OMRON

Machine Automation Controller NJ-series

Ethernet Connection Guide (TCP/IP)

OMRON Corporation

FQ-CR-Series Code Reader

Network Connection Guide

sysmac
always in control

P532-E1-01

Table of Contents

1. Related Manuals

The table below lists the manuals related to this document.

To ensure system safety, make sure to always read and heed the information provided in all Safety Precautions, Precautions for Safe Use, and Precaution for Correct Use of manuals for each device which is used in the system.

Cat. No.	Model	Manual name
W500	NJ501-□□□□	NJ-series CPU Unit Hardware User's Manual
W501	NJ501-□□□□	NJ-series CPU Unit Software User's Manual
W506	NJ501-□□□□	NJ-series CPU Unit Built-in EtherNet/IP Port User's Manual
W504	SYSMAC-SE2□□□	Sysmac Studio Version 1 Operation Manual
W502	NJ501-□□□□	NJ-series Instructions Reference Manual
Z315	FQ-CR1 series	Fixed Mount Multi Code Reader User's Manual
Z316	FQ-CR2 series	Fixed Mount 2D Code Reader User's Manual

2. Terms and Definition

Terms	Explanation and Definition
IP address	Ethernet uses an IP address to perform communications. The IP address (Internet Protocol Address) is an address that is used to identify a node (host computer or controller, etc.) on the Ethernet. IP addresses must be set and managed so they do not overlap.
Socket	A socket is an interface that allows you to directly use TCP or UDP functions from the user program. The NJ-series Machine Automation Controller performs socket communications by using the socket service instructions provided as standard features. To use the socket services, connections with a destination node must be established and terminated. In this document, establishment processing is called "socket open" or "TCP open" and termination processing is called "socket close" or "close". The socket services enable data exchange with destination nodes.
Active and Passive	Open processing is executed for each node to establish a connection. The open method depends on whether the node is opened as a server or client. In this document, the method used to open a node as a server is called "passive open" and the method used to open a node as a client is called "active open" or "open processing (active)".
Keep-alive function	When the keep-alive function is used with TCP/IP socket services, the keep-alive communications frame is used to check the status of the connection with the destination node (either a server or client) if there are no communications during the specified time interval. Checks are executed at a certain interval, and if there is no response to any of them then the connection is terminated.
Linger function	This is an option for the TCP socket that enables immediate open processing using the same port number without waiting until the port number opens after RST data is sent when the TCP socket closes. If the linger option is not specified, FIN data will be sent when a TCP socket is closed, and then approximately 1 minute will be required to confirm the transmission and perform other closing management with the destination node. Therefore, it may not be possible to immediately use TCP sockets with the same port number.

3. Remarks

(1) Understand the specifications of devices which are used in the system. Allow some margin for ratings and performance. Provide safety measures, such as installing safety circuit in order to ensure safety and minimize risks for abnormal occurrence.

(2) To ensure system safety, always read and heed the information provided in all Safety Precautions, Precautions for Safe Use, and Precaution for Correct Use of manuals for each device used in the system.

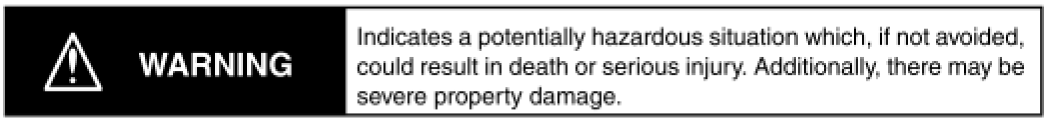
(3) The users are encouraged to confirm the standards and regulations that the system must conform to.

(4) It is prohibited to copy, to reproduce, and to distribute a part of or whole part of this document without the permission of OMRON Corporation.

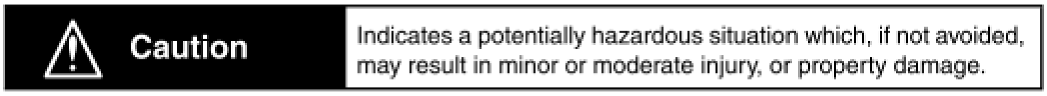
(5) This document provides the latest information as of March 2013. The information on this manual is subject to change for improvement without notice.

About Intellectual Property Right and Trademarks

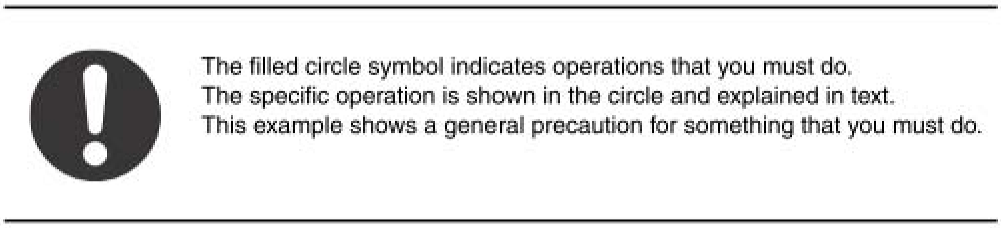
The following notation is used in this document.



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



⚠ Caution	Indicates a potentially hazardous situation which, if not avoided, may result in minor or moderate injury, or property damage.



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.

Precautions for Safe Use

Indicates precautions on what to do and what not to do to ensure using the product safely.

Precautions for Correct Use

Indicates precautions on what to do and what not to do to ensure proper operation and performance.

Additional Information

Provides useful information.
Additional information to increase understanding or make operation easier.

4. Overview

This document describes the procedure for connecting the Code Reader (FQ-CR series) of OMRON Corporation (hereinafter referred to as OMRON) to the NJ-series Machine Automation Controller (hereinafter referred to as Controller) through Ethernet, and provides the procedure for checking their connection.

Refer to the Ethernet communications settings of the prepared project file to understand the setting method and key points to connect the devices via Ethernet.

This project file is used to check the Ethernet connection by sending/receiving the message of VERGET /S (Acquire Software Version) to/from the destination device.

Obtain the latest "Sysmac Studio project file" from OMRON beforehand.

Name	File name	Version
Sysmac Studio project file (extension: SMC)	OMRON_FQCR_ETN(TCP)_EV100.SMC	Ver.1.00

5. Applicable Devices and Support Software

5.1. Applicable Devices

The following devices can be connected.

Manufacturer	Name	Model	Version
OMRON	NJ series CPU Unit	NJ501-□□□□	-
OMRON	Code Reader	FQ-CR10□□□□ FQ-CR15□□□□ FQ-CR20□□□□ FQ-CR25□□□□	

Additional Information

As applicable devices above, the devices listed in Section 5.2. are actually used in this document to check the connection. When using devices not listed in Section 5.2, check the connection by referring to the procedure in this document.

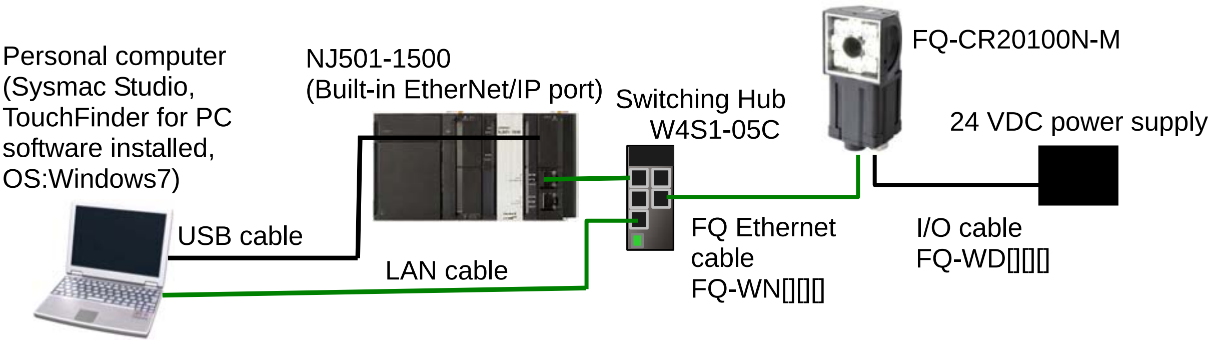
Additional Information

This document describes the procedure to establish the network connection. It does not provide information about operation, installation nor wiring method of each device.

For details on the above products (other than communication connection procedures), refer to the manuals for the corresponding products or contact your OMRON representative.

5.2. Device Configuration

The hardware components to reproduce the connection procedure of this document are as follows.



Manufacturer	Name	Model	Version
OMRON	NJ series CPU Unit (Built-in EtherNet/IP port)	NJ501-1500	
OMRON	Power Supply Unit	NJ-PA3001	
OMRON	Switching Hub	W4S1-05C	
OMRON	Sysmac Studio	SYSMAC-SE2□□□	Ver.1.00
OMRON	Sysmac Studio project file	OMRON_FQCR_ETN(TCP)_EV100.SMC	Ver.1.00
-	Personal computer (OS:Windows7)		
-	USB cable (USB 2.0 type B connector)		
-	LAN cable (Ethernet STP (Shielded twisted-pair) cable of category 5 or higher)		
OMRON	Code Reader	FQ-CR20100N-M	
OMRON	I/O cable	FQ-WD□□□	
OMRON	FQ Ethernet cable	FQ-WN□□□	
OMRON	Touch Finder for PC		Ver.1.20

Precautions for Correct Use

Obtain the latest Sysmac Studio project file from OMRON in advance.
(To obtain the file, contact your OMRON representative.)

Additional Information

It may not be possible to reproduce the same operation with different devices and versions. Check the configuration, model and version. If they are different from your configuration. Contact your OMRON representative.

Additional Information

In this document, a USB is used to connect with the Controller. For information on how to install a USB driver, refer to *A-1 Driver Installation for Direct USB Cable Connection* of the *Sysmac Studio Version 1 Operation Manual* (Cat.No. W504).

6. Ethernet Settings

This section describes the specifications such as communication parameters and variables that are set in this document.

Additional Information

This document and project file can be used to perform operations using the settings and command described in this section. Modifications are necessary to perform communications using different settings.

6.1. Ethernet Communications Settings

The settings required for Ethernet communications are shown below.

6.1.1. Communications Settings between Personal Computer and Code Reader

This document explains the procedure for setting the Code Reader using the personal computer with the setting example shown in the table below.

	Personal computer used for setting	Code Reader
IP address	10.5.5.101	10.5.5.100 (Default)
Subnet mask	255.255.255.0	255.255.255.0 (Default)
Gateway	-	Blank (Default)

*In this document, the gateway setting is unnecessary because the connection is made in the same segment.

6.1.2. Communications Settings between the Controller and Code Reader

This document explains the procedure for connecting the Controller and Code Reader using the setting example shown in the table below.

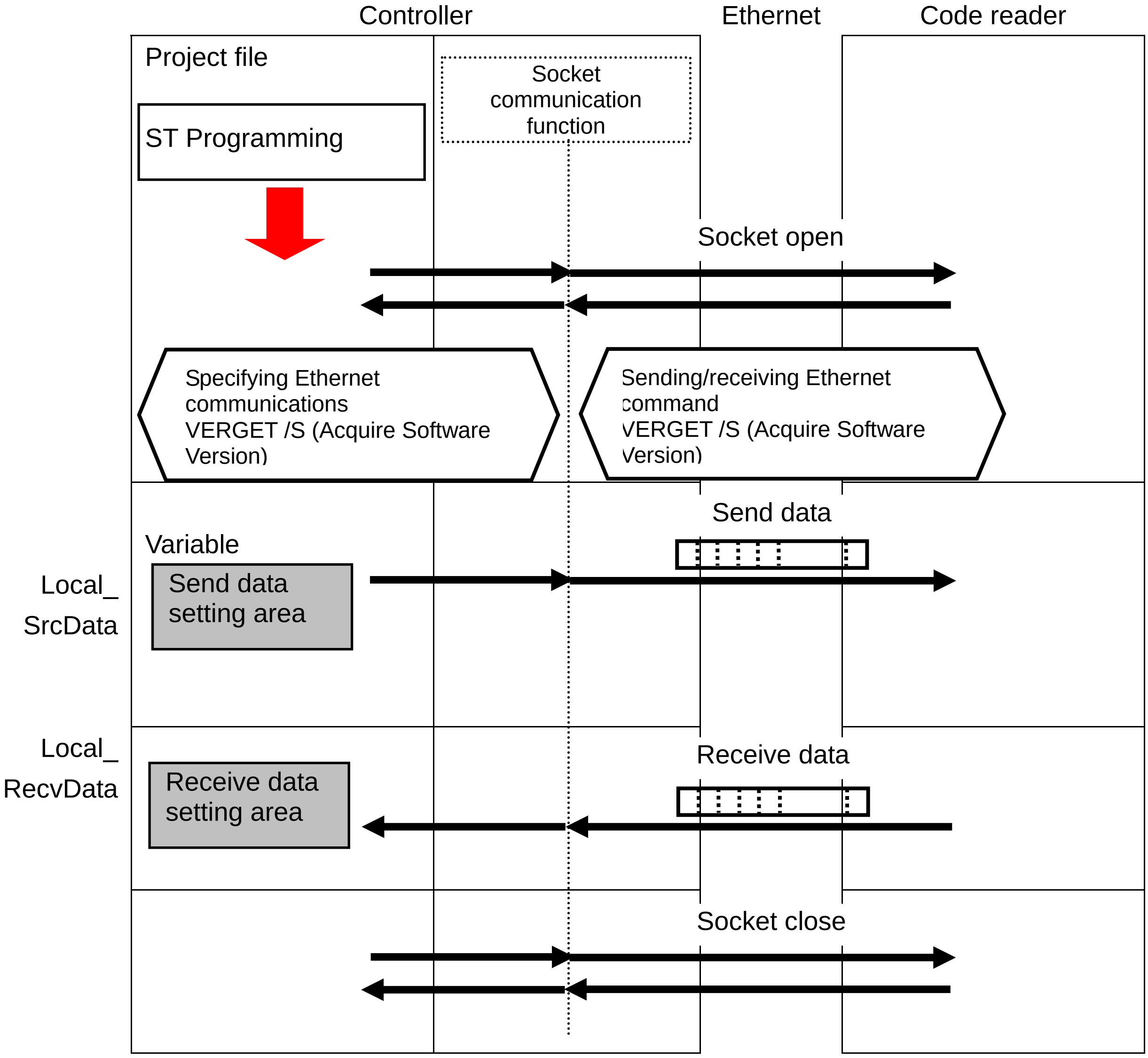
	Controller	Code Reader
IP address	192.168.250.1	192.168.250.2
Subnet mask	255.255.255.0	255.255.255.0 (Default)
Gateway	-	0.0.0.0 (Default)
Auto	-	OFF
Port number	(Set with the program)	9876 (Fixed)

*In this document, the gateway setting is unnecessary because the connection is made in the same segment.

6.2. Example of Checking Connection

This document shows an example of a Structured Text (ST) program in which the Controller executes socket open, send/receive, and socket close processing on the Code Reader.

The message of VERGET /S (Acquire Software Version) is sent and received between the Controller and Code Reader. The following figure outlines the operation.


Controller
Ethernet
Code reader
Project file
ST Programming
Socket communication function
Socket open
Specifying Ethernet communications VERGET /S (Acquire Software Version)
Sending/receiving Ethernet command VERGET /S (Acquire Software Version)
Send data
Variable
Local_ SrcData
Send data setting area
Local_ RecvData
Receive data
Receive data setting area
Socket close

7. Connection Procedure

This section describes how to connect the Controller on the Ethernet network.
This document explains the procedures for setting the Controller and Code Reader from the factory default setting. For the initialization, refer to *Section 8 Initialization Method*.

7.1. Work Flow

Take the following steps to connect the controllers via Ethernet.

7.2 Setting up the Code Reader	Set up the Code Reader.
↓ 7.2.1 Parameter Setting	Set the parameters of the Code Reader.
↓ **7.3 Setting up the Controller**	Set up the Controller.
↓ 7.4.1 Starting the Sysmac Studio and Importing the Project File	Start the Sysmac Studio Automation Software, and import the Sysmac Studio project file.
↓ 7.3.2 Checking the Parameters and Building	Check the set parameters, execute the program check on the project data and build the program.
↓ 7.3.3 Going Online and Transferring the Project Data	Connect online with the Sysmac Studio and transfer the project data to the Controller.
↓ **7.4 Connection Status Check**	Execute the project file that was transferred and confirm that Ethernet communications are normally performed.
↓ 7.4.1 Executing the Project File and Checking the Receive Data	Execute the project file and check if the correct data are written to the variables of the Controller.

Precautions for Correct Use

Obtain the latest Sysmac Studio project file from OMRON in advance.
(To obtain the file, contact your OMRON representative.)

7.2. Setting Up the Code Reader

Set up the Code Reader.

Precautions for Correct Use

Use a personal computer to set the parameters of the Code Reader.
Note that the settings of the personal computer may need to be changed.

7.2.1. Parameter Setting

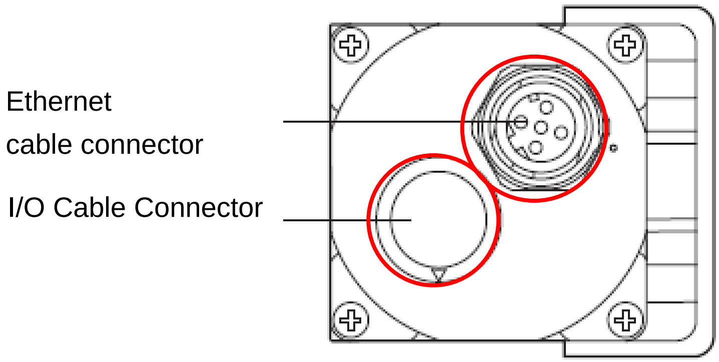
Set the parameters of the Code Reader.

PC tool for FQ (TouchFinder for PC) is used to set the parameters. Install the software in the personal computer beforehand.

Set the IP address of the personal computer to 10.5.5.101.

1 Connect the Ethernet cable connector, which is located at the bottom of the Code Reader, to the Switching Hub using the FQ Ethernet Cable.

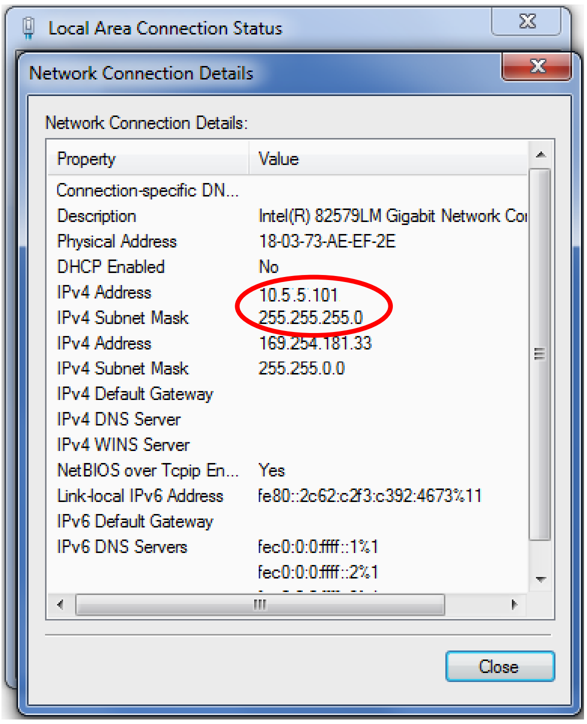
Connect the I/O cable to the I/O cable connector, and then turn ON the 24 VDC power supply.



2 Start TouchFinder for PC (PC tool for FQ) on the personal computer which is connected to the Switching Hub.

*Set the IP address of the personal computer to 10.5.5.101. Use the following procedure to check the IP address of the personal computer.

(1) Execute **Network and Sharing Center** from Control Panel.
(2) Double-click **Local Area Connection** on the Network and Sharing Center Window.
(3) Click the **Details** Button on the Local Area Connection Status Dialog Box.
(4) Check that the IP address is set to 10.5.5.101.



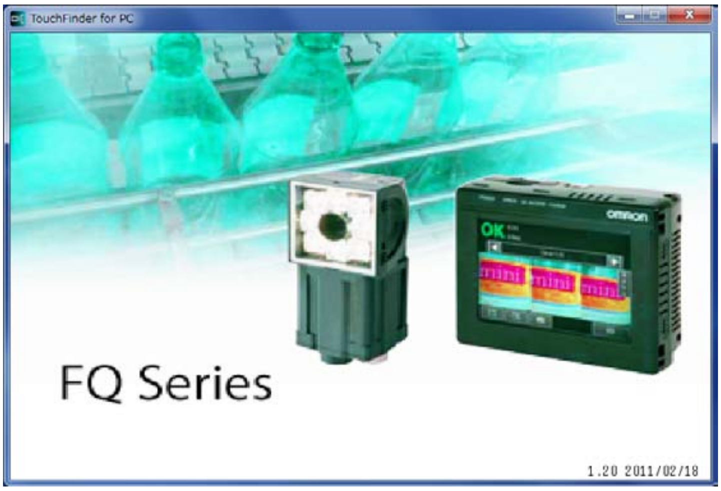
Network Connection Details

Network Connection Details:

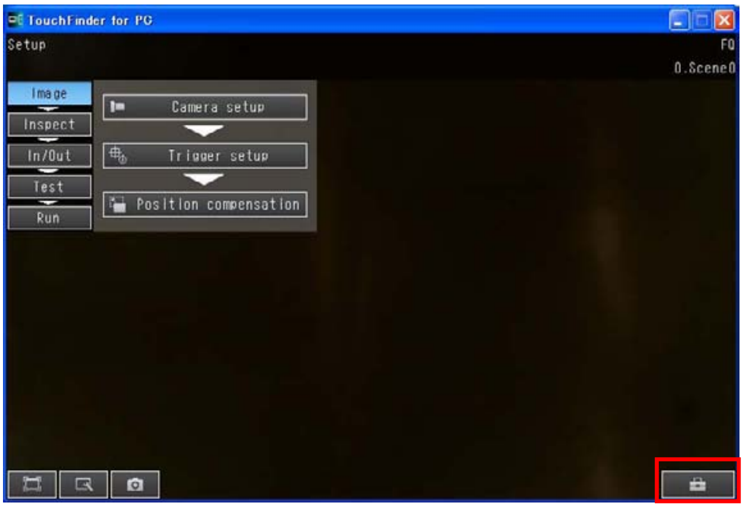
Property	Value
Connection-specific DN...	
Description	Intel(R) 82579LM Gigabit Network Co
Physical Address	18-03-73-AE-EF-2E
DHCP Enabled	No
IPv4 Address	10.5.5.101
IPv4 Subnet Mask	255.255.255.0
IPv4 Address	169.254.181.33
IPv4 Subnet Mask	255.255.0.0
IPv4 Default Gateway	
IPv4 DNS Server	
IPv4 WINS Server	
NetBIOS over Tcpip En...	Yes
Link-local IPv6 Address	fe80::2c62:c2f3:c392:4673%11
IPv6 Default Gateway	
IPv6 DNS Servers	fec0:0:0:ffff::1%1
	fec0:0:0:ffff::2%1

3 The start screen of Touch Finder for PC is displayed.

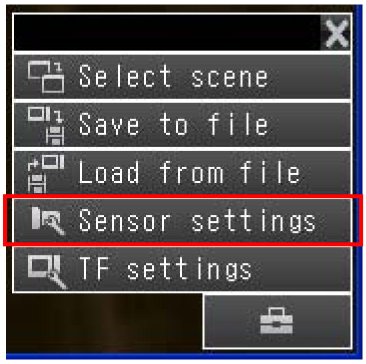
*Select a language at the first startup. In the following example, English is selected.



