OMRON

Machine Automation Controller NJ-series

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

V750 series RFID System

SYSMAC always in control

P543-E1-01

About Intellectual Property Right and Trademarks

Table of Contents

1. Related Manuals 1
2. Terms and Definition 2
3. Remarks 3
4. Overview 5
5. Applicable Devices and Support Software 6
5.1. Applicable Devices 6
5.2. Device Configuration 7
6. Ethernet Communications Settings 9
6.1. Ethernet Communications Settings 9
6.2. Example of Checking Connection 10
7. Connection Procedure 11
7.1. Work Flow 11
7.2. Setting Up the RFID Reader/Writer 12
7.3. Setting Up the Controller 18
7.4. Connection Status Check 25
8. Initialization Method 28
8.1. Controller 28
8.2. RFID Reader/Writer 29
9. Program 30
9.1. Overview 30
9.2. Destination Device Command 34
9.3. Error Detection Processing 37
9.4. Variables 40
9.5. ST Program 45
9.6. Timing Charts 62
9.7. Error Process 68
10. Revision History 74

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
Z235	V750-BA50C04-US V740-HS01[][]	V750-series UHF RFID System User’s Manual

2. Terms and Definition

Terms	Explanation and Definition
IP address	Ethernet uses an IP address to perform communications. The IP address (Internet Protocol address) is an address that is used to identify a node (host computer or controller, etc.) on Ethernet. IP addresses must be set and managed so they do not overlap.
Socket	A socket is an interface that allows you to directly use TCP or UDP functions from the user program. The 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 users are encouraged to confirm the standards and regulations that the system must conform to.
(4) It is prohibited to copy, to reproduce, and to distribute a part of or whole part of this document without the permission of OMRON Corporation.
(5) This document provides the latest information as of April 2013. The information on this manual is subject to change for improvement without notice.

The following notation is used in this document.

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

Precautions for Safe Use

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

Precautions for Correct Use

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

Additional Information

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

Symbols

The triangle symbol indicates precautions (including warnings).
The specific operation is shown in the triangle and explained in text.
This example indicates a general precaution.

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

4. Overview

This document describes the procedure for connecting the RFID Reader/Writer (V750 series) of OMRON Corporation (hereinafter referred to as OMRON) to the NJ-series Machine Automation Controller (hereinafter referred to as Controller) through Ethernet, and provides the procedure for checking their connection.
Refer to the Ethernet communications settings of the prepared project file to understand the setting procedure and key points to connect the devices via Ethernet.
The user program in this project file is used to check the Ethernet connection by sending/receiving the message of “GETR TYP FWV (read the product type and firmware version of the memory data)” to/from the destination device.

Prepare the latest Sysmac Studio project file beforehand. For information on how to obtain the file, contact your OMRON representative.

Name	File name	Version
Sysmac Studio project file (extension: smc)	OMRON_V750_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 given below.

Manufacturer	Name	Model
OMRON	NJ-series CPU Unit	NJ501-[][][][] NJ301-[][][][]
OMRON	RFID Reader/Writer (complies with FCC and EN)	V750-BA50C04-US
OMRON	Antenna	V740-HS01[][]
OMRON	Antenna cable	V740-A01 [][]M

Additional Information

As applicable devices above, the devices with the models and versions listed in Section 5.2. are actually used in this document to describe the procedure for connecting devices and checking the connection.

You cannot use devices with versions lower than the versions listed in Section 5.2.

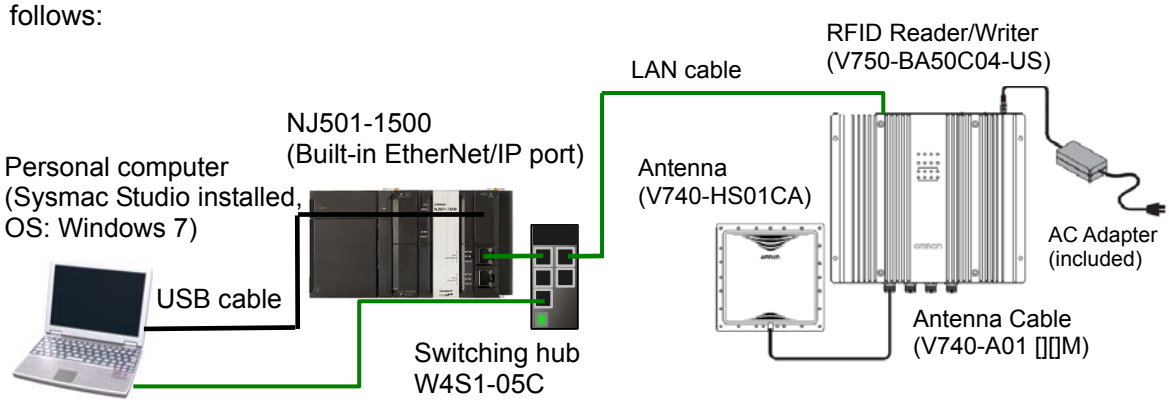
To use the above devices with versions not listed in Section 5.2 or versions higher than those listed in Section 5.2, check the differences in the specifications by referring to the manuals before operating the devices.

Additional Information

This document describes the procedure to establish the network connection. Except for the connection procedure, it does not provide information on operation, installation or wiring method. It also does not describe the function or operation of the devices. Refer to the manuals or contact your OMRON representative.

5.2. Device Configuration

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



Manufacturer	Name	Model	Version
OMRON	NJ-series CPU Unit (Built-in EtherNet/IP port)	NJ501-1500	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_V750_ETN(TCP) _EV101.smc	Ver.1.01
-	Personal computer (OS:Windows7)	-	
-	USB cable (USB 2.0 type B connector)	-	
-	LAN cable	-	
OMRON	RFID Reader/Writer	V750-BA50C04-US	Ver.102-10 2-103-0
OMRON	Antenna (Circular) (4 max.)	V740-HS01CA	
OMRON	Antenna cable	V740-A01[][]M	
OMRON	AC Adapter (included)	-	

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

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

6. Ethernet Communications Settings

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

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 Personal Computer and the RFID Reader/Writer

The setting example below is used to explain the procedure for setting the RFID Reader/Writer by using the personal computer.

Setting item	Personal computer used for setting	RFID Reader/Writer
IP address	192.168.1.1	192.168.1.200 (Default)
Subnet mask	255.255.255.0	255.255.255.0 (Default)
Gateway	---.---.---.---	192.168.1.254 (Default)

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

6.1.2. Communications Settings between the Controller and the RFID Reader/Writer

The setting example below is used to explain the procedure for connecting the Controller to the RFID Reader/Writer.

Setting item	NJ501-1500	RFID Reader/Writer
IP address	192.168.250.1	192.168.250.2
Subnet mask	255.255.255.0	255.255.255.0 (Default)
Gateway	-.-.-.-	192.168.1.254 (Default)
Host name	-	“V750-BA50C04-US” (Default)
Domain name	-	Blank (Default)
DHCP	-	OFF (Default)
TCP/IP port	(This is set by the program.)	7090 (Default)

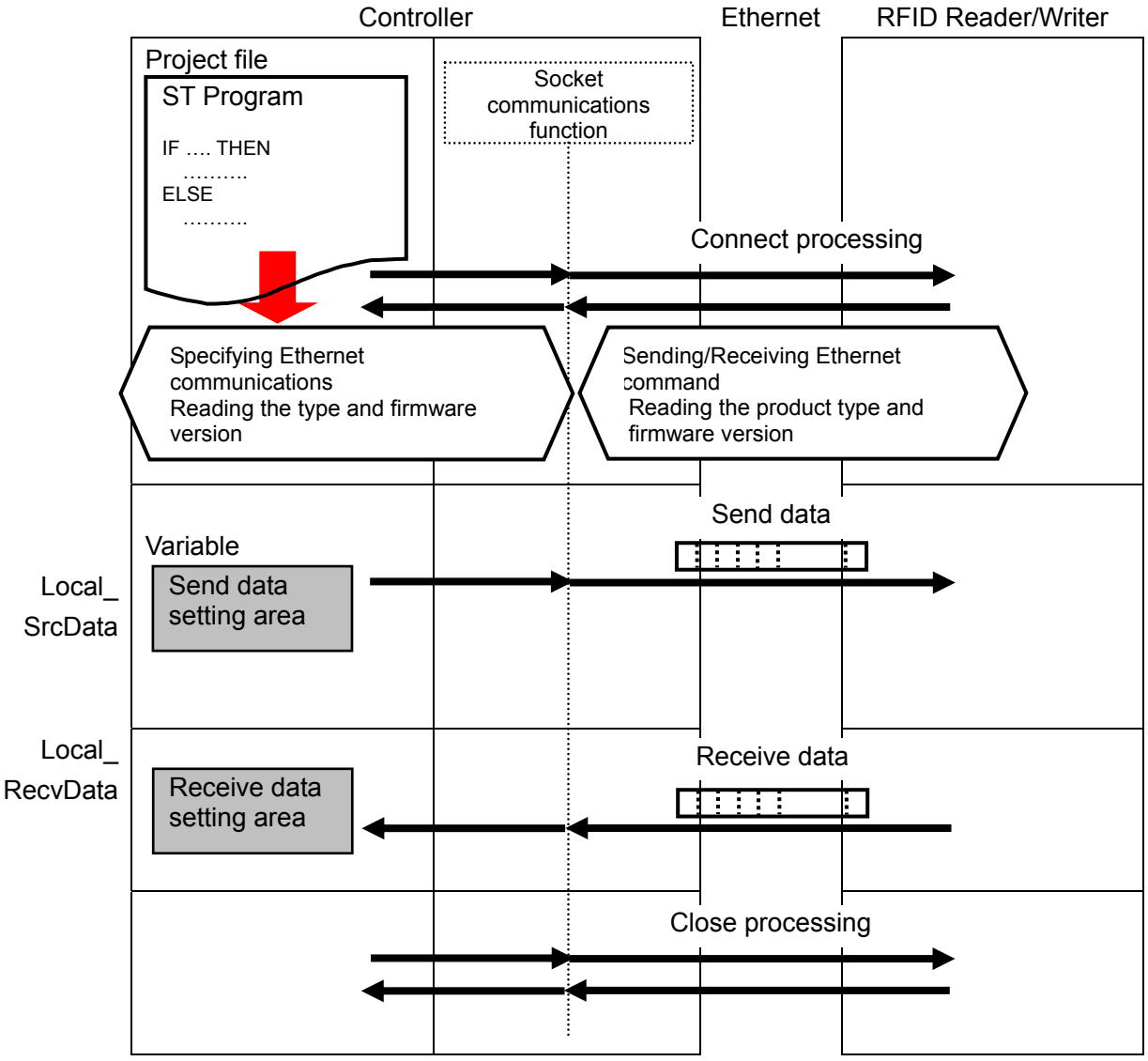
*In this document, the gateway setting is unnecessary because the connection is made in the same segment.

*This project file uses the default settings (keep-alive: use, linger option: Do not use) of the keep-alive and linger option functions for the TCP socket communications. 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 RFID Reader/Writer.

The Controller and RFID Reader/Writer send and receive the message of “GETR TYP FWV (read the product type and firmware version of the memory data)”. The following figure outlines the operation.



7. Connection Procedure

This section describes the procedure for connecting the RFID Reader/Writer to the Controller via Ethernet.

This document explains the procedures for setting the Controller and RFID Reader/Writer 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 RFID Reader/Writer to the Controller via Ethernet.

7.2 Setting Up the RFID Reader/Writer	Set up the RFID Reader/Writer.
↓	
7.2.1 Parameter Setting	Set the parameters of the RFID Reader/Writer.
↓	
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 Connection Status Check	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.

Precautions for Correct Use

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

7.2. Setting Up the RFID Reader/Writer

Set up the RFID Reader/Writer.

7.2.1. Parameter Setting

Set the parameters of the RFID Reader/Writer.

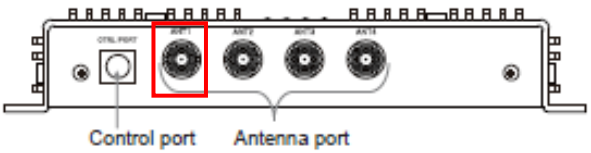
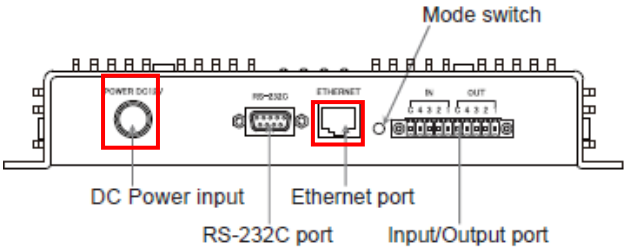
For the setting, a web browser (e.g., Internet Explore) that can execute Java software is required. Install necessary software so that Java software can operate.

