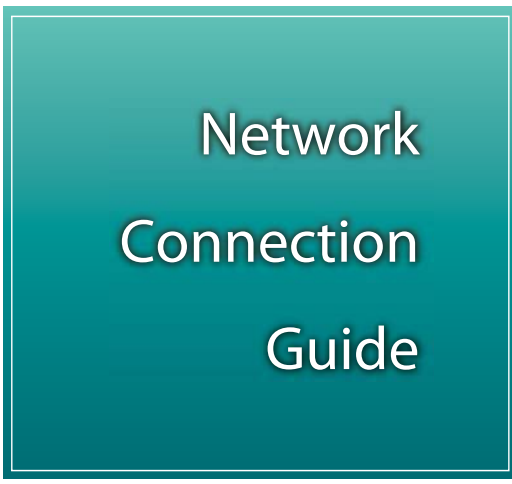
OMRON

Machine Automation Controller
NX Series

General Ethernet (TCP/IP) Connection Guide

Smart camera
F430-F Series



SYSMAC
always in control

Z443-E-01

About Copyrights and Trademarks

Contents

1. Related Manuals

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

The following Omron Corporation (hereinafter referred to as "Omron") manuals are related to this document:

Manual No.	Model	Manual Name
W535	NX Series	NX Series CPU Unit User's Hardware Manual
W593	NX Series	NX Series NX102 CPU Unit Hardware User's Manual
W578	NX Series	NX Series NX1P2 CPU Unit Hardware User's Manual
W501	NJ/NX Series	NJ/NX Series CPU Unit Software User's Manual
W506	NJ/NX Series	NJ/NX Series CPU Unit Built-in EtherNet/IP Port User's Manual
W504	SYSMAC-SE2□□□	Sysmac Studio Version 1 Operation Manual
W502	NJ/NX Series	NJ/NX Series Instructions Reference Manual
Z433	F430-F Series	Smart Camera F430-F Series User Manual
Z444	F430-F Series	Smart Camera F430-F Series User Manual for Communications Settings

2. Terms and Definitions

Below is a list of terms used in this manual and their definitions.

Term	Description / Definition
IP Address	Ethernet communicates using an IP address. An IP address (Internet Protocol address) is an address for identifying a node (host computer, controller, etc.) on Ethernet. IP addresses must be set and managed so that there are no duplicate addresses on the network.
Socket	A socket is an interface that allows you to directly use TCP or UDP functions from the user program. The NJ/NX Series machine automation controller performs socket communication using the standard socket service instruction. To use the socket service, you need to establish and disconnect a connection with the other node. In this document, the establishment process is called "socket open" or "TCP open", and the disconnection process is called "socket close" or "close". Socket service allows you to send and receive arbitrary data with another node.
Active and Passive	When a TCP socket connection is established, open processing is executed on each node. Currently, the open method differs depending on whether the node becomes a server or a client. In this document, processing when opening as a server is called "Passive Open", and processing when opening as a client is called "Active Open" or "Open Processing (Active)".
Keep-alive Function	If the TCP / IP socket service does not communicate with the other node (server or client) for longer than the set time, the keep-alive communication frame is used to check the connection with the other node. If there is no response, confirmation will be performed at regular intervals, and if there is no response to all confirmations, the connection will be disconnected.
Linger	It is an option of TCP socket that enables open processing by the same port number immediately without sending RST data at TCP socket close and waiting for port number release. If the linger option is not specified, FIN data will be issued when TCP is closed, and termination management such as delivery confirmation will be performed with the other node in about 1 minute. For this reason, it may not be possible to use the TCP socket of the same port number immediately.

3. Restrictions and Precautions

(1) Understand the specifications of devices which are used in the system. Allow some margin for ratings and performance.
(2) Provide safety measures, such as installing a safety circuit, in order to ensure safety and minimize the risk of abnormal occurrence.
(3) To ensure system safety, make sure to always read and follow the information provided in all Safety Precautions and Precautions for Safe Use in the manuals for each device which is used in the system. The user is encouraged to confirm the standards and regulations that the system must conform to.
(4) It is prohibited to copy, to reproduce, and to distribute a part or the whole of this document without the permission of OMRON Corporation.
(5) The information contained in this document is current as of January 2020.
It is subject to change for improvement without notice.

The following notations are used in this document.

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

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

Precautions for Safe Use

Precautions on what to do and what to avoid doing to ensure the safe use of the product.

Precautions for Correct Use

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

Note

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

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

4. Overview

This document describes the procedures for connecting the Omron smart camera (F430-F Series) to the NX Series Machine Automation Controller (hereinafter referred to as Controller) via EtherNet/IP), and for verifying their connections.
Ethernet communication connection can be made by understanding the setting procedure and the point in setting through Ethernet communication setting of the project file prepared in advance.
In this project file, the Ethernet connection is checked by sending a measurement execution command to the smart camera and receiving the measurement data from it.

Please prepare the latest file of "Sysmac Studio project file" from Omron beforehand.

Name	Filename	Version
Sysmac Studio Project File (Extension: SMC2)	OMRON_F430_NX_ETN(TCP)_V100.SMC2	Ver.1.00

CAUTION

The purpose of this document is to explain the wiring method, the communication settings required for connection of the devices and the step by step setting procedure. The program included with the document was created using the same procedures and can be used to confirm that all the settings have been made correctly and that a connection can be established (Confirm Connection). It is not a program created on the premise of constant use in the field, so its functionality and performance aspects have not been fully considered. When building an actual system, it is necessary to refer to the wiring method and communication setting contents and setting procedures described in this document, and to design a new program according to the customer's purpose for the program.

5. Applicable Devices and Device Configuration

5.1. Applicable Devices

The applicable devices that can be connected are as follows:

Manufacturer	Name	Model	Version
OMRON	NX Series CPU Unit	NX701-□□□□ NX102-□□□□ NX1P2-□□□□	Same or higher version as indicated in section 5.2.
OMRON	Smart camera	F430-F□□□□□□□-□□□	

Note

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

Note

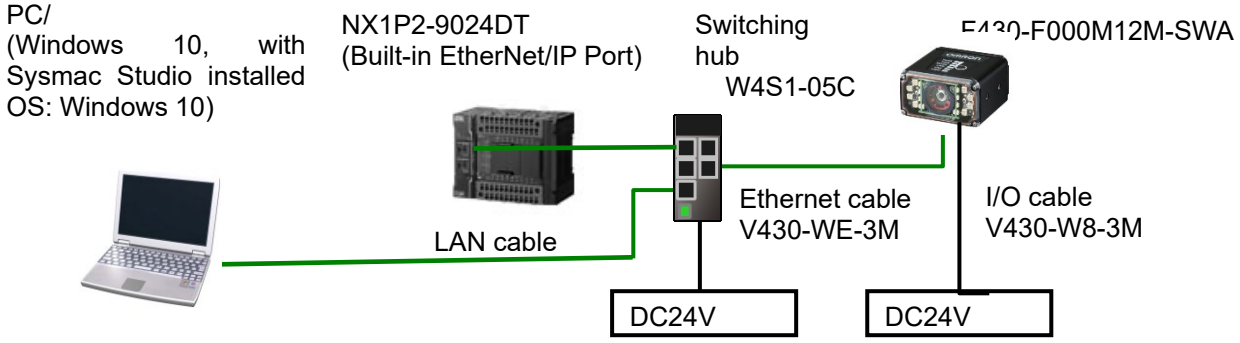
In this document, from among the above target devices, connection confirmation is performed using the devices listed in section 5.2. When using a device that is not described in section 5.2. Among the above target devices, check the connection referring to the contents of this document.

Precautions for Correct Use

In this document, the devices with models and versions listed in section 5.2.
You cannot use devices with versions lower than the versions listed in section 5.2.
To use the above devices with models not listed in 5.2. or versions higher than those listed in 5.2., check the differences in the specifications by referring to the manuals before operating the devices.

5.2. Device Configuration

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



Manufacturer	Name	Model	Version
OMRON	NX Series CPU Unit (Built-in EtherNet/IP Port)	NX1P2-9024DT	Ver.1.16
OMRON	Switching hub	W4S1-05C	
OMRON	Sysmac Studio	SYSMAC-SE2□□□	Ver.1.29
OMRON	Sysmac Studio Project File	OMRON_F430_NX_EIP_V100.csm2	Ver.1.00
-	Computer (OS: Windows 10)	-	
-	LAN cable (STP (shielded, twisted-pair) cable of Ethernet category 5 or higher)	-	
OMRON	Smart camera	F430-F000M12M-SWA	Ver.5.2.0
OMRON	I/O cable	V430-W8-3M	
OMRON	Ethernet cable	V430-WE-3M	
-	DC24V Power supply (Smart camera)	-	

Precautions for Correct Use

Please prepare the latest file of "Sysmac Studio project file" from Omron Corp. beforehand. (Contact Omron for information on how to obtain these files.)

Note

If the device configuration or versions are different, it may not be reproducible. After confirming the configuration, models and versions, if it is different from your configuration, please contact Omron.

Note

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

Note

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

Note

Refer to the "Smart Camera F430-F Series User Manual" (Z433) for the power supply specifications that can be used for DC24V power supply (for Smart camera).

6. Ethernet Settings

This manual explains communication parameter specifications and cable wiring.

Note

This document and this project file can only operate with the settings and commands described in this chapter. For communication other than this setting, it is necessary to modify the project file.

6.1. Ethernet Communications Settings

The settings for establishing Ethernet communication are as follows.

6.1.1. Communication setting between setting PC and Smart Camera

In this document, the setting procedure of the smart camera by the setting PC is explained using the setting contents in the table below as an example.

Setting item	On PC used for settings	Smart camera
IP Address	192.168.188.100	192.168.188.2 (default)
Subnet mask	255.255.0.0	255.255.0.0 (default)
Gateway	Blank (default)	0.0.0.0 (default)

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

6.1.2. Communication Setting Between Ethernet Unit and Smart Camera

The connection procedure of the Ethernet unit and the smart camera is explained using the settings in the table below as an example.

Setting item	NX1P2-9024DT (Built-in EtherNet/IP Port)	Smart camera
IP Address	192.168.188.1	192.168.188.2 (default)
Subnet mask	255.255.0.0	255.255.0.0 (default)
Gateway	-	0.0.0.0 (default)
Port number	(Set in the software)	49211 (Default)

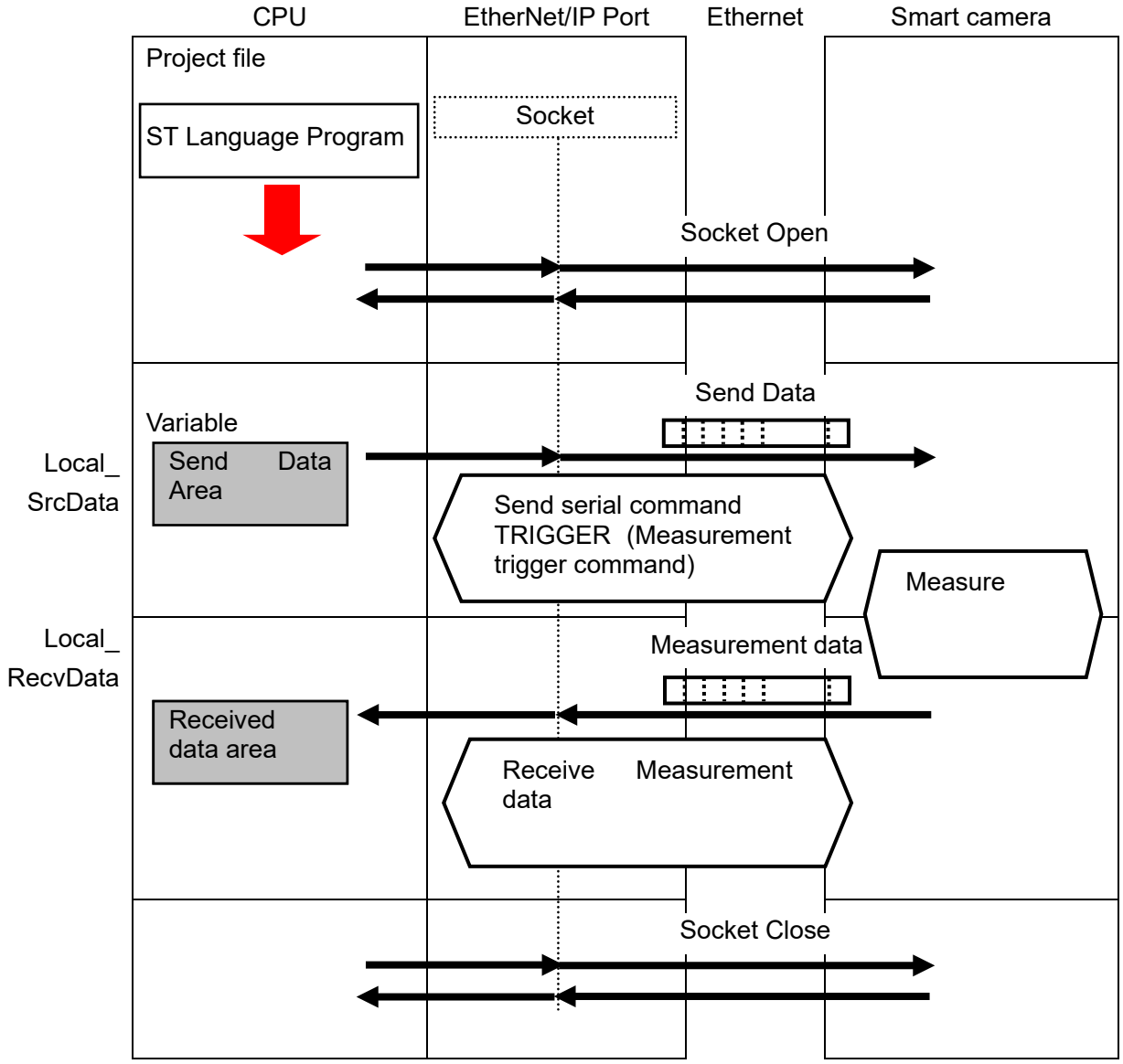
* For the use cases in this document, setting the gateway is unnecessary because the devices are connected within the same segment of the network.

6.2. Communication Verification Example

In this document, a program in structured text (hereinafter, ST) language is used as an example when executing "socket open", "send / receive" and "socket close" from the controller to the smart camera.

The controller sends an “inspection trigger command” to the smart camera. The smart camera sends its output data to the controller.

The Overview of the operation is shown below.



7. Steps for Connecting

This section describes the procedures for setting up an Ethernet connection on the controller (PCL).

The explanations of procedures for setting up the PLC and smart camera given in this document are based on the use of the factory default settings. If re-initializing the devices is required, refer to Section 8. Initialization Method.

7.1. Operation Flow

Use the following procedures to set up an Ethernet connection on the controller (PLC).

7.2. Smart Camera Setup	Set up the Smart camera.
▼	
7.2.1. Install the (AutoVision) software	Install the (AutoVision) software application used for setting up the Smart camera.
▼	
7.2.2. Set Parameters	Set the parameters for the Smart camera.
▽	
7.3. Controller Setup	Set up the Controller.
▼	
7.3.1 Start Sysmac Studio and load the project file	Start the "Sysmac Studio" (Automation Software) and load the Project File in to Sysmac Studio.
▼	
7.3.2 Verify parameters and execute builds	Check the configuration parameters and execute program check and build of the project data.
▼	
7.3.3 Connect online and transfer project data	Connect online in Sysmac Studio and transfer the project data to Controller.
▽	
7.4 Check the Connection Status	Execute the transferred project file and check that Ethernet communication is performed correctly.
▼	
7.4.1. Run the project file and confirm the receive data	Run the project file and check that the correct data is written to the controller variable.

Precautions for Correct Use

Please prepare the latest file of "Sysmac Studio project file" from Omron Corp. beforehand. (Contact Omron for information on how to obtain these files.)

7.2. Smart Camera Setup

Set up the Smart camera.

Precautions for Correct Use

Use a PC (personal computer) to set the parameters for the smart camera.
Note that there may be some changes required for the PC settings depending on the current state of PC.

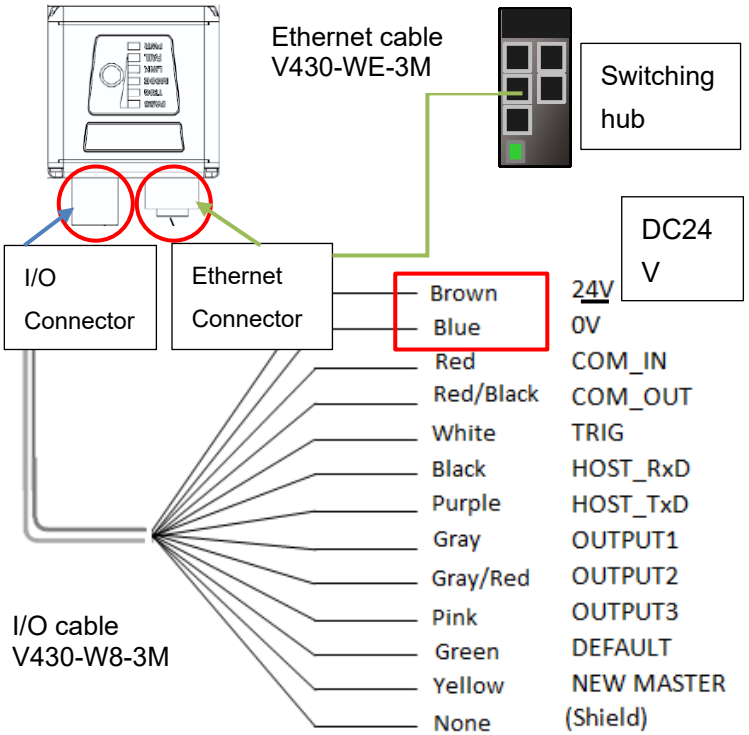
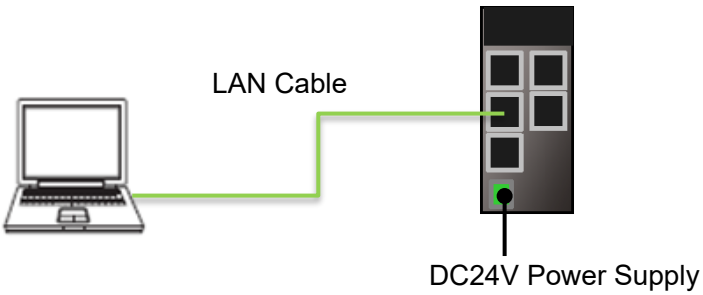
7.2.1. Install the (AutoVision) software

Install the (AutoVision) software application used for setting up the Smart camera. For more details on installing the Autovision software, please refer to the “AutoVision Quick Start Guide” (Z434).

7.2.2. Set Parameters

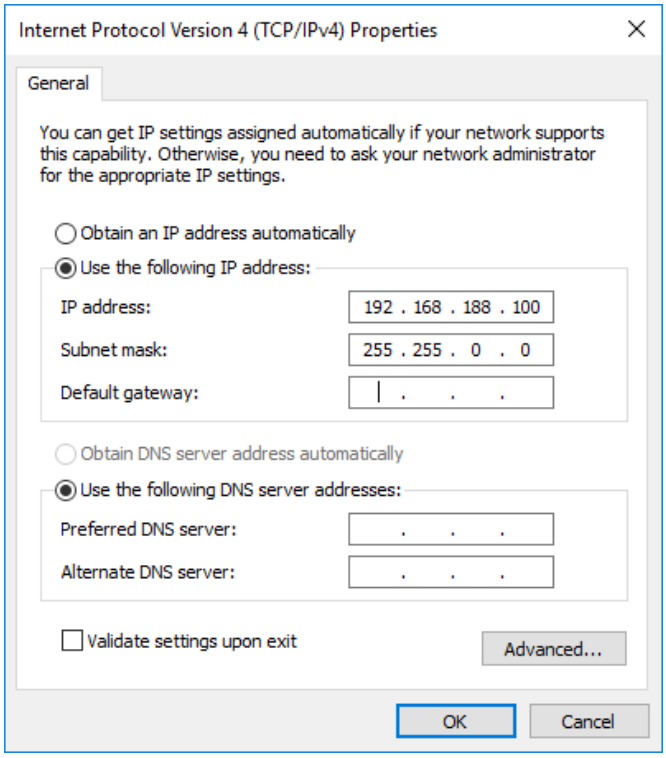
Set the parameters for the Smart camera.

Set the IP address of your computer to "192.168.188.100" and its subnet mask to "255.255.0.0".

