MITSUBISHI ELECTRIC

Changes for the Better

FATEC

Programmable Controllers Training Manual C Controller Basic Course

SAFETY PRECAUTIONS

(Read these precautions before exercise.)

When designing the system, always read the relevant manuals and give sufficient consideration to safety.

During the exercise, pay full attention to the following points and handle the product correctly.

[EXERCISE PRECAUTIONS]

⚠WARNING

- Do not touch the terminals while the power is on to prevent electric shock.
- Before opening the safety cover, turn off the power or ensure the safety.

⚠CAUTION

- Follow the instructor's direction during the exercise.
- Do not remove the module of the demonstration machine or change wirings without permission.
  Doing so may cause failures, malfunctions, personal injuries and/or a fire.
- Turn off the power before mounting or removing the module.
  Failure to do so may result in malfunctions of the module or electric shock.
- When the demonstration machine emits abnormal odor/sound, press the "Power switch" or "Emergency switch" to turn off.
- When a problem occurs, notify the instructor as soon as possible.

REVISIONS

*The manual number is given on the bottom left of the back cover.

Revision date	*Manual number	Description
March 2022	SH(NA)-082520ENG-A	First edition

TRADEMARKS

Microsoft and Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.
Unicode is either a registered trademark or a trademark of Unicode, Inc. in the United States and other countries.
The company names, system names and product names mentioned in this manual are either registered trademarks or trademarks of their respective companies.
In some cases, trademark symbols such as '™' or '®' are not specified in this manual.

CONTENTS

INTRODUCTION

To help users acquire the knowledge required for configuring a data collection system using the MES interface module, this manual describes the functions and specifications of hardware and software used to configure a system, explains the databases, and provides troubleshooting information.

RELEVANT MANUALS

Manual name [manual number]	Description	Available form
MELSEC iQ-R C Controller Module User's Manual (Startup) [SH-081366]	Performance specifications, procedures before operation, and troubleshooting of the C Controller module	e-Manual PDF
MELSEC iQ-R C Controller Module User's Manual (Application) [SH-081368]	Functions, devices, and parameters of the C Controller module	e-Manual PDF
MELSEC iQ-R C Controller Module Programming Manual [SH-081370]	Programming specifications and dedicated function library of the C Controller module	e-Manual PDF
MELSEC iQ-R C Controller/C Intelligent Function Module Programming Manual (Data Analysis) [SH-081755]	Programming specifications and dedicated function library relating to data analysis of the C Controller module/C Intelligent Function Module	e-Manual PDF
CW Workbench/CW-Sim Operating Manual [SH-081372]	System configuration, specifications, functions, and troubleshooting of CW Workbench/CW-Sim	e-Manual PDF
CW Configurator Operating Manual [SH-081381]	System configuration, parameter settings, and online operations of CW Configurator	e-Manual PDF

Point

e-Manual refers to the Mitsubishi Electric FA electronic book manuals that can be browsed using a dedicated tool.

e-Manual has the following features:

- Required information can be cross-searched in multiple manuals.
- Other manuals can be accessed from the links in the manual.
- The hardware specifications of each part can be found from the product figures.
- Pages that users often browse can be bookmarked.

MEMO

1 OVERVIEW

The C Controller module is a CPU module developed based on the multi-core ARM® and capable of executing multiple programs simultaneously. Featuring both robustness and timing accuracy, the C Controller module can serve as an alternative platform to personal computers or MCUs. Moreover, the fanless design adopted by the C Controller module, which prevents the spread of dust, is most suitable for use in a clean environment such as a microchip factory. Taking advantage of the excellent features of the MELSEC iQ-R series, such as high performance, flexibility, and robustness, the C Controller module realizes automation systems for a variety of industrial uses.

1.1 Features of the C Controller Module

Easy access to programmable controller devices with dedicated functions

Applications for handling programmable controllers, such as access to the C Controller module, I/O module, intelligent function module, network module, programmable controller CPU, and Motion CPU, can be created by using dedicated functions.
There are four types of dedicated functions: the CCPU/CITL functions, QBF functions, ISR (Interrupt Service Routine) QBF functions, and MD functions.

High-speed calculation realized by using function libraries

Processing that is difficult to achieve with a ladder program, such as FFT operations and digital filter operations used for vibration analysis and other processes, can be used as a function library.
Using this data analysis library, data analysis and judgment processing can be realized easily.
In addition, calculation can be performed at speeds higher than when the programmable controller CPU is used.

Easy connection with peripherals

Compatible with SLMP (MC protocol), which is a predefined protocol for programmable controllers, the C Controller module can read from/write to devices via the Ethernet ports.
Regardless of the communication destination device, data communications can be performed by the same communication method from an external device. In addition, the R12CCPU-V is compatible with CC-Link IE Field Network Basic as well.
CC-Link IE Field Network Basic is a network that can realize cyclic transmission by just implementing software, which allows various peripherals to be connected easily.

Checking made easy with LED indications

Without a personal computer, debugging, machine operating status check, and primary diagnostics when an error occurs can be performed easily by checking LED indications.
The dot matrix LED can display letters and symbols as well.

Easy maintenance without the need for battery replacement

Battery replacement is not required because MRAM (magnetoresistive random access memory) is adopted. Time and efforts associated with battery replacement can be saved, such as the recording of replacement dates, maintenance plan management, and the checking of ERR LED notifying a battery voltage drop.

Unauthorized access prevention by limiting access and setting account lockout

The C Controller Setting and Monitor Tool, FTP, login user for connection using telnet, and account lockout can be configured.
In addition, by setting access rights (read/write/execute) to each login user, a role such as the administrator and field operator can be set on a login user basis to prevent unauthorized access.

Security improved by stopping connection services

The states of services operating in the C Controller module can be set.
Whether to enable/disable services can be set with the C Controller Setting and Monitor Tool parameter setting to improve security.

Polling processing for waiting for processing completion not required

Event driven programming, which is common for MCUs/personal computers, is available.
CPU load reduction and high-speed response processing are realized because there is no need to wait for a processing completion signal from a module.

Load distribution realized by executing two tasks simultaneously

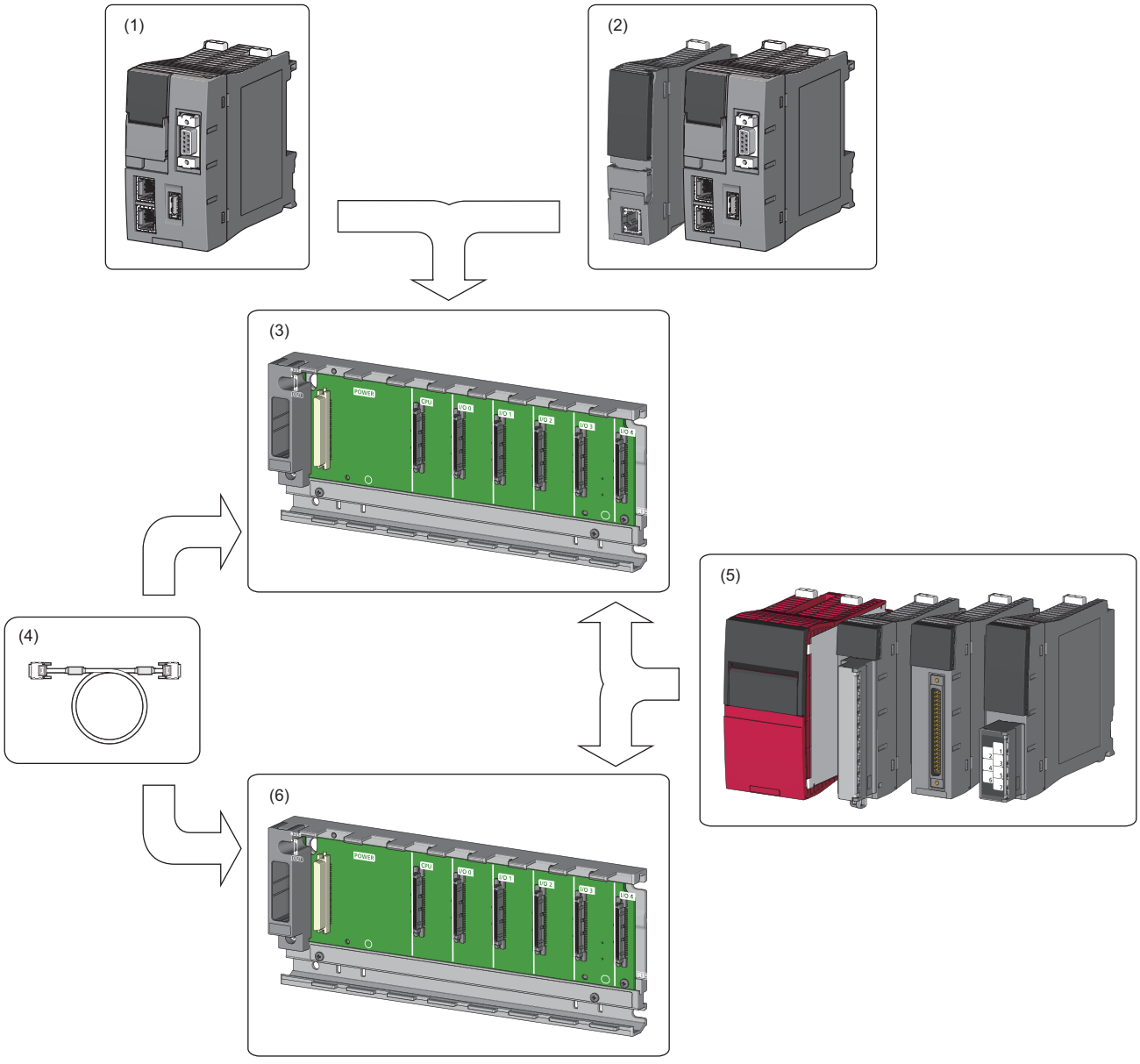
Two tasks can be executed simultaneously because a dual-core CPU is installed.
Since an interrupt service routine and a task can be executed simultaneously as well, CPU load distribution can be realized.

1.2 C Controller System Configuration

Select the C Controller module and modules with various functions according to the application, and mount them on the base unit, which operates as a backplane, to flexibly build a system.
For a C Controller system, a system can be flexibly and easily built, just by selecting necessary modules according to the application and mounting them on a base unit.

Overall System Configuration

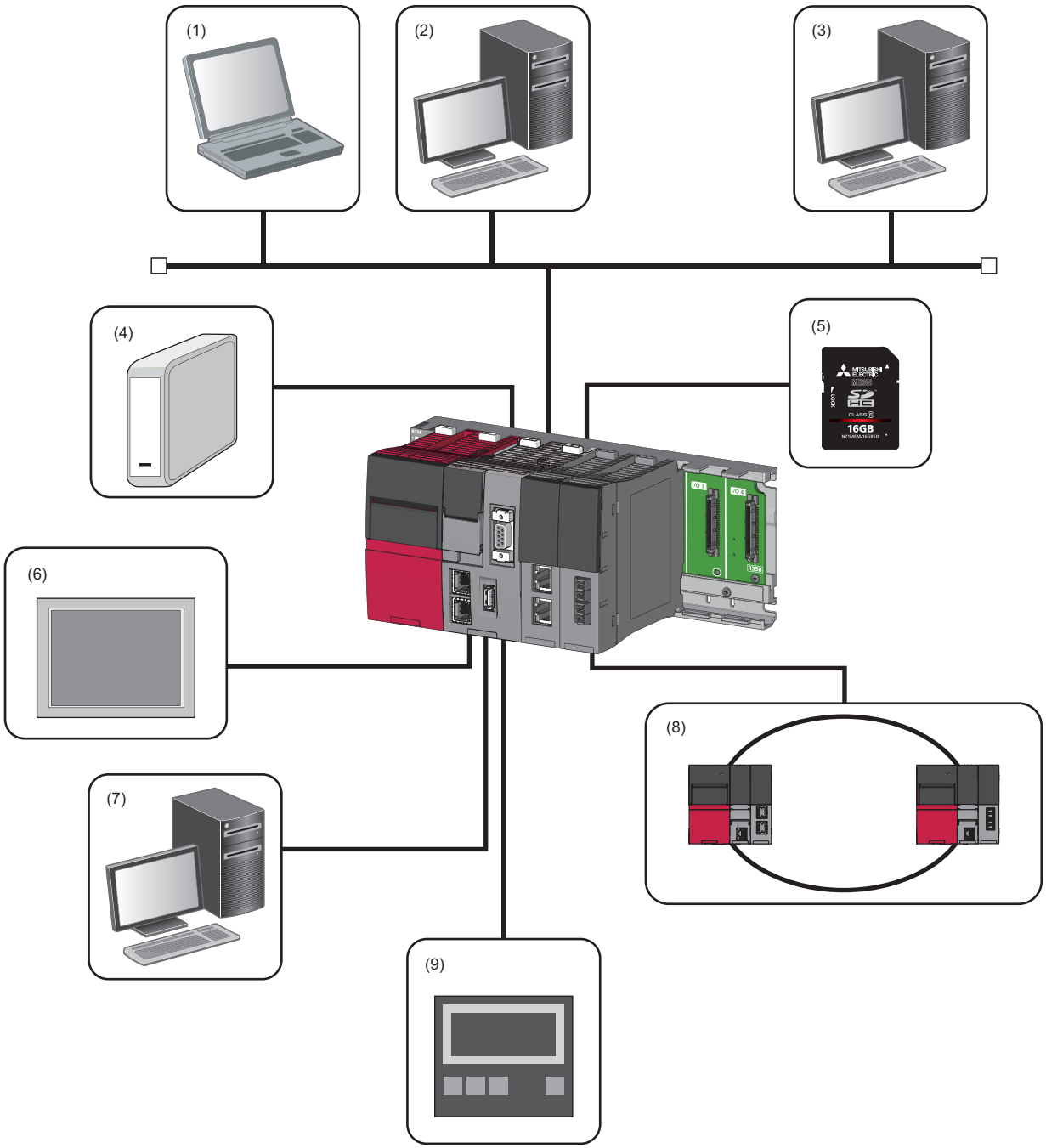
The following shows the overall C Controller system configuration.



(1) C Controller module
(2) Programmable controller CPU, process CPU, Motion CPU, and C Controller module
(3) Main base unit
(4) Power supply module, I/O module, and intelligent function module
(5) Extension base unit and RQ extension base unit

Peripheral configuration

The following figure shows a configuration with peripherals.



(1) Maintenance personal computer (Telnet function and FTP function)
(2) User program development environment (CW Workbench and CW-Sim)
(3) SNTP server
(4) USB Mass Storage Class compatible device
(5) SD memory card
(6) Connection via built-in Ethernet (HMI (Human Machine Interface) (GOT), SLMP-compatible device)
(7) CW Configurator
(8) Networks via network module (CC-Link IE Controller Network, CC-Link IE Field Network, MELSECNET/H Network, and CC-Link)
(9) Connection via built-in Ethernet (CC-Link IE Field Network Basic compatible device)

Point

- USB devices can be used for a firmware version of the C Controller module "03" or later.
- To install and connect a peripheral to the C Controller module, comply with the specifications of both the C Controller module and the peripheral.
- For details on access via each network module and access using Ethernet communication, refer to the following.

MELSEC iQ-R C Controller Module User's Manual (Application)

Applicable Software

The following table lists software that can be used for the MELSEC iQ-R C Controller module system. (Manual for each software used)

Software package		Version
CW Configurator	SW1DND-RCCPU-J	Version 1.00A or later
	SW1DND-RCCPU-E	
CW Workbench	SW1DND-CWWR-E/EZ/EVZ	Version 1.00A or later
CW-Sim	SW1DND-CWWSIMR-EZ	Version 1.00A or later
CW-Sim Standalone	SW1DND-CWWSIMSAR-E	Version 1.00A or later
Wind River Workbench	—	Version 3.3
GX Works3	SW1DND-GXW3-J	Version 1.007H or later
	SW1DND-GXW3-E	
GT Designer3	SW1DNC-GTWK3-J	Version 1.126G or later
	SW1DNC-GTWK3-E	
MT Works2	SW1DNC-MTW2-J	Version 1.110Q or later
	SW1DNC-MTW2-E	

SD Memory Card

One SD memory card can be inserted in the C Controller module.

SD memory cards that can be used

The following table lists the SD memory cards manufactured by Mitsubishi Electric that can be used.

Model	Description
NZ1MEM-2GBSD	SD memory card 2GB
NZ1MEM-4GBSD	SD memory card 4GB
NZ1MEM-8GBSD	SD memory card 8GB
NZ1MEM-16GBSD	SD memory card 16GB

For commercially available SD memory cards, refer to the Mitsubishi Electric FA global website. Before using a commercially available SD memory card, check that the card does not affect the control of the target system.
TECHNICAL BULLETIN No.FA-A-0023

Precautions

- Execute formatting for the SD memory card using the formatting function of CW Configurator.
- If an SD memory card other than those listed above is used, a problem, such as damage to data in the SD memory card and system operation stop, may occur.
- Data in the SD memory card may corrupt if any of the following operations is performed while the SD memory card is being accessed: powering off the system, resetting the C Controller module, or removing the SD memory card. Always power off or reset the C Controller module, or remove the SD memory card after access to the SD memory card stops.

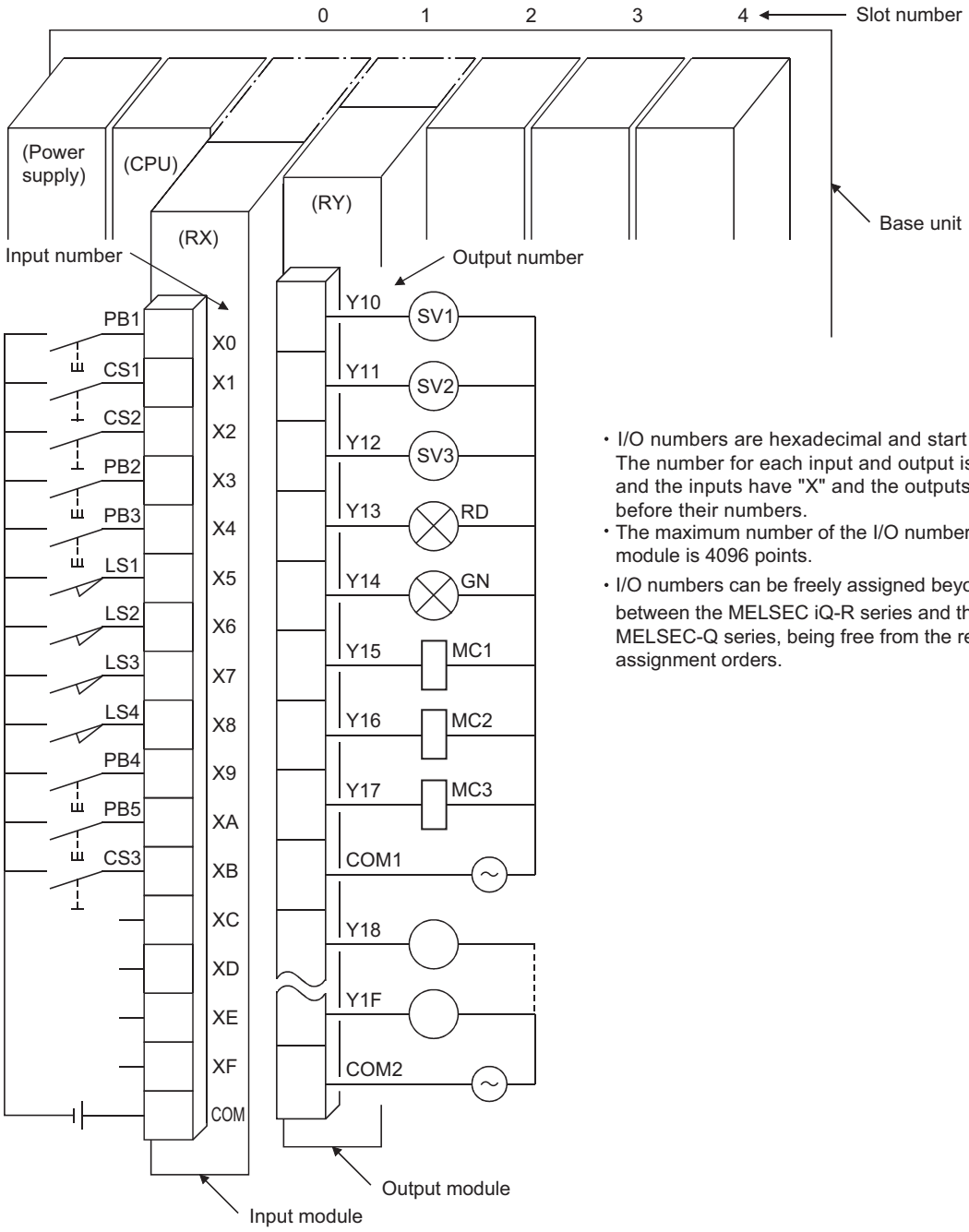
1.3 External I/O Signals and I/O Numbers

Point

Understand the rules for giving I/O numbers for the iQ-R series.

Wiring of I/O devices

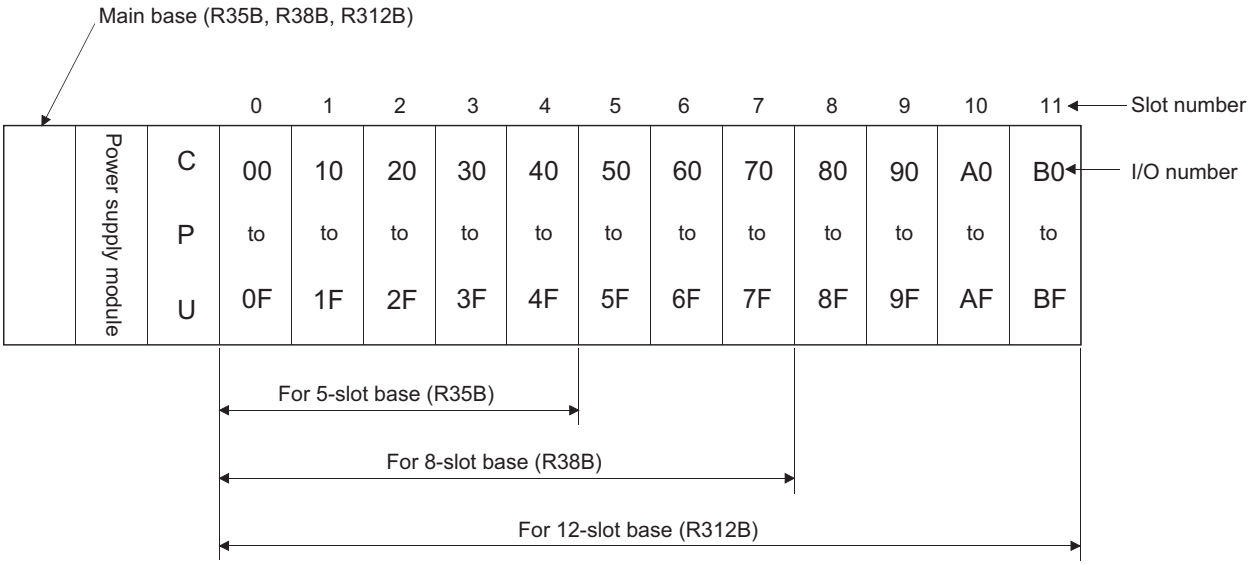
Signals from an external input device are handled in a program by being replaced with input numbers determined by the mounting position of the connected input module and terminal numbers. In addition, output numbers determined by the mounting position of the output module to which the external output device is connected and terminal numbers are used for outputs of a program's calculation results (coils).



- I/O numbers are hexadecimal and start from 0. The number for each input and output is common, and the inputs have "X" and the outputs have "Y" before their numbers.
- The maximum number of the I/O numbers for a CPU module is 4096 points.
- I/O numbers can be freely assigned beyond a boundary between the MELSEC iQ-R series and the MELSEC-Q series, being free from the restriction of assignment orders.

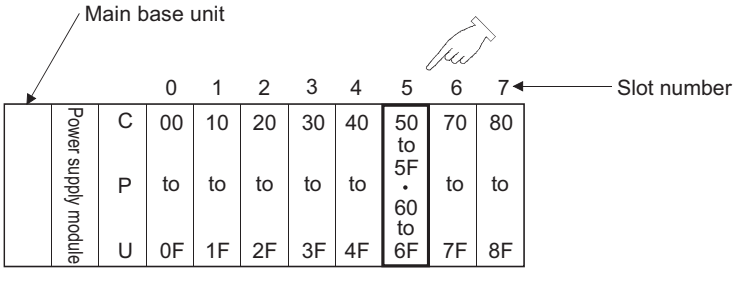
I/O numbers of the main base unit

The I/O numbers of the I/O module mounted on the main base unit are assigned as follows. The concept is the same for the I/O module and the intelligent function module.

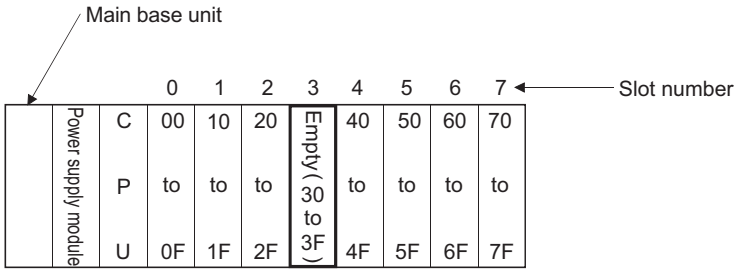


- I/O numbers for one slot (1 module) are assigned in units of 16 points (0 to FH), starting from the lowest number.
  In other words, basically a 16 point module is mounted on every slot.
  For example, the following figure shows I/O numbers when a 32 point module is mounted on slot 5.

When a module is replaced with a 32 point module, the numbers for the next module are changed. (The numbers are advanced.)



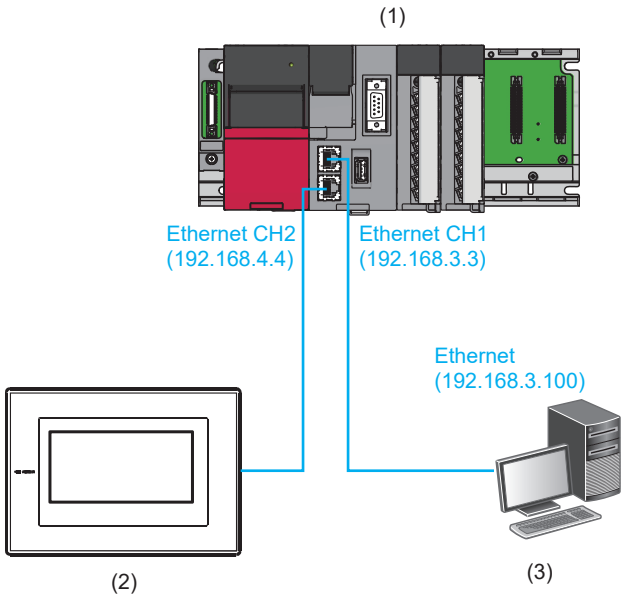
- I/O numbers are assigned to an empty slot (where an I/O module is not mounted) as well.
  For example, the following figure shows I/O numbers when slot 3 is empty. (Initial setting)
  The number of points to be assigned can be changed by the setting.



2 STARTING UP THE DEMONSTRATION MACHINE

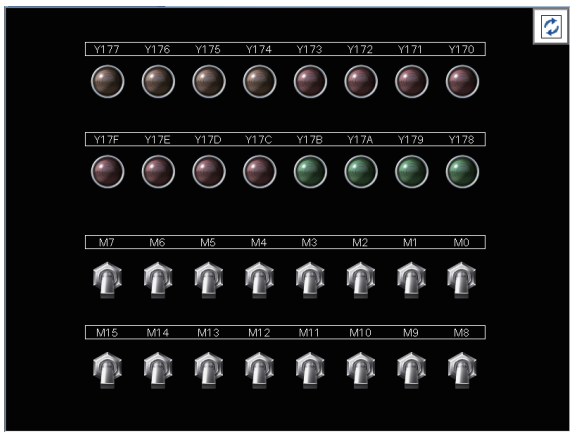
2.1 Demonstration Machine System Configuration

The following describes the system configuration of the demonstration machine.

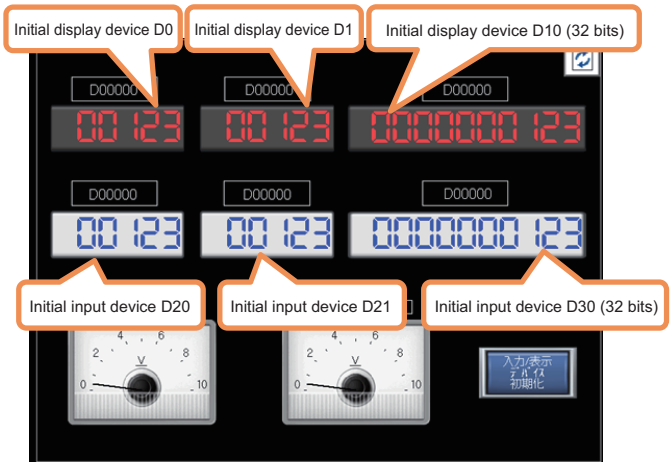


Device/software			Product model/description
(1)	PLC system	Main base unit	R35B
		Power supply module	R61P
		C Controller module	R12CCPU-V
		A/D converter module	R60AD4
		D/A converter module	R60DA4
(2)	GOT2000		GT2708-STBA
(3)	Personal computer		Personal computer running Windows
	Engineering tool	CW Configurator	SW1DND-RCCPU-J
		CW Workbench	SW1DND-CWWR-E

GOT screen display



Display 1



Display 2

Upper row: The display device is changeable

Lower row: Displays data

Upper row: The input device is changeable

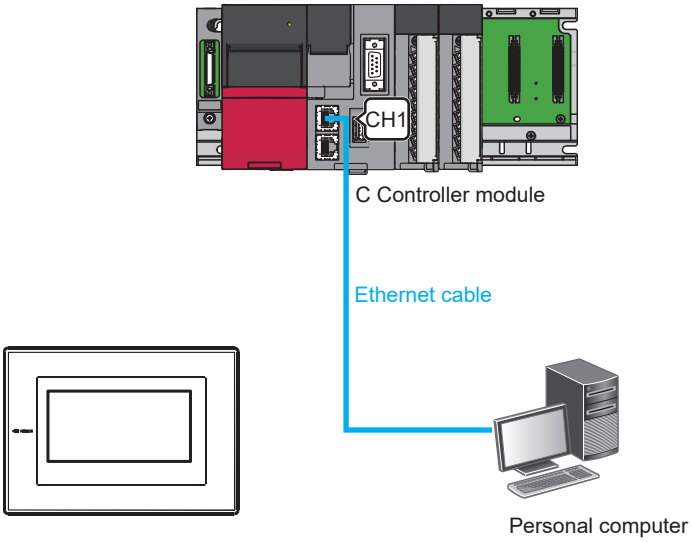
Lower row: Sets/ displays input data

- Touch  to switch screens.
- Touch the [Initialize input/display device] button to initialize the device number at top.

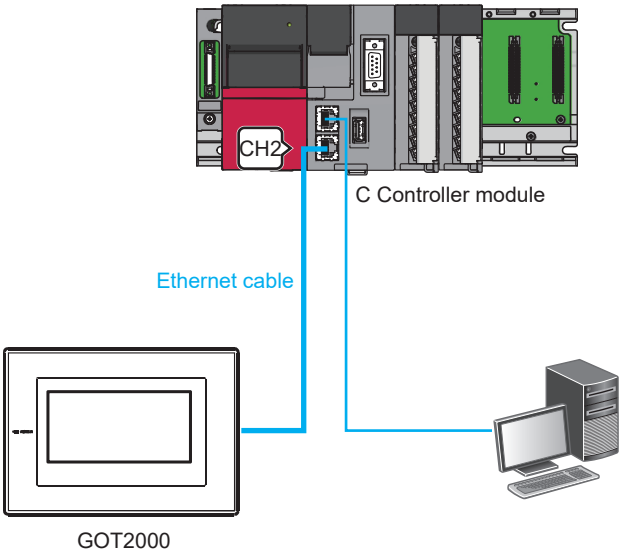
2.2 Wiring the Demonstration Machine

This section describes the procedure for wiring the demonstration machine.

1. Connect the C Controller module (CH1) and the personal computer using an Ethernet cable.



2. Connect the C Controller module (CH2) and the GOT using an Ethernet cable.



2.3 Procedures Before Operation

This section describes the procedure from startup of the C Controller module to program execution.
When operating the C Controller module for the first time, perform hardware diagnostics to check that the module has no problem before starting up the system.

Executing hardware diagnostics

Operating procedure

1. Mounting the C Controller module

Mount the power supply module and C Controller module on the base unit. (📖 MELSEC iQ-R Module Configuration Manual)

2. Powering on the system

Check that the wiring of the power supply and the power supply voltage are correct, and turn the power on.

3. Initializing the C Controller module

Initialize the C Controller module. (☞ Page 19 Executing initialization)

4. Executing hardware diagnostics

Check the hardware status of the C Controller module. (☞ Page 22 Executing hardware diagnostics)

Starting up the C Controller system

Operating procedure

1. Inserting the SD memory card

Insert the SD memory card to the C Controller module as necessary. (📖 MELSEC iQ-R C Controller Module User's Manual (Startup))

2. Mounting modules and connecting cables

Mount modules on the base unit and connect cables. (📖 MELSEC iQ-R Module Configuration Manual)

3. Powering on the system

Check the following items before powering on the system.

- The wiring of the power supply and the power supply voltage are correct.
- The C Controller module is in the STOP state.

4. Creating a project

On the personal computer with CW Configurator installed, create a project for the C Controller module to be used.
(☞ Page 37 Parameter Settings)

5. Connecting the personal computer and the C Controller module

Connect the personal computer with CW Configurator installed and the C Controller module. (☞ Page 44 Specifying the connection destination)

6. Setting parameters

Set system parameters, CPU parameters, and module parameters. (☞ Page 37 Parameter Settings)
Parameters other than the above need to be set to use the SD memory card function or to mount an intelligent function module. (📖 User's Manual (Application) for the module used)

7. Writing to the C Controller module

Write the parameters set with CW Configurator to the C Controller module. (☞ Page 46 Writing parameters to the C Controller module)

8. Resetting the C Controller system

Reset the system in either of the following ways.

