

Machine Automation Controller NJ-series

General-purpose Serial Connection Guide (RS-232C) OMRON Corporation

V750 series RFID System

Network
Connection
Guide

About Intellectual Property Right and Trademarks

Microsoft product screen shots reprinted with permission from Microsoft Corporation.

Windows is a registered trademark of Microsoft Corporation in the United States and other countries.

EtherCAT® is registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany.

Ethernet is a registered trademark of Xerox Corporation.

Java and all Java-related trademarks and logos are trademarks of Oracle Corporation, Inc., in the United States and other countries.

Company names and product names in this document are the trademarks or registered trademarks of their respective companies.

Table of Contents

| | |
|---|-----------|
| 1. Related Manuals | 4 |
| 2. Terms and Definition | 4 |
| 3. Remarks | 5 |
| 4. Overview | 7 |
| 5. Applicable Devices and Support Software | 8 |
| 5.1. Applicable Devices | 8 |
| 5.2. Device Configuration | 9 |
| 6. Serial Communications Settings | 11 |
| 6.1. Serial Communications Settings | 11 |
| 6.2. Cable Wiring Diagram | 13 |
| 6.3. Example of Checking Connection | 14 |
| 7. Connection Procedure | 15 |
| 7.1. Work Flow | 15 |
| 7.2. Setting Up the RFID Reader/Writer | 16 |
| 7.3. Setting Up the Controller | 21 |
| 7.4. Connection Status Check | 34 |
| 8. Initialization Method | 37 |
| 8.1. Controller | 37 |
| 8.2. Initializing the RFID Reader/Writer | 39 |
| 9. Program | 40 |
| 9.1. Overview | 40 |
| 9.2. Destination Device Command | 44 |
| 9.3. Error Detection Processing | 48 |
| 9.4. Variables | 51 |
| 9.5. ST Program | 55 |
| 9.6. Timing Charts | 70 |
| 9.7. Error Process | 74 |
| 10. Revision History | 80 |

1. Related Manuals

The table below lists the manuals related to this document.

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

| Cat.No | Model | Manual name |
|--------|--------------------------------|---|
| W500 | NJ501-□□□□ NJ301-□□□□ | NJ-series CPU Unit Hardware User's Manual |
| W501 | NJ501-□□□□ NJ301-□□□□ | NJ-series CPU Unit Software User's Manual |
| W494 | CJ1W-SCU□2 | CJ-series Serial Communications Units Operation Manual for NJ-series CPU Unit |
| W502 | NJ501-□□□□ NJ301-□□□□ | NJ-series Instructions Reference Manual |
| W504 | SYSMAC-SE2□□□□ | Sysmac Studio Version 1 Operation Manual |
| Z235 | V750-BA50C04-US V740-HS01□□ | V750-series UHF RFID System User's Manual |



2. Terms and Definition

| Terms | Explanation and Definition |
|-----------------|---|
| No-protocol | No-protocol Mode enables you to receive or send data by using SCU Send Serial (SerialSend) or SCU Receive Serial (SerialRcv) instructions. In this mode, messages are sent/received to/from a destination device. |
| Send message | A send message is a communications frame (command) sent from the Serial Communications Unit to the destination device. This is executed by the SerialSend instruction and sent to the destination device. |
| Receive message | A receive message is a communications frame (response) sent from the destination device to the Serial Communications Unit. The SerialRcv instruction is used to read data received from the destination device. |

3. Remarks

- (1) Understand the specifications of devices which are used in the system. Allow some margin for ratings and performance. Provide safety measures, such as installing safety circuit in order to ensure safety and minimize risks for abnormal occurrence.
- (2) To ensure system safety, always read and heed the information provided in all Safety Precautions, Precautions for Safe Use, and Precaution for Correct Use of manuals for each device used in the system.
- (3) The users are encouraged to confirm the standards and regulations that the system must conform to.
- (4) It is prohibited to copy, to reproduce, and to distribute a part of or whole part of this document without the permission of OMRON Corporation.
- (5) This document provides the latest information as of April 2013. The information contained in this document is subject to change for improvement without notice.

The following notation is used in this document.

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



Precautions for Safe Use

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



Precautions for Correct Use

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



Additional Information

Provides useful information.

Additional information to increase understanding or make operation easier.

Symbols



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



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

4. Overview

This document describes the procedure for connecting the RFID Reader/Writer (V750 series) of OMRON Corporation (hereinafter referred to as OMRON) with the NJ-series Machine Automation Controller (hereinafter referred to as Controller) via serial communications, and describes the procedure for checking their connection.

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

The user program in this project file is used to check the serial connection by executing the "GETR TYP FWV command (read the product type and firmware version of memory data)" on the destination device.

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

| Name | File name | Version |
|--|------------------------------|----------|
| Sysmac Studio project file (extension: SMC) | OMRON_V750_SERI232_EV101.smc | Ver.1.01 |

*Hereinafter, the Sysmac Studio project file is referred to as the "project file".

The user program in the project file is referred to as the "program".

Caution

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



5. Applicable Devices and Support Software

5.1. Applicable Devices

The applicable devices are given below.

| Manufacturer | Meaning | Model |
|--------------|--|--------------------------|
| OMRON | NJ-series CPU Unit | NJ501-□□□□ NJ301-□□□□ |
| OMRON | Serial Communications Unit | CJ1W-SCU□2 |
| OMRON | RFID Reader/Writer (Complies with FCC and EN) | V750-BA50C04-US |
| OMRON | Antenna | V740-HS01□□ |
| OMRON | Antenna Cable | V740-A01□□M |



Additional Information

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

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

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

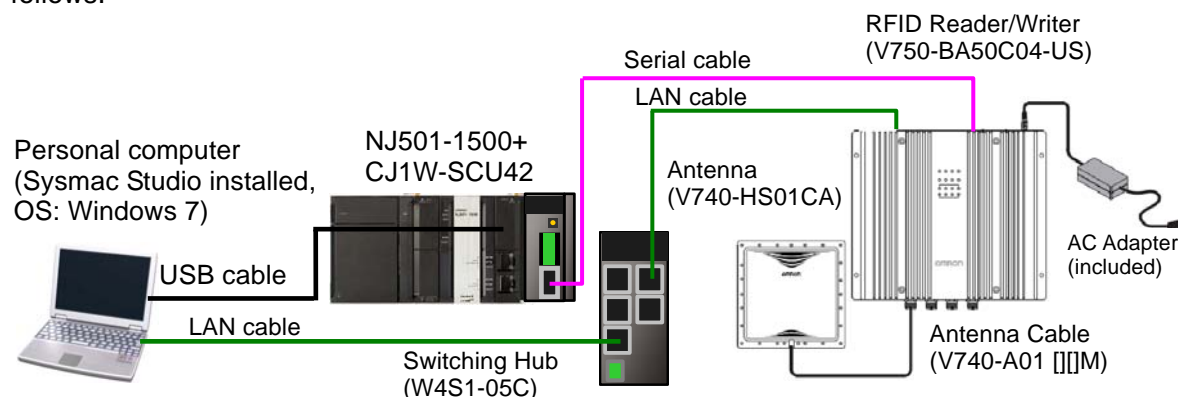


Additional Information

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

5.2. Device Configuration

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



| Manufacturer | Name | Model | Version |
|--------------|--------------------------------------|------------------------------|-------------------|
| OMRON | Serial Communications Unit | CJ1W-SCU42 | Ver.2.0 |
| OMRON | NJ-series CPU Unit | NJ501-1500 | Ver.1.03 |
| OMRON | Power Supply Unit | NJ-PA3001 | |
| OMRON | Switching Hub | W4S1-05C | Ver.1.0 |
| OMRON | Sysmac Studio | SYSMAC-SE2 | Ver.1.04 |
| OMRON | Sysmac Studio project file | OMRON_V750_SERI232_EV101.smc | Ver.1.01 |
| - | Personal computer (OS:Windows7) | - | |
| - | USB cable (USB 2.0 type B connector) | - | |
| - | Serial cable (RS-232C) | - | |
| - | LAN cable (for setting) | - | |
| OMRON | RFID Reader/Writer | V750-BA50C04-US | Ver.102-102-103-0 |
| OMRON | Antenna (Circular) (4 max.) | V740-HS01CA | |
| OMRON | Antenna Cable | V740-A01 | |
| OMRON | AC Adaptor (Included) | - | |



Precautions for Correct Use

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



Precautions for Correct Use

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



Additional Information

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



Additional Information

For information on the serial cable (RS-232C), refer to *3-3 RS-232C and RS-422A/485 Wiring* in the *CJ-series Serial Communications Units Operation Manual for NJ-series CPU Unit* (Cat.No. W494).



Additional Information

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

6. Serial Communications Settings

This section provides the specifications such as communications parameters and cable wiring that are set in this document.



Additional Information

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

6.1. Serial Communications Settings

The settings for serial communications are shown below.

6.1.1. Communications Settings between the Personal Computer and the RFID Reader/Writer

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

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

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

6.1.2. Communications Settings between the Serial Communications Unit and the RFID Reader/Writer

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

| Setting item | Serial Communications Unit | RFID Reader/Writer |
|----------------------------------|----------------------------|----------------------|
| Unit number | 0 | - |
| Communications (connection) port | Port 2 (RS-232C) | - |
| Serial communications mode | No-protocol | - |
| Data length | 7 bits (Default) | 7 bits (Default) |
| Stop bit | 2 bits (Default) | 2 bits (Default) |
| Parity | Even (Default) | Even (Default) |
| Baud rate | 57,600 bps | 57,600 bps (Default) |
| No-protocol Start Code | Yes 1 (SOH) | SOH (Fixed) |
| No-protocol End Code | Yes (CR+LF) | CR+LF (Fixed) |



Precautions for Correct Use

This manual describes the procedure for setting the CJ1W-SCU42 Serial Communications Unit when the unit number 0, communications port 2 and device name J01 are used. To connect devices under different conditions, refer to 9. *Program* and create a program by changing the variable names and setting values.

6.2. Cable Wiring Diagram

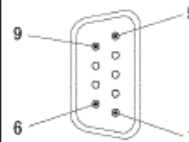
For details on the cable wiring, refer to *Section 3 Installation and Wiring* of the *CJ-series Serial Communications Units Operation Manual* for NJ-series CPU Unit (Cat. No. W494) and *Section 4 Diagnosis and Maintenance-Wiring for cable* of *V750-series UHF RFID System User's Manual*(Cat.No. Z235).

Check the connector configuration and pin assignment for wiring.

■Connector configuration and pin assignment

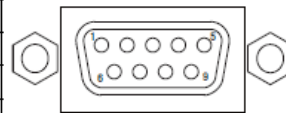
<OMRON CJ1W-SCU42> Applicable connector: D-sub 9-pin

| Pin | Abbreviation | Signal name | I/O |
|------|--------------|---------------------|--------|
| 1 | FG | Shield | --- |
| 2 | SD | Send data | Output |
| 3 | RD | Receive data | Input |
| 4 | RTS (RS) | Request to send | Output |
| 5 | CTS (CS) | Clear to send | Input |
| 6 | 5V | Power supply | --- |
| 7 | DSR (DR) | Data set ready | Input |
| 8 | DTR (ER) | Data terminal ready | Output |
| 9 | SG | Signal ground | --- |
| Hood | FG | Shield | --- |

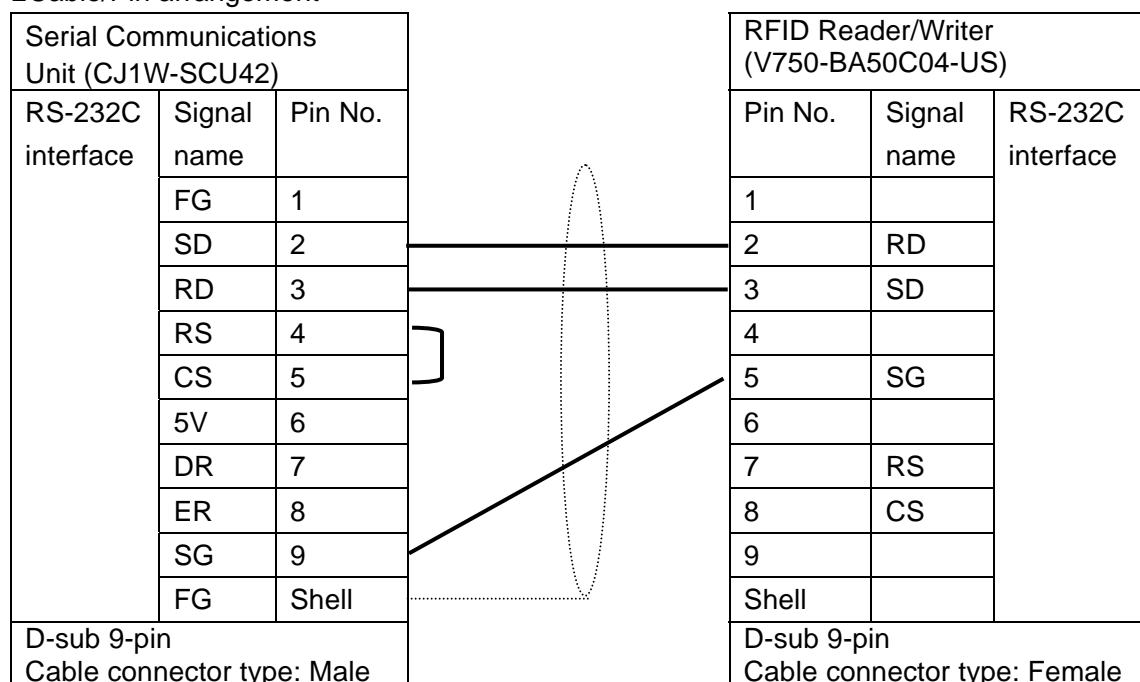


<OMRON V750-BA50C04-US> Applicable connector: D-sub 9-pin

| Pin No. | Name | Function | I/O |
|---------|------|-----------------|-----|
| 1 | --- | --- | --- |
| 2 | RD | Receive Data | IN |
| 3 | SD | Send Data | OUT |
| 4 | --- | --- | --- |
| 5 | SG | Signal Ground | --- |
| 6 | --- | --- | --- |
| 7 | RS | Request to Send | OUT |
| 8 | CS | Clear to Send | IN |
| 9 | --- | --- | --- |



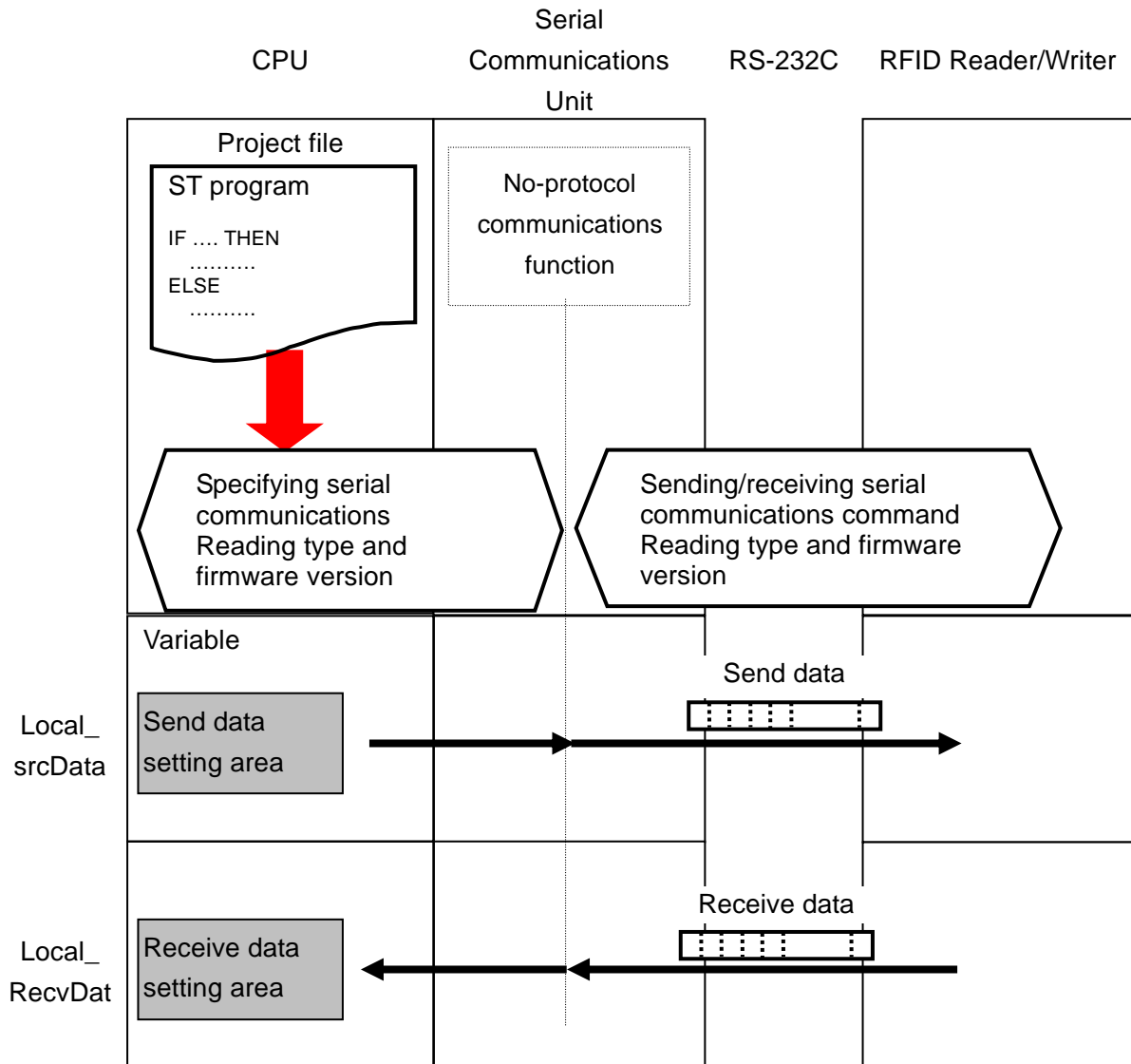
■Cable/Pin arrangement



6.3. Example of Checking Connection

This document shows an example of an ST (structured text) program in which the controller sends/receives messages to/from the RFID Reader/Writer.

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



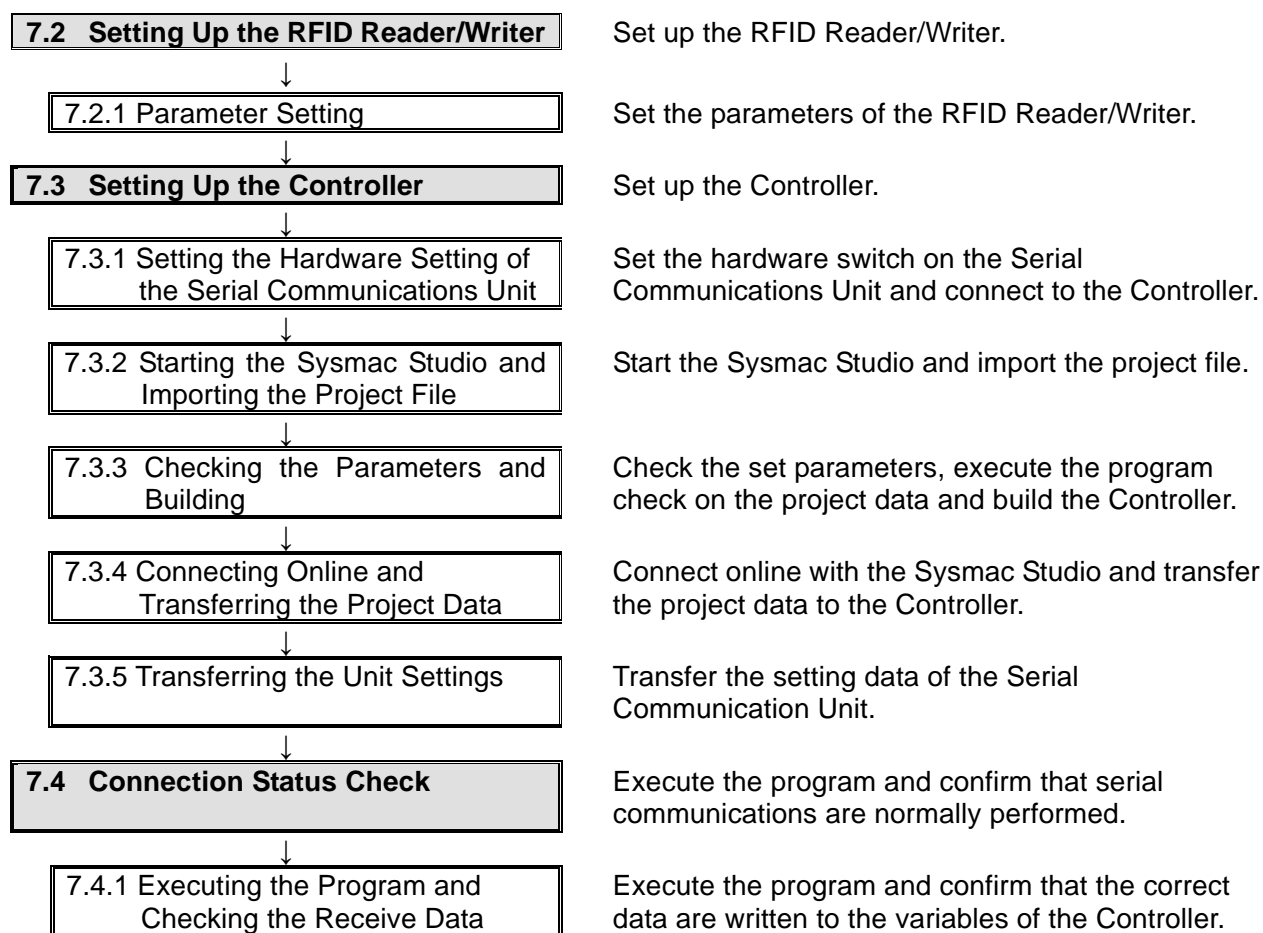
7. Connection Procedure

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

This document explains the procedures for setting up the Controller and RFID Reader/Writer from the factory default setting. For the initialization, refer to *Section 8 Initialization Method*.

7.1. Work Flow

Take the following steps to connect the RFID Reader/Writer to the Controller via serial communications.



Precautions for Correct Use

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

7.2. Setting Up the RFID Reader/Writer

Set up the RFID Reader/Writer.

7.2.1. Parameter Setting

Set the parameters for the RFID Reader/Writer.

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

Set the IP address of the personal computer to 192.168.1.1.



Precautions for Correct Use

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

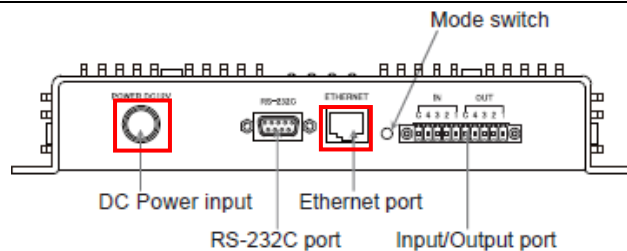
Note that you may need to change the settings of the personal computer depending on the status of the personal computer.

- 1 Connect the antenna to the antenna port on the side of the RFID Reader/Writer.



(Side of RFID Reader/Writer)

- 2 Connect the Switching Hub to the Ethernet port on the other side of the RFID Reader/Writer using the LAN cable. Connect the included AC Adapter cable to the DC power input.



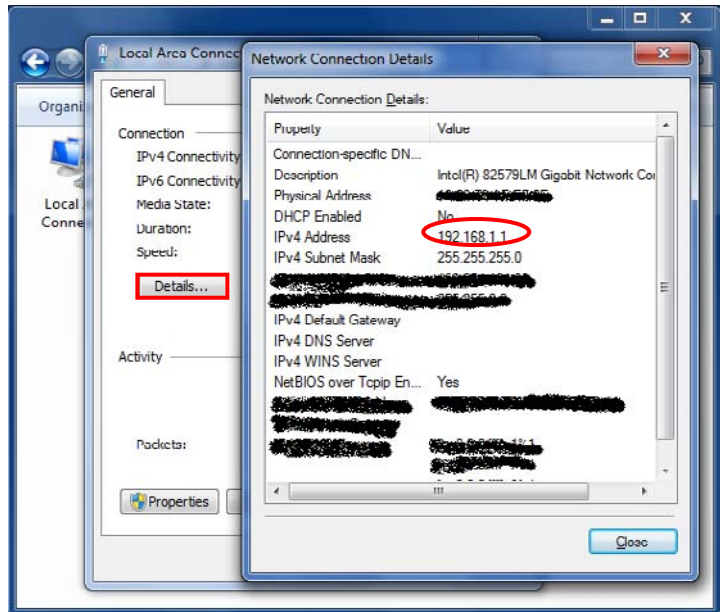
(Other side of RFID Reader/Writer)

- 3 Start Internet Explorer from the personal computer that is connected to the Switching Hub.

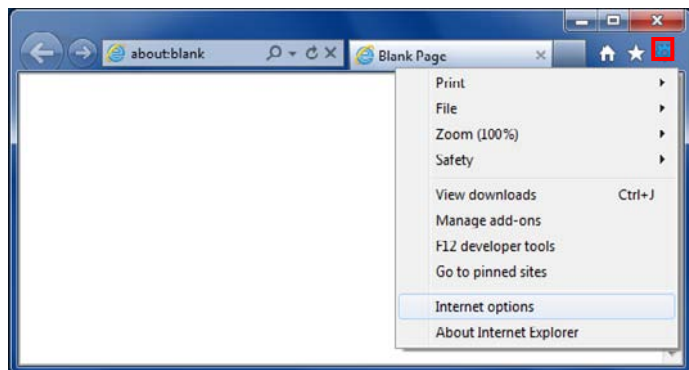


*Set the IP address of the personal computer to 192.168.1.1. Use the following procedure to check the IP address of the personal computer

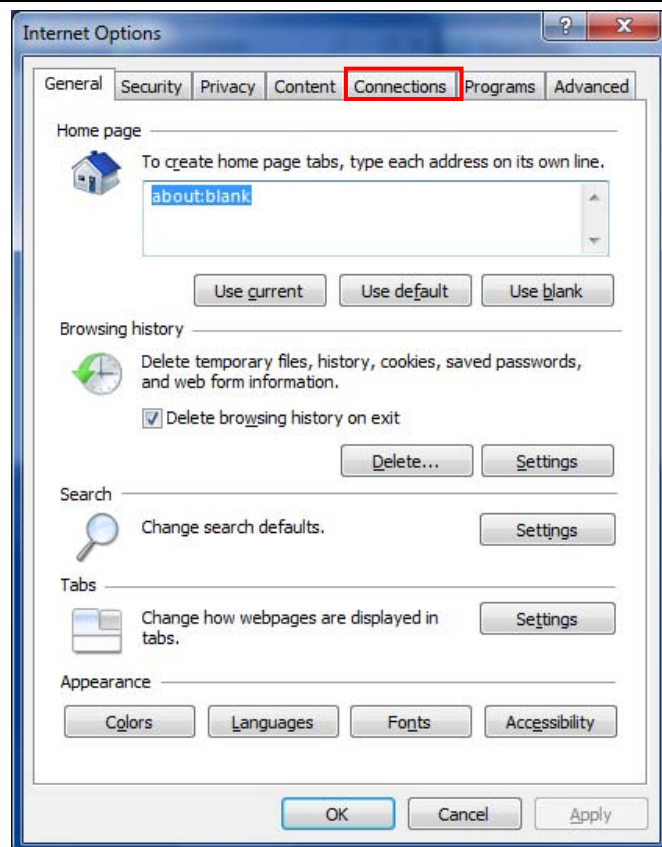
- (1) Click **Connect to the Internet View network status and tasks - Change adapter settings** on the Control Panel.
- (2) Double-click **Local Area Connection** on the Network Connections.
- (3) Click the **Details** Button on the Local Area Connection Status Dialog Box.
- (4) Confirm that the IP address is 192.168.1.1.



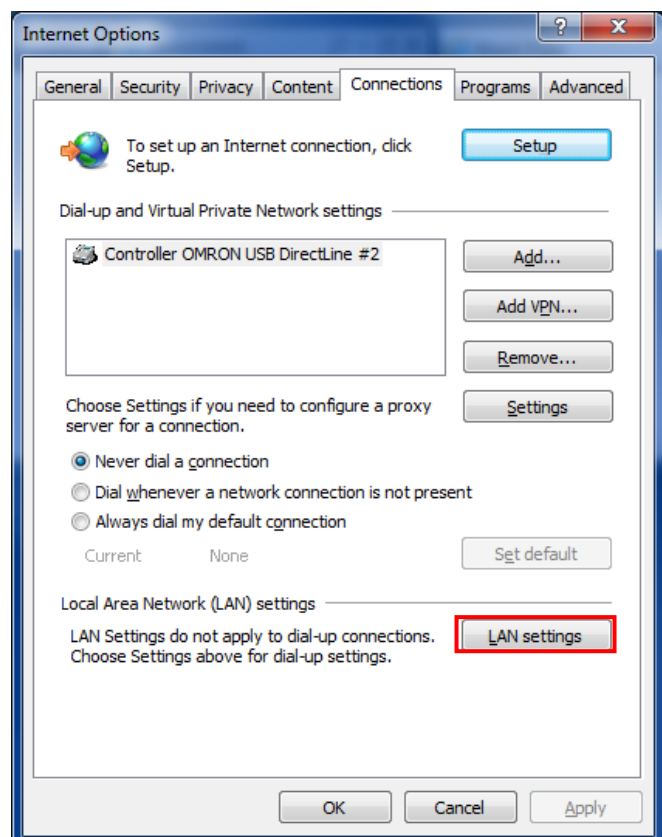
- 4 Click **Tool** (☰) on the command bar of Internet Explorer and select **Internet options**.



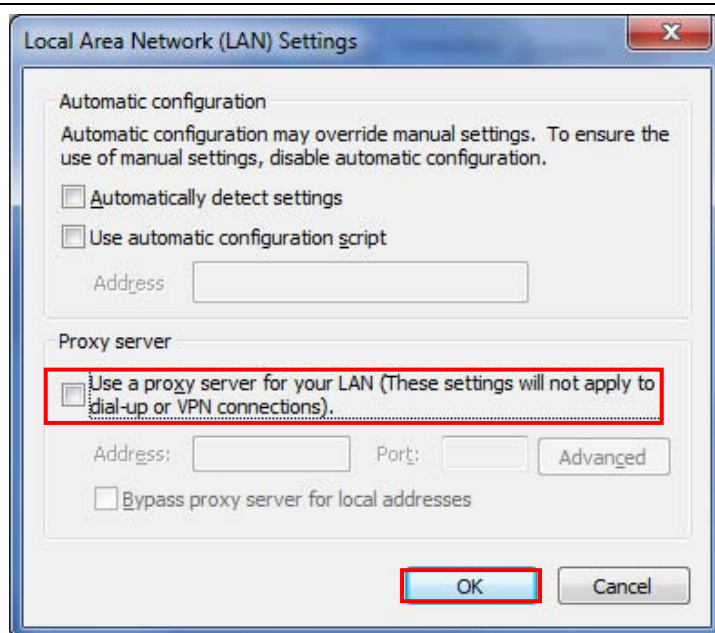
- 5 The Internet Options Dialog Box is displayed. Select the Connections Tab.



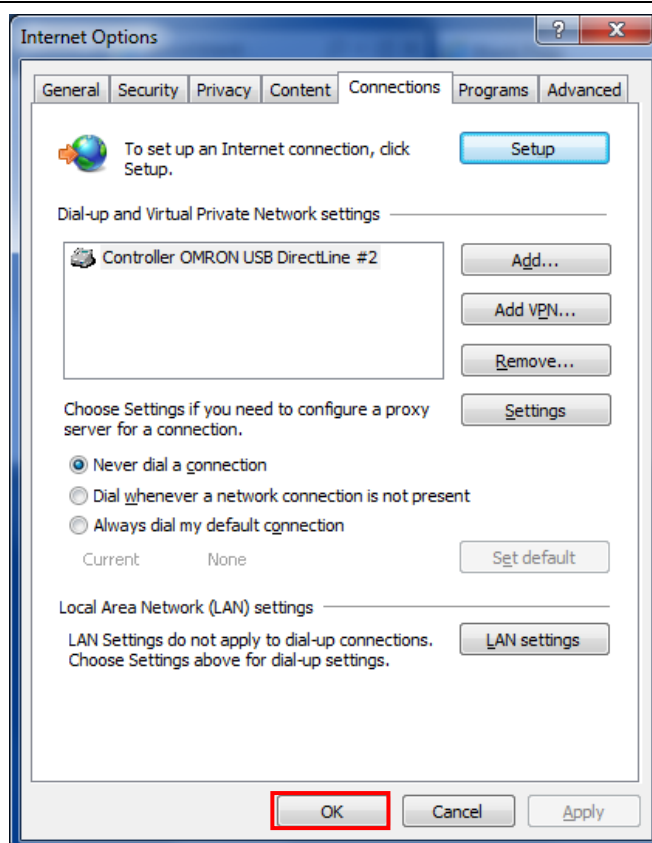
- 6 The Internet Options Dialog Box is displayed. click the **LAN settings** Button.



- 7 The Local Area Network (LAN) Settings Dialog Box is displayed. Confirm that the *Use a proxy server for your LAN* Check Box is cleared in the Proxy server Field and click the **OK** Button.



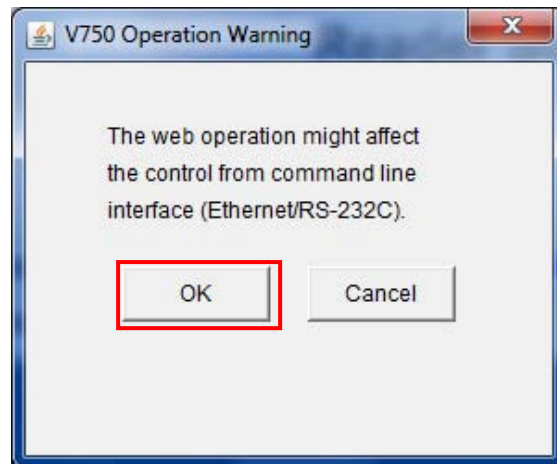
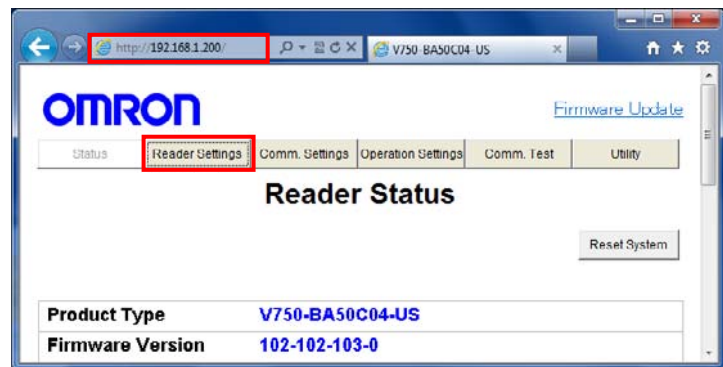
- 8 Click the **OK** Button on the Internet Options Dialog Box.



- 9 Enter `http://192.168.1.200` / in the address bar (Internet Explorer).

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

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

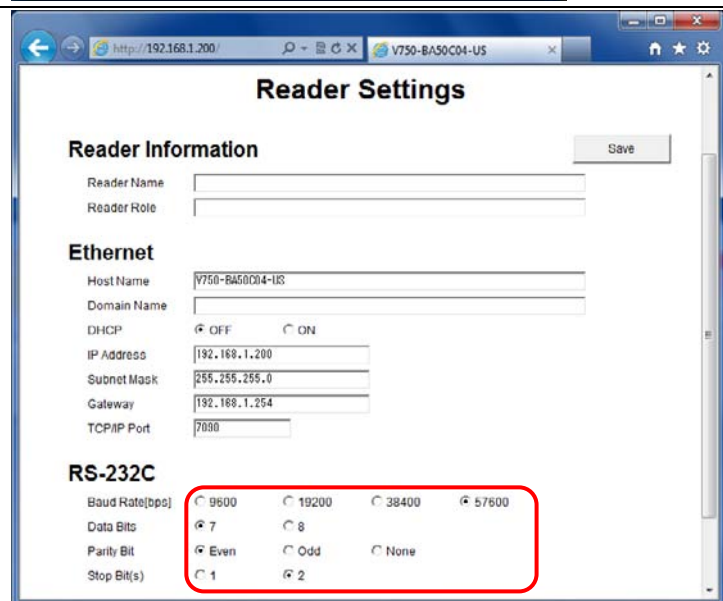


- 10 The Reader Settings Window shows the RS-232C settings. Confirm that the settings are made as follows (all default settings).

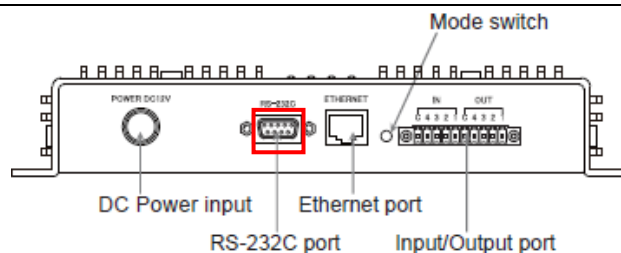
Baud Rate :57600 bps
Data Bits :7 bits
Parity Bit :Even
Stop Bit(s) :2 bits

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

*To change the settings, select a button of the corresponding parameter value. After changing, click the **Save** Button and cycle the power supply to the RFID Reader/Writer.



- 11 Connect the Serial Communications Unit to the RS-232C port on the other side of the RFID Reader/Writer using the Serial cable.



(Other side of RFID Reader/Writer)

7.3. Setting Up the Controller

Set up the Controller.

7.3.1. Setting the Hardware Settings of the Serial Communications Unit

Set the hardware switches on the Serial Communications Unit.



Precautions for Correct Use

Make sure that the power supply is OFF when you perform the settings.

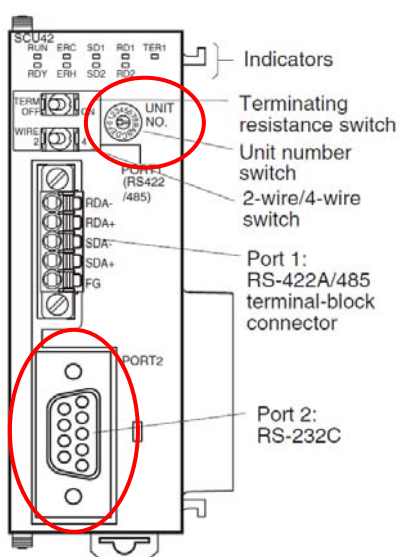
- 1 Make sure that the power supply to the Controller is OFF when you perform the settings.

*If the power supply is turned ON, settings may not be applicable as described in the following procedure.

Refer to the right figure and check the each part name.

*This setting is required to use the Port 2 of Serial Communications Unit.

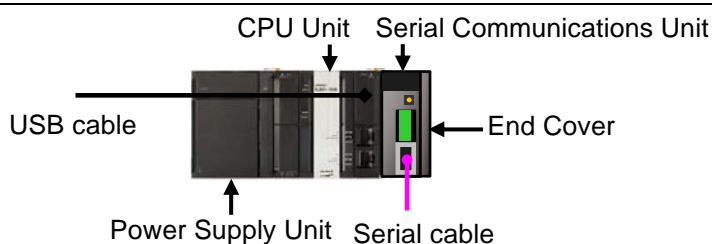
CJ1W-SCU42



- 2 Set the Unit No. Switch to 0. (The unit number is factory-set to 0.)



- 3 Connect the Serial Communications Unit to the Controller as shown on the right. Connect the serial communications cable and USB cable, and turn ON the power supply to the Controller.



7.3.2. Starting the Sysmac Studio and Importing the Project File

Start the Sysmac Studio and import the project file.

Install the Sysmac Studio and USB driver in the personal computer in advance.

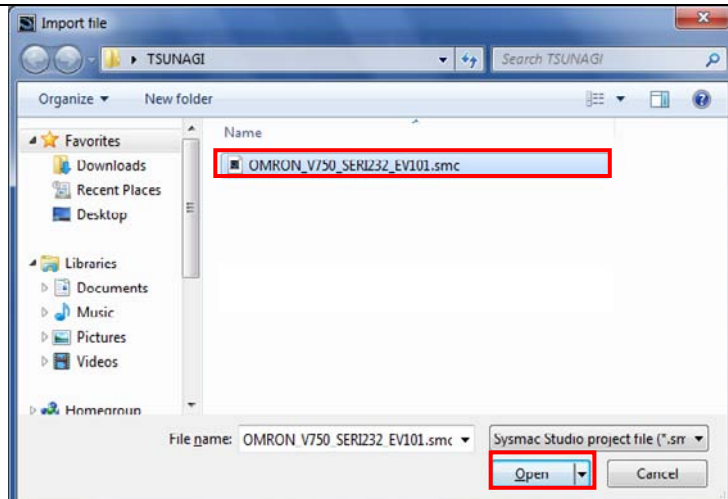
- 1 Start the Sysmac Studio.
Click the **Import** Button.

*If a dialog box is displayed at start confirming the access right, select an option to start.



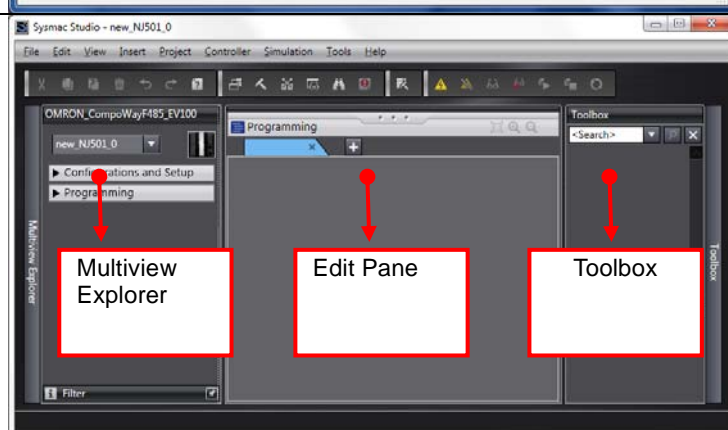
- 2 The Import file Dialog Box is displayed. Select OMRON_V750_SERI232_EV1 01.smc and click the **Open** Button.

*Obtain the project file from OMRON.



- 3 The OMRON_V750_SERI232_EV1 01 project is displayed. The left pane is called Multiview Explorer, the right pane is called Toolbox and the middle pane is called Edit Pane.

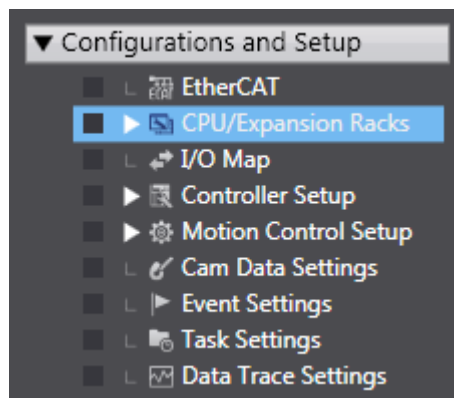
*If an error dialog box is displayed, check the version of the Sysmac Studio.



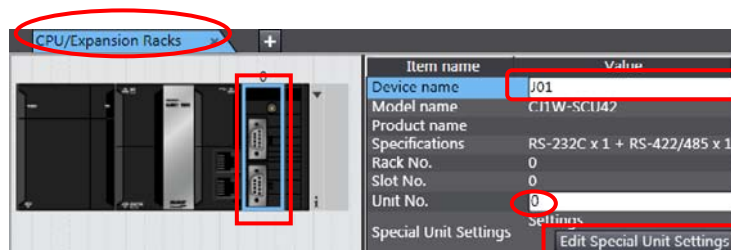
7.3.3. Checking the Parameters and Building

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

- 1 Double-click **CPU/Expansion Racks** under **Configurations and Setup** in the Multiview Explorer.



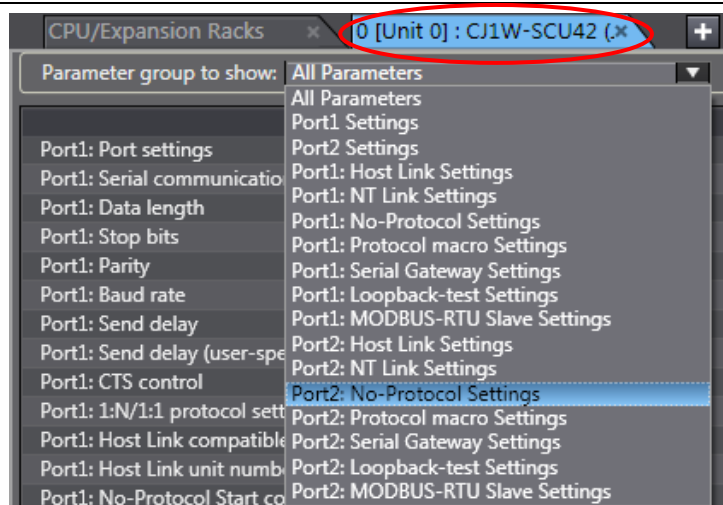
- 2 The CPU/Expansion Racks Tab is displayed on the Edit Pane. Select the Serial Communications Unit icon as shown on the right. Confirm that CJ1W-SCU42 is displayed, the device name is J01, and the unit number is 0.



*If the setting is different from the above, change the value.

Click **Edit Special Unit Settings**.

- 3 The 0 [Unit 0]: Tab is displayed. Select *Port2: No-Protocol Settings* from the pull-down list of Parameter group to show.

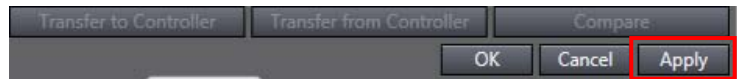
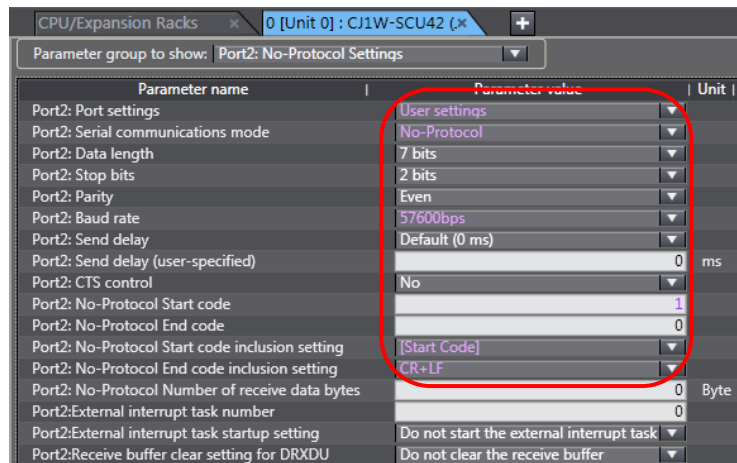


- 4 Parameter group to show is set to *Port2: No-Protocol Settings*.

The setting items for Port2: No-Protocol Settings are shown.

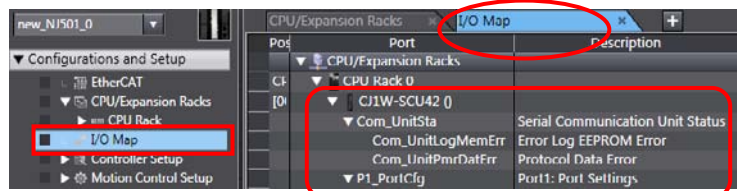
Confirm that the *Port2: Port Settings* is set to *User settings* and other settings are the same as those listed in Section 6.1.

*If the settings are different from the above, change the values from the pull-down list. After changing values, click the **Apply** Button.



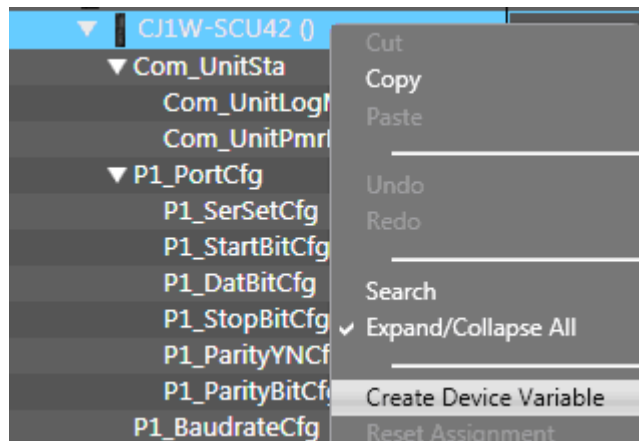
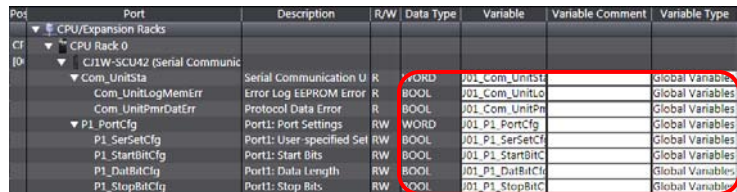
- 5 Double-click **I/O Map** under **Configurations and Setup** on the Multiview Explorer.

The I/O Map Tab is displayed and then the parameters for the Unit are listed.

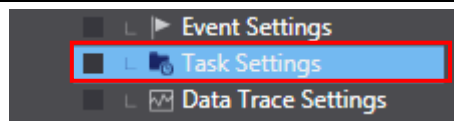


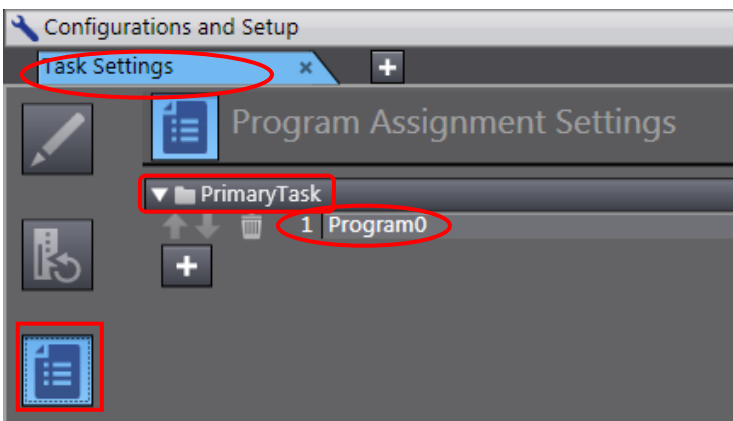
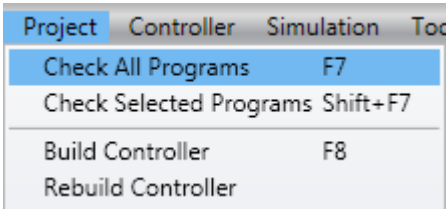
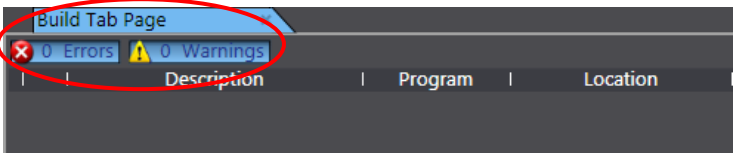
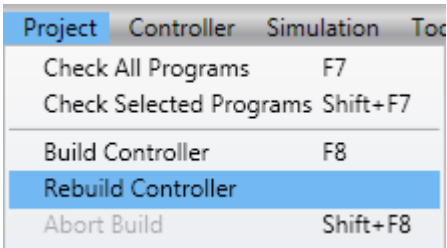

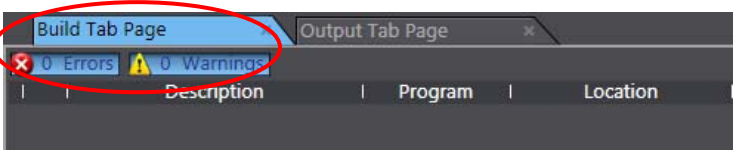
- 6 Confirm that data in the Variable Columns start with J01 and the Global Variable is set in each Variable Type Column.

*If the settings are different from the above, right-click on **CJ1W-SCU42** and select **Create Device Variable**.



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



- 8 The Task Settings Tab Page is displayed in the Edit Pane. Click the **Program Assignment Settings** Button and check that Program0 is set under PrimaryTask.
- 
- 9 Select **Check All Programs** from the Project Menu.
- 
- 10 The Build Tab Page is displayed in the Edit Pane. Confirm that "0 Errors" and "0 Warnings" are displayed.
- 
- 11 Select **Rebuild Controller** from the Project Menu.
- 
- A screen is displayed indicating the conversion is being performed.
- 
- 12 Confirm that "0 Errors" and "0 Warnings" are displayed in the Build Tab Page.
- 

7.3.4. Connecting Online and Transferring the Project Data

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

WARNING

Always confirm safety at the destination node before you transfer a user program, configuration data, setup data, device variables, or values in memory used for CJ-series Units from the Sysmac Studio.

The devices or machines may perform unexpected operation regardless of the operating mode of the CPU Unit.

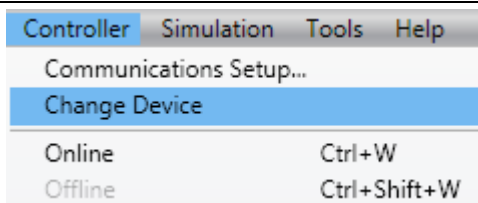


Caution

Always confirm safety before you reset the Controller or any components.

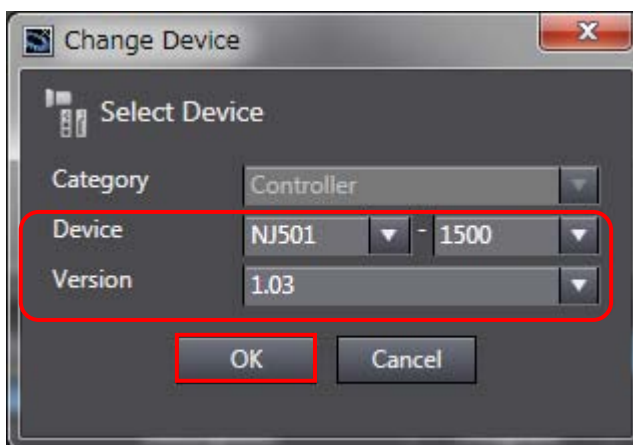


- 1 Select **Change Device** from the Controller Menu.



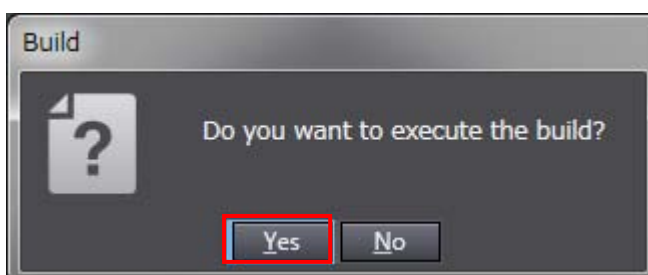
- 2 The Change Device Dialog Box is displayed.
Confirm that the Device and Version are set as shown on the right and click the **OK** Button.

*If the settings are different from the above, change the values from the pull-down list.

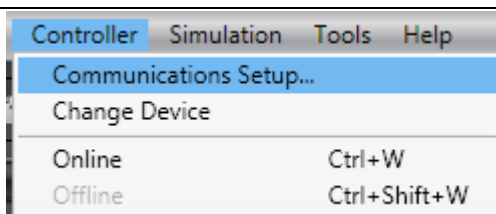


- 3 If the settings were changed in Step 2, the Build Dialog Box is displayed. Click the **Yes** Button.

*This dialog box is not displayed if no change was made.

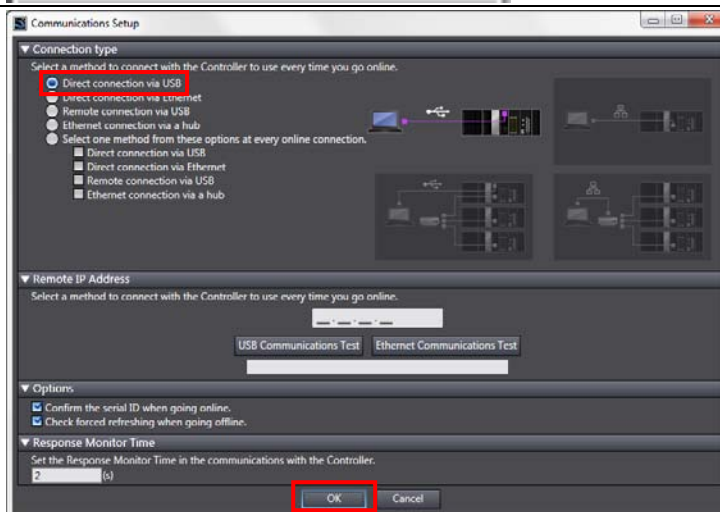


- 4 Select **Communications Setup** from the Controller Menu.

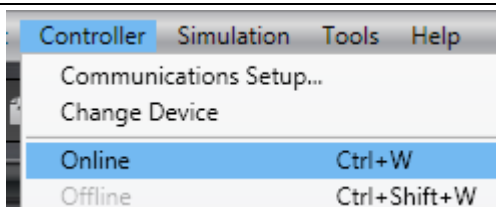


- 5 The Communications Setup Dialog Box is displayed. Select the *Direct connection via USB* Option in the Connection Type Field.

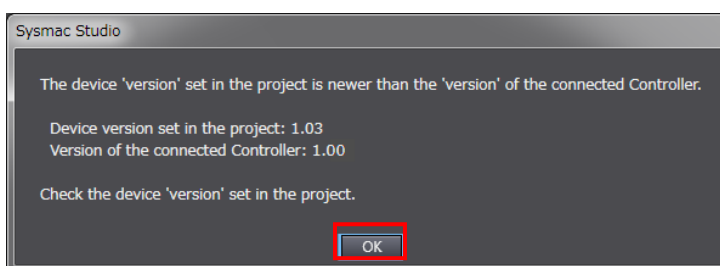
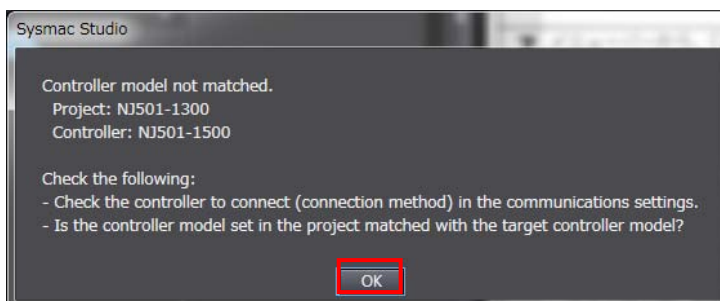
Click the **OK** Button.



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



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



7

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

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

*The displayed serial ID differs depending on the device.

Sysmac Studio

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

Yes

No

Sysmac Studio

Serial ID not matched.

Project:
Name: [new_NJ501_0]
Serial ID: [R01-07X11-0555]

Controller:
Name: [new_NJ501_0]
Serial ID: [R01-07X11-0550]

Do you want to continue the connection processing? (Y/N)

Yes

No

Sysmac Studio

Do you want to change the Serial ID in the project to the controller's Serial ID? (Y/N)
(It will be used at the ID check of next online connection.)

Yes

No



Additional Information

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

8

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

Configurations and Setup

9

Select **Synchronization** from the Controller Menu.

ControllerSimulationToolsHelp

Communications Setup...

Change Device

OnlineCtrl+W

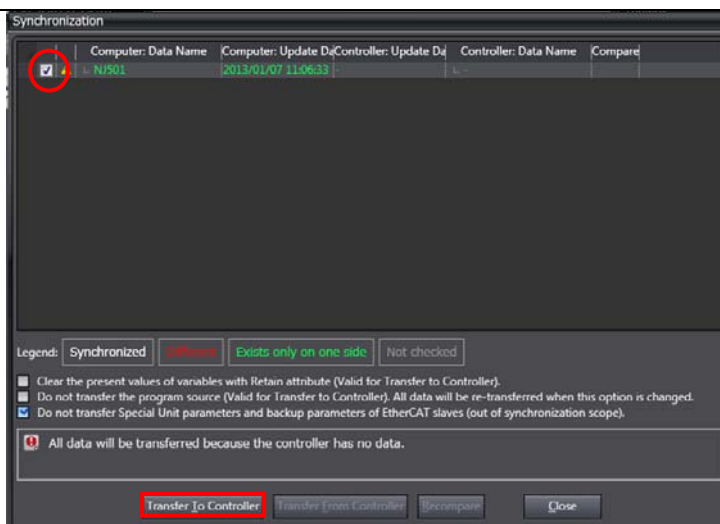
OfflineCtrl+Shift+W

SynchronizationCtrl+M

10 The Synchronization Dialog Box is displayed.

Confirm that the data to transfer (NJ501 in the right figure) is selected. Then, click the **Transfer to Controller** Button.

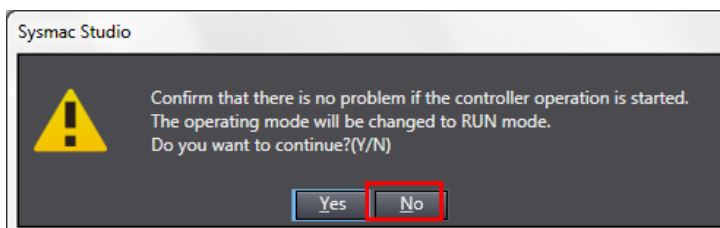
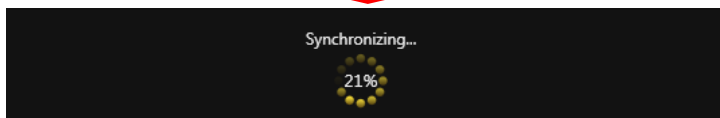
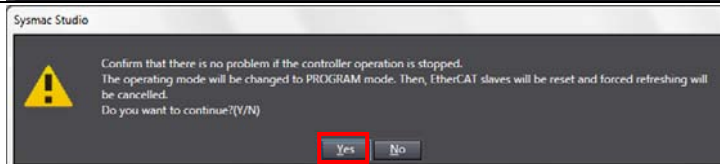
*After executing the **Transfer to Controller** Button, the Sysmac Studio project data is transferred to the Controller and the data are compared.



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

A screen stating "Synchronizing" is displayed.

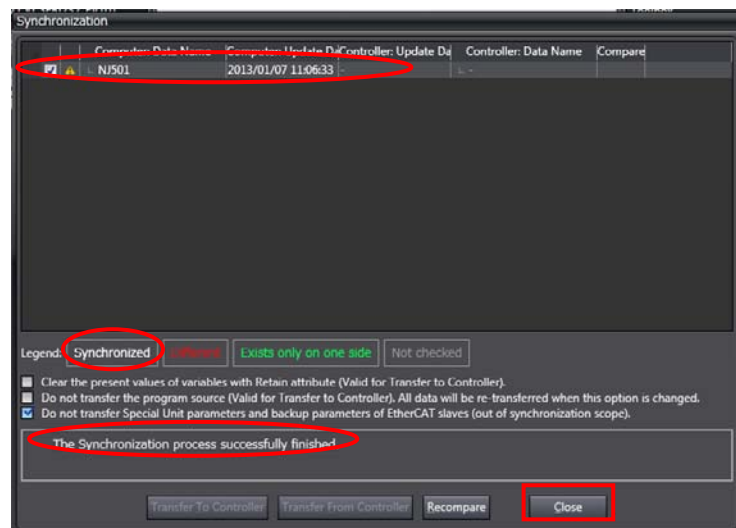
A confirmation dialog box is displayed. Click the **No** Button.



- 12 Confirm that the synchronized data is displayed with the color specified by "Synchronized" and that a message is displayed stating "The synchronization process successfully finished". If there is no problem, click the **Close** Button.

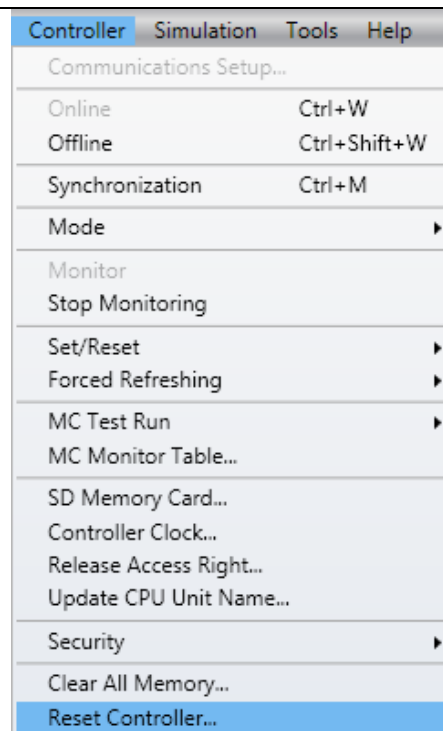
*A message stating "The synchronization process successfully finished" means that the project data of Sysmac Studio and that of the Controller match.

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

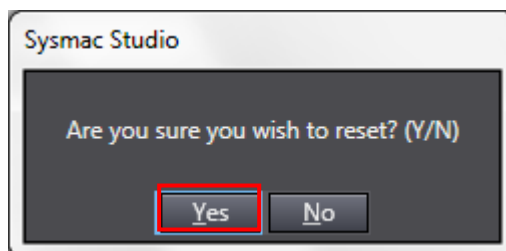
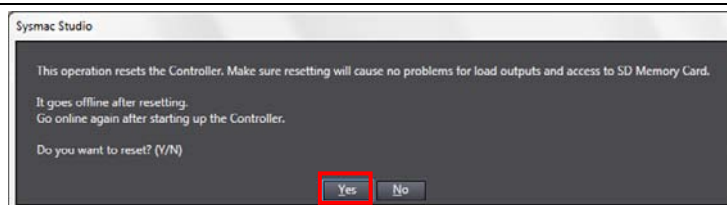


- 13 Select **Reset Controller** from the Controller Menu.

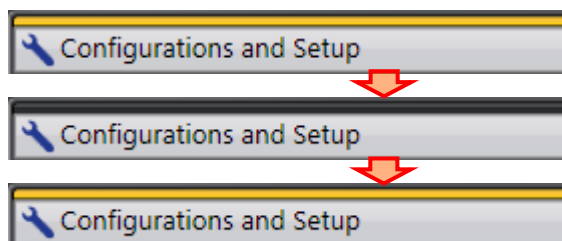
*When Mode is set to RUN Mode, Reset Controller cannot be selected. In this case, select **Mode - PROGRAM Mode** from the Controller Menu to change to PROGRAM mode and perform the procedure in this step.



- 14 A confirmation dialog box is displayed several times. Click the **Yes** Button.



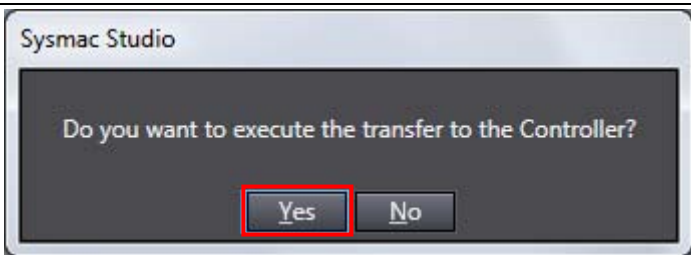

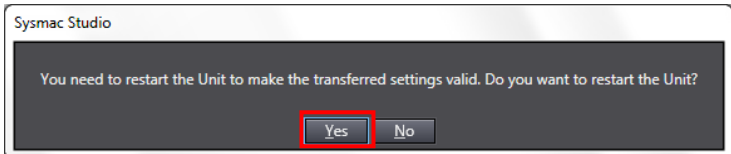
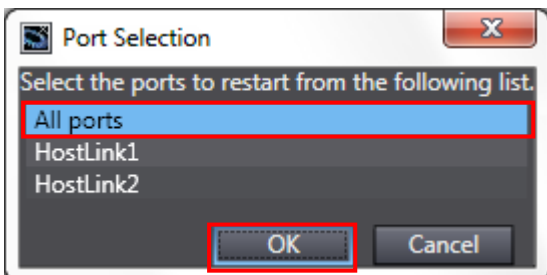
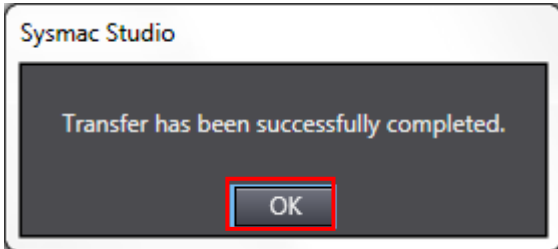
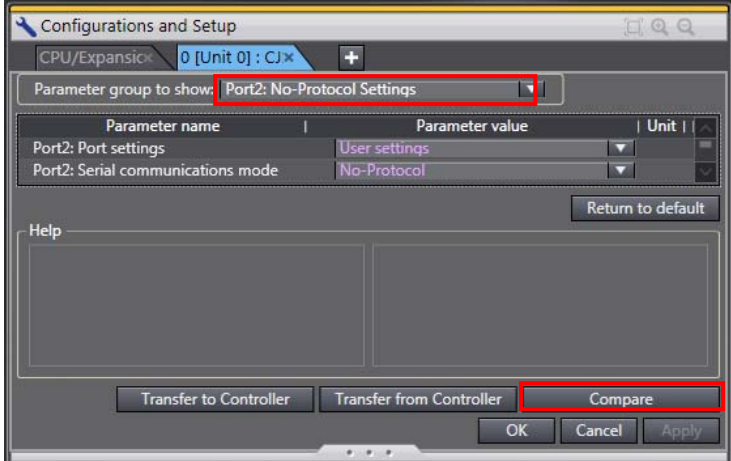
- 15 The Controller is reset, and Sysmac Studio goes offline. The yellow bar on the top of the Edit Pane disappears. Use steps 6 to 8 to go online again.



7.3.5. Transferring the Unit Settings

Transfer the setting data of the Serial Communication Unit.

| | | |
|---|--|--|
| 1 | Select Mode - PROGRAM Mode from the Controller Menu. | |
| 2 | A confirmation dialog box is displayed. Click the Yes Button. | |
| 3 | PROGRAM mode is displayed on the Controller Status Pane. | |
| 4 | Double-click CPU/Expansion Racks under Configurations and Setup in the Multiview Explorer. Select the Serial Communications Unit icon. Click Edit Special Unit Settings . | |
| 5 | The 0 [Unit 0]: Tab is displayed. Click the Transfer to Controller Button. | |

| 6 | <p>A confirmation dialog box is displayed.</p> <p>Click the Yes Button.</p> <p>A dialog box is displayed indicating transferring is being performed.</p> <p>A confirmation dialog box is displayed.</p> <p>Click the Yes Button.</p> |    | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|------------------------------------|--|---|----------------|-----------------|-----------------|----------------------|---------------|---------------|-----------------------------------|-------------|-------------|--------------------|--------|--------|------------------|--------|--------|---------------|------|------|------------------|----------|----------|-------------------|----------------|----------------|------------------------------------|---|---|--------------------|----|----|-------------------------------|---|---|
| 7 | <p>The Port Selection Dialog Box is displayed.</p> <p>Select All ports and click the OK Button.</p> |  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 8 | <p>A confirmation dialog box is displayed.</p> <p>Click the OK Button.</p> |  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 9 | <p>Select <i>Port2: No-Protocol Settings</i> from the pull-down list of Parameter group to show.</p> <p>Click the Compare Button.</p> |  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 10 | <p>Confirm that “≠” (mismatch) is not shown in the red frame on the right.</p> | <table><thead><tr><th>Parameter name</th><th>Parameter value</th><th>Compare results</th></tr></thead><tbody><tr><td>Port2: Port settings</td><td>User settings</td><td>User settings</td></tr><tr><td>Port2: Serial communications mode</td><td>No-Protocol</td><td>No-Protocol</td></tr><tr><td>Port2: Data length</td><td>7 bits</td><td>7 bits</td></tr><tr><td>Port2: Stop bits</td><td>2 bits</td><td>2 bits</td></tr><tr><td>Port2: Parity</td><td>Even</td><td>Even</td></tr><tr><td>Port2: Baud rate</td><td>57600bps</td><td>57600bps</td></tr><tr><td>Port2: Send delay</td><td>Default (0 ms)</td><td>Default (0 ms)</td></tr><tr><td>Port2: Send delay (user-specified)</td><td>0</td><td>0</td></tr><tr><td>Port2: CTS control</td><td>No</td><td>No</td></tr><tr><td>Port2: No-Protocol Start code</td><td>1</td><td>1</td></tr></tbody></table> | Parameter name | Parameter value | Compare results | Port2: Port settings | User settings | User settings | Port2: Serial communications mode | No-Protocol | No-Protocol | Port2: Data length | 7 bits | 7 bits | Port2: Stop bits | 2 bits | 2 bits | Port2: Parity | Even | Even | Port2: Baud rate | 57600bps | 57600bps | Port2: Send delay | Default (0 ms) | Default (0 ms) | Port2: Send delay (user-specified) | 0 | 0 | Port2: CTS control | No | No | Port2: No-Protocol Start code | 1 | 1 |
| Parameter name | Parameter value | Compare results | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Port2: Port settings | User settings | User settings | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Port2: Serial communications mode | No-Protocol | No-Protocol | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Port2: Data length | 7 bits | 7 bits | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Port2: Stop bits | 2 bits | 2 bits | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Port2: Parity | Even | Even | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Port2: Baud rate | 57600bps | 57600bps | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Port2: Send delay | Default (0 ms) | Default (0 ms) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Port2: Send delay (user-specified) | 0 | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Port2: CTS control | No | No | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Port2: No-Protocol Start code | 1 | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

7.4. Connection Status Check

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

Caution

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

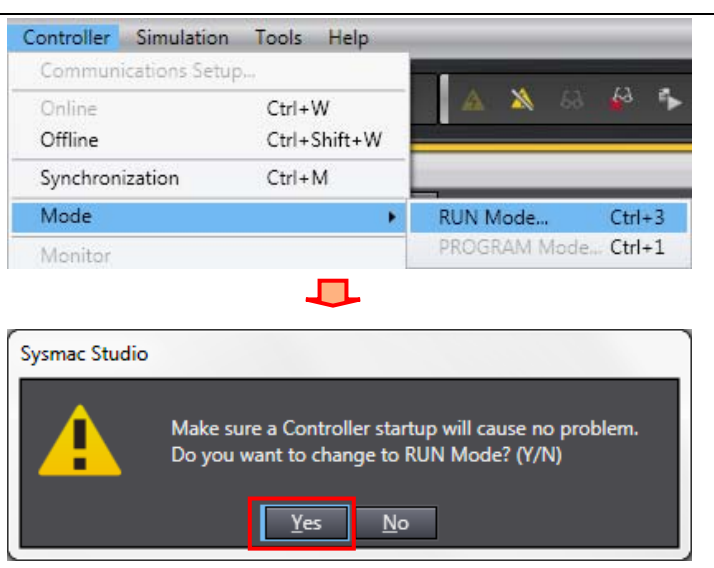
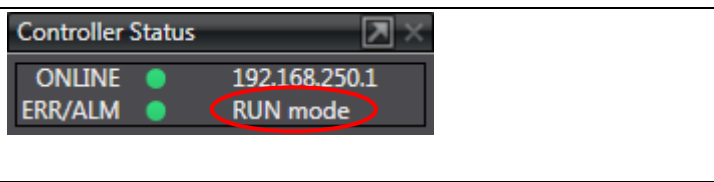
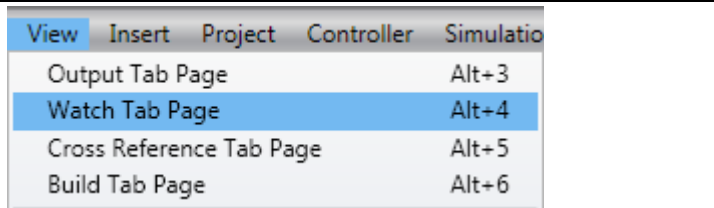


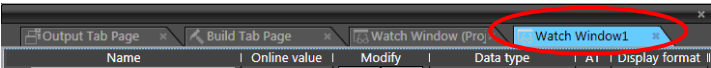
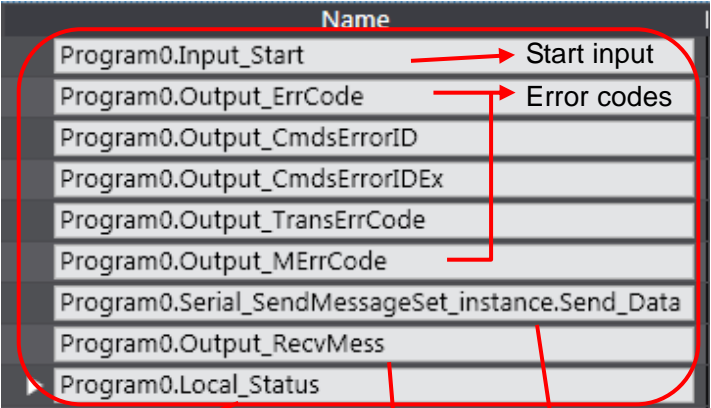
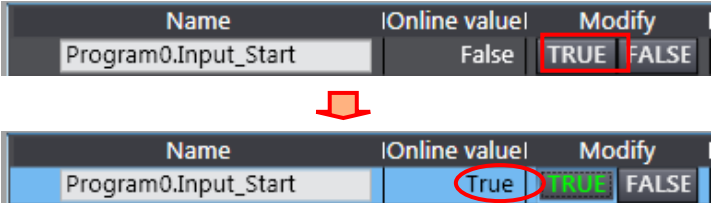
Precautions for Correct Use

Please confirm that the serial cable is connected before proceeding to the following steps. If it is not connected, turn OFF the power of the devices, and then connect the serial cable.

7.4.1. Executing the Program and Checking the Receive Data

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

| | |
|--|---|
| <p>1 Select Mode - RUN Mode from the Controller Menu.</p> <p>A confirmation dialog box is displayed. Click the Yes Button.</p> |  <p>Sysmac Studio</p> <p>Make sure a Controller startup will cause no problem. Do you want to change to RUN Mode? (Y/N)</p> <p>Yes No</p> |
| <p>2 RUN mode is displayed on the Controller Status Pane.</p> |  <p>Controller Status</p> <p>ONLINE ● 192.168.250.1</p> <p>ERR/ALM ● RUN mode</p> |
| <p>3 Select Watch Tab Page from the View Menu.</p> |  <p>View Insert Project Controller Simulation</p> <p>Output Tab Page Alt+3</p> <p>Watch Tab Page Alt+4</p> <p>Cross Reference Tab Page Alt+5</p> <p>Build Tab Page Alt+6</p> |

| | | |
|---|---|---|
| 4 | <p>The Watch Tab Page 1 is displayed in the lower section of the Edit Pane.</p> |  |
| 5 | <p>Confirm that the variables shown on the right are displayed in the Name Columns.</p> <p>*To add a variable, click <i>Input Name...</i></p> <p>*Program0 of the Name is omitted from the following descriptions.</p> |  <p>Start input</p> <p>Error codes</p> <p>Program execution status</p> <p>Receive data</p> <p>Send data</p> |
| 6 | <p>Click TRUE on the Modify Column of <i>Input_Start</i>.</p> <p>The Online value of <i>Input_Start</i> changes to True.</p> <p>The program is operated and serial communications are performed with the destination device.</p> |  |

- 7 When the communications ends normally, each error code changes to 0.

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

The Online value of *Local_Status.Done*, which indicates the program execution status, changes to True. In the case of error end, *Local_Status.Error* changes to True.

*When *Input_Start* changes to FALSE, each *Local_Status* variable also changes to False. For details, refer to 9.6 *Timing Charts*.

| Name | Online value | Modify |
|-------------------------------|--------------|---|
| Program0.Input_Start | True | <input checked="" type="checkbox"/> TRUE <input type="checkbox"/> FALSE |
| Program0.Output_ErrCode | 0000 | |
| Program0.Output_CmdsErrorID | 0000 | |
| Program0.Output_CmdsErrorIDEx | 0000 0000 | |
| Program0.Output_TransErrCode | 0000 | |
| Program0.Output_MErrCode | 0000 0000 | |

| Name | Online value | Modify |
|-------------------------|--------------|---|
| ▼ Program0.Local_Status | | |
| Busy | False | <input type="checkbox"/> TRUE <input checked="" type="checkbox"/> FALSE |
| Done | True | <input type="checkbox"/> TRUE <input checked="" type="checkbox"/> FALSE |
| Error | False | <input type="checkbox"/> TRUE <input checked="" type="checkbox"/> FALSE |

- 8 The response data received from the destination device is stored in *Output_RecvMess* (*Serial_SendMessageSet_instance.Send_Data* is a send command.)

Specify an area where you want to reference in the Watch Tab Page 1 as shown in the right figure.

*The response data differ depending on the device used

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

| Name |
|--|
| Program0.Serial_SendMessageSet_instance.Send_Data |
| Program0.Output_RecvMess |
| Online value |
| GETR typ fwv1E |
| GETR0000 typ="\$V750-BA50C04-US\$" fwv=102-102-103-012 |

Receive data

- Send command: "GETR"
- Response code: "0000" (normal)
- Product type: "typ="\$V750-BA50C04-US\$"
- Firmware version: "fwv=102-102-103-0"
- FCS (Horizontal parity): "12"

8. Initialization Method

This document explains the setting procedure from the factory default setting.

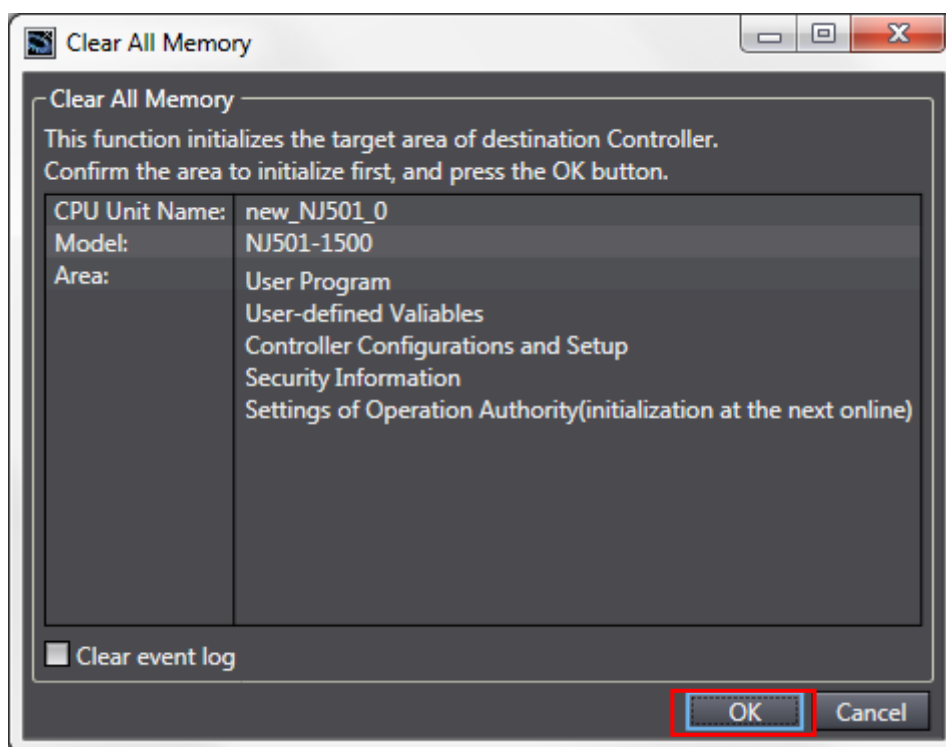
If the device settings are changed from the factory default setting, some settings may not be applicable as described in this procedure.

8.1. Controller

To initialize the Controller, it is necessary to initialize the CPU Unit and Serial Communications Unit.

8.1.1. CPU Unit

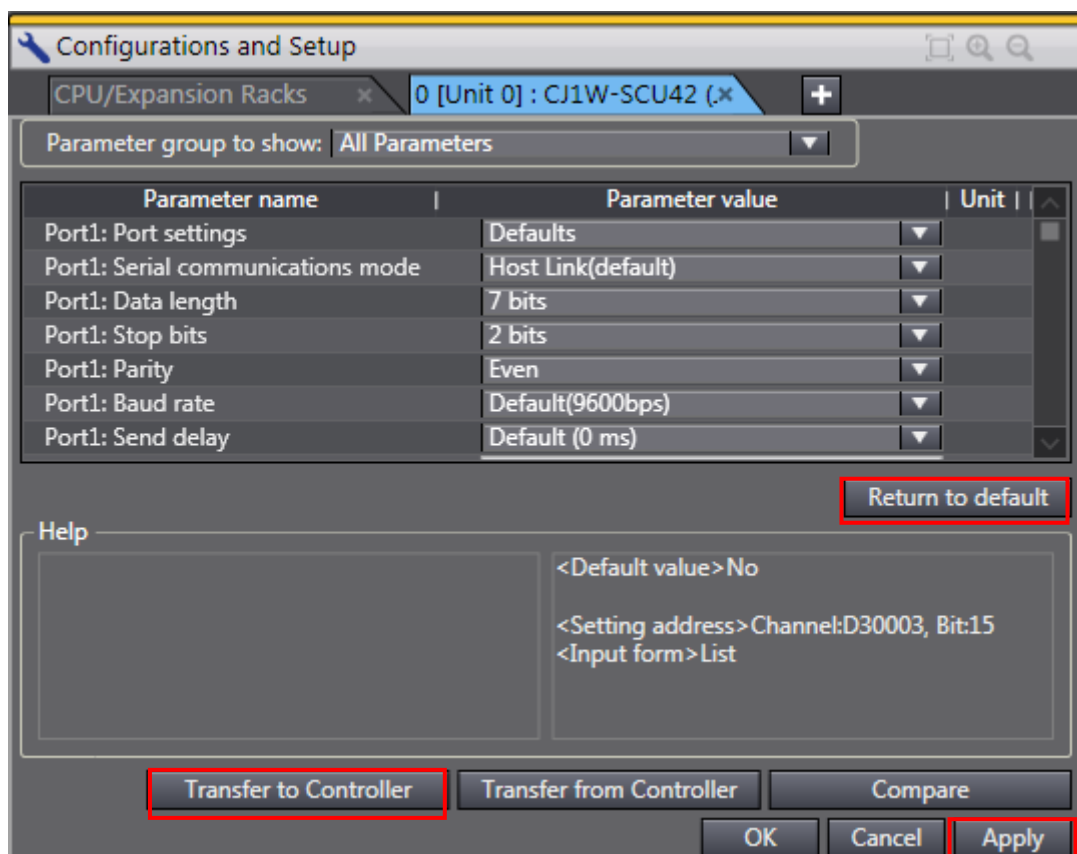
To initialize the settings of the Controller, select **Clear All Memory** from the Controller Menu of the Sysmac Studio.



8.1.2. Serial Communications Unit

To initialize the settings of the Serial Communications Unit, select **Edit Special Unit Settings** of CJ1W-SCU42 in CPU/Expansion Racks from the Sysmac Studio.

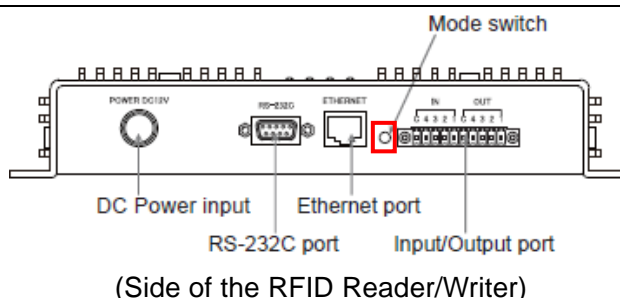
Click the **Return to default** Button and click the **Apply** Button. Then, click the **Transfer to Controller** Button.



8.2. Initializing the RFID Reader/Writer

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

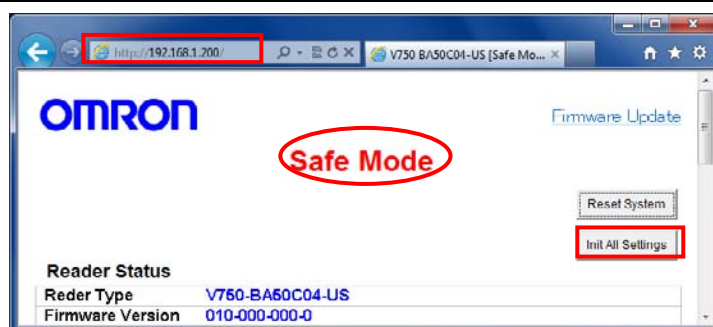
- 1 Press the mode switch at least one second and start the Safe Mode of the RFID Reader/Writer.



- 2 Type "http://192.168.1.200/" in the address bar (Internet Explorer).

The Safe Mode Window is displayed. Click the **Init All Settings** Button.

The RFID Reader/Writer will be initialized and restarted.



*The firmware version in the safe mode is 010-000-000-0.



Additional Information

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

9. Program

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

9.1. Overview

This section explains the specifications and functions of the program used to check the connection between the RFID Reader/Writer (V750 series) (hereinafter referred to as a destination device) to the Controller (Serial Communications Unit) (hereinafter referred to as an SCU Unit).

This program uses the serial communications of the SCU Unit to send/receive “GETR TYP FWV (read the product type and firmware version of the memory data) command” to/from the destination device and to detect a normal end or an error end.

The normal end of this program means a normal end of the serial communications.

The error end means an error end of the serial communications and an error end of the destination device (detected with the response data from the destination device).



Additional Information

OMRON has confirmed that normal communications can be performed using this project file under the OMRON evaluation conditions including the test system configuration, version of each product, and product Lot, No. of each device which was used for evaluation.

OMRON does not guarantee the normal operation under the disturbance such as electrical noise and the performance variation of the device.



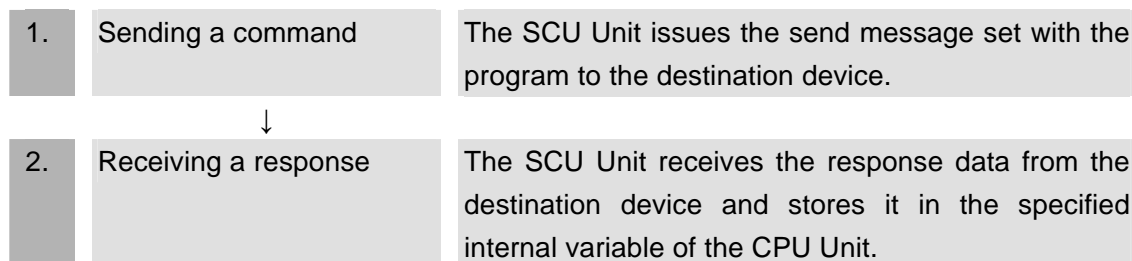
Additional Information

With Sysmac Studio, add the prefix “10#” (possible to omit) to decimal data and the prefix “16#” to hexadecimal data when it is necessary to distinguish between decimal and hexadecimal data. (e.g., “1000” or “10#1000” for decimal data and “16#03E8” for hexadecimal data, etc.)

Also, to specify a specific data type, add the prefix “<data type>#”. (e.g., “UINT#10#1000” and “WORD#16#03E8”, etc.)

9.1.1. Communications Data Flow

The following figure shows the data flow from when the Controller (SCU Unit) issues the serial communications command to the destination device until when the Controller receives the response data from the destination device.



*The response data is not sent after receiving a command or the response data is sent without the need for a command depending on the destination device and command. With this program, the Send/receive processing required/not required setting can be set for the General-purpose serial no-protocol communications sequence setting function block.

If Send only is set, the response data receive processing is not performed. If Receive only is set, the command data send processing is not performed.

9.1.2. Serial Communications Instruction and Send/Receive Message

This section outlines the function blocks for Serial Communications Unit (hereinafter referred to as serial communications instructions) and general operation of the send/receive message.



Additional Information

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

•Serial communications instructions

This program uses the following 2 types of standard instructions to perform serial communications.

| Name | Function block | Description |
|--------------------|----------------|--|
| SCU Send Serial | SerialSend | Sends data in No-protocol Mode from a serial port. (Send instruction) |
| SCU Receive Serial | SerialRcv | Reads the receive data from the serial port in No-protocol Mode. (Receive instruction) |

•Serial communications instructions argument data

•SCU Send Serial

| Instruction | Name | FB/ FUN | Graphic expression | ST expression |
|-------------|-----------------|------------|--------------------|--|
| SerialSend | SCU Send Serial | FB | | SerialSend_instance(Execute, Port, SrcDat, SendSize, Done, Busy, Error, ErrorID, ErrorIDEx); |

Variables

| Name | Meaning | I/O | Description | Valid range | Unit | Default |
|---------------------|------------------|-------|----------------------------------|-----------------------|-------|---------|
| Port | Destination port | Input | Destination port | --- | --- | --- |
| SrcDat[] (array) | Send data array | | Send data array | Depends on data type. | | * |
| SendSize | Send data size | | Data size to send from Src-Dat[] | 0 to 256 | Bytes | 1 |

* If you omit an input parameter, the default value is not applied. A building error will occur.

•SCU Receive Serial

| Instruction | Name | FB/ FUN | Graphic expression | ST expression |
|-------------|-----------------------|------------|--------------------|--|
| SerialRcv | SCU Receive Serial | FB | | SerialRcv_instance(Execute, Port, Size, DstDat, Done, Busy, Error, ErrorID, ErrorIDEx, RcvSize); |

Variables

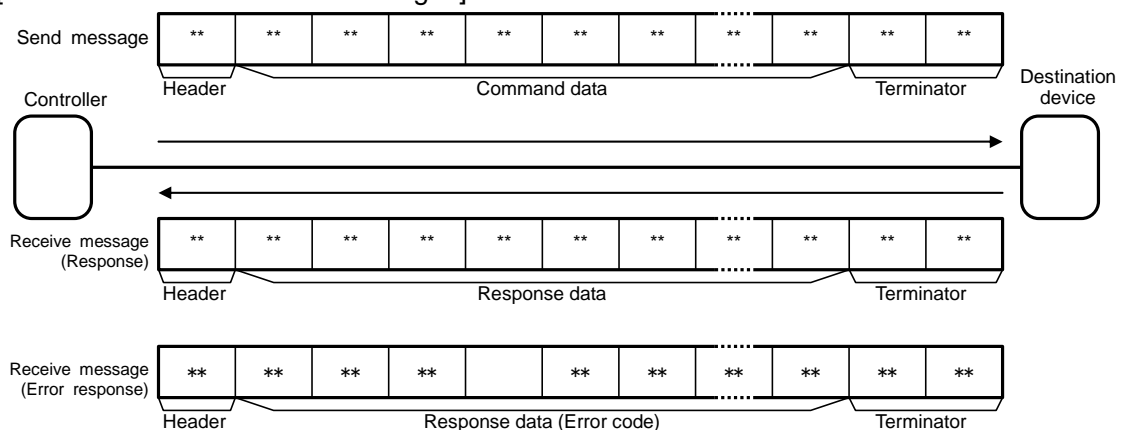
| Name | Meaning | I/O | Description | Valid range | Unit | Default |
|---------------------|---------------------------|--------|--|-----------------------|-------|---------|
| Port | Destination port | Input | Destination port | --- | --- | --- |
| Size | Receive data size | | Size of receive data stored in <i>DstDat[]</i> | 0 to 256 | Bytes | 1 |
| DstDat[] (array) | Receive data array | In-out | Receive data array | Depends on data type. | --- | --- |
| RcvSize | Receive data storage size | Output | Size of receive data that was actually stored in <i>DstDat[]</i> | 0 to 256 | Bytes | --- |

•The data type (_sPORT) of destination port Port

| Name | Meaning | Description | Data type | Valid range | Unit | Default |
|--------------|--------------------|--|-----------|------------------------|------|-----------|
| Port | Destination port | Destination port | _sPORT | --- | --- | --- |
| UnitNo | Unit number | Unit number of Serial Communications Unit | _eUnitNo | _CBU_No00 to _CBU_No15 | --- | _CBU_No00 |
| PhysicPortNo | Serial port number | Serial port number on Serial Communications Unit | USINT | 1 or 2 | | 1 |

•Send/Receive messages

[Overview of send/receive messages]



9.2. Destination Device Command

This section explains the destination device command used in this program.

9.2.1. Overview of the Command

This program uses the GETR TYP FWV command (read the product type and firmware version of memory data) to read information from the destination device.

| Command name | Description |
|--------------|-----------------------------------|
| GETR | Reads the Reader/Writer settings. |



Additional Information

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

9.2.2. Detailed Description of the Command

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

- Send message command format

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

- ASCII codes are sent except for the header and terminator.
- The terminator is automatically added to the send message by the SCU Unit.

| Data | Number of bytes | Remarks |
|--------------------------|------------------|--|
| Header (start code) | 1 | Fixed: SOH (#16#01) |
| Command code | 4 | Fixed: "GETR" (Destination device command) |
| (Space *1) | 1 | Fixed: " " (Space. Parameters and options are separated by a space.) |
| (Parameter or option *1) | 1 and greater *2 | Fixed: "typ" (product type), "fwv" (firmware version) (Option of the "GETR" command) |
| FCS | 2 | The horizontal parity is calculated based on the data after the start code (SOH) through just before the FCS, and the result converted in ASCII is added to the message. |
| Terminator | 2 | Fixed: CR+LF (16#0D0A) |

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

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

- Response format of the receive message

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

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

- The terminator is automatically removed from the receive message by the SCU Unit.

| Data | Number of bytes | Remarks |
|---------------------|-----------------|--|
| Header (start code) | 1 | Fixed: SOH(16#01) |
| Command code | 4 | Fixed: "GETR" (Destination device command) |
| Response code | 4 | Fixed: "0000" (Normal end) |
| (Space *) | 1 | Fixed: " " (Space. Data are separated by a space.) |
| (Response data *) | 1 and greater | Fixed: "typ=\$"[product type V750]\$"" (The product type is enclosed in "\$" and "\$".), "fwv=[firmware version]" (Firmware version) (The information of the options specified with the "GETR" command of this program is returned.) |
| FCS | 2 | The horizontal parity is calculated based on the data after the start code (SOH) through just before the FCS, and the result converted in ASCII is added to the message. |
| Terminator | 2 | Fixed: CR+LF(16#0D0A) |

- Response format of the receive message (error)

This is the response format for an error message received by the Controller from the destination device.

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

- The terminator is automatically removed from the receive message by the SCU Unit.

| Data | Number of bytes | Remarks |
|---------------------|-----------------|--|
| Header (start code) | 1 | Fixed: SOH(16#01) |
| Command code | 4 | Fixed: "GETR" (destination device command) or "ICMD" (undefined command) |
| Response code | 4 | Except for the ICMD code (16#140X (X=0 to 9, A to F)) Destination device error code (Refer to 8.8. Error Code List.) |
| FCS | 2 | The horizontal parity is calculated based on the data after the start code (SOH) through just before the FCS, and the result converted in ASCII is added to the message. |
| Terminator | 2 | Fixed: CR+LF (16#0D0A) |



Additional Information

For details on the error codes, refer to *Section 5 Command Line Interface* in the *V750-series UHF RFID System User's Manual*.

9.2.3. Command Settings

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

- Send data (command) settings

Set the send data in Serial_SendMessageSet function block.

<Specifications of the destination device>

- Data are stored in ASCII codes.

| Variable | Contents (Data type) | Set value |
|----------------|-----------------------------|------------------------------|
| Send_Header | Send header (STRING[5]) | " |
| Send_Addr | Send address (STRING[5]) | " (Setting unnecessary) |
| Send_Command | Send data (STRING[256]) | CONCAT('GETR',' typ',' fwv') |
| Send_Check | Add send check (STRING[5]) | StringLRC(Send_Command) |
| Send_Terminate | Send terminator (STRING[5]) | " |

*The header (SOH) and terminator (CR+LF) are added by the SCU Unit to the send message. Therefore, do not set any header or terminator.

| Variable | Contents (Data type) | Data | Description |
|-----------|----------------------------|--|--|
| Send_Data | Send message (STRING[256]) | CONCAT(Send_Header, Send_Addr, Send_Command, Send_Check, Send_Terminate) | Used as send data of SerialSend instruction (SerialSend_instance). |

[Calculation method of Send_Check (FCS)]

The FCS is the result of the horizontal parity calculation of the data right after the send header through just before the FCS. The lower 1 byte is converted into ASCII codes. (This project file uses StringLRC instruction (Function).)

- Calculation method

The StringLRC instruction is used to calculate the FCS value (sum of the character code of each character) based on the target strings right after the send header to the send data and to output two-byte ASCII characters.

Send_Check := StringLRC(Send_Command);



Additional Information

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

- Receive data (response) that is stored

The receive data is stored and checked by the Serial_ReceiveCheck function block.

<Specifications of the destination device>

- The response is stored in ASCII codes.

| Variable | Contents (Data type) | Storage area |
|-----------|-------------------------------|--|
| Recv_Data | Receive data (STRING[256]) | Receive buffer |
| Recv_Buff | Receive data (STRING[256]) | Receive data storage area (stores the receive buffer data) |

- Send/receive message

*Send message

| | | | | | | | | | | | | | | | |
|--------|---------|-----|-----|-----|------------------|-----|-----|-----|-----|-----|-----|-----|------------|------|------|
| 01 | 47 | 45 | 54 | 52 | 20 | 74 | 79 | 70 | 20 | 66 | 77 | 76 | **** | 0D | 0A |
| [SOH] | 'G' | 'E' | 'T' | 'R' | ' ' | 't' | 'y' | 'p' | ' ' | 'f' | 'w' | 'v' | FCS | [CR] | [LF] |
| Header | Command | | | | Data (Parameter) | | | | | | | | Terminator | | |

*Receive message 1 (at normal process)

| | | | | | | | | | | | | | | |
|--------|---------|-----|-----|-----|---------------|----|----|----|-----|-----|-----|-----|-----|-----|
| 01 | 47 | 45 | 54 | 52 | 30 | 30 | 30 | 30 | 20 | 74 | 79 | 70 | 3D | 22 |
| [SOH] | 'G' | 'E' | 'T' | 'R' | '0000' | | | | ' ' | 't' | 'y' | 'p' | '=' | ''' |
| Header | Command | | | | Response code | | | | | | | | | |

| | | | | | | | | | | |
|------------------|-----|-----|-----|-----|-----|-----|------------|------|------|------|
| **** | 22 | 20 | 66 | 77 | 76 | 3D | **** | **** | 0D | 0A |
| Product type | ''' | ' ' | 'f' | 'w' | 'v' | '=' | Version | FCS | [CR] | [LF] |
| Data (parameter) | | | | | | | Terminator | | | |

*Receive message 2 (at error process)

| | | | | | | | | |
|--------|---------|-----|-----|-----|---------------|------|------------|------|
| 01 | 47 | 45 | 54 | 52 | **** | **** | 0D | 0A |
| [SOH] | 'G' | 'E' | 'T' | 'R' | Response code | FCS | [CR] | [LF] |
| Header | Command | | | | | | Terminator | |

*Receive message 3 (at error process: Undefined)

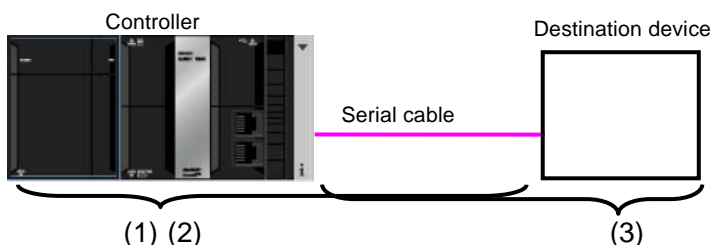
| | | | | | | | | |
|--------|---------|-----|-----|-----|---------------|------|------------|------|
| 01 | 49 | 43 | 4D | 44 | **** | **** | 0D | 0A |
| [SOH] | 'I' | 'C' | 'M' | 'D' | Response code | FCS | [CR] | [LF] |
| Header | Command | | | | | | Terminator | |

9.3. Error Detection Processing

This section explains the error detection processing of this program.

9.3.1. Error Detection in the Program

This program detects and handles errors of the following items (1) to (3). For error codes, refer to 9.7 *Error Processing*.



- (1)Errors at execution of the serial communications instruction (serial communications instruction errors)

Errors in the Unit, command format, or parameters at the execution of the SerialSend or SendCmd instruction are detected as "serial communications instruction errors". The error is detected with the error codes *ErrorID* and *ErrorIDEx* of the serial communications instruction. If the "Serial communications instruction error" is caused by a transmission error due to, for example, a character corruption or unmatched baud rate setting, the transmission error status (*J01_P2_TransErrSta*) device variable of the SCU Unit is stored in the output variable.

- (2) Timeout errors at execution of the program (Timeout errors)

When the send processing and receive processing are not normally performed and cannot be completed within the monitoring time, it is detected as a timeout error. The error is detected with the timer monitoring function in the program. For information on the time monitoring function of the timer in the program, refer to 9.3.2. *Time Monitoring Function*.

- (3)Errors in the destination device (Destination device errors)

The destination device errors include a command error, a parameter error, and an execution failure in the destination device. An error is detected with the response data which is returned from the destination device. For information on the send/receive messages, refer to 9.2. *Destination Device Command*.

Receive message at normal process

| | | | | | |
|------------|--------------|---------------|---------------|-----|------------|
| SOH | 'GETR' | '0000' | *•••* | ** | 16#0D0A |
| Start code | Command code | Response code | Response data | FCS | Terminator |

Receive message at error process

| | | | | |
|------------|--------------|---------------|-----|------------|
| SOH | 'GETR' | **** | ** | 16#0D0A |
| Start code | Command code | Response code | FCS | Terminator |

Receive message at undefined command error process

| | | | | |
|---------------|-----------------|------------------|-----|------------|
| SOH | 'ICMD' | **** | ** | 16#0D0A |
| Start code | Command code | Response code | FCS | Terminator |

9.3.2. Time Monitoring Function

This section explains the time monitoring function of this program.

- Time monitoring function using the timer in the program

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

[Time monitoring function of the timer in the program]

| Processing | Monitoring | Timeout value |
|--------------------|--|----------------------------|
| Send processing | Send processing monitoring time: Time from when the program waits for the send processing to be allowed until when the send processing ends. *An operation end of the SerialSend instruction means the end of the processing. | After 5 seconds (Default) |
| Receive processing | Receive processing start time: Time from the start to the end of the receive processing. *When the receive processing is repeated, the program monitors each receive processing separately. | After 5 seconds (Default) |
| Receive wait | Receive wait monitoring time: Receive waiting time between responses *The receive waiting time for the next response after the receive processing ends once is also set in the TrTime variable as the receive waiting time monitoring timer. If the next response does not arrive from the destination device within this time, it is detected that the receive processing ended. | After 0.3 second (Default) |

9.4. Variables

The variables used in this program are listed below.

9.4.1. List of Variables

The following tables list the data types, external variables (user-defined global variables/device variable for CJ-series Unit/system-defined variable) and internal variables that are used in this program.

•Data type (Structure)

[Communications processing status flags]

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

[Communications instruction execution flags]

| Name | Data type | Description |
|----------|-----------|--|
| sControl | STRUCT | Serial communications instruction execution flags |
| Send | BOOL | Send processing instruction TRUE: Executed / FALSE: Not executed |
| Recv | BOOL | Receive processing instruction TRUE: Executed / FALSE: Not executed |

[Timer enable flags]

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

[Send/receive processing required/not required setting flags]

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

•Data type (Union)

[Error code processing]

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

•External variables

[User-defined global variables]

| Variable name | Data type | Description |
|----------------------|-------------|---|
| Input_Start | BOOL | Communication start switch The program is started when this variable changes from FALSE to TRUE. |
| Output_RecvMess | STRING[256] | An area that stores the receive data (response) (256 bytes) |
| Output_ErrCode | WORD | An area that stores the error flag for a communications error or a timeout error that is detected at the send processing and receive processing. Normal end: 16#0000 |
| Output_CmdsErrorID | WORD | An area that stores the error code for an error that is detected at the send processing and receive processing. Normal end: 16#0000 |
| Output_CmdsErrorIDEx | DWORD | An area that stores the expansion error code for an error that is detected at the send processing and receive processing. Normal end: 16#00000000 |
| Output_TransErrCode | WORD | An area that stores the transmission error status (J01_P2_TransErrSta) at a communications error. Normal end: 16#0000 |
| Output_MErrCode | DWORD | An area that stores the destination device error code for a destination device error. Normal end: 16#00000000 |
| Output_ReceiveLength | INT | An area that stores the receive data size |

[Device variable for CJ-series Unit] (Serial Communications Unit)

| Variable name | Data type | Description |
|-----------------------------|-----------|--------------------------------|
| J01_P2_NopSerialSendExecSta | BOOL | Send processing executing flag |
| J01_P2_TransErr | BOOL | Transmission error |
| J01_P2_TransErrSta | BOOL | Transmission error status |
| J01_P2_NopRcvCompleteSta | BOOL | Receive completion |



Additional Information

For details on variables of the Serial Communications Unit, refer to *2-3 Device Variable for CJ-series Unit* in the *CJ-series Serial Communications Units Operation Manual for NJ-series CPU Unit* (Cat.No. W494).

[System-defined variable]

| Name | Data type | Description |
|-------------------|-----------|---|
| _Port_isAvailable | BOOL | Communications Port Enabled Flag TRUE: Enabled, FALSE: Not enabled |



Additional Information

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

•Internal variables (Instance variables)

The following tables list the internal variables used to execute the function blocks in the program. An internal variable is called an “instance”. The name of the function block to use is specified as the data type of the variable.

[Instances of user-defined function blocks]

| Variable name | Data type | Description |
|--------------------------------|----------------|---|
| Serial_ParameterSet_instance | ParameterSet | No-protocol serial communications parameter setting function block This variable sets the monitoring time of each processing from the send processing to the receive processing. |
| Serial_SendMessageSet_instance | SendMessageSet | No-protocol serial communications send data setting function block This variable sets the send/receive processing required/not required setting and sets a send message. |
| Serial_ReceiveCheck_instance | ReceiveCheck | No-protocol serial communications receive processing function block This variable stores the receive data and detects a normal end or an error end. |

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

[Instances of timers]

| Variable name | Data type | Description |
|------------------|-----------|--|
| Tfs_TON_instance | TON | Send processing monitoring timer This variable counts the time taken to perform the send processing. |
| Tfr_TON_instance | TON | Receive processing monitoring timer This variable counts the time taken to perform the receive processing. |
| Tr_TON_instance | TON | Receive wait monitoring timer This variable counts the time taken to wait for the receive data from the destination device. |

[Instances of communications instructions]

| Variable name | Data type | Description |
|---------------------|------------|--|
| SerialSend_instance | SerialSend | SCU send serial (no-protocol send processing) function block |
| SerialRcv_instance | SerialRcv | SCU receive serial (no-protocol receive processing) function block |



Additional Information

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

•Internal variables

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

9.5. ST Program

9.5.1. Functional Components of Program

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

| Major classification | Minor classification | Description |
|------------------------------------|---|--|
| 1. Communications processing | 1.1. Starting communications processing 1.2. Clearing the communications processing status flags 1.3. Communications processing in progress status | The communications processing is started. |
| 2. Initialization processing | 2.1. Initializing the timers 2.2. Initializing the instructions 2.3. Initializing the instruction execution flags 2.4. Initializing the timer enable flags 2.5. Initializing the error code storage areas 2.6. Setting each processing monitoring time and setting the communications parameters 2.7. Setting the send/receive processing required/not required setting and send data 2.8. Converting send data from a string to a BYTE array 2.9. Initializing the receive data storage areas 2.10. Initialization setting end processing | The parameters for serial communications are set and the error code storage areas are initialized. The send/receive required/not required setting is set and the send data and receive data are set. |
| 3. Send processing | 3.1. Determining the send processing status and setting the execution flag 3.2. Enabling the send processing time monitoring timer 3.3. Executing the send instruction | The processing is started when the send processing required/not required setting is set to Required and the initialization processing ends normally. |
| 4. Receive processing | 4.1. Determining the receive processing status and setting the execution flag 4.2. Enabling the receive waiting time monitoring timer 4.3. Enabling the receive processing time monitoring timer 4.4. Executing the receive instruction 4.5. Executing the destination device error detection instruction | The processing is started when the receive processing required/not required setting is set to Required and the send processing ends normally. When multiple receive data arrive, the receive processing is repeated. The receive data is stored and checked. |
| 5. Processing number error process | 5. Processing number error process | The error process is performed when a non-existent status processing number is detected. |

9.5.2. Program List

The program is shown below.

The communications setting and send data (command data) setting, which need to be changed depending on the destination device, are set in the function blocks (ParameterSet, SendMessageSet, and ReceiveCheck). For information on how to change these values, refer to *9.5.3 Detailed Description of Function Blocks*.

- Program: Program0 (General-purpose serial communications connection check program)

1. Communications processing

```
(* =====
  Name: NJ-series general-purpose serial no-protocol (RS-232C) communications
        connection check program
  Serial Unit: CJ1W-SCU42 (No-protocol, Unit number: 0, Serial port number: 2)
  Version information: V1.00 December 7, 2012 New release
  (C)Copyright OMRON Corporation 2012 All Rights Reserved.
  ===== *)

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

(* 1.1. Starting the communications processing
  Start communications processing when the communications start switch changes to ON
  when communications processing status flags have been cleared. *)
IF Input_Start AND
  NOT (Local_Status.Busy OR Local_Status.Done OR Local_Status.Error) THEN
  Local_Status.Busy:=TRUE;
  Local_State:=10; //To 10: Initialization processing
END_IF;

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

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


2. Initialization processing

```

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

    (* 2.2. Initializing the communications instructions *)
    SerialSend_instance(Execute:=FALSE,SrcDat:=Local_SrcData[0]);
    SerialRcv_instance(Execute:=FALSE,DstDat:=Local_RecvData[0]);

    (* 2.3. Initializing the communications instruction execution flags *)
    Local_ExecFlgs.Send:=FALSE;
    Local_ExecFlgs.Recv:=FALSE;

    (* 2.4. Initializing the processing time monitoring timer enable flags *)
    Local_TONflgs.Tfs:=FALSE;
    Local_TONflgs.Tfr:=FALSE;
    Local_TONflgs.Tr:=FALSE;

    (* 2.5. Initializing the error code storage areas *)
    Local_ErrCode.WordData:=WORD#16#0000;
    Output_ErrCode:=WORD#16#0000;
    Output_TransErrCode:=WORD#16#0000;
    Output_MErrCode:=DWORD#16#FFFFFFF;
    Output_CmdsErrorID:=WORD#16#FFFF;
    Output_CmdsErrorIDEx:=DWORD#16#FFFFFFF;

    (* 2.6. Setting each processing monitoring time and
        setting the general-purpose serial no-protocol-related parameters *)
    (* Set each processing monitoring time *)
    Serial_ParameterSet_instance(Execute:=TRUE);
    (* Set the port for communications instructions *)
    Local_Port.UnitNo:=_CBU_No00;
    Local_Port.PhysicPortNo:=USINT#2;

    (* 2.7. Setting the send/receive processing required/not required setting
        and setting the send data *)
    Serial_SendMessageSet_instance(Execute:=TRUE);
    (* Detect a setting error in the send/receive processing required/not required setting. *)
    Local_ComType.Send:=TestABit(Serial_SendMessageSet_instance.ComType,0);
    Local_ComType.Recv:=TestABit(Serial_SendMessageSet_instance.ComType,1);
    Local_ComType.Error:=NOT(Local_ComType.Send OR Local_ComType.Recv);
    IF Local_ComType.Error THEN
        Output_ErrCode:=WORD#16#0020;
        Local_InitialSettingOK:=FALSE;
    ELSE
        Local_InitialSettingOK:=TRUE;
    END_IF;

```

```

(* 2.8. Converting the send data from STRING to BYTE array *)
Local_SrcDataByte:=
  StringToAry(Serial_SendMessageSet_instance.Send_Data,Local_SrcData[0]);

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

(* 2.10. Initialization setting end processing
  Determine the next status
      based on the send/receive processing required/not required flag *)
IF NOT Local_InitialSettingOK THEN
  Local_Status.Busy:=FALSE;
  Local_Status.Error:=TRUE;
  Local_State:=0; //To 0: Communications not in progress status
ELSIF Local_ComType.Send THEN
  Local_State:=11; //To 11: Send processing
ELSIF Local_ComType.Recv THEN
  Local_State:=12; //To 12: Receive processing
END_IF;

```

3. Send processing

```

(* 3. Send processing
-Send data from the specified serial port *)
11:
  (* 3.1. Determining the send processing status and setting the execution flag *)
  (* 3.1.1. Timeout processing *)
  IF Tfs_TON_instance.Q THEN
    Local_ErrCode.BoolData[8]:=TRUE;
    Output_CmdsErrorID:=WORD#16#FFFF;
    Output_CmdsErrorIDEx:=DWORD#16#FFFFFFFF;
    Local_ExecFlgs.Send:=FALSE;
    Local_TONflgs.Tfs:=FALSE;
    Output_TransErrCode:=
      SEL(J01_P2_TransErr,WORD#16#0000,J01_P2_TransErrSta);

    (* Error end processing *)
    Local_ErrCode.BoolData[15]:=TRUE;
    Output_ErrCode:=Local_ErrCode.WordData;
    Local_Status.Busy:=FALSE;
    Local_Status.Error:=TRUE;
    Local_State:=0; //To 0: Communications not in progress status

  (* 3.1.2. Normal end processing *)
  ELSIF SerialSend_instance.Done AND NOT (J01_P2_NopSerialSendExecSta) THEN
    Local_ErrCode.BoolData[0]:=FALSE;
    Output_CmdsErrorID:=WORD#16#0000;
    Output_CmdsErrorIDEx:=WORD#16#00000000;
    Local_ExecFlgs.Send:=FALSE;
    Local_TONflgs.Tfs:=FALSE;
    Output_TransErrCode:=WORD#16#0000;
    Output_ErrCode:=Local_ErrCode.WordData;

    (* Determine the next status
      based on the send/receive processing required/not required flag *)
    IF Local_ComType.Recv THEN
      Local_State:=12; //To 12: Receive processing
    ELSE
      Local_Status.Busy:=FALSE;
      Local_Status.Done:=TRUE;
      Local_State:=0; //To 0: Communications not in progress status
    END_IF;
  
```

```

(* 3.1.3. Send error end processing *)
ELSIF SerialSend_instance.Error THEN
    Local_ErrCode.BoolData[0]:=TRUE;
    Output_CmdsErrorID:=SerialSend_instance.ErrorID;
    Output_CmdsErrorIDEx:=SerialSend_instance.ErrorIDEx;
    Local_ExecFlgs.Send:=FALSE;
    Local_TONflgs.Tfs:=FALSE;
    Output_TransErrCode:=
        SEL(J01_P2_TransErr,WORD#16#0000,J01_P2_TransErrSta);

    (* Error end processing *)
    Local_ErrCode.BoolData[15]:=TRUE;
    Output_ErrCode:=Local_ErrCode.WordData;
    Local_Status.Busy:=FALSE;
    Local_Status.Error:=TRUE;
    Local_State:=0; //To 0: Communications not in progress status

(* 3.1.4. Setting the send instruction execution flag *)
ELSIF _Port_isAvailable AND NOT (SerialSend_instance.Busy)
    AND NOT(J01_P2_NopSerialSendExecSta) THEN
    Local_ExecFlgs.Send:=TRUE;

(* 3.1.5. Setting the send processing timer enable flag *)
ELSE
    Local_TONFlgs.Tfs:=TRUE;
END_IF;

(* 3.2. Enabling the send processing time monitoring timer *)
Tfs_TON_instance(
    In:= Local_TONFlgs.Tfs,
    PT:=MULTIME(T#10ms,Serial_ParameterSet_instance.TfsTime));

(* 3.3. Executing the send instruction *)
SerialSend_instance(
    Execute:=Local_ExecFlgs.Send,
    Port:=Local_Port ,
    SrcDat:=Local_SrcData[0] ,
    SendSize:=Local_SrcDataByte);

```

4. Receive processing

```

(* 4. Receive processing
-Read the data from the receive buffer of the specified serial port *)
12:
  (* 4.1. Determining the receive processing status and setting the execution flag *)
  (* 4.1.1. Receive end processing *)
  IF Tr_TON_instance.Q THEN
    Local_TONFlgs.Tfr:=FALSE;
    Local_TONFlgs.Tr:=FALSE;
    Local_ErrCode.BoolData[1]:= FALSE;
    Output_CmdsErrorID:=WORD#16#0000;
    Output_CmdsErrorIDEx:=DWORD#16#00000000;

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

    (* Normal end processing
    Setting the communications processing end flag by error code detection *)
    Local_Status.Busy:=FALSE;

    (* Communications processing normal end *)
    IF (Local_ErrCode.WordData = WORD#16#0000)
      AND NOT(J01_P2_TransErr) THEN
      Output_TransErrCode:=WORD#16#0000;
      Local_RecvCheckFlg:=TRUE;

    (* Communications processing error end *)
    ELSE
      Local_Status.Error:=TRUE;
      Output_TransErrCode:=J01_P2_TransErrSta;
      Local_ErrCode.BoolData[15]:=TRUE;
      Output_ErrCode:=Local_ErrCode.WordData;
    END_IF;
    Local_State:=0; //To 0: Communications not in progress status

  (* 4.1.2. Timeout processing *)
  ELSIF Tfr_TON_instance.Q THEN
    Local_ErrCode.BoolData[9]:=TRUE;
    Output_CmdsErrorID:=WORD#16#FFFF;
    Output_CmdsErrorIDEx:=DWORD#16#FFFFFFFF;
    Local_ExecFlgs.Recv:=FALSE;
    Local_TONFlgs.Tfr:=FALSE;
    Local_TONFlgs.Tr:=FALSE;
    Output_TransErrCode:=
      SEL(J01_P2_TransErr,WORD#16#0000,J01_P2_TransErrSta);

    (* Error end processing *)
    Local_ErrCode.BoolData[15]:=TRUE;
    Output_ErrCode:=Local_ErrCode.WordData;
    Local_Status.Busy:=FALSE;
    Local_Status.Error:=TRUE;
    Local_State:=0; //To 0: Communications not in progress status

```

```

(* 4.1.3. Normal end processing *)
ELSIF SerialRcv_instance.Done THEN
    Local_RecvDataLength:=
        Local_RecvDataLength+SerialRcv_instance.RcvSize;
    Local_RecvCHNo:=Local_RecvDataLength;
    Local_TONFlgs.Tfr:=FALSE;
    Local_ExecFlgs.Recv:=FALSE;
    Local_TONFlgs.Tr:=TRUE; // To 4.1.5. Reading the receive data

(* 4.1.4. Error end processing *)
ELSIF SerialRcv_instance.Error THEN
    Local_ErrCode.BoolData[1]:=TRUE;
    Output_CmdsErrorID:=SerialRcv_instance.ErrorID;
    Output_CmdsErrorIDEx:=SerialRcv_instance.ErrorIDEx;
    Local_ExecFlgs.Recv:=FALSE;
    Local_TONFlgs.Tfr:=FALSE;
    Local_TONFlgs.Tr:=FALSE;
    Output_TransErrCode:=
        SEL(J01_P2_TransErr,WORD#16#0000,J01_P2_TransErrSta);

    (* Error end processing *)
    Local_ErrCode.BoolData[15]:=TRUE;
    Output_ErrCode:=Local_ErrCode.WordData;
    Local_Status.Busy:=FALSE;
    Local_Status.Error:=TRUE;
    Local_State:=0; //To 0: Communications not in progress status

(* 4.1.5. Reading the receive data
    When there is data to read: Receive processing continues. *)
ELSIF J01_P2_NopRcvCompleteSta THEN
    IF _Port_isAvailable AND NOT SerialRcv_instance.Busy THEN
        Local_ExecFlgs.Recv:=TRUE;
        Local_TONFlgs.Tfr:=TRUE;
        Local_TONFlgs.Tr:=FALSE;
    END_IF;
    (* When there is no data to read:
        -When no data is received, no processing is performed.
        -When data is already received, the waiting time to receive the response is monitored,
        and if there is no more response, the receive processing is ended
        after reading the data that was already received. *)

(* 4.1.6. Setting the timer enable flag *)
ELSE
    Local_TONFlgs.Tfr:=TRUE;
    (* Initialize the destination device error detection instruction execution flag *)
    Local_RecvCheckFlg:=FALSE;
END_IF;

```

```

(* 4.2. Enabling the receive waiting time monitoring timer *)
Tr_TON_instance(
  In:= Local_TONFlgs.Tr,
  PT:=MULTIME(T#100ms,Serial_ParameterSet_instance.TrTime));

(* 4.3. Enabling the receive processing time monitoring timer *)
Tfr_TON_instance(
  In:= Local_TONFlgs.Tfr,
  PT:=MULTIME(T#10ms,Serial_ParameterSet_instance.TfrTime));

(* 4.4. Executing the receive instruction *)
SerialRcv_instance(
  Execute:=Local_ExecFlgs.Rcv ,
  Port:=Local_Port ,
  Size:=Local_ReceiveSize,
  DstDat:=Local_RecvData[Local_RecvCHNo]);

(* 4.5. Executing the destination device error detection instruction *)
Serial_ReceiveCheck_instance(
  Execute:=Local_RecvCheckFlg,
  Recv_Buff:=Local_ReceiveMessage,
  Recv_Data:=Output_RecvMess,
  tLength:= Local_RecvDataLength,
  Done:=Local_Status.Done,
  Error:=Local_Status.Error,
  ErrorID:=Output_ErrCode,
  ErrorIDEx:=Output_MErrCode);

```

5. Processing number error process

```

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

```

9.5.3. Detailed Description of Function Block

The user-defined function blocks are shown below.

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

●ParameterSet function block

(General-purpose serial no-protocol communications parameter setting)

| Instruction | Meaning | ST expression |
|--------------|---|---|
| ParameterSet | General-purpose serial no-protocol communications parameter setting | Serial_ParameterSet_instance (Execute, TfsTime, TrTime, TfrTime); |

[Internal variables]

None

[Input/output]

| Name | I/O | Data type | Description |
|-----------|--------|-----------|---|
| Execute | Input | BOOL | Execution flag: The function block is executed when this variable changes to TRUE and it is stopped when this variable changes to FALSE. |
| TfsTime | Output | UINT | Send processing monitoring time: This variable sets the monitoring time of the send processing in increments of 10 ms. |
| TrTime | Output | UINT | Receive wait monitoring time: This variable sets the waiting time for the receive data in increments of 100 ms. |
| TfrTime | Output | UINT | Receive processing monitoring time: This variable sets the monitoring time of the receive processing in increments of 10 ms. |
| Busy | Output | BOOL | Busy |
| Done | Output | BOOL | Normal end |
| Error | Output | BOOL | Error end |
| ErrorID | Output | WORD | Error information |
| ErrorIDEx | Output | DWORD | Error information |

Not used
(Not used in this program.)

[External variables]

None

[Program]

```

(* =====
   Name: NJ-series general-purpose serial no-protocol communications
         parameter setting function block
   Applicable device: OMRON Corporation V750-series RFID system
   Version: V1.00 New release December 7, 2012
   (C)Copyright OMRON Corporation 2012 All Rights Reserved.
   ===== *)

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

  (* Maximum waiting time of response data for when a message, which is
     divided into multiple packets, is received. *)
  TrTime:= UINT#3; // Receive wait monitoring time: Setting unit 100ms<3->300ms>

END_IF;

RETURN;

```

•SendMessageSet function block

(General-purpose serial no-protocol communications sequence setting)

| Instruction | Meaning | ST expression |
|----------------|--|--|
| SendMessageSet | General-purpose serial no-protocol communications sequence setting | Serial_ParameterSet_instance(Execute, Send_Data, ComType); |

[Internal variables]

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

[Input/Output]

| Name | I/O | Data type | Description |
|-----------|--------|-------------|---|
| Execute | Input | BOOL | Execution: The function block is executed when this variable changes to TRUE and it is stopped when this variable changes to FALSE. |
| Send_Data | Output | STRING[256] | Send data: This variable sets a command that is sent to the destination device. |
| ComType | Output | BYTE | Send/receive type: This variable sets whether send/receive processing are required. 1: Send only, 2: Receive only, 3: Send and receive |
| Busy | Output | BOOL | Busy |
| Done | Output | BOOL | Normal end |
| Error | Output | BOOL | Error end |
| ErrorID | Output | WORD | Error code |
| ErrorIDEx | Output | DWORD | Expansion error code |

Not used
(Not used in this project.)

[Internal variable]

None

[Program]

```

(* =====
   Name: NJ-series general-purpose serial no-protocol communications
         sequence setting function block
   Applicable device: OMRON Corporation V750-series RFID system
   Version: V1.00 New release December 7, 2012
   (C)Copyright OMRON Corporation 2012 All Rights Reserved.
   ===== *)

IF Execute THEN

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

  (* Set the send data *)
  Send_Header:="";           // Header: 'None'
                             // Presence or absence of SCU Unit start code
                             // Set SOH(0x01)
  Send_Addr:="";             // Address (station number)
  Send_Command:=CONCAT('GETR',' typ',' fvv'); // Destination device command: GETR
  Send_Check:=StringLRC(Send_Command);
                             // FCS calculation: Horizontal parity
  Send_Terminate:="";        // Terminator: 'None'
                             // Presence or absence of SCU Unit end code
                             // Set CR+LF(0x0D0A)

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

END_IF;

RETURN;

```

●ReceiveCheck function block

(General-purpose serial no-protocol communications receive processing)

| Instruction | Meaning | ST expression |
|--------------|--|---|
| ReceiveCheck | General-purpose serial no-protocol communications receive processing | Serial_ReceiveCheck_instance(Execute, Recv_Data, Recv_Buff, Done, Error, ErrorID, ErrorIDEx); |

[Internal variables]

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

[Input/Output]

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

[External variable]

None

[Program]

```

(* =====
Name: NJ-series general-purpose serial no-protocol communications
    receive processing function block
Applicable device: OMRON Corporation V750-series RFID system
Version: V1.00 New release December 7, 2012
(C)Copyright OMRON Corporation 2012 All Rights Reserved.
===== *)

IF Execute THEN

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

    (* Detect the destination device error
    Normal: 5th to 8th bytes from the start of the data is '0000' *)
    IF EQascii(MID(Recv_Buff, UINT#4, UINT#5), '0000') THEN

        (* Detection of FCS
        Obtain FCS of the receive data (in buffer).
        Extract 2 characters from the right. *)
        Receive_Check:= RIGHT(Recv_Buff, USINT#2);

        (* FCS calculation of the receive data (in buffer) *)
        (* Data from the start of the receive data through just before "horizontal parity+CR+LF"
        Calc_Check:=stringLRC(LEFT(Recv_Buff, tLength-UINT#2));

        (* Comparing the obtained value to calculated FCS *)
        IF EQascii(Receive_Check, Calc_Check) THEN
            (* Normal end *)
            Done:= TRUE;                // Set the normal end flag
            Error:= FALSE;              // Set/reset the error flag
            ErrorID:= WORD#16#0000;     // Clear the error code
            ErrorIDEx:= DWORD#16#00000000; // Clear the destination device error code
        ELSE
            (* FCS value error end *)
            Done:= FALSE;              // Reset normal end flag
            Error:= TRUE;              // Set the error flag
            ErrorID:= WORD#16#2000;     // Set the error code
            ErrorIDEx:= STRING_TO_DWORD (Receive_Check);
        END_IF;

        (* Error: 5th to 8th bytes from the start of the data is not '0000' *)
        ELSE
            Done:= FALSE;              // Set the normal end flag
            Error:= TRUE;              // Set the error flag
            ErrorID:= WORD#16#1000;     // Set the error code
            (* Store the destination device error code
            Convert 5th to 8 characters from the left of the string from ASCII code to hexadecimal *)
            ErrorIDEx:= STRING_TO_DWORD ((MID(Recv_Buff, UINT#4, UINT#5)));

        END_IF;
    END_IF;

RETURN;

```

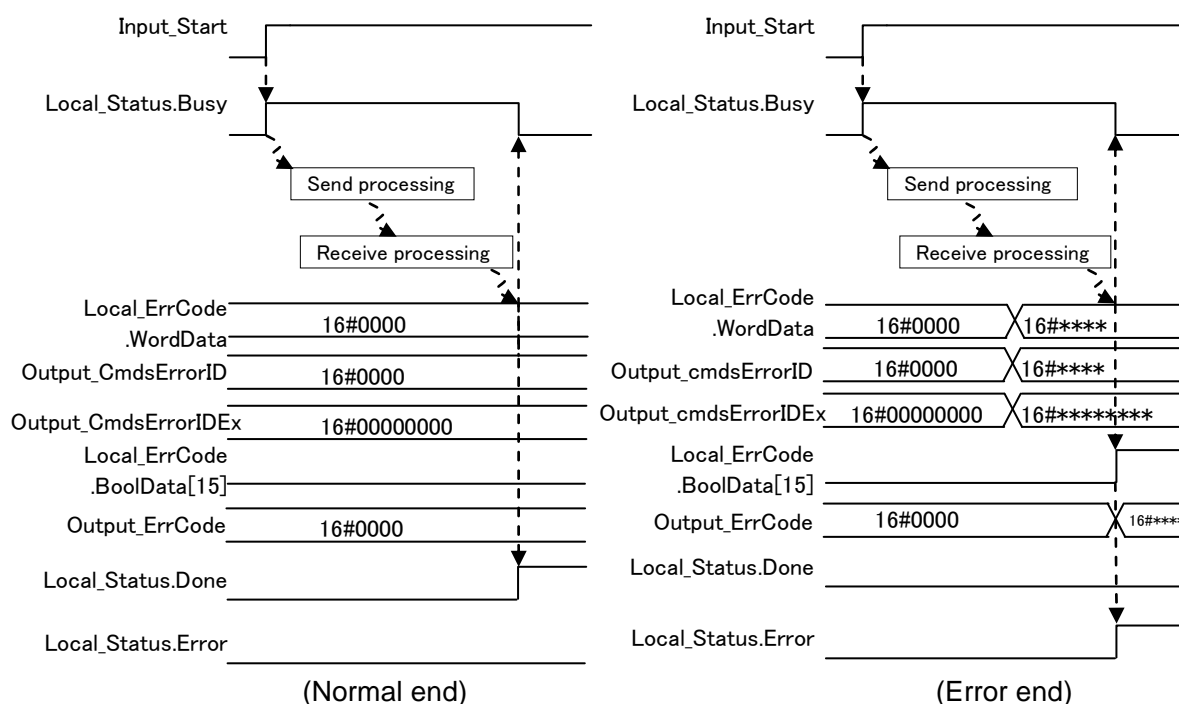
9.6. Timing Charts

This section explains the timing charts of the program.

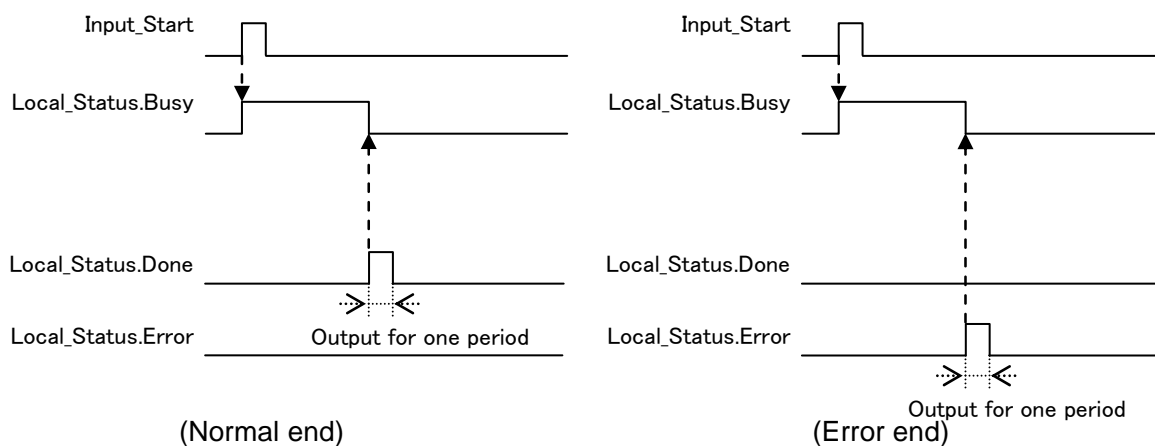
The definitions of the timing chart patterns are as follows:

| Pattern | Normal end | Error end (1) Serial communications instruction error | Error end (2) Timeout error | Error end (3) Destination device error |
|--------------------|------------|---|--------------------------------|--|
| Command | Normal | Error | Normal | Normal |
| Destination device | Normal | Normal or error | Normal or error | Error |
| Response | Yes | None | None | Yes |

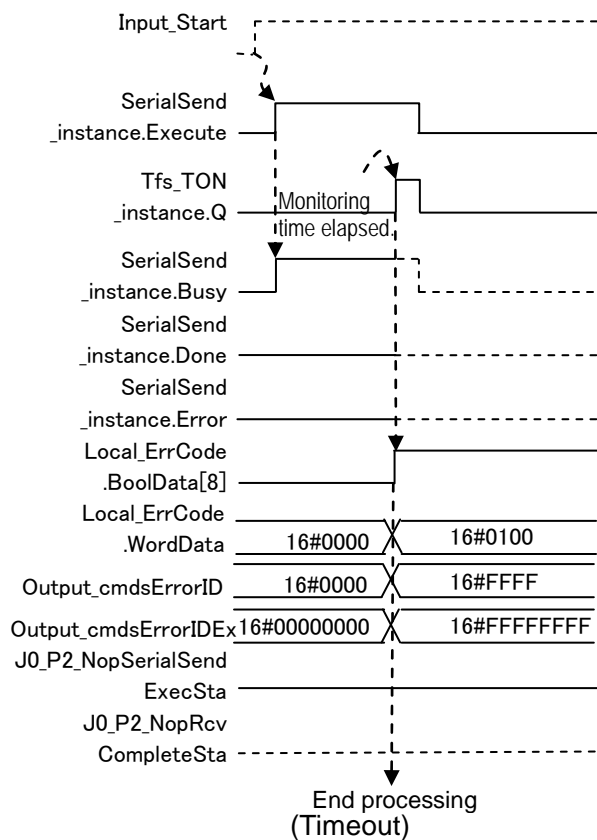
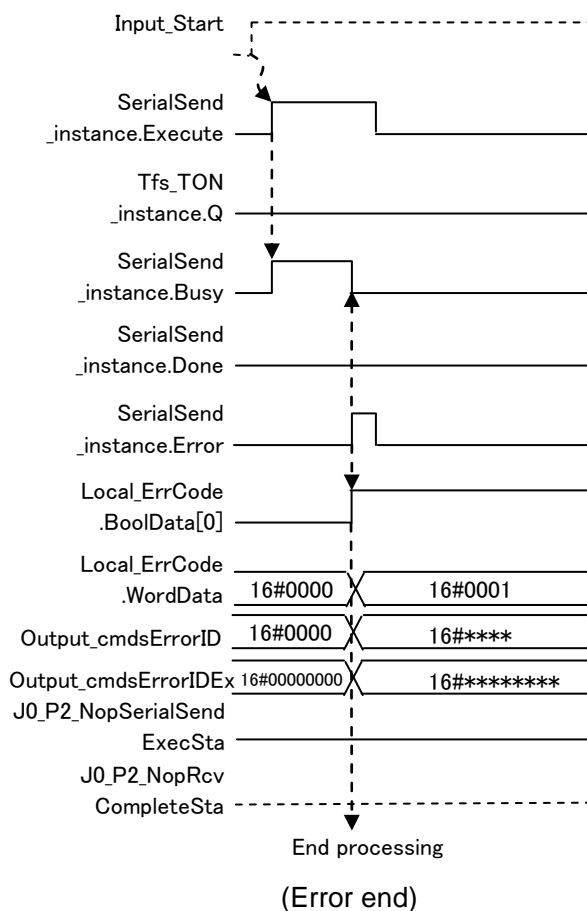
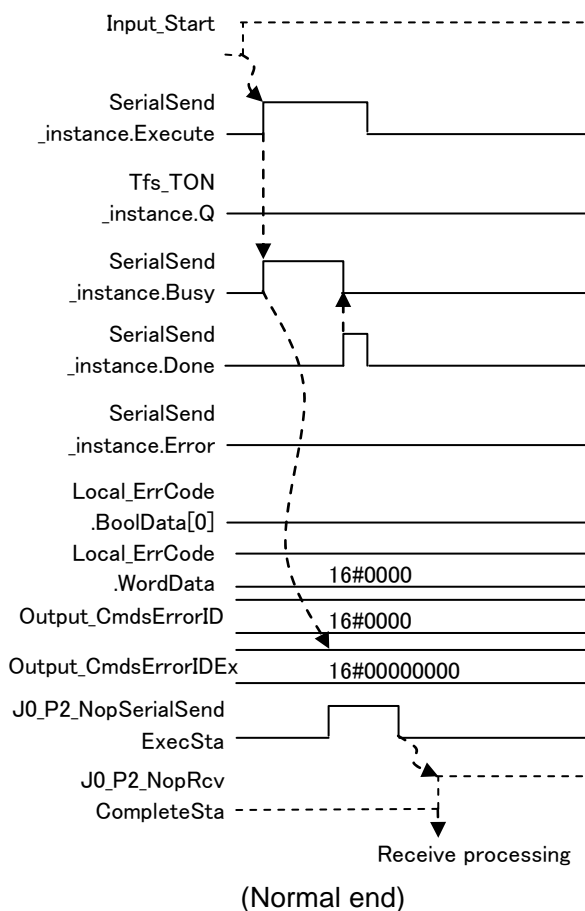
•Start&End processing



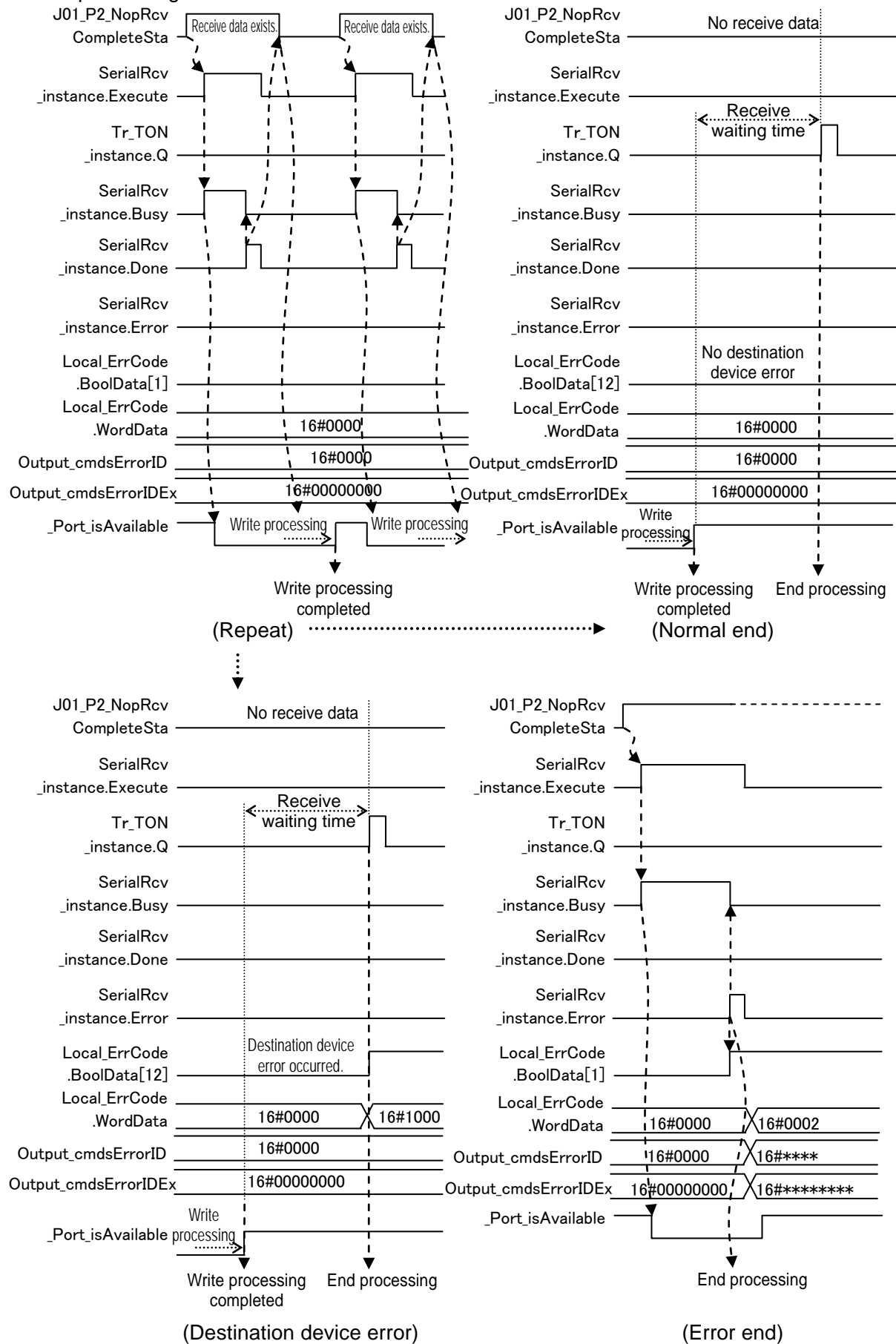
If *Input_Start* changes from TRUE to FALSE during execution, a normal end or an error end is output for one period after the processing is completed as described below.

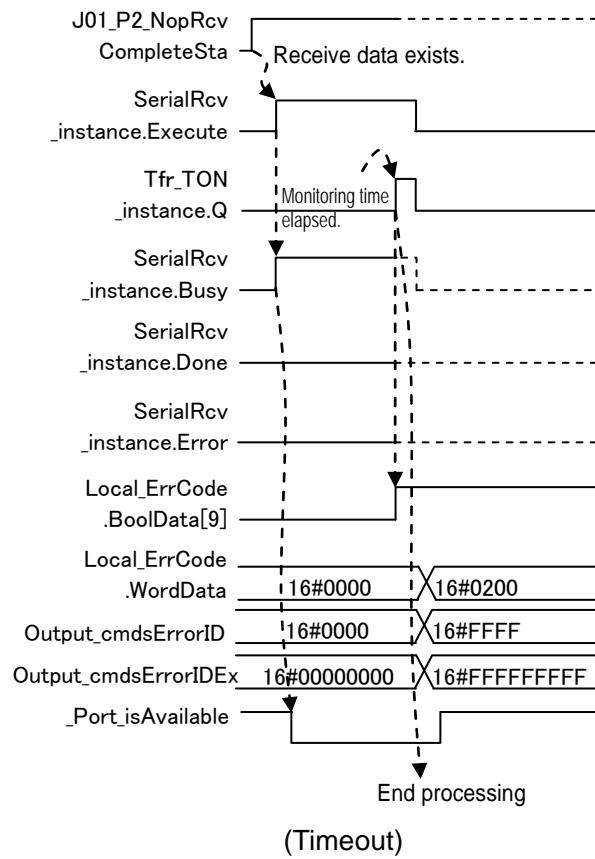


•Send processing



●Receive processing





9.7. Error Process

The error codes for this program are shown below.

Refer to the descriptions on the error codes (*Output_ErrCode*) listed in 9.7.1 Common Errors and check the detailed codes listed in 9.7.2 Transmission Errors to 9.7.4. Destination Device Errors.

9.7.1. Common Errors

The error codes commonly used for errors are shown below.

- Error code [*Output_ErrCode*]

The error information is stored in *Output_ErrCode*.

| Error code | Description |
|------------|--|
| 16#0000 | Normal end |
| 16#0001 | Send processing ended in error. (Serial communications instruction error) |
| 16#0002 | Receive processing ended in error. (Serial communications instruction error) |
| 16#0100 | The send processing could not be completed within the time limit. (Timeout error) |
| 16#0200 | The reception processing was not completed in time. (Timeout error) (Including the case the arrival of the expected response cannot be confirmed) |
| 16#0010 | The processing number is invalid. |
| 16#0020 | Send/receive required/not required detection is illegal. |
| 16#1000 | The response from the destination device is illegal. (Destination device error) |
| 16#2000 | The FCS value of the data received from the destination device is illegal. |
| 16#8000 | Transmission error (Transmission error) |

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

(Example) Transmission error + Send processing error

WORD#16#8000 (Transmission error)

+WORD#16#0001 (Send processing error)

↓

Output_ErrorID: WORD#16#8001

9.7.2. Transmission Errors

The error codes commonly used for transmission errors are shown below.

- Transmission error status [Output_TransErrCode]

When a transmission error occurs, *Output_TransErrCode* stores the sum of the data of transmission error status and the destination device error.

| Bit | Description |
|---------|---|
| 15 | 1:Transmission error 0: Normal |
| 14 | (Not used) |
| 13 | 1:Destination device FCS error 0:Normal |
| 12 | 1:Destination device error 0:Normal |
| 5 to 11 | (Not used) |
| 4 | 1:Overflow error 0:Normal |
| 3 | 1:Framing error 0:Normal |
| 2 | 1:Parity error 0:Normal |
| 0 and 1 | (Not used) |

9.7.3. Serial Communications Instruction Error

The error codes for an error end of the serial communications instructions (SerialSend instruction and SerialRcv instruction) are shown below.

- Serial communications instruction error codes [Output_CmdsErrorID and Output_CmdsErrorIDEx]

An error code of *ErrorID* is stored in *Output_CmdsErrorID* and an error code of *ErrorIDEx* is stored in *Output_CmdsErrorIDEx*.

[Output_CmdsErrorID]

| Code | Description |
|---------|---|
| 16#0000 | Normal end |
| 16#0400 | An input parameter for an instruction exceeded the valid range for an input variable. |
| 16#0406 | The data position specified for an instruction exceeded the data area range. |
| 16#0407 | The results of instruction processing exceeded the data area range of the output parameter. |
| 16#040D | The Unit specified for an instruction does not exist. |
| 16#0C00 | The Serial Communications Unit is not in the serial communications mode required to execute an instruction. |
| 16#0800 | An error occurred when a command was sent or received. |
| 16#0801 | The port is being used. |
| 16#FFFF | The instruction is not completed. |



Additional Information

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

[Output_CmdsErrorIDEx]

| Code | Description |
|-------------|--|
| 16#00000000 | Normal end |
| 16#00000205 | The serial communications mode is set to Host Link Mode. |
| 16#00000401 | The serial communications mode is set to Protocol Macro, NT Link, Echoback Test, or Serial Gateway Mode. |
| 16#00001001 | The command is too long. |
| 16#00001002 | The command is too short. |
| 16#00001003 | The value of SendSize does not match the number of send bytes. |
| 16#00001004 | The command format is incorrect. |
| 16#0000110C | Other parameter error |
| 16#00002201 | The SerialSend or SerialRcv instruction is already in execution. |
| 16#00002202 | The protocol is being switched, so execution is not possible. |
| 16#FFFFFFFF | The instruction is not completed. |



Additional Information

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

**Additional Information**

For details and troubleshooting of the SerialSend and SerialRcv instruction errors, refer to *9-3 Troubleshooting of Section 9 Troubleshooting and Maintenance* in the *CJ-series Serial Communications Units Operation Manual for NJ-series CPU Unit* (Cat. No. W494).

9.7.4. Destination Device Error

The error codes for destination device errors are shown below.

●Destination device error code [Output_MErrCode]

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

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

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

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

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

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



Additional Information

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

10. Revision History

| Revision code | Date of revision | Revision reason and revision page |
|---------------|------------------|-----------------------------------|
| 01 | 2013/04/15 | First edition |
| | | |
| | | |

OMRON Corporation Industrial Automation Company
Tokyo, JAPAN

Contact: www.ia.omron.com

Regional Headquarters

OMRON EUROPE B.V.

Wegalaan 67-69-2132 JD Hoofddorp
The Netherlands
Tel: (31)2356-81-300/Fax: (31)2356-81-388

OMRON ELECTRONICS LLC

One Commerce Drive Schaumburg,
IL 60173-5302 U.S.A.
Tel: (1) 847-843-7900/Fax: (1) 847-843-7787

OMRON ASIA PACIFIC PTE. LTD.

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

OMRON (CHINA) CO., LTD.

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

Authorized Distributor:

© OMRON Corporation 2013 All Rights Reserved.
In the interest of product improvement,
specifications are subject to change without notice.

Cat. No. P544-E1-01

0911(-)