



Sysmac Library for

MQTT Communications Library

User's Manual

SYSMAC-XR020

NOTE

1. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, mechanical, electronic, photocopying, recording, or otherwise, without the prior written permission of OMRON.
2. 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.
3. 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.
- ODVA, CIP, CompoNet, DeviceNet, and EtherNet/IP are trademarks of ODVA.

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

Copyrights

- Microsoft product screen shots used with permission from Microsoft.

Introduction

Thank you for purchasing an NX-series CPU Unit.

This manual contains information that is necessary to use the Function Block (hereafter, sometimes abbreviated to FB) of the MQTT Communications Library. Please read this manual and make sure you understand the functionality and performance of the product before you attempt to use it in a control system.

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

Make sure to read the user's manual for each product before use.

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

Features of the Library

The MQTT Communications Library is a collection of software functional objects that are used for Pub/Sub-type message exchange through the MQTT broker.

This library supports the MQTT protocol with a version 3.1.1.

Using this library enables the following:

- Connection control with MQTT broker
- As a Subscriber, receiving information from Publishers that are on the network
- As a Publisher, sending information to Subscribers that are on the network
- Checking communication with MQTT broker and response time

Intended Audience

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

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

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

Applicable Products

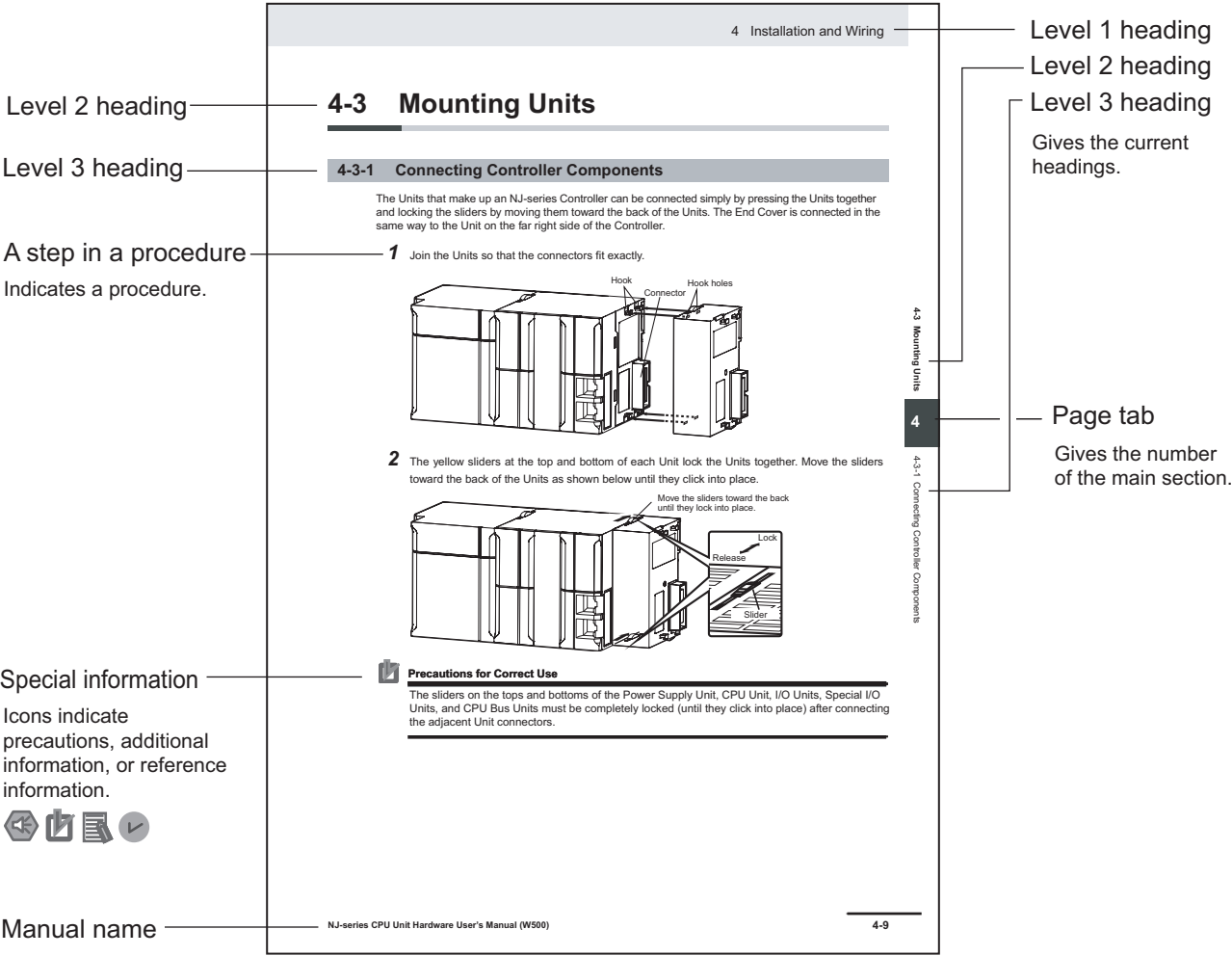
For the models and versions of NJ/NX-series CPU Unit, NY-series industrial PC, and Sysmac Studio that this library supports, refer to *Sysmac Library Version Information* in *SYSMAC-XR□□□□ Sysmac Library Catalog (Cat. No. P102)*.

You can download the catalog from the OMRON website (<https://www.fa.omron.co.jp/products/family/3459/download/catalog.html>).

Manual Structure

Page Structure

The following page structure is used in this manual.



Note This illustration is provided only as a sample. It may not literally appear in this manual.

Special Information

Special information in this manual is classified as follows:



Precautions for Safe Use

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



Precautions for Correct Use

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



Additional Information

Additional information to read as required.

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



Version Information

Information on differences in specifications and functionality for CPU Units with different unit versions and for different versions of the industrial-use PC, Sysmac Studio are given.

Sections in this Manual

		1
1	Sysmac Library Usage Procedure	2
2	MQTT Communications Library	3
3	Common Specifications of FBs	4
4	Individual Specifications of FBs	A
A	Appendix	I
I	Index	

CONTENTS

Introduction	1
Features of the Library	1
Intended Audience	1
Applicable Products	1
Manual Structure	2
Page Structure	2
Special Information	3
Sections in this Manual	5
Terms and Conditions Agreement	9
Warranty, Limitations of Liability	9
Application Considerations	10
Disclaimers	10
Statement of security responsibilities for assumed use cases and against threats	11
Safety Precautions	12
Definition of Precautionary Information	12
Symbols	12
Caution	13
Precautions for Correct Use	14
Using the Library	14
Operation	14
Regulations and Standards	15
Related Manuals	16
Revision History	18

Section 1 Sysmac Library Usage Procedure

1-1 Procedure to Use Sysmac Library Installed Using the Installer	1-2
1-1-1 Using a Newly Installed Sysmac Library	1-2
1-1-2 Using an Upgraded Sysmac Library	1-4
1-2 How to Use Sysmac Library in the CPU Unit	1-6

Section 2 MQTT Communications Library

2-1 Overview	2-2
2-1-1 Connection Control with MQTT Broker	2-2
2-1-2 Receiving Information from Publishers	2-2
2-1-3 Sending Information to Subscribers	2-2
2-1-4 Checking Communication with MQTT Broker and Response Time	2-2
2-2 Usage Method	2-3
2-2-1 Making a Client Operate as a Publisher	2-4
2-2-2 Making a Client Operate as a Subscriber	2-4
2-2-3 Checking Communication with MQTT Broker and Response Time	2-5
2-3 Packets of MQTT Protocol	2-6

Section 3 Common Specifications of FBs

3-1 Common Variables.....	3-2
3-1-1 Definition of Input Variables and Output Variables	3-2
3-1-2 Execute-type Function Blocks	3-2
3-1-3 Enable-type Function Blocks	3-3
3-2 Precautions	3-5
3-2-1 Nesting	3-5
3-2-2 Instruction Options	3-5
3-2-3 Re-execution of Function Blocks	3-5

Section 4 Individual Specifications of FBs

MQTTClient.....	4-2
Function Block and Function Information	4-2
Input Variables	4-2
Output Variables	4-3
Input-Output Variables	4-4
Structure	4-4
Function	4-5
Timing Charts.....	4-8
Additional Information	4-10
Precautions for Correct Use	4-10
Troubleshooting	4-11
Sample Programming	4-15
MQTTPubAryByte	4-24
Function Block and Function Information	4-24
Input Variables	4-24
Output Variables	4-25
Input-Output Variables	4-25
Structure	4-26
Function	4-26
Timing Charts.....	4-28
Precautions for Correct Use	4-32
Troubleshooting	4-32
Sample Programming	4-33
MQTTPubString.....	4-34
Function Block and Function Information	4-34
Input Variables	4-34
Output Variables	4-35
Input-Output Variables	4-35
Function	4-36
Timing Charts.....	4-36
Additional Information	4-36
Precautions for Correct Use	4-36
Troubleshooting	4-36
Sample Programming	4-37
MQTTSubAryByte	4-38
Function Block and Function Information	4-38
Input Variables	4-38
Output Variables	4-39
Input-Output Variables	4-40
Function	4-40
Timing Charts.....	4-42
Precautions for Correct Use	4-45
Troubleshooting	4-46
Sample Programming	4-47
MQTTSubString.....	4-48
Function Block and Function Information	4-48

Input Variables	4-48
Output Variables	4-49
Input-Output Variables	4-50
Function	4-50
Timing Charts.....	4-50
Precautions for Correct Use	4-50
Troubleshooting	4-50
Sample Programming	4-51
MQTTPing	4-52
Function Block and Function Information	4-52
Input Variables	4-52
Output Variables	4-53
Input-Output Variables	4-53
Function	4-53
Timing Charts.....	4-54
Precautions for Correct Use	4-55
Troubleshooting	4-56
Sample Programming	4-56

Appendix

A-1 Referring to Library Information.....	A-2
A-1-1 Library Attributes, and FB or FUN Attributes.....	A-2
A-1-2 Referring to Attributes of Libraries, Function Blocks, and Functions	A-2
A-2 Procedure for Connection to Azure IoT Hub via MQTT.....	A-5
A-2-1 Overall Procedure	A-5
A-2-2 System Configuration and Properties of Configuration Elements	A-6
A-2-3 Preliminary Preparations on Azure Side	A-8
A-2-4 Network Setting for PC.....	A-10
A-2-5 Network Setting for CPU Unit.....	A-10
A-2-6 Secure Socket Setting for CPU Unit	A-11
A-2-7 Creation of Program and Execution of Program	A-15
A-3 Procedure for Connection to AWS IoT via MQTT	A-22
A-3-1 Overall Procedure	A-22
A-3-2 System Configuration and Properties of Configuration Elements	A-23
A-3-3 Preliminary Preparations on AWS Side.....	A-24
A-3-4 Network Setting for PC.....	A-25
A-3-5 Network Setting for CPU Unit.....	A-25
A-3-6 Secure Socket Setting for CPU Unit	A-26
A-3-7 Creation of Program and Execution of Program	A-28

Index

Terms and Conditions Agreement

Warranty, Limitations of Liability

Warranties

● Exclusive Warranty

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

● Limitations

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

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

● Buyer Remedy

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

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

Limitation on Liability; Etc

OMRON COMPANIES SHALL NOT BE LIABLE FOR SPECIAL, INDIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, LOSS OF PROFITS OR PRODUCTION OR COMMERCIAL LOSS IN ANY

WAY CONNECTED WITH THE PRODUCTS, WHETHER SUCH CLAIM IS BASED IN CONTRACT, WARRANTY, NEGLIGENCE OR STRICT LIABILITY.

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

Application Considerations

Suitability of Use

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

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

Programmable Products

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

Disclaimers

Performance Data

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

Change in Specifications

Product specifications and accessories may be changed at any time based on improvements and other reasons. It is our practice to change part numbers when published ratings or features are changed, or when significant construction changes are made. However, some specifications of the Product may

be changed without any notice. When in doubt, special part numbers may be assigned to fix or establish key specifications for your application. Please consult with your Omron's representative at any time to confirm actual specifications of purchased Product.

Errors and Omissions

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

Statement of security responsibilities for assumed use cases and against threats

OMRON SHALL NOT BE RESPONSIBLE AND/OR LIABLE FOR ANY LOSS, DAMAGE, OR EXPENSES DIRECTLY OR INDIRECTLY RESULTING FROM THE INFECTION OF OMRON PRODUCTS, ANY SOFTWARE INSTALLED THEREON OR ANY COMPUTER EQUIPMENT, COMPUTER PROGRAMS, NETWORKS, DATABASES OR OTHER PROPRIETARY MATERIAL CONNECTED THERETO BY DISTRIBUTED DENIAL OF SERVICE ATTACK, COMPUTER VIRUSES, OTHER TECHNOLOGICALLY HARMFUL MATERIAL AND/OR UNAUTHORIZED ACCESS.

It shall be the users sole responsibility to determine and use adequate measures and checkpoints to satisfy the users particular requirements for (i) antivirus protection, (ii) data input and output, (iii) maintaining a means for reconstruction of lost data, (iv) preventing Omron Products and/or software installed thereon from being infected with computer viruses and (v) protecting Omron Products from unauthorized access.



Safety Precautions

Definition of Precautionary Information





The following notation is used in this user's manual to provide precautions required to ensure safe use of this library on the NX-series.

The safety precautions that are provided are extremely important for 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 that disassembly is prohibited.
	The triangle symbol indicates precautions (including warnings). The specific operation is shown in the triangle and explained in text. This example indicates a precaution for electric shock.
	The triangle symbol indicates precautions (including warnings). The specific operation is shown in the triangle and explained in text. This example indicates a general precaution.
	The filled circle symbol indicates operations that you must do. The specific operation is shown in the circle and explained in text. This example shows a general precaution for something that you must do.

Caution

Caution

Read all related manuals carefully before you use this library.



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



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



The Sysmac Library and manuals are assumed to be used by personnel that is given in Intended Audience in this manual. Otherwise, do not use them.



Perform the test run by holding an emergency stop switch in hand or otherwise prepare for rapid motor operation in an application to control the motor.
Also perform the test run by using parameters for which the motor does not rapidly accelerate or decelerate before you gradually adjust the parameters.



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



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



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



When you use actual devices, also use programs such as safety circuits, device interlocks, I/O with other devices, and other control procedures.



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



Create a user program that will produce the intended device operation.



Precautions for Correct Use

Using the Library

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

Operation

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

Regulations and Standards

Refer to the following manuals for Regulations and Standards.

- *NX-series NX102 CPU Unit Hardware User's Manual (Cat. No. W593)*
- *NX-series NX1P2 CPU Unit Hardware User's Manual (Cat. No. W578)*

Related Manuals

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

Manual name	Cat. No.	Model numbers	Application	Description
NX-series NX102 CPU Unit Hardware User's Manual	W593	NX102-□□□□	Learning the basic specifications of the NX102 CPU Units, including introductory information, designing, installation, and maintenance. Mainly hardware information is provided.	An introduction to the entire NX102 system is provided along with the following information on the CPU Unit. <ul style="list-style-type: none"> • Features and system configuration • Introduction • Part names and functions • General specifications • Installation and wiring • Maintenance and inspection
NX-series NX1P2 CPU Unit Hardware User's Manual	W578	NX1P2-□□□□	Learning the basic specifications of the NX1P2 CPU Units, including introductory information, designing, installation, and maintenance. Mainly hardware information is provided.	An introduction to the entire NX1P2 system is provided along with the following information on the CPU Unit. <ul style="list-style-type: none"> • Features and system configuration • Introduction • Part names and functions • General specifications • Installation and wiring • Maintenance and inspection
NJ/NX-series CPU Unit Software User's Manual	W501	NX701-□□□□ NX102-□□□□ NX1P2-□□□□ NJ501-□□□□ NJ301-□□□□ NJ101-□□□□	Learning how to program and set up an NJ/NX-series CPU Unit. Mainly software information is provided.	The following information is provided on a Controller built with an NJ/NX-series CPU Unit. <ul style="list-style-type: none"> • CPU Unit operation • CPU Unit features • Initial settings • Programming based on IEC 61131-3 language specifications
NJ/NX-series Instructions Reference Manual	W502	NX701-□□□□ NX102-□□□□ NX1P2-□□□□ NJ501-□□□□ NJ301-□□□□ NJ101-□□□□	Learning detailed specifications on the basic instructions of an NJ/NX-series CPU Unit.	The instructions in the instruction set (IEC 61131-3 specifications) are described.

Manual name	Cat. No.	Model numbers	Application	Description
NJ/NX-series CPU Unit Motion Control User's Manual	W507	NX701-□□□□ NX102-□□□□ NX1P2-□□□□ NJ501-□□□□ NJ301-□□□□ NJ101-□□□□	Learning about motion control settings and pro- gramming con- cepts.	The settings and opera- tion of the CPU Unit and programming concepts for motion control are descri- bed.
NJ/NX-series Motion Control Instruc- tions Reference Manual	W508	NX701-□□□□ NX102-□□□□ NX1P2-□□□□ NJ501-□□□□ NJ301-□□□□ NJ101-□□□□	Learning about the specifica- tions of the mo- tion control in- structions.	The motion control in- structions are described.
NJ/NX-series CPU Unit Built-in EtherNet/IP™ Port User's Manual	W506	NX701-□□□□ NX102-□□□□ NX1P2-□□□□ NJ501-□□□□ NJ301-□□□□ NJ101-□□□□	Using the built-in EtherNet/IP port on an NJ/NX- series CPU Unit.	Information on the built-in EtherNet/IP port is provid- ed. Information is provided on the basic setup, tag data links, and other features.
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.

Revision History

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

Cat. No. W625-E1-02

Revision code

Revision code	Date	Revised content
01	July 2021	Original production
02	October 2021	Revised for error correction

Sysmac Library Usage Procedure

The section describes the procedure to use Sysmac Library installed using the installer, and Sysmac Library in the CPU Unit.

1-1	Procedure to Use Sysmac Library Installed Using the Installer.....	1-2
1-1-1	Using a Newly Installed Sysmac Library	1-2
1-1-2	Using an Upgraded Sysmac Library.....	1-4
1-2	How to Use Sysmac Library in the CPU Unit.....	1-6

1-1 Procedure to Use Sysmac Library Installed Using the Installer

This section describes the procedure to use Sysmac Library installed using the installer.

There are two ways to use libraries.

- Using a Newly Installed Sysmac Library
- Using an Upgraded Sysmac Library

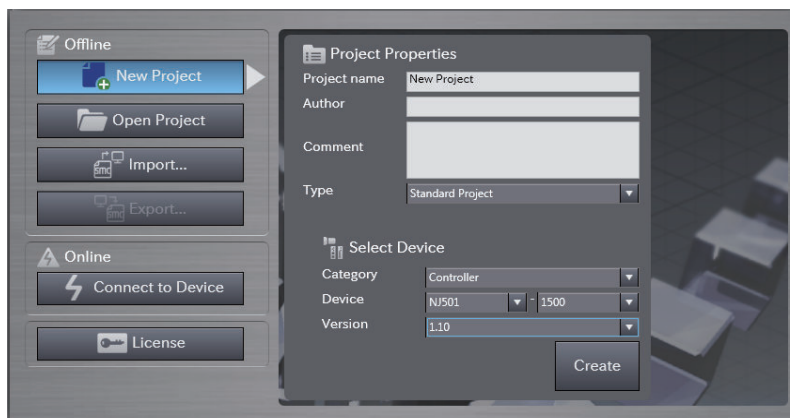


Version Information

Refer to *Applicable Products* on page 1 for the models and versions of Controller and Sysmac Studio that can use this library.

1-1-1 Using a Newly Installed Sysmac Library

- 1 Start the Sysmac Studio and open a project using Sysmac Library, or create a new one.

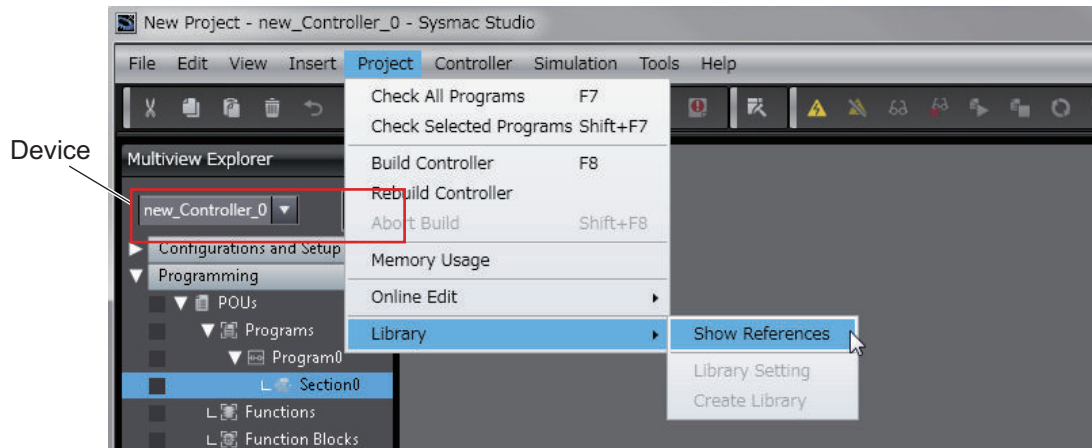


Precautions for Correct Use


If you create a new project, be sure to configure the settings as follows to enable use of the Sysmac Library. Without the settings below, you cannot proceed to Step 2 and later steps.

- Set the project type to Standard Project or Library Project.
- Set the device category to Controller.
- For the Controller and version in device selection, refer to *Applicable Products* on page 1.

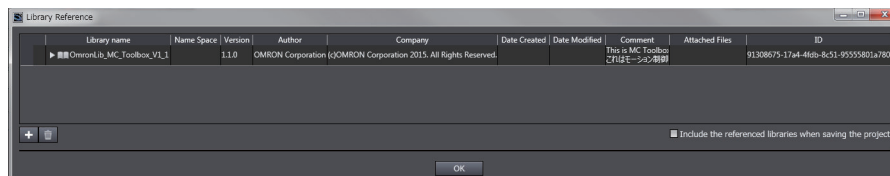
- 2 Select **Project – Library – Show References**.



Precautions for Correct Use

If you have more than one device registered in the project, make sure that the currently selected device is the NX102 CPU Unit or the NX1P2 CPU Unit. If the NX102 CPU Unit or the NX1P2 CPU Unit is not selected, the menu for browsing the library will not appear. When the selected device is the NX102 CPU Unit or the NX1P2 CPU Unit, the device icon displayed in Multiview Explorer changes to .

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



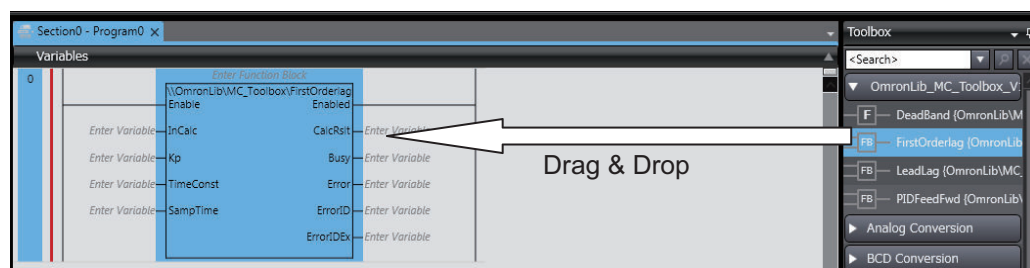
Sysmac Library is read into the project.

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

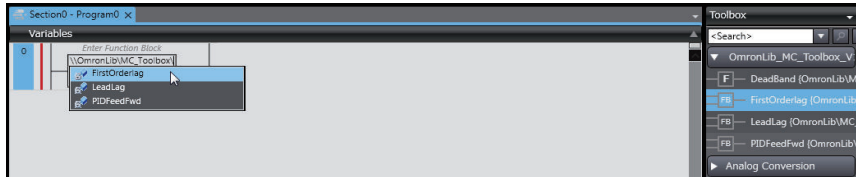
For the procedure for adding and setting libraries in the above screen, refer to the *Sysmac Studio Version 1 Operation Manual (Cat. No. W504)*.

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 in the Toolbox and drag and drop it onto the Ladder Editor.

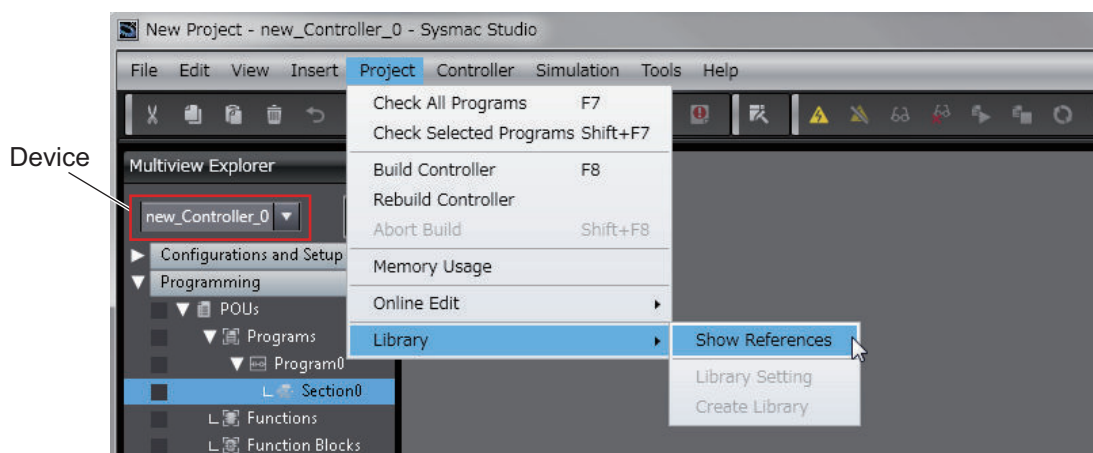


- Right-click the Ladder Editor, select **Insert Function Block** in the menu, and enter the fully qualified name ($\backslash\text{namespacename}\backslash\text{FBname}$).




1-1-2 Using an Upgraded Sysmac Library

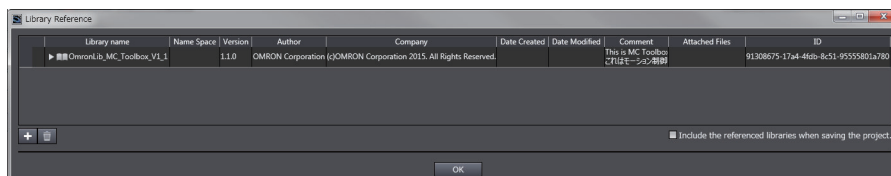
- 1 Start Sysmac Studio and open a project in which any old-version Sysmac Library is included.
- 2 Select **Project – Library – Show References**.



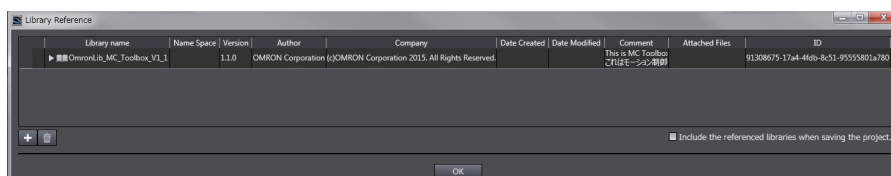
Precautions for Correct Use

If you have more than one device registered in the project, make sure that the currently selected device is the NX102 CPU Unit or the NX1P2 CPU Unit. If the NX102 CPU Unit or the NX1P2 CPU Unit is not selected, the menu for browsing the library will not appear. When the selected device is the NX102 CPU Unit or the NX1P2 CPU Unit, the device icon displayed in Multiview Explorer changes to .

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



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





Precautions for Correct Use

Upgrade the Sysmac Library version, and then execute All Program Check, and confirm that there are no errors in the Build Window Program Check results.
From the Main Menu, select **Project – All Program Check**.

1-2 How to Use Sysmac Library in the CPU Unit

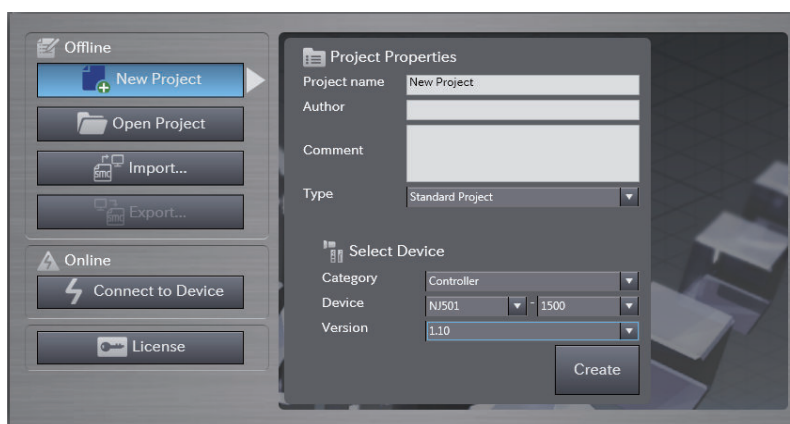
Even when Sysmac Library is not installed on your computer, you can use Sysmac Library by uploading it from the CPU Unit to your computer.

The procedure to use Sysmac Library in the CPU Unit is as follows.

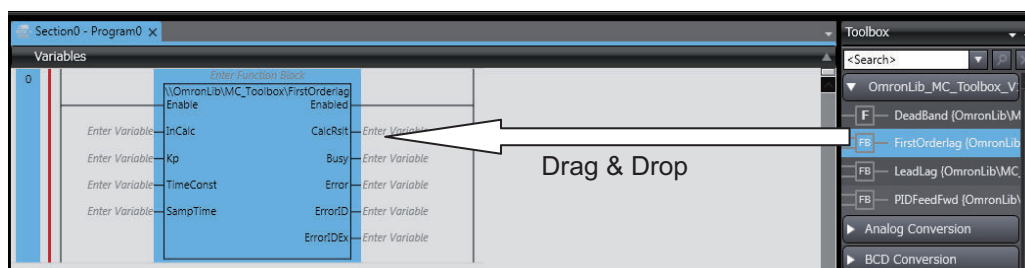
Version Information

Refer to *Applicable Products* on page 1 for the version of Sysmac Studio that can use this library.

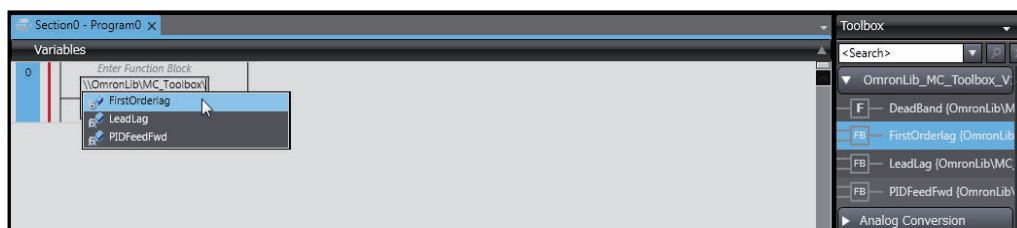
- 1 Start the Sysmac Studio and create a new project in which you want to use Sysmac Library.



- 2 Connect online to the CPU Unit.
- 3 Upload the POU's in which Sysmac Library is used.
Now, when you select the Ladder Editor or ST Editor, the function blocks included in the Sysmac Library used in the uploaded POU's appear in the Toolbox.
- 4 Insert the Sysmac Library's function blocks into the circuit using one of the following two methods.
 - Select the desired function block in the Toolbox and drag and drop it onto the Ladder Editor.



- Right-click the Ladder Editor, select **Insert Function Block** in the menu, and enter the fully qualified name (¥¥namespace¥¥FBname).



Precautions for Correct Use

- The Sysmac Studio installs Sysmac Library library files to the specified folder on the computer if they are not present. However, the Sysmac Studio does not install libraries 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 Library from a CPU Unit does not install the manual and help files for Sysmac Library, unlike installation using the installer. Please install the manual and help files using the installer if you need them.

