OMRON

Machine Automation Controller NJ-series

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

ZW-series Displacement Sensor

Network Connection Guide

SYSMAC always in control

P559-E1-01

About Intellectual Property Right and Trademarks

Table of Contents

1. Related Manuals

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

Cat.No	Model	Manual name
W500	NJ501-[][][][] NJ301-[][][][]	NJ-series CPU Unit Hardware User's Manual
W501	NJ501-[][][][] NJ301-[][][][]	NJ-series CPU Unit Software User's Manual
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
Z322	ZW-C1[]	Confocal Fiber Type Displacement Sensor User's Manual
Z332	ZW-CE1[]	ZW Series Displacement Sensor (Confocal Fiber Type) User's Manual

2. Terms and Definitions

Term	Explanation and Definition
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 of abnormal occurrence.

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

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

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

(5) This document provides the latest information as of July 2013. It is subject to change without notice for improvement.

The following notation is used in this document.

⚠ WARNING	Indicates a potentially hazardous situation which, if not avoided, 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.

Symbol

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

4. Overview

This document describes the procedure for connecting OMRON Corporation's Displacement Sensor (ZW series) with OMRON Corporation's 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 Sysmac Studio 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 "VR (version information acquisition)" command 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_ZW_SERI232C_EV101.smc	Ver.1.01

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

⚠ Caution

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

5. Applicable Devices and Support Software

5.1. Applicable Devices

The applicable devices are as follows:

Manufacturer	Name	Model	Version
OMRON	NJ-series CPU Unit	NJ501-[][][][] NJ301-[][][][]	Versions listed in Section 5.2 or higher versions
OMRON	Serial Communications Unit	CJ1W-SCU[]2	
OMRON	Confocal Fiber Type Displacement Sensor Controller	ZW-C1[]/CE1[]/CE1[]T	
OMRON	Sensor Head	ZW-S[][]	

Additional Information

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

Additional Information

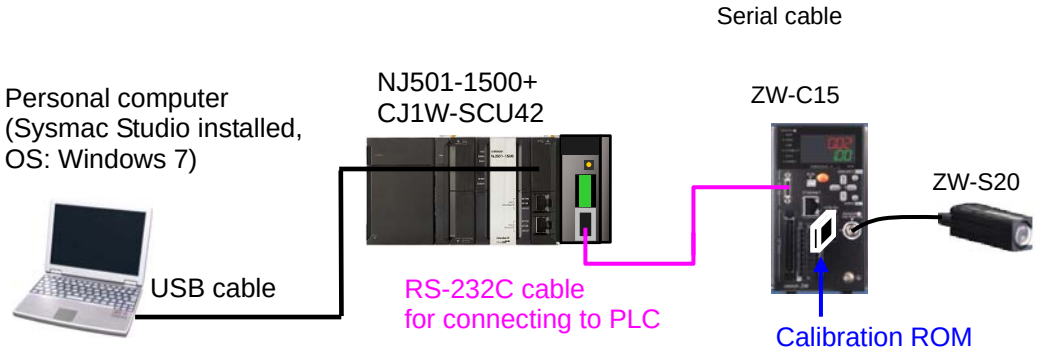
This document describes the procedure to establish the network connection. It does not provide information about operation, installation nor wiring method of each device.
For details on the above products (other than communication connection procedures), refer to the manuals for the corresponding products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

Additional Information

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

5.2. Device Configuration

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



Manufacturer	Name	Model	Version
OMRON	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	Sysmac Studio	SYSMAC-SE2[][][]	Ver.1.04
OMRON	Sysmac Studio project file	OMRON_ZW_SERI232C_EV101.smc	Ver.1.01
-	Personal computer (OS:Windows7)	-	
-	USB cable (USB 2.0 type B connector)	-	
OMRON	RS-232C cable for connecting to PLC	ZW-XPT2	
OMRON	Displacement Sensor Controller	ZW-C15	Ver.1.000
OMRON	Displacement Sensor Sensor Head	ZW-S20	
OMRON	Calibration ROM	(Included with Sensor Head.)	

Precautions for Correct Use

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

Precautions for Correct Use

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

Additional Information

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

Additional Information

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

The system configuration in this document uses USB for the connection between the personal computer and the Controller. For information on how to install the USB driver, refer to *A-1 Driver Installation for Direct USB Cable Connection* of the *Sysmac Studio Version 1 Operation Manual* (Cat.No. W504).

6. Serial Communications Settings

This section provides the specifications such as cable wiring and communications parameters 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.

Setting item	Serial Communications Unit	Displacement Sensor
Unit number	0	-
Communications (connection) port	Port 2 (RS-232C)	-
Serial communications mode	No-protocol	-
Data length	8 bits	8 bits (Default)
Stop bit	1 bit	1 bit (Default)
Parity	None	None (Default)
Baud rate	38,400 bps	38,400 bps (Default)
CTS control	None	OFF (Default)
No-protocol start code	None (Default)	-
No-protocol end code Terminator	Yes (16#0D)	[CR] (Default)

Precautions for Correct Use

This document 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.

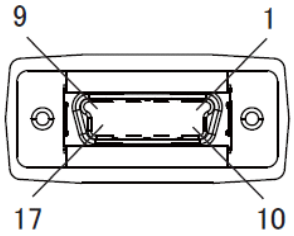
6.2. Cable Wiring Diagram

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).
Check the connector configuration and pin assignment for wiring.

■Connector configuration and pin assignment

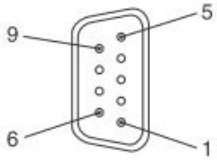
<OMRON ZW-C15> Applicable connector: 17-pin square connector

Pin No.	Signal name
1	GND(0V)
2	TXD(SD)
3	RXD(RD)
4	RTS(RS)
5	CTS(CS)
6 to 17	NC
Shell	FG

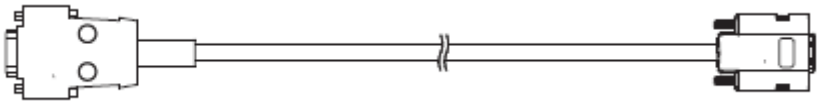


<OMRON CJ1W-SCU42> Applicable connector: D-sub 9-pin

Pin No.	Abbreviation	Signal name	I/O
1	FG	Shield	---
2	SD	Send data	Output
3	RD	Receive data	Input
4	RS	Request to send	Output
5	CS	Clear to send	Input
6	5V	Power supply	---
7	DR	Data set ready	Input
8	ER	Data terminal ready	Output
9	SG	Signal ground	---
Shell	FG	Shield	---



■Cable/Pin arrangement (RS-232C cable for connecting to PLC: ZW-XPT2)



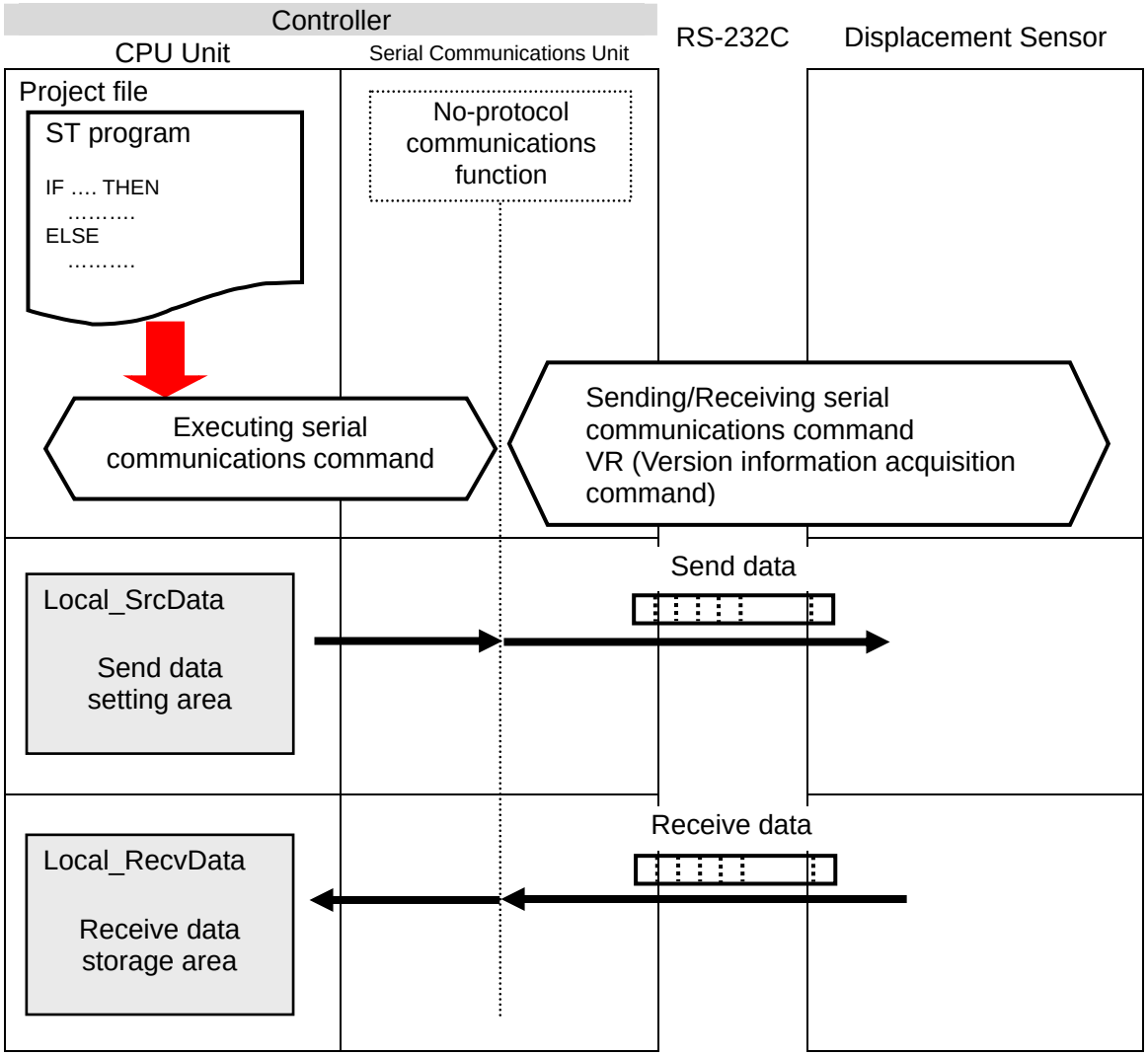
Serial Communications Unit (CJ1W-SCU42)		
RS-232C interface	Signal name	Pin No.
	FG	1
	SD	2
	RD	3
	RS	4
	CS	5
	5V	6
	DR	7
	ER	8
	SG	9
	FG	Shell
D-SUB 9-pin Cable connector type: Male		

Displacement Sensor (ZW-C15)		
Pin No.	Signal name	RS-232C interface
1	GND(0V)	
2	TXD(SD)	
3	RXD(RD)	
4	RTS(RS)	
5	CTS(CS)	
6 to 17	NC	
Shell	FG	
17-pin square connector Cable connector type: Male		

6.3. Example of Checking Connection

This document shows an example of an ST (structured text) program in which the Controller sends/receives a message to/from the Displacement Sensor.

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



7. Connection Procedure

This section describes the procedure for connecting the Displacement Sensor to the Controller via serial communications.
This document explains the procedures for setting up the Controller and Displacement Sensor from the factory default setting. For the initialization, refer to *Section 8 Initialization Method*.

7.1. Work Flow

Take the following steps to connect the Displacement Sensor to the Controller via serial communications.

