CommandCode;
N1_Ch1_Memory_Address := V680_Ch1_DeviceVariables.MemoryAddress;
N1_Ch1_Data_Size := V680_Ch1_DeviceVariables.DataSize;
N1_Ch1_Refresh_Count_Response := V680_Ch1_DeviceVariables.RefreshCountResponse;
N1_Ch1_Input_SID_response := V680_Ch1_DeviceVariables.InputSIDResponse;
N1_Ch1_Select_UID := V680_Ch1_DeviceVariables.SelectUID;
```

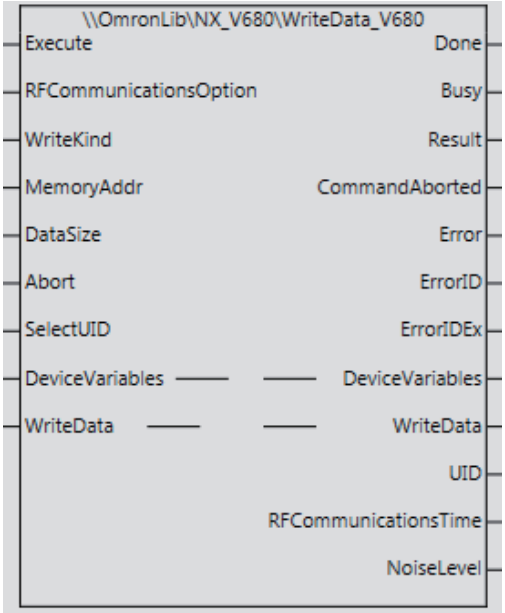
ST

```
// V680_CH1_IO Parameter Refresh
// Input Parameter Setting
V680_Ch1_DeviceVariables.Status := N1_Ch1_Status;
V680_Ch1_DeviceVariables.Refreshcount := N1_Ch1_Refresh_Count;
V680_Ch1_DeviceVariables.ResponseCode := N1_Ch1_Response_Code;
V680_Ch1_DeviceVariables.InputSID := N1_Ch1_Input_SID;
V680_Ch1_DeviceVariables.InputData1 := N1_Ch1_Input_Data_1;
V680_Ch1_DeviceVariables.UID := N1_Ch1_UID;
V680_Ch1_DeviceVariables.RFCommunicationTime := N1_Ch1_RF_Communications_Time;
V680_Ch1_DeviceVariables.NoiseLevel := N1_Ch1_Noise_level;


CASE CommandStage OF
    (*Idle*)
    0 :
    // If ReadTrig changes to TRUE, the command is executed
    IF ( ReadTrig = TRUE ) THEN
        // Read Parameter Setting
        RFCommunicationsOption := USINT#0;
        ReadKind := USINT#0;
        MemoryAddr := UINT#8;
        DataSize := UINT#16;
        // Communications Target Tag UID Setting (Used only when RF Communications
Option = Selective)
        SelectUID[0]:=BYTE#16#66;
        SelectUID[1]:=BYTE#16#78;
        SelectUID[2]:=BYTE#16#2D;
        SelectUID[3]:=BYTE#16#4B;
        SelectUID[4]:=BYTE#16#53;
        SelectUID[5]:=BYTE#16#05;
        SelectUID[6]:=BYTE#16#08;
        SelectUID[7]:=BYTE#16#E0;
        // Read Execution Flag Setting
        Execute := TRUE;
        // Transit to Read Execution Status
        CommandStage := UINT#1;
    END_IF;


    (*Acquisition of read execution result*)
    1 :
    IF (Busy = FALSE) THEN
        IF (Done = TRUE) THEN
        // Normal End
           (* --------------------↓Specify normal processing↓--------------------- *)

            (* ↑-------------------------------------------------------------↑ *)
            // Transit to Unit Operation Stop Wait Status
            CommandStage := UINT#2;

        ELSIF (Error = TRUE)THEN
            (* --------------------↓Specify error processing↓--------------------- *)

            (* ↑-------------------------------------------------------------↑ *)
            // Transit to Unit Operation Stop Wait Status
            CommandStage := UINT#2;
        END_IF;
    END_IF;
```

```
    (*Unit operation stop wait*)
    2 :
    // If ReadTrig changes to FALSE, transition to idle state occurs
    IF ( ReadTrig = FALSE ) THEN
        CommandStage := UINT#0;
        // Read Execution End Flag Setting
        Execute := FALSE;
    END_IF;

END_CASE;

// Read Data FB
ReadData_V680_instance
(
    Execute :=Execute,
    RFCommunicationsOption := RFCommunicationsOption,
    ReadKind := ReadKind,
    MemoryAddr := MemoryAddr,
    DataSize := DataSize,
    Abort := Abort,
    SelectUID := SelectUID,
    Done => Done,
    Busy => Busy,
    Result => Result,
    CommandAborted => CommandAborted,
    Error => Error,
    ErrorID => ErrorID,
    ErrorIDEx => ErrorIDEx,
    Warning => Warning,
    UID => UID,
    RFCommunicationsTime => RFCommunicationsTime,
    NoiseLevel => NoiseLevel,
    DeviceVariables := V680_Ch1_DeviceVariables,
    ReadData := R_Data
);

