Machine Automation Controller NJ-series

General-purpose Ethernet Connection Guide (TCP/IP) OMRON Corporation

ZW-series Displacement Sensor

Network

Connection

Guide

always in control

P558-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-[][][][] NJ301-[][][][]	NJ-series CPU Unit Hardware User's Manual
W501	NJ501-[][][][] NJ301-[][][][]	NJ-series CPU Unit Software User's Manual
W506	NJ501-[][][][] NJ301-[][][][]	NJ-series CPU Unit Built-in EtherNet/IP Port User's Manual
W504	SYSMAC-SE2[][][]	Sysmac Studio Version 1 Operation Manual
W502	NJ501-[][][][] NJ301-[][][][]	NJ-series Instructions Reference Manual
Z322	ZW-C1[]	Confocal Fiber Type Displacement Sensor User's Manual
Z332	ZW-CE1[]	ZW Series Displacement Sensor (Confocal Fiber Type) User's Manual

2. Terms and Definitions

Term	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 Ethernet. IP addresses must be set and managed so that 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 socket services enable data exchange with destination nodes. The NJ-series Machine Automation Controller performs socket communications by using the standard socket service instructions.
Connect processing/ Accept processing	Open processing is executed on each node to connect the TCP socket. The open method depends on whether the node is opened as a server or client. In this document, the processing executed to open a node as a client is called "connect processing" and the processing executed to open as a server is called "accept processing".
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 connect 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 of 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 user is encouraged to confirm the standards and regulations that the system must conform to.
(4) It is prohibited to copy, to reproduce, and to distribute a part of or whole part of this document without the permission of OMRON Corporation.
(5) The information contained in this document is current as of July 2013. It is subject to change without notice for improvement.

The following notation is used in this document.

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

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

Precautions for Safe Use

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

Precautions for Correct Use

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

Additional Information

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

Symbol

●	The filled circle symbol indicates operations that you must do. The specific operation is shown in the circle and explained in text. This example shows a general precaution for something that you must do.

4. Overview

This document describes the procedure for connecting OMRON Corporation's Displacement Sensor (ZW series) (hereinafter referred to as the Displacement Sensor) to the NJ-series Machine Automation Controller (hereinafter referred to as the Controller) via Ethernet, and provides the procedure for checking their connection.
Refer to the Ethernet communications settings of the prepared Sysmac Studio project file to understand the setting procedure and key points to connect the devices via Ethernet.
The user program in this Sysmac Studio project file is used to check the Ethernet connection by sending/receiving the message of "VR (Version information acquisition)" to/from the destination device.

Prepare the latest Sysmac Studio project file beforehand. To obtain the file, contact your OMRON representative.

Name	File name	Version
Sysmac Studio project file (extension: smc)	OMRON_ZW_ETN(TCP)_EV101.smc	Ver.1.01

*Hereinafter, the Sysmac Studio project file is referred to as the "project file".
The user program in the project file is referred to as the "program".

⚠ **Caution**

This document aims to explain the wiring method and communications settings necessary to connect the corresponding devices and provide the setting procedure. The program used in this document is designed to check if the connection was properly established, and is not designed to be constantly used at a site. Therefore, functionality and performances are not sufficiently taken into consideration. When you construct an actual system, please use the wiring method, communications settings and setting procedure described in this document as a reference and design a new program according to your application needs.

5. Applicable Devices and Support Software

5.1. Applicable Devices

The applicable devices are as follows:

Manufacturer	Name	Model	Version
OMRON	NJ-series CPU Unit	NJ501-[][][][] NJ301-[][][][]	Versions listed in Section 5.2 or higher versions
OMRON	Confocal Fiber Type Displacement Sensor Controller	ZW-C1[]/CE1[]CE1[]T	
OMRON	Sensor Head	ZW-S[][]	

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

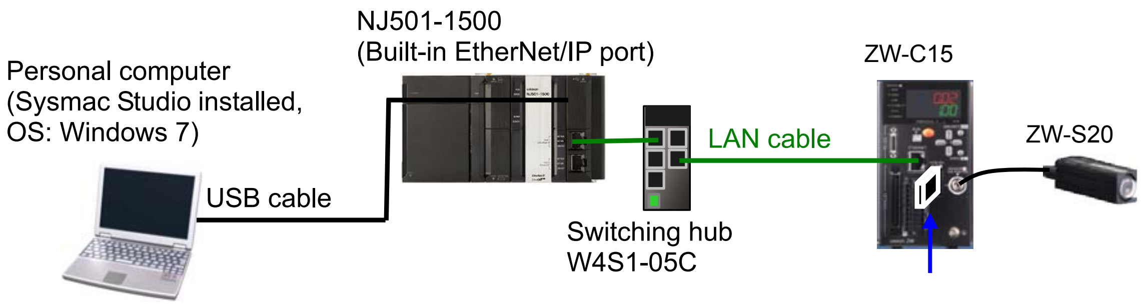
This document describes the procedure to establish the network connection. It does not provide information about operation, installation nor wiring method of each device.
For details on the above products (other than communication connection procedures), refer to the manuals for the corresponding products or contact your OMRON representative.

Additional Information

You can connect devices with the versions listed in Section 5.2 or higher versions.
For devices whose versions are not listed in Section 5.2, the versions are not managed or there is no version restriction.
To connect a device whose model number is not listed in Section 5.2, use the same version of the device that is listed.

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 Units (Built-in EtherNet/IP port)	NJ501-1500	Ver.1.03
OMRON	Power Supply Unit	NJ-PA3001	
OMRON	Switching Hub	W4S1-05C	Ver.1.0
OMRON	Sysmac Studio	SYSMAC-SE2[][][]	Ver.1.04
OMRON	Sysmac Studio project file	OMRON_ZW_ETN(TCP)_ EV101.smc	Ver.1.01
-	Personal computer (OS:Windows7)	-	
-	USB cable (USB 2.0 type B connector)	-	
-	LAN cable (STP (shielded, twisted-pair) cable of Ethernet category 5 or higher)	-	
OMRON	Displacement Sensor Controller	ZW-C15	Ver.1.000
OMRON	Displacement Sensor Sensor Head	ZW-S20	
OMRON	Calibration ROM	(Included with Sensor Head.)	

Precautions for Correct Use

Prepare the latest project file in advance.
To obtain the file, contact your OMRON representative.

Precautions for Correct Use

Update the Sysmac Studio to the version specified in this section or higher version using the auto update function. If a version not specified in this section is used, the procedures described in Section 7 and subsequent sections may not be applicable. In that case, use the equivalent procedures described in the Sysmac Studio Version 1 Operation Manual (Cat.No. W504).

Additional Information

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

Additional Information

The system configuration in this document uses USB for the connection between the personal computer and the Controller. For information on how to install the 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 Communications Settings

This section describes the specifications of the communication parameters that are set in this document and outlines the operation.

Additional Information

To perform communications without using the settings described in this section, you need to modify the program. For information on the program, refer to *Section 9. Program*.

6.1. Ethernet Communications Settings

The settings required for Ethernet communications are shown below.

6.1.1. Communications Settings between the Controller and the Displacement Sensor

The setting example below is used to explain the procedure for connecting the Controller to the Displacement Sensor.

Setting item	Controller	Displacement Sensor
IP address	192.168.250.1	192.168.250.2
Subnet mask	255.255.255.0	255.255.255.0 (Default)
Gateway	---.---.---.---	192.168.0.1 (Default)
Port number	Automatic allocation (Set by the program.)	9600 (Default)
Terminator	-	[CR] (Default)

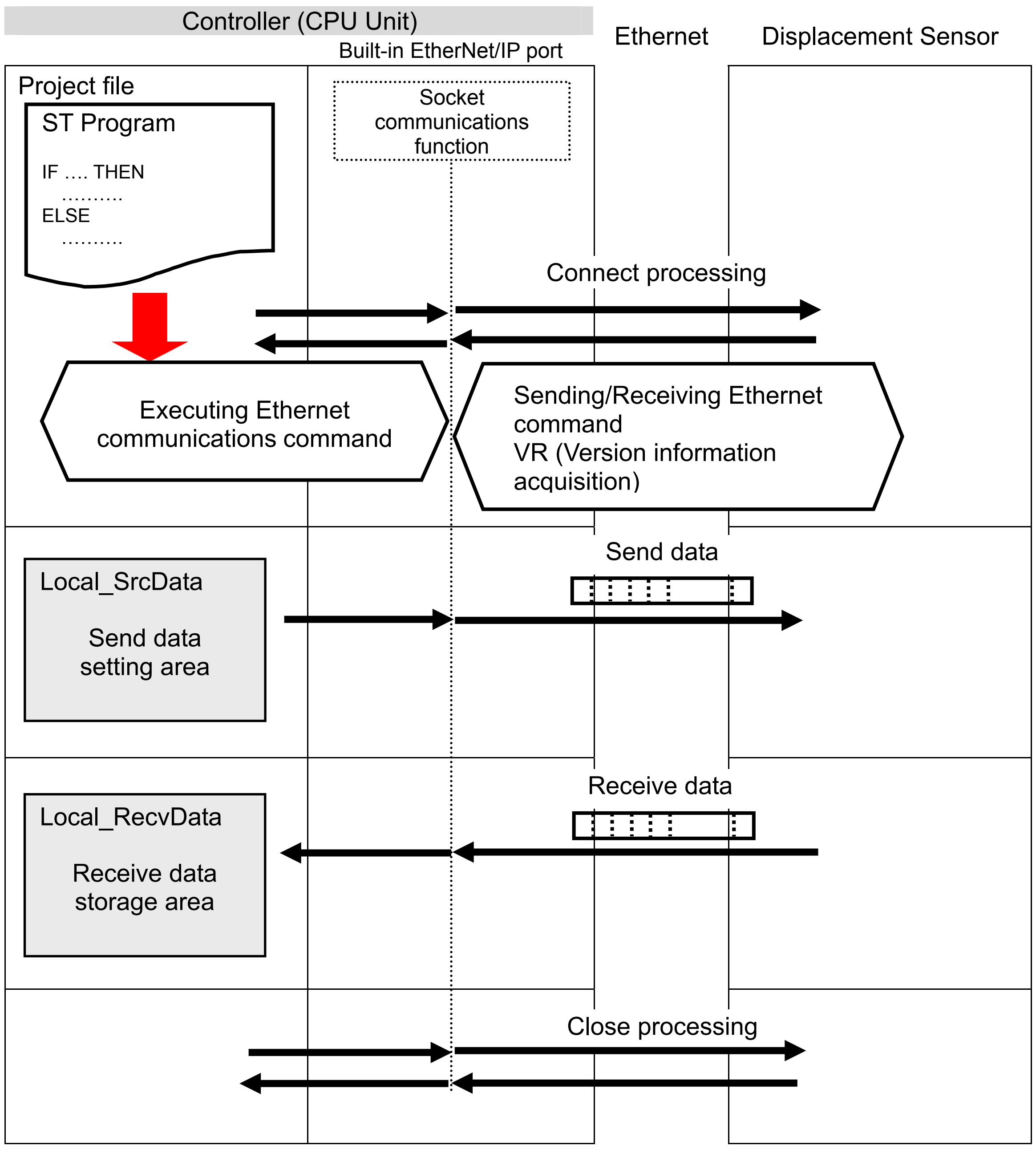
*In this document, the gateway setting is unnecessary because the connection is made in the same segment.

*This project file uses the default settings of the keep-alive and linger option functions for the TCP socket communications (keep-alive: use, linger option: Do not use). Use these functions according to the system when necessary.

6.2. Example of Checking Connection

This document shows an example of a Structured Text (ST) program in which the Controller executes the connect processing, send/receive processing, and close processing on the Displacement Sensor.

The Controller and Displacement Sensor send and receive the message of "VR (Version information acquisition). The following figure outlines the operation.



7. Connection Procedure

This section describes the procedure for connecting the Displacement Sensor to the Controller via Ethernet.
This document explains the procedures for setting up the Controller and Displacement Sensor 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 Displacement Sensor to the Controller via Ethernet.

7.2. Setting Up the Displacement Sensor	Set up the Displacement Sensor.
↓	
7.2.1. Parameter Setting	Set the parameters for the Displacement Sensor.
↓	
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 and import the 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 Controller.
↓	
7.3.3. Connecting Online and Transferring the Project Data	Connect online with the Sysmac Studio and transfer the project data to the Controller.
↓	
7.4. Checking the Ethernet Communications	Execute the program and confirm that Ethernet communications are normally performed.
↓	
7.4.1. Executing the Program and Checking the Receive Data	Execute the program and confirm that the correct data are written to the variables of the Controller.

7.2. Setting Up the Displacement Sensor

Set up the Displacement Sensor.

7.2.1. Parameter Setting

Set the parameters for the Displacement Sensor.

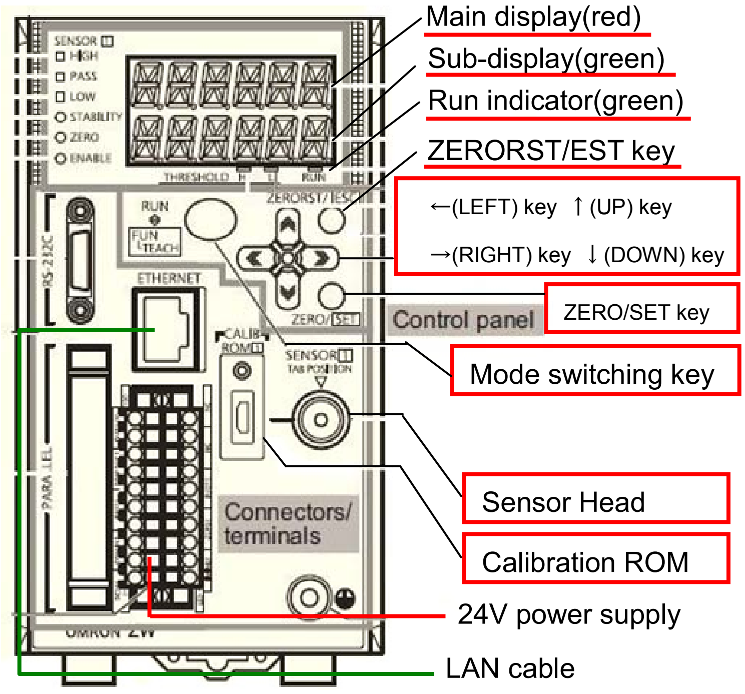
1 Check the keys and display used to set parameters for the Displacement Sensor.

Connect the Sensor Head.

Insert the Calibration ROM.

Connect the LAN cable.

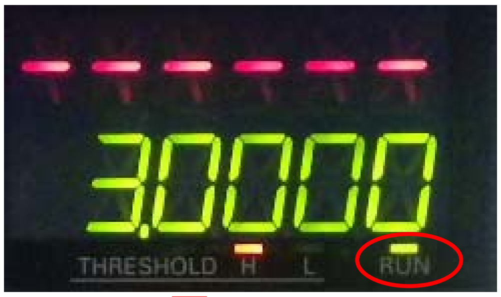
Turn ON the power supply to the Displacement Sensor.



2 After the startup screen is displayed, the RUN mode screen is displayed.

The RUN indicator is lit as shown on the right.

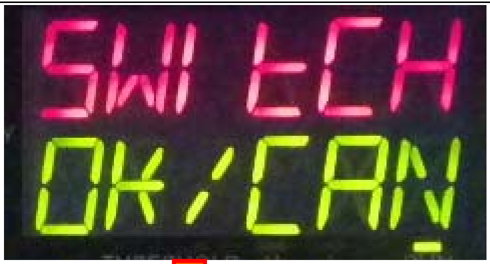
Hold down the **Mode switching** Key for two seconds.



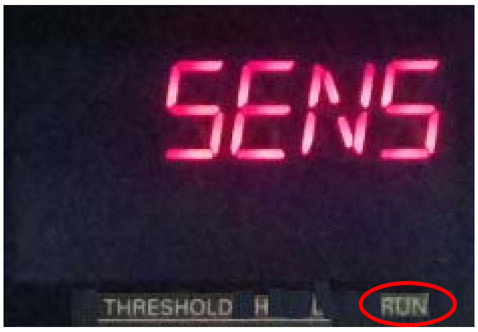
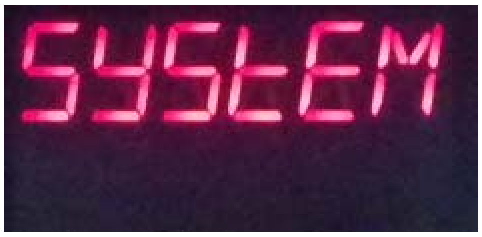
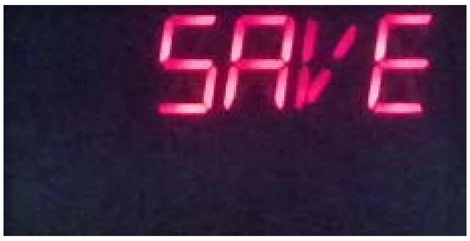
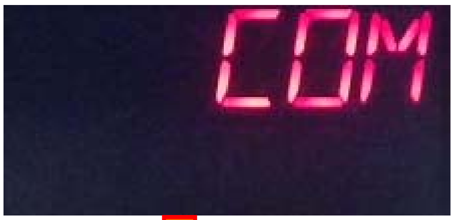
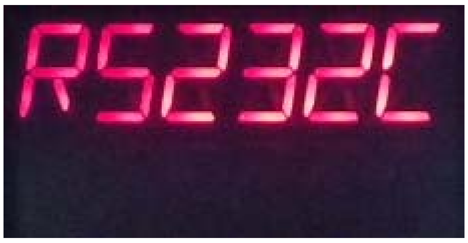
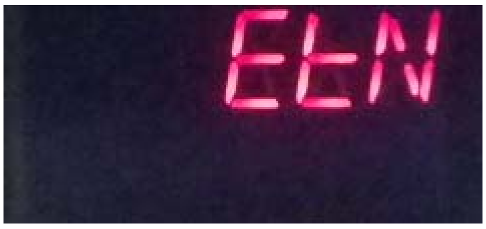
Hold down the **Mode switching** Key for two seconds.