7.2. Setting Up the Displacement Sensor	Set up the Displacement Sensor.
↓	
7.2.1. Parameter Setting	Set the parameters of the Displacement Sensor.
↓	
7.3. Setting Up the Controller	Set up the Controller.
↓	
7.3.1. Hardware Setting of the Serial Communications Unit	Set the hardware switch on the Serial Communications Unit and connect to the Controller.
↓	
7.3.2. Starting the Sysmac Studio and Importing the Project File	Start the Sysmac Studio and import the project file.
↓	
7.3.3. Checking the Parameters and Building	Check the set parameters, execute the program check on the project data and build the Controller.
↓	
7.3.4. Connecting Online and Transferring the Project Data	Connect online with the Sysmac Studio and transfer the project data to the Controller.
↓	
7.3.5. Transferring the Unit Settings	Transfer the setting data of the Serial Communication Unit.
↓	
7.4. Checking the Serial Communications	Execute the program and confirm that serial communications are normally performed.
↓	
7.4.1. Executing the Program and Checking the Receive Data	Execute the program and confirm that the correct data are written to the variables of the Controller.

7.2. Setting Up the Displacement Sensor

Set up the Displacement Sensor.

7.2.1. Parameter Setting

Set the parameters of the Displacement Sensor.

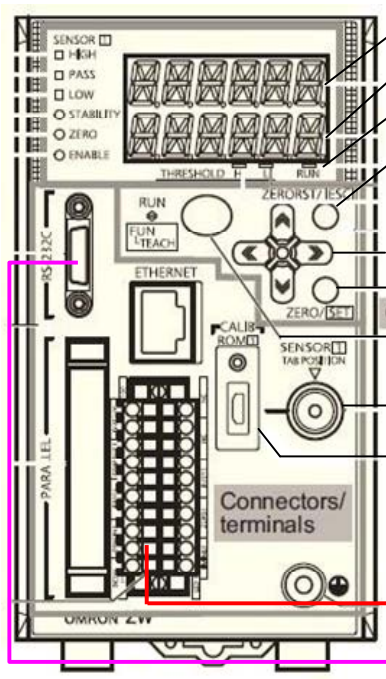
1 Check the keys and display used to set parameters of the Displacement Sensor.

Connect the Sensor Head.

Insert the Calibration ROM.

Connect the RS-232C cable (for connecting to PLC).

Turn ON the power supply to the Displacement Sensor.



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

The RUN indicator is lit as shown on the right.

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

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

3 A confirmation screen for mode switching is displayed.

Press the **ZERO/SET** Key.

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

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

7	DATA is displayed on the main display.	
	Press the **ZERO/SET** Key.	Press the **ZERO/SET** Key once.
	Confirm that 8bit (default value of the data length) is displayed on the sub-display. *If the value is different, change the value by pressing ↑ (UP) or ↓ (DOWN).	 <Setting range> 8bit/7bit Default: 8bit
	Press the **ZERORST/ESC** Key once. The first screen in this step is displayed again.	Press the **ZERORST/ESC** Key once.
	Press the → (RIGHT) Key once.	DATA is displayed. Press the → (RIGHT) Key once.
8	PARITY is displayed on the main display.	
	Press the **ZERO/SET** Key.	Press the **ZERO/SET** Key once.
	Confirm that OFF (default value of the parity) is displayed on the sub-display. *If the value is different, change the value by pressing ↑ (UP) or ↓ (DOWN).	 <Setting range> OFF/ODD/ EVEN Default: OFF
	Press the **ZERORST/ESC** Key once. The first screen in this step is displayed again. Press the → (RIGHT) Key once.	Press the **ZERORST/ESC** Key once. PARITY is displayed. Press the → (RIGHT) Key once.

9

STOP is displayed on the main display.

Press the **ZERO/SET** Key.

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

Confirm that 1bit (default value of the stop bit) is displayed on the sub-display.
*If the value is different, change the value by pressing ↑ (UP) or ↓ (DOWN).

<Setting range>
1bit/2bit
Default:1bit

Press the **ZERORST/ESC** Key once. The first screen in this step is displayed again.
Press the → (RIGHT) Key once.

Press the **ZERORST/ESC** Key.

STOP is displayed.

Press the → (RIGHT) Key once.

10

BAUD.RT is displayed on the main display.

Press the **ZERO/SET** Key.

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

Confirm that 38400 (default value of the baud rate) is displayed on the sub-display.
*If the value is different, change the value by pressing ↑ (UP) or ↓ (DOWN).

<Setting range>
9600/19200/
38400/57600/
115200
Default:
38400 bps

Press the **ZERORST/ESC** Key once. The first screen in this step is displayed again.
Press the → (RIGHT) Key once.

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

BAUD.RT is displayed.

Press the → (RIGHT) Key once.

11

CS/RS is displayed on the main display.

Press the **ZERO/SET** Key.

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

Confirm that OFF (default value of the CS/RS control) is displayed on the sub-display.
*If the value is different, change the value by pressing ↑ (UP) or ↓ (DOWN).

<Setting range>
OFF/ON
Default: OFF

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

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

12

RS232C is displayed on the main display.

Press → (RIGHT) or ← (LEFT) Key to change the main display content to DELIMI.

Press the → (RIGHT) or ← (LEFT) Key.

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

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

13

Confirm that CR is displayed on the sub-display.

*If the value is different, change the value by pressing ↑ (UP) or ↓ (DOWN).

<Setting range>
CR/LF/CRLF
Default: CR

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

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

14	The confirmation screen for mode switching is displayed. Press the **ZERO/SET** Key. The save confirmation screen is displayed. Press the **ZERO/SET** Key. The RUN mode screen is displayed.	 Press the **ZERO/SET** Key once.  Press the **ZERO/SET** Key once. 
15	Cycle the power supply to the Displacement Sensor. *The saved setting data will take effect after restarting.	

7.3. Setting Up the Controller

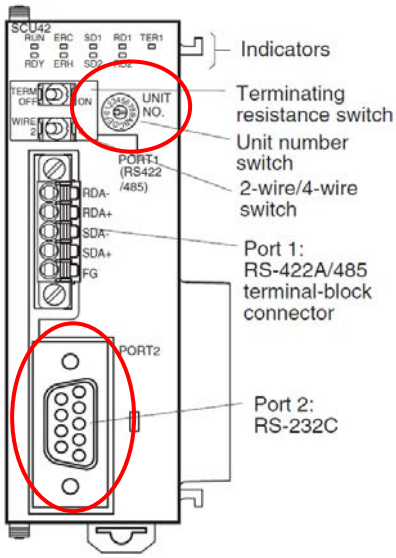
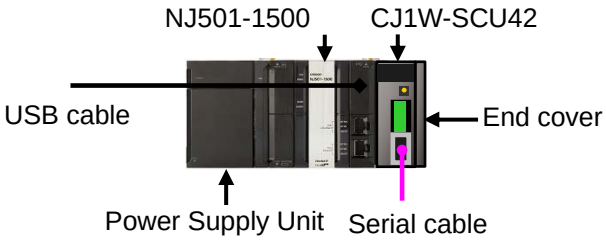
Set up the Controller.

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

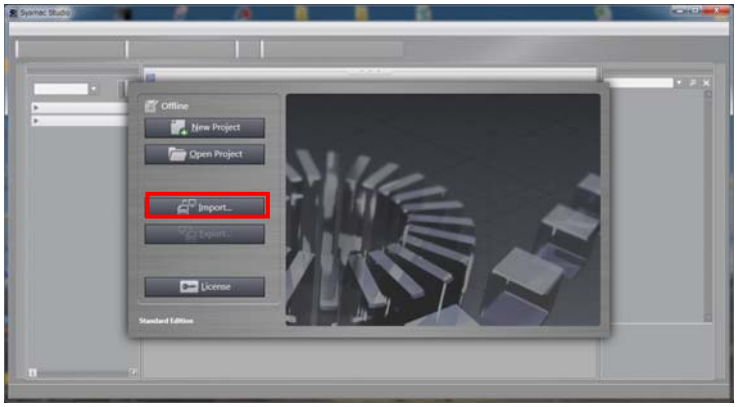
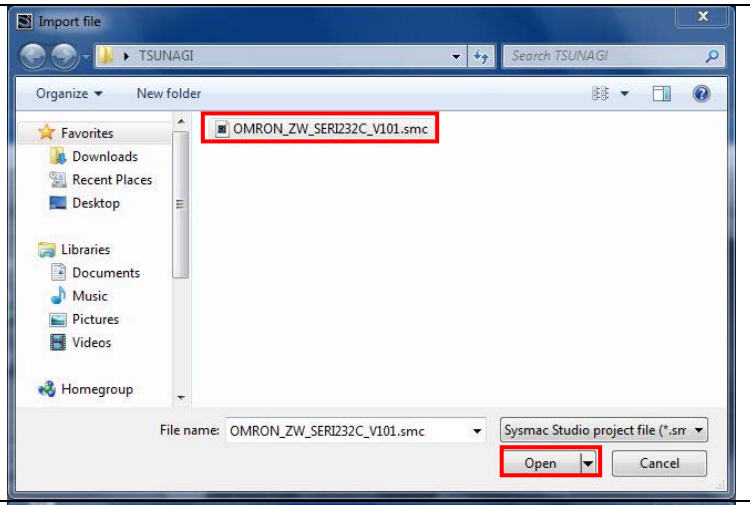
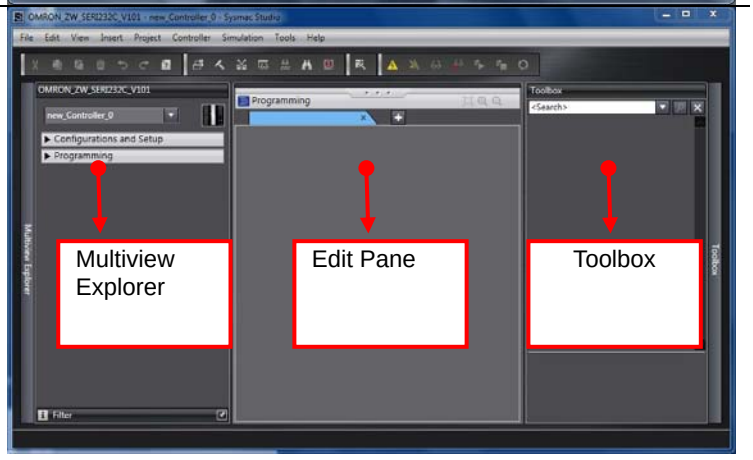
Make sure that the power supply is OFF when you perform the settings.

1	Make sure that the power supply to the Controller is OFF. *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	

7.3.2. Starting the Sysmac Studio and Importing the Project File

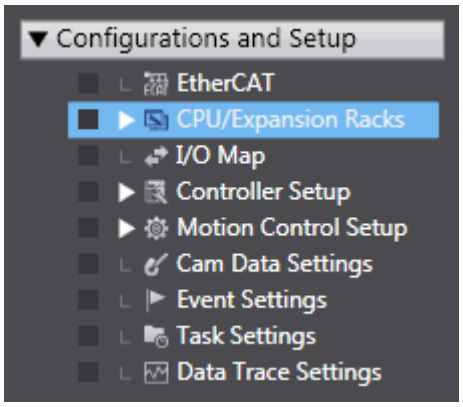
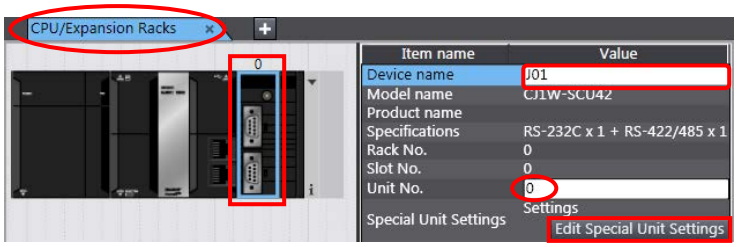
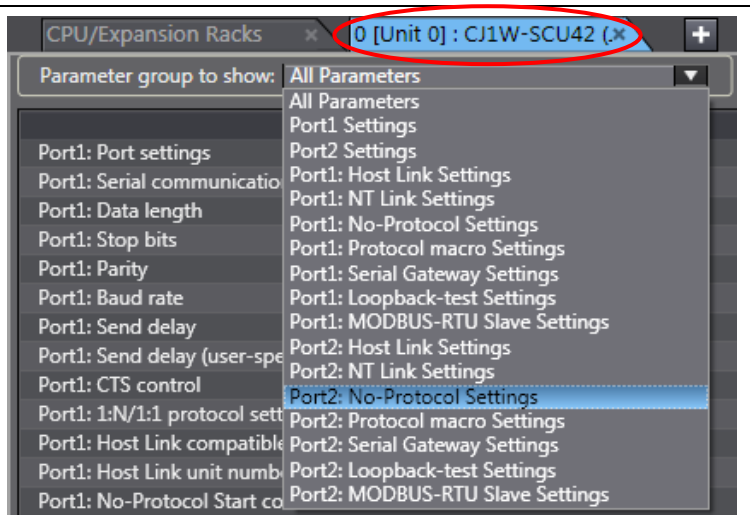
Start the Sysmac Studio and import the project file.