// V680_CH1_IO Parameter Refresh
// Output Parameter Setting
N1_Ch1_Operation_Command := V680_Ch1_DeviceVariables.OperationCommand;
N1_Ch1_RF_Communications_Option := V680_Ch1_DeviceVariables.RFCommunicationsOption;
N1_Ch1_Command_Code := V680_Ch1_DeviceVariables.CommandCode;
N1_Ch1_Memory_Address := V680_Ch1_DeviceVariables.MemoryAddress;
N1_Ch1_Data_Size := V680_Ch1_DeviceVariables.DataSize;
N1_Ch1_Refresh_Count_Response := V680_Ch1_DeviceVariables.RefreshCountResponse;
N1_Ch1_Input_SID_response := V680_Ch1_DeviceVariables.InputSIDResponse;
N1_Ch1_Select_UID := V680_Ch1_DeviceVariables.SelectUID;
```

WriteData_V680

This command writes data to the memory of an RF Tag in the communications range of the antenna mounted on the RFID Unit (NX-V680).

Program part names	Series	FB/ FUN	Graphic expression	STexpression
Write-Data_V680	RF Tag data write for V680	FB		WriteData_V680_instance(Execute := <Parameter>, RFCommunicationsOption := <Parameter>, WriteKind := <Parameter>, MemoryAddr := <Parameter>, DataSize := <Parameter>, Abort := <Parameter>, SelectUID := <Parameter>, Done => <Parameter>, Busy => <Parameter>, Result => <Parameter>, CommandAborted => <Parameter>, Error => <Parameter>, ErrorID => <Parameter>, ErrorIDEx => <Parameter>, UID => <Parameter>, RFCommunicationsTime => <Parameter>, NoiseLevel => <Parameter>, WriteData := <Parameter>, DeviceVariables := <Parameter>,);

Function Block and Function Information

Item	Description
Library file name	OmronLib_NX_V680_Vx_x.slr (x indicates the version)
Namespace	OmronLib\NX_V680
Function block and function number	00214
Source code	Do not publish.

Input variable

Variables	Name	Data Type	Description	Valid range	Unit	Initial value
Execute	Execute	BOOL	The processing is started when the variable changes to TRUE. TRUE: Execute FALSE: Do not execute	TRUE or FALSE	---	FALSE
RFCommunicationsOption	RF Communications Option	USINT	Specify the operation sequence during communications. 0: Trigger 1: Auto 2: Repeat 3: FIFO Trigger 4: FIFO Repeat 5: Multi Trigger 6: Multi Repeat 7: Selective*1	0 to 7	---	0
WriteKind	Write Type	USINT	Specify the type of the write command.*2 0: Normal 1: With error detection 2: With error correction	0 to 2	---	0
MemoryAddr	Memory Address	UINT	Specify the start address of the memory to which data is written from the RF Tag.*3	0 to 65535	Byte	0
DataSize	Data Size	UINT	Enter the size of the data to be written from the RF Tag.*4	1 to 8192	Byte	0
Abort	Abort	BOOL	The processing is aborted when the variable changes to TRUE.	TRUE or FALSE	---	FALSE
SelectUID	Select UID	ARRAY[0..7]OF BYTE	Specify the UID of the communications target RF Tag when the RF communications option is *Selective*.	Depends on data type.	---	16#00

*1. When using *Selective*, do not add the I/O entry *Select UID* to a channel used in the RFID Unit.

*2. Refer to the List of Commands in the *NX-series RFID Units User's Manual* (Z401) for details on the types of write commands and differences in their operation.

*3. Enter a value in consideration of the memory map of the RF Tag actually used.

*4. Enter a value in consideration of the memory capacity of the RF Tag actually used. When 0 is specified, no operation is performed and the processing ends.

Output Variables

Variables	Name	Data Type	Description	Valid range	Unit	Initial value
Done	Done	BOOL	The variable changes to TRUE when the processing is completed.	TRUE or FALSE	---	FALSE
Busy	Busy	BOOL	The variable changes to TRUE when the processing is acknowledged.	TRUE or FALSE	---	FALSE
Result	Result	BOOL	The value of the variable alternately changes between TRUE and FALSE each time the result of communications with the RF Tag is output.	TRUE or FALSE	---	FALSE
Command Aborted	Instruction Aborted	BOOL	The variable changes to TRUE when the processing is aborted.	TRUE or FALSE	---	FALSE
Error	Error	BOOL	This variable is TRUE while there is an error. TRUE: Error end FALSE: Normal end or Executing	TRUE or FALSE	---	FALSE
ErroriD	Error code	WORD	An error code is output. This is the error ID for an error end. The value is 16#0 for a normal end.	*1	---	0
ErrorIDEx	Expansion error code	DWORD	An expansion error code is output. This is the error ID for an error end. The value is 16#0 for a normal end.	*1	---	0
UID*2	UID	ARRAY[0..7]OF BYTE	The UID of the RF Tag with which communications are performed is output.	Depends on data type.	---	16#00
RFCommunicationsTime*3	RF Communications Time	UINT	The measured RF communications time is output.	0 to 65535	ms	0
NoiseLevel*4	Noise Level	UINT	The measured noise level is output.	0 to 99	---	0

*1. For details, refer to *Troubleshooting* on page 72.

*2. During use, add the I/O entry *UID* to the channel used in the RFID Unit.

*3. During use, add the I/O entry *RF Communications Time* to the channel used in the RFID Unit.

*4. During use, add the I/O entry *Noise Level* to the channel used in the RFID Unit.

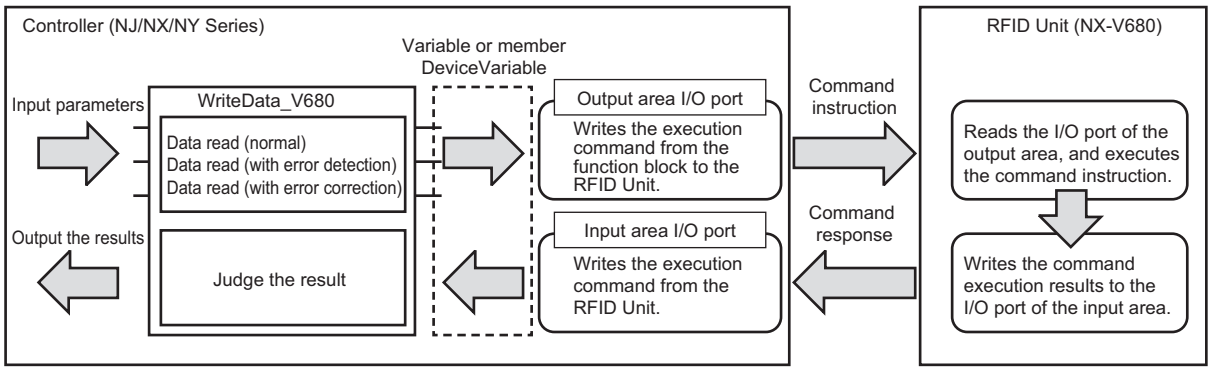
In-Out Variables

Variables	Name	Data Type	Description	Valid range	Unit	Initial value
DeviceVariables	device variable	Omron-Lib\RFID\sNX-V680DeviceVariables	Device variables of the RFID Unit (I/O port of the input area / output area)	Depends on data type.	---	---
WriteData	Write Data	ARRAY[Variable length] OF BYTE	Data array to be written to the RF Tag	Depends on data type.	---	-

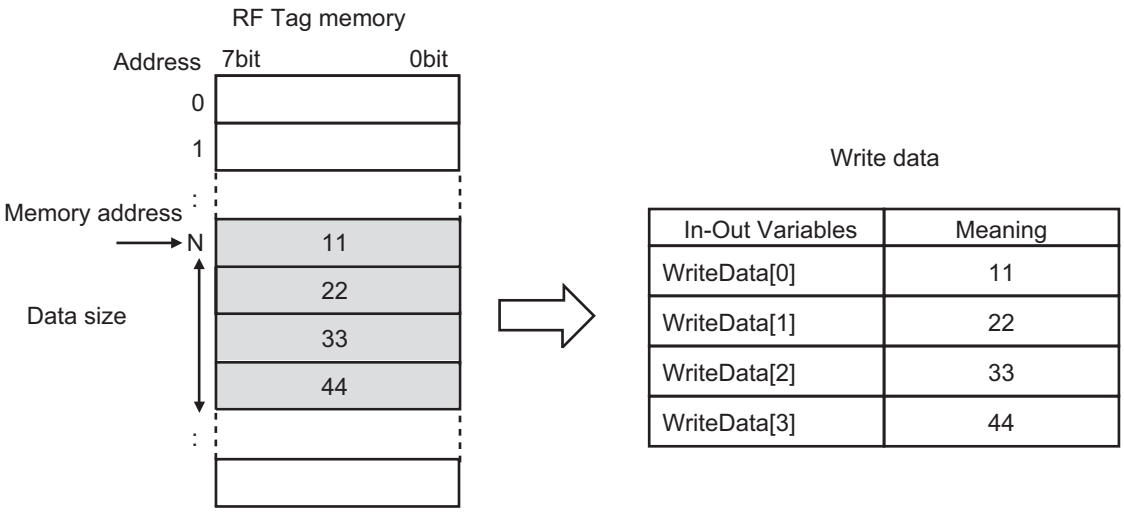
Note A BYTE array of any length can be specified. However, the array length must be equal to or more than the DataSize. Both 0 and n can be set for the array start number.

Operation

This function block controls the command for writing the memory of the RF Tag for an RFID Unit (NX-V680) that is exchanging data with an NJ/NX/NY-series Controller. Therefore, the device variables (the I/O port of the output area and the I/O port of the input area) of the RFID Unit to be controlled must be set in the input/output variable *DeviceVariables* of the function block.



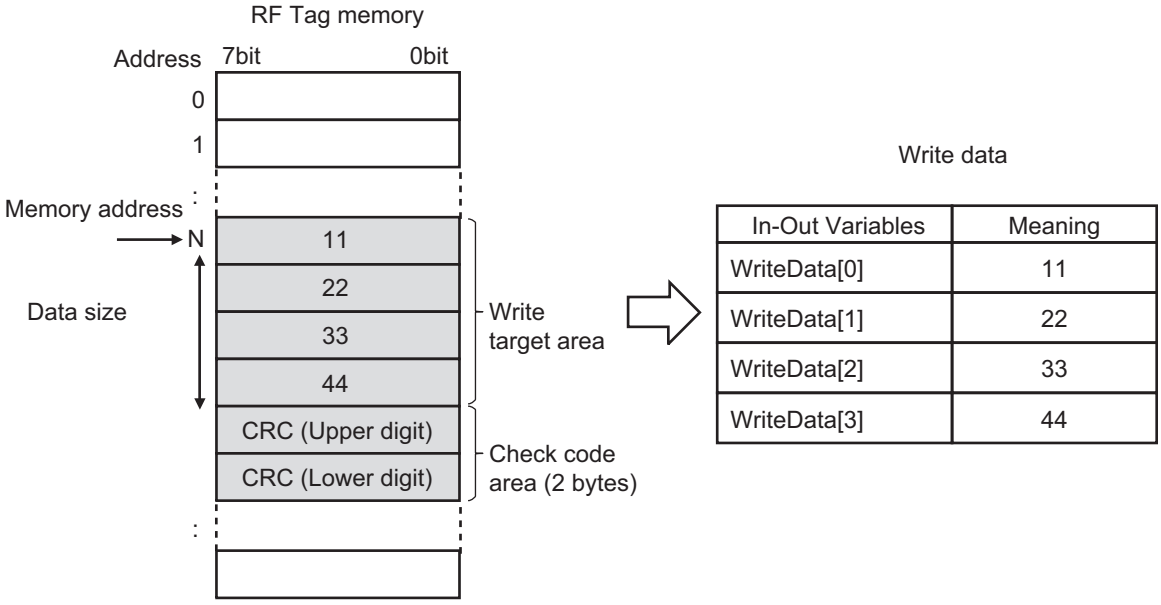
- When *Execute* changes to TRUE, the data of the byte array specified in *WriteData* is written to the memory of the RF Tag.



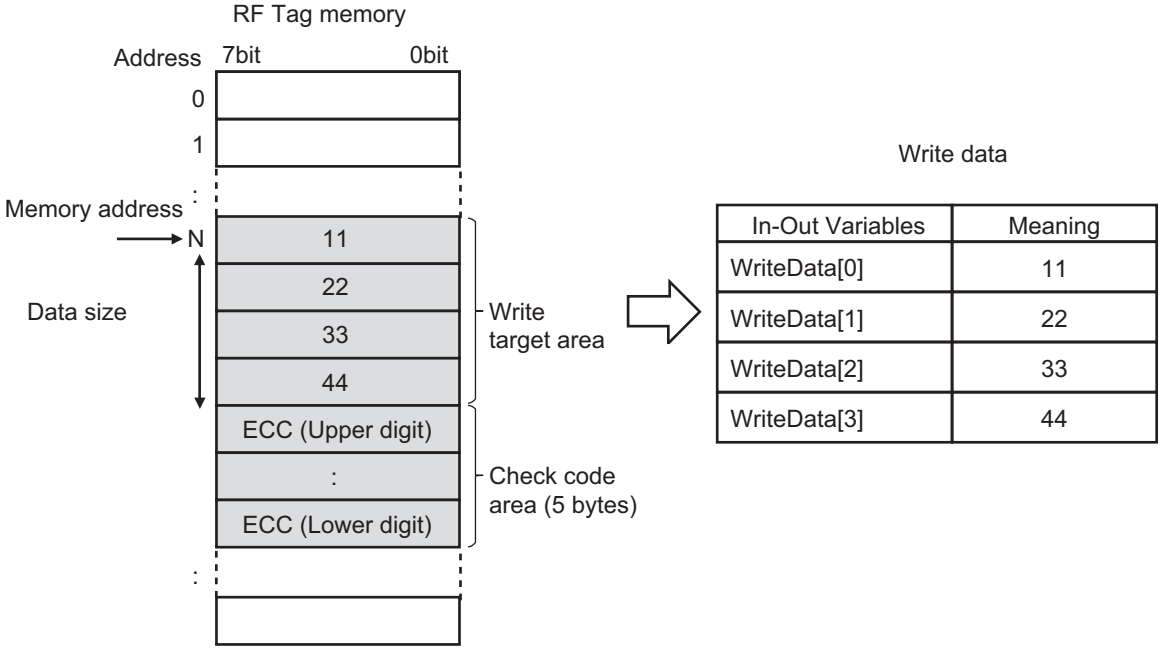
- The memory capacity written to the RF Tag in a single execution varies depending on the specified *WriteKind*. The values that can be specified in *MemoryAddr* and *DataSize* are described below.

WriteKind	MemoryAddr	DataSize
0: Normal	0 to 65535	1 to 8192
1: With error detection	0 to 65533	1 to 8190
2: With error correction	0 to 65530	1 to 510

- If *WriteKind* is executed as *With error detection*, data is written to the RF Tag target area, and the check code for error detection is written to the last two bytes of the area.



- If *WriteKind* is executed as *With error correction*, data is written to the RF Tag target area, and the check code for error correction is written to the last five bytes of the area.



Timing Chart

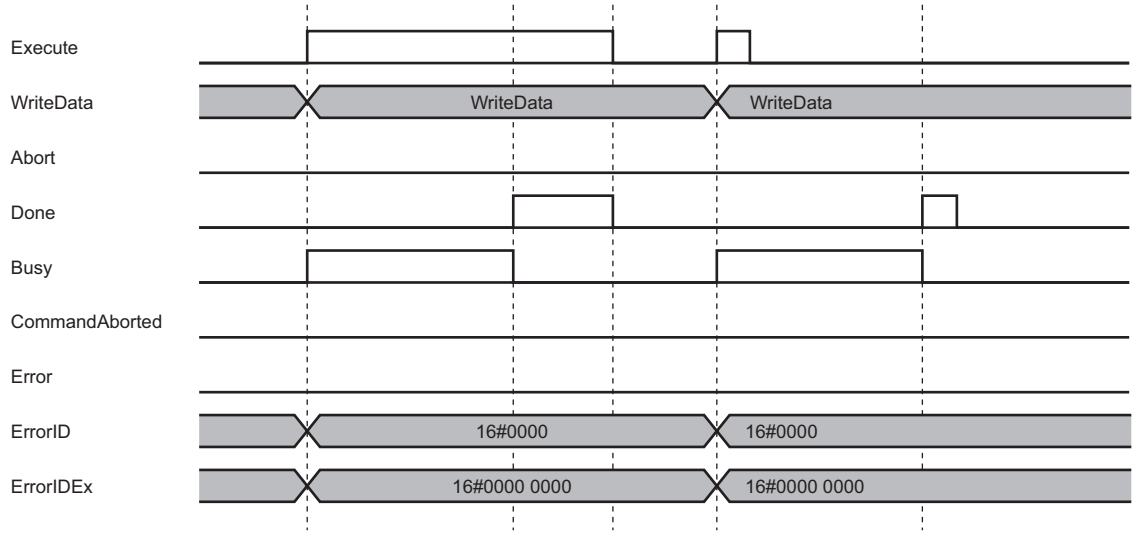
The timing charts are shown below.

Communicating with One RF Tag (RF Communications Option: Trigger, Auto, FIFO trigger or Selective)

If this function block is started by specifying Trigger, Auto, FIFO trigger, or Selective in *RFCommunicationsOption*, the result of communications with one RF Tag present in the communications range of the antenna is output.

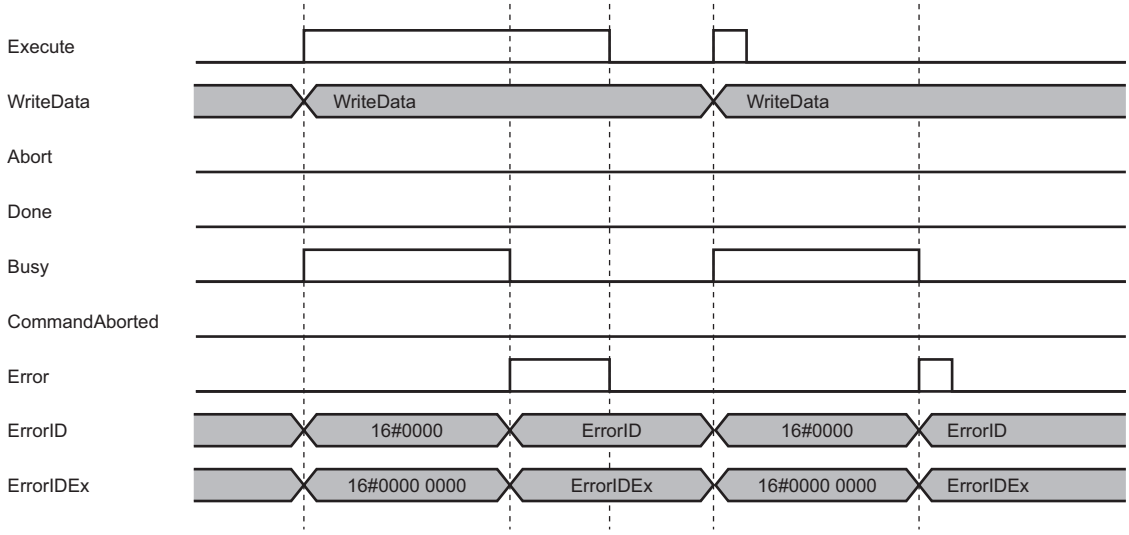
● Timing Chart for Normal End

- Busy (Executing) changes to TRUE when Execute (Execute) changes to TRUE.
- If writing of data to the memory of the RF Tag ends normally, *Done* changes to TRUE, and *Busy* (Executing) changes to FALSE. At the same time, a data string is output in UID, and a value is output in *RFCommunicationsTime* and *NoiseLevel*.