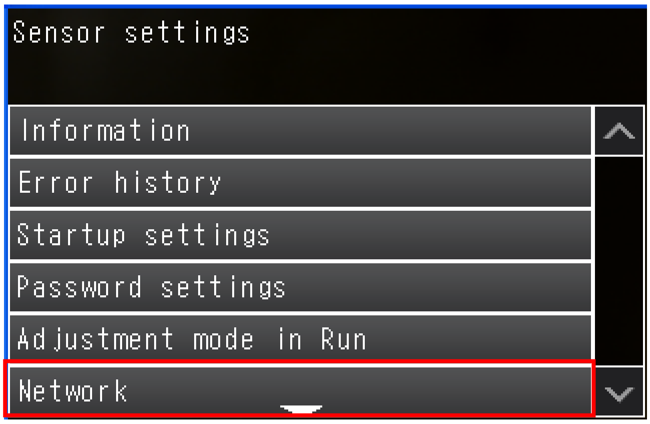
4 Click the 🧰 icon located at the right bottom of the Touch Finder for PC Window.

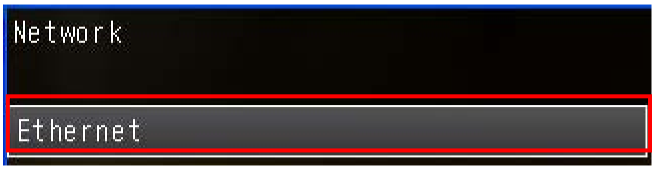
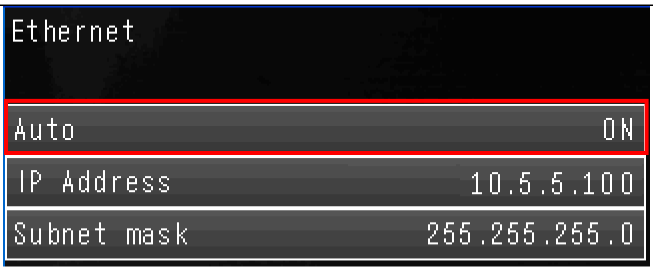
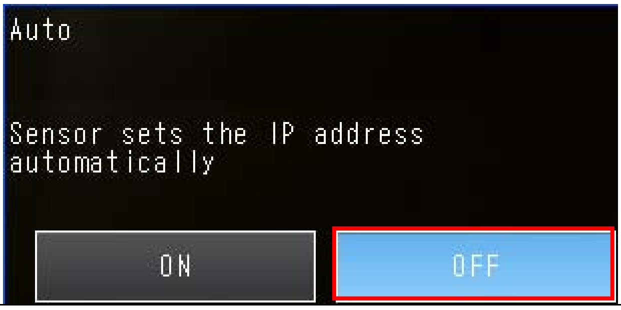


5 Click **Sensor Settings** on the pop-up screen.



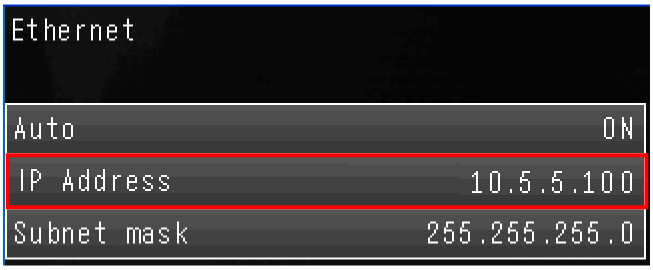
6 Click **Network** on the Sensor settings Menu.



7	Click **Ethernet** on the Network Menu.	
8	Turn OFF the auto setting of the Ethernet. Click **Auto** on the Ethernet Menu.	 Auto ON IP Address 10.5.5.100 Subnet mask 255.255.255.0
	Click **OFF** on the Auto Menu.	 Sensor sets the IP address automatically

9 Set the fixed IP address.

Click **IP Address** on the Ethernet Menu.



Ethernet

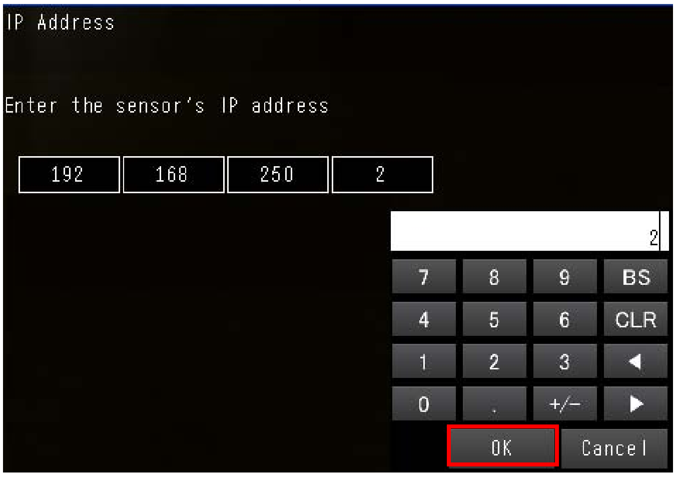
Auto ON

IP Address 10.5.5.100

Subnet mask 255.255.255.0

Click each octet on the IP Address Screen. A numeric keypad is displayed. Click the numeric keypad and enter each octet of the IP address.
Click **OK**.

Set the IP address to 192.168.250.2.

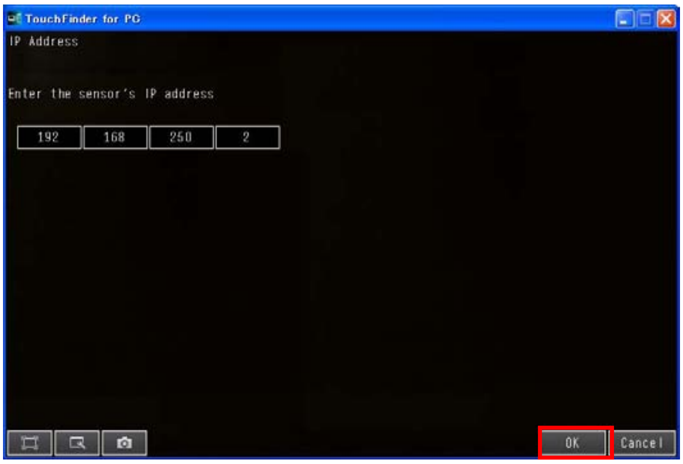


IP Address

Enter the sensor's IP address

192 168 250 2

Click **OK**. This completes the IP address setting.



TouchFinder for PC

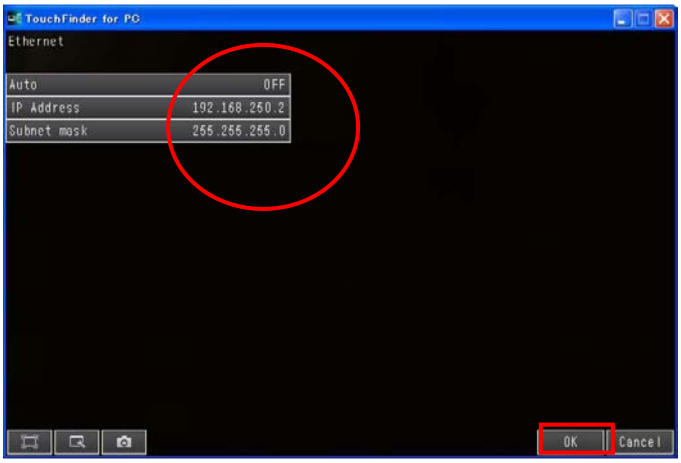
IP Address

Enter the sensor's IP address

192 168 250 2

10 Confirm the settings are made as follows and click **OK**.

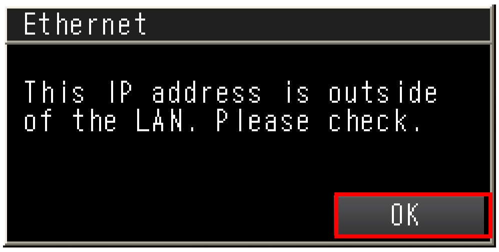
Auto: OFF
IP address: 192.168.250.2
Subnet mask: 255.255.255.0



Ethernet

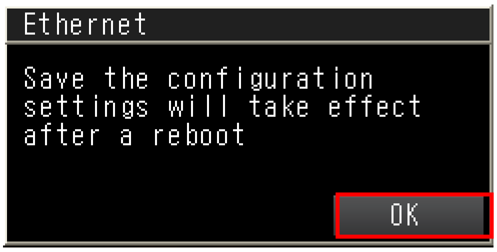
Auto	OFF
IP Address	192.168.250.2
Subnet mask	255.255.255.0

If the dialog box on the right is displayed, click **OK**.


Ethernet

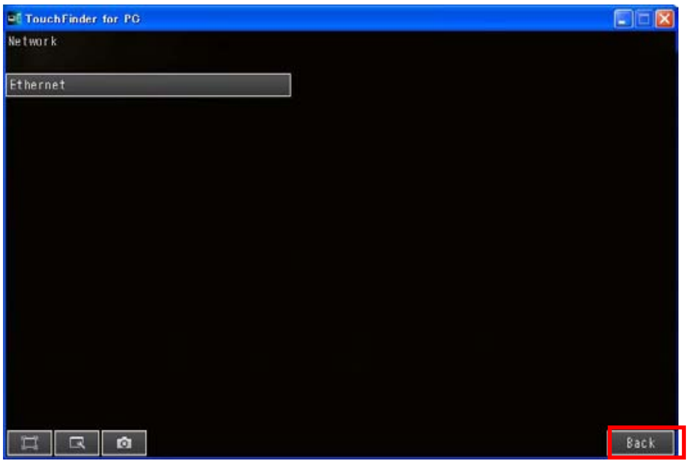
This IP address is outside of the LAN. Please check.

If the dialog box on the right is displayed, click **OK**.


Ethernet

Save the configuration settings will take effect after a reboot

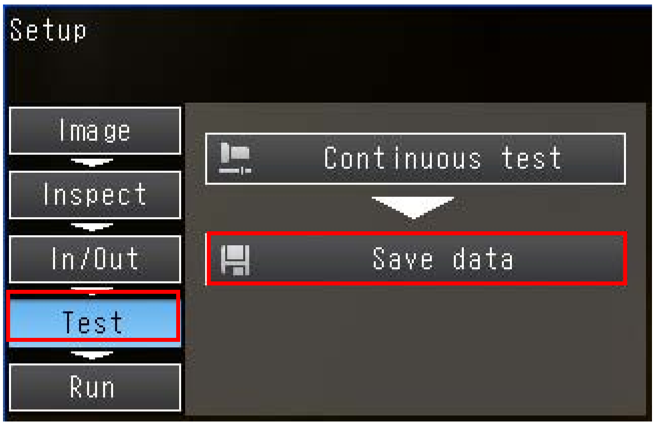
Click **Back** twice to return to the screen in step 4.



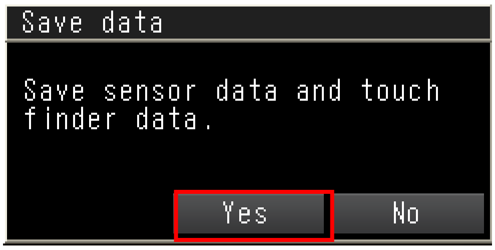
11 Save the data.

Click **Test** on the Setup Screen.

Click **Save** data.



Click **Yes** on the Save data Dialog Box.



Save sensor data and touch finder data.

12 Cycle the power supply to the Code Reader.

*The parameters that were changed after cycling the power supply are reflected.

7.3. Setting Up the Controller

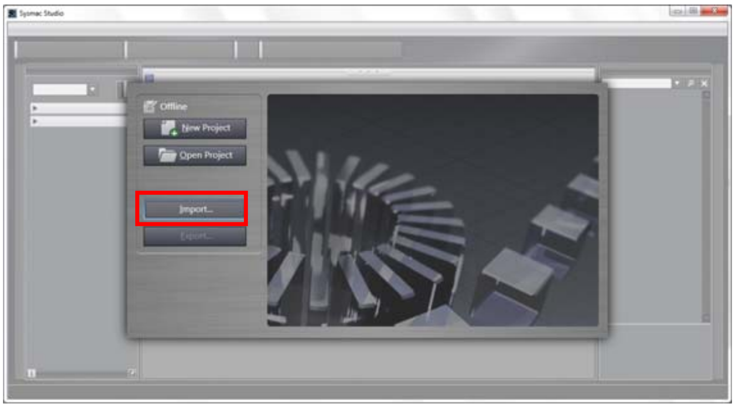
Set up the Controller.

7.3.1. Starting the Sysmac Studio and Importing the Project File

Start the Sysmac Studio Automation Software, and import the Sysmac Studio project file. The software and USB driver must be installed beforehand. Connect a USB cable to the personal computer and to the Controller, and turn ON the power supply to the Controller.

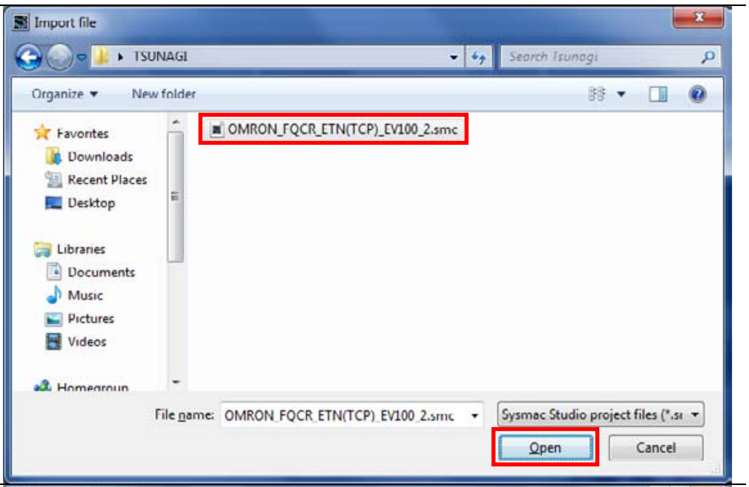
1 Start the Sysmac Studio.
Click the **Import** Button.

*If a confirmation dialog for an access right is displayed at start, select to start.



2 The Import File Dialog Box is displayed. Select OMRON_FQCR_ETN(TCP)_EV100.SMC (Sysmac Studio project file) and click the **Open** Button.

*Obtain the Sysmac Studio project file from OMRON.

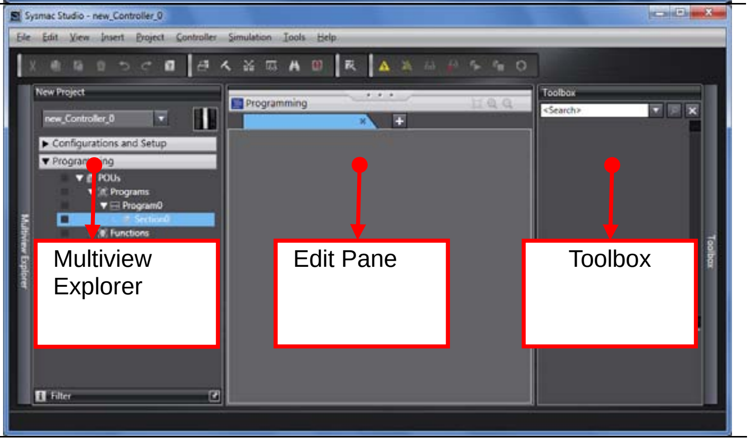


OMRON_FQCR_ETN(TCP)_EV100_2.smc

File name: OMRON_FQCR_ETN(TCP)_EV100_2.smc

Sysmac Studio project files (*.sı

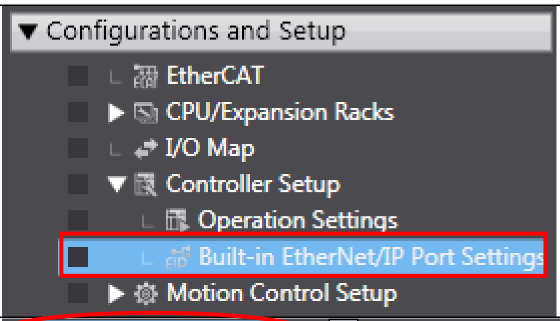
3 OMRON_FQCR_ETN(TCP)_EV100 project is displayed.
The left pane is called Multiview Explorer, the right pane is called Toolbox and the middle pane is called Edit Pane.



7.3.2. Checking the Parameters and Building

Check the set parameters, execute the program check on the project data and build the program.

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



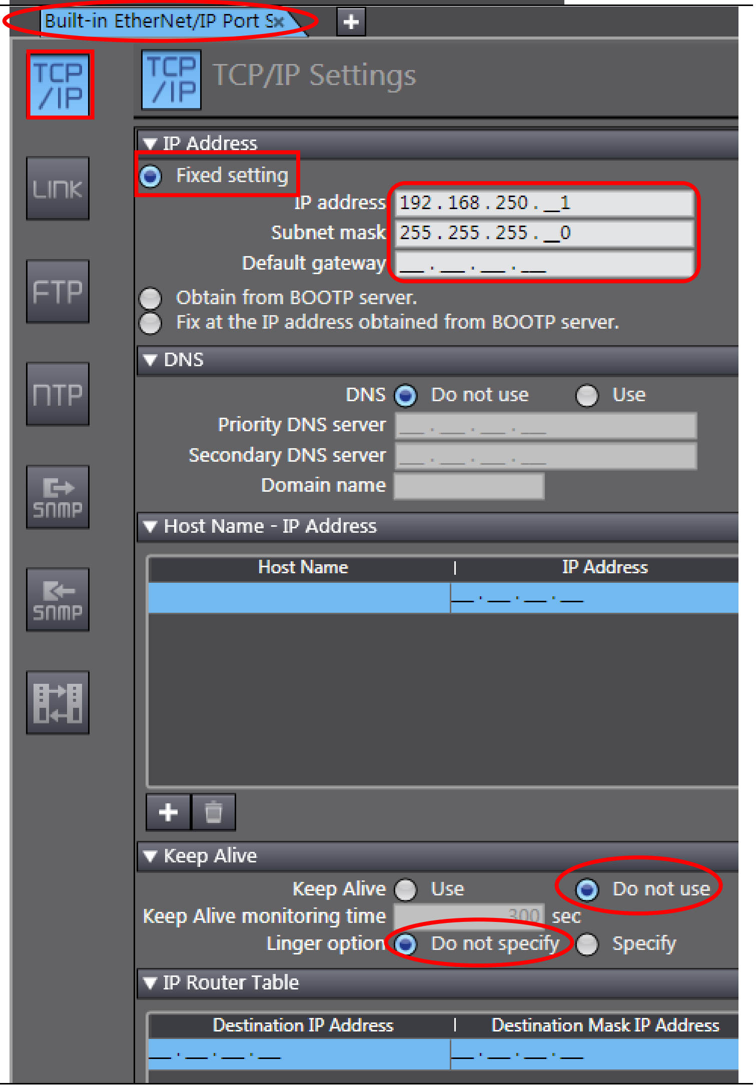
2 The Built-in EtherNet/IP Port Settings Tab Page is displayed in the Edit Pane.

Select the **TCP/IP Setting** Button, select the *Fixed Setting* Option in the IP Address Field, and check that the following settings are made.

IP address: 192.168.250.1
Subnet mask: 255.255.255.0
Default gateway:_._._._

Check that the Keep Alive settings are set as follows.

Keep Alive: Do not use
Linger option: Do not specify

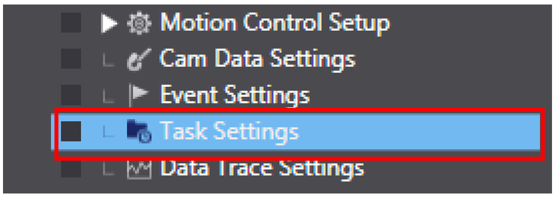


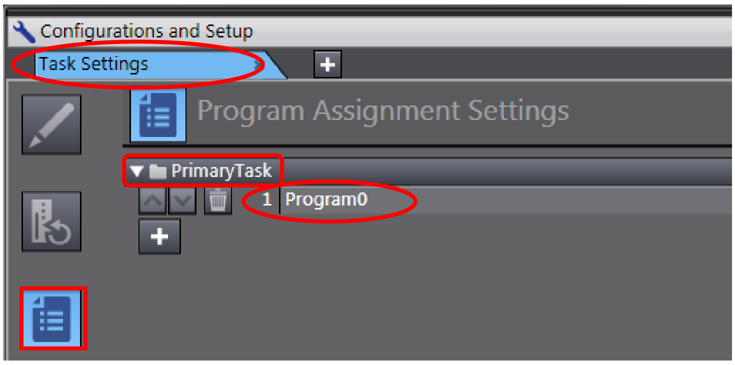
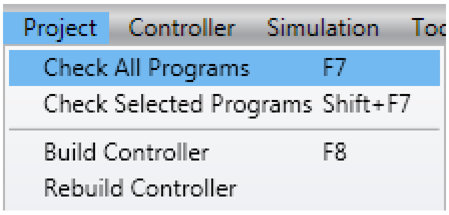
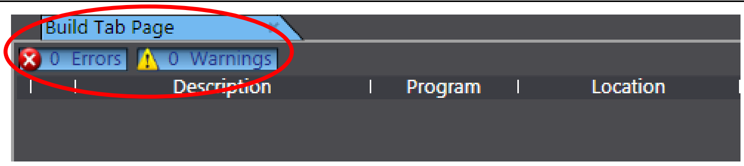
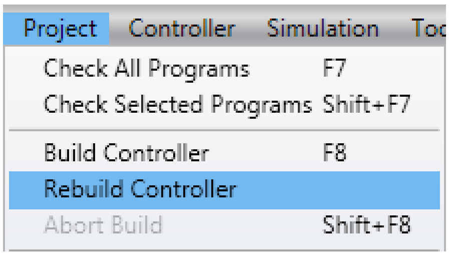
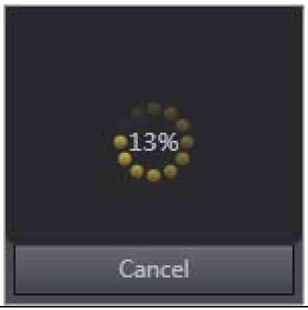
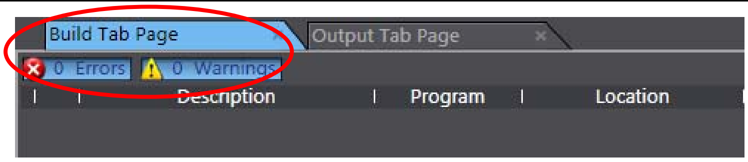
● Fixed setting
IP address 192 . 168 . 250 . _1
Subnet mask 255 . 255 . 255 . _0
Default gateway __ . __ . __ . __
○ Obtain from BOOTP server.
○ Fix at the IP address obtained from BOOTP server.

DNS ● Do not use ○ Use
Priority DNS server __ . __ . __ . __
Secondary DNS server __ . __ . __ . __
Domain name

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

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



4	The Task Settings Tab Page is displayed in the Edit Pane. Select the **Program Assignment Settings** Button and check that Program0 is set under PrimaryTask.	 Configurations and Setup Task Settings Program Assignment Settings PrimaryTask 1 Program0
5	Select ***Check All Programs*** from the Project Menu.	 Project Controller Simulation Too Check All Programs F7 Check Selected Programs Shift+F7 Build Controller F8 Rebuild Controller
6	The Build Tab Page is displayed in the Edit Pane. Check that “0 Errors” and “0 Warnings” are displayed.	 Build Tab Page 0 Errors 0 Warnings Description Program Location
7	Select ***Rebuild Controller*** from the Project Menu. A screen is displayed indicating the conversion is being performed.	 Project Controller Simulation Too Check All Programs F7 Check Selected Programs Shift+F7 Build Controller F8 Rebuild Controller Abort Build Shift+F8 
8	Check that “0 Errors” and “0 Warnings” are displayed in the Build Tab Page.	 Build Tab Page Output Tab Page 0 Errors 0 Warnings Description Program Location

7.3.3. Going Online and Transferring the Project Data

Connect online with the Sysmac Studio and transfer the project data to the Controller.