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

1	Confirm that the personal computer and the Controller are connected with the USB cable and turn ON the power supply to the Controller. 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_ZW_SERI232C_EV101.smc and click the **Open** Button. *Obtain the project file from OMRON.	
3	OMRON_ZW_SERI232C_EV101 project is displayed. The left pane is called Multiview Explorer, the right pane is called Toolbox and the middle pane is called Edit Pane. *If an error message is displayed stating “Failed to Load Descendants”, change the version of the Sysmac Studio to any version specified in 5.2. Device Configuration or higher version.	

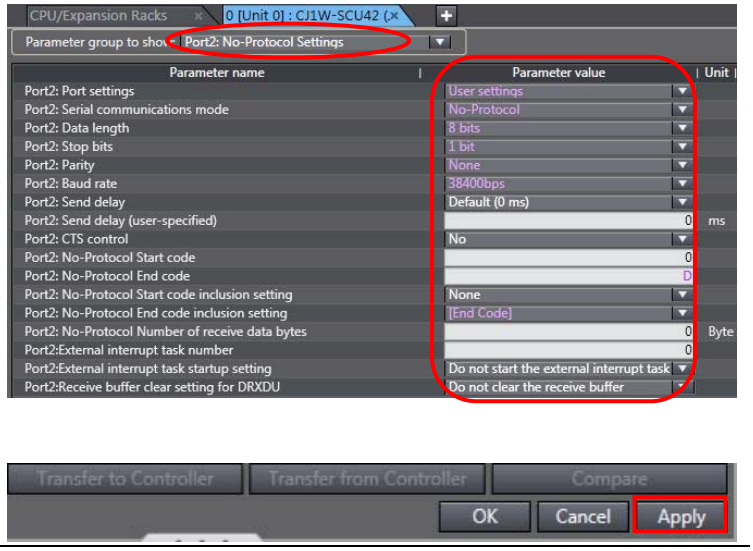
7.3.3. Checking the Parameters and Building

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

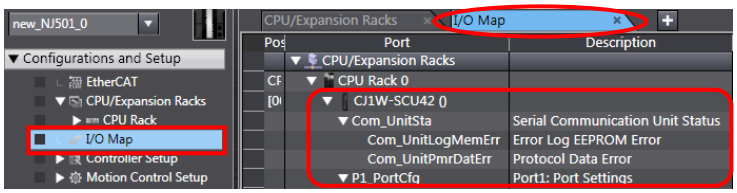
1	Double-click **CPU/Expansion Racks** under **Configurations and Setup** - **Controller Setup** in the Multiview Explorer.	
2	The CPU/Expansion Racks Tab is displayed on the Edit Pane. Select the Serial Communications Unit icon as shown on the right. Confirm that CJ1W-SCU42 is displayed, the device name is J01, and the unit number is 0. *If the setting value is different from 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 the values, click the **Apply** Button.

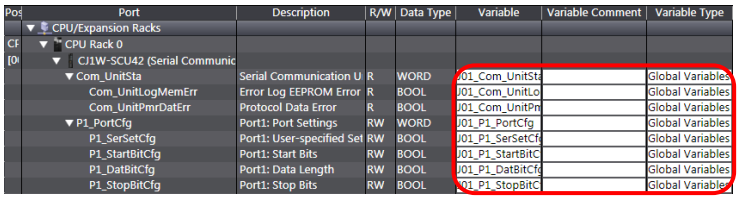


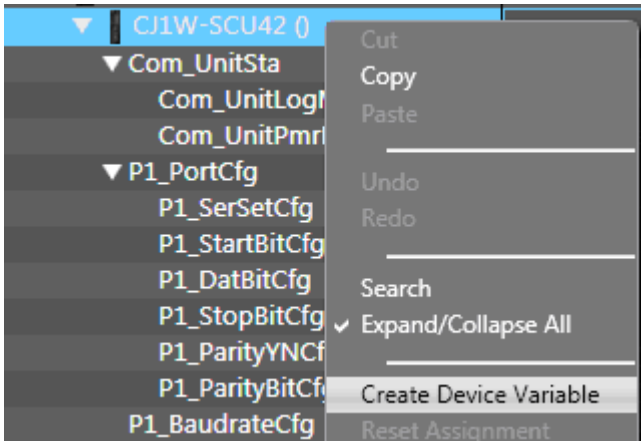
5 Double-click **I/O Map** under **Configurations and Setup** on the Multiview Explorer.
The I/O Map Tab is displayed and then the parameters for the Unit are listed.



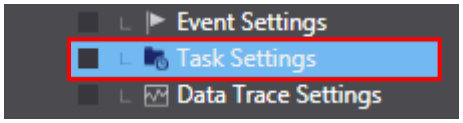
6 Confirm that data in the Variable Columns start with J01 and the Global Variable is set in each Variable Type Column.

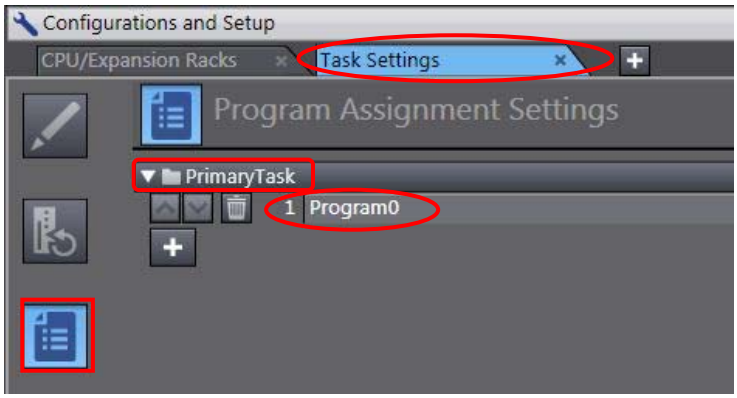
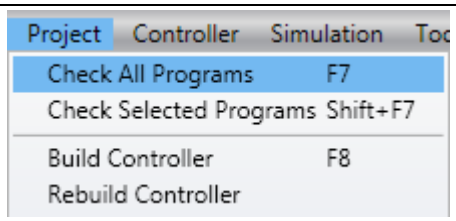
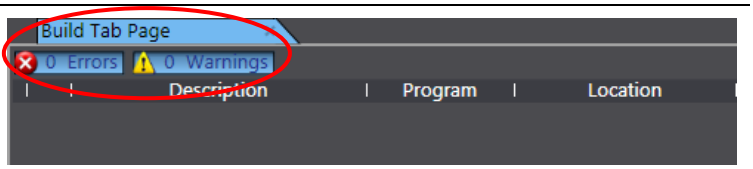
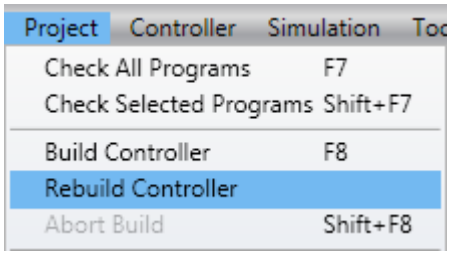
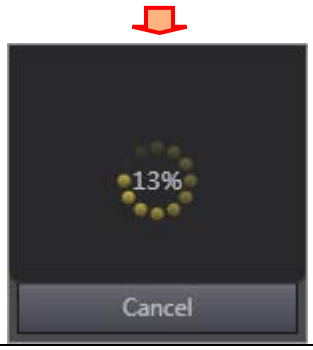
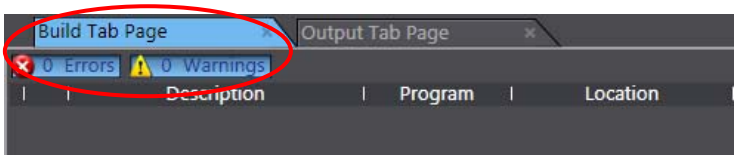
*If the settings are different from the above, right-click on **CJ1W-SCU42** and select ***Create Device Variable***.





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



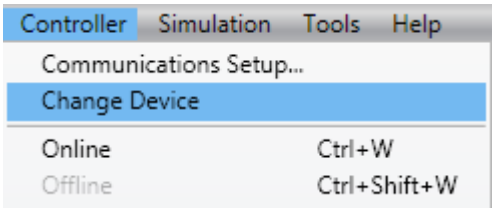
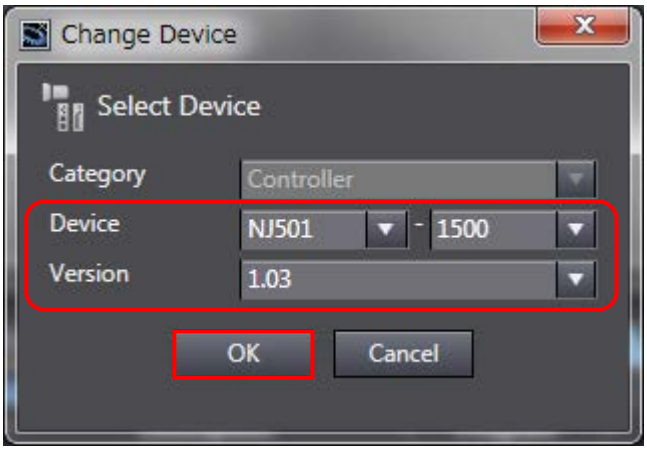
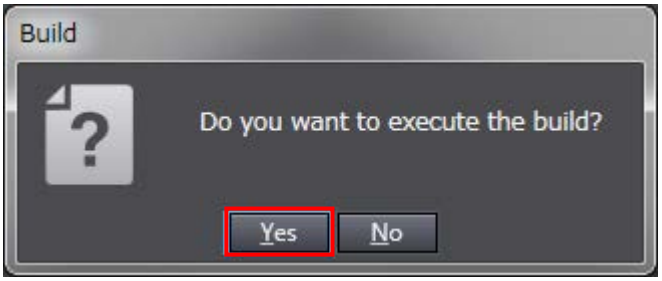
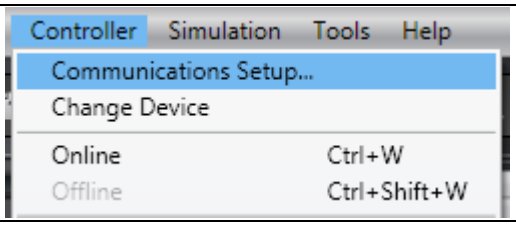
8	The Task Settings Tab Page is displayed in the Edit Pane. Click the **Program Assignment Settings** Button and confirm that Program0 is set under PrimaryTask.	
9	Select ***Check All Programs*** from the Project Menu.	
10	The Build Tab Page is displayed in the Edit Pane. Confirm that "0 Errors" and "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.