1	Connect the [Ethernet connector] of the smart camera to the switching hub with the [Ethernet cable]. Connect the [I/O cable] to [I/O connector] and turn on the 24VDC power supply. ∗ This document is only for verification of the power supply wiring of the I/O cable connection. Be careful not to short-circuit any other wires. ∗ Ground the shielded wire as needed. For more information on Grounding, please refer to the "Smart Camera F430-F Series User Manual” (Z433) - "2-7 Grounding and Power"	
2	Connect the computer to the switching hub with the LAN cable. Connect 24 VDC power supply (for Switching hub) to Switching hub.	

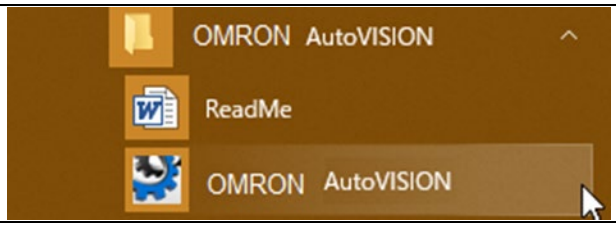
3 Set the IP Address of the PC.
Set the IP Address to "192.168.188.100"
set the subnet mask to "255.255.0.0".

Refer to Step 4 for the procedure to open the dialog on the right on a Windows 7 system.



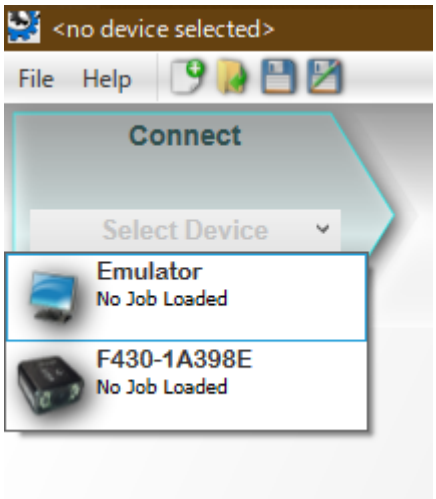
4 (1) From the Windows Start menu, select Control Panel - Network and Internet - Network and Sharing Center.
(2) Click on Local Area Connection. The Local Area Connection Status Dialog Box is displayed. Click Properties.
(3) In the [Local Area Connection Properties] dialog box, select [Internet Protocol Version 4 (TCP / IPv4)], and click the [Properties] button.
(4) Click the "OK" button

5 Launch the AutoVision software.



6 After starting AutoVision, if the smart camera is displayed in the device selection list, proceed to Step 8.

If the AutoVision startup screen does not appear, go to step 7.



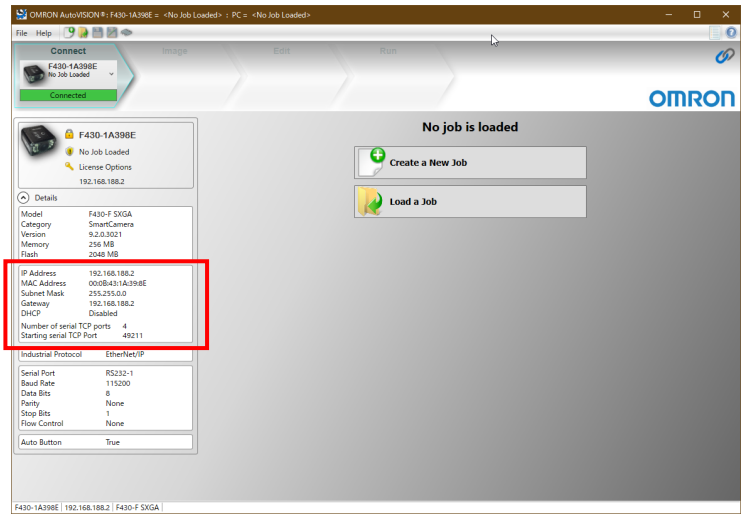
7 If the AutoVision startup screen does not appear, it means that communication between the smart camera and the PC has not been established so please check the following.

・Does the F430 and the PC have a proper physical (cable) connection?

・Are the respective IP Addresses on the PC and on the F430 smart camera set correctly?
→ Refer to 4. for setting the IP Address of the PC.
・Do a hardware reset of the F430.
→ When turning the power on, press and hold the setup button on the smart camera body until its light turns on.

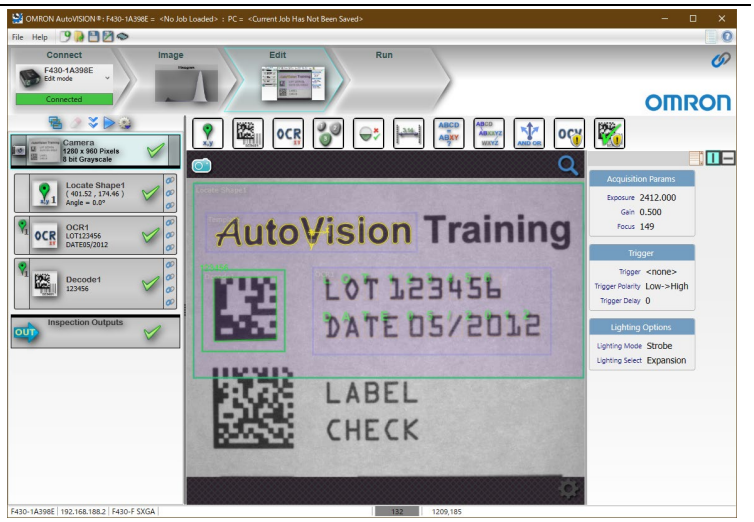
8 After selecting the smart camera, the settings screen will display.
Check the settings indicated by the red boxes.

If you need to change the IP address, for example when connecting multiple F430 devices, change the setting from [IP address] as necessary.

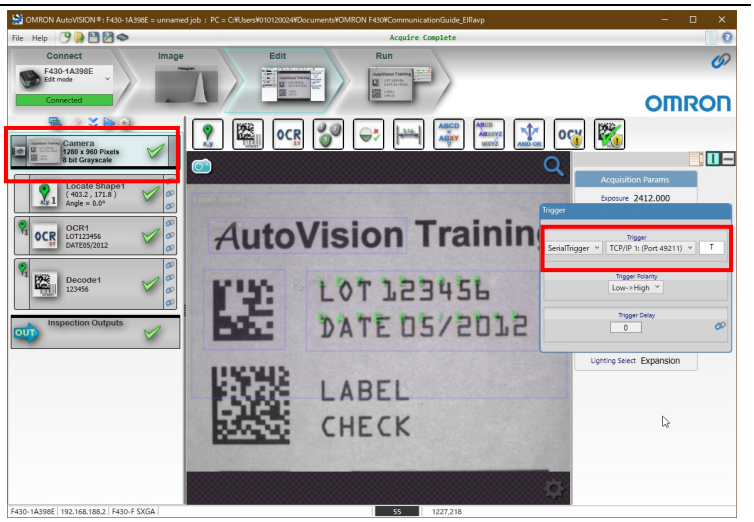


9 Create a new job and set the "Locate Shape", "OCR" and "Decode" tools.

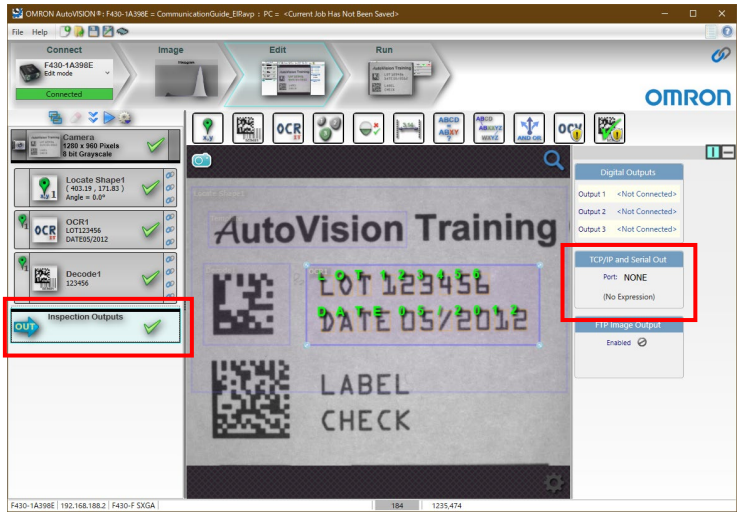
* In this chapter, you will create a job to output the detection points from the Locate Shape tool, the text string read by the OCR tool and character text decoded from a 2D Code using Serial(TCP/IP) communications.



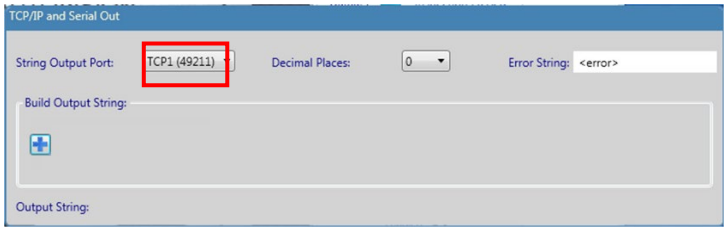
10 Select the camera tool and set the trigger to "Serial Trigger", "TCP / IP 1: (Port 49211)", "T".



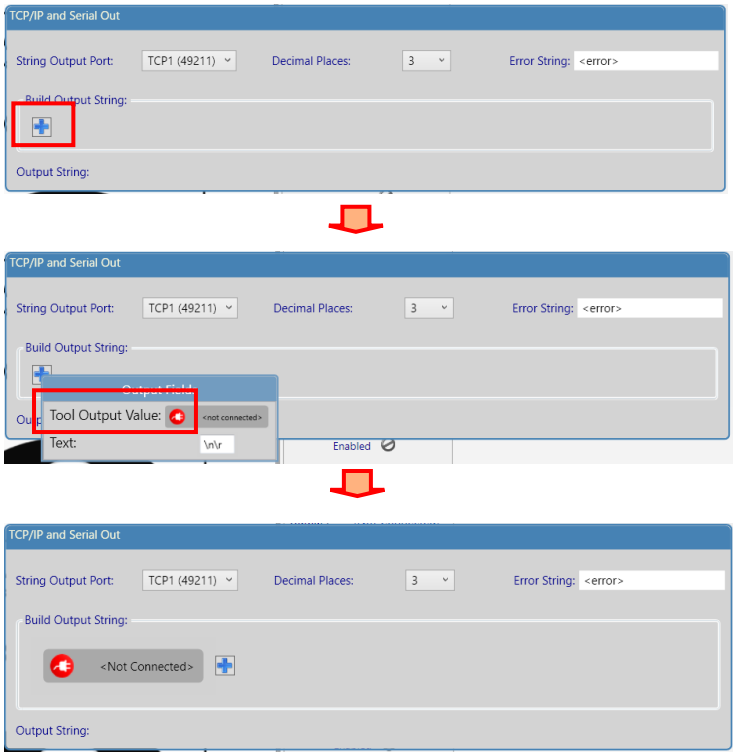
11 Click on Inspection Outputs and in its menu select “TCP/IP and Serial Out”.



12 In the popup dialog, set "String output port" to "TCP1 (49211)".



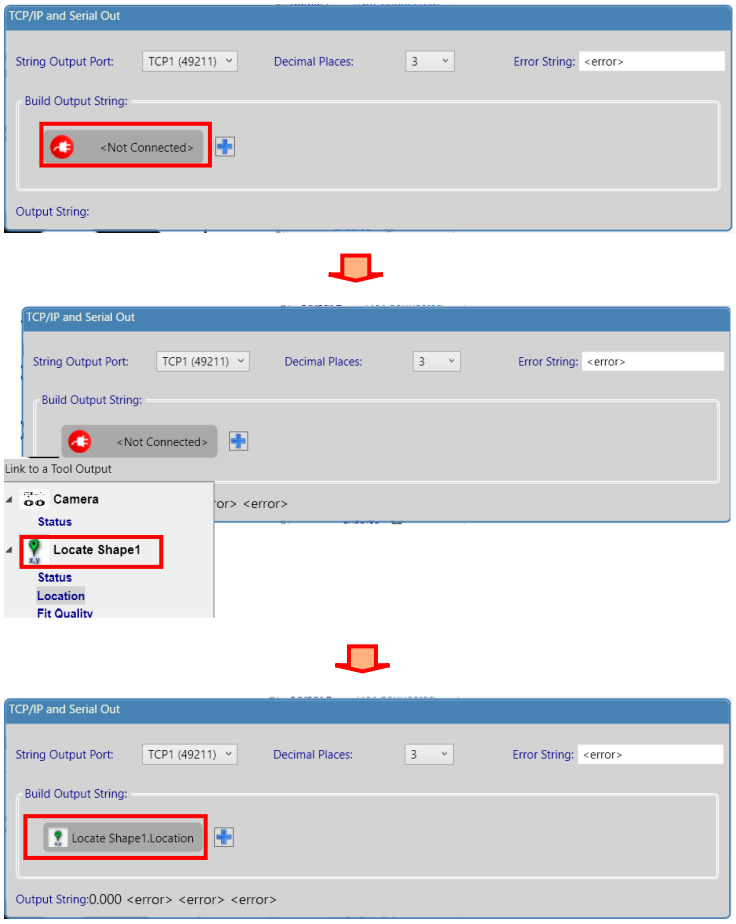
13 Click the  icon in the Output String and select Output Value.



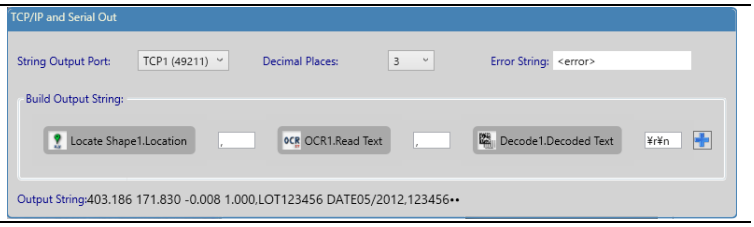
14 Click the icon in the red frame and select the data to output.

Here, select the location from “Locate Shape”.

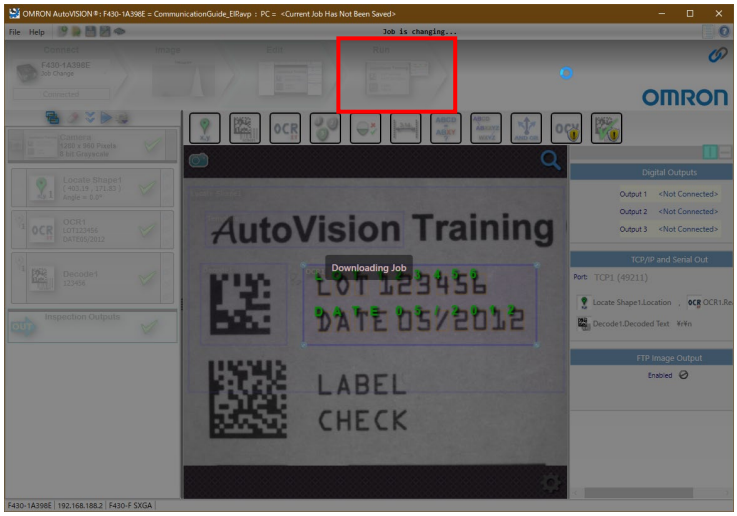
When selected, the current value is displayed in the "Output String" part of the red frame.



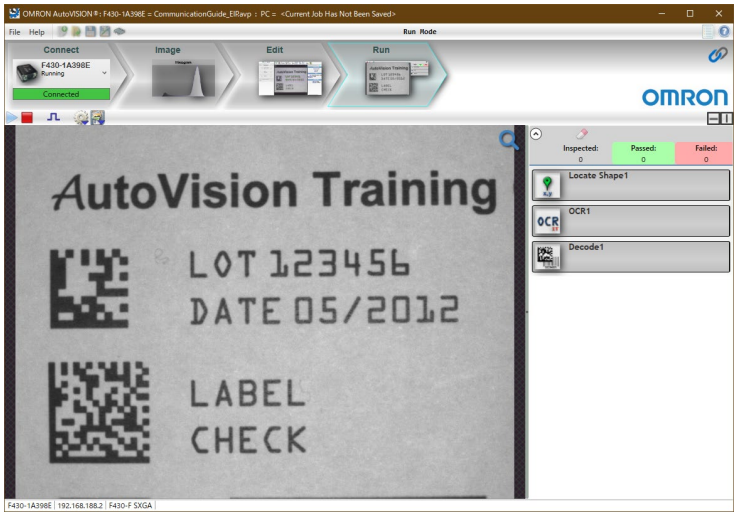
15 Repeat steps 13-14 to build the output string.



16 Go to Run view and download the job to smart camera.



17 The download is complete when you can successfully transition to the Run screen.



7.3. Controller Setup

Set up the Controller.

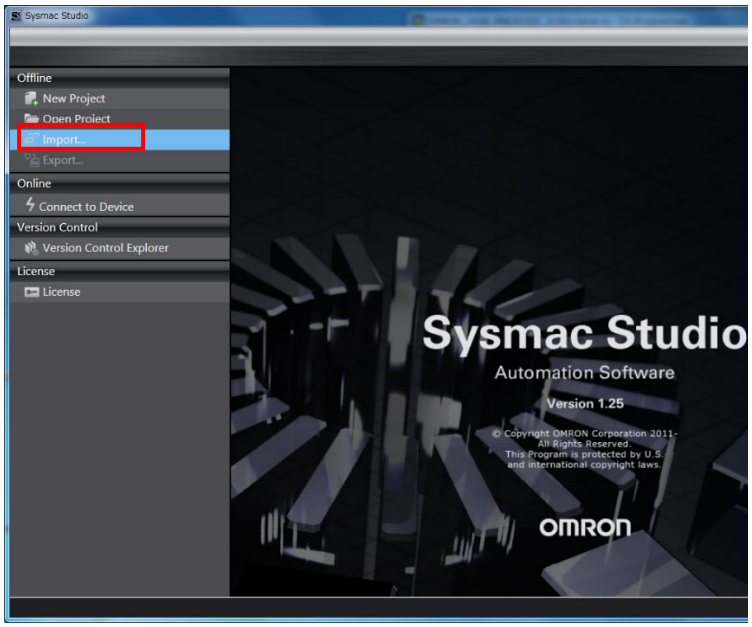
7.3.1. Start Sysmac Studio and Load the Project File

Start the "Sysmac Studio" (Automation Software) and load the Project File in to Sysmac Studio.

Install Sysmac Studio and USB driver on the computer beforehand. Also, connect the USB cable to the PC and the controller, and turn on the controller power.

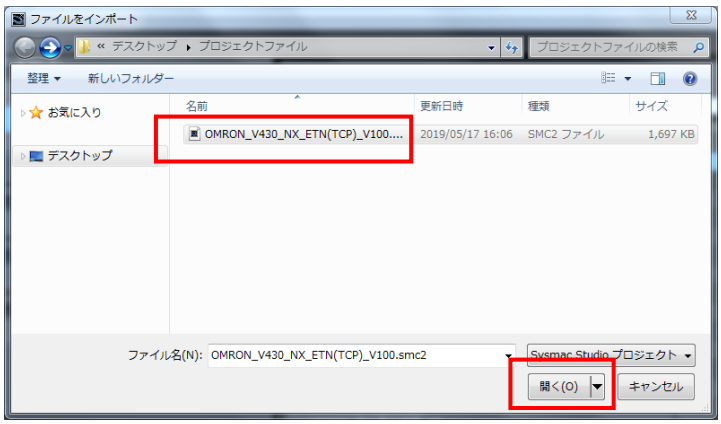
1 Start Sysmac Studio.
Click **Import**.

* If the User Account Control Dialog Box is displayed at startup, select the option to start.

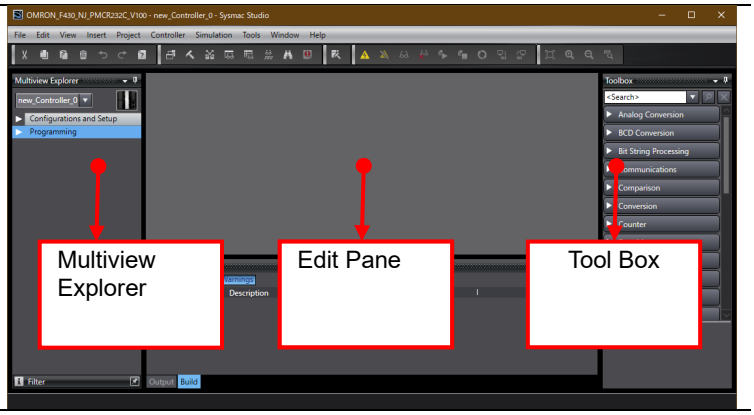


2 The [Import project] dialog is displayed. Select [OMRON_F430_NX_ETN (TCP)_V100.SMC2] (Sysmac Studio project file), and click [Open].