MQTT Communications Library

This section provides the overviews of function and usage method.

2-1	Overview	2-2
2-1-1	Connection Control with MQTT Broker	2-2
2-1-2	Receiving Information from Publishers.....	2-2
2-1-3	Sending Information to Subscribers	2-2
2-1-4	Checking Communication with MQTT Broker and Response Time	2-2
2-2	Usage Method.....	2-3
2-2-1	Making a Client Operate as a Publisher.....	2-4
2-2-2	Making a Client Operate as a Subscriber.....	2-4
2-2-3	Checking Communication with MQTT Broker and Response Time	2-5
2-3	Packets of MQTT Protocol	2-6

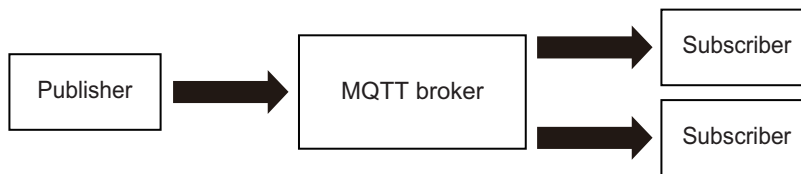
2-1 Overview

The MQTT Communications Library is a collection of software functional objects that are used for Pub/Sub-type message exchange through the MQTT broker.

This library supports the MQTT protocol with a version 3.1.1.

Using this library enables the following:

- Connection control with MQTT broker
- As a Subscriber, receiving information from Publishers that are on the network
- As a Publisher, sending information to Subscribers that are on the network
- Checking communication with MQTT broker and response time



2-1-1 Connection Control with MQTT Broker

This library establishes connection with the MQTT broker specified with an IP address or a host name, and monitors the connection status. Socket communications or secure socket communications can be selected as the communication method with the MQTT broker.

2-1-2 Receiving Information from Publishers

You can specify a topic filter for the MQTT broker, subscribe to the MQTT broker, and receive information from Publishers via the MQTT broker. For the topic filter, it is possible not only to specify a single topic name to subscribe to the single topic name, but also to use a wildcard for subscribing to multiple topics to specify them. The received information can be obtained in byte array or in STRING data type.

2-1-3 Sending Information to Subscribers

It is possible to send a message belonging to a topic to an MQTT broker and send information to a Subscriber who subscribes by specifying the topic to which the sent message belongs via the MQTT broker. It is also possible to set the QoS to specify the level at which the transmission of the message is guaranteed. The sent message can be specified as a byte array or STRING type.

2-1-4 Checking Communication with MQTT Broker and Response Time

Communication with the MQTT broker can be checked by sending a *PINGREQ* packet to it and receiving the *PINGRESP* packet.

In addition, you can measure the response time between when the *PINGREQ* packet is sent and when the *PINGRESP* packet is received.

2-2 Usage Method

This library provides the functions as an MQTT client in the form of the following function blocks.

Function block name	Function
MQTTClient	<ul style="list-style-type: none"> • Connection control with MQTT broker • Send and receive processing for messages • KeepAlive function
MQTTPubAryByte, MQTTPubString	<ul style="list-style-type: none"> • Request to send a PUBLISH message • Managing the message exchange status of the PUBLISH message
MQTTSubAryByte, MQTTSubString	<ul style="list-style-type: none"> • Requesting and canceling subscription • Managing the message exchange status of the message under subscription • Receiving the PUBLISH message received during subscription
MQTTPing	<ul style="list-style-type: none"> • Request to send a <i>PINGREQ</i> packet • Acknowledging the reception of the <i>PINGRESP</i> packet • Measuring a PING message response time

The relationship between the MQTTClient instruction and the other instructions is shown in the following figure.

The MQTTPubAryByte instruction, MQTTPubString instruction, MQTTSubAryByte instruction, MQTTSubString instruction, and MQTTPing instruction provide communication with an MQTT broker via MQTTClient instruction.

These instructions provide data exchange via MQTTClient instruction and variable *ClientReference*.

Set the address and TCP port number of the MQTT broker to the member variables *IpAdr* and *PortNo* of the input variable *ConnectionSettings* of the MQTTClient instruction.

When you use secure socket communications for sending and receiving messages, you need to make a secure socket setting in advance.

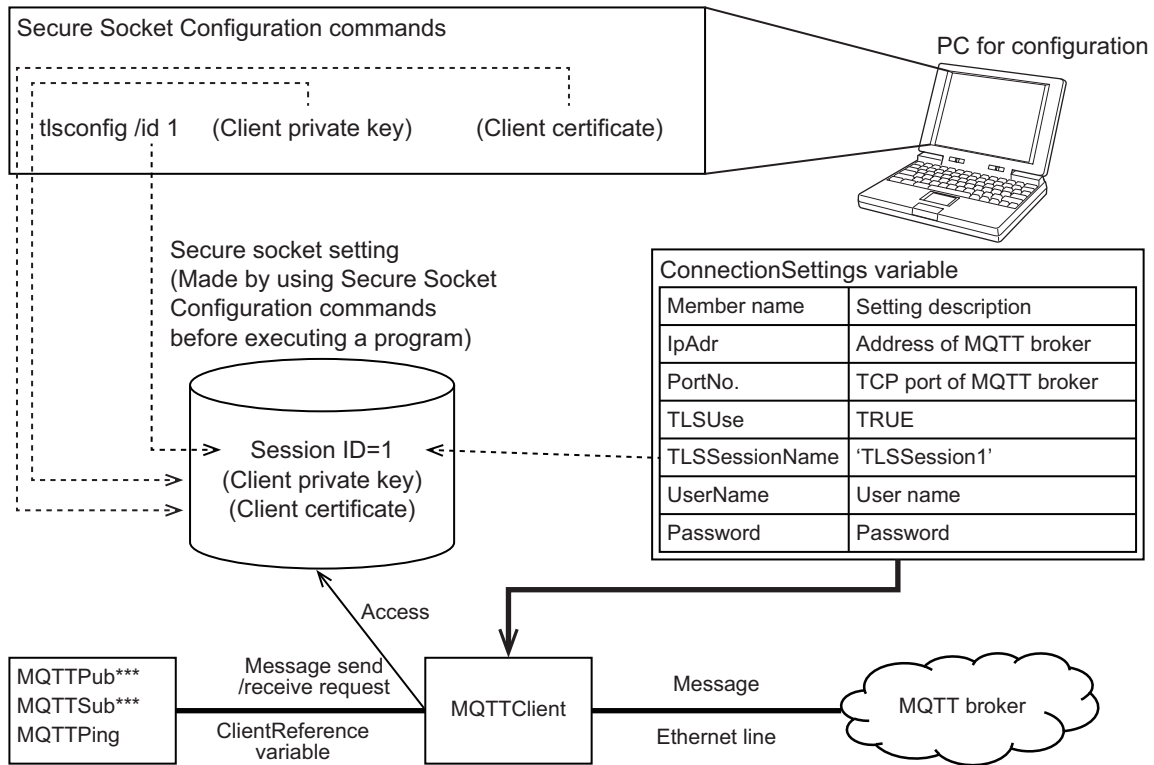
Refer to the *NJ/NX-series CPU Unit Built-in EtherNet/IP™ Port User's Manual (Cat. No. W506)* for details of the secure socket setting.

Set the member variable *TLSUse* of the input variable *ConnectionSettings* of the MQTTClient instruction to TRUE, and set *TLSSessionName*.

TLSSessionName is specified with a session ID that Secure Socket Configuration commands set regarded as a session name.

For example, when the session ID is 1, *TLSSessionName* becomes *TLSSession1*.

For MQTT brokers that do not require a client private key or client certificate, you may need to specify a user name and password.



You can make an MQTT client operate as not only a Publisher but also a Subscriber. In this case, use the MQTTPubAryByte instruction or the MQTTPubString instruction, and the MQTTSubAryByte instruction or the MQTTSubString instruction.

When you check communication with the MQTT broker and a response time, use the MQTTPing instruction.

2-2-1 Making a Client Operate as a Publisher

Whether you want to make an MQTT client operate as a Publisher or a Subscriber, you need to use the MQTTClient instruction.

To make a client operate as a Publisher, use the MQTTPubAryByte instruction or the MQTTPubString instruction.

- 1** Execute the MQTTClient instruction to establish connection with the MQTT broker.
- 2** Make the client operate as a Publisher.
After checking that the connection with the MQTT broker is established, execute the MQTTPubAryByte instruction or the MQTTPubString instruction to send the message you want to send. The MQTTPubAryByte instruction and the MQTTPubString instruction are different only in the data type in which to specify the message to send.
- 3** Stop execution of the MQTTClient instruction to cut the connection with the MQTT broker.

2-2-2 Making a Client Operate as a Subscriber

Whether you want to make an MQTT client operate as a Publisher or a Subscriber, you need to use the MQTTClient instruction.

To make a client operate as a Subscriber, use the MQTTSub AryByte instruction or the MQTTSubString instruction.

- 1** Execute the MQTTClient instruction to establish connection with the MQTT broker.
- 2** Make the client operate as a Subscriber.
After checking that the connection with the MQTT broker is established, execute the MQTTSub AryByte instruction or the MQTTSubString instruction to create the subscribed state.
The MQTTSub AryByte instruction and the MQTTSubString instruction are different only in the data type in which to output the received message.
 - Requesting subscription
Execute the MQTTSub AryByte instruction or the MQTTSubString instruction to put a topic to receive into the subscribed state.
 - Canceling subscription
To end the subscription, stop execution of the MQTTSub AryByte instruction or the MQTTSubString instruction.
- 3** Stop execution of the MQTTClient instruction to cut the connection with the MQTT broker.

2-2-3 Checking Communication with MQTT Broker and Response Time

The MQTTPing instruction is used for checking communication with the MQTT broker and a response time.

Even for checking, you need to use the MQTTClient instruction to establish connection with the MQTT broker in advance.

When operating as a Publisher, after confirming that the connection with the MQTT broker has been established, execute the MQTTPing instruction while the execution of the MQTTPub AryByte instruction and MQTTPubString instruction is stopped.

Similarly, when operating as a Subscriber, after confirming that the connection with the MQTT broker has been established, execute the MQTTPing instruction while the execution of the MQTTSub AryByte instruction and MQTTSubString instruction is stopped.

2-3 Packets of MQTT Protocol

This library uses the following MQTT protocol packets to exchange Pub/Sub type messages through the MQTT broker.

Packet name	Description
<i>CONNECT</i>	Request of connection to MQTT broker
<i>CONNACK</i>	Acknowledgement of connection
<i>PUBLISH</i>	Publication of message
<i>PUBACK</i>	Acknowledgement of publication
<i>PUBREC</i>	Reception of publication
<i>PUBREL</i>	Release of publication
<i>PUBCOMP</i>	Completion of publication
<i>SUBSCRIBE</i>	Subscription request
<i>SUBACK</i>	Acknowledgement of subscription
<i>UNSUBSCRIBE</i>	Unsubscription request
<i>UNSUBACK</i>	Acknowledgement of unsubscription
<i>PINGREQ</i>	PING request
<i>PINGRESP</i>	PING response
<i>DISCONNECT</i>	Disconnection from MQTT broker

Note For details of the packets, refer to the MQTT protocol with a version 3.1.1.

Common Specifications of FBs

This section describes the shared specifications of each FB in the Sysmac Library.

3-1	Common Variables	3-2
3-1-1	Definition of Input Variables and Output Variables	3-2
3-1-2	Execute-type Function Blocks	3-2
3-1-3	Enable-type Function Blocks	3-3
3-2	Precautions.....	3-5
3-2-1	Nesting	3-5
3-2-2	Instruction Options	3-5
3-2-3	Re-execution of Function Blocks	3-5

3-1 Common Variables

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

3-1-1 Definition of Input Variables and Output Variables

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

Variable	I/O	Data type	Function block type to use		Meaning	Definition
			Execute-type	Enable-type		
Execute	Input	BOOL	○		Execute	The processing is executed when the variable changes to TRUE.
Enable		BOOL		○	Execute	The processing is executed while the variable is TRUE.
Done	Output	BOOL	○		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	○	○	Executing	The variable is TRUE when the processing is in progress. It is FALSE when the processing is not in progress.
Error		BOOL	○	○	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	○	○	Error code	An error code is output.
ErrorIDEx		DWORD	○	○	Expansion error code	An expansion error code is output.

3-1-2 Execute-type Function Blocks

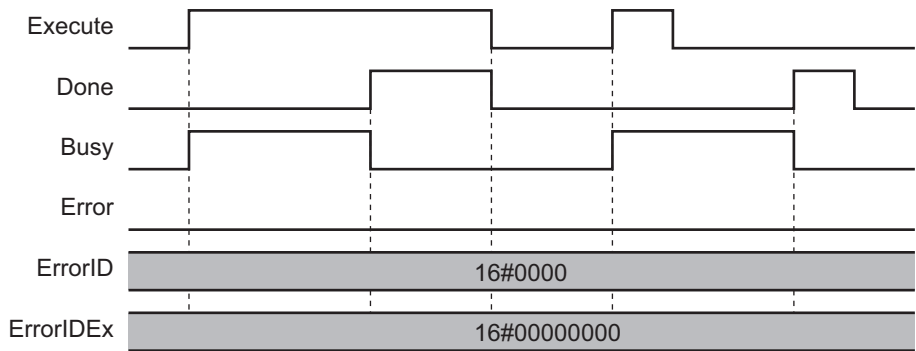
- Processing starts when *Execute* (Execute) changes to TRUE.
- When Execute changes to TRUE, *Busy* (Executing) also changes to TRUE. When processing is completed normally, *Busy* changes to FALSE and *Done* (Done) changes to TRUE.
- When continuously executing function blocks of the same instance, change the next *Execute* (Execute) to TRUE for at least one task period after *Done* (Done) changes to FALSE in the previous execution.
- If an error occurs in the function block, *Error* (Error) changes to TRUE and *Busy* changes to FALSE.

- If *Execute* is TRUE and *Done* or *Error* changes to TRUE, *Done* or *Error* changes to FALSE when *Execute* is changed to FALSE.
- If *Execute* is FALSE and *Done* or *Error* changes to TRUE, *Done* or *Error* changes to TRUE for only one task period.
- If an error occurs in the function block, 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 from FALSE to TRUE.

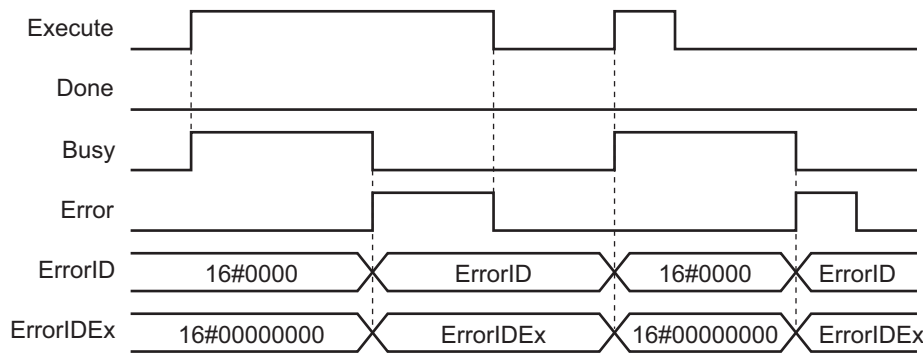
Timing Charts

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

● Normal End



● Errors



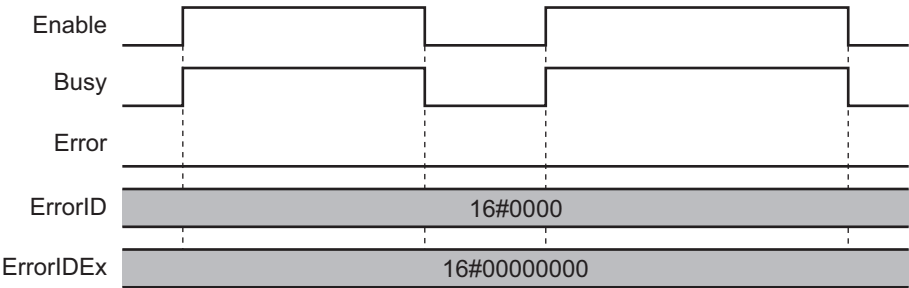
3-1-3 Enable-type Function Blocks

- Processing is executed while *Enable* (*Execute*) is TRUE.
- When *Enable* changes to TRUE, *Busy* (*Executing*) also changes to TRUE.
- If an error occurs in the function block, *Error* (*Error*) changes to TRUE and *Busy* changes to FALSE. When *Enable* changes from TRUE to FALSE, *Busy* and *Error* change to FALSE.
- If an error occurs in the function block, 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 from FALSE to TRUE.

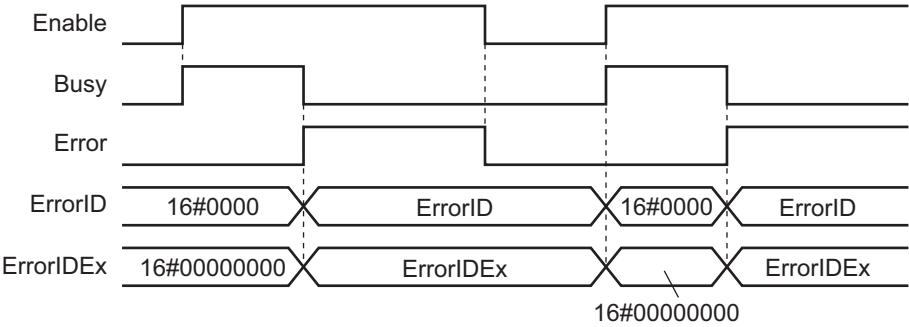
Timing Charts

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

● Normal End



● Errors



3-2 Precautions

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

3-2-1 Nesting

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

Refer to the *NJ/NX-series CPU Unit Software User's Manual (Cat. No. W501)* for details on the nesting function block.

3-2-2 Instruction Options

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

3-2-3 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.

Refer to the *NJ/NX-series CPU Unit Motion Control User's Manual (Cat. No. W507)* for details on re-execution.

4

Individual Specifications of FBs

This section describes the individual specifications of each FB in the MQTT Communications Library.

4

MQTTClient	4-2
MQTTPubAryByte.....	4-24
MQTTPubString	4-34
MQTTSubAryByte.....	4-38
MQTTSubString	4-48
MQTTPing.....	4-52

MQTTClient

This function block controls connection with the MQTT broker to send the messages generated by the instances of MQTTPubString, MQTTPubByte, MQTTSubByte, MQTTSubString, and MQTTPing, and receive the messages from the MQTT broker.

Function block name	Name	FB/FUN	Graphic expression	ST expression
MQTTClient	MQTT client	Function block	<p>MQTTClient_instance</p>	MQTTClient_instance(Enable:=, ClientReference:=, ClientID:=, ConnectionSettings:=, KeepAlive:=, Timeout:=, DiscardMsgTime:=, Connected=>, Busy=>, DiscardMsgNum=>, SessionPresent=>, Error=>, ErrorID=>, ErrorIDEx=>,);

Function Block and Function Information

Item	Description
Library file name	OmronLib_MQTT_Comm_Vx_x.slr (x shows the version.)
Namespace	OmronLib\MQTT_Comm
Function block and function number	00237
Source code published/not published	Not Published

Input Variables

Variable	Meaning	Data type	Description	Valid range	Unit	Default
Enable	Execute	BOOL	TRUE: Executes processing*1. FALSE: Stops processing.	TRUE, FALSE	---	FALSE
ClientID	Client identifier	STRING[256]	This is an identifier for the MQTT broker to identify the client.	---	---	"

Variable	Meaning	Data type	Description	Valid range	Unit	Default
Connec- tionSettings	Connection settings	OmronLib \\MQTT_Co mm\\sCon- nectionSet- tings	Configure communica- tions setup for the MQTT broker.	---	---	Refer to OmronLib \\MQTT_Co mm\\sCon- nectionSet- tings.
KeepAlive	Keep-alive timer	UINT	Set a keep-alive timer for the MQTT broker.	0 to 65535	s	60
Timeout	Timeout time	UINT	This is a timeout time for a request to connect to the MQTT broker.	0 to 20	s	10
Dis- cardMsg- Time	Message discarding time	UINT	Specify the maximum time spent waiting for the received message to be discarded.	0 to 65535	ms	1000

- *1. Even if a command to start processing (change *Enable* from FALSE to TRUE) is given during process to connect (*Busy* is TRUE), connection with the MQTT broker will not be established (*Connected* will not change to TRUE).

Output Variables

Variable	Meaning	Data type	Description	Valid range	Unit	Default
Connected	MQTT broker con- nected	BOOL	TRUE: Connected with MQTT broker FALSE: Unconnected with MQTT broker, or error end	TRUE, FALSE	---	---
Busy	Executing	BOOL	TRUE: Executing FALSE: Not executing	TRUE, FALSE	---	---
Error	Error	BOOL	TRUE: Error end FALSE: Normal end, exe- cution in progress, or exe- cution condition not met	TRUE, FALSE	---	---
ErrorID	Error code	WORD	This is the error ID for an error end. The value is 16#0 for a normal end.	*1	---	---
ErrorIDEx	Expansion error code	DWORD	This is the expansion er- ror ID for an error end. The value is 16#0 for a normal end.	*1	---	---
Dis- cardMsg- Num	Number of discarded messages	UDINT	This is the number of dis- carded message recep- tions.	Depends on data type	---	---
Session- Present	Sustain session	BOOL	TRUE: Sustain session present FALSE: Sustain session absent	TRUE, FALSE	---	---

- *1. Refer to *Troubleshooting* on page 4-11 for details.

Input-Output Variables

Variable	Meaning	Data type	Description	Valid range	Unit	Default
ClientReference	MQTT client variable	OmronLib MQTT_Comm sClientReference	This is data to be shared among function blocks in this library. Do not change the data. The data contents are not published.	---	---	---

Structure

● OmronLib\MQTT_Comm\sConnectionSettings

Connection settings to MQTT broker

Member	Member name	Data type	Valid range	Default	Description
IpAdr	Destination IP address or host name	STRING[201]	Depends on data type	"	This is an IP address or host name of the MQTT broker. If you specify by host name, you need to set the DNS server.
PortNo	Destination port number	UINT	0 to 65535	0	This is a port number of the MQTT broker. When 0 is specified, connection with the MQTT broker is established at 8883 for <i>TLSUse</i> =TRUE or at 1883 for <i>TLSUse</i> =FALSE.
TLSUse	Secure socket communications enable	BOOL	TRUE, FALSE	TRUE	When this flag is TRUE, secure socket communications are used to communicate with the MQTT broker. When this flag is FALSE, socket communications are used to communicate with the MQTT broker.
TLSSession-Name	TLS session name	STRING[17]	Depends on data type	"	This is a set TLS session name. The set TLS session name is specified with Secure Socket Configuration commands. The TLS session names range from TLSSession0 to TLSSession59. It is accessed in the case of <i>TLSUse</i> =TRUE.
UserName	User name	STRING[256]	Depends on data type	"	This is a login user name to the MQTT broker.
Password	Password	STRING[256]	Depends on data type	"	This is a login password to the MQTT broker.
WillCfg	Will configuration	OmronLib MQTT_Comm sWillCfg	---	---	Refer to OmronLib\MQTT_Comm\sWillCfg.
CleanSession	Clean session specification	BOOL	TRUE, FALSE	TRUE	When this flag is TRUE, at the time of disconnection from this library (<i>Enable</i> of the MQTTClient instruction is set to FALSE), the MQTT broker clears the subscription status etc. before the disconnection; at the time of reconnection (<i>Enable</i> of the MQTTClient instruction is set to TRUE), it establishes the connection as a new session.

● OmronLib\MQTT_Comm\sWillCfg

Will settings to MQTT broker

If an error occurs in the MQTT broker during communications, or if this library fails within the time of the keep-alive timer, the MQTT broker will define a message to publish on behalf of this library.

Member	Member name	Data type	Valid range	Default	Description
WillTopic	Will topic	STRING[512]	Depends on data type	"	This is a Will topic name.
WillMsg	Will message	STRING[256]	Depends on data type	"	Set a message to publish to the Will topic.
WillRetain	Will message save setting	BOOL	TRUE, FALSE	FALSE	When this flag is TRUE, after sending out a Will message to other clients, the MQTT broker retains that message.
WillQoS	QoS level of Will message	BYTE	0, 1, 2	0	Set the QoS level when the MQTT broker publishes a Will message in this library to other clients.
WillFlag	Will enable	BOOL	TRUE, FALSE	FALSE	Set this to TRUE when you use the Will function.

Function

When the value of the input variable *Enable* changes from FALSE to TRUE, a request for connection is submitted to the MQTT broker in accordance with the setting specified with the input variable *ConnectionSettings*. Connection with the MQTT broker is established via the socket service instructions and the secure socket service instructions, and a *CONNECT* packet is sent.

After a connection with the MQTT broker is established, this function block sends the messages generated by the instances of MQTTPubString, MQTTPubAryByte, MQTTSUBAryByte, MQTTSUBString, and MQTTPing instructions, and receives the messages from the MQTT broker. The secure socket service instructions or the socket service instructions are used for sending and receiving messages. While the connection is built with the MQTT broker, monitoring and keep-alive functions are performed to check for disconnection from the MQTT broker.

When you use function blocks (MQTTPubAryByte instruction, MQTTPubAryString instruction, MQTTSUBAryByte instruction, MQTTSUBString instruction, and MQTTPing instruction) that request multiple send messages from a single MQTTClient instruction, the priority of message send becomes higher in ascending order of function block execution after the MQTTClient instruction is executed.

If the value of *Enable* changes from TRUE to FALSE while the connection is built with the MQTT broker, a request to disconnect from the MQTT broker will be made.

After the *DISCONNECT* packet is sent, the MQTT broker is disconnected by using the socket service instructions or the secure socket service instructions, and the socket is closed.

If disconnection from the MQTT broker is detected during connection or if a message that violates the protocol is received, the MQTT broker will be disconnected and the function block will come to an error end.

● Client Identifier

This is an ID for the MQTT broker to identify this library. Be sure to specify a character string of one or more characters to a client identifier.

Depending on the setting of the MQTT broker to connect, you may need to register a client beforehand, and you may be able to use only the ID provided at the time of registration. If the value specified with the input variable *ClientID* is identical to the client identifier of another client connected with the MQTT broker, the connection with the other connected client will be closed when connection between this library and the MQTT broker is established.

● Connection Destination Setting

The IP address or host name of the MQTT broker to connect to and the port number to connect to are specified in the members *IpAddr* and *PortNo* of the structure *sConnectionSettings*.

If you specify by host name, you need to set the DNS setting server in advance.

● Secure Socket Communication Enabled

Specify the method for communication with the MQTT broker.

When TRUE is specified for the member *TLSUse* in the structure *sConnectionSettings*, secure socket communications are used.

When FALSE is specified for *TLSUse*, socket communications are used.

● Login Information

If the MQTT broker has specified a user name and password, specify the user name and password. Specify *ConnectionSettings.UserName* as the user name and *ConnectionSettings.Password* as the password.

If the MQTT broker does not specify a user name and password, it is not required.

● Will configuration

Specify the registration setting of a Will message that the MQTT broker issues when the connection is cut by an external factor, for the member *WillCfg* in the structure *sConnectionSettings*.

In registering the Will message, specify TRUE for the member *WillFlag* in the structure *sWillCfg* and, as with usual topic messages, specify a topic and a message for the members *WillTopic* and *WillMsg* in the structure *sWillCfg*.

If you do not register the Will message, specify FALSE for the member *WillFlag* in the structure *sWillCfg*.

If the connection is cut explicitly by the client, the Will message will not be issued even if it is registered.

● Clean Session Specification

Specify whether to save the current session status.

When TRUE is specified for the member *CleanSession* in the structure *sConnectionSettings*, at the time of connection, the previous session status that the client and the MQTT broker retain is cleared and a new session starts. At the time of disconnection, the session status will not be retained.

When FALSE is specified for *CleanSession*, at the time of connection, communications are restarted on the basis of the previous session status that the client and the MQTT broker retain. However, when the MQTT broker does not retain the session status, a new session is started. At the time of disconnection, the session status will be retained.

● Keep-alive Timer

If, during the connection with the MQTT broker, no message is sent from this client within the time specified with the input variable *KeepAlive*, a *PINGREQ* packet will be sent.

After the *PINGREQ* packet is sent, if the *PINGRESP* packet from the MQTT broker cannot be received even when the time which is 1.5 times as long as that specified with *KeepAlive* has elapsed, the connection with the MQTT broker will be cut.

If 0 is specified for *KeepAlive*, the *PINGREQ* packet will not be sent.

Some MQTT brokers do not support disabled keep-alive (*KeepAlive*=0).

If you specify 0 for *KeepAlive* and connect to such an MQTT broker, the connection may be cut by the MQTT broker.

The range of keep-alive interval values that can be specified varies with the MQTT broker.

If you request a value outside the range that the MQTT broker accepts, the connection may be cut by the MQTT broker or a *PINGRESP* packet reception time error may occur.

Make sure that *KeepAlive* has a shorter time than *KeepAlive Monitoring Time* of *KeepAlive* in **TCP/IP Settings** of the Controller.

For the range of values that can be specified with *KeepAlive*, check the specifications of the MQTT broker to connect.

● Timeout

If the connection request processing failed to be completed within the time specified with the input variable *Timeout*, this will be judged as a timeout and the output variable *Error* will change to TRUE.

When 0 is specified, a timeout will occur in 10 seconds, which is a default.

● Message Discarding Time

The MQTTClient instruction receives the message.

The MQTTPubAryByte instruction, MQTTPubString instruction, MQTTSubAryByte instruction, MQTTSubString instruction, and MQTTPing instruction refer to the data received by the MQTTClient instruction and confirm that it is the response for the transmission requested by each.

When the MQTTPubAryByte instruction, MQTTPubString instruction, MQTTSubAryByte instruction, MQTTSubString instruction, and MQTTPing instruction confirm that the response is for the transmission requested by them, the MQTTClient instruction is notified that the reception has been confirmed. If the MQTTClient instruction does not receive the reception notification from the MQTTPubAryByte instruction, MQTTPubString instruction, MQTTSubAryByte instruction, MQTTSubString instruction, and MQTTPing instruction within the time specified by the input variable *DiscardMsgTime* after receiving the message, it discards the received message and starts receiving the next message.

Execute the MQTTPubAryByte instruction, MQTTPubString instruction, MQTTSubAryByte instruction, MQTTSubString instruction, and MQTTPing instruction periodically at intervals shorter than the time specified by *DiscardMsgTime*.

When 0 is specified for *DiscardMsgTime*, the message will be discarded in 1000 ms, which is a default.

If any message over 65535 bytes is received, the message will be discarded without waiting for the acknowledgement of reception.

● Number of Messages Discarded

If, after messages are received, there are any messages discarded because the acknowledgement of reception made by MQTTPubAryByte instruction, MQTTPubString instruction, MQTTSubAryByte

instruction, MQTTSubString instruction, and MQTTPing instruction cannot be detected within the time specified with the input variable *DiscardMsgTime*, and any messages over 65535 bytes, the number of discarded messages will be output to the output variable *DiscardMsgNum*.

The number of discarded messages is cleared when the input variable *Enable* changes to TRUE, and refreshed during execution.