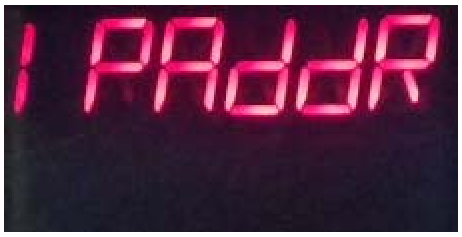
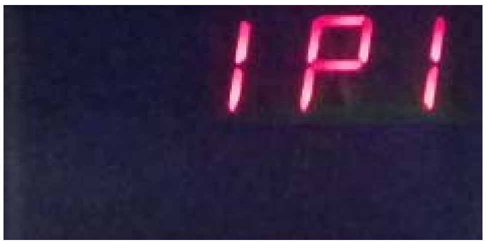
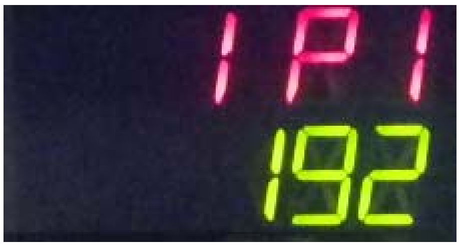
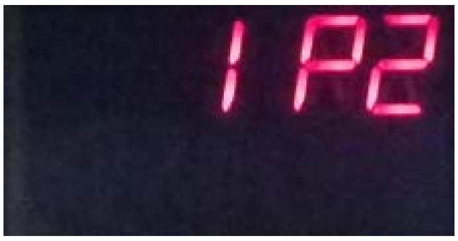
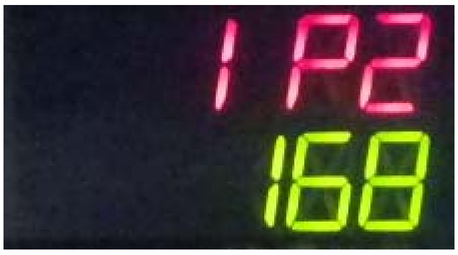
3 A confirmation screen for mode switching is displayed.

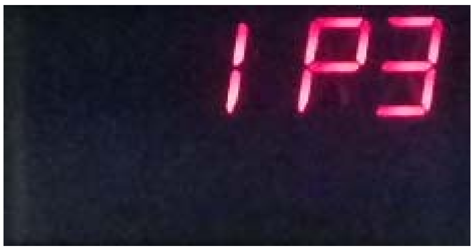
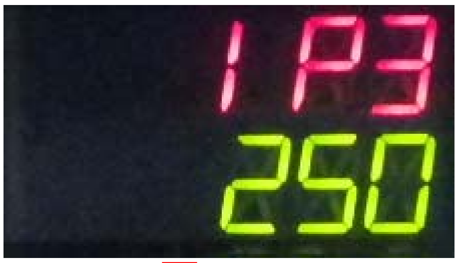
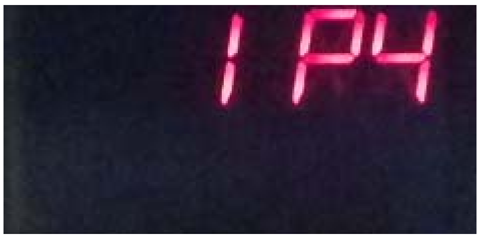
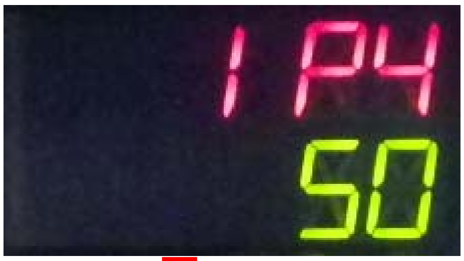
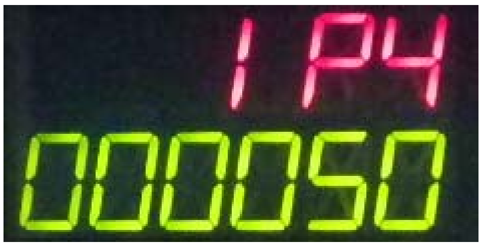
Press the **ZERO/SET** Key.



Press the **ZERO/SET** Key once.

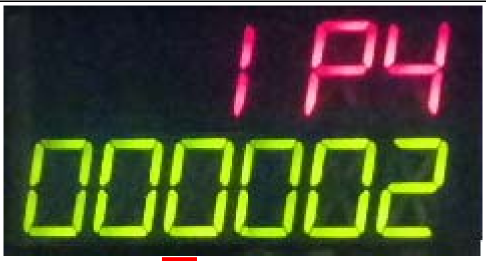
4	The FUN mode screen is displayed. The RUN indicator is not lit as shown on the right. Press → (RIGHT) or ← (LEFT) Key to change the main display content from SENS to SYSTEM. Press the **ZERO/SET** Key.	 Press the → (RIGHT) or ← (LEFT) Key.  Press the **ZERO/SET** Key once.
5	SAVE is displayed on the main display. Press → (RIGHT) or ← (LEFT) key to change the main display content from SAVE to COM. Press the **ZERO/SET** Key.	 Press the → (RIGHT) or ← (LEFT) Key.  Press the **ZERO/SET** Key once.
6	RS232C is displayed on the main display. Press the → (RIGHT) Key and change the main display content from RS232C to ETN. Press the **ZERO/SET** Key.	 Press the → (RIGHT) Key once.  Press the **ZERO/SET** Key once.

7	IPADDR is displayed on the main display. Press the **ZERO/SET** Key.	 Press the **ZERO/SET** Key once.
8	IP1 is displayed on the main display. Press the **ZERO/SET** Key. Confirm that 192 is displayed on the sub-display. *If the setting value is different, change the value by referring to steps 11 and 12. *In this step, you set 192 that is the first octet of IP address 192.168.250.2. Press the **ZERORST/ESC** Key once. The first screen in this step is displayed again. Press the → (RIGHT) Key once.	 Press the **ZERO/SET** Key.  Press the **ZERORST/ESC** Key. IP1 is displayed. Press the → (RIGHT) Key once.
9	IP2 is displayed on the main display. Press the **ZERO/SET** Key. Confirm that 168 is displayed on the sub-display. *If the setting value is different, change the value by referring to steps 11 and 12. *In this step, you set 168 that is the second octet of IP address 192.168.250.2. Press the **ZERORST/ESC** Key once. The first screen in this step is displayed again. Press the → (RIGHT) Key once.	 Press the **ZERO/SET** Key once.  Press the **ZERORST/ESC** Key once. IP2 is displayed. Press the → (RIGHT) Key once.

10	IP3 is displayed on the main display. Press the **ZERO/SET** Key. Confirm that 250 is displayed on the sub-display. *If the setting value is different, change the value by referring to steps 11 and 12. *In this step, you set 250 that is the third octet of IP address 192.168.250.2. Press the **ZERORST/ESC** Key once. The first screen in this step is displayed again. Press the → (RIGHT) Key once.	 Press the **ZERO/SET** Key once.  Press the **ZERORST/ESC** Key once. IP3 is displayed. Press the → (RIGHT) key once.
11	IP4 is displayed on the main display. Press the **ZERO/SET** Key. The default value is displayed on the sub-display. Press the **ZERO/SET** Key. The sub-display content changes, allowing you to change the value. Select a digit whose value you want to change by pressing the → (RIGHT) or ← (LEFT) Key. You can change the value of a blinking digit. Change the value by pressing the ↑ (UP) or ↓ (DOWN) Key. Change to 000002.	 Press the **ZERO/SET** Key once.  Press the **ZERO/SET** Key once.  Press the → (RIGHT) or ← (LEFT) Key. Press the ↑ (UP) or ↓ (DOWN) Key.

12

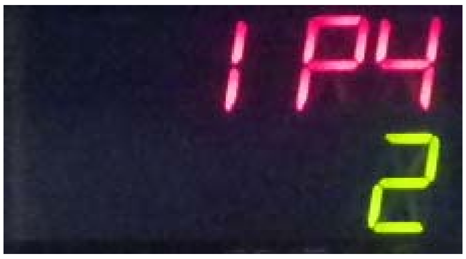
000002 is displayed on the sub-display.



Press the **ZERO/SET** Key.

Press the **ZERO/SET** Key once.

2 is displayed on the sub-display.



*In this step, you set 2 that is the fourth octet of IP address 192.168.250.2.

Press **ZERORST/ESC** Key once. The first screen in step 11 is displayed again.

Press the **ZERORST/ESC** Key once.

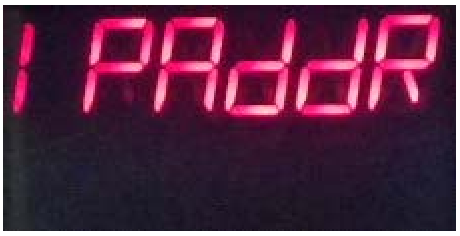
IP4 is displayed.

Press the **ZERORST/ESC** Key once.

Press the **ZERORST/ESC** Key once.

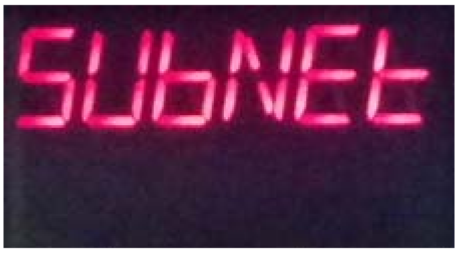
13

IPADDR is displayed on the main display.



Press the → (RIGHT) Key once to change the main display to SUBNET.

Press the → (RIGHT) Key once.

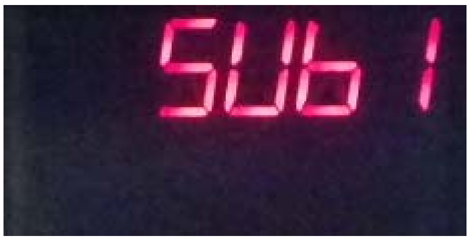


Press the **ZERO/SET** Key.

Press the **ZERO/SET** Key once.

14

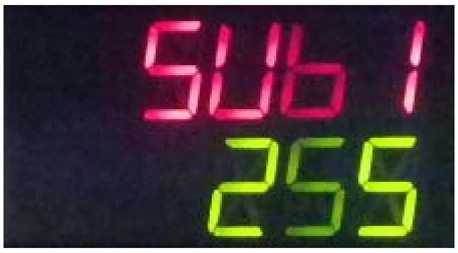
The main display content changes to SUB1.



Press the **ZERO/SET** Key.

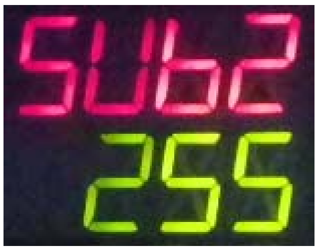
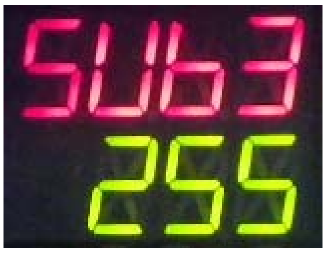
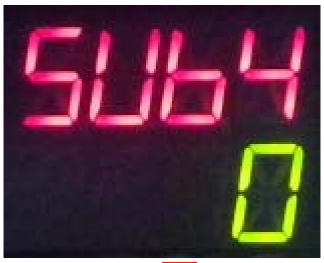
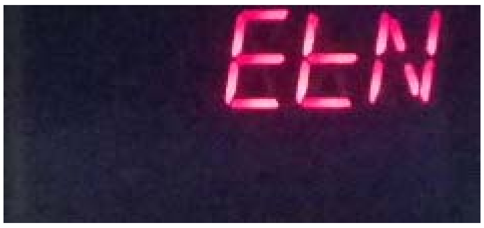
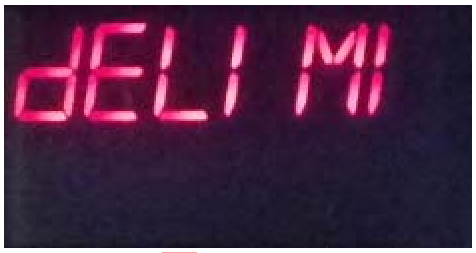
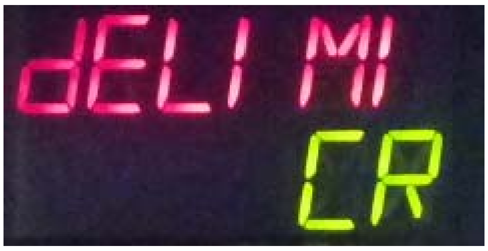
Press the **ZERO/SET** Key once.

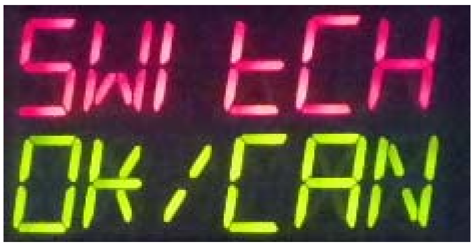
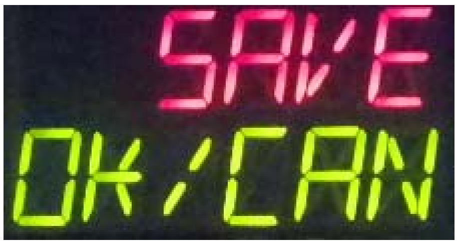
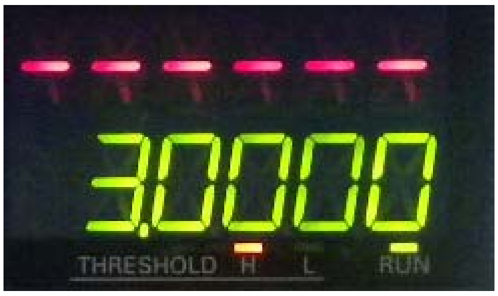
255 is displayed on the sub-display.



Press the **ZERORST/ESC** Key once.

Press the **ZERORST/ESC** Key once.

15	Press the → (RIGHT) Key and change the main display content to SUB2, SUB3 and SUB4. Press the **ZERO/SET** Key to check the setting values. Check that the values between SUB2 and SUB4 are as follows: •SUB2, SUB3: 255 •SUB4: 0 *In steps 14 and 15, you set subnet mask 255.255.255.0. After checking, press the **ZERORST/ESC** Key three times.	   Press the **ZERORST/ESC** Key three times.
16	ETN is displayed on the main display. Press the → (RIGHT) Key and change the main display content to DELIMI. Press the **ZERO/SET** Key.	 Press the → (RIGHT) Key once.  Press the **ZERO/SET** Key once.
17	Confirm that CR is displayed on the sub-display. *If the value is different, change the value by pressing ↑ (UP) or ↓ (DOWN). Hold down the **Mode switching** Key for two seconds.	 <Setting range> CR/LF/CRLF Default: CR Hold down the **Mode switching** Key for two seconds.

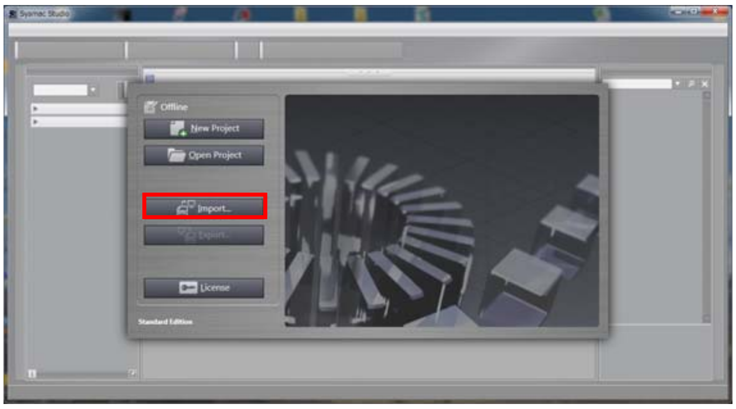
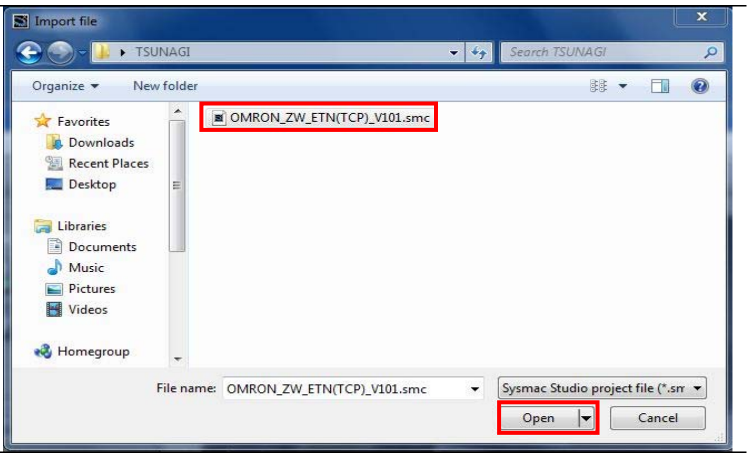
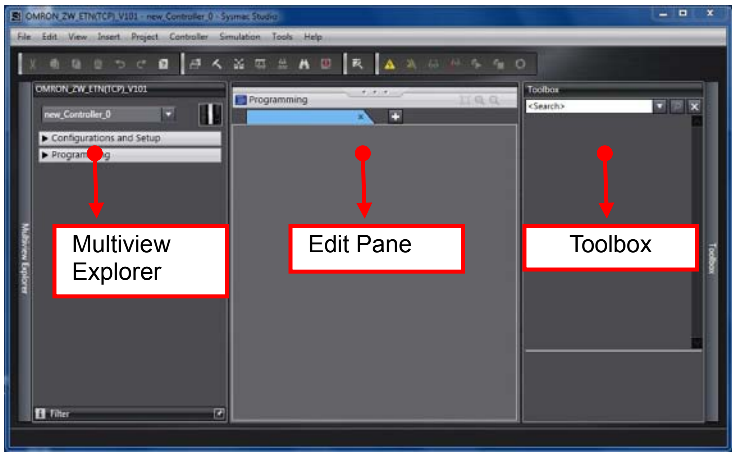
18	The confirmation screen for mode switching is displayed. Press the **ZERO/SET** Key.	
		Press the **ZERO/SET** Key once.
	The save confirmation screen is displayed. Press the **ZERO/SET** Key.	
		Press the **ZERO/SET** Key once.
	The RUN mode screen is displayed.	
19	Cycle the power supply to the Displacement Sensor. *The new IP address and subnet mask will take effect after restarting.	

7.3. Setting Up the Controller

Set up the Controller.

7.3.1. Starting the Sysmac Studio and Importing the Project File

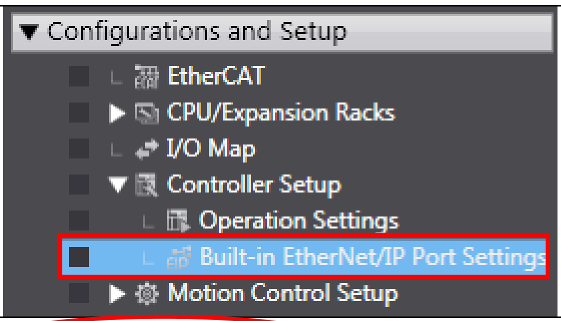
Start the Sysmac Studio and import the project file.
Install the Sysmac Studio and USB driver beforehand.

1	Confirm that the personal computer is connected to the Controller through a USB cable, and turn ON the power supply to the Controller. Start the Sysmac Studio and click the **Import** Button. *If a confirmation dialog for an access right is displayed at start, select to start.	 Offline New Project Open Project Import... Export... License Standard Edition
2	The Import File Dialog Box is displayed. Select OMRON_ZW_ETN(TCP)_EV101.smc and click the **Open** Button. *Obtain the project file from OMRON.	 Import file TSUNAGI Search TSUNAGI Organize New folder Favorites Downloads Recent Places Desktop Libraries Documents Music Pictures Videos Homegroup OMRON_ZW_ETN(TCP)_V101.smc File name: OMRON_ZW_ETN(TCP)_V101.smc Sysmac Studio project file (*.sm Open Cancel
3	OMRON_ZW_ETN(TCP)_V101 project is displayed. The left pane is called Multiview Explorer, the right pane is called Toolbox and the middle pane is called Edit Pane. *If an error message is displayed stating "Failed to Load Descendants", change the version of the Sysmac Studio to any version specified in 5.2. Device Configuration or higher version.	 OMRON_ZW_ETN(TCP)_V101 - new_Controller_0 - Sysmac Studio File Edit View Insert Project Controller Simulation Tools Help new_Controller_0 Configurations and Setup Programming Toolbox Multiview Explorer Edit Pane Toolbox

7.3.2. Checking the Parameters and Building

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

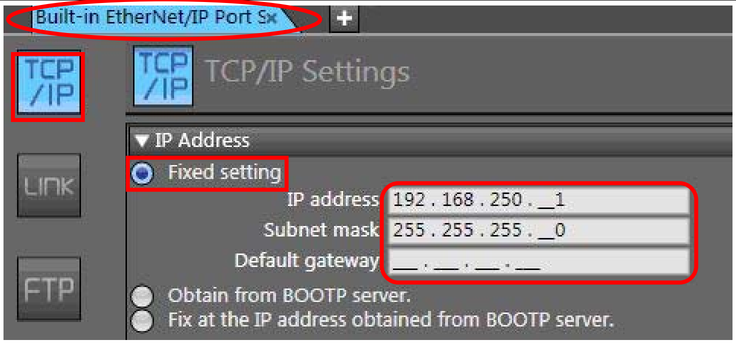
1 Double-click **Built-in EtherNet/IP Port Settings** under **Configurations and Setup** - **Controller Setup** in the Multiview Explorer.



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

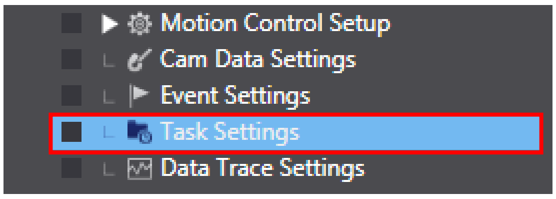
Select the **TCP/IP Settings** Button, select the *Fixed setting* Option in the IP Address Field, and make the following settings.

•IP address: 192.168.250.1
•Subnet mask: 255.255.255.0
•Default gateway: _._._._ (blank)

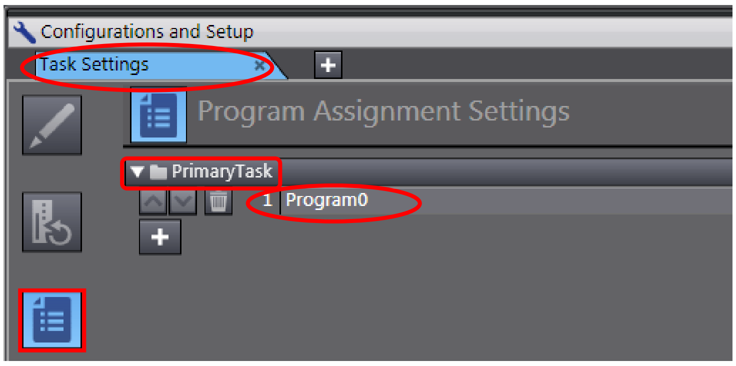

Built-in EtherNet/IP Port Se
TCP/IP
LINK
FTP
TCP/IP Settings
IP Address
Fixed setting
Obtain from BOOTP server.
Fix at the IP address obtained from BOOTP server.

IP address 192 . 168 . 250 . 1
Subnet mask 255 . 255 . 255 . 0
Default gateway __ . __ . __ . __

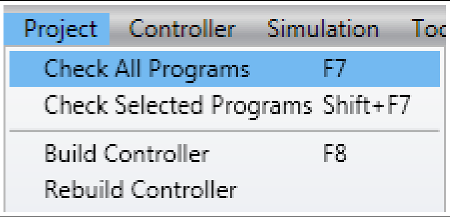
3 Double-click the **Task Settings** under **Configurations and Setup** in the Multiview Explorer

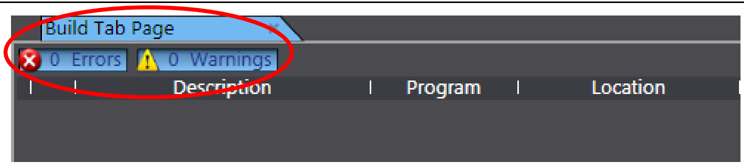
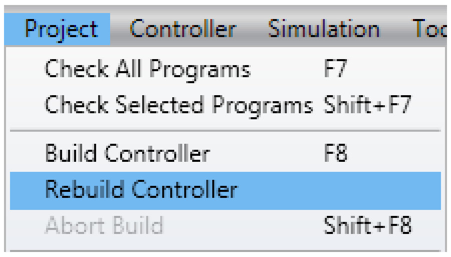
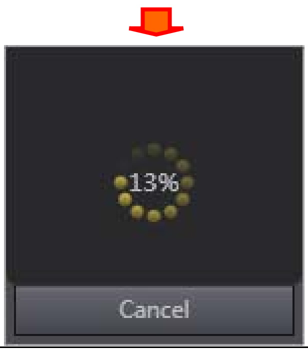
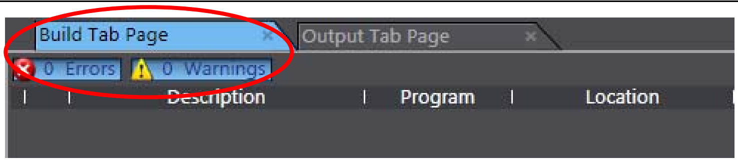


4 The Task Settings Tab Page is displayed in the Edit Pane. Click the **Program Assignment Settings** Button and confirm that Program0 is set under PrimaryTask.



5 Select ***Check All Programs*** from the Project Menu.



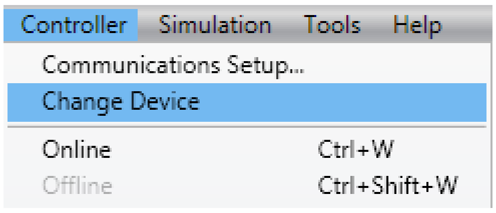
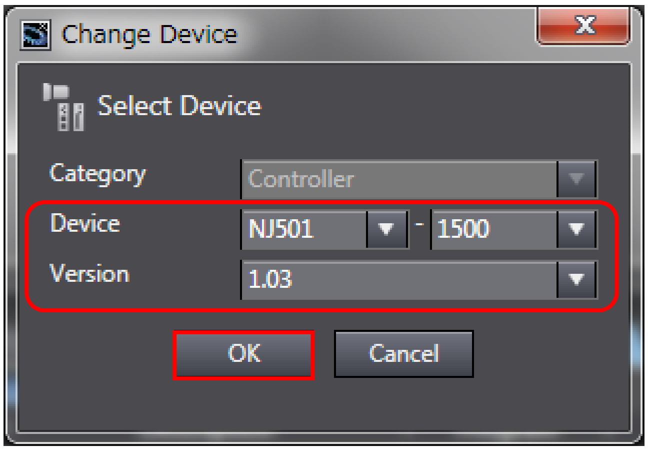
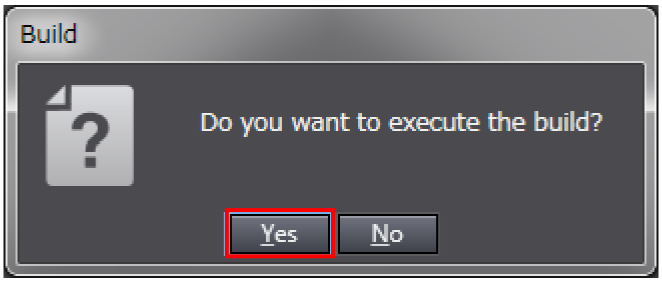
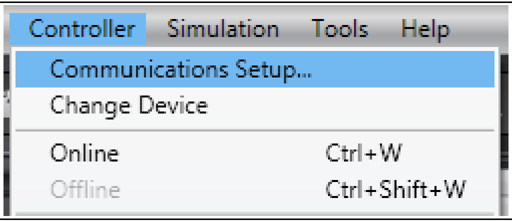
6	The Build Tab Page is displayed in the Edit Pane. Confirm that “0 Errors” and “0 Warnings” are displayed.	 Build Tab Page 0 Errors 0 Warnings Description Program Location
7	Select ***Rebuild Controller*** from the Project Menu.	 Project Controller Simulation Toc Check All Programs F7 Check Selected Programs Shift+F7 Build Controller F8 Rebuild Controller Abort Build Shift+F8
	A screen is displayed indicating the conversion is being performed.	
8	Confirm 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

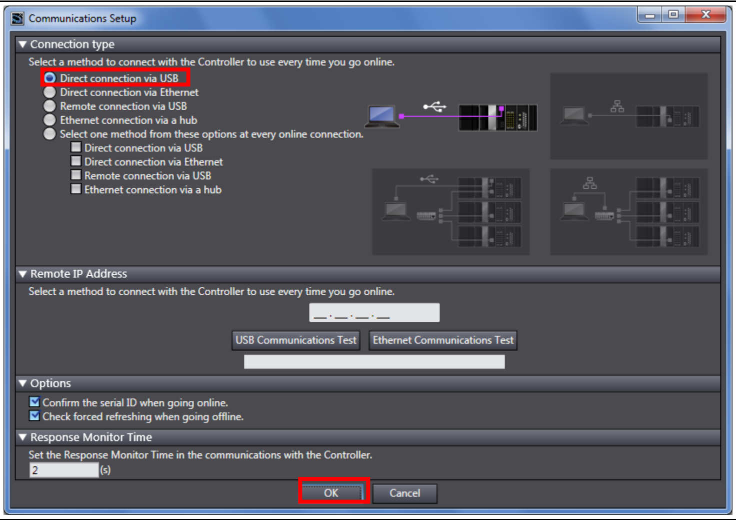
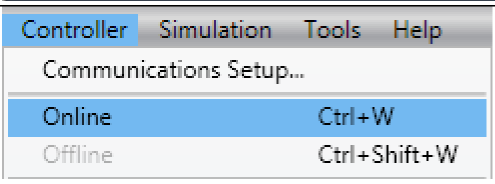
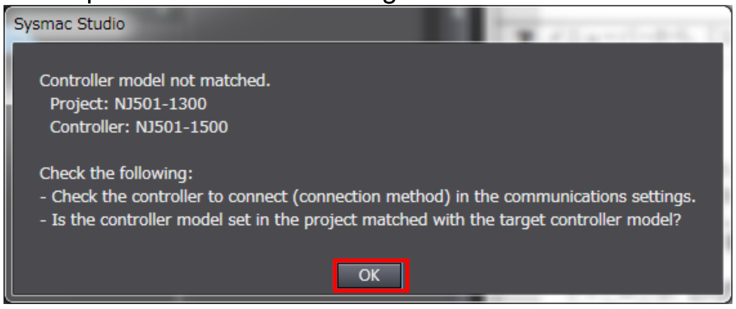
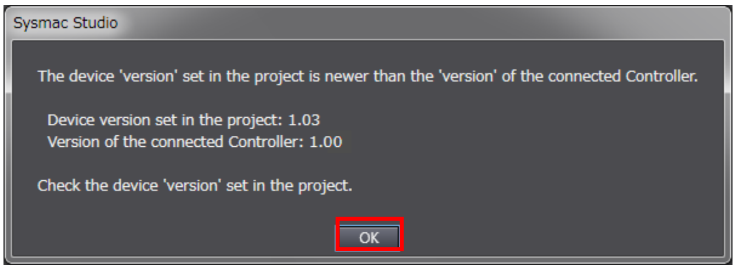
7.3.3. Connecting Online and Transferring the Project Data

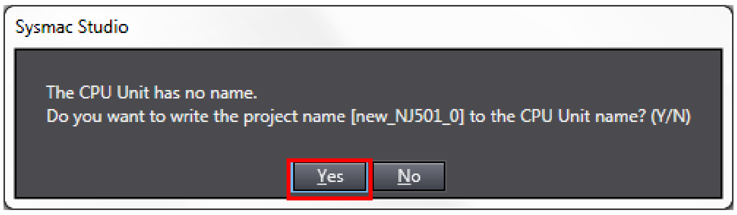
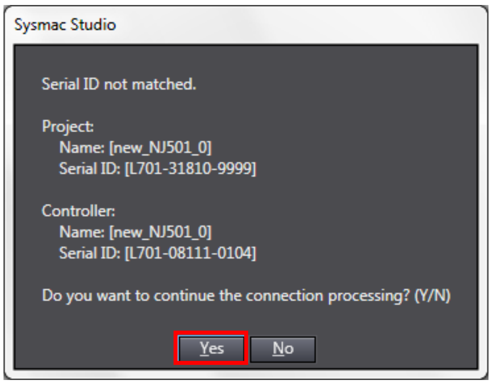
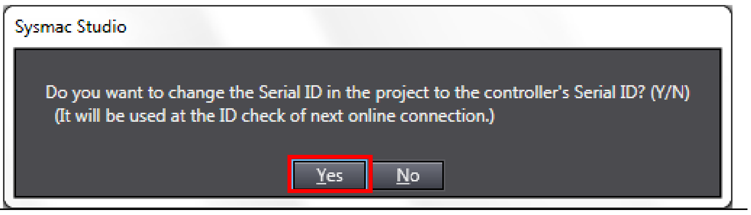
Connect online with the Sysmac Studio and transfer the project data to the Controller.

⚠ 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.

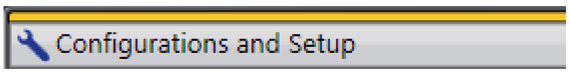
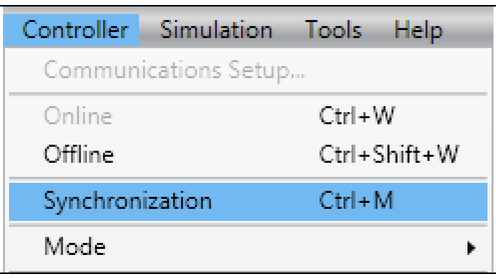
1	Select ***Change Device*** from the Controller Menu.	 Controller Simulation Tools Help Communications Setup... Change Device Online Ctrl+W Offline Ctrl+Shift+W
2	The Change Device Dialog Box is displayed. Confirm that the Device and Version are set as shown on the right and click the **OK** Button. *If the settings are different, change the values from the pull-down list.	 Change Device Select Device Category Controller Device NJ501 - 1500 Version 1.03
3	If settings are changed in step 2, the Build Dialog Box is displayed. Click the **Yes** Button. *This dialog box is not displayed if no change is made.	 Build Do you want to execute the build?
4	Select ***Communications Setup*** from the Controller Menu.	 Controller Simulation Tools Help Communications Setup... Change Device Online Ctrl+W Offline Ctrl+Shift+W

5	The Communications Setup Dialog Box is displayed. Select the *Direct Connection via USB* Option for 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
6	Select ***Online*** from the Controller Menu. *If the dialog on the right is displayed, the model or version of the Controller does not match that of the project file. Check the settings of the project file, return to step 1 and try again. Click the **OK** Button to close the dialog box. *The model and version displayed on the confirmation dialog box differ depending on the Controller used and the device setting of the project file.	 Controller Simulation Tools Help Communications Setup... Online Ctrl+W Offline Ctrl+Shift+W *Example of confirmation dialog box  Sysmac Studio Controller model not matched. Project: NJ501-1300 Controller: NJ501-1500 Check the following: - Check the controller to connect (connection method) in the communications settings. - Is the controller model set in the project matched with the target controller model? OK  Sysmac Studio The device 'version' set in the project is newer than the 'version' of the connected Controller. Device version set in the project: 1.03 Version of the connected Controller: 1.00 Check the device 'version' set in the project. OK

7	A confirmation dialog is displayed. Click the **Yes** Button. *The displayed dialog differs depending 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)  Sysmac Studio Serial ID not matched. Project: Name: [new_NJ501_0] Serial ID: [L701-31810-9999] Controller: Name: [new_NJ501_0] Serial ID: [L701-08111-0104] 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.)

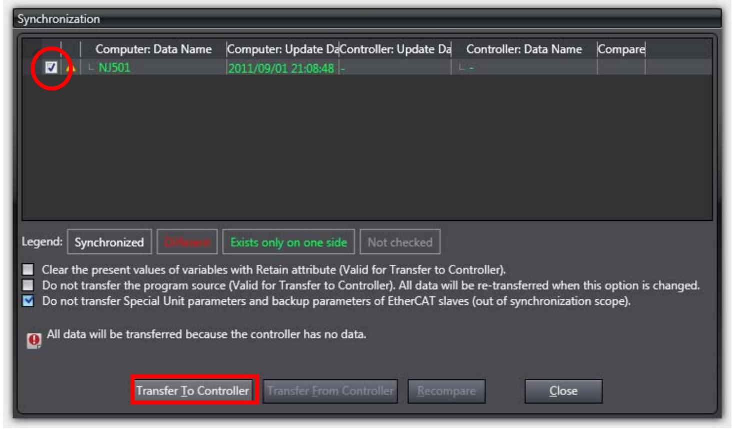
Additional Information

For details on online connections to a Controller, refer to *Section 5 Going Online with a Controller* in the *Sysmac Studio Version 1 Operation Manual* (Cat. No. W504).

8	When an online connection is established, a yellow bar is displayed on the top of the Edit Pane.	
9	Select ***Synchronization*** from the Controller Menu.	 Controller Simulation Tools Help Communications Setup... Online Ctrl+W Offline Ctrl+Shift+W Synchronization Ctrl+M Mode

10 The Synchronization Dialog Box is displayed.
Confirm that the data to transfer (NJ501 in the right figure) is selected. Then, click the **Transfer to Controller** Button.

*After executing **Transfer to Controller**, the Sysmac Studio project data is transferred to the Controller and the data are compared.



Synchronization

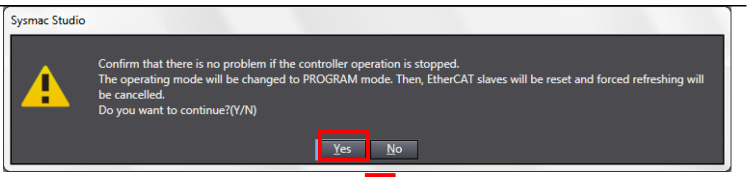
		Computer: Data Name	Computer: Update Da	Controller: Update Da	Controller: Data Name	Compare
✓	⚠	NJ501	2011/09/01 21:08:48	-	-	

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.
- [x] 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.

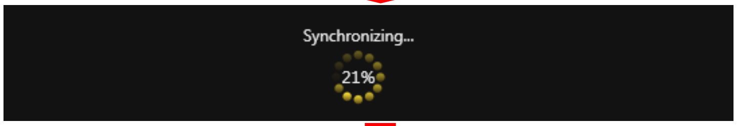
11 A confirmation dialog is displayed. Click the **Yes** Button.



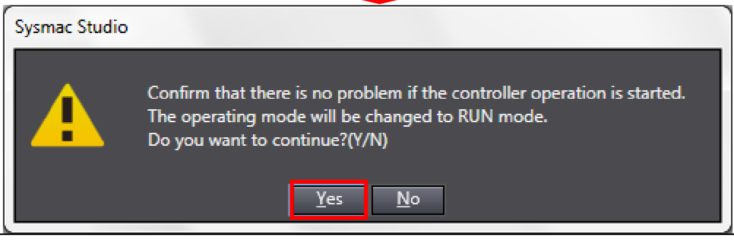
Sysmac Studio

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)

A screen stating "Synchronizing" is displayed.



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



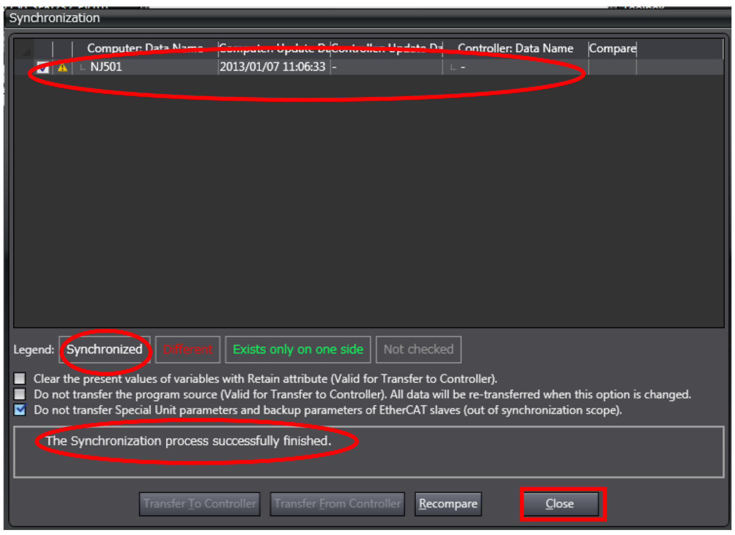
Sysmac Studio

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)

12 Confirm 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.

*A message stating "The synchronization process successfully finished" means that the project data of Sysmac Studio matches that of the Controller.

*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	2013/01/07 11:06:33	-	-	

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).

The Synchronization process successfully finished.

7.4. Checking the Ethernet Communications

Execute the program and confirm that Ethernet communications are normally performed.

⚠ Caution

Sufficiently confirm safety before you change the values of variables on a Watch Tab Page when the Sysmac Studio is online with the CPU Unit. Incorrect operation may cause the devices that are connected to Output Units to operate regardless of the operating mode of the Controller.

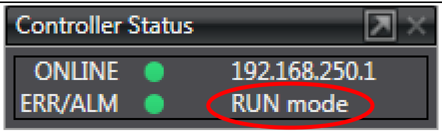
Precautions for Correct Use

Please confirm that the LAN cable is 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.

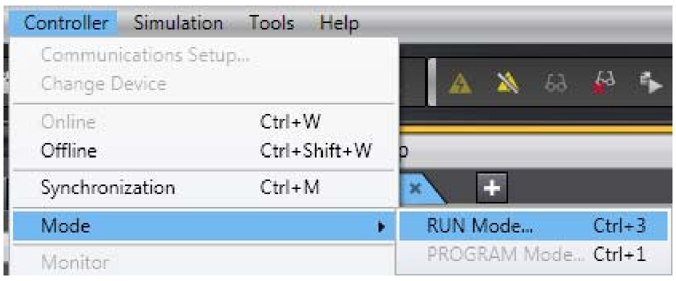
7.4.1. Executing the Program and Checking the Receive Data

Execute the program and confirm that the correct data are written to the variables of the Controller.

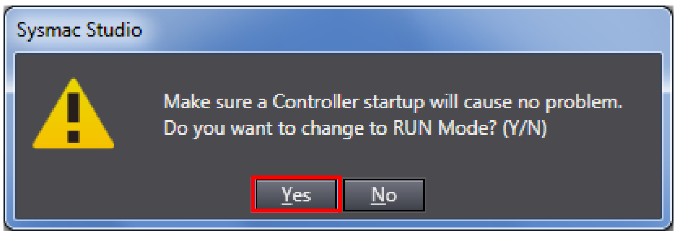
1 Confirm that RUN mode is displayed on the Controller Status Pane of the Sysmac Studio.



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

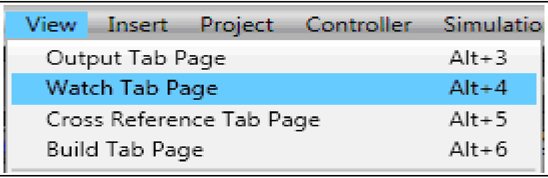


A confirmation dialog box is displayed. Click the **Yes** Button.

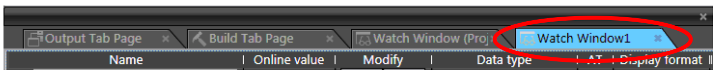


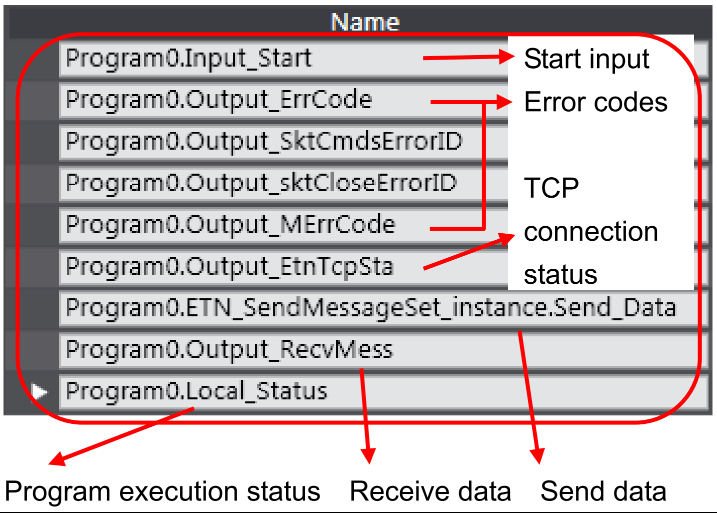
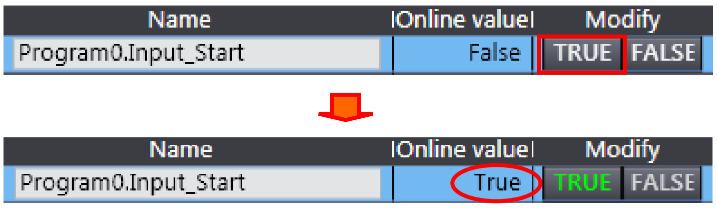
Make sure a Controller startup will cause no problem.
Do you want to change to RUN Mode? (Y/N)

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



3 The Watch Tab Page 1 is displayed in the lower section of the Edit Pane.

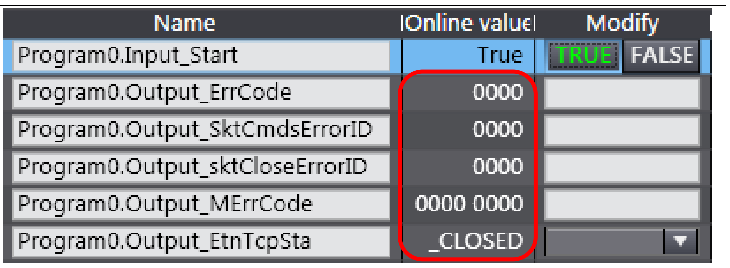


4	Confirm that the variables shown on the right are displayed in the Name Columns. *To add a variable, click *Input Name…* *Program0 of the Name is omitted from the following descriptions.	 Name Program0.Input_Start Program0.Output_ErrCode Program0.Output_SktCmdsErrorID Program0.Output_sktCloseErrorID Program0.Output_MErrCode Program0.Output_EtnTcpSta Program0.ETN_SendMessageSet_instance.Send_Data Program0.Output_RecvMess Program0.Local_Status Start input Error codes TCP connection status Program execution status Receive data Send data
5	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 |

6

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

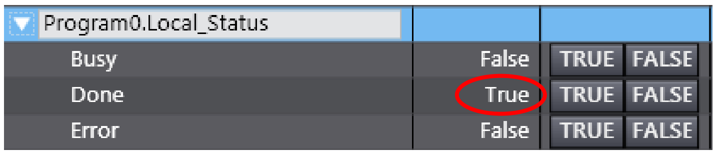
*In the case of error end, the error code corresponding to the error is stored. For details on error codes, refer to *9.7 Error Process*.



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

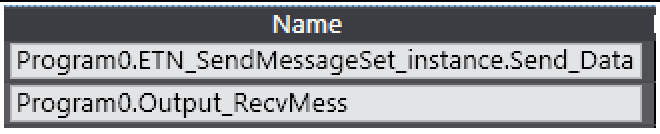
7

The response data received from the destination device is stored in *Output_RecvMess*.
(*ETN_SendMessageSet_instance.Send_Data* is the send command.)

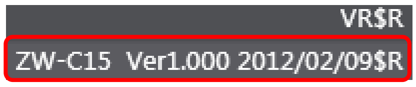
Specify an area where you want to reference in the Watch Tab Page1 as shown in the right figure.

*The response data differs depending on the device used.

*Refer to *9.2. Destination Device Command* for details on the command.



Name
Program0.ETN_SendMessageSet_instance.Send_Data
Program0.Output_RecvMess



VR$R
ZW-C15 Ver1.000 2012/02/09$R

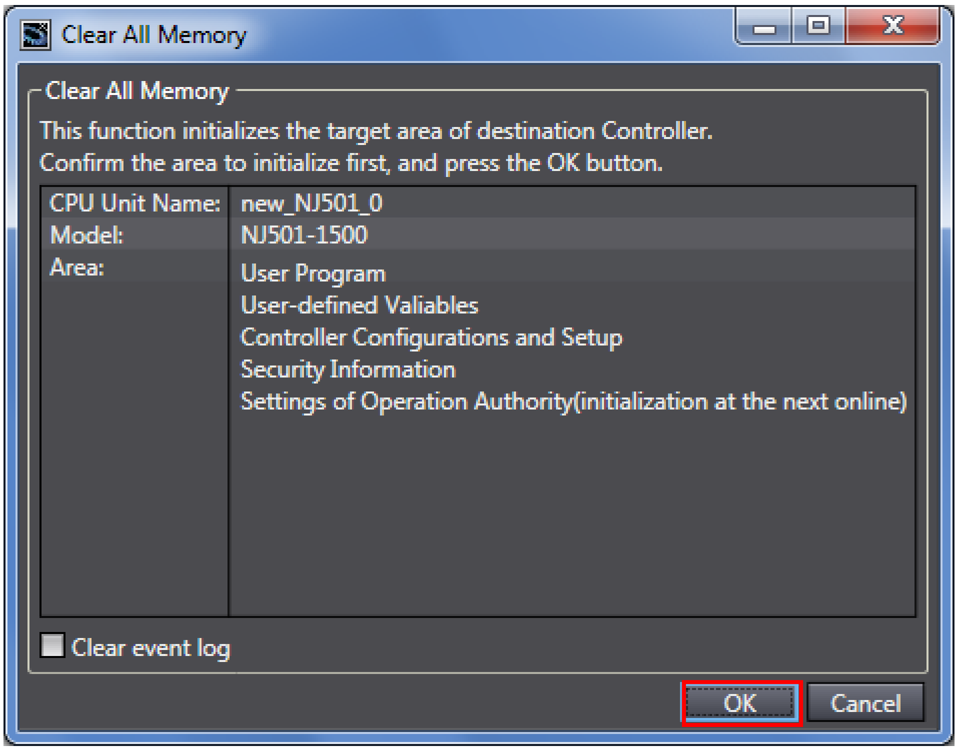
Receive data
- Version information
  - Product type: ZW-C15
  - Blank: (2 characters)
  - Version: Ver1.000
  - Blank: (1 character)
  - Release date: 2012/02/09
- Terminator: “$R”([CR])

8. Initialization Method

This document explains the setting procedure from the factory default setting.
Some settings may not be applicable as described in this document unless you use the devices with the factory default setting.

8.1. Initializing the Controller

To initialize the settings of the Controller, place the operating mode to PROGRAM mode, and select ***Clear All Memory*** from the Controller Menu of the Sysmac Studio. The Clear All Memory Dialog Box is displayed. Click the **OK** Button.



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. Initializing the Displacement Sensor

For the initialization of the Displacement Sensor, refer to *Initializing Settings* in *Setting the System* in *Chapter 3 SETTINGS FOR FUNCTIONS* of the *Confocal Fiber Type Displacement Sensor User's Manual* (Cat. No. *Z322*).

9. Program

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

9.1. Overview

This section explains the specifications and functions of the program used to check the connection between the Displacement Sensor (ZW series) (hereinafter referred to as the destination device) and the Controller (built-in EtherNet/IP port).

This program uses the socket service functions of the Controller to send and receive the “VR (read version information)” command to/from the destination device and to detect a normal end or an error end.
A normal end of this program means a normal end of the TCP socket communications.
An error end means an error end of the TCP socket communications and an error end of the destination device (detected with the response data from the destination device).

In this section, the prefix “10#" (possible to omit) is added to decimal data and the prefix "16#" to hexadecimal data when it is necessary to distinguish between decimal and hexadecimal data. (e.g., “1000” or “10#1000” for decimal data and “16#03E8” for hexadecimal data, etc.)
Also, to specify a specific data type, the prefix “<data type>#” is added. (e.g., “WORD#16#03E8”)

Additional Information

OMRON has confirmed that normal communications can be performed using this program 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 or the performance variation of the device.

9.1.1. Communications Data Flow

The following figure shows the data flow from when the Controller issues command data with TCP socket communications to the destination device until when the Controller receives the response data from the destination device. This program executes a series of processing from the connect processing to the close processing continuously. The receive processing is repeated when the response data is divided and multiple receive data arrive.

1.	Connect processing	The Controller issues a TCP socket connect request to the destination device, and establishes a TCP connection.
	↓	
2.	Sending a command	The Controller issues a send message (command data), which is set in the program, to the destination device.
	↓	
3.	Receiving a response	The Controller receives the receive message (response data) from the destination device and stores it in the specified internal variable (storage area).
	↓	
4.	Close processing	The Controller issues a close request to the destination device, and terminates the TCP connection.

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

9.1.2. TCP Socket Communications with Socket Service Instructions

This section explains the TCP socket service function blocks and outlines the general operation of the send/receive messages.

Additional Information

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

●Socket Service Instructions

This program uses the following 5 standard instructions to perform socket communications.

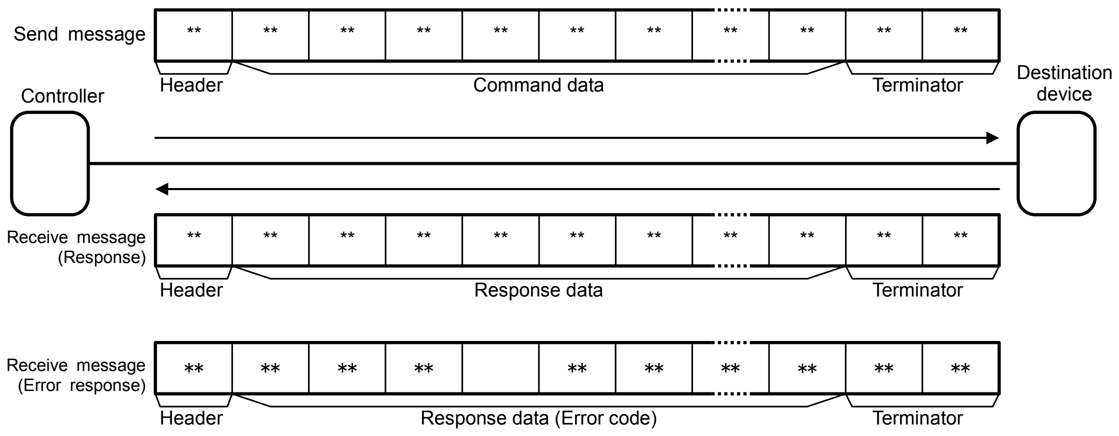
Name	Function blocks	Description
Connect TCP Socket	SktTCPConnect	Connects the TCP port of the destination device.
TCP Socket Send	SktTCPSend	Sends data from the specified TCP socket.
TCP Socket Receive	SktTCPRcv	Reads the data from the receive buffer for the specified TCP socket.
Close TCP Socket	SktClose	Closes the specified TCP socket.
Read TCP Socket Status	SktGetTCPStatus	Reads the status of the specified TCP socket. The program uses this instruction for the following 2 types of processing. (1) Processing to check the arrival of divided packets (2) Processing to check the close status at the close processing

*The socket obtained by the Connect TCP socket instruction (SktTCPConnect) 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	

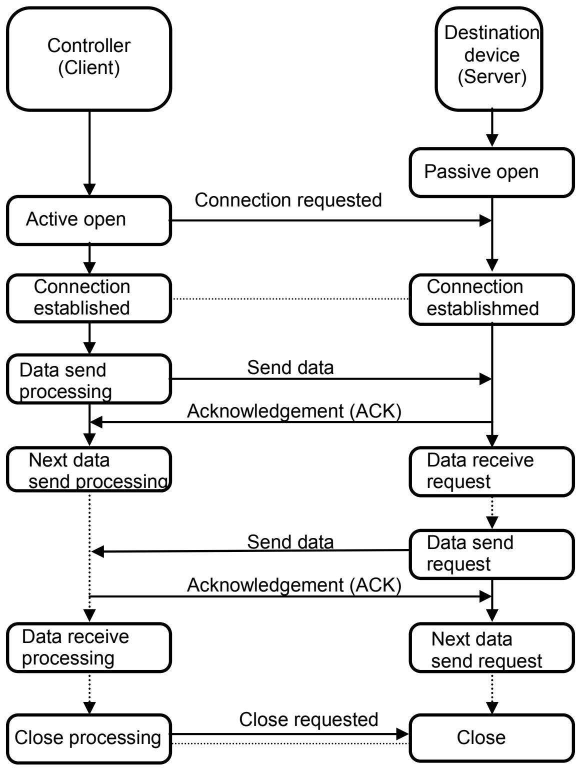
*1: The address indicates an IP address and a port number.
*2: A DNS or Hosts setting is required to use a host name.

●Send/Receive messages


Send message
**
Header
Command data
Terminator
Controller
Destination device
Receive message (Response)
Response data
Receive message (Error response)
Response data (Error code)

●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 program.

9.2.1. Overview of the Command

This program uses “VR (Version information acquisition)” command to read the controller information of the destination device.

Command	Description
VR	Acquire version information.

Additional Information

For details on the destination device command and message format, refer to *Command format* in *Chapter 5 Ethernet/RS-232C COMMUNICATION* of the *Confocal Fiber Type Displacement Sensor User's Manual* (Cat. No. Z322).

9.2.2. Detailed Description of the Command

This section explains the “VR (Version information acquisition)” command.

●Command format of the send message

This is the command format of the message that is sent by the Controller to the destination device according to the setting of the “VR (Version information acquisition)” command.

•ASCII codes are sent except for the terminator.

Data name	Number of bytes	Remarks
Header	-	None
Command	2	Fixed: ”VR”
Terminator	1	Fixed: [CR](16#0D) (Default)

●Command format of the receive message (normal)

This is the response format of the normal message received by the Controller from the destination device according to the setting of the “VR (Version information acquisition)” command.

•ASCII codes are received except for the terminator.

•The version information differs depending on the Displacement Sensor used.

Command		Number of bytes	Remarks
Header		-	None
Version information		-	-
	Product type	6	”ZW-C15”
	Blank	2	Fixed: ” “
	Version	8	“Ver1.000”
	Blank	1	Fixed: ” “
	Release data	10	“2012/02/09”
Terminator		1	Fixed: [CR](16#0D) (Default)

●Command format of the receive message (error)

This is the response format of the error message received by the Controller from the destination device according to the setting of the “VR (Version information acquisition)” command.

•ASCII codes are received except for the terminator.

Command	Number of bytes	Remarks
Error code	2	Fixed: ”ER”(16#4552)
Terminator	1	Fixed: [CR](16#0D) (Default)

9.2.3. Command Settings

This section explains the details on the "VR (Version information acquisition)" command settings.

●Send data (command) settings

The send data is set in the SendMessageSet function block.

Variable	Contents (Data type)	Set value
Send_Header	Send header (STRING[5])	‘’(Setting unnecessary)
Send_Addr	Send address (STRING[5])	‘’(Setting unnecessary)
Send_Command	Send data (STRING[256])	‘VR’
Send_Check	Addition of send check (STRING[5])	‘’(Setting unnecessary)
Send_Terminate	Send terminator (STRING[5])	‘$R’ ([CR]: 16#0D)

Variable	Contents (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.

●Receive data (response) that is stored

After a data check is performed on the receive data using the ReceiveCheck function block, the receive data is stored in the receive data storage area.

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

●Send/Receive messages

*Send message

56	52	0D
V	R	[CR]
Command		Terminator

*Receive message (at normal process)

5A	57	2D	43	31	35	20	20
Z	W	-	C	1	5	[SP]	[SP]
Product type						Blank	Blank

56	65	72	31	2E	30	30	30	20
V	e	r	1	.	0	0	0	[SP]
Version								Blank

32	30	31	32	2F	30	32	2F	30	39	0D
2	0	1	2	/	0	2	/	0	9	[CR]
Release data										Terminator

*Receive message (at error process)

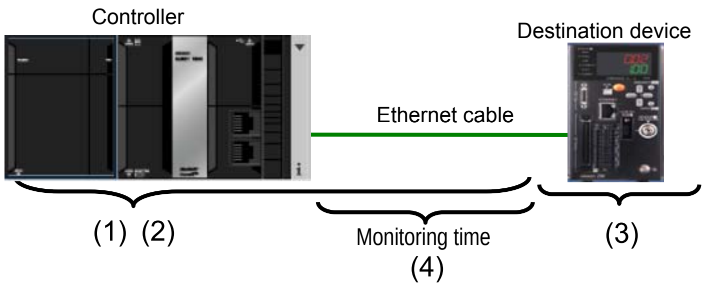
45	52	0D
E	R	[CR]
Error code		Terminator

9.3. Error Detection Processing

This section explains the error detection processing of this program.

9.3.1. Error Detection in the Program

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



(1)Communications errors in TCP socket communications using socket service instructions
Errors occurred in the program during TCP socket communications such as a command format error and a parameter error are detected as communications errors. An error is detected with the socket service instruction argument *ErrorID*.

(2)Timeout errors during communication with the destination device
When the connect 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. An error is detected with the timer in the program. For information on the time monitoring function of the timer in the program, refer to *9.3.2. Time Monitoring Function*.

(3)Errors in the destination device (Destination device error)
The destination device errors include a command error, a parameter error, and an execution failure in the destination device. An error is detected with "ER" (16#4552) which is returned from the destination device when an error occurs.

(4)TCP connection status error that occurs when ending the processing
This program always performs the close processing at the end of the whole processing regardless of whether each processing from the connect processing to the receive processing ends normally or in an error. When the close processing is operated abnormally, the next connect processing may not be performed normally. The TCP connection status variable *TcpStatus* of the SktGetTCPStatus instruction is used to detect whether the close processing ends normally. For the corrective action of the TCP connection status error, refer to *9.7.2 TCP Connection Status Error and Corrective Action*.

9.3.2. Time Monitoring Function

This section explains the time monitoring function of this program.

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

●Time monitoring function of the communication instruction processing

To avoid the status that keeps executing a communications process without stop due to abnormality, the timer in this program is used to abort the processing (timeout). The timeout value for each processing from the connect processing to the close processing is 5 seconds.

[Monitoring time of the communications instruction processing]

Processing	Monitoring	Variable name	Timeout time
Connect processing	Time from the start to the end of the processing	TopenTime	5 seconds (UINT#500)
Send processing	Time from the start to the end of the processing	TfsTime	5 seconds (UINT#500)
Receive processing	Time from the start to the end of the processing (Each receive processing)	TfrTime	5 seconds (UINT#500)
Close processing	Time from the start of the processing until the TCP socket enters the close status.	TcloseTime	5 seconds (UINT#500)

●Receive waiting function for divided packets/multiple response data

To repeat the receive processing, this function enables waiting for multiple responses that arrive continuously or the receive data that is divided. The timer of the program monitors the interval at which to receive the data. Then, if the next packet does not arrive from the destination device within the set time (300 ms: default) after each receive processing (If the receive data arrives at an interval of more than 300 ms), repeating the receive processing will end.

[Receive wait monitoring time]

Processing	Monitoring	Variable name	Maximum waiting time
TCP socket status read processing	Time from when SktGetTCPStatus_instance.DatRcvFlag is turned OFF until when the flag is turned ON.	TrTime	300 milliseconds (UINT#3)

●Resend/time monitoring function of 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 processing ends in an error, this program performs the close processing and stops the TCP/IP resend/time monitoring function.

*If a TCP connection status error occurs at the close processing, the TCP/IP resend/time monitoring function may still be operating. For information on the situation and corrective action, refer to *9.7.2. TCP Connection Status Error and Corrective Action*.

9.4. Variables

The table below lists the variables used in this program.

9.4.1. List of Variables

The data types, external variables (user-defined global variables/system-defined variables), and internal variables used in this program are listed below.

●Data type (Structure)

[Communications processing status flags]

Name		Data type	Description
sStatus		STRUCT	Structure of communications processing status flags
	Busy	BOOL	Communications processing in progress flag TRUE: Processing is in progress. FALSE: Processing is not in progress.
	Done	BOOL	Communications processing normal end flag TRUE: Normal end / FALSE: Other than normal end
	Error	BOOL	Communications processing error end flag TRUE: Error end / FALSE: Other than error end

[Socket service instruction execution flags]

Name		Data type	Description
sControl		STRUCT	Socket service instruction execution flags
	Send	BOOL	Send processing instruction TRUE: Executed / FALSE: Not executed
	Recv	BOOL	Receive processing instruction TRUE: Executed / FALSE: Not executed
	Open	BOOL	Connect processing instruction TRUE: Executed / FALSE: Not executed
	Close	BOOL	Close processing instruction TRUE: Executed / FALSE: Not executed
	Status	BOOL	TCP socket status read processing instruction TRUE: Executed / FALSE: Not executed

[Timer enable flags]

Name		Data type	Description
sTimerControl		STRUCT	Time monitoring timer enable flags
	Tfs	BOOL	Send processing time monitoring timer instruction TRUE: Enabled / FALSE: Not enabled
	Tfr	BOOL	Receive processing time monitoring timer instruction TRUE: Enabled / FALSE: Not enabled
	Topen	BOOL	Connect processing time monitoring timer instruction TRUE: Enabled / FALSE: Not enabled
	Tclose	BOOL	Close processing time monitoring timer instruction TRUE: Enabled / FALSE: Not enabled
	Tr	BOOL	Receive waiting time monitoring timer instruction TRUE: Enabled / FALSE: Not enabled

[Send/Receive processing required/not required setting flag]

Name	Data type	Description
sComType	STRUCT	Send/Receive processing required/not required setting flags
Send	BOOL	Send processing TRUE: Required / FALSE: Not required *Specify this when sending a command.
Recv	BOOL	Receive processing TRUE: Required / FALSE: Not required *Specify this when receiving a response.
Error	BOOL	Send/Receive processing required/not required setting error flag (This flag changes to ON when a setting error occurred.)

●Data type (Union)

[Error code processing]

Name	Data type	Description
uErrorFlags	UNION	Union for error code processing
BoolData	ARRAY[0..15] OF BOOL	2-byte error code is processed in units of 1 bit as 16-bit string. : TRUE (Error) / FALSE (Normal) •Communications error BoolData[0]: Send processing BoolData[1]: Receive processing BoolData[2]: Connect processing BoolData[3]: Close processing BoolData[4]: Processing number error •Timeout error BoolData[8]: Send processing BoolData[9]: Receive processing BoolData[10]: Connect processing BoolData[11]: Close processing •Others BoolData[5]: Send/Receive required/not required detection error BoolData[12]: Destination device error BoolData[6..7],[13..14]: Reserved BoolData[15]: Error
WordData	WORD	2-byte error code is processed as WORD at once.

●External variables

[User-defined global variables]

Variable name	Data type	Description
Input_Start	BOOL	Communications start switch The program starts when this flag changes from FALSE to TRUE.
Output_RecvMess	STRING[256]	An area that stores the receive data (response) (256 bytes)
Output_ErrCode	WORD	An area that stores the error flag for a communications error or a timeout error that is detected at the connect processing, TCP socket status read processing, receive processing or close processing. Normal end: 16#0000
Output_SktCmdsErrorID	WORD	An area that stores the error code for a communications error or a timeout error that is detected at the connect processing, TCP socket status read processing and receive processing. Normal end: 16#0000
Output_SktCloseErrorID	WORD	An area that stores the error code for a communications error or a timeout error that is detected at the close processing. Normal end: 16#0000
Output_EtnTcpSta	_eCONNECTION_STATE	An area that stores the TCP socket status _ESTABLISHED: Connect status _CLOSED: Close status
Output_MErrCode	DWORD	An area that stores the destination device's error code for an FCS error or a destination device error that is detected after the receive processing. Normal end: 16#00000000

[System-defined variable]

Variable name	Data type	Description
_EIP_EtnOnlineSta	BOOL	Status of built-in EtherNet/IP port communications function TRUE: Can be used, FALSE: Cannot be used

Additional Information

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

●Internal variables (instance variables)

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

[Instances for user-defined function blocks]

Variable name	Data type	Description
ETN_ParameterSet_instance	ParameterSet	Ethernet communications parameter setting function block This variable sets a destination IP address and monitoring time for each processing from the connect processing to the close processing.
ETN_SendMessageSet_instance	SendMessageSet	Ethernet communications send data setting function block This variable sets the send/receive processing required/not required setting and send data.
ETN_ReceiveCheck_instance	ReceiveCheck	Ethernet communications receive processing function block This variable stores the receive data and detects a normal end or an error end.

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

[Instances for timer]

Variable name	Data type	Description
SktTCPConnect_instance	SktTCPConnect	Connect TCP socket function block
SktTCPSend_instance	SktTCPSend	TCP socket send function block
SktTCPRcv_instance	SktTCPRcv	TCP socket receive function block
SktClose_instance	SktClose	Close TCP socket function block
SktGetTCPStatus_instance	SktGetTCPStatus	Read TCP socket status function block

Additional Information

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

[Instances for timer]

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

●Internal variables

Variable name	Data type	Description
Local_Status	sStatus	Communications processing status flags This variable is defined as sStatus structure.
Local_State	DINT	Processing number
Local_ErrCode	uErrorFlgs	An area in which an error code is edited. This variable is defined as uErrorFlgs union.
Local_ExecFlgs	sControl	Socket service instruction execution flags This variable is defined as sControl structure.
Local_SrcDataByte	UINT	The number of bytes of send data
Local_SrcData	ARRAY[0..255] OF BYTE	An area that stores the send data of the SktTCPSend instruction (256 bytes)
Local_RecvData	ARRAY[0..2000] OF BYTE	An area that stores the receive data of the SktTCPRcv instruction (2001 bytes)
Local_ReceiveMessage	STRING[256]	An area that stores the receive data after converted into a string. (256 characters)
Local_ReceiveSize	UINT	The size of the receive data of SktTCPRcv instruction
Local_RecvDataLength	UINT	The total byte length of the receive data
Local_RecvCHNo	UINT	The element number of Local_RecvData that stores the receive data
Local_RecvCheckFlg	BOOL	Destination device error detection instruction execution flag TRUE: Executed / FALSE: Not executed
Local_InitialSettingOK	BOOL	Initialization processing normal setting flag
Local_TONFlgs	sTimerControl	Timer enable flags This variable is defined as sTimerControl structure.
Local_ComType	sComType	Send/Receive processing required/not required setting flags This variable is defined as sControl structure.

9.5. ST Program

9.5.1. Functional Components of the 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 the communications processing 1.2. Clearing the communications processing status flags 1.3 Communications processing in progress status	The communications processing starts.
2. Initialization processing	2.1. Initializing the timer 2.2. Initializing the instructions 2.3. Initializing the instruction execution flags 2.4. Initializing the timer enable flags 2.5. Initializing the error code storage areas 2.6. Setting each processing monitoring time and Ethernet communications parameters 2.7. Setting the send/receive processing required/not required setting and send data 2.8. Converting the send data from a string to BYTE array 2.9. Initializing the receive data storage areas 2.10. Initialization setting end processing	The parameters of Ethernet communications are set and the error code storage areas are initialized. The send/receive required/not required setting, the send data and receive data are set.
3. Connect processing	3.1. Determining the connect processing status and setting the execution flag 3.2. Enabling the connect instruction monitoring timer 3.3. Executing the connect instruction	The connect processing is performed. The processing starts unconditionally after starting the communications processing and executing the initialization setting.
4. Send processing	4.1. Determining the send processing status and setting the execution flag 4.2. Enabling the send instruction monitoring timer 4.3. Executing the send instruction	The processing starts when the send processing is required and the connect processing ends 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 instruction monitoring timer 5.4 Executing the receive instruction 5.5 Executing the TCP socket status read processing 5.6 Executing the destination device error detection instruction	The processing starts when the receive processing is required and the send processing ends normally. If multiple receive data arrive, the receive processing is repeated. The receive data is stored and checked.
6. Close processing	6.1. Determining the close processing status and setting the execution flag 6.2. Enabling the close instruction monitoring timer 6.3. Executing the close instruction 6.4. Executing the read TCP socket status processing	The close processing is performed. The processing starts in the following cases. •When the receive processing is not required, and the send processing ends normally •When the receive processing ends normally •When any of the connect processing, send processing or receive processing ends in error

Major classification	Minor classification	Description
7. Processing number error process	7. Processing number error process	The error process is executed if a non-existent processing number is detected.

9.5.2. Program List

This section shows the program.

The function blocks (ParameterSet, SendMessageSet, and ReceiveCheck) are used to perform the communications settings, send data (command data) setting and receive data (response data) check that must be changed according to the destination device. To change these values, refer to *9.5.3 Detailed Description of Function Blocks*.

●Program: Program0

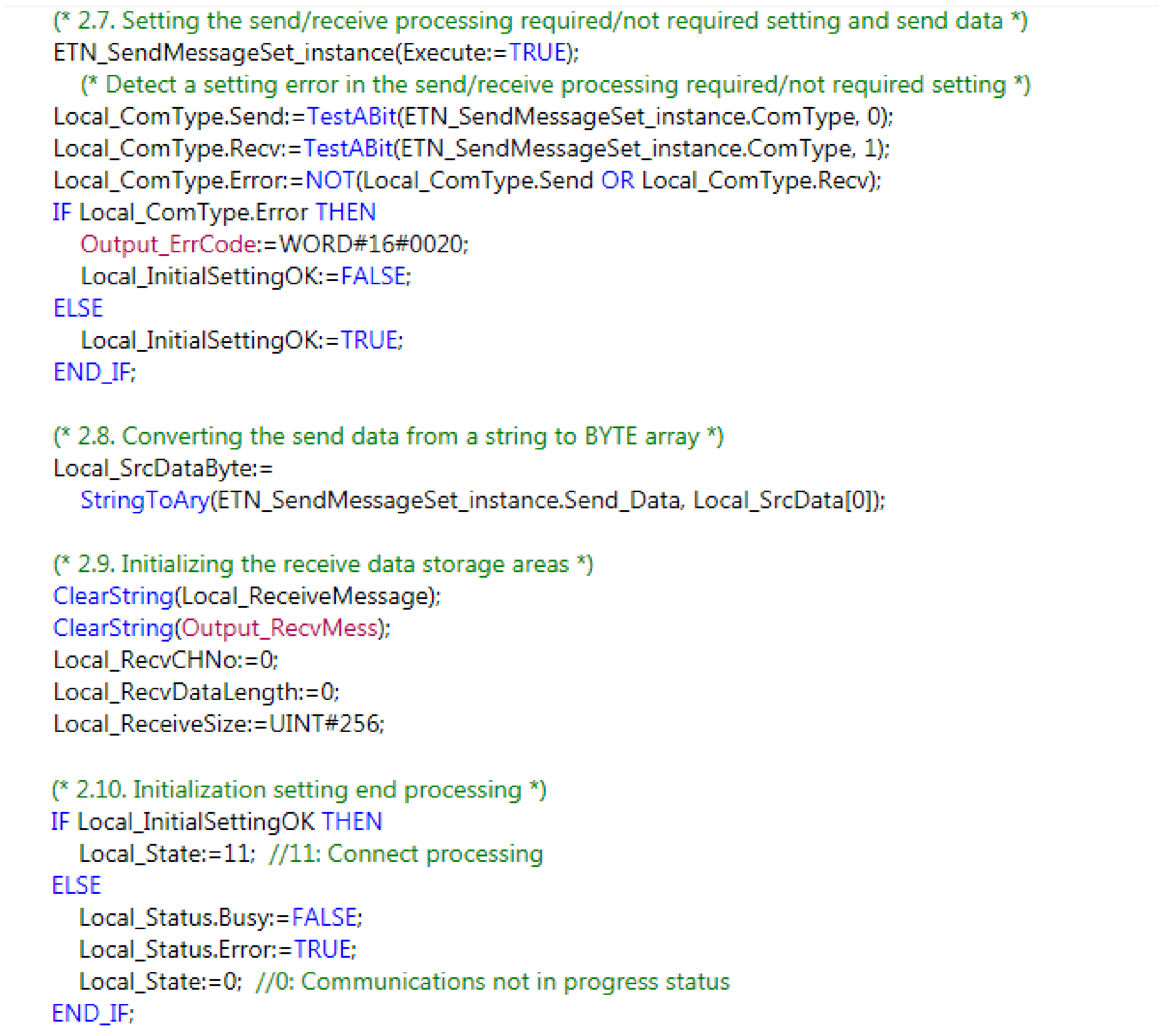
(General-purpose Ethernet communications connection check program)

1. Communications processing

```
(*============================================================
   Name: NJ-series Ethernet communications connection check program
   Version: V1.00 New release 29 November 2012
            V1.01 Update 19 February 2013
   (C)Copyright OMRON Corporation 2012 All Rights Reserved.
 ============================================================ *)

(* 1. Communications processing
    Communications start switch: Input_Start
    Communications processing status flags: Local_Status<STRUCT>
     .Busy: Communications in progress
     .Done: Communications normal end
     .Error: Communications error end
   Processing number: Local_State
    10: Initialization processing
    11: Connect processing
    12: Send processing
    13: Receive processing
    14: Close processing *)

(* 1.1. Starting the communications processing
   Start the communications processing when the communications start switch changes to ON
   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;  //10: Initialization processing
END_IF;

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

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

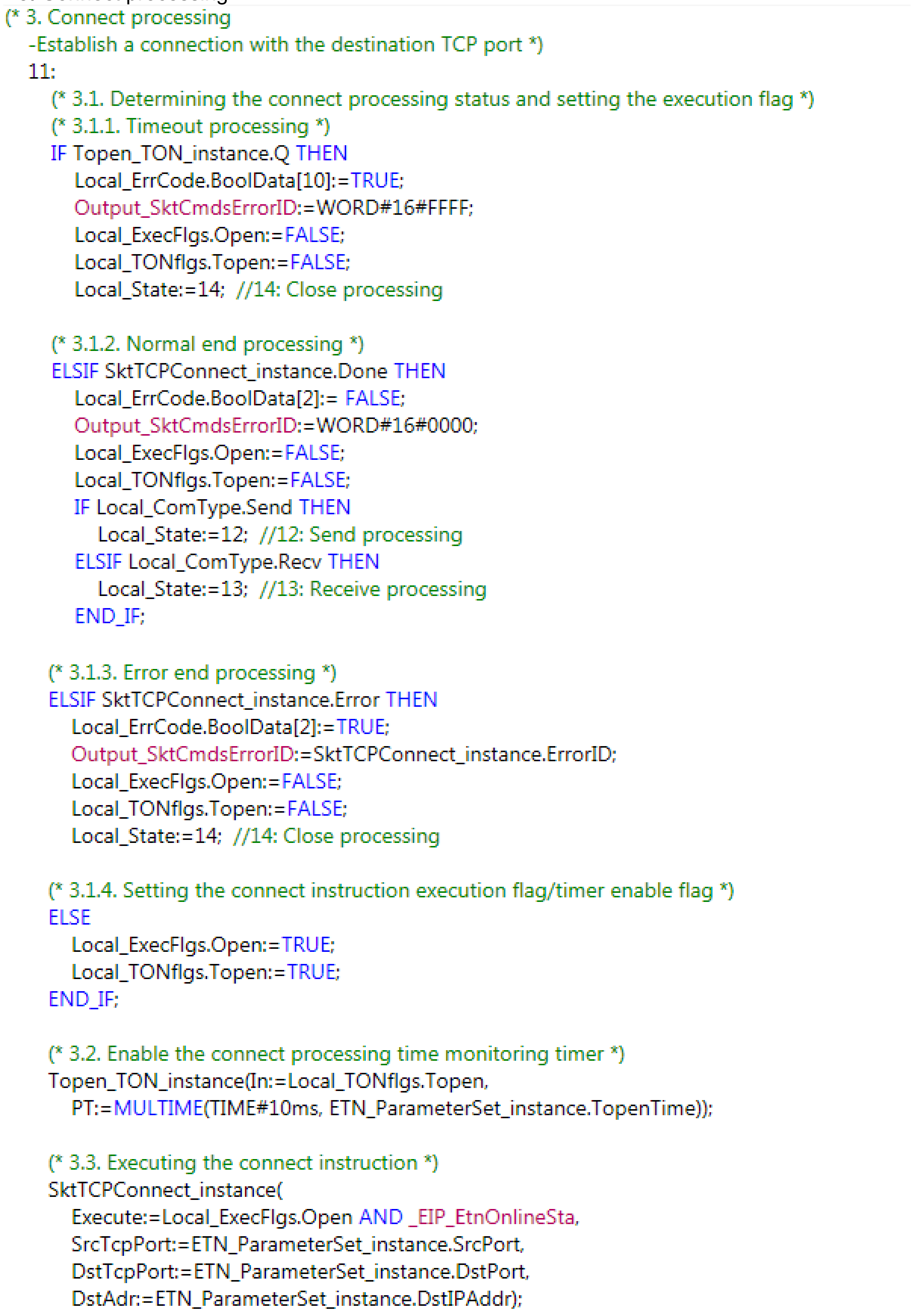
2. Initialization processing

```
(* 2. Initialization processing
   -Perform initialization for the whole communications and set the parameters.
   -Set the send data and initialize the receive data storage area. *)
   10:
      (* 2.1. Initializing the processing time monitoring timer *)
      Topen_TON_instance (In:=FALSE);
      Tfs_TON_instance (In:=FALSE);
      Tr_TON_instance (In:=FALSE);
      Tfr_TON_instance (In:=FALSE);
      Tclose_TON_instance(In:=FALSE);

      (* 2.2. Initializing the socket service instructions *)
      SktTCPConnect_instance(Execute:=FALSE);
      SktTCPSend_instance(Execute:=FALSE, SendDat:=Local_SrcData[0]);
      SktTCPRcv_instance(Execute:=FALSE, RcvDat:=Local_RecvData[0]);
      SktClose_instance(Execute:=FALSE);
      SktGetTCPStatus_instance(Execute:=FALSE);

      (* 2.3. Initializing the socket service instruction execution flags *)
      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 *)
      Local_TONflgs.Tfs:=FALSE;
      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 setting the 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 a setting error in the send/receive processing required/not required setting *)
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 the send data from a string to 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;  //11: Connect processing
ELSE
   Local_Status.Busy:=FALSE;
   Local_Status.Error:=TRUE;
   Local_State:=0;  //0: Communications not in progress status
END_IF;
```

3. Connect processing



```
(* 3. Connect processing
    -Establish a connection with the destination TCP port *)
    11:
        (* 3.1. Determining the connect 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;  //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;
            IF Local_ComType.Send THEN
                Local_State:=12;  //12: Send processing
            ELSIF Local_ComType.Recv THEN
                Local_State:=13;  //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;  //14: Close processing

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

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

        (* 3.3. Executing the connect 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

```
(* 4. Send processing
  -Send data from the specified TCP port. *)
  12:
    (* 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;  //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;
      Local_State:=SEL(Local_ComType.Recv,14,13);  //13: Receive processing/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;  //14: Close processing

    (* 4.1.4. Setting the send instruction execution flag/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 *)
    SktTCPSend_instance(
      Execute:=Local_ExecFlgs.Send AND _EIP_EtnOnlineSta,
      Size:=Local_SrcDataByte,
      Socket:=SktTCPConnect_instance.Socket,
      SendDat:=Local_SrcData[0]);
```

5. Receive processing

```
(* 5. Receive processing
  -Read the data from the receive buffer of the specified TCP socket. *)
  13:
    (* 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 the receive data from BYTE array to STRING *)
      Local_ReceiveMessage:=AryToString(Local_RecvData[0],Local_RecvDataLength);
      (* Setting the destination device error detection instruction execution flag *)
      Local_RecvCheckFlg:=TRUE;
      Local_State:=14;  //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;  //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;  // 5.1.5. Reading the receive data

    (* 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;  //14: Close processing

    (* 5.1.5. Reading the receive data *)
    ELSIF SktGetTCPStatus_instance.Done
        OR SktGetTCPStatus_instance.Error THEN
      Local_ExecFlgs.Status:=FALSE;
        (* When there is data to read: Continue 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:
          -When no data is received,
             do not perform any processing and
             execute the read TCP socket status at the next period.
          -When data has already been received,
             if there is no response when the maximum receive waiting time has elapsed
             read the data that has been already received and end the receive processing *)
```

```
(* 5.1.6. Setting the TCP status get instruction execution flag/timer enable flag *)
ELSE
    Local_ExecFlgs.Status:=TRUE;
    Local_TONflgs.Tfr:=TRUE;
        (* Initialize the destination device error detection instruction execution flag *)
    Local_RecvCheckFlg:=FALSE;
END_IF;

(* 5.2. Enabling the receive waiting time monitoring timer *)
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 *)
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 read TCP socket 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

```
(*  6. Close processing
     -Close the specified socket *)
   14:
       (* 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: Communications 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;

                (* Determine the processing result of the whole communications processing *)
                Local_Status.Busy:=FALSE;
                   (* Communications processing normal end *)
                IF Local_ErrCode.WordData = WORD#16#0000 THEN
                   Local_Status.Done:=TRUE;
                   Local_ErrCode.BoolData[15]:=FALSE;
                   (* Communications processing error end *)
                ELSE
                   Local_Status.Error:=TRUE;
                   Local_ErrCode.BoolData[15]:=TRUE;
                END_IF;
                Output_ErrCode:=Local_ErrCode.WordData;
                Local_State:=0;  //0: Communications 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: Communications not in progress status

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

(* 6.2. Executing 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 *)
SktClose_instance(Execute:=Local_ExecFlgs.Close AND _EIP_EtnOnlineSta,
   Socket:=SktTCPConnect_instance.Socket);

(* 6.4. Executing the read TCP socket status instruction *)
SktGetTCPStatus_instance(
   Execute:=Local_ExecFlgs.Status AND _EIP_EtnOnlineSta,
   Socket:=SktTCPConnect_instance.Socket);
```

7. Processing number error process

```
(* 7. Processing number error process
   -Error process for nonexistent processing number *)
   99:
      Output_ErrCode:=WORD#16#0010;
      Local_Status.Busy:=FALSE;
      Local_Status.Error:=TRUE;
      Local_State:=0;  //0: Communications not in progress status

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

   END_CASE;

END_IF;
```

9.5.3. Detailed Description of Function Blocks

The user-defined function blocks are shown below.

The code which you need to edit according to the destination device is indicated by the red frames on the function blocks below.

●ParameterSet function block

(General-purpose Ethernet communications parameter setting)

Instruction	Meaning	ST expression
ParameterSet	General-purpose Ethernet Communications parameter setting	ETN_ParameterSet_instance(Execute, TopenTime, TfsTime, TrTime, TfrTime, TcloseTime, SrcPort, DstIPAddr, DstPort);

[Internal variable]

None

[Input/Output]

Name	I/O	Data type	Description	
Execute	Input	BOOL	Execution flag: The function block is executed when this variable changes to TRUE and it is stopped when this variable changes to FALSE.	
TopenTime	Output	UINT	Connect processing monitoring time: This variable sets the monitoring time of the connect processing in increments of 10 ms.	
TfsTime	Output	UINT	Send processing monitoring time: This variable sets the monitoring time of the send processing in increments of 10 ms.	
TrTime	Output	UINT	Receive wait monitoring time: This variable sets the waiting time for the receive data in increments of 100 ms.	
TfrTime	Output	UINT	Receive processing monitoring time: This variable sets the monitoring time of the receive processing in increments of 10 ms.	
TcloseTime	Output	UINT	Close processing monitoring time: This variable sets the monitoring time of the close processing in increments of 10 ms.	
SrcPort	Output	UINT	Local port number: This variable sets the local port number.	
DstIPAddr	Output	STRING[256]	Destination IP address: This variable sets the destination IP address.	
DstPort	Output	UINT	Destination port number: This variable sets the destination port number.	
Busy	Output	BOOL	Busy	Not used (Not used in this program.)
Done	Output	BOOL	Normal end	
Error	Output	BOOL	Error end	
ErrorID	Output	WORD	Error code	
ErrorIDEx	Output	DWORD	Expansion error code	

[External variable]

None

[Program]

```
(* ==============================================================
   Name: NJ-series general-purpose Ethernet communications parameter setting function block
   Applicable device: OMRON Corporation ZW-series Displacement Sensor
   Version: V1.00 New release 29 November 2012
                V1.01 Update 19 February 2013
   (C)Copyright OMRON Corporation 2012 All Rights Reserved.
 ============================================================== *)

IF Execute THEN
   (* Set the Ethernet-related parameters *)
   SrcPort:= UINT#0;                  // Local port No.
   DstIPAddr:= '192.168.250.2';    // Destination IP address
   DstPort:= UINT#9600;              // Destination port No.

   (* Set the processing monitoring time: Maximum time from the start to end of the processing *)
   TopenTime := UINT#500;  // Connect monitoring time setting: Setting unit 10ms<500->5s>
   TfsTime:= UINT#500;       // Send monitoring time setting: Setting unit 10ms<500->5s>
   TfrTime:= UINT#500;       // Receive monitoring time setting: Setting unit 10ms<500->5s>
   TcloseTime:=UINT#500;   // Close monitoring time setting: Setting unit 10ms<500->5s>

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

END_IF;

RETURN;
```

●SendMessageSet function block

(General-purpose Ethernet communications send data setting)

Instruction	Meaning	ST expression
SendMessageSet	General-purpose Ethernet communications send data setting	ETN_SendMessageSet_instance(Execute, Send_Data, ComType);

[Internal variables]

Name	Data type	Description
Send_Header	STRING[5]	Send header: Header of the send message
Send_Addr	STRING[5]	Destination device address: Address of the destination device
Send_Command	STRING[256]	Destination device command: Command sent to the destination device
Send_Check	STRING[5]	Send check code: Check code of the send message
Send_Terminate	STRING[5]	Send terminator: Terminator of the send message

[Input/Output]

Name	I/O	Data type	Description	
Execute	Input	BOOL	Execution flag: The function block is executed when this variable changes to TRUE and it is stopped when this variable changes to FALSE.	
Send_Data	Output	STRING[256]	Send data: This variable sets a command that is sent to the destination device.	
ComType	Output	BYTE	Send/Receive type: This variable sets whether send/receive processing are required. 1:Send only, 2: Receive only, 3: Send and receive	
Busy	Output	BOOL	Busy	Not used (Not used in this project.)
Done	Output	BOOL	Normal end	
Error	Output	BOOL	Error end	
ErrorID	Output	WORD	Error code	
ErrorIDEx	Output	DWORD	Expansion error code	

[Internal variable]

None

[Program]

```
(* ============================================================
   Name: NJ-series general-purpose Ethernet communications send data setting function block
   Applicable device: OMRON Corporation ZW-series Displacement Sensor
   Version: V1.00 New release 29 November 2012
               V1.01 Update 19 February 2013
   (C)Copyright OMRON Corporation 2012 All Rights Reserved.
 ============================================================ *)

IF Execute THEN
   (* Set the send/receive processing required/not required setting *)
   ComType:= BYTE#16#03;      // 1: Send only, 2: Receive only, 3: Send/Receive

   (* Set the send data *)
   Send_Header:= '';           // Send header: None
   Send_Addr:= '';             // Destination device address: None
   Send_Command:='VR';         // Destination device command: VR
   Send_Check:='';             // FCS calculation : None
   Send_Terminate:= '$R';      // Send terminator: CR(0x0D): Fixed

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

END_IF;

RETURN;
```

●ReceiveCheck function block

(General-purpose Ethernet communications receive processing)

Instruction	Meaning	ST expression
ReceiveCheck	General-purpose Ethernet Communications receive processing	ETN_ReceiveCheck_instance(Execute, Recv_Data, Recv_Buff, Error, ErrorID, ErrorIDEx);

[Internal variables]

None

[Input/Output]

Name	I/O	Data type	Description	
Execute	Input	BOOL	Execution flag: The function block is executed when this variable changes to TRUE and it is stopped when this variable changes to FALSE.	
tLength	Input	UINT	Receive data length: The byte length of the receive data	
Recv_Data	In-out	STRING[256]	Receive data storage area: An area that stores the receive data after detection	
Recv_Buff	In-out	STRING[256]	Receive buffer: An area that temporarily stores the receive data that is used for detection.	
ErrorID	In-out	WORD	Error code: This variable stores 16#1000 for a destination device error and 16#2000 for an FCS error.	
ErrorIDEx	In-out	DWORD	Expansion error code: This variable stores the FCS determination result or the destination device error code.	
Busy	Output	BOOL	Busy	Not used (Not used in this program.)
Done	Output	BOOL	Normal end	
Error	Output	BOOL	Error end: TRUE when an error occurs.	

[External variable]

None

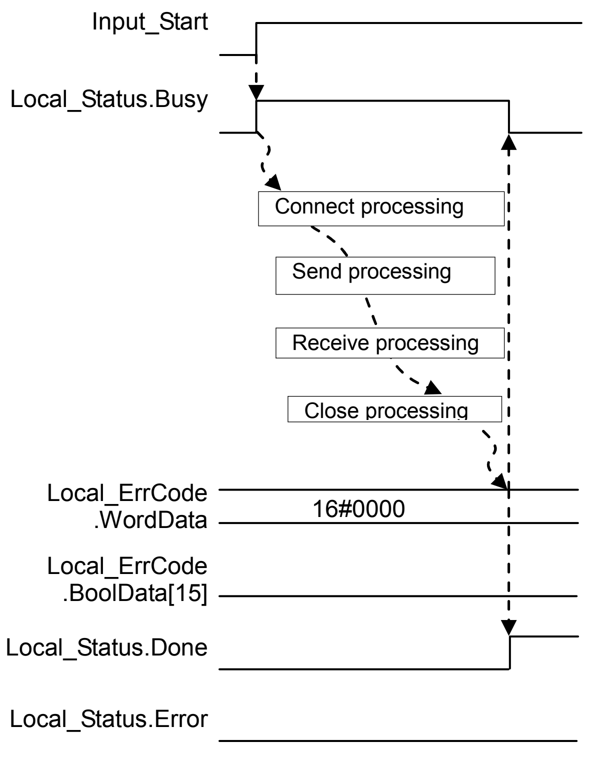
[Program]

```
(* ==============================================================
   Name: NJ-series general-purpose Ethernet communications receive processing function block
   Applicable device: OMRON Corporation ZW-series Displacement Sensor
   Version: V1.00 New release 29 November 2012
            V1.01 Update 19 February 2013
   (C)Copyright OMRON Corporation 2012 All Rights Reserved.
 ============================================================== *)

IF Execute THEN
    (* Store the receive buffer data in the receive data storage area *)
   Recv_Data:= Recv_Buff;

   (* Detect the destination device error
   (* Error: The code starts with 'ER' *)
   IF FIND(LEFT(Recv_Buff,2),'ER') = UINT#1 THEN
      Error:= TRUE;                          // Error flag set
      ErrorID:= WORD#16#1000;                // Error code set
      ErrorIDEx:= DWORD#16#45520000;         // Store the destination device error code (ER).

   (* Normal: The code does not start with 'ER'. *)
   ELSE
      Error:= FALSE;                         // Error flag reset
      ErrorID:= WORD#16#0000;                // Error code clear
      ErrorIDEx:= DWORD#16#00000000;         // Destination device error code clear

   END_IF;