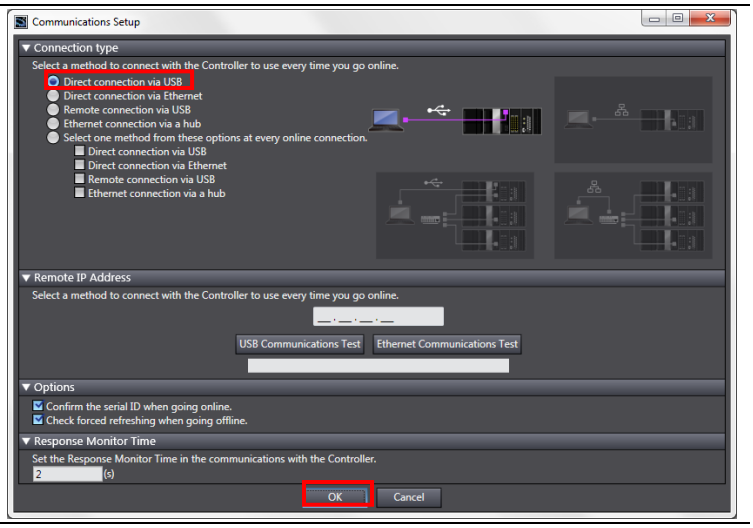
⚠ Caution

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

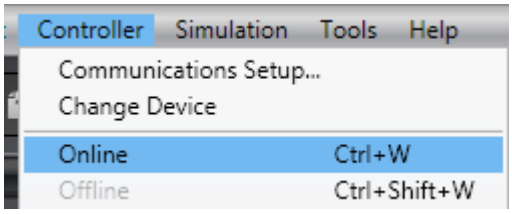
1	Select ***Change Device*** from the Controller Menu.	
2	The Change Device Dialog Box is displayed. Confirm that the Device and Version are set as shown on the right and click the **OK** Button. *If the settings are different, change the values from the pull-down list.	
3	If settings were changed in Step 2, the Build Dialog Box is displayed. Click the **Yes** Button. *This dialog box is not displayed if no change was made.	
4	Select ***Communications Setup*** from the Controller Menu.	

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

Click the **OK** Button.

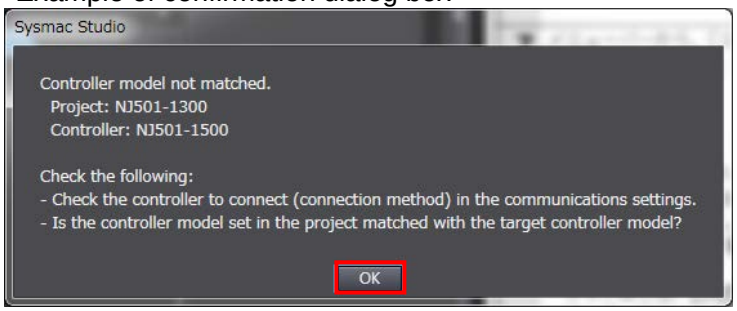


6 Select ***Online*** from the Controller Menu.

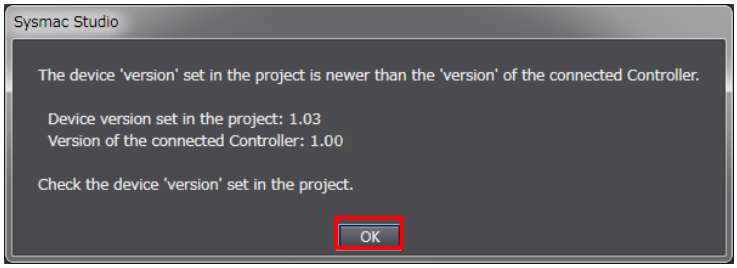


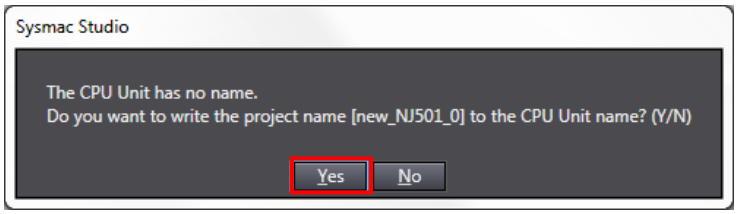
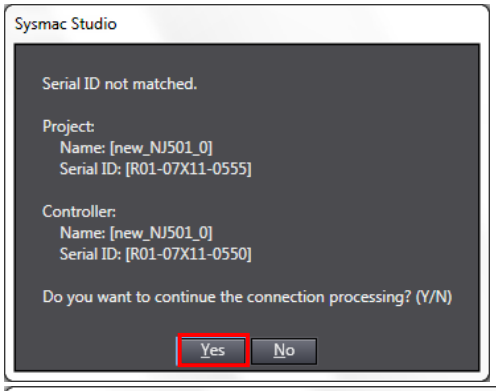
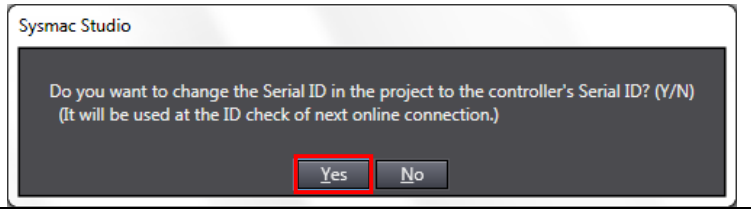
*If the dialog on the right is displayed, the model or version of the Controller does not match that of the project file. Check the device settings of the project file, return to step 1 and try again.
Click the **OK** Button to close the dialog box.

*Example of confirmation dialog box



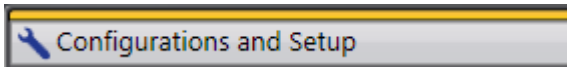
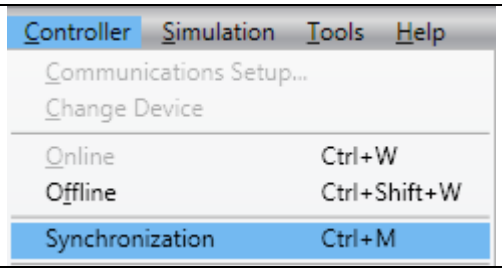
*The model and version displayed on the confirmation dialog box differ depending on the Controller used and the device settings of the project file.



7	A confirmation dialog is displayed. Click the **Yes** Button. *The displayed dialog differs depending on the status of the Controller used. Select the **Yes** Button to proceed with the processing. *The displayed serial ID differs depending on the device.	  

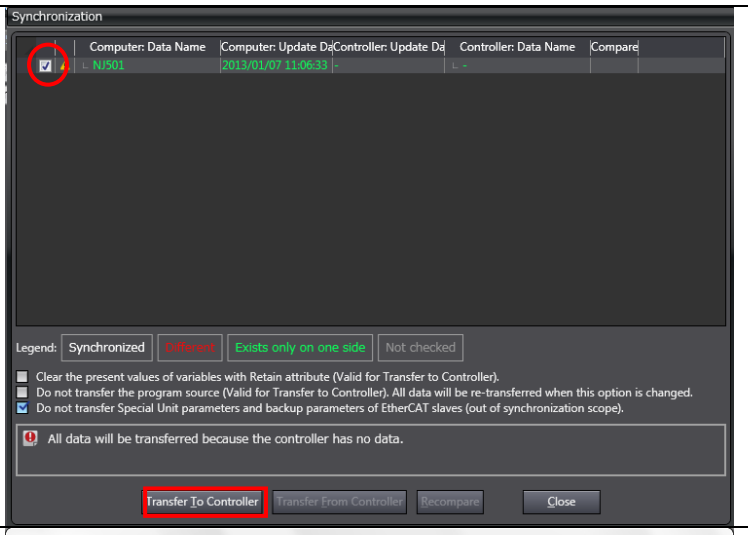
Additional Information

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

8	When an online connection is established, a yellow bar is displayed on the top of the Edit Pane.	
9	Select ***Synchronization*** from the Controller Menu.	

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

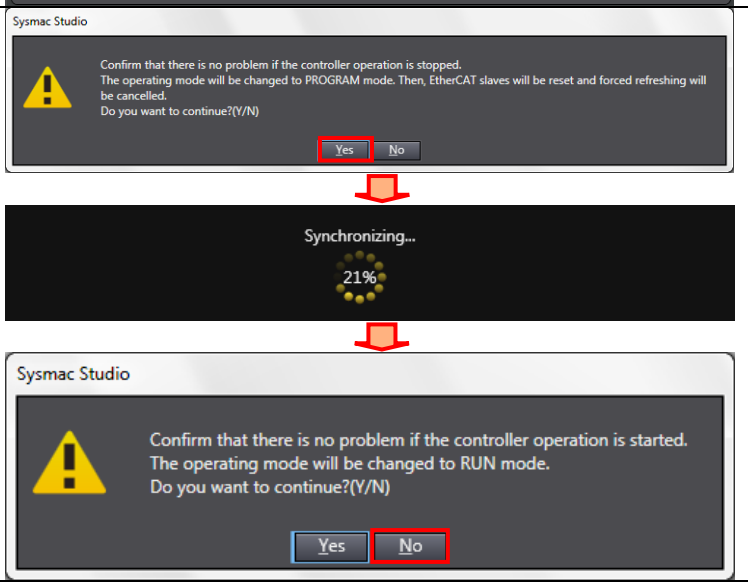
*After executing **Transfer to Controller**, the Sysmac Studio project data is transferred to the Controller and the data are compared.



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

A screen stating "Synchronizing" is displayed.

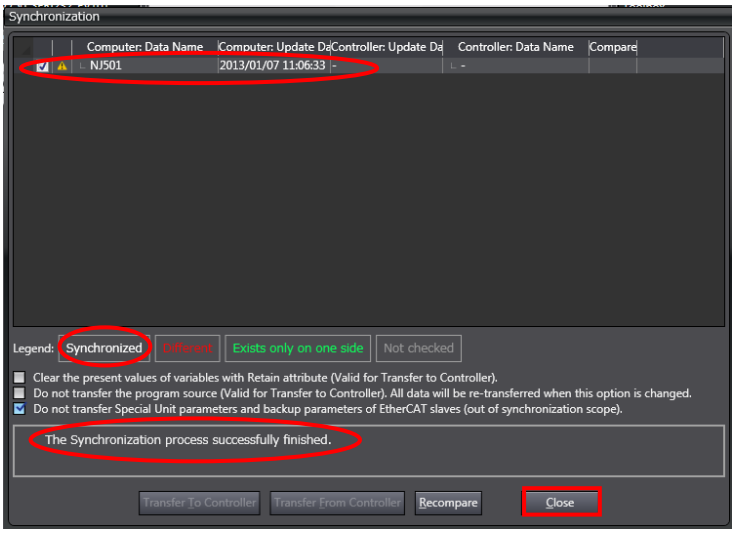
A confirmation dialog box is displayed. Click the **No** Button.

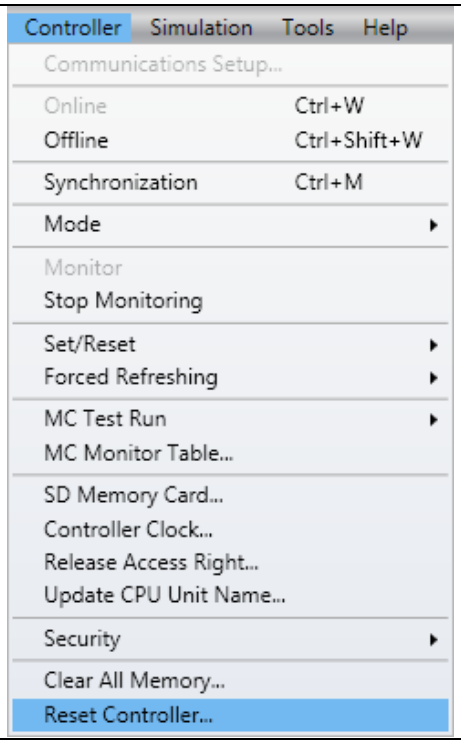
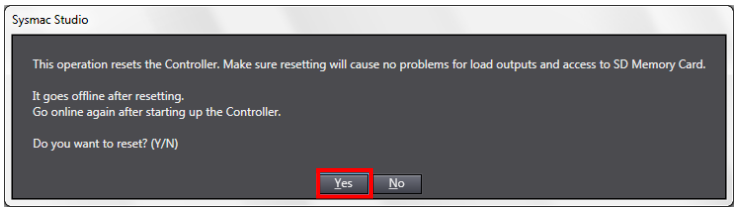
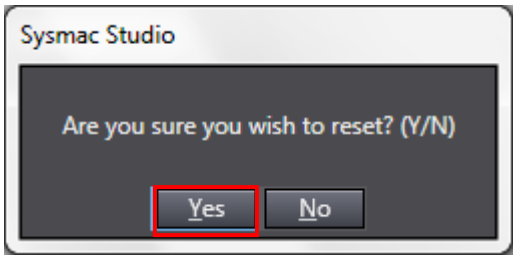
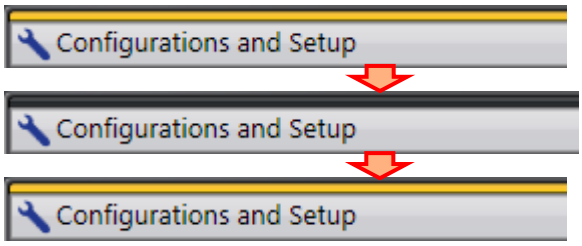


12 Confirm that the synchronized data is displayed with the color specified by “Synchronized” and that a message is displayed stating "The synchronization process successfully finished". If there is no problem, click the **Close** Button.