● Setting Changes During Execution

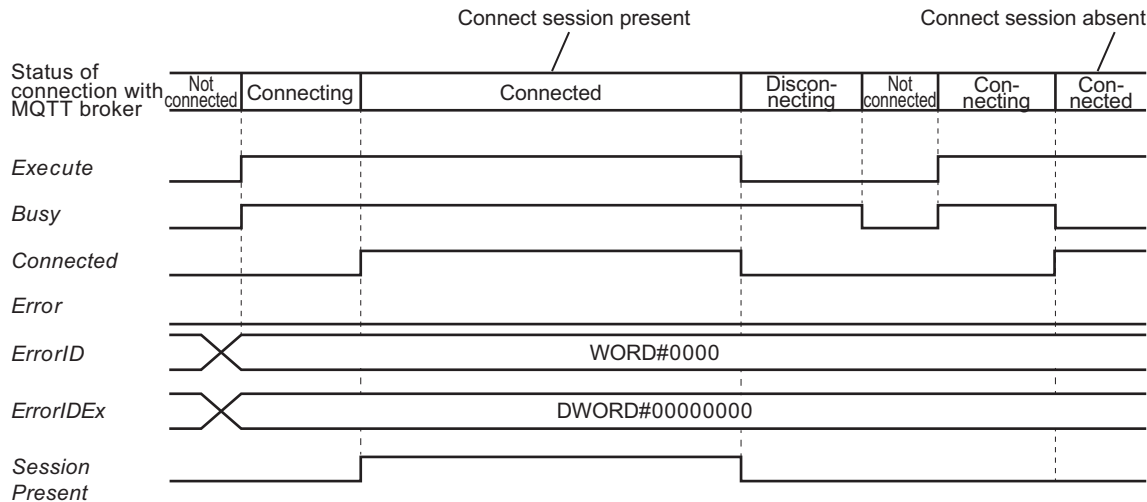
Even if the input variable *ClientID*, input variable *ConnectionSettings*, input variable *KeepAlive*, input variable *Timeout*, or input variable *DiscardMsgTime* is changed during execution, the changed value will not be reflected.

The setting when the input variable *Enable* has changed to TRUE is used in execution.

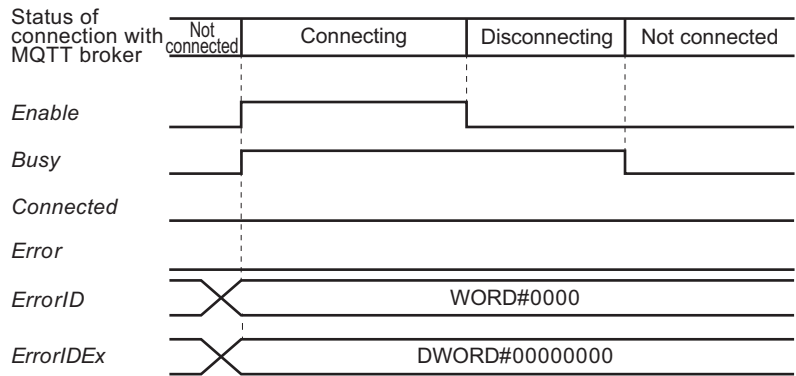
Timing Charts

The timing charts are shown below.

- When *Enable* (Execute) changes to TRUE while *Busy* (Executing) is FALSE, *Busy* (Executing) changes to TRUE.
- Then, when connection with the MQTT broker is established, *Connected* (MQTT broker connected) changes to TRUE. Even after the connection with the MQTT broker is established, *Busy* (Executing) remains TRUE. If the connection with the MQTT broker is established with sustain session present, *SessionPresent* (Sustain session) will change to TRUE.
- If *Enable* (Execute) is changed to FALSE while *Connected* (MQTT broker connected) is TRUE, processing to disconnect from the MQTT broker will be started, and *Connected* (MQTT broker connected) will change to FALSE. When the processing to disconnect from the MQTT broker is finished, *Busy* (Executing) changes to FALSE.
- Even if *Enable* (Execute) changes to FALSE while *Connected* (MQTT broker connected) is FALSE and *Busy* (Executing) is TRUE, *Busy* (Executing) will not change to FALSE immediately but remain TRUE. When the processing to disconnect from the MQTT broker is finished, *Busy* (Executing) changes to FALSE.
- If an error occurs while *Connected* (MQTT broker connected) is TRUE, *Connected* (MQTT broker connected) changes to FALSE, and after the processing to disconnect from the MQTT broker is finished, *Error* (Error) changes to TRUE.
- If an error occurs during the execution of this function block, *Error* (Error) changes to TRUE and *Busy* (Executing) changes to FALSE when the disconnection process with the MQTT broker is finished. You can find out the cause of the error by accessing the values output to *ErrorID* (Error code) and *ErrorIDEx* (Expansion error code).
- Timing Chart for Normal End
 - a) When execution is stopped after the connection with the MQTT broker

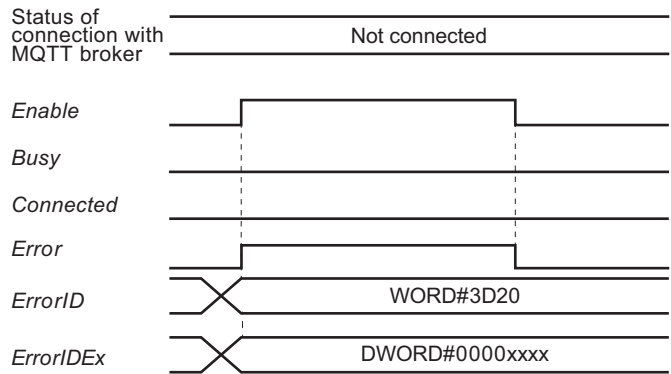


b) When execution is stopped during connection processing with the MQTT broker

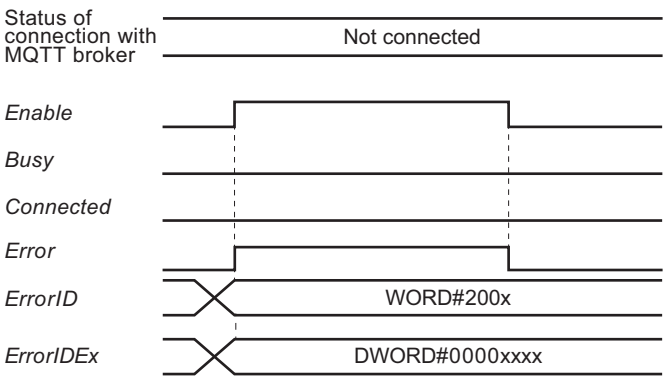


• Timing Chart for Error End

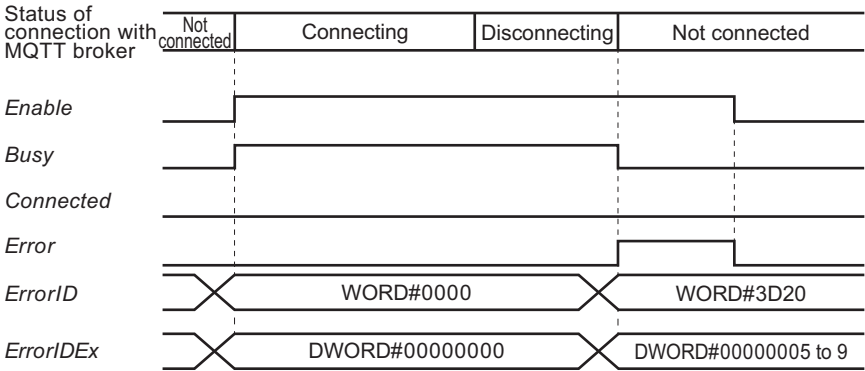
a) When there is an error in input parameters



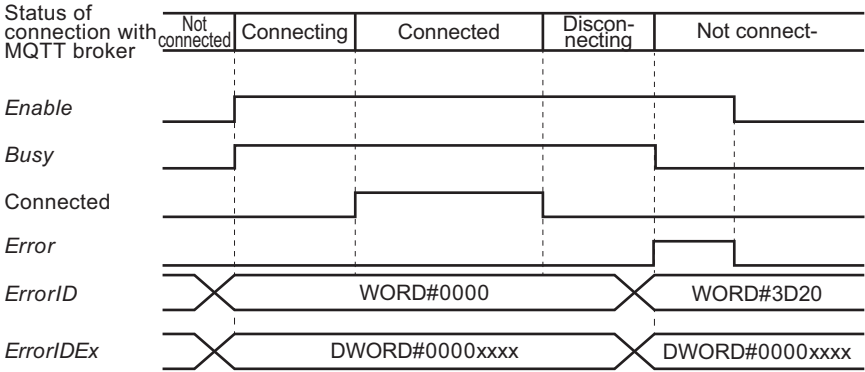
b) When communications cannot be started because of the EtherNet/IP port setting and status



c) When connection is rejected because of the verification of authority for the MQTT broker or other reasons



d) When the connection with the MQTT broker is cut by a line error that occurred during connection



Additional Information

- For this FB, use the socket service function. Refer to the *NJ/NX-series CPU Unit Built-in EtherNet/IP™ Port User's Manual (Cat. No. W506)* for details of the socket service function.
- One socket is used per use of one instance of this function block.

Precautions for Correct Use

- For this function block, even if the value of *Enable* changes to FALSE, the processing of the function block will not stop immediately. The value of *Busy* changes to FALSE when processing has stopped. Use this to confirm stop of processing.

- At the start of execution, unless the processing is in the stopped state, *Connected* will not change to TRUE. In starting execution, be sure to set *Enable* to TRUE after checking that *Busy* is FALSE.
- This function block, inside the object, uses the SktTCPConnect instruction, SktGetTCPStatus instruction, SktClose instruction, SktTLSConnect instruction, SktTLSClearBuf instruction, SktTLSWrite instruction, SktTLSRead instruction, and SktTLSDisconnect instruction for each instance, and may be executing a maximum of three instances simultaneously. For the number of simultaneous executions, refer to the *NJ/NX-series Instructions Reference Manual (Cat. No. W502)*.
- If the number of discarded messages *DiscardMsgNum* is refreshed when the execution status of the MQTTSubAryByte instruction or the MQTTSubString instruction is in the subscribed state, the message discarding time *DiscardMsgTime* may be too small as compared with the task period of a program that executes the MQTTClient instruction, or the task period of a program that executes the MQTTSubAryByte instruction or the MQTTSubString instruction may be too large as compared with the task period of a program that executes the MQTTClient instruction. Check and correct the value of the message discarding time or the task period of the program.
- The maximum size of one message that this function block can receive is 65535 bytes. Any message over 65535 bytes will be discarded.

Troubleshooting

Error code	Expansion error code	Status	Description	Corrective action
16#0000	16#00000000	Normal End	---	---
16#0400	16#00000000	Input Out of Range	<i>ConnectionSettings.IpAdr</i> is not set correctly.	Set the IP address or host name using 1 to 200 characters.
16#2000	16#00000000	Local IP Address Setting Error	The instruction was executed when there was a setting error in the local IP address.	Make sure that the local IP address is set correctly.
16#2002	16#00000000	Address Resolution Failed	Failed to resolve the address of the MQTT broker specified in <i>ConnectionSettings.IpAdr</i> .	Make sure it is the IP address or host name specified by the MQTT broker.
16#2003	16#00000000	Socket Status Error	An error occurred during socket communications with the MQTT broker.	<ul style="list-style-type: none"> • Make sure the Client ID is not duplicated with another MQTT client. • Make sure that the values of the following input variables match the MQTT broker specifications. <ol style="list-style-type: none"> a) <i>ConnectionSettings.IpAdr</i> b) <i>ConnectionSettings.PortNo</i> c) <i>ConnectionSettings.TLSUse</i> • Make sure that the input variable <i>KeepAlive</i> is less than the KeepAlive monitoring time in the built-in EtherNet/IP port setting.

Error code	Expansion error code	Status	Description	Corrective action
				<ul style="list-style-type: none"> Make sure that the topic name and QoS level of the MQTTClient, MQTTPubAry-Byte, MQTTTPubString, MQTTSubAry-Byte, and MQTTSubString instructions are those specified or supported by the MQTT broker. Check the MQTT broker settings. Check if there is a problem with the communication path such as FireWall or Gateway.
16#2004	16#00000000	Local IP Address Not Set	The instruction was executed when there was a setting error in the local IP address.	Make sure that the local IP address is set correctly.
16#2006	16#00000000	Socket Timeout	In socket service instruction, a timeout occurred.	<ul style="list-style-type: none"> Make sure that the ClientReference variable is not specified more than once in multiple MQTTClient instructions. Make sure that the values of the following input variables match the MQTT broker specifications. <ol style="list-style-type: none"> ConnectionSettings.IpAdr ConnectionSettings.PortNo ConnectionSettings.TLSUse Make sure that the input variable <i>KeepAlive</i> is less than the KeepAlive monitoring time in the built-in EtherNet/IP port setting. Check the MQTT broker settings. Check if there is a problem with the communication path such as FireWall or Gateway.
16#2007	16#00000000	Socket Handle Out of Range	The socket handle used in the MQTTClient instruction has been closed.	Execute the SktClose instruction with 0 set in the <i>Socket.Handle</i> and check if all sockets are not closed.
16#2008	16#00000000	Socket Communications Resource Overflow	Instructions were executed in excess of the socket service instruction resources that can be simultaneously executed.	<p>Make sure that the number of MQTTClient instructions and socket service instructions that are executed at the same time does not exceed the upper limit.</p> <p>The MQTTClient instruction uses up to three socket service instructions and one socket handle.</p>
16#200A	16#00000000	Illegal TLS Session Name	The TLS session name specified with <i>ConnectionSettings.TLSSessionName</i> is not in the secure socket setting.	Make sure that the TLS session name corresponding to the TLS session ID set by the Secure Socket Configuration commands is specified.

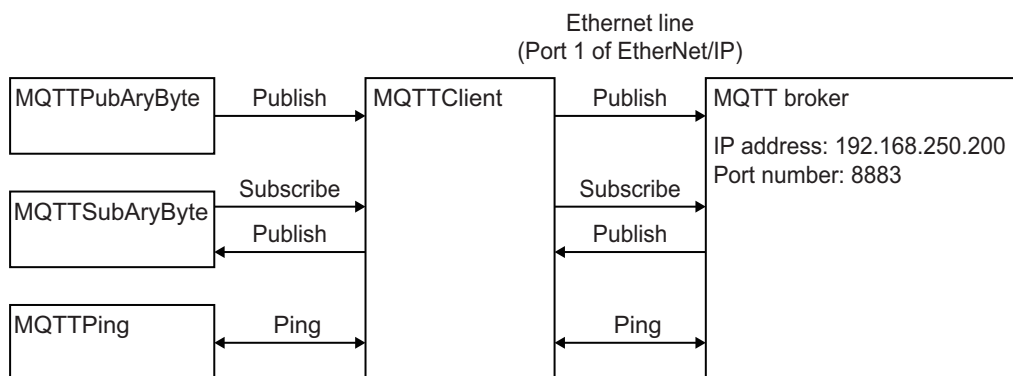
Error code	Expansion error code	Status	Description	Corrective action
16#200B	16#00000000	Certificate Access Failed	<ul style="list-style-type: none"> The TLS session certificate and secure socket setting specified in <i>ConnectionSettings.TLSSessionName</i> have not been transferred to the CPU Unit. A password is set to the client certificate, so the certificate could not be accessed. The secure socket setting does not exist, or the content of the secure socket setting is incorrect. 	Check the TLS session certificate and secure socket setting.
16#200C	16#00000000	TLS Session Establishment Error	TLS session establishment failed.	<p>Make sure that the values of the following input variables match the MQTT broker specifications.</p> <ul style="list-style-type: none"> <i>ConnectionSettings.PortNo</i> <i>ConnectionSettings.TLSUse</i>
16#200E	16#00000000	Invalid TLS Session Handle	The TLS session handle used in the MQTTClient instruction has been closed.	Execute the <i>SktClose</i> instruction with 0 set in the <i>Socket.Handle</i> and check if all sockets are not closed.
16#200F	16#00000000	Error in TLS	An error occurred during secure socket communications with the MQTT broker.	<ul style="list-style-type: none"> Make sure the Client ID is not duplicated with another MQTT client. Make sure that the ClientReference variable is not specified more than once in multiple MQTTClient instructions. Make sure that you have set the correct client private key and client certificate using the Secure Socket Configuration commands. Make sure that the topic name and QoS level of the MQTTClient, MQTTPubAryByte, MQTTPubString, MQTTSubAryByte, and MQTTSubString instructions are those specified or supported by the MQTT broker. If the MQTT broker requires certificate activation, make sure it is activated. Check the MQTT broker settings.
16#3D20	16#00000001	ClientID Input Value Out of Range	The input variable <i>ClientID</i> is not set correctly.	Set it using 1 to 255 characters.

Error code	Expansion error code	Status	Description	Corrective action
	16#00000002	Illegal Will Setting-related Value	When <i>WillCfg.WillFlag</i> is TRUE (Will function enabled), <i>WillCfg.WillTopic</i> is not set correctly.	When <i>WillCfg.WillFlag</i> is TRUE (Will function enabled), set <i>WillCfg.WillTopic</i> with 1 to 511 characters.
	16#00000003	WillQoS Input Value Out of Range	<i>WillCfg.WillQoS</i> is outside the valid range.	Make sure that it is in the valid range.
	16#00000004	Timeout Input Value Out of Range	The value of the input variable <i>Timeout</i> is outside the valid range.	Make sure that it is in the valid range.
	16#00000005	MQTT Broker Connection Rejected: Protocol Version Not Allowable	The MQTT broker does not support the MQTT protocol with a version 3.1.1.	<ul style="list-style-type: none"> Make sure that the MQTT broker supports version 3.1.1 of the MQTT protocol. Check the MQTT broker settings.
	16#00000006	MQTT Broker Connection Rejected: Identifier Rejected	The value of the input variable <i>ClientID</i> is a value that the MQTT broker does not support.	<ul style="list-style-type: none"> Make sure the value is supported by the MQTT broker. Check the MQTT broker settings.
	16#00000007	MQTT Broker Connection Rejected: MQTT Service Unavailable	The MQTT service of the MQTT broker cannot be used.	Check the MQTT broker settings.
	16#00000008	MQTT Broker Connection Rejected: Illegal User Name or Password	<i>ConnectionSettings.UserName</i> and <i>ConnectionSettings.Password</i> are invalid.	<ul style="list-style-type: none"> Make sure the user name and password specified by the MQTT broker are set. Check the MQTT broker settings.
	16#00000009	MQTT Broker Connection Rejected: No Authority	This client does not have the authority to make a connection.	<ul style="list-style-type: none"> Make sure the user name and password specified by the MQTT broker are set. Make sure that the client certificate or client private key has not expired. Check the MQTT broker settings.
	16#0000000A	Undefined Packet Received from MQTT Broker	An undefined packet was sent from the MQTT broker.	Check the MQTT broker settings.
	16#0000000B	Connection Processing Timeout	Connection with the MQTT broker failed to be established within the time set in the input variable <i>Timeout</i> .	<ul style="list-style-type: none"> Make sure that the <i>ClientReference</i> variable is not specified more than once in multiple <i>MQTTClient</i> instructions. Make sure that the values of the following input variables match the MQTT broker specifications. <ol style="list-style-type: none"> <i>ConnectionSettings.IpAdr</i> <i>ConnectionSettings.PortNo</i> <i>ConnectionSettings.TLSUse</i>

Error code	Expansion error code	Status	Description	Corrective action
				<ul style="list-style-type: none"> Make sure that the input variable <i>KeepAlive</i> is less than the KeepAlive monitoring time in the built-in EtherNet/IP port setting. Increase the time specified by <i>Timeout</i>. Check the MQTT broker settings. Check if there is a problem with the communication path such as FireWall or Gate-way.
	16#0000000C	Protocol Error	A packet that violates the protocol was received.	Check the MQTT broker settings.
	16#0000000D	PINGRESP Packet Reception Timeout	PINGRESP packet failed to be received within the time which is 1.5 times as long as that specified with the input variable <i>KeepAlive</i> .	<ul style="list-style-type: none"> Make sure that the topic name and QoS level of the MQTTClient, MQTTPubAryByte, MQTTTpubString, MQTTSubAryByte, and MQTTSubString instructions are those specified or supported by the MQTT broker. Check the MQTT broker settings. Check if there is a problem with the communication path.

Sample Programming

A Publisher sends a PUBLISH message that belongs to the topic of 'Tank/Temperature' to the MQTT broker, and a Subscriber specifies a topic filter of 'Tank/Temperature' and subscribes to the MQTT broker. The MQTT broker uses the MQTT broker on the local network to communicate using secure socket communications. The Publisher and the Subscriber use the port 1 of the built-in EtherNet/IP.



User Program Processing Procedure

The procedure for processing is as follows.

- 1 Use the MQTTClient instruction to establish connection with the MQTT broker.
- 2 While the connection with the MQTT broker is established, use the MQTTPubAryByte and MQTTPing instructions to send a PUBLISH message and check the response time.

- Sending a PUBLISH message
Execute the MQTTPubAryByte instruction to send a PUBLISH message. The send message is contained in pubMsg[], and the data size is specified with *MsgSize*.
 - Checking a response time
Execute the MQTTPing instruction to check a response time. The execution starts while the MQTTPubAryByte instruction is not being executed, so that a response time will be checked more accurately.
In this sample programming, if the response time exceeds 500 ms, it will be registered in the user information.
- 3** Start execution of the MQTTSubAryByte instruction to activate subscription.
Messages obtained during subscription are stored in rcvMsg[].
Before starting execution, make sure that connection with the MQTT broker is established.
 - 4** Stop execution of the MQTTSubAryByte instruction to stop subscription.
Before stopping execution, make sure that connection with the MQTT broker is established.
 - 5** Use the MQTTClient instruction to disconnect from the MQTT broker.



Precautions for Correct Use

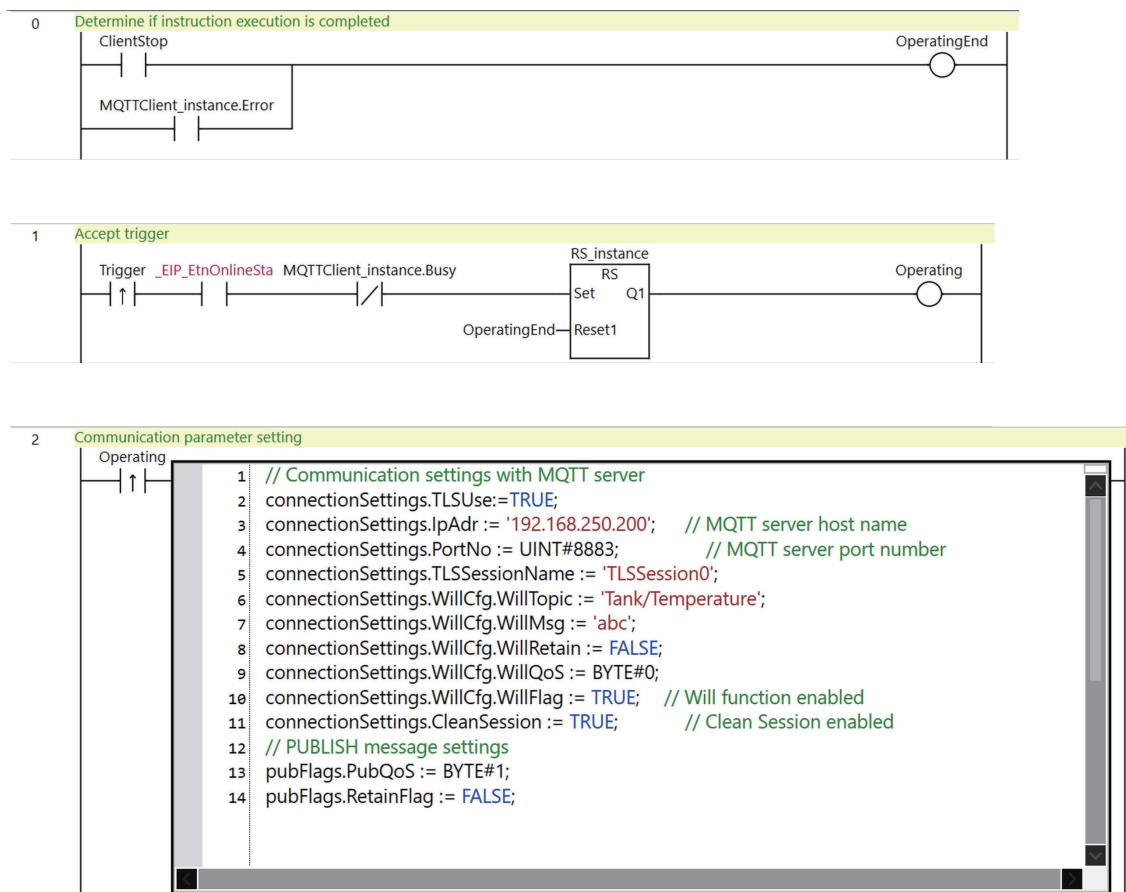
- To use a host name to specify the MQTT broker, set the DNS settings in the Controller's built-in EtherNet/IP port setting in advance.
- Use the Secure Socket Configuration commands in advance to set the secure socket setting to the CPU Unit. For details on Secure Socket Configuration commands, refer to *NJ/NX-series CPU Unit Built-in EtherNet/IP™ Port User's Manual (Cat. No. W506)*.
This sample program uses session ID 0 set in the secure socket setting.
- If the input variable *ClientID* specified in the MQTTClient instruction conflicts with other clients, the connection may be disconnected from the MQTT broker at an unexpected timing. Make sure that the *ClientID* is unique.

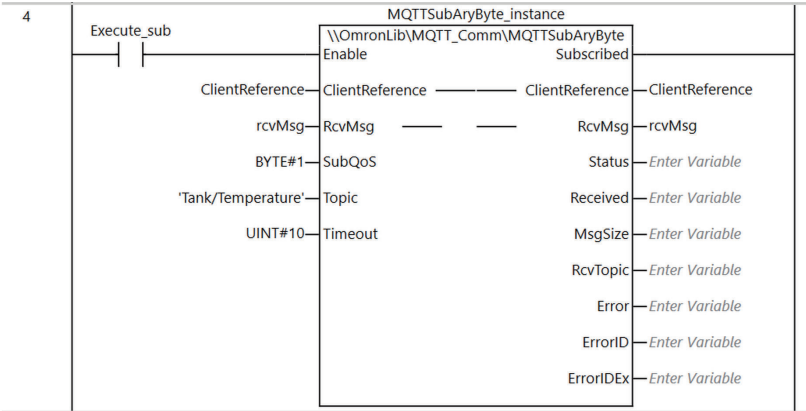
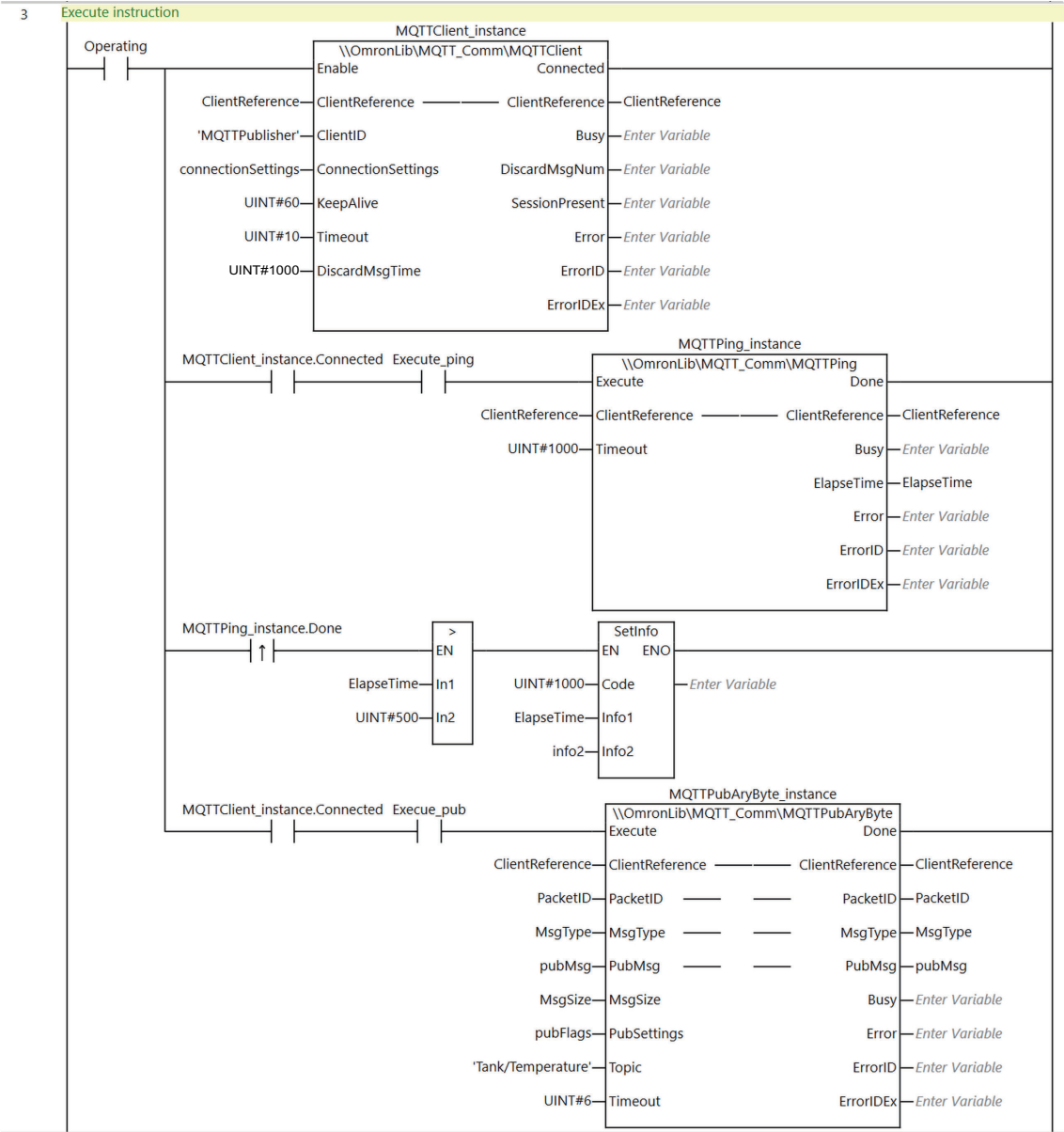
LD

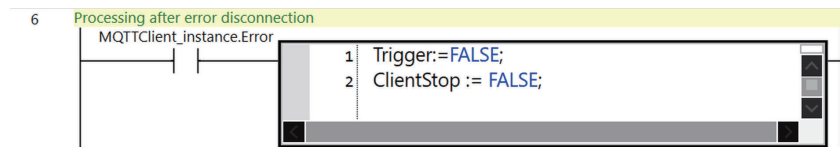
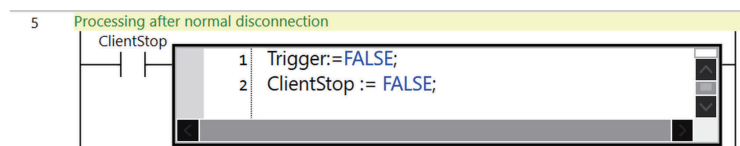
Internal variable	Name	Data type	Default	Comment
	Trigger	BOOL	FALSE	Execution condition
	Operating	BOOL	FALSE	Processing in progress
	OperatingEnd	BOOL	FALSE	Processing complete
	ClientStop	BOOL	FALSE	Stop command for MQTTClient instruction
	Execute_ping	BOOL	FALSE	Execution condition for MQTTPing instruction
	Execute_pub	BOOL	FALSE	Execution condition for MQTTPubAryByte instruction
	Execute_sub	BOOL	FALSE	Execution condition for MQTTSubAryByte instruction