- If *Execute* remains TRUE even after the execution of this function block is complete, the output value of *Done* is retained.
- If Execute changes to FALSE before the execution of this function block is complete, *Done* changes to TRUE only for one task period after the execution of the function block is complete.



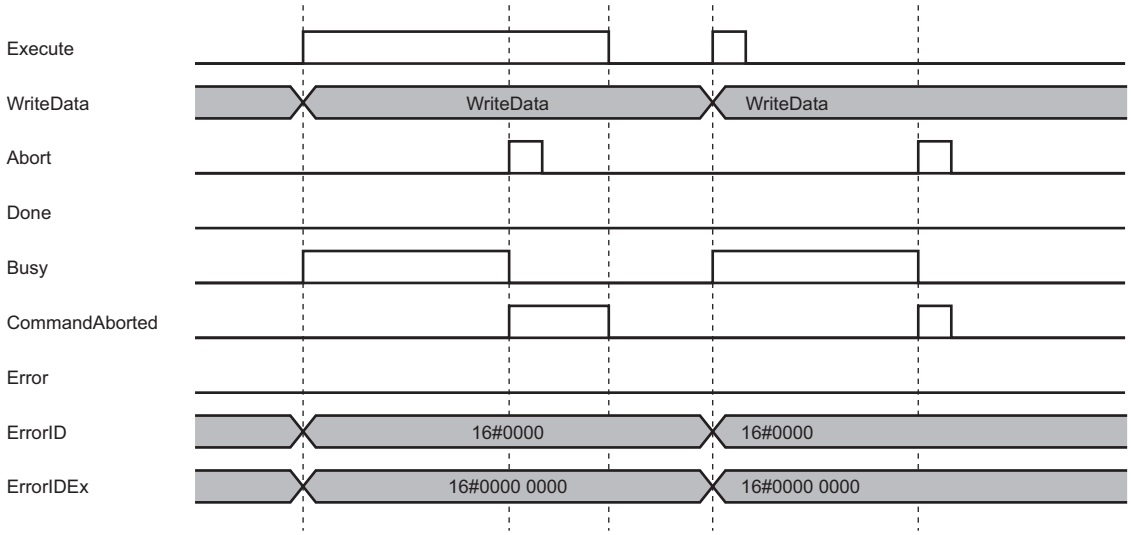
Timing Chart for Error End

- If an error occurs while the execution of this function block is in progress, *Error* changes to TRUE and *Busy* (Executing) changes to FALSE.
- You can find out the cause of the error by accessing the values output to ErrorID (Error Code) and ErrorIDEx (Expansion Error Code).
- If *Execute* remains TRUE even after the execution of this function block is complete, the output value of *Error* is retained.
- The output values of *ErrorID* (Error Code) and *ErrorIDEx* (Expansion Error Code) are retained until this function block is executed again.
- If *Execute* changes to FALSE before the execution of this function block is complete, *Error* changes to TRUE only for one task period after the execution of the function block is complete.



When the Function Block Is Aborted

To abort communications with the RF Tag while the execution of this function block is in progress, set *Abort* to TRUE. If *Abort* changes to TRUE, *Busy* (Executing) changes to FALSE and *Command-Aborted* changes to TRUE.

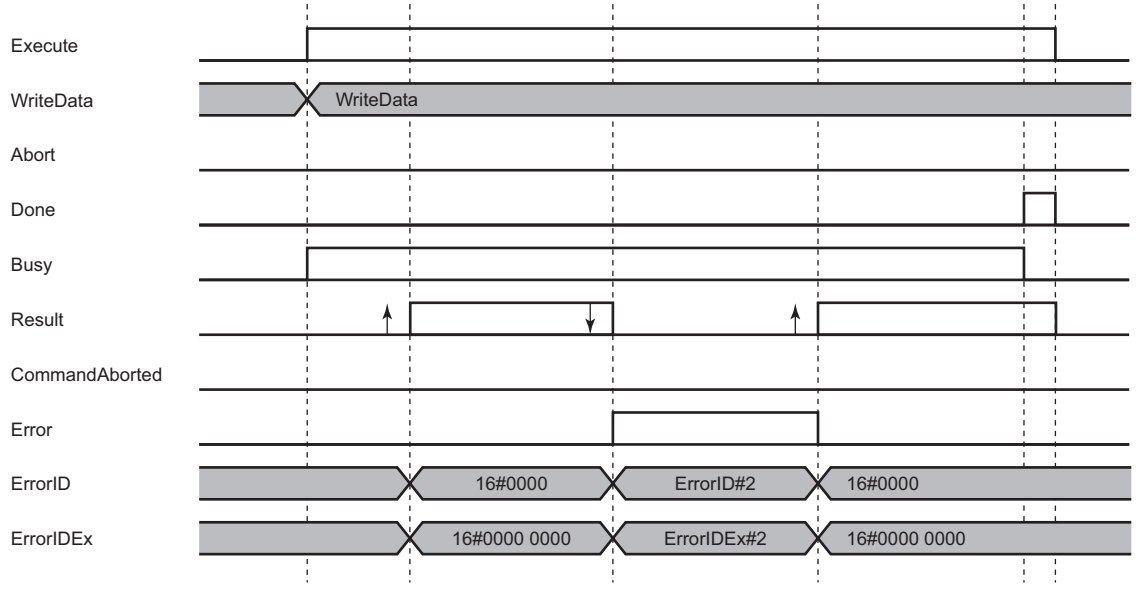


Communicating with Multiple RF Tags (RF Communications Option: Multi Trigger)

If this function block is started by specifying Multi trigger in *RFCommunicationsOption*, the result of communications with the multiple RF Tags present in the communications range of the antenna is output.

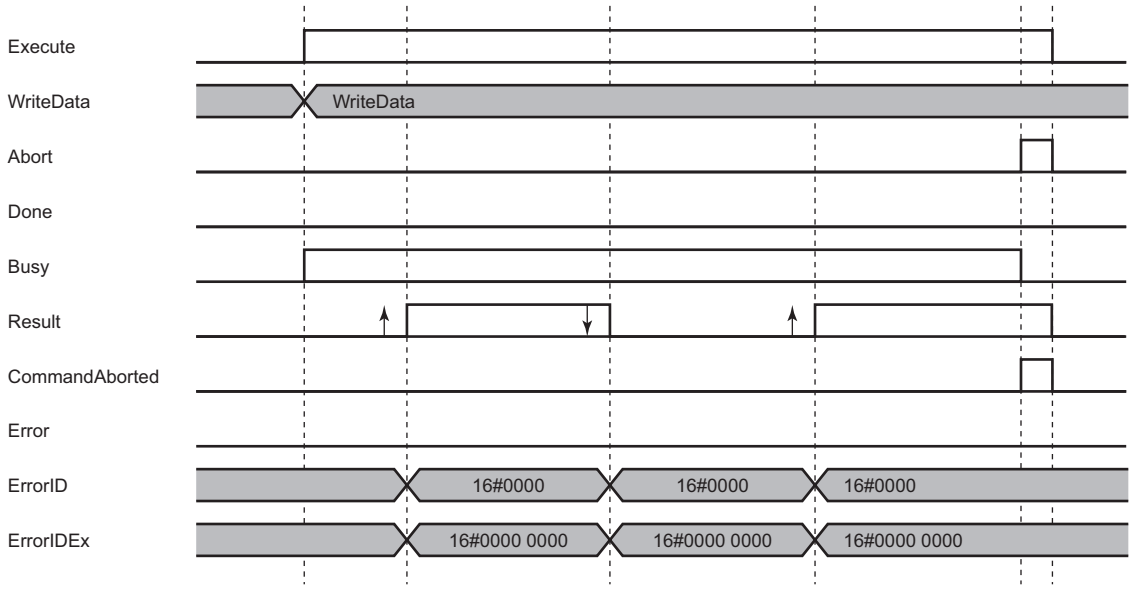
● Timing Chart for Communications with Multiple RF Tags

- Busy (Executing) changes to TRUE when Execute (Execute) changes to TRUE.
- Each time communications are performed with multiple RF Tags, the *Result* toggles between FALSE -> TRUE -> FALSE -> TRUE.
- When communications with all RF Tags in the communications range are complete, *Done* changes to TRUE and *Busy* (Executing) changes to FALSE.
- If writing of data to the memory of the RF Tag ends normally, a data string is output in UID, and a value is output in *RFCommunicationsTime* and *NoiseLevel* at the same time as a change in *Result*.
- If writing of data to the memory of the RF Tag ends in an error, *Error* changes to TRUE at the same time as a change in *Result*. You can find out the cause of the error by accessing the values output to ErrorID (Error Code) and ErrorIDEx (Expansion Error Code).
- *Result* changes to FALSE at the same time when *Done* changes to FALSE.



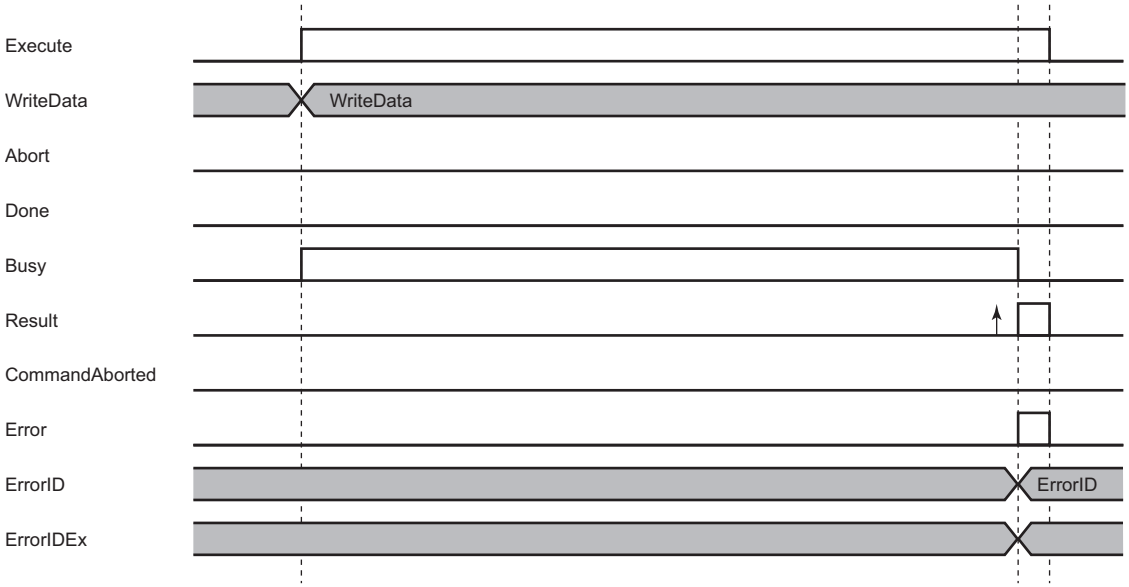
When the Function Block Is Aborted

- To abort communications with the RF Tag while the execution of this function block is in progress, set *Abort* to TRUE. If *Abort* changes to TRUE, *Busy* (Executing) changes to FALSE and *CommandAborted* changes to TRUE.
- *Result* changes to FALSE at the same time when *CommandAborted* changes to FALSE.



Timing Chart when the Tag is Missing

- If communications are not performed with the RF Tag even once while the execution of this function block is in progress, *Result* and *Error* change to TRUE, and *Busy* (Executing) changes to FALSE.
- You can find out the cause of the error by accessing the values output to ErrorID (Error Code) and ErrorIDEx (Expansion Error Code).
- If *Execute* remains TRUE even after the execution of this function block is complete, the output values of *Result* and *Error* are retained.
- The output values of *ErrorID* (Error Code) and *ErrorIDEx* (Expansion Error Code) are retained until this function block is executed again.
- If *Execute* changes to FALSE before the execution of this function block is complete, *Result* and *Error* change to TRUE only for one task period after the execution of the function block is complete.

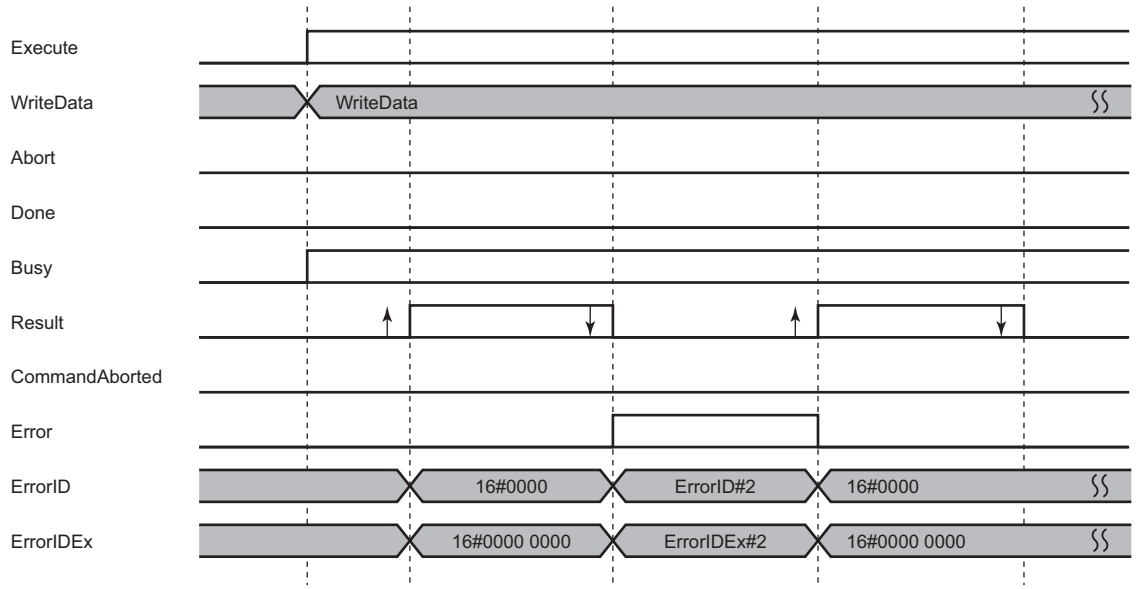


Communicating Repeatedly with an RF Tag (RF Communications Option: Repeat, FIFO Repeat or Multi Repeat)

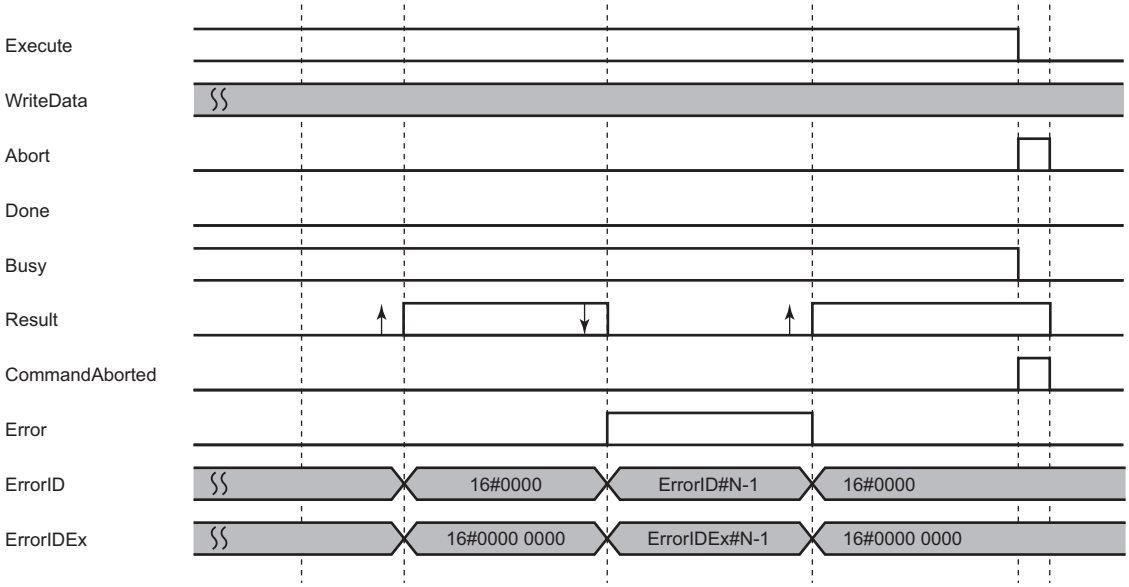
If this function block is started by specifying Repeat, FIFO repeat, or Multi-access repeat in RFCommunicationsOption (RF Communications Option), the result of repeated communications with an RF Tag that has moved into the communications range of the antenna is output.

● Timing Chart for Repeated Communications

- Busy (Executing) changes to TRUE when Execute (Execute) changes to TRUE.
- Each time communications are performed with multiple RF Tags, the *Result* toggles between FALSE -> TRUE -> FALSE -> TRUE.
- If writing of data to the memory of the RF Tag ends normally, a data string is output in *ReadData* and UID, and a value is output in *RFCommunicationsTime* and *NoiseLevel* at the same time as a change in *Result*.
- If writing of data to the memory of the RF Tag ends in an error, *Error* changes to TRUE at the same time as a change in *Result*. You can find out the cause of the error by accessing the values output to ErrorID (Error Code) and ErrorIDEx (Expansion Error Code).

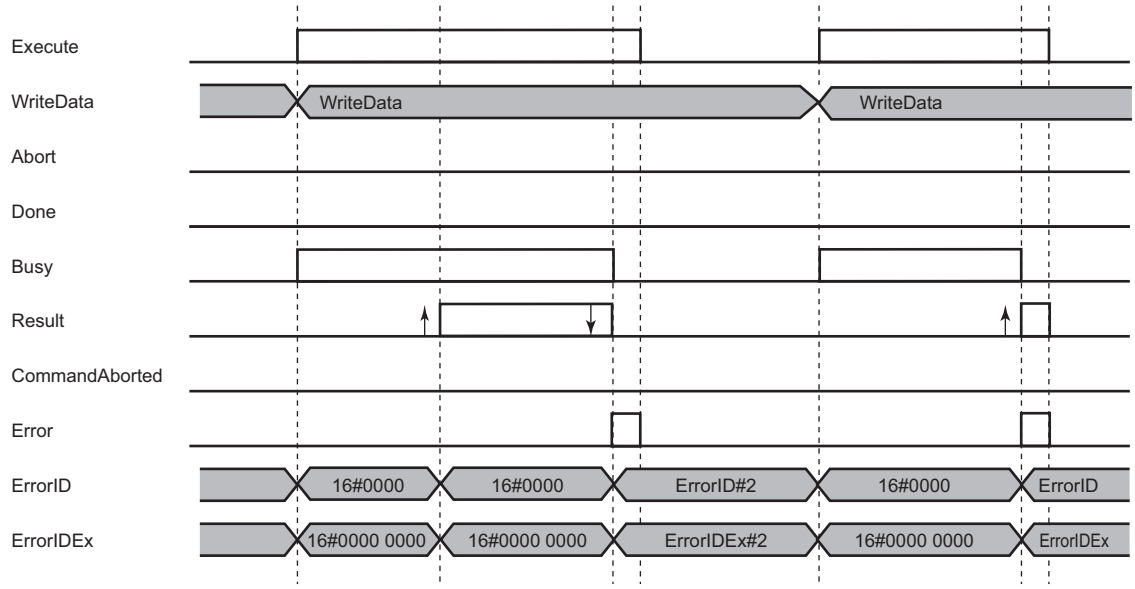


- To end repeated communications with an RF Tag, you must set *Abort* to TRUE. If *Abort* changes to TRUE, *Busy* (Executing) changes to FALSE and *CommandAborted* changes to TRUE.
- *Result* changes to FALSE at the same time when *CommandAborted* changes to FALSE.



Timing Chart for Error End

- If a specific error occurs while the execution of this function block is in progress, *Error* changes to TRUE and *Busy* (Executing) changes to FALSE, and the repetitive process is aborted.
- *Result* changes to FALSE at the same time when *Error* changes to FALSE.
- You can find out the cause of the error by accessing the values output to ErrorID (Error Code) and ErrorIDEx (Expansion Error Code).
- If *Execute* remains TRUE even after the execution of this function block is complete, the output value of *Error* is retained.
- The output values of *ErrorID* (Error Code) and *ErrorIDEx* (Expansion Error Code) are retained until this function block is executed again.
- If *Execute* changes to FALSE before the execution of this function block is complete, *Error* changes to TRUE only for one task period after the execution of the function block is complete.



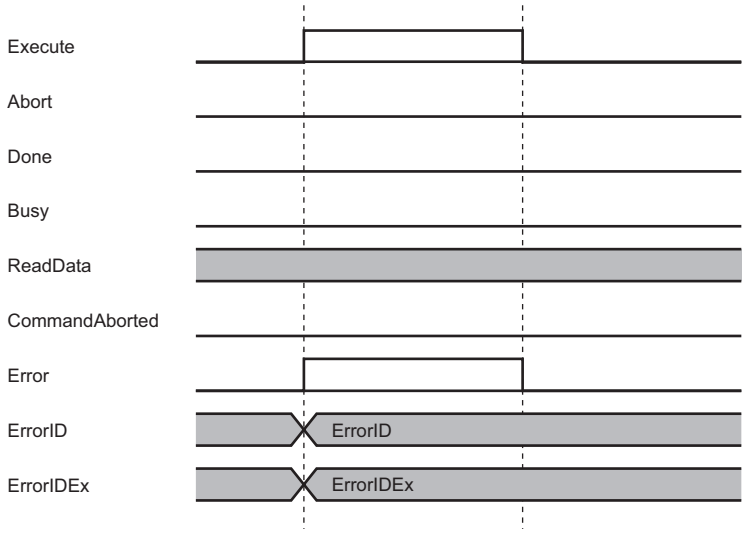
Note The specific errors due to which repeated communications are stopped are as described below.

Expansion error code	Expansion error code	Status
16#3D14	16#6AA00000	Antenna Configuration Error
	16#6AA10000	Amplifier Power Supply Error
	16#6AA20000	Amplifier Disconnection Detection
	16#6A720000	RF Tag Missing Error

Common Behavior

Timing Chart when an Error is Detected during Startup

- If an error is detected when this function block is started, *Error* changes to TRUE. *Busy* (Executing) remains FALSE.
- You can find out the cause of the error by accessing the values output to ErrorID (Error Code) and ErrorIDEx (Expansion Error Code).
- If *Execute* remains TRUE even after the execution of this function block is complete, the output value of *Error* is retained.
- The output values of *ErrorID* (Error Code) and *ErrorIDEx* (Expansion Error Code) are retained until this function block is executed again.



Precautions for Correct Use

- If the RFID Unit is in the Busy state due to the execution of a command already in progress, *Error* is output as TRUE indicating an error end, and the functions of this function block are not executed.