- Power off and on the system.
- Reset the C Controller module (☞ Page 21 Reset operation procedure).

9. Checking for errors

Check the status of the READY LED and ERROR LED of the C Controller module. If an error has occurred, perform troubleshooting. If an error has occurred in a module other than the C Controller module, refer to the manual for each module.

10. Creating a user program

Create a user program.

- Create a user program and perform debugging. (☞ Page 137 Creating a new user program)
- Create a script file. (☞ Page 82 Creating and storing a script file)
- Register the user program and script file into the C Controller module. (☞ Page 82 Creating and storing a script file)

11. Resetting the C Controller system

Reset the system in either of the following ways.

- Power off and on the system.
- Reset the C Controller module (☞ Page 21 Reset operation procedure).

12. Executing the program

Run the C Controller module and check that the BUS RUN LED turns on.

Executing initialization

Initialize the C Controller module in any of the following cases.

- The system operates for the first time.
- The C Controller module does not start up by executing the script file registered in the program memory.
- The user name/password set to the C Controller module is forgotten.

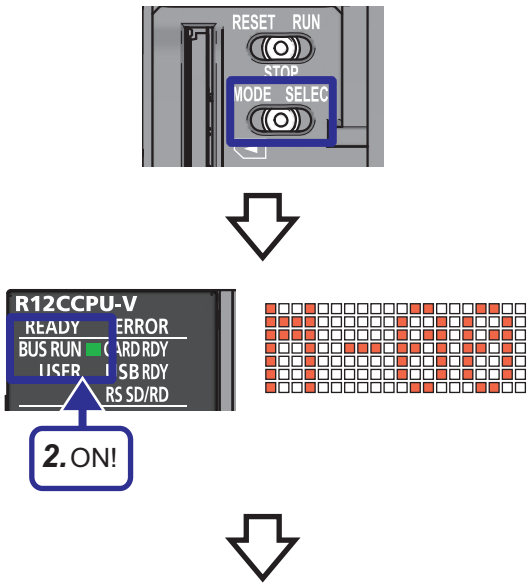
Point

When initialization is executed, the program memory, device/label memory, and data memory data are deleted. Backup necessary data before initialization.

Initialization execution procedure

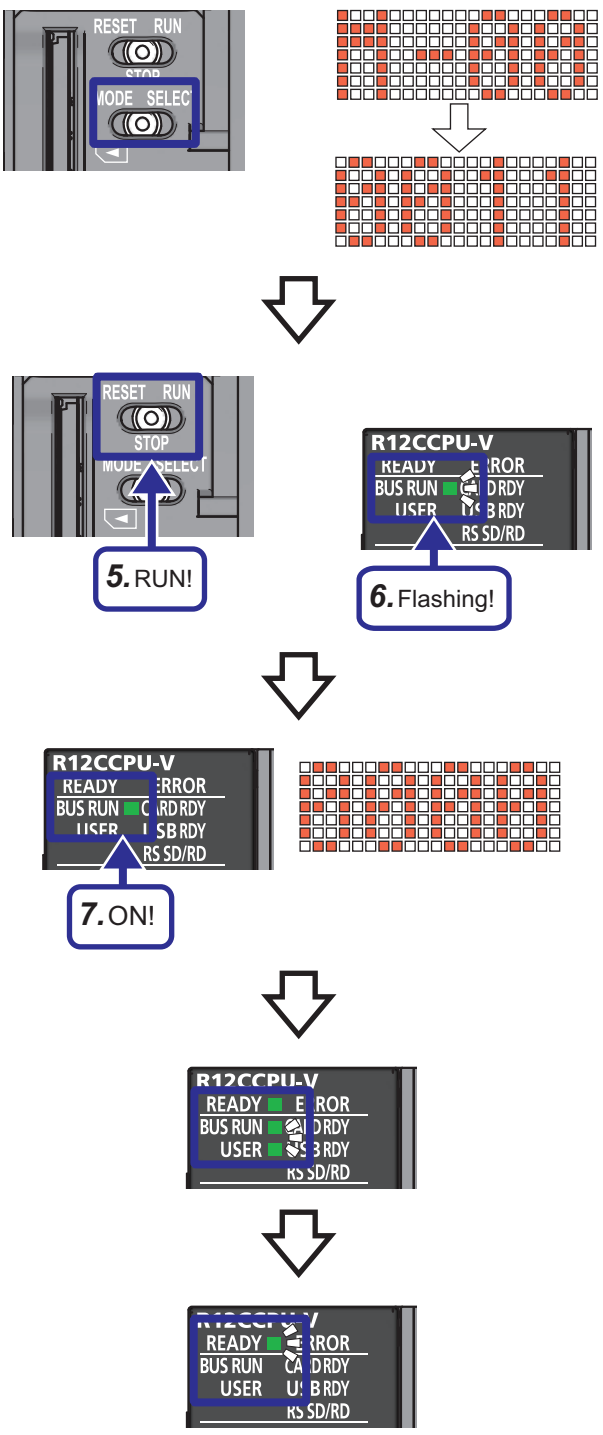
Before switch operation, check that the RESET/STOP/RUN switch is in the middle position.

Operating procedure



1. Hold the MODE/SELECT switch in the MODE position.

2. Power on the C Controller module. The BUS RUN LED turns on and "M-00" is displayed on the dot matrix LED.



3. Release the MODE/SELECT switch and put it back to the center position.

4. Set the MODE/SELECT switch to the SELECT position. Every time the switch is set to the SELECT position, the value of mode displayed on the dot matrix LED is changed. Repeat the action until "0011" is displayed.

5. Set the RESET/STOP/RUN switch to the RUN position to execute the selected mode.

6. The BUS RUN LED flashes during initialization.

7. Check that the BUS RUN LED turns on and "0000" is displayed on the dot matrix LED, then reset the C Controller module.

8. Initialization is executed after the module is reset. During initialization, the READY LED turns on and the BUS RUN LED and USER LED flash.

9. When initialization is completed successfully, the BUS RUN LED and USER LED turn off and the READY LED flashes.

10. Reset the C Controller module.

☞ Page 21 Reset operation procedure

■Modes that can be selected

Mode	Dot matrix LED indication	Description
10	0010	Default IP setting
11	0011	Module initialization setting

If initialization ends abnormally

If the initialization of the C Controller module ends abnormally, the ERROR LED flashes and the READY LED and USER LED turn on. If the initialization ends abnormally, initialize the C Controller module again.

Point

Do not reset the C Controller module while it is being initialized.
If the C Controller module is reset mistakenly, initialize it again.

Module state after initialization

When initialization is executed, the C Controller module is in the following state.

■Default IP setting

- The execution of the registered script file is stopped.*1
- The initial value is set for the IP address of the C Controller module.

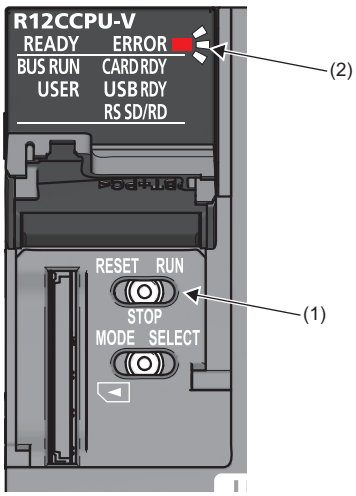
*1 The script file is renamed as "STARTUP.BAK" and its registration is cleared.

■Module initialization setting

- The default parameters are set for the data memory.
- The program memory, device/label memory, and data memory data are formatted.
- The security password is initialized. (Default password: password)

Reset operation procedure

Operate the switch according to the following procedure to reset the C Controller module.



1. Hold the RESET/STOP/RUN switch (1) in the RESET position.
2. Check that the ERROR LED (2) flashes several times and all the LEDs turn off.
3. Set the RESET/STOP/RUN switch (1) back to the STOP position.

Point

Do not reset the C Controller module while it is being initialized.
If the C Controller module is reset mistakenly, initialize it again.

Executing hardware diagnostics

Diagnose the hardware in the C Controller module.

Timing of hardware diagnostics

Use hardware diagnostics in the following cases.

- The system operates for the first time.
- Troubleshooting

Point

During hardware diagnostics, do not power off or reset the C Controller module. Otherwise, the C Controller module cannot start up correctly. In such a case, perform initialization.

Diagnostics types

The following table shows the modes of hardware diagnostics.

Mode	Dot matrix LED	Item	Description
0	M-00	Mode 1 to 6 diagnostics tests	Execute a diagnostics test for each mode in the order of mode numbers, starting from 1 to 6.
1	M-01	Program memory and data memory diagnostics test	Read data from the program memory and data memory, and perform error detection.
		Device/label memory diagnostics test	Check the device/label memory by reading/writing/verifying test data.
2	M-02	Ethernet diagnostics test	Diagnose the condition of the Ethernet ports (CH1 and CH2).
3	M-03	SD memory card interface diagnostics test	Diagnose the condition of the SD memory card slot.
4	M-04	RS-232 diagnostics test	Execute a self-loopback test for the RS-232 connector. Prepare wiring for self-loop back operation.
5	M-05	USB diagnostics test	Diagnose the condition of the USB connector.
6	M-06	Bus diagnostics test	Check the internal bus memory and registers by reading/writing/verifying test data.
7	M-07	Dot matrix LED test	Test the display condition of the dot matrix LED.

Executing diagnostics

This section describes the execution procedure for hardware diagnostics.

■Preparation

Execute the following before hardware diagnostics.

1. Mount the module.

Mount the power supply module and C Controller module on the base unit.

Point

Hardware diagnostics can be executed even when the C Controller module is not mounted on the CPU slot. To execute hardware diagnostics by mounting the C Controller module on a slot other than the CPU slot, mount another CPU module on the CPU slot. An error may be detected in the mounted CPU module, but hardware diagnostics can be executed.

2. Check the wiring.

- Check that the wiring for the power supply is correct.
- Use only cables for power supply wiring.

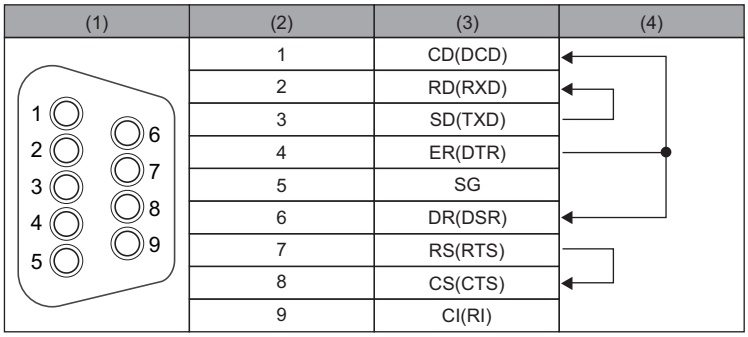
3. Make preparations required for each type of diagnostics.

4. Turn the power on.

- Check that the power supply voltage is within the range of the specifications.
- Check that the RESET/STOP/RUN switch is in the STOP position.

■Preparations required for each type of diagnostics

Execute the following before executing each mode (0 to 7) of hardware diagnostics.

Mode	Description
0	Make all the preparations required for executing modes 1 to 6.
1	Back up the data of the program memory, data memory, and device/label memory.
2	Check that a cable is not connected to the Ethernet ports.
3	Check that an SD memory card is not inserted.
4	Connect a cable to the RS-232 connector. The connector pin-outs and cable connections are as follows.  (1) Connector (2) Pin number (3) Signal abbreviation (4) Cable connection
5	Check that a cable is not connected to the USB connector.
6	Preparations for this mode are not required.

■Selecting a mode

The following describes how to select a mode.

1. Set the RESET/STOP/RUN switch to the RESET position and hold it in that position until step 4. Check that all the LEDs turn off.
2. Set the MODE/SELECT switch to the MODE position and hold it in that position until step 6.
3. Set the RESET/STOP/RUN switch back to the STOP position.
4. The BUS RUN LED turns on and "M-00" is displayed on the dot matrix LED.
5. Release the MODE/SELECT switch and put it back to the center position.
6. Set the MODE/SELECT switch to the SELECT position and select a diagnostics mode.

Every time the switch is set to the SELECT position, the displayed value of diagnostics mode is changed.
Repeat the action until the value of the target diagnostics mode is displayed on the dot matrix LED.

■Executing the mode

The following describes how to execute the selected mode.

●Executing modes 0 to 6

1. Set the RESET/STOP/RUN switch to the RUN position.

- While modes 0 to 1 are executed, the BUS RUN LED flashes, and the mode in progress and its progress ratio are displayed in turn on the dot matrix LED.

Example: "M-01" (diagnostics mode) ← displayed in turn → "0050" (progress rate: %)

- While modes 2 to 6 are executed, the BUS RUN LED flashes, and the mode in progress is displayed on the dot matrix LED.

2. Check that the BUS RUN LED turns on.

When the test is completed successfully, "0000" is displayed on the dot matrix LED.

3. Turn the power off.

Point

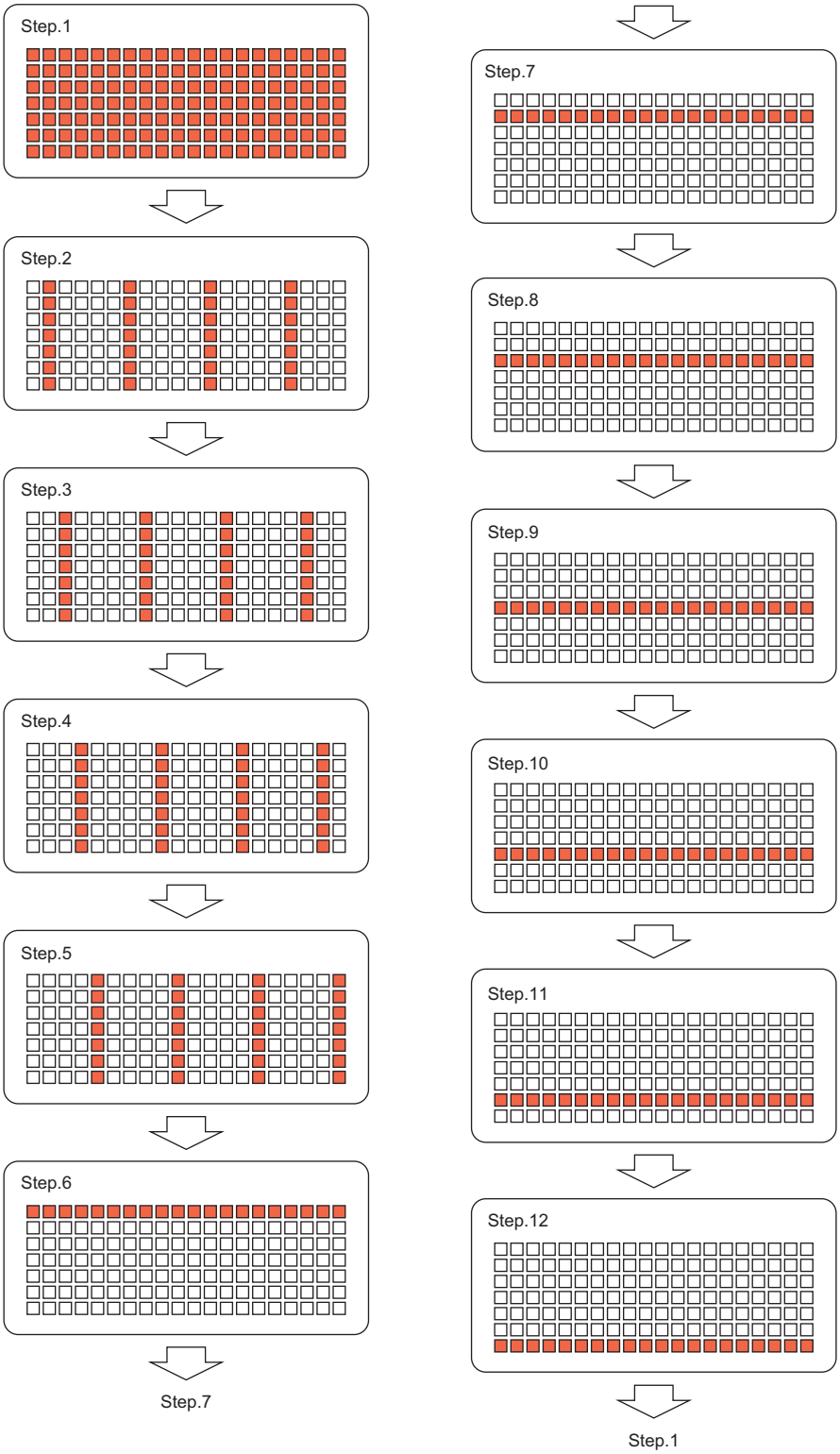
When the RESET/STOP/RUN switch is in the RUN position before taking the procedure for mode execution, set the switch back to the STOP position.

●Executing mode 7

1. Set the RESET/STOP/RUN switch to the RUN position.

2. Set the MODE/SELECT switch to the SELECT position.

Every time the switch is set to the SELECT position, the status of the dot matrix LED is changed.



■: On

□: Off

3. Turn the power off.

Point

Visually check that all the dots on the dot matrix LED turn on.
If some dot does not turn on, the possible cause is a hardware failure of the C Controller module. Please consult your local Mitsubishi Electric System & Service Co., Ltd. representative.

Operation upon error detection

This section describes the error content displayed when an error is detected.

■When "0000" is not displayed on the dot matrix LED

When an error is detected during diagnostics and setting operation, the ERROR LED flashes, and the value corresponding to the type of diagnostics in which an error occurred is displayed on the dot matrix LED. If nothing is displayed on the dot matrix LED and only the ERROR LED flashes, it is a system error.

Mode	Dot matrix LED	Type of diagnostics in which an error occurred
1	E010	Program memory and data memory diagnostics test Device/label memory diagnostics test
	E020	
	E030	
	E040	
2	E050	Ethernet diagnostics test (CH1)
	E060	Ethernet diagnostics test (CH2)
3	E070	SD memory card interface diagnostics test
4	E080	RS-232 diagnostics test*1
5	E090	USB diagnostics test
6	E0A0	Bus diagnostics test
	E0B0	
	E0C0	
	E0D0	
	E0E0	

*1 An error may occur if a cable for wiring is not connected correctly. Correctly connect the cable or do the wiring, and execute mode 4 diagnostics again.

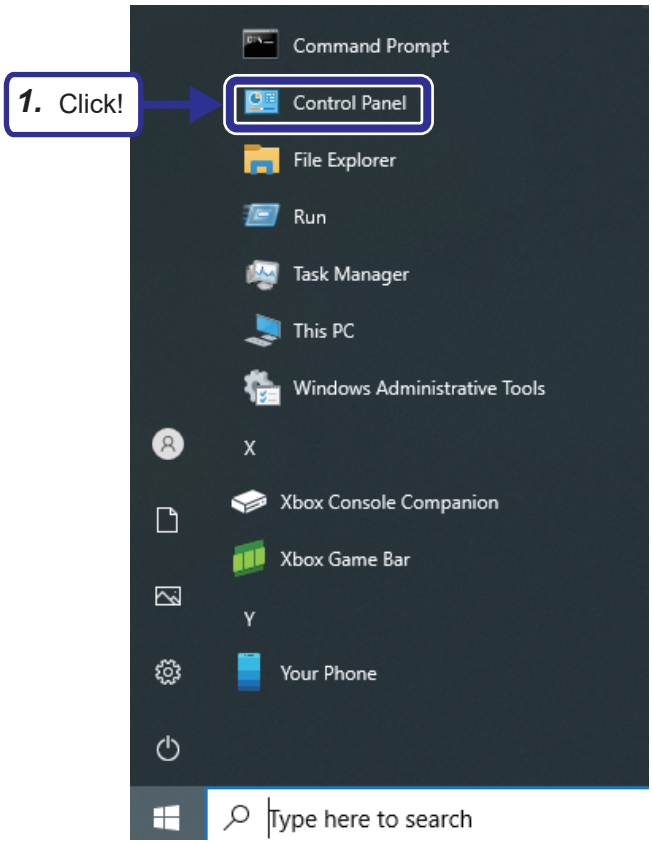
Point

If an abnormal end occurs, diagnostics or setting operation after that is not executed. Execute diagnostics and setting operation again. If an error occurs again after an abnormal end, the possible cause is a hardware failure of the C Controller module. Please consult your local Mitsubishi Electric System & Service Co., Ltd. representative.

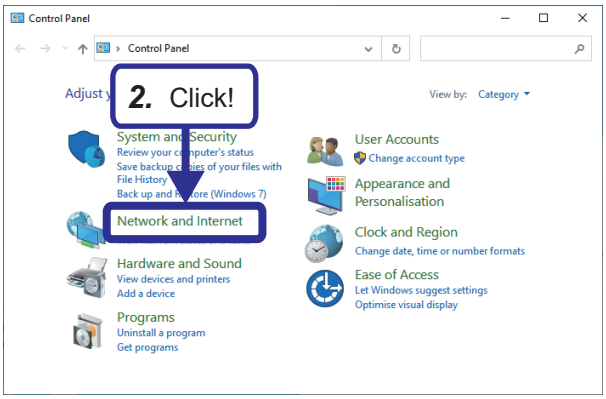
2.4 TCP/IP setting on the personal computer

Configure the TCP/IP setting on the personal computer.
This manual describes the procedure using a personal computer running Microsoft® Windows® 7 Operating System.

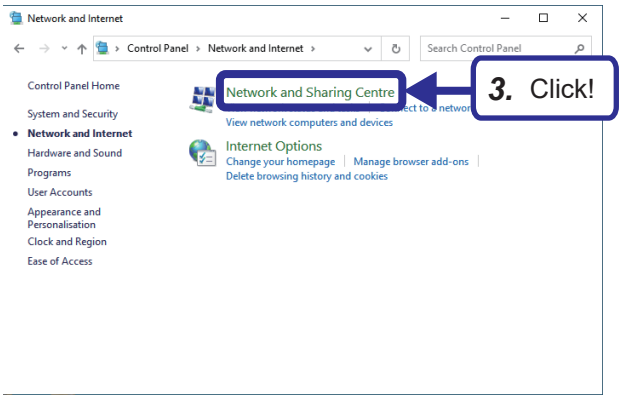
Operating procedure



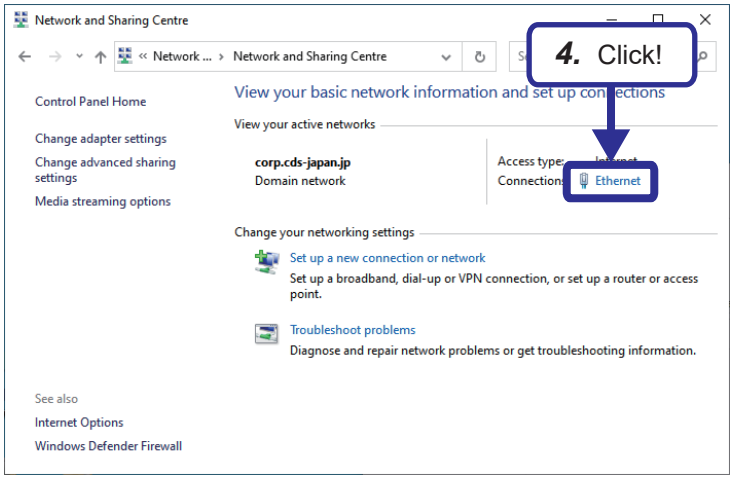
1. From the Start menu of Windows®, click [Windows System] ⇨ [Control Panel].



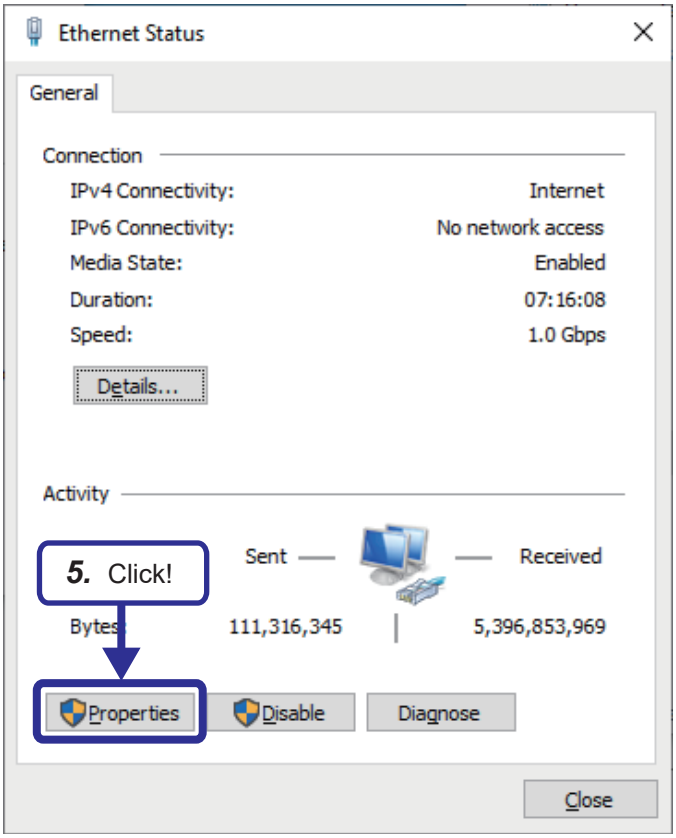
2. The Control Panel dialog box appears. Click "Network and Internet".



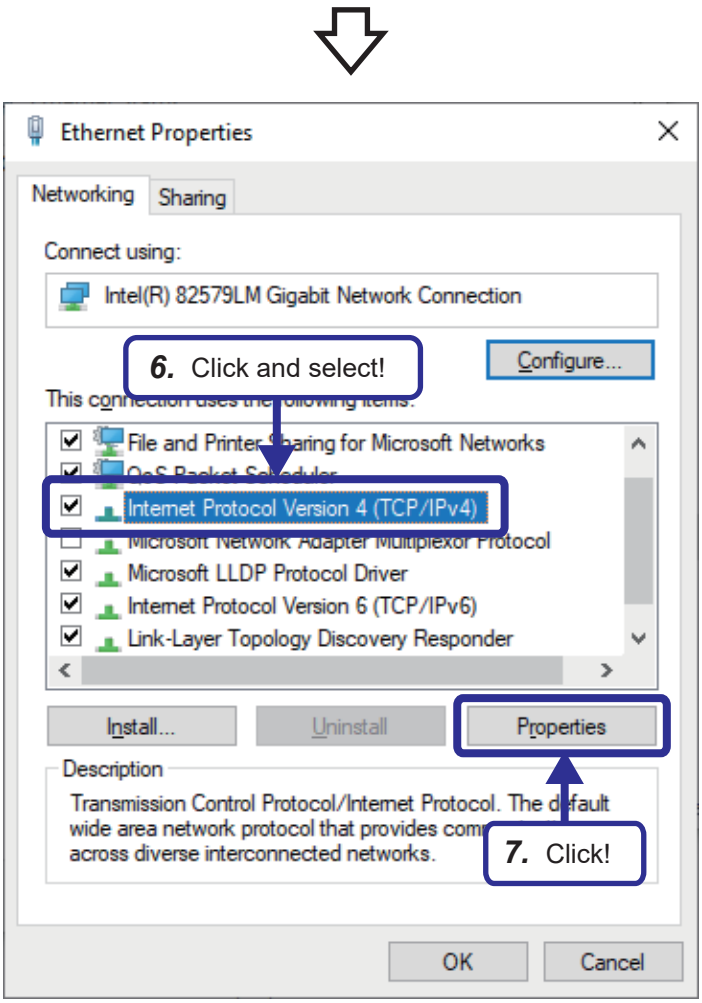
3. Click "Network and Sharing Centre".



4. Click "Ethernet".

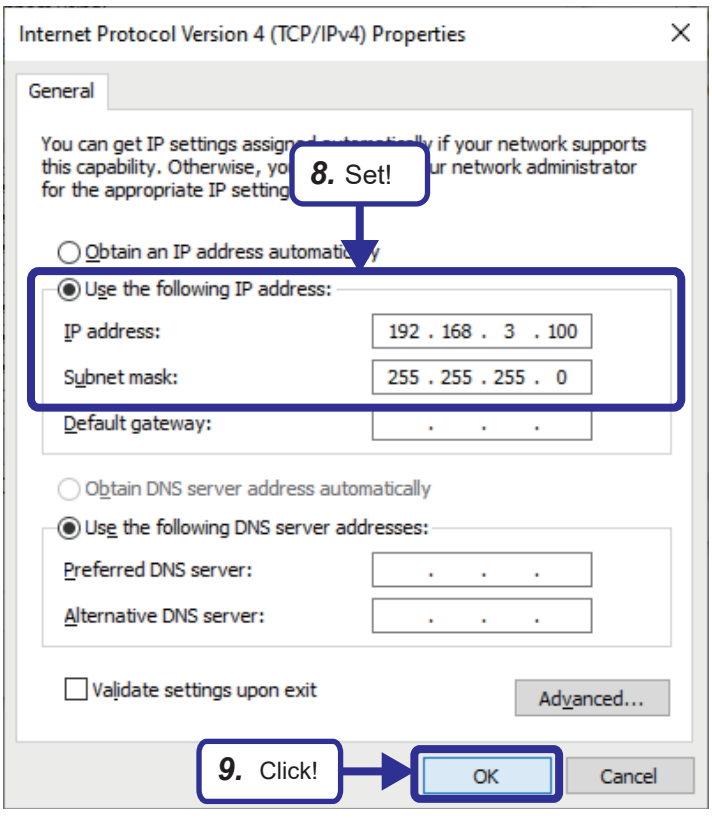


5. Click the [Properties] button.



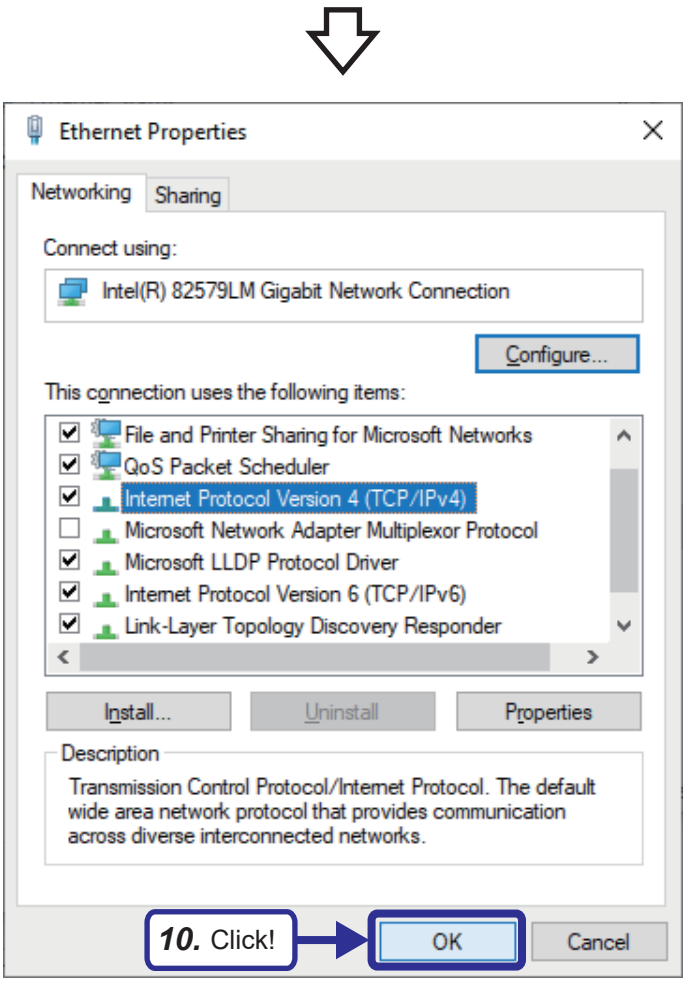
6. Select "Internet Protocol Version 4 (TCP/IPv4)".