Internal variable	Name	Data type	Default	Comment
	PubMsg	ARRAY[0..1999] OF BYTE	[2000(16#0)]	Message part of PUBLISH packet
	MsgSize	UINT	0	Data size of pubMsg
	PacketID	UINT	0	---
	MsgType	USINT	0	---
	RcvMsg	ARRAY[0..1999] OF BYTE	[2000(16#0)]	Message part of PUBLISH message received during subscription
	ElapsedTime	UINT	0	Ping response time
	Info2	UINT	0	---
	ConnectionSettings	OmronLib \\MQTT_Comm\\sConnectionSettings	(IpAdr :=", PortNo :=0, TLSUse :=False, TLSSessionName :=", UserName :=", Password :=", WillCfg := (WillTopic :=", WillMsg :=", WillRetain :=False, WillQoS :=16#0, WillFlag :=False), CleanSession :=False)	Communication settings with MQTT broker
	PubFlags	OmronLib \\MQTT_Comm\\sPubFlags	(PubQoS :=16#0, RetainFlags :=False)	Publication setup of PUBLISH message
	ClientReference	OmronLib \\MQTT_Comm\\sClientReference	---	Data to be shared among function blocks in this library
	MQTTClient_instance	OmronLib \\MQTT_Comm \\MQTTClient	---	---
	MQTTPing_Instance	OmronLib \\MQTT_Comm \\MQTTPing	---	---
	MQTTPubAryByte_instance	OmronLib \\MQTT_Comm \\MQTTPubAryByte	---	---
	MQTTSubAryByte_instance	OmronLib \\MQTT_Comm \\MQTTSubAryByte	---	---
	RS_instance	RS	---	---

External variable	Name	Data type	Constant	Comment
	_EIP_EtnOnlineSta	BOOL	<input checked="" type="checkbox"/>	---








ST

Internal variable	Name	Data type	Default	Comment
	Trigger	BOOL	FALSE	Execution condition
	DoMQTT	BOOL	FALSE	Processing in progress
	ClientStop	BOOL	FALSE	Stop command for MQTTClient instruction
	Stage	INT	0	State transition
	Enable_client	BOOL	FALSE	---
	Execute_pub	BOOL	FALSE	---
	Execute_ping	BOOL	FALSE	---
	Enable_subscribe	BOOL	FALSE	---
	PubMsg	ARRAY[0..1999] OF BYTE	[2000(16#0)]	Message part of <i>PUBLISH</i> packet
	MsgSize	UINT	0	Data size of PubMsg
	PacketID	UINT	0	---
	MsgType	USINT	0	---
	ConnectionSettings	OmronLib \\MQTT_Comm\\sCon- nectionSettings	(IpAdr :=", PortNo :=0, TLSUse := False, TLSSessionName :=", UserName :=", Pass- word :=", WillCfg := (WillTopic :=", WillMsg :=", WillRe- tain :=False, Will- QoS :=16#0, Will- Flag :=False), Cleant- Session :=False)	Communication set- tings with MQTT broker
	PubFlags	OmronLib \\MQTT_Comm\\sPub- Flags	(PubQoS :=16#0, Re- tainFlags :=False)	Publication setup of PUBLISH message
	ElapseTime	UINT	0	Ping response time
	PingDone	BOOL	FALSE	Execution of MQTTPing complete
	Info2	UINT	0	---

Internal variable	Name	Data type	Default	Comment
	RcvMsg	ARRAY[0..1999] OF BYTE	[2000(16#0)]	Message part of PUBLISH message received during subscription
	ClientReference	OmronLib \\MQTT_Comm\\sClientReference	---	Data to be shared among function blocks in this library
	MQTTClient_instance	OmronLib \\MQTT_Comm \\MQTTClient	---	---
	MQTTPubAryByte_instance	OmronLib \\MQTT_Comm \\MQTTPubAryByte	---	---
	MQTTPing_instance	OmronLib \\MQTT_Comm \\MQTTPing	---	---
	MQTTSubAryByte_instance	OmronLib \\MQTT_Comm \\MQTTSubAryByte	---	---
	R_TRIG_pingDone	R_TRIG	---	---

External variable	Name	Data type	Constant	Comment
	_EIP_EtnOnlineSta	BOOL		---

```
// Preparing for processing
IF (Trigger = TRUE) AND (DoMQTT = FALSE) AND (_EIP_EtnOnlineSta = TRUE) THEN
  DoMQTT := TRUE;
  Stage := INT#1;
  // Initialize instance
  MQTTClient_instance(Enable := FALSE, ClientReference := ClientReference);
  // Initialize instance
  MQTTPing_instance(Execute:=FALSE, ClientReference := ClientReference);
  // Initialize instance
  MQTTPubAryByte_instance(
    Execute := FALSE, ClientReference := ClientReference, PacketID:=PacketID, M
    sgType:=MsgType, PubMsg:=pubMsg);
  // Communication settings with MQTT broker
  connectionSettings.TLSUse:=TRUE;
  connectionSettings.IpAdr := '192.168.250.200';           // MQTT broker host name
  connectionSettings.PortNo := UINT#8883;                 // MQTT broker port numbe
r
  connectionSettings.TLSSessionName := 'TLSSession0';
  connectionSettings.WillCfg.WillTopic := 'Tank/Temperature';
  connectionSettings.WillCfg.WillMsg := 'abc';
  connectionSettings.WillCfg.WillRetain := TRUE;
  connectionSettings.WillCfg.WillQoS := BYTE#0;
```

```

        connectionSettings.WillCfg.WillFlag := FALSE;           // Will function enabled
        connectionSettings.CleanSession := TRUE;               // Clean Session enabled
        // PUBLISH message settings
        pubFlags.PubQoS := BYTE#1;
        pubFlags.RetainFlag := FALSE;
    END_IF;

    IF (DoMQTT = TRUE) THEN
        CASE Stage OF
            1 :           // Start connection request
                Enable_client := TRUE;
                Stage := INT#2;

            2:           // MQTTClient running
                IF ClientStop = TRUE THEN
                    Enable_client := FALSE;    // Stop MQTTClient
                    Stage := INT#3;
                ELSIF (MQTTClient_instance.Error = TRUE) THEN
                    Stage := INT#20;           // Error occurred
                END_IF;

            3:           // Waiting for connection disconnection with MQTT broker
                IF MQTTClient_instance.Busy = FALSE THEN
                    DoMQTT := FALSE;
                    Trigger := FALSE;
                    ClientStop := FALSE;
                    Stage := INT#10;           // Normal end
                ELSIF (MQTTClient_instance.Error = TRUE) THEN
                    Stage := INT#20;           // Error occurred
                END_IF;

            20:          // Error End
                Enable_client := FALSE;
                // Wait for all FB to stop running
                IF MQTTClient_instance.Busy = FALSE THEN
                    DoMQTT := FALSE;
                    Trigger := FALSE;
                    ClientStop := FALSE;
                END_IF;

        END_CASE;

    END_IF;

    MQTTClient_instance(
        Enable           := Enable_client,
        ClientReference   := ClientReference,

```

```

ClientID          := 'MQTTPublisher',
ConnectionSettings := connectionSettings,
KeepAlive         := UINT#60,
Timeout          := UINT#10,
DiscardMsgTime    := UINT#1000);

MQTTPing_instance(
    Execute          := MQTTClient_instance.Connected AND Execute_ping,
    ClientReference  := ClientReference,
    Timeout          := UINT#1000,
    ElapseTime       => elapseTime);

R_TRIG_pingDone(MQTTPubAryByte_instance.Done, pingDone);
IF MQTTClient_instance.Connected AND pingDone AND elapseTime > UINT#500
    THEN
        SetInfo(UINT#1000, elapseTime, info2);
END_IF;

MQTTPubAryByte_instance(
    Execute          := MQTTClient_instance.Connected AND Execute_pub,
    ClientReference  := ClientReference,
    PacketID         := PacketID,
    MsgType          := MsgType,
    PubMsg           := pubMsg,
    MsgSize          := MsgSize,
    PubSettings      := pubFlags,
    Topic            := 'Tank/Temperature',
    Timeout          := UINT#6);

MQTTSubAryByte_instance(
    Enable           := Enable_subscribe,
    ClientReference  := ClientReference,
    rcvMsg           := rcvMsg,
    SubQoS           := BYTE#1,
    Topic            := 'Tank/Temperature',
    Timeout          := UINT#10);

```

MQTTPubAryByte

This function block sends a PUBLISH message of the message specified in BYTE array to the MQTT broker via MQTTClient instance.

Function block name	Name	FB/ FUN	Graphic expression	ST expression
MQTTPubAryByte	Data publication in MQTT byte array	Function block	<div>MQTTPubAryByte_instance<div><div>\\OmronLib\MQTT_Comm\MQTPubAryByte</div><div><div>Execute</div><div>Done</div><div>ClientReference</div><div>ClientReference</div><div>PacketID</div><div>PacketID</div><div>MsgType</div><div>MsgType</div><div>PubMsg</div><div>PubMsg</div><div>MsgSize</div><div>Busy</div><div>PubSettings</div><div>Error</div><div>Topic</div><div>ErrorID</div><div>Timeout</div><div>ErrorIDEx</div></div></div></div>	MQTTPubAry-Byte_instance(Execute:=, ClientReference:=, PacketID:=, MsgType:=, PubMsg:=, MsgSize:=, PubSettings:=, Topic:=, Timeout:=, Done=>, Busy=>, Error=>, ErrorID=>, ErrorIDEx=>,);

Function Block and Function Information

Item	Description
Library file name	OmronLib_MQTT_Comm_Vx_x.slr (x shows the version.)
Namespace	OmronLib\MQTT_Comm
Function block and function number	00238
Source code published/not published	Not Published

Input Variables

Variable	Meaning	Data type	Description	Valid range	Unit	Default
Execute	Execute	BOOL	TRUE: Executes processing when this flag changes to TRUE. FALSE: Do not execute	TRUE, FALSE	---	FALSE

Variable	Meaning	Data type	Description	Valid range	Unit	Default
Topic	Topic	STRING[512]	Specify a topic name to publish.	Depends on data type	---	"
PubSettings	Publication setup	OmronLib \MQTT_Comm\PubFlags	Configure the QoS and retain settings in publication.	---	---	Refer to OmronLib \MQTT_Comm\PubFlags.
MsgSize	Message size	UINT	Specify the size of the message stored in PubMsg[].	0 to 65000	Bytes	0
Timeout	Timeout time	UINT	This is a timeout time. It is accessed when the QoS level is 1 or 2.	1 to 1000	s	6

Output Variables

Variable	Meaning	Data type	Description	Valid range	Unit	Default
Done	Done	BOOL	TRUE: Normal end FALSE: Error end, execution in progress, or execution condition not met	TRUE, FALSE	---	---
Busy	Executing	BOOL	TRUE: Executing FALSE: Not executing	TRUE, FALSE	---	---
Error	Error	BOOL	TRUE: Error end FALSE: Normal end, execution in progress, or execution condition not met	TRUE, FALSE	---	---
ErrorID	Error code	WORD	This is the error ID for an error end. The value is 16#0 for a normal end.	*1	---	---
ErrorIDEx	Expansion error code	DWORD	This is the expansion error ID for an error end. The value is 16#0 for a normal end.	*1	---	---

*1. Refer to *Troubleshooting* on page 4-32 for details.

Input-Output Variables

Variable	Meaning	Data type	Description	Valid range	Unit	Default
ClientReference	MQTT client variable	OmronLib \MQTT_ClientReference	This is data to be shared among function blocks in this library. Do not change the data. The data contents are not published.	Depends on data type	---	---
PubMsg ^{*1}	Message	ARRAY[*] OF BYTE	This is a message to be published to the specified topic.	Depends on data type	---	---

Variable	Meaning	Data type	Description	Valid range	Unit	Default
PacketID	Packet ID	UINT	This is a packet ID. If any number other than 0 is specified at the start of execution, it will be sent as the resend message with the specified packet ID. The packet ID used in publication is output. It is accessed when the QoS level is 1 or 2.	Depends on data type	---	---
MsgType	Message type	USINT	Send message type 0: <i>PUBLISH</i> 1: <i>PUBREL</i> It is accessed when the QoS level is 1 or 2 and <i>PacketID</i> is not 0.	0, 1	---	---

*1. The maximum number of array elements is 65000. In addition, subscripts of the array should start with 0.

Structure

OmronLib\MQTT_Comm\sPubFlags

Member	Member name	Data type	Valid range	Default	Description
PubQoS	QoS level in publication	BYTE	0, 1, 2	0	Set the QoS level when a message is published.
RetainFlag	Message retain setting	BOOL	TRUE, FALSE	FALSE	When this flag is TRUE, after sending out a message to other clients, the MQTT broker retains that message.

Function

In the case where the member *PubQoS* of the input variable *PubSettings* is 0, or in the case where the member *PubQoS* of the input variable *PubSettings* is 1 or 2 and the in-out variable *PacketID* is 0, when the input variable *Execute* changes to TRUE during the connection with the MQTT broker, a topic name specified for the input variable *Topic* and a *PUBLISH* packet of the message specified for the in-out variable *PubMsg* will be created in accordance with the setting specified with the input variable *PubSettings*.

A request to send the created *PUBLISH* packet is submitted to the MQTTClient instance. Message exchange is carried out according to the QoS level for publication. When sending messages is completed, the function block comes to a normal end.

When a *PUBLISH* packet is generated, *PacketID* changes to 0 if *PubSettings.PubQoS* is 0.

When *PubSettings.PubQoS* is 1 or 2, the packet ID used when a packet was generated is output.

In the case of a normal end, *MsgType* changes to 0.

If a timeout error occurs with *PubSettings.PubQoS* being 1 or 2, or in the case of an error end due to send failed, *MsgType* will output the type of a message to which an acknowledgement from the MQTT broker failed to be received.

In the case where the member *PubQoS* of the input variable *PubSettings* is 1 or 2 and the in-out variable *PacketID* is not 0, when the input variable *Execute* changes to TRUE during the connection with the MQTT broker, a message specified with *MsgType* will be created as the resend message (DUP=1 message) with the specified packet ID. In the case where 0 is specified for *MsgType*, a topic name specified for *Topic* and a *PUBLISH* packet of the message specified for *PubMsg* will be created in accordance with the setting specified with *PubSettings*. A request to send the created packet is submitted to the MQTTClient instance. Message exchange is carried out according to the QoS level; when exchanging messages is completed, the function block comes to a normal end.

● Topic Name

Specify the topic to which the PUBLISH message to send belongs in the input variable *Topic*. You can use / to nest topics.

The topic name does not accept a null characters, so specify a character string of one or more characters. In addition, for a PUBLISH message, you cannot use + or # as a wildcard. If any wildcard is included, *Topic* encounters an illegal input value error.

● QoS level

Specify the QoS level of a Publish message to send with the member *PubQoS* of the input variable *PubSettings*. Message delivery assurance vary depending on the QoS level.

QoS level	Assurance about transmission	Description
0	Up to once	A message is sent only once, and whether or not the recipient received is not guaranteed. A <i>PUBLISH</i> packet is sent only once, and an acknowledgement to the message is not made.
1	At least once	Although duplicate transmission may be made, a message is guaranteed to be received by the recipient at least once. A <i>PUBACK</i> packet to the <i>PUBLISH</i> packet is sent from the recipient, so the sender waits for the reception of the acknowledgement. If the acknowledgement cannot be received, the sender will resend the <i>PUBLISH</i> packet.
2	Accurately once	A message is guaranteed to be received by the recipient securely once without being lost or duplicated. A <i>PUBREC</i> packet for the <i>PUBLISH</i> packet is sent from the recipient, so the sender waits for the reception. If the <i>PUBREC</i> packet cannot be received, the sender will resend the <i>PUBLISH</i> packet. When the <i>PUBREC</i> packet is received, the sender sends a <i>PUBREL</i> packet. A <i>PUBCOMP</i> packet for the <i>PUBREL</i> packet is sent from the recipient, so the sender waits for the reception. If the <i>PUBCOMP</i> packet cannot be received, the sender will resend the <i>PUBREL</i> packet.

● Message Retain Setting

You can request the MQTT broker to retain the PUBLISH message you sent by specifying TRUE for the member *RetainFlag* of the input variable *PubSettings*.

To cancel the message requested to the MQTT broker to retain it, specify the same topic name as when requesting, and send a 0-byte message (empty message) by specifying TRUE to the member *RetainFlag* of the input variable *PubSettings*.

● Timeout Setting

If you specify 1 or 2 for the QoS level and the message exchange with the MQTT broker is not completed within the time specified by the input variable *Timeout*, a timeout error occurs and the execution terminates abnormally.

● Packet ID

If you execute the function block with 0 specified for the in-out variable *PacketID*, a message will be sent with a packet ID assigned automatically inside the library.

When a *PUBLISH* packet is generated, 0 is output to *PacketID* if *PubSettings.PubQoS* is 0. When *PubSettings.PubQoS* is 1 or 2, the packet ID used when a packet was generated is output to *PacketID*.

If you execute the function block with a number other than 0 specified for *PacketID*, a message will be sent as the resend message (DUP=1) with the specified packet ID.

The same message can be resent by repeating the execution until the function block comes to a normal end.

● Message Type

Specify the type of a message to send at the start of execution of the function block in the in-out variable *MsgType*.

It is accessed only when a number other than 0 is specified in the in-out variable *PacketID*.

When the function block comes to a normal end, 0 is output to *MsgType*.

If the function block terminates abnormally due to a timeout error or send failed error, the type of message for which the acknowledgment from the MQTT broker could not be received is output.

● Connection with MQTT broker cut during execution

If connection between the MQTT broker and this library is cut when execution of the function block is in progress (during *Busy*=TRUE), this function block will come to an error end.

To send the same message after reconnection, after confirming that the connection with the MQTT broker has resumed, re-execute the function block with the input variable *PubSettings*, input variable *Topic*, and in-out variable *PubMsg* being as the same as the previous values.

If the output variable *SessionPresent* of the MQTTClient instruction is TRUE at the time of reconnection and the in-out variable *PacketID* and the in-out variable *MsgType* are re-executed with the same values as when the previous function block ended, the PUBLISH message sent before the disconnected can be retransmitted.

● Setting Changes During Execution

Even if the input variable *Topic*, input variable *PubSettings*, input variable *Timeout*, in-out variable *PacketID*, or in-out variable *MsgType* are changed during execution, the changed value will not be reflected.

The setting when the input variable *Execute* has changed to TRUE is used in execution.

Please note that if *PacketID* and *MsgType* are changed during execution, the values updated in this function block may be overwritten.

Timing Charts

The timing charts are shown below.

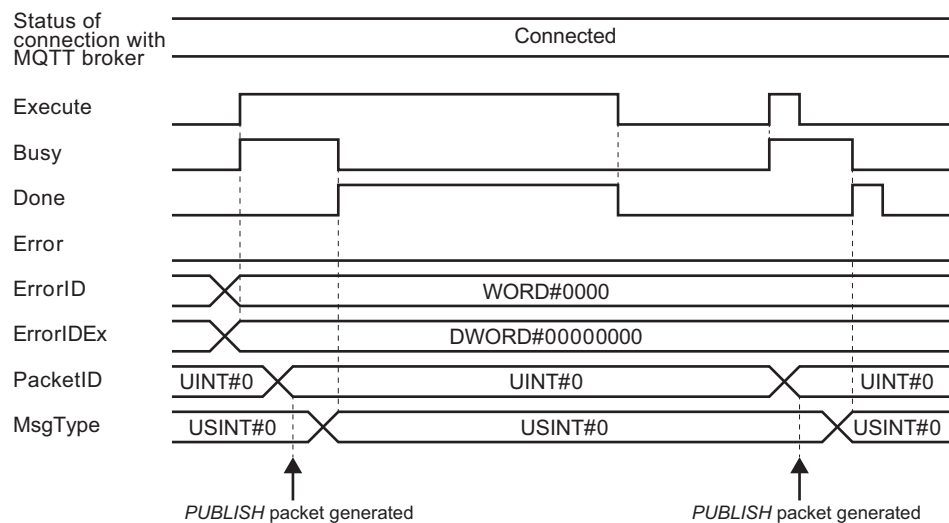
- *Busy* (Executing) changes to TRUE when *Execute* (Execute) changes to TRUE.
- When a *PUBLISH* packet is generated, the packet ID of the generated message is output to *PacketID*.
- When message exchange for the *PUBLISH* message is completed, *Done* (Done) changes to TRUE, *Busy* (Executing) changes to FALSE, and *MsgType* (Message type) outputs 0.
- If an error occurs when execution of the function block is in progress, *Error* (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).

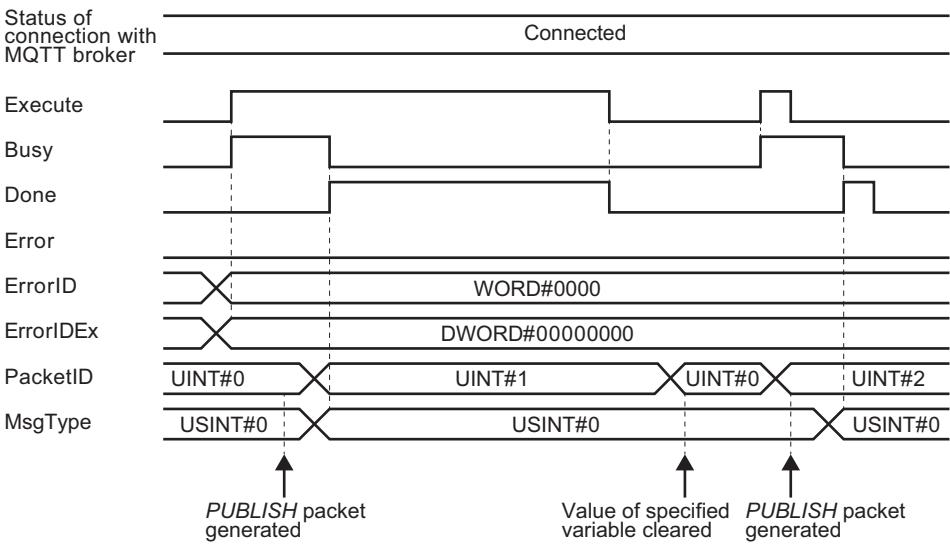
- In case of timeout error or a send failed error, the message type for which the acknowledgment from the MQTT broker could not be received is output to *MsgType* (Message type). By executing the function block again with the output value retained, the message that failed to be sent can be sent again.
- If *Execute* (Execute) changes to FALSE before execution of the function block is ended, *Done* (Done) and *Error* (Error) are TRUE only for one task period.
- If *Execute* (Execute) remains TRUE even after execution of the function block is ended, the output values of *Done* (Done) and *Error* (Error) are retained.

- Timing chart for normal end

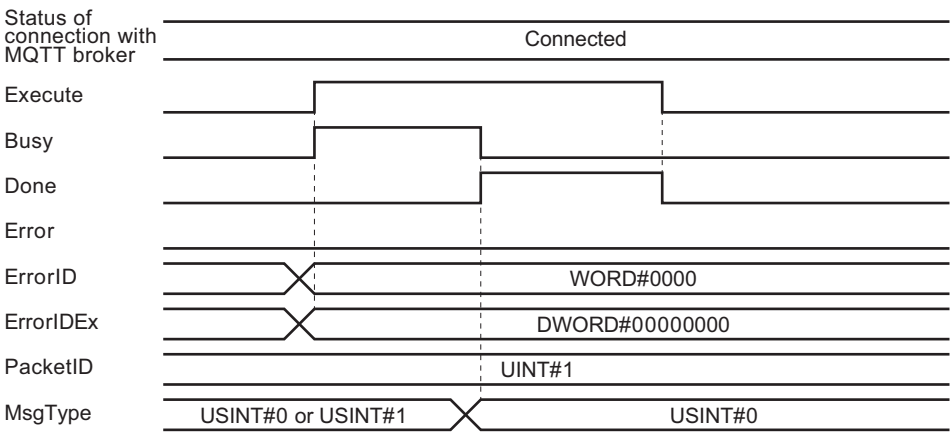
a) When the function block is executed with 0 specified for *PacketID* (Packet ID) at QoS level 0



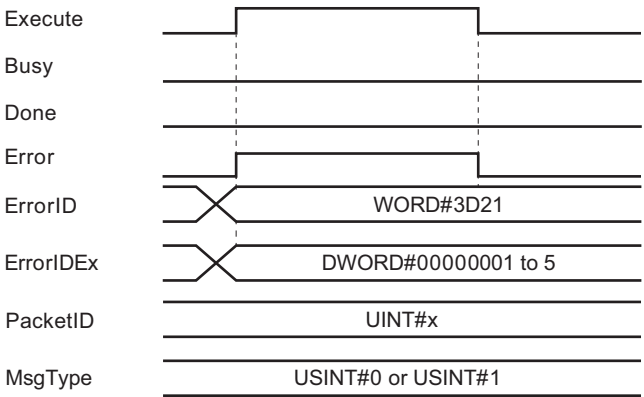
b) When the function block is executed with 0 specified for *PacketID* (Packet ID) at QoS level 1 or 2



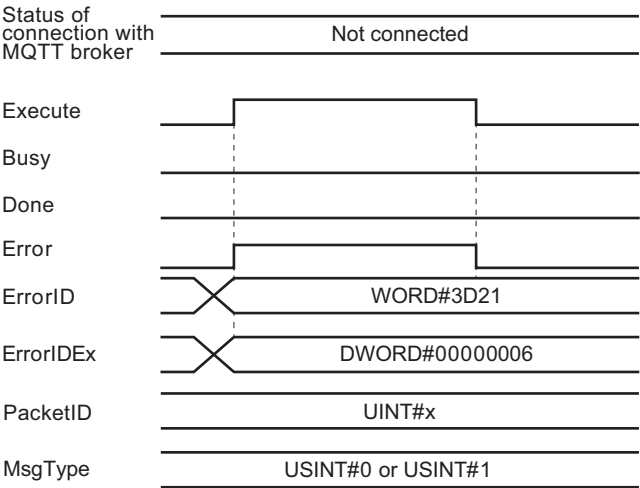
c) When the function block is executed with a number other than 0 specified for *PacketID* (Packet ID) at QoS level 1 or 2



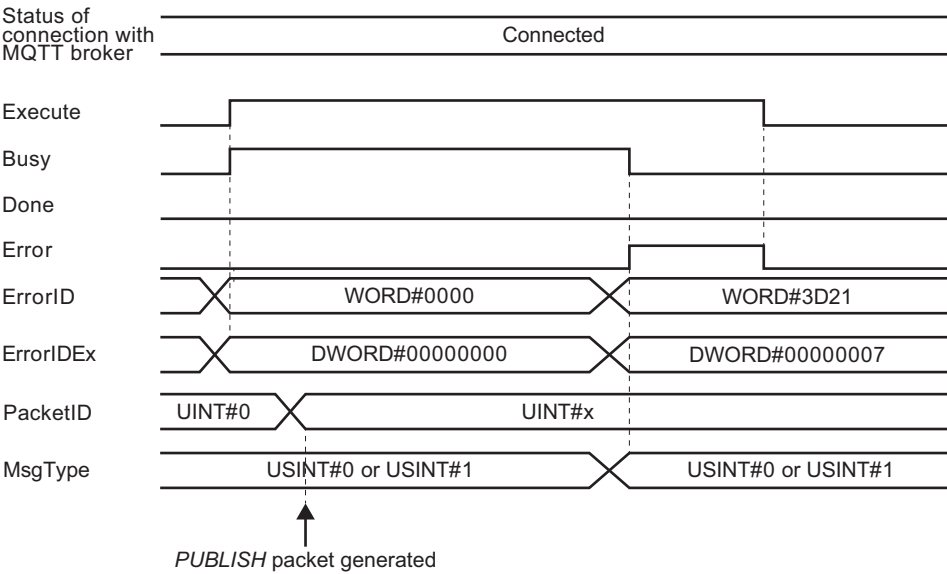
- Timing Chart for Error End
 - a) When there is an error in input parameters



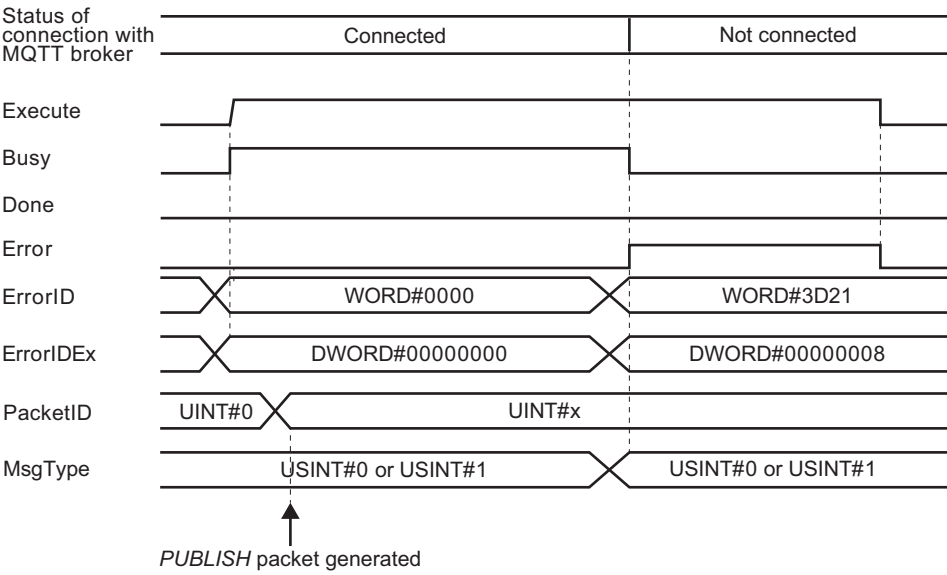
b) The MQTTClient instruction is not executed when this instruction is executed or an error has caused disconnection.



c) When a timeout error has occurred



d) When the connection with the MQTT broker is cut during execution of this instruction.



Precautions for Correct Use

- Execution of this function block will be continued until processing is ended even if the value of *Execute* changes to FALSE or the execution time exceeds the task period. The value of *Done* changes to TRUE when processing is ended. Use this to confirm normal ending of processing.
- If you cannot subscribe to messages sent by publish, make sure that the topic name you specify for publish or subscribe is correct.

Troubleshooting

Error code	Expansion error code	Status	Description	Corrective action
16#0000	16#00000000	Normal End	---	---
16#3D21	16#00000001	Illegal Topic Input Value	The value of the input variable <i>Topic</i> is illegal.	<ul style="list-style-type: none"> • Make sure it is a string of 1 to 511 characters. • Make sure that specifiable characters are used.
	16#00000002	PubQoS Input Value Out of Range	<i>PubSettings.PubQoS</i> is outside the valid range.	Make sure that it is in the valid range.
	16#00000003	MsgSize Input Value Out of Range	<ul style="list-style-type: none"> • The input variable <i>MsgSize</i> is outside the valid range. • The input variable <i>MsgSize</i> exceeds the range of the in-out variable <i>PubMsg</i>. 	<ul style="list-style-type: none"> • Make sure <i>MsgSize</i> is within the valid range. • Make sure that <i>MsgSize</i> does not exceed the number of elements in <i>PubMsg</i>.
	16#00000004	MsgType Input Value Out of Range	The in-out variable <i>MsgType</i> is outside the valid range.	<ul style="list-style-type: none"> • If <i>PubSettings.PubQoS</i> is 0, set <i>MsgType</i> to 0. • If <i>PubSettings.PubQoS</i> is 1, set <i>MsgType</i> to 0 or 1.
	16#00000005	Timeout Input Value Out of Range	The value of the input variable <i>Timeout</i> is outside the valid range.	Make sure that it is in the valid range.
	16#00000006	MQTT Broker Not Connected	Connection with the MQTT broker is not established when this instruction is executed.	Make sure that the instruction is executed when the output variable <i>Connected</i> of the MQTTClient instruction is TRUE.
	16#00000007	Publish Send Timeout	The publish message exchange was not completed within the time specified by the input variable <i>Timeout</i> .	<ul style="list-style-type: none"> • Make sure that the topic name and QoS level of the MQTTClient, MQTTPubArray, MQTTPubString, MQTTSubArray, and MQTTSubString instructions are those specified or supported by the MQTT broker. • Increase the time specified by <i>Timeout</i>. • Check the MQTT broker settings. • Check if there is a problem with the communication path.

Error code	Expansion error code	Status	Description	Corrective action
	16#00000008	Publish Send Failed	Publish message exchange failed.	<ul style="list-style-type: none">• Make sure that the topic name and QoS level of the MQTTClient, MQTTPubAryByte, MQTTTPubString, MQTTSubAryByte, and MQTTSubString instructions are those specified or supported by the MQTT broker.• Check the MQTT broker settings.

Sample Programming

Refer to the MQTTClient instruction *Sample Programming* on page 4-15.

MQTTPubString

This function block sends a PUBLISH message of the message specified in STRING data type to the MQTT broker via MQTTClient instance.

Function block name	Name	FB/ FUN	Graphic expression	ST expression
MQTTPubString	MQTT character string message publication	Function block	<div>MQTTPubString_instance</div> <div><div>\\OmronLib\MQTT_Comm\MQTTPubString</div><div><div>Execute</div><div>Done</div><div>ClientReference</div><div>ClientReference</div><div>PacketID</div><div>PacketID</div><div>MsgType</div><div>MsgType</div><div>PubMsg</div><div>PubMsg</div><div>PubSettings</div><div>Busy</div><div>Topic</div><div>Error</div><div>Timeout</div><div>ErrorID</div><div>ErrorIDEx</div></div></div>	MQTTPubString_instance(Execute:=, ClientReference:=, PacketID:=, MsgType:=, PubMsg:=, PubSettings:=, Topic:=, Timeout:=, Done=>, Busy=>, Error=>, ErrorID=>, ErrorIDEx=>,);

MQTTPubString_instance(

Execute:=,

ClientReference:=,

PacketID:=,

MsgType:=,

PubMsg:=,

PubSettings:=,

Topic:=,

Timeout:=,

Done=>,

Busy=>,

Error=>,

ErrorID=>,

ErrorIDEx=>,

);

Function Block and Function Information

Item	Description
Library file name	OmronLib_MQTT_Comm_Vx_x.slr (x shows the version.)
Namespace	OmronLib\MQTT_Comm
Function block and function number	00239
Source code published/not published	Not Published

Input Variables

Variable	Meaning	Data type	Description	Valid range	Unit	Default
Execute	Execute	BOOL	TRUE: Executes processing when this flag changes to TRUE. FALSE: Do not execute	TRUE, FALSE	---	FALSE

Variable	Meaning	Data type	Description	Valid range	Unit	Default
Topic	Topic	STRING[512]	Specify a topic name to publish.	Depends on data type	---	"
PubSettings	Publication setup	OmronLib \MQTT_Comm\sPubFlags	Configure the QoS and retain settings in publication.	---	---	Refer to OmronLib \MQTT_Comm\sPubFlags.
Timeout	Timeout time	UINT	This is a timeout time. It is accessed when the QoS level is 1 or 2.	1 to 1000	s	6

Output Variables

Variable	Meaning	Data type	Description	Valid range	Unit	Default
Done	Done	BOOL	TRUE: Normal end FALSE: Error end, execution in progress, or execution condition not met	TRUE, FALSE	---	---
Busy	Executing	BOOL	TRUE: Executing FALSE: Not executing	TRUE, FALSE	---	---
Error	Error	BOOL	TRUE: Error end FALSE: Normal end, execution in progress, or execution condition not met	TRUE, FALSE	---	---
ErrorID	Error code	WORD	This is the error ID for an error end. The value is 16#0 for a normal end.	*1	---	---
ErrorIDEx	Expansion error code	DWORD	This is the expansion error ID for an error end. The value is 16#0 for a normal end.	*1	---	---

*1. Refer to *Troubleshooting* on page 4-36 for details.

Input-Output Variables

Variable	Meaning	Data type	Description	Valid range	Unit	Default
ClientReference	MQTT client variable	OmronLib \MQTT_Comm\sClientReference	This is data to be shared among function blocks in this library. Do not change the data. The data contents are not published.	Depends on data type	---	---
PubMsg	Message	STRING[1986]	This is a message to be published to the specified topic.	Depends on data type	---	---

Variable	Meaning	Data type	Description	Valid range	Unit	Default
PacketID	Packet ID	UINT	This is a packet ID. If any number other than 0 is specified at the start of execution, it will be sent as the resend message with the specified packet ID. The packet ID used in publication is output. It is accessed when the QoS level is 1 or 2.	Depends on data type	---	---
MsgType	Message type	USINT	Send message type 0: <i>PUBLISH</i> 1: <i>PUBREL</i> It is accessed when the QoS level is 1 or 2 and PacketID is not 0.	0.1	---	---

Function

This is different from the *MQTTPubAryByte* instruction in the data type in which to specify the message, but they are the same in the function. Refer to *MQTTPubAryByte* on page 4-24 for details on the function.

Timing Charts

ErrorID (Error code) is different from *ErrorIDEx* (Expansion error code), but the timing charts are the same as those for the *MQTTPubAryByte* instruction. Refer to *MQTTPubAryByte* on page 4-24 for details on the timing charts. Refer to *Troubleshooting* on page 4-36 for details on *ErrorID* (Error code) and *ErrorIDEx* (Expansion error code).

Additional Information

For this FB, use the socket service function. Refer to the *NJ/NX-series CPU Unit Built-in EtherNet/IP™ Port User's Manual (Cat. No. W506)* for details of the socket service function.

Precautions for Correct Use

- Execution of this function block will be continued until processing is ended even if the value of *Execute* changes to FALSE or the execution time exceeds the task period. The value of *Done* changes to TRUE when processing is ended. Use this to confirm normal ending of processing.
- If you cannot subscribe to messages sent by publish, make sure that the topic name you specify for publish or subscribe is correct.

Troubleshooting

Error code	Expansion error code	Status	Description	Corrective action
16#0000	16#00000000	Normal End	---	---

Error code	Expansion error code	Status	Description	Corrective action
16#3D22	16#00000001	Illegal Topic Input Value	The value of the input variable <i>Topic</i> is illegal.	<ul style="list-style-type: none"> Make sure it is a string of 1 to 511 characters. Make sure that specifiable characters are used.
	16#00000002	PubQoS Input Value Out of Range	<i>PubSettings.PubQoS</i> is outside the valid range.	Make sure that it is in the valid range.
	16#00000003	MsgType Input Value Out of Range	The in-out variable <i>MsgType</i> is outside the valid range.	<ul style="list-style-type: none"> If <i>PubSettings.PubQoS</i> is 0, set <i>MsgType</i> to 0. If <i>PubSettings.PubQoS</i> is 1, set <i>MsgType</i> to 0 or 1.
	16#00000004	Timeout Input Value Out of Range	The value of the input variable <i>Timeout</i> is outside the valid range.	Make sure that it is in the valid range.
	16#00000005	MQTT Broker Not Connected	Connection with the MQTT broker is not established when this instruction is executed.	Make sure that the instruction is executed when the output variable <i>Connected</i> of the MQTTClient instruction is TRUE.
	16#00000006	Publish Send Timeout	The publish message exchange was not completed within the time specified by the input variable <i>Timeout</i> .	<ul style="list-style-type: none"> Make sure that the topic name and QoS level of the MQTTClient, MQTTPubAry-Byte, MQTTTPubString, MQTTSubAry-Byte, and MQTTSubString instructions are those specified or supported by the MQTT broker. Increase the time specified by <i>Timeout</i>. Check the MQTT broker settings. Check if there is a problem with the communication path.
	16#00000007	Publish Send Failed	Publish message exchange failed.	<ul style="list-style-type: none"> Make sure that the topic name and QoS level of the MQTTClient, MQTTPubAry-Byte, MQTTTPubString, MQTTSubAry-Byte, and MQTTSubString instructions are those specified or supported by the MQTT broker. Check the MQTT broker settings.

Sample Programming

Create a program for MQTTPubString (MQTT character string message publication) instruction by referencing *Sample Programming* on page 4-15 for MQTTClient instruction.

MQTTSubAryByte

This function block subscribes to the specified topic, and reads out the message part of a PUBLISH message in the received specified topic in BYTE array.

Function block name	Name	FB/ FUN	Graphic expression	ST expression
MQTTSubAryByte	Data subscription request in MQTT byte array	Function block	<p>MQTTSubAryByte_instance</p>	<pre>MQTTSubscribe_instance(Enable:=, ClientReference:=, RcvMsg:=, SubQoS:=, Topic:=, Timeout:=, Subscribed=>, Status=>, Received=>, MsgSize=>, RcvTopic=> Error=>, ErrorID=>, ErrorIDEx=>,);</pre>

Function Block and Function Information

Item	Description
Library file name	OmronLib_MQTT_Comm_Vx_x.slr (x shows the version.)
Namespace	OmronLib\MQTT_Comm
Function block and function number	00240
Source code published/not published	Not Published

Input Variables

Variable	Meaning	Data type	Description	Valid range	Unit	Default
Enable	Execute	BOOL	TRUE: Executes processing*1. FALSE: Stops processing.	TRUE, FALSE	---	FALSE

Variable	Meaning	Data type	Description	Valid range	Unit	Default
Topic	Topic	STRING[512]	Specify a topic targeted for subscription request.	Depends on data type	---	"
SubQoS	Maximum QoS level in subscription	BYTE	Set the maximum QoS level when a message is subscribed to.	0, 1, 2	---	0
Timeout	Timeout time	UINT	This is a timeout time for subscription request.	0 to 65535	s	10

- *1. Even if a command to start processing (change *Enable* from FALSE to TRUE) is given in a case other than that where the execution is stopped (*Status* is 0), the subscribed state (*Subscribed* is TRUE) will not be created.

Output Variables

Variable	Meaning	Data type	Description	Valid range	Unit	Default
Subscribed	Subscription request complete	BOOL	TRUE: Subscription in progress FALSE: Subscription not in progress	TRUE, FALSE	---	---
Status	Execution status	SINT	Execution status of function block 0: Execution stopped 1: Requesting subscription 2: Subscription in progress 3: Canceling subscription 4: Waiting for MQTT broker reconnection	0 to 4	---	---
Received	Reception completed	BOOL	Changes to TRUE after receiving a message from the MQTT broker.	TRUE, FALSE	---	---
Error	Error	BOOL	TRUE: Error end FALSE: Normal end, execution in progress, or execution condition not met	TRUE, FALSE	---	---
ErrorID	Error code	WORD	This is the error ID for an error end. The value is 16#0 for a normal end.	*1	---	---
ErrorIDex	Expansion error code	DWORD	This is the expansion error ID for an error end. The value is 16#0 for a normal end.	*1	---	---
MsgSize	Message size	UINT	This is a size of the message stored in RcvMsg[].	Depends on data type	Bytes	---
RcvTopic	Receive topic	STRING[512]	This is a topic name to which the received message belongs.	Depends on data type	---	---

- *1. Refer to *Troubleshooting* on page 4-46 for details.

Input-Output Variables

Variable	Meaning	Data type	Description	Valid range	Unit	Default
RcvMsg ^{*1}	Message	ARRAY[*] OF BYTE	This is a message part of a PUBLISH message in the specified topic.	Depends on data type	---	---
ClientReference	MQTT client variable	OmronLib \\MQTT_Comm\\sClientReference	This is data to be requested among function blocks in this library. Do not change the data. The data contents are not published.	---	---	---

*1. The maximum number of array elements is 65535. In addition, subscripts of the array should start with 0.

Function

When the input variable *Enable* changes from FALSE to TRUE with the function block in the execution stopped state, a *SUBSCRIBE* packet in the topic specified with the input variable *Topic* is created. A request to send the created *SUBSCRIBE* packet is submitted to the MQTTClient instance. When subscription is successful, the output variable *Subscribed* changes to TRUE, and the message part of a *PUBLISH* packet that matches the topic specified with *Topic* sent from the MQTT broker is read out. When the QoS level of the *PUBLISH* packet is 1 or 2, message exchange is carried out with the MQTT broker. If the connection with the MQTT broker is cut during subscription, *Subscribe* becomes FALSE and the instruction waits for the MQTT broker to reconnect.

If *Enable* changes from TRUE to FALSE, an *UNSUBSCRIBE* packet will be created and a send request will be submitted to the MQTTClient instance. When subscription cancellation is successful, the output variable *Status* changes to 0 (Execution stopped).

● Topic Filter

Specify a topic to be subscribed to in the input variable *Topic*.

You can use / to nest topics.

The topic name does not accept a null character, so specify a character string of one or more characters. You can use wildcards with # in *SUBSCRIBE* packets.

Wildcard	Description
"#"	A multi-level wildcard. You can specify only wildcards or after the topic delimiter /. You can use it only for the last character. Valid examples: "#", "animal/#" Illegal examples: "animal/dog#", "animal/#/color"

Wildcards cannot be specified with +. If + is included, an illegal *Topic* input value error will occur.

You cannot use a topic filter that starts with the wildcard # to receive a message from the topic that starts with \$. Specify a topic filter that does not include wildcards or a topic filter that contains # in the second and subsequent layers. (Example of a topic filter that can receive a message from "\$SYS/monitor/Client": "\$SYS/monitor/Client" or "\$SYS/monitor/#", Example that cannot receive a message: "#")

● QoS level

Specify the maximum QoS level of a message that is subscribed to, in the input variable *SubQoS*.

If the MQTT broker permits subscription at the maximum QoS level that is different from the QoS level specified with the *SUBSCRIBE* packet, the subscription will be carried out at the maximum QoS level that the MQTT broker permits. In addition, if the QoS level at which a Publisher sent a *PUBLISH* message is lower than the maximum QoS level, the message will be subscribed to at the QoS level that the Publisher sent the *PUBLISH* message.

● Timeout of Subscription Request

When subscription is requested, a *SUBSCRIBE* packet is sent and the reception of a *SUBACK* packet from the MQTT broker is awaited.

If the *SUBACK* packet failed to be received within the time specified with the input variable *Timeout*, this will be judged as a timeout error and the function block will terminate with an error.

When 0 is specified for *Timeout*, a timeout will occur in 10 seconds, which is a default.

● Disconnection from MQTT broker during subscription

If the connection with the MQTT broker is cut during subscription, this function block will not come to an error end, but will enter an MQTT broker reconnection waiting state and wait for reconnection.

When the reconnection with the MQTT broker is detected, the subscribed state is restored. However, if the reconnection is made in the clean session enabled state, the subscribed state will be restored after a subscription request is submitted again.

● *Enable* (Execute) changing to FALSE while MQTT broker reconnection is awaited

When the input variable *Enable* changes from TRUE to FALSE while MQTT broker reconnection is awaited, the function block will not enter the execution stopped state immediately after that change.

The reconnection waiting state continues until reconnection with the MQTT broker is made; after the reconnection, the subscription is canceled and then the function block enters execution stopped state.

● Message Read-out

When a *PUBLISH* packet that matches the topic specified with the input variable *Topic* is received, TRUE is output to the output variable *Received* for only one period.

The message part of the *PUBLISH* packet is output to the In-out variable *RcvMsg* when the message is received.

The size of the message part is output to the output variable *MsgSize*. The topic name to which the received *PUBLISH* packet belongs is output to the output variable *RcvTopic*.

If the data size of the message part of the received *PUBLISH* packet is larger than the size of *RcvMsg*, only the size of *RcvMsg* is stored.

When the topic name to which the received *PUBLISH* packet belongs is larger than *RcvTopic*, only the size of *RcvTopic* is stored.

● Setting Changes During Execution

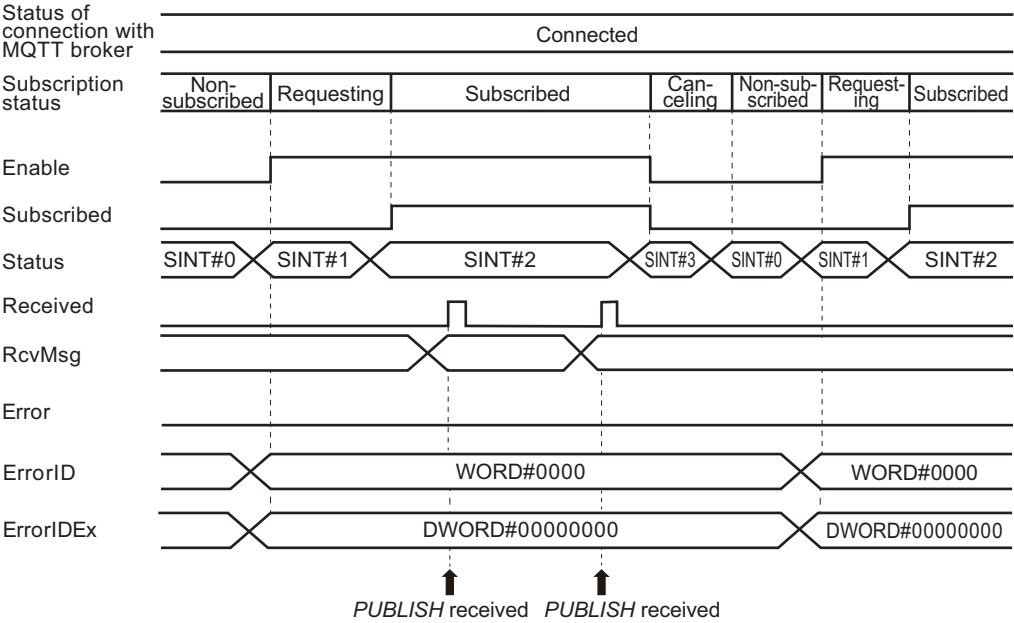
Even if the input variable *Topic*, input variable *SubQoS*, or input variable *Timeout* is changed during execution, the changed value will not be reflected.

The setting when the input variable *Enable* has changed to TRUE is used in execution.

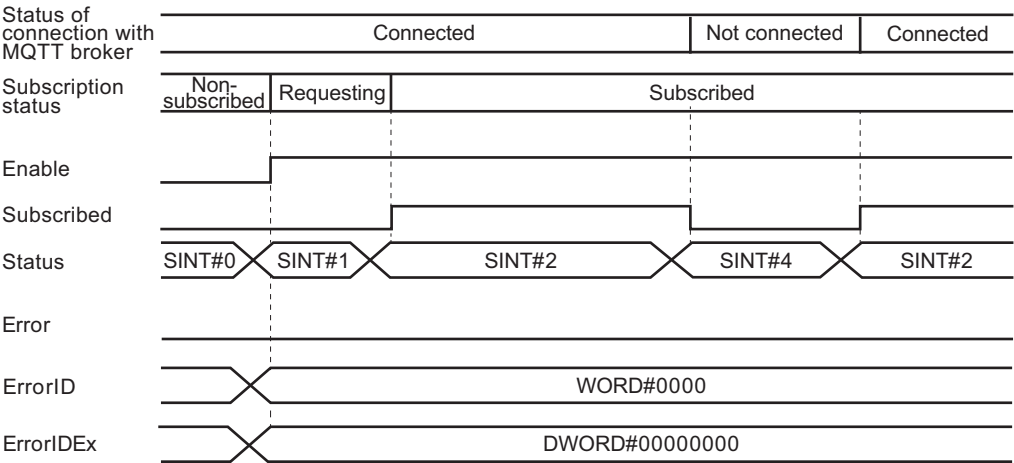
Timing Charts

The timing charts are shown below.

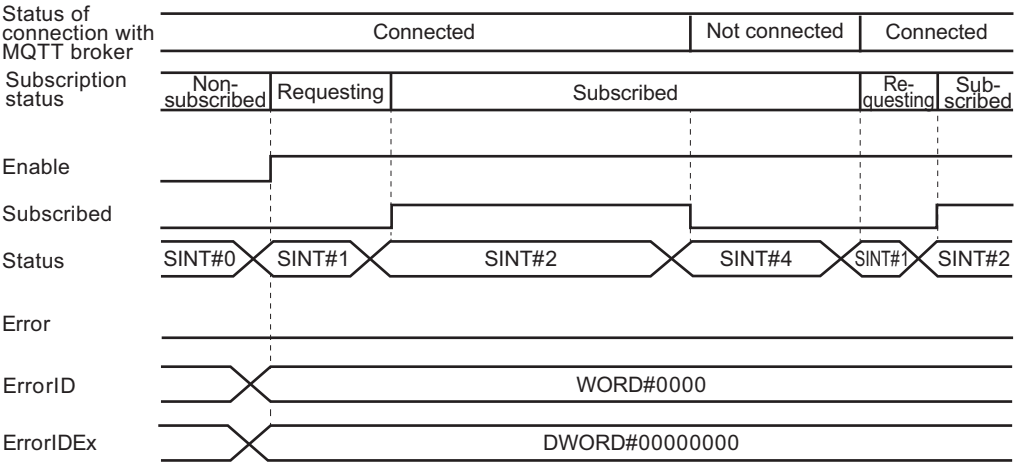
- When *Enable* (Execute) changes from FALSE to TRUE with *Status* (Execution status) being 0 (Execution stopped), *Status* (Execution status) changes to 1 (Requesting subscription).
- Then, when subscription is successful, *Status* (Execution status) changes to 2 (Subscription in progress), and *Subscribed* (Subscription request complete) changes to TRUE.
- When a *PUBLISH* packet that belongs to the topic that matches *Topic* is received with *Status* (Execution status) being 2 (Subscription in progress), *Received* (Reception complete) changes to TRUE for only one period.
- When *Enable* (Execute) changes from TRUE to FALSE with *Status* (Execution status) being 1 (Requesting subscription) or 2 (Subscription in progress), *Status* (Execution status) changes to 3 (Canceling subscription), and *Subscribed* (Subscription request complete) changes to FALSE.
- Then, when subscription cancellation is completed, *Status* (Execution status) changes to 0 (Execution stopped).
- If the connection with the MQTT broker is cut with *Status* (Execution status) being 2 (Subscription in progress), *Status* (Execution status) changes to 4 (Waiting for MQTT broker reconnection), and *Subscribed* (Subscription request complete) changes to FALSE.
- If reconnection is made in the clean session disabled state with *Status* (Execution status) being 4 (Waiting for MQTT broker reconnection), *Status* (Execution status) changes to 2 (Subscription in progress), and *Subscribed* (Subscription request complete) changes to TRUE. If reconnection is made in the clean session enabled state, *Status* (Execution status) changes to 1 (Requesting subscription).
- Even if *Enable* (Execute) changes from TRUE to FALSE with *Status* (Execution status) being 4 (Waiting for MQTT broker reconnection), *Status* (Execution status) will remain 4 (Waiting for MQTT broker reconnection). Then, if reconnection is made in the clean session disabled state, *Status* (Execution status) changes to 3 (Canceling subscription). If reconnection is made in the clean session enabled state, *Status* (Execution status) changes to 0 (Execution stopped).
- If an error occurs when execution of the function block is in progress, *Error* (Error) changes to TRUE. You can find out the cause of the error by accessing the values output to *ErrorID* (Error code) and *ErrorIDEx* (Expansion error code).
- Timing Chart for Normal End
 - a) When subscription request and cancellation are repeated



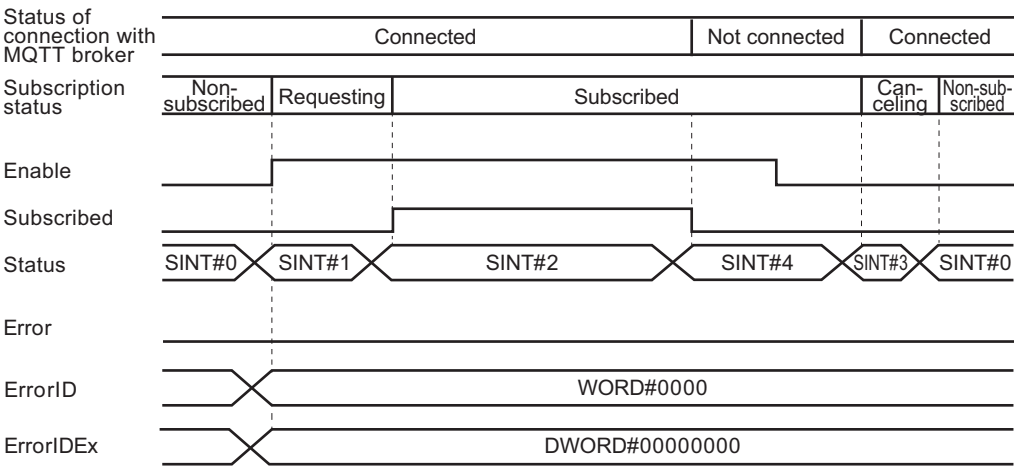
b) When the connection with the MQTT broker is cut during subscription and reconnection is made in the clean session disabled state



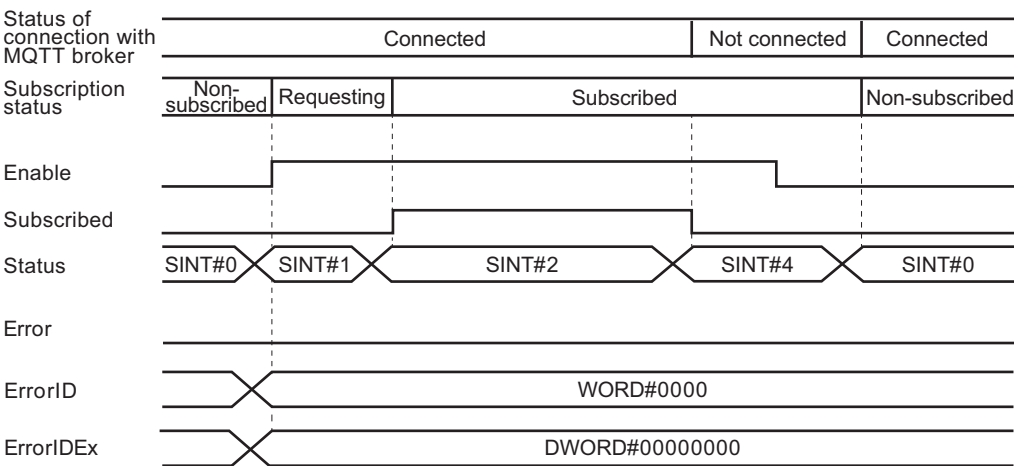
c) When the connection with the MQTT broker is cut during subscription and reconnection is made in the clean session enabled state



d) When *Enable* (Execute) changes to FALSE while reconnection is awaited and reconnection is made in the clean session disabled state

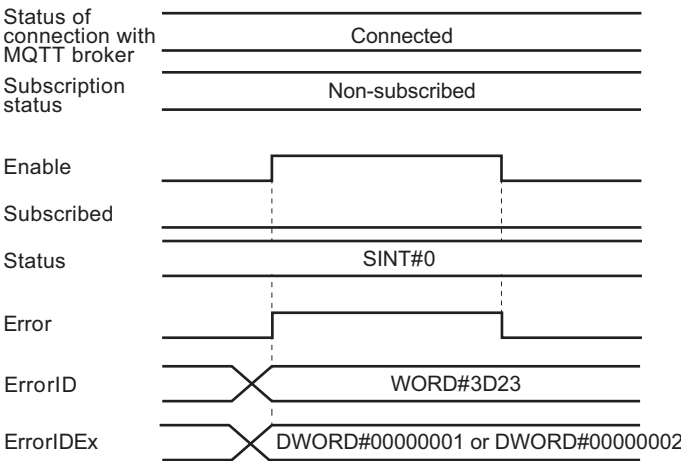


e) When *Enable* (Execute) changes to FALSE while reconnection is awaited and reconnection is made in the clean session enabled state

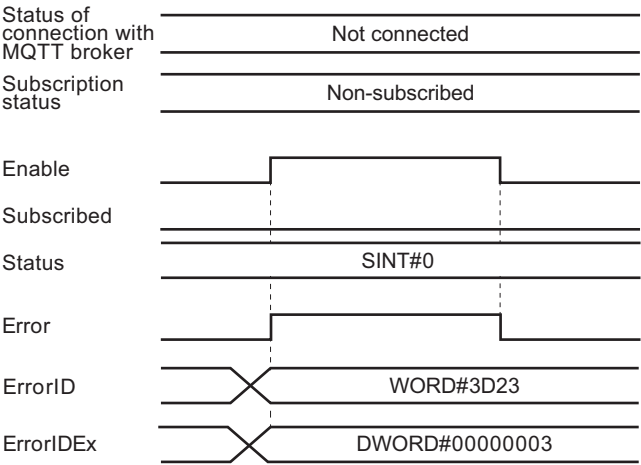


• Timing Chart for Error End

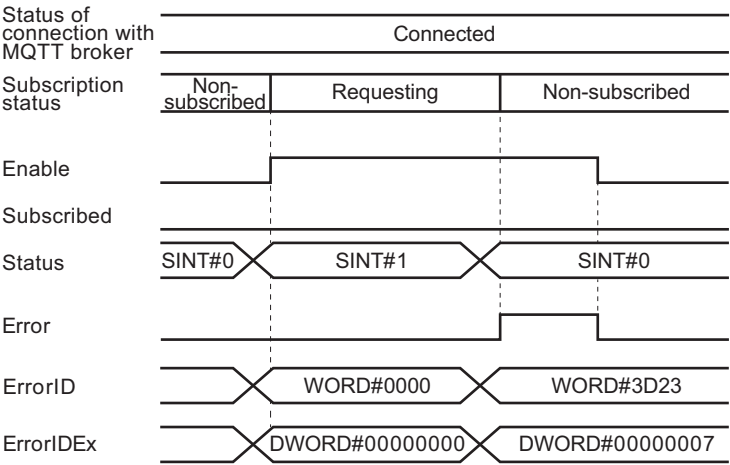
a) When there is an error in input parameters



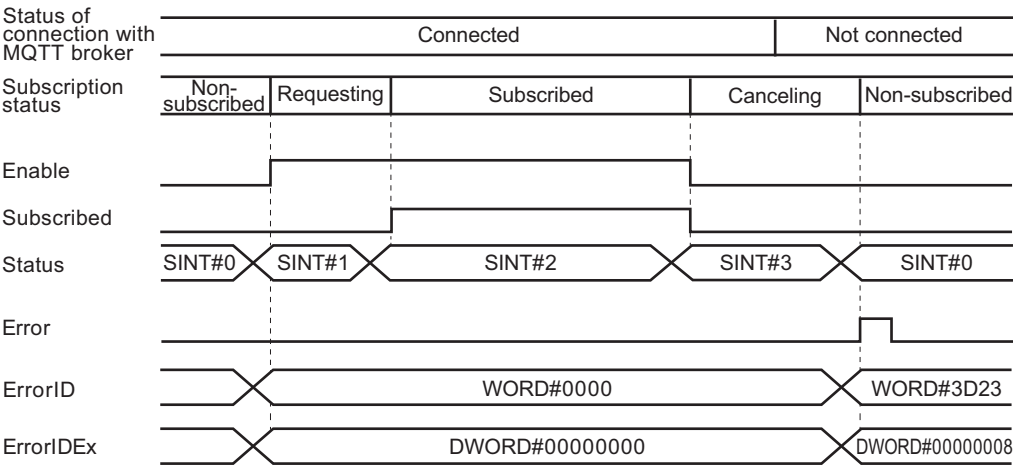
b) When the MQTTClient instruction is not executed or an error has caused disconnection



c) When a rejection response to subscription request is received from the MQTT broker



d) When an error occurs at the time of subscription cancellation



Precautions for Correct Use

- For this function block, even if the value of *Enable* changes to FALSE, the processing of the function block will not stop immediately. The value of *Status* changes to 0 when processing has stopped. Use this to confirm stop of processing.

- At the start of execution, unless the processing is in the stopped state, *Subscribed* will not change to TRUE. In starting execution, be sure to set *Enable* to TRUE after checking that *Status* is 0.
- If you cannot subscribe to messages sent by publish or subscribe to messages from MQTT brokers, make sure that the topic name you specify for publish or subscribe is correct.

Troubleshooting

Error code	Expansion error code	Status	Description	Corrective action
16#0000	16#00000000	Normal End	---	---
16#3D23	16#00000001	Illegal Topic Input Value	The value of the input variable <i>Topic</i> is illegal.	<ul style="list-style-type: none"> • Make sure it is a string of 1 to 511 characters. • Make sure that specifiable characters are used.
	16#00000002	SubQoS Input Value Out of Range	The input variable <i>SubQoS</i> is outside the valid range.	Make sure that it is in the valid range.
	16#00000003	MQTT Broker Not Connected	Connection with the MQTT broker is not established when this instruction is executed.	Make sure that the instruction is executed when the output variable <i>Connected</i> of the MQTTClient instruction is TRUE.
	16#00000005	Subscription Request Send Failed	Sending a subscription request failed.	<ul style="list-style-type: none"> • Make sure that the topic name and QoS level of the MQTTClient, MQTTPubAry-Byte, MQTTTPubString, MQTTSubAry-Byte, and MQTTSubString instructions are those specified or supported by the MQTT broker. • Check the MQTT broker settings.
	16#00000006	Subscription Request Timeout	Subscription request was not completed within the time specified with the input variable <i>Timeout</i> .	<ul style="list-style-type: none"> • Make sure that the topic name and QoS level of the MQTTClient, MQTTPubAry-Byte, MQTTTPubString, MQTTSubAry-Byte, and MQTTSubString instructions are those specified or supported by the MQTT broker. • Increase the time specified by <i>Timeout</i>. • Check the MQTT broker settings. • Check if there is a problem with the communication path.
	16#00000007	Subscription Registration Failed	Subscription registration failed.	<ul style="list-style-type: none"> • Make sure that the topic name and QoS level of the MQTTClient, MQTTPubAry-Byte, MQTTTPubString, MQTTSubAry-Byte, and MQTTSubString instructions are those specified or supported by the MQTT broker. • Check the MQTT broker settings.
	16#00000008	Subscription Cancellation Failed	Subscription cancellation failed.	<ul style="list-style-type: none"> • Make sure that the topic name and QoS level of the MQTTClient, MQTTPubAry-Byte, MQTTTPubString, MQTTSubAry-Byte, and MQTTSubString instructions are those specified or supported by the MQTT broker. • Check the MQTT broker settings.

Error code	Expansion error code	Status	Description	Corrective action
	16#00000009	Subscription Cancellation Timeout	Subscription cancellation was not completed within the time specified with the input variable <i>Timeout</i> .	<ul style="list-style-type: none">• Make sure that the topic name and QoS level of the MQTTClient, MQTTPubArrayByte, MQTTPubString, MQTTSubArrayByte, and MQTTSubString instructions are those specified or supported by the MQTT broker.• Increase the time specified by <i>Timeout</i>.• Check the MQTT broker settings.• Check if there is a problem with the communication path.

Sample Programming

Refer to the MQTTClient instruction *Sample Programming* on page 4-15.

MQTTSubString

This function block subscribes to the specified topic, and reads out the message part of a PUBLISH message in the received specified topic in STRING data type.

Function block name	Name	FB/ FUN	Graphic expression	ST expression
MQTTSubString	MQTT character string message subscription request	Function block	<p>MQTTSubString_instance</p>	<pre>MQTTSubString_instance(Enable:=, ClientReference:=, RcvMsg:=, SubQoS:=, Topic:=, Timeout:=, Subscribed=>, Status=>, Received=>, RcvTopic=>, Error=>, ErrorID=>, ErrorIDEx=>,);</pre>

Function Block and Function Information

Item	Description
Library file name	OmronLib_MQTT_Comm_Vx_x.slr (x shows the version.)
Namespace	OmronLib\MQTT_Comm
Function block and function number	00241
Source code published/not published	Not Published

Input Variables

Variable	Meaning	Data type	Description	Valid range	Unit	Default
Enable	Execute	BOOL	TRUE: Executes processing*1. FALSE: Stops processing.	TRUE, FALSE	---	FALSE

Variable	Meaning	Data type	Description	Valid range	Unit	Default
Topic	Topic	STRING[512]	Specify a topic targeted for subscription request.	Depends on data type	---	"
SubQoS	Maximum QoS level in subscription	BYTE	Set the maximum QoS level when a message is subscribed to.	0, 1, 2	---	0
Timeout	Timeout time	UINT	This is a timeout time for subscription request.	0 to 65535	s	10

- *1. Even if a command to start processing (change *Enable* from FALSE to TRUE) is given in a case other than that where the execution is stopped (*Status* is 0), the subscribed state (*Subscribed* is TRUE) will not be created.

Output Variables

Variable	Meaning	Data type	Description	Valid range	Unit	Default
Subscribed	Subscription request complete	BOOL	TRUE: Subscription in progress FALSE: Subscription not in progress	TRUE, FALSE	---	---
Status	Execution status	SINT	Execution status of function block 0: Execution stopped 1: Requesting subscription 2: Subscription in progress 3: Canceling subscription 4: Waiting for MQTT broker reconnection	0 to 4	---	---
Received	Reception completed	BOOL	Changes to TRUE after receiving a message from the MQTT broker.	TRUE, FALSE	---	---
Error	Error	BOOL	TRUE: Error end FALSE: Normal end, execution in progress, or execution condition not met	TRUE, FALSE	---	---
ErrorID	Error code	WORD	This is the error ID for an error end. The value is 16#0 for a normal end.	*1	---	---
ErrorIDex	Expansion error code	DWORD	This is the expansion error ID for an error end. The value is 16#0 for a normal end.	*1	---	---
RcvTopic	Receive topic	STRING[512]	This is a topic name to which the received message belongs.	Depends on data type	---	---

- *1. Refer to *Troubleshooting* on page 4-50 for details.

Input-Output Variables

Variable	Meaning	Data type	Description	Valid range	Unit	Default
RcvMsg	Message	STRING[1986]	This is a message part of a PUBLISH message in the specified topic.	Depends on data type	---	---
ClientReference	MQTT client variable	OmronLib \MQTT_Comm\ ClientReference	This is data to be requested among function blocks in this library. Do not change the data. The data contents are not published.	---	---	---

Function

This is different from the MQTTSubAryByte instruction in the data type in which to output the read-out message, but they are the same in the function. Refer to *MQTTSubAryByte* on page 4-38 for details on the function.

Timing Charts

ErrorID (Error code) is different from *ErrorIDEx* (Expansion error code), but the timing charts are the same as those for the MQTTSubAryByte instruction. Refer to *MQTTSubAryByte* on page 4-38 for details on the timing charts. Refer to *Troubleshooting* on page 4-50 for details on *ErrorID* (Error code) and *ErrorIDEx* (Expansion error code).

Precautions for Correct Use

- For this function block, even if the value of *Enable* changes to FALSE, the processing of the function block will not stop immediately. The value of *Status* changes to 0 when processing has stopped. Use this to confirm stop of processing.
- At the start of execution, unless the processing is in the stopped state, *Subscribed* will not change to TRUE. In starting execution, be sure to set *Enable* to TRUE after checking that *Status* is 0.
- If you cannot subscribe to messages sent by publish or subscribe to messages from MQTT brokers, make sure that the topic name you specify for publish or subscribe is correct.

Troubleshooting

Error code	Expansion error code	Status	Description	Corrective action
16#0000	16#00000000	Normal End	---	---
16#3D24	16#00000001	Illegal Topic Input Value	The value of the input variable <i>Topic</i> is illegal.	<ul style="list-style-type: none"> • Make sure it is a string of 1 to 511 characters. • Make sure that specifiable characters are used.
	16#00000002	SubQoS Input Value Out of Range	The input variable <i>SubQoS</i> is outside the valid range.	Make sure that it is in the valid range.

Error code	Expansion error code	Status	Description	Corrective action
	16#00000003	MQTT Broker Not Connected	Connection with the MQTT broker is not established when this instruction is executed.	Make sure that the instruction is executed when the output variable <i>Connected</i> of the MQTTClient instruction is TRUE.
	16#00000005	Subscription Request Send Failed	Sending a subscription request failed.	<ul style="list-style-type: none"> Make sure that the topic name and QoS level of the MQTTClient, MQTTPubAry-Byte, MQTTPubString, MQTTSubAry-Byte, and MQTTSubString instructions are those specified or supported by the MQTT broker. Check the MQTT broker settings.
	16#00000006	Subscription Request Timeout	Subscription request was not completed within the time specified with the input variable <i>Timeout</i> .	<ul style="list-style-type: none"> Make sure that the topic name and QoS level of the MQTTClient, MQTTPubAry-Byte, MQTTPubString, MQTTSubAry-Byte, and MQTTSubString instructions are those specified or supported by the MQTT broker. Increase the time specified by <i>Timeout</i>. Check the MQTT broker settings. Check if there is a problem with the communication path.
	16#00000007	Subscription Registration Failed	Subscription registration failed.	<ul style="list-style-type: none"> Make sure that the topic name and QoS level of the MQTTClient, MQTTPubAry-Byte, MQTTPubString, MQTTSubAry-Byte, and MQTTSubString instructions are those specified or supported by the MQTT broker. Check the MQTT broker settings.
	16#00000008	Subscription Cancellation Failed	Subscription cancellation failed.	<ul style="list-style-type: none"> Make sure that the topic name and QoS level of the MQTTClient, MQTTPubAry-Byte, MQTTPubString, MQTTSubAry-Byte, and MQTTSubString instructions are those specified or supported by the MQTT broker. Check the MQTT broker settings.
	16#00000009	Subscription Cancellation Timeout	Subscription cancellation was not completed within the time specified with the input variable <i>Timeout</i> .	<ul style="list-style-type: none"> Make sure that the topic name and QoS level of the MQTTClient, MQTTPubAry-Byte, MQTTPubString, MQTTSubAry-Byte, and MQTTSubString instructions are those specified or supported by the MQTT broker. Increase the time specified by <i>Timeout</i>. Check the MQTT broker settings. Check if there is a problem with the communication path.

Sample Programming

Create a program for MQTTSubString (MQTT character string message subscription request) instruction by referencing *Sample Programming* on page 4-15 for MQTTClient instruction.

MQTTPing

This function block sends a PING message via MQTTClient, and measures the time it takes for a response to be returned from the MQTT broker.

Function block name	Name	FB/ FUN	Graphic expression	ST expression
MQTTPing	MQTTPing message send	Function block	<div>MQTTPing_instance </div>	MQTTPing_instance(Execute:=, ClientReference:=, Timeout:=, Done=:=, Busy=:=, ElapsedTime=:=, Error=:=, ErrorID=:=, ErrorIDEx=:=,);

Function Block and Function Information

Item	Description
Library file name	OmronLib_MQTT_Comm_Vx_x.slr (x shows the version.)
Namespace	OmronLib\MQTT_Comm
Function block and function number	00242
Source code published/not published	Not Published

Input Variables

Variable	Meaning	Data type	Description	Valid range	Unit	Default
Execute	Execute	BOOL	TRUE: Executes processing when this flag changes to TRUE. FALSE: Do not execute	TRUE, FALSE	---	FALSE
Timeout	Timeout time	UINT	This is a wait time for a response from the MQTT broker.	Depends on data type	ms	1000

Output Variables

Variable	Meaning	Data type	Description	Valid range	Unit	Default
Done	Done	BOOL	TRUE: Normal end FALSE: Error end, execution in progress, or execution condition not met	TRUE, FALSE	---	---
Busy	Executing	BOOL	TRUE: Executing FALSE: Not executing	TRUE, FALSE	---	---
Error	Error	BOOL	TRUE: Error end FALSE: Normal end, execution in progress, or execution condition not met	TRUE, FALSE	---	---
ErrorID	Error code	WORD	This is the error ID for an error end. The value is 16#0 for a normal end.	*1	---	---
ErrorIDEx	Expansion error code	DWORD	This is the expansion error ID for an error end. The value is 16#0 for a normal end.	*1	---	---
EIapseTime	Elapsed time	UINT	The time between when the <i>PINGREQ</i> packet is sent and when the <i>PINGRESP</i> packet is received	Depends on data type	ms	---

*1. Refer to *Troubleshooting* on page 4-56 for details.

Input-Output Variables

Variable	Meaning	Data type	Description	Valid range	Unit	Default
ClientReference	MQTT client variable	OmronLib \MQTT_Comm\CLIENTReference	This is data to be shared among function blocks in this library. Do not change the data. The data contents are not published.	---	---	---

Function

When the input variable *Execute* changes to TRUE during the connection with the MQTT broker, a *PINGREQ* packet is created. A request to send the created *PINGREQ* packet is submitted to the MQTTClient instance. When the *PINGRESP* packet sent from the MQTT broker is received, the function block comes to a normal end.

The time it took for the *PINGRESP* packet to be received after the *PINGREQ* packet was sent is measured, and the measurement result is output to the output variable *EIapseTime*.

● Timeout

If the *PINGRESP* packet failed to be received within the time specified with the input variable *Timeout*, this will be judged as a timeout and the output variable *Error* will change to TRUE.

When 0 is specified, a timeout will occur in 1,000 ms, which is a default.

● **Elapsed Time**

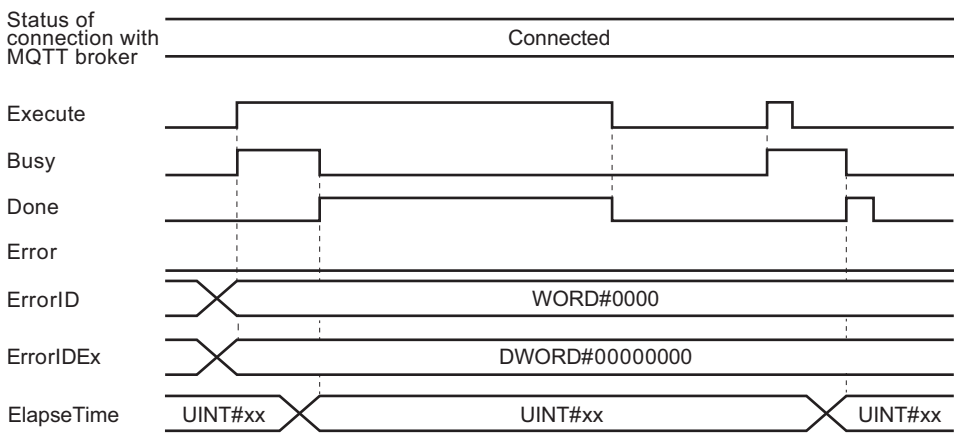
When execution is completed successfully, the time it took for the *PINGRESP* packet to be received after the *PINGREQ* packet was sent will be output.
In case of an error end, the value will not be refreshed.

Timing Charts

The timing charts are shown below.

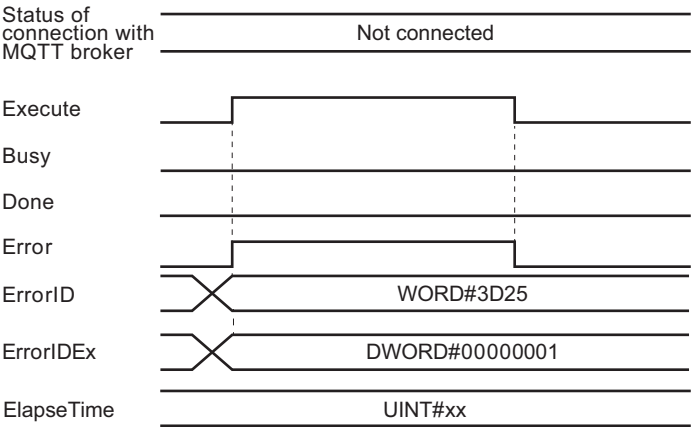
- *Busy* (Executing) changes to TRUE when *Execute* (Execute) changes to TRUE.
- When the *PINGRESP* packet is received, *Done* (Done) changes to TRUE, *Busy* (Executing) changes to FALSE, and the elapsed time is output to *ElapseTime*.
- If an error occurs when execution of the function block is in progress, *Error* (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* (Execute) changes to FALSE before execution of the function block is ended, *Done* (Done) and *Error* (Error) are TRUE only for one task period.
- If *Execute* (Execute) remains TRUE even after execution of the function block is ended, the output values of *Done* (Done) and *Error* (Error) are retained.

• Timing Chart for Normal End

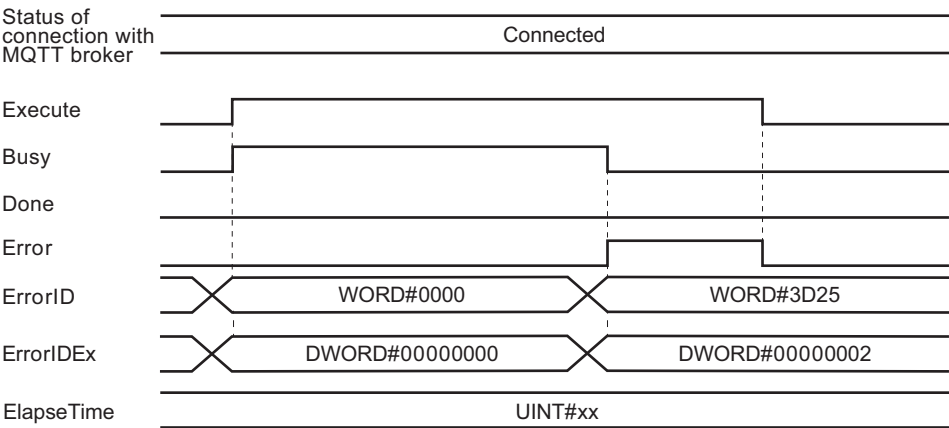


• Timing Chart for Error End

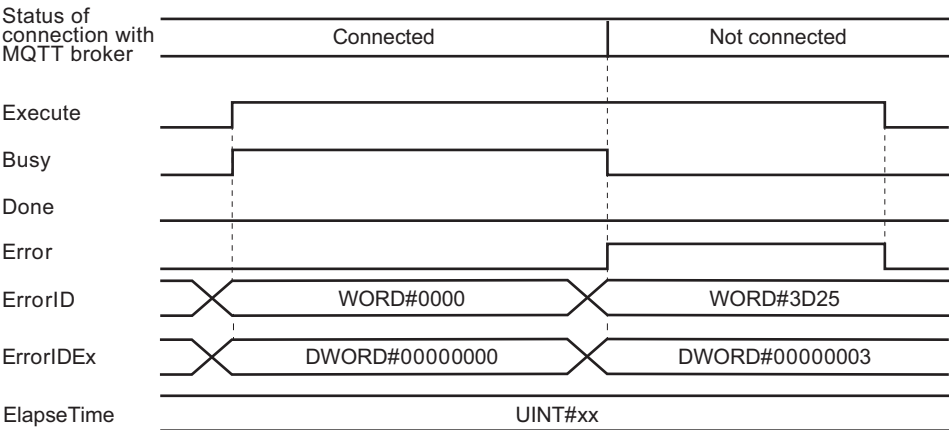
- a) The MQTTClient instruction is not executed when this instruction is executed or an error has caused disconnection.



b) When a timeout has occurred during execution of this instruction.



c) When the connection with the MQTT broker is cut during execution of this instruction.



Precautions for Correct Use

- Execution of this function block will be continued until processing is ended even if the value of *Execute* changes to FALSE or the execution time exceeds the task period. The value of *Done* changes to TRUE when processing is ended. Use this to confirm normal ending of processing.
- The time of the range between when a request to send a *PINGREQ* packet is submitted and when the reception of a *PINGRESP* packet is acknowledged is measured inside this function block, so the measurement time is affected by the task period.

- In order to measure the time it took for a response to be returned from the MQTT broker more accurately, execute this function block while the MQTTPubAryByte instruction, MQTTPubString instruction, MQTTSubAryByte instruction, and MQTTSubString instruction are in the execution stopped state.

Troubleshooting

Error code	Expansion error code	Status	Description	Corrective action
16#0000	16#00000000	Normal End	---	---
16#3D25	16#00000001	MQTT Broker Not Connected	Connection with the MQTT broker is not established when this instruction is executed.	Make sure that the instruction is executed when the output variable <i>Connected</i> of the MQTTClient instruction is TRUE.
	16#00000002	PINGRESP Packet Reception Timeout	The PINGRESP packet from the MQTT broker can not be received within the time specified by the input variable <i>Timeout</i> .	<ul style="list-style-type: none"> • Make sure that the topic name and QoS level of the MQTTClient, MQTTPubAryByte, MQTTTPubString, MQTTSubAryByte, and MQTTSubString instructions are those specified or supported by the MQTT broker. • Increase the time specified by <i>Timeout</i>. • Check the MQTT broker settings. • Check if there is a problem with the communication path.
	16#00000003	MQTT Broker Disconnection	The connection with the MQTT broker is cut during execution of this instruction.	<ul style="list-style-type: none"> • Make sure that the topic name and QoS level of the MQTTClient, MQTTPubAryByte, MQTTTPubString, MQTTSubAryByte, and MQTTSubString instructions are those specified or supported by the MQTT broker. • Check the MQTT broker settings.

Sample Programming

Refer to the MQTTClient instruction *Sample Programming* on page 4-15.



Appendix

This section describes information that is convenient to know, such as library information reference methods, FB or FUN source code reference methods, etc.

A-1	Referring to Library Information	A-2
A-1-1	Library Attributes, and FB or FUN Attributes	A-2
A-1-2	Referring to Attributes of Libraries, Function Blocks, and Functions.....	A-2
A-2	Procedure for Connection to Azure IoT Hub via MQTT	A-5
A-2-1	Overall Procedure	A-5
A-2-2	System Configuration and Properties of Configuration Elements	A-6
A-2-3	Preliminary Preparations on Azure Side	A-8
A-2-4	Network Setting for PC.....	A-10
A-2-5	Network Setting for CPU Unit.....	A-10
A-2-6	Secure Socket Setting for CPU Unit.....	A-11
A-2-7	Creation of Program and Execution of Program	A-15
A-3	Procedure for Connection to AWS IoT via MQTT	A-22
A-3-1	Overall Procedure	A-22
A-3-2	System Configuration and Properties of Configuration Elements	A-23
A-3-3	Preliminary Preparations on AWS Side.....	A-24
A-3-4	Network Setting for PC.....	A-25
A-3-5	Network Setting for CPU Unit.....	A-25
A-3-6	Secure Socket Setting for CPU Unit.....	A-26
A-3-7	Creation of Program and Execution of Program	A-28

A-1 Referring to Library Information

When you make an inquiry to OMRON about a 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.

A-1-1 Library Attributes, and FB or FUN Attributes

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

Library Attributes

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 the 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, .

*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 the 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 the 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, .

*2. It is provided in English and Japanese.

A-1-2 Referring to Attributes of Libraries, Function Blocks, and Functions

You can refer to the library attributes of library information, and FB or FUN attributes at the following Sysmac Studio locations.

- Library Reference Dialog Box
- Toolbox
- Programming screen

Library Reference Dialog Box

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

(1) Library file name (2) Library version (3) Library creator (4) Library comments

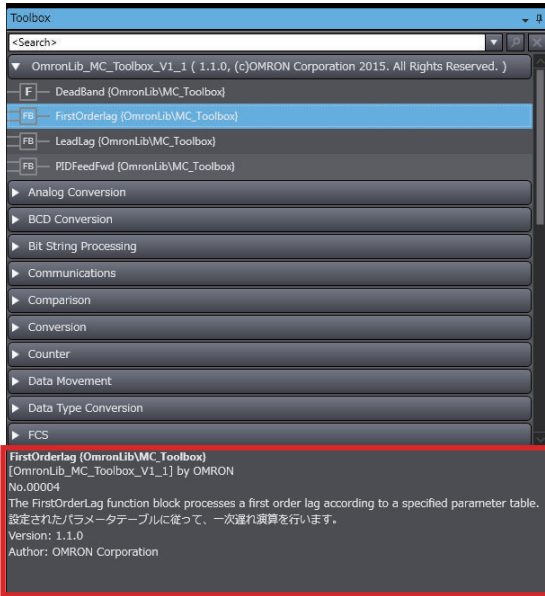
The screenshot shows the 'Library Reference' dialog box. It contains a table with columns: Library name, Name Space, Version, Author, Company, Date Creat, Date Modi, and Comment. The table lists several libraries, including 'OmronLib_MC_Toolbox_V1_1' and 'DeadBand (OmronLib\MC_Toolbox)'. Below the table is a section for 'Namespace - Using' with a table of attributes. Callouts (1) through (10) point to specific elements: (1) Library file name, (2) Library version, (3) Library creator, (4) Library comments, (5) FB/FUN name, (6) Name space, (7) FB/FUN version, (8) FB/FUN creator, (9) FB/FUN version, and (10) FB/FUN comments.

Library name	Name Space	Version	Author	Company	Date Creat	Date Modi	Comment
OmronLib_MC_Toolbox_V1_1		1.1.0	OMRON Corporation	©OMRON Corporation 2015. All Rights Reserved.			This is MC_Toolbox library. これはモーション制御ツールボックスライブラリです。
DeadBand (OmronLib\MC_Toolbox)	OmronLib\MC_Toolbox	1.1.0	OMRON Corporation		03/16/2015	08/10/2015	No.00006 The DeadBand function block control the output value to zero when the input value is zero. It is used to eliminate the effect of the input offset.
FirstOrderLag (OmronLib\MC_Toolbox)	OmronLib\MC_Toolbox	1.1.0	OMRON Corporation		04/01/2015	08/10/2015	No.00004 The FirstOrderLag function block performs the first-order lag operation. It is used to simulate the lag of the system.
LeadLag (OmronLib\MC_Toolbox)	OmronLib\MC_Toolbox	1.1.0	OMRON Corporation		04/01/2015	08/10/2015	No.00005 The LeadLag function block performs the lead-lag operation. It is used to simulate the lead-lag of the system.
PIDFeedFwd (OmronLib\MC_Toolbox)	OmronLib\MC_Toolbox	1.1.0	OMRON Corporation		04/01/2015	08/10/2015	No.00003 The PIDFeedFwd function block performs the PID feedforward operation. It is used to improve the control performance.

Name	In/Out	Data Type	Edge	Initial Value	Retain	Constant	Comment
Enable	Input	BOOL	No Edge	False	<input type="checkbox"/>	<input type="checkbox"/>	
InCalc	Input	LREAL	No Edge	0.0	<input type="checkbox"/>	<input type="checkbox"/>	
Kp	Input	LREAL	No Edge	1.0	<input type="checkbox"/>	<input type="checkbox"/>	
TimeConst	Input	LREAL	No Edge	1.0	<input type="checkbox"/>	<input type="checkbox"/>	
SampTime	Input	LREAL	No Edge	1.0	<input type="checkbox"/>	<input type="checkbox"/>	
Enabled	Output	BOOL	No Edge		<input type="checkbox"/>	<input type="checkbox"/>	

Toolbox

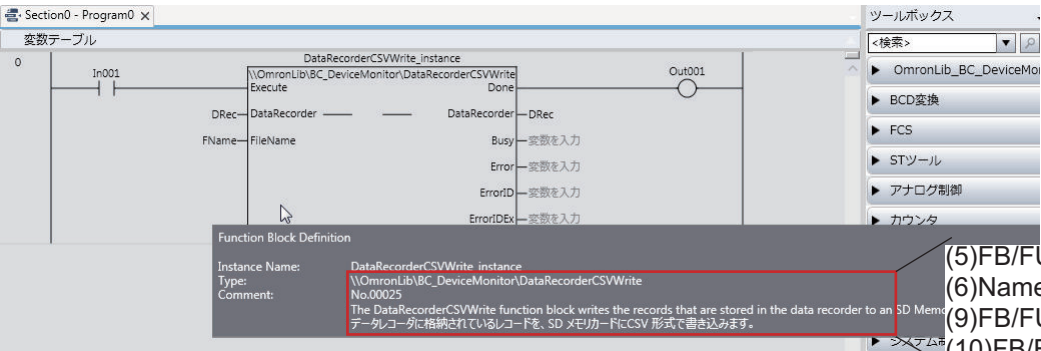
Select a function block or function to display its library information at the bottom of the Toolbox Pane. The text "**by OMRON**" which is shown on the right of the library name (1) indicates that this library was provided by OMRON.



- (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

Programming Screen

Place the mouse on a function block and function to display the library information in a tooltip.

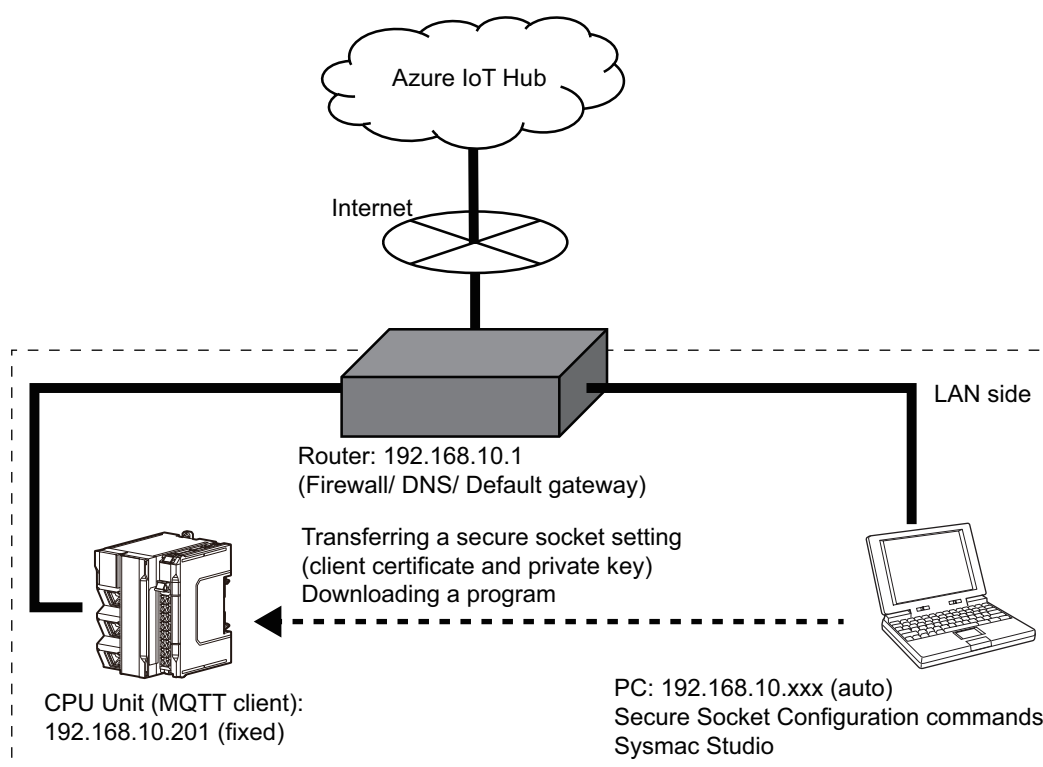


- (5)FB/FUN name
- (6)Name space
- (9)FB/FUN number
- (10)FB/FUN comment

A-2 Procedure for Connection to Azure IoT Hub via MQTT

In the configuration in the following figure, the procedure shown here allows the CPU Unit as an MQTT Publisher to send data to the Azure IoT Hub and the CPU Unit as an MQTT Subscriber to receive the data from the Azure IoT Hub.

The MQTT over TLS protocol encrypts data, so it is sent and received safely.



Additional Information

As of May 2021, the Azure IoT Hub is not a general-purpose messaging broker between a publisher and a subscriber.

The Azure IoT Hub cannot send to a subscriber the message that a publisher sent to the Azure IoT Hub.

For details, refer to the manual of the Azure IoT Hub.

A-2-1 Overall Procedure

In the context of the procedure shown here, when you connect to the Azure IoT Hub, there are three methods below for the Azure IoT Hub authenticating a device (MQTT client).

Type of authentication	Description
Symmetric key	When the CPU Unit communicates with the Azure IoT Hub, it is authenticated by using a user name and a password (SAS token).

Type of authentication	Description
X.509 self-signed	When the CPU Unit communicates with the Azure IoT Hub, it is authenticated by using the X.509 certificate signed by your company or yourself instead of a root certification authority.
X.509 CA-signed	When the CPU Unit communicates with the Azure IoT Hub, it is authenticated by using the X.509 certificate purchased from a root certification authority.

These authentication types are linked with individual devices that connect with the Azure IoT Hub; in creating a device on the Azure IoT Hub, the customer specifies the type.

The authentication type affects the setting procedure with Secure Socket Configuration commands on the PC and the parameters in a program on the Sysmac Studio.

This section shows procedures for all the authentication types; however, if any of them is unnecessary, skip that procedure.

Refer to *A-2-6 Secure Socket Setting for CPU Unit* on page A-11 for the setting procedure with Secure Socket Configuration commands, and refer to *Creation of Program* on page A-15 for a program on the Sysmac Studio.

The overall procedure shown here is as follows.

- 1** Preliminary Preparations on Azure Side
Refer to *A-2-3 Preliminary Preparations on Azure Side* on page A-8.
- 2** Network Setting for PC
Refer to *A-2-4 Network Setting for PC* on page A-10.
- 3** Network Setting for CPU Unit
Refer to *A-2-5 Network Setting for CPU Unit* on page A-10.
- 4** Secure Socket Setting for CPU Unit
Refer to *A-2-6 Secure Socket Setting for CPU Unit* on page A-11.
- 5** Creation of Program and Execution of Program
Refer to *A-2-7 Creation of Program and Execution of Program* on page A-15.



Additional Information

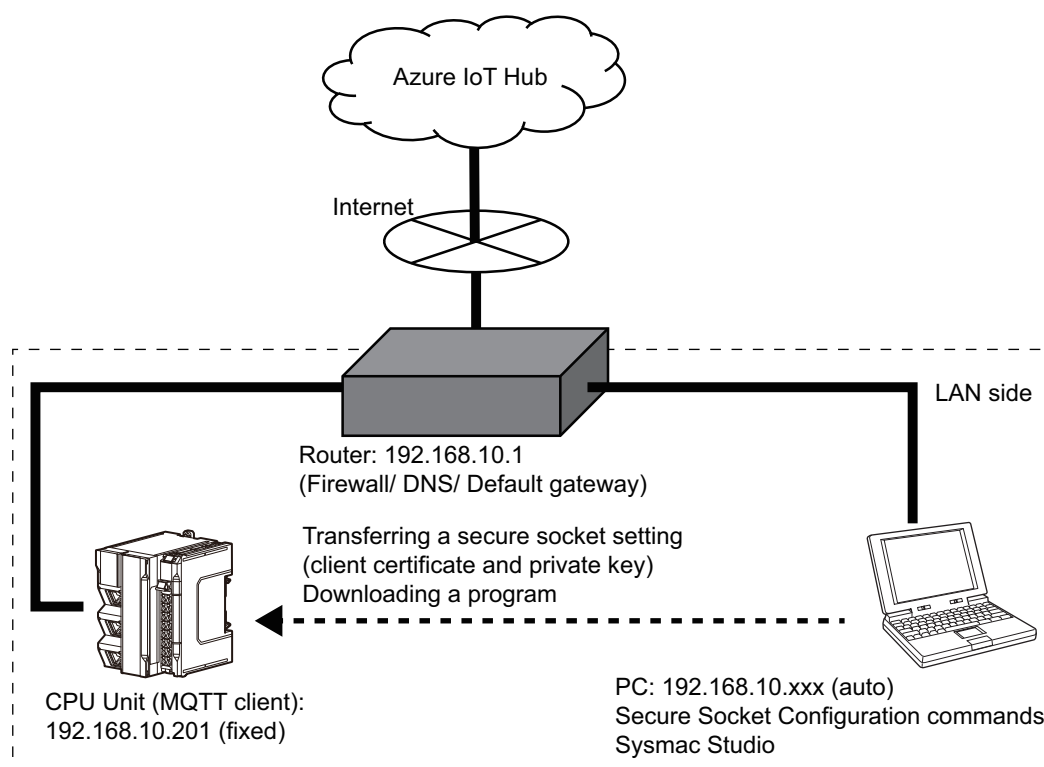
You need to be careful in deciding on an authentication type, considering the operational costs and following the security policy of your company.

Refer to the following Azure website etc.

Security practices for Azure IoT device manufacturers (<https://docs.microsoft.com/en-us/azure/iot-dps/concepts-device-oem-security-practices>)

A-2-2 System Configuration and Properties of Configuration Elements

The configuration shown below is used to explain the procedure.



Precautions for Correct Use

- When you build an intranet via global address, because of network security issues, consider using a VPN and installing a firewall after consulting with a specialist well-versed in the network field.
- Depending on the firewall setting made by a carrier, some communication applications cannot be used. If there are any communication applications that cannot be used, be sure to check with the carrier.

List and Properties of Configuration Elements

The list and properties of configuration elements are shown in the table below.

Config-uration element	Property	Value	Description
Azure IoT Hub	This is an IoT Hub to be made on the Azure. It fills the role of an MQTT broker.		
	IoT Hub name	testiothubnx1	Refer to <i>A-2-3 Preliminary Preparations on Azure Side</i> on page A-8.
	IoT Hub host name	testiothubnx1.azuredevi-ces.net	
	Port number	8883	This is a port number that is used when you connect to the IoT Hub.
	Network – Connection Method	Public endpoint	All networks come to have the right to access this IoT hub.
	Shared access policy	Enabling all the Permission items of iothubowner	Refer to <i>A-2-3 Preliminary Preparations on Azure Side</i> on page A-8.
Router	This is a router that has a global IP address. Here, it also has functions of Firewall, DNS, and default gateway.		

Config-uration element	Property	Value	Description
	IP address	192.168.10.1	This is an IP address on the LAN side of the router. For details, read the manual of the router.
	Subnet mask	255.255.255.0*1	This is a subnet mask on the LAN side of the router. For details, read the manual of the router.
CPU Unit (MQTT client)	This fills the role of a Publisher or a Subscriber. In the Azure IoT Hub, it is written as a device.		
	IP address – Port 1	192.168.10.201 (fixed)	Refer to <i>A-2-5 Network Setting for CPU Unit</i> on page A-10.
	Subnet mask – Port 1	255.255.255.0	
	DNS address	192.168.10.1	
	Default gateway	192.168.10.1	
	Device ID	This procedure changes a device ID for description, according to the device authentication type. Refer to <i>Properties of Devices</i> on page A-8.	
PC (Sysmac Studio)	This downloads settings and programs to the CPU Unit, by means of the Sysmac Studio.		
	IP address	192.168.10.xxx (auto)	This depends on the IP address auto-assignment setting made by the DHCP function of the router. As with the CPU Unit, it is also possible to set the fixed IP address that does not overlap with other devices or PCs.

*1. For bit specification, it is 24.

Properties of Devices

This section describes the properties of devices for each authentication type.

Type of authentication*1	Device ID	Session ID*2	Session name*3	Other properties*4
Symmetric key	testdevicesas	0	TLSSession0	SAS token
X.509 self-signed	testdeviceself	1	TLSSession1	Self-signed certificate file and private key file for self-signed certificate
X.509 CA-signed	testdeviceca	2	TLSSession2	CA-signed certificate file and private key file for CA-signed certificate

*1. Refer to the required items, according to the device authentication type to select.

*2. You can use any values from 0 to 59. Here, values in the table are used.

*3. This indicates TLSSession <Session ID>.

*4. Refer to *A-2-3 Preliminary Preparations on Azure Side* on page A-8.

A-2-3 Preliminary Preparations on Azure Side

This section describes the preparations required to connect the CPU Unit to the Azure IoT Hub.

Obtaining the required items on the Azure IoT Hub side requires the following preparations on the Azure side.

- Creating an Azure account (<https://azure.microsoft.com/>)
- Creating an Azure IoT Hub
- Creating a shared access policy

The easiest way to create a shared access policy is to use `iothubowner`, which sets an extensive access policy, and check off all the Permission items. However, when you use it actually, create such a policy as allows the necessary minimum.

- Creating a device

Note that creating a device requires the following preparations in advance according to the device authentication type.

- When the device authentication type is "symmetric key", you need to create a SAS token.
In the method of using the Azure CLI, `az iot hub generate-sas-token` can be used.
- When the device authentication type is "X.509 self-signed", you need to prepare a self-signed certificate.
Open SSL can be used for creation of the self-signed certificate. For details, read the manual of Open SSL.
- When the device authentication type is "X.509 CA-signed", you need to prepare a CA-signed certificate.
After drawing up a certificate signing request (CSR), you need such a procedure as submitting the CSR to a certification authority. For details, search the Internet etc.

The required items on the Azure IoT Hub side are shown in the table below.

Item	Description
IoT Hub name	This is a name of the IoT Hub to be specified when you create an Azure IoT Hub. You need to set a name unique to the whole Azure. Here, "testiothubnx1" is used.
IoT Hub host name	This is a host name of the IoT Hub to be settled automatically on the basis of the IoT Hub name. In this procedure, it is used as a host name etc. of the MQTT broker that the MQTT client connects to. Here, "testiothubnx1.azure-devices.net" is used.
Device ID	This is a name of the device (MQTT client) to be specified when you create a device on the IoT Hub. When the MQTT client connects to the Azure IoT Hub, it is used as a client identifier to specify etc. This procedure uses a different device ID for description, according to the device authentication type. Here, "testdevicesas", "testdeviceself", or "testdeviceca" is used.
SAS token (character string)	This item is required when the device authentication type is "symmetric key". In this procedure, it is used as a password when the MQTT client connects to the Azure IoT Hub. Here, "SharedAccessSignature sr=testiothubnx1.azuredevices.net%2Fdevices%2Ftestdevicesas&sig=%2B***%3D&se=****", which was created by using the Azure CLI, is used.
Self-signed certificate file and private key file for self-signed certificate	These are required when the device authentication type is "X.509 self-signed". The files are in PEM format. Here, "device.crt.pem" and "device.key.pem" are used.

Item	Description
CA-signed certificate file and private key file for CA-signed certificate	These are required when the device authentication type is "X.509 CA-signed". The files are in PEM format. Here, "device.crt.pem" and "device.key.pem" are used.

A-2-4 Network Setting for PC

For the PC, a special network setting is not necessary as long as it is connected with that router and set to automatically obtain the IP address and the DNS server address by means of the DHCP function.

When you manually set the IP address and DNS server for the PC, apply the same setting as that in *A-2-5 Network Setting for CPU Unit* on page A-10 to also the PC.

A-2-5 Network Setting for CPU Unit

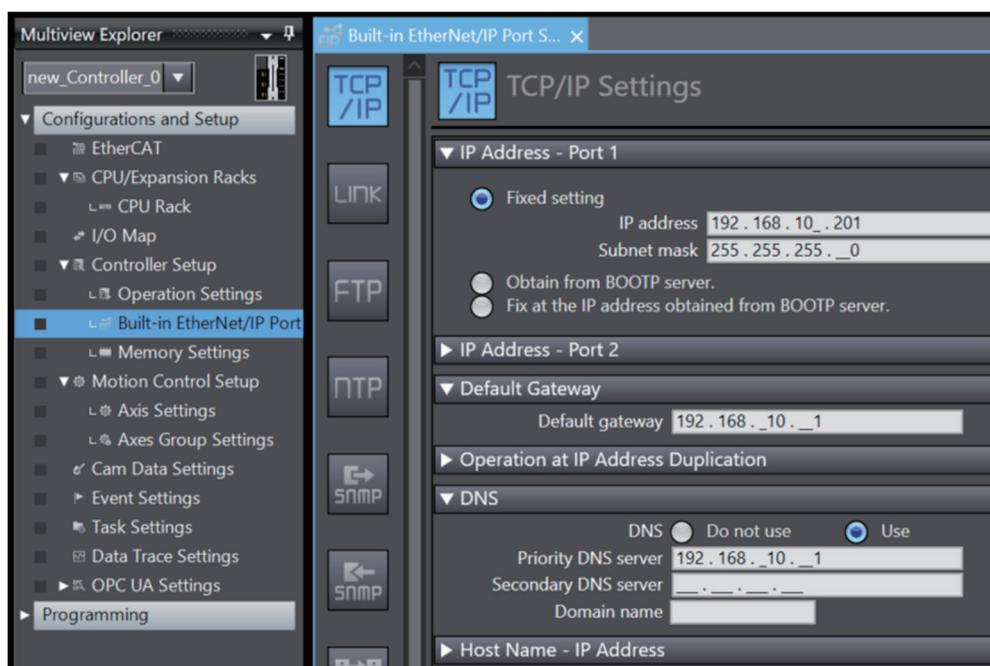


Precautions for Correct Use

After this setting is downloaded to the CPU Unit, third parties will be able to access the CPU Unit from the Internet.

Take security measures as required.

- 1 Start the Sysmac Studio, and select **Multiview Explorer – Configurations and Setup – Controller Setup – Built-in EtherNet/IP Port – TCP/IP**.
- 2 Input the setting as follows.



Item	Value	Description
IP Address – Port 1 – IP address	192.168.10.201	In order to prevent the IP address of the CPU Unit from being changed by the DHCP function of the router, set the fixed IP address so that it will be assigned also on the router side. Set the IP address that does not overlap with other devices or PCs.
IP Address – Port 1 – Subnet mask	255.255.255.0	Set the same value as the setting of a subnet mask on the LAN side of the router.
Default Gateway – Default gateway	192.168.10.1	Specify an IP address on the LAN side of the router.
DNS	Use	
DNS – Priority DNS server	192.168.10.1	Specify an IP address on the LAN side of the router. Specify the IP address of the other DNS server as required.

3 Download the setting to the CPU Unit.

Use a ping or other commands to check that the IP address of the port 1 of the CPU Unit can be accessed from the PC.

A-2-6 Secure Socket Setting for CPU Unit

Here, use Secure Socket Configuration commands on the PC to set and transfer the information required for secure socket communications to the CPU Unit.



Precautions for Correct Use

- If the client certificate, private key, or secure socket setting related to secure socket communications is stolen, leaked, or falsified by any third party, this may result in such network security issues as data on the server is acquired illegally, data on the server is falsified, and communication with the server is disabled.
The customer must manage the client certificate, private key, and secure socket setting, and take measures to prevent them from being stolen, leaked, or falsified.
Especially when you acquire the private key, be careful not to leak it by using an encrypted safe communication path or the like.
Methods for managing the private key include storing it in a location where the possibility of leakage is eliminated as much as possible.
- In order to reduce the risk of unauthorized access by a third party using the Secure Socket Configuration commands, consider setting operation authority verification to the CPU Unit.



Additional Information

- "<Sysmac Studio Installed Folder>" shown in a command to be executed is a folder in which to install the Sysmac Studio, which is by default as follows.
 - a) For the Sysmac Studio with a 32-bit version
For the OS with a 32-bit version, C:\Program Files\OMRON\Sysmac Studio
For the OS with a 64-bit version, C:\Program Files (x86)\OMRON\Sysmac Studio
 - b) For the Sysmac Studio with a 64-bit version
C:\Program Files\OMRON\Sysmac Studio
- If operation authority verification is set in the CPU Unit, the following will appear when you execute a command, and you will need to enter a password.

```
Operation authority: Administrator
Password: |
```

Entering the correct password will execute the command.

- Refer to the *NJ/NX-series CPU Unit Built-in EtherNet/IP™ Port User's Manual (Cat. No. W506)* for detailed usage method and troubleshooting of the Secure Socket Configuration commands.
-

- 1 Check that the operating mode of the CPU Unit is PROGRAM mode.
- 2 Perform the following steps to start the command prompt.
 - 1) Click **OMRON – Sysmac Studio** from the Windows Start Menu.
A folder is displayed.
 - 2) Double-click a shortcut to `tlsconfig` in the folder.
The command prompt is started.
- 3 Make a secure socket setting of the device according to the device authentication type.
 - Refer to *For Device with Device Authentication Type of "Symmetric Key"* on page A-13 for the device with a device authentication type of "symmetric key".
 - Refer to *For Device with Device Authentication Type of "X.509 Self-Signed"* on page A-13 for the device with a device authentication type of "X.509 self-signed".
 - Refer to *For Device with Device Authentication Type of "X.509 CA-Signed"* on page A-14 for the device with a device authentication type of "X.509 CA-signed".

This concludes the secure socket setting for the CPU Unit.

The CPU Unit does not need restarting or the like.

If you make a wrong setting, you can delete the setting by using the following command.

- To delete the secure socket setting for the specified session ID

```
<Sysmac Studio Installed Folder>\TLSSettingTool>tlsconfig delSessionInfo /id <Session ID> /ip:192.168.10.201
```

- To delete the secure socket setting for all the session IDs

```
<Sysmac Studio Installed Folder>\TLSSettingTool>tlsconfig delAllSessionInfo /ip:192.168.10.201
```

In addition, when an error occurs during communications, you can check a log of the error if you executed the following command in advance.

```
<Sysmac Studio Installed Folder>\TLSSettingTool>tlsconfig setLogLevel /enable /ip:192.168.10.201
```

For Device with Device Authentication Type of "Symmetric Key"

- 1 Execute the following command to inform the CPU Unit that the session ID 0 does not use a client certificate.

```
<Sysmac Studio Installed Folder>\TLSSettingTool>tlsconfig setSessionInfo /id
0 /ip:192.168.10.201 /f
```

000: Success is displayed, and the command comes to a normal end. The secure socket setting for the session ID 0 ends normally.

If any error number other than **000: Success** is displayed, refer to the error number of that command in *A-2-5 Network Setting for CPU Unit* on page A-10 and *NJ/NX-series CPU Unit Built-in EtherNet/IP™ Port User's Manual (Cat. No. W506)* to take action.

- 2 Execute the following command to check that the setting is made in the CPU Unit correctly.

```
<Sysmac Studio Installed Folder>\TLSSettingTool>tlsconfig getAllSessionInfo
/ip:192.168.10.201
```

The following results are displayed.

The setting is made correctly if the private key file name and the client certificate file name are **none**.

```
Count=1
Id=0
PrivateKey=none
Certificate=none
Description=

000:Success
```

For Device with Device Authentication Type of "X.509 Self-Signed"

Commands and results in this example are based on the case where the self-signed certificate file is in "C:\certs\self-signed.device.cert.pem". In the same way, they are based on the case where the private key file is in "C:\private\self-signed.device.key.pem".

Correct the values of /key and /cert, according to the file locations.

- 1 Execute the following command to set a private key file and a self-signed certificate file for the session ID 1 to the CPU Unit.

```
<Sysmac Studio Installed Folder>\TLSSettingTool>tlsconfig setSessionInfo /id
1 /key "C:\private\self-signed.device.key.pem" /cert "C:\certs\self-signed.
device.cert.pem" /ip:192.168.10.201 /f
```

000: Success is displayed, and the private key file and the self-signed certificate file are transferred to the session ID 1.

If any error number other than **000: Success** is displayed, refer to the error number of that command in *A-2-5 Network Setting for CPU Unit* on page A-10 and *NJ/NX-series CPU Unit Built-in EtherNet/IP™ Port User's Manual (Cat. No. W506)* to take action.

2 Execute the following command to check that the setting is made in the CPU Unit correctly.

```
<Sysmac Studio Installed Folder>\TLSSettingTool>tlsconfig getAllSessionInfo /ip:192.168.10.201
```

The following results are displayed.

The setting is made correctly if the private key file name and the self-signed certificate file name are consistent with those that were set.

```
Count=1
Id=1
PrivateKey=self-signed.device.key.pem
Certificate=self-signed.device.cert.pem
Description=

000:Success
```

For Device with Device Authentication Type of "X.509 CA-Signed"

Commands and results in this example are based on the case where the CA-signed certificate file is in "C:\certs\ca-signed.device.cert.pem". In the same way, they are based on the case where the private key file is in "C:\private\ca-signed.device.key.pem".

Correct the values of /cert and /key, according to the file locations.

1 Execute the following command to set a private key file and a CA-signed certificate file for the session ID 2 to the CPU Unit.

```
<Sysmac Studio Installed Folder>\TLSSettingTool>tlsconfig setSessionInfo /id 2 /key "C:\private\ca-signed.device.key.pem" /cert "C:\certs\ca-signed.device.cert.pem" /ip:192.168.10.201 /f
```

000: Success is displayed, and the private key file and the CA-signed certificate file are transferred to the session ID 2.

If any error number other than **000: Success** is displayed, refer to the error number of that command in *A-2-5 Network Setting for CPU Unit* on page A-10 and *NJ/NX-series CPU Unit Built-in EtherNet/IP™ Port User's Manual (Cat. No. W506)* to take action.

2 Execute the following command to check that the setting is made in the CPU Unit correctly.

```
<Sysmac Studio Installed Folder>\TLSSettingTool>tlsconfig getAllSessionInfo /ip:192.168.10.201
```

The following results are displayed.

The setting is made correctly if the private key file name and the CA-signed certificate file name are consistent with those that were set.

```
Count=1
Id=2
PrivateKey=ca-signed.device.key.pem
Certificate=ca-signed.device.cert.pem
Description=

000:Success
```

A-2-7 Creation of Program and Execution of Program

Create a program on the PC, and operate the program actually on the CPU Unit.

The program enables the following:

- Checking communication with the Azure IoT Hub
- As an MQTT Publisher, the CPU Unit sending a message to the Azure IoT Hub
- As an MQTT Subscriber, the CPU Unit receiving the message from the Azure IoT Hub

Adding MQTT Communications Library to Sysmac Studio

Following the procedure in *Section 1 Sysmac Library Usage Procedure* on page 1-1, add the MQTT Communications Library to the Sysmac Studio, and make it ready to be used by a program.

Creation of Program

Before use, rewrite <***> in the following program according to the environment.

After the completion of rewriting, download it to the CPU Unit.



Precautions for Correct Use

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

● ST

Internal variable	Name	Data type	Default
	Connect	BOOL	FALSE
	Ping	BOOL	FALSE
	Publish	BOOL	FALSE
	Subscribe	BOOL	FALSE
	ins_MQTTClient	OmronLib\MQTT_Comm \MQTTClient	
	ins_MQTTPing	OmronLib\MQTT_Comm \MQTTPing	
	ins_MQTTPubString	OmronLib\MQTT_Comm \MQTTPubString	
	ins_MQTTSubString	OmronLib\MQTT_Comm \MQTTSubString	
	ClientReference	OmronLib\MQTT_Comm \sClientReference	
	ClientID	STRING[256]	
	ConnectionSettings	OmronLib\MQTT_Comm \sConnectionSettings	
	KeepAlive	UINT	60

Internal variable	Name	Data type	Default
	Timeout	UINT	10
	DiscardMsgTime	UINT	1000
	Connected	BOOL	
	Busy	BOOL	
	DiscardMsgNum	UDINT	
	SessionPresent	BOOL	
	Error	BOOL	
	ErrorID	WORD	
	ErrorIDEx	DWORD	
	Png_Timeout	UINT	1000
	Png_Done	BOOL	
	Png_Busy	BOOL	
	Png_ElapseTime	UINT	
	Png_Error	BOOL	
	Png_ErrorID	WORD	
	Png_ErrorIDEx	DWORD	
	Pub_PacketID	UINT	
	Pub_MsgType	USINT	
	Pub_PubMsg	STRING[256]	
	Pub_PubSettings	OmronLib\MQTT_Comm lsPubFlags	
	Pub_Topic	STRING[512]	
	Pub_Timeout	UINT	6
	Pub_Done	BOOL	
	Pub_Busy	BOOL	
	Pub_Error	BOOL	
	Pub_ErrorID	WORD	
	Pub_ErrorIDEx	DWORD	
	Sub_RcvMsg	STRING[512]	
	Sub_SubQoS	BYTE	16#0
	Sub_Topic	STRING[512]	
	Sub_Timeout	UINT	10
	Sub_Subscribed	BOOL	
	Sub_Status	SINT	
	Sub_Received	BOOL	
	Sub_RcvTopic	STRING[512]	
	Sub_Error	BOOL	
	Sub_ErrorID	WORD	
	Sub_ErrorIDEx	DWORD	

// The notations [SAS], [Self-signed], and [CA-signed] in the comments refer to the authentication type of device selected by you on Azure.

```
IF P_First_RunMode THEN
  // <Device ID> that you input for creating the device on Azure settings.
  // e.g. [SAS]'testdevicesas'
  //      [Self-signed]'testdeviceself'
```



```

//      [CA-signed]'testdeviceca'
ClientID := '<Device ID>';
// <IoT Hub host name> that you can check on Azure.
// e.g. 'testiothubnx1.azure-devices.net'
ConnectionSettings.IpAdr := '<IoT Hub host name>.azure-devices.net';
ConnectionSettings.PortNo := 8883;
ConnectionSettings.TLSUse := True;
// TLSession<Session ID>, <Session ID> is number that you input on secure soc
ket settings.
// e.g. [SAS]'TLSession0'
//      [Self-signed]'TLSession1'
//      [CA-signed]'TLSession2'
ConnectionSettings.TLSSessionName := 'TLSession<Session ID>';
// '<IoT Hub host name>.azure-devices.net/<Device ID>/?api-version=2018-06-30'
,
// e.g. [SAS]'testiothubnx1.azure-devices.net/testdevicesas/?api-version=2018
-06-30'
//      [Self-signed]'testiothubnx1.azure-devices.net/testdeviceself/?api-ver
sion=2018-06-30'
//      [CA-signed]'testiothubnx1.azure-devices.net/testdeviceca/?api-version
=2018-06-30'
ConnectionSettings.UserName := '<IoT Hub host name>.azure-devices.net/<Device
ID>/?api-version=2018-06-30';
// [SAS]Password is '<SAS token string>'
//      e.g. SharedAccessSignature sr=testiothubnx1.azure-devices.net%2Fdevice
s%2Ftestdevicesas&sig=%2Bxxx%3D&se=xxx
// [Self-signed] [CA-signed]Password is '' (empty)
ConnectionSettings.Password := '';
ConnectionSettings.CleanSession := True;
// 'devices/<Device ID>/messages/events/'
// e.g. [SAS]'devices/testdevicesas/messages/events/'
//      [Self-signed]'devices/testdeviceself/messages/events/'
//      [CA-signed]'devices/testdeviceca/messages/events/'
Pub_Topic := 'devices/<Device ID>/messages/events/';
Pub_PubMsg := 'Hello!';
Pub_PubSettings.PubQoS := 16#1;
Pub_PubSettings.RetainFlag := False;
// 'devices/<Device ID>/messages/devicebound/#'
// e.g. [SAS]'devices/testdevicesas/messages/devicebound/#'
//      [Self-signed]'devices/testdeviceself/messages/devicebound/#'
//      [CA-signed]'devices/testdeviceca/messages/devicebound/#'
Sub_Topic := 'devices/<Device ID>/messages/devicebound/#';
END_IF;

ins_MQTTClient(
    Enable := Connect,
    ClientReference := ClientReference,

```

```

ClientID := ClientID,
ConnectionSettings := ConnectionSettings,
KeepAlive := KeepAlive,
Timeout := Timeout,
DiscardMsgTime := DiscardMsgTime,
Connected => Connected,
Busy => Busy,
DiscardMsgNum => DiscardMsgNum,
SessionPresent => SessionPresent,
Error => Error,
ErrorID => ErrorID,
ErrorIDEx => ErrorIDEx);

ins_MQTTPing(
    Execute := ins_MQTTClient.Connected AND Ping,
    ClientReference := ClientReference,
    Timeout := Png_Timeout,
    Done => Png_Done,
    Busy => Png_Busy,
    ElapseTime => Png_ElapseTime,
    Error => Png_Error,
    ErrorID => Png_ErrorID,
    ErrorIDEx => Png_ErrorIDEx);

ins_MQTTPubString(
    Execute := ins_MQTTClient.Connected AND Publish,
    ClientReference := ClientReference,
    PacketID := Pub_PacketID,
    MsgType := Pub_MsgType,
    PubMsg := Pub_PubMsg,
    PubSettings := Pub_PubSettings,
    Topic := Pub_Topic,
    Timeout := Pub_Timeout,
    Done => Pub_Done,
    Busy => Pub_Busy,
    Error => Pub_Error,
    ErrorID => Pub_ErrorID,
    ErrorIDEx => Pub_ErrorIDEx);

ins_MQTTSubString(
    Enable := ins_MQTTClient.Connected AND Subscribe,
    ClientReference := ClientReference,
    RcvMsg := Sub_RcvMsg,
    SubQoS := Sub_SubQoS,
    Topic := Sub_Topic,
    Timeout := Sub_Timeout,
    Subscribed => Sub_Subscribed,

```

```

Status => Sub_Status,
Received => Sub_Received,
RcvTopic => Sub_RcvTopic,
Error => Sub_Error,
ErrorID => Sub_ErrorID,
ErrorIDEx => Sub_ErrorIDEx);

```

Checking Communication to Azure IoT Hub

Check that the CPU Unit sends PING (*PINGREQ* packet) to the Azure IoT Hub and that the CPU Unit can receive the response (*PINGRESP* packet) from the Azure IoT Hub.

- 1** Change the CPU Unit into RUN mode.
- 2** Set the internal variable Connect to TRUE to connect to the IoT Hub.
ins_MQTTClient.Connected changes to TRUE, and you can check the connection to the Azure IoT Hub.
If it does not change to TRUE, check and correct the parameters in *A-2-5 Network Setting for CPU Unit* on page A-10, *A-2-6 Secure Socket Setting for CPU Unit* on page A-11, and *Creation of Program* on page A-15. Or, refer to the *MQTTClient* on page 4-2 (MQTT client) instruction to take action.
- 3** Set the internal variable Ping to TRUE to send and receive PING.
ins_MQTTPing.Done changes to TRUE, and you can check that the PING send/receive process ended normally.
If it does not change to TRUE, refer to the *MQTTPing* on page 4-52 (MQTTPing message send) instruction to take action.
- 4** Set the internal variable Ping to FALSE to cancel the PING send/receive ready state.
- 5** Set the internal variable Connect to FALSE to cancel the connection to the Azure IoT Hub.
ins_MQTTClient.Done changes to FALSE, and you can check that the connection to the Azure IoT Hub was canceled.

Checking Operation as Publisher

Check that the CPU Unit as an MQTT Publisher sends a message to the Azure IoT Hub and that the Azure IoT Hub receives the message.

- 1** Change the CPU Unit into RUN mode.
- 2** Set the internal variable Connect to TRUE to connect to the Azure IoT Hub.
ins_MQTTClient.Connected changes to TRUE, and you can check the connection to the Azure IoT Hub.
If it does not change to TRUE, check and correct the parameters in *A-2-5 Network Setting for CPU Unit* on page A-10, *A-2-6 Secure Socket Setting for CPU Unit* on page A-11, and *Creation*

of *Program* on page A-15. Or, refer to the *MQTTClient* on page 4-2 (MQTT client) instruction to take action.

- 3** Set the internal variable Publish to TRUE to publish a message.
ins_MQTTPubString.Done changes to TRUE, and you can check that the message was published.
If it does not change to TRUE, refer to the *MQTTPubString* on page 4-34 (MQTT character string message publication) instruction to take action.
- 4** Use the Service Bus Explorer^{*1} etc. to check that the message 'Hello!' was sent to the Azure IoT Hub.
- 5** Set the internal variable Publish to FALSE to cancel the publication ready state.
- 6** Set the internal variable Connect to FALSE to cancel the connection to the Azure IoT Hub.
ins_MQTTClient.Done changes to FALSE, and you can check that the connection to the Azure IoT Hub was canceled.

*1. For the Service Bus Explorer, refer to <https://github.com/paolosavatori/ServiceBusExplorer/releases>.
When you use this tool for checking, you need Primary connection string of the shared access policy that was created in A-2-3 *Preliminary Preparations on Azure Side* on page A-8.

Checking Operation as Subscriber

Register a topic that the CPU Unit as an MQTT Subscriber subscribes to in the Azure IoT Hub.
Check that the Azure IoT Hub sends a message to the CPU Unit and that the CPU Unit receives the message.

- 1** Change the CPU Unit into RUN mode.
- 2** Set the internal variable Connect to TRUE to connect to the Azure IoT Hub.
ins_MQTTClient.Connected changes to TRUE, and you can check the connection to the Azure IoT Hub.
If it does not change to TRUE, check and correct the parameters in A-2-5 *Network Setting for CPU Unit* on page A-10, A-2-6 *Secure Socket Setting for CPU Unit* on page A-11, and *Creation of Program* on page A-15. Or, refer to the *MQTTClient* on page 4-2 (MQTT client) instruction to take action.
- 3** Set the internal variable Subscribe to TRUE to make ready to subscribe to and receive a message.
The internal variable Sub_Status value changes to 2 (Subscription in progress).
If it does not change to 2, refer to the *MQTTSubString* on page 4-48 (MQTT character string message subscription request) instruction to take action.
- 4** Send a message from the Azure to that device (CPU Unit).
You can acquire the topic name specified by the Azure IoT Hub to the internal variable Sub_RcvTopic, and the message sent from the Azure to Sub_RcvMsg.

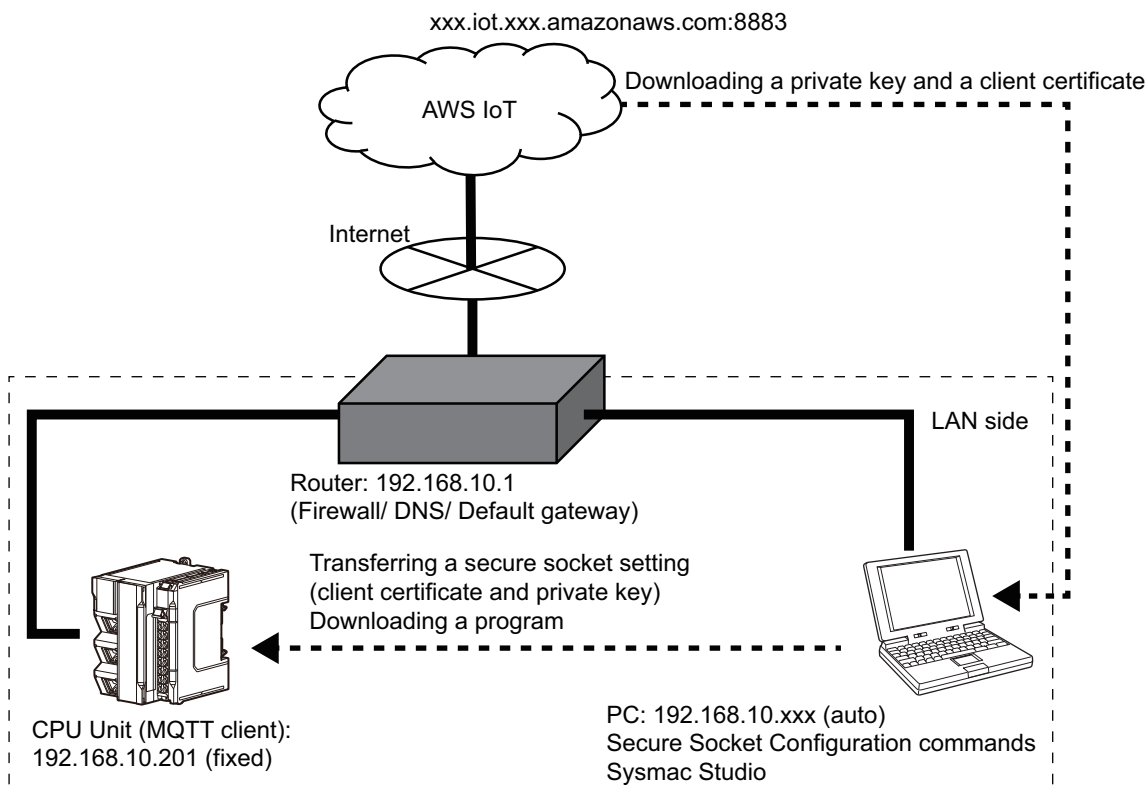
If you cannot acquire the message, refer to the *MQTTSubString* on page 4-48 (MQTT character string message subscription request) instruction to take action.

- 5** Set the internal variable `Subscribe` to `FALSE` to cancel the subscription ready state. The internal variable `Sub_Status` value changes to `0` (Execution stopped), and you can check that the state of subscription in progress was canceled.
- 6** Set the internal variable `Connect` to `FALSE` to cancel the connection to the Azure IoT Hub. `ins_MQTTClient.Connected` changes to `FALSE`, and you can check that the connection to the Azure IoT Hub was canceled.

A-3 Procedure for Connection to AWS IoT via MQTT

In the configuration in the following figure, the procedure shown here allows the CPU Unit as an MQTT Publisher to send data to the AWS IoT and the CPU Unit as an MQTT Subscriber to receive the data from the AWS IoT.

The MQTT over TLS protocol encrypts data, so it is sent and received safely.



A-3-1 Overall Procedure

The overall procedure shown here is as follows.

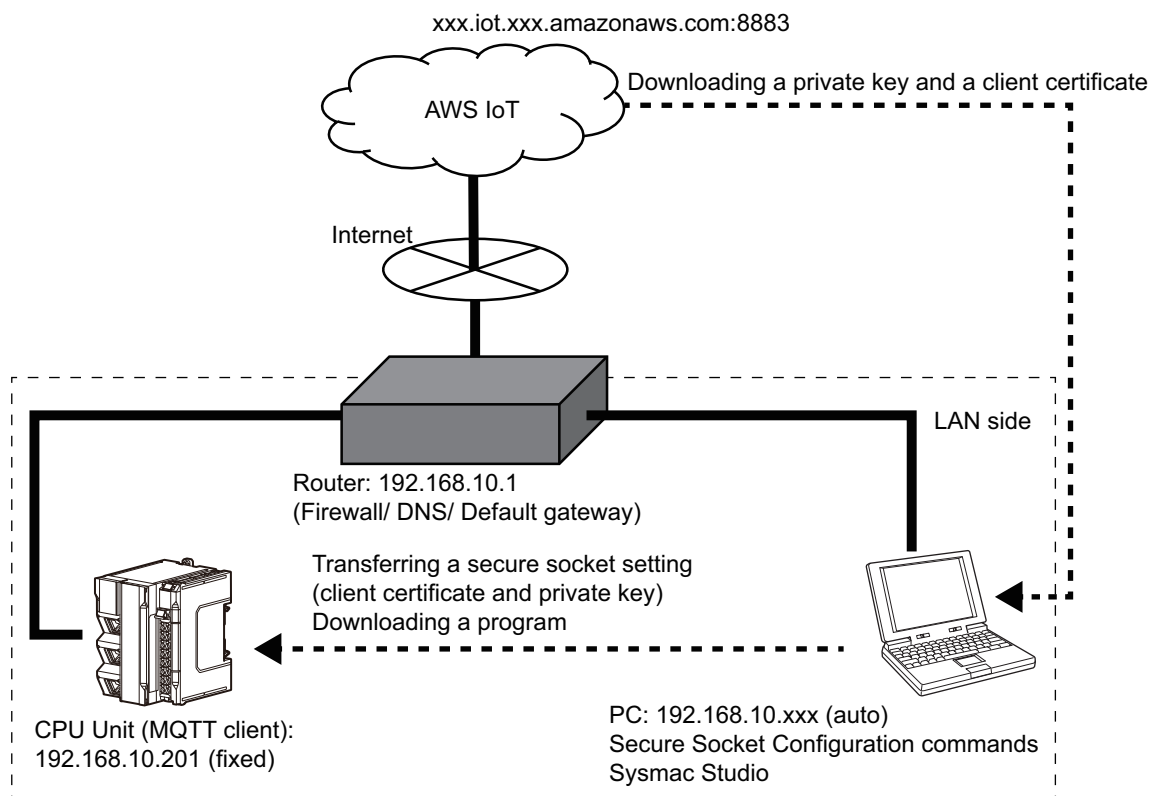
- 1** Preliminary Preparations on AWS Side
Refer to *A-3-3 Preliminary Preparations on AWS Side* on page A-24.
- 2** Network Setting for PC
Refer to *A-3-4 Network Setting for PC* on page A-25.
- 3** Network Setting for CPU Unit
Refer to *A-3-5 Network Setting for CPU Unit* on page A-25.
- 4** Secure Socket Setting for CPU Unit
Refer to *A-3-6 Secure Socket Setting for CPU Unit* on page A-26.

5 Creation of Program and Execution of Program

Refer to A-3-7 *Creation of Program and Execution of Program* on page A-28.

A-3-2 System Configuration and Properties of Configuration Elements

The configuration shown below is used to explain the procedure.



Precautions for Correct Use

- When you build an intranet via global address, because of network security issues, consider using a VPN and installing a firewall after consulting with a specialist well-versed in the network field.
- Depending on the firewall setting made by a carrier, some communication applications cannot be used. If there are any communication applications that cannot be used, be sure to check with the carrier.

List and Properties of Configuration Elements

The list and properties of configuration elements are shown in the table below.

Configuration element	Property	Value	Description
AWS IoT	This is an AWS IoT service to be made on the AWS. It fills the role of an MQTT broker.		
	Endpoint	xxx.iot.xxx.amazonaws.com	Refer to A-3-3 <i>Preliminary Preparations on AWS Side</i> on page A-24.
	Port number	8883	This is a port number that is used when you connect to the AWS IoT.

Configuration element	Property	Value	Description
Router	This is a router that has a global IP address. Here, it also has functions of Firewall, DNS, and default gateway.		
	IP address	192.168.10.1	This is an IP address on the LAN side of the router. For details, read the manual of the router.
	Subnet mask	255.255.255.0*1	This is a subnet mask on the LAN side of the router. For details, read the manual of the router.
CPU Unit (MQTT client)	This fills the role of a Publisher or a Subscriber. In the AWS IoT, it is written as a thing.		
	IP address – Port 1	192.168.10.201 (fixed)	Refer to <i>A-3-5 Network Setting for CPU Unit</i> on page A-25.
	Subnet mask – Port 1	255.255.255.0	
	DNS address	192.168.10.1	
	Default gateway	192.168.10.1	
PC (Sysmac Studio)	This downloads settings and programs to the CPU Unit, by means of the Sysmac Studio.		
	IP address	192.168.10.xxx (auto)	This depends on the IP address auto-assignment setting made by the DHCP function of the router. As with the CPU Unit, it is also possible to set the fixed IP address that does not overlap with other devices or PCs.

*1. For bit specification, it is 24.

A-3-3 Preliminary Preparations on AWS Side

This section describes the preparations required to connect the CPU Unit to the AWS IoT.

Obtaining the required items on the AWS IoT side requires the following preparations on the AWS side.

- Creating an AWS account (<https://portal.aws.amazon.com/billing/signup#/start>)
- Creating a policy
The easiest way to create a policy is to set Action: "iot:*", Resource ARN: "*", and Effect: "Allow". However, when you use it actually, create such a policy as allows the necessary minimum.
- Creating a certificate and attaching the policy
A certificate can be created by **One-click certificate creation** on the AWS.
You can acquire the certificate, public key, and private key. Attach the policy to the acquired certificate.
- Registering things (as required)
Associating the policy with the certificate eliminates the need to register things. Registering things is useful for managing things individually or the like.

The required items on the AWS IoT side are shown in the table below.

Item	Description
Endpoint	This is the URL of the entry point for an AWS web service. You can check it from the setting of the AWS IoT. It is used as a host name etc. of the MQTT broker that the MQTT client connects to. Here, "xxx.iot.xxx.amazonaws.com" is used.
Client certificate	This is an X.509 client certificate file in PEM format. It is transferred to the CPU Unit by <i>A-3-6 Secure Socket Setting for CPU Unit</i> on page A-26. In this procedure, the client certificate file that can be created on the AWS and downloaded is used. Here, "xxx-certificate.pem.crt" is used.
Private key	This is a private key in PEM format. It is transferred to the CPU Unit by <i>A-3-6 Secure Socket Setting for CPU Unit</i> on page A-26. In this procedure, the private key file that can be created on the AWS and downloaded is used. Here, "xxx-private.pem.key" is used.

A-3-4 Network Setting for PC

For the PC, a special network setting is not necessary as long as it is connected with that router and set to automatically obtain the IP address and the DNS server address by means of the DHCP function.

When you manually set the IP address and DNS server for the PC, apply the same setting as that in *A-3-6 Secure Socket Setting for CPU Unit* on page A-26 to also the PC.

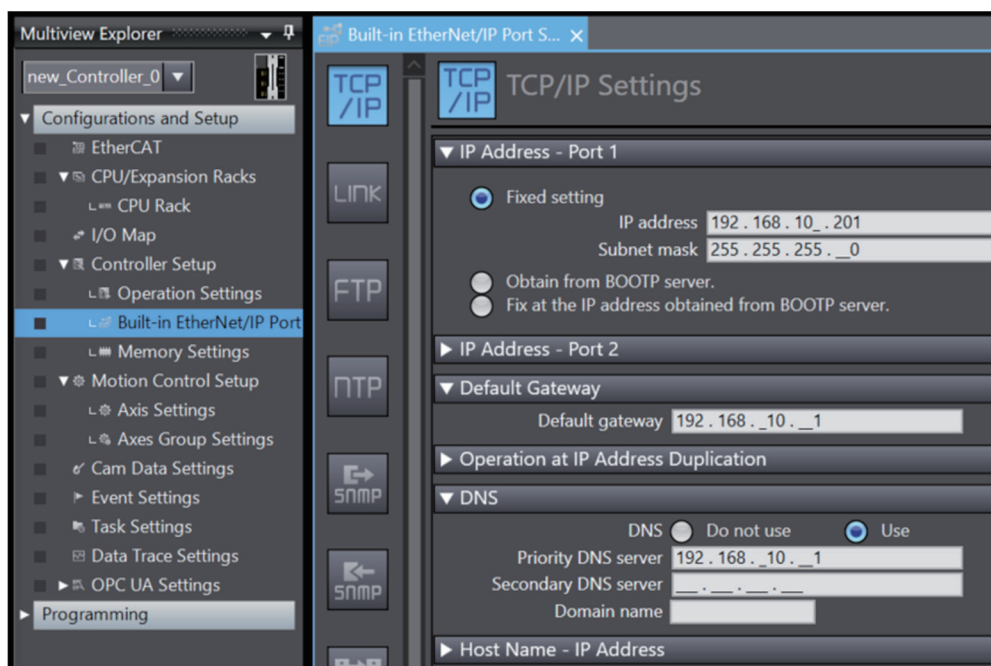
A-3-5 Network Setting for CPU Unit



Precautions for Correct Use

After this setting is downloaded to the CPU Unit, third parties will be able to access the CPU Unit from the Internet.
Take security measures as required.

- 1** Start the Sysmac Studio, and select **Multiview Explorer – Configurations and Setup – Controller Setup – Built-in EtherNet/IP Port – TCP/IP**.
- 2** Input the setting as follows.



Item	Value	Description
IP Address – Port 1 – IP address	192.168.10.201	In order to prevent the IP address of the CPU Unit from being changed by the DHCP function of the router, set the fixed IP address so that it will be assigned also on the router side. Set the IP address that does not overlap with other devices or PCs.
IP Address – Port 1 – Subnet mask	255.255.255.0	Set the same value as the setting of a subnet mask on the LAN side of the router.
Default Gateway – Default gateway	192.168.10.1	Specify an IP address on the LAN side of the router.
DNS	Use	
DNS – Priority DNS server	192.168.10.1	Specify an IP address on the LAN side of the router. Specify the IP address of the other DNS server as required.

3 Download the setting to the CPU Unit.

Use a ping or other commands to check that the IP address of the port 1 of the CPU Unit can be accessed from the PC.

A-3-6 Secure Socket Setting for CPU Unit

Here, use Secure Socket Configuration commands on the PC to set and transfer the information required for secure socket communications to the CPU Unit.



Precautions for Correct Use

- If the client certificate, private key, or secure socket setting related to secure socket communications is stolen, leaked, or falsified by any third party, this may result in such network security issues as data on the server is acquired illegally, data on the server is falsified, and communication with the server is disabled.
The customer must manage the client certificate, private key, and secure socket setting, and take measures to prevent them from being stolen, leaked, or falsified.
Especially when you acquire the private key, be careful not to leak it by using an encrypted safe communication path or the like.
Methods for managing the private key include storing it in a location where the possibility of leakage is eliminated as much as possible.
- In order to reduce the risk of unauthorized access by a third party using the Secure Socket Configuration commands, consider setting operation authority verification to the CPU Unit.



Additional Information

- "<Sysmac Studio Installed Folder>" shown in a command to be executed is a folder in which to install the Sysmac Studio, which is by default as follows.
 - a) For the Sysmac Studio with a 32-bit version
For the OS with a 32-bit version, C:\Program Files\OMRON\Sysmac Studio
For the OS with a 64-bit version, C:\Program Files (x86)\OMRON\Sysmac Studio
 - b) For the Sysmac Studio with a 64-bit version
C:\Program Files\OMRON\Sysmac Studio
- The value of /id in the command execution example shown here can be set as desired within the allowable range of session IDs of the CPU Unit.
For the range of session IDs of the CPU Unit, refer to the *NJ/NX-series CPU Unit Built-in EtherNet/IP™ Port User's Manual (Cat. No. W506)*.
- If operation authority verification is set in the CPU Unit, the following will appear when you execute a command, and you will need to enter a password.

```
Operation authority: Administrator
Password: |
```

- Entering the correct password will execute the command.
- Refer to the *NJ/NX-series CPU Unit Built-in EtherNet/IP™ Port User's Manual (Cat. No. W506)* for detailed usage method and troubleshooting of the Secure Socket Configuration commands.

- 1 Check that the operating mode of the CPU Unit is PROGRAM mode.
- 2 Perform the following steps to start the command prompt.
 - 1) Click **OMRON – Sysmac Studio** from the Windows Start Menu.
A folder is displayed.
 - 2) Double-click a shortcut to `tlconfig` in the folder.
The command prompt is started.
- 3 Execute the following command to transfer a private key file and a client certificate file for the session ID 0 to the CPU Unit.
Commands and results in this example are based on the case where the private key file is in "`C:\private\xxx-private.pem.key`". In the same way, they are based on the case where the client certificate file is in "`C:\certs\xxx-certificate.pem.crt`".
Correct the values of /key and /cert, according to the file locations.

```
<Sysmac Studio Installed Folder>\TLSSettingTool>tlsconfig setSessionInfo /id
0 /key "C:\private\xxx-private.pem.key" /cert "C:\certs\xxx-certificate.pem
.crt" /ip:192.168.10.201 /f
```

000: Success is displayed, and the private key file and the client certificate file are transferred to the session ID 0.

If any error number other than **000: Success** is displayed, refer to the error number of that command in *A-3-4 Network Setting for PC* on page A-25 and *NJ/NX-series CPU Unit Built-in EtherNet/IP™ Port User's Manual (Cat. No. W506)* to take action.

4 Execute the following command to check that the setting is made in the CPU Unit correctly.

```
<Sysmac Studio Installed Folder>\TLSSettingTool>tlsconfig getAllSessionInfo
/ip:192.168.10.201
```

- The following results are displayed. The setting is made correctly if the private key file name and the client certificate file name are consistent with those that were set.

```
Count=1
Id=0
PrivateKey=xxx-private.pem.key
Certificate=xxx-certificate.pem.crt
Description=

000:Success
```

This concludes the secure socket setting for the CPU Unit.

The CPU Unit does not need restarting or the like.

If you make a wrong setting, you can delete the setting by using the following command.

- To delete the secure socket setting for the specified session ID

```
<Sysmac Studio Installed Folder>\TLSSettingTool>tlsconfig delSessionInfo /id <Se
ssion ID> /ip:192.168.10.201
```

- To delete the secure socket setting for all the session IDs

```
<Sysmac Studio Installed Folder>\TLSSettingTool>tlsconfig delAllSessionInfo /ip:
192.168.10.201
```

Also, the Secure Socket Configuration commands have a function to enable or disable a communications log.

When an error occurs during communications, you can check a log of the error if you executed the following command in advance.

```
<Sysmac Studio Installed Folder>\TLSSettingTool>tlsconfig setLogLevel /enable /ip:
192.168.10.201
```

A-3-7 Creation of Program and Execution of Program

Create a program on the PC, and operate the program actually on the CPU Unit.

The program enables the following:

- Checking communication with the AWS IoT
- As an MQTT Publisher, the CPU Unit sending a message to the AWS IoT
- As an MQTT Subscriber, the CPU Unit receiving the message from the AWS IoT

Adding MQTT Communications Library to Sysmac Studio

Following the procedure in *Section 1 Sysmac Library Usage Procedure* on page 1-1, add the MQTT Communications Library to the Sysmac Studio, and make it ready to be used by a program.

Creation of Program

Before use, rewrite <***> in the following program according to the environment.

After the completion of rewriting, download it to the CPU Unit.



Precautions for Correct Use

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

● ST

Internal variable	Name	Data type	Default
	Connect	BOOL	FALSE
	Ping	BOOL	FALSE
	Publish	BOOL	FALSE
	Subscribe	BOOL	FALSE
	ins_MQTTClient	OmronLib\MQTT_Comm \MQTTClient	
	ins_MQTTPing	OmronLib\MQTT_Comm \MQTTPing	
	ins_MQTTPubString	OmronLib\MQTT_Comm \MQTTPubString	
	ins_MQTTSubString	OmronLib\MQTT_Comm \MQTTSubString	
	ClientReference	OmronLib\MQTT_Comm \sClientReference	
	ClientID	STRING[256]	
	ConnectionSettings	OmronLib\MQTT_Comm \sConnectionSettings	
	KeepAlive	UINT	60
	Timeout	UINT	10
	DiscardMsgTime	UINT	1000
	Connected	BOOL	
	Busy	BOOL	
	DiscardMsgNum	UDINT	
	SessionPresent	BOOL	
	Error	BOOL	
	ErrorID	WORD	

Internal variable	Name	Data type	Default
	ErrorIDEx	DWORD	
	Png_Timeout	UINT	1000
	Png_Done	BOOL	
	Png_Busy	BOOL	
	Png_ElapseTime	UINT	
	Png_Error	BOOL	
	Png_ErrorID	WORD	
	Png_ErrorIDEx	DWORD	
	Pub_PacketID	UINT	
	Pub_MsgType	USINT	
	Pub_PubMsg	STRING[256]	
	Pub_PubSettings	OmronLib\MQTT_Comm \sPubFlags	
	Pub_Topic	STRING[512]	
	Pub_Timeout	UINT	6
	Pub_Done	BOOL	
	Pub_Busy	BOOL	
	Pub_Error	BOOL	
	Pub_ErrorID	WORD	
	Pub_ErrorIDEx	DWORD	
	Sub_RcvMsg	STRING[512]	
	Sub_SubQoS	BYTE	16#0
	Sub_Topic	STRING[512]	
	Sub_Timeout	UINT	10
	Sub_Subscribed	BOOL	
	Sub_Status	SINT	
	Sub_Received	BOOL	
	Sub_RcvTopic	STRING[512]	
	Sub_Error	BOOL	
	Sub_ErrorID	WORD	
	Sub_ErrorIDEx	DWORD	

```

IF P_First_RunMode THEN
  // ClientID is specified any value.
  ClientID := 'DemoID';
  // <Endpoint> that you can check on AWS IoT setting.
  // e.g. 'xxx.iot.xxx.amazonaws.com'
  ConnectionSettings.IpAdr := '<Endpoint>';
  ConnectionSettings.PortNo := 8883;
  ConnectionSettings.TLSUse := True;
  // TLSsession<Session ID>, <Session ID> is number that you input on secure socket settings.
  // e.g. 'TLSsession0'
  ConnectionSettings.TLSsessionName := 'TLSsession0';
  // User name and password are not used, because this system use certificate for authorizing the MQTT client.

```

```

ConnectionSettings.UserName := '';
ConnectionSettings.Password := '';
ConnectionSettings.WillCfg.WillFlag := False;
ConnectionSettings.CleanSession := True;
// Topic name for the message that Publisher send.
Pub_Topic := 'test/mytopic';
// JSON format message that Publisher send.
Pub_PubMsg := '{"message$": $"Hello!$"}';
Pub_PubSettings.PubQoS := 16#1;
Pub_PubSettings.RetainFlag := False;
// Topic name for the message that Subscriber receive.
Sub_Topic := 'test/#';
END_IF;

ins_MQTTClient(
    Enable := Connect,
    ClientReference := ClientReference,
    ClientID := ClientID,
    ConnectionSettings := ConnectionSettings,
    KeepAlive := KeepAlive,
    Timeout := Timeout,
    DiscardMsgTime := DiscardMsgTime,
    Connected => Connected,
    Busy => Busy,
    DiscardMsgNum => DiscardMsgNum,
    SessionPresent => SessionPresent,
    Error => Error,
    ErrorID => ErrorID,
    ErrorIDEx => ErrorIDEx);

ins_MQTTPing(
    Execute := ins_MQTTClient.Connected AND Ping,
    ClientReference := ClientReference,
    Timeout := Png_Timeout,
    Done => Png_Done,
    Busy => Png_Busy,
    ElapseTime => Png_ElapseTime,
    Error => Png_Error,
    ErrorID => Png_ErrorID,
    ErrorIDEx => Png_ErrorIDEx);

ins_MQTTPubString(
    Execute := ins_MQTTClient.Connected AND Publish,
    ClientReference := ClientReference,
    PacketID := Pub_PacketID,
    MsgType := Pub_MsgType,
    PubMsg := Pub_PubMsg,

```

```

PubSettings := Pub_PubSettings,
Topic := Pub_Topic,
Timeout := Pub_Timeout,
Done => Pub_Done,
Busy => Pub_Busy,
Error => Pub_Error,
ErrorID => Pub_ErrorID,
ErrorIDEx => Pub_ErrorIDEx);

ins_MQTTSubString(
  Enable := ins_MQTTClient.Connected AND Subscribe,
  ClientReference := ClientReference,
  RcvMsg := Sub_RcvMsg,
  SubQoS := Sub_SubQoS,
  Topic := Sub_Topic,
  Timeout := Sub_Timeout,
  Subscribed => Sub_Subscribed,
  Status => Sub_Status,
  Received => Sub_Received,
  RcvTopic => Sub_RcvTopic,
  Error => Sub_Error,
  ErrorID => Sub_ErrorID,
  ErrorIDEx => Sub_ErrorIDEx);

```

Checking Communication to AWS IoT

Check that the CPU Unit sends PING (*PINGREQ* packet) to the AWS IoT and that the CPU Unit can receive the response (*PINGRESP* packet) from the AWS IoT.

- 1** Change the CPU Unit into RUN mode.
- 2** Set the internal variable Connect to TRUE to connect to the AWS IoT.
 ins_MQTTClient.Connected changes to TRUE, and you can check the connection to the AWS IoT.
 If it does not change to TRUE, check and correct the parameters in *A-3-5 Network Setting for CPU Unit* on page A-25, *A-3-6 Secure Socket Setting for CPU Unit* on page A-26, and *Creation of Program* on page A-29. Or, refer to the *MQTTClient* on page 4-2 (MQTT client) instruction to take action.
- 3** Set the internal variable Ping to TRUE to send and receive PING.
 ins_MQTTPing.Done changes to TRUE, and you can check that the PING send/receive process ended normally.
 If it does not change to TRUE, refer to the *MQTTPing* on page 4-52 (MQTTPing message send) instruction to take action.
- 4** Set the internal variable Ping to FALSE to cancel the PING send/receive ready state.

- 5** Set the internal variable `Connect` to `FALSE` to cancel the connection to the AWS IoT.
`ins_MQTTClient.Done` changes to `FALSE`, and you can check that the connection to the AWS IoT was canceled.

Checking Operation as Publisher

Check that the CPU Unit as an MQTT Publisher sends a message to the AWS IoT and that the AWS IoT receives the message.

- 1** Change the CPU Unit into RUN mode.
- 2** Set the internal variable `Connect` to `TRUE` to connect to the AWS IoT.
`ins_MQTTClient.Connected` changes to `TRUE`, and you can check the connection to the AWS IoT.
 If it does not change to `TRUE`, check and correct the parameters in *A-3-5 Network Setting for CPU Unit* on page A-25, *A-3-6 Secure Socket Setting for CPU Unit* on page A-26, and *Creation of Program* on page A-29. Or, refer to the *MQTTClient* on page 4-2 (MQTT client) instruction to take action.
- 3** Set the internal variable `Publish` to `TRUE` to publish a message.
`ins_MQTTPubString.Done` changes to `TRUE`, and you can check that the message was published.
 If it does not change to `TRUE`, refer to the *MQTTPubString* (MQTT character string message publication) instruction to take action.
- 4** Use an MQTT test client on the AWS IoT to check that the MQTT test client can receive the message `'{"message": "Hello!"}'` in the 'test/mytopic' topic.
- 5** Set the internal variable `Publish` to `FALSE` to cancel the publication ready state.
- 6** Set the internal variable `Connect` to `FALSE` to cancel the connection to the AWS IoT.
`ins_MQTTClient.Connected` changes to `FALSE`, and you can check that the connection to the AWS IoT was canceled.

Checking Operation as Subscriber

Register a topic that the CPU Unit as an MQTT Subscriber subscribes to the AWS IoT.
 Check that the AWS IoT sends a message to the CPU Unit and that the CPU Unit receives the message.

- 1** Change the CPU Unit into RUN mode.
- 2** Set the internal variable `Connect` to `TRUE` to connect to the AWS IoT.
`ins_MQTTClient.Connected` changes to `TRUE`, and you can check the connection to the AWS IoT.

If it does not change to TRUE, check and correct the parameters in *A-3-5 Network Setting for CPU Unit* on page A-25, *A-3-6 Secure Socket Setting for CPU Unit* on page A-26, and *Creation of Program* on page A-29. Or, refer to the *MQTTClient* on page 4-2 (MQTT client) instruction to take action.

- 3** Set the internal variable `Subscribe` to TRUE to make ready to subscribe to and receive a message.
The internal variable `Sub_Status` value changes to 2 (Subscription in progress).
If it does not change to 2, refer to the *MQTTSubString* on page 4-48 (MQTT character string message subscription request) instruction to take action.
- 4** Use an MQTT test client on the AWS IoT to specify the topic name 'test/topicfromaws' and the message payload (Message), then publish (Publish) them to the topic.
You can acquire 'test/topicfromaws' to the internal variable `Sub_RcvTopic`, and the message sent from the Azure to `Sub_RcvMsg`.
If you cannot acquire the message, refer to the *MQTTSubString* on page 4-48 (MQTT character string message subscription request) instruction to take action.
- 5** Set the internal variable `Subscribe` to FALSE to cancel the subscription ready state.
The internal variable `Sub_Status` value changes to 0 (Execution stopped), and you can check that the state of subscription in progress was canceled.
- 6** Set the internal variable `Connect` to FALSE to cancel the connection to the AWS IoT.
`ins_MQTTClient.Connected` changes to FALSE, and you can check that the connection to the AWS IoT was canceled.



Index



Index

D

Data publication in MQTT byte array.....4-24

Data subscription request in MQTT byte array.....4-38

M

MQTT character string message publication.....4-34

MQTT character string message subscription request....4-48

MQTT client.....4-2

MQTTClient.....4-2

MQTTPing.....4-52

MQTTPing message send.....4-52

MQTTPubAryByte.....4-24

MQTTPubString.....4-34

MQTTSubAryByte.....4-38

MQTTSubString.....4-48

OMRON Corporation Industrial Automation Company
Kyoto, JAPAN

Contact: www.ia.omron.com

Regional Headquarters

OMRON EUROPE B.V.

Wegalaan 67-69, 2132 JD Hoofddorp
The Netherlands
Tel: (31)2356-81-300/Fax: (31)2356-81-388

OMRON ELECTRONICS LLC

2895 Greenspoint Parkway, Suite 200
Hoffman Estates, IL 60169 U.S.A.
Tel: (1) 847-843-7900/Fax: (1) 847-843-7787

OMRON ASIA PACIFIC PTE. LTD.

No. 438A Alexandra Road # 05-05/08 (Lobby 2),
Alexandra Technopark,
Singapore 119967
Tel: (65) 6835-3011/Fax: (65) 6835-2711

OMRON (CHINA) CO., LTD.

Room 2211, Bank of China Tower,
200 Yin Cheng Zhong Road,
PuDong New Area, Shanghai, 200120, China
Tel: (86) 21-5037-2222/Fax: (86) 21-5037-2200

Authorized Distributor:

© OMRON Corporation 2021 All Rights Reserved.
In the interest of product improvement,
specifications are subject to change without notice.

Cat. No. W625-E1-02

1021