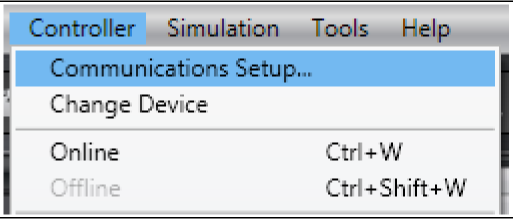
⚠ WARNING

Always confirm safety at the destination node before you transfer a user program, configuration data, setup data, device variables, or values in memory used for CJ-series Units from the Sysmac Studio.
The devices or machines may perform unexpected operation regardless of the operating mode of the CPU Unit.

Additional Information

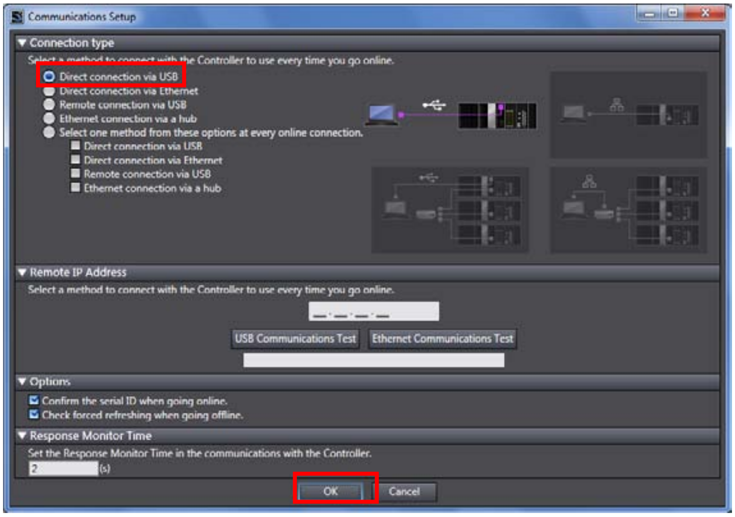
For details on the online connections to a Controller, refer to *Section 5 Going Online with a Controller* in the *Sysmac Studio Version 1.0 Operation Manual* (Cat. No. W504).

1 Select ***Communications Setup*** from the Controller Menu.



2 The Communications Setup Dialog Box is displayed.
Select the *Direct Connection via USB* Option from Connection Type.

Click the **OK** Button.



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 Ethernet
- ○ 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

Select a method to connect with the Controller to use every time you go online.

__.__.__.__

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.

2 (s)

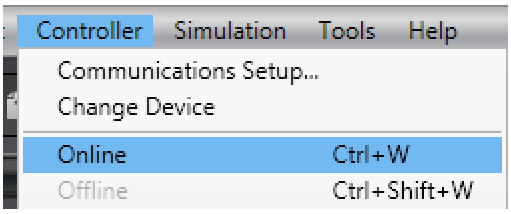
OK | Cancel

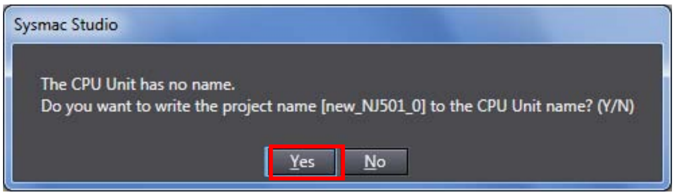
3 Select ***Online*** from the Controller Menu.

A confirmation dialog box is displayed. Click the **Yes** Button.

*The displayed dialog depends on the status of the Controller used. Select the **Yes** Button to proceed with the processing.

*The displayed serial ID differs depending on the device.

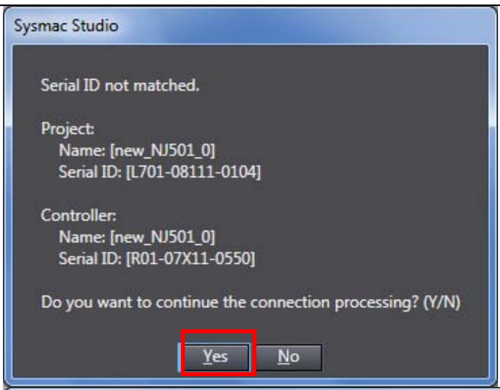
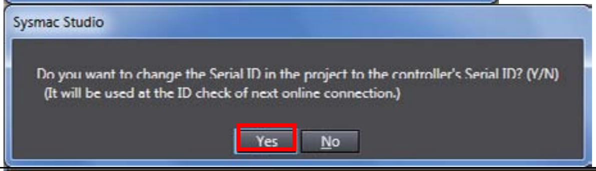
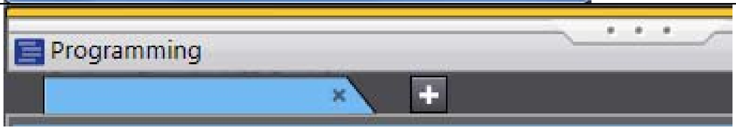
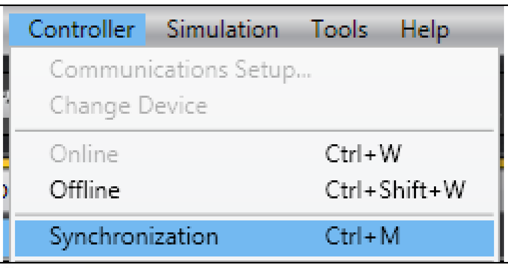
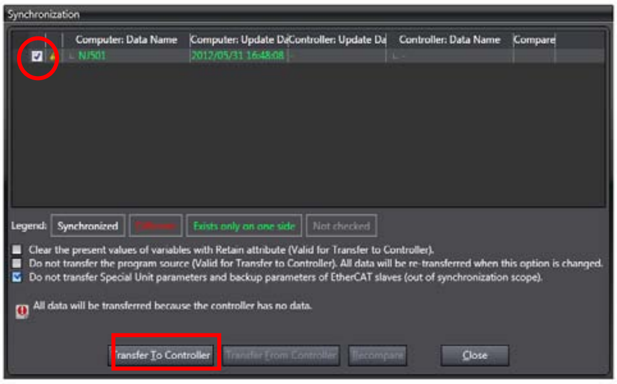
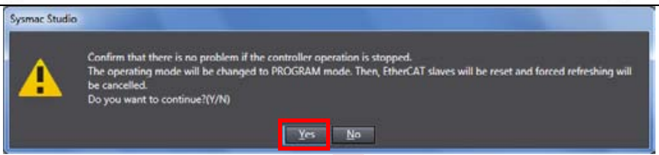
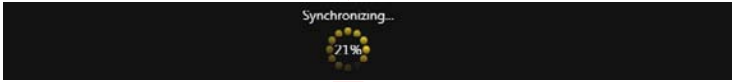
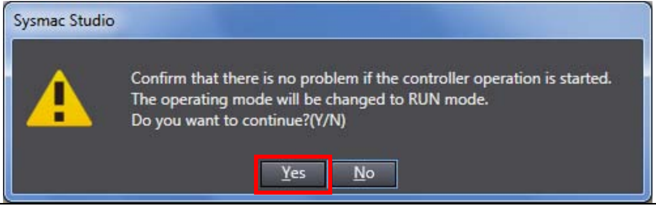


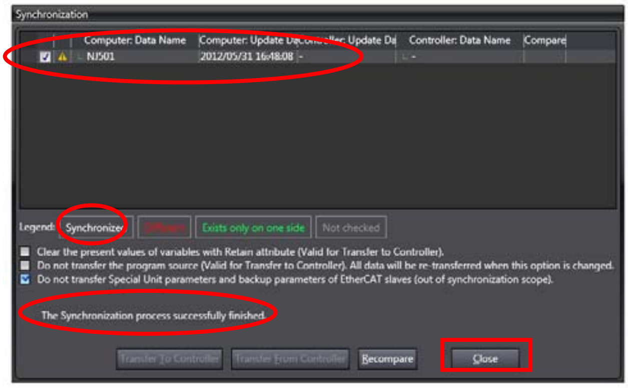


Sysmac Studio

The CPU Unit has no name.
Do you want to write the project name [new_NJ501_0] to the CPU Unit name? (Y/N)

Yes | No

		 Sysmac Studio Serial ID not matched. Project: Name: [new_NJ501_0] Serial ID: [L701-08111-0104] Controller: Name: [new_NJ501_0] Serial ID: [R01-07X11-0550] Do you want to continue the connection processing? (Y/N)  Sysmac Studio Do you want to change the Serial ID in the project to the controller's Serial ID? (Y/N) (It will be used at the ID check of next online connection.)
4	When an online connection is established, a yellow bar is displayed on the top of the Edit Pane.	
5	Select ***Synchronization*** from the Controller Menu.	 Controller Simulation Tools Help Communications Setup... Change Device Online Ctrl+W Offline Ctrl+Shift+W Synchronization Ctrl+M
6	The Synchronization Dialog Box is displayed. Check that the data to transfer (NJ501 in the right figure) is selected. Then, click the **Transfer to Controller** Button.	 Synchronization ☑ NJ501 2012/05/31 16:48:08 Computer: Data Name Computer: Update Da Controller: Update Da Controller: Data Name Compare 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 program source (Valid for Transfer to Controller). All data will be re-transferred when this option is changed. ☑ Do not transfer Special Unit parameters and backup parameters of EtherCAT slaves (out of synchronization scope). All data will be transferred because the controller has no data. Transfer To Controller Transfer From Controller Recompare Close
7	A confirmation dialog is displayed. Click the **Yes** Button. A screen stating "Synchronizing" is displayed. A confirmation dialog is displayed. Click the **Yes** Button.	 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. Do you want to continue?(Y/N)  Synchronizing...  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	Check that the synchronized data is displayed with the color specified by “Synchronized” and that a message is displayed stating "The synchronization process successfully finished". If there is no problem, click the **Close** Button. *If the synchronization fails, check the wiring and repeat the procedure described in this section.	

Synchronization

	Computer: Data Name	Computer: Update Da	Controller: Update Da	Controller: Data Name	Compare
☑ ⚠	NJ501	2012/05/31 16:48:08	-	-	

Legend: Synchronize | 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 program source (Valid for Transfer to Controller). All data will be re-transferred when this option is changed.
☑ Do not transfer Special Unit parameters and backup parameters of EtherCAT slaves (out of synchronization scope).

The Synchronization process successfully finished.

7.4. Connection Status Check

Execute the project file that was transferred and confirm that Ethernet communications are normally performed.

Precautions for Correct Use

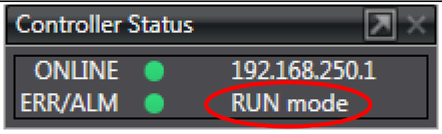
Please confirm that the LAN cable has been connected before proceeding to the following steps.

If it is not connected, turn OFF the power to the devices, and then connect the LAN cable.

7.4.1. Executing the Project File and Checking the Receive Data

Execute the project file and check if the correct data are written to the variables of the Controller.

1 Check that RUN mode is displayed on the Controller Status Pane of the Sysmac Studio.

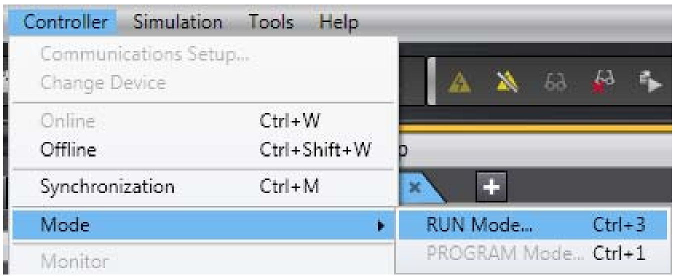


Controller Status

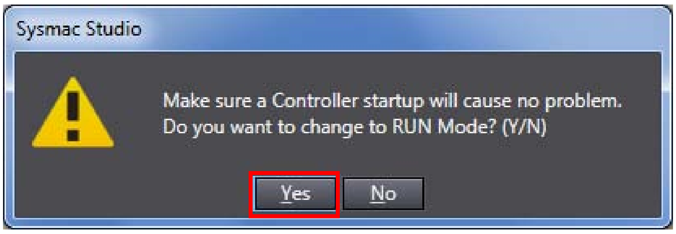
ONLINE ● 192.168.250.1

ERR/ALM ● RUN mode

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



A confirmation dialog box is displayed. Click the **Yes** Button.



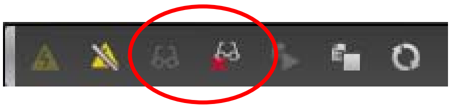
Sysmac Studio

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

2 Check the Monitor Button and Stop Monitoring Button on the toolbar of the Sysmac Studio to see if the Controller is in monitor status.

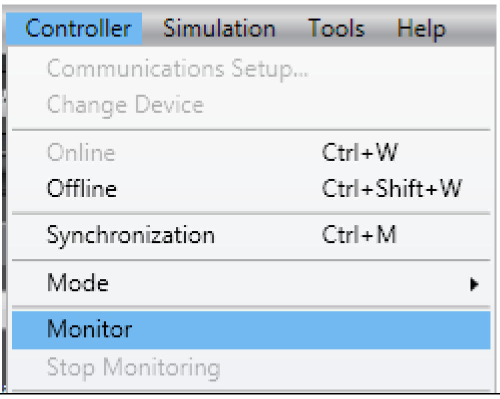
Check that the Monitor Button is selected and grayed out and that the Stop Monitoring Button is selectable (monitor status) as shown in the right figure.

*If the Controller is not in monitor status, select ***Monitor*** from the Controller Menu of the Sysmac Studio.

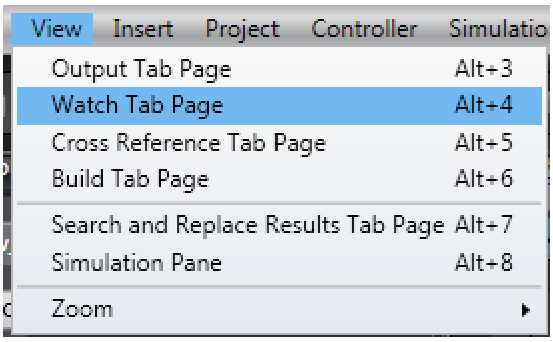


Monitor

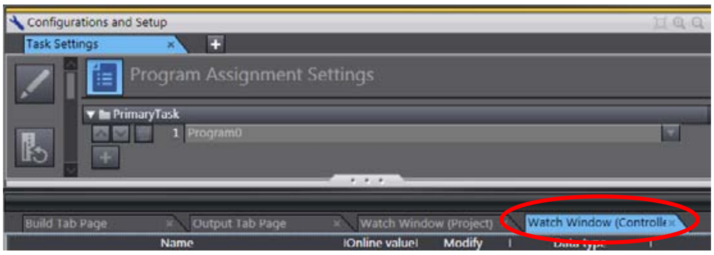
Stop Monitoring



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



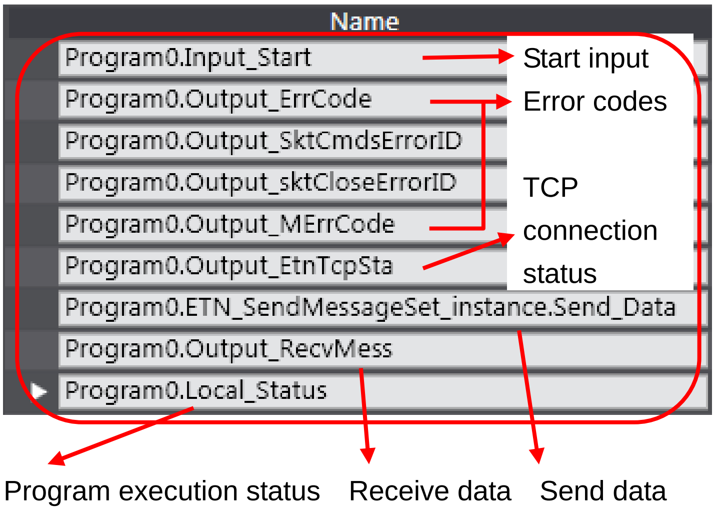
4 The Watch Tab Page is displayed in the lower section of the Edit Pane.



5 Check that the variables shown on the right are displayed in the Name Columns.

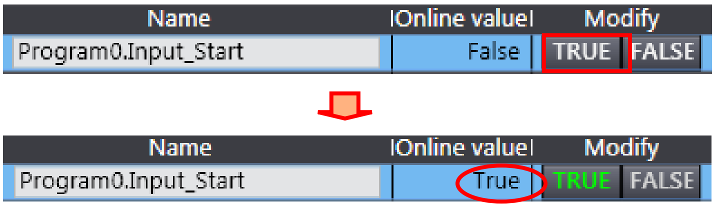
*To add a variable, click *Input Name…*

*Program0 of the Name is omitted from the following descriptions.



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

6 Click **TRUE** on the Modify Column of *Input_Start*.
The Online value of *Input_Start* changes to True.
The program is operated and Ethernet communications are performed with the destination device.

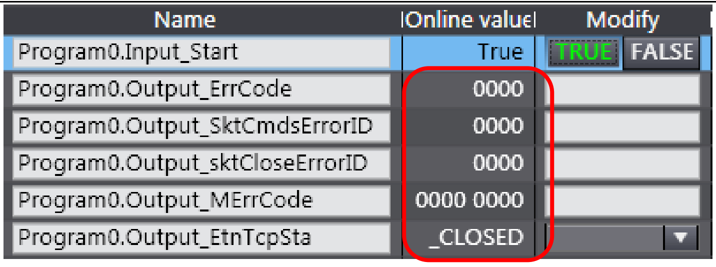


Name	Online value	Modify
Program0.Input_Start	False	TRUE FALSE

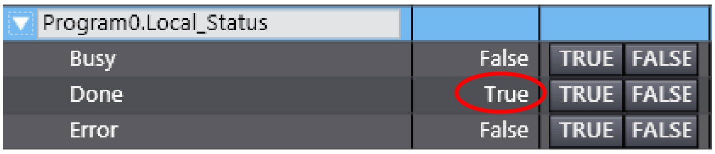
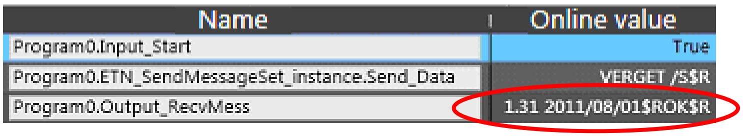
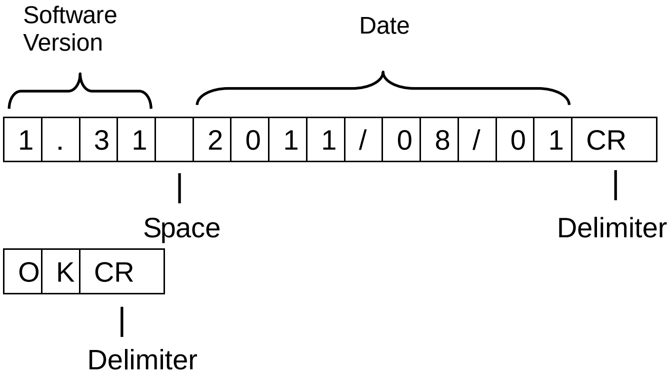
Name	Online value	Modify
Program0.Input_Start	True	TRUE FALSE

7 When the communications end normally, each error code changes to 0.
TCP connection status (*Output_EtnTcpSta*) changes to *_CLOSED*.

*In the case of error end, the error code for an error is stored. For details on error codes, refer to *9.7 Error Process*.



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	

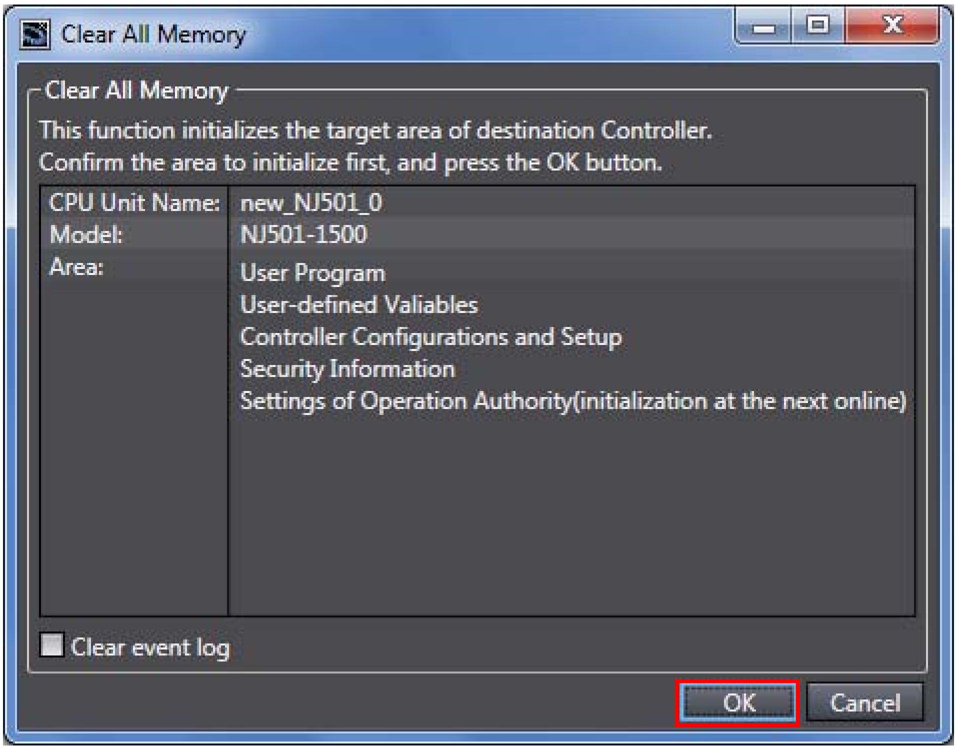
	The Online value of *Local_Status.Done*, which indicates the execution status of the program, changes to True. In the case of error end, *Local_Status.Error* changes to True. *When *Input_Start* changes to FALSE, each *Local_Status* variable also changes to False. For details, refer to *9.6 Timing Charts*.	 Program0.Local_Status Busy False TRUE FALSE Done True TRUE FALSE Error False TRUE FALSE
8	The response data received from the destination device is stored in *Output_RecvMess*. (*ETN_SendMessageSet_instance.Send_Data* is a send command.) Specify variables you want to see in the Watch Tab Page as shown in the right figure and check them. *The response data differ depending on the device used. *Refer to *9.2. Destination Device Command* for details on the command.	 Name / Online value Program0.Input_Start True Program0.ETN_SendMessageSet_instance.Send_Data VERGET /S$R Program0.Output_RecvMess 1.31 2011/08/01$ROK$R <Response format>  Software Version Date 1 . 3 1 2 0 1 1 / 0 8 / 0 1 CR Space Delimiter O K CR Delimiter

8. Initialization Method

This document explains the setting procedure from the factory default setting.
If the device settings have been changed from the factory default setting, some settings may not be applicable as described in this procedure.

8.1. Controller

To initialize the settings of the Controller, select ***Clear All Memory*** from the Controller Menu of the Sysmac Studio.



Clear All Memory

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_NJ501_0
Model:	NJ501-1500
Area:	User Program User-defined Valiables Controller Configurations and Setup Security Information Settings of Operation Authority(initialization at the next online)

□ Clear event log

8.2. Code Reader

For information on how to initialize the Code Reader, refer to *Initializing the Sensor and Touch Finder* under *7-9 Functions Related to the System* in the user's manual for each Code Reader.

9. Project File

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

9.1. Overview

This section explains the specifications and functions of the project file used to check the connection between the Code Reader (FQ-CR series) (hereinafter referred to as destination device) and the Controller (built-in EtherNet/IP port) (hereinafter referred to as Controller).

The project file is a Sysmac Studio project file.
The following data has already been set in this project file.
•Communications settings of the Controller and task settings of program
•A program and function blocks to perform socket communications
•Variable tables and data type definitions of the variables used in ST programs

In this project file, the socket service functions of the Controller are used to perform VERGET /S (Acquire Software Version) for the destination device and to detect whether the processing ends normally or in an error.
The normal end of this project file indicates that the TCP socket communications end normally.
The error end indicates that the TCP socket communications ends in error and a destination device error occurs (judged on the response data from the destination device).

This project file does not use keep-alive or linger functions of the TCP socket options. Use them in your application when necessary.