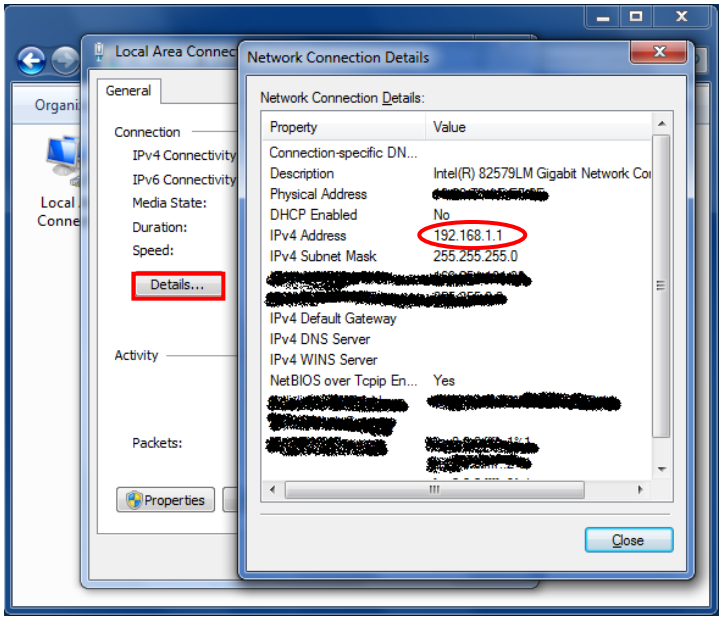
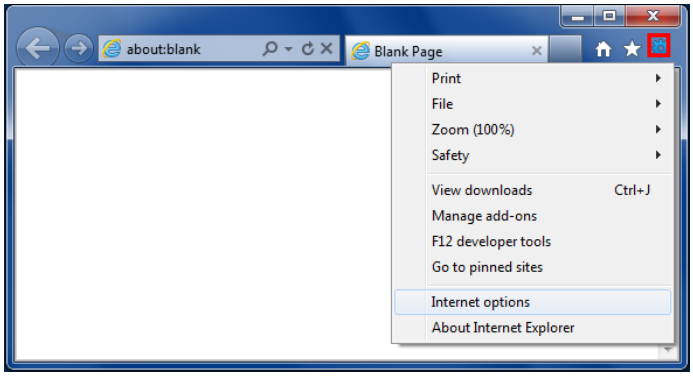
Set the IP address of the personal computer to 192.168.1.1.

Precautions for Correct Use

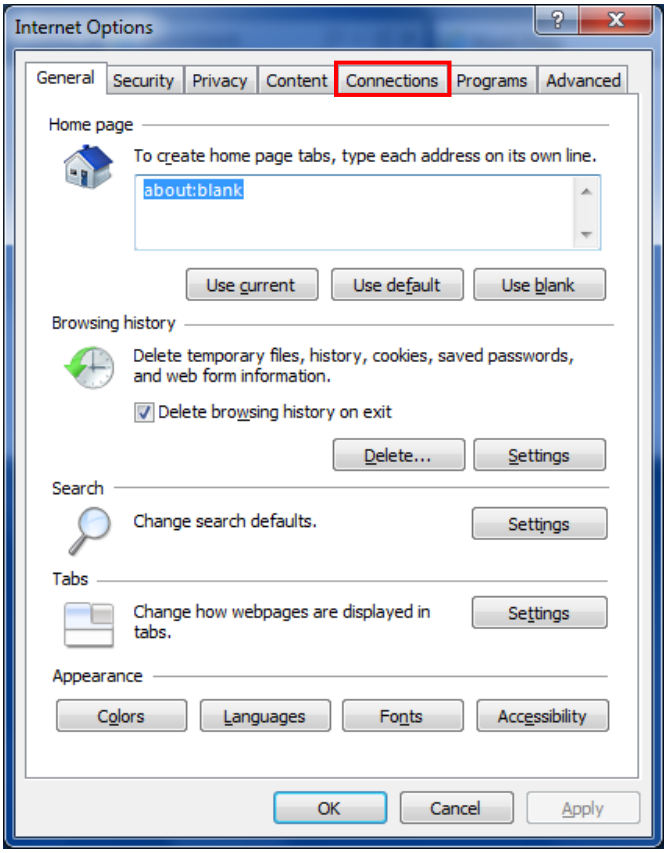
Set the parameters of the RFID Reader/Writer by using the Ethernet communications of the personal computer.

Note that you may need to change the settings of the personal computer depending on the status of the personal computer.

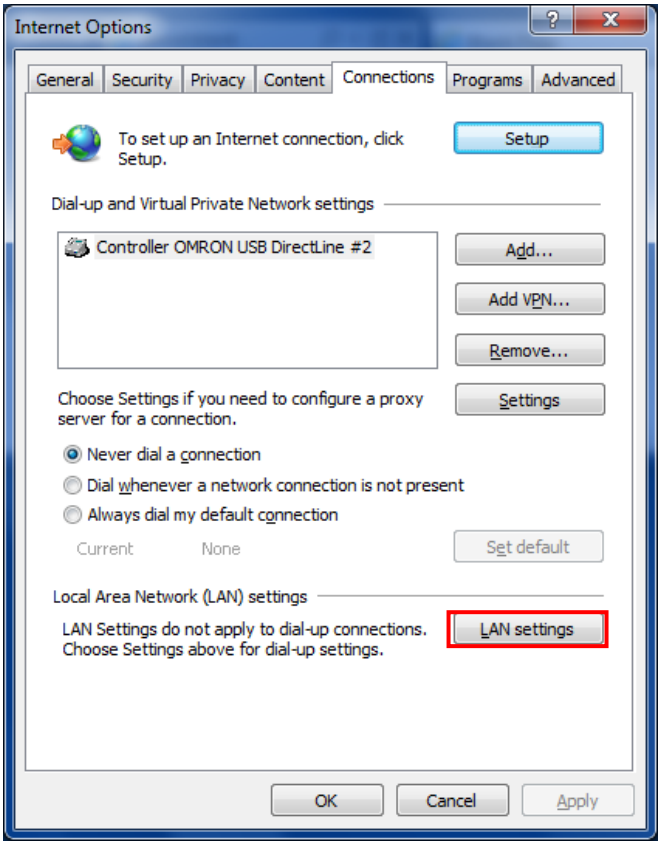
1	Connect the antenna to the antenna port on the side of the RFID Reader/Writer.	 (Side of RFID Reader/Writer)
2	Connect the Switching Hub to the Ethernet port on the other side of the RFID Reader/Writer to using the LAN cable. Connect the included AC Adapter cable to the DC power input.	 (Other side of RFID Reader/Writer)

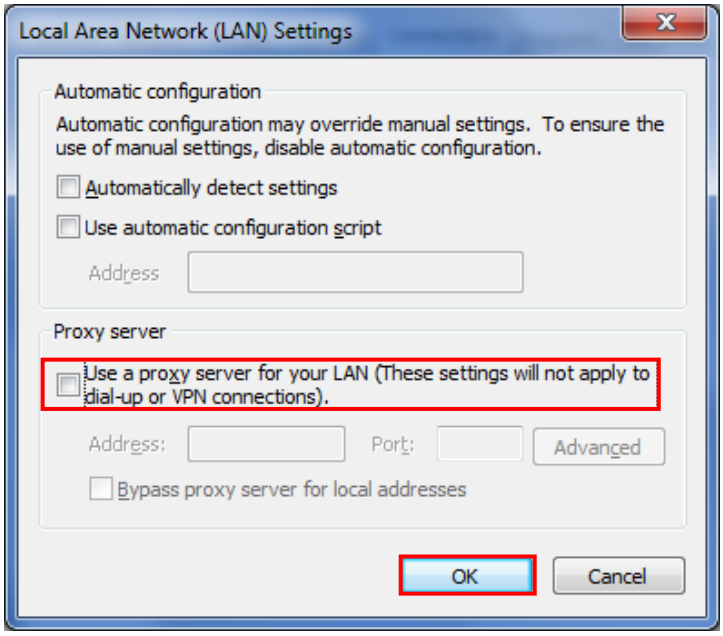
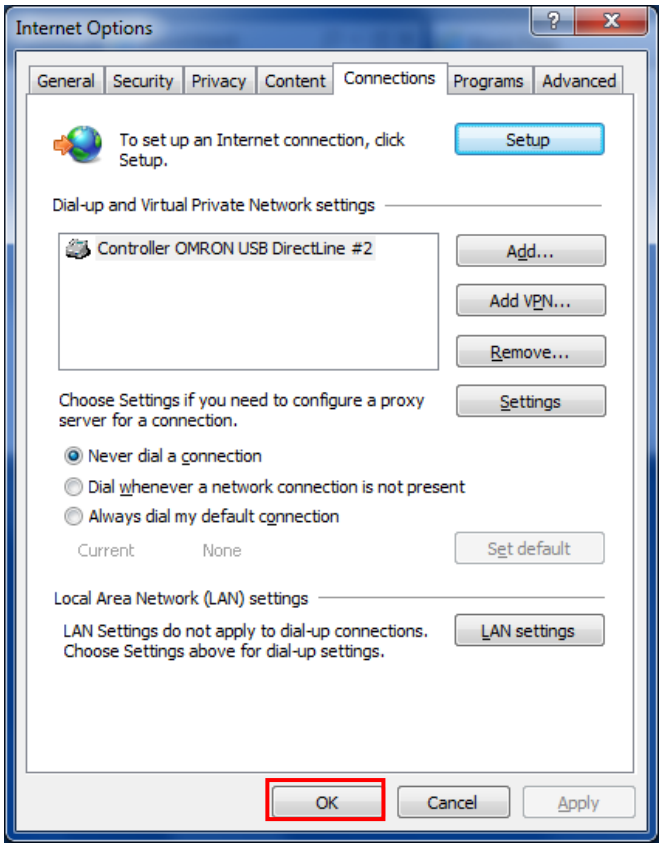
3	Start Internet Explorer from the personal computer that is connected to the Switching Hub. *Set the IP address of the personal computer to 192.168.1.1. Use the following procedure to check the IP address of the personal computer. (1)Click **Connect to the Internet View network status and tasks** - **Change adapter settings** on the Control Panel. (2)Double-click **Local Area Connection** on the Network Connections. (3)Click the **Details** Button on the Local Area Connection Status Dialog Box. (4)Confirm that the IP address is 192.168.1.1.	 
4	Click **Tool** ( ) on the command bar of Internet Explorer and select ***Internet options***.	

5 The Internet Options Dialog Box is displayed. Select the Connections Tab.



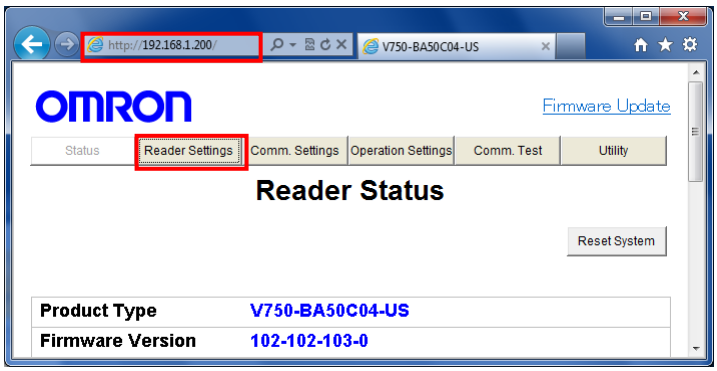
6 The Internet Options Dialog Box is displayed. Click the **LAN settings** Button.



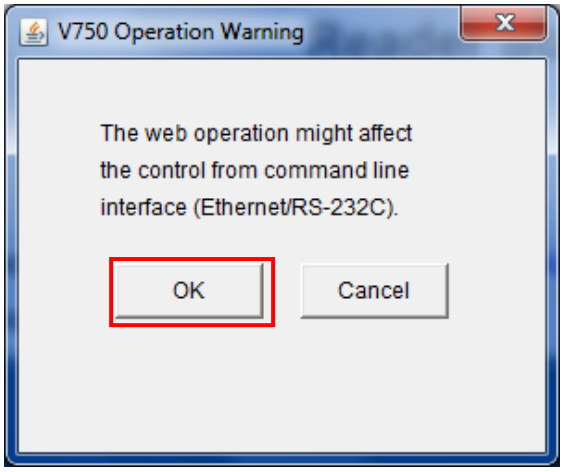
7	The Local Area Network (LAN) Settings Dialog Box is displayed. Confirm that the *Use a proxy server for your LAN* Check Box is cleared from the Proxy server Field, and click the **OK** Button.	
8	Click the **OK** Button on the Internet Options Dialog Box.	

9

Type http://192.168.1.200 / in the address bar ( ) of Internet Explorer.



The Reader Status Window is displayed. Click the **Reader Settings** Button.



The V750 Operation Warning Dialog Box is displayed. Click the **OK** Button.

10

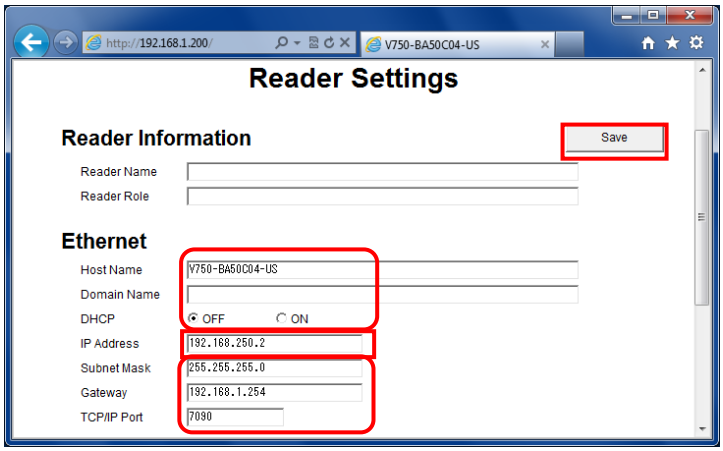
The Reader Settings Window shows the Ethernet settings. Make the settings as follows and click the Save Button.

Host Name
:V750-BA50C04-US
Domain Name
:Blank
DHCP: OFF
IP Address
:192.168.250.2
Subnet Mask
:255.255.255.0
Gateway
:192.168.1.254
TCP/IP Port
:7090

*If the settings are different from the above, change the corresponding set values.

Exit Internet Explorer.

*If Internet Explorer does not exit, the IP address of the RFID Reader/Writer will be changed and the screen will not be displayed.



*The gateway setting is unnecessary. However, if you leave the Gateway Field blank, an error will occur. Therefore, use the default setting.

11	Cycle the power supply to the RFID Reader/Writer. *The new parameters will be enabled after the power supply is cycled.

Additional Information

In addition to changing the Ethernet settings on the web browser screen, you can set and read the Ethernet parameters by using the setting commands (SETR and GETR). For information on the specifications of the setting commands, refer to *Section 5 Command Line Interface* in the *V750-series UHF RFID System User's Manual* (Cat. No. Z235).

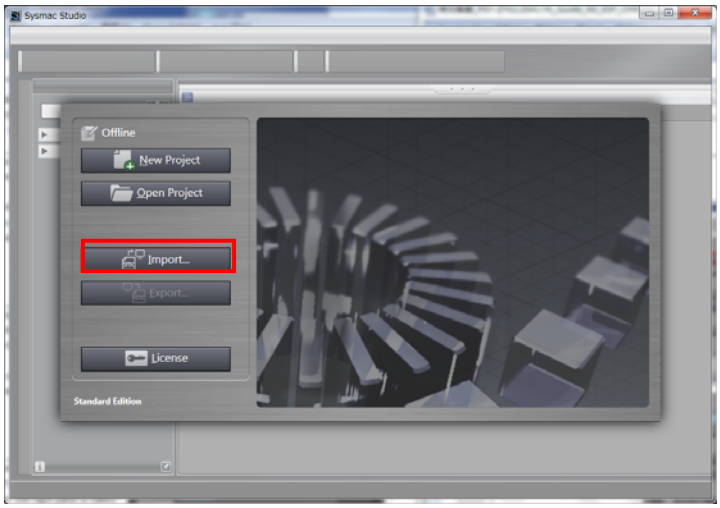
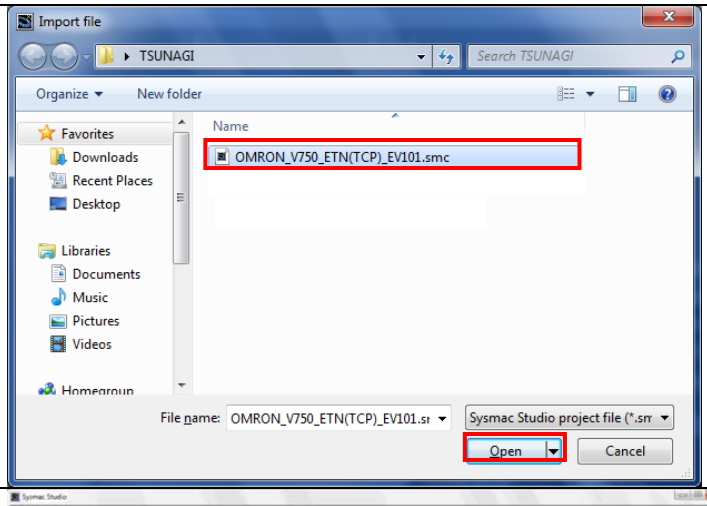
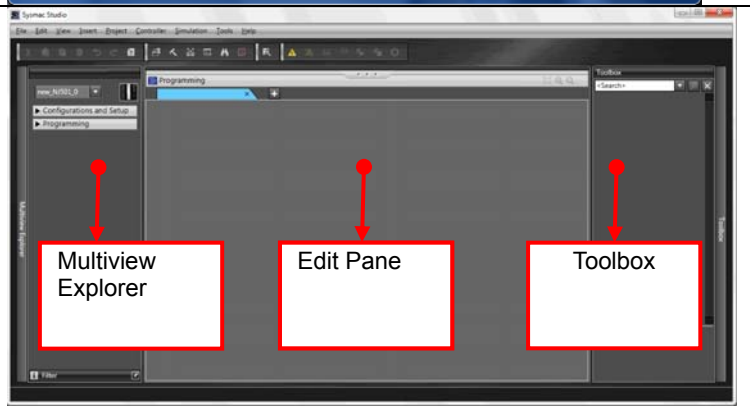
7.3. Setting Up the Controller

Set up the Controller.

7.3.1. Starting the Sysmac Studio and Importing the Project File

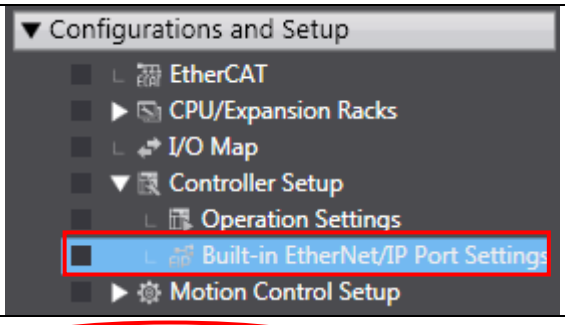
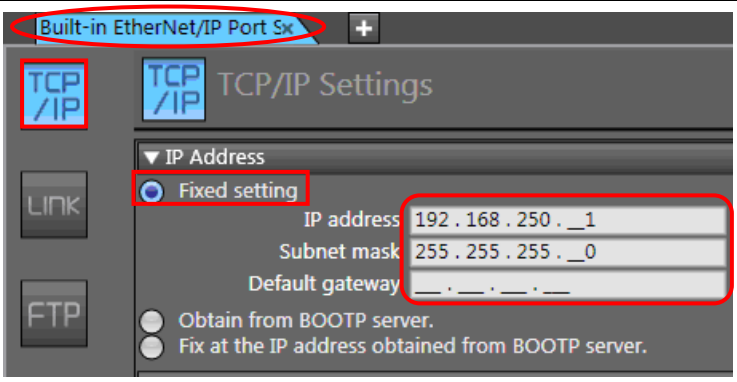
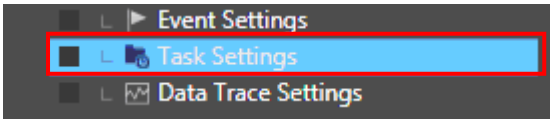
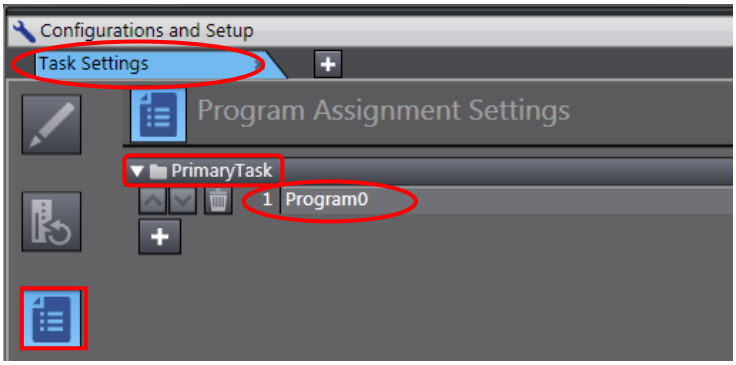
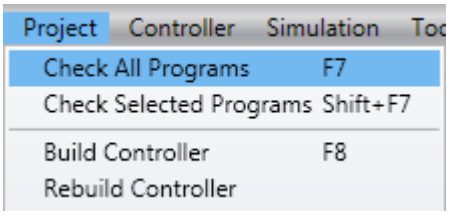
Start the Sysmac Studio and import the project file.

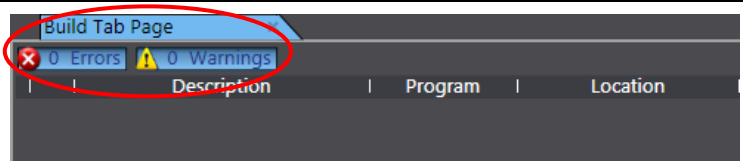
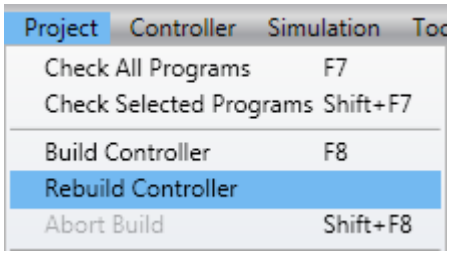
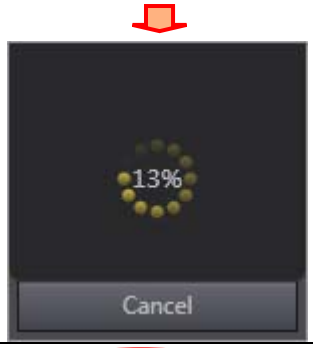
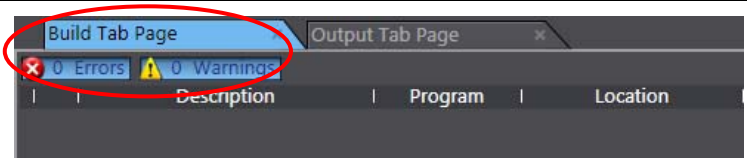
Install the Sysmac Studio and USB driver beforehand.

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.	
2	The Import File Dialog Box is displayed. Select OMRON_V750_ETN(TCP)_EV101.smc and click the **Open** Button. *Obtain the project file from OMRON.	
3	OMRON_V750_ETN(TCP)_EV101 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 dialog box is displayed, check the version of the Sysmac Studio.	

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. Click the **TCP/IP Settings** Button, select the *Fixed setting* Option in the IP Address Field, and confirm that the following settings are made. •IP address: 192.168.250.1 •Subnet mask: 255.255.255.0 •Default gateway: _._._._ (blank)	
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.	

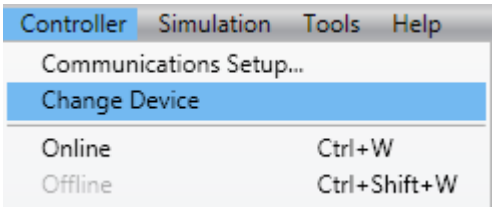
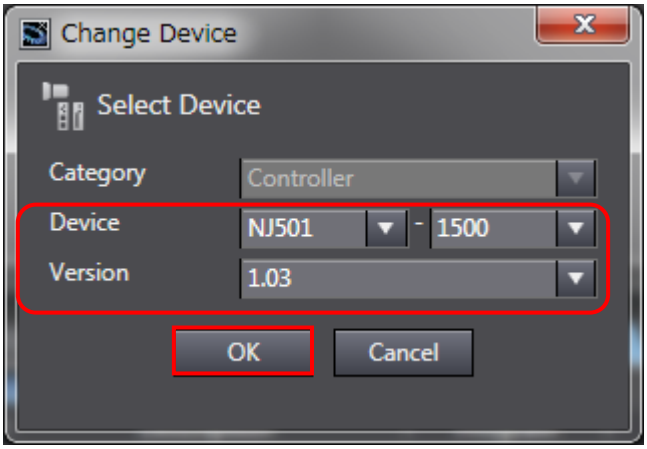
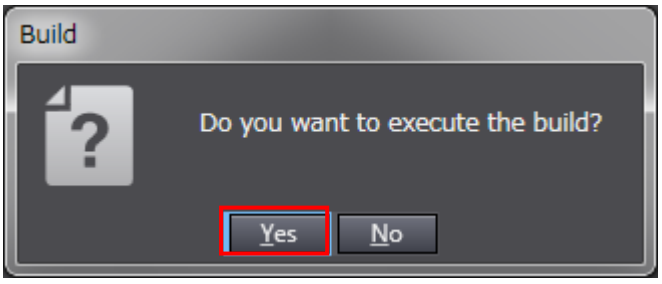
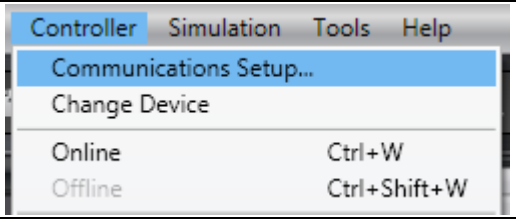
6	The Build Tab Page is displayed in the Edit Pane. Confirm that “0 Errors” and “0 Warnings” are displayed.	
7	Select ***Rebuild Controller*** from the Project Menu. A screen is displayed indicating the conversion is being performed.	 
8	Confirm that “0 Errors” and “0 Warnings” are displayed in the Build Tab Page.	

7.3.3. Going Online and Transferring the Project Data

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

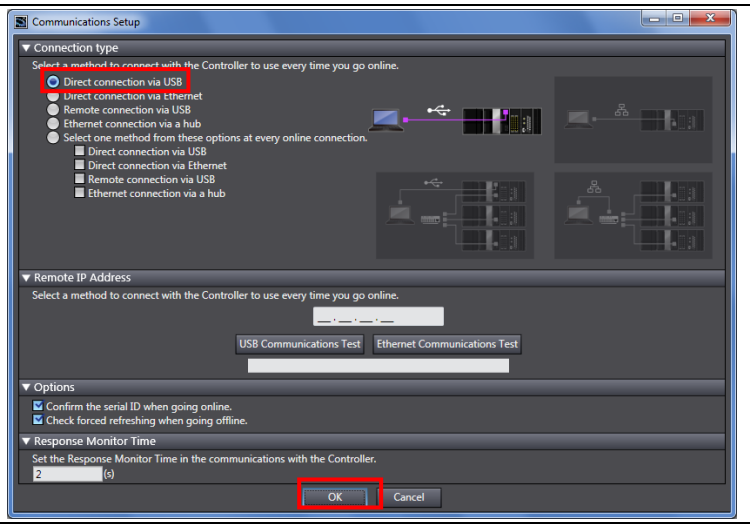
⚠ WARNING

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

1	Select ***Change Device*** from the Controller Menu.	
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 from the above, change the values from the pull-down list.	
3	If settings were changed in step 2, the Build Dialog Box is displayed. Click the **Yes** Button. *This dialog box is not displayed if no change was made.	
4	Select ***Communications Setup*** from the Controller Menu.	

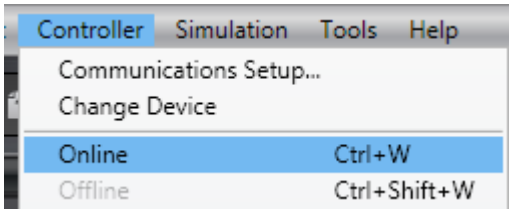
5 The Communications Setup Dialog Box is displayed.
Select the *Direct Connection via USB* Option from Connection Type.

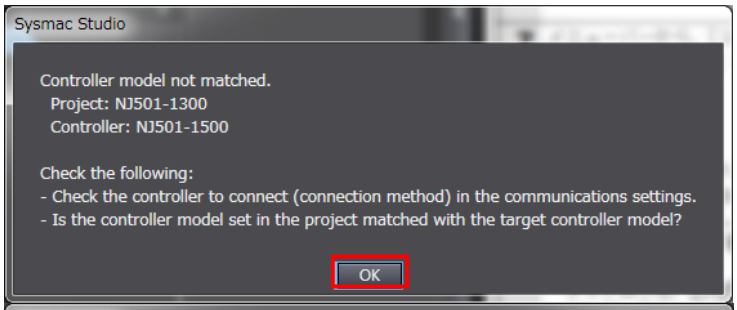
Click the **OK** Button.

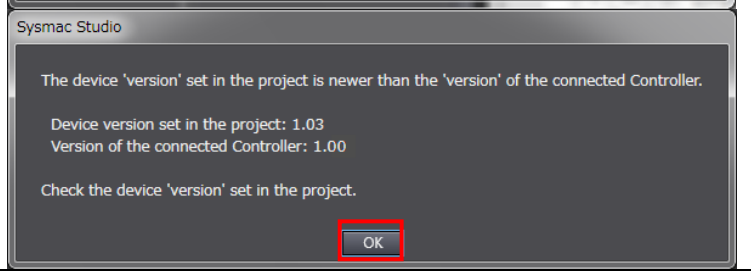


6 Select ***Online*** from the Controller Menu.

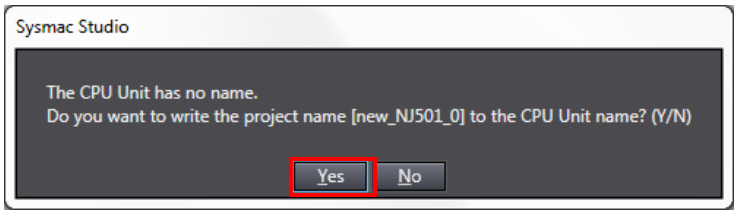
*If the dialog on the right is displayed, the model or version of the Controller does not match those of the project file. Check the settings of the project file, return to step 1 and try again.
Click the **OK** Button to close the dialog box.



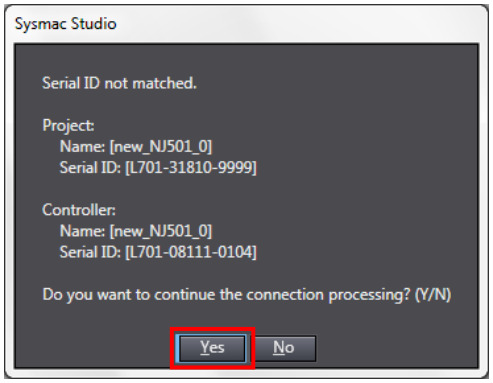




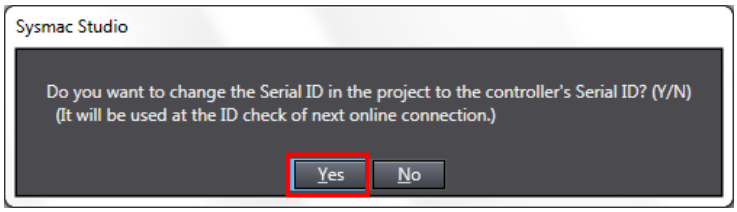
7 A confirmation dialog is displayed. Click the **Yes** Button.



*The displayed dialog differs depending on the status of the Controller used. Click the **Yes** Button to proceed with the processing.



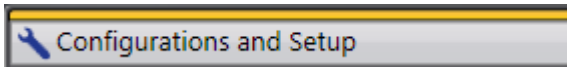
*The displayed serial ID differs depending on the device.



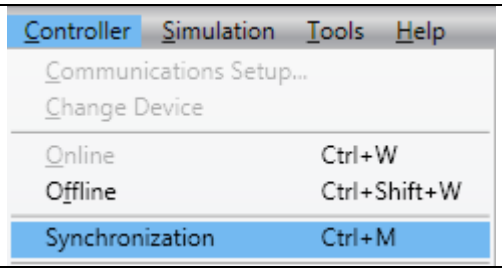
Additional Information

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

8 When an online connection is established, a yellow bar is displayed on the top of the Edit Pane.

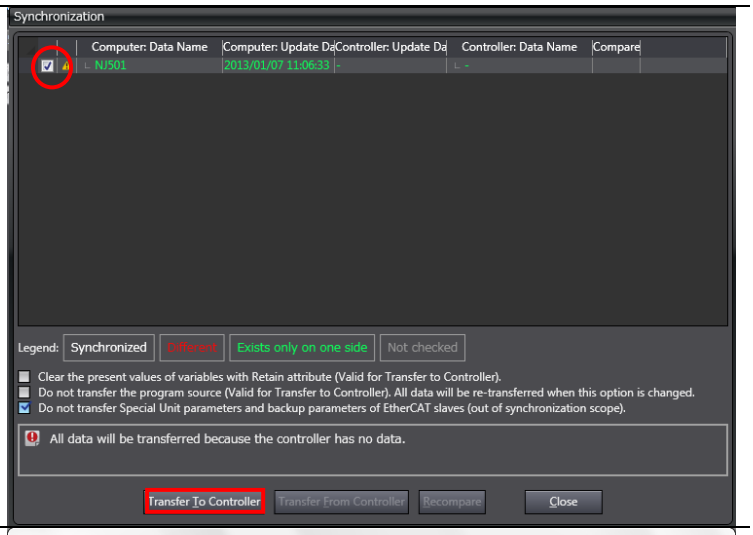


9 Select ***Synchronization*** from the Controller Menu.

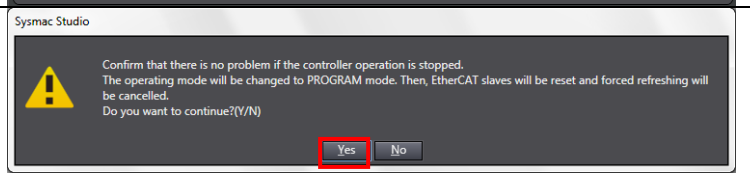


10 The Synchronization Dialog Box is displayed.
Confirm that the data to transfer (NJ501 in the right figure) is selected. Then, click the **Transfer to Controller** Button.

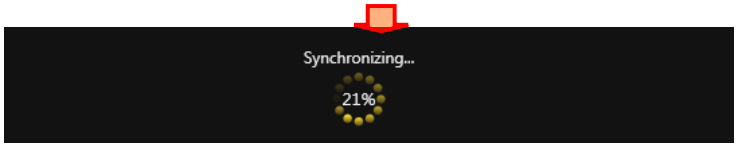
*After executing the **Transfer to Controller**, the Sysmac Studio project data is transferred to the Controller and the data are compared.



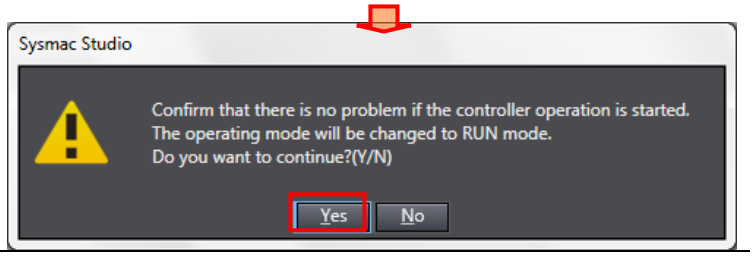
11 A confirmation dialog is displayed. Click the **Yes** Button.



A screen stating "Synchronizing" is displayed.



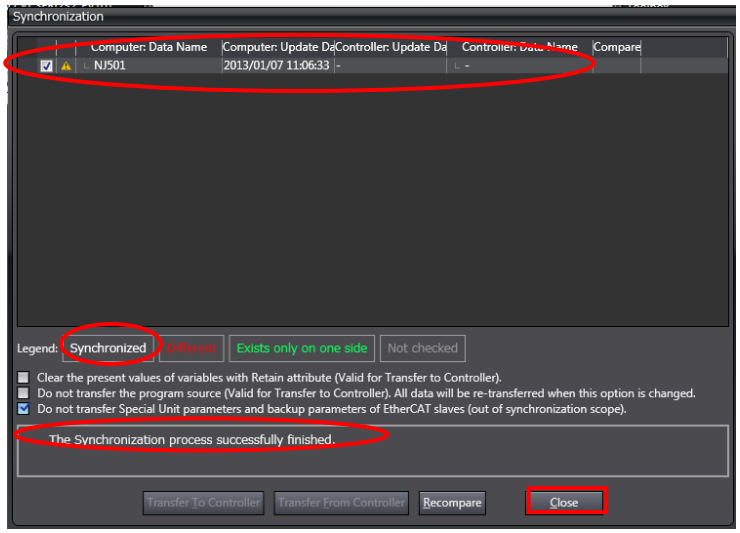
A confirmation dialog box is displayed. Click the **Yes** Button.



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 and that of the Controller match.

*If the synchronization fails, check the wiring and repeat the procedure described in this section.



7.4. Connection Status Check

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.

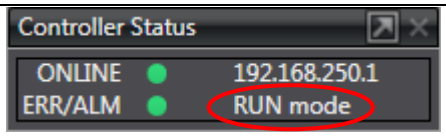
Precautions for Correct Use

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.

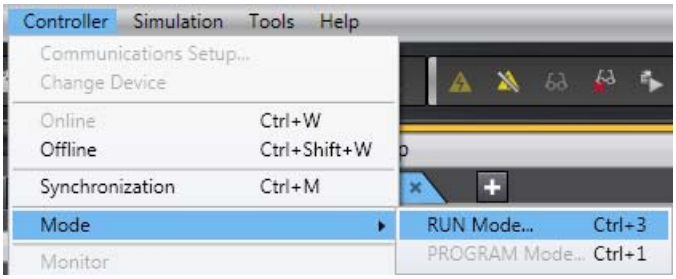
7.4.1. Executing the Program and Checking the Receive Data

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

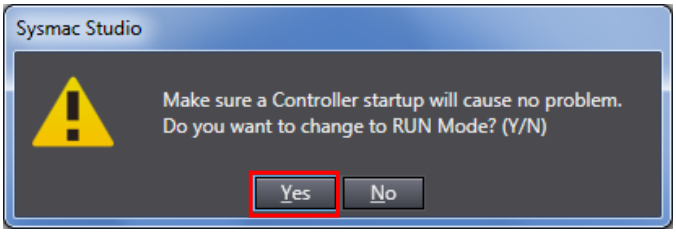
1 Confirm that RUN mode is displayed on the Controller Status Pane of the Sysmac Studio.



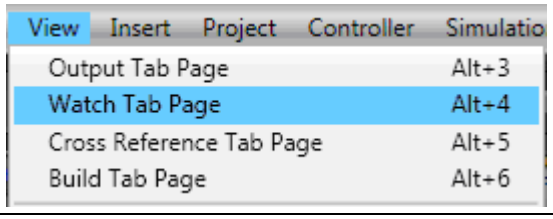
If PROGRAM mode is shown, select ***Mode*** - ***RUN Mode*** from the Controller Menu.



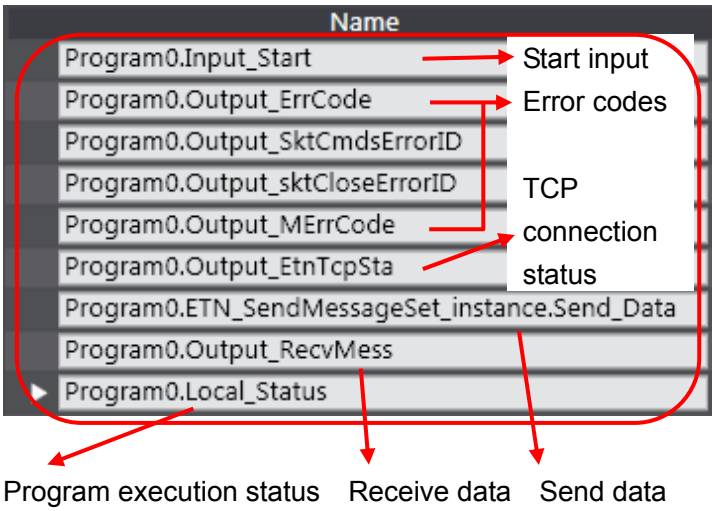
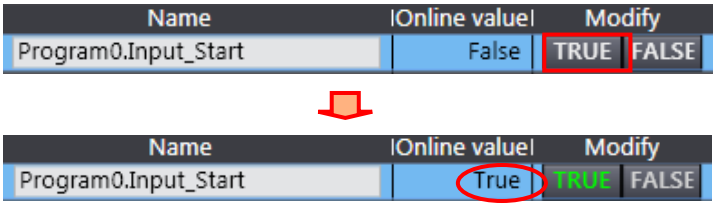
A confirmation dialog box is displayed. Click the **Yes** Button.



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

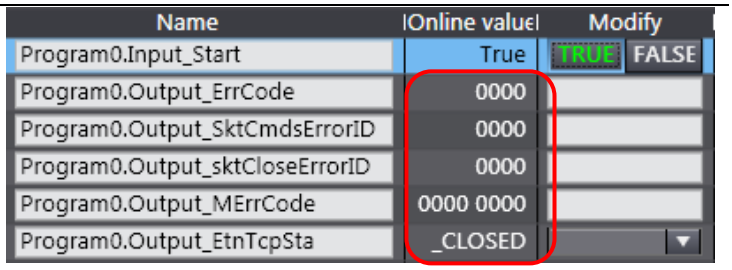
6

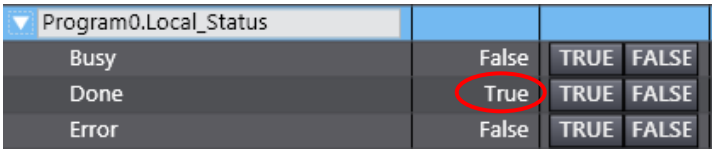
When the communications ends normally, each error code changes to 0.
The TCP connection status (*Output_EtnTcpSta*) changes to *_CLOSED*.

*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*.

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*.



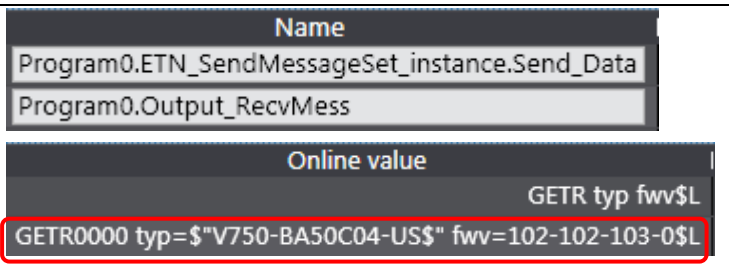


7

The response data received from the destination device is stored in *Output_RecvMess.* (*ETN_SendMessageSet_instance.Send_Data* is a send command.)
In the Watch Tab Page 1, specify an area to reference as shown in the right figure.

*The response data differ depending on the device used

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



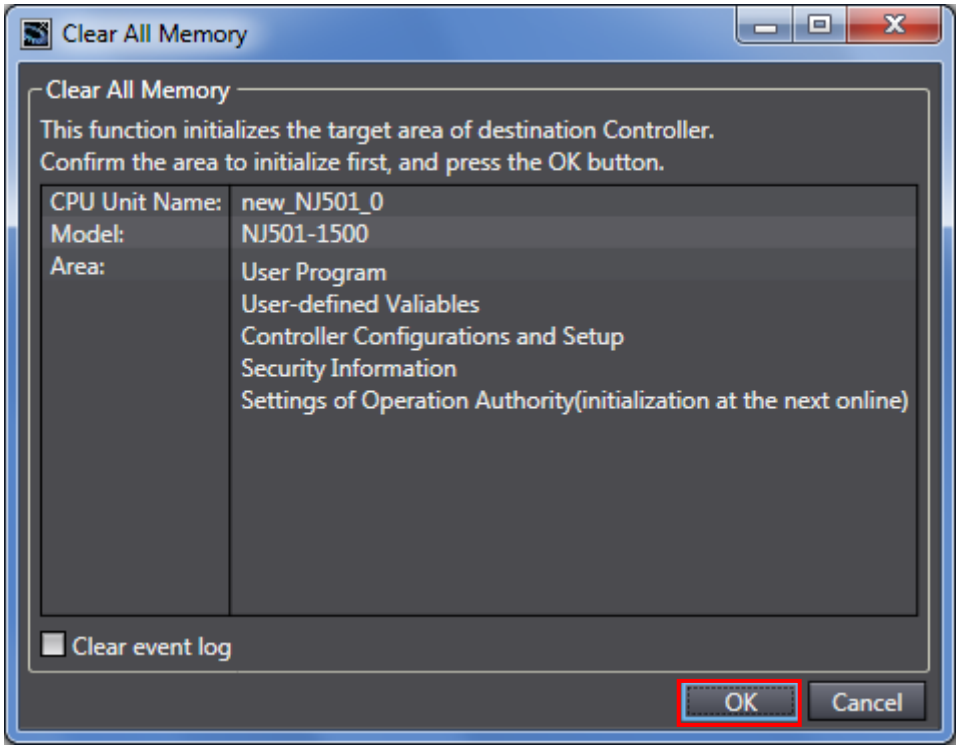
Receive data
•Send command: “GETR”
•Response code: “0000” (normal)
•Model: ”typ=$”V750-BA50C04-US$””
•Firmware version: “fwv=102-102-103-0”
•Terminator: “$L”([LF])

8. Initialization Method

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

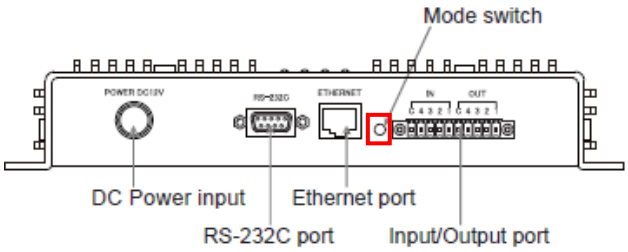
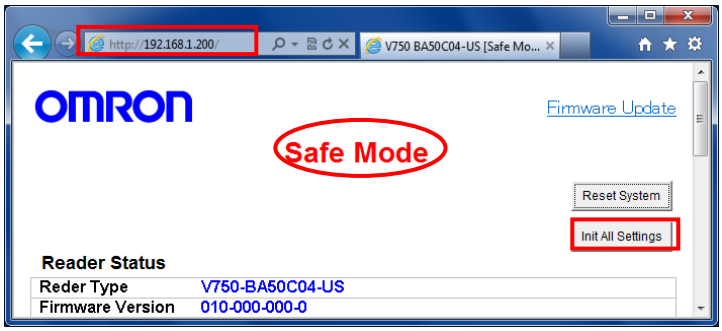
8.1. Controller

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



8.2. RFID Reader/Writer

Use the following procedure to initialize the settings of the RFID Reader/Writer.

1	Press the mode switch at least one second and start the Safe Mode of the RFID Reader/Writer.	 (Side of the RFID Reader/Writer)
2	Type “http://192.168.1.200/" in the address bar ( ) of the Internet Explorer. The Safe Mode Window is displayed. Click the **Init All Settings** Button. The RFID Reader/Writer will be initialized and restarted. *The firmware version in the safe mode is 010-000-000-0.	

Additional Information

For the initialization of the RFID Reader/Writer, refer to *Mode switch* in *Names and Functions of Components* in *Reader* of *Section 2 Specifications and Performance* and *Mode* in *Section 3 Mode and Function* in the *V750-series UHF RFID System User's Manual* (Cat. No. Z235).

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 RFID Reader/Writer (V750 series) (hereinafter referred to as destination device) and the Controller (built-in EtherNet/IP port) (hereinafter referred to as Controller).

This program uses the socket service functions of the Controller to execute “GETR TYP FWV command (read the product type and firmware version of the memory data)” on the destination device and to detect a normal end or an error end.
The normal end of this program means a normal end of the TCP socket communications.
The 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).

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 and the performance variation of the device.

Additional Information

With Sysmac Studio, add the prefix “10#" (possible to omit) 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, add the prefix “<data type>#”. (e.g., “UINT#10#1000” and “WORD#16#03E8”, etc.)

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 are sent.

1.	Connect processing	The Controller issues a TCP open 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.
	↓	
4.	Close processing	The Controller issues a close request to the destination device, and terminates the TCP connection.

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

If Send only is set, the response 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 communications performed by using the TCP socket service function blocks (hereinafter referred to as socket service instructions) and outlines the general operation of the send/receive message.

Additional Information

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

●TCP Socket Services with Socket Service Instructions

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

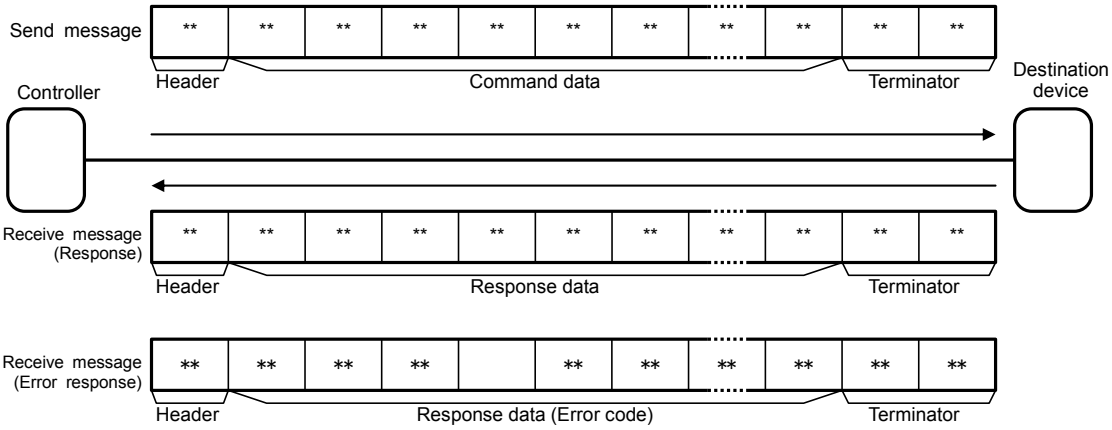
Name	Function blocks	Description
Connect TCP Socket	SktTCPConnect	Connects the TCP port of the destination device.
TCP Socket Send	SktTCPSend	Sends data from a specified TCP socket.
TCP Socket Receive	SktTCPRcv	Reads the data from the receive buffer for a specified TCP socket.
Close TCP Socket	SktClose	Closes a specified TCP socket.
Read TCP Socket Status	SktGetTCPStatus	Reads the status of a specified TCP socket. By using this instruction, this program checks if the receive processing is completed at the receive processing and checks the closing 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	

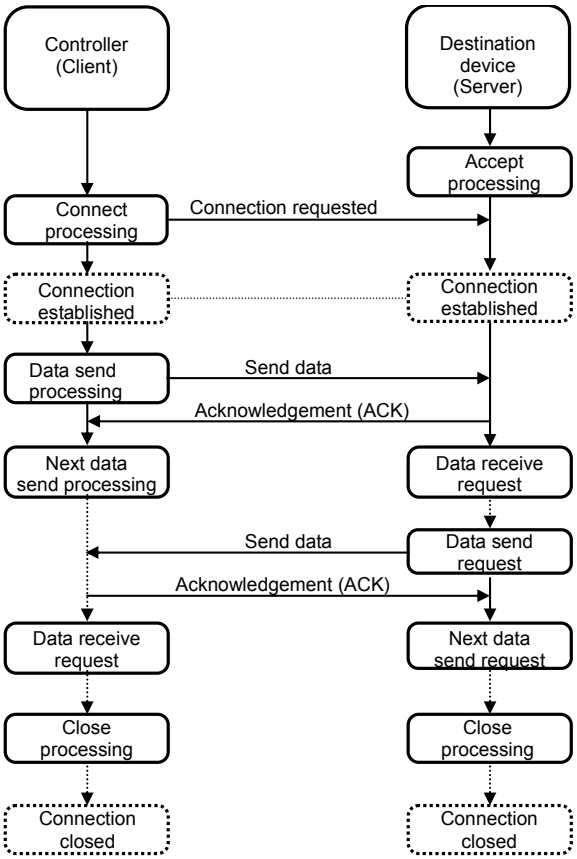
*1: The address indicates an IP address and a port number.
*2: A DNS or Hosts setting is required to use a host name.

●Send/Receive message



●Communications sequence

TCP communications are performed between the destination device (server) and the 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 “GETR TYP FWV (read the product type and firmware version of the memory data) command” to read the information of the destination device.

Command	Description
GETR	Read the Reader/Writer settings.

Additional Information

For details, refer to *Section 5 Command Line Interface* in the *V750-series UHF RFID System User's Manual* (Cat. No. Z235).

9.2.2. Detailed Description of the Command

This section explains the formats used to read the information on the destination device by executing the GETR TYP FWV (read the product type and firmware version of the memory data) 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 GETR TYP FWV (read the product type and firmware version of the memory data) command.

•ASCII codes are sent except for the header and terminator.

Data	Number of bytes	Remarks
Command code	4	Fixed: ”GETR”
(Space *1)	1	Fixed: ” ” (Space)
(Parameter and option *1)	1 and greater *2	Fixed: ”typ” (product type) + ”+”fwv” (firmware version)
Terminator	1	Fixed: [LF](16#0A)

*1: When this is not used, the terminator is moved forward.

*2: Any number of bytes can be set for parameters and 3 bytes for options.

●Command format of the receive message

This is the response format of the message received by the Controller from the destination device according to the setting of the GETR TYP FWV (read the product type and firmware version of the memory data) command.

•ASCII codes are received except for the header and terminator.

Command	Number of bytes	Remarks
Command code	4	Fixed: ”GETR” or Fixed: ”ICMD”
Response code	4	Destination error code (Refer to *9.7.1. Error Code List*.)
(Space *)	1	Fixed: ” ” (Space. Data are separated by a space.)
(Response data *)	1 and greater	Fixed: ”typ=$”[product type V750]$”” (The product type is enclosed in $” and $”.),”fwv=[Firmware version]” (Firmware version) (The information of GETR command options specified with this program is returned.)
Terminator	1	Fixed: [LF](16#0A)

*The terminator is moved forward for an error message when there is no response data because the command is undefined or the parameter of the send command is illegal.

9.2.3. Command Settings

This section explains the details on the settings of the GETR TYP FWV (read the product type and firmware version of the memory data) command.

●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])	CONCAT('GETR’,‘ typ,’ fwv')
Send_Terminate	Send terminator (STRING[5])	‘$L’ ([LF]: 16#0A)

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 as output receive data.

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 message

*Send message

47	45	54	52	20	74	79	70	20	66	77	76	0A
'G'	'E'	'T'	'R'	' '	't'	'y'	‘p’	' '	‘f’	‘w’	‘v’	[LF]
Send command												Terminator

*Receive message 1 (at normal process)

47	45	54	52	30	30	30	30	20	74	79	70	3D	22
'G'	'E'	'T'	'R'	‘0000’				' '	't'	'y'	‘p’	‘=’	‘“’
Command				Response code				Data (parameter)					

•••	22	20	66	77	76	3D	•••	0A
Product type	‘”’	' '	‘f’	‘w’	‘v’	‘=’	Version	[LF]
	Data (parameter)							Terminator

*Receive message 2 (at error process)

47	45	54	52	•••	0A
'G'	'E'	'T'	'R'	Response code	[LF]
Command					Terminator

*Receive message 3 (at error process: undefined)

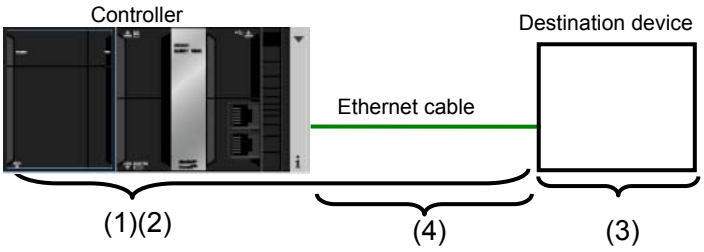
49	43	4D	44	•••	0A
'I'	'C'	'M'	'D'	Response code	[LF]
Command					Terminator

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 of the following items (1) to (4). For information on error codes, refer to *9.7. Error Process.*



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

Errors occurred in the program during TCP socket communications such as command format error and 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 time monitoring function 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 the response code, which is returned from the destination device when an error occurs. For information on the send/Receive messages, refer to *9.2. Destination Device Command*.

Receive message at normal process

‘GETR’	‘0000’	*···*	16#0A
Command code	Response code	Response data	Terminator

Receive message at error process

‘GETR’	****	16#0A
Command code	Response code	Terminator

Receive message at error process for undefined command

‘ICMD’	****	16#0A
Command code	Response code	Terminator

(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. The TCP connection status variable *TcpStatus* of the SktGetTCPStatus instruction is used to detect whether the close processing ended normally. When the close processing is operated abnormally, the next connect processing may not be performed normally. For the corrective actions of the TCP connection status errors, refer to *9.7.2 TCP Connection Status Errors and Corrective Actions*.

9.3.2. Time Monitoring Function

This section explains the time monitoring function of this program.

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

●Time monitoring function of the communication instruction processing

To avoid errors that keep a communications process executing without a stop, 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 (for 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. If the next response does not arrive from the destination device within the maximum waiting time, it is detected that the receive processing ended.

[Receive waiting time]

Processing	Monitoring	Variable name	Maximum waiting time
Receive wait	Interval to receive data	TrTime	300 ms (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 the 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 be still operating. For information on the situation and corrective actions, refer to *9.7.2. TCP Connection Status Error and Corrective Actions*.

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 the 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
uErrorFlgs	UNION	Error code processing union
BoolData	ARRAY[0..15] OF BOOL	2-byte error code is handled 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	Communication start switch The program is started when this variable 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 and 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 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	The status of built-in EtherNet/IP port communications 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
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.

[Instances for communications instructions]

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).

●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 to send
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 BOOL	An area that stores the receive data of the SktTCPRcv instruction (2000 bytes)
Local_ReceiveMessage	STRING[256]	An area that stores the receive data that was converted into a string. (256 characters)
Local_ReceiveSize	UINT	The size of the receive data of the SktTCPRcv instruction
Local_RecvDataLength	UINT	The total byte length of the receive data
Local_RecvCHNo	UINT	The array number of the receive data stored in Local_RecvData
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 communications processing 1.2. Clearing the communications processing status flags 1.3 Communications processing in progress status	The communications processing is started.
2. Initialization processing	2.1. Initializing the 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 send data from a string to a BYTE array 2.9. Initializing the receive data storage areas 2.10. Initialization setting end processing	The Ethernet parameters are set and the error code storage areas are initialized. The send/receive required/not required setting, 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 is performed 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 is performed when the send processing required/not required setting is set to 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 is performed when the receive processing required/not required setting is set to Required and the send processing ends normally. If multiple receive data arrive, the receive processing is repeated. The receive data is stored and checked.

Major classification	Minor classification	Description
6. Close processing	6.1. Determining the close processing status and setting the execution flag 6.2. Enabling the close instruction monitoring timer 6.3. Executing the close instruction 6.4. Executing the TCP socket status read processing	The close processing is performed. The processing is performed in the following cases. •When the receive processing required/not required setting is set to 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
7. Processing number error process	7. Processing number error process	The error process is performed when a non-existent processing number was detected.

9.5.2. Program List

This section shows the details on 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. For information on how 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 general-purpose Ethernet communications connection check program
   Version: V1.00 New release November 29, 2012
   (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 communications processing
    Start 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: To 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 setting the 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 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;  //To 11:connect processing
ELSE
   Local_Status.Busy:=FALSE;
   Local_Status.Error:=TRUE;
   Local_State:=0;  //To 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;  //To 14: Close processing

      (* 3.1.2. Normal end processing *)
      ELSIF SktTCPConnect_instance.Done THEN
         Local_ErrCode.BoolData[2]:= FALSE;
         Output_SktCmdsErrorID:=WORD#16#0000;
         Local_ExecFlgs.Open:=FALSE;
         Local_TONflgs.Topen:=FALSE;
         IF Local_ComType.Send THEN
            Local_State:=12;  //To 12: Send processing
         ELSIF Local_ComType.Recv THEN
            Local_State:=13;  //To 13: Receive processing
         END_IF;

      (* 3.1.3. Error end processing *)
      ELSIF SktTCPConnect_instance.Error THEN
         Local_ErrCode.BoolData[2]:=TRUE;
         Output_SktCmdsErrorID:=SktTCPConnect_instance.ErrorID;
         Local_ExecFlgs.Open:=FALSE;
         Local_TONflgs.Topen:=FALSE;
         Local_State:=14;  //To 14: Close processing

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

      (* 3.2. Enabling 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;  //To 14: Close processing

      (* 4.1.2. Normal end processing *)
      ELSIF SktTCPSend_instance.Done THEN
         Local_ErrCode.BoolData[0]:=FALSE;
         Output_SktCmdsErrorID:=WORD#16#0000;
         Local_ExecFlgs.Send:=FALSE;
         Local_TONflgs.Tfs:=FALSE;
         Local_State:=SEL(Local_ComType.Recv,14,13);
                        //To 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;  //To 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);
         (* Set the destination device error determination instruction execution flag *)
         Local_RecvCheckFlg:=TRUE;
         Local_State:=14;  //To 14: Close processing

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

      (* 5.1.3. Normal end processing *)
      ELSIF SktTCPRcv_instance.Done THEN
         Local_RecvDataLength:=Local_RecvDataLength+SktTCPRcv_instance.RcvSize;
         Local_RecvCHNo:=Local_RecvDataLength;
         Local_ExecFlgs.Recv:=FALSE;
         Local_TONflgs.Tfr:=FALSE;
         Local_TONflgs.Tr:=TRUE;  //To 5.1.5. Receive data read processing

      (* 5.1.4. Error end processing *)
      ELSIF SktTCPRcv_instance.Error THEN;
         Local_ErrCode.BoolData[1]:=TRUE;
        Output_SktCmdsErrorID:=SktTCPRcv_instance.ErrorID;
         Local_ExecFlgs.Recv:=FALSE;
         Local_TONflgs.Tfr:=FALSE;
         Local_State:=14;  //To 14: Close processing

      (* 5.1.5. Receive data read processing *)
      ELSIF SktGetTCPStatus_instance.Done
            OR SktGetTCPStatus_instance.Error THEN
         Local_ExecFlgs.Status:=FALSE;
            (* When there is data to read: Receive processing continues. *)
         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, no processing is performed,
              and Read TCP socket status is re-executed at the next period.
           -When data is already received,
              if the maximum receive waiting time has elapsed and there is no more data,
              the receive processing is ended after reading the data that was already received. *)

(* 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 TCP socket status read 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;  //To 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;  //To 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;  //To 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. Enable 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 TCP socket status read 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;  //To 0: Communications not in progress status

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

    END_CASE;

END_IF;
```

9.5.3. Detailed Description of Function Blocks

The 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 RFID system V750 series
   Version: V1.00 New release November 29, 2012
   (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#7090;             // Destination port No.

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

   (* Maximum waiting time of packet interval when a response, which is
      divided into multiple packets, is received.
      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 RFID system V750 series
   Version: V1.00 New release November 29, 2012
   (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:=CONCAT('GETR',' typ', ' fwv');   // Destination device command: GETR
   Send_Check:= '';                  // FCS calculation : None
   Send_Terminate:= '$L';            // Send terminator: LF(0x0A): 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]

Name	Data type	Description
Receive_Check	STRING[5]	FCS receive value: FCS receive result of the receive data
Calc_Check	STRING[5]	FCS calculation value: FCS calculation result of the receive data

[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 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

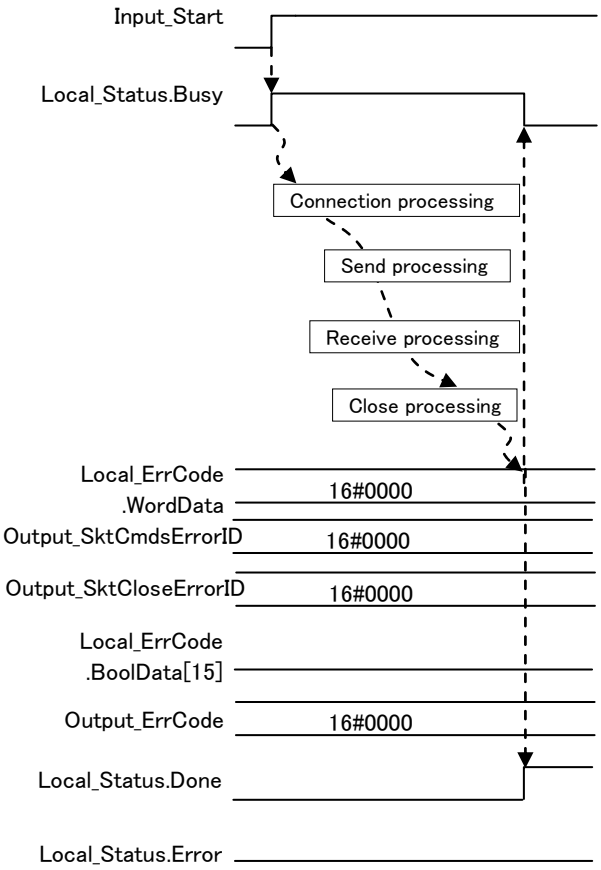
[Program]

```
(* ============================================================
   Name: NJ-series general-purpose Ethernet communications receive processing function block
   Applicable device: OMRON Corporation RFID system V750 series
   Version: V1.00 New release November 29, 2012
   (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
      Normal: 5th to 8th characters from the start of the data is '0000' *)
   IF EQascii(MID(Recv_Buff, UINT#4, UINT#5), '0000') THEN
   (* No FCS detection *)
   (* Normal end *)
      Error:= FALSE;                        // Error flag reset
      ErrorID:= WORD#16#0000;               // Error code clear
      ErrorIDEx:= DWORD#16#00000000;        // Destination device error code clear

   (* Error: 5th to 8th characters from the start of the data is not '0000' *)
   ELSE
      Error:=TRUE;                          // Error flag set
      ErrorID:=WORD#16#1000;                // Error code set

            (* Store the destination device error code
       Convert 5th to 8 characters from the left of the string ASCII code to hexadecimal *)
      ErrorIDEx:= STRING_TO_DWORD (MID(Recv_Buff, UINT#4, UINT#5));

   END_IF;
END_IF;