- When using the RFID Unit (NX-V680C2), integrate the device variables used as the input/output variables for this function block in either channel Ch1 or Ch2. If the device variables of different channels are set together, the operation will not be performed properly.
- Before executing this function block, carefully read the manual of the NX-V680C□ to use and ensure the safety for use.
- If the operation parameter *Data storage order* of the RFID Unit is *Descending*, do not specify the data size of the odd bytes. If you do so, the correct data will not be written to the RF Tag.
- Do not add the I/O entry *Output Data 2 to 8* to a channel used in the RFID Unit. If the total size of the output data increases, correct operation will not be performed.

Troubleshooting

The list of error codes output when this function block ends in an error is shown below.

Expansion error code	Expansion error code	Status	Description	Corrective action
16#0000	16#00000000	Normal End	---	---
16#3D14	16#00000001	Invalid RF Communications Option	The value of RFCommunicationsOption is outside the valid range.	Set the correct value in RFCommunicationsOption.
	16#00000002	Invalid Write Type	The value of WriteKind is outside the valid range.	Set the correct value in WriteKind.
	16#00000003	Invalid Memory Address	The value of MemoryAddr is outside the valid range.	Set the correct value in MemoryAddr.
	16#00000004	Invalid Data Size	The value of DataSize is outside the valid range.	Set the correct value in DataSize.
	16#00000005	Invalid Data Array	The array length of WriteData is below DataSize.	Set the array length of WriteData to DataSize or above.
	16#00000006	Execution Disabled	This function block cannot be started because the command is not ready for execution in the RFID Unit.	Check the status of the RFID Unit to confirm that the command is ready for execution, and then execute the function block again.
	16#6AA00000	Antenna Configuration Error	An unsupported antenna is connected. * Combination of the NX-V680C2 Unit and the V680-H01-V2 Unit	Connect an antenna other than V680-H01-V2 with NX-V680C2. Alternatively, use NX-V680C1.
	16#6AA10000	Amplifier Power Supply Error	No power is supplied to drive the amplifier.	Check the input of the I/O power supply.
	16#6AA20000	Amplifier Disconnection Detection	An amplifier disconnection is detected. The amplifier could not be recognized.	Connect the amplifier. Alternatively, replace the amplifier.
	16#6A720000	RF Tag Missing Error	There is no RF Tag in the communications range.	Adjust the equipment so that the RF Tag enters inside the communications range.
	16#6A700000	RF Tag Communications Failure	An error occurred during communications with the RF Tag, preventing a normal end.	Change the movement speed of the RF Tag and the distance between RF Tags so that they are within the specified range. Also, implement noise countermeasures if there is excessive ambient noise.
	16#6A710000	RF Tag Verification Error	The correct data could not be written to the RF Tag.	Change the movement speed of the RF Tag and the distance between RF Tags so that they are within the specified range. Also, implement noise countermeasures if there is excessive ambient noise.

Expansion error code	Expansion error code	Status	Description	Corrective action
16#3D14	16#6A730000	RF Tag Data Loss	Correct data is not written to the RF Tag, and there is a possibility that the data has been lost.	Perform re-processing when the RF Tag exists in the communications range of the reader/writer. Change the movement speed of the RF Tag and the distance between RF Tags so that they are within the specified range. Also, implement noise countermeasures if there is excessive ambient noise.
	16#6A7A0000	RF Tag Address Error	The address of the RF Tag is incorrect.	Check the memory capacity of the RF Tag being used, and correct the specified address so that it is within the range of the memory capacity.
	16#6A7D0000	RF Tag Write Protect Error	An attempt was made to write to a write-protected area of the RF Tag.	Correct the specified address and number of bytes to the correct value. Alternatively, remove write protection.
	16#6A790000	RF Tag Response Error	The RF Tag returned an error response, preventing a normal end.	Implement noise countermeasures if there is excessive ambient noise. [1]
	16#6A7E0000	RF Tag Lock Error	An attempt was made to write to a locked area of the RF Tag.	Change the RF Tag.
	16#6A7F0000	RF Tag Customer Code Error	Communications were performed with an RF Tag that cannot be used.	Change the RF Tag.
	16#6AC00000	Undefined Command	It cannot be executed because it is an undefined command.	Set the correct variables in DeviceVariables.
	16#6AC10000	Invalid Command Parameter	The command cannot be executed because the command parameter is erroneous.	Set the correct variables in DeviceVariables.
	16#6AC20000	Command Execution Failure	The command cannot be executed because the command execution conditions have not been established.	Correct to a RF communications option that can be executed for the command.

*1. Change the RF Tag if the problem is not solved with noise countermeasures.

Sample programming

Program Description

Write the 16-byte data to memory address 8 of the RF Tag.

Preconditions

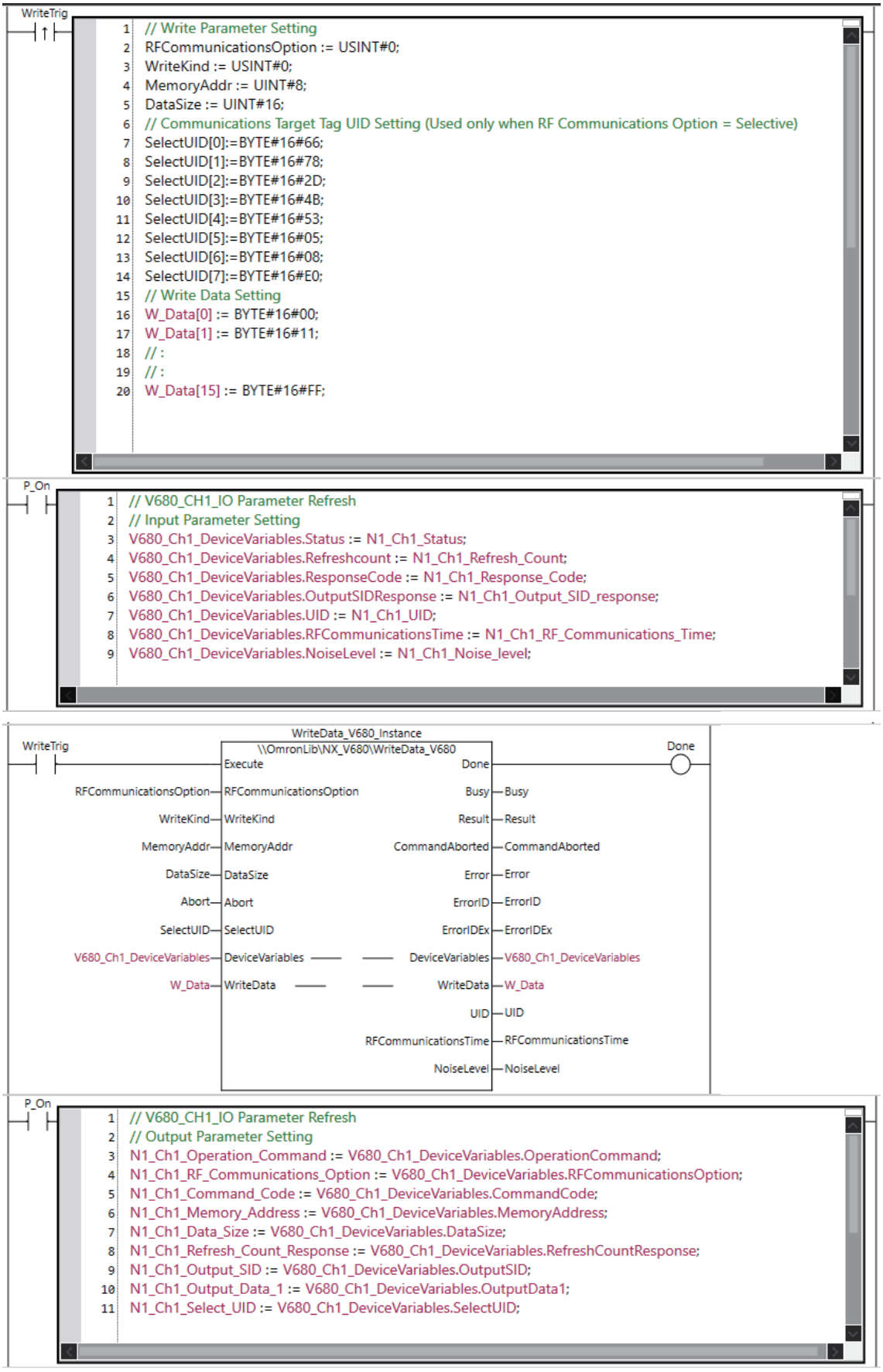
Create device variables for the RFID Unit to be operated, and use external references to the variables in the user program. Refer to the *Sysmac Studio Version 1 Operation Manual* (W504) for details on how to create device variables.

Main Variables

Variable or member	Name	Data Type	Default value	Description
WriteData_V680_Instance	Write Data V680 FB	OmronLib\NX_V680\WriteData_V680	---	Function block for implementing write for the NX-V680
V680_Ch1_DeviceVariables	Ch1 Device Variable	OmronLib\NX_V680\sV680DeviceVariables	---	Structure of Ch1 Device Variable
W_Data	Buffer for Write Data	ARRAY[0..8191] OF BYTE	16#00	Set the write data
WriteTrig	Write Execution Flag	BOOL	FALSE	The variable is TRUE when write is executed
RFCommunicationsOption	RF Communications Option	USINT	0	Set the RF communications option
WriteKind	Write Type	USINT	0	Set the write type
MemoryAddr	Memory Address	UINT	0	Set the communications start address
DataSize	Data Size	UINT	0	Set the communications data size
Abort	Execution Stop Flag	BOOL	FALSE	The variable is TRUE when write execution has stopped
SelectUID	Select UID	ARRAY[0..7] OF BYTE	16#00	Set the UID of the communications target tag
CommandStage	Command Busy	UINT	0	Execution status of the ST program
Execute	Execution Flag	BOOL	FALSE	Execution status flag in the ST program
Done	Execution Completion Flag	BOOL	---	Output variable for *WriteData_V680_Instance*
Busy	Enable	BOOL	---	Output variable for *WriteData_V680_Instance*
Result	Result Flag	BOOL	---	Output variable for *WriteData_V680_Instance*

Variable or member	Name	Data Type	Default value	Description
CommandAborted	Abort Flag	BOOL	---	Output variable for *WriteData_V680_Instance*
Error	Error Flag	BOOL	---	Output variable for *WriteData_V680_Instance*
ErroriD	Error code	WORD	---	Output variable for *WriteData_V680_Instance*
ErrorIDEx	Expansion error code	DWORD	---	Output variable for *WriteData_V680_Instance*
UID	UID	ARRAY[0..7] OF BYTE	---	Output variable for *WriteData_V680_Instance*
RFCommunications-Time	RF Communications Time	UINT	---	Output variable for *WriteData_V680_Instance*
NoiseLevel	Noise Level	UINT	---	Output variable for *WriteData_V680_Instance*

Ladder Diagram



● Code of Inline ST (Zeroth Line of Ladder Diagram):