END_IF;
```

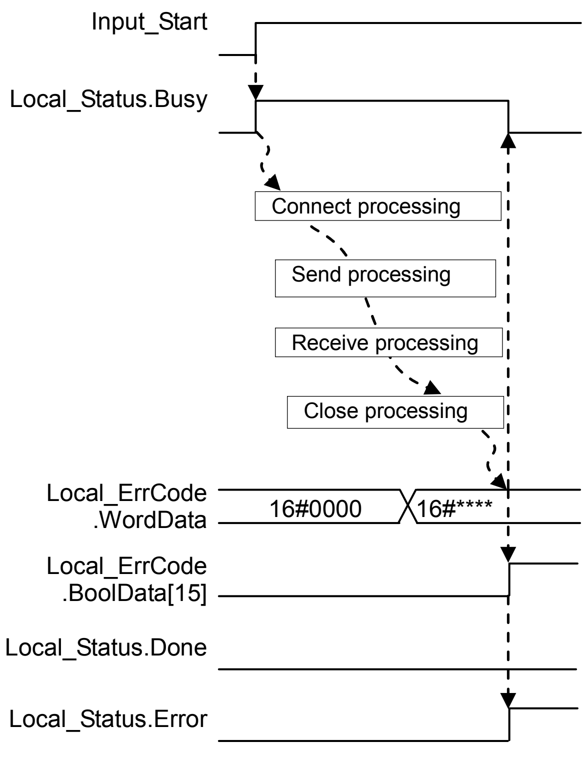
9.6. Timing Charts

The timing charts of this program are shown below.

●Start & End processing

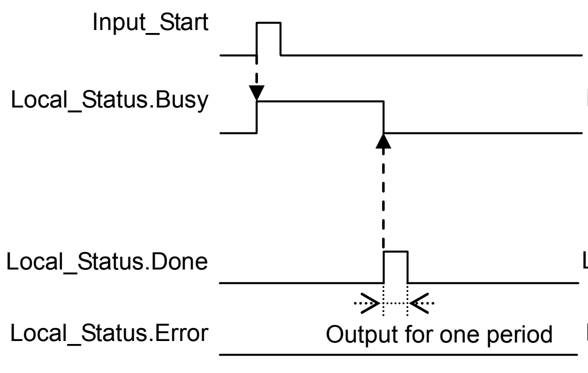


(Normal end)

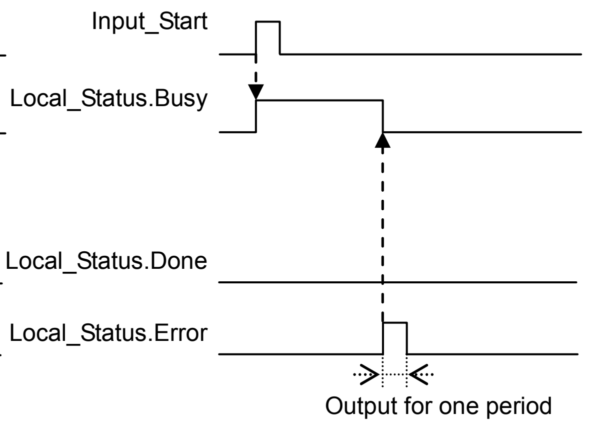


(Error end)

If *Input_Start* changes from TRUE to FALSE during execution, a normal end or an error end is output for one period after the processing is completed as shown below.

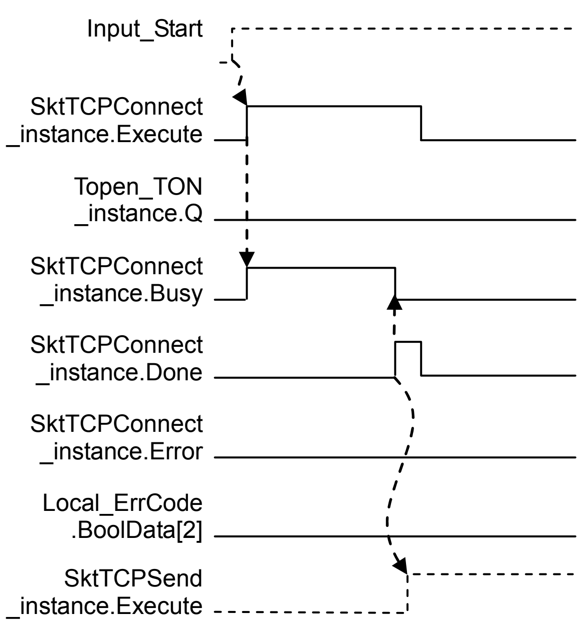


(Normal end)

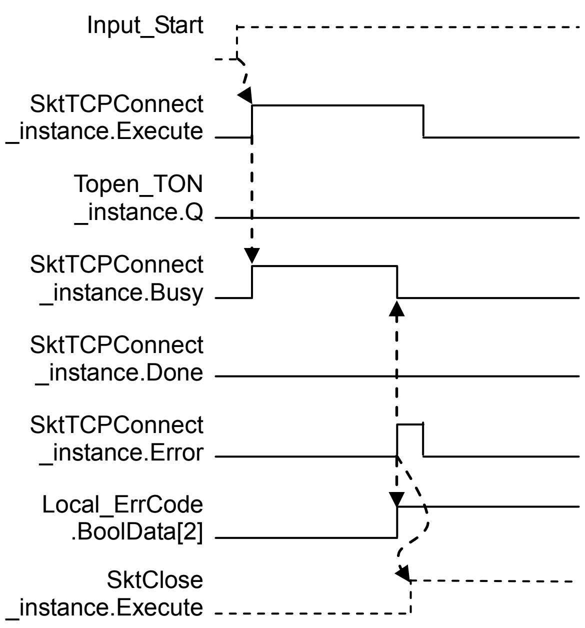


(Error end)

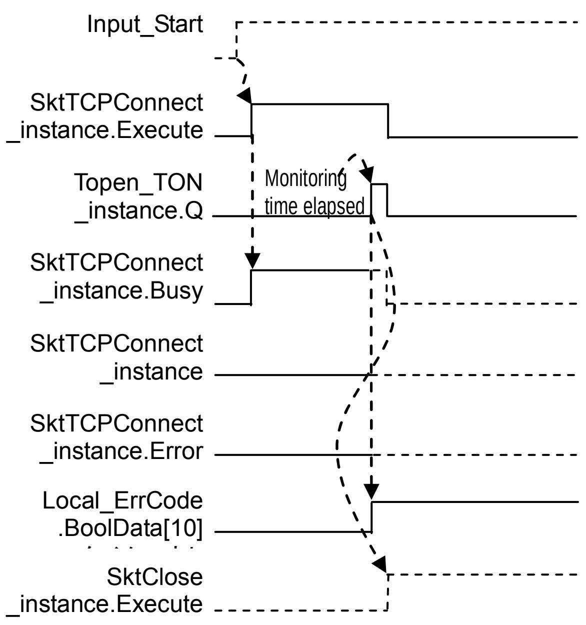
●Connect processing


Input_Start
SktTCPConnect _instance.Execute
Topen_TON _instance.Q
SktTCPConnect _instance.Busy
SktTCPConnect _instance.Done
SktTCPConnect _instance.Error
Local_ErrCode .BoolData[2]
SktTCPSend _instance.Execute

(Normal end)

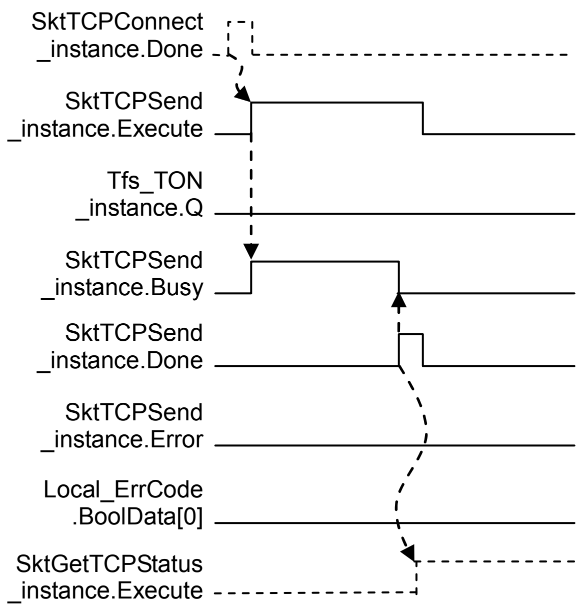

Input_Start
SktTCPConnect _instance.Execute
Topen_TON _instance.Q
SktTCPConnect _instance.Busy
SktTCPConnect _instance.Done
SktTCPConnect _instance.Error
Local_ErrCode .BoolData[2]
SktClose _instance.Execute

(Error end)

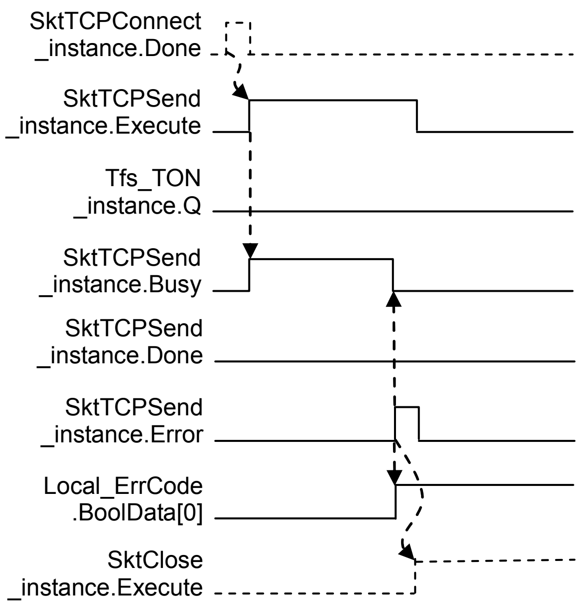

Input_Start
SktTCPConnect _instance.Execute
Topen_TON _instance.Q
Monitoring time elapsed
SktTCPConnect _instance.Busy
SktTCPConnect _instance
SktTCPConnect _instance.Error
Local_ErrCode .BoolData[10]
SktClose _instance.Execute

(Timeout)

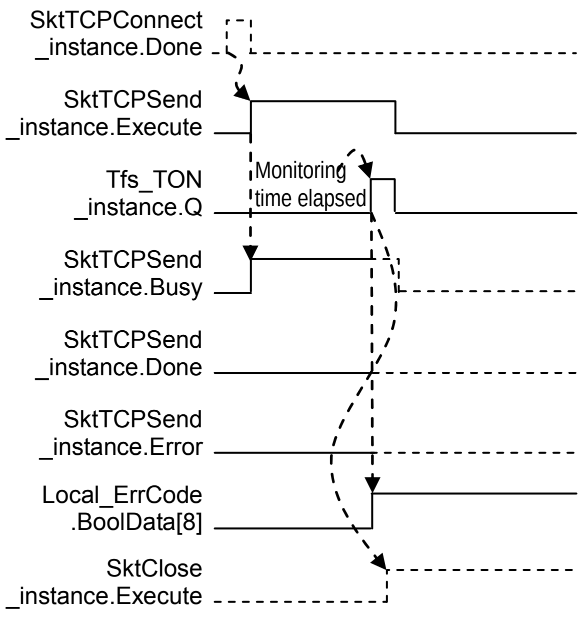
●Send processing



(Normal end)

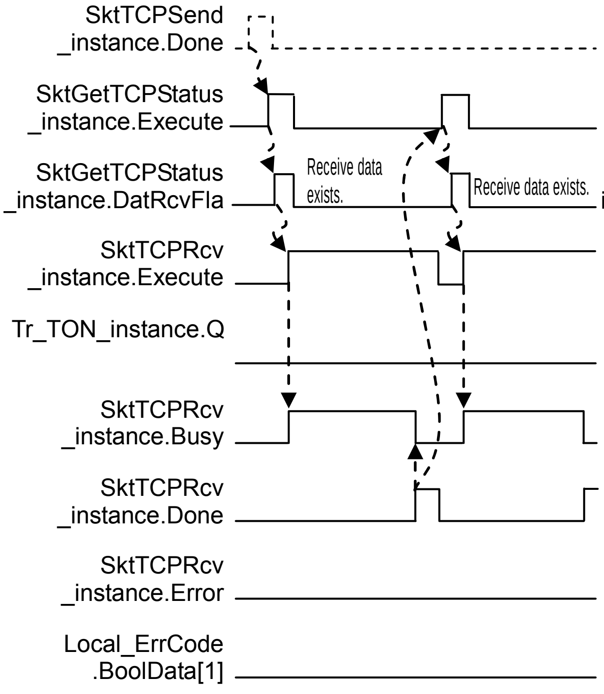


(Error end)

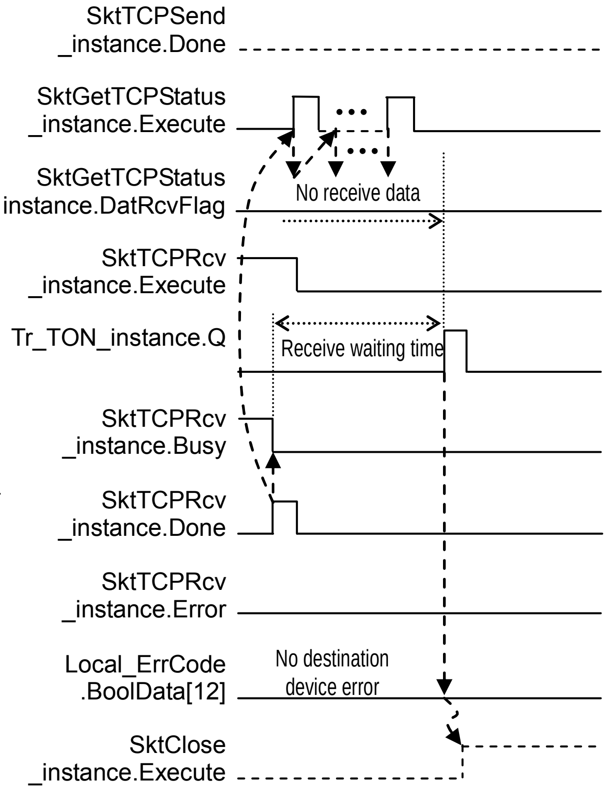


(Timeout)

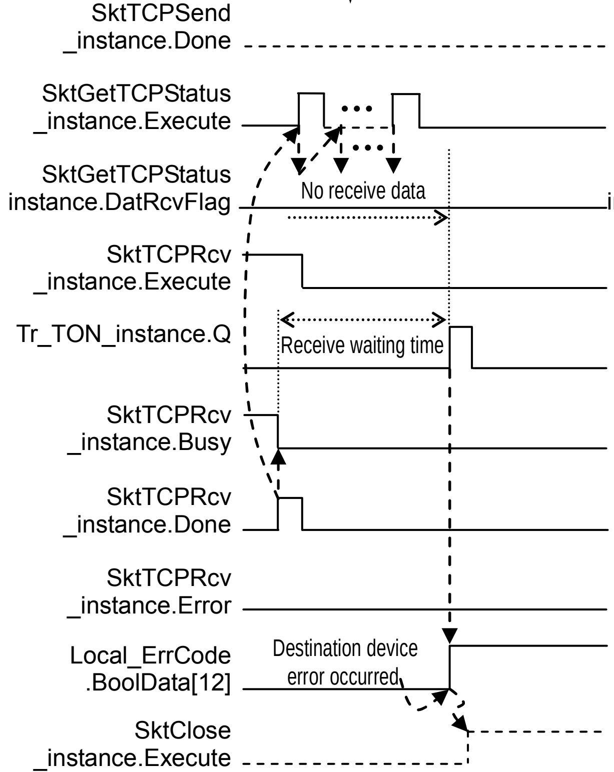
●Receive processing


SktTCPSend _instance.Done
SktGetTCPStatus _instance.Execute
SktGetTCPStatus _instance.DatRcvFla
Receive data exists.
Receive data exists.
SktTCPRcv _instance.Execute
Tr_TON_instance.Q
SktTCPRcv _instance.Busy
SktTCPRcv _instance.Done
SktTCPRcv _instance.Error
Local_ErrCode .BoolData[1]

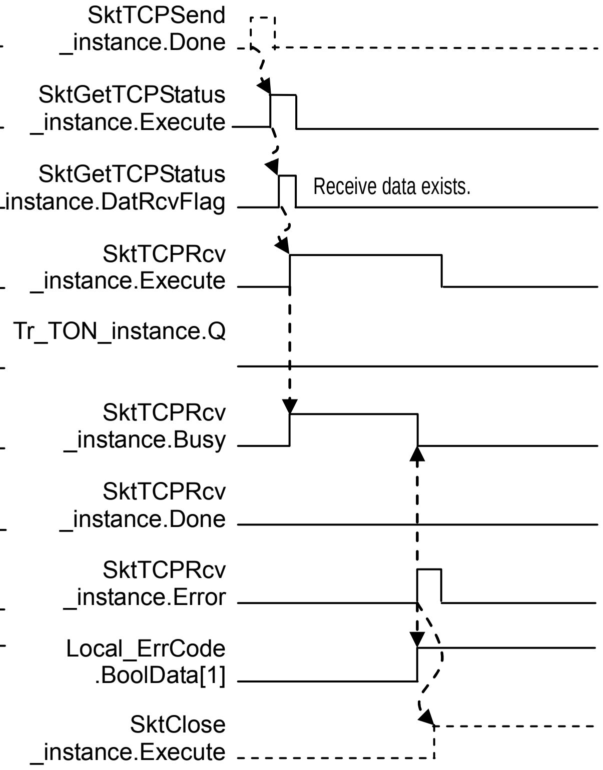
(Repetition)


SktTCPSend _instance.Done
SktGetTCPStatus _instance.Execute
SktGetTCPStatus instance.DatRcvFlag
No receive data
SktTCPRcv _instance.Execute
Tr_TON_instance.Q
Receive waiting time
SktTCPRcv _instance.Busy
SktTCPRcv _instance.Done
SktTCPRcv _instance.Error
Local_ErrCode .BoolData[12]
No destination device error
SktClose _instance.Execute

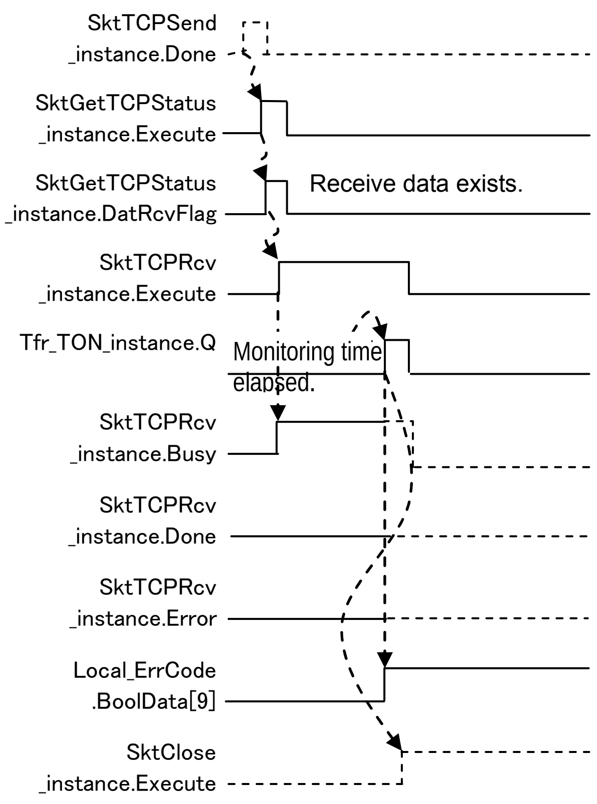
(Normal end)


SktTCPSend _instance.Done
SktGetTCPStatus _instance.Execute
SktGetTCPStatus instance.DatRcvFlag
No receive data
SktTCPRcv _instance.Execute
Tr_TON_instance.Q
Receive waiting time
SktTCPRcv _instance.Busy
SktTCPRcv _instance.Done
SktTCPRcv _instance.Error
Local_ErrCode .BoolData[12]
Destination device error occurred
SktClose _instance.Execute

(Destination device error)

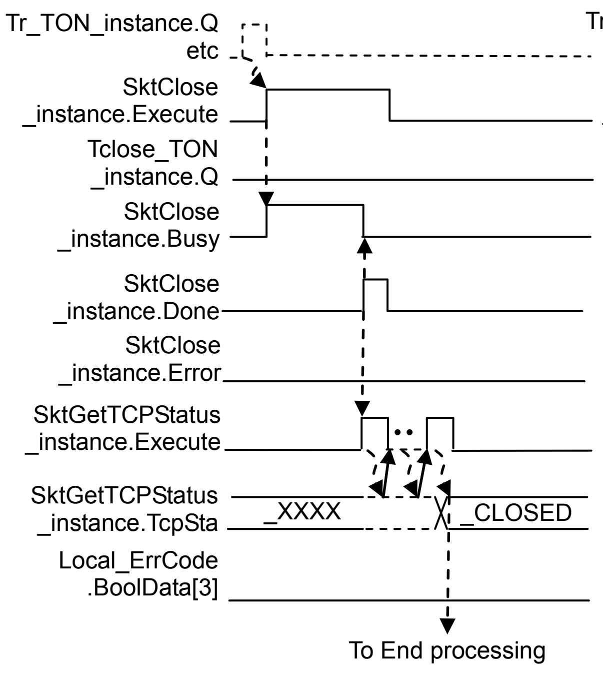

SktTCPSend _instance.Done
SktGetTCPStatus _instance.Execute
SktGetTCPStatus instance.DatRcvFlag
Receive data exists.
SktTCPRcv _instance.Execute
Tr_TON_instance.Q
SktTCPRcv _instance.Busy
SktTCPRcv _instance.Done
SktTCPRcv _instance.Error
Local_ErrCode .BoolData[1]
SktClose _instance.Execute

(Error end)

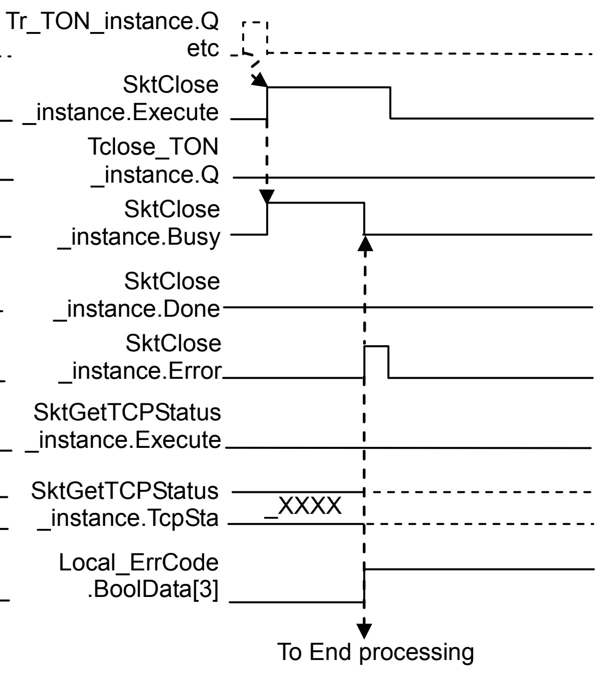

SktTCPSend _instance.Done
SktGetTCPStatus _instance.Execute
SktGetTCPStatus _instance.DatRcvFlag
Receive data exists.
SktTCPRcv _instance.Execute
Tfr_TON_instance.Q
Monitoring time elapsed.
SktTCPRcv _instance.Busy
SktTCPRcv _instance.Done
SktTCPRcv _instance.Error
Local_ErrCode .BoolData[9]
SktClose _instance.Execute

(Timeout)

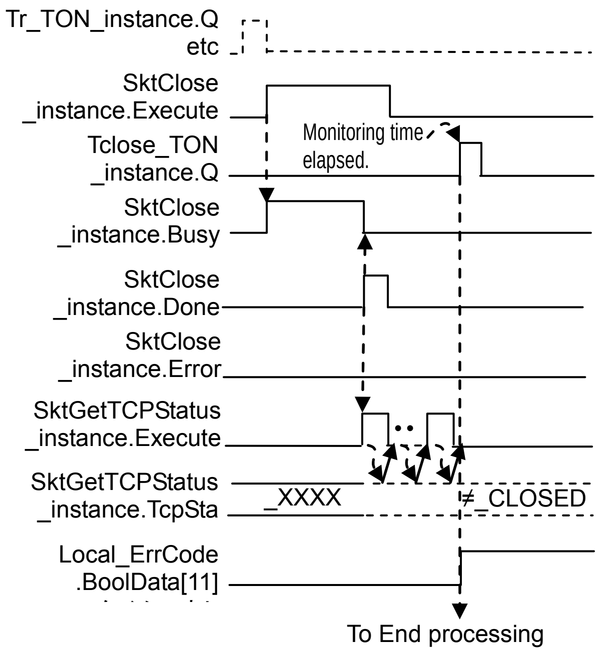
●Close processing



Normal end)



(Error end)



(Timeout)

9.7. Error Process

9.7.1. Error Code List

The error codes of this program are shown below.

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

If the connect processing, send processing, receive processing or close processing ends in error or times out, the error flag will be set in the *Output_ErrCode* variable.

Error flag	Description
16#0000	Normal end
16#0001	The send processing ended in error.
16#0002	The receive processing ended in error.
16#0004	The connect processing ended in error.
16#0008	The close processing ended in error.
16#0100	The send processing did not end in time.
16#0200	The receive processing did not end in time. (Including when an arrival of the response cannot be checked.)
16#0400	The connect processing did not end in time.
16#0800	The 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 occurrence

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

(Example) Error end of connect TCP socket instruction + Time out of close status check

WORD#16#8000 (Error occurrence)
+WORD#16#0001 (Error end of TCP socket connect instruction)
+WORD#16#0100 (Timeout of close status check)
↓
Output_ErrorID: WORD#16#8101

●Error codes [Output_SktCmdsErrorID], [Output_SkTcloseErrorID]

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

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.

Error code	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 connected. 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 ended without completing the executing of an instruction.

Additional Information

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

Additional Information

For details on the socket service errors and troubleshooting, 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).

●TCP connection status error [Output_EtnTcpSta]

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

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).

●Destination device error code

The destination device error code is stored in the *Output_MErrCode* variable.

If a destination device error occurs, the response data will be “ER”.

Error No.	Error name
16#0000 0000	Normal end
16#4552 0000	The error response from the destination device (“ER” is received.)

Additional Information

For details and troubleshooting the destination device errors, refer to *Troubleshooting* in *Chapter 7 APPENDIX* of the *Confocal Fiber Type Displacement Sensor User's Manual* (Cat. No. Z322).

9.7.2. TCP Connection Status Error and Corrective Action

This section describes the status when a TCP connection status error occurs and explains the corrective action.

●Affects of a TCP connection status error

After a TCP connection status error occurs, if this program is executed again without taking any corrective action or without notifying the error, then the destination node specified with the destination IP address (*DstAdr*) input variable and destination port (*DstTcpPort*) input variable may not be waiting for a connection. (Hereinafter this error is referred to as a connect 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*.)

●Status when a TCP connection status error occurs

Both the TCP connection status error after the close processing and a connect 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. In this situation, although all processing (until the close processing) of the program ended in the Controller, the close processing completion notification is not received from the destination device (The completion of the close processing is not confirmed 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 that the destination device is disconnected from other device before resetting the communications port of the destination device.

●Controller status when a TCP connection status error occurs

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

•When a connect processing request is made again by re-executing the program

•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 restarts or the power supply is turned OFF

10. Revision History

Revision code	Date of revision	Revision reason and revision page
01	Jul. 31, 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 ELECTRONICS LLC
One Commerce Drive Schaumburg,
IL 60173-5302 U.S.A.
Tel: (1) 847-843-7900/Fax: (1) 847-843-7787

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

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

Authorized Distributor:

Cat. No. P558-E1-01

0911(-)