Additional Information

OMRON has confirmed that normal communications can be performed using this project file under the OMRON evaluation conditions including the test system configuration, version of each product, and product Lot, No. of each device which was used for evaluation.
OMRON does not guarantee the normal operation under the disturbance such as electrical noise and the performance variation of the device.

Additional Information

With Sysmac Studio, a data type + "#" are prefixed to decimal data and a data type + "#" + "16" + "#" are prefixed to hexadecimal data when it is necessary to distinguish between decimal and hexadecimal data. (e.g., INT#1000 decimal -> INT#16#03E8 hexadecimal. For DINT, a data type + "#" are unnecessary.

9.1.1. Communications Data Flow

The following figure shows the data flow from issuing a command with TCP socket communications from the Controller to the destination device to receiving the response data from the destination device. This project file executes a series of processing from the TCP open to the close processing continuously. Receive processing is performed repeatedly when the response data is divided and multiple receive data are sent.

1.	TCP open processing	The Controller issues a TCP open request to the destination device and a TCP connection is established.
	↓	
2.	Command send processing	The send message set with the ST program is sent from the Controller to the destination device.
	↓	
3.	Response receive processing	The response data, which was received by the Controller from the destination device, is stored in specified internal memory.
	↓	
4.	Close processing	The Controller issues a close request to the destination device, and the TCP connection is terminated.

*The response data is not sent after receiving a command or the response data is sent immediately after a connection is established depending on the destination device and command. With this project file, "Send/receive processing required/not required setting" can be set for the "General-purpose Ethernet communications sequence setting function block". If "Send only" is set, the response receive processing is not performed. If "Receive only" is set, the command send processing is not performed.

9.1.2. TCP Socket Communications with Socket Service Instructions

This section outlines TCP socket communications performed by using the TCP socket service function blocks (hereinafter referred to as socket service instructions) and send/receive process of the message.

Additional Information

For details, refer to *Communications Instructions* under *Section 2 Instruction Descriptions* of *NJ-series Instructions Reference Manual* (Cat. No. W502).

●TCP Socket Services with Socket Service Instructions

In this project file, socket communications are performed by using the following 5 types of standard instructions.

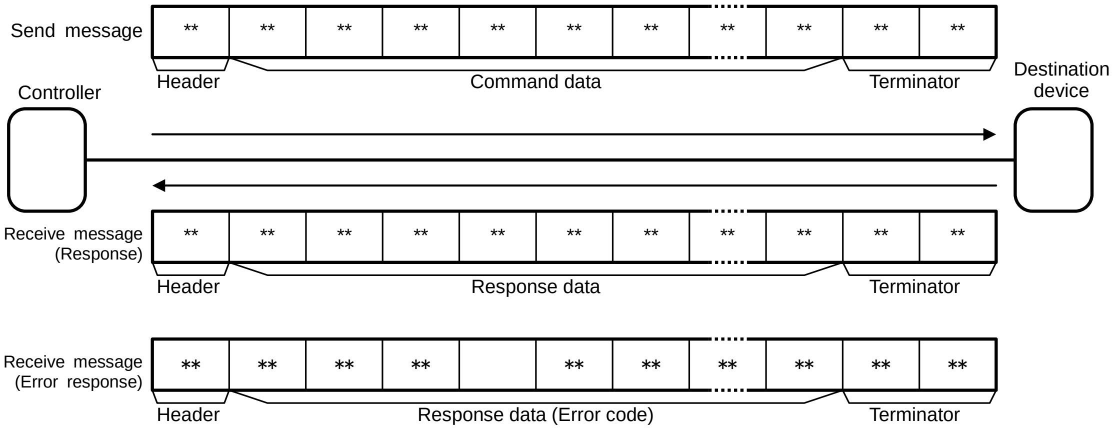
Name	Function blocks	Description
Connect TCP Socket	SktTCPConnect	Connects the TCP port of the destination device using an active open.
TCP Socket Send	SktTCPSend	Sends data from a specified TCP socket.
TCP Socket Receive	SktTCPRcv	Reads data received from a specified TCP socket.
Close TCP/UDP Socket	SktClose	Closes a specified TCP socket.
Read TCP Socket Status	SktGetTCPStatus	Reads the status of a specified TCP socket. In this project file, this instruction is used to check if receive processing is completed during receive processing and to check the closing status during close processing.

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

Variable	Meaning	Description	Data type	Valid range	Default
Socket	Socket	Socket	_sSOCKET	-	-
Handle	Handle	Handle for data communications	UDINT	Depends on data type	-
SrcAdr	Local address	Local address *1	_sSOCKET_ADDRESS	-	-
PortNo	Port number	Port number	UINT	1 to 65535	
IpAdr	IP address	IP address or host name *2	STRING	Depends on data type	
DstAdr	Destination address	Destination address *1	_sSOCKET_ADDRESS	-	-
PortNo	Port number	Port number	UINT	1 to 65535	
IpAdr	IP address	IP address or host name *2	STRING	Depends on data type	

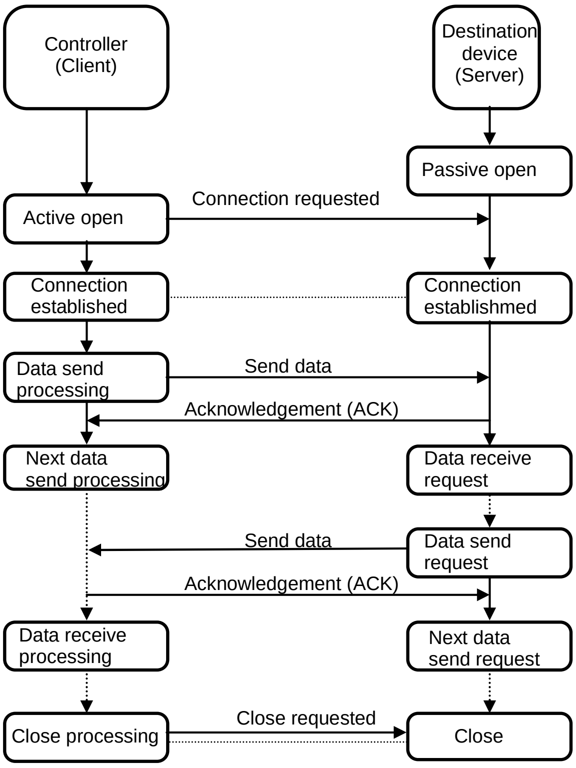
*1: The address indicates an IP address and a port number.
*2: A DNS or Hosts setting is required to use a host name.

●Send/receive message



●Communications sequence

TCP communications are performed between the destination device (server) and Controller (client) in the following procedure.



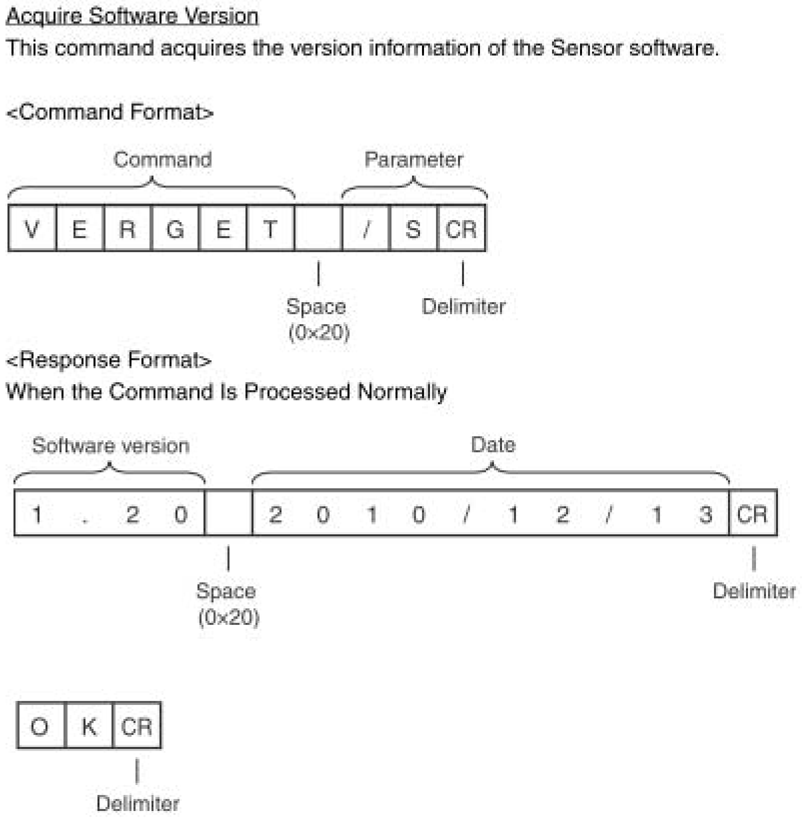
9.2. Destination Device Command

This section explains the destination device command used in this project file.

9.2.1. Overview of the Command

This project file uses VERGET /S (Acquire Software Version) command to perform Ethernet communications with the destination device.

Command	Description
VERGET /S	Acquire software version



Acquire Software Version

This command acquires the version information of the Sensor software.

<Command Format>

V	E	R	G	E	T		/	S	CR

Command
Parameter
Space
(0×20)
Delimiter

<Response Format>

When the Command Is Processed Normally

1	.	2	0		2	0	1	0	/	1	2	/	1	3	CR

Software version
Date
Space
(0×20)
Delimiter

O	K	CR

Delimiter

Additional Information

For details, refer to *Controlling the Sensor from an External Device (Procedure for No-protocol Command/Response Communications)* in *8-2 Outputting/Controlling with Ethernet* in the user's manual for each Code Reader.

9.2.2. Command Settings

This section explains the details on the settings for VERGET /S (Acquire Software Version) command.

●Send data (Command) settings

Set the send data in SendMessageSet_instance function block.

<Specifications of the destination device>

•Data is stored in ASCII code.

Variable	Description (data type)	Set value
Send_Header	Send Header (STRING[5])	“” (None)
Send_Addr	Send address (STRING[5])	”” (None)
Send_Command	Send data (STRING[256])	"VERGET /S"
Send_Check	Addition of send check (STRING[5])	“” (None)
Send_Terminate	Send terminator (STRING[5])	'$R' ([CR]:#16#0D)

Variable	Description (data type)	Data	Description
Send_Data	Send message (STRING[256])	CONCAT(Send_Header, Send_Addr, Send_Command, Send_Check, Send_Terminate)	Used as send data of SktTCPSend instruction (SktTCPSend_instance).

●Receive data (response) that is stored

After a data check is performed on the receive data using the ReceiveCheck_instance function block, the receive data is stored as output receive data.

<Specifications of the destination device>

•Data is stored in ASCII code.

Variable	Description (data type)	Storage area
Recv_Data	Receive data (STRING[256])	Receive buffer
Recv_Buff	Receive data (STRING[256])	Receive data storage area (stores the receive buffer data)

●Send/receive message

Send message

56	45	52	47	45	54	20	2F	53	0D
'V'	'E'	'R'	'G'	'E'	'T'	''	'/'	'S'	[CR]

(Normal operation)

Receive message 1

31	2E	33	31	20	32	30	31	31	2F	30	38	2F	30	31	0D
'1'	'.'	'3'	'1'	' '	'2'	'0'	‘1’	'1 '	'/'	'0'	‘8’	'/'	'0'	‘1’	[CR]

Receive message 2

4F	4B	0D
'O'	'K'	[CR]

(Error operation)

Receive message

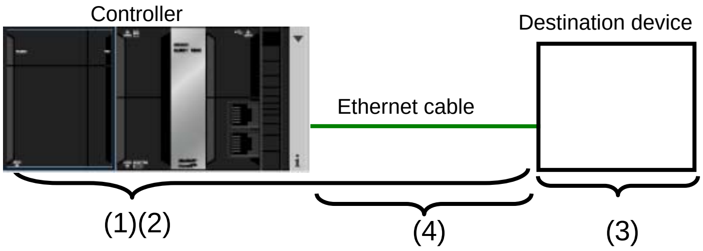
45	52	0D
'E'	'R'	[CR]

9.3. Error Detection Processing

This section explains the error detection processing of this project file.

9.3.1. Error Detection in the Project File

This project file detects and handles errors of the following items (1) to (4). For information on error codes, refer to *9.7.1. Error Code List*.



(1)Communications errors in TCP socket communications using socket service instructions

Errors occurred in a program during TCP socket communications such as Unit error, command format error and parameter error are detected as communications errors. The error is detected with the socket service instruction argument ErrorID.

(2)Timeout errors during communication with the destination device

When open processing, send processing, receive processing, or close processing is not normally performed and cannot be completed within the monitoring time, it is detected as a timeout error. The error is detected with the time monitoring function in the project file. For information on the time monitoring function by using the timer in the project file, refer to *9.3.2. Time Monitoring Function*.

(3)Errors in the destination device (Destination device error)

The destination device error includes a command error, parameter error, and execution failure in the destination device. The error is detected with the response data which is sent from the destination device. With this project file, the destination device error is detected with the error code, which is returned from the destination device when an error occurs. For information on the send/receive messages, refer to *9.2. Destination Device Command*.

(Receive message for error process)

45	52	0D
'E'	'R'	[CR]

(4)TCP connection status errors when ending the processing

With this project file, the close processing is always performed at the end of the whole processing regardless of whether each processing from the open processing to the receive processing ends normally or in an error. The TCP connection status variable *TcpStatus* of the SktGetTCPStatus instruction is used to detect whether the close processing ends normally. When the close processing is operated abnormally, the next open processing may not be performed normally. For information on the corrective actions for TCP connection status errors, refer to *9.7.2 TCP Connection Status Errors and Corrective Actions*.

9.3.2. Time Monitoring Function

This section explains the time monitoring function of this project file.

You can change the monitoring time settings by changing the variables of the ParameterSet function block.

●Time monitoring function using the timer in the project file

To prepare against errors that may prevent the execution of the processing from ending, the timer in this project file is used to abort the processing (timeout). The timeout value for each processing from the open processing to the close processing is 5 seconds (default).

[Time monitoring function using the timer in the project file]

Processing	Monitoring	Variable name	Timeout time (Default)
Open processing	Time from the start of the open processing to the end	TopenTime	After 5 seconds (UINT#500)
Send processing	Time from the start of the send processing to the end	TfsTime	After 5 seconds (UINT#500)
Receive processing	Time from the start of the receive processing to the end *When receive processing is repeated, the timer monitoring timer monitors each receive processing separately.	TfrTime	After 5 seconds (UINT#500)
Close processing	Time from the start of the close processing to the end *The time monitoring timer confirms the normal TCP connection status after the close processing and detects that the processing is completed.	TcloseTime	After 5 seconds (UINT#500)

●Time monitoring function of the Controller (socket service)

The Controller has a time monitor function as a socket service. This function monitors the time taken to receive data that are sent separately. *TrTime*=UINT#3 (300 ms) (default) is stored in the *TimeOut* parameter of the SktTCPRcv socket service instruction when receive processing is performed. For the receive waiting time for the next response after the receive processing ends once, *TrTime* variable is also set for the receive waiting time monitoring timer with this project file. If the next response is not received from the destination device within this time, it is detected that the receive processing ends.

Additional Information

For information on the time monitoring function of the socket service, refer to *Communications Instructions - SktTCPRcv* in *Section 2 Instruction Descriptions* of the *NJ-series Instructions Reference Manual* (Cat. No. W502).

●Resend/time monitoring functions of the Controller (TCP/IP)

When a communication problem occurs, TCP/IP automatically resends the data and monitors the processing time if there is no error in the Controller. If the processing ends in an error, this project file performs the close processing and stops the TCP/IP resend/time monitoring function. If a TCP connection status error occurs during close processing, the TCP/IP resend/time monitoring function of the Controller may be operating. For information on the status and corrective actions, refer to *9.7.2. TCP Connection Error Status and Corrective Actions*.

9.4. Variables

The table below lists the variables used in this project file.

9.4.1. List of Variables

The variables necessary to execute this project file are listed below.

●Input variable

The following table shows the variable used to operate this project file.

Name	Data type	Description
Input_Start	BOOL	This project file is started by turning OFF (FALSE) and then ON (TRUE). After checking the normal end output or error end output, turn ON and then OFF.

●Output variables

The following table lists the variables that contain the execution results of this project file.

Name	Data type	Description
Output_RecvMess	STRING[256]	Stores the receive data (response). (256-byte area 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 or close processing. 16#0000 is stored for a normal end.
Output_SktCmdsErrorID	WORD	Stores each socket service instruction's error code for a communications error or timeout error detected during open processing, send processing or receive processing. 16#0000 is stored for a normal end.
Output_SkTcloseErrorID	WORD	Stores the SktTcpClose instruction's error code for a communications error or timeout error detected during close processing rather than an error detected during open processing, send processing or receive processing. 16#0000 is stored for a normal end.
Output_EtnTcpSta	_eCONNECTION_STATE	Stores the TCP connection status when a communications error or timeout error is detected during close processing. _*CLOSED* is stored for a normal end.
Output_MErrCode	DWORD	Stores the error code for an FCS calculation error or a destination device error detected after the receive processing. 16#00000000 is stored for a normal end.

●Internal variables

The following table lists the variables used only for operations of this project file.

Name	Data type	Description
Local_Status	sStatus (STRUCT)	Program execution status
Busy	BOOL	TRUE while executing this project file. FALSE while not executing this project file.
Done	BOOL	TRUE for a normal end of this project file. FALSE when Input_Start changes to FALSE.
Error	BOOL	TRUE for an error end of this project file. FALSE when Input_Start changes to FALSE.
Local_State	DINT	Status processing number
Local_ErrCode	uErrorFlgs (UNION)	Sets an error code.
Local_ErrCode. WordData	WORD	Expresses an error code in WORD.
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: 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/not required detection error: Error (TRUE)/Normal (FALSE) BoolData[12]: Destination device error: Error (TRUE)/Normal (FALSE) BoolData[6..7],[13..14]: Reserved BoolData[15]: Error
Local_ExecFlgs	sControl (STRUCT)	Socket service instruction execution flag
Send	BOOL	Send processing instruction　Executed (TRUE)/Not executed (FLASE)
Recv	BOOL	Receive processing instruction　Executed (TRUE)/Not executed (FLASE)
Open	BOOL	Open processing instruction　Executed (TRUE)/Not executed (FLASE)
Close	BOOL	Close processing instruction　Executed (TRUE)/Not executed (FLASE)
Status	BOOL	TCP status instruction　Executed (TRUE)/Not executed (FLASE)
Local_SrcDataByte	UINT	Sets the number of send data bytes.

Name	Data type	Description
Local_SrcData	ARRAY[0..2000] OF BYTE	An area that stores the data sent by the SktTCPSend instruction (SktTCPSend_instance). (256-byte area is secured.)
Local_RecvData	ARRAY[0..2000] OF BOOL	Stores the data (response) received by the SktTCPRcv instruction (SktTCPRcv_instance). (256-byte area is secured.)
Local_ReceiveMessage	STRING[256]	Stores the STRING data (response) received by *Local_RecvData*. (256-character area is secured.)
Local_RecvCheckFlg	BOOL	Destination device error detection instruction execution flag Executed (TRUE)/Not executed (FLASE)
Local_InitialSettingOK	BOOL	Initialization processing normal setting flag
Local_TONFlgs	sTimerControl (STRUCT)	Timer enable flag
Tfs	BOOL	Send processing time monitoring timer instruction Enabled (TRUE)/Disabled (FALSE)
Tfr	BOOL	Receive processing time monitoring timer instruction Enabled (TRUE)/Disabled (FALSE)
Topen	BOOL	Open processing time monitoring timer instruction Enabled (TRUE)/Disabled (FALSE)
Tclose	BOOL	Close processing time monitoring timer instruction Enabled (TRUE)/Disabled (FALSE)
Tr	BOOL	Next response receive waiting time monitoring timer instruction Enabled (TRUE)/Disabled (FALSE)
Local_ComType	sControl (STRUCT)	Sets the send/receive processing required/not required setting.
Send	BOOL	Send processing Required (TRUE)/Not required (FALSE). *When send processing is required and receive processing is not required: This program skips receive processing without waiting for receive data during send processing, and shifts to close processing. This is specified when no response data is sent for the sent command.
Recv	BOOL	Receive processing Required (TRUE)/Not required (FALSE). *When send processing is required and receive processing is required: This program waits for the receive data after the send processing. After checking that data is received, this program shifts to the receive processing. This is specified when response data is sent for the sent command.
Error	BOOL	Send/receive processing required/not required setting error flag (Set this flag when a setting error occurs.)

●Variables used to initialize socket service instructions

Name	Data type	Description
NULL_SOCKET	_sSOCKET	Socket service instruction initialization data (Retain/Constant: Enabled) Default value (Handle := 0, SrcAdr := (PortNo := 0, IpAdr := ''), DstAdr := (PortNo := 0, IpAdr := '')) (Used for all socket service instructions.)
NULL_ARRAYOFBYTE_1	ARRAY[0..0] OF BYTE	Send socket service instruction initialization data array (Retain/Constant: Enabled) Default value [0] (Used for SktTCPSend instruction.)
NULL_ARRAYOFBYTE_2	ARRAY[0..0] OF BYTE	Receive socket service instruction initialization data array (Retain/Constant: Enabled) Default value [0] (Used for SktTCPRcv instruction.)

9.4.2. List of Variables Used in Function Block/Function

The internal variables used to execute the function blocks in the program are listed below. The internal variable is called the “instance”. The name of the function block to use is specified as the data type of the variable.

●Instances of user-defined function blocks

Variable name	Data type	Description
ETN_ParameterSet_instance	ParameterSet	Ethernet setting (Destination IP address, etc.) Monitoring time of each processing from the open processing to the close processing
ETN_SendMessageSet_instance	SendMessageSet	Sets the send/receive processing required/not required setting and sets a send message.
ETN_ReceiveCheck_instance	ReceiveCheck	Stores receive data and detects whether the operation ended normally or ended in error.

*For information on the user-defined function blocks, refer to *9.5.3 Detailed Description of Function Blocks*.

●Instances of timers used in the program

Variable name	Data type	Description
Topen_TON_instance	TON	Counts the time taken to perform the open processing.
Tfs_TON_instance	TON	Counts the time taken to perform the send processing.
Tfr_TON_instance	TON	Counts the time taken to perform the receive processing.
Tclose_TON_instance	TON	Counts the time taken to perform the close processing.
Tr_TON_instance	TON	Counts the time waiting for the next response.

9.4.3. List of System Variables

The variable necessary to execute the project file is shown below.

●System variable (External variable)

Name	Data type	Description
_EIP_EtnOnlineSta	BOOL	Communication function status of the Controller: TRUE: Can be used. FALSE: Cannot be used.

Additional Information

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

9.5. Program (ST language)

9.5.1. Functional Components of ST Program

This program is written in the ST language. The functional components are as follows.

Major classification	Minor classification	Description
1. Communications processing	1.1. Starting communications processing 1.2. Clearing the communications processing status flags 1.3 Communications processing in progress status	The communications processing is started.
2. Initialization processing	2.1. Initializing the processing time monitoring timer 2.2. Initializing the socket service instructions 2.3. Initializing the socket service instruction execution flags 2.4. Initializing the processing time monitoring timer enable flags 2.5. Initializing the error code storage areas 2.6. Setting each processing monitoring time and Ethernet related parameters 2.7. Setting the send/receive processing required/not required setting and send data 2.8. Converting send data from a string to a BYTE array 2.9. Initializing the receive data storage areas 2.10. Initialization setting end processing	The Ethernet parameters are set and the error code storage area is initialized. The send/receive required/not required setting, send data and receive data are set.
3. Open processing	3.1. Determining the open processing status and setting the execution flag 3.2. Enabling the open processing time monitoring timer 3.3. Executing the open instruction (TCP.Active open processing)	The TCP open (Active) processing is executed. After starting the communication processing and executing initialization settings, the processing is executed unconditionally.
4. Send processing	4.1. Determining the send processing status and setting the execution flag 4.2. Enabling the send processing time monitoring timer 4.3. Executing the send instruction	The processing is executed when the send processing required/not required setting is set to Required and the open processing ended normally.
5. Receive processing	5.1 Determining the receive processing status and setting the execution flag 5.2 Enabling the receive waiting time monitoring timer 5.3 Enabling the receive processing time monitoring timer 5.4 Executing the receive instruction 5.5 Executing the get TCP status instruction 5.6 Executing the destination device error detection instruction	The processing is executed when the receive processing required/not required setting is set to Required and the send processing ended normally. If multiple receive data arrive, the receive processing is repeated. The receive data is stored and checked.