7. Click the [Properties] button.

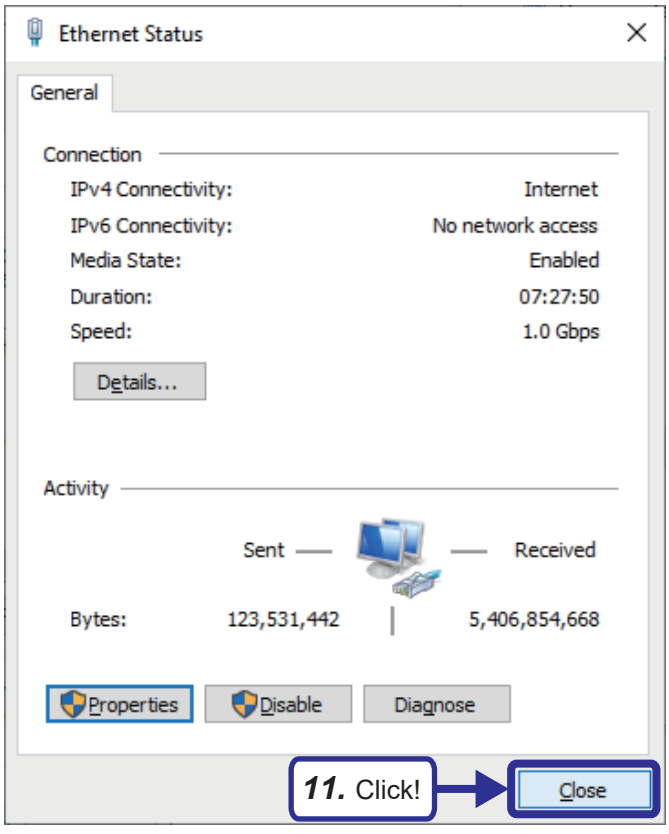


8. Select "Use the following IP address" and configure the settings as follows.
[Setting details]
IP address: 192.168.3.100
Subnet mask: 255.255.255.0

9. Click the [OK] button.



10. Click the [OK] button.



11. Click the [Close] button.

3 OPERATING CW CONFIGURATOR

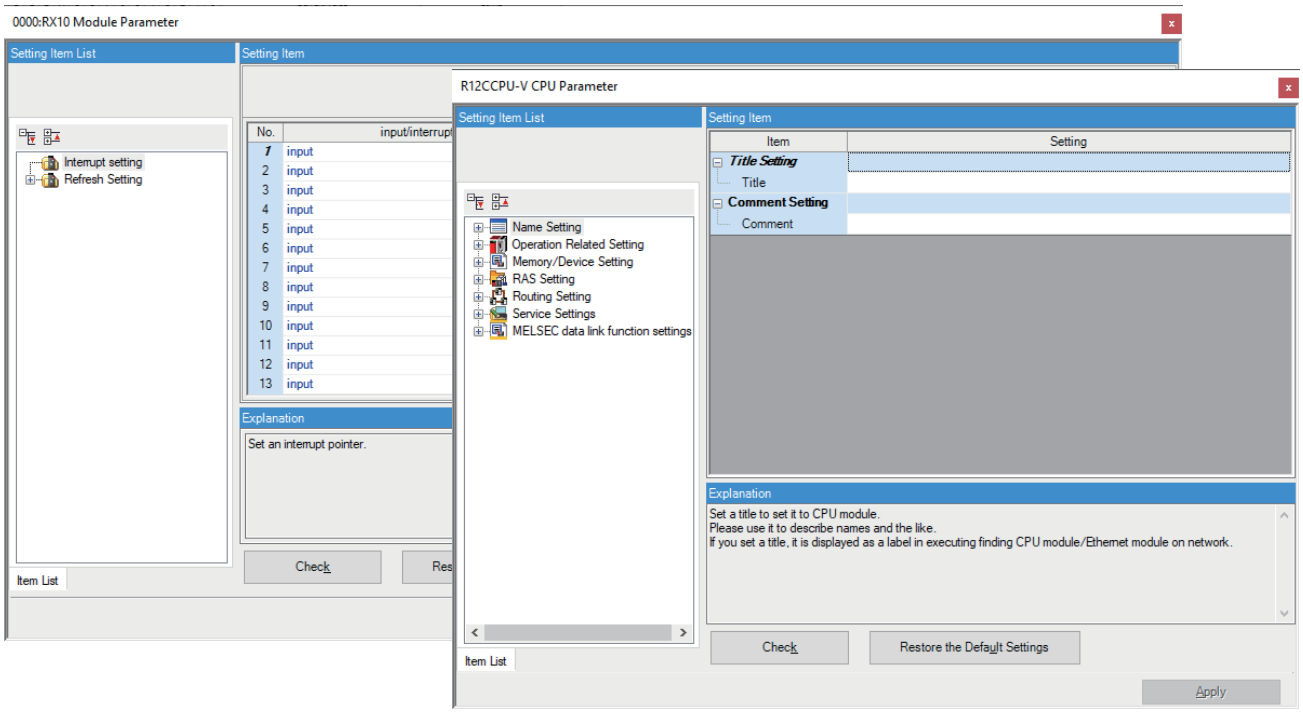
CW Configurator is a dedicated software package for setting and monitoring the parameters of the C Controller module.

3.1 Main functions of CW Configurator

CW Configurator manages module configuration data and parameters for each C Controller module on a project basis.
The main functions of CW Configurator are as follows.

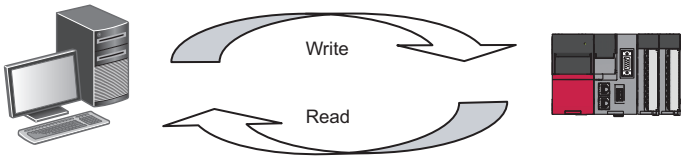
Parameter setting function

This function allows parameters of C Controller modules, I/O modules, and intelligent function modules to be set.



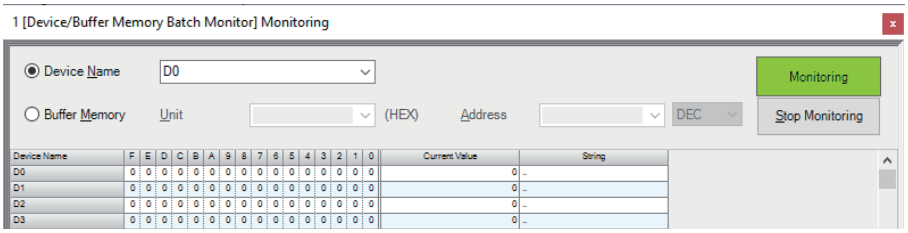
Programmable controller read/write function

With the "Read from PLC"/"Write to PLC" functions, a set parameter can be read/written from/to the C Controller module.



Module operation check function

With a personal computer connected to the C Controller module, device contents of the C Controller module and intelligent function modules can be monitored to check their actions.

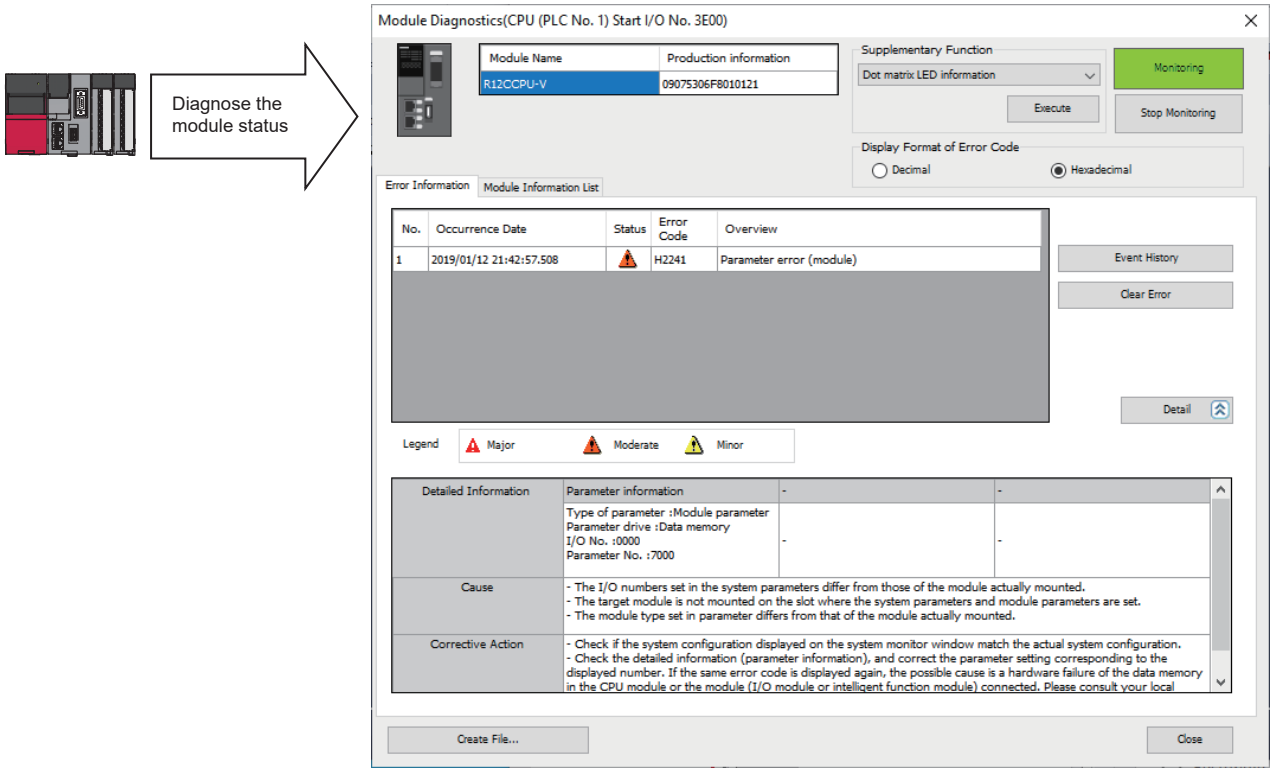


Diagnostic function

This function provides diagnostics for the current error status and error history of the module and network. The diagnostic function helps to shorten the recovery time.

Detailed information of intelligent function modules can also be checked via system monitoring. This feature further shortens the recovery time when an error occurs.

Module diagnostics (module diagnostics window)



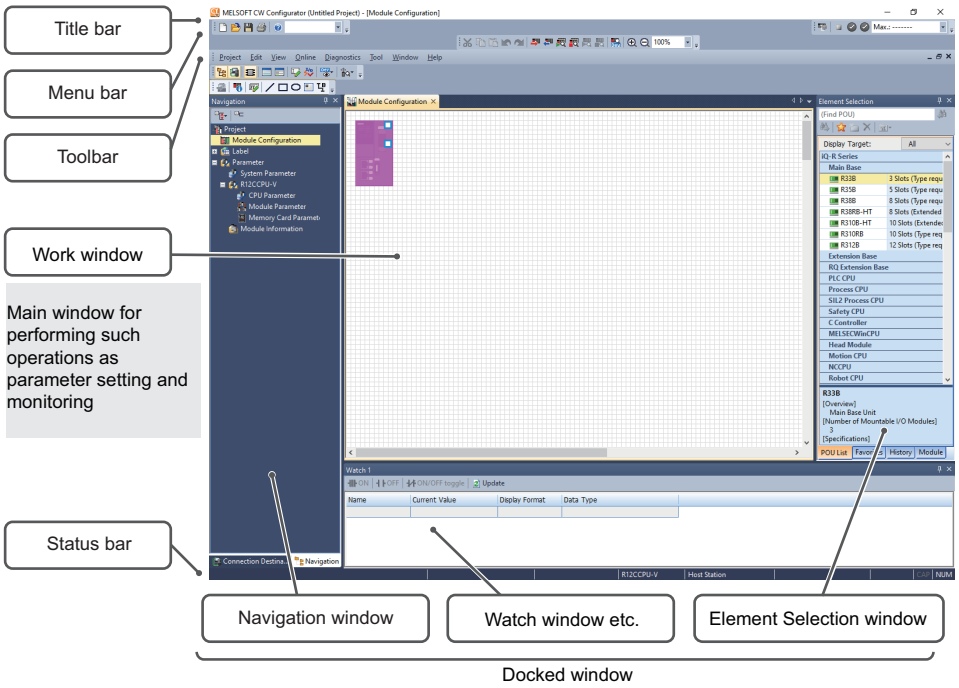
3.2 CW Configurator Screen Layout

This section describes the screen layout when starting CW Configurator.

Main frame

The following figure shows the entire main frame layout.

In this screen, a work window and each docking window are displayed.



Name			Description	Reference
(1)	Title bar		Information such as the project name is displayed.	—
(2)	Menu bar		Menu items for executing each function are displayed.	—
(3)	Toolbar		Tool buttons for executing each function are displayed.	—
(4)	Work window		The main window for such operations as parameter setting and monitoring	—
(5)	Docking window	Navigation window (Project view)	The project contents are displayed in tree form.	☞ Page 35 Navigation window
		Watch window	The current values of the registered devices are displayed.	📖CW Configurator Operating Manual
		Element Selection window	Elements for the Module Configuration window are displayed in list form.	☞ Page 36 Element Selection window
(6)	Status bar		Information of the project being edited is displayed.	—

Window operation

■Displaying docking windows

[View] ⇨ [Docking Window] ⇨ [(target item)]

■Switching docking windows and work windows

Press Ctrl + Tab to switch windows or files.

Select with Ctrl + ←/→/↑/↓.

■Arranging work windows

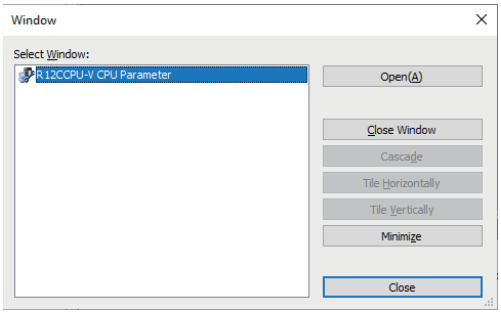
A list of currently opened windows can be displayed.

In addition, the specified window can be opened or arranged.

When multiple windows are opened, the target window can be displayed efficiently.

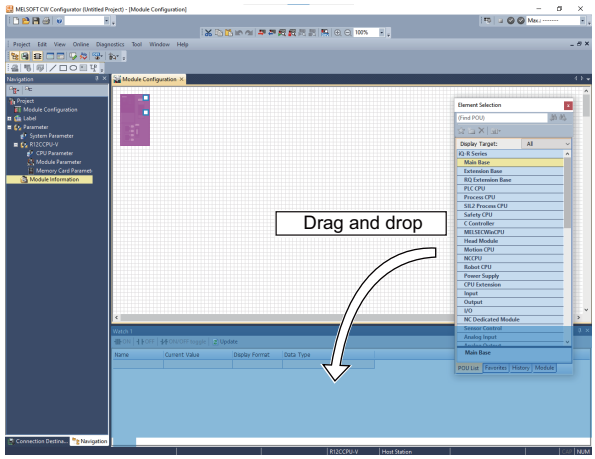
Window

[Window] ⇨ [Window]

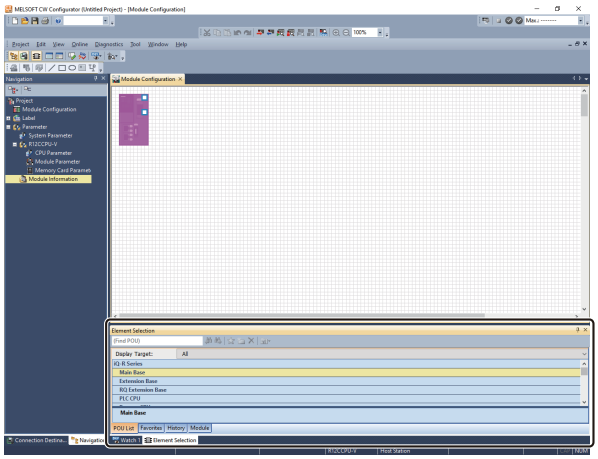


■Switching between docking and floating for docking windows

- Docked display: Drag and drop the title bar of a floating docking window into the guidance in the main frame to dock that window into the main frame.



Fit the title bar into the guidance.



The title bar is docked.

- Floating display: Drag the title bar of a docked window to any location to display that window separately from the main frame.

■Switching between docking and floating for work windows

- Docked display: Select a floating work window, and select [Window] ⇨ [Docking].
- Floating: Select a docked work window, and select [Window] ⇨ [Floating].

Point

For a window docked once, the docked display and the floating display can be switched by double-clicking its title bar.

Customizing/resetting toolbars

Set the types of tool buttons to be displayed on each toolbar.
The selected items in the list are displayed as tool buttons.

Operating procedure

■Customizing

1. Click in the toolbar, and select [Show/Hide Buttons] ⇨ [(toolbar name)].
2. Select a tool button to be displayed from the list.

■Resetting

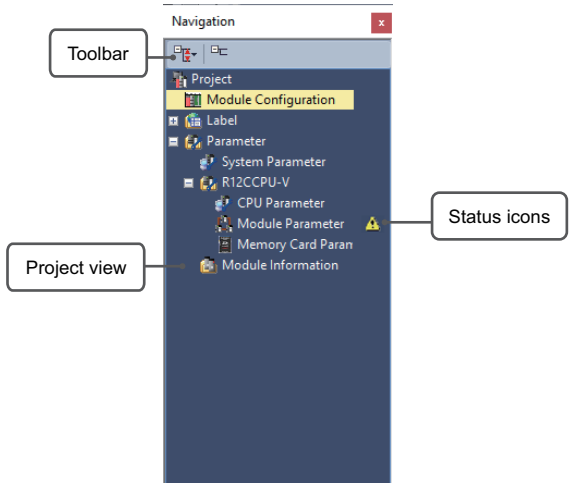
Click in the toolbar, and select [Show/Hide Buttons] ⇨ [Reset].

Navigation window

The navigation window displays the contents of a project in tree form.
Data can be newly created and the edit window can be displayed via the tree.

Window

[View] ⇨ [Docking Window] ⇨ [Navigation] ( )



Displayed items

Name	Description	Reference
Status display icon	The icon representing the status of a project is displayed.	☞ Page 35 Status display icon

Simple display

Click on the toolbar to hide the folders that are not used.

Status display icon

The following table shows the icons representing project statuses.

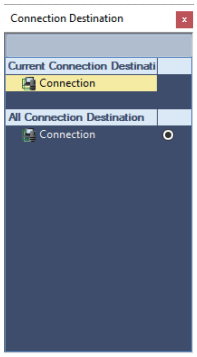
Icon	Status	Display timing	Item	Description
	Mismatch between parameters	Offline	Module folder	Displayed when a mismatch occurs between the system parameters and the module properties.
	Unconfirmed required setting		Module parameter	Displayed when the [Apply] button has never been pressed in the setting window for the setting-required module parameter (network).

Connection Destination window

The Connection Destination window displays the connection destinations set for the C Controller module in list form.

Window

[View] ⇨ [Docking Window] ⇨ [Connection Destination] ( )



For details on how to set a connection destination, refer to the following.

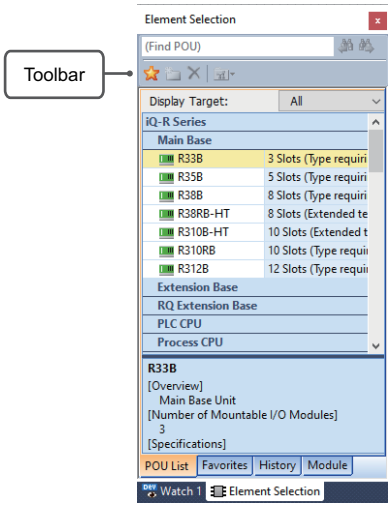
☞ Page 44 Specifying the connection destination

Element Selection window

The Element Selection window displays elements for the Module Configuration window in list form.

Window

[View] ⇨ [Docking Window] ⇨ [Element Selection] ( )



Enter a search string (an element name or a keyword included in the description of an element) into the toolbar, and the focus moves to the matched element.

From the display targets, only the elements included in the selected category can be displayed.

Pasting elements

■Pasting into the Module Configuration window

When the Module Configuration window is displayed, elements that can be pasted are displayed in the Element Selection window.

Paste an element by dragging and dropping it into the Module Configuration window from the list.

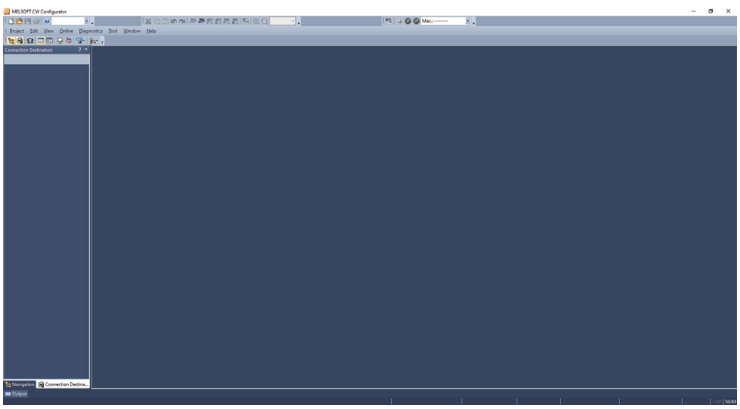
3.3 Parameter Settings

This section describes the procedure for setting parameters with CW Configurator and writing them to the C Controller module.

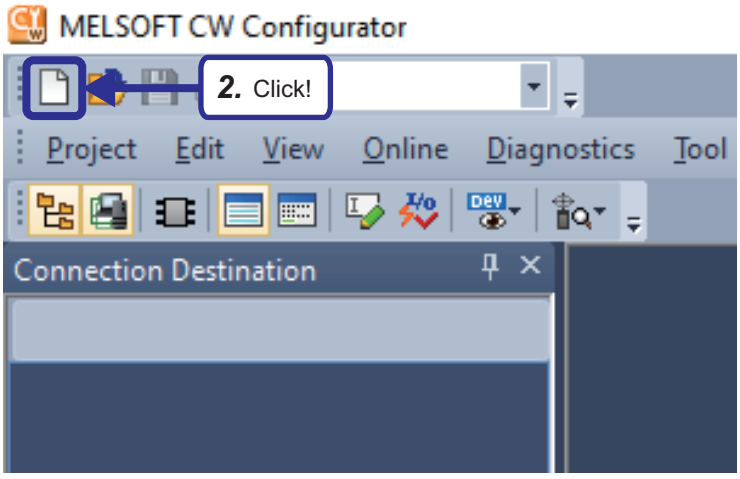
Starting up CW Configurator

To set parameters, start up CW Configurator.

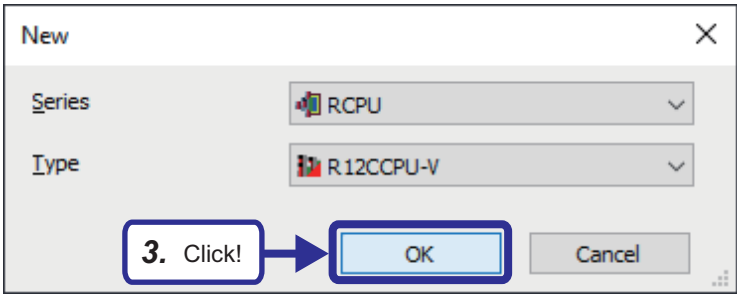
Operating procedure



1. On the Windows® Start screen ⇨ click [MELSOFT] ⇨ [CW Configurator].



2. Click on the toolbar or [Project] ⇨ [New] menu (Ctrl + N).



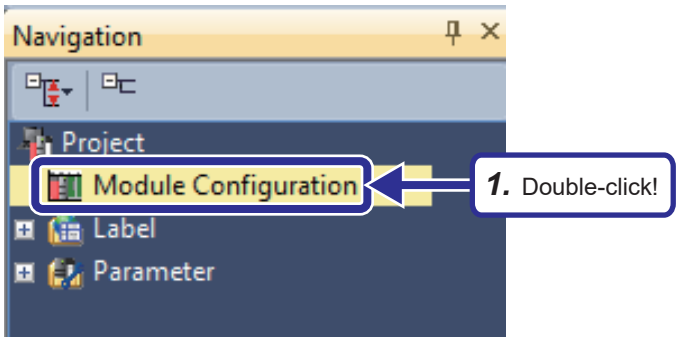
3. Click the [OK] button.

Parameter setting procedure with CW Configurator

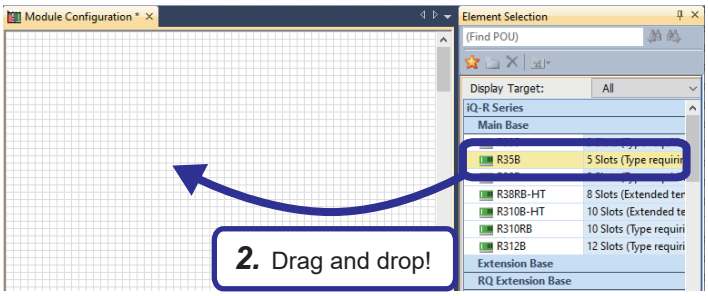
Adding module data

Add a module onto the Module Configuration window, and module parameters can be set.

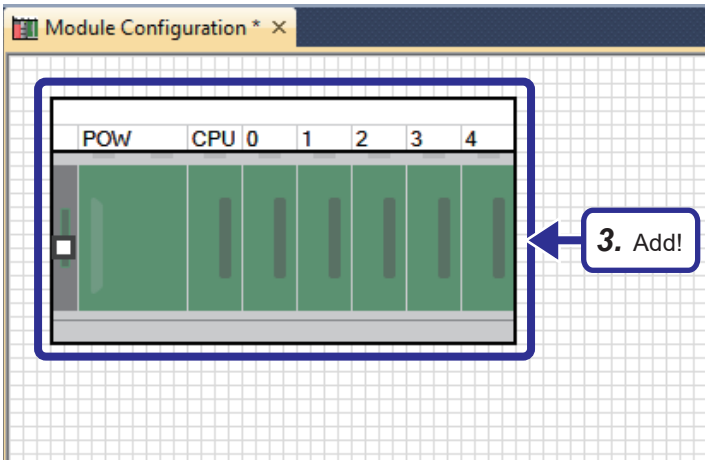
Operating procedure



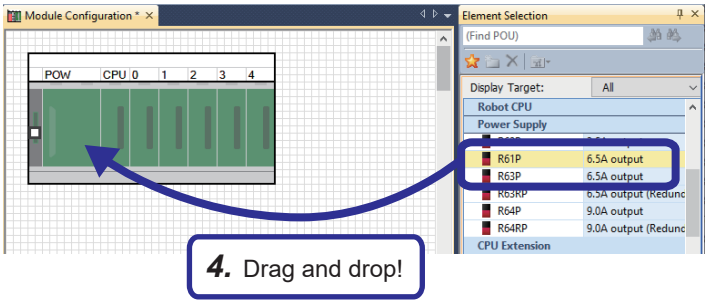
1. Double-click "Module Configuration Diagram" in the Navigation window.
When the dialog box about parameter information appears, click the [OK] button.



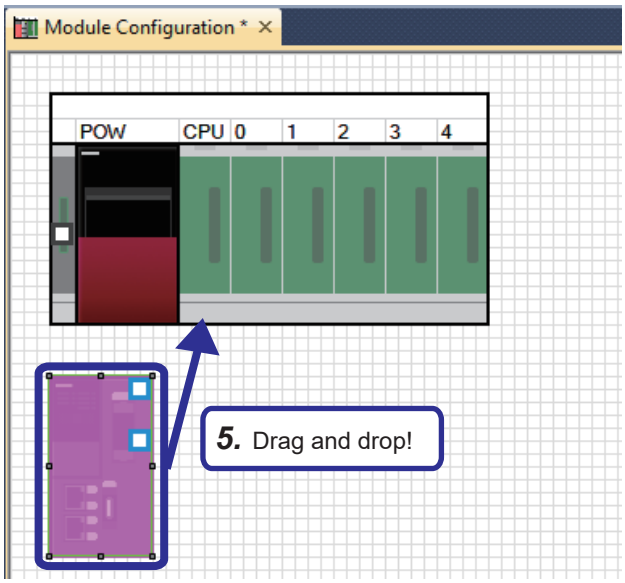
2. When the module configuration diagram dialog box appears, select "R35B" from the "Main Base" section on the Element Selection window, and drag and drop it to the Module Configuration window.



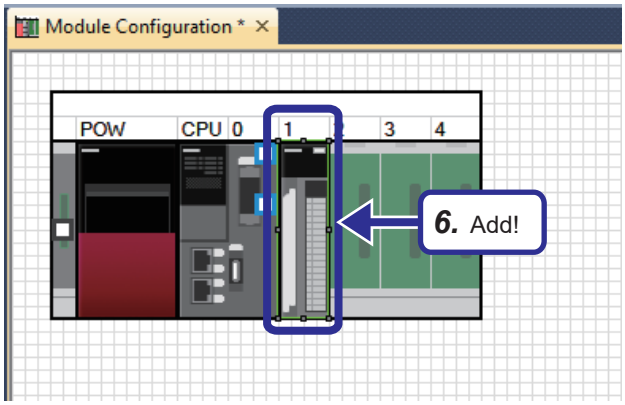
3. "R35B" is added to the Module Configuration window.



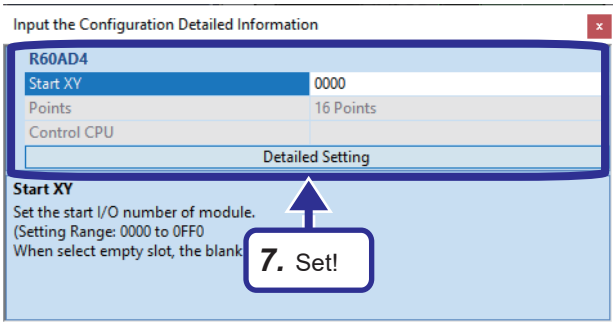
4. Select "R61P" from the "Power Supply" section on the Element Selection window, and drag and drop it to the power supply slot of the R35B that was added to the Module Configuration window.
While the R61P is being dragged and dropped, locations where it can be placed are highlighted.



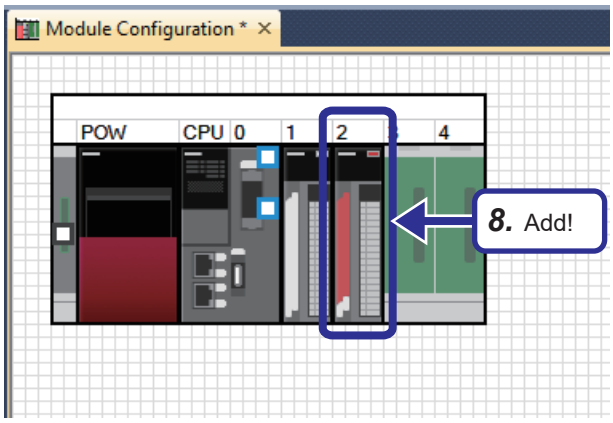
5. Add the R12CCPU-V already placed on the Module Configuration window to the CPU slot of the R35B.



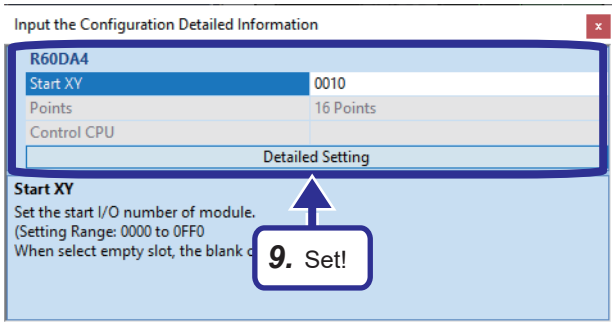
6. In the same way as when adding a power supply, select "R60AD4" from the "Analog Input" section on the Element Selection window, and add it to slot number 1 of the R35B.



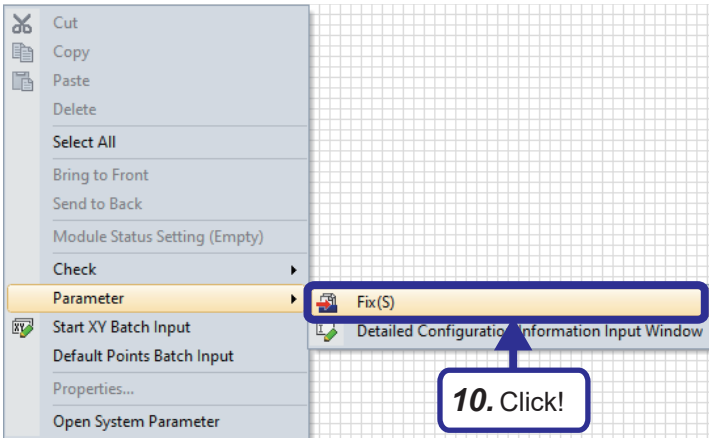
7. Right-click the added R60AD4 ⇨ click [Parameter] ⇨ [Input Detailed Configuration Information Window] menu, and when the input detailed configuration information window appears, set as follows.
[Setting details]
start XY: 0000



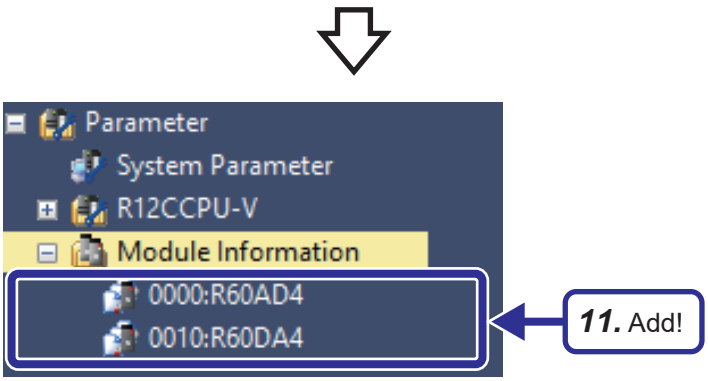
8. In the same way, select "R60DA4" from the "Analog output" section on the Element Selection window, and add it to slot number 2 of the R35B.



9. In the same way as procedure **7.**, set as follows.
[Setting details]
start XY: 0010



10. After the setting, right-click ⇨ click [Parameter] ⇨ [Fix] menu to apply the parameters.
(Click the [OK] button when the confirmation window for adding module labels appears.)

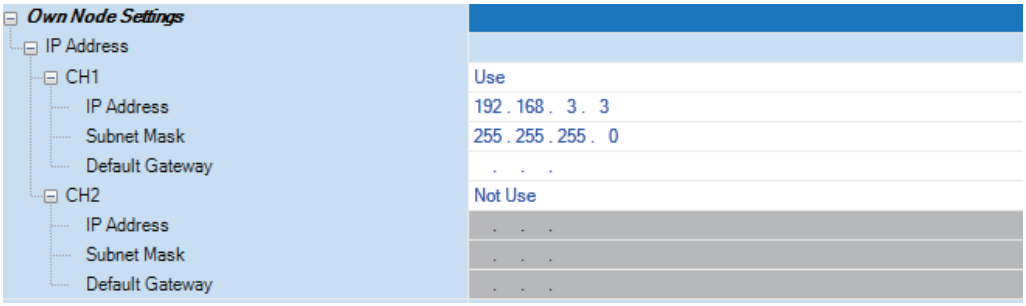


11. The data of the specified modules are added to the Navigation window.

Own node settings

The own node settings refer to the settings necessary for the C Controller module to communicate with an external device.