RETURN;
```

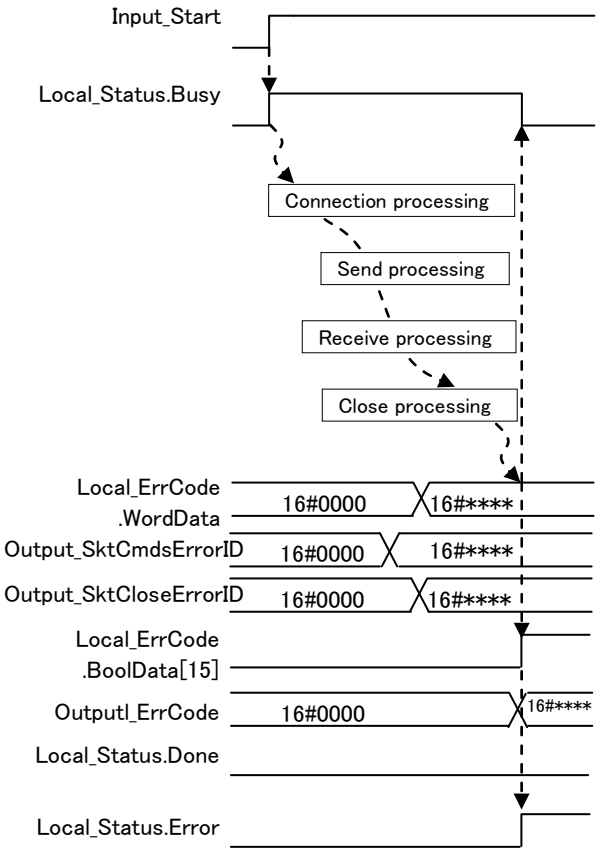
9.6. Timing Charts

The timing charts of this program are shown below.

●Start & End processing

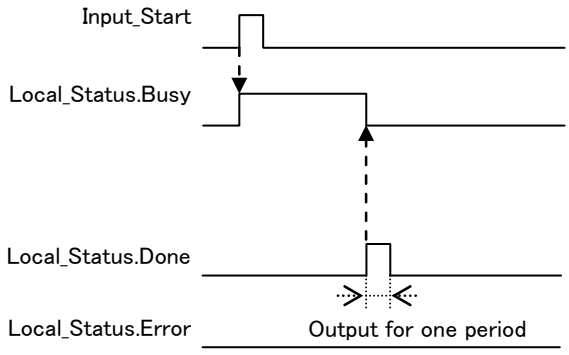


(Normal end)

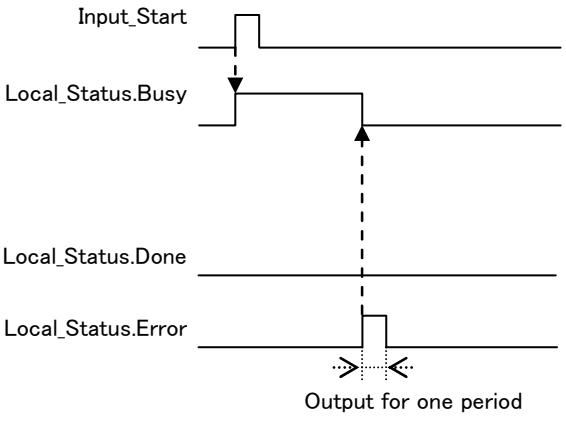


(Error end)

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.

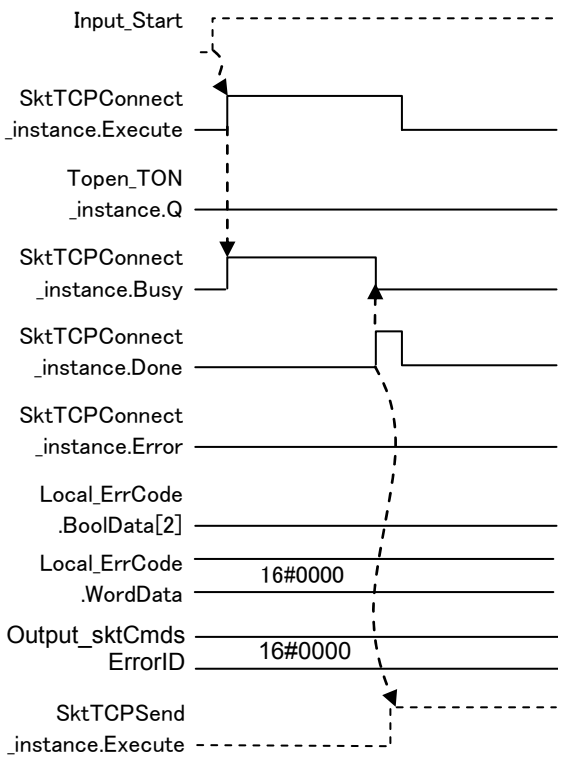


(Normal end)

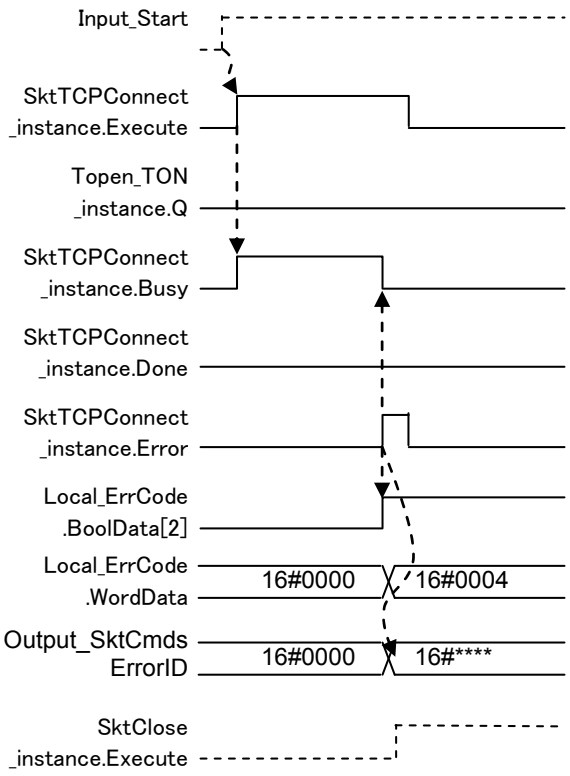


(Error end)

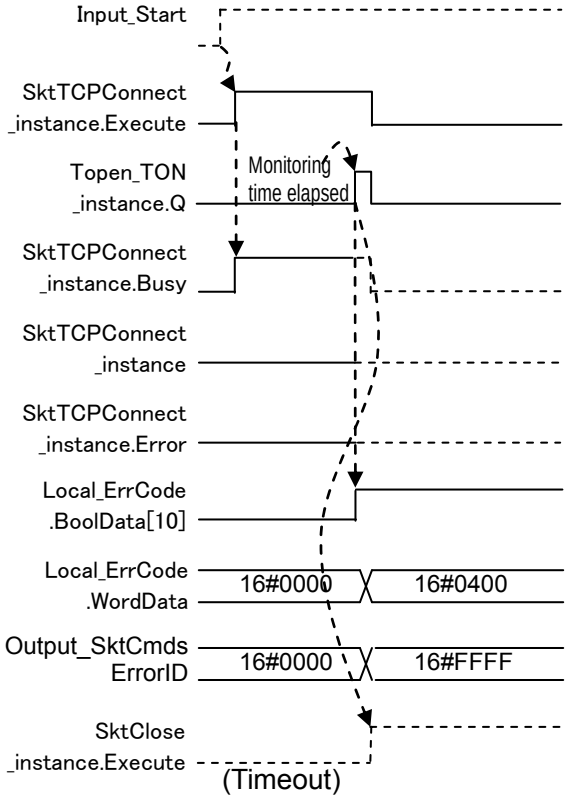
●Connect processing



(Normal end)

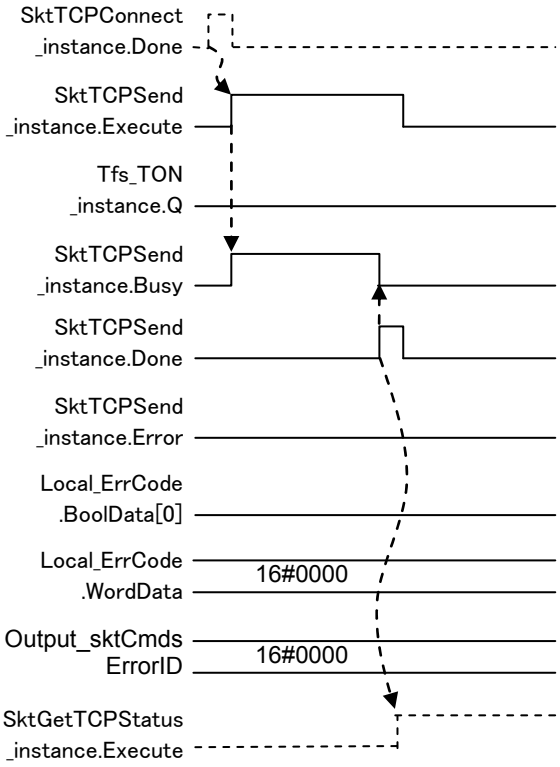


(Error end)

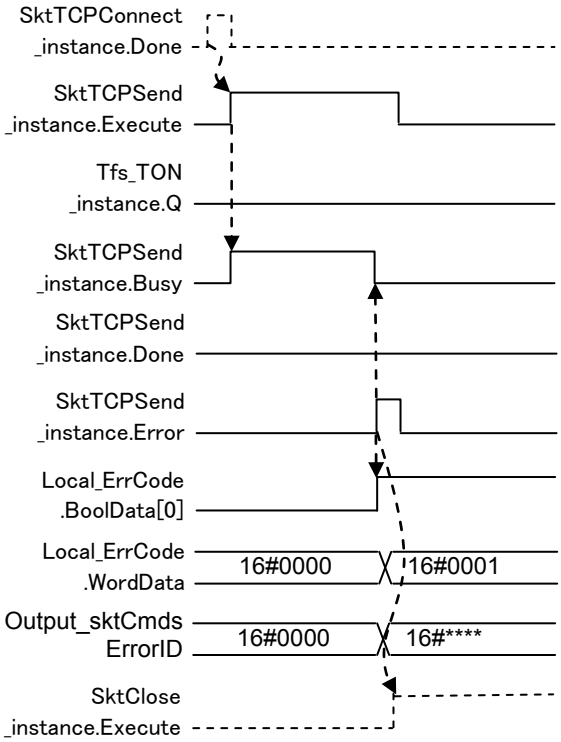


(Timeout)

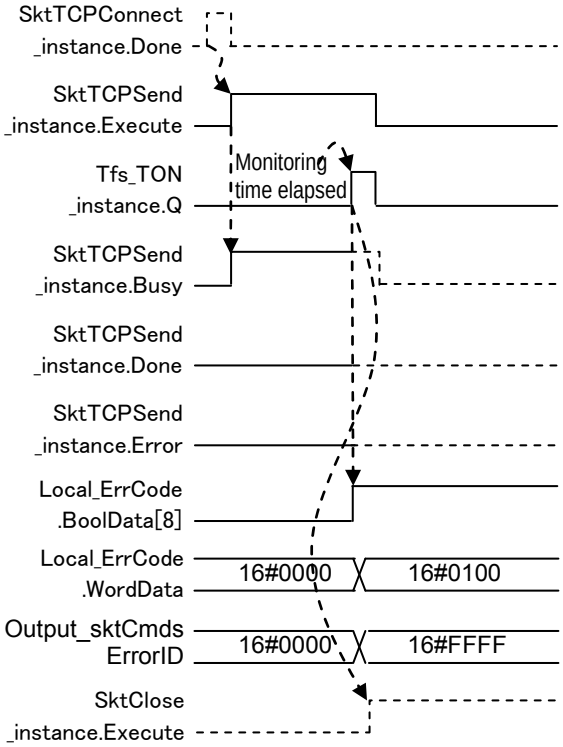
●Send processing



(Normal end)

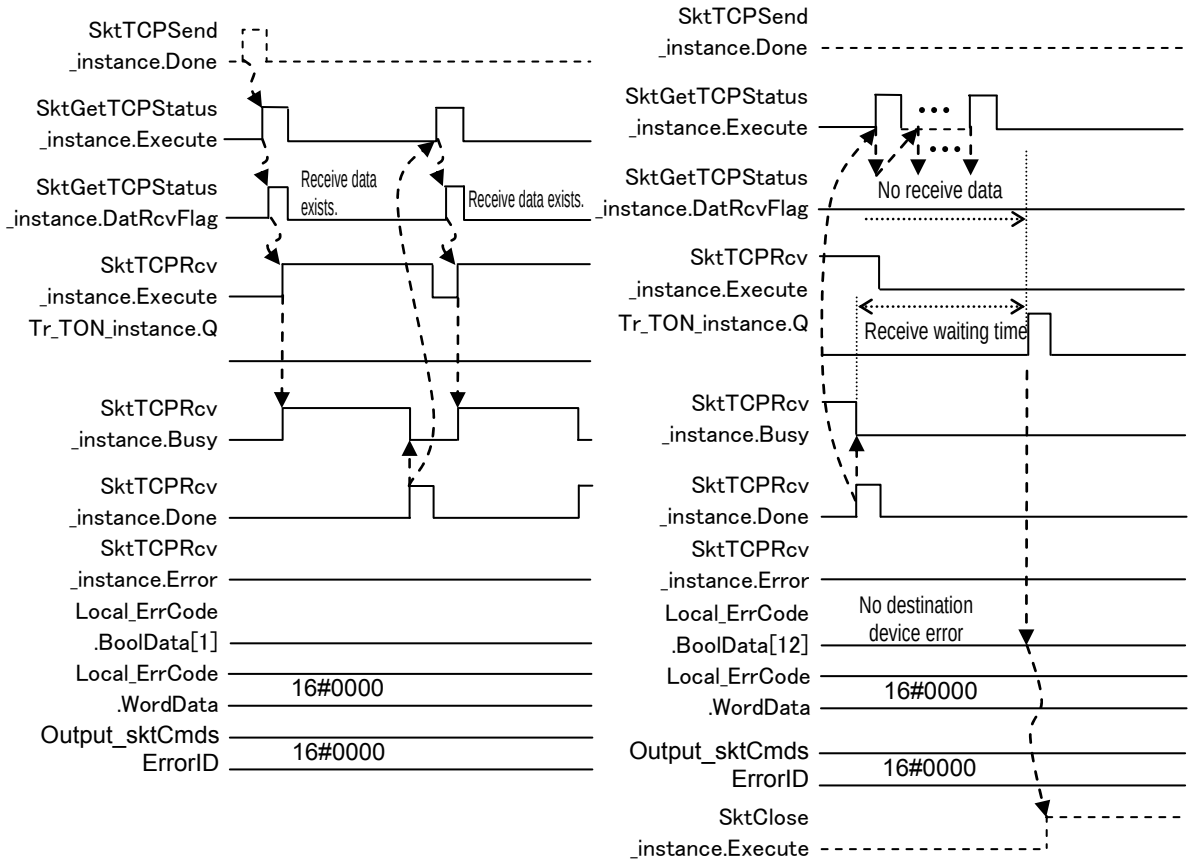


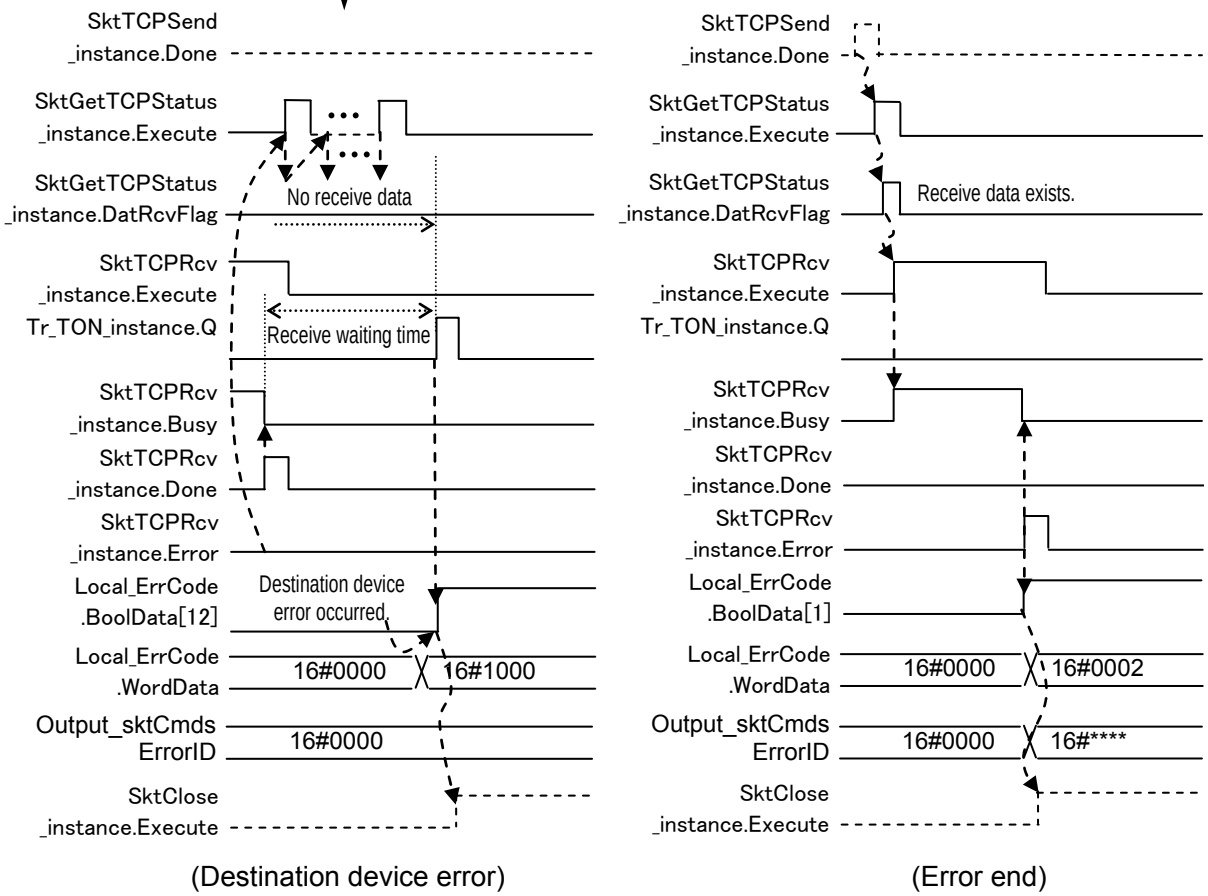
(Error end)

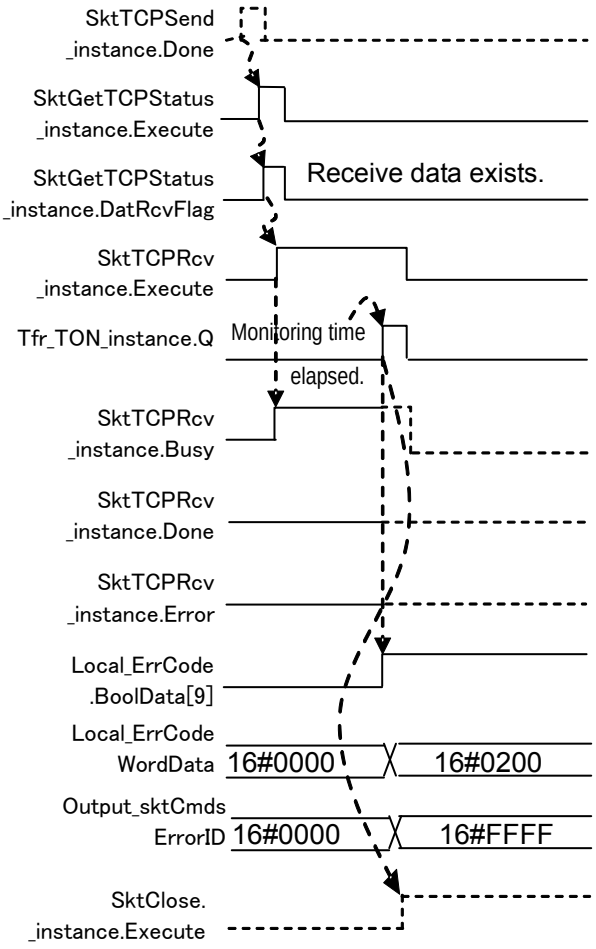


(Timeout)

●Receive processing

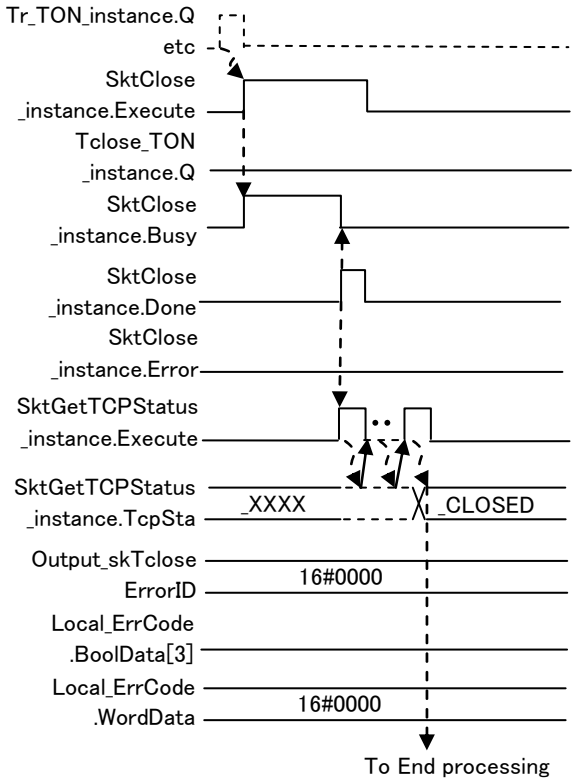




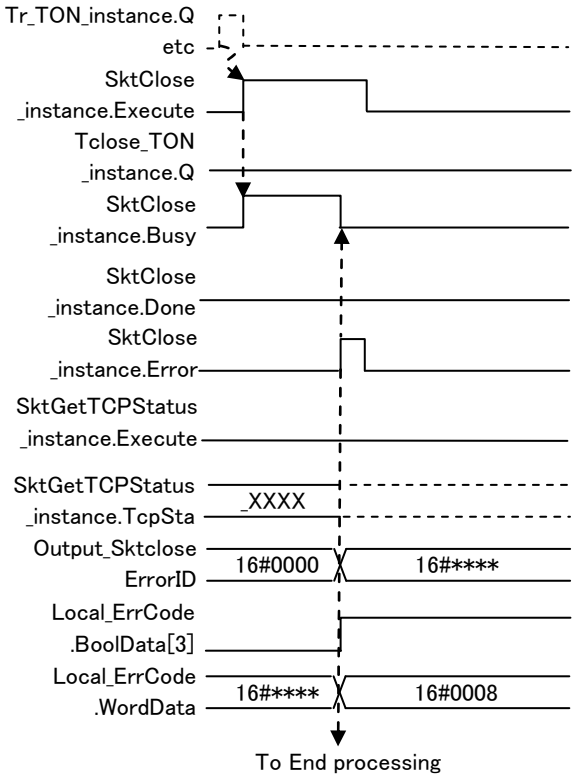

SktTCPSend
_instance.Done
SktGetTCPStatus
_instance.Execute
SktGetTCPStatus
_instance.DatRcvFlag
Receive data exists.
SktTCPRcv
_instance.Execute
Tfr_TON_instance.Q
Monitoring time
elapsed.
SktTCPRcv
_instance.Busy
SktTCPRcv
_instance.Done
SktTCPRcv
_instance.Error
Local_ErrCode
.BoolData[9]
Local_ErrCode
WordData
16#0000
16#0200
Output_sktCmds
ErrorID
16#0000
16#FFFF
SktClose.
_instance.Execute

(Timeout)

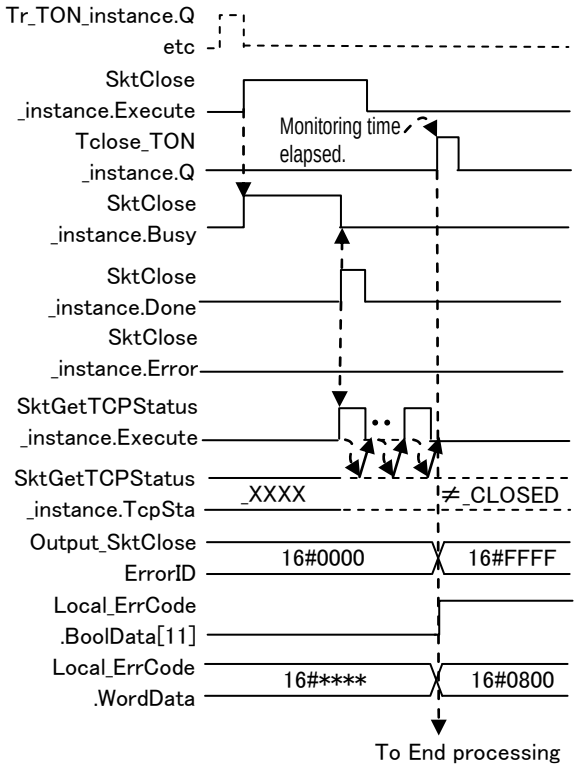
●Close processing



(Normal end)



(Error end)



(Timeout)

9.7. Error Process

9.7.1. Error Code List

The error codes for 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 timed 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) The connect TCP socket instruction error end + Close status check timeout

WORD#16#8000 (Error occurrence)
+WORD#16#0001 (TCP socket connect instruction ended in error)
+WORD#16#0100 (Close status check timeout)

↓

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 ends 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 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. (If the close processing ends in error, check this also.)

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

●Destination device error code

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

When 16#2000 is stored in *Output_ErrCode*, the FCS value of the data received from the destination device is stored in Output_MErrCode.

When 16#1000 is stored in *Output_ErrCode*, the error number is stored in *Output_MErrCode* as the destination device error code.

Bit	31 ... 24	23 ... 16	15 ... 8	7 ... 0
	16#0000		Response code	
			16#**:Main	16#**:Sub

Category	Response Code		Response Name	Description
	Main	Sub		
Normal end	00	00	Normal end	The received command ended normally with no error.
Command error	10	00	Parity error	A parity error has occurred in one of the characters of the command frame (For only RS-232C).
	11	00	Framing error	A framing error has occurred in one of the characters of the command frame (For only RS-232C).
	12	00	Overrun erro	An overrun error has occurred in one of the characters of the command frame (For only RS-232C).
	13	00	FCS error	The command frame has an incorrect FCS (For only RS-232C).
	14	0X (See Note1)	Command code error	Incorrect command has been received. The response code is ICMD.
		1X (See Note1)	Command parameter erro	Command parameter is incorrect.
		2X (See Note1)	Command option error	Command option is incorrect.
	15	00	Process error	Specified command can not be executed. Ex. Caused by executing a communication command when the last command is being executed. Ex. Caused by incorrect setting of filtering condition.