Major classification	Minor classification	Description
6. Close processing	6.1. Determining the close processing status and setting the execution flag 6.2. Enabling the close processing time monitoring timer 6.3. Executing the close instruction 6.4. Executing the get TCP status instruction	The close processing is executed. The processing is executed in the following cases. •When the receive processing not required setting is set and the sent processing ends normally •When the receive processing ends normally •Immediately after an error end of open processing, send processing or receive processing
7. Processing number error process	7. Processing number error process	The error process is executed when a non-existent processing number is detected.

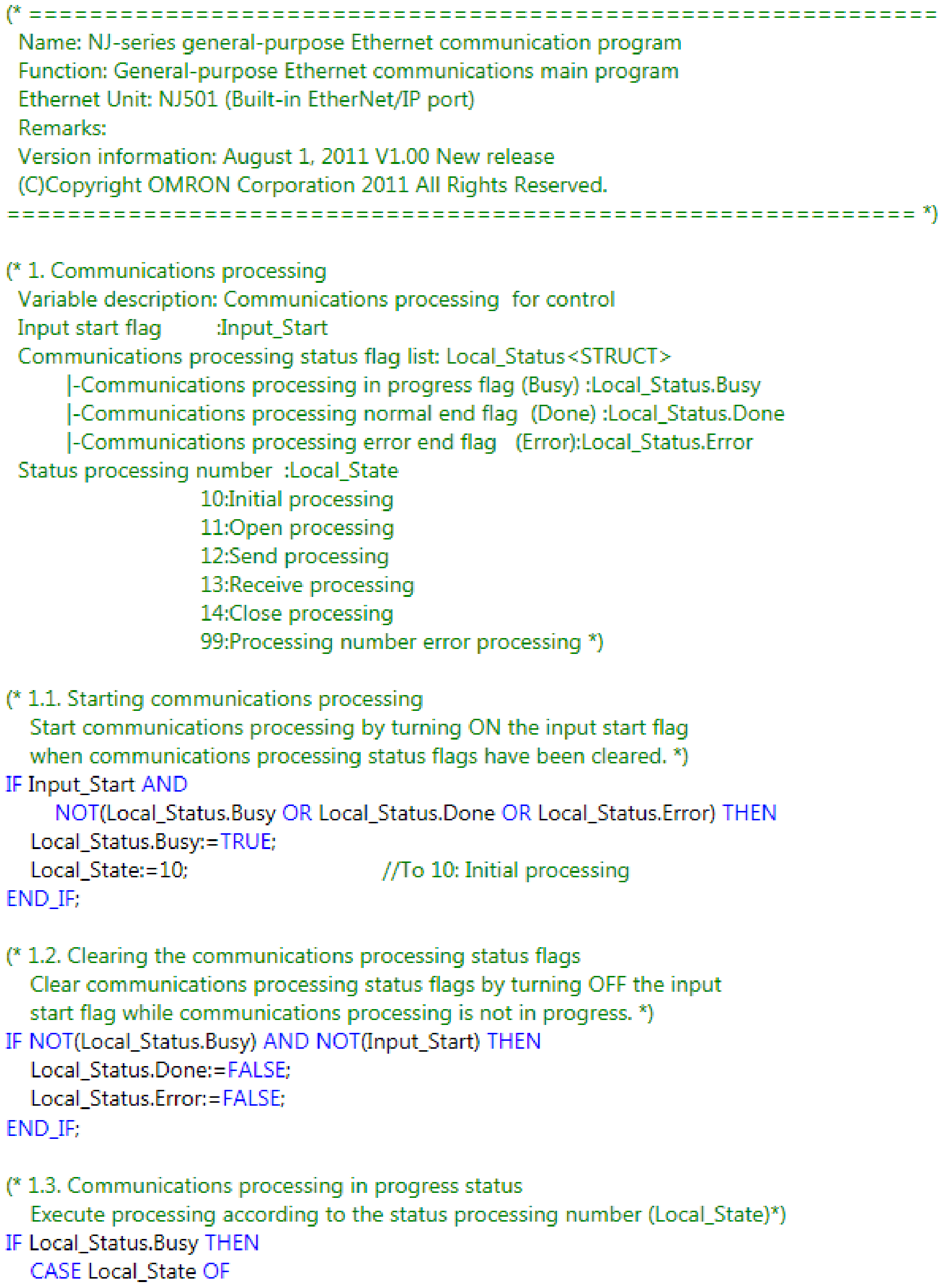
9.5.2. Detailed Description of Main Program

The main program is shown below.

The communications settings, send data (command) setting and receive data (response data) check that must be changed according to the destination device are performed in the function blocks (ETN_ParameterSet, ETN_SendMessageSet, and ETN_ReceiveCheck). For information on how to change these values, refer to *9.5.3 Detailed Description of Function Blocks*.

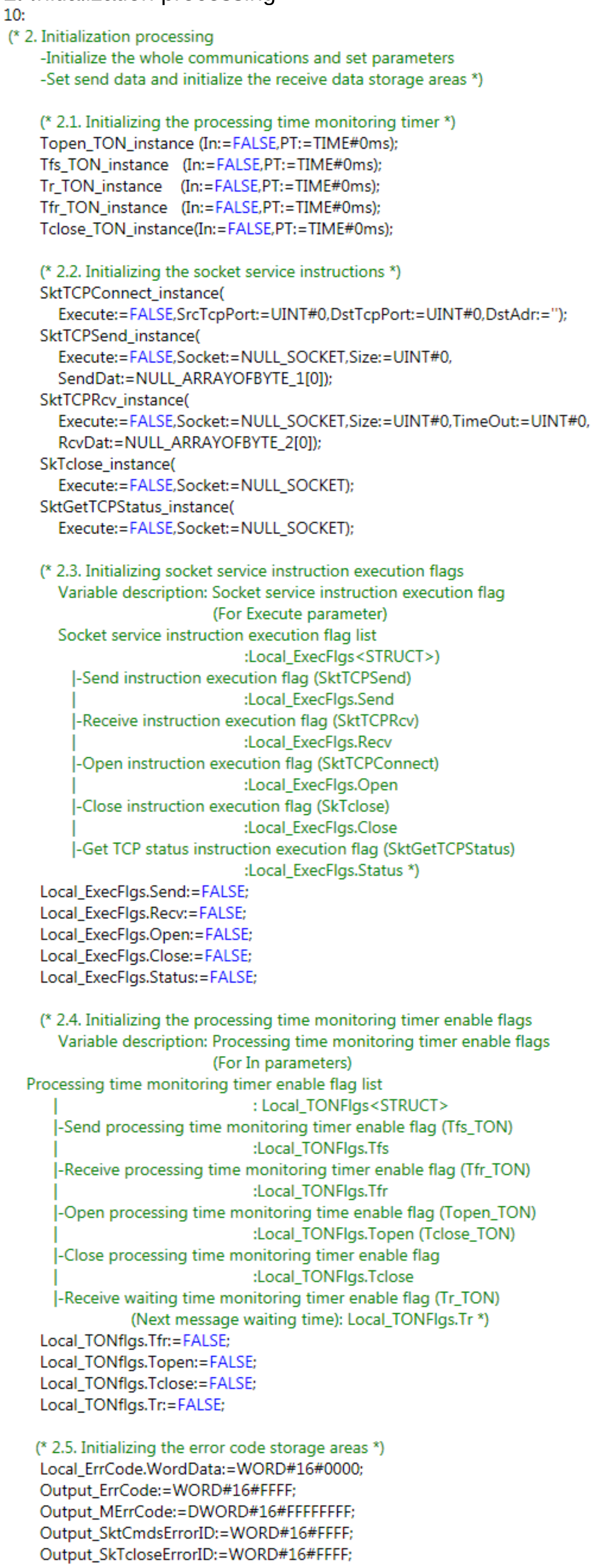
[Main program:Program0]

1. Communications processing

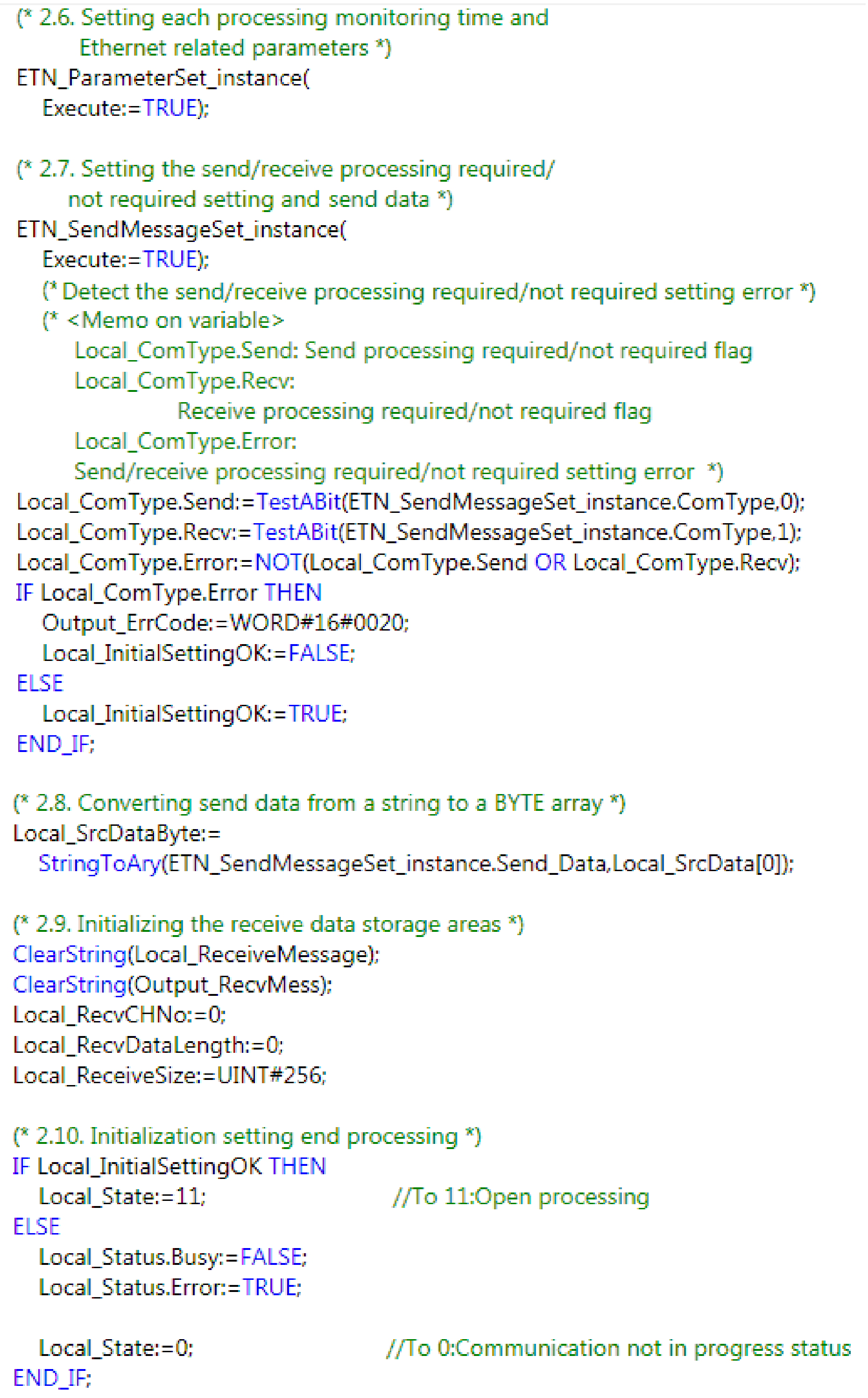


```
(* =============================================================
  Name: NJ-series general-purpose Ethernet communication program
  Function: General-purpose Ethernet communications main program
  Ethernet Unit: NJ501 (Built-in EtherNet/IP port)
  Remarks:
  Version information: August 1, 2011 V1.00 New release
  (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 list: Local_Status<STRUCT>
        |-Communications processing in progress flag (Busy) :Local_Status.Busy
        |-Communications processing normal end flag  (Done) :Local_Status.Done
        |-Communications processing error end flag   (Error):Local_Status.Error
  Status processing number  :Local_State
                        10:Initial processing
                        11:Open processing
                        12:Send processing
                        13:Receive processing
                        14:Close processing
                        99:Processing number error processing *)

(* 1.1. Starting communications processing
   Start communications processing by turning ON the input start flag
   when communications processing status flags have been 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;                     //To 10: Initial processing
END_IF;

(* 1.2. Clearing the communications processing status flags
   Clear communications processing status flags by turning OFF the input
   start flag while communications processing is not in progress. *)
IF NOT(Local_Status.Busy) AND NOT(Input_Start) THEN
   Local_Status.Done:=FALSE;
   Local_Status.Error:=FALSE;
END_IF;

(* 1.3. Communications processing in progress status
   Execute processing according to the status processing number (Local_State)*)
IF Local_Status.Busy THEN
   CASE Local_State OF
```

2. Initialization processing

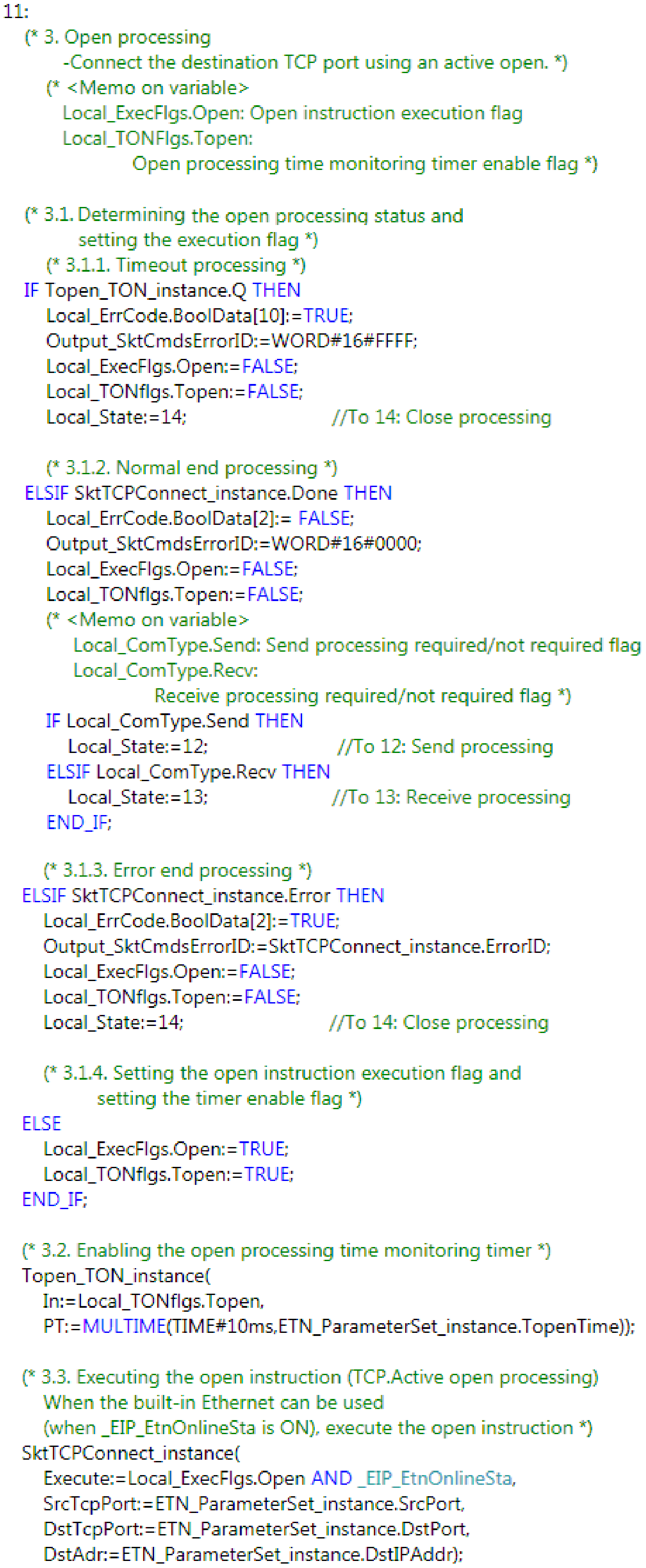


```
10:
(* 2. Initialization processing
    -Initialize the whole communications and set parameters
    -Set send data and initialize the receive data storage areas *)

    (* 2.1. Initializing the processing time monitoring timer *)
    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. Initializing the socket service instructions *)
    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. Initializing socket service instruction execution flags
      Variable description: Socket service instruction execution flag
                                (For Execute parameter)
      Socket service instruction execution flag list
                                    :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
        |-Get TCP 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. Initializing the processing time monitoring timer enable flags
      Variable description: Processing time monitoring timer enable flags
                                (For In parameters)
  Processing time monitoring timer enable flag list
      |                            : Local_TONFlgs<STRUCT>
      |-Send processing time monitoring timer enable flag (Tfs_TON)
      |                            :Local_TONFlgs.Tfs
      |-Receive processing time monitoring timer enable flag (Tfr_TON)
      |                            :Local_TONFlgs.Tfr
      |-Open processing time monitoring time enable flag (Topen_TON)
      |                            :Local_TONFlgs.Topen (Tclose_TON)
      |-Close processing time monitoring timer enable flag
      |                            :Local_TONFlgs.Tclose
      |-Receive waiting time monitoring timer enable flag (Tr_TON)
                     (Next message waiting time): Local_TONFlgs.Tr *)
    Local_TONflgs.Tfr:=FALSE;
    Local_TONflgs.Topen:=FALSE;
    Local_TONflgs.Tclose:=FALSE;
    Local_TONflgs.Tr:=FALSE;

   (* 2.5. Initializing the error code storage areas *)
    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;
```



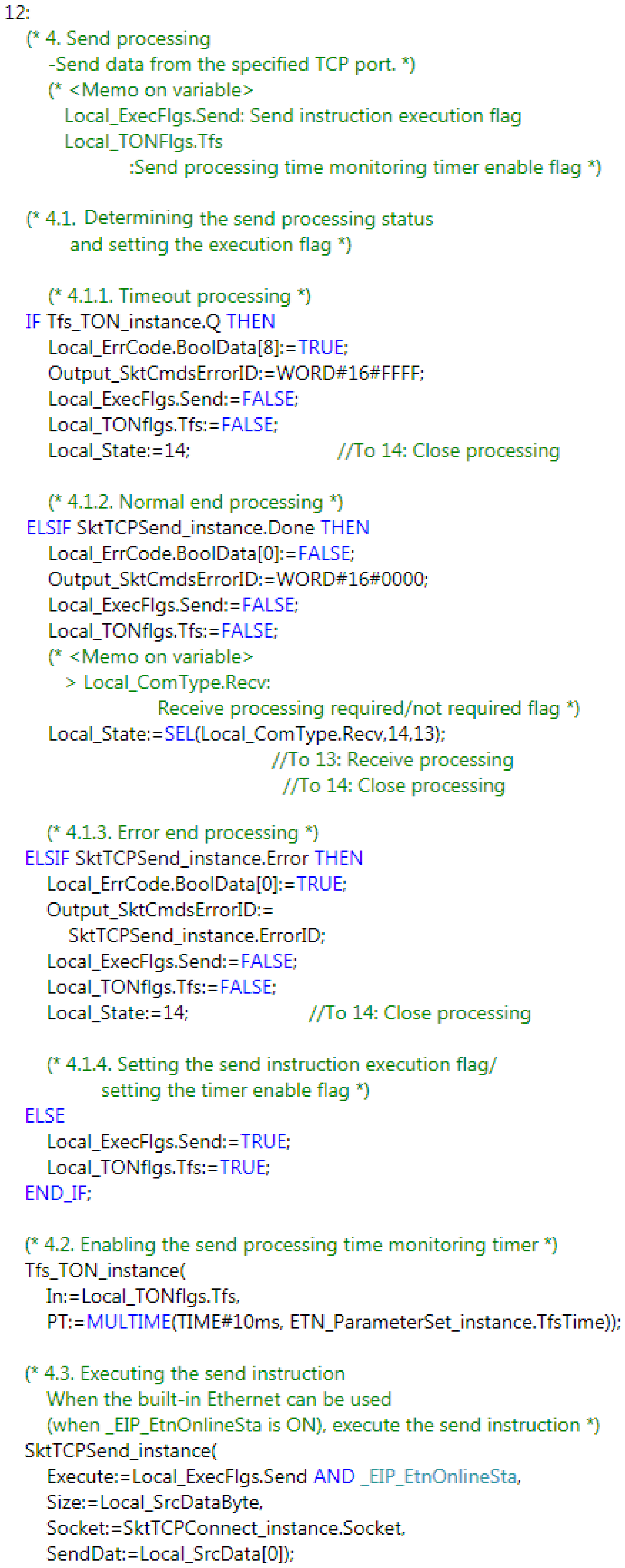
```
(* 2.6. Setting each processing monitoring time and
        Ethernet related parameters *)
ETN_ParameterSet_instance(
    Execute:=TRUE);

(* 2.7. Setting the send/receive processing required/
        not required setting and send data *)
ETN_SendMessageSet_instance(
    Execute:=TRUE);
    (* Detect the send/receive processing required/not required setting error *)
    (* <Memo on variable>
         Local_ComType.Send: Send processing required/not required flag
         Local_ComType.Recv:
                         Receive processing required/not required flag
         Local_ComType.Error:
         Send/receive processing required/not 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. Converting send data from a string to a BYTE array *)
Local_SrcDataByte:=
    StringToAry(ETN_SendMessageSet_instance.Send_Data,Local_SrcData[0]);

(* 2.9. Initializing the receive data storage areas *)
ClearString(Local_ReceiveMessage);
ClearString(Output_RecvMess);
Local_RecvCHNo:=0;
Local_RecvDataLength:=0;
Local_ReceiveSize:=UINT#256;

(* 2.10. Initialization setting end processing *)
IF Local_InitialSettingOK THEN
    Local_State:=11;                     //To 11:Open processing
ELSE
    Local_Status.Busy:=FALSE;
    Local_Status.Error:=TRUE;

    Local_State:=0;                      //To 0:Communication not in progress status
END_IF;
```