Window



Displayed items

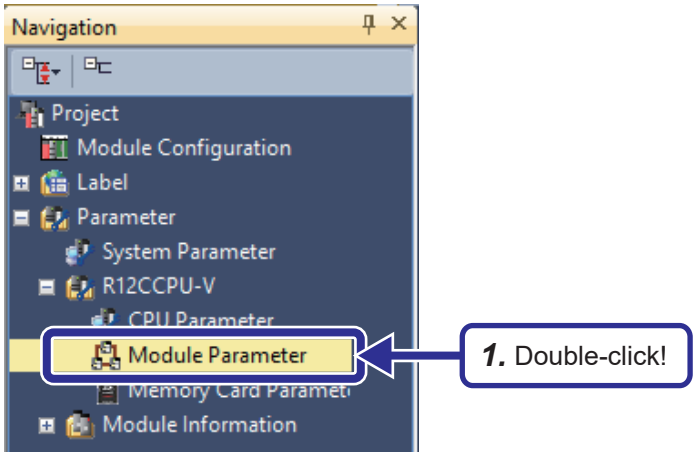
—: there is no setting

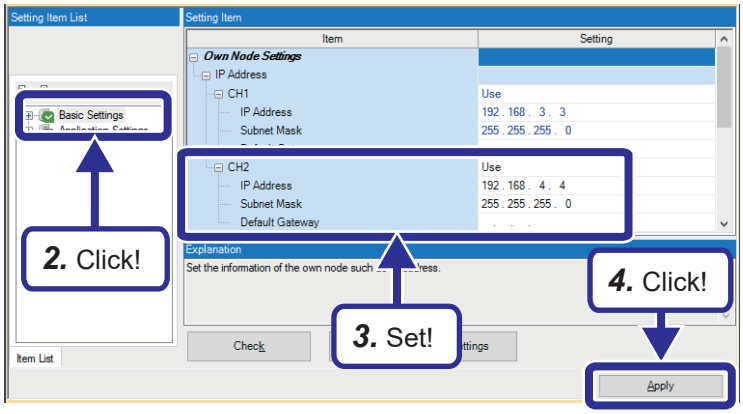
Item			Description	Setting range	Default
IP address setting	Use of CH1	—	Set whether to use the Ethernet Port (CH1).	• Not Use • Use	Use
		IP address	Set the IP address so that the external device to be communicated with has an address of the same class and subnet. Set the IP address so that it belongs to a network different from the network for CH2.	• 0.0.0.1 to 223.255.255.254	192.168.3.3
		Subnet mask	When the IP address of the default gateway is set and when communicating with an external device in a different network via routers, set the subnet mask pattern of the default gateway. All the devices on the same subnetwork must have the common subnet mask. This setting is not required when communications are performed in a single network.	• 128.0.0.0 to 255.255.255.252	255.255.255.0
		Default gateway	Set the IP address of the device through which to access an external device in a different network (default gateway). Set a value for the default gateway IP address that satisfies the following conditions. • The IP address class must be Class A, B, or C. • The subnet address of the default gateway is the same as the subnet address of the C Controller module in the own station. • The host address is neither all "0" nor all "1".	• —(empty) • 0.0.0.1 to 223.255.255.254	—
	Use of CH2	—	Set whether to use the Ethernet Port (CH2).	• Not Use • Use	Not Use
		IP address	Same as CH1	• 0.0.0.1 to 223.255.255.254	—
		Subnet mask		• 128.0.0.0 to 255.255.255.252	
		Default gateway		• —(empty) • 0.0.0.1 to 223.255.255.254	

Point

When using both CH1 and CH2, set different values for the IP address and subnet mask.

Operating procedure



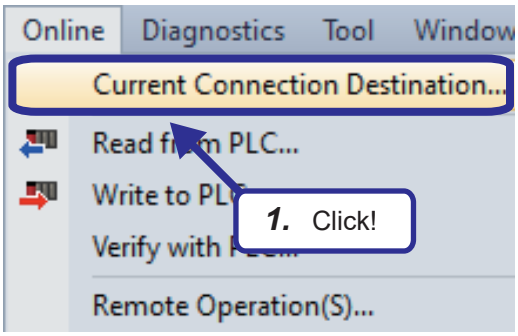


1. Double-click "Module Parameter" of the R12CCPU-V in the Navigation window.
2. Select "Basic Settings" from the list of setting items.
3. Set "Basic Settings" as follows.
   [Setting details]
   Use of CH2: Use
   IP address: 192.168.4.4
   Subnet mask: 255.255.255.0
4. Click the [Apply] button.
5. Save the created and set data.
   Project name: school_ccpu.cp5

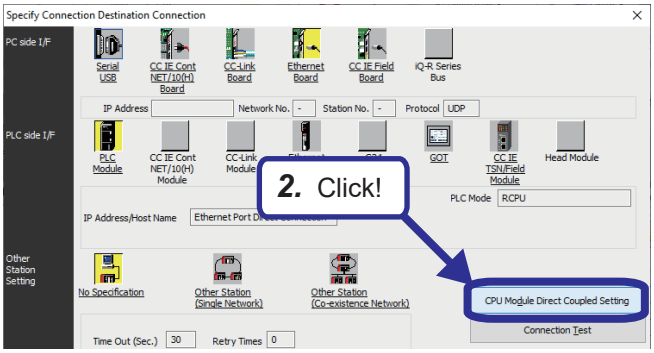
Specifying the connection destination

Specify the connection destination.

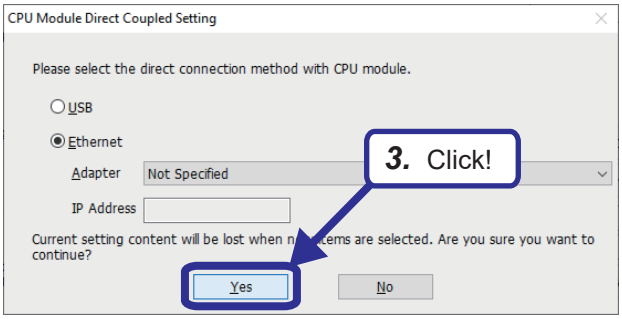
Operating procedure



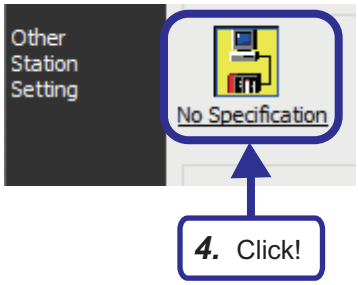
1. Select [Online] ⇨ [Current Connection Destination] on the menu bar of the engineering tool.



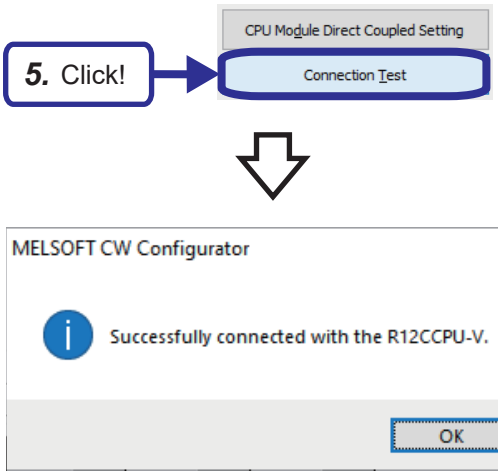
2. Click the [CPU Module Direct Coupled Setting] button on the "Specify Connection Destination Connection" window.
The CPU Module Direct Coupled Setting dialog box appears.



3. Select a method of connection with the CPU module and click the [Yes] button.

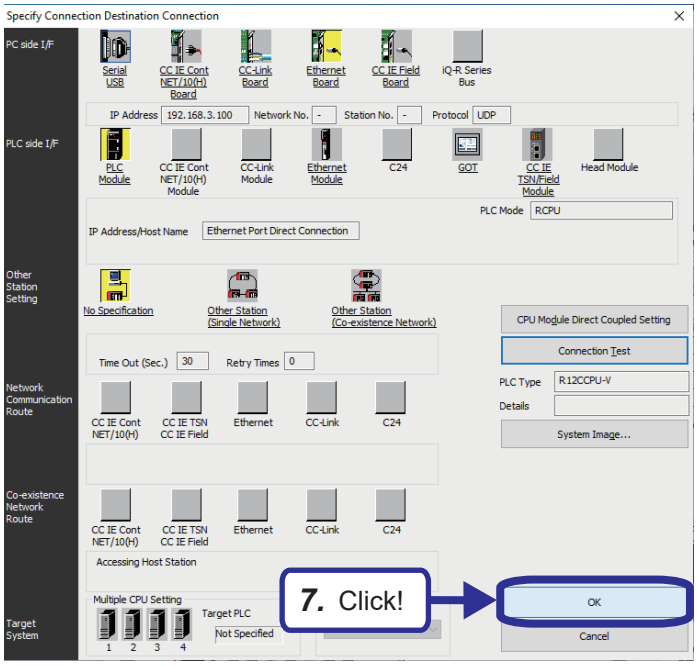


4. Click "No Specification" for the other station setting.



5. Click the [Connection Test] button.

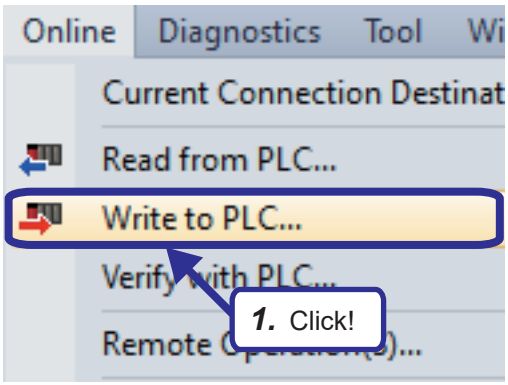
6. Check that the CPU module is connected successfully.



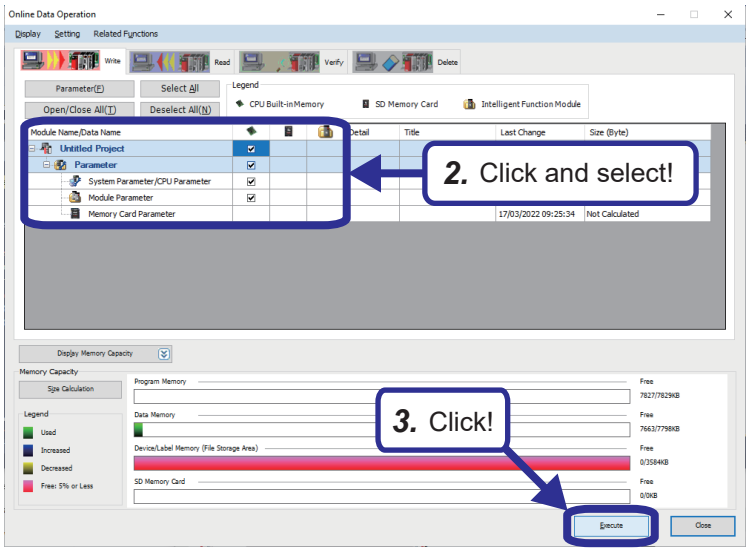
7. Click the [OK] button.

Writing parameters to the C Controller module

Write the parameters set with CW Configurator to the C Controller module.

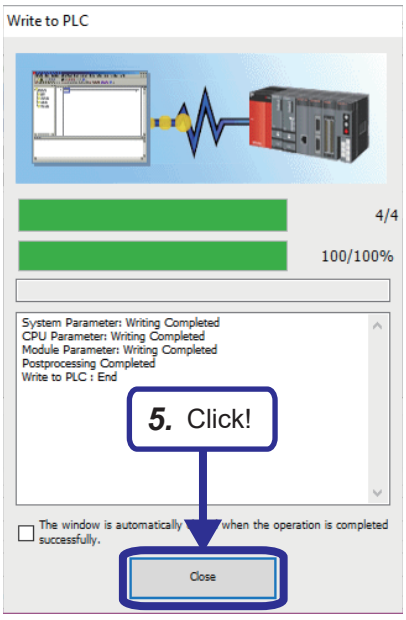


1. Select [Online] ⇨ [Write to PLC] from the menu of the engineering tool.



2. Select "System Parameter/CPU Parameter" and "Module Parameter".

3. Click the [Execute] button.



4. In the window for choosing whether to overwrite parameters or not, click the [Yes to all] button.

5. When the writing completion message appears as the writing completes, click [Close].

6. Reset the C Controller after writing the parameters. For details on the reset procedure, refer to the following.

☞ Page 21 Reset operation procedure

4 DEVICE ACCESS

4.1 Device List

This section describes the devices that can be used in the C Controller module. The devices are used to access peripherals (such as an I/O module and intelligent function module) of the C Controller system.

Device list

The following table lists the device names that can be used and the ranges of use.

Classification	Type	Device name	Default value			Number of points setting	Setting range
			Number of points	Range of use			
User device	Bit device	Input	4096 points	X0 to FFF	Hexadecimal	Not allowed	—
		Output	4096 points	Y0 to FFF	Hexadecimal		
	Bit device	Internal relay	61440 points	M0 to 61439	Decimal	Not allowed	—
		Link relay	655360 points	B0 to 9FFFF	Hexadecimal		
	Word device	Data register	4184064 points	D0 to 4184063	Decimal		
		Link register	1048576 points	W0 to FFFFF	Hexadecimal		
System device	Bit device	Special relay	4096 points	SM0 to 4095	Decimal	Not allowed	—
	Word device	Special register	4096 points	SM0 to 4095	Decimal		
Link direct device*1	Bit device	Link input	16384 points	Jn\X0 to 3FFF	Hexadecimal		
		Link output	16384 points	Jn\Y0 to 3FFF	Hexadecimal		
		Link relay	32768 points	Jn\B0 to 7FFF	Hexadecimal		
		Link special relay	512 points	Jn\SB0 to 1FF	Hexadecimal		
	Word device	Link register	131072 points	Jn\W0 to 1FFFF	Hexadecimal		
		Link special register	512 points	Jn\SW0 to 1FF	Hexadecimal		
Module access device	Word device	Module access device	268435456 points	Un\G0 to 268435455	Decimal	Not allowed	—
CPU buffer memory access device	Word device	CPU buffer memory access device	268435456 points	U3En\G0 to 268435455	Decimal	Not allowed	—
	Word device	Fixed scan communication area access device	0 points	—	Decimal	Allowed	U3En\HG0 to 12287
File register	Word device	File register	1835008 points	ZR0 to 1835007	Decimal	Not allowed	—
Pointer	—	Interrupt pointer	1024 points	I0 to I15, I50 to I1023	Decimal	Not allowed	—

*1 The number of points and range of use for a link direct device differ depending on the network module. For details on the number of points and range of use for link direct devices, refer to the manual for the network module used.

Point

Use only devices in the list.

Device description

The following table shows an overview of devices that can be used.

For details on devices, refer to the following.

MELSEC iQ-R CPU Module User's Manual (Application)

Device name			Description
User device	Input	X	This device provides the CPU module with commands and/or data using an external device, such as a pushbutton, transfer switch, limit switch, and device switch.
	Output	Y	This device outputs the control results of the program to various devices, such as an external signal light/digital HMI/electromagnetic switch (electromagnetic contactor)/ solenoid.
	Internal relay	M	This device is used as an auxiliary relay within the CPU module.
	Data register	D	This device can store numerical values.
	Link relay	B	This device is used on the C Controller module side when data is refreshed between the network module and the C Controller module.
	Link register	W	
System device	Special relay	SM	This is the internal relay or internal register for which the specification is defined in the C Controller module, where the status of the C Controller module is stored.
	Special register	SD	
Link direct device	Link input	Jn\X	This device directly accesses link relays and/or link registers of the network module in the CC-Link IE Controller Network and/or CC-Link IE Field Network.
	Link output	Jn\Y	
	Link relay	Jn\B	
	Link special relay	Jn\SB	
	Link register	Jn\W	
	Link special register	Jn\SW	
Module access device	Module access device	Un\G	This device directly accesses from the CPU module to the buffer memory of the intelligent function module mounted on the main base unit and extension base unit.
CPU buffer memory access device	CPU buffer memory access device	U3En\G	This device accesses memory used by the built-in function of the CPU module, such as data writing/reading between CPU modules on the multiple CPU system and Ethernet function.
	Fixed scan communication area access device	U3En\HG	
File register	File register	ZR	This device holds data while the power is turned off. It exists in the file storage area of the device/label memory.
Pointer	Interrupt pointer	I	When the interrupt function is used, this device executes the corresponding routine.

Device access function

By using the dedicated function library, data can be read from/written to the devices and buffer memory of the intelligent function module managed by the CPU module or C Controller module.

Point

By using a peripheral (such as CW Configurator), data can be read from/written to the devices and buffer memory of the C Controller module as well.

4.2 C Controller module dedicated functions

C Controller module dedicated functions are one of the C Controller module "dedicated function libraries". When they are used for user programs, each module in the MELSEC iQ-R series can be controlled easily.

Function list

This section introduces the most basic C Controller module dedicated functions.

Besides those listed below, C Controller module dedicated functions useful for controlling each module and MELSEC communication functions also exist. For each function, refer to the following.

📖 C Controller Module Programming Manual

■Device access

The following table lists the functions used for device access.

Function name	Description	Reference
CCPU_X_In_BitEx	Reads the input signal (X) in units of bits (one point).	☞ Page 50 CCPU_X_In_BitEx
CCPU_X_In_WordEx	Reads the input signal (X) in units of words (16 points).	☞ Page 51 CCPU_X_In_WordEx
CCPU_Y_Out_BitEx	Outputs the output signal (Y) in units of bits (one point).	☞ Page 52 CCPU_Y_Out_BitEx
CCPU_Y_Out_WordEx	Outputs the output signal (Y) in units of words (16 points).	☞ Page 53 CCPU_Y_Out_WordEx
CCPU_ReadDevice	Reads data from the internal user device and internal system device of the C Controller module.	☞ Page 54 CCPU_ReadDevice
CCPU_WriteDevice	Writes data to the internal user device and internal system device of the C Controller module.	☞ Page 55 CCPU_WriteDevice

■User LED control

User LED control includes the dot matrix LED control of the C Controller module.

Function name	Description	Reference
CCPU_SetDotMatrixLED	Sets a value to be displayed in the dot matrix LED control of the C Controller module.	☞ Page 56 CCPU_SetDotMatrixLED

CCPU_X_In_BitEx

This function reads the input signal (X) in units of bits (one point).

■Format

short CCPU_X_In_BitEx (short sFlg, unsigned short usXNo, unsigned short* pusData)

■Argument

Argument	Name	Description	IN/OUT
sFlg	Access flag	Specifies the access flag. • 0: Normal access • Others: Reserved	IN
usXNo	Input signal	Specifies the input signal (X).	IN
pusData	Data storage location	Specifies the storage location of read data. One of the following values is stored according to the value of the input signal (X). • 0: OFF • 1: ON	OUT

■Description

- This function reads the input signal (X) specified by the input signal (usXNo) in units of bits (one point).
- The value of the read input signal (X) is stored in the data storage location (pusData).
- The CCPU_X_In_BitEx function operates for the mounted module corresponding to the specification by the input signal (usXNo) regardless of the type in the parameter settings (I/O assignment). When the specified area is "Empty", it ends normally with non-processing (read data is 0). When it is "Output module", an I/O assign error occurs.

■Return value

Return value	Description
0 (0000H)	Normally finished
Other than 0	Failed For details when the function fails, refer to the following. 📖 MELSEC iQ-R C Controller Module Programming Manual

CCPU_X_In_WordEx

This function reads the input signal (X) in units of words (16 points).

■Format

short CCPU_X_In_WordEx (short sFlg, unsigned short usXNo, unsigned short usSize, unsigned short* pusDataBuf, unsigned short usBufSize)

■Argument

Argument	Name	Description	IN/OUT
sFlg	Access flag	Specifies the access flag. • 0: Normal access • Others: Reserved	IN
usXNo	Start input signal	Specifies the start input signal (X). (Specify a multiple of 16.)	IN
usSize	Read size	Specifies the read size in units of words.	IN
pusData	Data storage location	Specifies the storage location of read data.	OUT
usBufSize	Data storage location size	Specifies the data storage location size in units of words.	IN

■Description

- This function reads input signals (X) amounting to the size specified by the read size (usSize), starting from the start input signal (X) specified by the start input signal (usXNo), and stores them in the data storage location (pusDataBuf).
- For the data storage location size (usBufSize), specify the area size of the data storage location (pusDataBuf).
- The CCPU_X_In_WordEx function operates for the mounted module corresponding to the specification by the input signal (usXNo) regardless of the type in the parameter settings (I/O assignment). When the specified area is "Empty", it ends normally with non-processing (read data is 0). When it is "Output module", an I/O assign error occurs.
- As shown below, read data is stored in the data storage location (pusDataBuf) in the order from the earliest to the oldest, starting from the lower bits.

pusDataBuf	Description
pusDataBuf[0]	Data in usXNo+FH to usXNo
pusDataBuf[1]	Data in usXNo+1FH to usXNo+10H
⋮	⋮
pusDataBuf[usSize-1]	Data in usXNo+(usSize–1)×16+FH to usXNo+(usSize–1)×16

Precautions

For the data storage location size (usBufSize), set a value larger than the value of the read size (usSize).

■Return value

Return value	Description
0 (0000H)	Normally finished
Other than 0	Failed For details when the function fails, refer to the following. 📖 MELSEC iQ-R C Controller Module Programming Manual

CCPU_Y_Out_BitEx

This function outputs the output signal (Y) in units of bits (one point).

■Format

short CCPU_Y_Out_BitEx (short sFlg, unsigned short usYNo, unsigned short usData)

■Argument

Argument	Name	Description	IN/OUT
sFlg	Access flag	Specifies the access flag. • 0: Normal access • Others: Reserved	IN
usYNo	Output signal	Specifies the output signal (Y).	IN
usData	Data storage location	Specifies the storage location of output data. (Specify a value for bit 0.) • 0: Off • 1: On	IN

■Description

- This function outputs (ON/OFF) the output signal (Y) specified by the output signal (usYNo) in units of bits (one point).
- It turns OFF/ON according to the specified value for bit 0 of the data storage location (usData). (The values of bits 1 to 7 are ignored.)
- When the function is executed while the operating status of the CPU module is not RUN, an error in the STOP/PAUSE state occurs.
- When the function is executed for "Input module", an I/O assign error occurs.
- Do not specify an output module managed by another CPU module for the output signal (usYNo).
  Otherwise, operation for the output module is treated as non-processing.

■Return value

Return value	Description
0 (0000H)	Normally finished
Other than 0	Failed For details when the function fails, refer to the following. 📖 MELSEC iQ-R C Controller Module Programming Manual

CCPU_Y_Out_WordEx

This function outputs the output signal (Y) in units of words (16 points).

■Format

short CCPU_Y_Out_WordEx (short sFlg, unsigned short usYNo, unsigned short usSize, unsigned short* pusDataBuf, unsigned short usBufSize)

■Argument

Argument	Name	Description	IN/OUT
sFlg	Access flag	Specifies the access flag. • 0: Normal access • Others: Reserved	IN
usYNo	Start output signal	Specifies the start output signal (Y). (Specify a multiple of 16.)	IN
usSize	Output size	Specifies the output size in units of words.	IN
pusDataBuf	Data storage location	Specifies the storage location of output data.	IN
usBufSize	Data storage location size	Specifies 0.	IN

■Description

- This function outputs (ON/OFF) data in the data storage location (pusDataBuf) to output signals (Y) amounting to the size specified by the data size (usSize), starting from the start output signal (Y) specified by the start output signal (usYNo).
- When the function is executed while the operating status of the CPU module is not RUN, an error in the STOP/PAUSE state occurs.
- When the function is executed for "Input module", an I/O assign error occurs.
- Do not specify an output module managed by another CPU module for the output signal (usYNo).
  Otherwise, operation for the output module is treated as non-processing.
- As shown below, store output data in the data storage location (pusDataBuf) in the order from the earliest to the oldest, starting from the lower bits.

pusDataBuf	Description
pusDataBuf[0]	Normally finished
pusDataBuf[1]	Failed
⋮	⋮
pusDataBuf[usSize-1]	Data in usYNo+(usSize–1)×16+FH to usYNo+(usSize–1)×16

■Return value

Return value	Description
0 (0000H)	Normally finished
Other than 0	Failed For details when the function fails, refer to the following. MELSEC iQ-R C Controller Module Programming Manual

CCPU_ReadDevice

This function reads data from the internal user device and internal system device of the C Controller module.

■Format

short CCPU_ReadDevice (short sDevType, unsigned long ulDevNo, unsigned long ulSize, unsigned short* pusDataBuf, unsigned long ulBufSize)

■Argument

Argument	Name	Description	IN/OUT
sDevType	Device type	Specifies the device type. ☞ Page 55 Device type	IN
ulDevNo	Start device No.	Specifies the start device number. (For bit devices, only a multiple of 16 can be specified.)	IN
ulSize	Data size	Specifies the read data size in units of words.	IN
pusDataBuf	Data storage location	Specifies the storage location of read data.	OUT
ulBufSize	Data storage location size	Specifies the data storage location size in units of words.	IN

■Description

This function reads data of devices amounting to the size specified by the data size (ulSize), starting from the device specified by the device type (sDevType) and start device number (ulDevNo), and stores it in the data storage location (pusDataBuf).

Precautions

For the data storage location size (ulBufSize), set a value larger than the value of the data size (ulSize).

■Return value

Return value	Description
0 (0000H)	Normally finished
Other than 0	Failed For details when the function fails, refer to the following. 📖 MELSEC iQ-R C Controller Module Programming Manual

CCPU_WriteDevice

This function writes data to the internal user device and internal system device of the C Controller module.

■Format

short CCPU_WriteDevice (short sDevType, unsigned long ulDevNo, unsigned long ulSize, unsigned short* pusDataBuf, unsigned long ulBufSize)

■Argument

Argument	Name	Description	IN/OUT
sDevType	Device type	Specifies the device type. ☞ Page 55 Device type	IN
ulDevNo	Start device No.	Specifies the start device number. (For bit devices, only a multiple of 16 can be specified.)	IN
ulSize	Data size	Specifies the write data size in units of words.	IN
pusDataBuf	Data storage location	Specifies the storage location of write data.	IN
ulBufSize	Data storage location size	Specifies 0.	IN

■Description

This function writes data in the data storage location (pusDataBuf) amounting to the size specified by the data size (ulSize) to devices starting from the device specified by the device type (sDevType) and start device number (ulDevNo).

■Return value

Return value	Description
0 (0000H)	Normally finished
Other than 0	Failed For details when the function fails, refer to the following. 📖 MELSEC iQ-R C Controller Module Programming Manual

Device type

Device type refers to the device type to be specified for C Controller module dedicated functions.
Devices are defined in the header file "CCPUFunc.h".

Point

A device type can be specified either by a code or device name.

Device (Device name)	Device type		
	Code		Device name
	Decimal	Hexadecimal	
Internal relay (M)	4	4H	Dev_CCPU_M
Special relay (SM)	5	5H	Dev_CCPU_SM
Data register (D)	13	DH	Dev_CCPU_D
Special register (SD)	14	EH	Dev_CCPU_SD
Link relay (B)	23	17H	Dev_CCPU_B
Link register (W)	24	18H	Dev_CCPU_W
File register (ZR)	200	DCH	Dev_CCPU_ZR

CCPU_SetDotMatrixLED

This function sets a value to be displayed in the dot matrix LED control of the C Controller module.

■Format

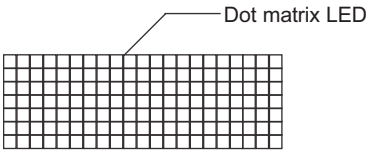
short CCPU_SetDotMatrixLED(unsigned short usLedMode, char* pcData)

■Argument

Argument	Name	Description	IN/OUT
usLedMode	Output mode	Specifies the output mode for the dot matrix LED. (If reserved is specified, the function finishes normally with non-processing.) • 0: Dot mode • 1: ASCII mode • Others: Reserved	IN
pcData	LED data	Specifies the LED data.	IN

• Specifies the LED data (pcData) as follows.

· Mode 0: In dot mode



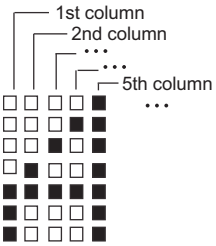
pcData[0] to pcData[19]: Dot matrix LED data (7×20)

Data specified by the following format is displayed in each column.

Data format for each column: Bit pattern of 0 for upper one bit, 1 for lower 7 bits when LED is turned on, and 0 when LED is turned off

Ex.

When the following bit patterns are output to the dot matrix LED



1st column: 0000 0111b=07H→pcData[0]=0x07

2nd column: 0000 1100b=0cH→pcData[1]=0x0c

3rd column: 0001 0100b=14H→pcData[2]=0x14

4th column: 0010 0100b=24H→pcData[3]=0x24

5th column: 0111 1111b=7fH→pcData[4]=0x7f

6 to 20th column: 0000 0000b=00H→pcData[5] to pcData[19]=0x00

· Mode 1: In ASCII mode

The specified string is displayed in pcData[0] to pcData[3].

The following table lists the characters that can be specified (ASCII code).

×: Character specification not allowed

Bit		Upper 4 bits															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Lower 4 bits	0	×	×	SP	0	×	P	×	×	×	×	×	×	×	×	×	×
	1	×	×	×	1	A	Q	×	×	×	×	×	×	×	×	×	×
	2	×	×	×	2	B	R	×	×	×	×	×	×	×	×	×	×
	3	×	×	×	3	C	S	×	×	×	×	×	×	×	×	×	×
	4	×	×	×	4	D	T	×	×	×	×	×	×	×	×	×	×
	5	×	×	%	5	E	U	×	×	×	×	×	×	×	×	×	×
	6	×	×	×	6	F	V	×	×	×	×	×	×	×	×	×	×
	7	×	×	×	7	G	W	×	×	×	×	×	×	×	×	×	×
	8	×	×	×	8	H	X	×	×	×	×	×	×	×	×	×	×
	9	×	×	×	9	I	Y	×	×	×	×	×	×	×	×	×	×
	A	×	×	×	×	J	Z	×	×	×	×	×	×	×	×	×	×
	B	×	×	×	×	K	×	×	×	×	×	×	×	×	×	×	×
	C	×	×	×	×	L	×	×	×	×	×	×	×	×	×	×	×
	D	×	×	-	×	M	×	×	×	×	×	×	×	×	×	×	×
	E	×	×	.	×	N	×	×	×	×	×	×	×	×	×	×	×
	F	×	×	/	×	O	×	×	×	×	×	×	×	×	×	×	×

If a character other than the above is specified, an error is returned.

If NULL is in the middle of a string, data after that is not displayed and treated as empty. (Displayed as left-aligned)

■Description

In accordance with the method specified by the output mode (usLedMode), values specified for LED data (pcData) are displayed on the dot matrix LED.

Precautions

- To display data on the dot matrix LED, "USER" must be selected for the operation selection mode.
  (MELSEC iQ-R C Controller Module User's Manual (Startup))
- When the MODE/SELECT switch is operated, if the operation is being checked or the selected operation is being checked, even with "USER" selected for the operation selection mode, an error occurs when the CCPU_SetDotMatrixLED function is executed.

■Return value

Return value	Description
0 (0000H)	Normally finished
Other than 0	Failed For details when the function fails, refer to the following. (MELSEC iQ-R C Controller Module Programming Manual

4.3 Exercise 1 Device Control

Check the operation using a program that performs control for inputting and outputting device values of the C Controller module.

The engineering tool CW Workbench is used for the exercise. For details on CW Workbench, refer to the following.

☞ Page 137 CW Workbench

Creating a project

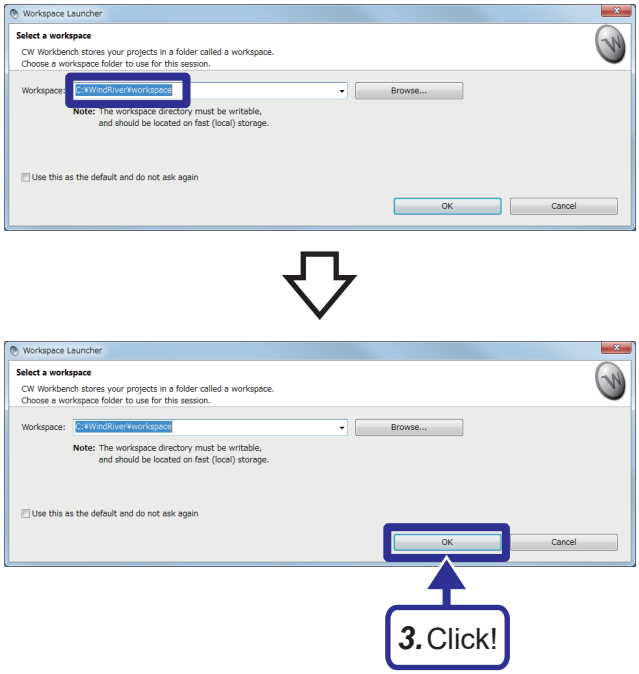
Make preparations in advance of the exercise.

In this manual, CW Workbench is assumed to have already been installed.

In addition, as a workspace, create "C:\CCPU_CWW_Prj\enshu" on a personal computer.

Starting up CW Workbench

Operating procedure



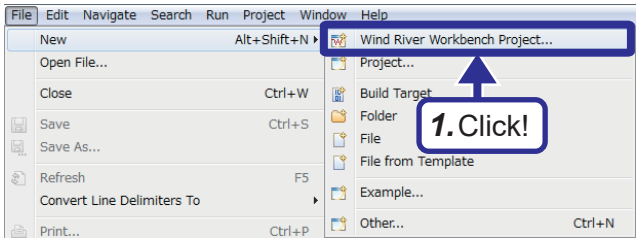
1. Select [Start] ⇨ [All Apps] ⇨ [Wind River] ⇨ [CW Workbench 3.3] ⇨ [CW Workbench 3.3].
2. After CW Workbench starts up, enter the workspace save destination folder.
   [Setting details]
   C:\CCPU_CWW_Prj\enshu
3. Click the [OK] button.

Point

- The size of each window and the icon layout in the initialized state of CW Workbench differ depending on the personal computer used. When the actual screen is different from the screen described in this manual, adjust the size of each window.
- To enlarge, delete, or restore the initial state of each window, select [Window] on the menu ⇨ [New Window].