		0X (See Note1)	Filter error	Specified filter settins is incorrect. Ex. Caused by incorrect setting of filtering condition.
	18	00	Frame length error	A command received from the host exceeds the receive buffer (512 Bytes).
RF Tag communication error	70	00	LBT busy error	Channel none by can LBT use. (The electric wave cannot be sent.)
		1X (See Note1)	Communication error	During the transaction after tag detection, communication error or process time out has occurred, and consequently the transaction can not be completed normally. Specified password does not match to the one of the target tag.
		2X (See Note1)	Communication error	During the transaction after tag detection, communication error or process time out has occurred, and consequently the transaction can not be completed normally ." In the case of ID write/Data write, a part of data in the tag may have been written.
	71	00	Verification error	The reader has not written the data to the tag by reason of verification error.
	7A	00	Address specification error	Specifying Bank/Address in the tag memory is incorrect and command can not be executed.
	7B	00	Data write error	During the data write into the detected tag, sufficient power is not supplied to the tag.
	7C	1X (See Note1)	Antenna derection error	At the R/W starts up, an appropriate antenna has not been connected to the specified antenna port.
		2X (See Note1)	Antenna error	Error occurred with the antenna connected to the specified antenna port (even though the antenna is detected normally when start up).
	7E	00	Lock error	When data write or read command is sent for the locked area. It depends on the tag's chip specifications. (For Monza chip, when these commands are sent for Lock Bit of User Memory because this area does not exist.)(See Note2)
	7F	0X (See Note1)	Tag error	The tag has been rejected the command process.
System error	9A	XX (See Note1)	System error	An error that blocks command execution has been detected in the hardware (such as malfunction of inner circuit or temporary execution error caused by noise).

Note1: 'x' character in response code means one character in the list of 0 to 9 or A to F.

Note2: Depends on the specification of IC chip equipped in the RF tag. (It occurs at Monza chip when it specified the lock bit which does not exist in its memory map.

Additional Information

For details and troubleshooting the destination device errors, refer to *Section 7 Troubleshooting Alarms and Errors* in the *V750-series UHF RFID System User's Manual* (Cat. No. Z235).

9.7.2. TCP Connection Status Error and Corrective Actions

This section describes the situation in which the TCP connection status error occurs and explains the corrective actions.

●Affects of the TCP connection status error

After a TCP connection status error occurs, if this program is executed again without 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*.)

●Situation in which the TCP connection status error occurs

Both a 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 (It is not confirmed that the close processing is completed at the destination device).

●Corrective actions

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

Precautions for Correct Use

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

●State of the Controller at a TCP connection status error

When a TCP connection status error occurs, the processing of this 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 is restarted or the power supply is turned OFF

10. Revision History

Revision code	Date of revision	Revision reason and revision page
01	2013/04/15	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. P543-E1-01 0911(-)