* About Sysmac Studio Sample project file for Smart camera F430-F Series Machine Automation Controller NX Series General Ethernet (TCP/IP) Connection Guide, contact your OMRON representative.



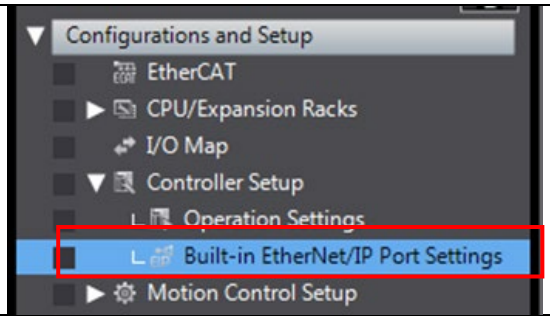
3 The [OMRON_F430_ETN(TCP)_V100] project screen is displayed. The left side of the screen is called "Multiview Explorer", the right side is called "Toolbox", and the center is called "Edit window".



7.3.2. Verify Parameters and Execute Builds

Check the configuration parameters and execute program check and build of the project data.

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



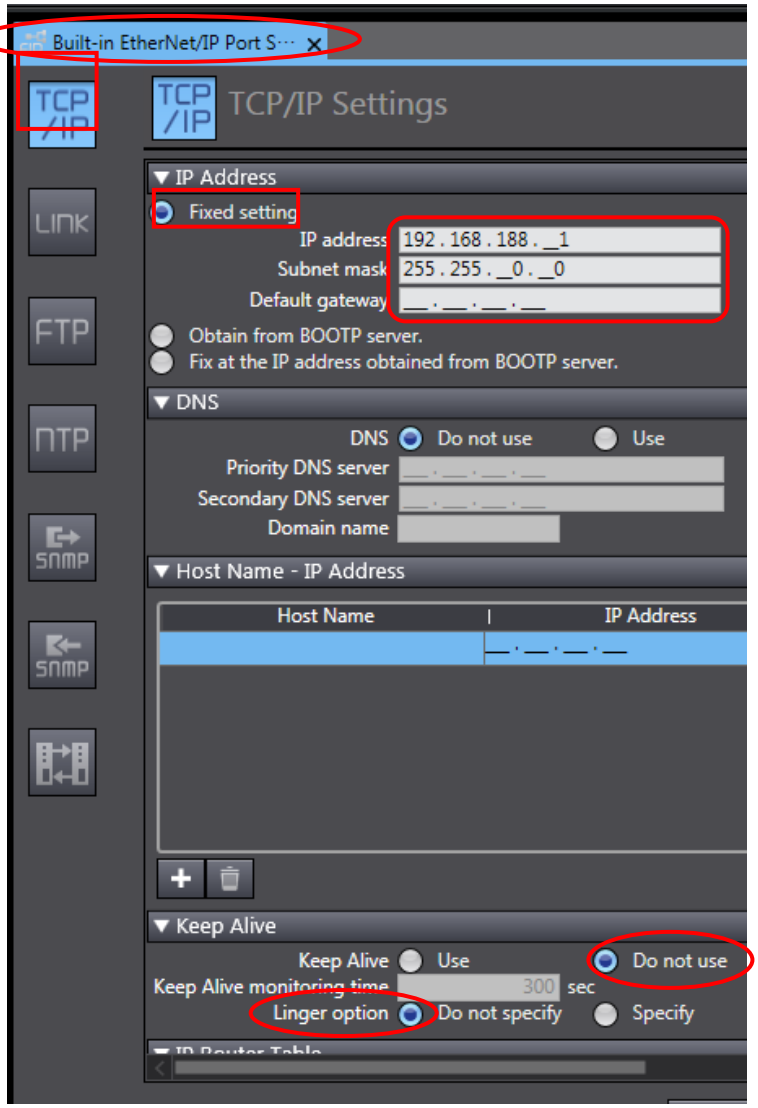
2 The Built-in EtherNet/IP Port Settings Tab Page is displayed in the Edit Pane.

Select [TCP/IP], select the [Fixed] check box for [IP Address], and check that the settings are as follows.

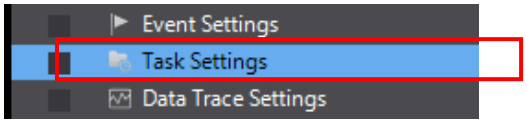
IP Address: 192.168.188.1
Subnet mask: 255.255.0.0
Default Gateway: _._._._

Verify the [KeepAlive] is set as shown here.

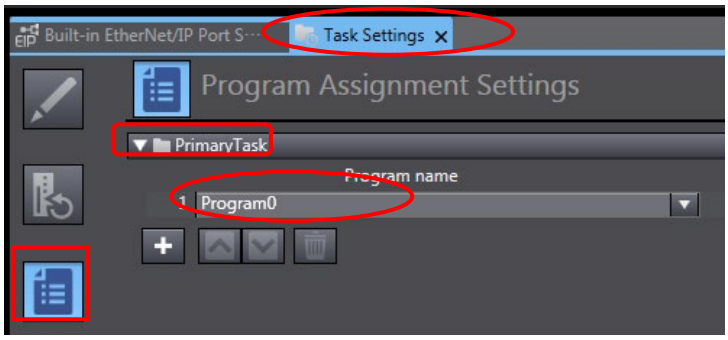
KeepAlive: Do not use
Linger option:
Do not use



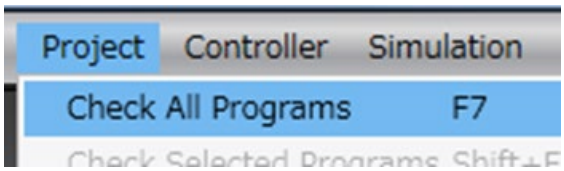
3 Double-click "Task Settings" under "Configurations and Setup" in the Multiview Explorer.



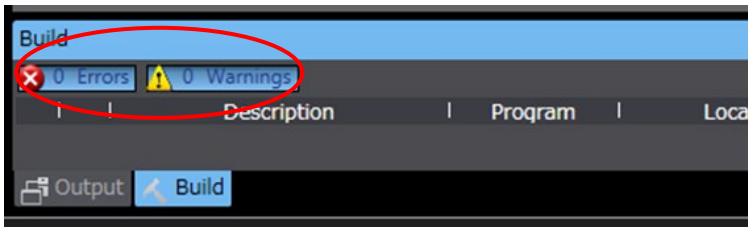
4 The "Task Settings" Tab Page is displayed in the Edit Pane.
Select [Program Assignment Settings] and confirm that [Primary Task] is set to [Program0].



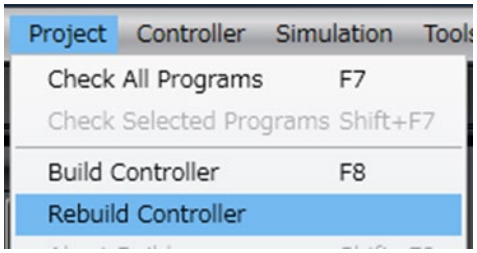
5 From the Main Menu in Sysmac Studio, select [Project] – [Check All Programs].



6 The [Build Tab Page] will be displayed under the [Edit Window].
Confirm that [0] is shown for both Errors and Warnings

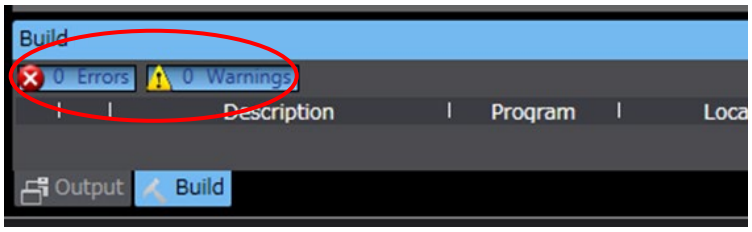


7 From the Menu Bar, select [Project] – [Rebuild Controller].



The build progress screen is displayed.

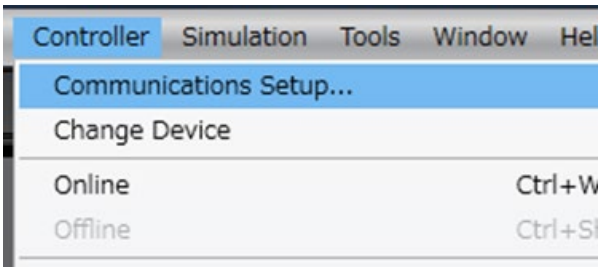
8 On the Build tab page, confirm that [0] is shown for both Errors and Warnings.



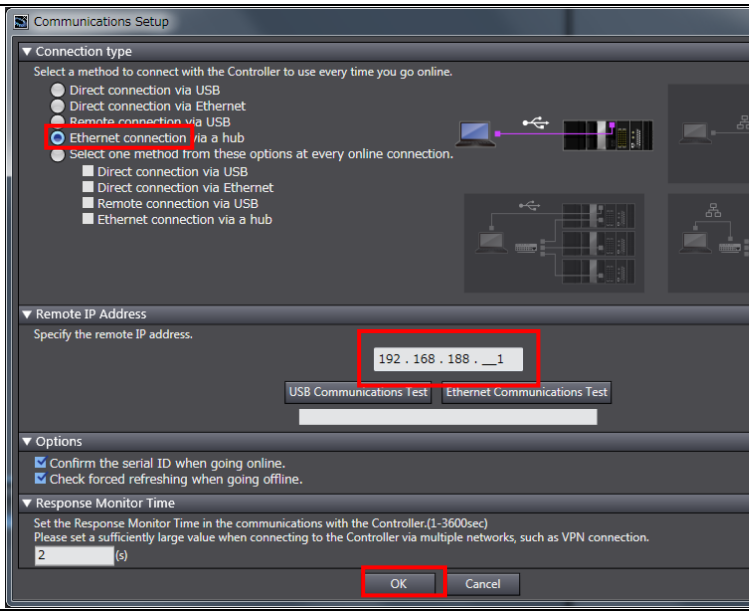
7.3.3. Connect Online and Transfer Project Data

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

1 From the Menu Bar, select [Controller] – [Communications Setup].



2 The [Communications Setup] dialog opens.
Make sure that [Ethernet connection via a hub] is selected for the [Connection Type].
Or enter [192.168.188.1] in [IP Address to connect to USB-Remote / Ethernet connection via a hub].
Press [Ethernet Communications Test] and confirm that [Communications test OK] is displayed.
Click [OK].

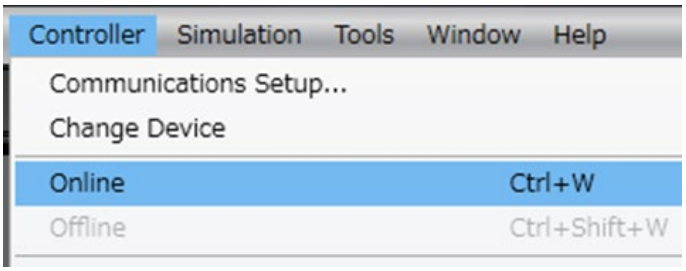


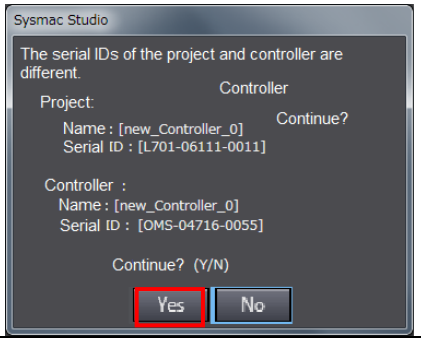
3 From the Menu Bar, select [Controller] – [Online].

The Confirm dialog is displayed.
Click [Yes].

* The dialog that is displayed differs depending on the status of the controller you are using, but make the selection to proceed with processing.

* The serial ID displayed differs by device used.

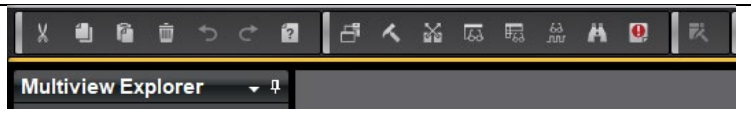




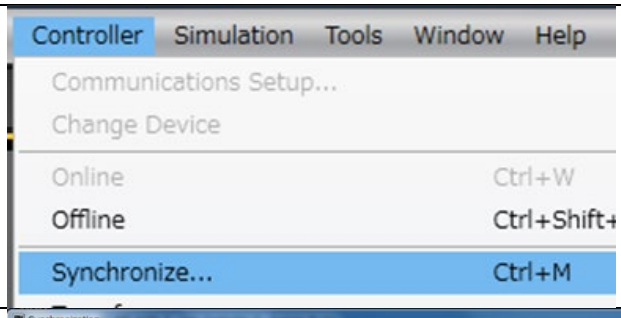
Note

Refer to Chapter 5 "Controller Configurations and Setup" in the "Sysmac Studio Version 1.0 Operation Manual" (W504) for details on online connection to the controller.

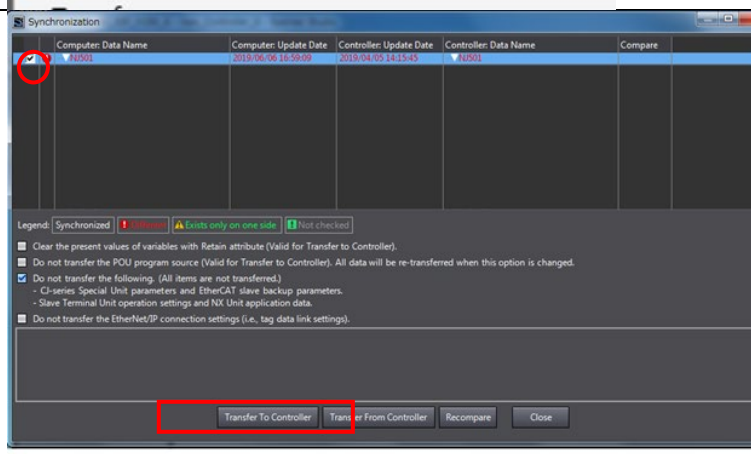
4 When you are online, a yellow frame will be displayed in the upper part of the [Edit window].



5 From the Menu Bar, select [Controller] – [Synchronize].



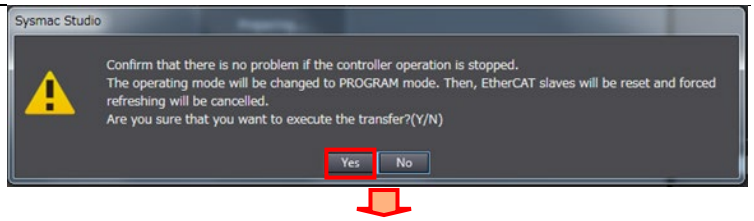
6 The [Synchronization] dialog opens.
Make sure that the data you want to transfer (in the right figure, [NX1P2]) is checked, and click [Transfer to Controller].

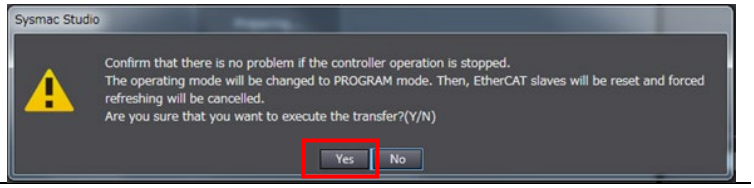


7 The Confirm dialog is displayed. Click [Yes].

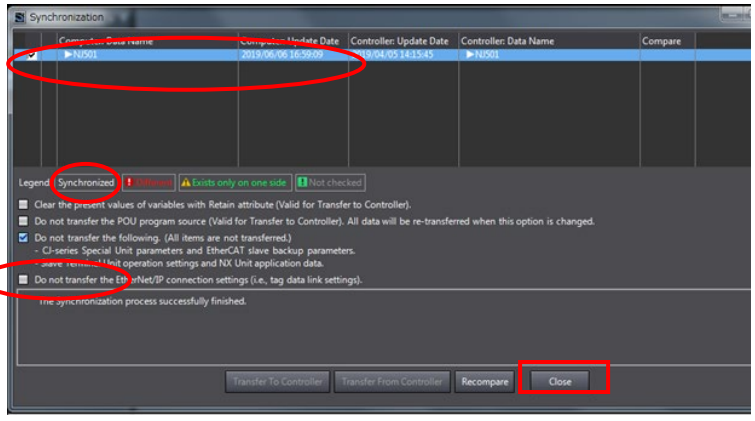
The "Synchronizing" dialog appears.

The Confirm dialog is displayed. Click [Yes].





8 The text color of the syncronized data changes as shown for [Synchronized].
Confirm that "The Synchronization process successfully finished" message is displayed.
If there is no problem, click [Close].
* If synchronization fails, check the wiring and re-execute the procedure in this section.



7.4. Check the Connection Status

Run the transferred project file and check that Ethernet communication is performed correctly.

Precautions for Correct Use

Before performing the following steps, confirm that the Ethernet cable is connected securely. If they are not connected, first turn the power to each device OFF and then connect the Ethernet cable.

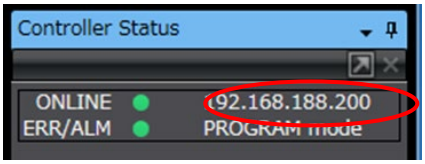
7.4.1. Run the project file and confirm the receive data

Run the project file and check that the correct data is written to the controller variable.

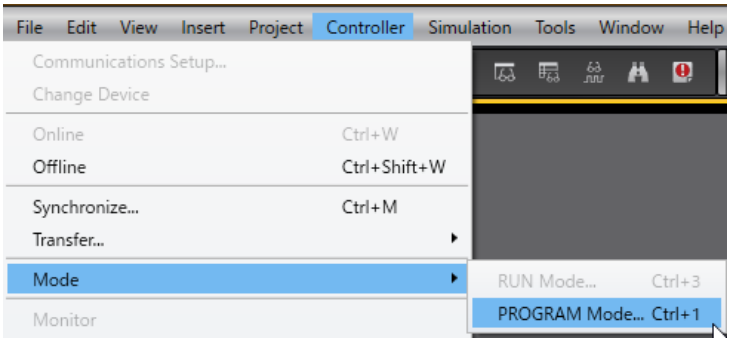
Precautions for Safe Use

Be sure to check the safety carefully before executing a project file.
Regardless of the operation mode of the unit, the connected equipment may malfunction and cause injury.

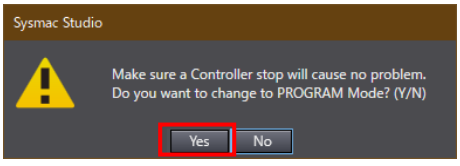
1 Confirm that [Operation mode] is displayed in the [Controller Status] window of Sysmac Studio.



If the mode is [Program mode], select [Controller]-[Operation mode]-[Run mode] from the menu bar.

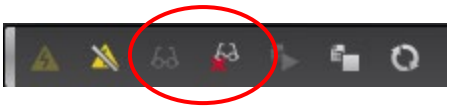


The Confirm dialog is displayed. Click [Yes].



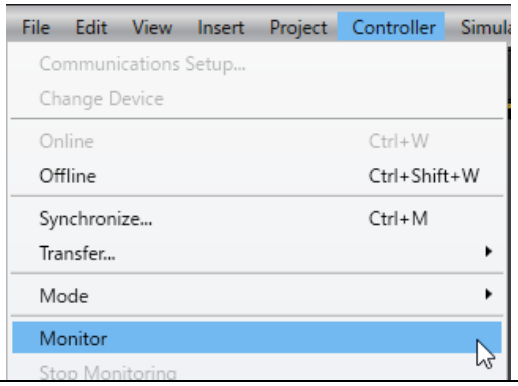
2 Confirm the monitoring status of the controller by selecting either [Monitor] or [Stop Monitoring] from the Sysmac Studio toolbar.
As shown in the right figure, check that the [Monitor] button is selected (Monitoring), and that the [Stop Monitoring] button is selectable.

* If the Monitoring state is currently stopped, from the Sysmac Studio menu bar select [Controller]—[Monitor].

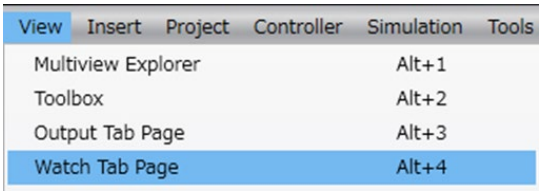


Monitor

Stop Monitoring



3 From Menu Bar in Sysmac Studio, select [View] - [Watch Tab Page].



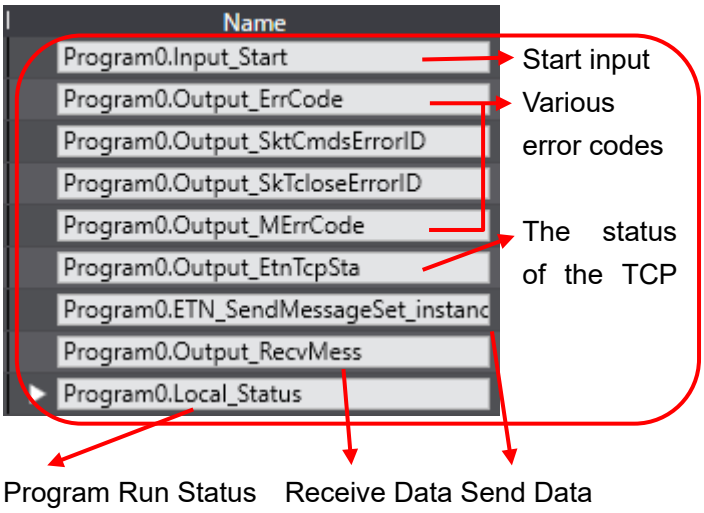
4 The [Watch Tab Page] is displayed under the [Edit Window].

5 Confirm that the variables shown on the right are displayed in the [Name] area.

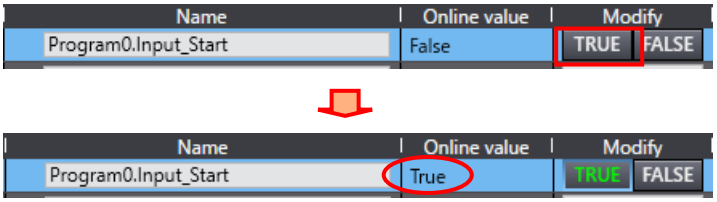
* If the required variable is not displayed, click [Input Name ...] to add it.

* In the following explanation, "Program0" in [Name] is omitted.

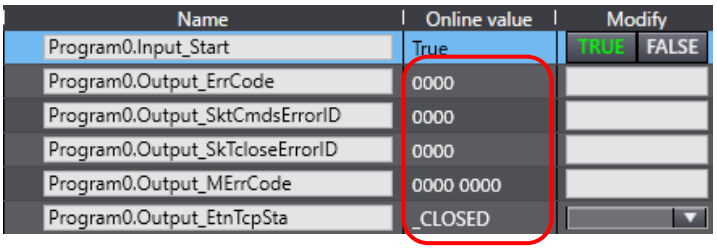
I



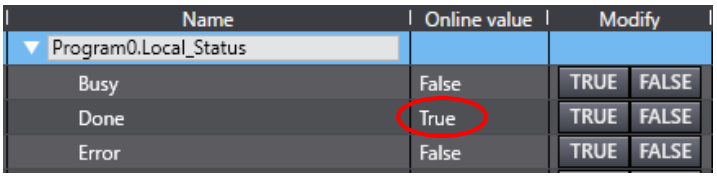
6 Click [TRUE] in the [Modify] area of [Input_Start].
The [Online value] for [Input_Start] becomes [True].
The program runs and performs Ethernet communication with the other device.



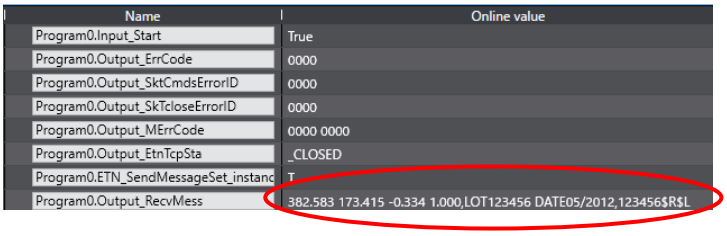
7 When communication completes normally, each error code will be "0".
The TCP Connection status (Output_EtnTcpSta) will become "CLOSED".
* If it terminates abnormally, an error code is stored according to the abnormality that occurred. Refer to section 9.7. "Error Processing".



In addition, [Online value] of [Local_Status.Done] indicating the program execution status will be [True]. If there is an abnormal termination, [Local_Status.Error] becomes [True].
* If [Input_Start] is set to [FALSE], each variable of [Local_Status] will also be [False]. For more information refer to section 9.6. "Timing Chart".



8 Response data received from the partner device is stored in Output_RecvMess.
(ETN_SendMessageSet_instance.Send_Data is the sent command)
As shown in the right figure, specify the area you want to reference in the [Watch Window] and check the selection.



* The content shown on the right depends on your environment.

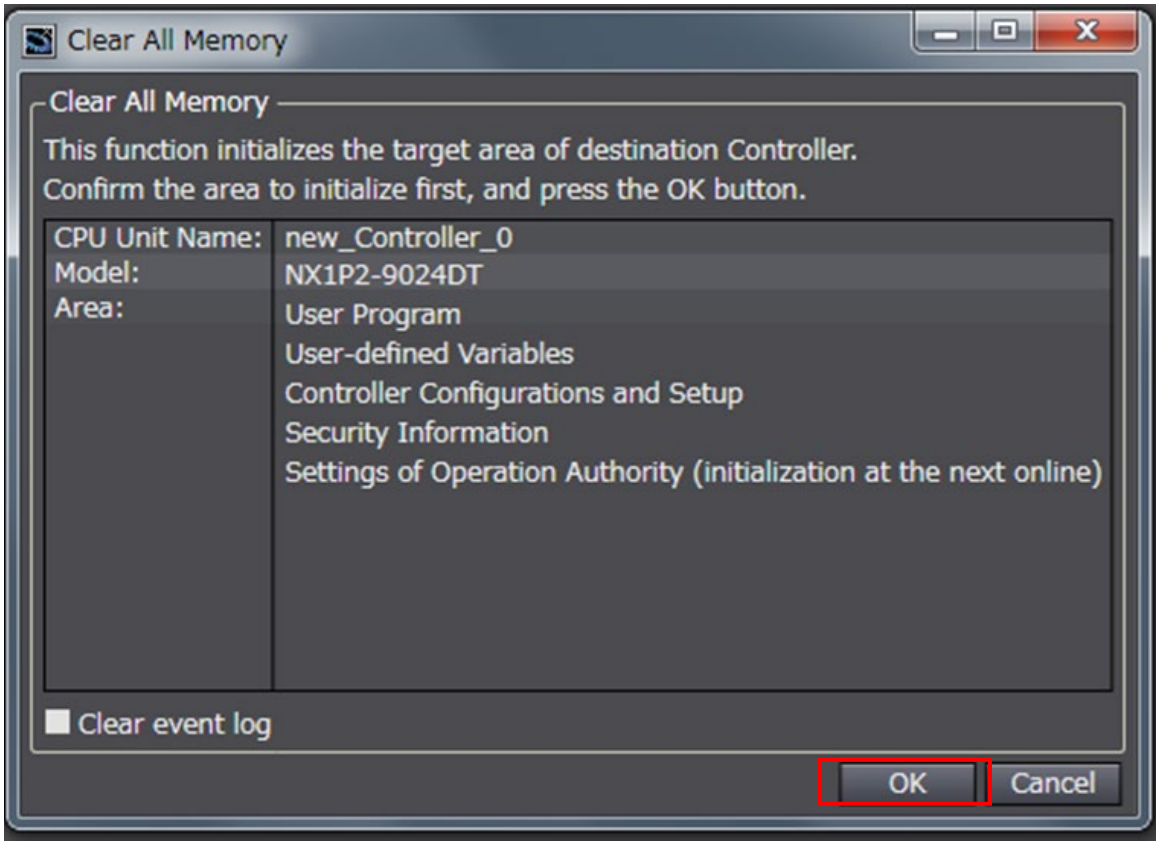
* For details on the command, refer to "9.2.2. Command Settings"

8. Initializing the System

The explanations of procedures given in this document are based on the use of the factory default settings on all the devices.
When using a device that has been changed from the default settings, various settings may not be able to proceed according to the procedures described.

8.1. Controller

To return the controller to its original settings, from the Sysmac Studio menu bar select [Controller]－[Clear All Memory].



8.2. Smart camera

For information on how to initialize a smart camera, consult our branch or sales office.

:

9. Project file

The details of the project file used in this document are shown.

9.1. Overview

This chapter describes the specifications and functions of the project file used for connecting a smart camera (F430-F series) (hereinafter referred to as "Partner device") and a controller (Built-in EtherNet/IP port) (hereinafter referred to as "Built-in EtherNet/IP port").

A "Project file" is a "Sysmac Studio" project file.
This project file includes all the following data.

- Built-in EtherNet / IP port communication settings and program task settings
- Programs and Function Blocks used for socket communication
- Data type definitions and "variable table" for variables used in ST language program

This project file uses the socket service function of Built-in EtherNet/IP port to execute "T (Measurement trigger) commands" on the smart camera and determine whether it ended normally or with an error.
The normal termination of this project file is the normal termination of TCP socket communication.
Also, its termination with error is considered as TCP socket communication termination with error.

The TCP socket options, "keep-alive" function and "linger" function are not used in this project file. Please consider the need to use them when designing your own application.

Note

This project file only confirms that communication is possible for this test configuration, product versions, and product lots used for evaluation.
Operation is not guaranteed under disturbances such as electrical noise or variations in the performance of the device itself.

Note

In Sysmac Studio, if it is necessary to distinguish between decimal data and hexadecimal data, at the beginning of decimal data, variable type name + '#' is used and variable type name + '#' + '16' is used at the beginning of hexadecimal data to distinguish the two data formats. For example, "INT#1000" decimal → "INT#16#03E8" hexadecimal. For DINT, the variable name ＋'#' is not required)

9.1.1. Communication Data Flow

This is the flow from issuing a command from the Built-in EtherNet/IP port via TCP socket communication and receiving the response data from the Smart camera. In this project file, a series of processes from TCP Open to Close is executed continuously. When response data is divided and received as multiple data packets, the Receive process is repeated.

1.	TCP Open Processing	A TCP Open request is issued from the Built-in EtherNet/IP port to the Smart camera and a TCP connection is established.
	▼	
2.	Send Command Processing	The Send message set in the program with the ST language is issued from Built-in EtherNet/IP port to the partner device (Smart camera).
	▼	
3.	Response Receive Processing	The response data from the Smart camera, received through the Built-in EtherNet/IP port, is stored in the internal memory of the specified CPU Unit.
	▼	
4.	Close processing	A TCP Close request is issued from the Built-in EtherNet/IP port to the Smart camera and the TCP connection is terminated.

* Depending on the partner device or the command to be used, there are cases of not sending response data after receiving command and sending response data immediately after establishing connection. Therefore, in this project file, it is possible to make settings in the "General Ethernet Communications Send/Receive Sequence" Function Block for the necessity of Send / Receive processing".
If you set "Send only", "Response Receive processing" will not be executed. Likewise, if you set "Receive only", "Command send processing" will not be executed.

9.1.2. TCP Socket Communications via Socket Service Commands

This section gives an overview of the general behavior of TCP socket communication and Send / Receive messages by the TCP Socket Service Function Block (hereinafter referred to as socket service instruction).

Note

For more detailed information, refer to Chapter 2 “Instruction Descriptions" in "Machine Automation Controller NJ/NX-series Instructions Reference Manual" (W502).

● TCP Socket Service using Socket Service Commands

In this project file, socket communication is established using the following five types of commands that are included standard.

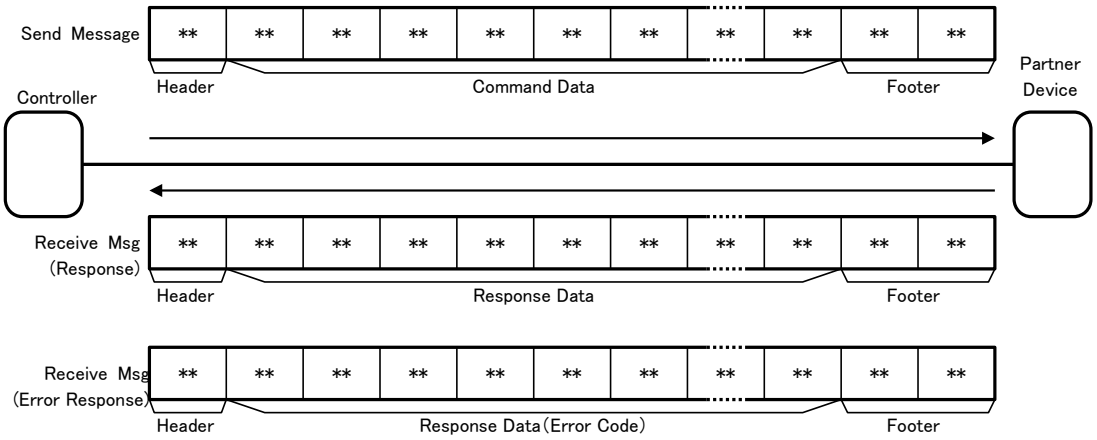
Name	Function Block	Description
TCP socket connect	SktTCPConnect	Connect to the Smart camera's TCP port with Active Open.
TCP socket send	SktTCPSend	Sends data from the specified TCP socket.
TCP socket receive	SktTCPRcv	Reads data received from the specified TCP socket.
TCP/UDP socket close	SktClose	Closes the specified TCP socket.
Get TCP socket status Reads status of socket	SktGetTCPStatus	Reads status received from the specified TCP socket. This project file is used to confirm Receive complete at end of Receive processing, and Close status at Close processing.

* The Socket acquired by the TCP socket connect command (SktTCPConnect: SktTCPConnect_instance) is used as an input parameter of another socket service command. The specifications of structure "sSOCKET" of data type of "Socket" are as follows.

Variable	Name	Content	Data type	Valid Range	Factory default
Socket	Socket	Socket	_sSOCKET	-	-
Handle	Handle	Handle for Sent and Received Data	UDINT	Depends on data type.	-
SrcAdr	Source address	Self-address ※ 1	_sSOCKET_ADDRESS	-	-
PortNo	Port number	Port number	UINT	0 to 65535	
IpAdr	IP Address	IP address or host name *2	STRING	Depends on data type.	
DstAdr	Partnering address	Partnering 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.	

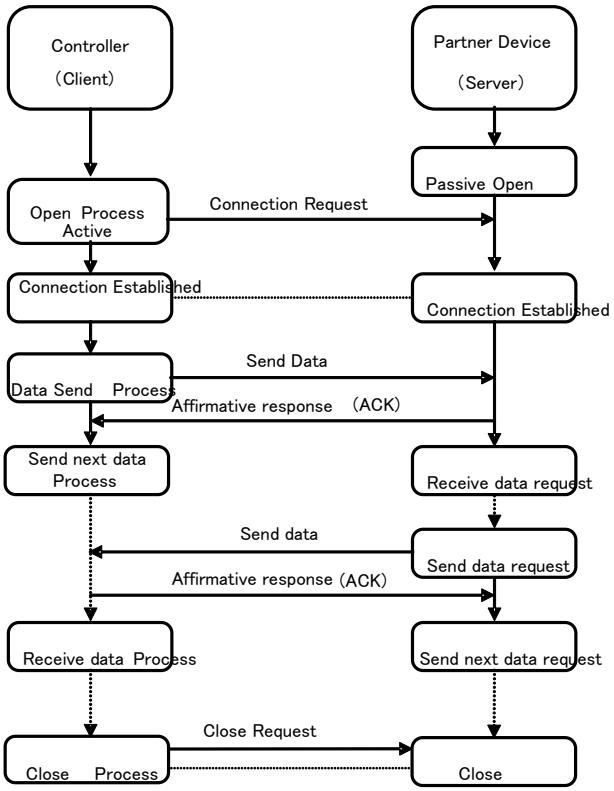
*1: An address refers to an IP address and a port number.
*2: When using a hostname, DNS and Hosts settings must also be used.

● Send / Receive Message



● Send / Receive Sequence

When using TCP communication between an external device (server) and controller (client), the process proceeds as follows.



9.2. Partner Device Command

An explanation of the partner device commands in this project file.

9.2.1. Command Overview

In this project file, Ethernet communication is performed with the external device using the “TRIGGER (Serial Trigger)” command.

Command	Content
TRIGGER	Execute inspection

Note

For details, refer to sections “3-2 Serial (TCP) Control and Output” and “4-1-4 Serial Commands” in the "F430-F Series User Manual for Communication Settings" (SDNC-748).

9.2.2. Command Settings

This section explains setting details for the "TRIGGER” (Inspection trigger) command.

● Settings for Send Data (Command)

Send Data is set by Function Block: SendMessageSet_instance.

<Partnering device specifications>

・Data is stored in ASCII code.

Variable	Content (Data format)	Setting value
Send_Header	Send Header (STRING[5])	‘’(None)
Send_Addr	Send Address (STRING[5])	‘’(None)
Send_Command	Send Data (STRING[256])	“TRIGGER”
Send_Check	Additional Send Check (STRING[5])	‘’(None)
Send_Terminate	Send Terminate (STRING[5])	‘’(None)

Variable	Content (Data format)	Data	Description
Send_Data	Send Message (STRING[256])	CONCAT(Send_Header, Send_Addr, Send_Command, Send_Check, Send_Terminate)	Used as send data of SktTCPSend command (SktTCPSend_instance).

●Contents of received data (response)

Received data is stored as output received data after data check by the function block: ReceiveCheck_instance.

<Partnering device specifications>

・Data is stored in ASCII code.

Variable	Content (Data format)	Storage area description
Recv_Data	Receiving Communication Data (STRING[256])	Receive buffer
Recv_Buff	Receiving Communication Data (STRING[256])	Receive data storage area (The data of the receive buffer is stored as it is.)

● Sent/Received Message

Send Message

54	52	49	47	47	45	52
'T'	'R'	'I'	'G'	'G'	'E'	'R'

(Normal processing: measurement data 369 171 -1,LOT123456 DATE05/2012,123456<CR><LF>)

Receive Message

33	36	39	20	31	37	31	20	2D	31	2C	4C
'3'	'6'	'9'	' '	'1'	'7'	'1'	' '	'-'	'1'	','	'L'
4F	54	31	32	33	34	35	36	20	44	41	54
'O'	'T'	'1'	'2'	'3'	'4'	'5'	'6'	' '	'D'	'A'	'T'
45	30	35	2F	32	30	31	32	2C	31	32	33
'E'	'0'	'5'	'/'	'2'	'0'	'1'	'2'	','	'1'	'2'	'3'
34	35	36	0D	0A							
'4'	'5'	'6'	CR	LF							

(At Error Processing)

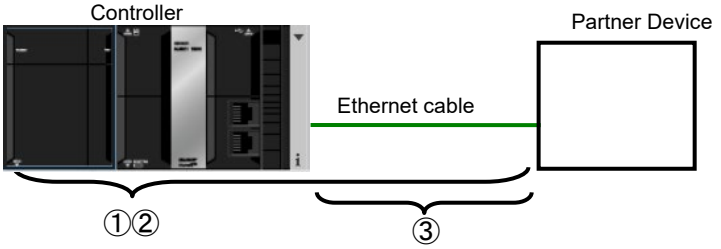
Receive Message (None)

9.3. Error Judgment Processing

This section describes the error judgment processing in this project file.

9.3.1. Project File Error Judgment Processing

In this project file, error judgment processing is performed for the following three items ① to ③. For more information on Error codes, refer to section 9.7.1. "Error Code List".



① TCP socket communication error due to socket service command

Errors during execution of TCP socket communication detected on the program, such as abnormalities in communication hardware, command format, and parameters, are judged as a "Communication Error". Judgment is performed by the "ErrorID" argument of the socket service command.

② Timeout error for communication with partnering device (Smart camera)

If the Open, Send, Receive, or Close processes do not terminate normally and each process is not completed within the monitoring time, it is judged as a "Timeout error". Judgment is performed by time monitoring in the project file. For details on the time monitoring function by the timer in the project file, refer to section 9.3.2. "Time Monitoring".

③ TCP connection status error at the end of processing

In this project file, regardless of whether the Open process to Receive process terminates normally or with an error, the operation procedure is to finish all processing that follows the last Close processing. Therefore, the TCP connection status variable "TcpStatus" of the "SktGetTCPStatus" command judges whether the Close processing ended normally. If and error occurs in the Close processing, the next Open processing may not be performed correctly. For details on how to handle TCP connection status errors, refer to section 9.7.2. "TCP connection status and actions".

9.3.2. Time monitoring function

This section describes the Time monitoring function in this project file.

The setting of the monitoring time can be changed by the variable used in the "ParameterSet" Function Block.

●Time monitoring function by timer in project file

In this project file, it is possible to interrupt the process (Timeout) by the timer in the project file, if the process is not completed due to some error condition. The timeout value is 5 seconds (initial value) for each process from Open to Close.

[Time monitoring function by timer in project file]

Process	Monitoring content	Variable name	Timeout (Default)
Open Processing	Time from start to end of Open processing	TopenTime	After 5 seconds (UINT#500)
Send processing	Time from start to end of Send process	TfsTime	After 5 seconds (UINT#500)
Receive Processing	Time from start to end of Receive process * If Receive processing is repeated, it will be monitored for each receive process.	TfrTime	After 5 seconds (UINT#500)
Close Processing	Time from start to end of Close process * After confirming that the TCP connection status after Close processing is normal, it is determined that the processing is completed.	TcloseTime	After 5 seconds (UINT#500)

● Time monitoring function by Built-in EtherNet/IP port (socket service)

Built-in EtherNet/IP port has a function to monitor the time of incoming data divided and arrived as socket service. Store "TrTime = UINT # 3 (300 ms)" (initial value) in the parameter "TimeOut" of the socket service instruction [SktTCPRcv_instance] during Receive processing. Also, in this project file, the variable "TrTime" is also set in the receive wait time monitoring timer for the receive waiting time of the next response after the completion of one receive. If the next response does not arrive from the other device within this time, it is determined that the receive process has ended.

Note

Refer to "SktTCPRcv" instruction in Chapter 2 "Instruction Descriptions" in "Machine Automation Controller NJ/NX-series Instructions Reference Manual" (W502) for the time monitoring function by socket service.

● Re-send/time monitoring function by Built-in EtherNet IP port (TCP/IP)

When a communication error occurs, TCP/IP automatically monitors data re-send and processing time if there is no error in Built-in EtherNet/IP port. In this project file, if it terminates with error during processing, the TCP/IP Resend / Time monitoring function is stopped by Close processing. However, if Close processing indicates "TCP connection status error", the TCP/IP Resend / Time monitoring function in Built-in EtherNet/IP port may continue to operate. Refer to "9.7.2. TCP connection status error status and corrective action" for the status and corrective action.

9.4. Variables Used

Variables used in this project file.

9.4.1. List of Variables Used

A list of the variables required to run this project file.

● Input Variables

Variables used for operation of this project file.

Variable name	Data format	Description
Input_Start	BOOL	Starts this project file by OFF(FALSE)→ON(TRUE) operation. Turns from ON→OFF after output confirmation (either normal output or output with error).

● Output Variable

Variable that reflects the execution result of this project file.

Variable name	Data format	Description
Output_RecvMess	STRING[256]	Stores received data (response) (Reserve area for 256 bytes)
Output_ErrCode	WORD	Stores error results (flags) for communication errors and timeout errors detected during open processing, send processing, receive processing, and close processing. At normal termination, "#0000" is stored.
Output_SktCmdsErrorID	WORD	The error code of each socket service instruction for communication error and timeout error detected in open process, send process, and receive process is stored. At normal termination, "#0000" is stored.
Output_SkTcloseErrorID	WORD	The error code of SktTcpClose instruction of communication error and timeout error detected at close process is stored separately from the open process, send process and receive process errors. At normal termination, "#0000" is stored.
Output_EtnTcpSta	_eCONNECTION_STATE	Stores the TCP connection status when communication error or timeout error is detected during close processing. At normal termination, "_CLOSED" is stored.
Output_MErrCode	DWORD	An error code is stored when an FCS calculation error or a device error is detected as a result of reception processing. At normal termination, "#00000000" is stored.

● Internal Variable

Variables only used for calculations in this project file.

Variable name	Data format	Description
Local_Status	sStatus (STRUCT)	Run status of the program
Busy	BOOL	It becomes TRUE while this project file is being executed and becomes FALSE when it is not being executed.
Done	BOOL	It becomes TRUE when this project file ends normally and becomes FALSE when [Input_Start] turns from TRUE → FALSE.
Error	BOOL	It becomes TRUE when this project file ends with an error and becomes FALSE when [Input_Start] turns from TRUE → FALSE.
Local_State	DINT	State processing number
Local_ErrCode	uErrorFlgs (UNION)	Sets the error code.
Local_ErrCode. WordData	WORD	Error code is expressed in WORD.
Local_ErrCode. BoolData	ARRAY[0..15] OF BOOL	・Communication Error BoolData[0]: Send processing : Error (TRUE) / Normal (FALSE) BoolData[1]: Receive processing : Error (TRUE) / Normal (FALSE) BoolData[2]: Open processing : Error (TRUE) / Normal (FALSE) BoolData[3]: Close processing : Error (TRUE) / Normal (FALSE) BoolData[4]: Processing Number : Error (TRUE) / Normal (FALSE) ・Timeout Error BoolData[8]: Send processing : Error (TRUE) / Normal (FALSE) BoolData[9]: Receive processing : Error (TRUE) / Normal (FALSE) BoolData[10]: Open processing : Error (TRUE) / Normal (FALSE) BoolData[11]: Close processing : Error (TRUE) / Normal (FALSE) ・ Other BoolData[5]: Send / Receive required judgment error: Error (TRUE) / Normal (FALSE) BoolData[12]: Partnering device error: Error (TRUE) / Normal (FALSE) BoolData[6..7],[13..14]: Reserve BoolData[15]: Error occurs
Local_ExecFlgs	sControl (STRUCT)	Socket service command activation flag

Variable name	Data format	Description
Send	BOOL	Execute Send processing command (TRUE) / Do not execute (FALSE)
Recv	BOOL	Execute Receive processing command (TRUE) / Do not execute (FALSE)
Open	BOOL	Execute Open processing command (TRUE) / Do not execute (FALSE)
Close	BOOL	Execute Close processing command (TRUE) / Do not execute (FALSE)
Status	BOOL	Execute TCP status command (TRUE) / Do not execute (FALSE)
Local_SrcDataByte	UINT	Set number of bytes for Send data.
Local_SrcData	ARRAY[0..2000] OF BYTE	Area for storing Send data for SktTCPSend command (SktTCPSend_instance). (Reserve area for 256 bytes)
Local_RecvData	ARRAY[0..2000] OF BOOL	Area for storing Receive data (Response) for SktTCPRcv command (SktTCPRcv_instance). (Reserve area for 256 bytes)
Local_ReceiveMessage	STRING[256]	Stores Local_RecvData Receive STRING data (Response). (Reserve area for 256 characters)
Local_RecvCheckFlg	BOOL	External device error judgment command activation flag Execute (TRUE) / Don't execute (FALSE)
Local_InitialSettingOK	BOOL	Initial processing normal setting flag
Local_TONFlgs	sTimerControl (STRUCT)	Timer execute flag
Tfs	BOOL	Execute Send processing time monitoring timer command (TRUE) / Do not execute (FALSE)
Tfr	BOOL	Execute Receive processing time monitoring timer command (TRUE) / Do not execute (FALSE)
Topen	BOOL	Execute Open processing time monitoring timer command (TRUE) / Do not execute (FALSE)
Tclose	BOOL	Execute Close processing time monitoring timer command (TRUE) / Do not execute (FALSE)
Tr	BOOL	Execute Wait time till receive next response time monitoring timer command (TRUE) / Do not execute (FALSE)
Local_ComType	sControl (STRUCT)	Set Send / Receive processing required or not required.
Send	BOOL	Send processing required (TRUE) / Not required (FALSE). * In the case of Send Processing Required / Receive Processing Not Required: At the time of receive processing, when there is no wait for arrival of receive data, receive processing is skipped and transition to Close processing is performed. Specify when no response data is sent for command transmission.

Variable name		Data format	Description
	Recv	BOOL	Receive processing required (TRUE) / Not required (FALSE). * In the case of Send Processing Required / Receive Processing Required: It waits for the arrival of receive data after transmission processing. After confirming the arrival of the received data, transit to the receive processing. Specify when response data is sent for command transmission.
	Error	BOOL	Send / Receive processing required / not required setting error flag (A flag is set when a setting error occurs.)

● Variables used to initialize socket service command

Variable name	Data format	Description
NULL_SOCKET	_sSOCKET	Command initialization data for internal socket service (Hold / constant: valid) Default(Handle := 0, SrcAdr := (PortNo := 0, IpAdr := ''), DstAdr := (PortNo := 0, IpAdr := '')) (Used for All Socket commands)
NULL_ARRAYOFBYTE_1	ARRAY[0..0] OF BYTE	Command initialization data for internal socket service Send (Hold / constant: valid) Default is [0] (used with SktTCPSend command)
NULL_ARRAYOFBYTE_2	ARRAY[0..0] OF BYTE	Command initialization data array for internal_receive socket service (hold / constant: invalid) Default is [0] (used with SktTCPRcv command)

9.4.2. List of Variables for Function Blocks / Functions used

This is a list of user-defined function blocks in the program when this project file is executed.

Refer to "9.5.3 Detailed Explanation of Function Block" for the variables of the following function block.

Variable name	Data format	Description
ETN_ParameterSet_instance	ParameterSet	Ethernet setting (destination IP address etc.) Monitoring time of each process from Open to Close
ETN_SendMessageSet_instance	SendMessageSet	Set whether to require send / receive processing and send message.
ETN_ReceiveCheck_instance	ReceiveCheck	Determines storage of received data and whether receive data it is normal / error.

● Timer

Timer used by this project file.

Variable name	Data format	Description
Topen_TON_instance	TON	Measures time for Open processing.
Tfs_TON_instance	TON	Measures time for Send processing.
Tfr_TON_instance	TON	Measures time for Receive processing.
Tclose_TON_instance	TON	Measures time for Close processing.
Tr_TON_instance	TON	Measures Wait time till Receive next response.

9.4.3. List of System Variables

A list of the variables required to run this project file.

● System Variables (External Variables)

Variable name	Data format	Description
_EIP_EtnOnlineSta	BOOL	Built-in EtherNet/IP port communication status function TRUE: Can be used. FALSE: Cannot be used.

Note

For details on system variables and communication commands, refer to Chapter 2 “Instruction Descriptions" in "Machine Automation Controller NJ/NX-series Instructions Reference Manual" (W502).

9.5. Program (ST Language)

9.5.1. Functional Configuration of Program in ST Language

This project file is programmed in ST language. The function configuration is as follows.

Major classification	Minor classification	Content
1. Communications processing	1.1.Communication processing start 1.2.Communication processing status flag column clear 1.3.Communication processing execution status	Start communication process.
2. Initialization processing	2.1. Processing time monitor timer initialization 2.2. Socket service command initialization 2.3. Socket service command activation flag initialization 2.4. Processing time monitor timer execute flag initialization 2.5. Error code storage area initialization 2.6. Various process monitoring time setting and Ethernet related parameter setting 2.7.Setting for Send / Receive requirement and Send data setting 2.8. Convert send data from STRING format to byte array 2.9. Receive data storage area initialization 2.10. End initialization process	Perform Ethernet parameter settings and initialization of error code storage area. Set whether to require Send and Receive and other settings for Send data and Receive data.
3. Open processing	3.1. Set judgement of Open processing status and activation flag 3.2. Run Open processing time monitoring timer 3.3. Open command activation (TCP. Active Open processing)	Perform TCP Open (Active) processing. Processing starts unconditionally after startup and initialization of communication processing.
4. Send processing	4.1. Set judgement of Send processing status and activation flag 4.2. Run Send processing time monitoring timer 4.3. Send command activation	Processing is started when Send processing requirement is "required" and Open processing ends normally.
5. Receive processing	5.1. Set judgement of Receive processing status and activation flag 5.2. Run Receive wait time monitor timer 5.3. Run Receive processing time monitor timer 5.4. Receive command activation 5.5. Acquire TCP status processing activation 5.6. Partnering device error judgement command activation	Processing is started when Receive processing requirement is "required" and Send processing ends normally. If multiple incoming data arrives, repeat the receiving process. Store received data and check received data.
6. Close processing	6.1. Set judgement of Close processing status and activation flag 6.2. Run Close processing time monitor timer 6.3. Close command activation 6.4. Acquire TCP status processing activation	Close processing is performed. This processing starts when the following conditions are met. ・When the Receive processing required is set to "not-required" and the Send process ends normally ・When the Receive process ends normally ・Immediately after any one of the Open

		processing, Send processing, or Receive processing ends with an error
7. Error processing	7. Processing number error processing	Execute error processing when a nonexistent processing number is detected.

9.5.2. Detailed Explanation of the Main Program

The following are included in the project file.
Communication settings, transmission data (command) settings, and reception data (response data) confirmation that need to be changed depending on the connected device are performed in the function block (ETN_ParameterSet_instance, ETN_SendMessageSet_instance, ETN_ReceiveCheck_instance). If you want to change these values, refer to "9.5.3 Detailed Explanation of Function Block".

[Main Program: Program0]

1. Communication processing

```
(* ================================================================================ *)
(* 名称：NJｼﾘｰｽﾞ汎用Ethernet通信ﾌﾟﾛｸﾞﾗﾑ                                          *)
(* 機能：汎用Ethernet通信ﾒｲﾝﾌﾟﾛｸﾞﾗﾑ                                              *)
(* Ethernetﾕﾆｯﾄ：NX1P2-9024DT（内蔵EtherNet/IPﾎﾟｰﾄ）                                *)
(* 備考：                                                                         *)
(*                                                                                *)
(* ﾊﾞｰｼﾞｮﾝ情報：2011/08/01 V1.00 新規                                              *)
(*                                                                                *)
(* (C)Copyright OMRON Corporation 2011 All Rights Reserved.                       *)
(* ================================================================================ *)

(* 1. 通信処理 *)
(* 変数説明：通信処理 制御用 ================================================== )
(                                                                                )
( 入力起動ﾌﾗｸﾞ        ：Input_Start                                             )
(                                                                                )
( 通信処理ｽﾃｰﾀｽﾌﾗｸﾞ列  ：Local_Status<STRUCT>                                    )
(  |                      ↓                                                     )
(  ├通信処理実行中ﾌﾗｸﾞ  (Busy)  ：Local_Status.Busy                             )
(  ├通信処理正常終了ﾌﾗｸﾞ (Done)  ：Local_Status.Done                             )
(  └通信処理異常終了ﾌﾗｸﾞ (Error) ：Local_Status.Error                            )
(                                                                                )
( 状態処理番号        ：Local_State                                             )
(                    10：初期処理                                                )
(                    11：ｵｰﾌﾟﾝ処理                                               )
(                    12：送信処理                                                )
(                    13：受信処理                                                )
(                    14：ｸﾛｰｽﾞ処理                                               )
(                    99：処理番号異常処理                                        )
(                                                                                )
( ================================================================================ *)

(* 1.1. 通信処理開始
   通信処理ｽﾃｰﾀｽﾌﾗｸﾞ列がｸﾘｱな状態で入力起動ﾌﾗｸﾞがONされた場合に通信処理を開始*)
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:初期処理へ
END_IF;

(* 1.2. 通信処理ｽﾃｰﾀｽﾌﾗｸﾞ列ｸﾘｱ
   通信処理非実行状態で入力起動ﾌﾗｸﾞOFFにより通信処理ｽﾃｰﾀｽﾌﾗｸﾞ列ｸﾘｱ *)
IF NOT(Local_Status.Busy) AND NOT(Input_Start) THEN
   Local_Status.Done:=FALSE;
   Local_Status.Error:=FALSE;
END_IF;

(* 1.3. 通信処理実行中状態
   状態処理番号(Local_State)に応じた処理を実行 *)
IF Local_Status.Busy THEN
   CASE Local_State OF
```

2. Initialization processing

```
10: (* ================================================================== *)
    (* 2. 初期処理                                                        *)
    (*   ・通信全体の各種初期化とﾊﾟﾗﾒｰﾀ設定                               *)
    (*   ・送信ﾃﾞｰﾀの設定と受信ﾃﾞｰﾀ格納ｴﾘｱの初期化                       *)
    (* ================================================================== *)

    (* 2.1. 処理時間監視ﾀｲﾏの初期化 *)
    Topen_TON_instance (In:=FALSE,PT:=TIME#0ms);
    Tfs_TON_instance   (In:=FALSE,PT:=TIME#0ms);
    Tr_TON_instance    (In:=FALSE,PT:=TIME#0ms);
    Tfr_TON_instance   (In:=FALSE,PT:=TIME#0ms);
    Tclose_TON_instance(In:=FALSE,PT:=TIME#0ms);

    (* 2.2. ｿｹｯﾄｻｰﾋﾞｽ用命令の初期化 *)
    SktTCPConnect_instance(
        Execute:=FALSE,SrcTcpPort:=UINT#0,DstTcpPort:=UINT#0,DstAdr:='');
    SktTCPSend_instance(
        Execute:=FALSE,Socket:=NULL_SOCKET,Size:=UINT#0,
        SendDat:=NULL_ARRAYOFBYTE_1[0]);
    SktTCPRcv_instance(
        Execute:=FALSE,Socket:=NULL_SOCKET,Size:=UINT#0,TimeOut:=UINT#0,
        RcvDat:=NULL_ARRAYOFBYTE_2[0]);
    SkTclose_instance(
        Execute:=FALSE,Socket:=NULL_SOCKET);
    SktGetTCPStatus_instance(
        Execute:=FALSE,Socket:=NULL_SOCKET);

    (* 2.3. ｿｹｯﾄｻｰﾋﾞｽ用命令起動ﾌﾗｸﾞの初期化 *)
    (* 変数説明：ｿｹｯﾄｻｰﾋﾞｽ用命令起動ﾌﾗｸﾞ(Executeﾊﾟﾗﾒｰﾀ用) ================ )
    (                                                                    )
    (  ｿｹｯﾄｻｰﾋﾞｽ用命令起動ﾌﾗｸﾞ列：Local_ExecFlgs<STRUCT>                  )
    (     |                                  ↓                         )
    (     ├送信命令起動ﾌﾗｸﾞ   (SktTCPSend)      :Local_ExecFlgs.Send       )
    (     ├受信命令起動ﾌﾗｸﾞ   (SktTCPRcv)       :Local_ExecFlgs.Recv       )
    (     ├ｵｰﾌﾟﾝ命令起動ﾌﾗｸﾞ  (SktTCPConnect)   :Local_ExecFlgs.Open       )
    (     ├ｸﾛｰｽﾞ命令起動ﾌﾗｸﾞ  (SkTclose)        :Local_ExecFlgs.Close      )
    (     └TCPｽﾃｰﾀｽ取得命令起動ﾌﾗｸﾞ                                      )
    (                        (SktGetTCPStatus):Local_ExecFlgs.Status     )
    ( ================================================================== *)
    Local_ExecFlgs.Send:=FALSE;
    Local_ExecFlgs.Recv:=FALSE;
    Local_ExecFlgs.Open:=FALSE;
    Local_ExecFlgs.Close:=FALSE;
    Local_ExecFlgs.Status:=FALSE;

    (* 2.4. 処理時間監視ﾀｲﾏ実行ﾌﾗｸﾞの初期化 *)
    (* 変数説明：処理時間監視ﾀｲﾏ実行ﾌﾗｸﾞ(Inﾊﾟﾗﾒｰﾀ用) ==================== )
    (                                                                    )
    (  処理時間監視ﾀｲﾏ実行ﾌﾗｸﾞ列：Local_TONFlgs<STRUCT>                  )
    (     |                                  ↓                         )
    (     ├送信処理時間監視ﾀｲﾏ実行ﾌﾗｸﾞ  (Tfs_TON) :Local_TONFlgs.Tfs      )
    (     ├受信処理時間監視ﾀｲﾏ実行ﾌﾗｸﾞ  (Tfr_TON) :Local_TONFlgs.Tfr      )
    (     ├ｵｰﾌﾟﾝ処理時間監視ﾀｲﾏ実行ﾌﾗｸﾞ(Topen_TON)                      )
    (     |                                  :Local_TONFlgs.Topen       )
    (     ├ｸﾛｰｽﾞ処理時間監視ﾀｲﾏ実行ﾌﾗｸﾞ(Tclose_TON)                     )
    (     |                                  :Local_TONFlgs.Tclose      )
    (     └受信待ち時間監視ﾀｲﾏ実行ﾌﾗｸﾞ  (Tr_TON)                         )
    (         (次ﾒｯｾｰｼﾞ待ち時間)              :Local_TONFlgs.Tr          )
    (                                                                    )
    ( ================================================================== *)
    Local_TONFlgs.Tfs:=FALSE;
    Local_TONFlgs.Tfr:=FALSE;
    Local_TONFlgs.Topen:=FALSE;
    Local_TONFlgs.Tclose:=FALSE;
    Local_TONFlgs.Tr:=FALSE;

    (* 2.5. ｴﾗｰｺｰﾄﾞ格納ｴﾘｱの初期化 *)
    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. 各処理監視時間設定およびEthernet関連ﾊﾟﾗﾒｰﾀ設定 *)
ETN_ParameterSet_instance(
    Execute:=TRUE);

(* 2.7. 送受信処理要否の設定および送信ﾃﾞｰﾀの設定 *)
ETN_SendMessageSet_instance(
    Execute:=TRUE);
    (* 送受信処理要否の設定異常判定 *)
(* ＜変数メモ＞
      ＞ Local_ComType.Send : 送信処理要否ﾌﾗｸﾞ
      ＞ Local_ComType.Recv : 受信処理要否ﾌﾗｸﾞ
      ＞ Local_ComType.Error : 送受信処理要否設定ｴﾗｰ *)
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. 送信ﾃﾞｰﾀをString形式からﾊﾞｲﾄ配列に変換 *)
Local_SrcDataByte:=
    StringToAry(ETN_SendMessageSet_instance.Send_Data,Local_SrcData[0]);
(* 2.9. 受信ﾃﾞｰﾀ格納ｴﾘｱの初期化 *)
ClearString(Local_ReceiveMessage);
ClearString(Output_RecvMess);
Local_RecvCHNo:=0;
Local_RecvDataLength:=0;
Local_ReceiveSize:=UINT#256;

(* 2.10. 初期設定終了処理 *)
IF Local_InitialSettingOK THEN
    Local_State:=11;                                    //11:ｵｰﾌﾟﾝ処理へ
ELSE
    Local_Status.Busy:=FALSE;
    Local_Status.Error:=TRUE;

    Local_State:=0;                                     //0:通信非実行状態へ
END_IF;
```

3. Open processing

```
11: (* ================================================================== *)
    (* 3. ｵｰﾌﾟﾝ処理                                                       *)
    (*  ・相手TCPﾎﾟｰﾄにActive接続                                         *)
    (* ================================================================== *)
    (* ＜変数メモ＞
       ＞ Local_ExecFlgs.Open : ｵｰﾌﾟﾝ命令起動ﾌﾗｸﾞ
       ＞ Local_TONFlgs.Topen  : ｵｰﾌﾟﾝ処理時間監視ﾀｲﾏ実行ﾌﾗｸﾞ *)

    (* 3.1. ｵｰﾌﾟﾝ処理状況の判定と起動ﾌﾗｸﾞｾｯﾄ *)

        (* 3.1.1. ﾀｲﾑｱｳﾄ処理 *)
    IF Topen_TON_instance.Q THEN
        Local_ErrCode.BoolData[10]:=TRUE;
        Output_SktCmdsErrorID:=WORD#16#FFFF;
        Local_ExecFlgs.Open:=FALSE;
        Local_TONFlgs.Topen:=FALSE;
        Local_State:=14;                                    //14:ｸﾛｰｽﾞ処理へ

        (* 3.1.2. 正常終了処理 *)
    ELSIF SktTCPConnect_instance.Done THEN
        Local_ErrCode.BoolData[2]:= FALSE;
        Output_SktCmdsErrorID:=WORD#16#0000;
        Local_ExecFlgs.Open:=FALSE;
        Local_TONFlgs.Topen:=FALSE;
    (* ＜変数メモ＞
         ＞ Local_ComType.Send : 送信処理要否ﾌﾗｸﾞ
         ＞ Local_ComType.Recv : 受信処理要否ﾌﾗｸﾞ *)
        IF Local_ComType.Send THEN
            Local_State:=12;                                //12:送信処理へ
        ELSIF Local_ComType.Recv THEN
            Local_State:=13;                                //13:受信処理へ
        END_IF;
        (* 3.1.3. 異常終了処理 *)
    ELSIF SktTCPConnect_instance.Error THEN
        Local_ErrCode.BoolData[2]:=TRUE;
        Output_SktCmdsErrorID:=SktTCPConnect_instance.ErrorID;
        Local_ExecFlgs.Open:=FALSE;
        Local_TONFlgs.Topen:=FALSE;
        Local_State:=14;                                    //14:ｸﾛｰｽﾞ処理へ

        (* 3.1.4. ｵｰﾌﾟﾝ命令起動ﾌﾗｸﾞｾｯﾄ／ﾀｲﾏ実行ﾌﾗｸﾞｾｯﾄ *)
    ELSE
        Local_ExecFlgs.Open:=TRUE;
        Local_TONFlgs.Topen:=TRUE;
    END_IF;

    (* 3.2. ｵｰﾌﾟﾝ処理時間監視ﾀｲﾏ実行 *)
    Topen_TON_instance(
        IN:=Local_TONFlgs.Topen,
        PT:=MULTIME(TIME#10ms,ETN_ParameterSet_instance.TopenTime));

    (* 3.3. ｵｰﾌﾟﾝ命令起動(TCP.Activeｵｰﾌﾟﾝ処理)
        内蔵ETN利用可能(_EIP_EtnOnlineStaがON)状態でｵｰﾌﾟﾝ命令起動 *)
    SktTCPConnect_instance(
        Execute:=Local_ExecFlgs.Open AND _EIP_EtnOnlineSta,
        SrcTcpPort:=ETN_ParameterSet_instance.SrcPort,
        DstTcpPort:=ETN_ParameterSet_instance.DstPort,
        DstAdr:=ETN_ParameterSet_instance.DstIPAddr);
```

4. Send processing

```
12: (* ================================================================== *)
    (* 4. 送信処理                                                        *)
    (*  ・指定したTCPﾎﾟｰﾄからﾃﾞｰﾀを送信                                   *)
    (* ================================================================== *)
    (* <変数メモ>
       > Local_ExecFlgs.Send : 送信命令起動ﾌﾗｸﾞ
       > Local_TONFlgs.Tfs  : 送信処理時間監視ﾀｲﾏ実行ﾌﾗｸﾞ *)

    (* 4.1. 送信処理状況の判定と起動ﾌﾗｸﾞｾｯﾄ *)

        (* 4.1.1. ﾀｲﾑｱｳﾄ処理 *)
    IF Tfs_TON_instance.Q THEN
        Local_ErrCode.BoolData[8]:=TRUE;
        Output_SktCmdsErrorID:=WORD#16#FFFF;
        Local_ExecFlgs.Send:=FALSE;
        Local_TONFlgs.Tfs:=FALSE;
        Local_State:=14;                                    //14:ｸﾛｰｽﾞ処理へ

        (* 4.1.2. 正常終了処理 *)
    ELSIF SktTCPSend_instance.Done THEN
        Local_ErrCode.BoolData[0]:=FALSE;
        Output_SktCmdsErrorID:=WORD#16#0000;
        Local_ExecFlgs.Send:=FALSE;
        Local_TONFlgs.Tfs:=FALSE;
        (* <変数メモ>
           > Local_ComType.Recv : 受信処理要否ﾌﾗｸﾞ *)
        Local_State:=SEL(Local_ComType.Recv,14,13); //13:受信処理へ
                                                    //14:ｸﾛｰｽﾞ処理へ

        (* 4.1.3. 異常終了処理 *)
    ELSIF SktTCPSend_instance.Error THEN
        Local_ErrCode.BoolData[0]:=TRUE;
        Output_SktCmdsErrorID:=
            SktTCPSend_instance.ErrorID;
        Local_ExecFlgs.Send:=FALSE;
        Local_TONFlgs.Tfs:=FALSE;
        Local_State:=14;                                    //14:ｸﾛｰｽﾞ処理へ

        (* 4.1.4. 送信命令起動ﾌﾗｸﾞｾｯﾄ／ﾀｲﾏ実行ﾌﾗｸﾞｾｯﾄ *)
    ELSE
        Local_ExecFlgs.Send:=TRUE;
        Local_TONFlgs.Tfs:=TRUE;
    END_IF;

    (* 4.2. 送信処理時間監視ﾀｲﾏ実行 *)
    Tfs_TON_instance(
        In:=Local_TONFlgs.Tfs,
        PT:=MULTIME(TIME#10ms, ETN_ParameterSet_instance.TfsTime));

    (* 4.3. 送信命令起動
        内蔵ETN利用可能(_EIP_EtnOnlineStaがON)状態で送信命令起動 *)
    SktTCPSend_instance(
        Execute:=Local_ExecFlgs.Send AND _EIP_EtnOnlineSta,
        Size:=Local_SrcDataByte,
        Socket:=SktTCPConnect_instance.Socket,
        SendDat:=Local_SrcData[0]);
```

5. Receive processing

```
13: (* ================================================================== *)
    (* 5. 受信処理                                                        *)
    (*  ・指定したTCPｿｹｯﾄの受信ﾊﾞｯﾌｧ内のﾃﾞｰﾀを読み出し                    *)
    (* ================================================================== *)
    (* <変数メモ>
       > Local_ExecFlgs.Recv : 受信命令起動ﾌﾗｸﾞ
       > Local_ExecFlgs.Status : TCPｽﾃｰﾀｽ取得命令起動ﾌﾗｸﾞ
       > Local_TONFlgs.Tfr  : 受信処理時間監視ﾀｲﾏ実行ﾌﾗｸﾞ
       > Local_TONFlgs.Tr  : 受信待ち時間監視ﾀｲﾏ実行ﾌﾗｸﾞ
                                    (次ﾒｯｾｰｼﾞ待ち時間)                  *)

    (* 5.1. 受信処理状況の判定と起動ﾌﾗｸﾞｾｯﾄ *)

        (* 5.1.1. 受信終了処理 *)
    IF Tr_TON_instance.Q THEN
        Local_ExecFlgs.Status:=FALSE;
        Local_TONflgs.Tfr:=FALSE;
        Local_TONflgs.Tr:=FALSE;

        (* 受信ﾃﾞｰﾀのBYTE配列->STRING型変換 *)
        Local_ReceiveMessage:=
                AryToString(Local_RecvData[0],Local_RecvDataLength);

        (* 相手機器異常判定命令起動ﾌﾗｸﾞｾｯﾄ *)
        Local_RecvCheckFlg:=TRUE;

        Local_State:=14;                                //14:ｸﾛｰｽﾞ処理へ

        (* 5.1.2. ﾀｲﾑｱｳﾄ処理 *)
    ELSIF Tfr_TON_instance.Q THEN
        Local_ErrCode.BoolData[9]:=TRUE;
        Output_SktCmdsErrorID:=WORD#16#FFFF;
        Local_ExecFlgs.Recv:=FALSE;
        Local_ExecFlgs.Status:=FALSE;
        Local_TONflgs.Tfr:=FALSE;
        Local_State:=14;                                //14:ｸﾛｰｽﾞ処理へ

        (* 5.1.3. 正常終了処理 *)
    ELSIF SktTCPRcv_instance.Done THEN
        Local_RecvDataLength
            :=Local_RecvDataLength+SktTCPRcv_instance.RcvSize;
        Local_RecvCHNo:=Local_RecvDataLength;

        Local_ExecFlgs.Recv:=FALSE;
        Local_TONflgs.Tfr:=FALSE;
        Local_TONflgs.Tr:=TRUE;             // 5.1.5. 受信ﾃﾞｰﾀ読み出し処理へ

        (* 5.1.4. 異常終了処理 *)
    ELSIF SktTCPRcv_instance.Error THEN;
            Local_ErrCode.BoolData[1]:=TRUE;
            Output_SktCmdsErrorID:=
                SktTCPRcv_instance.ErrorID;

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

        Local_State:=14;                                //14:ｸﾛｰｽﾞ処理へ
```

```
        (* 5.1.5.  受信ﾃﾞｰﾀ読み出し処理 *)
    ELSIF SktGetTCPStatus_instance.Done
            OR SktGetTCPStatus_instance.Error THEN
        Local_ExecFlgs.Status:=FALSE;

            (* 読み出すﾃﾞｰﾀがある場合：受信処理を継続 *)
        IF SktGetTCPStatus_instance.DatRcvFlag THEN
            Local_ExecFlgs.Recv:=TRUE;
            Local_TONflgs.Tfr:=TRUE;
            Local_TONflgs.Tr:=FALSE;
        END_IF;
            (* 読み出すﾃﾞｰﾀがない場合：
                ・ﾃﾞｰﾀを全く受信していない場合は何も処理せず、
                  次のｻｲｸﾙでTCPｽﾃｰﾀｽ取得を再起動する
                ・既にﾃﾞｰﾀを受信している場合はﾚｽﾎﾟﾝｽ受信待ち時間を監視して
                  次のﾚｽﾎﾟﾝｽがなくﾀｲﾑｱｳﾄすれば、既に受信済みのﾃﾞｰﾀを
                  読み出して受信処理終了
                *)

        (* 5.1.6.  TCPｽﾃｰﾀｽ取得命令起動ﾌﾗｸﾞｾｯﾄ／ﾀｲﾏ実行ﾌﾗｸﾞｾｯﾄ *)
    ELSE
        Local_ExecFlgs.Status:=TRUE;
        Local_TONflgs.Tfr:=TRUE;

            (* 相手機器異常判定命令起動ﾌﾗｸﾞ初期化 *)
        Local_RecvCheckFlg:=FALSE;
    END_IF;

    (* 5.2.  受信待ち時間監視ﾀｲﾏ実行(次ﾚｽﾎﾟﾝｽ待ち時間) *)
    Tr_TON_instance(
        In:=Local_TONflgs.Tr,

        PT:=MULTIME(TIME#100ms,ETN_ParameterSet_instance.TrTime));

    (* 5.3.  受信処理時間監視ﾀｲﾏ実行 *)
    Tfr_TON_instance(
        In:=Local_TONflgs.Tfr,
        PT:=MULTIME(TIME#10ms,ETN_ParameterSet_instance.TfrTime));

    (* 5.4.  受信命令起動
        内蔵ETN利用可能(_EIP_EtnOnlineStaがON)状態で受信命令起動 *)
    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.  TCPｽﾃｰﾀｽ取得命令起動
        内蔵ETN利用可能(_EIP_EtnOnlineStaがON)状態でTCPｽﾃｰﾀｽ取得命令起動 *)
    SktGetTCPStatus_instance(
        Execute:=Local_ExecFlgs.Status AND _EIP_EtnOnlineSta,
        Socket:=SktTCPConnect_instance.Socket);

    (* 5.6.  相手機器異常判定命令起動 *)
    ETN_ReceiveCheck_instance(
        Execute:=Local_RecvCheckFlg,
        Recv_Buff:=Local_ReceiveMessage,
        Recv_Data:=Output_RecvMess,
        tLength:=Local_RecvDataLength,
        ErrorID:=Local_ErrCode.WordData,
        ErrorIDEx:=Output_MErrCode);
```

6. Close processing

```
14: (* ================================================================== *)
    (* 6.ｸﾛｰｽﾞ処理                                                       *)
    (*   ・指定したｿｹｯﾄをｸﾛｰｽﾞ                                           *)
    (* ================================================================== *)
    (* ＜変数メモ＞
       ＞ Local_ExecFlgs.Close : ｸﾛｰｽﾞ命令起動ﾌﾗｸﾞ
       ＞ Local_ExecFlgs.Staus : TCPｽﾃｰﾀｽ取得命令起動ﾌﾗｸﾞ
       ＞ Local_TONFlgs.Tclose  : ｸﾛｰｽﾞ処理時間監視ﾀｲﾏ実行ﾌﾗｸﾞ *)

    (* 6.1. ｸﾛｰｽﾞ処理状況の判定と起動ﾌﾗｸﾞｾｯﾄ *)

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

        Local_State:=0;                                  //0:通信非実行状態へ

        (* 6.1.2. 正常終了処理 *)
    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;

                (* 通信処理全体の処理結果判定 *)
                Local_Status.Busy:=FALSE;

                    (* 通信処理正常終了 *)
                IF Local_ErrCode.WordData = WORD#16#0000 THEN
                    Local_Status.Done:=TRUE;
                    Local_ErrCode.BoolData[15]:=FALSE;

                    (* 通信処理異常終了 *)
                ELSE
                    Local_Status.Error:=TRUE;
                    Local_ErrCode.BoolData[15]:=TRUE;
                END_IF;
                Output_ErrCode:=Local_ErrCode.WordData;

                Local_State:=0;                          //0:通信非実行状態へ

            END_IF;
        END_IF;

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

        Local_State:=0;                                  //0:通信非実行状態へ
```

```
            (* 6.1.4. ｸﾛｰｽﾞ命令起動ﾌﾗｸﾞｾｯﾄ／ﾀｲﾏ実行ﾌﾗｸﾞｾｯﾄ *)
        ELSE
            Local_ExecFlgs.Close:=TRUE;
            Local_TONflgs.Tclose:=TRUE;

        END_IF;

        (* 6.2. ｸﾛｰｽﾞ処理時間監視ﾀｲﾏ実行 *)
        Tclose_TON_instance(
            In:= Local_TONflgs.Tclose,
            PT:=MULTIME(TIME#10ms,ETN_ParameterSet_instance.TcloseTime));

        (* 6.3. ｸﾛｰｽﾞ命令起動
            内蔵ETN利用可能(_EIP_EtnOnlineStaがON)状態でｸﾛｰｽﾞ命令起動 *)
        SkTclose_instance(
            Execute:=Local_ExecFlgs.Close AND _EIP_EtnOnlineSta,
            Socket:=SktTCPConnect_instance.Socket);

        (* 6.4. TCPｽﾃｰﾀｽ取得命令起動
            内蔵ETN利用可能(_EIP_EtnOnlineStaがON)状態でTCPｽﾃｰﾀｽ取得命令起動 *)
        SktGetTCPStatus_instance(
            Execute:=Local_ExecFlgs.Status AND _EIP_EtnOnlineSta,
            Socket:=SktTCPConnect_instance.Socket);
```

7. Processing number error processing

```
    99: (* ========================================================== *)
        (* 7. 処理番号異常処理                                        *)
        (*  ・存在しない状態処理番号が設定された場合の異常処理       *)
        (* ========================================================== *)

        Output_ErrCode:=WORD#16#0010;
        Local_Status.Busy:=FALSE;
        Local_Status.Error:=TRUE;
        Local_State:=0;                                  //0:通信非実行状態へ

    ELSE
        Local_State:=99;                                 //99:処理番号異常処理へ

    END_CASE;

END_IF;
```

9.5.3. Detailed Explanation of Function Blocks

The function block of this project file is shown below.

Data in the Function Block that will change depending on the partnering device is highlighted in the red frame of the function block shown below.

● ETN_ParameterSet_instance: Content of Function Block (ParameterSet)

Command	Name	FB/FUN	Graphic display	ST Expression
ParameterSet	General Ethernet communication Parameter Settings	FB	No	ETN_ParameterSet_instance (Execute, TfsTime, TrTime, TfrTime, , TopenTime, TcloseTime, SrcPort, DstIPAddr, DstPort);

・Input/Output variable table (arguments)

・Input

Variable name	Data format	Name	Content	Valid Range	Unit	Factory default
Execute	BOOL	Execute	When OFF (FALSE) turns to ON (TRUE), Function Block is activated. (Normally: TRUE)	Depends on data type.	-	-

・Output

Variable name	Data format	Name	Content	Valid Range	Unit	Factory default
TopenTime	UINT	Monitoring time for Open	The monitoring time for Open is set in 10ms increments.	Depends on data type.	-	-
TfsTime	UINT	Monitoring time for Send	The monitoring time for Send processing is set in 10ms increments.	Depends on data type.	-	-
TrTime	UINT	Monitoring time for Wait before Receive	Wait time for Receive data to arrive is set in 100ms increments.	Depends on data type.	-	-
TfrTime	UINT	Receive processing time	The monitoring time for Receive processing is set in 10ms increments.	Depends on data type.	-	-
TcloseTime	UINT	Monitoring time for Close	The monitoring time for Close processing is set in 10ms increments.	Depends on data type.	-	-
SrcPort	UINT	Local PortNo	Set Local port.	Depends on data type.	-	-
DstIPAddr	STRING [256]	Partner IP address	Set the IP address of partnering device.	According to setting on partnering device	-	-
DstPort	UINT	Partner PortNo	Set the PortNo of partnering device.	According to setting on partnering device	-	-
Busy	BOOL	Executing	Not used (Not used in this project file)	-	-	-
Done	BOOL	Normal End				
Error	BOOL	Error end				
ErrorID	WORD	Error				

		Information				
ErrorIDEx	DWORD	Error Information				

・Internal Variable Table: None

・Program

```
(* ================================================================================ *)
(* 名称：NJｼﾘｰｽﾞ 汎用Ethernet通信ﾊﾟﾗﾒｰﾀ設定ﾌｧﾝｸｼｮﾝﾌﾞﾛｯｸ                          *)
(* 機能：各処理監視時間設定およびEthernet関連ﾊﾟﾗﾒｰﾀ設定                          *)
(*                                                                                *)
(* 対象機器：                                                                     *)
(*   ﾒｰｶ名     ：ｵﾑﾛﾝ株式会社                                                    *)
(*   機器名称  ：スマートカメラ                                                   *)
(*   ｼﾘｰｽﾞ/形式：F430-F ｼﾘｰｽﾞ                                                    *)
(*   備考      ：                                                                 *)
(*                                                                                *)
(* ﾊﾞｰｼﾞｮﾝ情報：2018/11/30 V1.00 新規                                             *)
(*                                                                                *)
(* (C)Copyright OMRON Corporation 2018 All Rights Reserved.                       *)
(* ================================================================================ *)

(* 変数説明：引数 戻り値 ============================================================ )
(                                                                                  )
( 引数：      名称      ﾃﾞｰﾀ型    内容                                           )
(     ・入力：Execute    BOOL      起動ﾌﾗｸﾞ                                       )
(                                                                                  )
(     ・出力：TopenTime  UINT      ｵｰﾌﾟﾝ処理監視時間                               )
(           TfsTime    UINT      送信処理監視時間                                  )
(           TrTime     UINT      受信待ち処理監視時間                              )
(           TfrTime    UINT      受信処理監視時間                                  )
(           TcloseTime UINT      ｸﾛｰｽﾞ処理監視時間                                )
(           SrcPort    UINT      自PortNo                                          )
(           DstIPAddr  UINT      相手機器IPｱﾄﾞﾚｽ                                  )
(           DstPort    UINT      相手機器PortNo                                    )
(           Busy       BOOL      未使用                                            )
(           Done       BOOL      未使用                                            )
(           Error      BOOL      未使用                                            )
(           ErrorID    WORD      未使用                                            )
(           ErrorIDEx  DWORD     未使用                                            )
(                                                                                  )
(     ・入出力：なし                                                               )
(                                                                                  )
( 戻り値：なし                                                                     )
(                                                                                  )
( ================================================================================ *)

IF Execute THEN

    (* Ethernet関連ﾊﾟﾗﾒｰﾀ設定 *)
    SrcPort:= UINT#0;            // 自ﾎﾟｰﾄNo
    DstIPAddr:= '192.168.188.2';   // 相手IPｱﾄﾞﾚｽ
    DstPort:= UINT#49211;        // 相手ﾎﾟｰﾄNo

    (* 処理監視時間設定：処理開始～終了までの最大時間 *)
    TopenTime := UINT#500; // ｵｰﾌﾟﾝ処理監視時間設定：設定単位10ms＜500⇒5s＞
    TfsTime:= UINT#500;    // 送信処理監視時間設定：設定単位10ms＜500⇒5s＞
    TfrTime:= UINT#500;    // 受信処理監視時間：設定単位10ms＜500⇒5s＞
    TcloseTime:=UINT#500;  // ｸﾛｰｽﾞ処理監視時間：設定単位10ms＜500⇒5s＞

    (* ﾚｽﾎﾟﾝｽを複数ﾊﾟｹｯﾄで分割受信する場合のﾊﾟｹｯﾄ間隔の最大待ち時間（受信命令）
       および次のﾚｽﾎﾟﾝｽの最大待ち時間（受信待ち時間監視ﾀｲﾏ） *)
    TrTime:= UINT#3;       // 受信待ち監視時間：設定単位100ms＜3⇒300ms＞

END_IF;

RETURN;
```

●ETN_SendMessageSet_instance: Content of Function Block (SendMessageSet)

Command	Name	FB/FUN	Graphic display	ST Expression
SendMessageSet	General Ethernet communication Set Send/Receive Sequence	FB	No	ETN_SendMessageSet_instance (Execute, Send_Data, ComType);

・Input/Output variable table (arguments)

・Input

Variable name	Data format	Name	Content	Valid Range	Unit	Factory default
Execute	BOOL	Execute	When OFF (FALSE) turns to ON (TRUE), Function Block is activated. (Normally: TRUE)	Depends on data type.	-	-

・Output

Variable name	Data format	Name	Content	Valid Range	Unit	Factory default
Send_Data	STRING [256]	Send Data	Set the command to send to Partner Device.	According to Data type	-	-
ComType	BYTE	Send/Receive type	Set whether to Send / Receive. 1: Send only, 2: Receive only, 3: Send/Receive	1 to 3	-	-
Busy	BOOL	Executing	Not used (Not used in this project file)	-	-	-
Done	BOOL	Normal End				
Error	BOOL	Error end				
ErrorID	WORD	Error Information				
ErrorIDEx	DWORD	Error Information				

・Internal Variable Table

Variable name	Data format	Name	Content	Valid Range	Unit	Factory default
Send_Header	STRING[5]	Send Header	Header of Send message	Depends on data type.	-	-
Send_Addr	STRING[5]	Partnering device address	Partnering device address	Depends on data type.	-	-
Send_Command	STRING[256]	Send Data	Command to send to Partner Device	Depends on data type.	-	-
Send_Check	STRING[5]	Send Check Code	Checks Send Message Code	Depends on data type.	-	-
Send_Terminate	STRING[5]	Send Terminator	Terminator for Send message	Depends on data type.	-	-

・Program

```
(* ================================================================================ *)
(* 名称：NJｼﾘｰｽﾞ 汎用Ethernet通信送受信ｼｰｹﾝｽ設定ﾌｧﾝｸｼｮﾝﾌﾞﾛｯｸ                    *)
(* 機能：送信／受信処理の要否および送信ﾃﾞｰﾀ設定                                *)
(*                                                                              *)
(* 対象機器：                                                                   *)
(*   ﾒｰｶ名     ：ｵﾑﾛﾝ株式会社                                                   *)
(*   機器名称  ：スマートカメラ                                                 *)
(*   ｼﾘｰｽﾞ/形式：F430-F ｼﾘｰｽﾞ                                                  *)
(*   備考      ：                                                               *)
(*                                                                              *)
(* ﾊﾞｰｼﾞｮﾝ情報：2018/11/30 V1.00 新規                                           *)
(*                                                                              *)
(* (C)Copyright OMRON Corporation 2018 All Rights Reserved.                     *)
(* ================================================================================ *)

(* 変数説明：引数 戻り値 ============================================================ )
(                                                                               )
( 引数：      名称      ﾃﾞｰﾀ型    内容                                        )
(     ・入力：Execute    BOOL      起動ﾌﾗｸﾞ                                    )
(                                                                               )
(     ・出力：SendData   STRING[256] 送信ﾃﾞｰﾀ                                  )
(           ComType    BYTE      送信／受信処理の要否設定                      )
(           Busy       BOOL      未使用                                        )
(           Done       BOOL      未使用                                        )
(           Error      BOOL      未使用                                        )
(           ErrorID    WORD      未使用                                        )
(           ErrorIDEx  DWORD     未使用                                        )
(                                                                               )
(     ・入出力：なし                                                            )
(                                                                               )
( 戻り値：なし                                                                  )
(                                                                               )
( ================================================================================ *)

IF Execute THEN

    (* 送信／受信処理の要否設定 *)
    ComType:= BYTE#16#03;  // 1:送信のみ、2:受信のみ、3:送信／受信の両方

    (* 送信ﾃﾞｰﾀ設定 *)
    Send_Header:= '';            // ﾍｯﾀﾞ
    Send_Addr:= '';              // ｱﾄﾞﾚｽ
    Send_Command:= 'TRIGGER';    // 相手機器ｺﾏﾝﾄﾞ：計測実行
    Send_Check:='';              // SUM計算
    Send_Terminate:= '';         // ﾀｰﾐﾈｰﾀ

    (* 送信ﾃﾞｰﾀの連結 *)
    Send_Data:=
        CONCAT(Send_Header,Send_Addr,Send_Command,Send_Check,Send_Terminate);

END_IF;

RETURN;
```

●ETN_ReceiveCheck_instance: Content of Function Block (ReceiveCheck)

Command	Name	FB/FUN	Graphic display	ST Expression
ReceiveCheck	General Ethernet communication Receive Processing	FB	No	ETN_ReceiveCheck_instance (Execute, Recv_Data, Recv_Buff, Error, ErrorID, ErrorIDEx);

・Input/Output variable table (arguments)

・Input

Variable name	Data format	Name	Content	Valid Range	Unit	Factory default
Execute	BOOL	Execute	When OFF (FALSE) turns to ON (TRUE), Function Block is activated.	Depends on data type.	-	-
tLength	UINT	Receive data length	No. of bytes for Receive buffer data	Depends on data type.	-	-

・Input/Output

Variable name	Data format	Name	Content	Valid Range	Unit	Factory default
Recv_Data	STRING[256]	Receive data	Receive Data Storage Result	Depends on data type.	-	-
Recv_Buff	STRING[256]	Receive buffer	Receive buffer data	Depends on data type.	-	-
ErrorID	WORD	Error Information	Error Code: Partner device error = #16#1000 FCS error = #16#2000	-	-	-
ErrorIDEx	DWORD	Error Information	Error Code: FCS Receive result / Partner device error code	-	-	-

・Output

Variable name	Data format	Name	Content	Valid Range	Unit	Factory default
Busy	BOOL	Executing	Not used (Not used in this project file)	-	-	-
Done	BOOL	Normal End				
Error	BOOL	Error end	Error end	-	-	-

・Internal Variable Table

Variable name	Data format	Name	Content	Valid Range	Unit	Factory default
Receive_Check	STRING[5]	Receive FCS	FCS Receive result of Receive data	Depends on data type.	-	-
Calc_Check	STRING[5]	Calculated Receive FCS	FCS calculation result of receive data	Depends on data type.	-	-

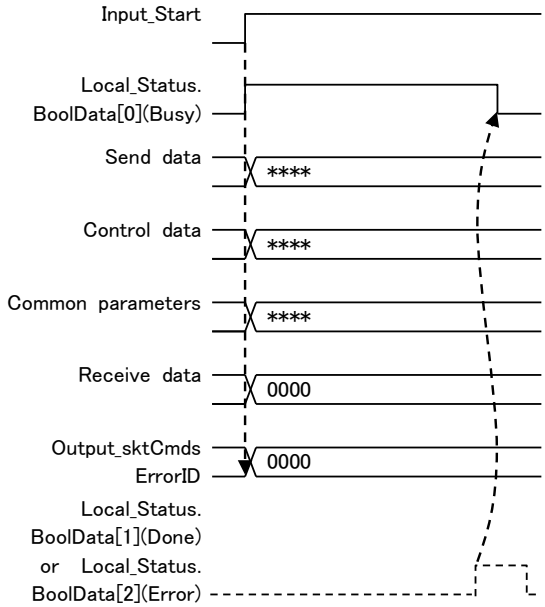
・Program

```
(* ================================================================================ *)
(* 名称：NJｼﾘｰｽﾞ 汎用Ethernet通信受信処理ﾌｧﾝｸｼｮﾝﾌﾞﾛｯｸ                              *)
(* 機能：受信ﾃﾞｰﾀ格納および受信処理結果判定                                        *)
(*                                                                                *)
(* 対象機器：                                                                     *)
(*   ﾒｰｶ名     ：ｵﾑﾛﾝ株式会社                                                     *)
(*   機器名称  ：スマートカメラ                                                   *)
(*   ｼﾘｰｽﾞ/形式：F430-Fｼﾘｰｽﾞ                                                    *)
(*   備考      ：                                                                 *)
(*                                                                                *)
(* ﾊﾞｰｼﾞｮﾝ情報：2018/11/30 V1.00 新規                                             *)
(*                                                                                *)
(* (C)Copyright OMRON Corporation 2018 All Rights Reserved.                       *)
(* ================================================================================ *)

(* 変数説明：引数 戻り値 ============================================================ )
(                                                                                  )
( 引数：      名称      ﾃﾞｰﾀ型    内容                                           )
(     ・入力：Execute   BOOL      起動ﾌﾗｸﾞ                                       )
(           tLength     UINT      受信ﾃﾞｰﾀ長                                     )
(                                                                                  )
(     ・出力：Busy      BOOL      未使用                                         )
(           Done        BOOL      未使用                                         )
(           Error       BOOL      ｴﾗｰﾌﾗｸﾞ                                       )
(                                                                                  )
(     ・入出力：Recv_Data STRING[256] 受信ﾃﾞｰﾀ格納ｴﾘｱ                           )
(           Recv_Buff STRING[256] 受信ﾊﾞｯﾌｧ                                     )
(           ErrorID   WORD      ｴﾗｰｺｰﾄﾞ                                        )
(           ErrorIDEx DWORD     FCS受信結果あるいは相手機器ｴﾗｰｺｰﾄﾞ              )
(                                                                                  )
( 戻り値：なし                                                                     )
(                                                                                  )
( ================================================================================ *)


IF Execute THEN

    (* CheckSUMの判定 ：不要 *)

    (* 受信ﾊﾞｯﾌｧのﾃﾞｰﾀを受信ﾃﾞｰﾀ格納ｴﾘｱに格納 *)
    Recv_Data:= Recv_Buff;

    (* 相手機器異常の判定 *)
    (* F430はシリアル(TCP)通信ではエラーレスポンスは返さない *)
    Error:= FALSE;                      // ｴﾗｰﾌﾗｸﾞ ﾘｾｯﾄ
    ErrorID:= WORD#16#0000;             // ｴﾗｰｺｰﾄﾞ ｸﾘｱ
    ErrorIDEx:= DWORD#16#00000000;      // 相手機器ｴﾗｰｺｰﾄﾞ ｸﾘｱ

END_IF;

RETURN;
```

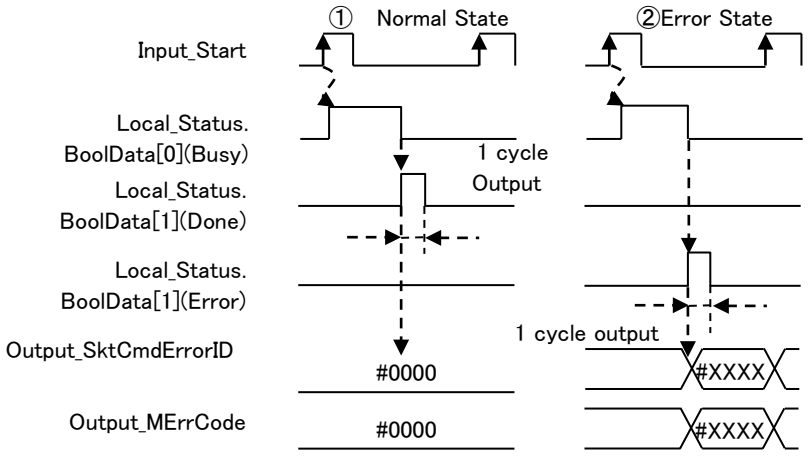
9.6. Timing Chart

Timing Chart of Program in ST Language.

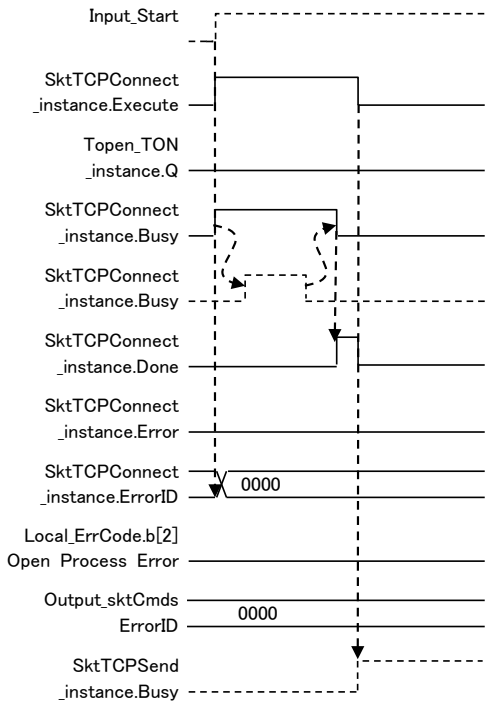
Start &Setting



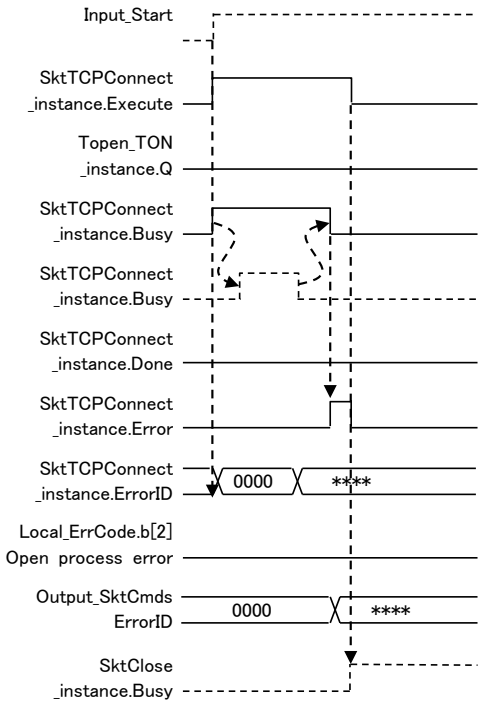
If [Input_Start] is changed from "True (ON)" to "False (OFF)" during execution of the following, one cycle of End Normal or End with Error is output after processing is completed.



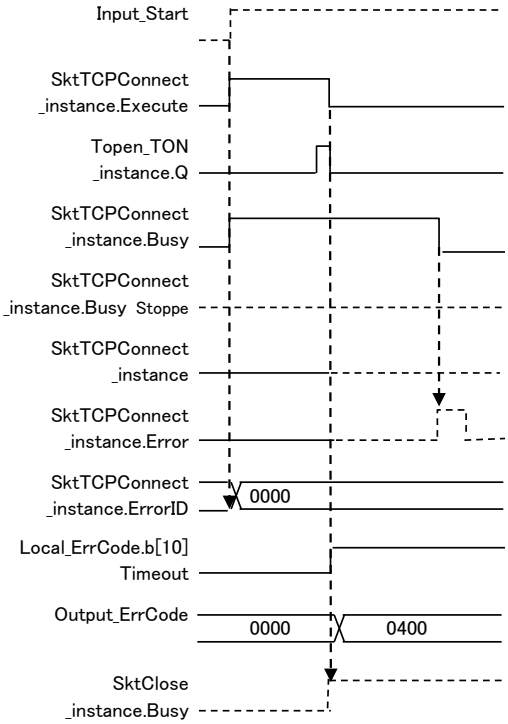
● Open Processing



(End Normal)

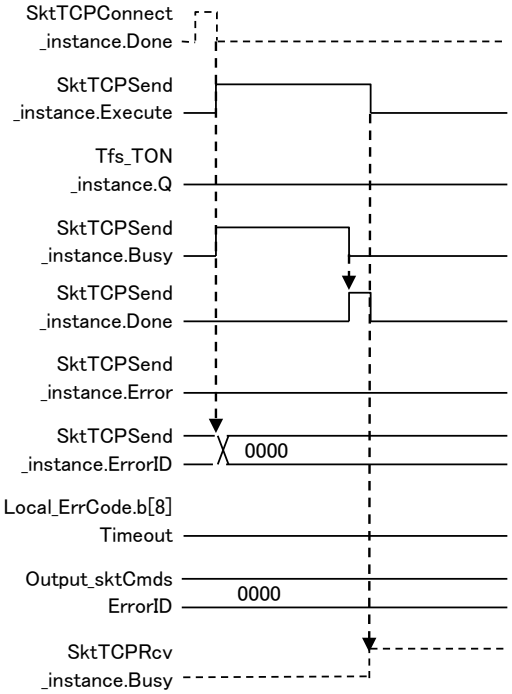


(End with Error)

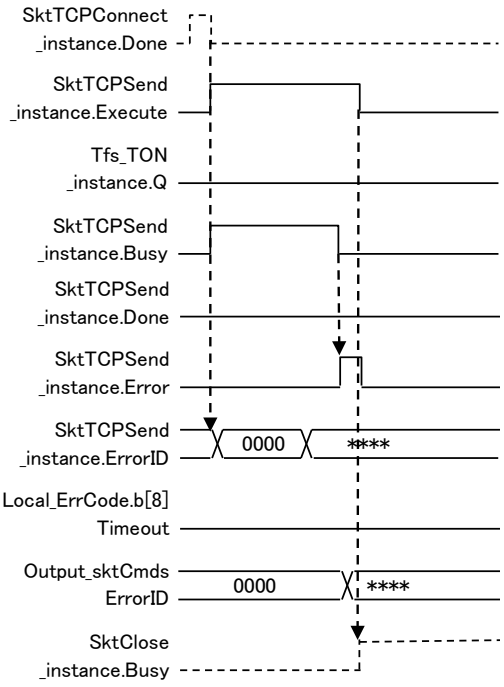


(Timeout)

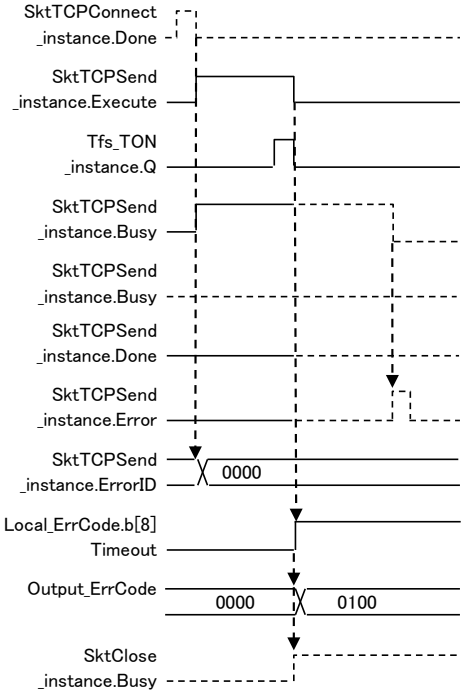
● Send Processing



(End Normal)

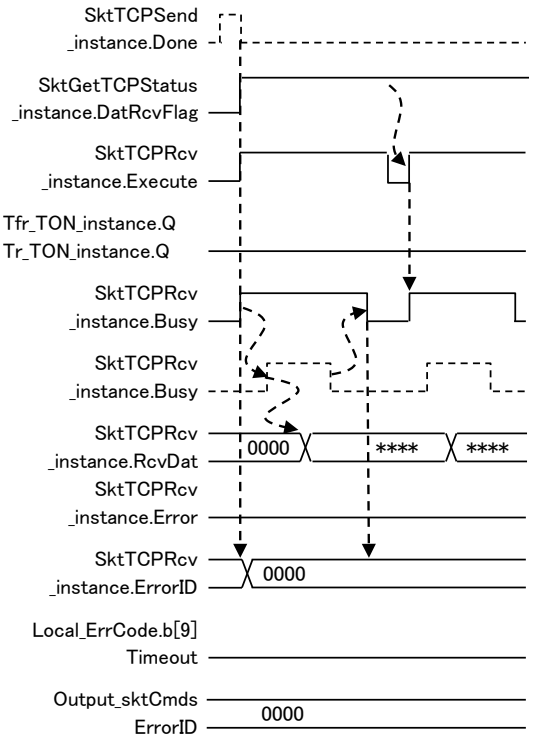


(End with Error)

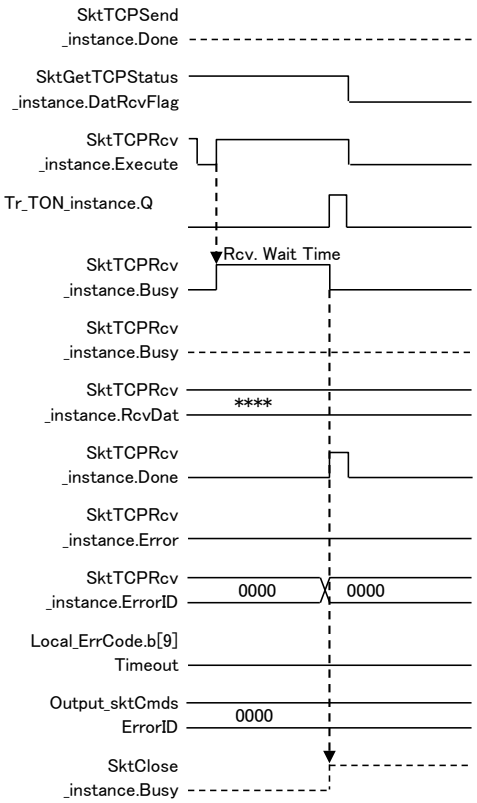


(Timeout)

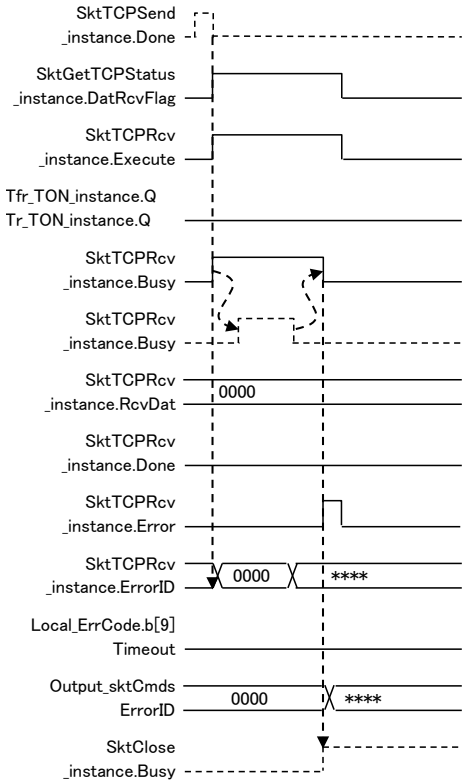
● Receive Processing



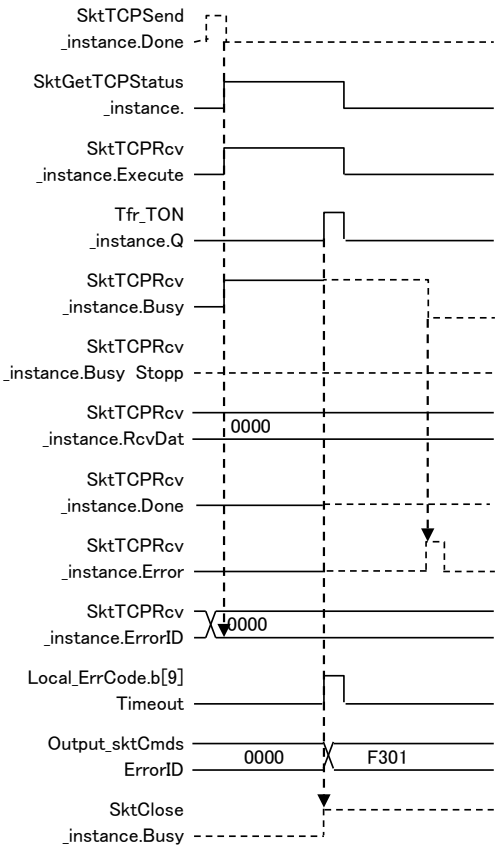
(Repeat)



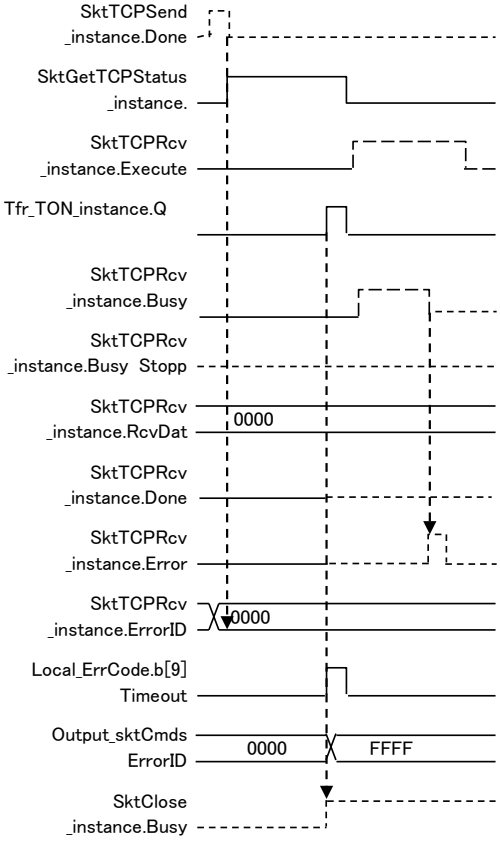
(End Normal)



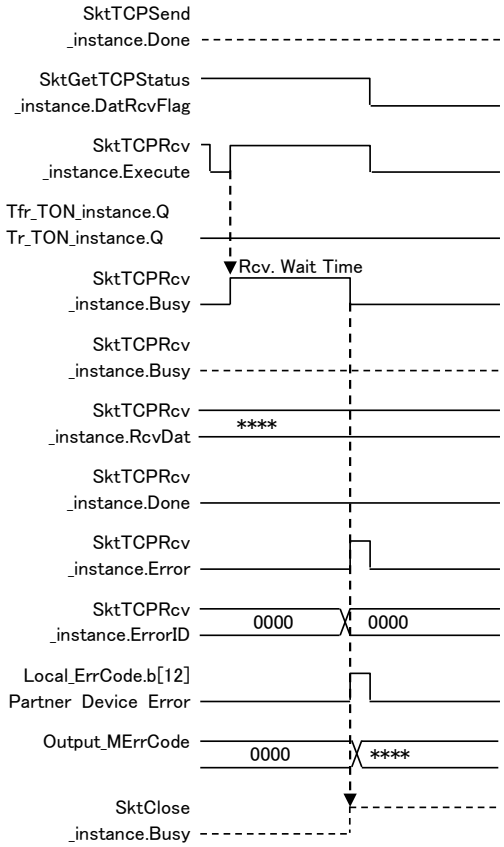
(End with Error)



(Timeout: Receive Error)

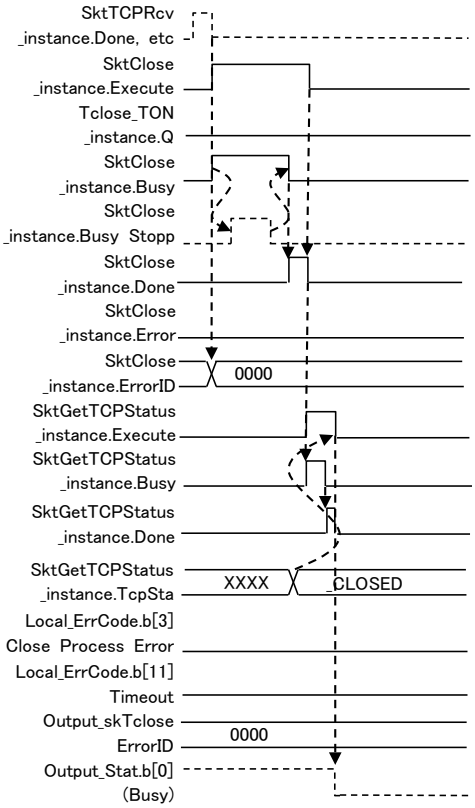


(Timeout: No Receive data)

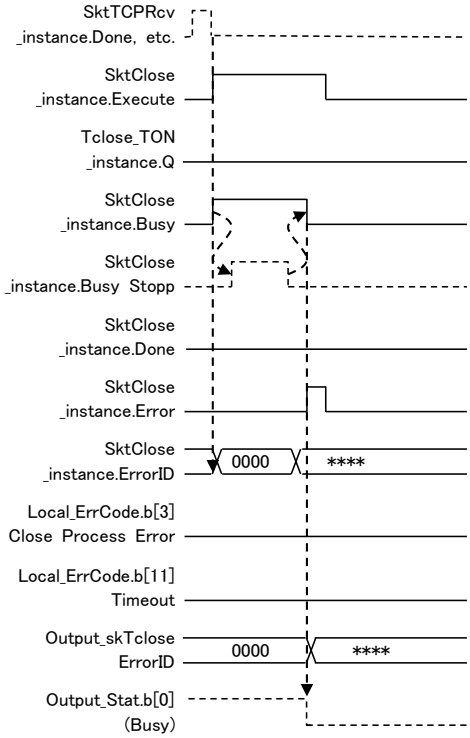


(Partner Device Error)

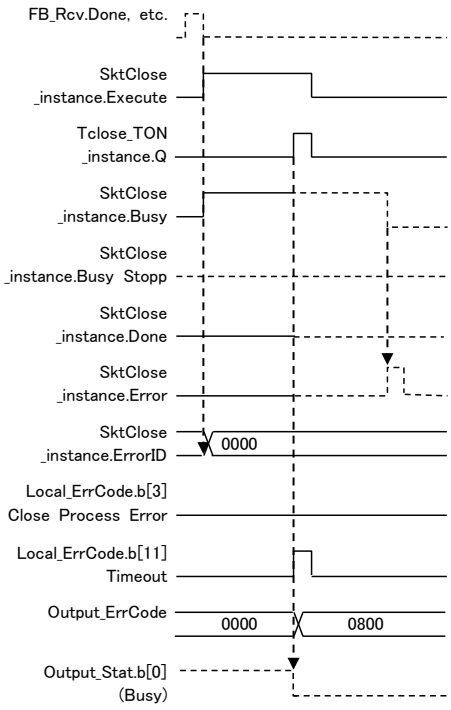
● Close processing



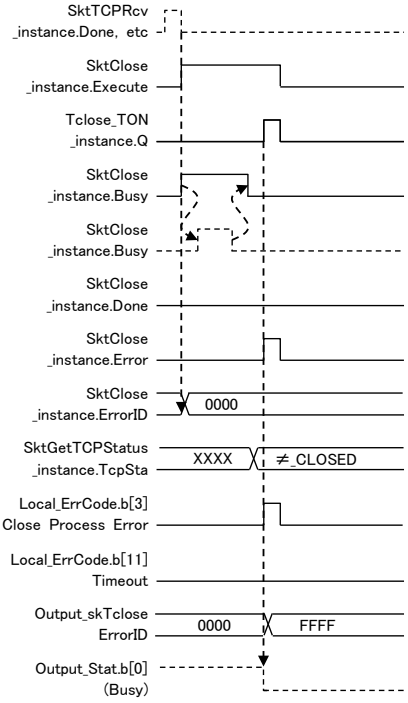
(End Normal)



(End with Error)



(Timeout)



(Status Error)

9.7. Error Processing

9.7.1. Error Code List

The list of error code that occur when this ST language is executed is shown below.

● TCP Connection Status Error [Output_EtnTcpSta]

If TCP connection status does not become normal status (_CLOSED) in time after Close processing, TCP connection status code is set to variable [Output_EtnTcpSta]. (If Close processing ends with error, check together.)

Error code enumerator [_eCONNECTION_STATE]	Content
_CLOSED	Closed state (Normal State)
_LISTEN	Waiting for a connection
_SYN SENT	Sent SYN in active state.
_SYN RECEIVED	SYN sent and received.
_ESTABLISHED	It has been established.
_CLOSE WAIT	After FIN has been sent, the state is waiting to end.
_FIN WAIT1	Finished and FIN sent.
_CLOSING	Finished and FIN exchange complete. Waiting for FIN acknowledgment (ACK).
_LAST ACK	Finished after receiving FIN. Waiting for FIN acknowledgment (ACK).
_FIN WAIT2	Received FIN acknowledgment (ACK). Waiting for FIN.
_TIME WAIT	Waiting for silence, twice the maximum segment survival time (2 MSL) after termination.

● Error codes [Output_SktCmdsErrorID], [Output_SkTcloseErrorID]

If an error occurs in any of Open processing, Send processing, or Receive processing, after setting the error code to the variable [Output_SktCmdsErrorID], Close processing is done.

If an error occurs during Close processing, the error code is set to the variable [Output_SkTcloseErrorID] and processing ends. The main error codes are as follows.

(O: Open process (SktTCPConnect command), S: Send process (SktTCPSend command), R: Receive process (SktTCPRcv command) C: Close process (SktClose command)

Error code	O	S	R	C	Content
#16#0000	○	○	○	○	Normal End
#16#0400	○	○	○	-	The input parameter of the instruction is out of the range of the 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 remote 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 specified by the input variable "SrcTcpPort" is already open. The partner node specified by the input variable "DstAdr" does not exist. The partner node specified by the input variables "DstAdr" and "DstTcpPort" is not waiting for connect ・SktTCPRcv Instruction Specified socket is currently receiving The connection of the specified socket has not been established. ・SktTCPSend Instruction Specified socket is currently sending. The connection of the specified socket has not been established.
#16#2006	-	-	○	-	A socket service command timeout occurred.
#16#2007	-	○	○	○	The handle specified in the socket service instruction is invalid.
#16#2008	○	○	○	○	The instruction executed exceeds the resources of the socket service instructions that can be executed simultaneously.
#16#FFFF	○	○	○	○	Ended without completing the execution of the instruction.

Note

For details, see the following Appendices in "Machine Automation Controller NJ/NX-series Instructions Reference Manual" (W502): "A-1 Error Codes That You Can Check with ErrorID", "A-2 Error Codes", “A-3 Instructions You Cannot Use in Event Tasks”.

Note

For details and measures for Built-in EtherNet / IP port error, refer to “Chapter 9 Socket Service” in “Machine Automation Controller NJ / NX Series CPU Unit Built-in EtherNet/IP Port User's Manual (W506). See "9-7 Precautions in Using Socket Services".

● Error Flag (Terminate with Error / Timeout) [Output_ErrCode]

If any of the Open, Send, Receive, or Close process ends with an error or times out, an error flag is set in the variable [Output_ErrCode], and an error code is stored in either the variable [Output_SktCmdsErrorID] or the variable [Output_SkTcloseErrorID].

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

(O: Open process (SktTCPConnect command), S: Send process (SktTCPSend command), R: Receive process (SktTCPRcv command) C: Close process (SktClose command)

Error Flag	O	S	R	C	Content
#16#0000	○	○	○	○	Normal End
#16#0001		○			Send processing ended with error.
#16#0002			○		Receive processing ended with error.
#16#0004	○				Open processing ended with error.
#16#0008				○	Close processing ended with error.
#16#0100		○			Send processing did not complete in time.
#16#0200			○		Receive processing did not complete in time. (This includes the case where an expected response was not received)
#16#0400	○				Open processing did not complete in time.
#16#0800				○	Close processing did not complete in time.
#16#0010					Processing number error
#16#0020					Send / Receive required judgment error:
#16#1000					Partner Device Error
#16#2000					Partner Device FCS (Checksum) Error
#16#8000	○	○	○	○	An error occurred.

* The error flag stores the value obtained by adding the error flag detected in each process.

● Partner Device Error Code

If the received data from the Partner device is Error data, the Error code is stored in the variable [Output_MErrCode].

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

9.7.2. TCP Connection Status Errors and Corrective Actions

This subsection describes situations in which "TCP connection status error" occurs and the corrective action for it.

● Implications of TCP connection status errors

If this project file is re-executed without taking any corrective action after "TCP connection status error" occurs, or without noticing the occurrence of "TCP connection status error", "The other node specified with the input variable" destination IP address (DstAdr) "and" destination port (DstTcpPort) "may not be waiting for connection (hereinafter referred to as" open processing error "). This is affected by the "TCP connection status error" at the end of the previous communication processing. (For details on errors that can occur, refer to "9.7.1 Error Code List".)

● TCP connection status errors

The cause of the "TCP connection status error" after Close processing and the "Open processing error" during the next communication processing may be that the close processing of the other device is not completed. This means that although all the processing of this project file has been completed in the controller (up to the Close process), the Close completion notification from the other device has not been received (the completion of the Close process of the other device is unconfirmed).

● Corrective Actions

Because the Close process on the partner device may not be completed, verify whether the communication port on the partner device is closed. If the result is that it is not closed, or its status cannot be verified, the communication port of the partner device must be reset. The method of resetting the communication port of the partner device may be software restart processing or restart processing by power cycle OFF → ON. For details, refer to the manual for the partner device.

Precautions for Correct Use

Execute the reset process of the communication port of the partner device after confirming that the partner device is not connected to another device.

●Status of controller (Built-in EtherNet/IP port) when TCP connection status is abnormal

When "TCP connection status error" occurs, processing by this project file is finished, but Resend / Time Monitor by Built-in EtherNet/IP port (TCP/IP function) may be operating (refer to "9.3.2. Time monitoring function"). However, Re-send will stop in the following situations, so there is no need to manually stop it.

- When an Open processing request is made again by starting a project file
- If during Re-send, the communication failure, such as cable disconnection, is resolved
- When Resend processing ends by result of TCP/IP time monitoring (timeout)
- When there is a controller restart, or power to the controller is turned OFF.

10. Revision History

Revision Symbol	Revised year and date	Revised Page and Reason
01	April 2022	First Publication

OMRON Corporation **Industrial Automation Company**
Kyoto, JAPAN
Contact: www.ia.omron.com

Regional Headquarters
OMRON EUROPE B.V.
Wegalaan 67-69, 2132 JD Hoofddorp
The Netherlands
Tel: (31)2356-81-300/Fax: (31)2356-81-388

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

OMRON ASIA PACIFIC PTE. LTD.
438B Alexandra Road, #08-01/02 Alexandra
Technopark, Singapore 119968
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. Z443-E-01 0622 (0422)