Creating a new project

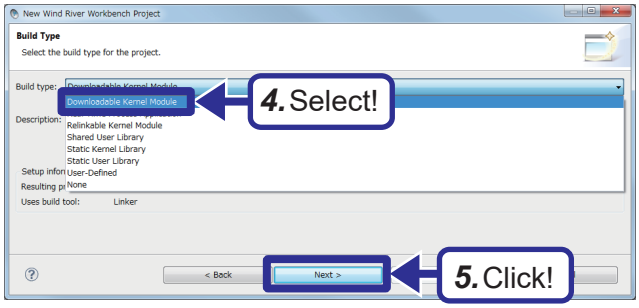
Operating procedure



1. Click [File] on the menu ⇨ [New] ⇨ [Wind River Workbench Project].

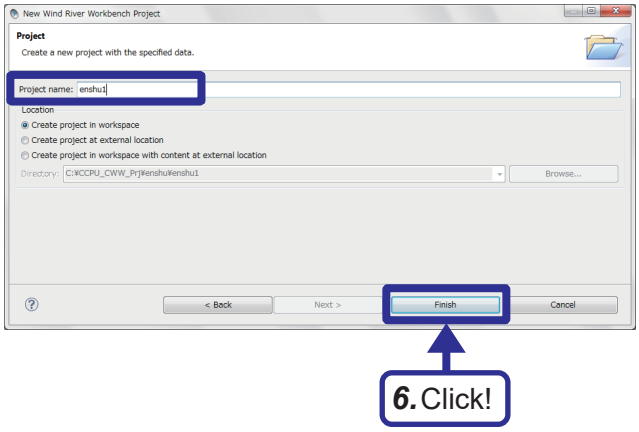
2. Select "Wind River VxWorks6.9".

3. Click the [Next] button.



4. Select "Downloadable Kernel Module".

5. Click the [Next] button.



6. Enter "enshu1" and click the [Finish] button.

Point

To create a project in ☞ Page 119 Exercise 2 A/D conversion, D/A conversion, enter "enshu2".

Setting properties for the project

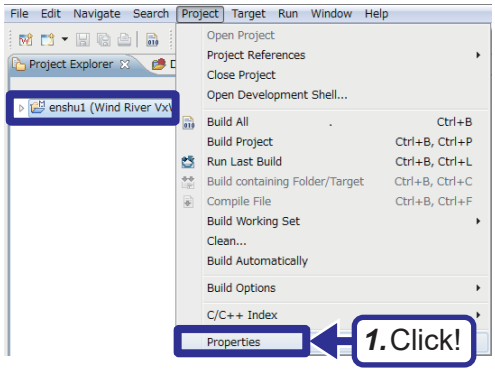
Configure settings for converting (building) the created project into a module that can be executed by the C Controller module.

Point

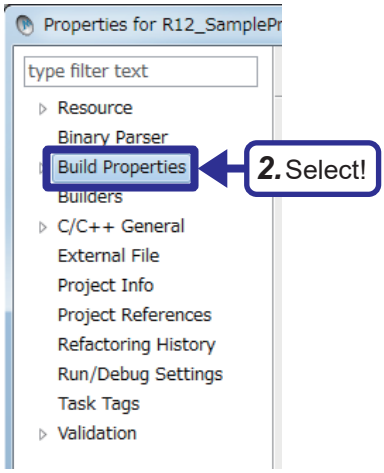
Build: Compile source code according to the processor and establish linkage with the include files.

■Setting the processor to be used

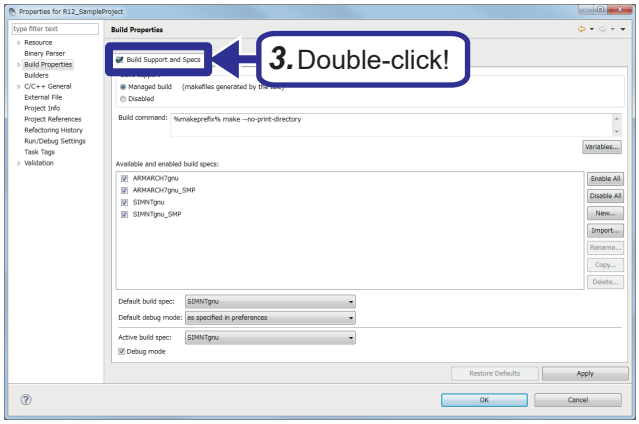
Operating procedure



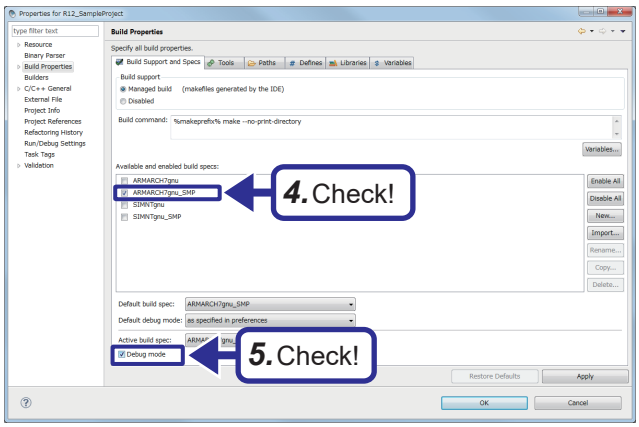
1. In the Project Explorer window, select the created project and click [Project] on the menu ⇨ [Properties].



2. Select "Build Properties" from the tree on the left of the window.



3. Click the [Build Support and Specs] tab.

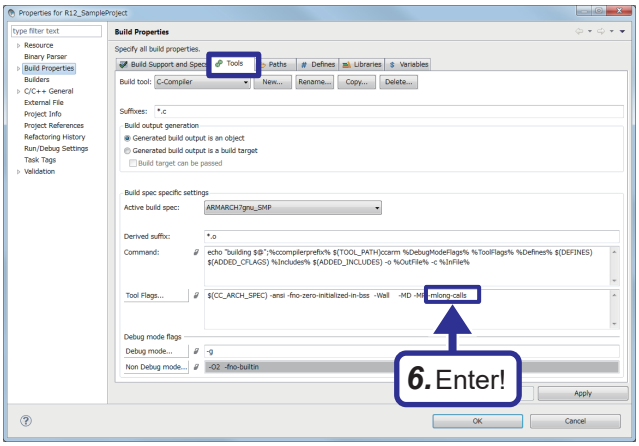


4. In "Available and enabled build specs", select only the checkbox for "ARMARCH7gnu_SMP".

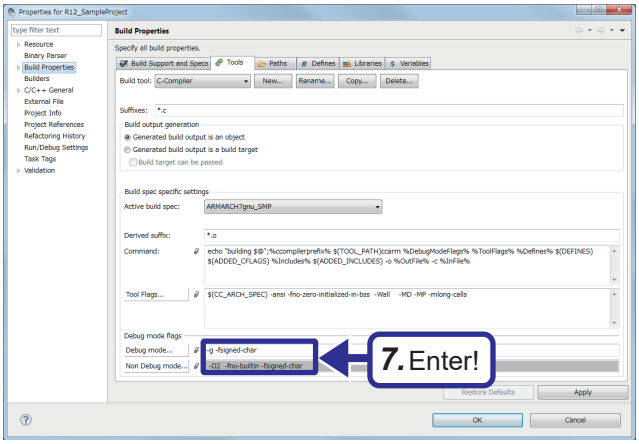
5. Select the checkbox for "Debug mode".

Point

Clear the checkbox for "Debug mode" when actually commissioning and operating the system.



6. Select the [Tools] tab and enter "-mlong-calls" for "Tool Flags".

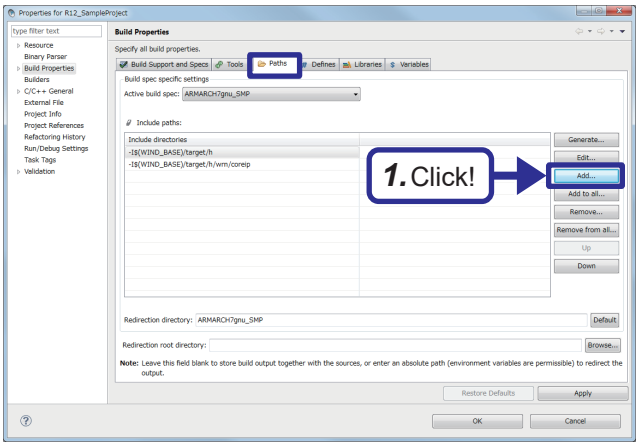


7. In the same way, enter "-fsigned-char" for [Debug mode] and [Non Debug mode] of "Debug mode flags".

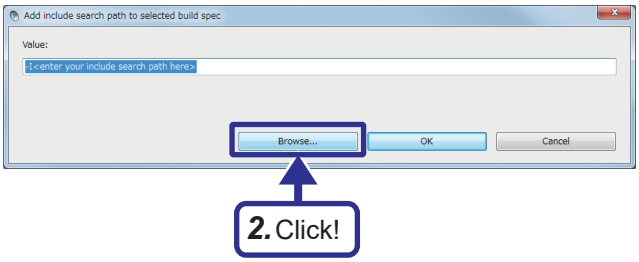
■Setting include files

Create in advance an include folder for storing include files in the following location.
C:\CCPU_CWW_Prj\Include

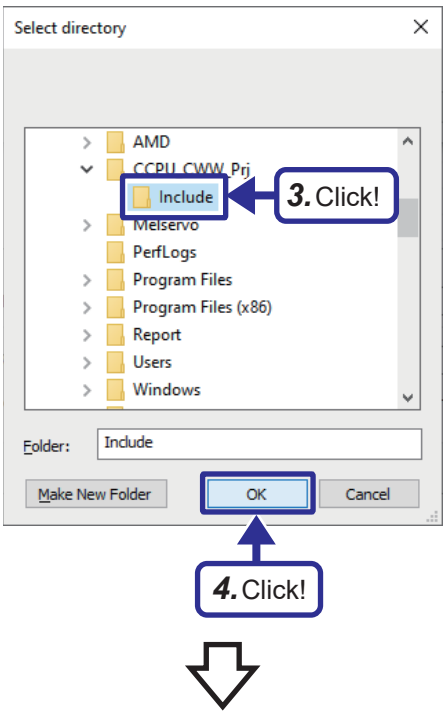
Operating procedure



1. Select the [Paths] tab and click the [Add] button.

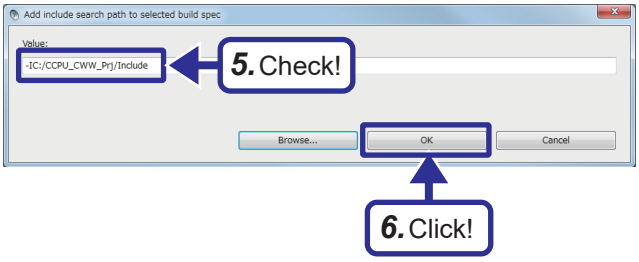


2. Click the [Browse] button.



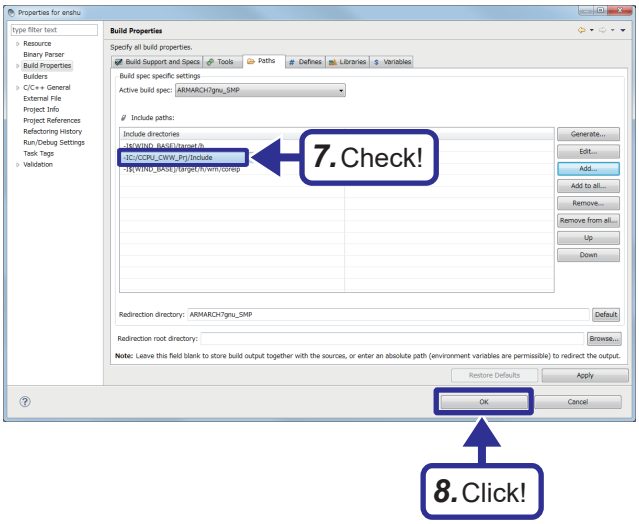
3. Select the include folder "C:\CCPU_CWW_Prj\Include" created in advance.

4. Click the [OK] button.



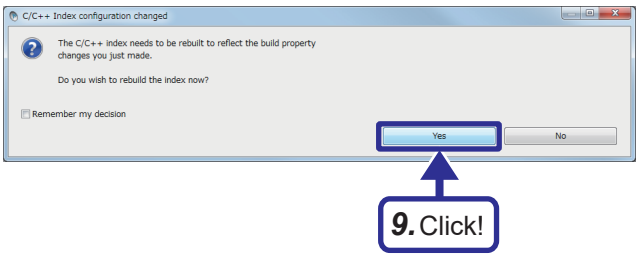
5. In the "Add include search path to selected build spec" window, check that the selected folder is specified.

6. Click the [OK] button.



7. In "Include paths", check that the added include path is displayed.

8. Click the [OK] button.

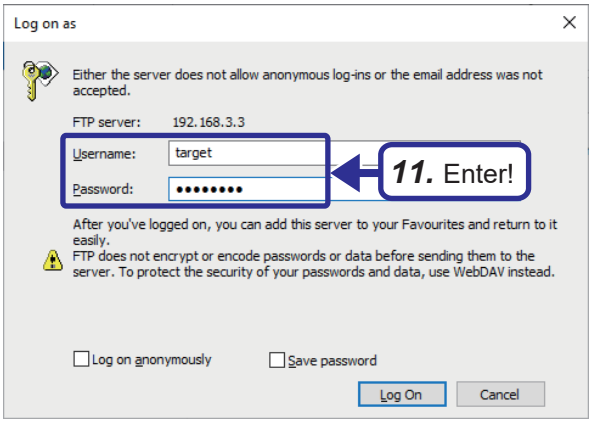


9. When the following message appears after the [OK] button is clicked, click the [Yes] button.

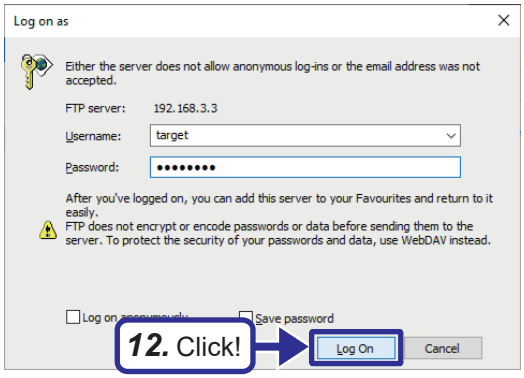
Point

When setting a project for "enshu2", because the include files are already added to the include folder, the subsequent steps do not need to be followed.

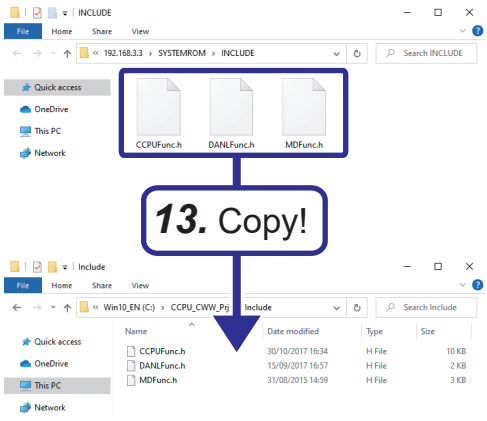
10. Add include files to the created include folder "C:\CCPU_CWW_Prj\Include".
To acquire include files stored in the C Controller module, start up Explorer and enter the following in the address field.
ftp://192.168.3.3/SYSTEMROM/INCLUDE/



11. In the "Log on as" window, enter the following user name and password.
[Setting details]
User name: target
Password: password



12. Click the [Log On] button.



13. Copy the include files stored in the C Controller module to "C:\CCPU_CWW_Prj\Include".

Exercise 1.1 Switch input and lamp output

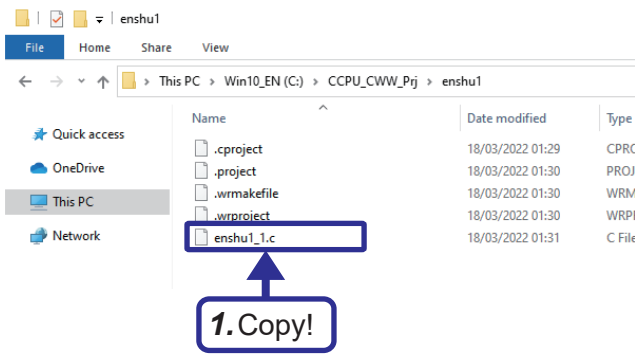
Acquire ON/OFF information of switches M0 to M15 of the demonstration machine, and among lamps Y170 to Y17F of the demonstration machine, turn on the lamps with the same bit numbers as the bit numbers of switches turned on. In addition, display the number of switch operations on the dot matrix LED, and when the number of operations exceeds 25, stop the processing and reset the lamp output and dot matrix LED display.
For operation check, the following shows an overview of the procedure.

Operating procedure

1. Copy the program enshu1_1.c to the project folder C:\CCPU_CWW_Prj\enshu\enshu1 created this time to add it to the project. (☞ Page 66 Procedure for adding a program)
2. Rebuild enshu1_1.c in debug mode. (☞ Page 67 Generating a module for execution)
3. Connect the C Controller module and CW Workbench. (☞ Page 69 Connecting the C Controller module and CW Workbench)
4. Debug the created program to check if it operates correctly. (☞ Page 73 Debugging the user program)
5. Rebuild the program enshu1_1.c that underwent debugging by canceling the debug mode, and store the created user program on the C Controller module. (☞ Page 80 Registering a module for execution)
6. Create a script and store it in the C Controller module. (☞ Page 82 Registering a module for execution)
7. Reset the C Controller module and set the switch on the front to the RUN position.
8. Turn on/off switches M0 to M15 of the demonstration machine, check the lamp outputs from Y170 to Y17F, and check that the number of switch operations is displayed on the dot matrix LED of the C Controller module. (☞ Page 83 Checking the operation)
9. After the operation is checked completely, disconnect CW Workbench from the C Controller module (☞ Page 72 Connecting and disconnecting) once, delete the user program and script stored in the C Controller module, and reset the C Controller module.

Procedure for adding a program

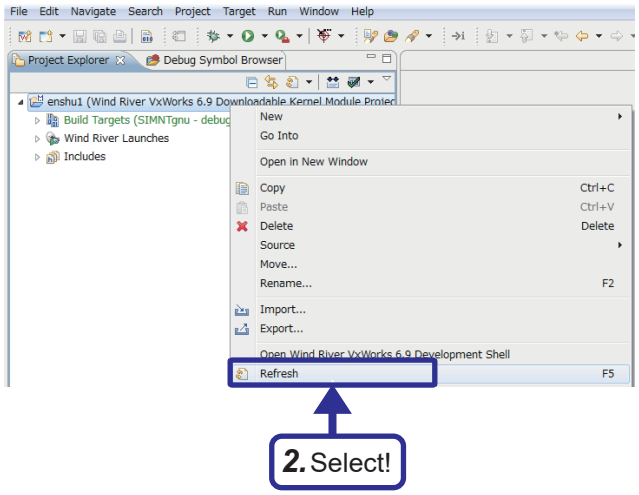
Operating procedure



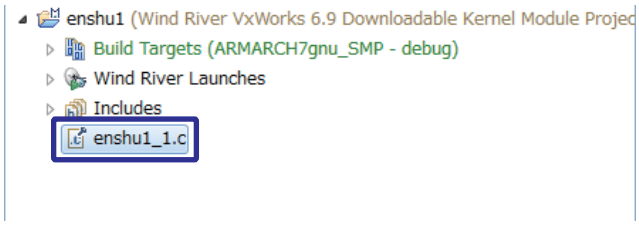
1. Copy the already programmed enshu1_1.c to C:\CCPU_CWW_Prj\enshu\enshu1.

Point

- To add the program in ☞ Page 119 Exercise 2 A/D conversion, D/A conversion, copy the program to C:\CCPU_CWW_Prj\enshu\enshu2.
- The file name of the program to be copied differs depending on the exercise. For the settings in Exercise 2.2, copy enshu2_2.c.



2. In the "Project Explorer" window, select a project to which a program is to be added, and right-click the mouse ⇨ select [Refresh].



3. The program copied in **1.** is added to the project.

Point

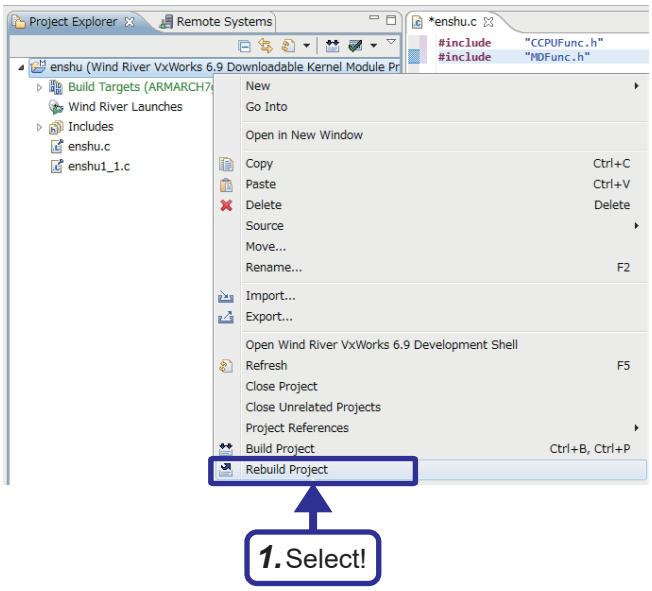
In this exercise, a user program is assumed to have already been created. For details on how to create a new user program, refer to the following.

☞ Page 137 Creating a new user program

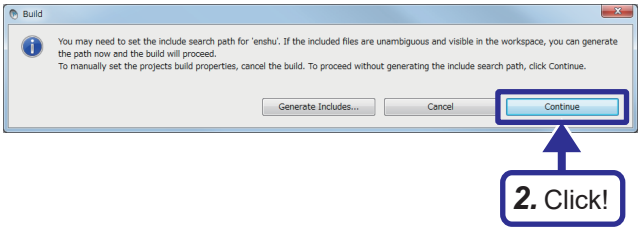
Generating a module for execution

Convert (build) the created program into a module that can be executed by the C Controller module.

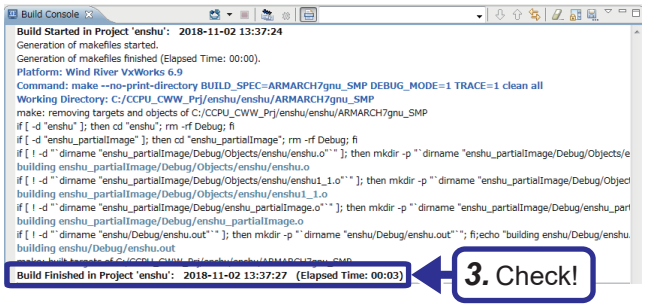
Operating procedure



1. In the "Project Explorer" window, select the created project and right-click the mouse ⇨ select [Rebuild Project].



2. When the message on the left appears, click the [Continue] button.



3. Check that "Build Finished..." is displayed in the Build Console window.

Point

If "Build Finished..." is not displayed and an error occurs, check the error description and correct the program.
After correcting the program, start from step **1.** again.

Point

When the build process finishes normally, the folder in which a user program is generated is as follows.

- Debug mode
  (Workspace folder)\(Project name folder)\ARMARCH7gnu_SMP\(Project name folder)\Debug
- Non Debug mode
  (Workspace folder)\(Project name folder)\ARMARCH7gnu_SMP\(Project name folder)\NonDebug

When a user program is generated in an imported project, the above folders change depending on the folder in which the imported project exists and the project structure.
For the folder in which a user program is generated, check with the imported project.

Precautions

■If the build result is an error

Error information (source file names, line numbers, error descriptions) is displayed in red in the "Build Console" window.
Double-click a line where the source file name and line number are displayed in red to jump to where an error occurred in the source file.
Repeat source code correction & building until error information (red) disappears from the build result.

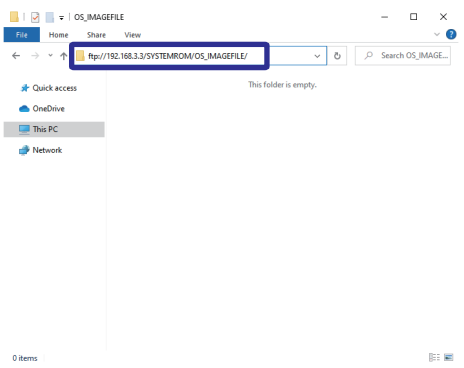
■When the error "command not found" occurs

An unsupported compiler may be used.
In the [Build Support and Specs] tab of "Build Properties", check that the checkbox for "ARMARCH7gnu_SMP" is selected.
Select only the checkbox for "ARMARCH7gnu_SMP".

Connecting the C Controller module and CW Workbench

Connect the Ethernet port (CH1) of the C Controller module and CW Workbench to perform debugging with CW Workbench.

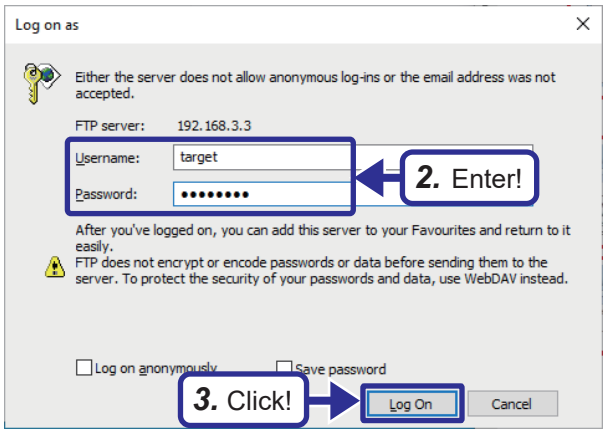
Operating procedure



1. Using Explorer, connect to ftp://192.168.3.3/SYSTEMROM/OS_IMAGEFILE/.

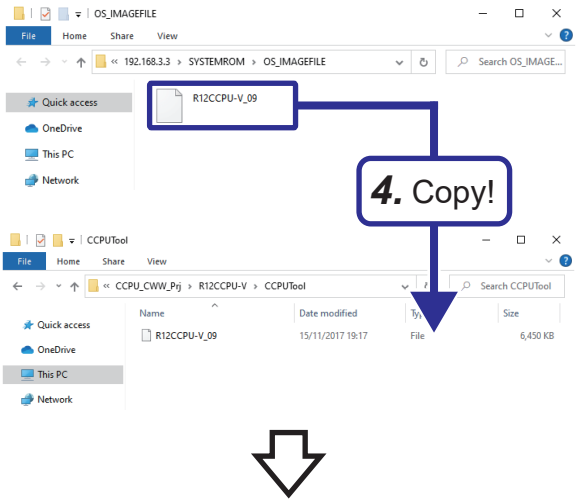
Point

To communicate with the C Controller module using a personal computer, the same VxWorks image file must be specified on both sides.



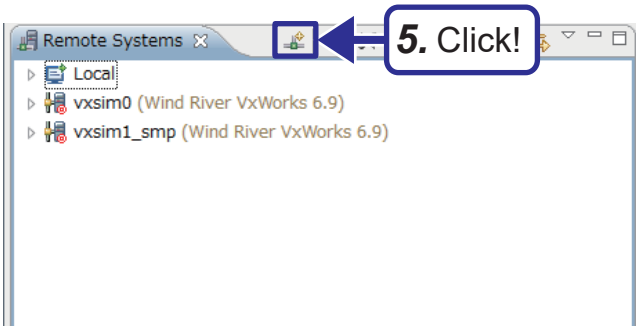
2. In the "Log on as" window, enter the following user name and password.
[Setting details]
User name: target
Password: password

3. Click the [Log On] button.

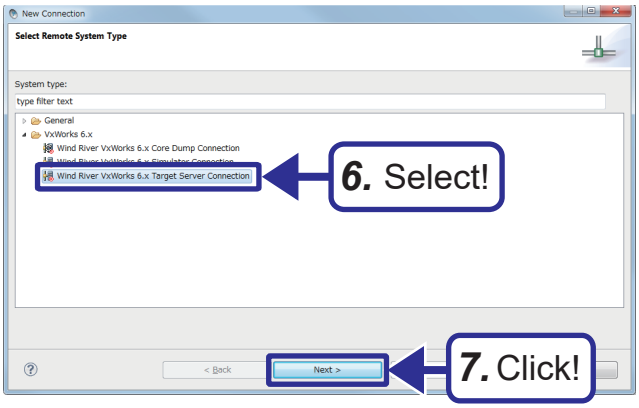


4. Create the "C:\CCPU_CWW_Prj\R12CCPU-V\CCPUTool" folder and copy the VxWorks image file*1 stored in the C Controller module to "C:\CCPU_CWW_Prj\R12CCPU-V\CCPUTool".

*1 The file name is "R12CCPU-V_XX". "XX" at the end of the file name is the upper two digits of the serial number.

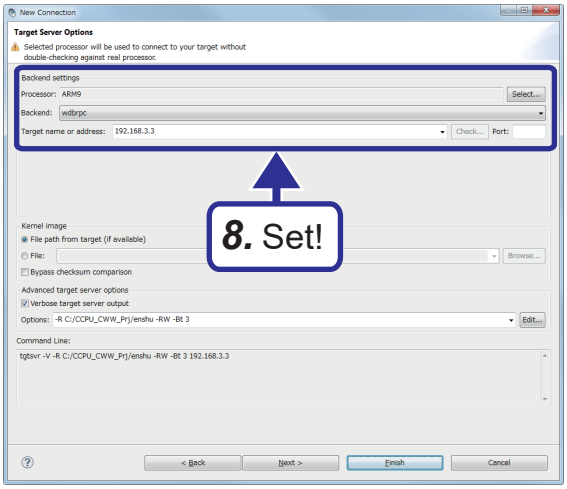


5. Click in the Remote Systems window.

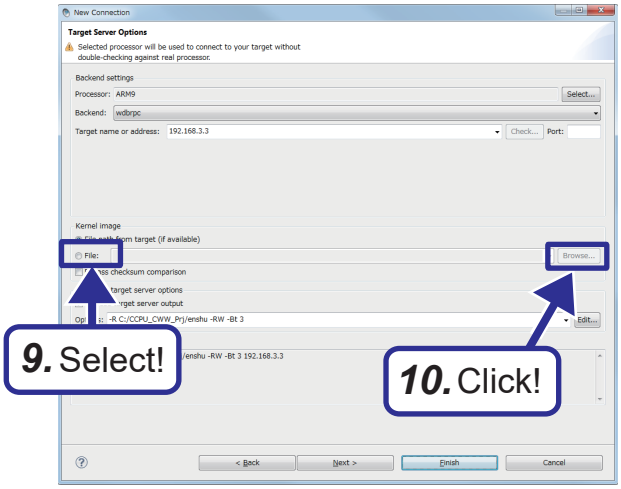


6. Select "Wind River VxWorks 6.x Target Server Connection" in the "New Connection" window.

7. Click the [Next] button.

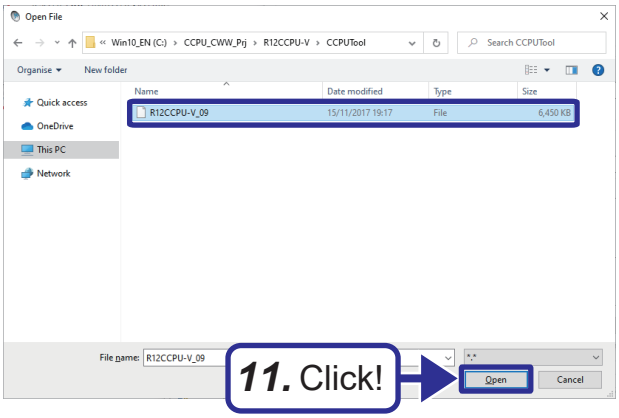


8. For the setting items in "Backend settings", set the following.
[Setting details]
Processor: ARM9 (Click the [Select] button to select from the tree.)
Backend: wdbrpc
IP Address: 192.168.3.3 (Default)
Port: Blank

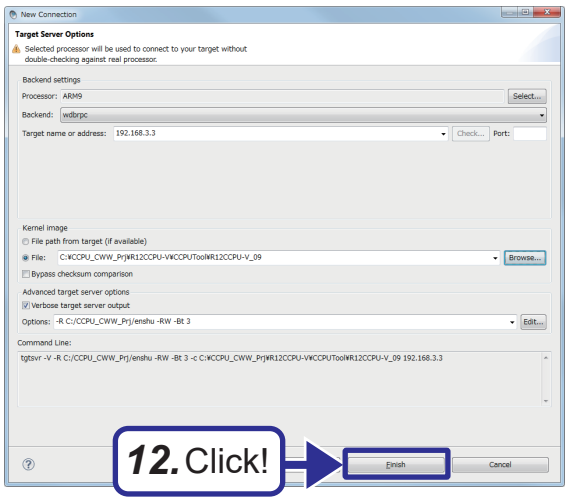


9. Select "File" for "Kernel image".

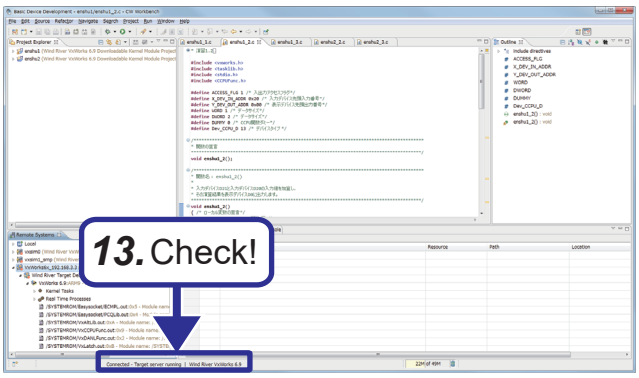
10. Click the [Browse] button.



11. Select the VxWorks image file (C:\CCPU_CWW_Prj\R12CCPU-V\CCPUTool) copied in step **4.** from the tree, and click the [Open] button.



12. Click the [Finish] button.

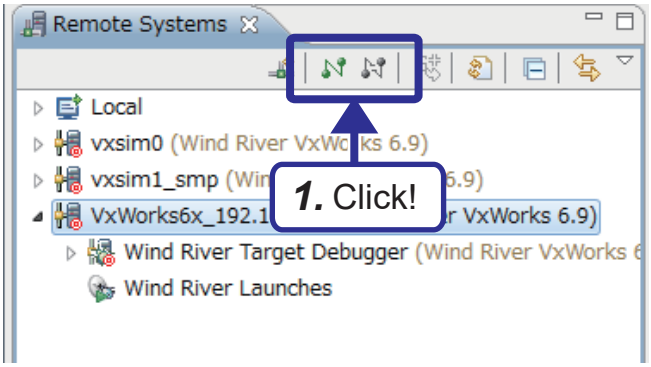


13. When "connected - target server running" is displayed at the bottom of the Remote Systems window, the connection is completed.