3. Open processing



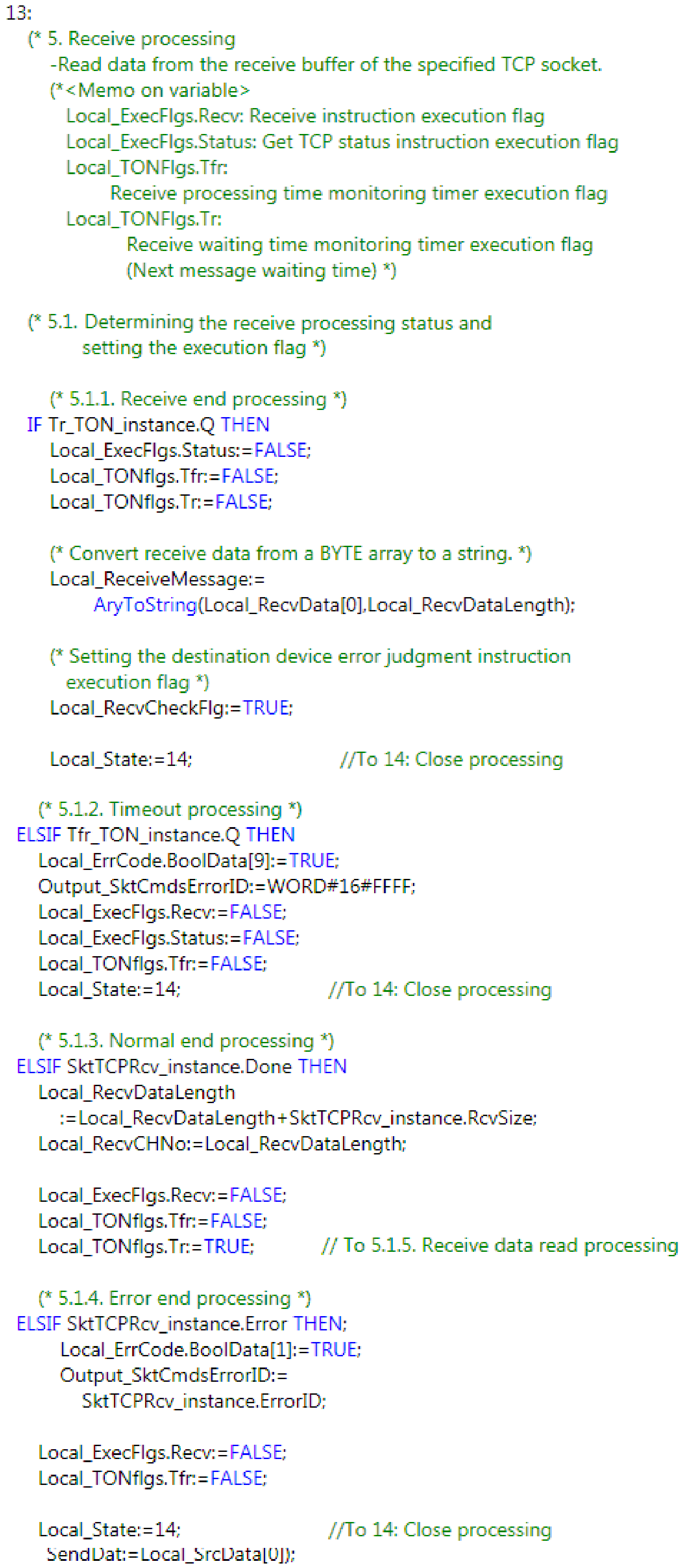
```
11:
   (* 3. Open processing
         -Connect the destination TCP port using an active open. *)
      (* <Memo on variable>
         Local_ExecFlgs.Open: Open instruction execution flag
         Local_TONFlgs.Topen:
                   Open processing time monitoring timer enable flag *)

   (* 3.1. Determining the open processing status and
           setting the execution flag *)
      (* 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;                  //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;
      (* <Memo on variable>
          Local_ComType.Send: Send processing required/not required flag
          Local_ComType.Recv:
                   Receive processing required/not required flag *)
      IF Local_ComType.Send THEN
         Local_State:=12;               //To 12: Send processing
      ELSIF Local_ComType.Recv THEN
         Local_State:=13;               //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;                  //To 14: Close processing

      (* 3.1.4. Setting the open instruction execution flag and
              setting the timer enable flag *)
   ELSE
      Local_ExecFlgs.Open:=TRUE;
      Local_TONflgs.Topen:=TRUE;
   END_IF;

   (* 3.2. Enabling the open processing time monitoring timer *)
   Topen_TON_instance(
      In:=Local_TONflgs.Topen,
      PT:=MULTIME(TIME#10ms,ETN_ParameterSet_instance.TopenTime));

   (* 3.3. Executing the open instruction (TCP.Active open processing)
      When the built-in Ethernet can be used
      (when _EIP_EtnOnlineSta is ON), execute the open instruction *)
   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

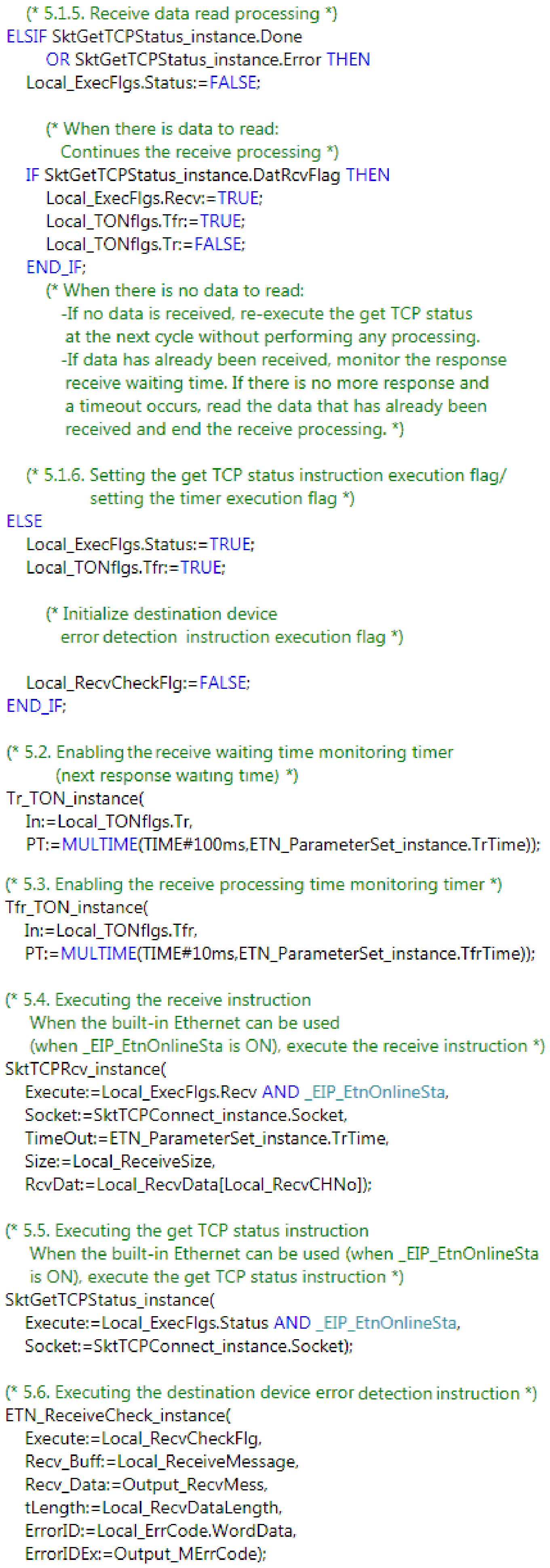


```
12:
(* 4. Send processing
   -Send data from the specified TCP port. *)
   (* <Memo on variable>
     Local_ExecFlgs.Send: Send instruction execution flag
     Local_TONFlgs.Tfs
             :Send processing time monitoring timer enable flag *)

(* 4.1. Determining the send processing status
      and setting the execution flag *)

   (* 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;                //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;
   (* <Memo on variable>
     > Local_ComType.Recv:
                 Receive processing required/not required flag *)
   Local_State:=SEL(Local_ComType.Recv,14,13);
                              //To 13: Receive processing
                               //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;              //To 14: Close processing

   (* 4.1.4. Setting the send instruction execution flag/
          setting the timer enable flag *)
ELSE
   Local_ExecFlgs.Send:=TRUE;
   Local_TONflgs.Tfs:=TRUE;
END_IF;

(* 4.2. Enabling the send processing time monitoring timer *)
Tfs_TON_instance(
   In:=Local_TONflgs.Tfs,
   PT:=MULTIME(TIME#10ms, ETN_ParameterSet_instance.TfsTime));

(* 4.3. Executing the send instruction
   When the built-in Ethernet can be used
   (when _EIP_EtnOnlineSta is ON), execute the send instruction *)
SktTCPSend_instance(
   Execute:=Local_ExecFlgs.Send AND _EIP_EtnOnlineSta,
   Size:=Local_SrcDataByte,
   Socket:=SktTCPConnect_instance.Socket,
   SendDat:=Local_SrcData[0]);
```

5. Receive processing

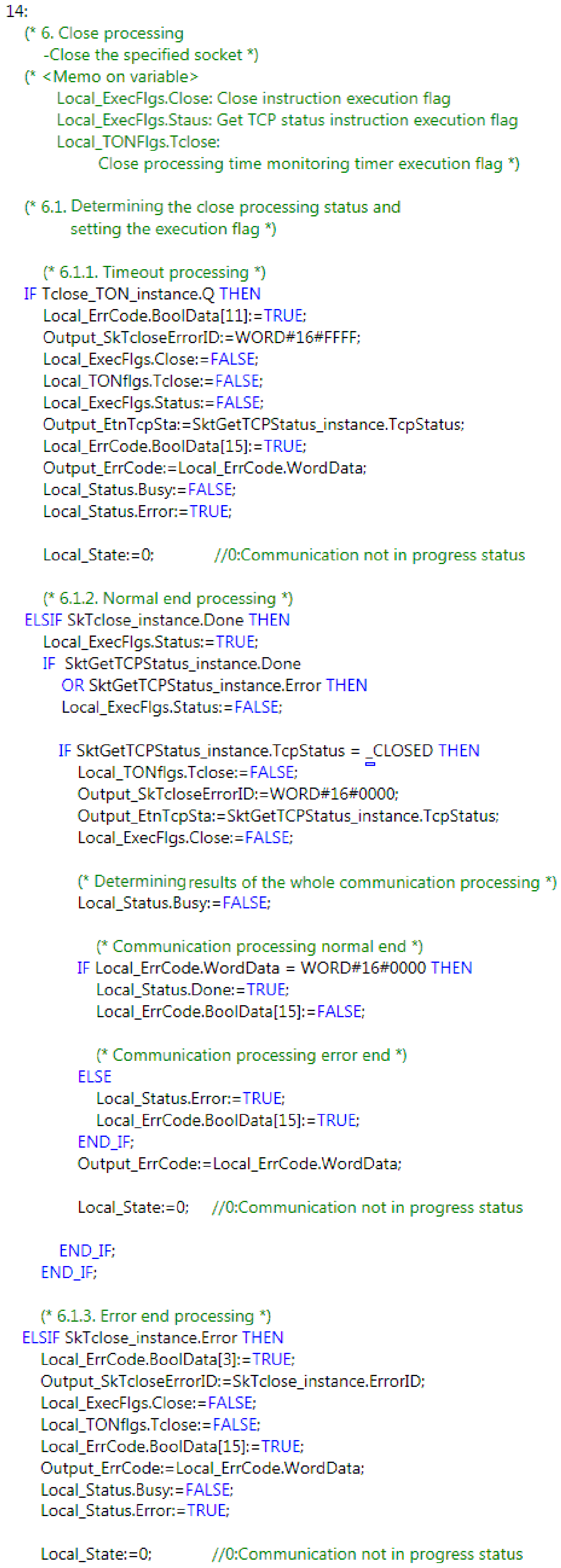


```
13:
  (* 5. Receive processing
     -Read data from the receive buffer of the specified TCP socket.
     (*<Memo on variable>
       Local_ExecFlgs.Recv: Receive instruction execution flag
       Local_ExecFlgs.Status: Get TCP status instruction execution flag
       Local_TONFlgs.Tfr:
             Receive processing time monitoring timer execution flag
       Local_TONFlgs.Tr:
               Receive waiting time monitoring timer execution flag
               (Next message waiting time) *)

  (* 5.1. Determining the receive processing status and
         setting the execution flag *)

     (* 5.1.1. Receive end processing *)
  IF Tr_TON_instance.Q THEN
     Local_ExecFlgs.Status:=FALSE;
     Local_TONflgs.Tfr:=FALSE;
     Local_TONflgs.Tr:=FALSE;

     (* Convert receive data from a BYTE array to a string. *)
     Local_ReceiveMessage:=
           AryToString(Local_RecvData[0],Local_RecvDataLength);

     (* Setting the destination device error judgment instruction
       execution flag *)
     Local_RecvCheckFlg:=TRUE;

     Local_State:=14;                  //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;                  //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;         // 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;                  //To 14: Close processing
     SendDat:=Local_SrcData[0]);
```

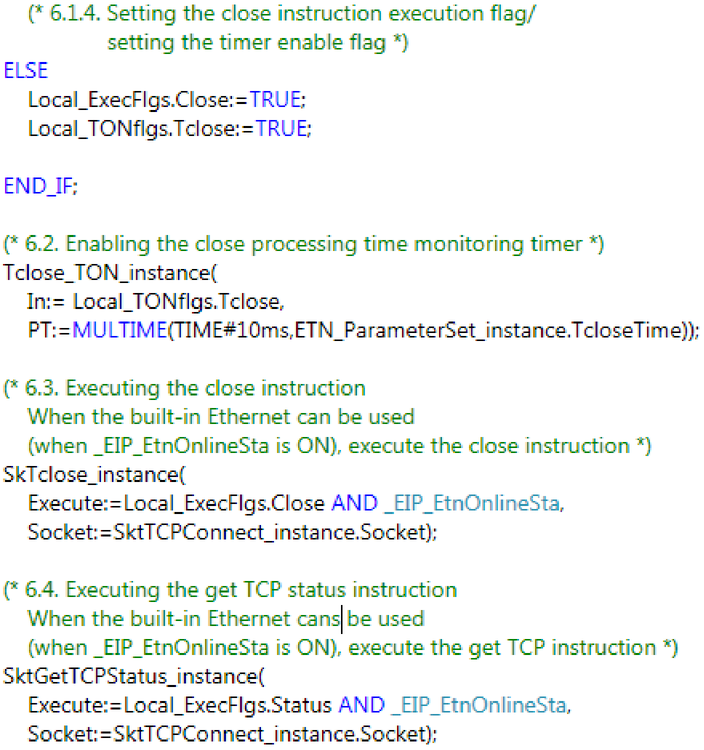


```
    (* 5.1.5. Receive data read processing *)
ELSIF SktGetTCPStatus_instance.Done
        OR SktGetTCPStatus_instance.Error THEN
    Local_ExecFlgs.Status:=FALSE;

        (* When there is data to read:
           Continues the receive processing *)
    IF SktGetTCPStatus_instance.DatRcvFlag THEN
        Local_ExecFlgs.Recv:=TRUE;
        Local_TONflgs.Tfr:=TRUE;
        Local_TONflgs.Tr:=FALSE;
    END_IF;
        (* When there is no data to read:
           -If no data is received, re-execute the get TCP status
            at the next cycle without performing any processing.
           -If data has already been received, monitor the response
            receive waiting time. If there is no more response and
            a timeout occurs, read the data that has already been
            received and end the receive processing. *)

    (* 5.1.6. Setting the get TCP status instruction execution flag/
                 setting the timer execution flag *)
ELSE
    Local_ExecFlgs.Status:=TRUE;
    Local_TONflgs.Tfr:=TRUE;

        (* Initialize destination device
           error detection  instruction execution flag *)

    Local_RecvCheckFlg:=FALSE;
END_IF;

(* 5.2. Enabling the receive waiting time monitoring timer
          (next response waiting time) *)
Tr_TON_instance(
    In:=Local_TONflgs.Tr,
    PT:=MULTIME(TIME#100ms,ETN_ParameterSet_instance.TrTime));

(* 5.3. Enabling the receive processing time monitoring timer *)
Tfr_TON_instance(
    In:=Local_TONflgs.Tfr,
    PT:=MULTIME(TIME#10ms,ETN_ParameterSet_instance.TfrTime));

(* 5.4. Executing the receive instruction
     When the built-in Ethernet can be used
     (when _EIP_EtnOnlineSta is ON), execute the receive instruction *)
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. Executing the get TCP status instruction
     When the built-in Ethernet can be used (when _EIP_EtnOnlineSta
     is ON), execute the get TCP status instruction *)
SktGetTCPStatus_instance(
    Execute:=Local_ExecFlgs.Status AND _EIP_EtnOnlineSta,
    Socket:=SktTCPConnect_instance.Socket);

(* 5.6. Executing the destination device error detection instruction *)
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

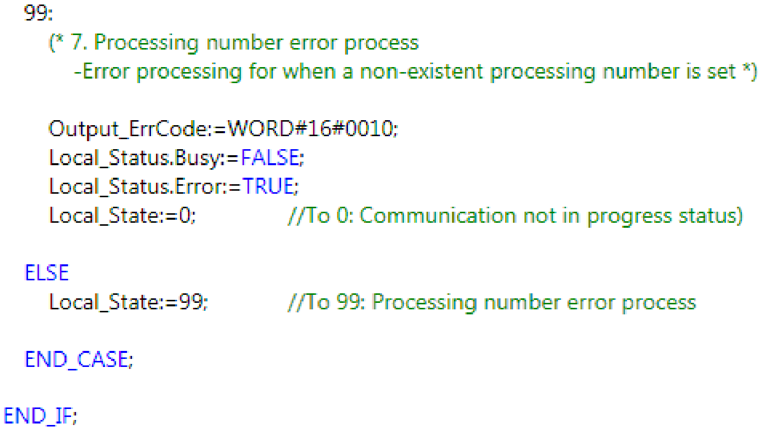


```
14:
(* 6. Close processing
  -Close the specified socket *)
(* <Memo on variable>
    Local_ExecFlgs.Close: Close instruction execution flag
    Local_ExecFlgs.Staus: Get TCP status instruction execution flag
    Local_TONFlgs.Tclose:
        Close processing time monitoring timer execution flag *)

(* 6.1. Determining the close processing status and
     setting the execution flag *)

  (* 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;          //0:Communication not in progress status

  (* 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;

      (* Determining results of the whole communication processing *)
      Local_Status.Busy:=FALSE;

        (* Communication processing normal end *)
      IF Local_ErrCode.WordData = WORD#16#0000 THEN
        Local_Status.Done:=TRUE;
        Local_ErrCode.BoolData[15]:=FALSE;

        (* Communication processing error end *)
      ELSE
        Local_Status.Error:=TRUE;
        Local_ErrCode.BoolData[15]:=TRUE;
      END_IF;
      Output_ErrCode:=Local_ErrCode.WordData;

      Local_State:=0;    //0:Communication not in progress status

    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;          //0:Communication not in progress status
```



```
    (* 6.1.4. Setting the close instruction execution flag/
              setting the timer enable flag *)
ELSE
    Local_ExecFlgs.Close:=TRUE;
    Local_TONflgs.Tclose:=TRUE;

END_IF;

(* 6.2. Enabling the close processing time monitoring timer *)
Tclose_TON_instance(
    In:= Local_TONflgs.Tclose,
    PT:=MULTIME(TIME#10ms,ETN_ParameterSet_instance.TcloseTime));

(* 6.3. Executing the close instruction
    When the built-in Ethernet can be used
    (when _EIP_EtnOnlineSta is ON), execute the close instruction *)
SkTclose_instance(
    Execute:=Local_ExecFlgs.Close AND _EIP_EtnOnlineSta,
    Socket:=SktTCPConnect_instance.Socket);

(* 6.4. Executing the get TCP status instruction
    When the built-in Ethernet cans be used
    (when _EIP_EtnOnlineSta is ON), execute the get TCP instruction *)
SktGetTCPStatus_instance(
    Execute:=Local_ExecFlgs.Status AND _EIP_EtnOnlineSta,
    Socket:=SktTCPConnect_instance.Socket);
```

7. Processing number error process



```
    99:
        (* 7. Processing number error process
            -Error processing for when a non-existent processing number is set *)

        Output_ErrCode:=WORD#16#0010;
        Local_Status.Busy:=FALSE;
        Local_Status.Error:=TRUE;
        Local_State:=0;            //To 0: Communication not in progress status)

    ELSE
        Local_State:=99;           //To 99: Processing number error process

    END_CASE;

END_IF;
```

9.5.3. Detailed Description of Function Blocks

The function blocks used in this project file are shown below.

Data that need to be changed depending on the destination device are set in the red frames on the function blocks below.

●Description of ParameterSet function block

Instruction	Meaning	FB/FUN	Graphic expression	ST expression
ParameterSet	General-purpose Ethernet Communications Parameter setting	FB	None	ETN_ParameterSet_instance (Execute, TfsTime, TrTime, TfrTime, , TopenTime, TcloseTime, SrcPort, DstIPAddr, DstPort);

•In-out variable table (arguments)

•Input

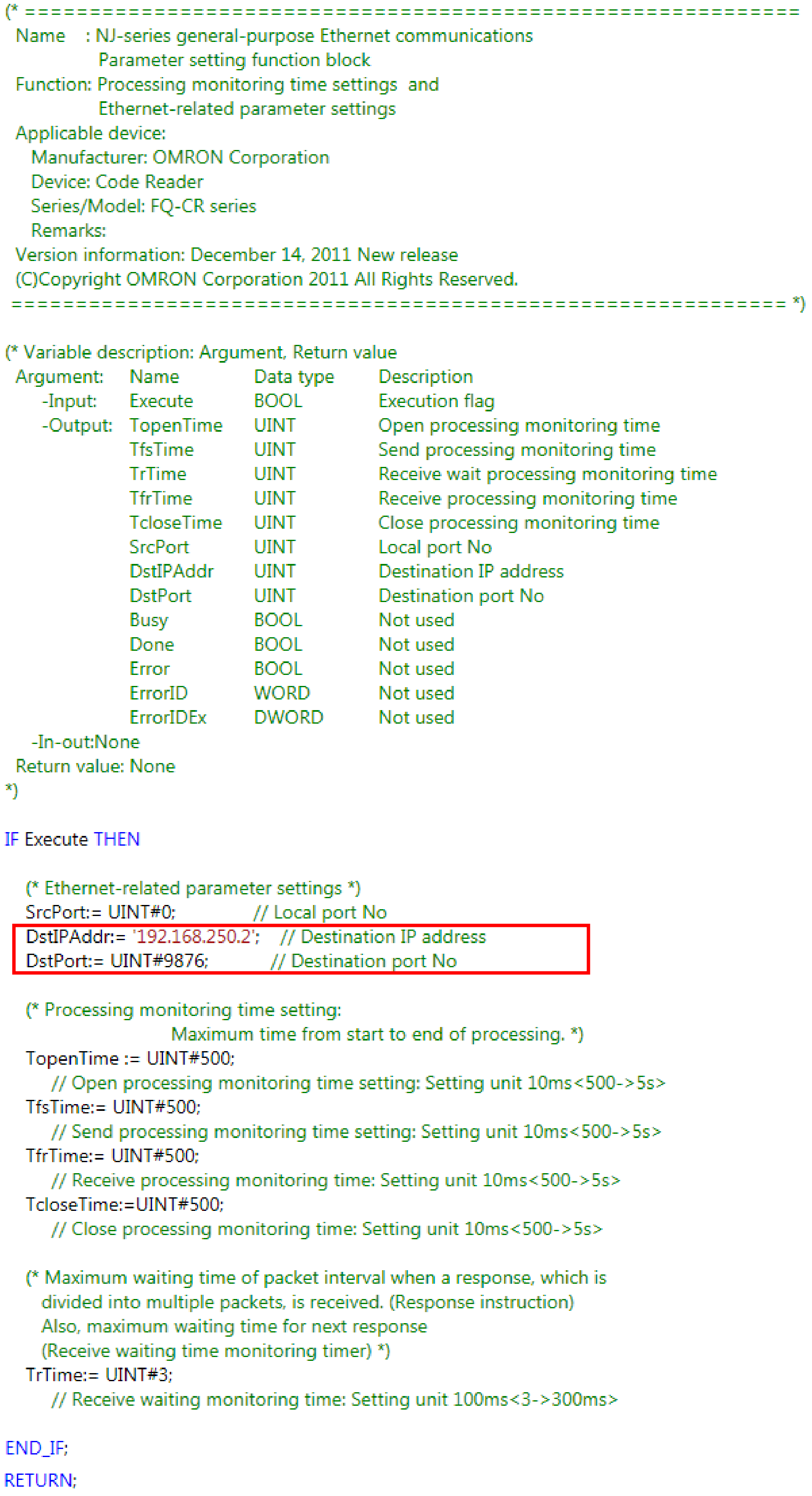
Name	Data type	Meaning	Description	Valid range	Unit	Default
Execute	BOOL	Execute	The function block is executed when this parameter changes from OFF (FALSE) to ON (TRUE). (Always: TRUE)	Depends on data type	-	-