```
//Write Parameter Setting
RFCommunicationsOption := USINT#0;
WriteKind := USINT#0;
MemoryAddr := UINT#8;
DataSize := UINT#16;
//Communications Target Tag UID Setting (Used only when RF Communications Option =
Selective)
SelectUID[0]:=BYTE#16#66;
SelectUID[1]:=BYTE#16#78;
SelectUID[2]:=BYTE#16#2D;
SelectUID[3]:=BYTE#16#4B;
SelectUID[4]:=BYTE#16#53;
SelectUID[5]:=BYTE#16#05;
SelectUID[6]:=BYTE#16#08;
SelectUID[7]:=BYTE#16#E0;
//Write Data Setting
W_Data[0] := BYTE#16#00;
W_Data[1] := BYTE#16#11;
//      :
//      :
W_Data[15] := BYTE#16#FF;
```

● Code of Inline ST (First Line of Ladder Diagram):

```
//V680_CH1_IO Parameter Refresh
//Input Parameter Setting
V680_Ch1_DeviceVariables.Status := N1_Ch1_Status;
V680_Ch1_DeviceVariables.Refreshcount := N1_Ch1_Refresh_Count;
V680_Ch1_DeviceVariables.ResponseCode := N1_Ch1_Response_Code;
V680_Ch1_DeviceVariables.OutputSIDResponse := N1_Ch1_Output_SID_response;
V680_Ch1_DeviceVariables.UID := N1_Ch1_UID;
V680_Ch1_DeviceVariables.RFCommunicationsTime := N1_Ch1_RF_Communications_Time;
V680_Ch1_DeviceVariables.NoiseLevel := N1_Ch1_Noise_level;
```

● Code of Inline ST (Third Line of Ladder Diagram):

```
//V680_CH1_IO Parameter Refresh
//Output Parameter Setting
N1_Ch1_Operation_Command := V680_Ch1_DeviceVariables.OperationCommand;
N1_Ch1_RF_Communications_Option := V680_Ch1_DeviceVariables.RFCommunicationsOption;
N1_Ch1_Command_Code := V680_Ch1_DeviceVariables.CommandCode;
N1_Ch1_Memory_Address := V680_Ch1_DeviceVariables.MemoryAddress;
N1_Ch1_Data_Size := V680_Ch1_DeviceVariables.DataSize;
N1_Ch1_Refresh_Count_Response := V680_Ch1_DeviceVariables.RefreshCountResponse;
N1_Ch1_Output_SID := V680_Ch1_DeviceVariables.OutputSID;
N1_Ch1_Output_Data_1 := V680_Ch1_DeviceVariables.OutputData1;
N1_Ch1_Select_UID := V680_Ch1_DeviceVariables.SelectUID;
```

ST

```
//V680_CH1_IO Parameter Refresh
//Input Parameter Setting
V680_Ch1_DeviceVariables.Status := N1_Ch1_Status;
V680_Ch1_DeviceVariables.Refreshcount := N1_Ch1_Refresh_Count;
V680_Ch1_DeviceVariables.ResponseCode := N1_Ch1_Response_Code;
V680_Ch1_DeviceVariables.OutputSIDResponse := N1_Ch1_Output_SID_response;
V680_Ch1_DeviceVariables.UID := N1_Ch1_UID;
V680_Ch1_DeviceVariables.RFCommunicationsTime := N1_Ch1_RF_Communications_Time;
V680_Ch1_DeviceVariables.NoiseLevel := N1_Ch1_Noise_level;

CASE CommandStage OF
    (*Idle*)
    0 :
    // If WriteTrig changes to TRUE, the command is executed
    IF ( WriteTrig = TRUE ) THEN
        //Write Parameter Setting
        RFCommunicationsOption := USINT#0;
        WriteKind := USINT#0;
        MemoryAddr := UINT#8;
        DataSize := UINT#16;
        //Communications Target Tag UID Setting (Used only when RF Communications
Option = Selective)
        SelectUID[0]:=BYTE#16#66;
        SelectUID[1]:=BYTE#16#78;
        SelectUID[2]:=BYTE#16#2D;
        SelectUID[3]:=BYTE#16#4B;
        SelectUID[4]:=BYTE#16#53;
        SelectUID[5]:=BYTE#16#05;
        SelectUID[6]:=BYTE#16#08;
        SelectUID[7]:=BYTE#16#E0;
        //Write Data Setting
        W_Data[0] := BYTE#16#00;
        W_Data[1] := BYTE#16#11;
        //        :
        //        :
        W_Data[15] := BYTE#16#FF;
        //Write Execution Flag Setting
        Execute := TRUE;
        //Transit to Write Execution Status
        CommandStage := UINT#1;
    END_IF;

    (*Write execution result acquisition*)
    1 :
    IF (Busy = FALSE) THEN
        IF (Done = TRUE) THEN
        //Normal End
            (* --------------------↓Specify normal processing↓--------------------- *)

            (* ↑-------------------------------------------------------------↑ *)
            //Transit to Unit Operation Stop Wait Status
            CommandStage := UINT#2;

        ELSIF (Error = TRUE)THEN
            (* --------------------↓Specify error processing↓--------------------- *)

            (* ↑-------------------------------------------------------------↑ *)
            //Transit to Unit Operation Stop Wait Status
            CommandStage := UINT#2;
        END_IF;
    END_IF;
```

```
    (*Unit operation stop wait*)
    2:
    // If WriteTrig changes to FALSE, transition to idle state occurs
    IF ( WriteTrig = FALSE ) THEN
        CommandStage := UINT#0;
        //Write Execution End Flag Setting
        Execute := FALSE;
    END_IF;

END_CASE;