Point

If "connected - target server running" is not displayed, check that the C Controller module is powered on correctly, and try again from ☞ Page 69 Connecting the C Controller module and CW Workbench.

■Connecting and disconnecting



1. To connect or disconnect the created connection destination, click the applicable button in the Remote Systems window.

: Connect

: Disconnect

Debugging the user program

Check that the created program operates correctly.

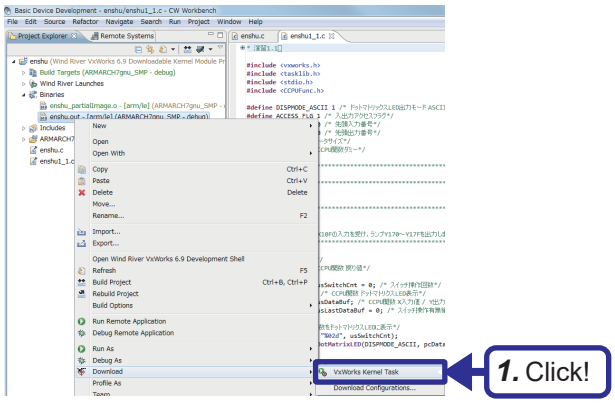
■Downloading the user program into the C Controller module

For debugging, download the module for execution into the memory of the C Controller module.
When the module for execution is downloaded, the program can be executed without a script file.

Point

Script file: A file in which information such as the load destination of the user program to be activated when the C Controller module starts up and the order of startup is described

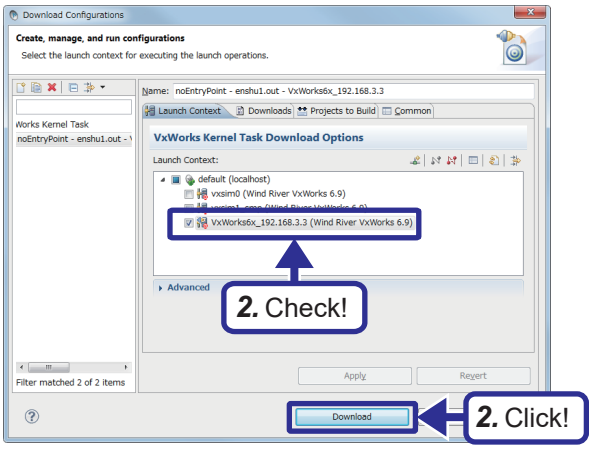
Operating procedure



1. In the Project Explorer window, select the created module file "enshu1.out", and right-click the mouse ⇨ select [Download] ⇨ [VxWorks Kernel Task].

Point

- To debug a program in ☞ Page 119 Exercise 2 A/D conversion, D/A conversion, enter "enshu2.out".
- When operating in Non Debug mode, two files with enshu☐.out in the file name are displayed. For the file for debug mode, "-debug" is displayed at the end.

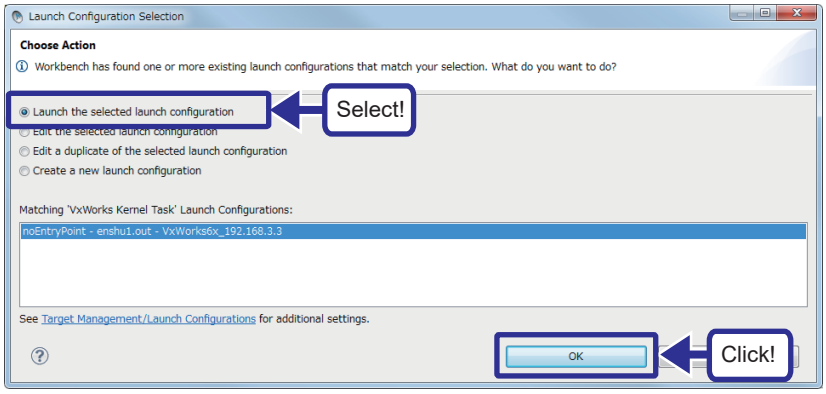


2. In the [Launch Context] tab, select only the checkbox for "VxWorks6x_192.168.3.3(Wind River VxWorks 6.9)", and click the [Download] button.

Point

When performing the operation in 2 for the second time or later, the "Launch Configuration Selection" window is displayed.

Select "Launch the selected launch configuration" and click the [OK] button.

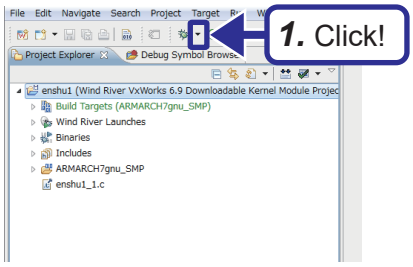


■Debugging the user program

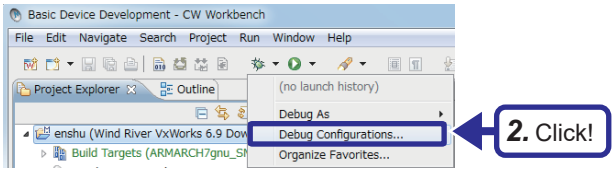
Check that the debug mode is applied to the project before operating the project.

☞ Page 60 Setting properties for the project

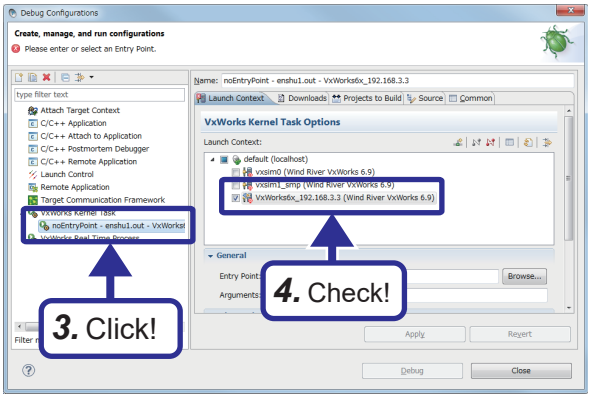
Operating procedure



1. In the Project Explorer window, select the created project and click the [▼] button on the right of the toolbar .



2. Click [Debug Configurations] on the menu.

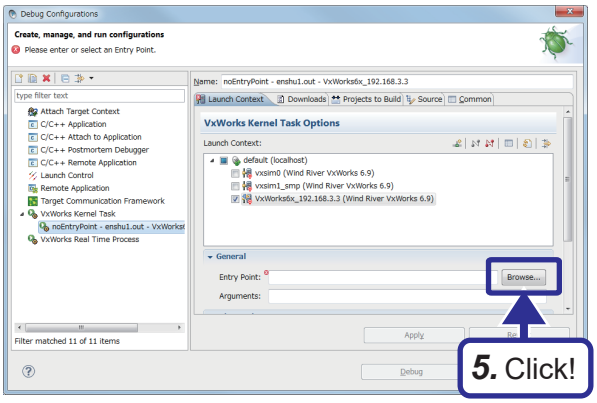


3. Click the module "enshu1.out" downloaded from "VxWorks Kernel Task".

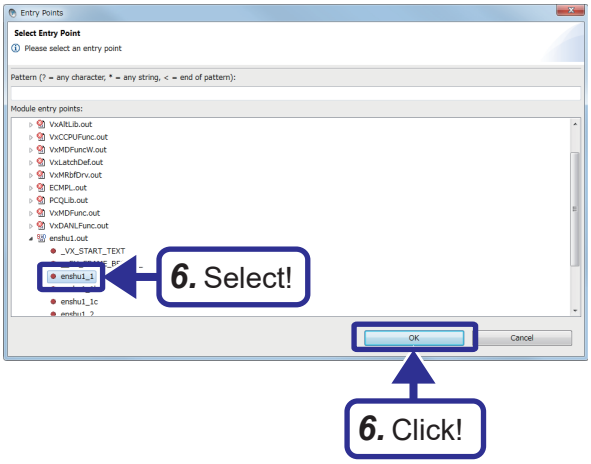
Point

To debug a program in ☞ Page 119 Exercise 2 A/D conversion, D/A conversion, enter "enshu2.out".

4. Select the checkbox for the target server indicating connection with the C Controller module.



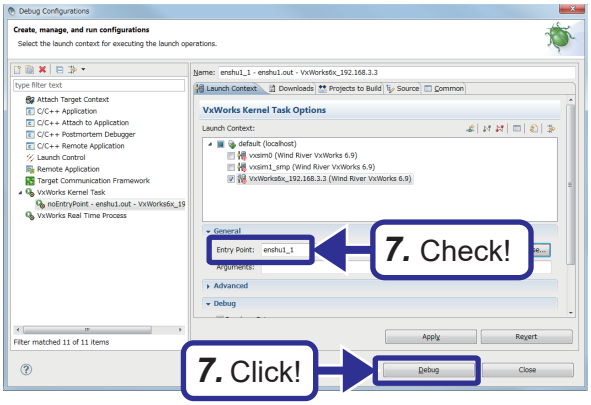
5. Click the [Browse] button.



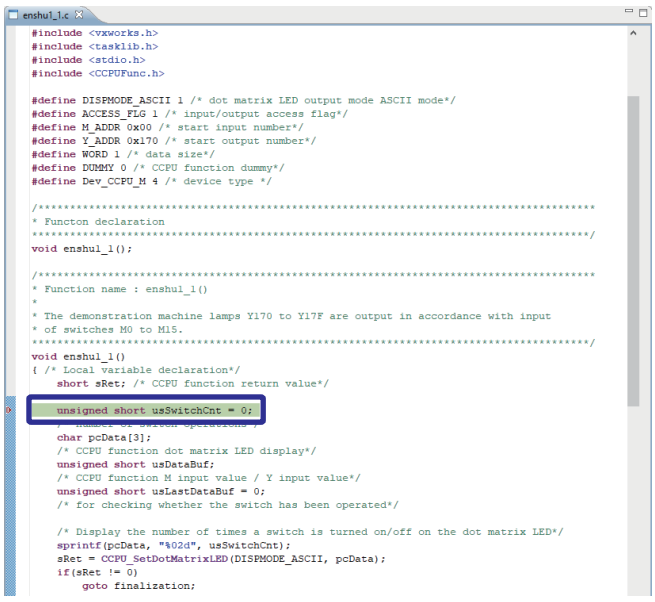
6. Select the debug start function "enshu1_1" and click the [OK] button.

Point

For the debug start function, name it according to the program name used for each exercise.
Example: For exercise 2.2, select "enshu2_2".

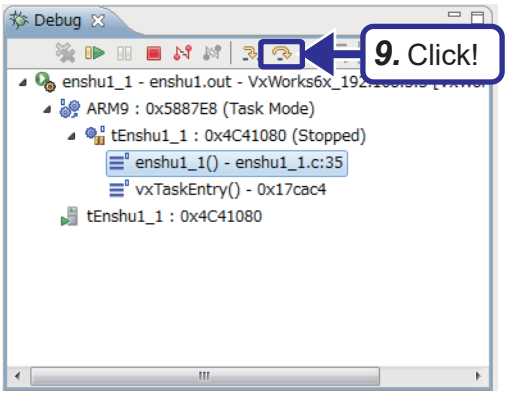


7. Check that the function name selected in step **6.** is set for "Entry Point", and click the [Debug] button.

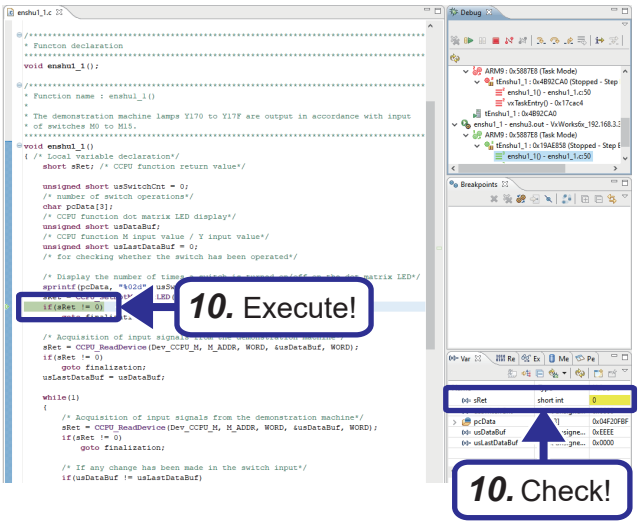


```
enshu1_1.c
#include <vxworks.h>
#include <tasklib.h>
#include <stdio.h>
#include <CCPUFunc.h>

#define DISPMODE_ASCII 1 /* dot matrix LED output mode ASCII mode*/
#define ACCESS_FLG 1 /* input/output access flag*/
#define M_ADDR 0x00 /* start input number*/
#define Y_ADDR 0x170 /* start output number*/
#define WORD 1 /* data size*/
#define DUMMY 0 /* CCPU function dummy*/
#define Dev_CCPU_M 4 /* device type */

/*****************************************************************************
* Funcion declaration
*****************************************************************************/
void enshu1_1();

/*****************************************************************************
* Function name : enshu1_1()
*
* The demonstration machine lamps Y170 to Y17F are output in accordance with input
* of switches M0 to M15.
*****************************************************************************/
void enshu1_1()
{ /* Local variable declaration*/
    short sRet; /* CCPU function return value*/

    unsigned short usSwitchCnt = 0;
    char pcData[3];
    /* CCPU function dot matrix LED display*/
    unsigned short usDataBuf;
    /* CCPU function M input value / Y input value*/
    unsigned short usLastDataBuf = 0;
    /* for checking whether the switch has been operated*/

    /* Display the number of times a switch is turned on/off on the dot matrix LED*/
    sprintf(pcData, "%02d", usSwitchCnt);
    sRet = CCPU_SetDotMatrixLED(DISPMODE_ASCII, pcData);
    if(sRet != 0)
        goto finalization;
```

8. Debugging starts and program execution stops at the head of the function specified for "Entry Point".



9. Use the [1 step execution] button in the Debug window to perform debugging one step at a time.



10. Click the Variables window tab at the lower right of the window to check or change the value of each variable. Here, check that the return value for the CCPU function "sRet" is 0 (normal value).

· Using step execution in step **9.**, debug up to the line "→" points to.

· In the [Variables] tab, check that the value of sRet is 0 (normal value).

11. Repeat steps **9.** and **10.**, and debug the created program entirely.

Point

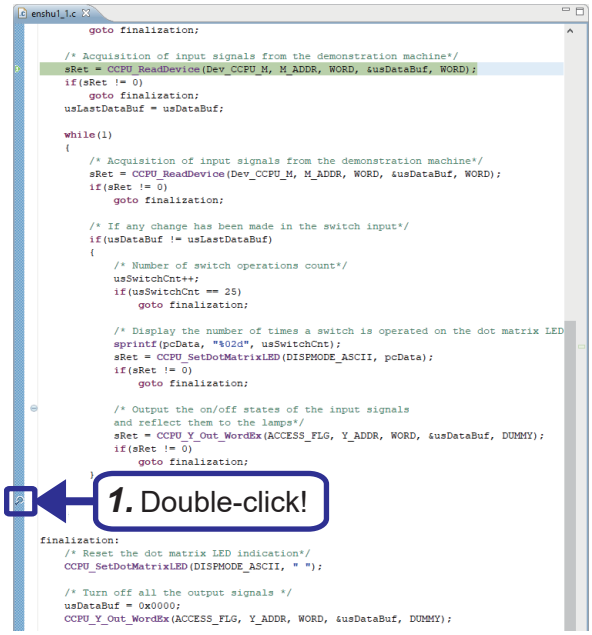
If the return value for the C Controller module dedicated function is not 0, perform troubleshooting by referring to the following.

C Controller Module Programming Manual

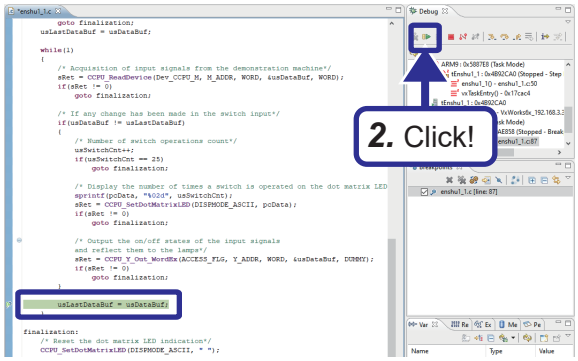
■Debug procedure using Breakpoint

Instead of debugging one step at a time, debugging can be performed by specifying Breakpoint at any point in the program.

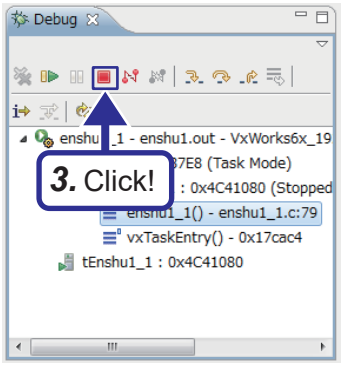
Operating procedure



1. Double-click the left end of the source file to insert Breakpoint.



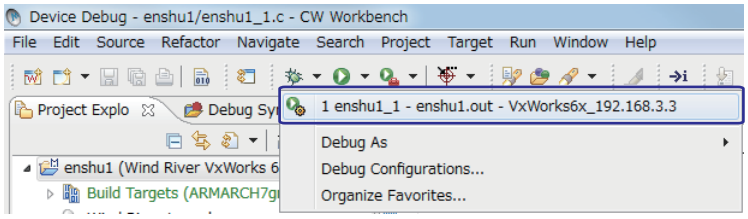
2. Click .
The program is executed up to the point specified by Breakpoint.



3. Click in the Debug window.

Point

To start debugging again, click the [▼] button on the right of the toolbar , and select the already generated debug structure at the top of the popup window displayed.

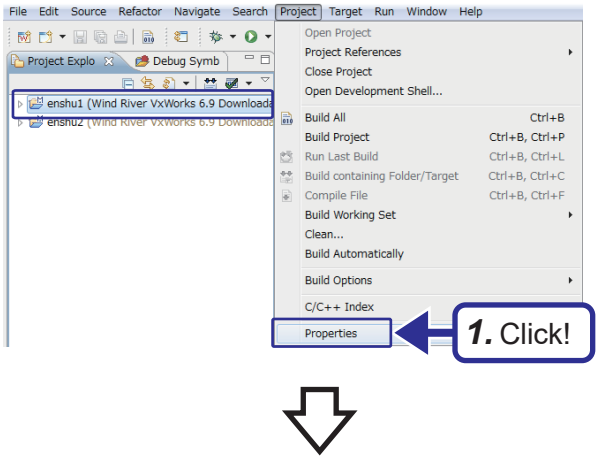


Registering a module for execution

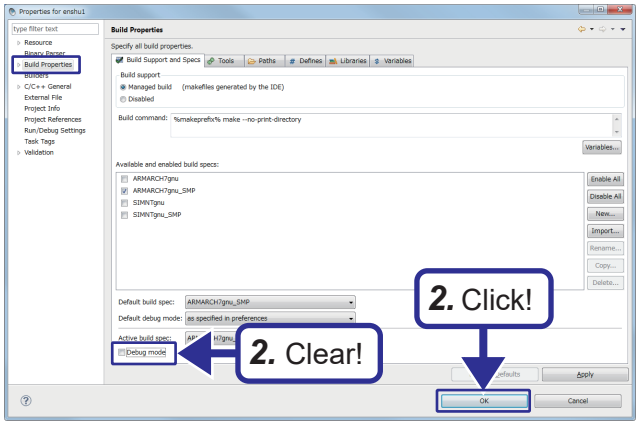
Build the created program into a program for operation, and store it in the C Controller module.

■Building a user program

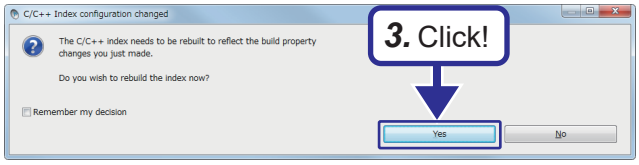
Operating procedure



1. In the Project Explorer window, select the created project and click [Project] on the menu ⇨ [Properties].



2. Select "Build Properties" from the tree on the left of the window, clear the checkbox for "Debug mode", and click the [OK] button.

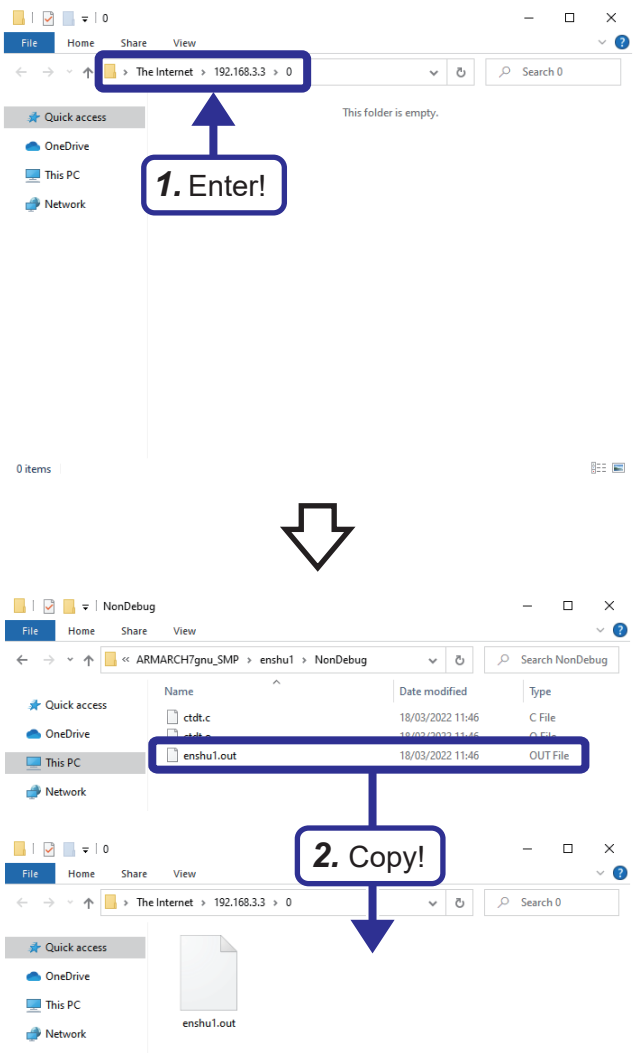


3. When the following message appears after the [OK] button is clicked, click the [Yes] button.

4. Build according to ☞ Page 67 Generating a module for execution.

■Storing the user program

Operating procedure



1. Start up Explorer and enter the following for the address field of the C Controller module.
ftp://192.168.3.3/0

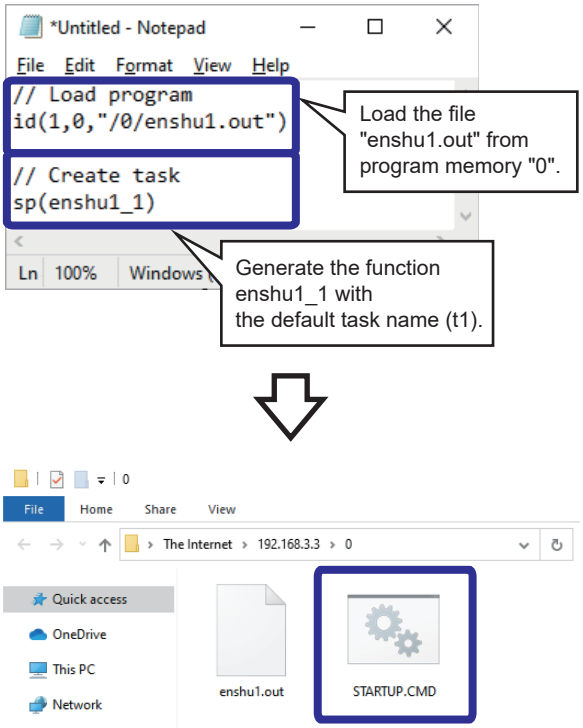
2. Drag and drop the created user program "enshu1.out" to the program memory "0" in the C Controller module to copy it.
The created user program "enshu1.out" is stored under the following.
C:\CCPU_CWW_Prj\enshu\enshu1\ARMARCH7gnu_SMP\enshu1\NonDebug

Point

To copy a program in ☞ Page 119 Exercise 2 A/D conversion, D/A conversion, enter "enshu2.out".

■Creating and storing a script file

Operating procedure



1. Open a text file and write the content of a script file for loading a user program and generating a task as shown in the window.
2. Save the file with the file name "STARTUP.CMD".
3. Copy the created script file to the following program memory in the C Controller module.
   ftp://192.168.3.3/0

Point

A user program and script file can be stored not only in the program memory, but also in the SD memory card. If a script file is stored in both areas, the script file in the SD memory card starts up first.

Checking the operation

Execute the program registered in the C Controller module to check its operation.

For operation, use the RESET/STOP/RUN switch on the front of the C Controller module.

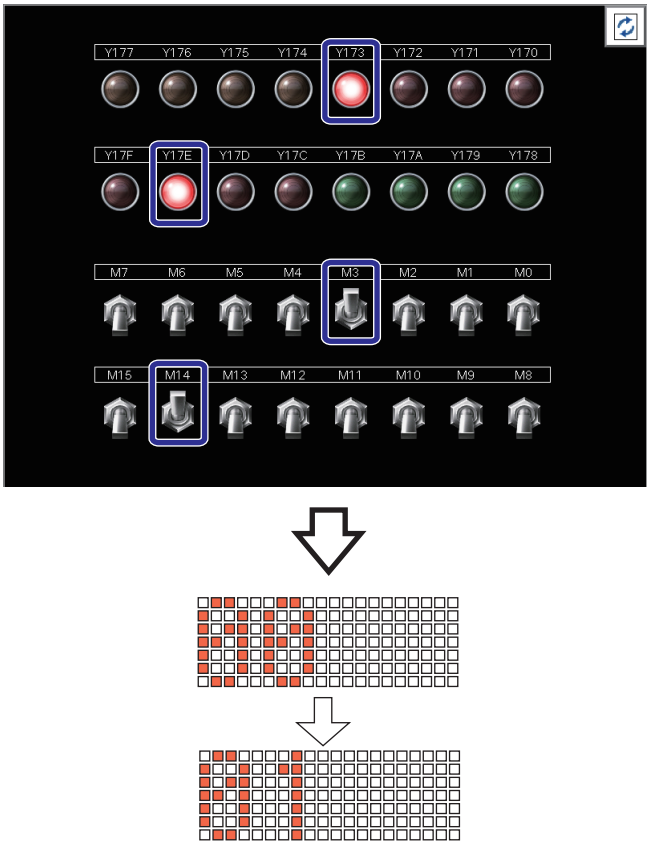
The applications of the RUN/STOP/RESET switch are as follows.

- RUN: Output (Y) from the user program, allowed to be written to the buffer memory
- STOP: Output (Y) from the user program, not allowed to be written to the buffer memory
- RESET: Reset the module

Point

- Program operation in the C Controller module is executed regardless of whether the status of the switch is RUN or STOP.
- For details on the RUN/STOP/RESET switch, refer to the following.
  (MELSEC iQ-R C Controller Module User's Manual (Startup))

Operating procedure



1. Reset the C Controller module and set the switch on the front to the RUN position.
   For details on the reset procedure, refer to the following.
   Page 21 Reset operation procedure
2. When switches M0 to M15 of the demonstration machine are turned on/off, lamps Y170 to Y17F with the same bit numbers as those switches turn on/off.
3. The number of times a switch is turned on/off is displayed on the dot matrix LED of the C Controller module.
4. In addition, the processing is finished when the number of switch operations exceeds 25, and the lamp output and dot matrix LED display are reset.

Source code

The following shows the source code of the program enshu1_1.

```
/*****************************************************************************************
* Exercise 1.1
*
* This is a sample program in which the C language controller outputs
* the demonstration machine lamps Y170 to Y17F in accordance with input of
* demonstration machine switches M0 to M15.
*
*****************************************************************************************/

#include <vxworks.h>
#include <tasklib.h>
#include <stdio.h>
#include <CCPUFunc.h>

#define DISPMODE_ASCII 1 /* dot matrix LED output mode ASCII mode*/
#define ACCESS_FLG 1 /* input/output access flag*/
#define M_ADDR 0x00 /* start input number*/
#define Y_ADDR 0x170 /* start output number*/
#define WORD 1 /* data size*/
#define DUMMY 0 /* CCPU function dummy*/
#define Dev_CCPU_M 4 /* device type */

/*****************************************************************************************
* Funcion declaration
*****************************************************************************************/
void enshu1_1();

/*****************************************************************************************
* Function name : enshu1_1()
*
* The demonstration machine lamps Y170 to Y17F are output in accordance with input
* of switches M0 to M15.
*****************************************************************************************/
void enshu1_1()
{ /* Local variable declaration*/
    short sRet; /* CCPU function return value*/

    unsigned short usSwitchCnt = 0;
    /* number of switch operations*/
    char pcData[3];
    /* CCPU function dot matrix LED display*/
    unsigned short usDataBuf;
    /* CCPU function M input value / Y input value*/
    unsigned short usLastDataBuf = 0;
    /* for checking whether the switch has been operated*/

    /* Display the number of times a switch is turned on/off on the dot matrix LED*/
    sprintf(pcData, "%02d", usSwitchCnt);
    sRet = CCPU_SetDotMatrixLED(DISPMODE_ASCII, pcData);
    if(sRet != 0)
        goto finalization;

    /* Acquisition of input signals from the demonstration machine*/
    sRet = CCPU_ReadDevice(Dev_CCPU_M, M_ADDR, WORD, &usDataBuf, WORD);
    if(sRet != 0)
        goto finalization;
    usLastDataBuf = usDataBuf;

    while(1)
    {
        /* Acquisition of input signals from the demonstration machine*/
        sRet = CCPU_ReadDevice(Dev_CCPU_M, M_ADDR, WORD, &usDataBuf, WORD);
        if(sRet != 0)
            goto finalization;

        /* If any change has been made in the switch input*/
        if(usDataBuf != usLastDataBuf)
```

```
        {
            /* Number of switch operations count*/
            usSwitchCnt++;
            if(usSwitchCnt == 25)
                goto finalization;

            /* Display the number of times a switch is operated on the dot matrix LED*/
            sprintf(pcData, "%02d", usSwitchCnt);
            sRet = CCPU_SetDotMatrixLED(DISPMODE_ASCII, pcData);
            if(sRet != 0)
                goto finalization;

            /* Output the on/off states of the input signals
            and reflect them to the lamps*/
            sRet = CCPU_Y_Out_WordEx(ACCESS_FLG, Y_ADDR, WORD, &usDataBuf, DUMMY);
            if(sRet != 0)
                goto finalization;
        }

        usLastDataBuf = usDataBuf;
    }

finalization:
    /* Reset the dot matrix LED indication*/
    CCPU_SetDotMatrixLED(DISPMODE_ASCII, " ");

    /* Turn off all the output signals */
    usDataBuf = 0x0000;
    CCPU_Y_Out_WordEx(ACCESS_FLG, Y_ADDR, WORD, &usDataBuf, DUMMY);

    return;
}
```

Exercise 1.2 Input device and display device

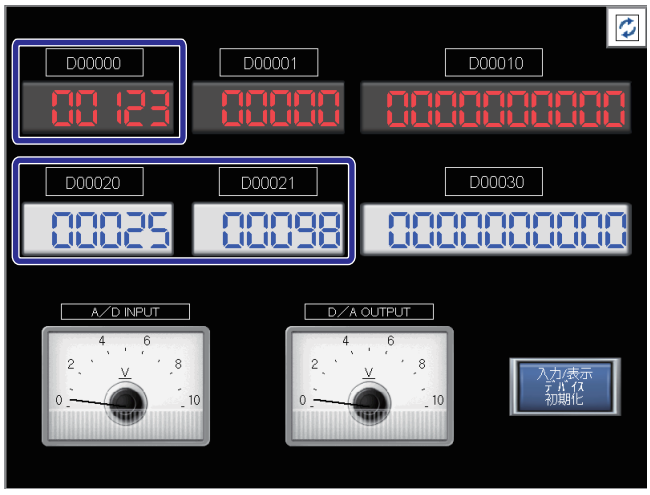
Add the input values of the input devices D20 and D21 of the demonstration machine, and display the result in the display device D0.

Operating procedure

1. Copy the program enshu1_2.c to the project folder C:\CCPU_CWW_Prj\enshu\enshu1 to add it to the project. (☞ Page 66 Procedure for adding a program)
2. Rebuild enshu1_2.c in debug mode. (☞ Page 67 Generating a module for execution)
3. Connect the C Controller module and CW Workbench. (☞ Page 72 Connecting and disconnecting)
4. Debug the created program to check if it operates correctly. (☞ Page 73 Debugging the user program)
5. Rebuild the program enshu1_2.c that underwent debugging by canceling the debug mode, and store the created user program on the C Controller module. (☞ Page 80 Registering a module for execution)
6. Create a script and store it in the C Controller module. (☞ Page 82 Registering a module for execution)
   [Script details]
   Load a program: ld (1, 0, "/0/enshu1.out")
   Generate a task: sp (enshu1_2)
7. Reset the C Controller module and set the switch on the front to the RUN position.
8. Change the values of the input devices D20 and D21 of the demonstration machine, and check the change of the value of the display device D0. (☞ Page 86 Checking the operation)
9. After the operation is checked completely, disconnect CW Workbench from the C Controller module (☞ Page 72 Connecting and disconnecting) once, and delete the user program and script stored in the C Controller module.

Checking the operation

Operating procedure



1. Reset the C Controller module and set the switch on the front to the RUN position.
2. Input values to the input devices D20 and D21 of the demonstration machine, and the addition result is displayed in the display device D0.