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

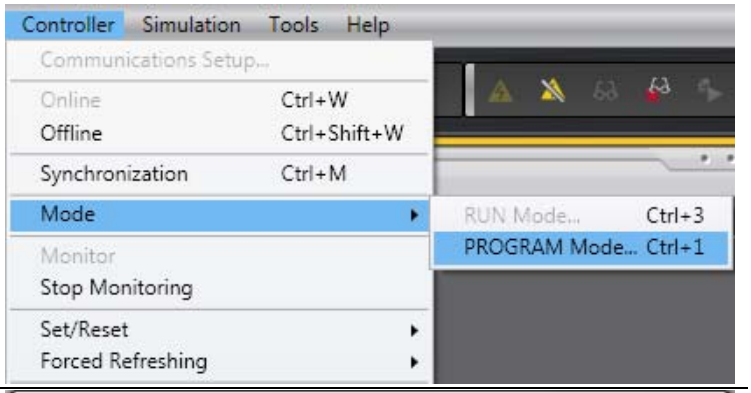
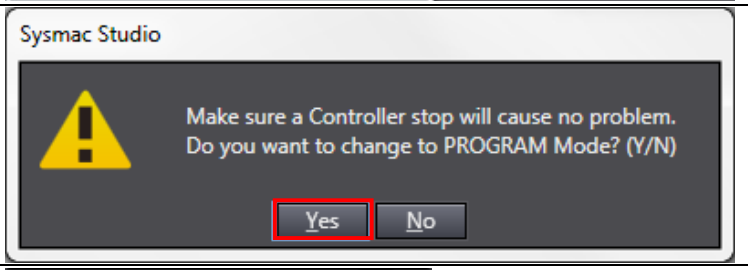
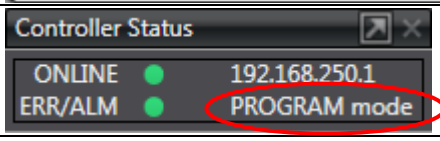
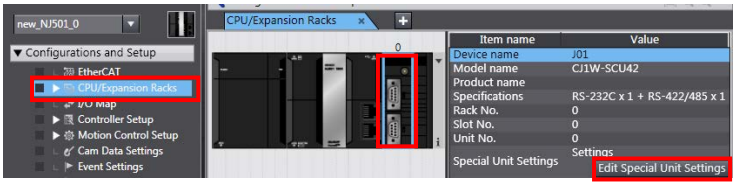
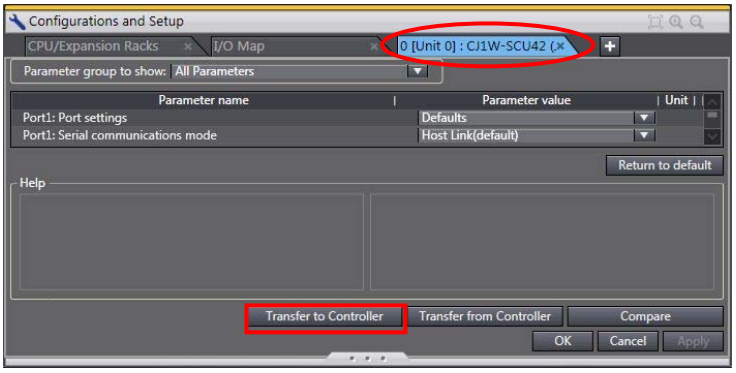
*If the synchronization fails, check the wiring and repeat the procedure described in this section.



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.	

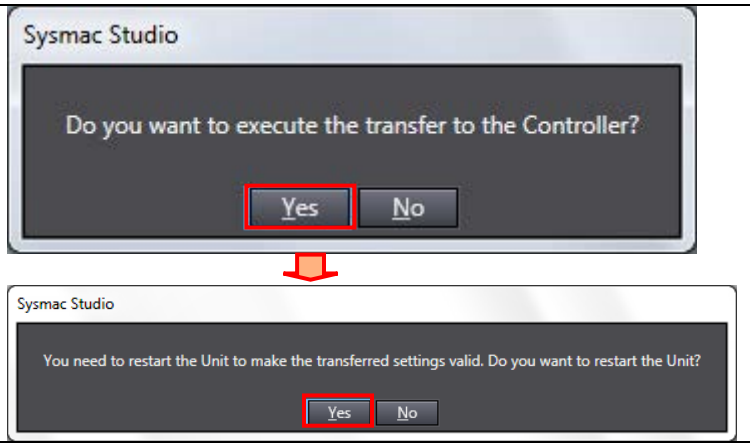
7.3.5. Transferring the Unit Settings

Transfer the setting data of the Serial Communication Unit.

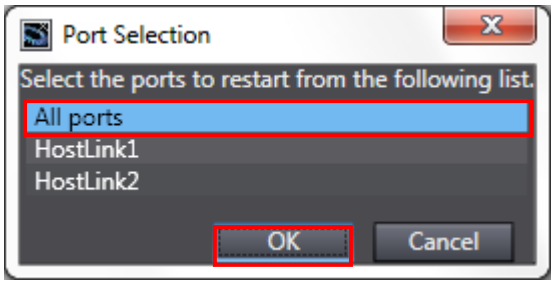
1	Select ***Mode*** - ***PROGRAM Mode*** from the Controller Menu.	
2	A confirmation dialog box is displayed. Click the **Yes** Button.	
3	PROGRAM mode is displayed on the Controller Status Pane.	
4	Double-click **CPU/Expansion Racks** under **Configurations and Setup** in the Multiview Explorer. Select 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 A confirmation dialog box is displayed.
Click the **Yes** Button.

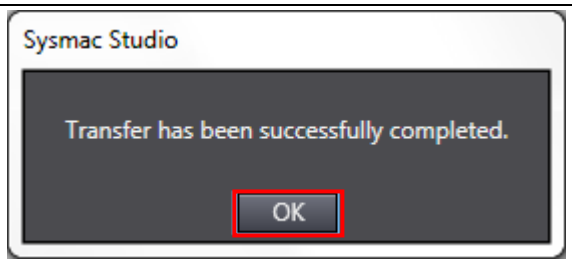
A dialog box is displayed indicating transferring is being performed, and a confirmation dialog box is displayed.
Click the **Yes** Button.



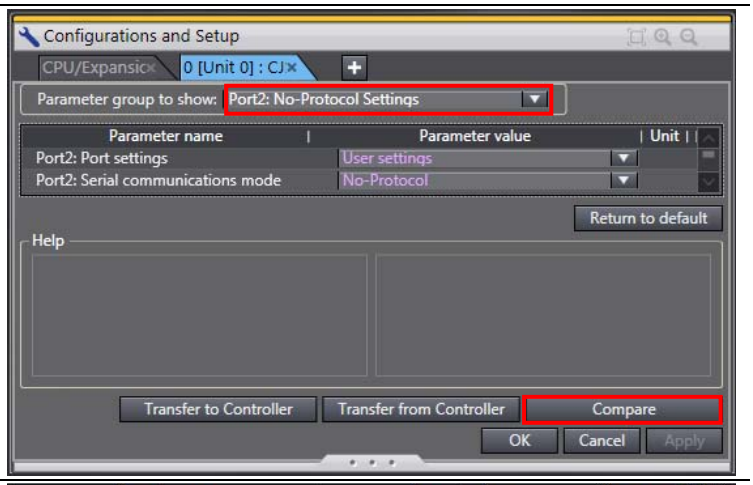
7 The Port Selection Dialog Box is displayed.
Select **All ports** and click the **OK** Button.



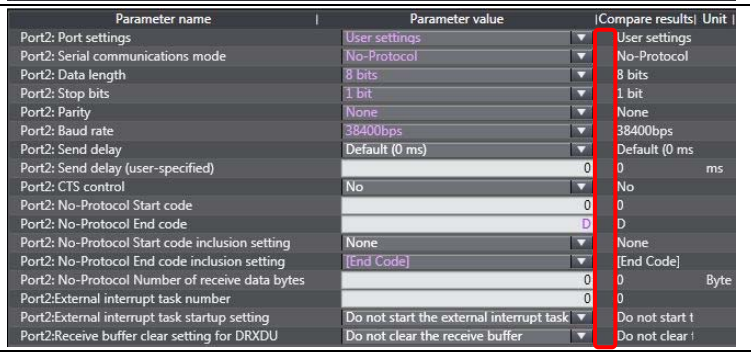
8 A confirmation dialog box is displayed.
Click the **OK** Button.



9 Select *Port2: No-Protocol Settings* from the pull-down list of Parameter group to show.
Click the **Compare** Button.



10 Confirm that "≠" (mismatch) is not shown in the red frame on the right.



7.4. Checking the Serial Communications

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

⚠ Caution

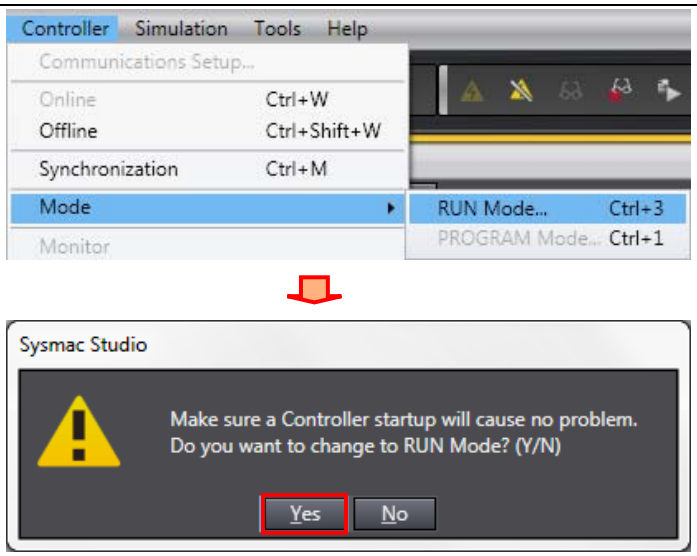
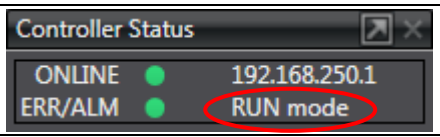
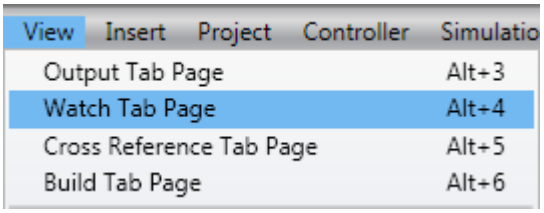
Sufficiently confirm safety before you change the values of variables on a Watch Tab Page when the Sysmac Studio is online with the CPU Unit. Incorrect operation may cause the devices that are connected to Output Units to operate regardless of the operating mode of the Controller.

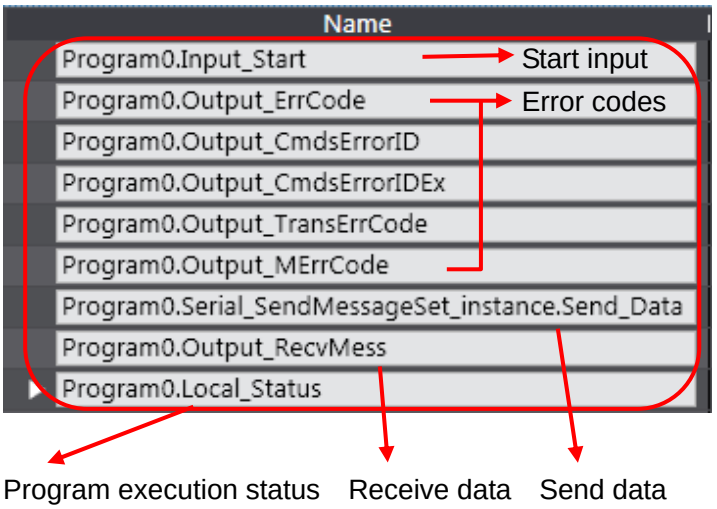
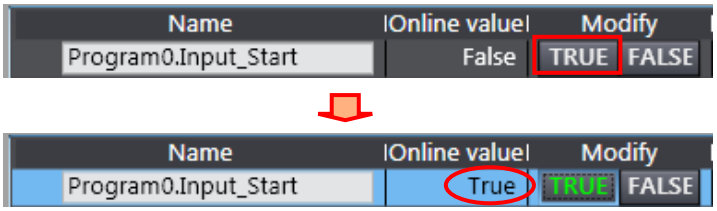
Precautions for Correct Use

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.

1	Select ***Mode*** - ***RUN Mode*** from the Controller Menu. A confirmation dialog box is displayed. Click the **Yes** Button.	
2	RUN mode is displayed on the Controller Status Pane.	
3	Select ***Watch Tab Page*** from the View Menu.	

4	The Watch Tab Page 1 is displayed in the lower section of the Edit Pane.	
5	Confirm that the variables shown on the right are displayed in the Name Columns. *To add a variable, click *Input Name…* *Program0 of the Name is omitted from the following descriptions.	
6	Click **TRUE** on the Modify Column of *Input_Start*. The online value of *Input_Start* changes to True. The program is operated and serial communications are performed with the destination device.	

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	TRUE 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	TRUE FALSE
Done	True	TRUE FALSE
Error	False	TRUE FALSE

8

The response data received from the destination device is stored in *Output_RecvMess*. (*Serial_SendMessageSet_instance.Send_Data* is the send command.)
Specify an area where you want to reference in the Watch Tab Page1 as shown in the right figure.

*The response data differs depending on the device used.

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

Name
Program0.Serial_SendMessageSet_instance.Send_Data
Program0.Output_RecvMess

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

Receive data

•Version information

Product type: ZW-C15
Blank: (2 characters)
Version: Ver1.000
Blank: (1 character)
Release date: 2012/02/09

8. Initialization Method

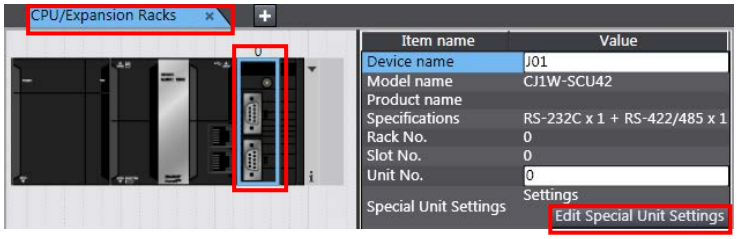
This document explains the setting procedure from the factory default setting.
If the device settings are changed from the factory default setting, some settings may not be applicable as described in this procedure.

8.1. Initializing the Controller

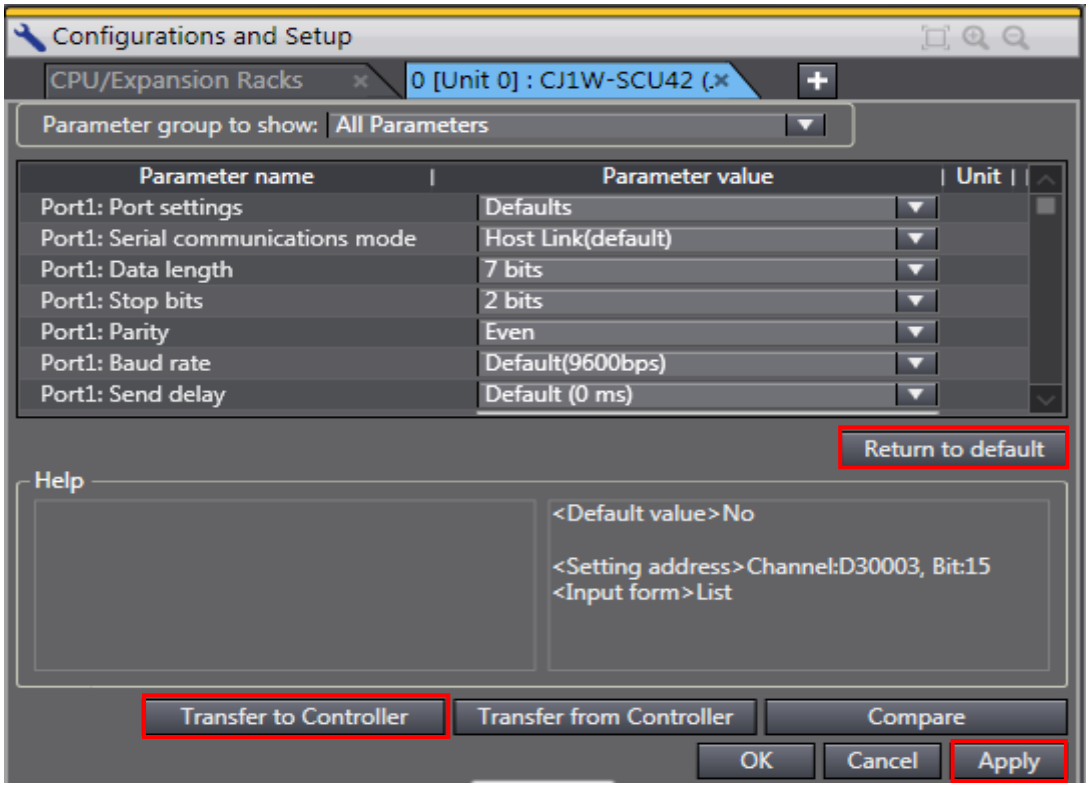
To initialize the settings of the Controller, it is necessary to initialize the Serial Communications Unit and the CPU Unit. Place the operating mode to PROGRAM mode before initialization.

8.1.1. 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.

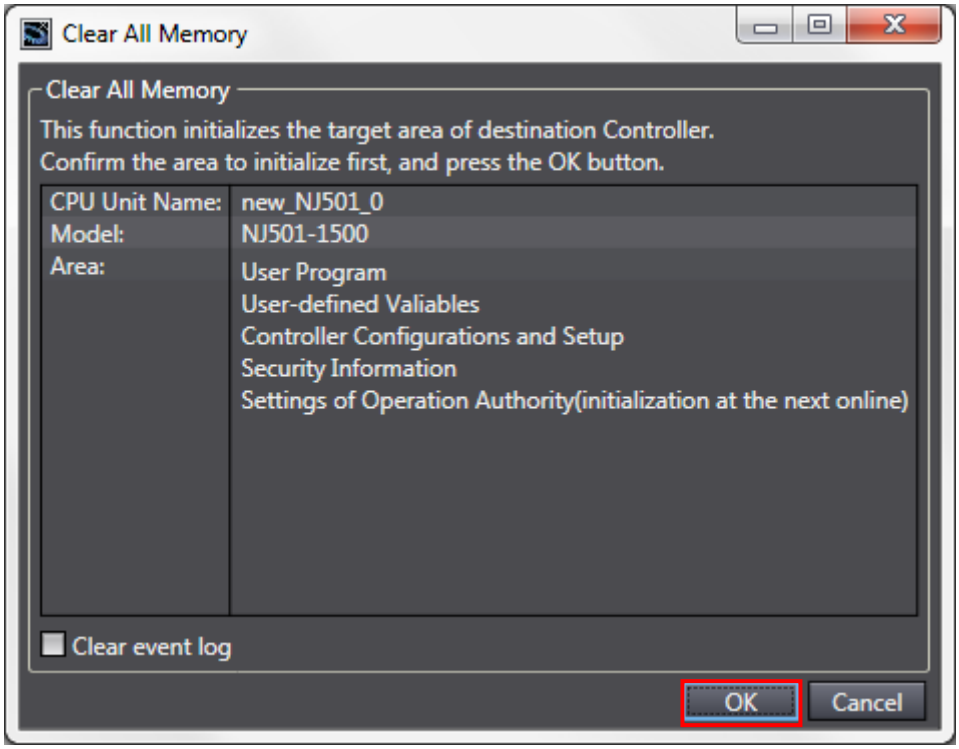


Click the **Return to default** Button and click the **Apply** Button. Then, click the **Transfer to Controller** Button.



8.1.2. CPU Unit

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



8.2. Initializing the Displacement Sensor

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

9. Program

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

9.1. Overview

This section explains the specifications and functions of the program used to check the connection between the Displacement Sensor (ZW series) (hereinafter referred to as the destination device) 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 “VR (version information acquisition command)” to/from the destination device and to detect a normal end or an error end.
A normal end of this program means a normal end of the serial communications.
An 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)

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

Additional Information

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

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.

1.	Sending a command	The SCU Unit issues the send message (command data) set with the program to the destination device.
	↓	
2.	Receiving a response	The SCU Unit receives the receive data (response data) from the destination device and stores it in the specified internal variable of the CPU Unit.

*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 Instructions and Send/Receive Messages

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

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 the 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 SerialSend Execute Done Port Busy SrcDat Error SendSize ErrorID ErrorIDEx	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 *SrcDat[]*	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 SerialRcv Execute Done Port Busy Size Error DstDat ErrorID ErrorIDEx RcvSize	SerialRcv_instance(Execute, Port, Size, DstDat, Done, Busy, Error, ErrorID, ErrorIDEx, RcvSize);

Variables

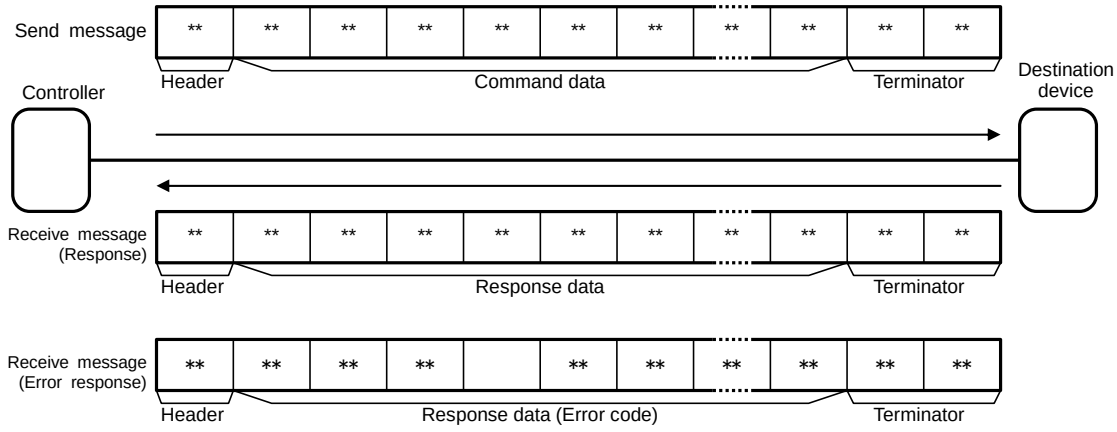
Name	Meaning	I/O	Description	Valid range	Unit	Default
Port	Destination port	Input	Destination port	---	---	---
Size	Receive data size		Size of receive data stored in *DstDat[]*	0 to 256	Bytes	1
DstDat[] (array)	Receive data array	In-out	Receive data array	Depends on data type.	---	---
RcvSize	Receive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

9.2.1. Overview of the Command

This program uses the VR (version information acquisition) command to read information from the destination device.

Command	Description
VR	Acquire version information.

Additional Information

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

9.2.2. Detailed Description of the Command

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

●Command format of the send message

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

•ASCII codes are sent except for the terminator.

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

●Command format of the receive message (normal)

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

•ASCII codes are received except for the terminator.

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

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

●Command format of the receive message (error)

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

•ASCII codes are received except for the terminator.

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

9.2.3. Command Settings

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

●Send data (command) settings

The send data is set in the SendMessageSet function block.

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

*The SCU Unit adds the terminator (CR) to the send message. Therefore, do not set the terminator.

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

●Receive data (response) that is stored

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

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

●Send/Receive messages

*Send message

56	52	0D
V	R	[CR]
Command		Terminator

*Receive message (at normal process)

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

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

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

*Receive message (at error process)

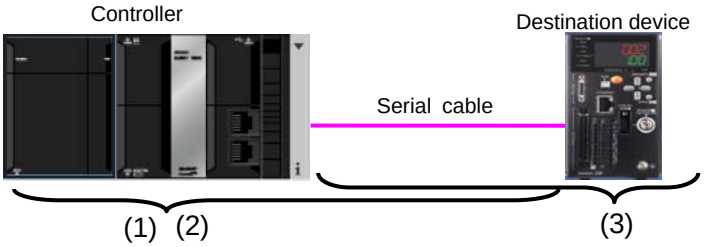
45	52	0D
E	R	[CR]
Error code		Terminator

9.3. Error Detection Processing

This section explains the error detection processing of this program.

9.3.1. Error Detection in the Program

This program detects and handles the following errors (1) to (3). For information on the error codes, refer to *9.7. Error Process.*



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

Errors in the Unit, the command format, or the parameters at the execution of the SerialSend or SendCmd instruction are detected as "serial communications instruction errors". An error is detected with the error code (*ErrorID)* and the expansion error code(*ErrorIDEx)* of the serial communications instructions. If the "Serial communications instruction error" is caused by a transmission error due to, for example, a character corruption or an 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 or receive processing are not normally performed and cannot be completed within the monitoring time, it is detected as a “timeout error”. An error is detected with the timer 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*.

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 a communications process executing without stop due to abnormality, the timer is used in this program to abort the processing (timeout). The timeout value for each processing from the send processing to the receive 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.	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.	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.	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	Structure of communications processing status flags
Busy	BOOL	Communications processing in progress flag TRUE: Processing is in progress. FALSE: Processing is not in progress.
Done	BOOL	Communications processing normal end flag TRUE: Normal end / FALSE: Other than normal end
Error	BOOL	Communications processing error end flag TRUE: Error end / FALSE: Other than error end

[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
uErrorFlags	UNION	Error code processing union
BoolData	ARRAY[0..15] OF BOOL	2-byte error code is processed in units of 1 bit as 16-bit string. : TRUE (Error) / FALSE (Normal) •Communications error BoolData[0]: Send processing BoolData[1]: Receive processing •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 starts when this flag changes from FALSE to TRUE.
Output_RecvMess	STRING[256]	An area that stores the receive data (response) (256 bytes)
Output_ErrCode	WORD	An area that stores the error flag for a communications error or a timeout error that is detected at the send processing or 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 or receive processing. Normal end: 16#0000
Output_CmdsErrorIDEx	DWORD	An area that stores the extension error code for an error that is detected at the send processing or 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 variables 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 the 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]

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 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 the 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	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 BYTE	An area that stores the receive data of the SerialRcv instruction (2001 bytes)
Local_ReceiveMessage	STRING[256]	An area that stores the receive data after converted into a string. (256 characters)
Local_ReceiveSize	UINT	Size of receive data of SerialRcv instruction
Local_RecvDataLength	UINT	Total byte length of receive data
Local_RecvCHNo	UINT	The element number in Local_RecvData that stores the receive data
Local_RecvCheckFlg	BOOL	Destination device error detection instruction execution flag TRUE: Executed / FALSE: Not executed
Local_InitialSettingOK	BOOL	Initialization processing normal setting flag
Local_TONFlgs	sTimerControl	Timer enable flags This variable is defined as sTimerControl structure.
Local_ComType	sComType	Send/Receive processing required/not required setting flags This variable is defined as sComType structure.
Local_Port	_sPORT	Used port

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 the communications processing 1.2. Clearing the communications processing status flags 1.3. Communications processing in progress status	The communications processing starts.
2. Initialization processing	2.1. Initializing the 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 starts 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 starts when the receive processing required/not required setting is set to Required and the send processing ends normally. The receive processing is repeated when multiple receive data arrive. The receive data is stored and checked.
5. Processing number error process	5. Processing number error process	The error processing is executed 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 the 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 Communications Unit: CJ1W-SCU42 (No-protocol, Unit number: 0, Serial port number :2)
   Version: V1.00 New release 7 December 2012
          V1.01 Update 25 February 2013
   (C)Copyright OMRON Corporation 2012 All Rights Reserved.
 ============================================================= *)

(* 1. Communications processing
     Communications start switch: Input_Start
     Communications processing status flags: Local_Status<STRUCT>
       .Busy: Communications in progress
       .Done: Communications normal end
       .Error: Communications error end
      Processing number: Local_State
     10:Initialization processing
     11: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;  //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
   Perform 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 commnications 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#FFFFFFFF;
      Output_CmdsErrorID:=WORD#16#FFFF;
      Output_CmdsErrorIDEx:=DWORD#16#FFFFFFFF;

      (* 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 transition state
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;  //0: Communications not in progress status
ELSIF Local_ComType.Send THEN
   Local_State:=11;  //11: Send processing
ELSIF Local_ComType.Recv THEN
   Local_State:=12;  //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;  //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 transition state
      based on the send/receive processing required/not required flag *)
       IF Local_ComType.Recv THEN
          Local_State:=12;  //12: Receive processing
       ELSE
          Local_Status.Busy:=FALSE;
          Local_Status.Done:=TRUE;
          Local_State:=0;  //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;  //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);

            (* Reset the communications processing in progress flag *)
            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;  //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;  //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;  // 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;  //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 ends
      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.Recv ,
       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;  //0: Communications not in progress status

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

   END_CASE;

END_IF;
```

9.5.3. Detailed Description of the Function Blocks

The user-defined function blocks are shown below.

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

●ParameterSet function block

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

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

[Internal variables]

None

[Input/output]

Variable name	I/O	Data type	Description	
Execute	Input	BOOL	Execution flag: The function block is executed when this flag changes to TRUE and it is stopped when this flag 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	Not used (Not used in this program.)
Done	Output	BOOL	Normal end	
Error	Output	BOOL	Error end	
ErrorID	Output	WORD	Error information	
ErrorIDEx	Output	DWORD	Error information	

[External variables]

None

[Program]

```
(* ============================================================
   Name: NJ-series general-purpose serial no-protocol communications
         parameter setting function block
   Applicable device: OMRON Corporation ZW-series Displacement Sensor
   Version: V1.00 New release 7 December 2012
          V1.01 Update 25 February 2013
   (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 the interval at which multiple messages are 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 send data setting)

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

[Internal variables]

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

[Input/Output]

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

[External variable]

None

[Program]

```
(* ============================================================
   Name: NJ-series general-purpose serial no-protocol communications
   send data setting function block
   Applicable device: OMRON Corporation ZW-series Displacement Sensor
   Version: V1.00 New release 7 December 2012
          V1.01 Update 25 February 2013
   (C)Copyright OMRON Corporation 2012 All Rights Reserved.
 ============================================================ *)

IF Execute THEN

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

   (* Set the send data *)
   Send_Header:='';          // Header
   Send_Addr:='';            // Address (Station No.)
   Send_Command:='VR';       // Destination device command: VR
   Send_Check:='';           // FCS calculation: None
   Send_Terminate:='';       // Terminator: 'None'
                             // Presence or absence of SCU Unit end code
                             // Set CR(0x0D).

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

END_IF;

RETURN;
```

●RecieveCheck 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 receive data
Calc_Check	STRING[5]	FCS calculation value: FCS calculation result of receive data

[Input/Output]

Variable name	I/O	Data type	Description	
Execute	Input	BOOL	Execution flag: The function block is executed when this flag changes to TRUE and it is stopped when this flag changes to FALSE.	
tLength	Input	UINT	Receive data length: Byte length of 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 the destination device error code.	
Busy	Output	BOOL	Busy	Not used (Not used in this program.)
Rcv_Size	Output	INT	Storage receive data length: Data length of receive data	

[External variable]

None

[Program]

```
(* ================================================================
   Name: NJ-series general-purpose serial no-protocol communications
       receive processing function block
   Applicable device: OMRON Corporation ZW-series Displacement Sensor
   Version: V1.00 New release 7 December 2012
          V1.01 Update 25 February 2013
   (C)Copyright OMRON Corporation 2012 All Rights Reserved.
 ================================================================ *)

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

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

   (* Normal: The code does not start with 'ER'. *)
   ELSE
      Done:= TRUE;                     // Set normal end flag.
      Error:= FALSE;                   // Reset error flag.
      ErrorID:= WORD#16#0000;          // Clear error code.
      ErrorIDEx:= DWORD#16#00000000;   // Clear destination device error code.

   END_IF;

END_IF;

RETURN;
```

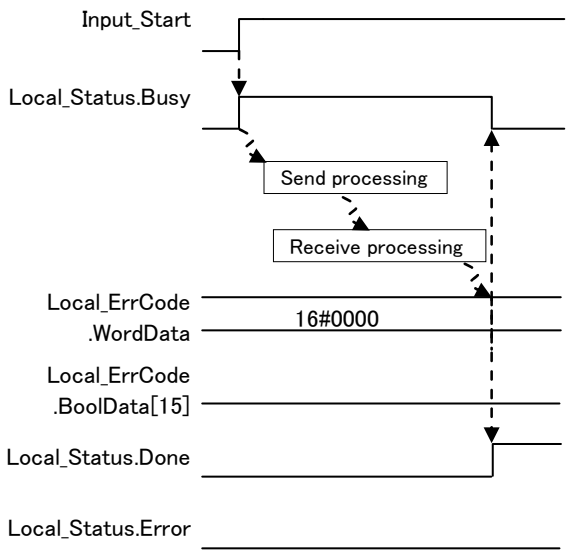
9.6. Timing Charts

This section explains the timing charts of the program.