•Output

Name	Data type	Meaning	Description	Valid range	Unit	Default
TopenTime	UINT	Open monitoring time	Sets the monitoring time of the open processing in increments of 10 ms.	Depends on data type	-	-
TfsTime	UINT	Send monitoring time	Sets the monitoring time of the send processing in increments of 10 ms.	Depends on data type	-	-
TrTime	UINT	Receive wait monitoring time	Sets the waiting time for the receive data in increments of 100 ms.	Depends on data type	-	-
TfrTime	UINT	Receive processing time	Sets the monitoring time of the receive processing in increments of 10 ms.	Depends on data type	-	-
TcloseTime	UINT	Close monitoring time	Sets the monitoring time of the close processing in increments of 10 ms.	Depends on data type	-	-
SrcPort	UINT	Local port number	Sets the local port.	Depends on data type	-	-
DstIPAddr	STRING [256]	Destination IP address	Sets the destination IP address.	Depends on data type	-	-
DstPort	UINT	Destination port number	Sets the destination port number.	Depends on the destination device	-	-
Busy	BOOL	Executing	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 general-purpose Ethernet communications
              Parameter setting function block
  Function: Processing monitoring time settings  and
              Ethernet-related parameter settings
  Applicable device:
    Manufacturer: OMRON Corporation
    Device: Code Reader
    Series/Model: FQ-CR series
    Remarks:
  Version information: December 14, 2011 New release
  (C)Copyright OMRON Corporation 2011 All Rights Reserved.
 ============================================================ *)

(* Variable description: Argument, Return value
  Argument:    Name          Data type     Description
      -Input:  Execute       BOOL          Execution flag
      -Output: TopenTime     UINT          Open processing monitoring time
               TfsTime       UINT          Send processing monitoring time
               TrTime        UINT          Receive wait processing monitoring time
               TfrTime       UINT          Receive processing monitoring time
               TcloseTime    UINT          Close processing monitoring time
               SrcPort       UINT          Local port No
               DstIPAddr     UINT          Destination IP address
               DstPort       UINT          Destination port No
               Busy          BOOL          Not used
               Done          BOOL          Not used
               Error         BOOL          Not used
               ErrorID       WORD          Not used
               ErrorIDEx     DWORD         Not used
    -In-out:None
  Return value: None
*)

IF Execute THEN

   (* Ethernet-related parameter settings *)
   SrcPort:= UINT#0;              // Local port No
   DstIPAddr:= '192.168.250.2';    // Destination IP address
   DstPort:= UINT#9876;            // Destination port No

   (* 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;
      // Send 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>

   (* Maximum waiting time of packet interval when a response, which is
     divided into multiple packets, is received. (Response instruction)
     Also, maximum waiting time for next response
     (Receive waiting time monitoring timer) *)
   TrTime:= UINT#3;
      // Receive waiting monitoring time: Setting unit 100ms<3->300ms>

END_IF;
RETURN;
```

●Description of SendMessageSet function block

Instruction	Meaning	FB/FUN	Graphic expression	ST expression
SendMessageSet	General-purpose Ethernet communications sequence setting	FB	None	ETN_SendMessageSet_instance (Execute, Send_Data, ComType);

•In-out variable table (arguments)

•Input

Name	Data type	Meaning	Description	Valid range	Unit	Default
Execute	BOOL	Execute	The function block is executed when this parameter changes from OFF (FALSE) to ON (TRUE). (Always: TRUE)	Depends on data type	-	-

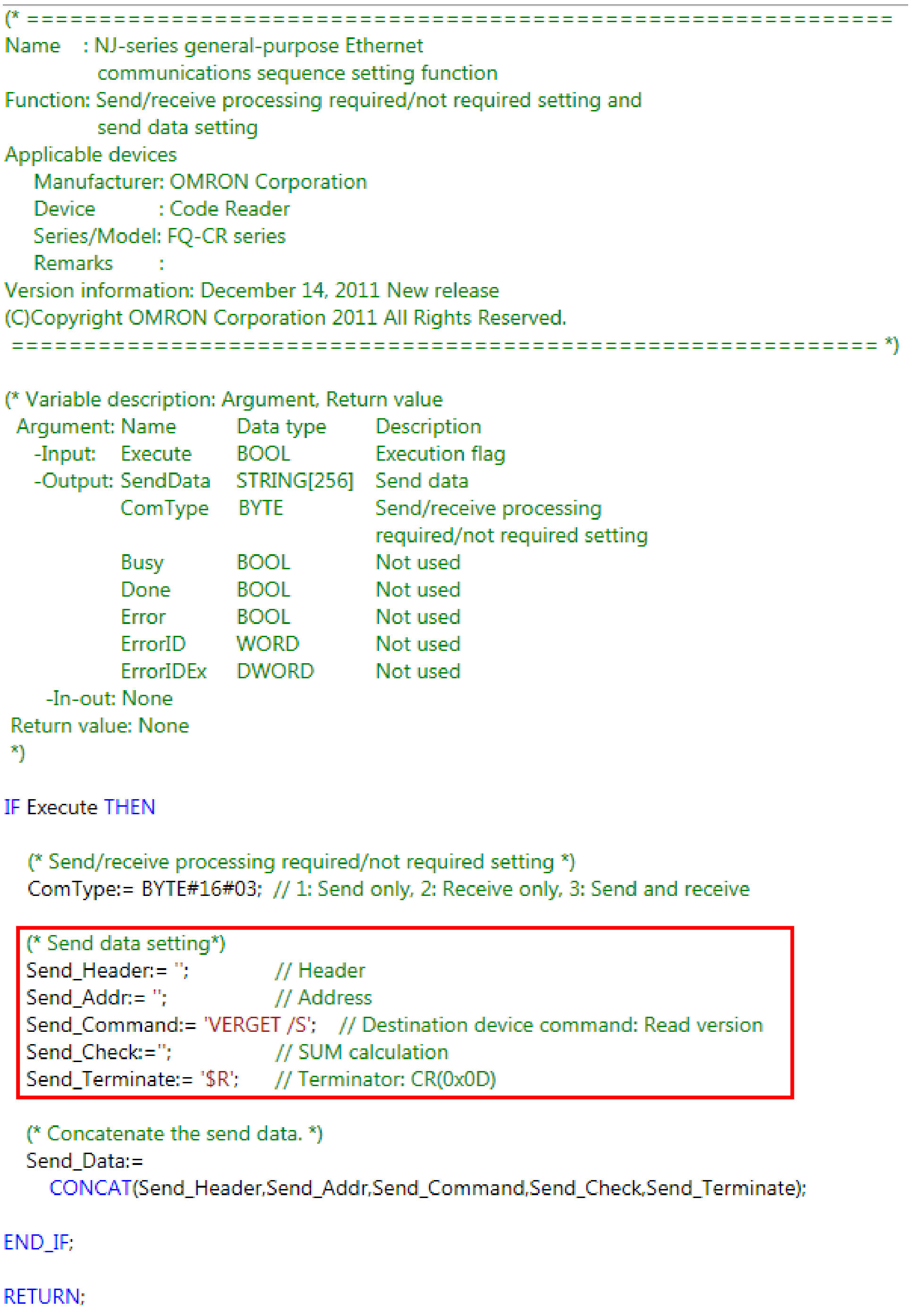
•Output

Name	Data type	Meaning	Description	Valid range	Unit	Default
Send_Data	STRING[256]	Send data	Sets a command that is sent to the destination device.	Depends on data type	-	-
ComType	BYTE	Send/receive type	Sets whether send/receive processing are required. 1:Send only, 2: Receive only, 3: Send and receive	1 to 3	-	-
Busy	BOOL	Executing	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

Name	Data type	Meaning	Description	Valid range	Unit	Default
Send_Header	STRING[5]	Send header	Header of send message	Depends on data type	-	-
Send_Addr	STRING[5]	Destination device address	Destination device address	Depends on data type	-	-
Send_Command	STRING[256]	Send data	Command sent to the destination device	Depends on data type	-	-
Send_Check	STRING[5]	Send check code	Check code of the send message	Depends on data type	-	-
Send_Terminate	STRING[5]	Send terminator	Send message terminator	Depends on data type	-	-

- Program



```
(* ================================================================
Name    : NJ-series general-purpose Ethernet
          communications sequence setting function
Function: Send/receive processing required/not required setting and
          send data setting
Applicable devices
   Manufacturer: OMRON Corporation
   Device        : Code Reader
   Series/Model: FQ-CR series
   Remarks       :
Version information: December 14, 2011 New release
(C)Copyright OMRON Corporation 2011 All Rights Reserved.
 ================================================================ *)

(* Variable description: Argument, Return value
 Argument: Name         Data type      Description
   -Input:   Execute      BOOL           Execution flag
   -Output: SendData     STRING[256]    Send data
             ComType      BYTE           Send/receive processing
                                         required/not required setting
             Busy         BOOL           Not used
             Done         BOOL           Not used
             Error        BOOL           Not used
             ErrorID      WORD           Not used
             ErrorIDEx    DWORD          Not used
    -In-out: None
 Return value: None
 *)

IF Execute THEN

  (* Send/receive processing required/not required setting *)
  ComType:= BYTE#16#03;  // 1: Send only, 2: Receive only, 3: Send and receive

  (* Send data setting*)
  Send_Header:= '';            // Header
  Send_Addr:= '';              // Address
  Send_Command:= 'VERGET /S';    // Destination device command: Read version
  Send_Check:='';              // SUM calculation
  Send_Terminate:= '$R';       // Terminator: CR(0x0D)

  (* Concatenate the send data. *)
  Send_Data:=
     CONCAT(Send_Header,Send_Addr,Send_Command,Send_Check,Send_Terminate);

END_IF;

RETURN;
```

●Description of ReceiveCheck function block

Instruction	Meaning	FB/FUN	Graphic expression	ST expression
ReceiveCheck	General-purpose Ethernet Communications Receive processing	FB	None	ETN_ReceiveCheck_instance (Execute, Recv_Data, Recv_Buff, Error, ErrorID, ErrorIDEx);

•In-out variable table (arguments)

•Input

Name	Data type	Meaning	Description	Valid range	Unit	Default
Execute	BOOL	Execute	The function block is executed when this parameter 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

Name	Data type	Meaning	Description	Valid range	Unit	Default
Recv_Data	STRING[256]	Receive data	Receive data storage result	Depends on data type	-	-
Recv_Buff	STRING[256]	Receive buffer	Receive buffer data	Depends on data type	-	-
ErrorID	WORD	Error information	Error code: Destination device error=16#1000 FCS error=16#2000	-	-	-
ErrorIDEx	DWORD	Error information	Error code: FCS receive result/destination device error code	-	-	-

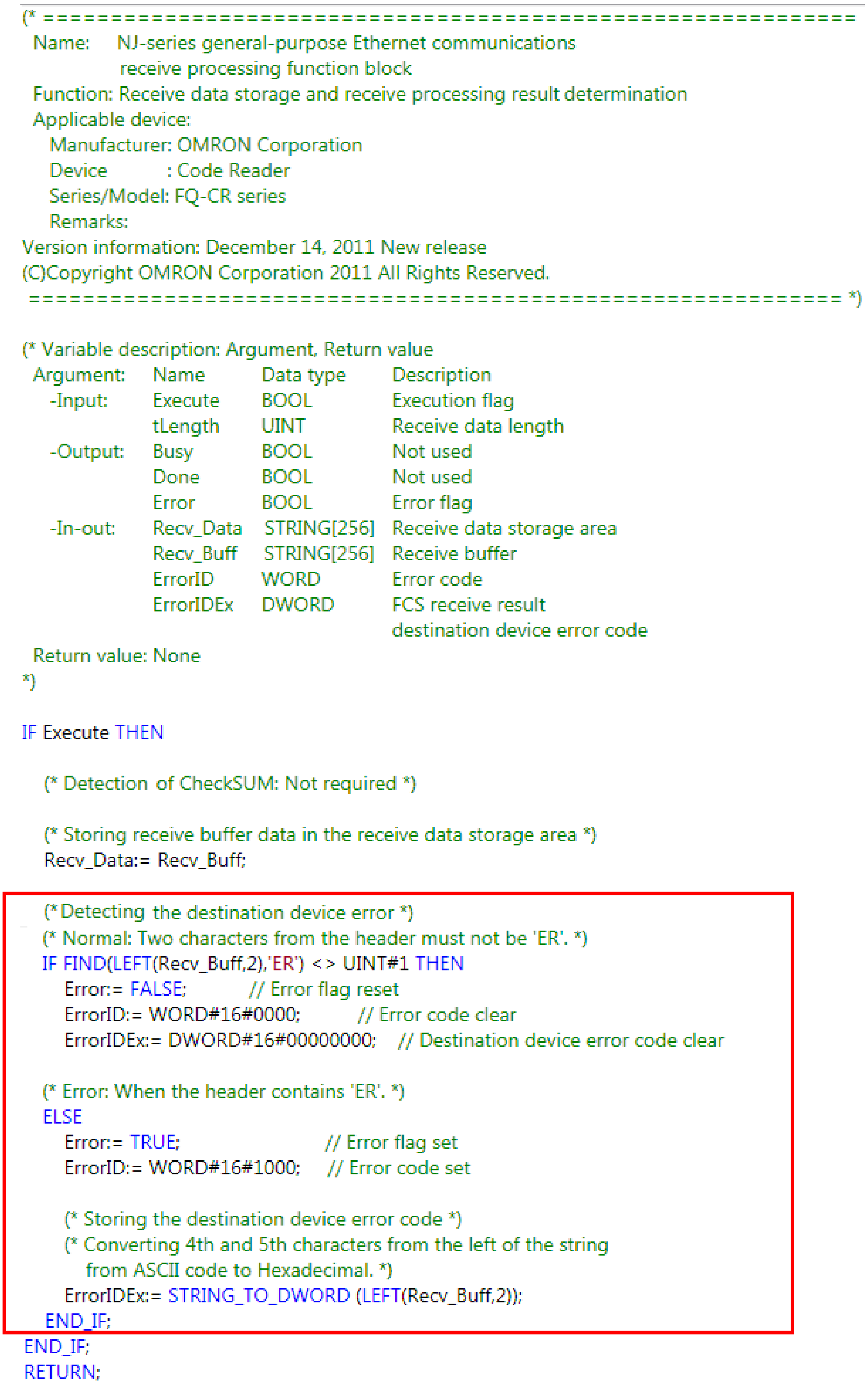
•Output

Name	Data type	Meaning	Description	Valid range	Unit	Default
Busy	BOOL	Executing	Not used (Not used in this project.)	-	-	-
Done	BOOL	Normal end				
Error	BOOL	Error end	Error end	-	-	-

•Internal variable table

Name	Data type	Meaning	Description	Valid range	Unit	Default
Receive_ Check	STRING[5]	Receive FCS	FCS receive result of the receive data	Depends on data type	-	-
Calc_Che ck	STRING[5]	Receive FCS calculation value	FCS calculation result of the receive data	Depends on data type	-	-

•Program

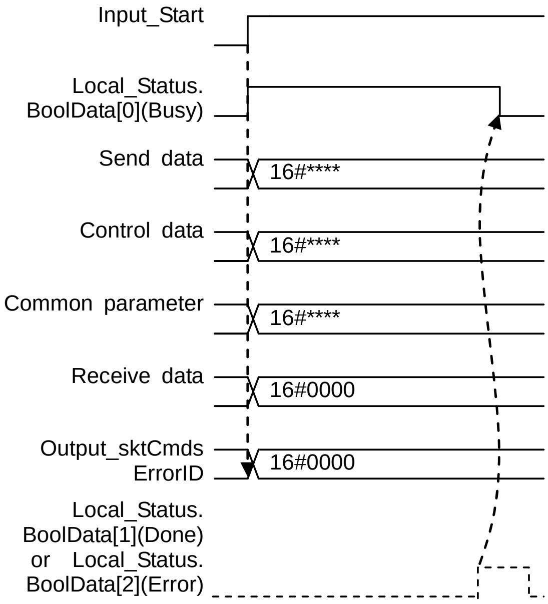


```
(* ============================================================
  Name:     NJ-series general-purpose Ethernet communications
            receive processing function block
  Function: Receive data storage and receive processing result determination
  Applicable device:
    Manufacturer: OMRON Corporation
    Device        : Code Reader
    Series/Model: FQ-CR series
    Remarks:
Version information: December 14, 2011 New release
(C)Copyright OMRON Corporation 2011 All Rights Reserved.
 ============================================================ *)

(* Variable description: Argument, Return value
  Argument:  Name        Data type     Description
    -Input:  Execute     BOOL          Execution flag
             tLength     UINT          Receive data length
    -Output: Busy        BOOL          Not used
             Done        BOOL          Not used
             Error       BOOL          Error flag
    -In-out: Recv_Data   STRING[256]   Receive data storage area
             Recv_Buff   STRING[256]   Receive buffer
             ErrorID     WORD          Error code
             ErrorIDEx   DWORD         FCS receive result
                                       destination device error code
  Return value: None
*)

IF Execute THEN

   (* Detection  of CheckSUM: Not required *)

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

   (*Detecting  the destination device error *)
   (* Normal: Two characters from the header must not be 'ER'. *)
   IF FIND(LEFT(Recv_Buff,2),'ER') <> UINT#1 THEN
      Error:= FALSE;          // Error flag reset
      ErrorID:= WORD#16#0000;         // Error code clear
      ErrorIDEx:= DWORD#16#00000000;    // Destination device error code clear

   (* Error: When the header contains 'ER'. *)
   ELSE
      Error:= TRUE;                   // Error flag set
      ErrorID:= WORD#16#1000;     // Error code set