Source code

The following shows the source code of the program enshu1_2.

```
/*****************************************************************************************
* Exercise 1.2
*
* This is a sample program in which the C language controller adds the input values
* of the input device D21 and D20 of the demonstration machine,
* and displays the result on the display device D0.
*
*****************************************************************************************/

#include <vxworks.h>
#include <tasklib.h>
#include <stdio.h>
#include <CCPUFunc.h>

#define D_DEV_IN_ADDR 20 /* start input number of the input device*/
#define D_DEV_OUT_ADDR 00 /* start output number of the display device*/
#define WORD 1 /* data size*/
#define DWORD 2 /* data size*/
#define DUMMY 0 /* CCPU function dummy*/
#define Dev_CCPU_D 13 /* device type */

/*****************************************************************************************
* Funcion declaration
*****************************************************************************************/
void enshu1_2();

/*****************************************************************************************
* Function name : enshu1_2()
*
* The input values of the input devices D20 and D21 are added,
* and the operation result is displayed in the display device D0.
*****************************************************************************************/
void enshu1_2()
{ /* Local variable declaration*/
    short sRet;
    /* CCPU function return value*/
    unsigned short pusDataBuf[2];
    /* CCPU function input device value, display device value*/
    unsigned short pusLastDataBuf[2] = {0, 0};
    /* for checking whether the switch has been operated*/
    unsigned short usDataBuf;
    /*  addition result*/

    /* Acquisition of the input device value from the demonstration machine*/
    sRet = CCPU_ReadDevice(Dev_CCPU_D, D_DEV_IN_ADDR, DWORD, pusDataBuf, DWORD);
    if(sRet != 0)
        goto finalization;

    /* Addition operation*/
    usDataBuf = pusDataBuf[0] + pusDataBuf[1];

    /* Output to the display device of the demonstration machine*/
    sRet = CCPU_WriteDevice(Dev_CCPU_D, D_DEV_OUT_ADDR, WORD, &usDataBuf, WORD);
    if(sRet != 0)
        goto finalization;

    pusLastDataBuf[0] = pusDataBuf[0];
    pusLastDataBuf[1] = pusDataBuf[1];

    while(1)
    {
        /* Acquisition of the input device value from the demonstration machine*/
        sRet = CCPU_ReadDevice(Dev_CCPU_D, D_DEV_IN_ADDR, DWORD, pusDataBuf, DWORD);
        if(sRet != 0)
            goto finalization;
```

```
        /* If any change has been made in the switch input*/
        if(pusDataBuf[0] != pusLastDataBuf[0] || pusDataBuf[1] != pusLastDataBuf[1])
        {
           /* Addition operation*/
            usDataBuf = pusDataBuf[0] + pusDataBuf[1];

           /* Output to the display device of the demonstration machine*/
           sRet = CCPU_WriteDevice(Dev_CCPU_D, D_DEV_OUT_ADDR, WORD, &usDataBuf, WORD);
           if(sRet != 0)
               goto finalization;
        }

        pusLastDataBuf[0] = pusDataBuf[0];
        pusLastDataBuf[1] = pusDataBuf[1];
    }

finalization:
     /* Turn off all the output signals */
     usDataBuf = 0x00000000;
     sRet = CCPU_WriteDevice(Dev_CCPU_D, D_DEV_OUT_ADDR, WORD, &usDataBuf, WORD);

    return;
}
```

Exercise 1.3 Executing two programs

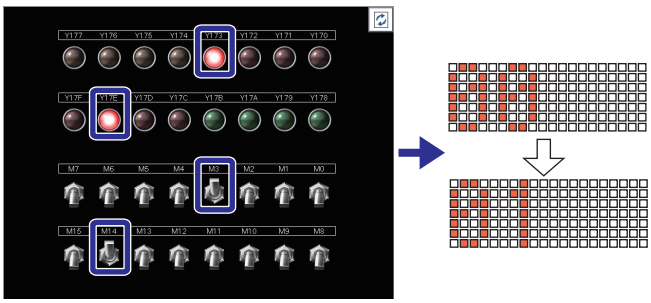
Start up a task for executing enshu1_1() and a task for executing enshu1_2() at the same time.
In exercise 1.3, check that two tasks can operate at the same time. Generate two tasks, tEnshu1_1 and tEnshu1_2, with the API function of VxWorks taskSpawn(), so that tEnshu1_1 executes enshu1_1() and tEnshu1_2 executes enshu1_2().

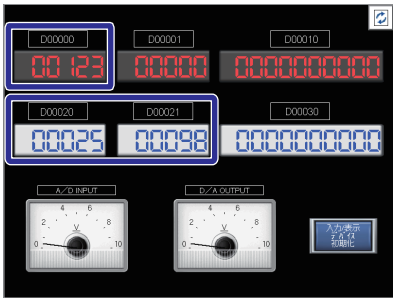
Operating procedure

1. Copy the program enshu1_3.c to the project folder C:\CCPU_CWW_Prj\enshu\enshu1 to add it to the project. (☞ Page 66 Procedure for adding a program)
2. Rebuild enshu1_3.c in debug mode. (☞ Page 67 Generating a module for execution)
3. Connect the C Controller module and CW Workbench. (☞ Page 72 Connecting and disconnecting)
4. Debug the created program to check if it operates correctly. (☞ Page 73 Debugging the user program)
5. Rebuild the program enshu1_3.c that underwent debugging by canceling the debug mode, and store the created user program on the C Controller module. (☞ Page 80 Registering a module for execution)
6. Create a script and store it in the C Controller module. (☞ Page 81 Storing the user program)
   [Script details]
   Load a program: ld (1, 0, "/0/enshu1.out")
   Generate a task: sp (enshu1_3)
7. Reset the C Controller module and set the switch on the front to the RUN position.
8. Check that the operations in exercises 1.1 and 1.2 can be executed at the same time. (☞ Page 83 Checking the operation)
9. After the operation is checked completely, disconnect CW Workbench from the C Controller module (☞ Page 72 Connecting and disconnecting) once, and delete the user program and script stored in the C Controller module.

Checking the operation

Operating procedure





1. Reset the C Controller module and set the switch on the front to the RUN position.
2. The operations in exercises 1.1 and 1.2 can be executed at the same time.

Source code

The following shows the source code of the program enshu1_3.

```
#include <vxworks.h>
#include <tasklib.h>

extern void enshu1_1();
extern void enshu1_2();

void enshu1_3()
{
	/* Generate the task to execute enshul_1*/
	taskSpawn(
		"tEnshul_1", /* task name*/
		101, /* task priority*/
		VX_FP_TASK, /* task option*/
		4096, /* stack size*/
		(FUNCPTR)enshul_1, /* entry point*/
		0, 0, 0, 0, 0, 0, 0, 0, 0, 0 /* specify the argument for the function to be executed*/
	);

	/* Generate the task to execute enshul_2*/
	taskSpawn(
		"tEnshul_2", /* task name*/
		255, /* task priority*/
		VX_FP_TASK, /* task option*/
		4096, /* stack size*/
		(FUNCPTR)enshul_2, /* entry point*/
		0, 0, 0, 0, 0, 0, 0, 0, 0, 0 /* specify the argument for the function to be executed*/
	);

	return;
}
```

5 OPERATING INTELLIGENT FUNCTION MODULES

5.1 Intelligent Function Modules

Types of intelligent function modules

An intelligent function module is a module that can realize functions that cannot be provided by or whose applications are limited in use for the C Controller module.

Therefore, intelligent function modules with functions required for specific purposes can be selected for use.

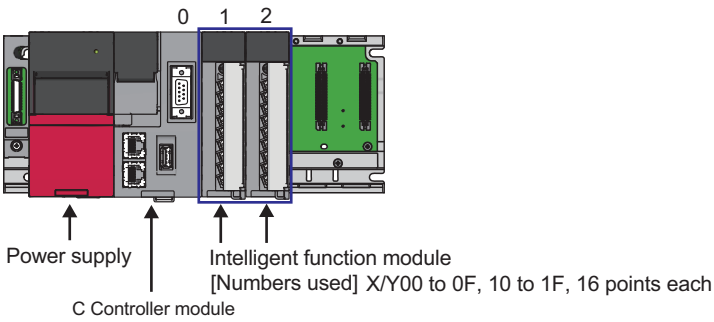
The C Controller module can use MELSEC iQ-R series intelligent function modules.

The following table lists examples of modules.

Name	Number of occupied I/O points	Function	Internal current consumption (5VDC)
Analog-digital converter module (R60AD4)	16 points (I/O assignment: intelligent 16 points)	0 to 20mADC → 0 to 32000 0 to ±10V → 0 to ±32000 Input module for the above conversions	0.22A
Digital-analog converter module (R60DA4)	16 points (I/O assignment: intelligent 16 points)	0 to 32000 → 0 to 20mADC ±32000 → 0 to ±10V Output module for the above conversions	0.16A

Combination with the C Controller module

An intelligent function module is used while mounted on any I/O slot of the main base unit and extension base unit.

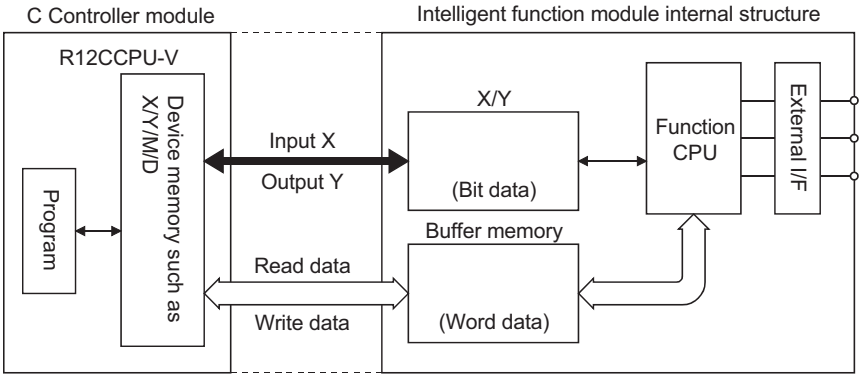


5.2 Exchange of Information between Intelligent Function Modules and the C Controller Module

Roughly speaking, two types of information is exchanged.

Information in units of bits--Signals using input and output X, Y

Information in units of words--16-bit or 32-bit data



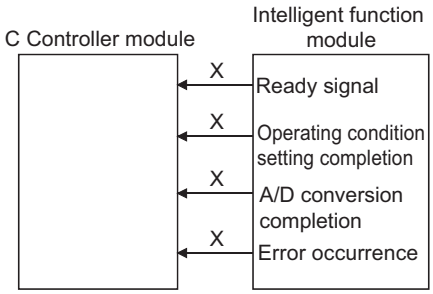
I/O signals for the C Controller module

Of signals sent/received between the C Controller module and intelligent function modules, those in units of bits use input and output X, Y.

X, Y here are different from external input and output, and used for sequence programs as signals specific to the intelligent function module. Note that for I/O numbers, numbers assigned according to the mounting position of the intelligent function module are used.

Input signal X from the intelligent function module

X used in a program refers to a signal to be input from the intelligent function module to the C Controller module and originates from the intelligent function module side. In a program, as bit information for checking the module status, this signal is read by calling the bus interface function (CCPU_X_In_BitEx() or CCPU_X_In_WordEx()).

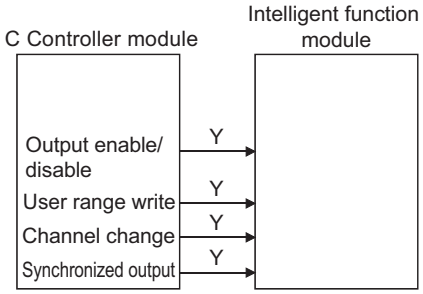


For example, input signals from an analog-digital converter module are as follows.

- Ready signal
  This signal inputs the information that the intelligent function module operates normally and is ready to the CPU when the power is turned on.
- A/D conversion completion flag
  This signal is turned on when conversion is completed at all the conversion enabled channels. When reading a digital output value, this signal is used as an interlock.

Output signal Y to the intelligent function module

This signal is output to the intelligent function module by calling the bus interface function (CCPU_Y_Out_BitEx() or CCPU_Y_Out_WordEx()) and originates from the C Controller module side.

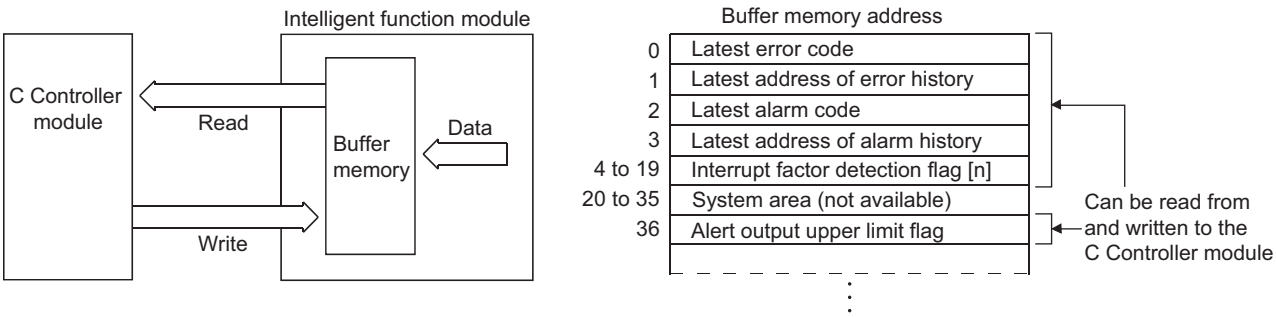


For example, an output signal to an analog-digital converter module is as follows.

- CH☐ Output enable/disable flag
  This signal specifies whether to output an analog value converted through D/A conversion or offset value, per channel.

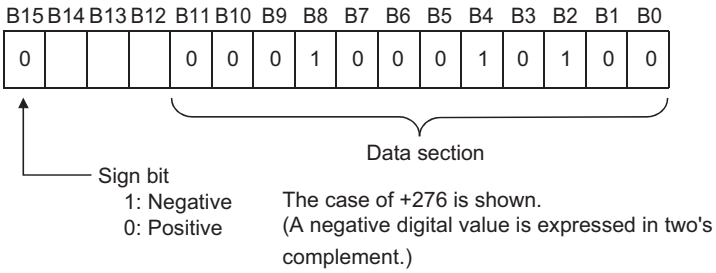
Data communication with an intelligent function module

Data is sent/received in units of 16 bits or 32 bits. The intelligent function module has buffer memory to store data.



Memory map example: Digital-analog converter module R60DA4

- Data can be read from/written to the buffer memory by calling the C Controller module dedicated function (CCPU_FromBuf() or CCPU_ToBuf()).
- The buffer memory has an address specific to each intelligent function module for each word (short type (16 bits)).
  The addresses of the buffer memory start from 0, and data is read/written by specifying them. The minimum unit of measurement is 1 word. 17 to 32-bit data is handled by using 2 words (32 bits).
  The following figure shows an example of 16 bits for the digital-analog converter module; the digital quantity is written from the CPU. In the contents, a digital value of -32000 to 32000 is set using 16-bit signed binary data.



Point

RAM is used for the buffer memory.

5.3 Methods of Communication with Intelligent Function Modules

Types of methods of communication with intelligent function modules

The C Controller can perform communications with intelligent function modules by the following methods.

Communication method		Function
CW Configurator	Intelligent function module monitor	Monitors I/O signals of the intelligent function module and the buffer memory, and writes data.
	Intelligent function module parameter	Writes the initial settings for the parameters of the intelligent function module to the C Controller module, and automatically reflects them in the buffer memory of each intelligent function module when the C Controller module starts up.
C Controller module dedicated functions	CCPU_X_In_BitEx	Reads input signals X of the intelligent function module in units of bits. ☞ Page 50 CCPU_X_In_BitEx
	CCPU_X_In_WordEx	Reads input signals X of the intelligent function module in units of 16 bits. ☞ Page 51 CCPU_X_In_WordEx
	CCPU_Y_In_BitEx	Reads output signals Y of the intelligent function module in units of bits. 📖 MELSEC iQ-R C Controller Module Programming Manual
	CCPU_Y_In_WordEx	Reads output signals Y of the intelligent function module in units of 16 bits. 📖 MELSEC iQ-R C Controller Module Programming Manual
	CCPU_Y_Out_BitEx	Outputs data to output signals Y of the intelligent function module in units of bits. ☞ Page 52 CCPU_Y_Out_BitEx
	CCPU_Y_Out_WordEx	Outputs data to output signals Y of the intelligent function module in units of 16 bits. ☞ Page 53 CCPU_Y_Out_WordEx
	CCPU_FromBuf	Reads values stored in the buffer memory of the intelligent function module in units of 16 bits. For details, refer to the following. ☞ Page 96 CCPU_FromBuf
	CCPU_ToBuf	Writes values to the buffer memory of the intelligent function module in units of 16 bits. For details, refer to the following. ☞ Page 97 CCPU_ToBuf

CCPU_FromBuf

This function reads data from the CPU buffer memory of the CPU module installed in the specified module position and the buffer memory of the intelligent function module installed in the specified module position. (FROM instruction)

■Format

short CCPU_FromBuf (unsigned short usIoNo, unsigned long ulOffset, unsigned long ulSize, unsigned short* pusDataBuf, unsigned long ulBufSize)

■Argument

Argument	Name	Description	IN/OUT
usIoNo	Module position	Specify the module position. Start I/O No. ÷ 16 (0H to FFH, 3E0H to 3E3H)	IN
ulOffset	Offset	Specify the offset in units of words.	IN
ulSize	Data size	Specify the read data size in units of words.	IN
pusDataBuf	Data storage location	Specify the storage location of read data.	OUT
ulBufSize	Data storage size	Specify the data storage location size in units of words.	IN

■Description

- This communication method reads data amounting to the size specified by the data size (ulSize) from the CPU buffer memory of the CPU module specified by the module position (usIoNo) and the buffer memory of the intelligent function module specified by the module position (usIoNo), and stores that data in the data storage location (pusDataBuf).
- Read data by specifying the offset address from the start of the CPU buffer memory of the CPU module and the buffer memory of the intelligent function module each for the offset (ulOffset).
- To access the CPU buffer memory of the multi-CPU (CPU No.1 to CPU No.4), specify 3E0H to 3E3H (CPU No.1 to CPU No.4) for the module position (usIoNo). However, the CPU buffer memory can be accessed only when the multiple CPU settings are configured.

Precautions

For the data storage location size (ulBufSize), set a value larger than the value of the data size (ulSize).

■Return value

Return value	Description
0 (0000H)	Normally finished
Other than 0	Failed For details on when the function fails, refer to the following. 📖 MELSEC iQ-R C Controller Module Programming Manual

CCPU_ToBuf

This function writes data to the CPU buffer memory of the CPU module (host CPU module) installed in the specified module position and the buffer memory of the intelligent function module installed in the specified module position. (TO instruction)

■Format

short CCPU_ToBuf (unsigned short usIoNo, unsigned long ulOffset, unsigned long ulSize, unsigned short* pusDataBuf, unsigned long ulBufSize)

■Argument

Argument	Name	Description	IN/OUT
usIoNo	Module position	Specify the module position as follows. For the CPU buffer memory, access can be made to the host CPU module only. Start I/O No. ÷ 16 (0H to FFH, 3E0H to 3E3H)	IN
ulOffset	Offset	Specify the offset in units of words.	IN
ulSize	Data size	Specify the write data size in units of words.	IN
pusDataBuf	Data storage location	Specify the storage location of write data.	IN
ulBufSize	Data storage size	Specify 0.	IN

■Description

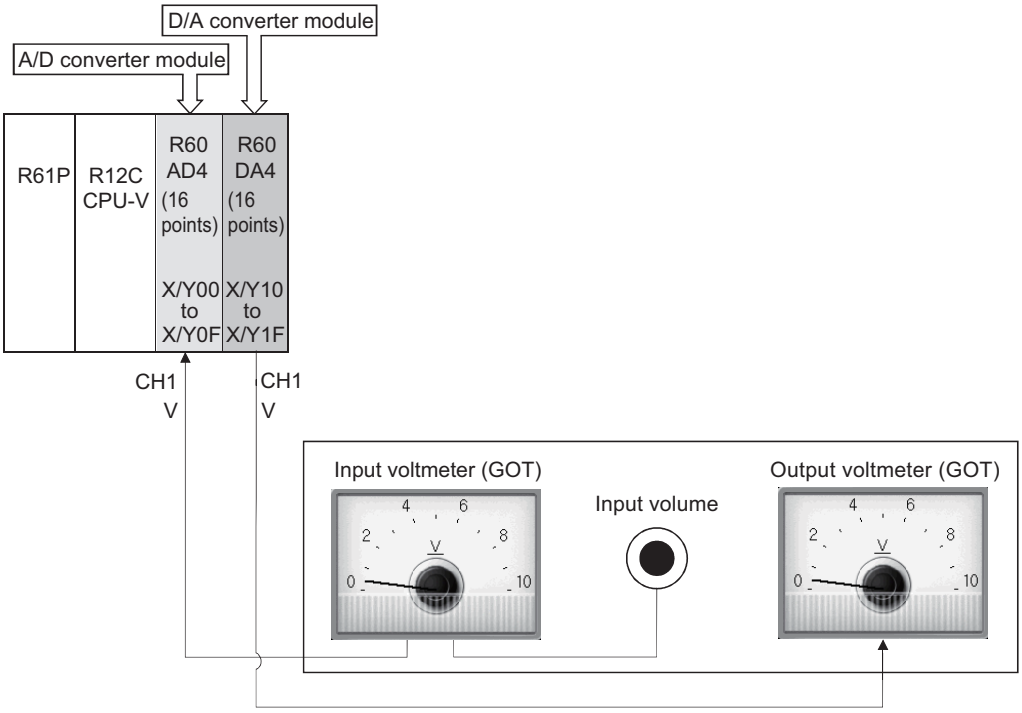
- This communication method writes data in the data storage location (pusDataBuf) amounting to the data size (ulSize) to the CPU buffer memory of the CPU module (host CPU module) specified by the module position (usIoNo) and the buffer memory of the intelligent function module specified by module position (usIoNo).
  Write data by specifying the offset address from the start of the CPU buffer memory of the CPU module (host CPU module) and the buffer memory of the intelligent function module each for the offset (ulOffset).
- To access the CPU buffer memory (host CPU module) of the multi-CPU (CPU No.1 to CPU No.4), specify 3E0H to 3E3H (CPU No.1 to CPU No.4) for the module position (usIoNo). However, the CPU buffer memory (host CPU module) can be accessed only when the multiple CPU settings are configured.
- When the operating status of the CPU module is not RUN, an error in the STOP/PAUSE state (-28640) occurs if the CCPU_ToBuf function is executed.

■Return value

Return value	Description
0 (0000H)	Normally finished
Other than 0	Failed For details on when the function fails, refer to the following. 📖 MELSEC iQ-R C Controller Module Programming Manual

5.4 Exercise Structure for Intelligent Function Modules

Convert analog signals/digital data input by the input volume tab or digital switch of the demonstration machine using the analog-digital converter module/digital-analog converter module.

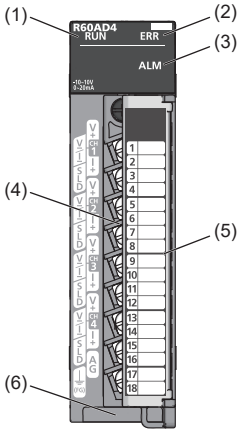


5.5 Analog-Digital Converter Module R60AD4

Part names

This section describes the part names of the R60AD4.

For details, refer to the user's manual.



No.	Name	Description
(1)	RUN LED	Indicates the operating status of the module. On: Normal operation Flashing (every 1s): In offset/gain setting mode Flashing (every 400ms): Selecting a module for online module change Off: 5V power supply turned off or watchdog timer error, ready to change modules during online module change
(2)	ERR LED	Indicates the module error occurrence status.*1 On: Error Off: Normal operation
(3)	ALM LED	Indicates the module alarm status.*1 On: Warning (process alarm or rate alarm) Flashing: Input signal error detected Off: Normal operation
(4)	Terminal block	18-point screw terminal block. Connect input signal wires of a device such as an external device.
(5)	Terminal block cover	Cover for preventing electric shocks when the power is turned on
(6)	Production information marking	Shows the production information (16 digits) of the module.

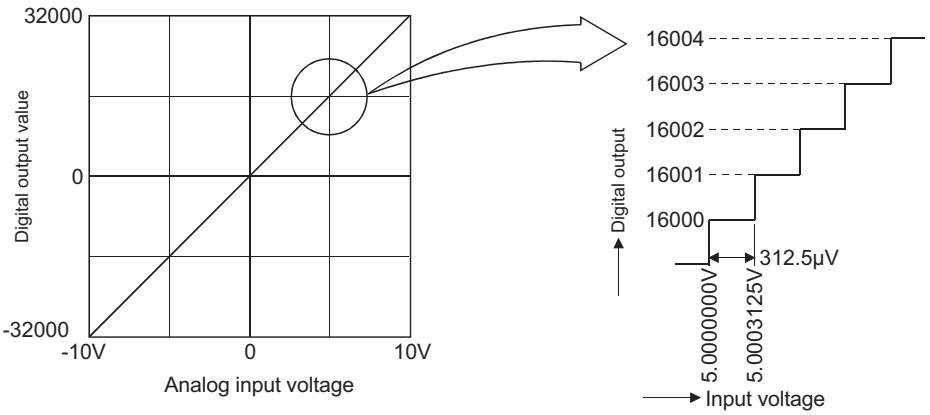
*1 For details, refer to the following.

MELSEC iQ-R Analog-Digital Converter Module User's Manual (Application)

A/D conversion characteristics

A/D conversion characteristics for voltage input

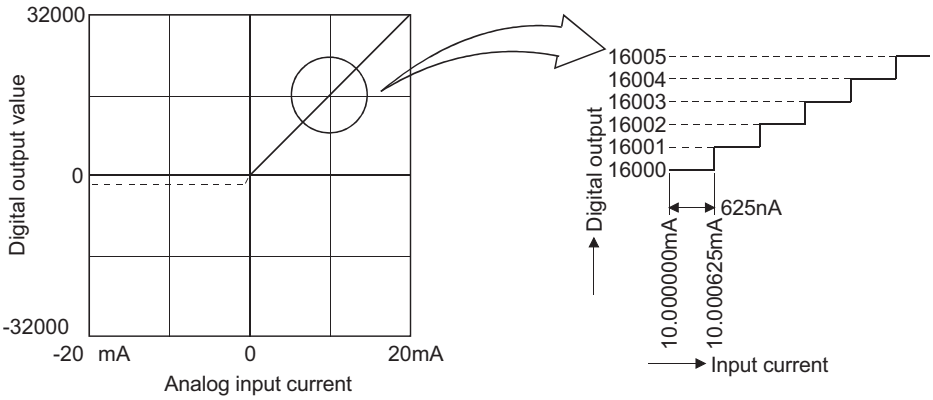
(When the input range setting is -10 to 10V.)



The analog-digital converter module converts analog inputs from outside into digital quantities to make arithmetic operation by the CPU possible. For voltage input, when -10V is input, a digital quantity of -32000 is output, and when 10V is input, 32000 is output. Therefore, an input of 312.5μV is equivalent to a digital quantity of 1, and a value smaller than 312.5μV, which cannot be converted, is discarded.

A/D conversion characteristics for current input

(When the input range setting is 0 to 20mA.)



For current input, when 0mA is input, 0 is output, and when 20mA is input, 32000 is output. 625nA is equivalent to a digital quantity of 1, and a value smaller than 625nA, which cannot be converted, is discarded.

Point

A voltage/current value equivalent to the digital value 1 in A/D conversion (maximum resolution) differs depending on the input range setting.

List of I/O signals and buffer memory area assignment

List of I/O signals

The following tables list the I/O signals of the analog-digital converter module.

Point

- The I/O numbers (X/Y) below indicate numbers when the start I/O number of the A/D converter module is set to 0.
- The use-prohibited signals below are used by the system, and cannot be used by the customer. If the customer uses any of them (turns from off to on), the function as the A/D converter module cannot be guaranteed.

■Input signal

Device No.	Signal name
X0	Module READY
X1 to X7	Use prohibited
X8	Warning output signal
X9	Operating condition setting completion flag
XA	Offset/gain setting mode status flag
XB	Channel change completed flag
XC	Input signal error detection signal
XD	Maximum value/minimum value reset completed flag
XE	A/D conversion completed flag
XF	Error flag

■Output signal

Device No.	Signal name
Y0 to Y8	Use prohibited
Y9	Operating condition setting request
YA	User range write request
YB	Channel change request
YC	Use prohibited
YD	Maximum value/minimum value reset request
YE	Use prohibited
YF	Error clear request

Buffer memory area assignment

Two types are available: R mode, in which the module operates according to the map of the buffer memory areas newly assigned by the MELSEC iQ-R series, and Q compatible mode, in which the module operates by converting the buffer memory map into an equivalent of the buffer memory map of the MELSEC-Q series.

Below are lists of buffer memory addresses when the R mode is used.

For details, refer to the following.

MELSEC iQ-R Analog-Digital Converter Module User's Manual (Application)

Point

Of all the buffer memory areas, do not write data to system areas and areas whose data type is monitor. If data is written to them, malfunction may occur.

■Un\G0~Un\G399

Address (decimal)	Address (hexadecimal)	Name	Default value	Data type	Auto refresh	Necessity of Y9*1
0	0H	Latest error code	0	Monitor	○	—
1	1H	Latest address of error history	0	Monitor	○	—
2	2H	Latest alarm code	0	Monitor	○	—
3	3H	Latest address of alarm history	0	Monitor	○	—
4 to 19	4H to 13H	Interrupt factor detection flag [n]*2	0	Monitor	○	—
20 to 35	14H to 23H	System area	—	—	—	—
36	24H	Warning output flag (process alarm upper limit)	0000H	Monitor	○	—
37	25H	Warning output flag (process alarm lower limit)	0000H	Monitor	○	—
38	26H	Warning output flag (rate alarm upper limit)	0000H	Monitor	○	—
39	27H	Warning output flag (rate alarm lower limit)	0000H	Monitor	○	—
40	28H	Input signal error detection flag	0000H	Monitor	○	—
41	29H	System area	0000H	—	—	—
42	2AH	A/D conversion completed flag	0000H	Monitor	○	—
43 to 89	2BH to 59H	System area	—	—	—	—
90	5AH	Level data 0	0	Control	○	—
91	5BH	Level data 1	0	Control	○	—
92	5CH	Level data 2	0	Control	○	—
93	5DH	Level data 3	0	Control	○	—
94	5EH	Level data 4	0	Control	○	—
95	5FH	Level data 5	0	Control	○	—
96	60H	Level data 6	0	Control	○	—
97	61H	Level data 7	0	Control	○	—
98	62H	Level data 8	0	Control	○	—
99	63H	Level data 9	0	Control	○	—
100 to 123	64H to 7BH	System area	—	—	—	—
124 to 139	7CH to 8BH	Interrupt factor mask [n]*2	0	Control	×	—
140 to 155	8CH to 9BH	System area	—	—	—	—
156 to 171	9CH to ABH	Interrupt factor reset request [n]*2	0	Control	×	—
172 to 199	ACH to C7H	System area	—	—	—	—
200 to 215	C8H to D7H	Interrupt factor occurrence setting [n]*2	0	Setting	×	○
216 to 231	D8H to E7H	System area	—	—	—	—
232 to 247	E8H to F7H	Condition target setting [n]*2	0	Setting	×	○
248 to 263	F8H to 107H	System area	—	—	—	—
264 to 279	108H to 117H	Condition target channel setting [n]*2	0	Setting	×	○
280 to 295	118H to 127H	System area	—	—	—	—
296, 297	128H, 129H	Mode switching setting	0	Setting	×	○
298 to 399	12AH to 18FH	System area	—	—	—	—

*1 Item enabled when 'Operating condition setting request' (Y9) turns on and off
*2 [n] in the table is an interrupt setting number. (n= 1 to 16)