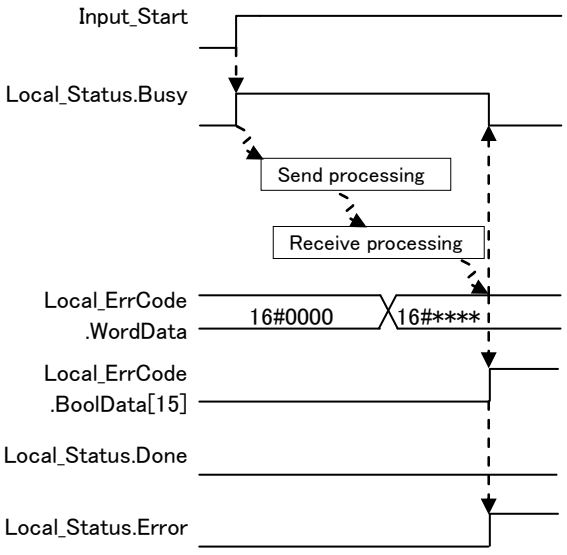
The definitions of the timing chart patterns are as follows:

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

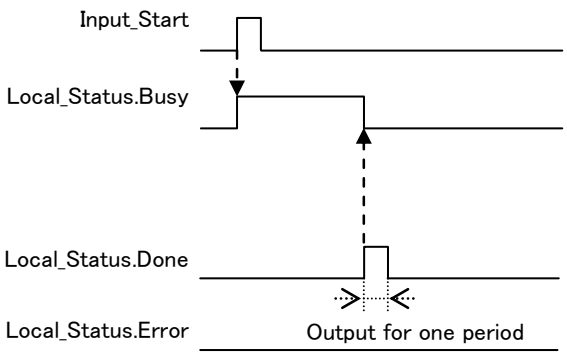


(Normal end)

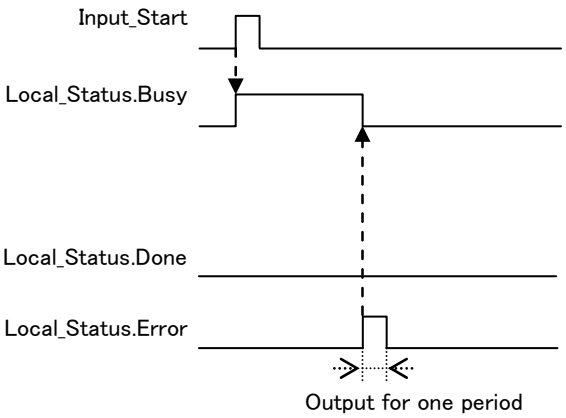


(Error end)

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

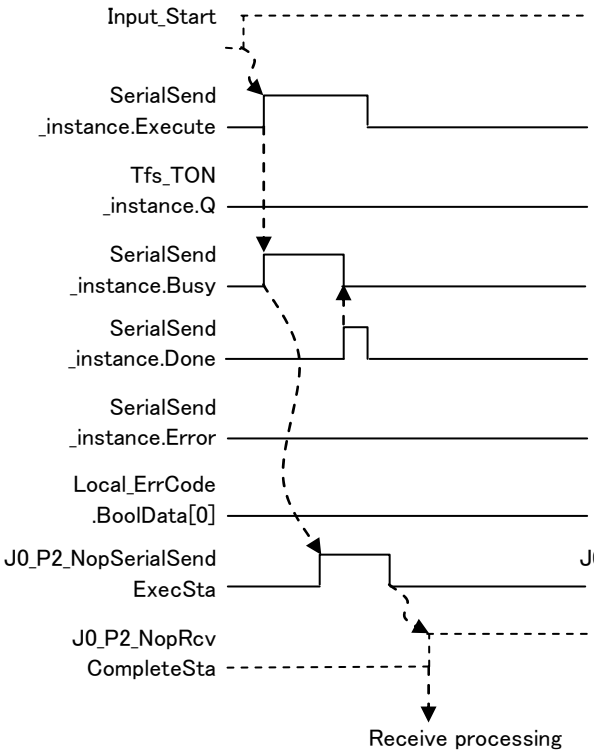


(Normal end)

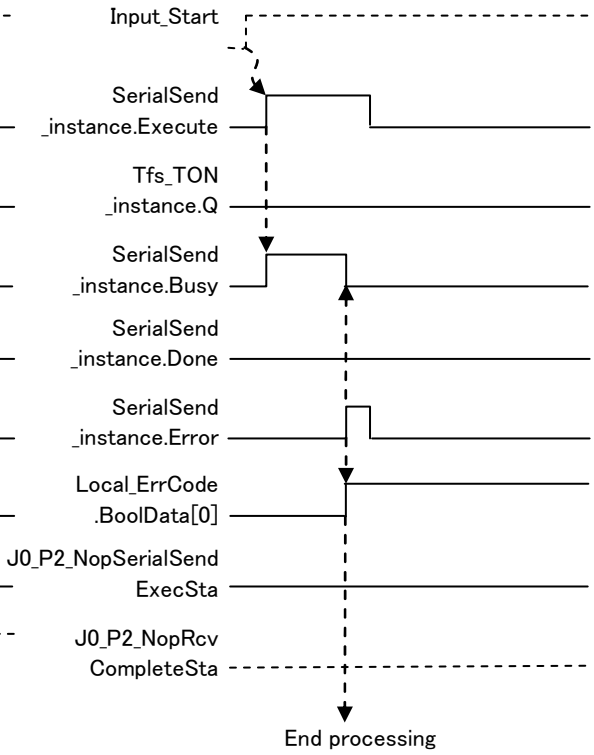


(Error end)

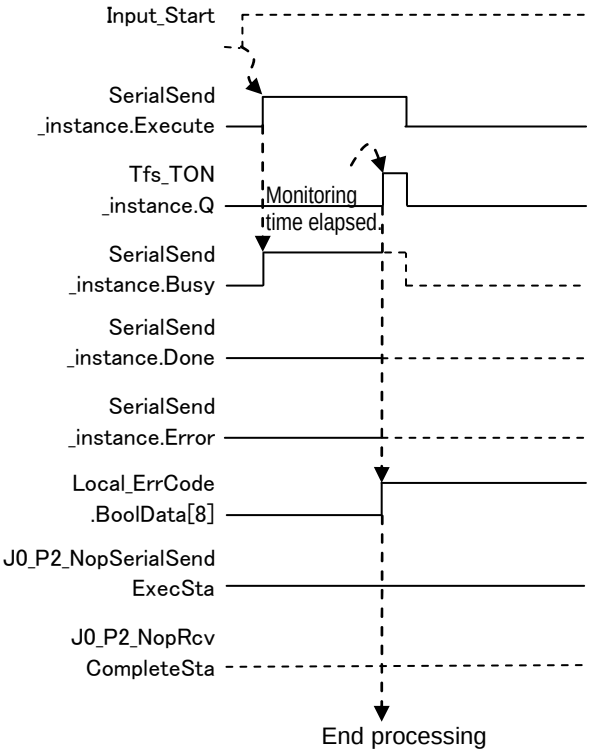
●Send processing



(Normal end)

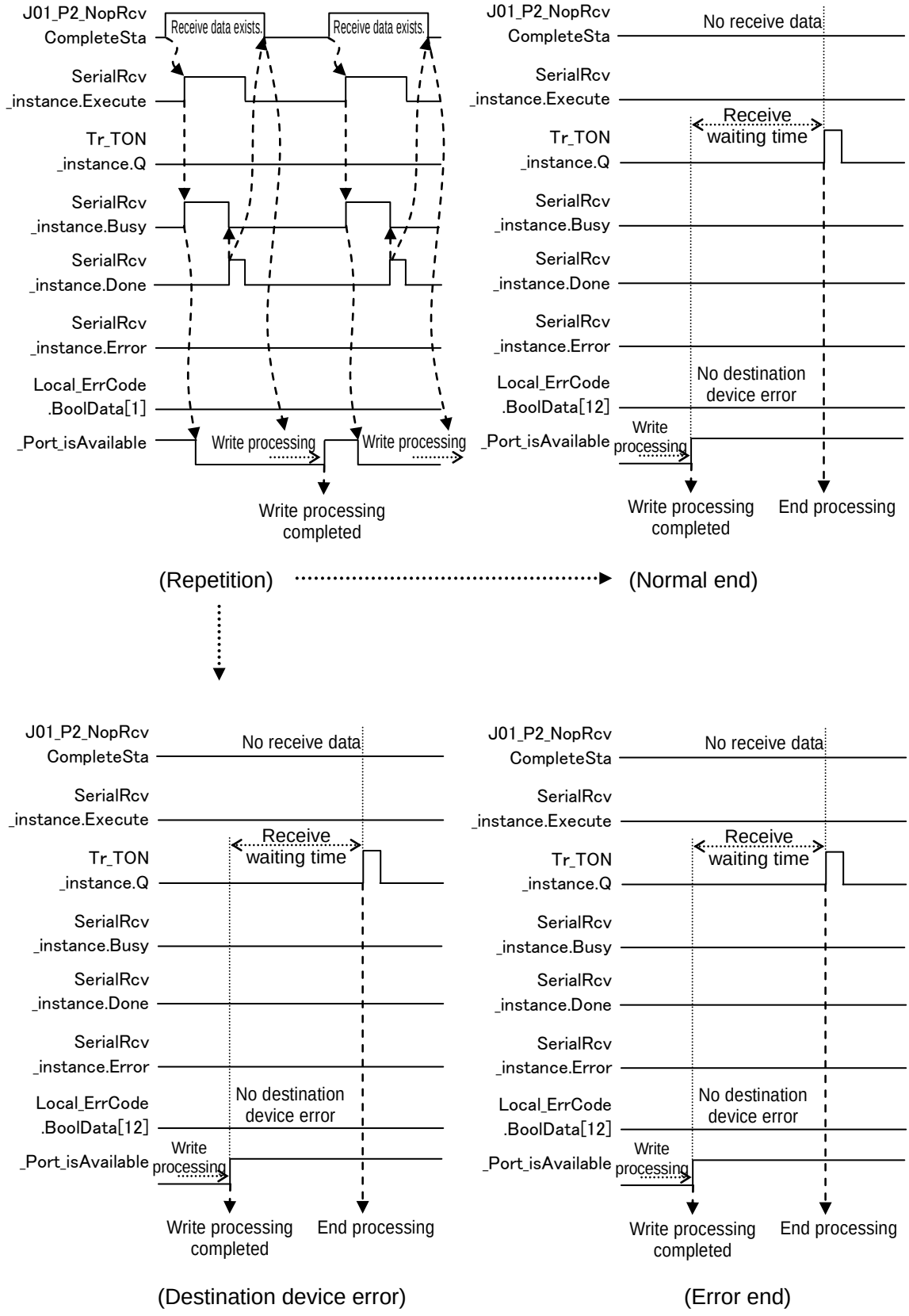


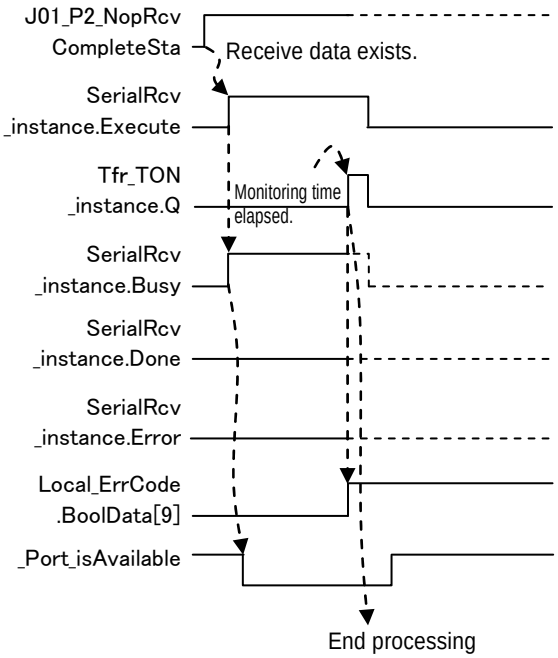
(Error end)



(Timeout)

●Send processing




J01_P2_NopRcv
CompleteSta
Receive data exists.
SerialRcv
_instance.Execute
Tfr_TON
_instance.Q
Monitoring time
elapsed.
SerialRcv
_instance.Busy
SerialRcv
_instance.Done
SerialRcv
_instance.Error
Local_ErrCode
.BoolData[9]
_Port_isAvailable
End processing

(Timeout)

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	The send processing ended in error. (Serial communications instruction error)
16#0002	The receive processing ended in error. (Serial communications instruction error)
16#0100	The send processing did not end in time. (Timeout error)
16#0200	The receive processing did not end in time. (Timeout error) (Including when an arrival of the response cannot be checked.)
16#0010	Processing number error
16#0020	Send/Receive required/not required detection error
16#1000	The response from the destination device is illegal. (Destination device error)
16#8000	Transmission error (Transmission error occurred.)

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

(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 transmission error status data and the destination device error.

Bit	Description	
15	1:Transmission error	0:Normal
14	(Not used)	
13	1:Destination device checksum error	0:Normal
12	1:Destination device error	0:Normal
5 to 11	(Not used)	
4	1:Overrun 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 Errors

The error codes used when the serial communications instructions (SerialSend instruction and SerialRcv instruction) end in error 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 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]

A destination device error code is stored in *Output_MErrCode*.

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

Code	Description
#16#00000000	Normal end
#16#45520000	The error response from the destination device (“ER” is received.)

Additional Information

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

10. Revision History

Revision code	Date of revision	Revision reason and revision page
01	Jul. 31, 2013	First edition

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

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

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

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

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

Authorized Distributor:

Cat. No. P559-E1-01 0911(-)