//Write Data FB
WriteData_V680_instance
(
    Execute :=Execute,
    RFCommunicationsOption := RFCommunicationsOption,
    WriteKind := WriteKind,
    MemoryAddr := MemoryAddr,
    DataSize := DataSize,
    Abort := Abort,
    SelectUID := SelectUID,
    Done => Done,
    Busy => Busy,
    Result => Result,
    CommandAborted => CommandAborted,
    Error => Error,
    ErrorID => ErrorID,
    ErrorIDEx => ErrorIDEx,
    UID => UID,
    RFCommunicationsTime => RFCommunicationsTime,
    NoiseLevel => NoiseLevel,
    DeviceVariables := V680_Ch1_DeviceVariables,
    WriteData := W_Data
);

//V680_CH1_IO Parameter Refresh
//Output Parameter Setting
N1_Ch1_Operation_Command := V680_Ch1_DeviceVariables.OperationCommand;
N1_Ch1_RF_Communications_Option := V680_Ch1_DeviceVariables.RFCommunicationsOption;
N1_Ch1_Command_Code := V680_Ch1_DeviceVariables.CommandCode;
N1_Ch1_Memory_Address := V680_Ch1_DeviceVariables.MemoryAddress;
N1_Ch1_Data_Size := V680_Ch1_DeviceVariables.DataSize;
N1_Ch1_Refresh_Count_Response := V680_Ch1_DeviceVariables.RefreshCountResponse;
N1_Ch1_Output_Data_1 := V680_Ch1_DeviceVariables.OutputData1;
N1_Ch1_Output_SID := V680_Ch1_DeviceVariables.OutputSID;
N1_Ch1_Select_UID := V680_Ch1_DeviceVariables.SelectUID;
```

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.

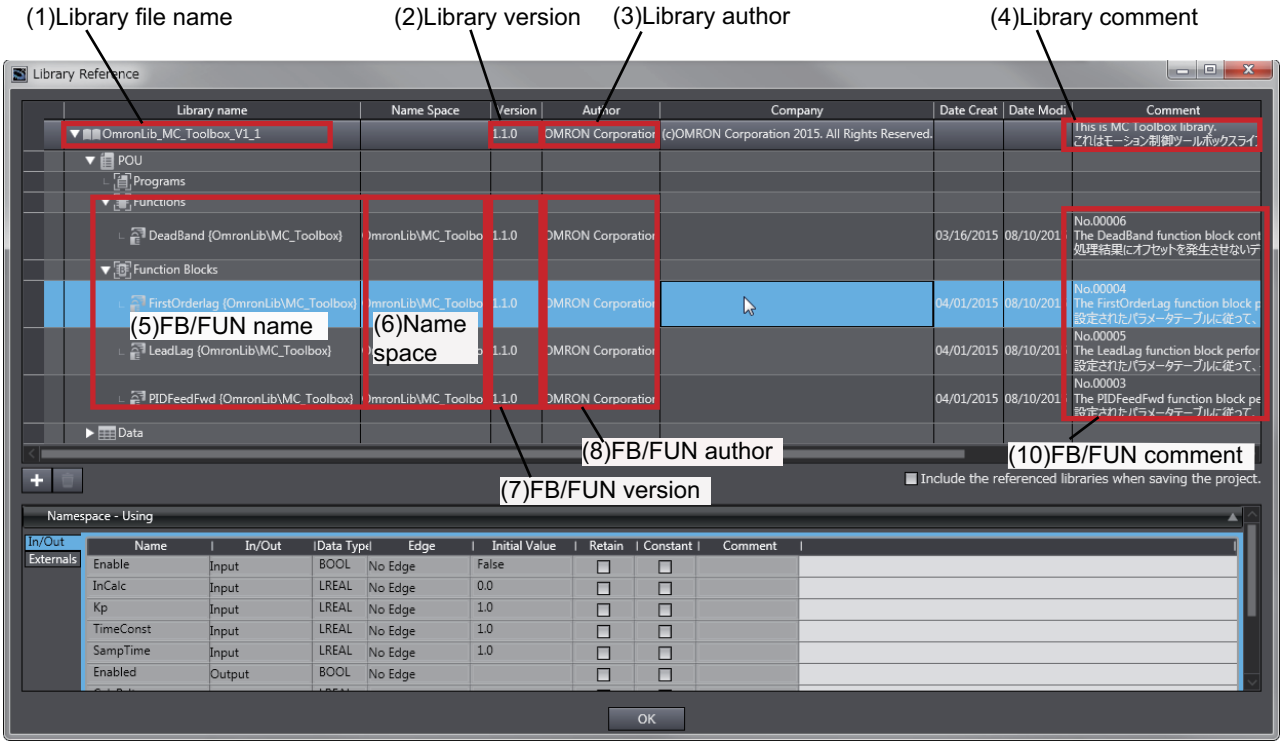
Referring to Attributes of Libraries, Function Blocks and Functions

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

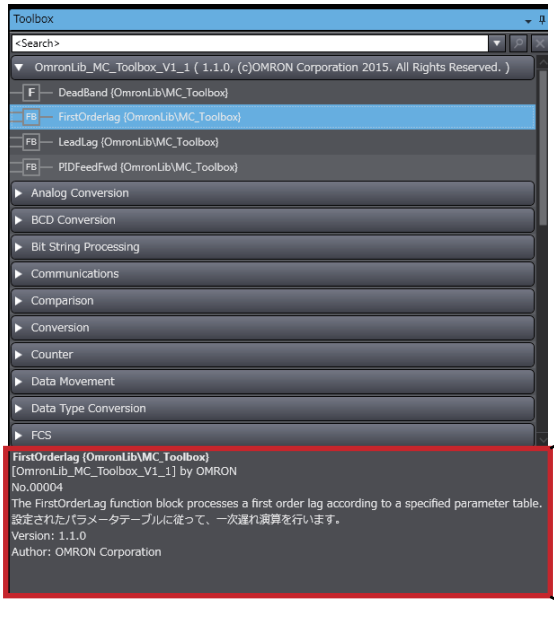
When you refer to the libraries, the library information is displayed at the locations shown below.



(b) Toolbox Pane

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

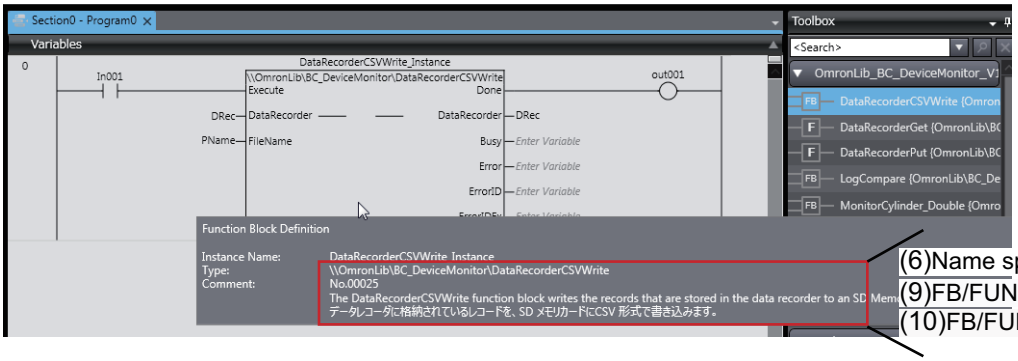
The text “by OMRON” which is shown on the right of the library name (1) indicates that this library was provided by OMRON.



(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.

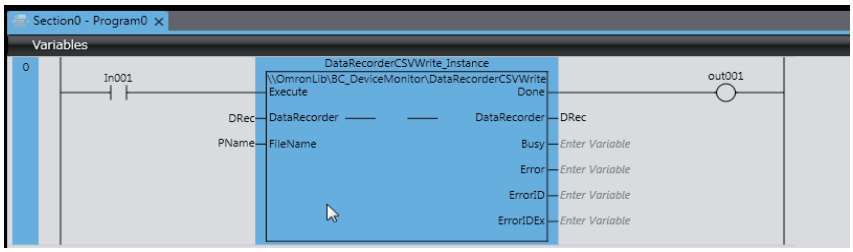
The following are the examples of items that you may need to customize.

- 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.

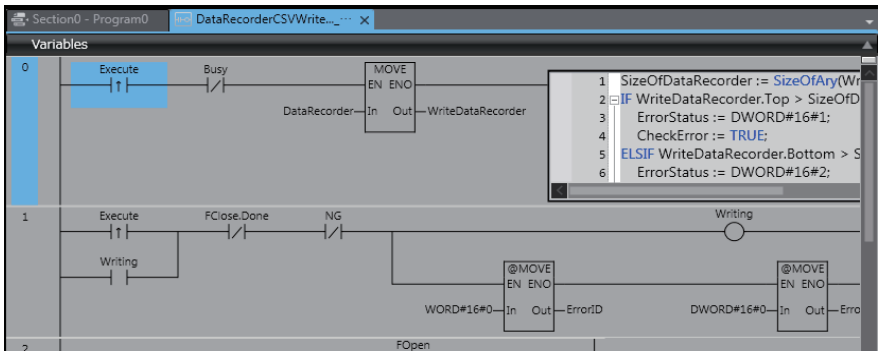
Use the following procedure to refer to the source codes of function blocks and functions.

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.