      (* Storing the destination device error code *)
      (* Converting 4th and 5th characters from the left of the string
         from ASCII code to Hexadecimal. *)
      ErrorIDEx:= STRING_TO_DWORD (LEFT(Recv_Buff,2));
   END_IF;
END_IF;
RETURN;
```

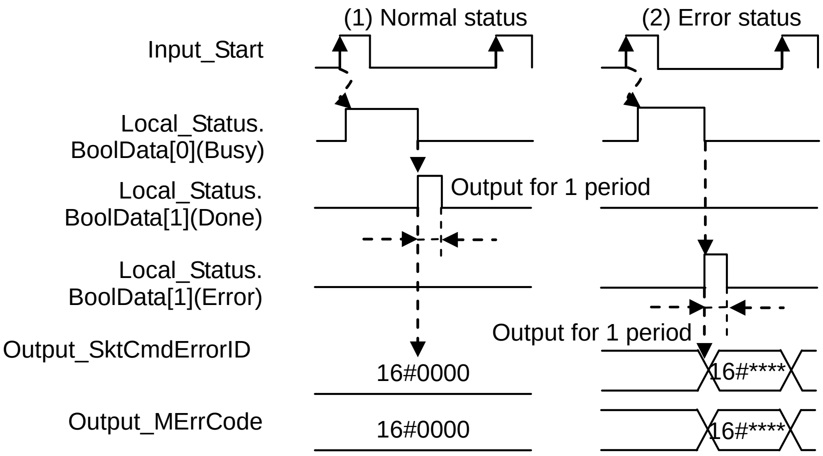
9.6. Timing Charts

The timing charts of the ST program are shown below.

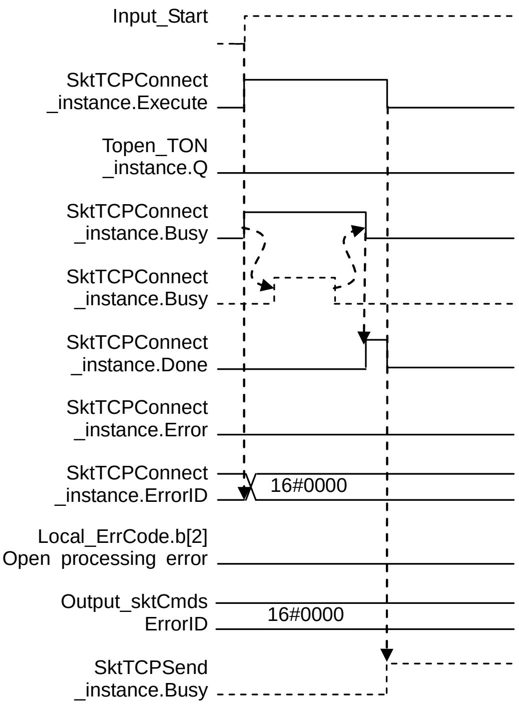
Execute & setting



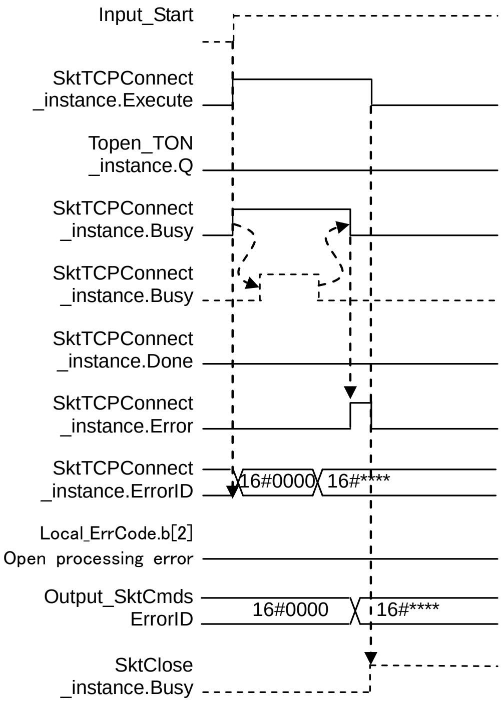
If *Input_Start* changes from True (ON) to False (OFF) during execution, a normal end or an error end is output for one period after the processing is completed.



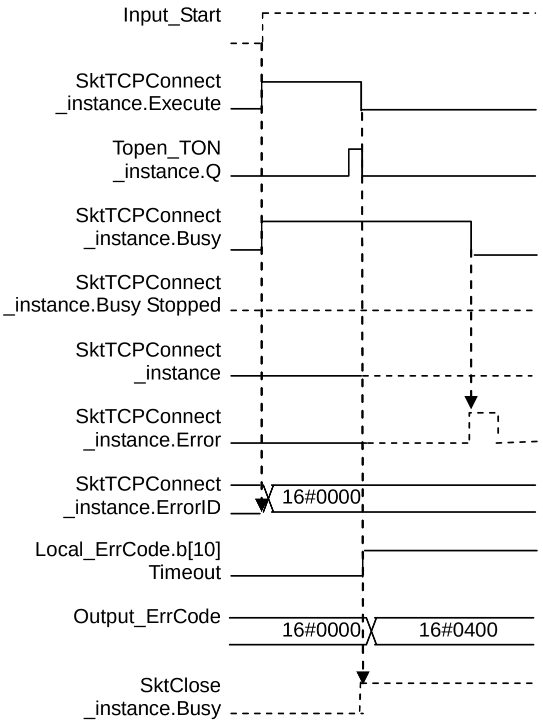
●Open processing


Input_Start
SktTCPConnect _instance.Execute
Topen_TON _instance.Q
SktTCPConnect _instance.Busy
SktTCPConnect _instance.Busy
SktTCPConnect _instance.Done
SktTCPConnect _instance.Error
SktTCPConnect _instance.ErrorID
16#0000
Local_ErrCode.b[2] Open processing error
Output_sktCmds ErrorID
16#0000
SktTCPSend _instance.Busy

(Normal end)

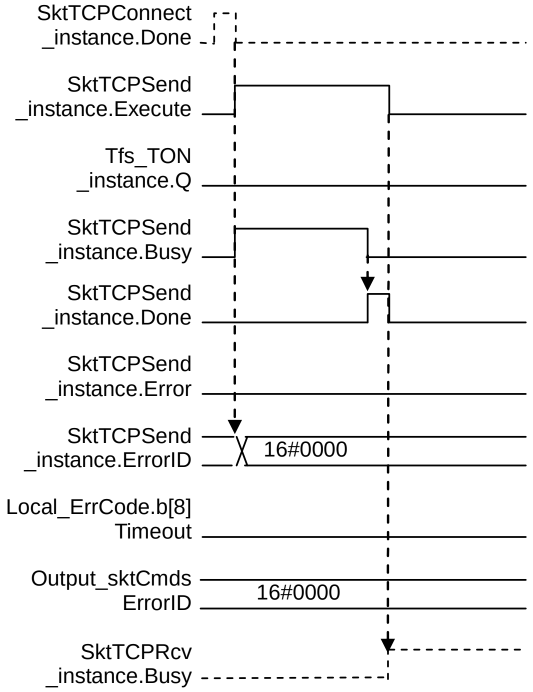

Input_Start
SktTCPConnect _instance.Execute
Topen_TON _instance.Q
SktTCPConnect _instance.Busy
SktTCPConnect _instance.Busy
SktTCPConnect _instance.Done
SktTCPConnect _instance.Error
SktTCPConnect _instance.ErrorID
16#0000
16#****
Local_ErrCode.b[2] Open processing error
Output_SktCmds ErrorID
16#0000
16#****
SktClose _instance.Busy

(Error end)

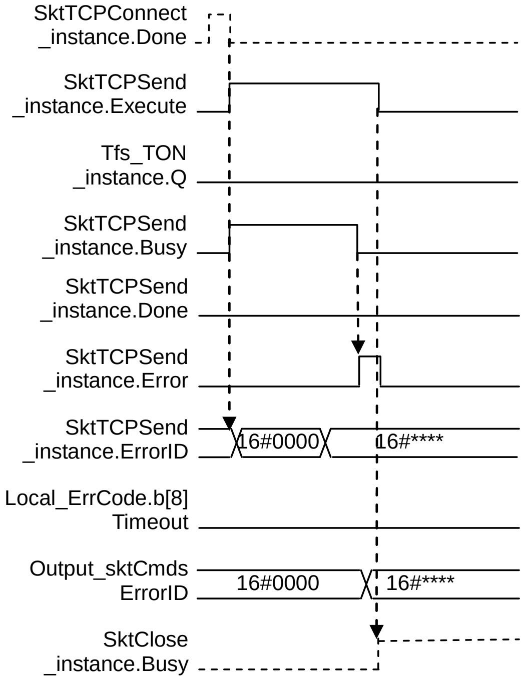

Input_Start
SktTCPConnect _instance.Execute
Topen_TON _instance.Q
SktTCPConnect _instance.Busy
SktTCPConnect _instance.Busy Stopped
SktTCPConnect _instance
SktTCPConnect _instance.Error
SktTCPConnect _instance.ErrorID
16#0000
Local_ErrCode.b[10] Timeout
Output_ErrCode
16#0000
16#0400
SktClose _instance.Busy

(Timeout)

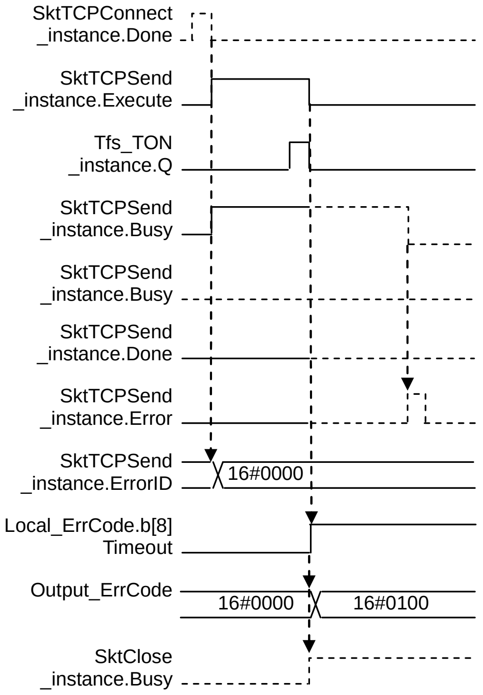
●Send processing



(Normal end)

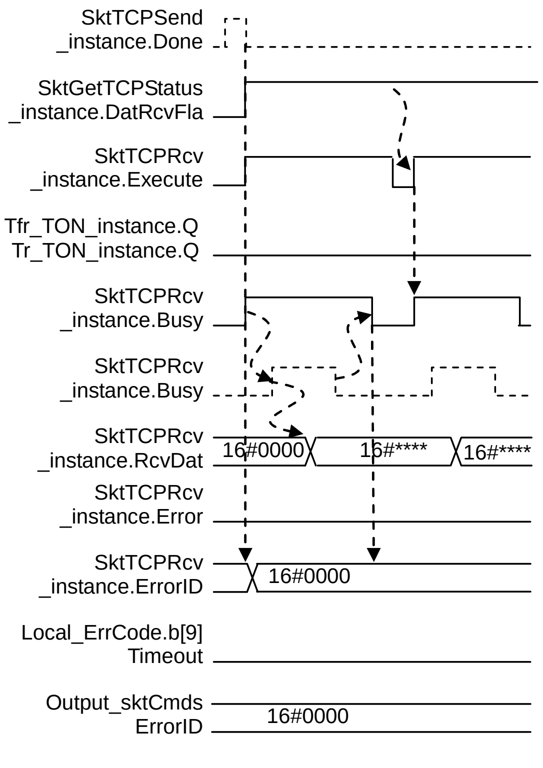


(Error end)

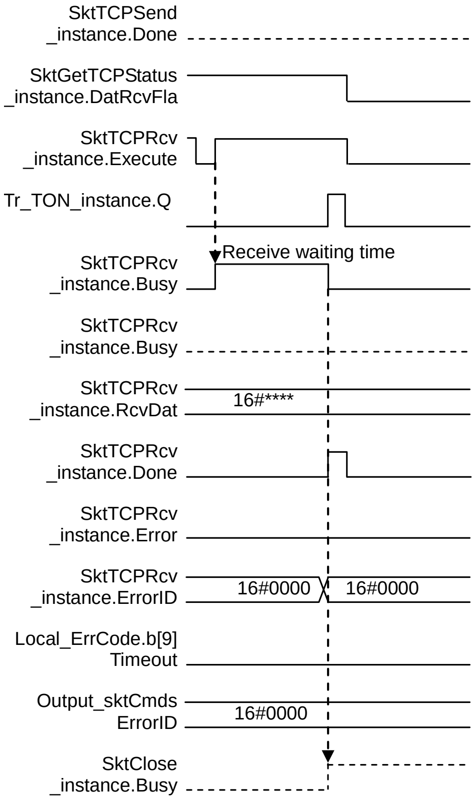


(Timeout)

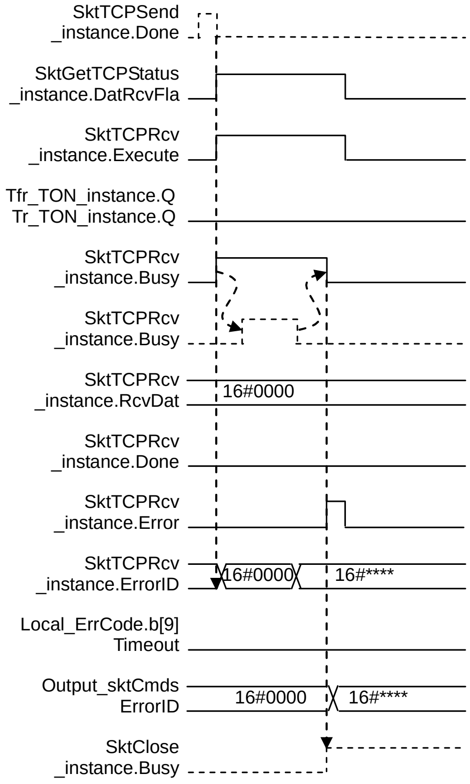
●Receive processing



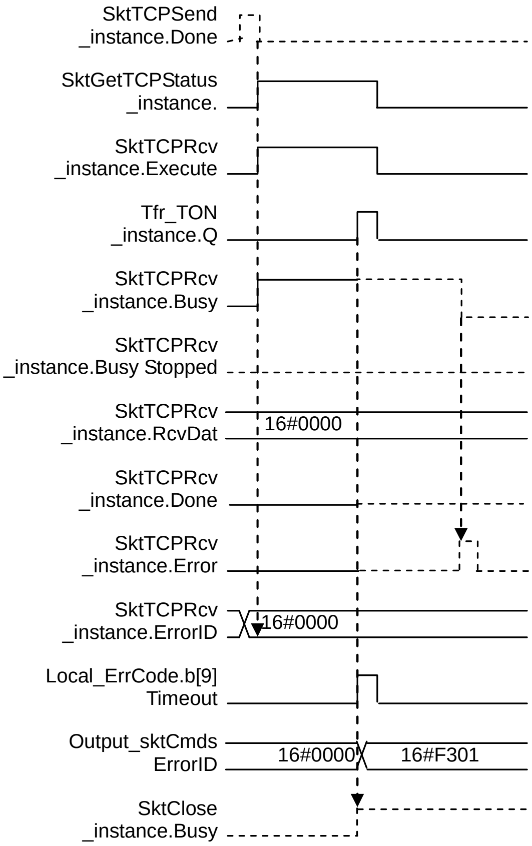
(Repeat)



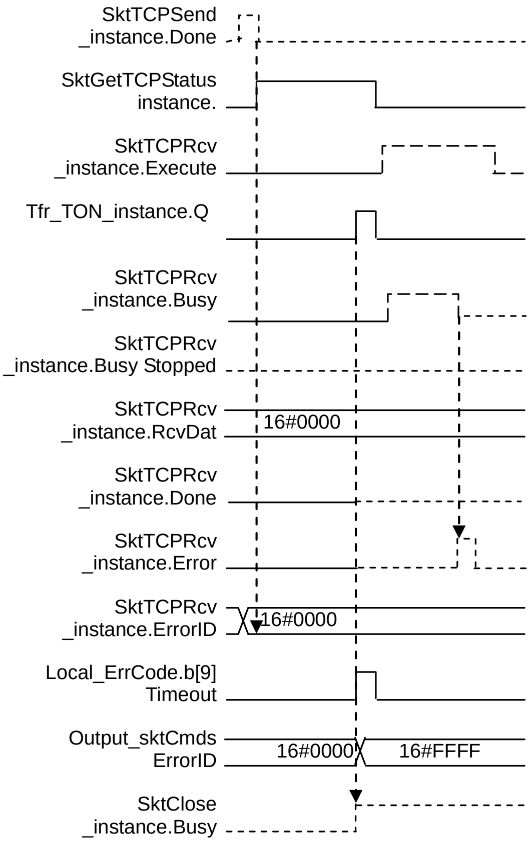
(Normal end)



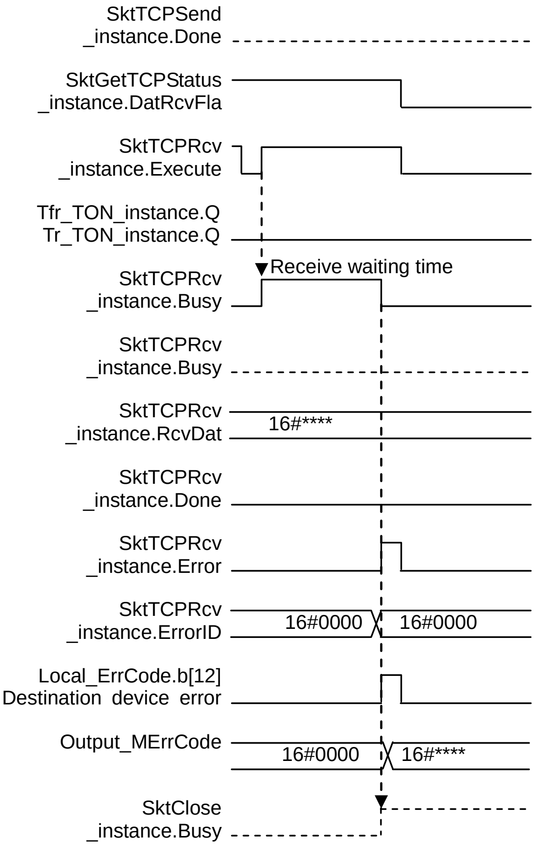
(Error end)



(Timeout: Receive error)

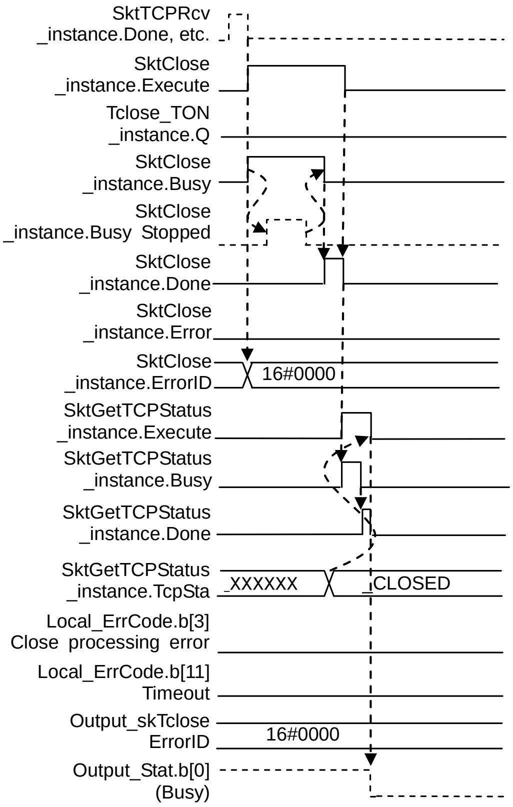


(Timeout: No receive data)

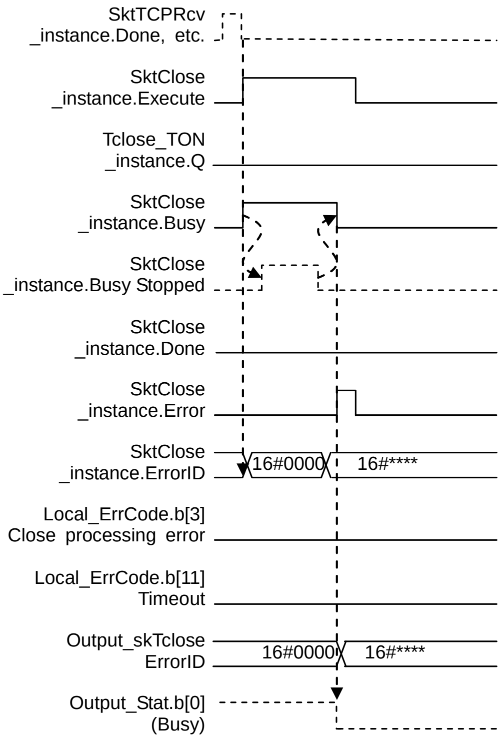


(Destination device error)

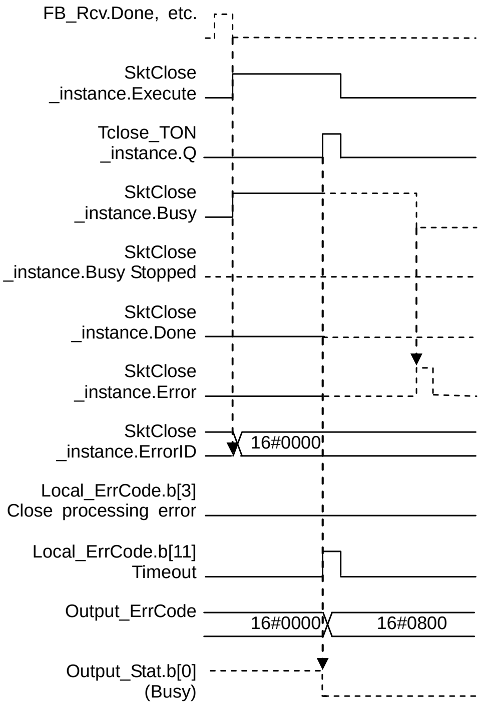
●Close processing



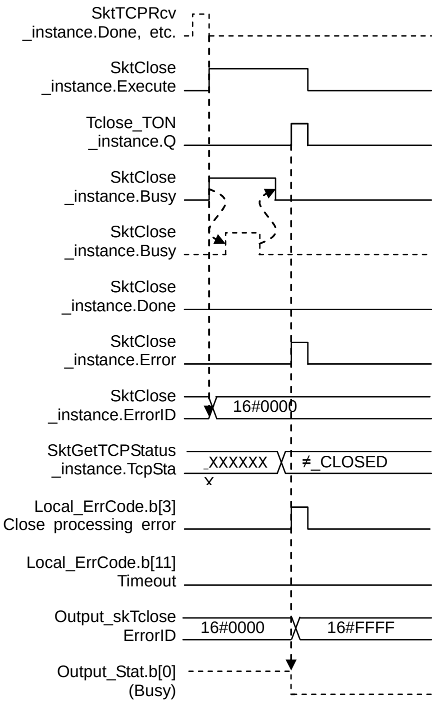
(Normal end)



(Error end)



(Timeout)



(Status error)

9.7. Error Process

9.7.1. Error Code List

The error codes for this ST program are shown below.

●TCP connection status error [Output_EtnTcpSta]

If the TCP connection status does not enter the normal status (*_CLOSED*) within a time after the close processing, a TCP connection status code is set in the *Output_EtnTcpSta* variable. (If the close processing ends in error, check this also.)

Error code enumerator [_eCONNECTION_STATE]	Description
_CLOSED	Connection closed. (Normal status)
_LISTEN	Waiting for connection
_SYN SENT	SYN sent in active status.
_SYN RECEIVED	SYN sent and received.
_ESTABLISHED	Already established.
_CLOSE WAIT	FIN received and waiting for completion.
_FIN WAIT1	Completed and FIN sent.
_CLOSING	Completed and exchanged FIN. Awaiting ACK.
_LAST ACK	FIN received and completed. Awaiting ACK.
_FIN WAIT2	Completed and ACK received. Awaiting FIN.
_TIME WAIT	After closing, pauses twice the maximum segment life (2MSL).

●Error code [Output_SktCmdsErrorID], [Output_SkTcloseErrorID]

If an error occurs in the open processing, send processing or receive processing, the error code is stored in the *Output_SktCmdsErrorID* variable and then the close processing is executed.

If an error occurs in the close processing, the error code is stored in the *Output_SkTcloseErrorID* variable and the processing ends. The main error codes are shown below.

(O: Open processing (SktTCPConnect instruction), S: Send processing (SktTCPSend instruction), R: Receive processing (SktTCPRcv instruction), C: Close processing (SktClose instruction). Target processing is indicated by ○.

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 results of instruction processing exceeded the data area range of the output parameter.
16#2000	○	-	-	-	An instruction was executed when there was a setting error in the local IP address.
16#2002	○	-	-	-	Address resolution failed for a destination node with the domain name that was specified in the instruction.
16#2003	○	○	○	-	The status was not suitable for execution of the instruction. •SktTCPConnect Instruction The TCP port that is specified with the *SrcTcpPort* input variable is already open. The destination node that is specified with *DstAdr* input variable does not exist. The destination node that is specified with *DstAdr* and *DstTcpPort* input variables are not waiting for a connection. •SktTCPRcv Instruction The specified socket is receiving data. The specified socket is not connected. •SktTCPSend Instruction The specified socket is sending data. The specified socket is not connected.
16#2006	-	-	○	-	A timeout occurred for a socket service instruction.
16#2007	-	○	○	○	The handle that is specified for the socket service instruction is not correct.
16#2008	○	○	○	○	The maximum resources that you can use for socket service instructions at the same time was exceeded.
16#FFFF	○	○	○	○	Processing ends without completing the executing of an instruction.

Additional Information

For details, refer to *Appendix A-1 Error Codes Related to Instructions, A-2 Error Code Descriptions* and *A-3 Error Code Details* in the *NJ-series Instructions Reference Manual* (Cat. No. W502).

Additional Information

For details on socket service errors and countermeasures, refer to *9-7 Precautions in Using Socket Services* of *Chapter 9 Socket Service* in the *NJ-series CPU Unit Built-in EtherNet/IP Port User's Manual* (Cat. No. W506).

●Error flag (Error end/timeout) [Output_ErrCode]

If an error end or a timeout occurs for the open processing, send processing, receive processing or close processing, the error flag will be set in the *Output_ErrCode* variable and the error code will be stored in the *Output_SktCmdsErrorID* variable or the *Output_SkTcloseErrorID* variable.

(If an error end or a timeout occurs for the close processing, check also the *Output_EtnTcpSta* variable for the TCP connection status error.)

(O: Open processing (SktTCPConnect instruction), S: Send processing (SktTCPSend instruction), R: Receive processing (SktTCPRcv instruction), C: Close processing (SktClose instruction). Target processing is indicated by ○.

Error Flag	O	S	R	C	Description
16#0000	○	○	○	○	Normal end
16#0001		○			Send processing ended in error.
16#0002			○		Receive processing ended in error.
16#0004	○				Open processing ended in error.
16#0008				○	Close processing ended in error.
16#0100		○			Send processing did not end in time.
16#0200			○		Receive processing did not end in time. (Including when an arrival of the response cannot be checked.)
16#0400	○				Open processing did not end in time.
16#0800				○	Close processing did not end in time.
16#0010					Processing number error
16#0020					Send/receive required/not required detection error
16#1000					Destination device error
16#2000					Destination device FCS (checksum) error
16#8000	○	○	○	○	Error occurs

*The error flags detected for each processing are added and the addition result is stored in the error flag.

●Destination device error code

If there is an error in the data received from the destination device, the error code is stored in the *Output_MErrCode* variable.

Error code	Description
16#00000000	Normal end
“ER”	The response from the destination device is illegal
16#FFFFFFFF	Not executed

Additional Information

For details and corrective actions for the destination device errors, refer to *Command Format* under *Controlling the Sensor from an External Device (Procedure for No-protocol Command/Response Communications)* in *8-2 Outputting/Controlling with Ethernet* of the user's manual for each Code Reader.

9.7.2. TCP Connection Status Errors and Corrective Actions

This section describes the situation in which the TCP connection status errors occur and explains the corrective actions.

●Affects of the TCP connection errors

After a TCP connection status error occurs, if no corrective action is taken or the error is not noticed and this project file is executed again, then the destination node specified with destination address input variable (*DstAdr*) and destination TCP port number input variable (*DstTcpPort*) may not be waiting for a connection. Hereinafter this error is referred to as an open processing error. This may be affected by the TCP connection status error that occurred when the previous communication processing ended. (For error details, refer to *9.7.1 Error Code List*.)

●Situation in which the TCP connection status error occurs

Both a TCP connection status error after the close processing and an open processing error that occurs when the next communications processing is performed can be caused by the fact that the close processing is not completed at the destination device. Although, all processing (until the close processing) of the project file ended in the Controller, the close processing completion notification is not received from the destination device (It is not confirmed that the close processing is completed at the destination device).

●Corrective action

The close processing may not be completed at the destination device. Check if the communications port of the destination device is closed. If not closed or not possible to check, reset the communications port of the destination device. The communications port of the destination device can be reset by executing restart operation from the software or by cycling the power supply. For details, refer to the manual for each destination device.

Precautions for Correct Use

Make sure the destination device is disconnected from other device before resetting the communications port of the destination device.

●State of the Controller at a TCP connection status error

When a TCP connection status error occurs, the processing of this project file is completed. However, the resend/time monitoring function of the Controller (TCP/IP function), which is described in 9.3.2. Time Monitoring Function, may be operating. This resend processing stops in the following cases. Therefore, you do not have to stop it.

- When the open processing request is made again by restarting the project file
- When a communications problem such as cable disconnection is cleared during resend processing
- When the resend processing is completed with the TCP/IP time monitoring (timeout) function
- When the Controller is restarted or the power supply is turned OFF

10. Revision History

Revision code	Date of revision	Revision reason and revision page
01	Mar. 26, 2013	First edition

OMRON Corporation **Industrial Automation Company**
Tokyo, 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.
No. 438A Alexandra Road # 05-05/08 (Lobby 2),
Alexandra Technopark,
Singapore 119967
Tel: (65) 6835-3011/Fax: (65) 6835-2711

OMRON ELECTRONICS LLC
One Commerce Drive Schaumburg,
IL 60173-5302 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: