Machine Automation Controller NJ-series

General Ethernet (TCP/IP) Connection Guide

OMRON Corporation
Auto Focus Multi Code Reader
V330-F / V430-F-series

Network

Connection

Guide

Z412-E1-01

About Copyrights and Trademarks

Contents

1. Related Manuals

To ensure system safety, make sure to always read and follow the information provided in all Safety Precautions and Precautions for Safe Use in the manuals for each device which is used in the system.
The following OMRON Corporation (hereinafter referred to as “OMRON”) manuals are related to this document:

Cat. No.	Model	Manual name
W500	NJ Series	NJ-series CPU Unit Hardware User’s Manual
W501	NJ/NX Series	NJ/NX-series CPU Unit Software User’s Manual
W506	NJ/NX Series	NJ/NX-series CPU Unit Built-in EtherNet/IP Port User’s Manual
W504	SYSMAC-SE2□□□	Sysmac Studio Version 1 Operation Manual
W502	NJ/NX Series	Machine Automation Controller Instructions Reference Manual
Z432	V320-F/V330-F/V420-F/V430-F Series	MicroHAWK V320-F/V330-F/V420-F/V430-F Series Barcode Reader User Manual
Z407	V320-F/V330-F/V420-F/V430-F Series	Autofocus Multicode Reader MicroHAWK V320-F/V330-F/V420-F/V430-F Series User Manual for Communication Settings

2. Terms and Definitions

Term	Description/Definition
IP Address	Ethernet uses IP addresses to achieve communications. Each IP address (specifically, Internet Protocol address) identifies a specific node (host computer, controller, etc.) on an Ethernet network, IP addresses must be set and managed so that they are not duplicated.
Socket	A socket is an interface that allows you to directly use TCP or UDP functions from a user program. The NJ Series Machine Automation Controller performs socket communication using standard socket service instructions. To use socket services, you need to establish a connection with a remote node and disconnect it after use. In this document, processing for establishing a connection is referred to as “socket open” or “TCP open” and for disconnecting it as “socket close” or “close”. You can use the socket services to send and receive arbitrary data to and from the remote node.
Active and Passive	When you open a TCP socket connection with nodes, open processing is executed for each node. The method to open a connection differs depending on whether the node is to serve as a client or server. In this document, processing to open a connection as a server is referred to as “passive open” and as a client is referred to as “active open” or “active open processing”.
keep-alive Function	When a remote node (server or client) does not respond for a set period of time or longer in TCP/IP socket services, the keep-alive function sends a communications frame to the node to check the connection status. If the node does not respond to it, the function performs this check at a certain interval, and closes the connection if it does not respond to all check frames.
linger function	This is a TCP socket option that sends RST data when the TCP socket is closed. This enables immediate open processing using the same port number, without waiting for the port to be opened. If the linger option is not specified, the controller issues FIN data when the TCP socket is closed and, after that, performs end control such as a send data arrival check with the remote node for approximately 1 minute. Therefore, TCP sockets with the same port number may not be used immediately.

3. Restrictions and Precautions

(1) Before building a system, understand the specifications of devices which are used in the system. Allow some margin for ratings and performance, and provide safety measures such as installing a safety circuit in order to minimize the risk in case of failure.
(2) To ensure system safety, make sure to read and follow the information provided in all Safety Precautions and Precautions for Safe Use in the manuals for each device which is used in the system.
(3) The user is encouraged to confirm the standards and regulations that the system must conform to.
(4) It is prohibited to copy, to reproduce, and to distribute a part or the whole of this document without the permission of OMRON Corporation.
(5) The information contained in this document is current as of March 2023.
It is subject to change for improvement without notice.

The following notations are used in this document.

⚠ WARNING	Indicates a potentially hazardous situation which, if not avoided, may result in minor or moderate injury, or may result in serious injury or death. 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.

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.

Note

Additional information to read as required.
This information is provided to increase understanding or make operation easier.

Symbols

❗	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.

4. Overview

This document describes the procedures for connecting the OMRON code reader products (V330-F/V430-F Series) to an NJ Series Machine Automation Controller (hereinafter referred to as the controller) via Ethernet and for checking their connections.
You can establish an Ethernet communication connection by understanding the setting procedures and key points of setup through the Ethernet communication settings in the project file prepared in advance.
In this project file, the Ethernet connection is checked by sending a read trigger command to the code reader and receiving the read data from it.

Obtain the latest version of the Sysmac Studio Project File from OMRON in advance.

Name	Filename	Version
Sysmac Studio Compact Project File (Extension: smc2)	OMRON_V330_V430_NJ_ETN(TCP)_V100.smc2	Ver. 1.00

⚠ Caution

The purpose of this document is to describe the wiring methods, communication settings, and setting procedures required to establish a connection for communications with applicable devices. In addition, the program used in this document is designed to check that the connection has been correctly performed (connection check). Since the program is not intended for permanent use on-site, full consideration is not given to functionality and performance. When configuring an actual system, please refer to the wiring methods, communication settings, and setting procedures described in this document to design and create a program that meets your purpose.

5. Applicable Products and Support Tools

5.1. Applicable Products

The applicable devices that are required to ensure a connection are as follows:

Manufacturer	Name	Model	Version
OMRON	NJ Series CPU Unit	NJ501-1500 NJ501-1400 NJ501-1300 NJ301-□□□□	Same or later version as indicated in section 5.2.
OMRON	Code reader	V330-F□□□□□□□-□□□ V430-F□□□□□□□-□□□	

Note

This document describes the procedures for establishing the communication connection of the device, and does not describe the operation, installation and wiring method of the device. For details on the above products (other than communication connection procedures), please refer to the instruction manual for the product or contact OMRON.

Note

From among the above applicable devices, this document uses the devices listed in section 5.2 for the connection check. When using devices that are not described in section 5.2, check the connection according to this document.

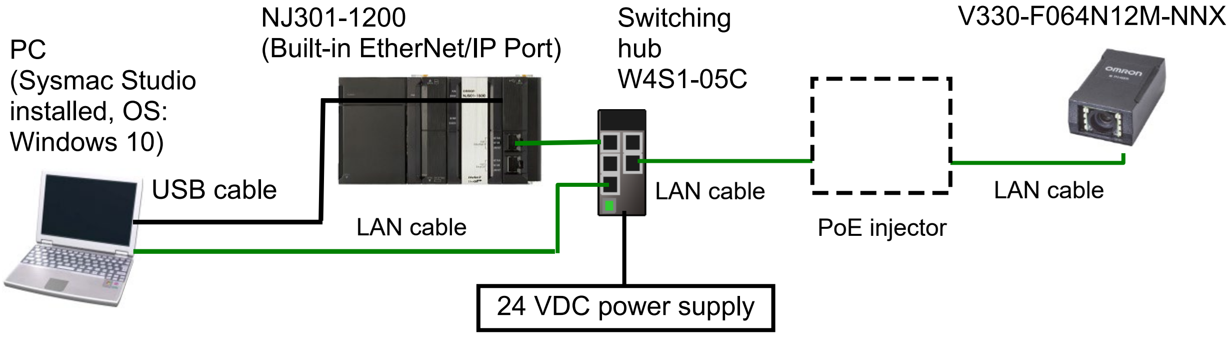
Precautions for Correct Use

The connection and connection check procedures described in this document use the devices listed in section 5.2, from among the above applicable devices.
You cannot use devices with versions earlier than the versions listed in section 5.2.
To use models that are not listed in section 5.2. or versions that are later than those listed in section 5.2., check the differences in the specifications according to their instruction manuals before operating the devices.

5.2. Device Configuration

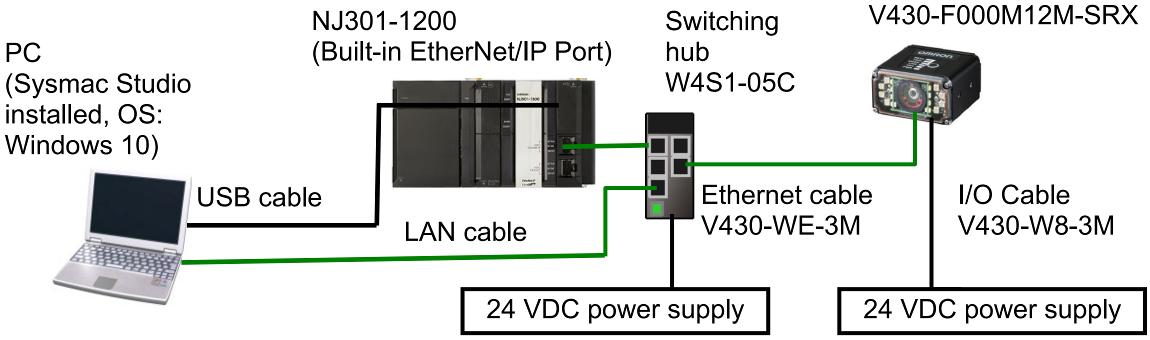
The system components required for reproducing the connection procedures described in this document are as follows.

- Configuration with V330-F



Manufacturer	Name	Model	Version
OMRON	NJ Series CPU Unit (Built-in EtherNet/IP Port)	NJ301-1200	Ver. 1.19
OMRON	Power Supply Unit	NJ-PA3001	
OMRON	Switching hub	W4S1-05C	
OMRON	Sysmac Studio	SYSMAC-SE2□□□	Ver. 1.44
OMRON	Sysmac Studio Project File	OMRON_V330_V430_NJ_ETN(TCP)_V100. smc2	Ver. 1.00
---	PC (OS: Windows 10)	---	
---	USB cable (USB 2.0-compliant B-type connector)	---	
---	LAN cable (STP (shielded, twisted-pair) cable of Ethernet category 5 or higher)		
OMRON	Code reader	V330-F064N12M-NNX	Ver. 2.1.0
OMRON	PoE (Power over Ethernet) injector	Select one that can be powered via Ethernet.	---
---	24 VDC power supply	---	

- Configuration with V430-F



Manufacturer	Name	Model	Version
OMRON	NJ Series CPU Unit (Built-in EtherNet/IP Port)	NJ301-1200	Ver. 1.19
OMRON	Power Supply Unit	NJ-PA3001	
OMRON	Switching hub	W4S1-05C	
OMRON	Sysmac Studio	SYSMAC-SE2□□□	Ver. 1.44
OMRON	Sysmac Studio Project File	OMRON_V330_V430_NJ_ETN(TCP)_V100.smc2	Ver. 1.00
	PC (OS: Windows 7)		
	USB cable (USB 2.0-compliant B-type connector)		
	LAN cable (STP (shielded, twisted-pair) cable of Ethernet category 5 or higher)		
OMRON	Code reader	V430-F000M12M-SRX	Ver. 2.1.0
OMRON	I/O Cable	V430-W8-3M	
OMRON	Ethernet cable	V430-WE-3M	
---	24 VDC power supply	---	

Precautions for Correct Use

Obtain the latest version of the Sysmac Studio Project File from OMRON in advance. (Contact OMRON for information on how to obtain this file.)

Note

The configuration may not be reproduced if the system component models or versions differ. Check your configuration and, if there is any difference in the models or versions, contact OMRON.

Note

This document assumes that the USB is used to connect the controller. For information on how to install the USB driver, refer to *A-1 Driver Installation for Direct USB Cable Connection* in *Appendices* of the *Sysmac Studio Version 1 Operation Manual* (Cat. No. W504).

Note

Refer to the *Industrial Switching Hub W4S1 Series User Manual* (0969584-7) for power supply specifications that can be used for 24 VDC power supply (for the switching hub).

Note

Refer to the *MicroHAWK V320-F/V330-F/V420-F/V430-F Series Barcode Reader User Manual* (Cat. No. Z432) for the power supply specifications that can be used for 24 VDC power supply (for the code reader).

6. Ethernet Settings

This section shows the specifications of the communication parameter settings, variable names and other information provided in this document.

Note

This document and the project file only cover the operations that you can perform using the settings and commands described in this section. To use communication settings that are not described here, you need to modify the project file.

6.1. Ethernet Communication Settings

The settings required to perform Ethernet communications are as follows.

6.1.1. Communications Settings for Setting PC and Code Reader

This document assumes that you use the settings below to set the code reader using a setting PC.

Parameter name	Setting PC	Code reader
IP address	192.168.188.100	192.168.188.2 (default)
Subnet mask	255.255.0.0	255.255.0.0 (default)
Gateway	Blank (default)	0.0.0.0 (default)

* For the use cases in this document, setting the gateway is unnecessary because the devices are connected within the same segment of the network.

6.1.2. Communication Settings for Ethernet Unit and Code Reader

It is assumed that you use the settings below to connect the Ethernet Unit and the code scanner.

Parameter name	NJ301-1200 (Built-in EtherNet/IP Port)	Code reader
IP address	192.168.188.1	192.168.188.2 (default)
Subnet mask	255.255.0.0	255.255.0.0 (default)
Gateway	---	0.0.0.0 (default)
Port number	(set by software part)	2001 (default)

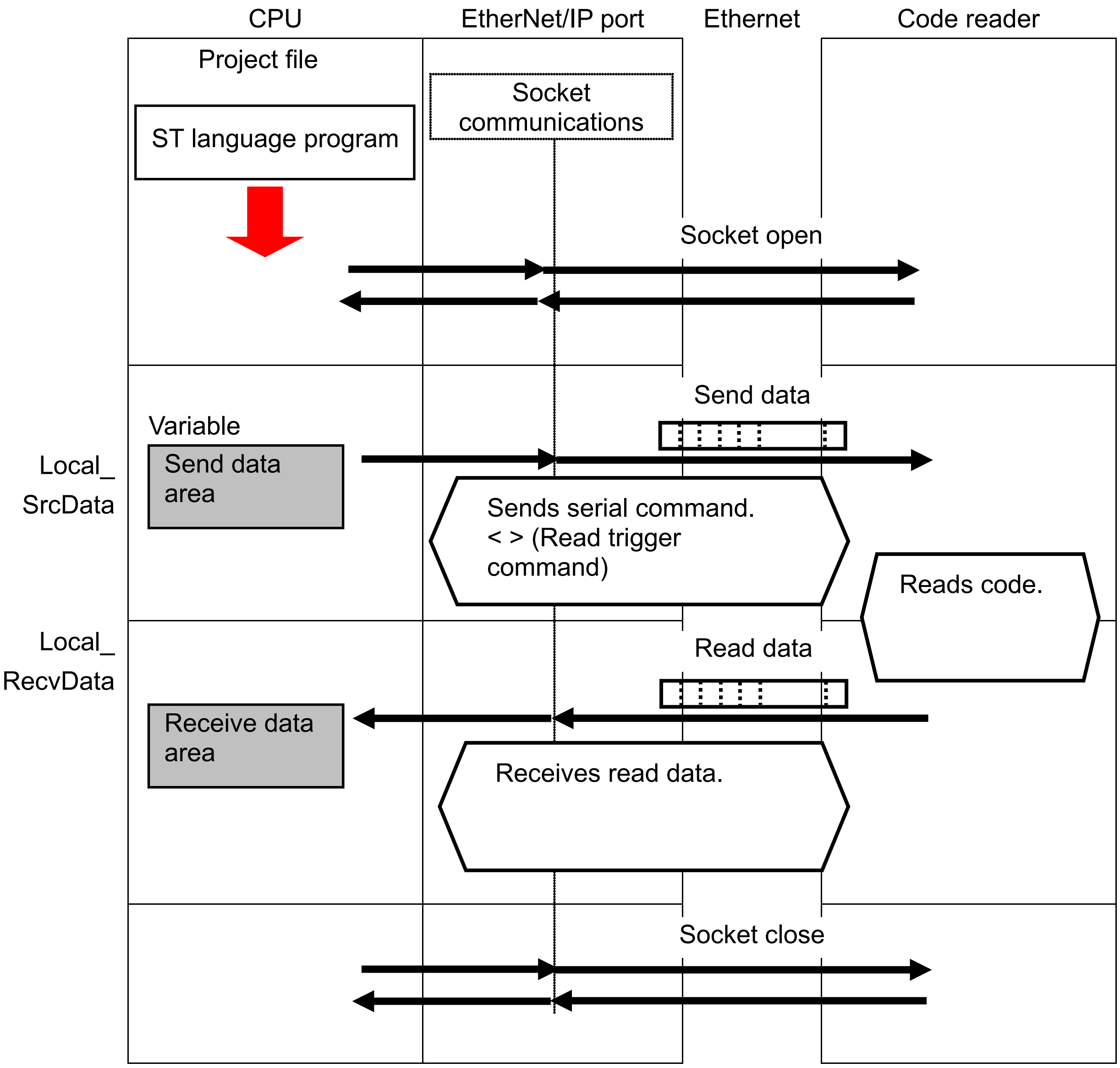
* For the use cases in this document, setting the gateway is unnecessary because the devices are connected within the same segment of the network.

6.2. Example of Connection Check for Communications

This document assumes that you use a program in structured text (hereinafter, ST) language to execute “socket open”, “send and receive”, and “socket close” from the controller to the code reader.

The controller sends a “read trigger” command to the code reader. The code reader sends the read data back to the controller.

An overview of the operation is shown below.



7. Connection Procedure

This section describes the procedures for connecting the controller to an Ethernet network.
In this document, it is assumed that the controller and the code reader use the factory default settings. For how to initialize the devices, refer to *Section 8. Initializing the System*.

7.1. Operation Flow

The procedures for connecting and setting up the controller via Ethernet are as follows.

7.2. Code Reader Setup	Set up the code reader.
▼	
7.2.1. Setting the Parameters	Set the parameters for the code reader.
▽	
7.3. Controller Setup	Set up the controller.
▼	
7.3.1 Starting the Sysmac Studio and Loading the Project File	Start the Sysmac Studio Automation Software and load the Sysmac Studio Project File.
▼	
7.3.2 Checking Parameters and Executing builds	Check the setting parameters. Then, perform program checks and builds on project data.
▼	
7.3.3 Going Online and Transferring the Project Data	Place the Sysmac Studio online and transfer the project data to the controller.
▽	
7.4. Checking the Connection Status	Execute the transferred project file to check that Ethernet communications work correctly.
▼	
7.4.1 Executing the Project File and Checking the Receive Data	Execute the project file and check that correct data is written to controller variables.

Precautions for Correct Use

Obtain the latest version of the Sysmac Studio Project File from OMRON in advance.
(Contact OMRON for information on how to obtain this file.)

7.2. Code Reader Setup

Set up the code reader.

Precautions for Correct Use

Use a PC (personal computer) to set the parameters for the code reader.

Note that you may need to change the PC settings depending on the condition of your PC.

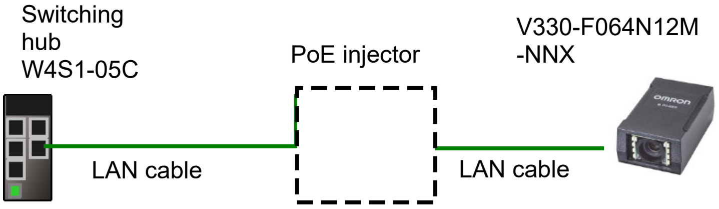
7.2.1. Setting the Parameters

Set the parameters for the code reader.

Set the IP address of your PC to *192.168.188.101* and its subnet mask to *255.255.0.0*.

1 [Using V330-F]

Connect the cord reader and the switching hub to the PoE injector with a LAN cable.



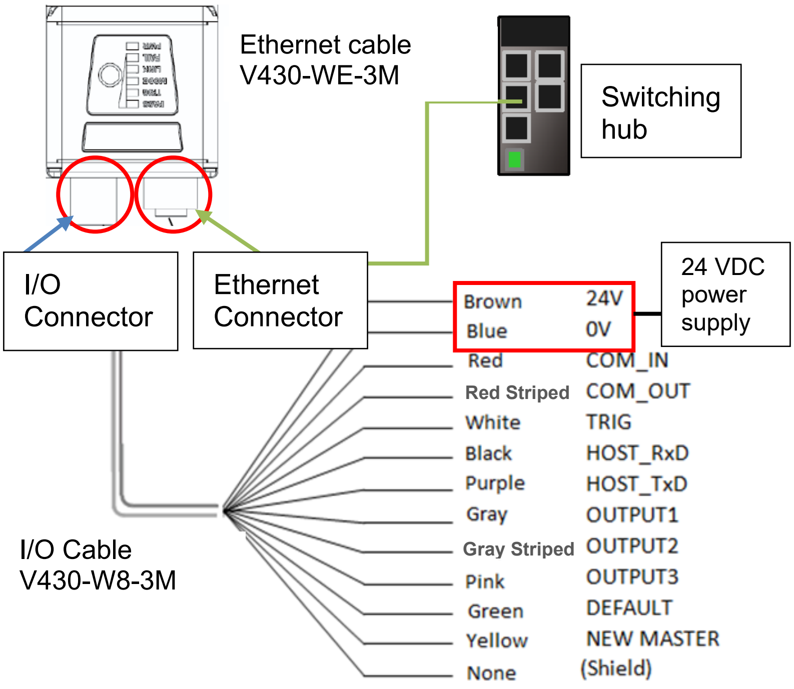
[Using V430-F]

Connect the Ethernet connector of the code reader to the switching hub with the Ethernet cable.

Connect the I/O cable to the I/O connector and turn ON the 24 VDC power supply.

* In this document, only the power supply wires of the I/O cable are connected and checked. Be careful not to short-circuit any other wires.

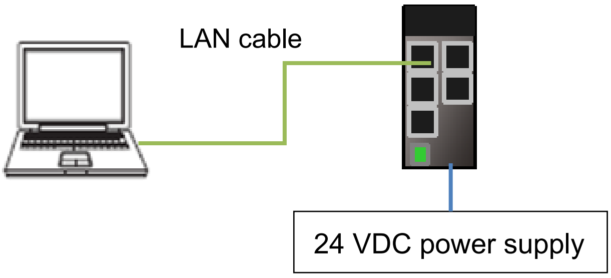
* Ground the shield wire as needed. For more information on grounding, please refer to *Grounding* in *Appendices* of the *MicroHAWK V320-F/V330-F/V420-F/V430-F Series Barcode Reader User Manual* (Cat. No. Z432).



Brown	24V
Blue	0V
Red	COM_IN
Red Striped	COM_OUT
White	TRIG
Black	HOST_RxD
Purple	HOST_TxD
Gray	OUTPUT1
Gray Striped	OUTPUT2
Pink	OUTPUT3
Green	DEFAULT
Yellow	NEW MASTER
None	(Shield)

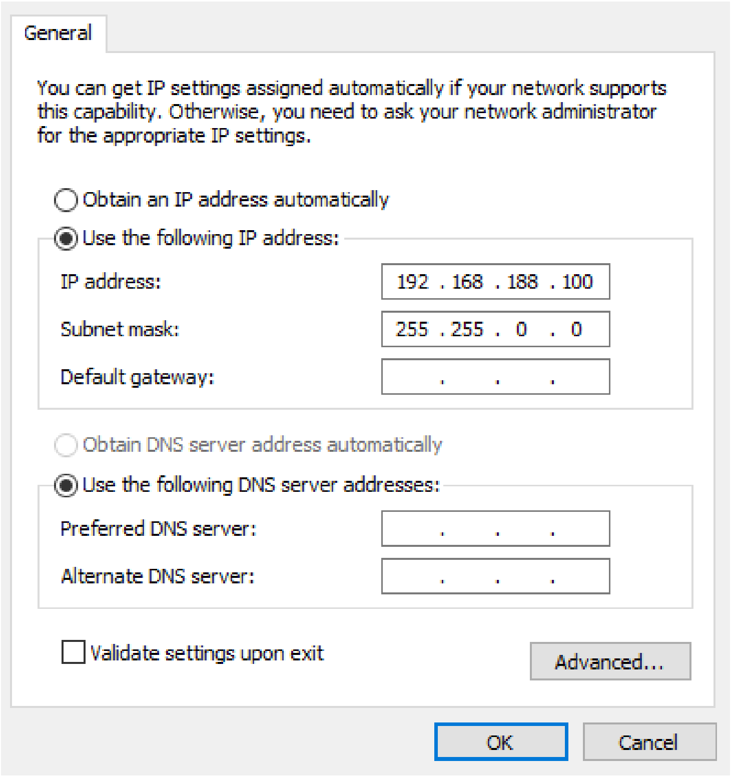
2 Connect the PC to the switching hub with a LAN cable.

Connect 24 VDC power supply (for the switching hub) to the switching hub.



3 Set the IP Address of the PC.
For the IP address, enter *192.168.188.100.*
For the subnet mask, enter *255.255.0.0.*

For how to open the screen shown on the right in Windows 10, please refer to step 4.



General

You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.

○ Obtain an IP address automatically

● Use the following IP address:

IP address:	192 . 168 . 188 . 100
Subnet mask:	255 . 255 . 0 . 0
Default gateway:	. . .

○ Obtain DNS server address automatically

● Use the following DNS server addresses:

Preferred DNS server:	. . .
Alternate DNS server:	. . .

□ Validate settings upon exit

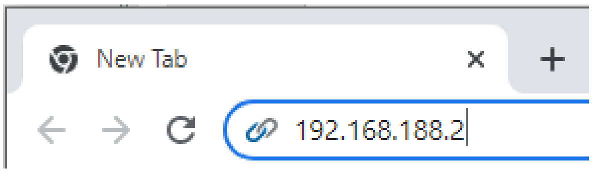
4 (1) From the Windows **Start** Menu, select **Control Panel – Network and Internet – Network and Sharing Center**.

(2) Click on **Local Area Connection**. The **Local Area Connection Status** Dialog Box is displayed. Click **Properties**.

(3) In the **Local Area Connection Properties** Dialog Box, select *Internet Protocol Version 4 (TCP / IPv4)*, and click the **Properties** Button.

(4) Click the **OK** Button.

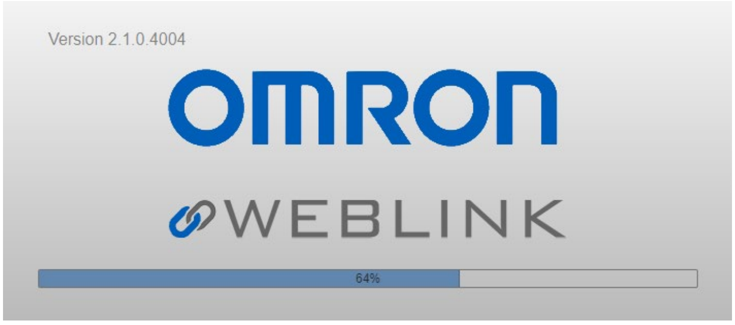
5 Start your browser and enter *http://192.168.188.2.*
"Google Chrome" is the recommended browser.



192.168.188.2

6 When the WebLink startup screen is displayed, go to step 8.

If you cannot access by WebLink, go to step 7.

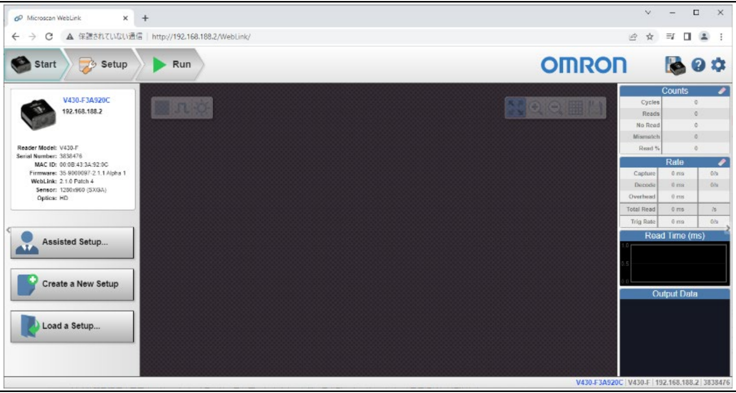


7 If the WebLink startup screen does not appear, it means that communications are not established between the code reader and the PC. Please check the following.

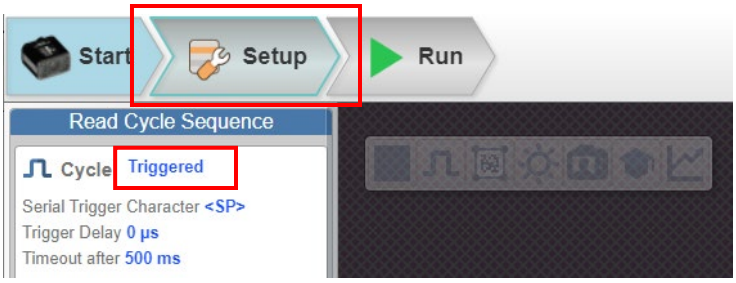
- The code reader and the PC have a proper physical (cable) connection.
  → Refer to steps 1 and 2 for checking the connection.
- The IP Addresses of the PC and code reader are set correctly.
  → Refer to step 4 for setting the IP address of the PC.

For other measures that can be taken, please refer to *When unable to access by WebLink* in *Q&A* in *Appendices* of the *MicroHAWK V320-F/V330-F/V420-F/V430-F Series Barcode Reader User Manual* (Cat. No. Z432).

8 The WebLink screen appears.



9 Click on the **Setup** Tab and, in **Read Cycle Sequence**, set **Cycle** to *Triggered*.



Read Cycle Sequence

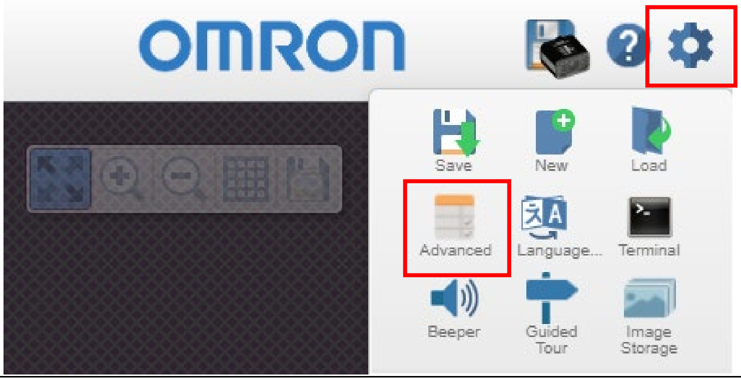
Cycle Triggered

Serial Trigger Character <SP>

Trigger Delay 0 µs

Timeout after 500 ms

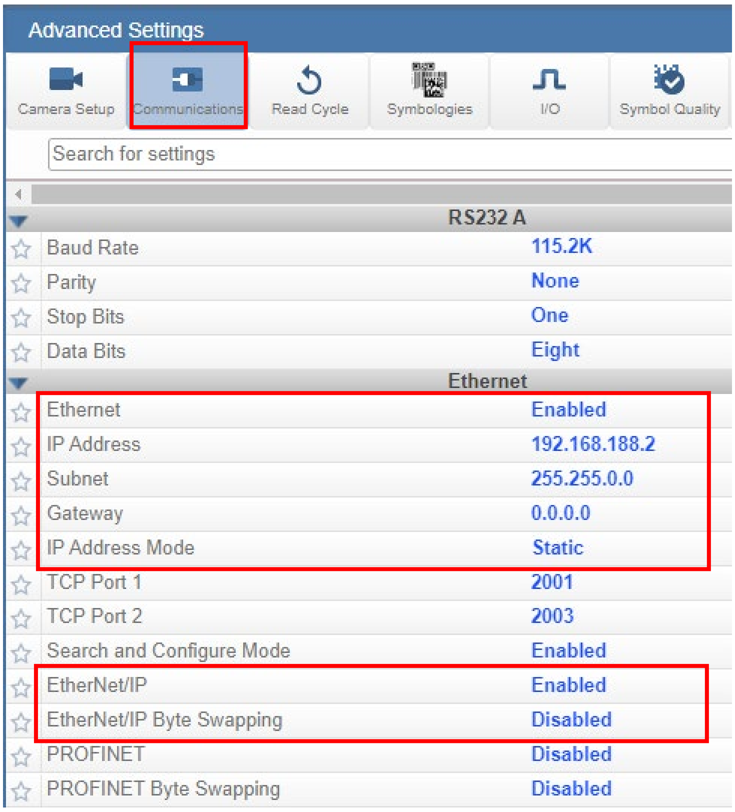
10 Click on the gear icon on the upper right of the screen and select **Advanced**.



11 The Advanced Settings Screen appears.
Select the **Communications** Tab and check the settings for **Ethernet** shown in the red frame.

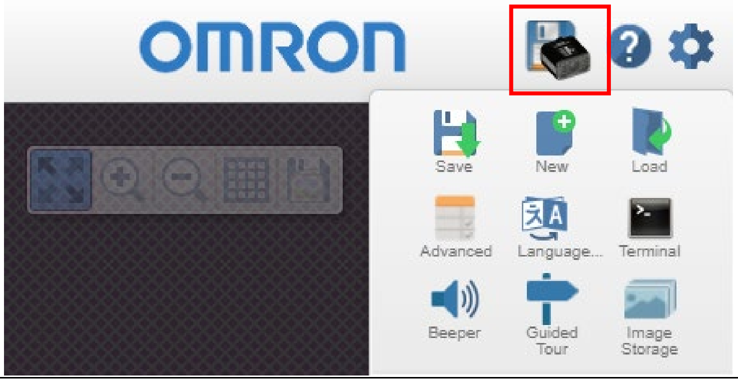
To use the defaults, you do not need to change the settings.

If you need to change the IP address, for example when connecting multiple code readers, change the **IP Address** and subsequent settings as necessary.

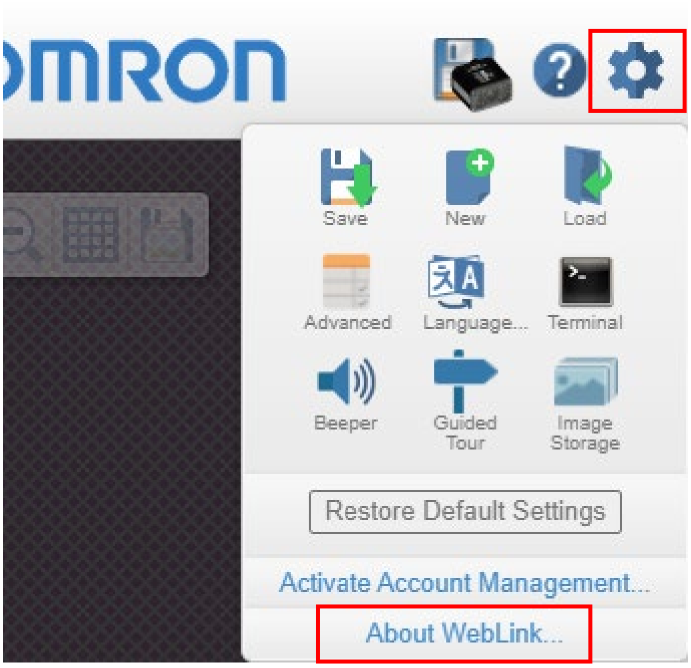


RS232 A	
Baud Rate	115.2K
Parity	None
Stop Bits	One
Data Bits	Eight
Ethernet	
Ethernet	Enabled
IP Address	192.168.188.2
Subnet	255.255.0.0
Gateway	0.0.0.0
IP Address Mode	Static
TCP Port 1	2001
TCP Port 2	2003
Search and Configure Mode	Enabled
EtherNet/IP	Enabled
EtherNet/IP Byte Swapping	Disabled
PROFINET	Disabled
PROFINET Byte Swapping	Disabled

12 Click on the icon shown in the red frame to save the settings to the code reader.

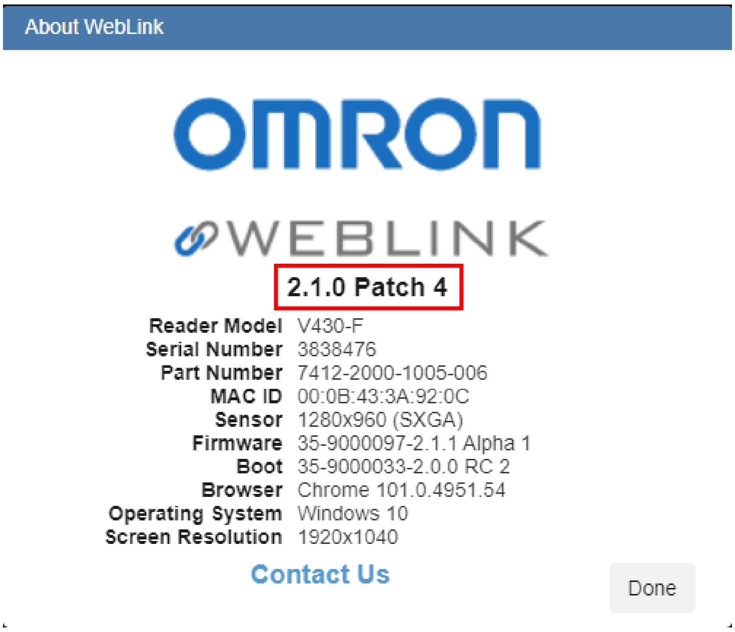


13 Finally, check the version number of the code reader.
Click on the gear icon on the upper right of the screen and select **About WebLink**.



14 **About WebLink** is displayed, so you can check the current version of the code reader.

Please update the code reader to the latest version if necessary.



About WebLink

2.1.0 Patch 4

Reader Model	V430-F
Serial Number	3838476
Part Number	7412-2000-1005-006
MAC ID	00:0B:43:3A:92:0C
Sensor	1280x960 (SXGA)
Firmware	35-9000097-2.1.1 Alpha 1
Boot	35-9000033-2.0.0 RC 2
Browser	Chrome 101.0.4951.54
Operating System	Windows 10
Screen Resolution	1920x1040

7.3. Controller Setup

Set up the controller.

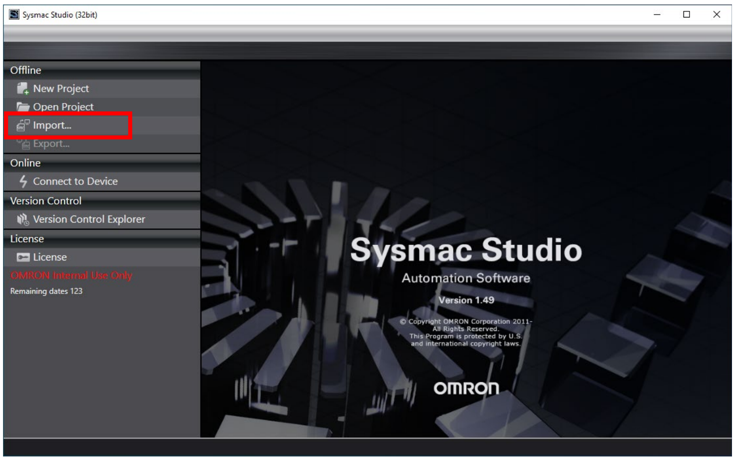
7.3.1. Starting the Sysmac Studio and Loading the Project File

Start the Sysmac Studio Automation Software and load the Sysmac Studio Project File.
Install the Sysmac Studio and USB driver on the PC beforehand. In addition, connect the PC and the controller with a USB cable, and turn ON the power supply to the controller.

1 Start the Sysmac Studio.

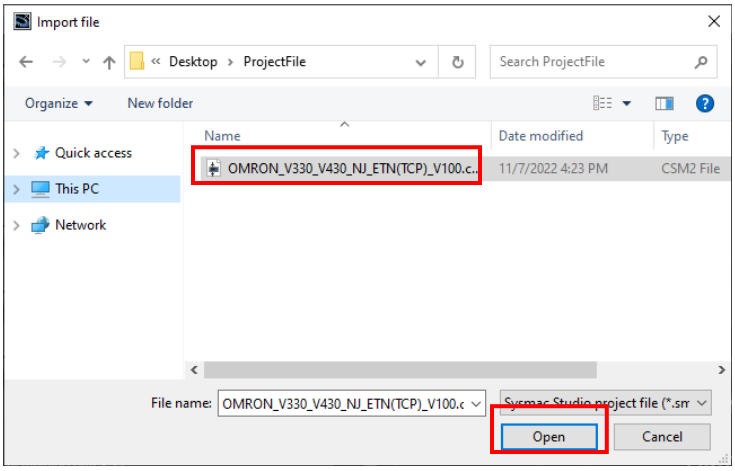
Click **Import**.

* If a user account control dialog box is displayed at startup, select the option to start.



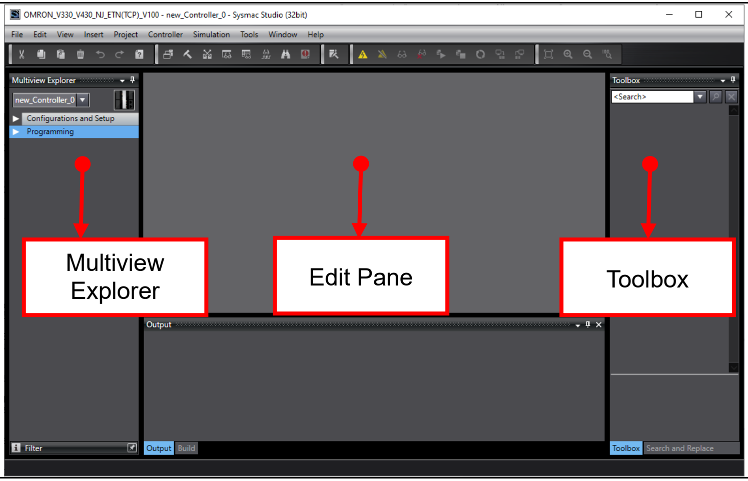
2 The Import file Dialog Box is displayed. Select the project file **OMRON_V330_V430_NJ_ETN(TCP)_V100. smc2** (Sysmac Studio Project File) and click **Open**.

* Obtain the latest version of the Sysmac Studio Project File from the OMRON website.



3 The OMRON_V330_V430_ETN (TCP)_V100 Project Window is displayed.

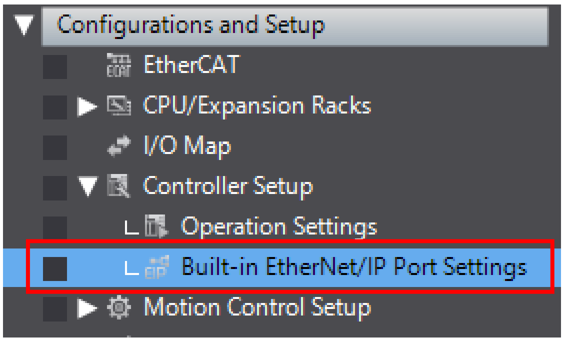
The window consists of three panes: “Multiview Explorer” on the left side, “Edit Pane” in the center, and “Toolbox” on the right side.



7.3.2. Checking Parameters and Executing builds

Check the setting parameters. Then, perform program checks and builds on project data.

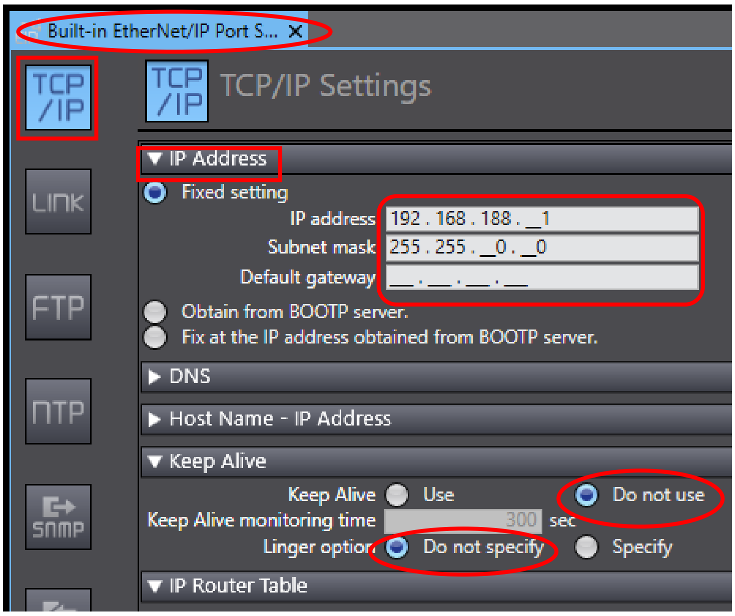
1 Double-click **Built-in EtherNet/IP Port Settings** under **Configurations and Setup – Controller Setup** in the Multiview Explorer.



2 The **Built-in EtherNet/IP Port Settings** Tab Page is displayed in the Edit Pane.

Select **TCP/IP**, select the **Fixed setting** Option in **IP Address**, and check that the settings are as follows.
IP Address: 192.168.188.1
Subnet mask: 255.255.0.0
Default gateway: _._._._

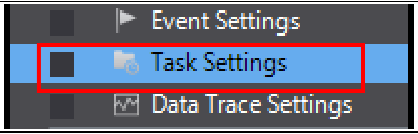
Check that **Keep Alive** is set as follows.
Keep Alive: Do not use
Linger option: Do not specify



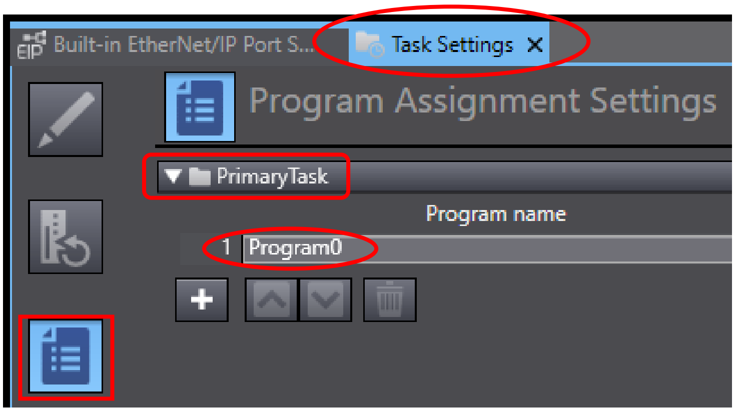
● Fixed setting
IP address 192 . 168 . 188 . _1
Subnet mask 255 . 255 . _0 . _0
Default gateway __ . __ . __ . __
○ Obtain from BOOTP server.
○ Fix at the IP address obtained from BOOTP server.

Keep Alive ○ Use ● Do not use
Keep Alive monitoring time 300 sec
Linger option ● Do not specify ○ Specify

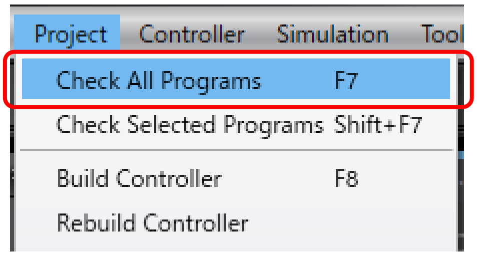
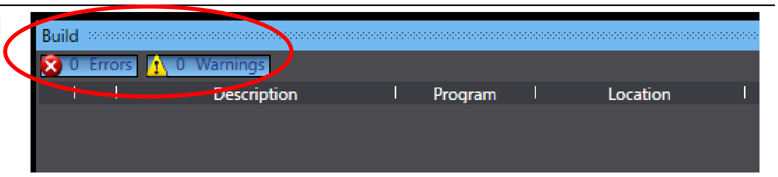
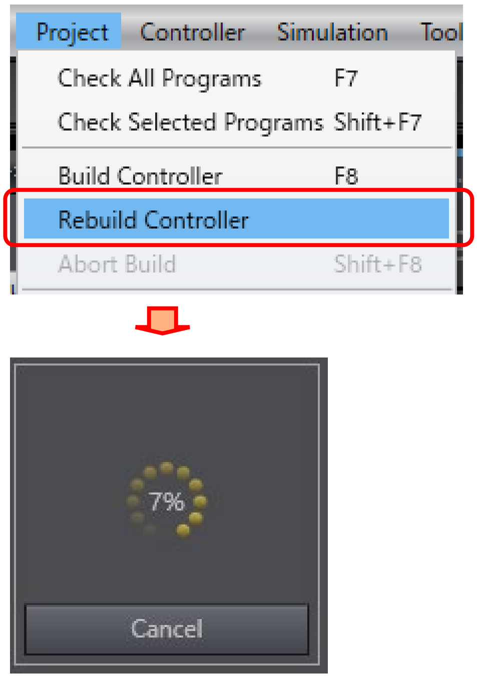
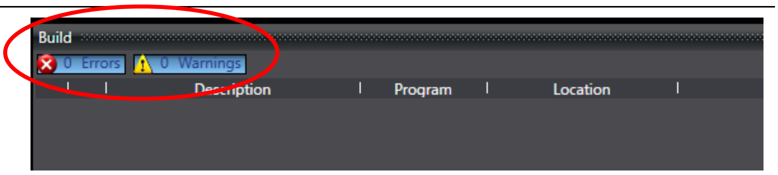
3 Double-click **Task Settings** under **Configurations and Setup** in the Multiview Explorer.



4 The **Task Settings** Tab Page is displayed in the Edit Pane. Select **Program Assignment Settings** and confirm that **Primary Task** is set to *Program0*.

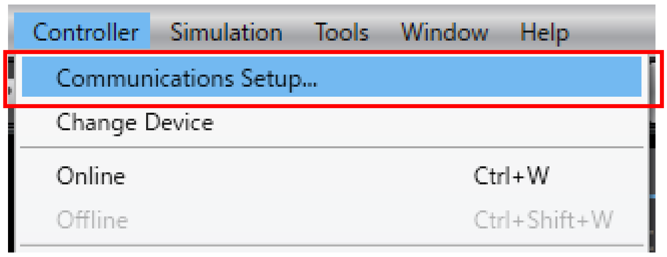
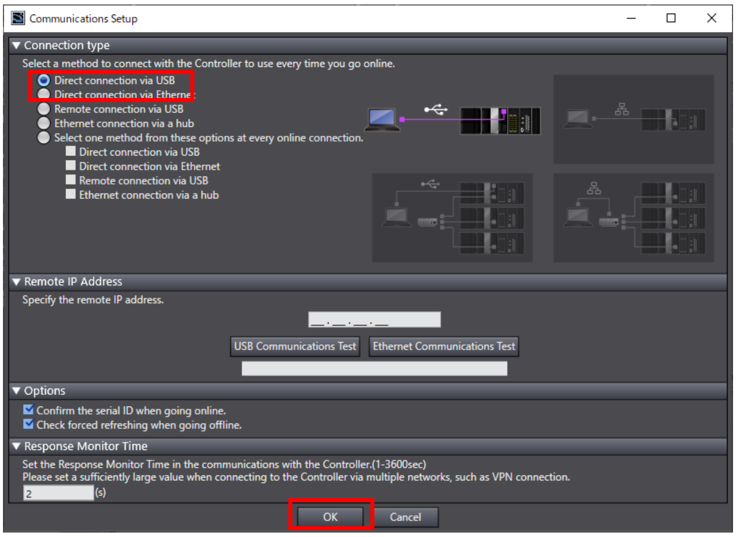
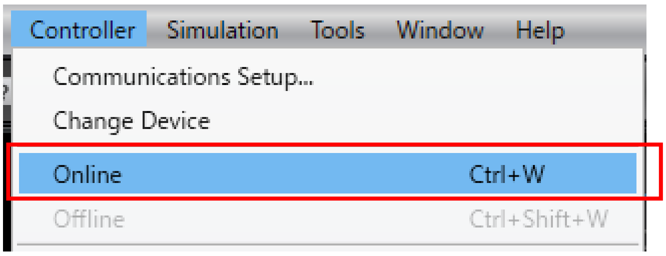
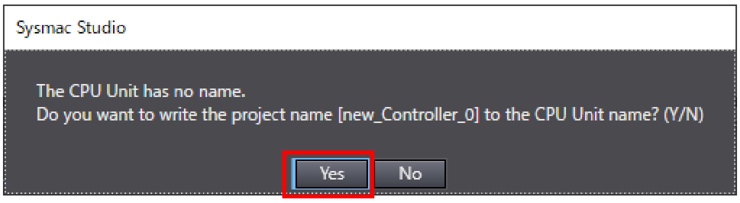


1 Program0

5	Select **Check All Programs** from the **Project** Menu.	 Project Controller Simulation Tool Check All Programs F7 Check Selected Programs Shift+F7 Build Controller F8 Rebuild Controller
6	The **Build** Tab Page is displayed under the Edit Pane. Confirm that *0* is shown for both **Errors** and **Warnings**.	 Build 0 Errors 0 Warnings Description Program Location
7	Select **Rebuild Controller** from the **Project** Menu. A dialog box showing the progress of conversion appears.	 Project Controller Simulation Tool Check All Programs F7 Check Selected Programs Shift+F7 Build Controller F8 Rebuild Controller Abort Build Shift+F8 7% Cancel
8	In the **Build** Tab Page, confirm that *0* is shown for both **Errors** and **Warnings**.	 Build 0 Errors 0 Warnings Description Program Location

7.3.3. Going Online and Transferring the Project Data

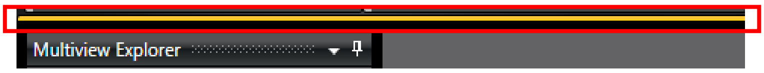
Place the Sysmac Studio online and transfer the project data to the controller.

1	Select **Communications Setup** from the **Controller** Menu.	 Controller Simulation Tools Window Help Communications Setup... Change Device Online Ctrl+W Offline Ctrl+Shift+W
2	The Communications Setup Dialog Box is displayed. In **Connection type**, select the **Direct connection via USB** Option. Click **OK**.	Communications Setup ▼ Connection type Select a method to connect with the Controller to use every time you go online. ● Direct connection via USB ○ Direct connection via Etherne ○ Remote connection via USB ○ Ethernet connection via a hub ○ Select one method from these options at every online connection. □ Direct connection via USB □ Direct connection via Ethernet □ Remote connection via USB □ Ethernet connection via a hub ▼ Remote IP Address Specify the remote IP address. __.__.__.__ USB Communications Test / Ethernet Communications Test ▼ Options ☑ Confirm the serial ID when going online. ☑ Check forced refreshing when going offline. ▼ Response Monitor Time Set the Response Monitor Time in the communications with the Controller.(1-3600sec) Please set a sufficiently large value when connecting to the Controller via multiple networks, such as VPN connection. 2 (s) OK / Cancel
3	Select **Online** from the **Controller** Menu. A confirmation dialog box appears. Click **Yes**. * The dialog box displayed differs depending on the status of the controller being used. Select **Yes** to proceed with the operation.	 Controller Simulation Tools Window Help Communications Setup... Change Device Online Ctrl+W Offline Ctrl+Shift+W Sysmac Studio The CPU Unit has no name. Do you want to write the project name [new_Controller_0] to the CPU Unit name? (Y/N) Yes / No

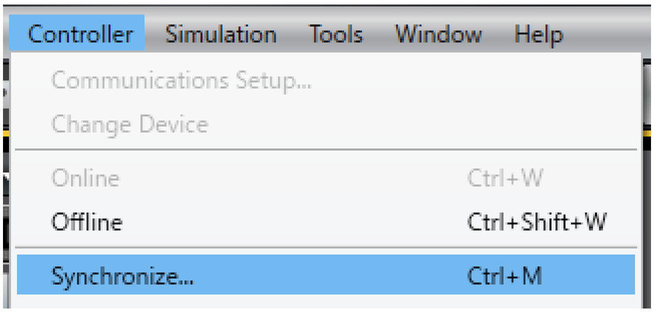
Note

Refer to *Section 6 Online Connections to a Controller* in the *Sysmac Studio Version 1 Operation Manual* (Cat. No. W504) for details on online connection to the controller.

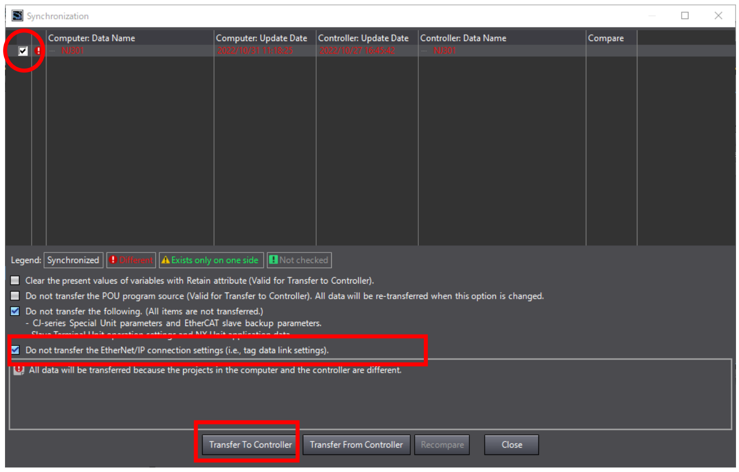
4 When you are online, a yellow border appears in the upper part of the Edit Pane.



5 Select **Synchronize** from the **Controller** Menu.



6 The Synchronization Dialog Box is displayed.
Confirm that the check box for the data to transfer (i.e., **NJ301** on the figure on the right) is selected, and click **Transfer to Controller**.


Synchronization

	Computer: Data Name	Computer: Update Date	Controller: Update Date	Controller: Data Name	Compare
☑	NJ301	2022/10/31 11:18:25	2022/10/27 16:45:42	NJ301	

Legend: Synchronized | Different | Exists only on one side | Not checked

□ Clear the present values of variables with Retain attribute (Valid for Transfer to Controller).

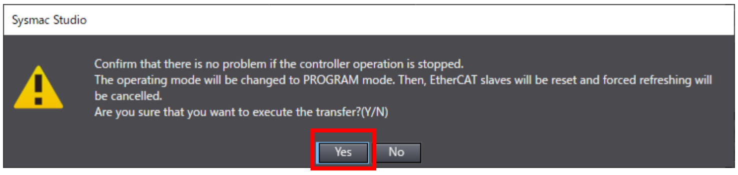
□ Do not transfer the POU program source (Valid for Transfer to Controller). All data will be re-transferred when this option is changed.

☑ Do not transfer the following. (All items are not transferred.)
- CJ-series Special Unit parameters and EtherCAT slave backup parameters.
- Slave Terminal Unit operation settings and NX Unit application data.

☑ Do not transfer the EtherNet/IP connection settings (i.e., tag data link settings).

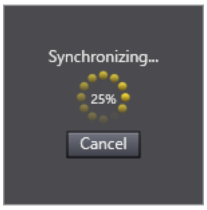
All data will be transferred because the projects in the computer and the controller are different.

7 A confirmation dialog box appears. Click **Yes**.

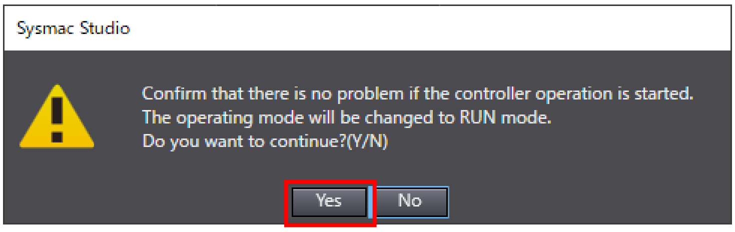

Sysmac Studio

Confirm that there is no problem if the controller operation is stopped.
The operating mode will be changed to PROGRAM mode. Then, EtherCAT slaves will be reset and forced refreshing will be cancelled.
Are you sure that you want to execute the transfer?(Y/N)

The Synchronizing Dialog Box appears.



A confirmation dialog box appears. Click **Yes**.

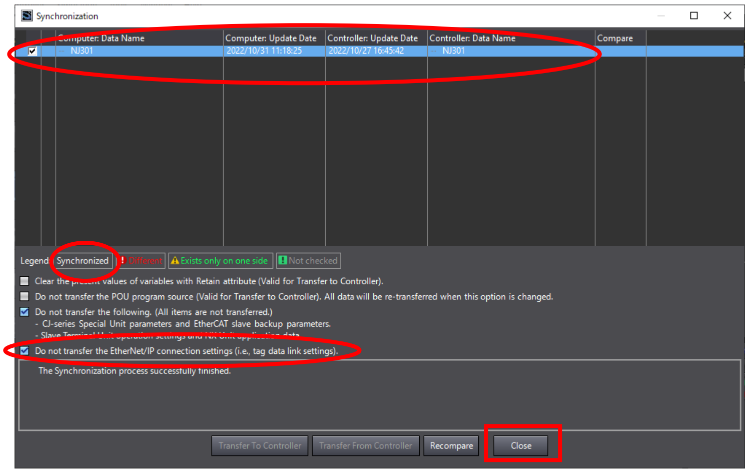

Sysmac Studio

Confirm that there is no problem if the controller operation is started.
The operating mode will be changed to RUN mode.
Do you want to continue?(Y/N)

8 Confirm that the synchronized data is now shown in the text color of **Synchronized** and the following message is displayed: *The Synchronization process successfully finished.*

If there is no problem, click **Close**.

* If synchronization fails, check the physical connections and redo the procedure.



Synchronization

	Computer: Data Name	Computer: Update Date	Controller: Update Date	Controller: Data Name	Compare
☑	NJ301	2022/10/31 11:18:25	2022/10/27 16:45:42	NJ301	

Legend Synchronized | ! Different | ⚠ Exists only on one side | ! Not checked

☐ Clear the present values of variables with Retain attribute (Valid for Transfer to Controller).

☐ Do not transfer the POU program source (Valid for Transfer to Controller). All data will be re-transferred when this option is changed.

☑ Do not transfer the following. (All items are not transferred.)
- CJ-series Special Unit parameters and EtherCAT slave backup parameters.
- Slave Terminal Unit operation settings and NX Unit application data

☑ Do not transfer the EtherNet/IP connection settings (i.e., tag data link settings).

The Synchronization process successfully finished.

7.4. Checking the Connection Status

Execute the transferred project file to check that Ethernet communications work correctly.

Precautions for Correct Use

Before performing the following steps, confirm that the LAN cable is connected securely.
If it is not connected, first turn OFF the power supply to the device and then connect the LAN cable.

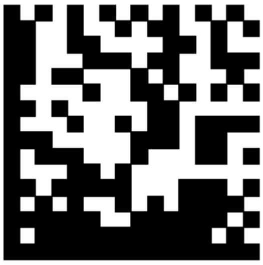
7.4.1. Executing the Project File and Checking the Receive Data

Execute the project file and check that correct data is written to controller variables.

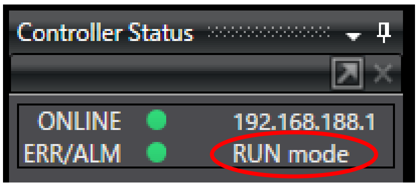
Precautions for Safe Use

Confirm the system safety before you execute the project file.
The connected devices may malfunction regardless of the operating mode of the unit, resulting in injury.

1 This document uses the 2D code shown in the right figure as an example of reading.
Set the code reader to the position where it can read the 2D code in the right figure.



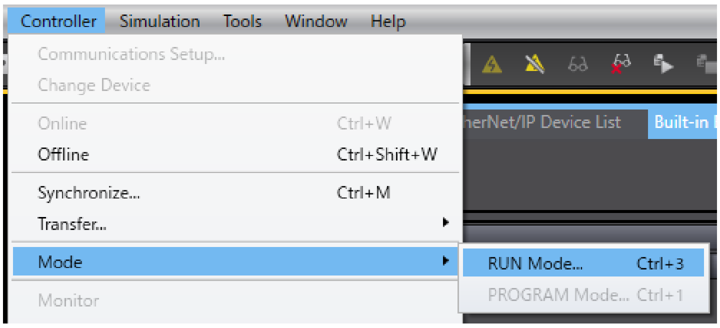
2 Confirm that the *RUN mode* is shown in the Controller Status Pane of the Sysmac Studio.



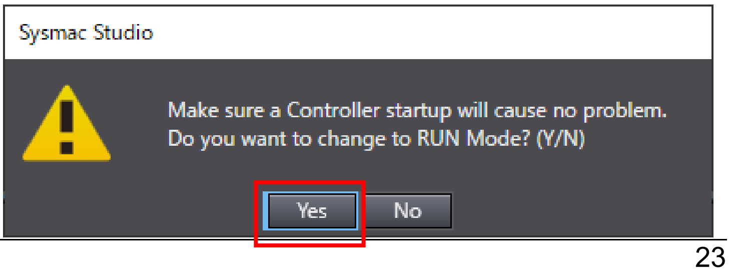
Controller Status

ONLINE ● 192.168.188.1
ERR/ALM ● RUN mode

If *PROGRAM mode* is shown, select **Mode – RUN Mode** from the **Controller** Menu.



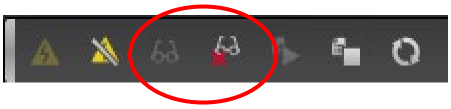
A confirmation dialog box appears. Click **Yes**.



Sysmac Studio

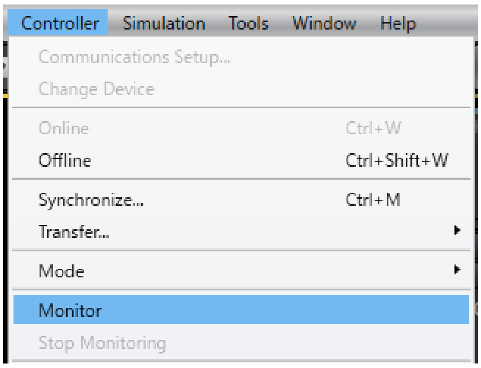
Make sure a Controller startup will cause no problem.
Do you want to change to RUN Mode? (Y/N)

3 Check that the controller is in a Monitor state by the **Monitor** and **Stop Monitoring** Buttons in the Sysmac Studio toolbar.
The controller is in a Monitor state if the **Monitor** Button is selected (not selectable) and the **Stop Monitoring** Button is selectable, as shown in the figure on the right.
* If the controller is in a Stop Monitoring state, select **Monitor** from the **Controller** Menu in the Sysmac Studio.

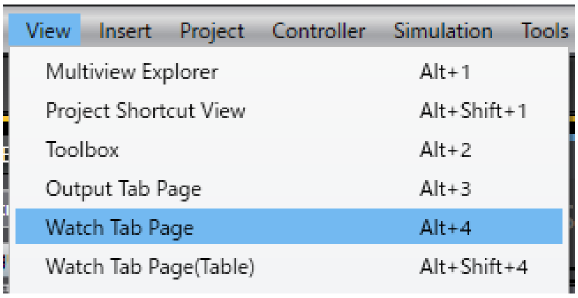


Monitor

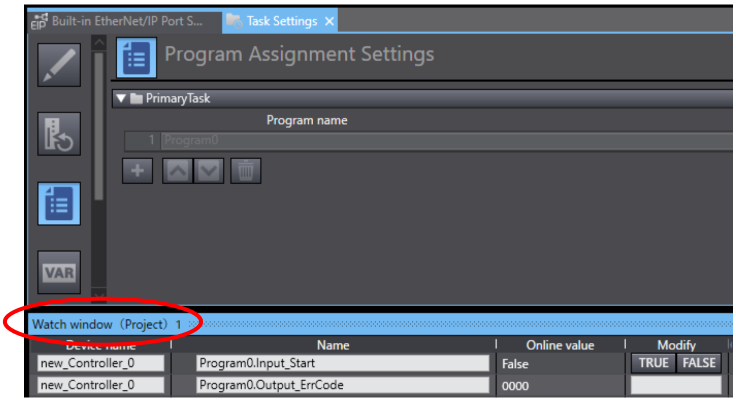
Stop Monitoring



4 Select **Watch Tab Page** from the **View** Menu.



5 The **Watch window** Tab Page is displayed under the Edit Pane.



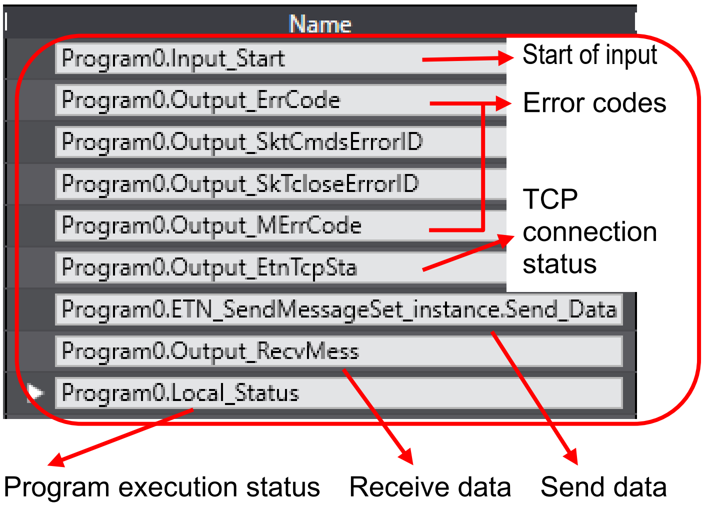
Watch window (Project) 1

Device name	Name	Online value	Modify
new_Controller_0	Program0.Input_Start	False	TRUE FALSE
new_Controller_0	Program0.Output_ErrCode	0000	

6 Confirm that the variables shown in the figure on the right are listed in the **Name** column.

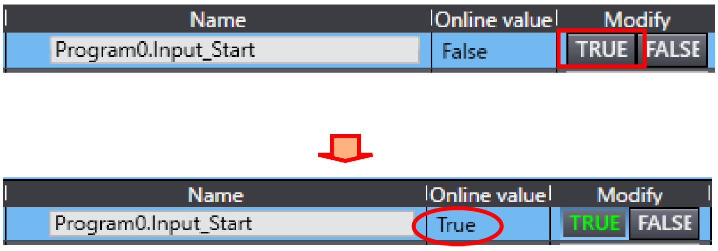
* If any of the required variables are not listed, click **Input Name** and add them.

* In the following description, "Program0" of the variable names in the Name column is omitted.



Name
Program0.Input_Start
Program0.Output_ErrCode
Program0.Output_SktCmdsErrorID
Program0.Output_SkTcloseErrorID
Program0.Output_MErrCode
Program0.Output_EtnTcpSta
Program0.ETN_SendMessageSet_instance.Send_Data
Program0.Output_RecvMess
Program0.Local_Status

7 Click **TRUE** in the **Modify** column of *Input_Start*. The **Online value** of *Input_Start* changes to *True*. The program starts running and the controller performs Ethernet communications with the code reader.

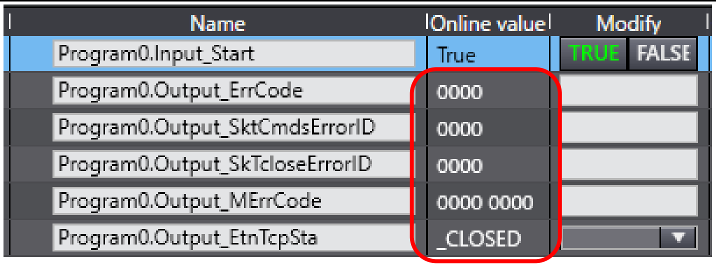


Name	Online value	Modify
Program0.Input_Start	False	TRUE FALSE

Name	Online value	Modify
Program0.Input_Start	True	TRUE FALSE

8 When the communications have ended normally, the values of the error codes are *0*. The value of the TCP connection status (Output_EtnTcpSta) is *_CLOSED*.

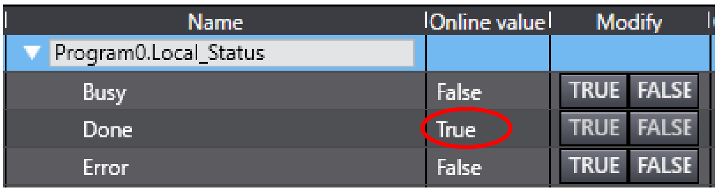
* If the program ends with an error, the error code will be stored according to the error that occurred. Refer to *9.7. Error Processing* for details on error codes.



Name	Online value	Modify
Program0.Input_Start	True	TRUE FALSE
Program0.Output_ErrCode	0000	
Program0.Output_SktCmdsErrorID	0000	
Program0.Output_SkTcloseErrorID	0000	
Program0.Output_MErrCode	0000 0000	
Program0.Output_EtnTcpSta	_CLOSED	▼

In addition, the **Online value** of *Local_Status.Done* indicating the program execution status is *True*. If the program ends with an error, the value of *Local_Status.Error* is *True*.

* If you click **FALSE** for *Input_Start*, the values of *Local_Status* also change to *False*. For more information, refer to *9.6. Timing Chart*.

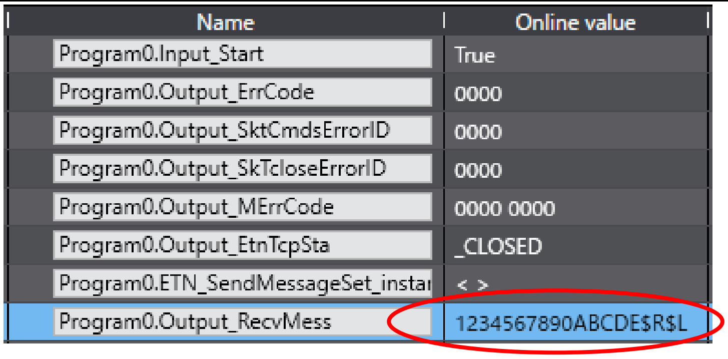


Name	Online value	Modify
▼ Program0.Local_Status		
Busy	False	TRUE FALSE
Done	True	TRUE FALSE
Error	False	TRUE FALSE

9 The response data received from the code reader is stored in *Output_RecvMess*. (ETN_SendMessageSet_instance.Send_Data is a send command.) Specify and check the referenced area in the Watch Tab Page, as shown in the figure on the right.

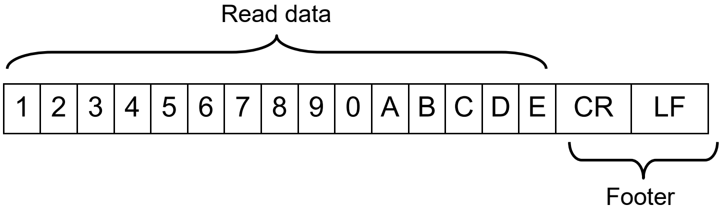
* The receive data in the figure on the right varies depending on your environment.

* For details on the command, refer to *9.2.2. Command Settings*.



Name	Online value
Program0.Input_Start	True
Program0.Output_ErrCode	0000
Program0.Output_SktCmdsErrorID	0000
Program0.Output_SkTcloseErrorID	0000
Program0.Output_MErrCode	0000 0000
Program0.Output_EtnTcpSta	_CLOSED
Program0.ETN_SendMessageSet_instai	< >
Program0.Output_RecvMess	1234567890ABCDERL

Response Format



Read data

1	2	3	4	5	6	7	8	9	0	A	B	C	D	E	CR	LF

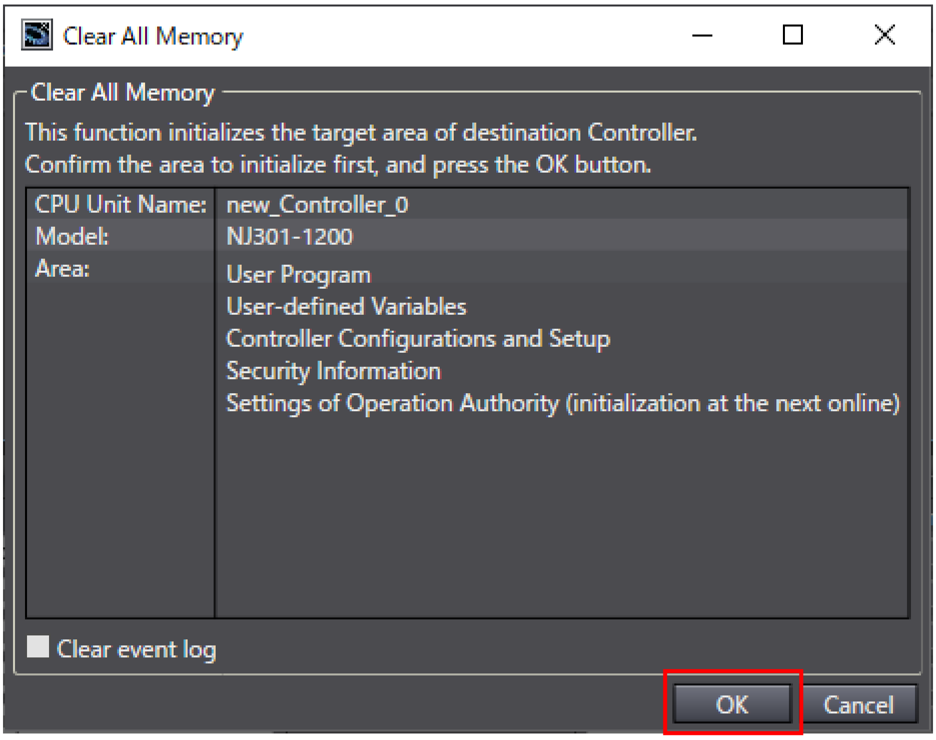
Footer

8. Initializing the System

This document assumes that each device uses the factory default settings.
If you change their settings from the defaults, you may not be able to perform various setting procedures as described.

8.1. Initializing the Controller

To initialize the controller, initialize the CPU Unit.
Before initialization, place the controller in PROGRAM mode, and select **Clear All Memory** from **Controller** Menu in the Sysmac Studio. When the Clear All Memory Dialog Box is displayed, confirm the contents and click **OK**.



Clear All Memory

This function initializes the target area of destination Controller.
Confirm the area to initialize first, and press the OK button.

CPU Unit Name:	new_Controller_0
Model:	NJ301-1200
Area:	User Program User-defined Variables Controller Configurations and Setup Security Information Settings of Operation Authority (initialization at the next online)

□ Clear event log

8.2. Initializing the Code Reader

For information on initializing the code reader, please refer to *How to initialize the settings?* in *Q&A* in *Appendices* of the *MicroHAWK V320-F/V330-F/V420-F/V430-F Series Barcode Reader User Manual* (Cat. No. Z432).

9. Project File

This section describes the details of the project file used in this document.

9.1. Overview

This section describes the specifications and functions of the project file used for connecting a V330-F/V430-F Series Code Reader (hereinafter referred to as “code reader”) to a controller’s built-in EtherNet/IP port (hereinafter referred to as “built-in EtherNet/IP port”).

“Project file” here refers to a Sysmac Studio Project File.
The project file contains the following data.

- Built-in EtherNet/IP port communication settings and program task settings
- Program and function blocks for socket communications
- Variable tables and data type definition of variables used in the ST language program

This project file uses the socket service function of the built-in EtherNet/IP port to execute the “< >” (Read trigger) command on the code reader and judges whether it reaches the normal end or error end.
In the project file, “normal end” means that TCP socket communications have ended normally. On the other hand,“error end” means that TCP socket communications have ended with an error.

The project file does not use the keep-alive and linger functions, which are TCP socket options. Consider using them as needed when designing your application.

Note

We have verified in our test configuration that the project file enables communications for the product versions and product lot used for evaluation.
However, we do not guarantee its operations where there are electrical noise or other disturbances, or variations in the performance of the devices themselves.

Note

In the Sysmac Studio, if it is necessary to distinguish between decimal data and hexadecimal data, add “*Variable Type* and #” to the beginning of the decimal data and *“Variable Type*, 16, and #” to the beginning of the hexadecimal data. (Example: INT#1000 for decimal data, INT#16#03E8 for hexadecimal data, etc. For DINT, “*Variable Type* and #” is not required.)

9.1.1. Communications Data Flow

This is the flow from issuing a TCP socket communications command from the built-in EtherNet/IP port to the code reader and receiving response data from the code reader. The project file executes a processing sequence of TCP open to TCP close in a continuous manner. If response data is divided and arrives as multiple pieces of receive data, receive processing will be repeated.

1.	TCP Open Processing	The built-in EtherNet/IP port issues a TCP open request to the code reader to establish a TCP connection.
	▼	
2.	Command Send Processing	The built-in EtherNet/IP port issues a send message that is set in the ST language program to the code reader.
	▼	
3.	Response Receive Processing	The built-in EtherNet/IP port stores the response data received from the code reader in the internal memory of the specified CPU Unit.
	▼	
4.	Close Processing	The built-in EtherNet/IP port issues a close request to the code reader to close the TCP connection.

* Depending on the code reader or the command used, response data may not be sent after the command is received or response data may be sent immediately after a connection is established. For this reason, this project file allows you to set whether or not send/receive processing is required in the Ethernet Communications Sequence Setting function block.
If *Send only* is set, response receive processing will not be executed. If *Receive only* is set, command send processing will not be executed.

9.1.2. TCP Socket Communications Using Socket Service Instructions

This section provides an overview of function blocks for TCP socket services (hereinafter referred to as “socket service instructions”) and the general movement of send and receive messages.

Note

For details, refer to *EtherNet/IP Communications Instructions* in *Section 2 Instruction Descriptions* of the *Machine Automation Controller NJ/NX-series Instructions Reference Manual* (Cat. No. W502).

● TCP Socket Services Using Socket Service Instructions

This project file uses the following five standard instructions to implement socket communications.

Name	Function block	Description
TCP Socket Connect	SktTCPConnect	Connects to a TCP port on the code reader by active open.
TCP Socket Send	SktTCPSend	Sends data from the specified TCP socket.
TCP Socket Receive	SktTCPRcv	Reads data received from the specified TCP socket.
TCP/UDP Socket Close	SktClose	Closes the specified TCP socket.
Get TCP Socket Status	SktGetTCPStatus	Reads the status of the specified TCP socket. The project file uses this instruction to check the completion of receiving in receive processing and to check the closed status in close processing.

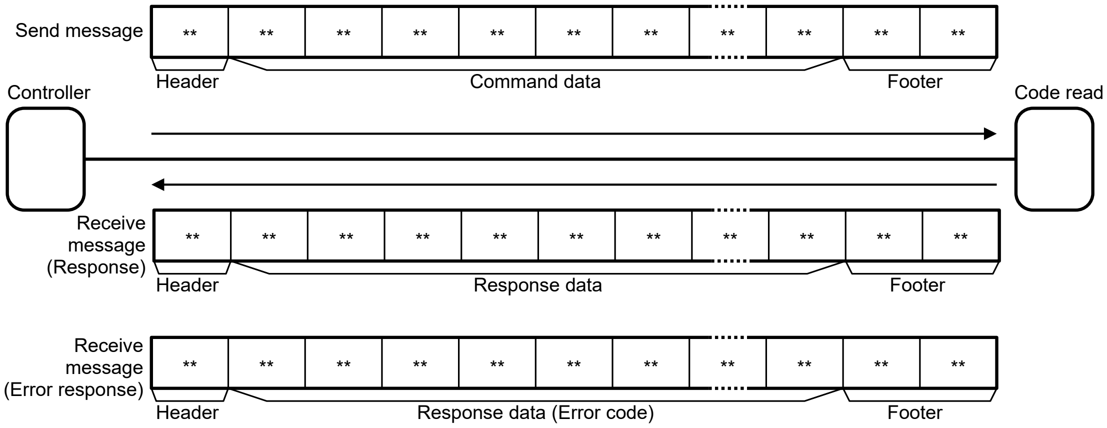
* The *Socket* obtained by the Connect TCP Socket instruction (SktTCPConnect: SktTCPConnect_instance) is used as an input parameter for other socket service instructions. The specifications of the data type structure _sSOCKET of *Socket* are as follows.

Variable	Name	Description	Data type	Valid range	Initial value
Socket	Socket	Socket	_sSOCKET	---	---
Handle	Handle	Handle for sending/receiving data	UDINT	Depends on data type.	---
SrcAdr	Source Address	Local node address*1	_sSOCKET_ ADDRESS	---	---
PortNo	Port No.	Port number	UINT	0 to 65535	
IpAdr	IP Address	IP address or host name*2	STRING	Depends on data type.	
DstAdr	Destination Address	Remote node address*1	_sSOCKET_ ADDRESS	---	---
PortNo	Port No.	Port number	UINT	1 to 65535	
IpAdr	IP Address	IP address or host name*2	STRING	Depends on data type.	

*1: “Address” refers to an IP address and a port number.

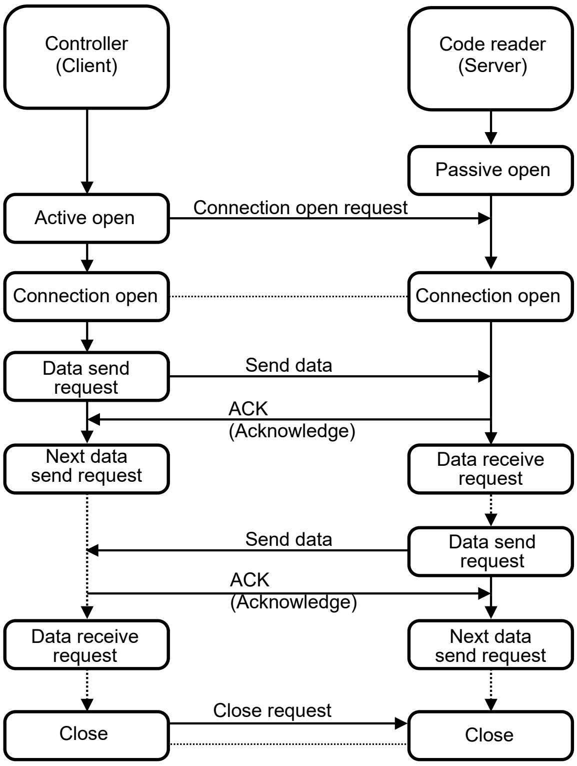
*2: DNS or Hosts settings are required to use a host name.

- Send and Receive Messages



- Communications Sequence

The figure below shows the processing flow of TCP communications between the code reader (server) and the controller (client).



9.2. Code Reader Command

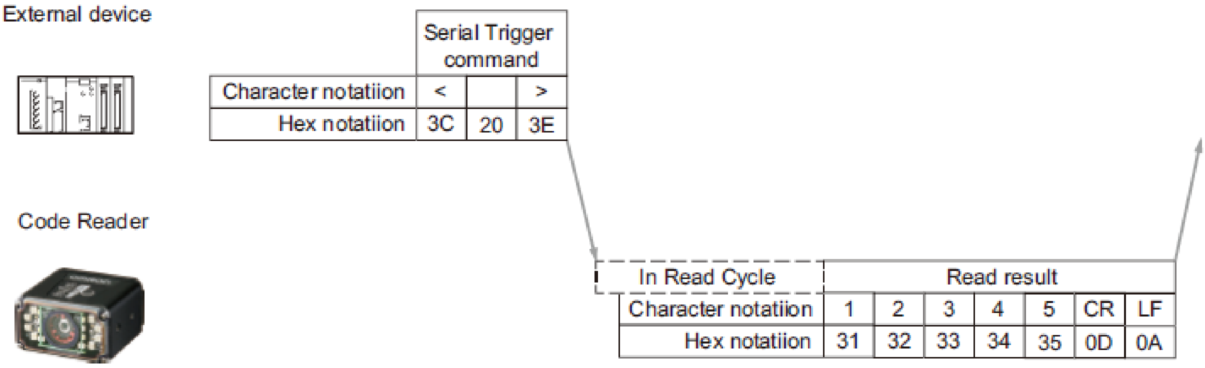
This section describes the code reader command in the project file.

9.2.1. Command Overview

This project file uses the “< >” (Read trigger) command to trigger Ethernet communications with the code reader. The code reader sends the read data back to the controller.

Command	Description
< >	Read trigger

- Read string: 12345, Character (Delimited): Space, Preamble: None, Postamble: CRLF



	Serial Trigger command		
Character notatiion	<		>
Hex notatiion	3C	20	3E

In Read Cycle	Read result						
Character notatiion	1	2	3	4	5	CR	LF
Hex notatiion	31	32	33	34	35	0D	0A

Note

For more information, please refer to *Change the Command that Executes Read* in *3-2 Communications Settings (Serial(TCP))* in the *Autofocus Multicode Reader MicroHAWK V320-F/V330-F/V420-F/V430-F Series User Manual* (Cat. No. Z407).

9.2.2. Command Settings

This section describes in detail the settings of the “< >” (Read trigger) command.

- Send Data (Command) Settings

Send data is set by the function block SendMessageSet_instance.

Code Reader Specifications:

• The data is stored in ASCII code.

Variable	Setting (Data format)	Setting
Send_Header	Send header (STRING[5])	‘’(None)
Send_Addr	Send address (STRING[5])	‘’(None)
Send_Command	Send data (STRING[256])	“< >”
Send_Check	Send check addition (STRING[5])	‘’(None)
Send_Terminate	Send terminator (STRING[5])	‘’(None)

Variable	Setting (Data format)	Data	Description
Send_Data	Send message (STRING[256])	CONCAT(Send_Header, Send_Addr, Send_Command, Send_Check, Send_Terminate)	Used as send data for the SktTCPSend instruction (SktTCPSend_ instance).

- Stored Contents of Receive Data (Response)

Receive data is stored as output receive data after a data check by the function block ReceiveCheck_instance.

Code Reader Specifications:

• The data is stored in ASCII code.

Variable	Setting (Data format)	Description of storage area
Recv_Data	Receive data (STRING[256])	Receive buffer
Recv_Buff	Receive data (STRING[256])	Receive data storage area (Stores receive buffer data as is.)

- Send and Receive Messages

Send message	3C	20	3E
	'<'	' '	'>'

(Normal processing: Decoded string)

Receive message	31	32	33	34	35	36	37	38	39	30	0D	0A
	‘1’	‘2’	‘3’	‘4’	‘5’	‘6’	‘7’	‘8’	‘9’	‘0’	CR	LF

(Error processing)

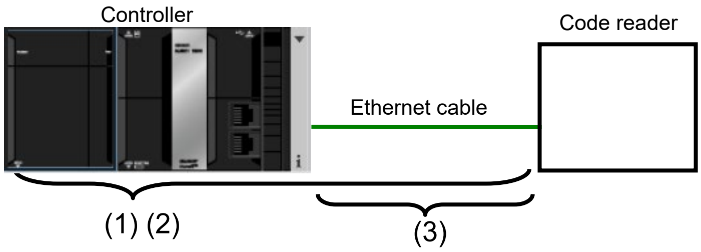
Receive message (None)

9.3. Error Judgment Processing

This section describes error judgment processing in the project file.

9.3.1. Error Judgment in the Project File

In this project file, error judgment processing is executed for the following three types of errors (1) to (3). Refer to *9.7.1. Error Code List* for information on error codes.



(1) Communications error during TCP socket communications using socket service instructions

An error that was detected by a program in TCP socket communications, such as a communications hardware error, command format error, or parameter error, is judged as a "communications error". This judgment is made based on the socket service instruction argument "ErrorID".

(2) Timeout error during communications with the code reader

An error that occurred due to abnormal open, send, receive, or close processing that failed to complete within the monitoring time is judged as a timeout error. This judgment is made based on timer monitoring in the project file. Refer to *9.3.2. Time Monitoring Function* for information on time monitoring using the internal timers of the project file.

(3) TCP connection status error at end of processing

The project file uses a procedure in which the overall processing ends after the last close processing is done, regardless of whether the open to receive processing steps have ended normally or ended with an error. Therefore, judgment of whether close processing has ended normally is made based on the TCP connection status variable *TcpStatus* in the SktGetTCPStatus instruction. If there is an error in close processing, the next open processing may not be executed correctly. Refer to *9.7.2. TCP Connection Status Error Situation and Correction* for information on how to correct a TCP connection status error.

9.3.2. Time Monitoring Function

This section describes the time monitoring function in the project file.
The monitoring time settings can be changed by using variables in the function block *ParameterSet*.

● Time Monitoring Using Internal Timers of the Project File

Assuming that processing has the executing status and does not end due to an error, the project file uses its internal timers to interrupted the processing (i.e., timeout). The timeout is set to *5* s (default) for each processing phase from open to close.

Time Monitoring Using Internal Timers of the Project File

Processing	Monitoring description	Variable name	Timeout (default)
Open processing	Time from start to end of open processing	TopenTime	After 5 s (UINT#500)
Send processing	Time from start to end of send processing	TfsTime	After 5 s (UINT#500)
Receive processing	Time from start to end of receive processing * If receive processing is repeated, the software part monitors the time for each repetition of receive processing.	TfrTime	After 5 s (UINT#500)
Close processing	Time from start to end of close processing * The software part checks that the TCP connection status is normal after close processing to judge the end of the processing.	TcloseTime	After 5 s (UINT#500)

● Time Monitoring Using the Built-in EtherNet/IP Port (Socket Service)

The built-in EtherNet/IP port has a time monitoring function for receive data that arrives in segments, as a socket service. In receiving processing, it stores the *TimeOut* parameter of the socket service instruction SktTCPRcv_instance to *TrTime=UINT#3(300ms)* (initial value). The project file also sets the variable *TrTime* as the Receive Wait Time Monitoring Timer for the next response receive wait time after completion of receiving a response. If the next response from the code reader does not arrive within this time, it will be judged that the receive processing has ended.

Note

For information on time monitoring using the socket service, refer to *SktTCPRcv Instruction* in *Section 2 Instruction Descriptions* of the *Machine Automation Controller NJ/NX-series Instructions Reference Manual* (Cat. No. W502).

● Resending and Time Monitoring Using the Built-in EtherNet/IP Port (TCP/IP)

If a communications error occurs, TCP/IP automatically resends the data and monitors the processing time if there is no problem with the built-in EtherNet/IP port. If processing ends with an error in the middle of it, the project file stops the resending and time monitoring via TCP/IP in close processing. However, if the close processing shows a TCP connection status error, the resending and time monitoring via TCP/IP may continue to be active in the built-in EtherNet/IP port. Refer to *9.7.2. TCP Connection Status Error Situation and Correction* for information on the error situation and correction.

9.4. Variables Used

This section describes variables used in the project file.

9.4.1. Lists of Variables Used

Below are lists of variables required in order to execute this project file.

● Input Variable

The following variable is used to manipulate the project file.

Variable name	Data type	Description
Input_Start	BOOL	Executes the project file when the value changes from OFF (*FALSE*) to ON (*TRUE*). The value changes from ON to OFF after the check of normal end or error end output.

● Output Variables

The following variables reflect the execution results of the project file.

Variable name	Data type	Description
Output_RecvMess	STRING[256]	Stores receive data (response). (An area of 256 words is secured.)
Output_ErrCode	WORD	Stores the error result (flag) for a communications error or timeout error detected during open processing, send processing, receive processing, and close processing. *#0000* is stored when the processing ends normally.
Output_ SktCmdsErrorID	WORD	Stores the error code for a communications error or timeout error detected for each socket service instruction in open processing, send processing, and receive processing. *#0000* is stored when the processing ends normally.
Output_ SkTcloseErrorID	WORD	Stores the error code for a communications error or timeout error detected for the SktTcpClose instruction in close processing, aside from errors in open processing, sending processing, and receiving processing. *#0000* is stored when the processing ends normally.
Output_ EtnTcpSta	_eCONNECTION _STATE	Stores the TCP connection status when a communications error or timeout error is detected in close processing. *_CLOSED* is stored when the processing ends normally.
Output_MErrCode	DWORD	Stores the error code of an FCS calculation error or code reader error detected as a result of receive processing. *#00000000* is stored when the processing ends normally.

- Internal Variables

The following variables are used only for the purpose of calculation in the project file.

Variable name		Data type	Description
Local_Status		sStatus (STRUCT)	Program execution status
	Busy	BOOL	Changes to *TRUE* when the project file is executed and to *FALSE* when it is not executed.
	Done	BOOL	Changes to TRUE when the project file ends normally and to FALSE when *Input_Start* changes from *TRUE* to *FALSE*.
	Error	BOOL	Changes to *TRUE* when the project file ends with an error and to *FALSE* when *Input_Start* changes from *TRUE* to *FALSE*.
Local_State		DINT	State Processing No.
Local_ErrCode		uErrorFlgs (UNION)	Sets an error code.
	Local_ErrCode. WordData	WORD	Expresses the error code as WORD data.
	Local_ErrCode. BoolData	ARRAY [0..15] OF BOOL	• Communications error BoolData[0]: Send processing: Error (*TRUE*)/Normal (*FALSE*) BoolData[1]: Receive processing: Error (*TRUE*)/Normal (*FALSE*) BoolData[2] Open processing: Error (*TRUE*)/Normal (*FALSE*) BoolData[3]: Close processing: Error (*TRUE*)/Normal (*FALSE*) BoolData[4]: Processing number: Error (*TRUE*)/Normal (*FALSE*) • Timeout error BoolData[8]: Send processing: Error (*TRUE*)/Normal (*FALSE*) BoolData[9]: Receive processing: Error (*TRUE*)/Normal (*FALSE*) BoolData[10] Open processing: Error (*TRUE*)/Normal (*FALSE*) BoolData[11]: Close processing: Error (*TRUE*)/Normal (*FALSE*) • Others BoolData[5]: Send/Receive required judgment error: Error (*TRUE*)/Normal (*FALSE*) BoolData[12]: Code reader error: Error (*TRUE*)/Normal (*FALSE*) BoolData[6..7],[13..14]: Reserved BoolData[15]: Error occurred

Variable name	Data type	Description
Local_ExecFlgs	sControl (STRUCT)	Socket service instruction execution flag
Send	BOOL	Send Processing instruction: Executed (*TRUE*)/Not executed (*FALSE*)
Recv	BOOL	Receive Processing instruction: Executed (*TRUE*)/Not executed (*FALSE*)
Open	BOOL	Open Processing instruction: Executed (*TRUE*)/Not executed (*FALSE*)
Close	BOOL	Close Processing instruction: Executed (*TRUE*)/Not executed (*FALSE*)
Status	BOOL	TCP Status instruction: Executed (*TRUE*)/Not executed (*FALSE*)
Local_SrcDataByte	UINT	Sets the number of bytes of send data.
Local_SrcData	ARRAY [0..2000] OF BYTE	Send data storage area for SktTCPSend instruction (SktTCPSend_instance). (An area of 256 words is secured.)
Local_RecvData	ARRAY [0..2000] OF BOOL	Receive data (response) storage area for SktTCPRcv instruction (SktTCPRcv_instance). (An area of 256 words is secured.)
Local_ ReceiveMessage	STRING[256]	Local_RecvDataReceived string data (response) storage area. (An area of 256 characters is secured.)
Local_ RecvCheckFlg	BOOL	Code Reader Error Judgment Instruction Execution Flag: Executed (*TRUE*)/Not executed (*FALSE*)
Local_ InitialSettingOK	BOOL	Initialization Normal Setting Flag
Local_TONFlgs	sTimerControl (STRUCT)	Timer Execution Flag
Tfs	BOOL	Send Processing Time Monitoring Timer Instruction: Executed (*TRUE*)/Not executed (*FALSE*)
Tfr	BOOL	Receive Processing Time Monitoring Timer Instruction: Executed (TRUE)/Not executed (FALSE)
Topen	BOOL	Open Processing Time Monitoring Timer Instruction: Executed (*TRUE*)/Not executed (*FALSE*)
Tclose	BOOL	Close Processing Time Monitoring Timer Instruction: Executed (*TRUE*)/Not executed (*FALSE*)
Tr	BOOL	Next Response Receive Wait Time Monitoring Timer Instruction: Executed (*TRUE*)/Not executed (*FALSE*)
Local_ComType	sControl (STRUCT)	Sets whether or not send processing or receive processing is required.
Send	BOOL	Send processing: Required (*TRUE*)/Not required (*FALSE*) * If send processing is required, but receive processing is not required: The program will skip receive processing and go to close processing without waiting for receive data in send processing. Specify this value when response data is not sent back to the command sent.

Variable name		Data type	Description
	Recv	BOOL	Receive processing: Required (*TRUE*)/Not required (*FALSE*) * If both send processing and receive processing are required: The program will wait for the arrival of receive data after send processing. The program will go to receive processing after checking the arrival of receive data. Specify this value when response data is sent back to the command sent.
	Error	BOOL	Send/Receive Processing Required Setting Error Flag (This flag is set if there is a setting error.)

● Variables for Initializing Socket Service Instructions

Variable name	Data type	Description
NULL_SOCKET	_sSOCKET	Internal socket service instruction initialization data (Retain constants: Enabled) Initial value (Handle:=0, SrcAdr:=(PortNo:=0, IpAdr:=''), DstAdr:=(PortNo:=0, IpAdr:='')) (Used for all socket instructions.)
NULL_ ARRAYOFBYTE_1	ARRAY [0..0] OF BYTE	Internal send socket service instruction initialization data (Retain constants: Enabled) Initial value [0] (Use for the SktTCPSend instruction)
NULL_ ARRAYOFBYTE_2	ARRAY [0..0] OF BYTE	Internal receive socket service instruction initialization data (Retain constants: Disabled) Initial value [0] (Use for the SktTCPRcv instruction)

9.4.2. Lists of Variables Used in User-defined Function Blocks/Functions

Below are lists of function blocks that must be user-defined in programs in order to execute this project file.

For information on the following function block variables, refer to *9.5.3. Detailed Explanation of Function Blocks*.

Variable name	Data type	Description
ETN_ParameterSet_instance	ParameterSet	Ethernet settings (Remote IP address, etc.) Monitoring time from open processing to close processing
ETN_SendMessageSet_instance	SendMessageSet	Send/receive processing required setting and send message setting.
ETN_ReceiveCheck_instance	ReceiveCheck	Receive data storage and normal/error judgment

- Timers

The following timers are used in the project file.

Variable name	Data type	Description
Topen_TON_instance	TON	Measures the monitoring time for open processing.
Tfs_TON_instance	TON	Measures the monitoring time for send processing.
Tfr_TON_instance	TON	Measures the monitoring time for receive processing.
Tclose_TON_instance	TON	Measures the monitoring time for close processing.
Tr_TON_instance	TON	Measures the processing time for the next response receive wait time.

9.4.3. Lists of System-defined Variables

Below are lists of variables required in order to execute this project file.

- System-defined Variables (External Variables)

Variable name	Data type	Description
_EIP_EtnOnlineSta	BOOL	Built-in EtherNet/IP port's communications status: *TRUE*: Available, *FALSE*: Not available

Note

For information on system variables and communications instructions, refer to *EtherNet/IP Communications Instructions* in *Section 2 Instruction Descriptions* of the *Machine Automation Controller NJ/NX-series Instructions Reference Manual* (Cat. No. W502).

9.5. Programs (ST Language)

9.5.1. Functional Components of the ST Language Program

This project file is written in the ST language. The functional components of the project file are as follows.

Category	Subcategory	Description
1. Communications Processing	1.1. Communications Processing Start 1.2. Communications Processing Status Flag String Clearing 1.3. Communications Processing Executing Status	Executes communications processing.
2. Initialization	2.1. Processing Time Monitoring Timer Initialization 2.2. Socket Service Instruction Initialization 2.3. Socket Service Instruction Execution Flag Initialization 2.4. Processing Time Monitoring Timer Execution Flag Initialization 2.5. Error Code Storage Area Initialization 2.6. Processing Monitoring Time Setting and Ethernet-related Parameter Setting 2.7. Send/Receive Processing Required Setting and Send Data Setting 2.8. Send Data Conversion from String to Byte Array 2.9. Receive Data Storage Area Initialization 2.10. Initialization End Processing	Sets Ethernet parameters and initializes the error code storage area. Sets whether or not the send/receive processing is required, send data, and receive data.
3. Open Processing	3.1. Open Processing Status Judgment and Execution Flag Setting 3.2. Open Processing Time Monitoring Timer Execution 3.3. Open Instruction Execution (TCP Active Open Processing)	Executes TCP open (active) processing. Processing starts after communications processing is started and initial setup is done.

Category	Subcategory	Description
4. Send Processing	4.1. Send Processing Status Judgment and Execution Flag Setting 4.2. Send Processing Time Monitoring Timer Execution 4.3. Send Instruction Execution	Starts processing if the Send Processing Required Flag is set to *Required* and open processing has ended normally.
5. Receive Processing	5.1. Receive Processing Status Judgment and Execution Flag Setting 5.2. Receive Wait Time Monitoring Timer Execution 5.3. Receive Processing Time Monitoring Timer Execution 5.4. Receive Instruction Execution 5.5. Get TCP Status Processing Execution 5.6. Code Reader Error Judgment Instruction Execution	Starts processing if the Receive Processing Required Flag is set to *Required* and send processing has ended normally. If receive data arrives in segments, receive processing is repeated. Stores and checks the receive data.
6. Close Processing	6.1. Close Processing Status Judgment and Execution Flag Setting 6.2. Close Processing Time Monitoring Timer Execution 6.3. Close Instruction Execution 6.4. Get TCP Status Processing Execution	Executes close processing. Processing starts in the following cases. • Receive Processing Required Flag is set to *Not required* and send processing has ended normally. • Receive processing ends normally. • Open processing, send processing, or receive processing ends with an error.
7. Processing Error Processing	7. Processing No. Error Processing	Executes error processing if a non-existent processing number is detected.

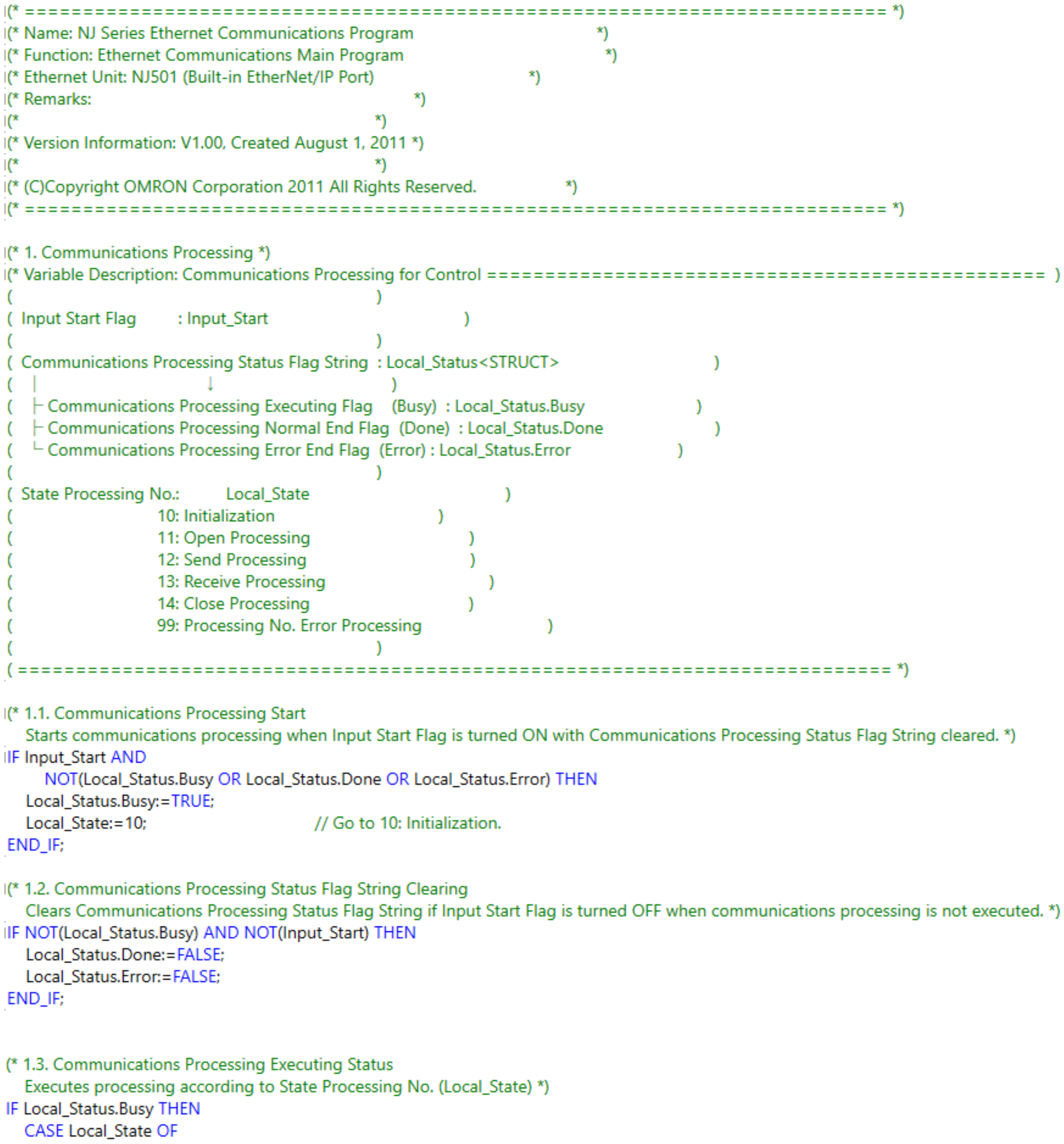
9.5.2. Detailed Explanation of the Main Program

A detailed explanation of the project file is given below.

Communication settings that need to be changed depending on the code reader, send data (command) settings, and receive data (response data) are checked in function blocks (ETN_ParameterSet_instance, ETN_SendMessageSet_instance, and ETN_ReceiveCheck_instance). For how to change the values of these settings, refer to *9.5.3 Detailed Explanation of Function Blocks*.

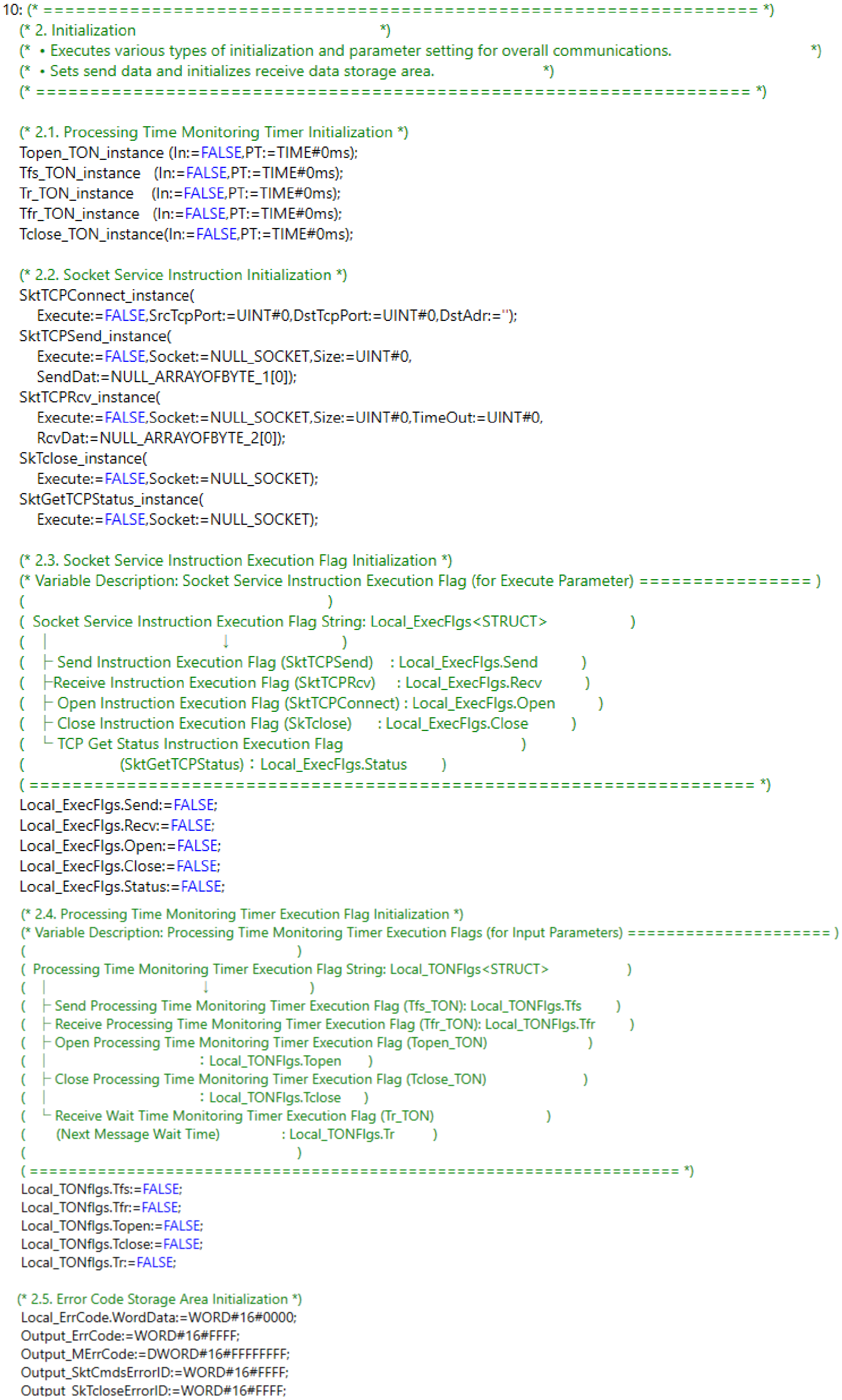
Main Program: Program0

1. Communications Processing

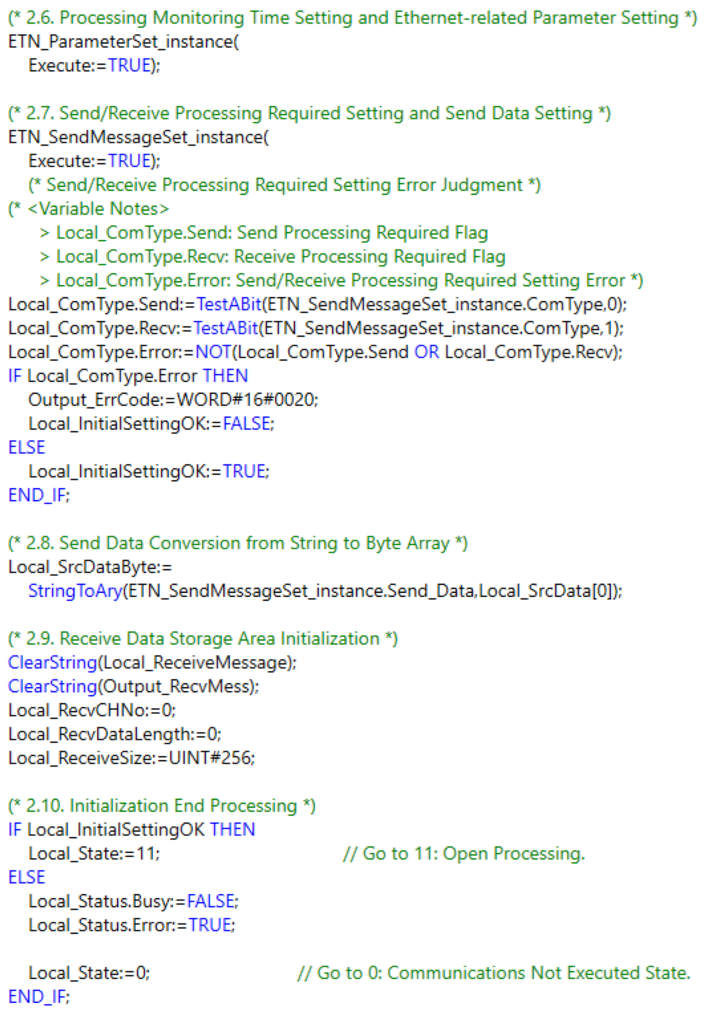


```
(* ======================================================================== *)
(* Name: NJ Series Ethernet Communications Program                          *)
(* Function: Ethernet Communications Main Program                            *)
(* Ethernet Unit: NJ501 (Built-in EtherNet/IP Port)                    *)
(* Remarks:                                                    *)
(*                                                         *)
(* Version Information: V1.00, Created August 1, 2011 *)
(*                                                         *)
(* (C)Copyright OMRON Corporation 2011 All Rights Reserved.               *)
(* ======================================================================== *)

(* 1. Communications Processing *)
(* Variable Description: Communications Processing for Control ============================================== )
(                                                          )
(  Input Start Flag         : Input_Start                          )
(                                                          )
(  Communications Processing Status Flag String  : Local_Status<STRUCT>                    )
(   |                          ↓                           )
(   ├Communications Processing Executing Flag    (Busy)  : Local_Status.Busy                  )
(   ├Communications Processing Normal End Flag  (Done)  : Local_Status.Done                  )
(   └Communications Processing Error End Flag  (Error) : Local_Status.Error               )
(                                                          )
(  State Processing No.:         Local_State                               )
(                         10: Initialization                         )
(                         11: Open Processing                          )
(                         12: Send Processing                          )
(                         13: Receive Processing                         )
(                         14: Close Processing                         )
(                         99: Processing No. Error Processing                  )
(                                                          )
( ======================================================================== *)

(* 1.1. Communications Processing Start
   Starts communications processing when Input Start Flag is turned ON with Communications Processing Status Flag String cleared. *)
IF Input_Start AND
     NOT(Local_Status.Busy OR Local_Status.Done OR Local_Status.Error) THEN
   Local_Status.Busy:=TRUE;
   Local_State:=10;                          // Go to 10: Initialization.
END_IF;

(* 1.2. Communications Processing Status Flag String Clearing
   Clears Communications Processing Status Flag String if Input Start Flag is turned OFF when communications processing is not executed. *)
IF NOT(Local_Status.Busy) AND NOT(Input_Start) THEN
   Local_Status.Done:=FALSE;
   Local_Status.Error:=FALSE;
END_IF;


(* 1.3. Communications Processing Executing Status
   Executes processing according to State Processing No. (Local_State) *)
IF Local_Status.Busy THEN
   CASE Local_State OF
```

2. Initialization

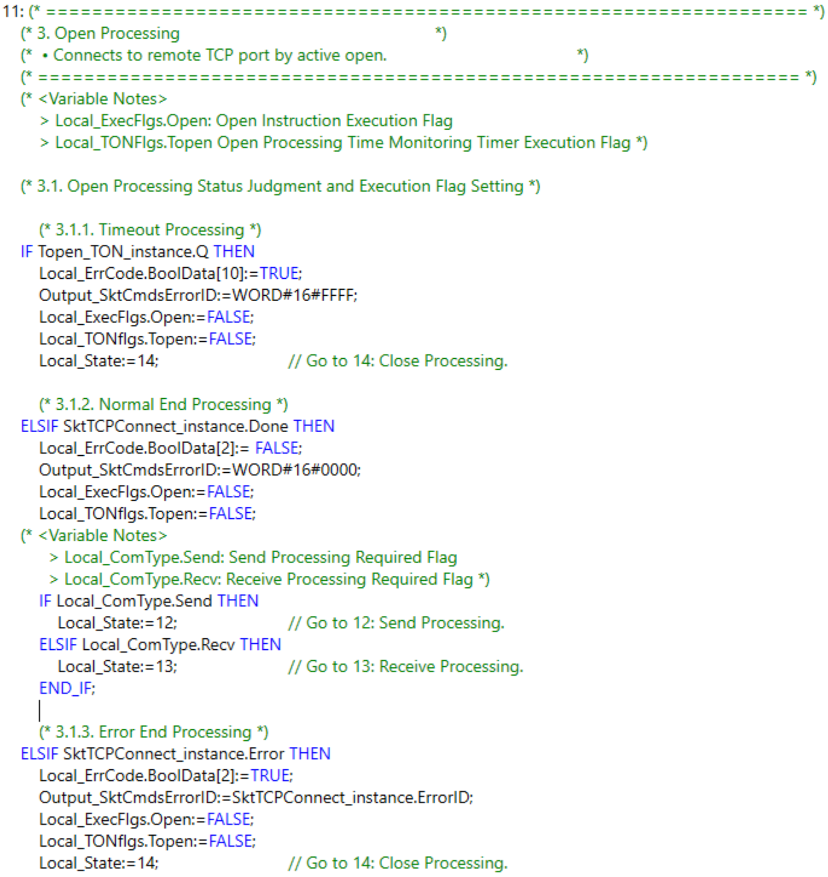


```
10: (* ================================================================= *)
  (* 2. Initialization                                        *)
  (*  • Executes various types of initialization and parameter setting for overall communications.                *)
  (*  • Sets send data and initializes receive data storage area.                  *)
  (* ================================================================= *)

  (* 2.1. Processing Time Monitoring Timer Initialization *)
  Topen_TON_instance (In:=FALSE,PT:=TIME#0ms);
  Tfs_TON_instance   (In:=FALSE,PT:=TIME#0ms);
  Tr_TON_instance    (In:=FALSE,PT:=TIME#0ms);
  Tfr_TON_instance   (In:=FALSE,PT:=TIME#0ms);
  Tclose_TON_instance(In:=FALSE,PT:=TIME#0ms);

  (* 2.2. Socket Service Instruction Initialization *)
  SktTCPConnect_instance(
     Execute:=FALSE,SrcTcpPort:=UINT#0,DstTcpPort:=UINT#0,DstAdr:='');
  SktTCPSend_instance(
     Execute:=FALSE,Socket:=NULL_SOCKET,Size:=UINT#0,
     SendDat:=NULL_ARRAYOFBYTE_1[0]);
  SktTCPRcv_instance(
     Execute:=FALSE,Socket:=NULL_SOCKET,Size:=UINT#0,TimeOut:=UINT#0,
     RcvDat:=NULL_ARRAYOFBYTE_2[0]);
  SkTclose_instance(
     Execute:=FALSE,Socket:=NULL_SOCKET);
  SktGetTCPStatus_instance(
     Execute:=FALSE,Socket:=NULL_SOCKET);

  (* 2.3. Socket Service Instruction Execution Flag Initialization *)
  (* Variable Description: Socket Service Instruction Execution Flag (for Execute Parameter) ================ )
  (                                                  )
  (  Socket Service Instruction Execution Flag String: Local_ExecFlgs<STRUCT>                  )
  (    |                              ↓                     )
  (    ├ Send Instruction Execution Flag (SktTCPSend)    : Local_ExecFlgs.Send          )
  (    ├Receive Instruction Execution Flag (SktTCPRcv)     : Local_ExecFlgs.Recv          )
  (    ├ Open Instruction Execution Flag (SktTCPConnect) : Local_ExecFlgs.Open          )
  (    ├ Close Instruction Execution Flag (SkTclose)      : Local_ExecFlgs.Close          )
  (    └ TCP Get Status Instruction Execution Flag                              )
  (                  (SktGetTCPStatus) : Local_ExecFlgs.Status        )
  ( ================================================================= *)
  Local_ExecFlgs.Send:=FALSE;
  Local_ExecFlgs.Recv:=FALSE;
  Local_ExecFlgs.Open:=FALSE;
  Local_ExecFlgs.Close:=FALSE;
  Local_ExecFlgs.Status:=FALSE;

  (* 2.4. Processing Time Monitoring Timer Execution Flag Initialization *)
  (* Variable Description: Processing Time Monitoring Timer Execution Flags (for Input Parameters) ==================== )
  (                                                  )
  (  Processing Time Monitoring Timer Execution Flag String: Local_TONFlgs<STRUCT>                  )
  (    |                              ↓                     )
  (    ├ Send Processing Time Monitoring Timer Execution Flag (Tfs_TON): Local_TONFlgs.Tfs        )
  (    ├ Receive Processing Time Monitoring Timer Execution Flag (Tfr_TON): Local_TONFlgs.Tfr        )
  (    ├ Open Processing Time Monitoring Timer Execution Flag (Topen_TON)                   )
  (    |                             : Local_TONFlgs.Topen       )
  (    ├ Close Processing Time Monitoring Timer Execution Flag (Tclose_TON)                   )
  (    |                             : Local_TONFlgs.Tclose      )
  (    └ Receive Wait Time Monitoring Timer Execution Flag (Tr_TON)                      )
  (      (Next Message Wait Time)              : Local_TONFlgs.Tr         )
  (                                                  )
  ( ============================================================== *)
  Local_TONflgs.Tfs:=FALSE;
  Local_TONflgs.Tfr:=FALSE;
  Local_TONflgs.Topen:=FALSE;
  Local_TONflgs.Tclose:=FALSE;
  Local_TONflgs.Tr:=FALSE;

  (* 2.5. Error Code Storage Area Initialization *)
  Local_ErrCode.WordData:=WORD#16#0000;
  Output_ErrCode:=WORD#16#FFFF;
  Output_MErrCode:=DWORD#16#FFFFFFFF;
  Output_SktCmdsErrorID:=WORD#16#FFFF;
  Output_SkTcloseErrorID:=WORD#16#FFFF;
```

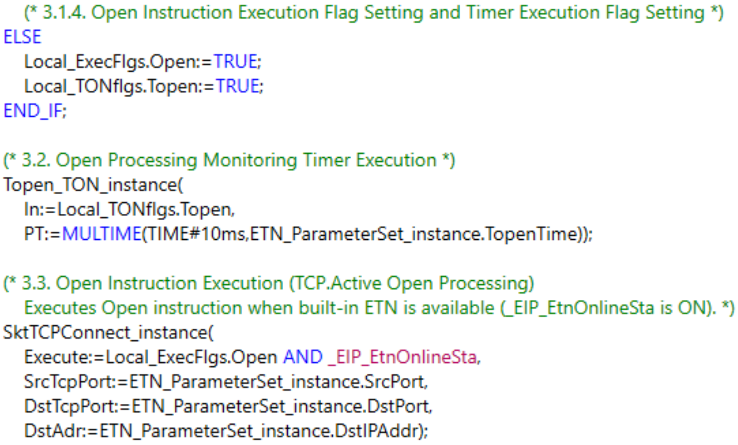


```
(* 2.6. Processing Monitoring Time Setting and Ethernet-related Parameter Setting *)
ETN_ParameterSet_instance(
    Execute:=TRUE);

(* 2.7. Send/Receive Processing Required Setting and Send Data Setting *)
ETN_SendMessageSet_instance(
    Execute:=TRUE);
    (* Send/Receive Processing Required Setting Error Judgment *)
(* <Variable Notes>
      > Local_ComType.Send: Send Processing Required Flag
      > Local_ComType.Recv: Receive Processing Required Flag
      > Local_ComType.Error: Send/Receive Processing Required Setting Error *)
Local_ComType.Send:=TestABit(ETN_SendMessageSet_instance.ComType,0);
Local_ComType.Recv:=TestABit(ETN_SendMessageSet_instance.ComType,1);
Local_ComType.Error:=NOT(Local_ComType.Send OR Local_ComType.Recv);
IF Local_ComType.Error THEN
    Output_ErrCode:=WORD#16#0020;
    Local_InitialSettingOK:=FALSE;
ELSE
    Local_InitialSettingOK:=TRUE;
END_IF;

(* 2.8. Send Data Conversion from String to Byte Array *)
Local_SrcDataByte:=
    StringToAry(ETN_SendMessageSet_instance.Send_Data,Local_SrcData[0]);

(* 2.9. Receive Data Storage Area Initialization *)
ClearString(Local_ReceiveMessage);
ClearString(Output_RecvMess);
Local_RecvCHNo:=0;
Local_RecvDataLength:=0;
Local_ReceiveSize:=UINT#256;

(* 2.10. Initialization End Processing *)
IF Local_InitialSettingOK THEN
    Local_State:=11;                        // Go to 11: Open Processing.
ELSE
    Local_Status.Busy:=FALSE;
    Local_Status.Error:=TRUE;

    Local_State:=0;                   // Go to 0: Communications Not Executed State.
END_IF;
```

3. Open Processing

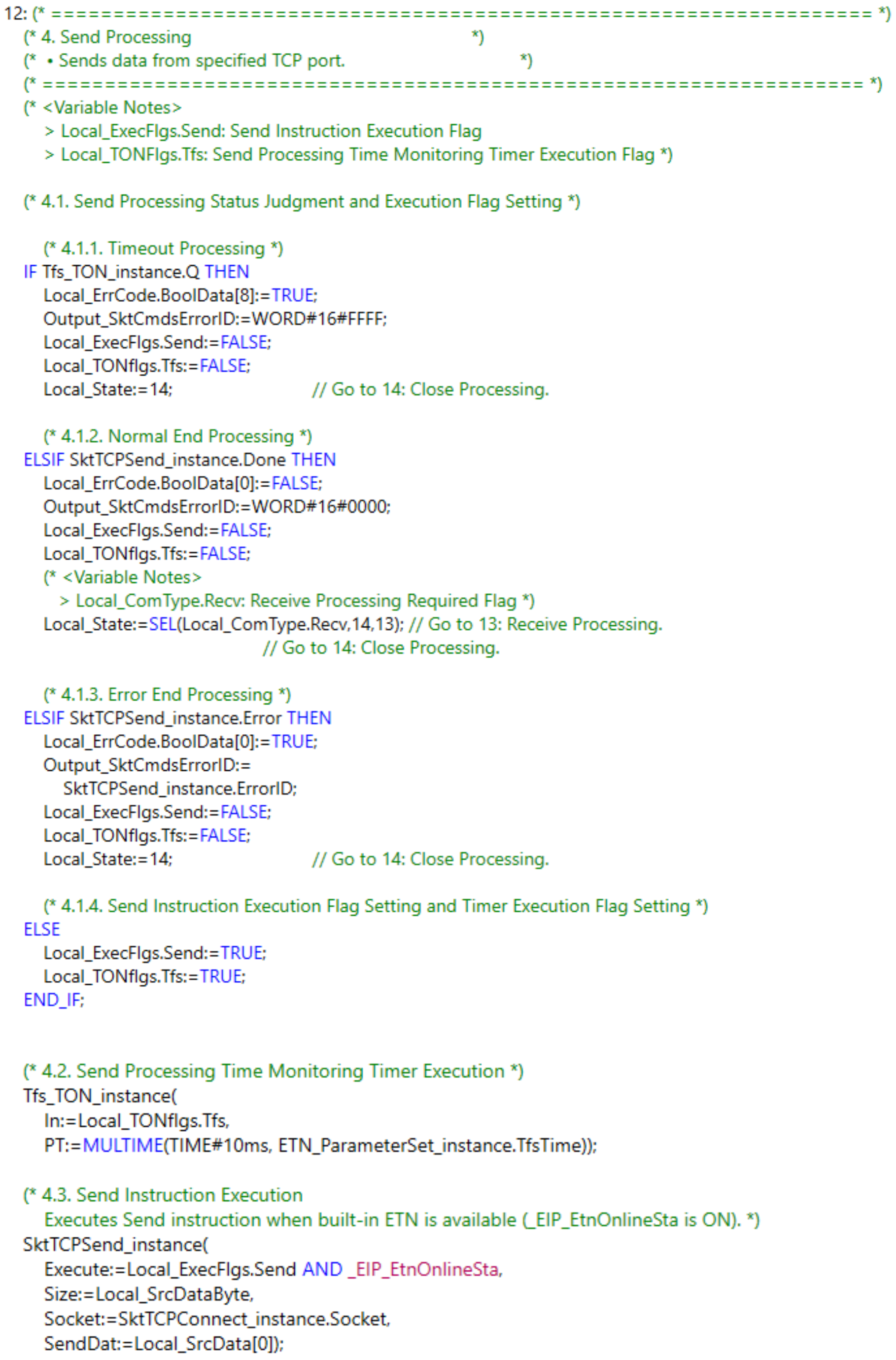


```
11: (* ===================================================================== *)
    (* 3. Open Processing                                  *)
    (*  • Connects to remote TCP port by active open.                       *)
    (* ===================================================================== *)
    (* <Variable Notes>
       > Local_ExecFlgs.Open: Open Instruction Execution Flag
       > Local_TONFlgs.Topen Open Processing Time Monitoring Timer Execution Flag *)

    (* 3.1. Open Processing Status Judgment and Execution Flag Setting *)

       (* 3.1.1. Timeout Processing *)
    IF Topen_TON_instance.Q THEN
       Local_ErrCode.BoolData[10]:=TRUE;
       Output_SktCmdsErrorID:=WORD#16#FFFF;
       Local_ExecFlgs.Open:=FALSE;
       Local_TONflgs.Topen:=FALSE;
       Local_State:=14;              // Go to 14: Close Processing.

       (* 3.1.2. Normal End Processing *)
    ELSIF SktTCPConnect_instance.Done THEN
       Local_ErrCode.BoolData[2]:= FALSE;
       Output_SktCmdsErrorID:=WORD#16#0000;
       Local_ExecFlgs.Open:=FALSE;
       Local_TONflgs.Topen:=FALSE;
    (* <Variable Notes>
        > Local_ComType.Send: Send Processing Required Flag
        > Local_ComType.Recv: Receive Processing Required Flag *)
       IF Local_ComType.Send THEN
          Local_State:=12;           // Go to 12: Send Processing.
       ELSIF Local_ComType.Recv THEN
          Local_State:=13;           // Go to 13: Receive Processing.
       END_IF;

       (* 3.1.3. Error End Processing *)
    ELSIF SktTCPConnect_instance.Error THEN
       Local_ErrCode.BoolData[2]:=TRUE;
       Output_SktCmdsErrorID:=SktTCPConnect_instance.ErrorID;
       Local_ExecFlgs.Open:=FALSE;
       Local_TONflgs.Topen:=FALSE;
       Local_State:=14;              // Go to 14: Close Processing.
```



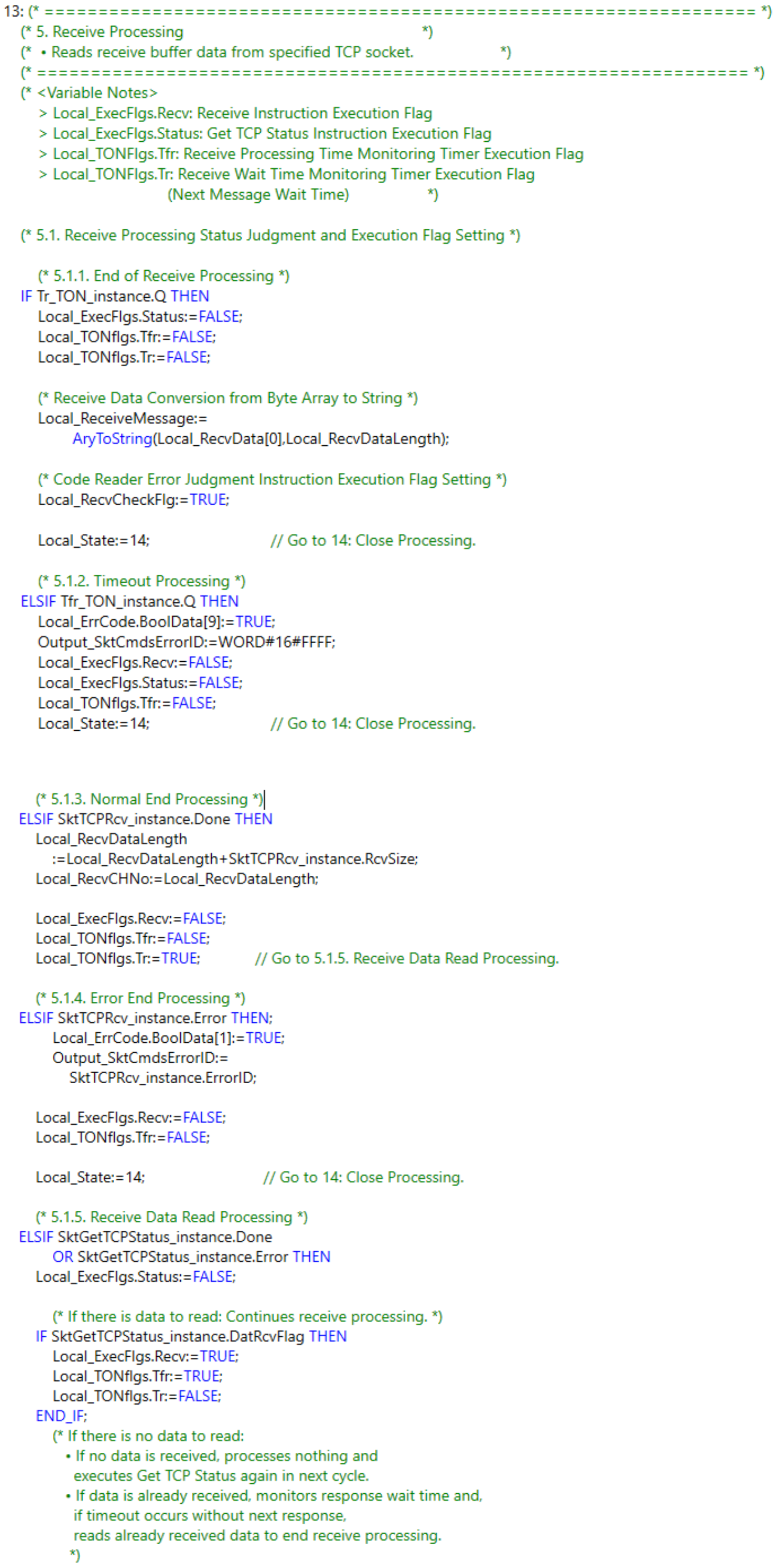
```
   (* 3.1.4. Open Instruction Execution Flag Setting and Timer Execution Flag Setting *)
ELSE
   Local_ExecFlgs.Open:=TRUE;
   Local_TONflgs.Topen:=TRUE;
END_IF;

(* 3.2. Open Processing Monitoring Timer Execution *)
Topen_TON_instance(
   In:=Local_TONflgs.Topen,
   PT:=MULTIME(TIME#10ms,ETN_ParameterSet_instance.TopenTime));

(* 3.3. Open Instruction Execution (TCP.Active Open Processing)
   Executes Open instruction when built-in ETN is available (_EIP_EtnOnlineSta is ON). *)
SktTCPConnect_instance(
   Execute:=Local_ExecFlgs.Open AND _EIP_EtnOnlineSta,
   SrcTcpPort:=ETN_ParameterSet_instance.SrcPort,
   DstTcpPort:=ETN_ParameterSet_instance.DstPort,
   DstAdr:=ETN_ParameterSet_instance.DstIPAddr);
```

4. Send Processing

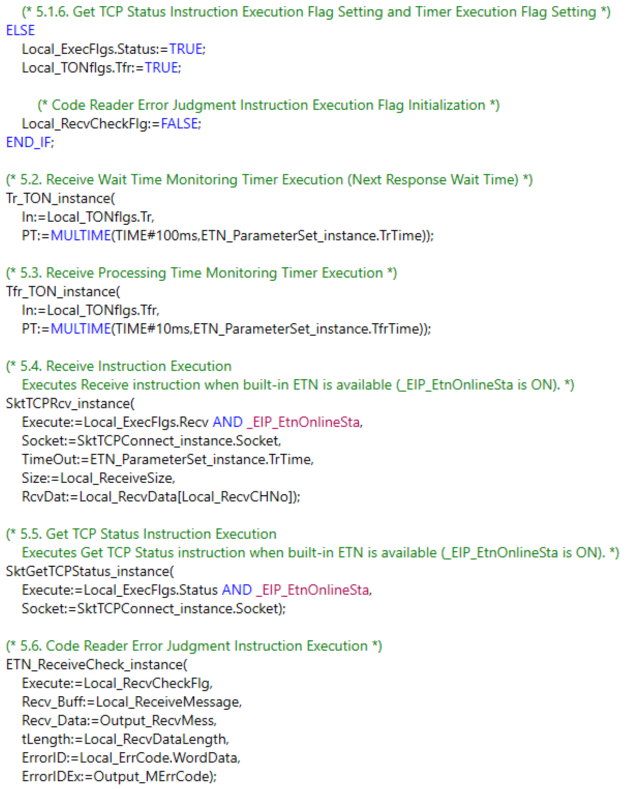


```
12: (* ================================================================== *)
  (* 4. Send Processing                                   *)
  (*  • Sends data from specified TCP port.                    *)
  (* ================================================================== *)
  (* <Variable Notes>
     > Local_ExecFlgs.Send: Send Instruction Execution Flag
     > Local_TONFlgs.Tfs: Send Processing Time Monitoring Timer Execution Flag *)

  (* 4.1. Send Processing Status Judgment and Execution Flag Setting *)

     (* 4.1.1. Timeout Processing *)
  IF Tfs_TON_instance.Q THEN
     Local_ErrCode.BoolData[8]:=TRUE;
     Output_SktCmdsErrorID:=WORD#16#FFFF;
     Local_ExecFlgs.Send:=FALSE;
     Local_TONflgs.Tfs:=FALSE;
     Local_State:=14;                    // Go to 14: Close Processing.

     (* 4.1.2. Normal End Processing *)
  ELSIF SktTCPSend_instance.Done THEN
     Local_ErrCode.BoolData[0]:=FALSE;
     Output_SktCmdsErrorID:=WORD#16#0000;
     Local_ExecFlgs.Send:=FALSE;
     Local_TONflgs.Tfs:=FALSE;
     (* <Variable Notes>
       > Local_ComType.Recv: Receive Processing Required Flag *)
     Local_State:=SEL(Local_ComType.Recv,14,13); // Go to 13: Receive Processing.
                                          // Go to 14: Close Processing.

     (* 4.1.3. Error End Processing *)
  ELSIF SktTCPSend_instance.Error THEN
     Local_ErrCode.BoolData[0]:=TRUE;
     Output_SktCmdsErrorID:=
        SktTCPSend_instance.ErrorID;
     Local_ExecFlgs.Send:=FALSE;
     Local_TONflgs.Tfs:=FALSE;
     Local_State:=14;                    // Go to 14: Close Processing.

     (* 4.1.4. Send Instruction Execution Flag Setting and Timer Execution Flag Setting *)
  ELSE
     Local_ExecFlgs.Send:=TRUE;
     Local_TONflgs.Tfs:=TRUE;
  END_IF;


  (* 4.2. Send Processing Time Monitoring Timer Execution *)
  Tfs_TON_instance(
     In:=Local_TONflgs.Tfs,
     PT:=MULTIME(TIME#10ms, ETN_ParameterSet_instance.TfsTime));

  (* 4.3. Send Instruction Execution
     Executes Send instruction when built-in ETN is available (_EIP_EtnOnlineSta is ON). *)
  SktTCPSend_instance(
     Execute:=Local_ExecFlgs.Send AND _EIP_EtnOnlineSta,
     Size:=Local_SrcDataByte,
     Socket:=SktTCPConnect_instance.Socket,
     SendDat:=Local_SrcData[0]);
```

5. Receive Processing

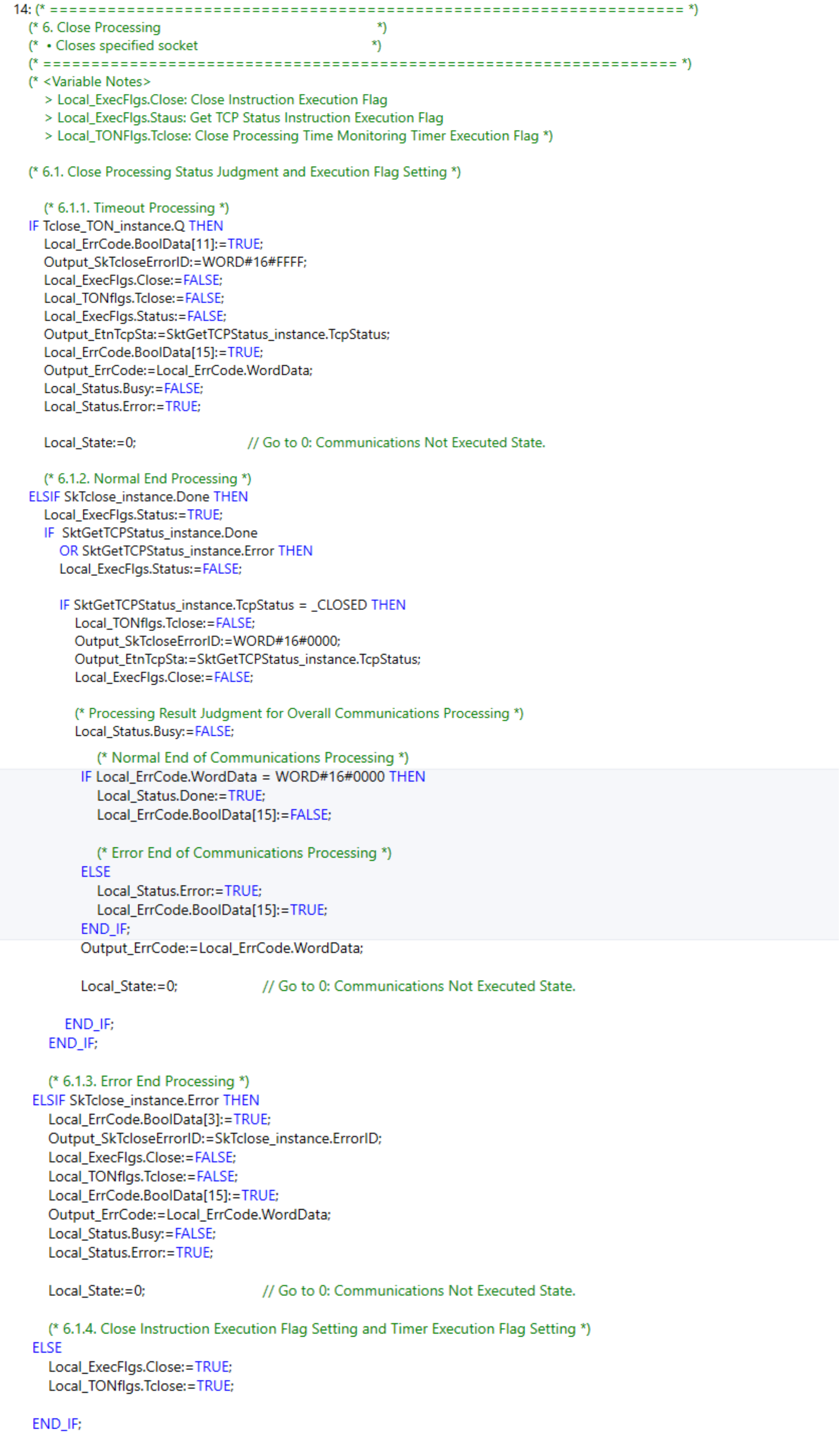


```
13: (* ================================================================ *)
  (* 5. Receive Processing                                         *)
  (*  • Reads receive buffer data from specified TCP socket.                 *)
  (* ================================================================ *)
  (* <Variable Notes>
     > Local_ExecFlgs.Recv: Receive Instruction Execution Flag
     > Local_ExecFlgs.Status: Get TCP Status Instruction Execution Flag
     > Local_TONFlgs.Tfr: Receive Processing Time Monitoring Timer Execution Flag
     > Local_TONFlgs.Tr: Receive Wait Time Monitoring Timer Execution Flag
                       (Next Message Wait Time)              *)

  (* 5.1. Receive Processing Status Judgment and Execution Flag Setting *)

     (* 5.1.1. End of Receive Processing *)
  IF Tr_TON_instance.Q THEN
     Local_ExecFlgs.Status:=FALSE;
     Local_TONflgs.Tfr:=FALSE;
     Local_TONflgs.Tr:=FALSE;

     (* Receive Data Conversion from Byte Array to String *)
     Local_ReceiveMessage:=
          AryToString(Local_RecvData[0],Local_RecvDataLength);

     (* Code Reader Error Judgment Instruction Execution Flag Setting *)
     Local_RecvCheckFlg:=TRUE;

     Local_State:=14;                    // Go to 14: Close Processing.

     (* 5.1.2. Timeout Processing *)
  ELSIF Tfr_TON_instance.Q THEN
     Local_ErrCode.BoolData[9]:=TRUE;
     Output_SktCmdsErrorID:=WORD#16#FFFF;
     Local_ExecFlgs.Recv:=FALSE;
     Local_ExecFlgs.Status:=FALSE;
     Local_TONflgs.Tfr:=FALSE;
     Local_State:=14;                    // Go to 14: Close Processing.


     (* 5.1.3. Normal End Processing *)
  ELSIF SktTCPRcv_instance.Done THEN
     Local_RecvDataLength
        :=Local_RecvDataLength+SktTCPRcv_instance.RcvSize;
     Local_RecvCHNo:=Local_RecvDataLength;

     Local_ExecFlgs.Recv:=FALSE;
     Local_TONflgs.Tfr:=FALSE;
     Local_TONflgs.Tr:=TRUE;          // Go to 5.1.5. Receive Data Read Processing.

     (* 5.1.4. Error End Processing *)
  ELSIF SktTCPRcv_instance.Error THEN;
        Local_ErrCode.BoolData[1]:=TRUE;
        Output_SktCmdsErrorID:=
           SktTCPRcv_instance.ErrorID;

     Local_ExecFlgs.Recv:=FALSE;
     Local_TONflgs.Tfr:=FALSE;

     Local_State:=14;                    // Go to 14: Close Processing.

     (* 5.1.5. Receive Data Read Processing *)
  ELSIF SktGetTCPStatus_instance.Done
        OR SktGetTCPStatus_instance.Error THEN
     Local_ExecFlgs.Status:=FALSE;

        (* If there is data to read: Continues receive processing. *)
     IF SktGetTCPStatus_instance.DatRcvFlag THEN
        Local_ExecFlgs.Recv:=TRUE;
        Local_TONflgs.Tfr:=TRUE;
        Local_TONflgs.Tr:=FALSE;
     END_IF;
        (* If there is no data to read:
          • If no data is received, processes nothing and
           executes Get TCP Status again in next cycle.
          • If data is already received, monitors response wait time and,
           if timeout occurs without next response,
           reads already received data to end receive processing.
          *)
```



```
    (* 5.1.6. Get TCP Status Instruction Execution Flag Setting and Timer Execution Flag Setting *)
ELSE
    Local_ExecFlgs.Status:=TRUE;
    Local_TONflgs.Tfr:=TRUE;

        (* Code Reader Error Judgment Instruction Execution Flag Initialization *)
    Local_RecvCheckFlg:=FALSE;
END_IF;

(* 5.2. Receive Wait Time Monitoring Timer Execution (Next Response Wait Time) *)
Tr_TON_instance(
    In:=Local_TONflgs.Tr,
    PT:=MULTIME(TIME#100ms,ETN_ParameterSet_instance.TrTime));

(* 5.3. Receive Processing Time Monitoring Timer Execution *)
Tfr_TON_instance(
    In:=Local_TONflgs.Tfr,
    PT:=MULTIME(TIME#10ms,ETN_ParameterSet_instance.TfrTime));

(* 5.4. Receive Instruction Execution
    Executes Receive instruction when built-in ETN is available (_EIP_EtnOnlineSta is ON). *)
SktTCPRcv_instance(
    Execute:=Local_ExecFlgs.Recv AND _EIP_EtnOnlineSta,
    Socket:=SktTCPConnect_instance.Socket,
    TimeOut:=ETN_ParameterSet_instance.TrTime,
    Size:=Local_ReceiveSize,
    RcvDat:=Local_RecvData[Local_RecvCHNo]);

(* 5.5. Get TCP Status Instruction Execution
    Executes Get TCP Status instruction when built-in ETN is available (_EIP_EtnOnlineSta is ON). *)
SktGetTCPStatus_instance(
    Execute:=Local_ExecFlgs.Status AND _EIP_EtnOnlineSta,
    Socket:=SktTCPConnect_instance.Socket);

(* 5.6. Code Reader Error Judgment Instruction Execution *)
ETN_ReceiveCheck_instance(
    Execute:=Local_RecvCheckFlg,
    Recv_Buff:=Local_ReceiveMessage,
    Recv_Data:=Output_RecvMess,
    tLength:=Local_RecvDataLength,
    ErrorID:=Local_ErrCode.WordData,
    ErrorIDEx:=Output_MErrCode);
```

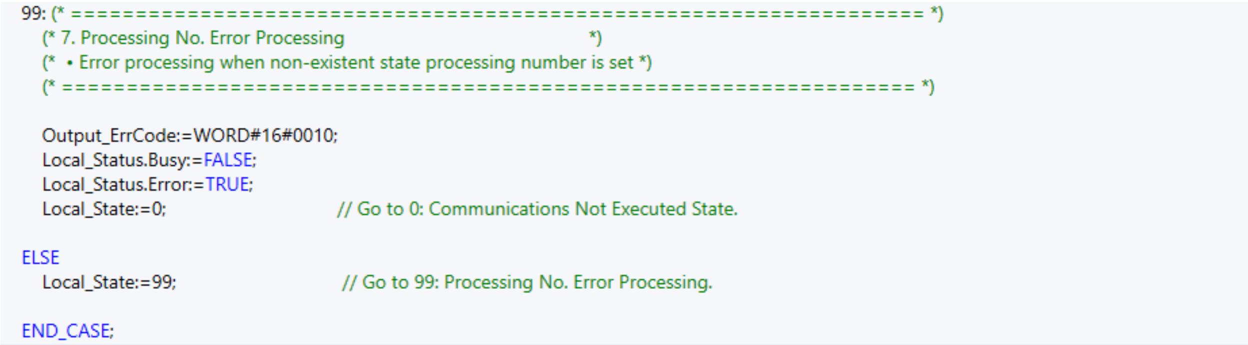
6. Close Processing



```
14: (* ====================================================================== *)
(* 6. Close Processing                                              *)
(*  • Closes specified socket                                       *)
(* ====================================================================== *)
(* <Variable Notes>
    > Local_ExecFlgs.Close: Close Instruction Execution Flag
    > Local_ExecFlgs.Staus: Get TCP Status Instruction Execution Flag
    > Local_TONFlgs.Tclose: Close Processing Time Monitoring Timer Execution Flag *)

(* 6.1. Close Processing Status Judgment and Execution Flag Setting *)

    (* 6.1.1. Timeout Processing *)
IF Tclose_TON_instance.Q THEN
    Local_ErrCode.BoolData[11]:=TRUE;
    Output_SkTcloseErrorID:=WORD#16#FFFF;
    Local_ExecFlgs.Close:=FALSE;
    Local_TONflgs.Tclose:=FALSE;
    Local_ExecFlgs.Status:=FALSE;
    Output_EtnTcpSta:=SktGetTCPStatus_instance.TcpStatus;
    Local_ErrCode.BoolData[15]:=TRUE;
    Output_ErrCode:=Local_ErrCode.WordData;
    Local_Status.Busy:=FALSE;
    Local_Status.Error:=TRUE;

    Local_State:=0;                    // Go to 0: Communications Not Executed State.

    (* 6.1.2. Normal End Processing *)
ELSIF SkTclose_instance.Done THEN
    Local_ExecFlgs.Status:=TRUE;
    IF  SktGetTCPStatus_instance.Done
        OR SktGetTCPStatus_instance.Error THEN
        Local_ExecFlgs.Status:=FALSE;

        IF SktGetTCPStatus_instance.TcpStatus = _CLOSED THEN
            Local_TONflgs.Tclose:=FALSE;
            Output_SkTcloseErrorID:=WORD#16#0000;
            Output_EtnTcpSta:=SktGetTCPStatus_instance.TcpStatus;
            Local_ExecFlgs.Close:=FALSE;

            (* Processing Result Judgment for Overall Communications Processing *)
            Local_Status.Busy:=FALSE;

                (* Normal End of Communications Processing *)
            IF Local_ErrCode.WordData = WORD#16#0000 THEN
                Local_Status.Done:=TRUE;
                Local_ErrCode.BoolData[15]:=FALSE;

                (* Error End of Communications Processing *)
            ELSE
                Local_Status.Error:=TRUE;
                Local_ErrCode.BoolData[15]:=TRUE;
            END_IF;
            Output_ErrCode:=Local_ErrCode.WordData;

            Local_State:=0;                // Go to 0: Communications Not Executed State.

        END_IF;
    END_IF;

    (* 6.1.3. Error End Processing *)
ELSIF SkTclose_instance.Error THEN
    Local_ErrCode.BoolData[3]:=TRUE;
    Output_SkTcloseErrorID:=SkTclose_instance.ErrorID;
    Local_ExecFlgs.Close:=FALSE;
    Local_TONflgs.Tclose:=FALSE;
    Local_ErrCode.BoolData[15]:=TRUE;
    Output_ErrCode:=Local_ErrCode.WordData;
    Local_Status.Busy:=FALSE;
    Local_Status.Error:=TRUE;

    Local_State:=0;                    // Go to 0: Communications Not Executed State.

    (* 6.1.4. Close Instruction Execution Flag Setting and Timer Execution Flag Setting *)
ELSE
    Local_ExecFlgs.Close:=TRUE;
    Local_TONflgs.Tclose:=TRUE;

END_IF;
```

```
(* 6.2. Close Processing Time Monitoring Timer Execution *)
Tclose_TON_instance(
   In:= Local_TONflgs.Tclose,
   PT:=MULTIME(TIME#10ms,ETN_ParameterSet_instance.TcloseTime));

(* 6.3. Close Instruction Execution
   Executes Close instruction when built-in ETN is available (_EIP_EtnOnlineSta is ON). *)
SkTclose_instance(
   Execute:=Local_ExecFlgs.Close AND _EIP_EtnOnlineSta,
   Socket:=SktTCPConnect_instance.Socket);

(* 6.4. Get TCP Status Instruction Execution
   Executes Get TCP Status instruction when built-in ETN is available (_EIP_EtnOnlineSta is ON). *)
SktGetTCPStatus_instance(
   Execute:=Local_ExecFlgs.Status AND _EIP_EtnOnlineSta,
   Socket:=SktTCPConnect_instance.Socket);
```

7. Processing No. Error Processing



```
99: (* ================================================================= *)
   (* 7. Processing No. Error Processing                                *)
   (*  • Error processing when non-existent state processing number is set *)
   (* ================================================================= *)

   Output_ErrCode:=WORD#16#0010;
   Local_Status.Busy:=FALSE;
   Local_Status.Error:=TRUE;
   Local_State:=0;                    // Go to 0: Communications Not Executed State.

ELSE
   Local_State:=99;                   // Go to 99: Processing No. Error Processing.

END_CASE;
```

```
END_IF;
```

9.5.3. Detailed Explanation of Function Blocks

This project file uses the following function blocks.

In the printout of function blocks given below, data that is variable depending on the code reader is shown in red frames.

● Details of the ETN_ParameterSet_instance Function Block (ParameterSet)

Instruction	Name	FB/FUN	Graphic expression	ST expression
ParameterSet	Ethernet Communications Parameter Settings	FB	None	ETN_ParameterSet_instance (Execute, TfsTime, TrTime, TfrTime, , TopenTime, TcloseTime, SrcPort, DstIPAddr, DstPort);

• In-out Variable Table

• Input

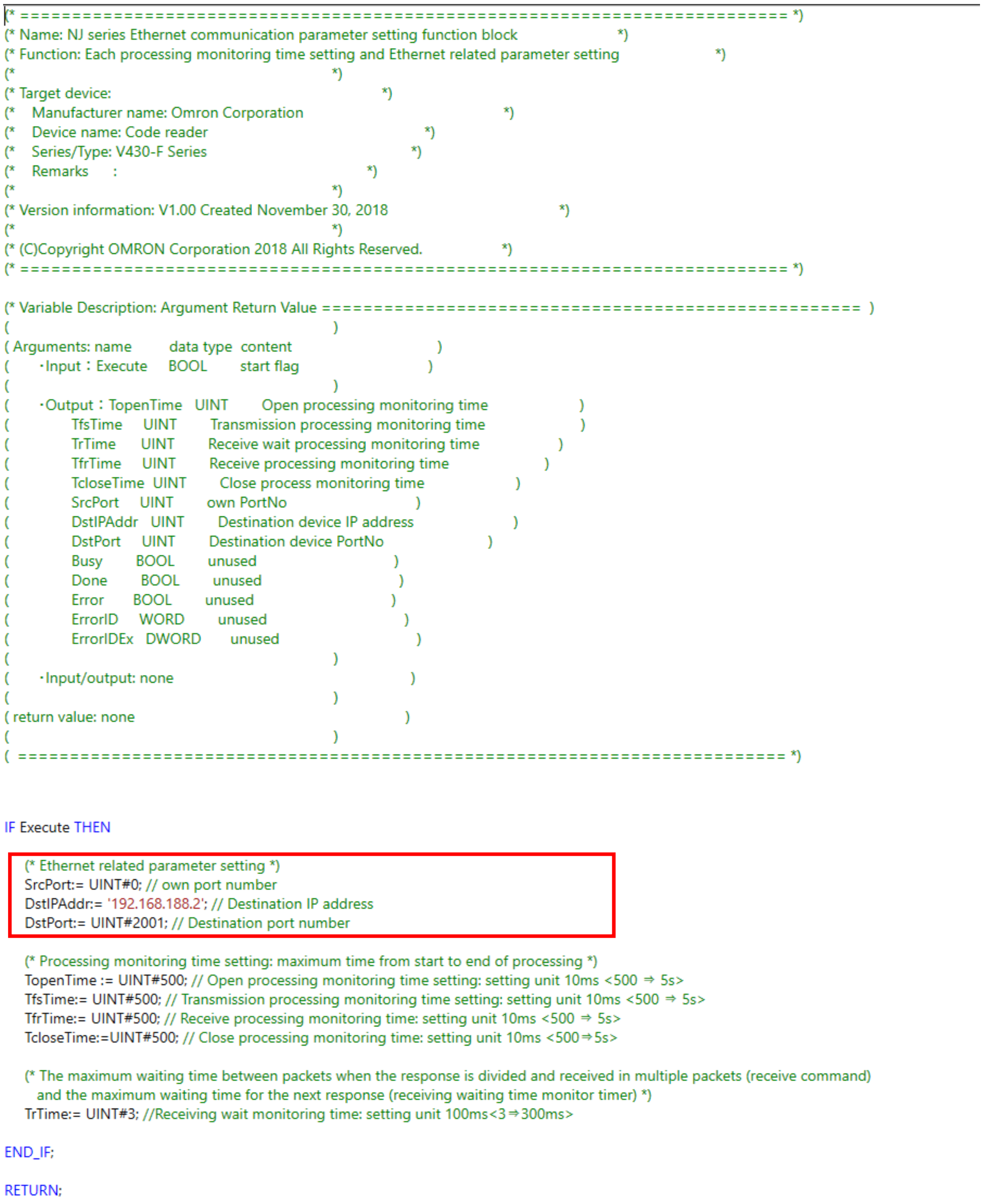
Variable name	Data type	Name	Description	Valid range	Unit	Initial value
Execute	BOOL	Execute	Executes the function block when the value changes from OFF (*FALSE*) to ON (*TRUE*). (Always *TRUE*)	Depends on data type.	---	---

• Output

Variable name	Data type	Name	Description	Valid range	Unit	Initial value
TopenTime	UINT	Open Monitoring Time	Sets the monitoring time for open processing in increments of 10 ms.	Depends on data type.	---	---
TfsTime	UINT	Send Monitoring Time	Sets the monitoring time for send processing in increments of 10 ms.	Depends on data type.	---	---
TrTime	UINT	Receive Wait Monitoring Time	Sets the arrival standby time for receive data in increments of 100 ms.	Depends on data type.	---	---
TfrTime	UINT	Receive Processing Time	Sets the monitoring time for receive processing in increments of 10 ms.	Depends on data type.	---	---
TcloseTime	UINT	Close Monitoring Time	Sets the monitoring time for close processing in increments of 10 ms.	Depends on data type.	---	---
SrcPort	UINT	Source Port No.	Sets the local port.	Depends on data type.	---	---
DstIPAddr	STRING [256]	Destination IP Address	Sets the remote IP address.	Depends on code reader.	---	---
DstPort	UINT	Destination Port No.	Sets the remote port number.	Depends on code reader.	---	---
Busy	BOOL	Busy	Not used (Not used in this project.)	---	---	---
Done	BOOL	Normal End				
Error	BOOL	Error End				
ErrorID	WORD	Error Information				
ErrorIDEx	DWORD	Error Information				

• Internal Variable Table: None

- Program



```
(* ========================================================================= *)
(* Name: NJ series Ethernet communication parameter setting function block          *)
(* Function: Each processing monitoring time setting and Ethernet related parameter setting          *)
(*                                                   *)
(* Target device:                                        *)
(*    Manufacturer name: Omron Corporation                           *)
(*    Device name: Code reader                                  *)
(*    Series/Type: V430-F Series                               *)
(*    Remarks     :                                      *)
(*                                                   *)
(* Version information: V1.00 Created November 30, 2018                         *)
(*                                                   *)
(* (C)Copyright OMRON Corporation 2018 All Rights Reserved.                 *)
(* ========================================================================= *)

(* Variable Description: Argument Return Value =================================================== )
(                                                    )
( Arguments: name       data type  content                       )
(      ･Input：Execute    BOOL       start flag                    )
(                                                    )
(      ･Output：TopenTime  UINT       Open processing monitoring time              )
(            TfsTime    UINT       Transmission processing monitoring time         )
(            TrTime     UINT       Receive wait processing monitoring time         )
(            TfrTime    UINT       Receive processing monitoring time            )
(            TcloseTime  UINT       Close process monitoring time              )
(            SrcPort    UINT       own PortNo                     )
(            DstIPAddr   UINT       Destination device IP address           )
(            DstPort    UINT       Destination device PortNo            )
(            Busy      BOOL       unused                    )
(            Done      BOOL       unused                    )
(            Error     BOOL       unused                    )
(            ErrorID    WORD       unused                    )
(            ErrorIDEx   DWORD      unused                    )
(                                                    )
(      ･Input/output: none                            )
(                                                    )
( return value: none                                 )
(                                                    )
(  ======================================================================== *)


IF Execute THEN

  (* Ethernet related parameter setting *)
  SrcPort:= UINT#0; // own port number
  DstIPAddr:= '192.168.188.2'; // Destination IP address
  DstPort:= UINT#2001; // Destination port number

  (* Processing monitoring time setting: maximum time from start to end of processing *)
  TopenTime := UINT#500; // Open processing monitoring time setting: setting unit 10ms <500 ⇒ 5s>
  TfsTime:= UINT#500; // Transmission processing monitoring time setting: setting unit 10ms <500 ⇒ 5s>
  TfrTime:= UINT#500; // Receive processing monitoring time: setting unit 10ms <500 ⇒ 5s>
  TcloseTime:=UINT#500; // Close processing monitoring time: setting unit 10ms <500⇒5s>

  (* The maximum waiting time between packets when the response is divided and received in multiple packets (receive command)
    and the maximum waiting time for the next response (receiving waiting time monitor timer) *)
  TrTime:= UINT#3; //Receiving wait monitoring time: setting unit 100ms<3⇒300ms>

END_IF;

RETURN;
```

● Details of the ETN_SendMessageSet_instance Function Block (SendMessageSet)

Instruction	Name	FB/FUN	Graphic expression	ST expression
SendMessageSet	Ethernet Communications Sequence Setting	FB	None	ETN_SendMessageSet_ instance (Execute, Send_Data, ComType);

- In-out Variable Table
- Input

Variable name	Data type	Name	Description	Valid range	Unit	Initial value
Execute	BOOL	Execute	Executes the function block when the value changes from OFF (*FALSE*) to ON (*TRUE*). (Always *TRUE*)	Depends on data type.	---	---

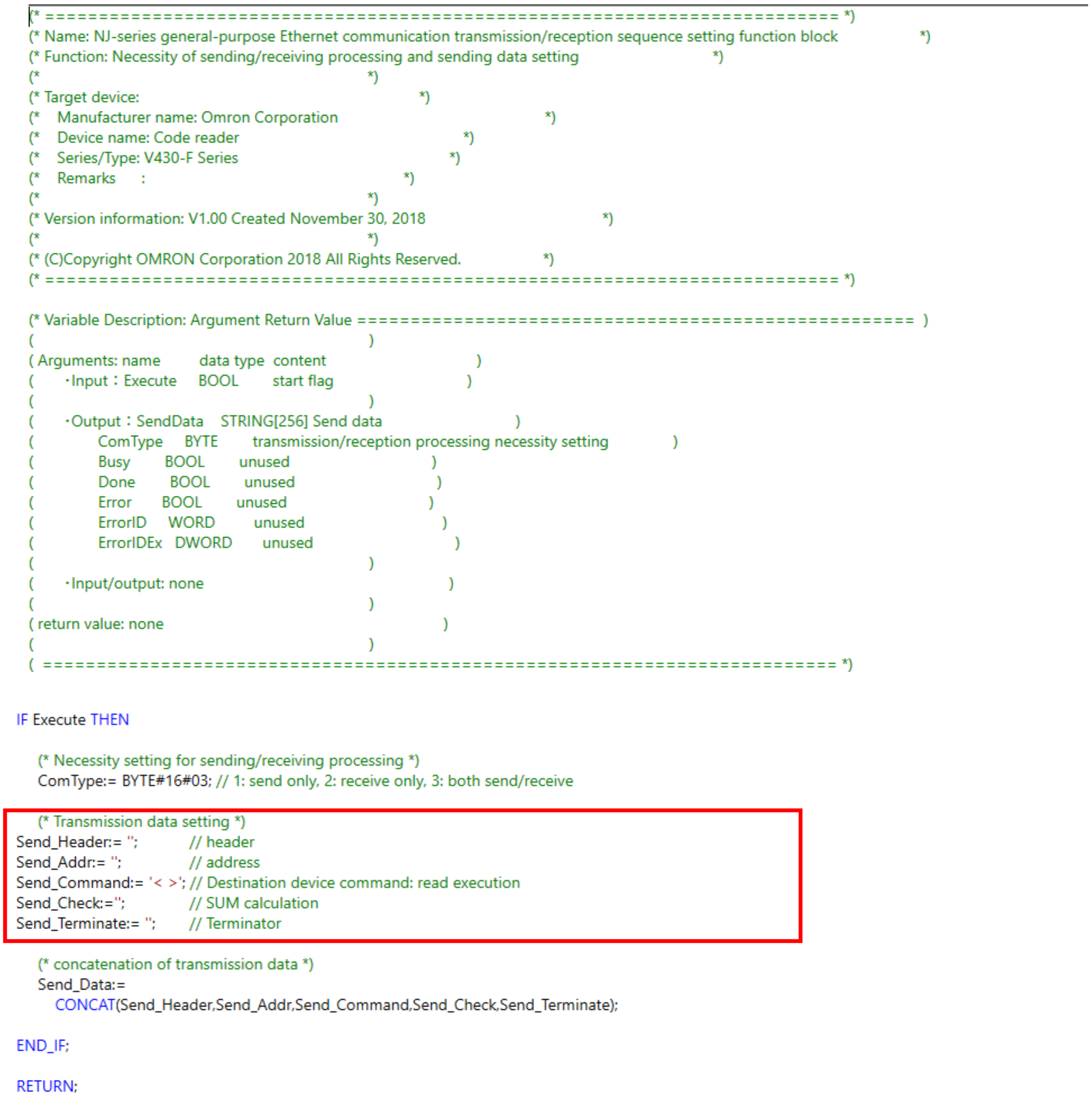
- Output

Variable name	Data type	Name	Description	Valid range	Unit	Initial value
Send_Data	STRING [256]	Send Data	Sets the send command to the code reader.	Depends on data type.	---	---
ComType	BYTE	Communication Type	Sets whether to execute send, receive, or send and receive processing. 1: Send only, 2: Receive only, 3: Send and receive	1 to 3	---	---
Busy	BOOL	Busy	Not used (Not used in this project.)	---	---	---
Done	BOOL	Normal End				
Error	BOOL	Error End				
ErrorID	WORD	Error Information				
ErrorIDEx	DWORD	Error Information				

- Internal Variable Table

Variable name	Data type	Name	Description	Valid range	Unit	Initial value
Send_ Header	STRING[5]	Send Header	Send message header	Depends on data type.	---	---
Send_ Addr	STRING[5]	Code Reader Address	Code reader address	Depends on data type.	---	---
Send_ Command	STRING[256]	Send Data	Send command to the code reader	Depends on data type.	---	---
Send_ Check	STRING[5]	Send Check Code	Send message check code	Depends on data type.	---	---
Send_Ter minate	STRING[5]	Send Terminator	Send message terminator	Depends on data type.	---	---

- Program



```
(* ======================================================================== *)
(* Name: NJ-series general-purpose Ethernet communication transmission/reception sequence setting function block        *)
(* Function: Necessity of sending/receiving processing and sending data setting                 *)
(*                                                     *)
(* Target device:                                          *)
(*    Manufacturer name: Omron Corporation                          *)
(*    Device name: Code reader                               *)
(*    Series/Type: V430-F Series                              *)
(*    Remarks     :                                       *)
(*                                                     *)
(* Version information: V1.00 Created November 30, 2018                     *)
(*                                                     *)
(* (C)Copyright OMRON Corporation 2018 All Rights Reserved.                *)
(* ======================================================================== *)

(* Variable Description: Argument Return Value ============================================= )
(                                                      )
( Arguments: name        data type  content                          )
(      ･Input : Execute    BOOL       start flag                      )
(                                                      )
(      ･Output : SendData    STRING[256] Send data                        )
(            ComType    BYTE       transmission/reception processing necessity setting          )
(            Busy       BOOL       unused                       )
(            Done       BOOL       unused                       )
(            Error      BOOL       unused                       )
(            ErrorID    WORD       unused                       )
(            ErrorIDEx  DWORD      unused                       )
(                                                      )
(      ･Input/output: none                                     )
(                                                      )
( return value: none                                         )
(                                                      )
( ======================================================================== *)


IF Execute THEN

   (* Necessity setting for sending/receiving processing *)
   ComType:= BYTE#16#03; // 1: send only, 2: receive only, 3: both send/receive

   (* Transmission data setting *)
Send_Header:= '';           // header
Send_Addr:= '';             // address
Send_Command:= '< >'; // Destination device command: read execution
Send_Check:='';             // SUM calculation
Send_Terminate:= '';        // Terminator

   (* concatenation of transmission data *)
   Send_Data:=
      CONCAT(Send_Header,Send_Addr,Send_Command,Send_Check,Send_Terminate);

END_IF;

RETURN;
```

- Details of the ETN_ReceiveCheck_instance Function Block (ReceiveCheck)

Instruction	Name	FB/FUN	Graphic expression	ST expression
ReceiveCheck	Ethernet Communications Receive Processing	FB	None	ETN_ReceiveCheck_instance (Execute, Recv_Data, Recv_Buff, Error, ErrorID, ErrorIDEx);

- In-out Variable Table
- Input

Variable name	Data type	Name	Description	Valid range	Unit	Initial value
Execute	BOOL	Execute	Executes the function block when the value changes from OFF (*FALSE*) to ON (*TRUE*).	Depends on data type.	---	---
tLength	UINT	Receive Data Length	Byte length of receive buffer data	Depends on data type.	---	---

- In-out

Variable name	Data type	Name	Description	Valid range	Unit	Initial value
Recv_Data	STRING[256]	Receive Data	Receive data storage result	Depends on data type.	---	---
Recv_Buff	STRING[256]	Receive Buffer	Receive data buffer	Depends on data type.	---	---
ErrorID	WORD	Error Information	Error code: Code reader error = #16#1000 FCS error = #16#2000	---	---	---
ErrorIDEx	DWORD	Error Information	Error: code: FCS receive result/Code reader error code	---	---	---

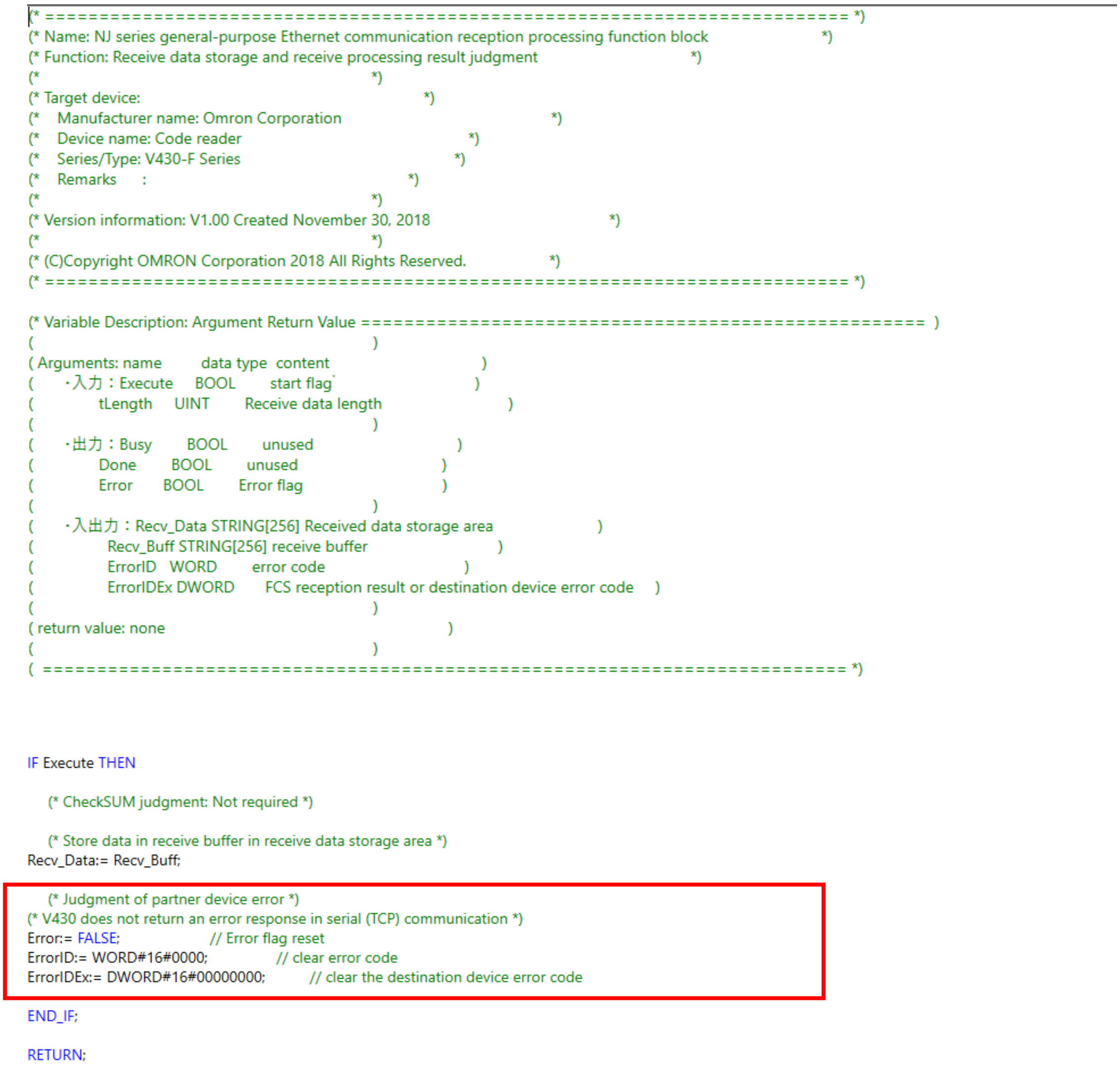
- Output

Variable name	Data type	Name	Description	Valid range	Unit	Initial value
Busy	BOOL	Busy	Not used (Not used in this project.)	---	---	---
Done	BOOL	Normal End				
Error	BOOL	Error End	Error end	---	---	---

- Internal Variable Table

Variable name	Data type	Name	Description	Valid range	Unit	Initial value
Receive_Check	STRING[5]	Receive FCS	FCS receive result of receive data	Depends on data type.	---	---
Calc_Check	STRING[5]	Receive FCS Calculation Value	FCS calculation result of receive data	Depends on data type.	---	---

- Program

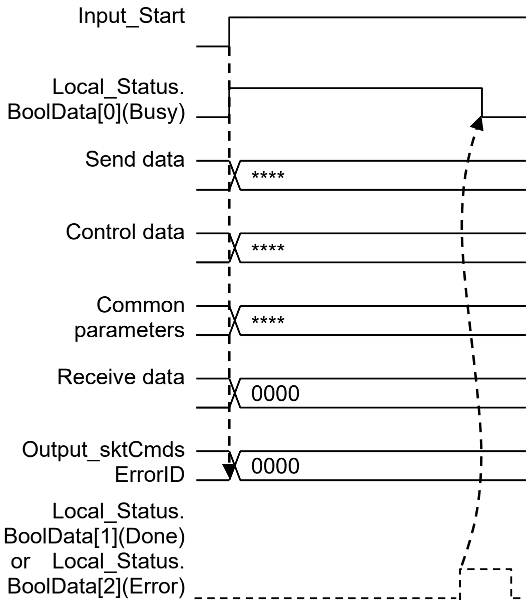


```
(* =========================================================================== *)
(* Name: NJ series general-purpose Ethernet communication reception processing function block          *)
(* Function: Receive data storage and receive processing result judgment                    *)
(*                                              *)
(* Target device:                                     *)
(*   Manufacturer name: Omron Corporation                          *)
(*   Device name: Code reader                                *)
(*   Series/Type: V430-F Series                             *)
(*   Remarks     :                                  *)
(*                                              *)
(* Version information: V1.00 Created November 30, 2018                        *)
(*                                              *)
(* (C)Copyright OMRON Corporation 2018 All Rights Reserved.                *)
(* =========================================================================== *)

(* Variable Description: Argument Return Value =================================================== )
(                                               )
( Arguments: name        data type  content                    )
(      ・入力：Execute     BOOL       start flag                   )
(           tLength     UINT       Receive data length                  )
(                                               )
(      ・出力：Busy        BOOL       unused                    )
(           Done        BOOL       unused                   )
(           Error       BOOL       Error flag                   )
(                                               )
(      ・入出力：Recv_Data STRING[256] Received data storage area                 )
(             Recv_Buff STRING[256] receive buffer                        )
(             ErrorID   WORD       error code                      )
(             ErrorIDEx DWORD      FCS reception result or destination device error code    )
(                                               )
( return value: none                                    )
(                                               )
(  =========================================================================== *)



IF Execute THEN

    (* CheckSUM judgment: Not required *)

    (* Store data in receive buffer in receive data storage area *)
Recv_Data:= Recv_Buff;

    (* Judgment of partner device error *)
(* V430 does not return an error response in serial (TCP) communication *)
Error:= FALSE;                // Error flag reset
ErrorID:= WORD#16#0000;             // clear error code
ErrorIDEx:= DWORD#16#00000000;          // clear the destination device error code

END_IF;

RETURN;
```

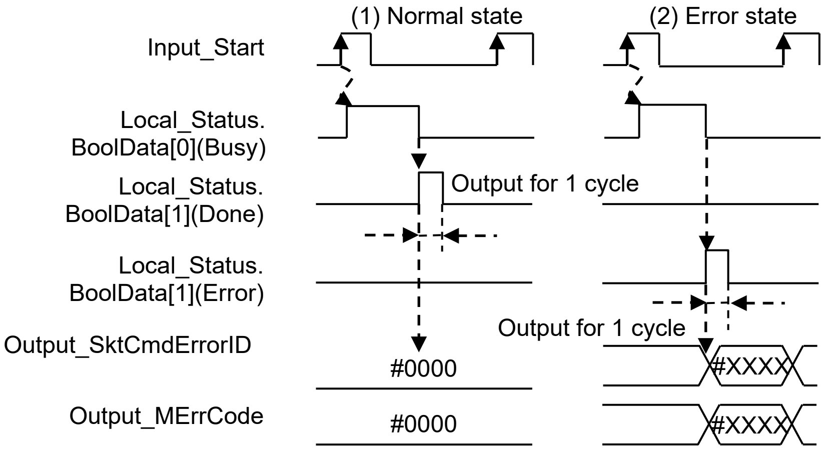
9.6. Timing Chart

The timing chart for the ST language program is shown below.

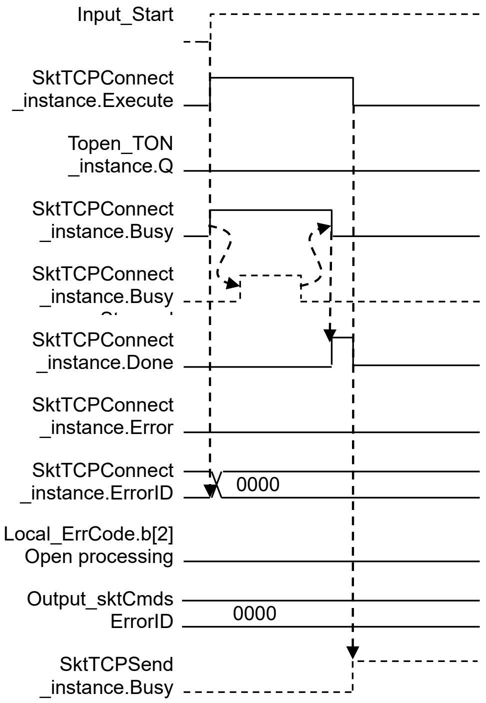
Start and Setup



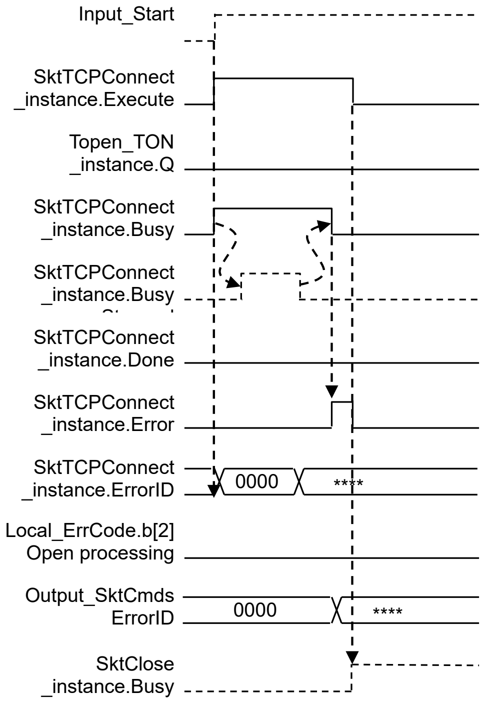
If *Input_Start* is changed from *True* (ON) to *False* (OFF) during execution, Normal End or Error End is output for one cycle after processing is completed as shown below.



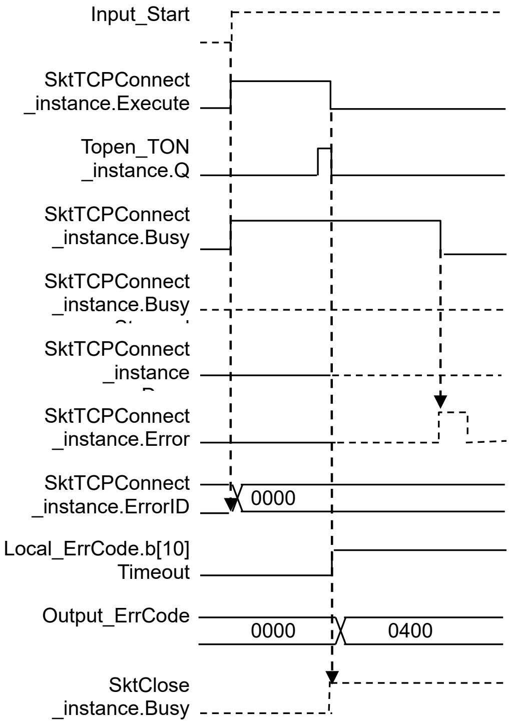
● Open Processing


Input_Start
SktTCPConnect _instance.Execute
Topen_TON _instance.Q
SktTCPConnect _instance.Busy
SktTCPConnect _instance.Busy
SktTCPConnect _instance.Done
SktTCPConnect _instance.Error
SktTCPConnect _instance.ErrorID
0000
Local_ErrCode.b[2] Open processing
Output_sktCmds ErrorID
0000
SktTCPSend _instance.Busy

(Normal End)

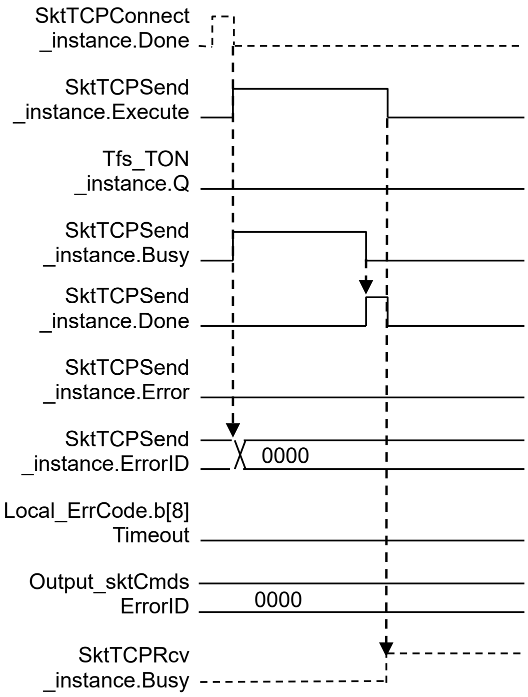

Input_Start
SktTCPConnect _instance.Execute
Topen_TON _instance.Q
SktTCPConnect _instance.Busy
SktTCPConnect _instance.Busy
SktTCPConnect _instance.Done
SktTCPConnect _instance.Error
SktTCPConnect _instance.ErrorID
0000
****
Local_ErrCode.b[2] Open processing
Output_SktCmds ErrorID
0000
****
SktClose _instance.Busy

(Error End)

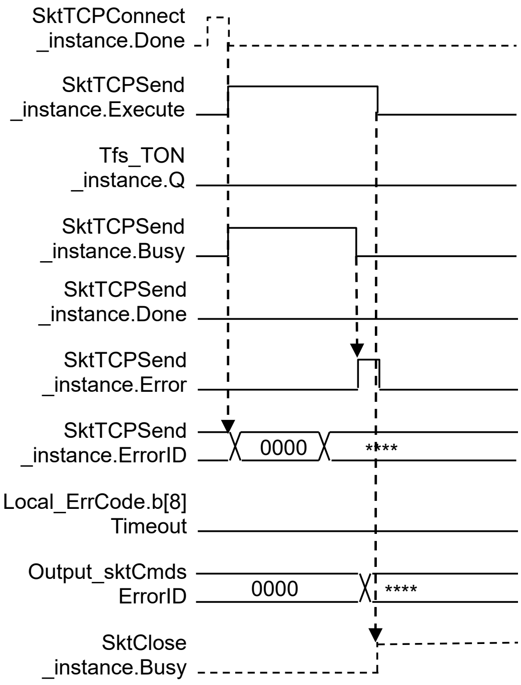

Input_Start
SktTCPConnect _instance.Execute
Topen_TON _instance.Q
SktTCPConnect _instance.Busy
SktTCPConnect _instance.Busy
SktTCPConnect _instance
SktTCPConnect _instance.Error
SktTCPConnect _instance.ErrorID
0000
Local_ErrCode.b[10] Timeout
Output_ErrCode
0000
0400
SktClose _instance.Busy

(Timeout)

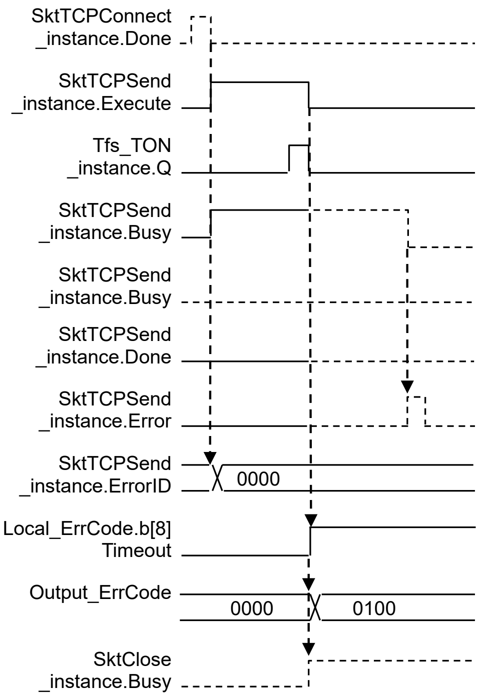
● Send Processing



(Normal End)

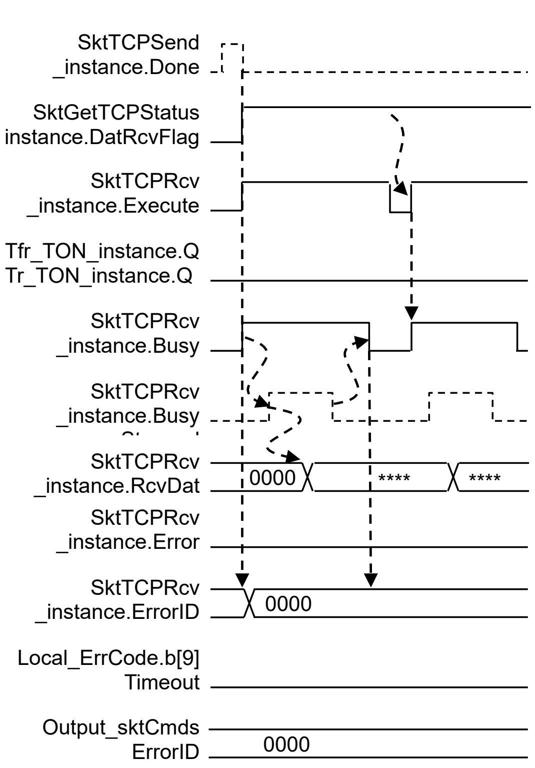


(Error End)

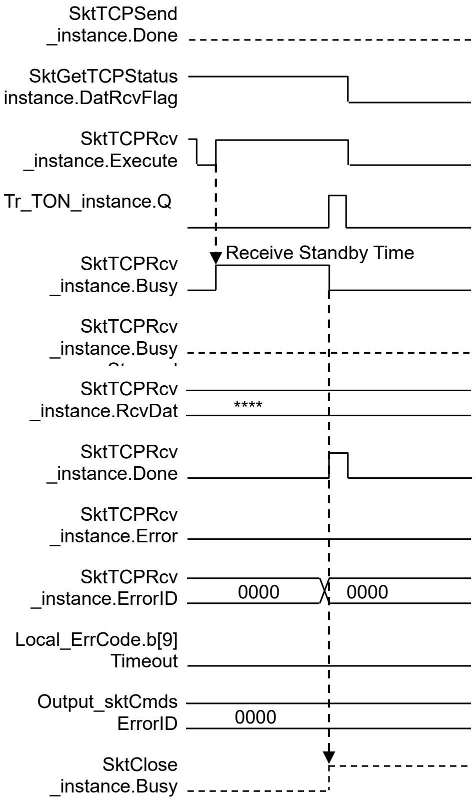


(Timeout)

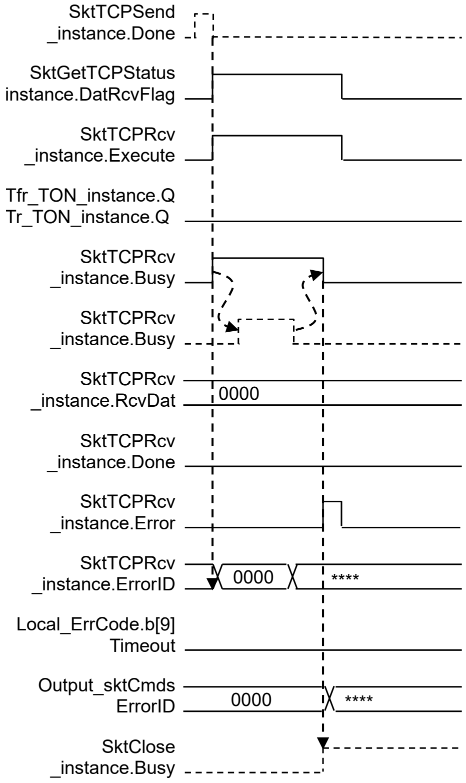
- Receive Processing



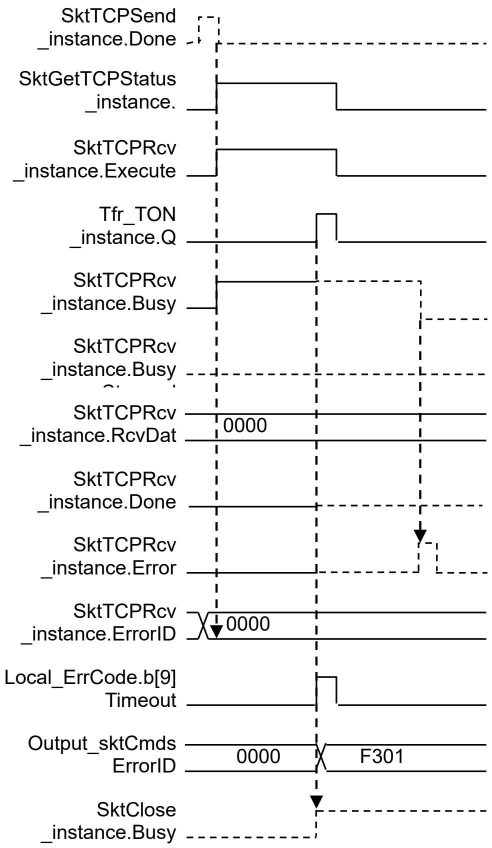
(Repetition)



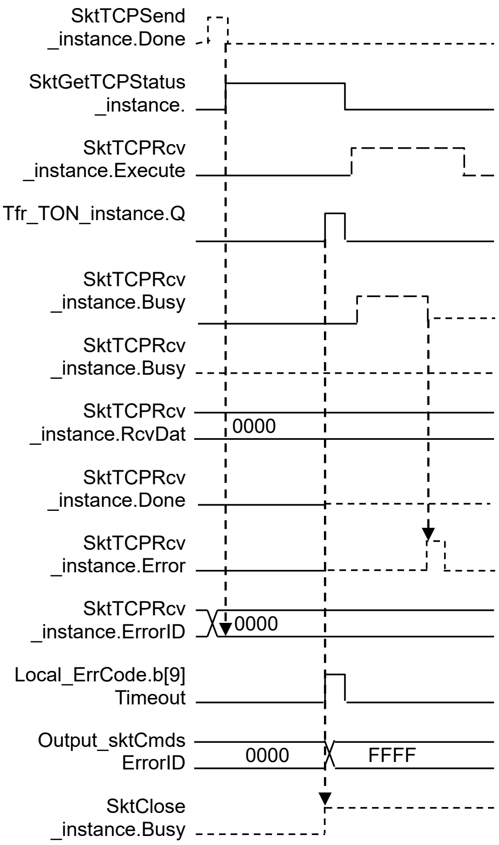
(Normal End)



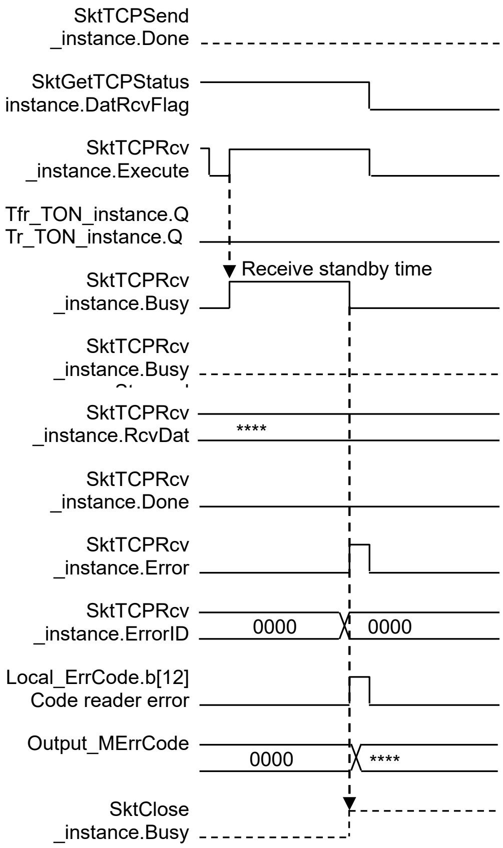
(Error End)


SktTCPSend _instance.Done
SktGetTCPStatus _instance.
SktTCPRcv _instance.Execute
Tfr_TON _instance.Q
SktTCPRcv _instance.Busy
SktTCPRcv _instance.Busy
SktTCPRcv _instance.RcvDat
0000
SktTCPRcv _instance.Done
SktTCPRcv _instance.Error
SktTCPRcv _instance.ErrorID
0000
Local_ErrCode.b[9] Timeout
Output_sktCmds ErrorID
0000
F301
SktClose _instance.Busy

(Timeout: Receive error)

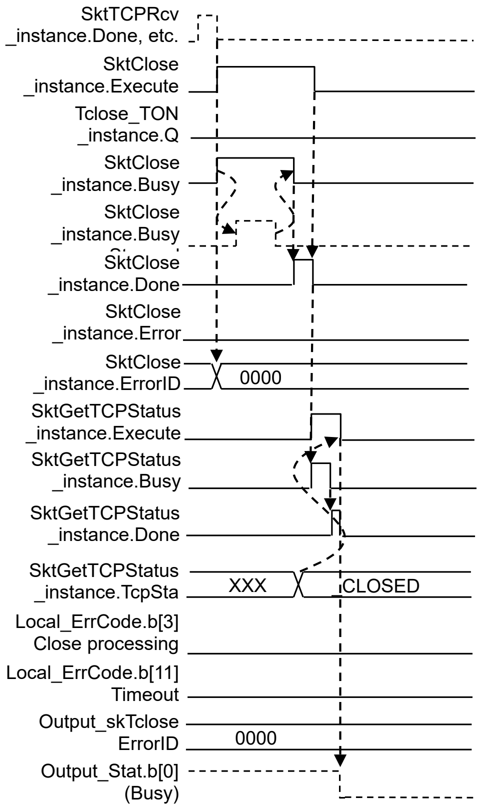

SktTCPSend _instance.Done
SktGetTCPStatus _instance.
SktTCPRcv _instance.Execute
Tfr_TON_instance.Q
SktTCPRcv _instance.Busy
SktTCPRcv _instance.Busy
SktTCPRcv _instance.RcvDat
0000
SktTCPRcv _instance.Done
SktTCPRcv _instance.Error
SktTCPRcv _instance.ErrorID
0000
Local_ErrCode.b[9] Timeout
Output_sktCmds ErrorID
0000
FFFF
SktClose _instance.Busy

(Timeout: No receive data)

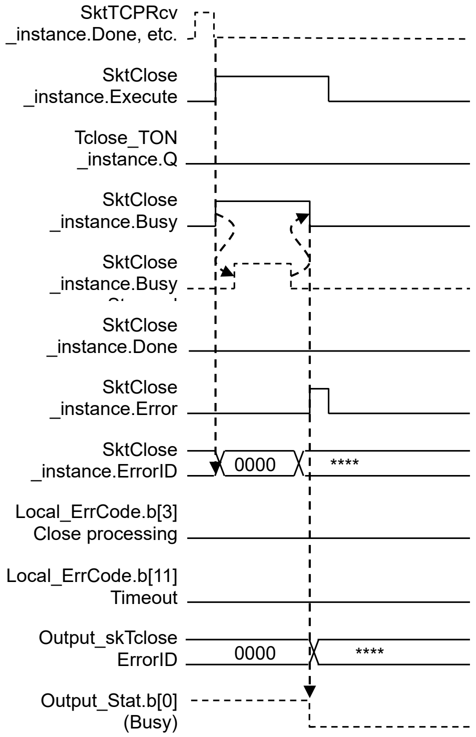

SktTCPSend _instance.Done
SktGetTCPStatus instance.DatRcvFlag
SktTCPRcv _instance.Execute
Tfr_TON_instance.Q Tr_TON_instance.Q
Receive standby time
SktTCPRcv _instance.Busy
SktTCPRcv _instance.Busy
SktTCPRcv _instance.RcvDat
****
SktTCPRcv _instance.Done
SktTCPRcv _instance.Error
SktTCPRcv _instance.ErrorID
0000
0000
Local_ErrCode.b[12] Code reader error
Output_MErrCode
0000
****
SktClose _instance.Busy

(Code reader error)

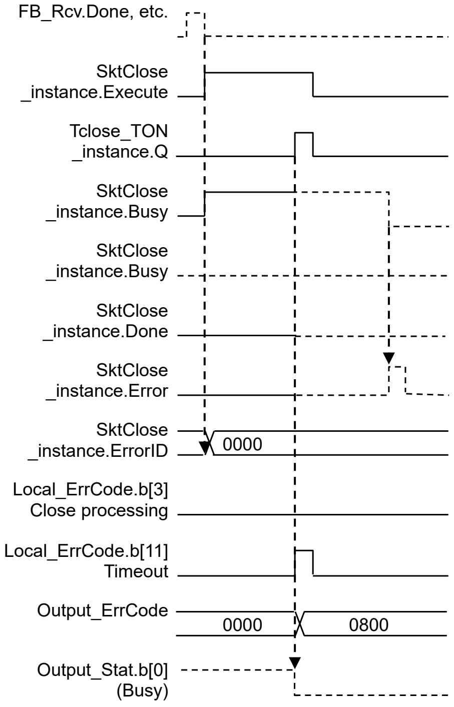
● Close Processing



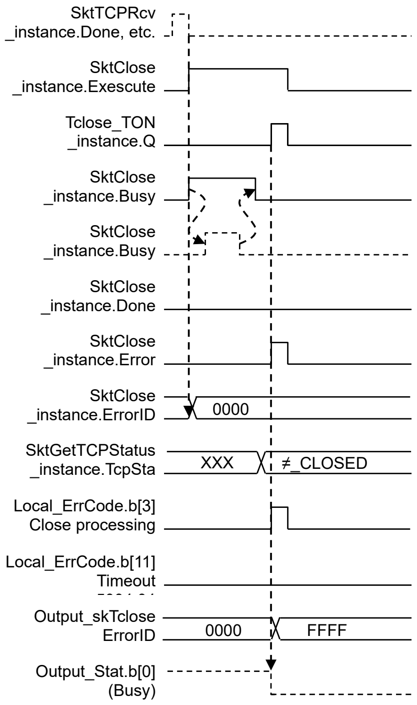
(Normal End)



(Error End)



(Timeout)



(Status Error)

9.7. Error Processing

9.7.1. Error Code List

This section lists error codes that can occur during the execution of the ST language program.

- TCP Connection Status Error (*Output_EtnTcpSta*)

  If the TCP connection status does not return to the normal state (*_CLOSED*) within the specified time after close processing, the TCP connection status code is set in the variable *Output_EtnTcpSta*. (If close processing ends with an error, the variable is checked together.)

Error code enumerator [_eCONNECTION_STATE]	Description
_CLOSED	Connection closed (Normal state)
_LISTEN	Waiting for a connection
_SYN SENT	SYN sent in an active state
_SYN RECEIVED	SYN sent and received
_ESTABLISHED	Connection established
_CLOSE WAIT	Waiting for a finish after FIN received
_FIN WAIT1	Finished and FIN sent
_CLOSING	Finished and FIN exchanged Waiting for FIN acknowledgment (ACK)
_LAST ACK	FIN received and finished Waiting for FIN acknowledgment (ACK)
_FIN WAIT2	FIN acknowledgment (ACK) received Waiting for FIN
_TIME WAIT	Waiting for a silence of twice the maximum segment lifetime (2 MSL) after a finish

● Error Codes (Output_SktCmdsErrorID, Output_SkTcloseErrorID)

If an error occurs in open processing, send processing, or receive processing, the error code is set in the variable *Output_SktCmdsErrorID* before execution of close processing. If an error occurs in close processing, the error code is set in the variable *Output_SkTcloseErrorID* and the processing ends. The table below shows the main error codes.

(O: Open processing (SktTCPConnect instruction), S: Send processing (SktTCPSend instruction), R: Receive processing (SktTCPRcv instruction), C: Close processing (SktClose instruction), ○: Applicable processing)

Error code	O	S	R	C	Description
#16#0000	○	○	○	○	Normal end
#16#0400	○	○	○	---	An input parameter for an instruction exceeded the valid range for an input variable.
#16#0407	---	○	○	---	The calculation result of the instruction exceeded the valid range for the data area for output parameters.
#16#2000	○	---	---	---	The instruction was executed with a local IP address setting error.
#16#2002	○	---	---	---	The instruction failed to resolve the address of the remote node with the specified domain name.
#16#2003	○	○	○	---	The instruction was not executed in appropriate state. • SktTCPConnect instruction The TCP port specified by the input variable *SrcTcpPort* is already open. The remote node specified by the input variable *DstAdr* does not exist. The remote node specified by the input variables *DstAdr* and *DstTcpPort* is not waiting for a connect request. • SktTCPRcv instruction The specified socket is in receive processing. A connection is not established for the specified socket. • SktTCPSend instruction The specified socket in send processing. A connection is not established for the specified socket.
#16#2006	---	---	○	---	A timeout occurred for the socket service instruction.
#16#2007	---	○	○	○	The handle specified in the socket service instruction is invalid.
#16#2008	○	○	○	○	The instruction was executed in excess of the resources available for simultaneously executable socket service instructions.
#16#FFFF	○	○	○	○	The instruction ended before completion of the execution.

Note

For details, refer to *A-1 Error Codes That You Can Check with ErrorID* and *A-2 Error Codes* in *Appendices* of the *Machine Automation Controller NJ/NX-series Instructions Reference Manual* (Cat. No. W502).

Note

For the details and corrections of the built-in EtherNet/IP port, refer to *8-7 Precautions in Using Socket Services* in *Section 8 Socket Service* of the *Machine Automation Controller NJ/NX-series CPU Unit Built-in EtherNet/IP Port User’s Manual* (Cat. No. W506).

● Error Flags (Error End, Timeout) (*Output_ErrCode*)

If open, send, receive, or close processing ends with an error or times out, an error flag is set in the variable *Output_ErrCode*, and an error code is stored in the variable *Output_SktCmdsErrorID* or *Output_SkTcloseErrorID*.

(If close processing ends with error or times out, the TCP connection status error variable *Output_EtnTcpSta* is also checked together.)

(O: Open processing (SktTCPConnect instruction), S: Send processing (SktTCPSend instruction), R: Receive processing (SktTCPRcv instruction), C: Close processing (SktClose instruction), ○: Applicable processing)

Error flag	O	S	R	C	Description
#16#0000	○	○	○	○	Normal end
#16#0001		○			Send processing ended with an error
#16#0002			○		Receive processing ended with an error
#16#0004	○				Open processing ended with an error
#16#0008				○	Close processing ended with an error
#16#0100		○			Send processing not completed within specified time
#16#0200			○		Receive processing not completed within specified time (This includes cases where response to be received was not received.)
#16#0400	○				Open processing not completed within specified time
#16#0800				○	Close processing not completed within specified time
#16#0010					Processing number error
#16#0020					Send/Receive required judgment error
#16#1000					Code reader error
#16#2000					Code reader FCS (checksum) error
#16#8000	○	○	○	○	Error occurred

* Each error flag stores the sum of error flag values detected in each processing.

● Code Reader Error Codes

If the receive data from the code reader is error data, an error code is stored in the variable *Output_MErrCode*.

Error code	Description
#16#00000000	Normal End
#16#FFFFFFFF	Not executed

9.7.2. TCP Connection Status Error and Correction

This section describes the situation and corrections if a TCP connection status error occurs.

● Effect of a TCP Connection Status Error

If, after the occurrence of a TCP connection status error, you execute the project file again without taking any corrective action or without noticing the error, the following error may occur: *The remote node specified by the input variable DstAdr (Destination Address) or DstTcpPort (Destination Port) is not waiting for a connect request.* (Hereinafter, this error is referred to as “open processing error”.) This is considered as the effect of the TCP connection status error at the end of the previous communications processing. Refer to *9.7.1 Error Code List* for details of errors that occurred.

● Situation When a TCP Connection Status Error Occurs

Both a TCP connection status error after close processing and an open processing error in the next communications processing due to the effect of the TCP connection status error can occur because the close processing has not completed in the code reader. In this situation, despite that the controller has ended all processing steps (up to close processing) in the project file, it has not received the close completion notification from the code reader (i.e., the completion of the close processing in the code reader is not confirmed).

● Correction

Check whether the communications port of the code reader is closed since the close processing may not be completed in the code reader. As a result, if the communications port of the code reader is not closed or its state cannot be confirmed, the communications port must be reset. To reset the communications port of the code reader, you can use software restart or turn OFF and then ON the power supply. For details, refer to the manual for the code reader.

Precautions for Correct Use

Reset the communication port of the code reader after confirming that it is not connected to another device.

● Situation When a TCP Connection Status Error Occurs in the Controller (Built-in EtherNet/IP Port)

When a TCP connection status error occurs, the project file has ended its processing, but resending and time monitoring by the built-in EtherNet/IP port (TCP/IP function) may be active, as described in *Resending and Time Monitoring Using the Built-in EtherNet/IP Port (TCP/IP)* in *9.3.2. Time Monitoring Function*. However, this resending will stop under the following situations, so there is no particular need to consciously stop it.

• The project file is executed and an open processing request is issued again.

• A communications problem such as cable disconnection is resolved during resending.

• Resend processing is ended by the TCP/IP time monitoring (timeout) function.

• The controller is restarted or turned OFF.

10. Revision History

Revision Code	Revision Date	Revised Page and Reason
01	March 2023	First Publication

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 ASIA PACIFIC PTE. LTD.
438B Alexandra Road, #08-01/02 Alexandra
Technopark, Singapore 119968
Tel: (65) 6835-3011 Fax: (65) 6835-2711

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

Authorized Distributor:

Cat. No. Z412-E1-01 0323