■Un\G400~Un\G3599

Address Decimal (Hexadecimal)								Name	Default value	Data type	Auto refresh	Necessity of Y9[*1]
CH1	CH2	CH3	CH4	CH5	CH6	CH7	CH8					
400 (190H)	600 (258H)	800 (320H)	1000 (3E8H)	1200 (4B0H)	1400 (578H)	1600 (640H)	1800 (708H)	CH□ Digital output value	0	Monitor	○	—
401 (191H)	601 (259H)	801 (321H)	1001 (3E9H)	1201 (4B1H)	1401 (579H)	1601 (641H)	1801 (709H)	System area	—	—	—	—
402 (192H)	602 (25AH)	802 (322H)	1002 (3EAH)	1202 (4B2H)	1402 (57AH)	1602 (642H)	1802 (70AH)	CH□ Digital operation value	0	Monitor	○	—
403 (193H)	603 (25BH)	803 (323H)	1003 (3EBH)	1203 (4B3H)	1403 (57BH)	1603 (643H)	1803 (70BH)	System area	—	—	—	—
404 (194H)	604 (25CH)	804 (324H)	1004 (3ECH)	1204 (4B4H)	1404 (57CH)	1604 (644H)	1804 (70CH)	CH□ Maximum value	0	Monitor	○	—
405 (195H)	605 (25DH)	805 (325H)	1005 (3EDH)	1205 (4B5H)	1405 (57DH)	1605 (645H)	1805 (70DH)	System area	—	—	—	—
406 (196H)	606 (25EH)	806 (326H)	1006 (3EEH)	1206 (4B6H)	1406 (57EH)	1606 (646H)	1806 (70EH)	CH□ Minimum value	0	Monitor	○	—
407 (197H)	607 (25FH)	807 (327H)	1007 (3EFH)	1207 (4B7H)	1407 (57FH)	1607 (647H)	1807 (70FH)	System area	—	—	—	—
408 (198H)	608 (260H)	808 (328H)	1008 (3F0H)	1208 (4B8H)	1408 (580H)	1608 (648H)	1808 (710H)	CH□ Difference conversion status flag	0	Monitor	○	—
409 (199H)	609 (261H)	809 (329H)	1009 (3F1H)	1209 (4B9H)	1409 (581H)	1609 (649H)	1809 (711H)	CH□ Logging hold flag	0	Monitor	○	—
410 to 429 (19AH to 1ADH)	610 to 629 (262H to 275H)	810 to 829 (32AH to 33DH)	1010 to 1029 (3F2H to 405H)	1210 to 1229 (4BAH to 4CDH)	1410 to 1429 (582H to 595H)	1610 to 1629 (64AH to 65DH)	1810 to 1829 (712H to 725H)	System area	—	—	—	—
430 (1AEH)	630 (276H)	830 (33EH)	1030 (406H)	1230 (4CEH)	1430 (596H)	1630 (65EH)	1830 (726H)	CH□ Range setting monitor	0000H	Monitor	×	—
431 (1AFH)	631 (277H)	831 (33FH)	1031 (407H)	1231 (4CFH)	1431 (597H)	1631 (65FH)	1831 (727H)	System area	—	—	—	—
432 (1B0H)	632 (278H)	832 (340H)	1032 (408H)	1232 (4D0H)	1432 (598H)	1632 (660H)	1832 (728H)	CH□ Difference conversion reference value	0000H	Monitor	×	—
433 (1B1H)	633 (279H)	833 (341H)	1033 (409H)	1233 (4D1H)	1433 (599H)	1633 (661H)	1833 (729H)	System area	—	—	—	—
434 (1B2H)	634 (27AH)	834 (342H)	1034 (40AH)	1234 (4D2H)	1434 (59AH)	1634 (662H)	1834 (72AH)	CH□ Head pointer	0	Monitor	×	—
435 (1B3H)	635 (27BH)	835 (343H)	1035 (40BH)	1235 (4D3H)	1435 (59BH)	1635 (663H)	1835 (72BH)	CH□ Latest pointer	0	Monitor	×	—
436 (1B4H)	636 (27CH)	836 (344H)	1036 (40CH)	1236 (4D4H)	1436 (59CH)	1636 (664H)	1836 (72CH)	CH□ Number of logging data	0	Monitor	×	—
437 (1B5H)	637 (27DH)	837 (345H)	1037 (40DH)	1237 (4D5H)	1437 (59DH)	1637 (665H)	1837 (72DH)	CH□ Trigger pointer	0	Monitor	×	—
438 (1B6H)	638 (27EH)	838 (346H)	1038 (40EH)	1238 (4D6H)	1438 (59EH)	1638 (666H)	1838 (72EH)	CH□ Current logging read pointer	-1	Monitor	×	—
439 (1B7H)	639 (27FH)	839 (347H)	1039 (40FH)	1239 (4D7H)	1439 (59FH)	1639 (667H)	1839 (72FH)	CH□ Previous logging read pointer	-1	Monitor	×	—
440 (1B8H)	640 (280H)	840 (348H)	1040 (410H)	1240 (4D8H)	1440 (5A0H)	1640 (668H)	1840 (730H)	CH□ Logging read points monitor value	0	Monitor	×	—
441 (1B9H)	641 (281H)	841 (349H)	1041 (411H)	1241 (4D9H)	1441 (5A1H)	1641 (669H)	1841 (731H)	CH□ Logging cycle monitor value (s)	0	Monitor	×	—
442 (1BAH)	642 (282H)	842 (34AH)	1042 (412H)	1242 (4DAH)	1442 (5A2H)	1642 (66AH)	1842 (732H)	CH□ Logging cycle monitor value (ms)	0	Monitor	×	—
443 (1BBH)	643 (283H)	843 (34BH)	1043 (413H)	1243 (4DBH)	1443 (5A3H)	1643 (66BH)	1843 (733H)	CH□ Logging cycle monitor value (μs)	0	Monitor	×	—
444 (1BCH)	644 (284H)	844 (34CH)	1044 (414H)	1244 (4DCH)	1444 (5A4H)	1644 (66CH)	1844 (734H)	CH□ Trigger occurrence time (first/ last two digits of the year)	0	Monitor	×	—

Address Decimal (Hexadecimal)								Name	Default value	Data type	Auto refresh	Necessity of Y9[*1]
CH1	CH2	CH3	CH4	CH5	CH6	CH7	CH8					
445 (1BDH)	645 (285H)	845 (34DH)	1045 (415H)	1245 (4DDH)	1445 (5A5H)	1645 (66DH)	1845 (735H)	CH□ Trigger occurrence time (month/day)	0	Monitor	×	—
446 (1BEH)	646 (286H)	846 (34EH)	1046 (416H)	1246 (4DEH)	1446 (5A6H)	1646 (66EH)	1846 (736H)	CH□ Trigger occurrence time (hour/minute)	0	Monitor	×	—
447 (1BFH)	647 (287H)	847 (34FH)	1047 (417H)	1247 (4DFH)	1447 (5A7H)	1647 (66FH)	1847 (737H)	CH□ Trigger occurrence time (second/day of week)	0	Monitor	×	—
448 (1C0H)	648 (288H)	848 (350H)	1048 (418H)	1248 (4E0H)	1448 (5A8H)	1648 (670H)	1848 (738H)	CH□ Trigger occurrence time (millisecond)	0	Monitor	×	—
449 to 469 (1C1H to 1D5H)	649 to 669 (289H to 29DH)	849 to 869 (351H to 365H)	1049 to 1069 (419H to 42DH)	1249 to 1269 (4E1H to 4F5H)	1449 to 1469 (5A9H to 5BDH)	1649 to 1669 (671H to 685H)	1849 to 1869 (739H to 74DH)	System area	—	—	—	—
470 (1D6H)	670 (29EH)	870 (366H)	1070 (42EH)	1270 (4F6H)	1470 (5BEH)	1670 (686H)	1870 (74EH)	CH□ Difference conversion trigger	0	Control	○	—
471 (1D7H)	671 (29FH)	871 (367H)	1071 (42FH)	1271 (4F7H)	1471 (5BFH)	1671 (687H)	1871 (74FH)	CH□ Logging hold request	0	Control	○	—
472 (1D8H)	672 (2A0H)	872 (368H)	1072 (430H)	1272 (4F8H)	1472 (5C0H)	1672 (688H)	1872 (750H)	CH□ Conversion value shift amount	0	Control	○	—
473 to 499 (1D9H to 1F3H)	673 to 699 (2A1H to 2BBH)	873 to 899 (369H to 383H)	1073 to 1099 (431H to 44BH)	1273 to 1299 (4F9H to 513H)	1473 to 1499 (5C1H to 5DBH)	1673 to 1699 (689H to 6A3H)	1873 to 1899 (751H to 76BH)	System area	—	—	—	—
500 (1F4H)	700 (2BCH)	900 (384H)	1100 (44CH)	1300 (514H)	1500 (5DCH)	1700 (6A4H)	1900 (76CH)	CH□ A/D conversion enable/disable setting	0	Setting	×	○
501 (1F5H)	701 (2BDH)	901 (385H)	1101 (44DH)	1301 (515H)	1501 (5DDH)	1701 (6A5H)	1901 (76DH)	CH□ Average process specification	0	Setting	×	○
502 (1F6H)	702 (2BEH)	902 (386H)	1102 (44EH)	1302 (516H)	1502 (5DEH)	1702 (6A6H)	1902 (76EH)	CH□ Time (for averaging)/count (for averaging)/moving average/primary delay filter constant setting	0	Setting	×	○
503 (1F7H)	703 (2BFH)	903 (387H)	1103 (44FH)	1303 (517H)	1503 (5DFH)	1703 (6A7H)	1903 (76FH)	System area	—	—	—	—
504 (1F8H)	704 (2C0H)	904 (388H)	1104 (450H)	1304 (518H)	1504 (5E0H)	1704 (6A8H)	1904 (770H)	CH□ Scaling enable/disable setting	1	Setting	×	○
505 (1F9H)	705 (2C1H)	905 (389H)	1105 (451H)	1305 (519H)	1505 (5E1H)	1705 (6A9H)	1905 (771H)	System area	—	—	—	—
506 (1FAH)	706 (2C2H)	906 (38AH)	1106 (452H)	1306 (51AH)	1506 (5E2H)	1706 (6AAH)	1906 (772H)	CH□ Scaling upper limit value	0	Setting	×	○
507 (1FBH)	707 (2C3H)	907 (38BH)	1107 (453H)	1307 (51BH)	1507 (5E3H)	1707 (6ABH)	1907 (773H)	System area	—	—	—	—
508 (1FCH)	708 (2C4H)	908 (38CH)	1108 (454H)	1308 (51CH)	1508 (5E4H)	1708 (6ACH)	1908 (774H)	CH□ Scaling lower limit value	0	Setting	×	○
509 (1FDH)	709 (2C5H)	909 (38DH)	1109 (455H)	1309 (51DH)	1509 (5E5H)	1709 (6ADH)	1909 (775H)	System area	—	—	—	—
510 (1FEH)	710 (2C6H)	910 (38EH)	1110 (456H)	1310 (51EH)	1510 (5E6H)	1710 (6AEH)	1910 (776H)	CH□ Digital clipping enable/disable setting	1	Setting	×	○
511 (1FFH)	711 (2C7H)	911 (38FH)	1111 (457H)	1311 (51FH)	1511 (5E7H)	1711 (6AFH)	1911 (777H)	System area	—	—	—	—
512 (200H)	712 (2C8H)	912 (390H)	1112 (458H)	1312 (520H)	1512 (5E8H)	1712 (6B0H)	1912 (778H)	CH□ Alert output setting (process alarm)	1	Setting	×	○
513 (201H)	713 (2C9H)	913 (391H)	1113 (459H)	1313 (521H)	1513 (5E9H)	1713 (6B1H)	1913 (779H)	CH□ Alert output setting (rate alarm)	1	Setting	×	○
514 (202H)	714 (2CAH)	914 (392H)	1114 (45AH)	1314 (522H)	1514 (5EAH)	1714 (6B2H)	1914 (77AH)	CH□ Process alarm upper upper limit value	0	Setting	×	○

Address Decimal (Hexadecimal)								Name	Default value	Data type	Auto refresh	Necessity of Y9[*1]
CH1	CH2	CH3	CH4	CH5	CH6	CH7	CH8					
515 (203H)	715 (2CBH)	915 (393H)	1115 (45BH)	1315 (523H)	1515 (5EBH)	1715 (6B3H)	1915 (77BH)	System area	—	—	—	—
516 (204H)	716 (2CCH)	916 (394H)	1116 (45CH)	1316 (524H)	1516 (5ECH)	1716 (6B4H)	1916 (77CH)	CH☐ Process alarm upper lower limit value	0	Setting	×	○
517 (205H)	717 (2CDH)	917 (395H)	1117 (45DH)	1317 (525H)	1517 (5EDH)	1717 (6B5H)	1917 (77DH)	System area	—	—	—	—
518 (206H)	718 (2CEH)	918 (396H)	1118 (45EH)	1318 (526H)	1518 (5EEH)	1718 (6B6H)	1918 (77EH)	CH☐ Process alarm lower upper limit value	0	Setting	×	○
519 (207H)	719 (2CFH)	919 (397H)	1119 (45FH)	1319 (527H)	1519 (5EFH)	1719 (6B7H)	1919 (77FH)	System area	—	—	—	—
520 (208H)	720 (2D0H)	920 (398H)	1120 (460H)	1320 (528H)	1520 (5F0H)	1720 (6B8H)	1920 (780H)	CH☐ Process alarm lower lower limit value	0	Setting	×	○
521 (209H)	721 (2D1H)	921 (399H)	1121 (461H)	1321 (529H)	1521 (5F1H)	1721 (6B9H)	1921 (781H)	System area	—	—	—	—
522 (20AH)	722 (2D2H)	922 (39AH)	1122 (462H)	1322 (52AH)	1522 (5F2H)	1722 (6BAH)	1922 (782H)	CH☐ Rate alarm alert detection cycle setting	0	Setting	×	○
523 (20BH)	723 (2D3H)	923 (39BH)	1123 (463H)	1323 (52BH)	1523 (5F3H)	1723 (6BBH)	1923 (783H)	System area	—	—	—	—
524 (20CH)	724 (2D4H)	924 (39CH)	1124 (464H)	1324 (52CH)	1524 (5F4H)	1724 (6BCH)	1924 (784H)	CH☐ Rate alarm upper limit value	0	Setting	×	○
525 (20DH)	725 (2D5H)	925 (39DH)	1125 (465H)	1325 (52DH)	1525 (5F5H)	1725 (6BDH)	1925 (785H)	System area	—	—	—	—
526 (20EH)	726 (2D6H)	926 (39EH)	1126 (466H)	1326 (52EH)	1526 (5F6H)	1726 (6BEH)	1926 (786H)	CH☐ Rate alarm lower limit value	0	Setting	×	○
527 (20FH)	727 (2D7H)	927 (39FH)	1127 (467H)	1327 (52FH)	1527 (5F7H)	1727 (6BFH)	1927 (787H)	System area	—	—	—	—
528 (210H)	728 (2D8H)	928 (3A0H)	1128 (468H)	1328 (530H)	1528 (5F8H)	1728 (6C0H)	1928 (788H)	CH☐ Input signal error detection setting	0	Setting	×	○
529 (211H)	729 (2D9H)	929 (3A1H)	1129 (469H)	1329 (531H)	1529 (5F9H)	1729 (6C1H)	1929 (789H)	CH☐ Input signal error detection setting value	50	Setting	×	○
530 to 534 (212H to 216H)	730 to 734 (2DAH to 2DEH)	930 to 934 (3A2H to 3A6H)	1130 to 1134 (46AH to 46EH)	1330 to 1334 (532H to 536H)	1530 to 1534 (5FAH to 5FEH)	1730 to 1734 (6C2H to 6C6H)	1930 to 1934 (78AH to 78EH)	System area	—	—	—	—
535 (217H)	735 (2DFH)	935 (3A7H)	1135 (46FH)	1335 (537H)	1535 (5FFH)	1735 (6C7H)	1935 (78FH)	CH☐ Logging enable/ disable setting	1	Setting	×	○
536 (218H)	736 (2E0H)	936 (3A8H)	1136 (470H)	1336 (538H)	1536 (600H)	1736 (6C8H)	1936 (790H)	CH☐ Logging data setting	1	Setting	×	○
537 (219H)	737 (2E1H)	937 (3A9H)	1137 (471H)	1337 (539H)	1537 (601H)	1737 (6C9H)	1937 (791H)	CH☐ Logging cycle setting value	4	Setting	×	○
538 (21AH)	738 (2E2H)	938 (3AAH)	1138 (472H)	1338 (53AH)	1538 (602H)	1738 (6CAH)	1938 (792H)	CH☐ Logging cycle unit specification	1	Setting	×	○
539 (21BH)	739 (2E3H)	939 (3ABH)	1139 (473H)	1339 (53BH)	1539 (603H)	1739 (6CBH)	1939 (793H)	CH☐ Post-trigger logging points	5000	Setting	×	○
540 (21CH)	740 (2E4H)	940 (3ACH)	1140 (474H)	1340 (53CH)	1540 (604H)	1740 (6CCH)	1940 (794H)	CH☐ Level trigger condition setting	0	Setting	×	○
541 (21DH)	741 (2E5H)	941 (3ADH)	1141 (475H)	1341 (53DH)	1541 (605H)	1741 (6CDH)	1941 (795H)	CH☐ Trigger data	*2	Setting	×	○
542 (21EH)	742 (2E6H)	942 (3AEH)	1142 (476H)	1342 (53EH)	1542 (606H)	1742 (6CEH)	1942 (796H)	CH☐ Trigger setting value	0	Setting	×	○
543 (21FH)	743 (2E7H)	943 (3AFH)	1143 (477H)	1343 (53FH)	1543 (607H)	1743 (6CFH)	1943 (797H)	System area	—	—	—	—
544 (220H)	744 (2E8H)	944 (3B0H)	1144 (478H)	1344 (540H)	1544 (608H)	1744 (6D0H)	1944 (798H)	CH☐ Read interrupt enable/disable setting	1	Setting	×	○
545 (221H)	745 (2E9H)	945 (3B1H)	1145 (479H)	1345 (541H)	1545 (609H)	1745 (6D1H)	1945 (799H)	CH☐ Logging read points setting value	1000	Setting	×	○

Address Decimal (Hexadecimal)								Name	Default value	Data type	Auto refresh	Necessity of Y9*1
CH1	CH2	CH3	CH4	CH5	CH6	CH7	CH8					
546 to 597 (222H to 255H)	746 to 797 (2EAH to 31DH)	946 to 997 (3B2H to 3E5H)	1146 to 1197 (47AH to 4ADH)	1346 to 1397 (542H to 575H)	1546 to 1597 (60AH to 63DH)	1746 to 1797 (6D2 to 705H)	1946 to 1997 (79AH to 7CDH)	System area	—	—	—	—
598 (256H)	798 (31EH)	998 (3E6H)	1198 (4AEH)	1398 (576H)	1598 (63EH)	1798 (706H)	1998 (7CEH)	CH□ Range setting	0	Setting	×	○
599 (257H)	799 (31FH)	999 (3E7H)	1199 (4AFH)	1399 (577H)	1599 (63FH)	1799 (707H)	1999 (7CFH)	System area	—	—	—	—
2000 to 3599 (7D0H to E0FH)												

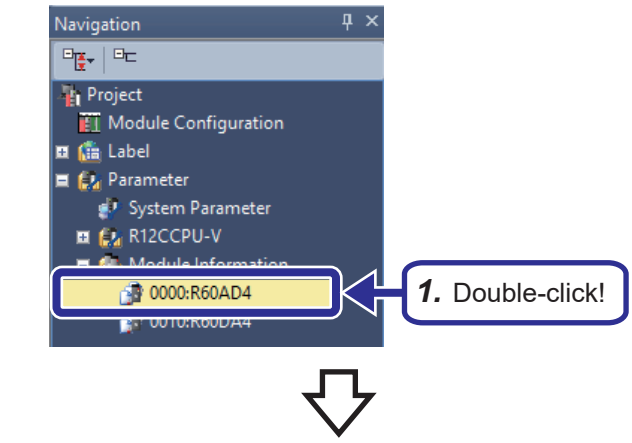
*1 Item enabled when 'Operating condition setting request' (Y9) turns on and off

*2 The default values are as follows.
CH1: 402, CH2: 602, CH3: 802, CH4: 1002, CH5: 1202, CH6: 1402, CH7: 1602, CH8: 1802

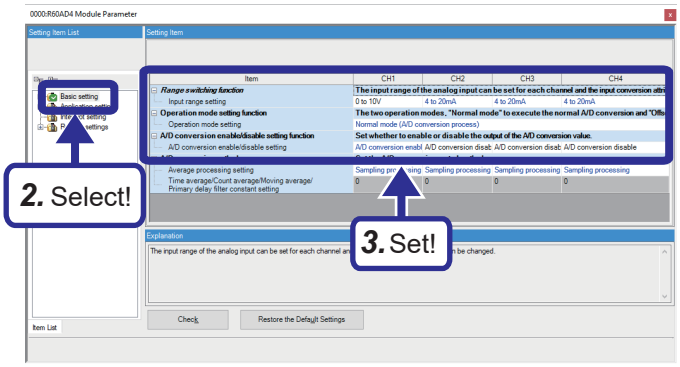
Setting intelligent function module data

This section describes how to set data of intelligent function modules.
By adding an intelligent function module to a project, data of an intelligent function module (such as parameters and switch settings) can be set.

Operating procedure

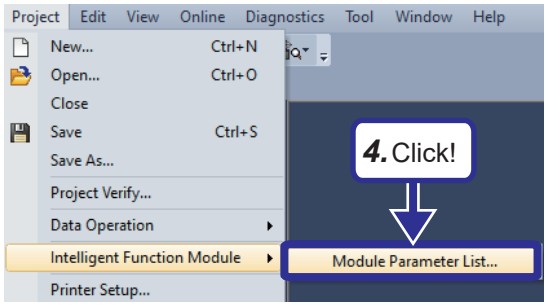


1. [Parameter] in the Navigation window ⇨ [Module Information] ⇨ double-click [0000: R60AD4].

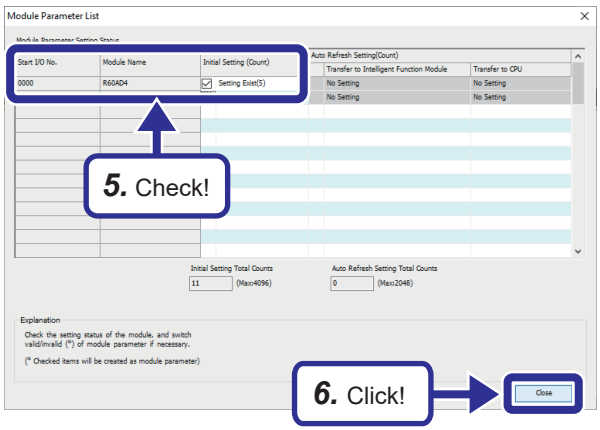


2. Select "Basic Settings" from the list of setting items.

3. Set "Basic Settings" as follows.
[Setting details]
Input range setting (CH1): 0 to 10V
A/D conversion enable/disable setting (CH2 to CH4): A/D conversion disable



4. [Project] on the menu ⇨ [Intelligent Function Module] ⇨ click [Module Parameter List].



5. In the Module Parameter List window, check that the checkbox for "Setting Exist" for the initial setting for R60AD4 is selected.

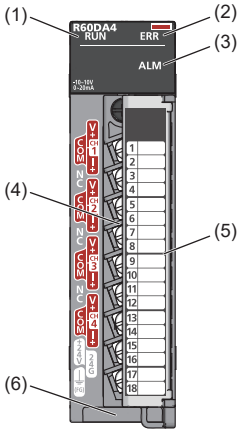
6. Click the [Close] button.

5.6 Digital-Analog Converter Module R60DA4

Part names

This section describes the part names of the R60DA4.

For details, refer to the user's manual.



No.	Name	Description
(1)	RUN LED	Indicates the operating status of the module. On: Normal operation Flashing (every 1s): In offset/gain setting mode Flashing (every 400ms): Selecting a module for online module change Off: 5V power supply turned off or watchdog timer error, ready to change modules during online module change
(2)	ERR LED	Indicate the module error occurrence status.*1 On: Error Off: Normal operation
(3)	ALM LED	Indicate the module alarm status.*1 On: Warning output generated Off: Normal operation
(4)	Terminal block	18-point screw terminal block. Connect output signal wires of a device such as an external device.
(5)	Terminal block cover	Cover for preventing electric shocks when the power is turned on
(6)	Production information marking	Shows the production information (16 digits) of the module.

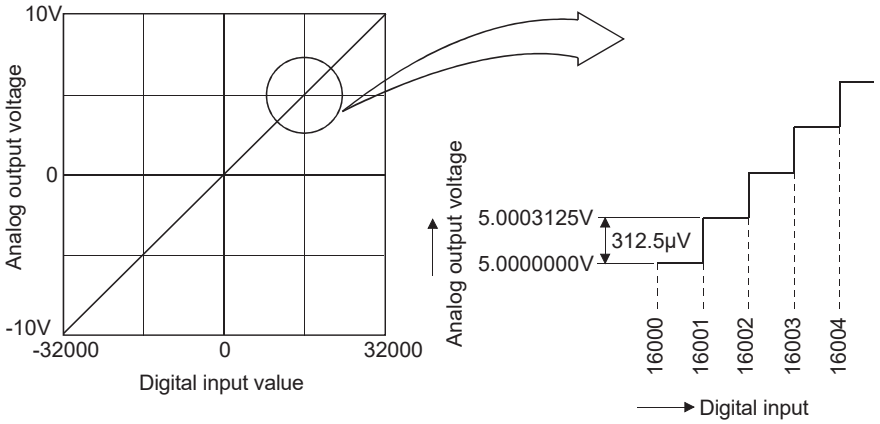
*1 For details, refer to the following.

MELSEC iQ-R Digital-Analog Converter Module User's Manual (Application)

D/A conversion characteristics

D/A conversion characteristics for voltage output

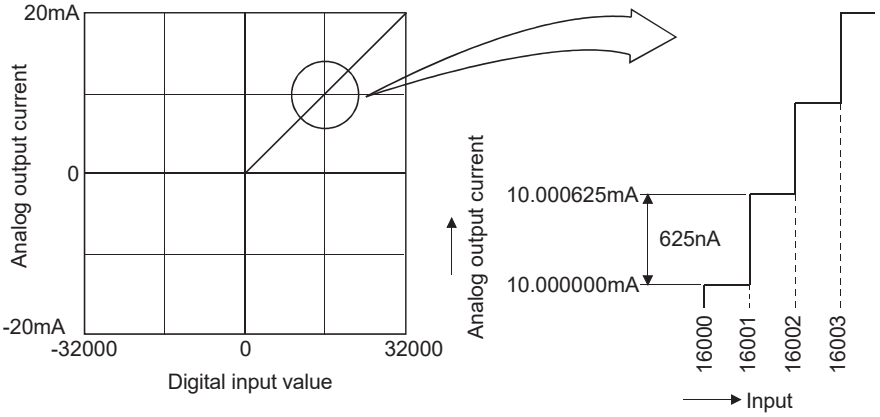
(When the output range setting is -10V to 10V.)



The digital-analog converter module converts digital quantities output from the CPU into analog values to output them externally. When a digital quantity of -32000 is input, -10V is output, and when a digital quantity of 32000 is input, 10V is output. Therefore, the digital input value 1 is equivalent to the analog amount 312.5μV, and a digital input smaller than 1 cannot be converted.

D/A conversion characteristics for current output

(When the output range setting is 0 to 20mA.)



For current output, 0 is converted to 0mA, and 32000 is converted to 20mA. The digital input value 1 is equivalent to the analog amount 625nA, and a digital input smaller than 1 cannot be converted.

Point

A voltage/current value equivalent to the digital value 1 in D/A conversion (maximum resolution) differs depending on the input range setting.

List of I/O signals and buffer memory area assignment

List of I/O signals

The following tables list the I/O signals for the digital-analog converter module.

Point

- The I/O signals (X/Y) below indicate signals when the start I/O number of the D/A converter module is set to 0.
- The use-prohibited signals below are used by the system, and cannot be used by the customer. If the customer uses any of them (turns from off to on), the function as the D/A converter module cannot be guaranteed.

■Input signal

Device No.	Signal name
X0	Module READY
X1 to X6	Use prohibited
X7	External power supply READY flag
X8	Use prohibited
X9	Operating condition setting completion flag
XA	Offset/gain setting mode status flag
XB	Channel change completed flag
XC	Set value change completed flag
XD	Disconnection detection signal
XE	Warning output signal
XF	Error flag

■Output signal

Device No.	Signal name
Y0	Use prohibited
Y1	CH1 Output enable/disable flag
Y2	CH2 Output enable/disable flag
Y3	CH3 Output enable/disable flag
Y4	CH4 Output enable/disable flag
Y5*1	CH5 Output enable/disable flag
Y6*1	CH6 Output enable/disable flag
Y7*1	CH7 Output enable/disable flag
Y8*1	CH8 Output enable/disable flag
Y9	Operating condition setting request
YA	User range write request
YB	Channel change request
YC	Value change request
YD	Use prohibited
YE	Warning output clear request
YF	Error clear request

*1 For the R60DA4, Y5 to Y8 are use prohibited.

Buffer memory area assignment

Two types are available: R mode, in which the module operates according to the map of the buffer memory areas newly assigned by the MELSEC iQ-R series, and Q compatible mode, in which the module operates by converting the buffer memory map into an equivalent of the buffer memory map of the MELSEC-Q series.

Below are lists of buffer memory addresses when the R mode is used.

For details, refer to the following.

MELSEC iQ-R Digital-Analog Converter Module User's Manual (Application)

Point

Of all the buffer memory areas, do not write data to system areas and areas whose data type is monitor. If data is written to them, malfunction may occur.

■Un\G0~Un\G399

Address (decimal)	Address (hexadecimal)	Name	Default value	Data type	Auto refresh	Necessity of Y9*1
0	0H	Latest error code	0	Monitor	○	—
1	1H	Latest address of error history	0	Monitor	○	—
2	2H	Latest alarm code	0	Monitor	○	—
3	3H	Latest address of alarm history	0	Monitor	○	—
4 to 19	4H to 13H	Interrupt factor detection flag [n]*2	0	Monitor	○	—
20 to 35	14H to 23H	System area	—	—	—	—
36	24H	Alert output upper limit flag	0000H	Monitor	○	—
37	25H	Alert output lower limit flag	0000H	Monitor	○	—
38	26H	Disconnection detection flag	0000H	Monitor	○	—
39 to 59	27H to 3BH	System area	—	—	—	—
60	3CH	Output mode	0000H	Monitor	×	○
61 to 123	3DH to 7BH	System area	—	—	—	—
124 to 139	7CH to 8BH	Interrupt factor mask [n]*2	0	Control	×	—
140 to 155	8CH to 9BH	System area	—	—	—	—
156 to 171	9CH to ABH	Interrupt factor reset request [n]*2	0	Control	×	—
172 to 187	ACH to BBH	System area	—	—	—	—
188	BCH	Step action wave output request	0	Control	×	—
189 to 199	BDH to C7H	System area	—	—	—	—
200 to 215	C8H to D7H	Interrupt factor generation setting [n]*2	0	Setting	×	○
216 to 231	D8H to E7H	System area	—	—	—	—
232 to 247	E8H to F7H	Condition target setting [n]*2	0	Setting	×	○
248 to 263	F8H to 107H	System area	—	—	—	—
264 to 279	108H to 117H	Condition target channel setting [n]*2	0	Setting	×	○
280 to 295	118H to 127H	System area	—	—	—	—
296, 297	128H, 129H	Mode switching setting	0	Setting	×	○
298 to 399	12AH to 18FH	System area	—	—	—	—

*1 Item enabled when 'Operating condition setting request' (Y9) turns on and off

*2 [n] in the table is an interrupt setting number. (n= 1 to 16)

■Un\G400 to Un\G3599

Address Decimal (Hexadecimal)								Name	Default value	Data type	Auto refresh	Necessity of Y9[*1]
CH1	CH2	CH3	CH4	CH5	CH6	CH7	CH8					
400 (190H)	600 (258H)	800 (320H)	1000 (3E8H)	1200 (4B0H)	1400 (578H)	1600 (640H)	1800 (708H)	CH□ Setting value check code	0	Monitor	○	—
401 (191H)	601 (259H)	801 (321H)	1001 (3E9H)	1201 (4B1H)	1401 (579H)	1601 (641H)	1801 (709H)	CH□ Wave output status monitor	0	Monitor	○	—
402 to 429 (192H to 1ADH)	602 to 629 (25AH to 275H)	802 to 829 (322H to 33DH)	1002 to 1029 (3EAH to 405H)	1202 to 1229 (4B2H to 4CDH)	1402 to 1429 (57AH to 595H)	1602 to 1629 (642H to 65DH)	1802 to 1829 (70AH to 725H)	System area	—	—	—	—
430 (1AEH)	630 (276H)	830 (33EH)	1030 (406H)	1230 (4CEH)	1430 (596H)	1630 (65EH)	1830 (726H)	CH□ Range setting monitor	0 —	Monitor	×	—
431 (1AFH)	631 (277H)	831 (33FH)	1031 (407H)	1231 (4CFH)	1431 (597H)	1631 (65FH)	1831 (727H)	CH□ HOLD/CLEAR function setting monitor	0	Monitor	×	—
432 (1B0H)	632 (278H)	832 (340H)	1032 (408H)	1232 (4D0H)	1432 (598H)	1632 (660H)	1832 (728H)	CH□ Wave output conversion cycle monitor (L)	0	Monitor	×	—
433 (1B1H)	633 (279H)	833 (341H)	1033 (409H)	1233 (4D1H)	1433 (599H)	1633 (661H)	1833 (729H)	CH□ Wave output conversion cycle monitor (H)		Monitor	×	—
434 (1B2H)	634 (27AH)	834 (342H)	1034 (40AH)	1234 (4D2H)	1434 (59AH)	1634 (662H)	1834 (72AH)	CH□ Wave pattern output count monitor	0	Monitor	×	—
435 (1B3H)	635 (27BH)	835 (343H)	1035 (40BH)	1235 (4D3H)	1435 (59BH)	1635 (663H)	1835 (72BH)	System area	—	—	—	—
436 (1B4H)	636 (27CH)	836 (344H)	1036 (40CH)	1236 (4D4H)	1436 (59CH)	1636 (664H)	1836 (72CH)	CH□ Wave output current address monitor (L)	0	Monitor	×	—
437 (1B5H)	637 (27DH)	837 (345H)	1037 (40DH)	1237 (4D5H)	1437 (59DH)	1637 (665H)	1837 (72DH)	CH□ Wave output current address monitor (H)		Monitor	×	—
438 (1B6H)	638 (27EH)	838 (346H)	1038 (40EH)	1238 (4D6H)	1438 (59EH)	1638 (666H)	1838 (72EH)	CH□ Wave output current digital value monitor	0	Monitor	×	—
439 (1B7H)	639 (27FH)	839 (347H)	1039 (40FH)	1239 (4D7H)	1439 (59FH)	1639 (667H)	1839 (72FH)	System area	—	—	—	—
440 (1B8H)	640 (280H)	840 (348H)	1040 (410H)	1240 (4D8H)	1440 (5A0H)	1640 (668H)	1840 (730H)	CH□ Wave output digital value out-of-range address monitor (L)	0	Monitor	×	—
441 (1B9H)	641 (281H)	841 (349H)	1041 (411H)	1241 (4D9H)	1441 (5A1H)	1641 (669H)	1841 (731H)	CH□ Wave output digital value out-of-range address monitor (H)		Monitor	×	—
442 (1BAH)	642 (282H)	842 (34AH)	1042 (412H)	1242 (4DAH)	1442 (5A2H)	1642 (66AH)	1842 (732H)	CH□ Wave output warning address monitor (L)	0	Monitor	×	—
443 (1BBH)	643 (283H)	843 (34BH)	1043 (413H)	1243 (4DBH)	1443 (5A3H)	1643 (66BH)	1843 (733H)	CH□ Wave output warning address monitor (H)		Monitor	×	—
444 to 459 (1BCH to 1CBH)	644 to 659 (284H to 293H)	844 to 859 (34CH to 35BH)	1044 to 1059 (414H to 423H)	1244 to 1259 (4DCH to 4EBH)	1444 to 1459 (5A4H to 5B3H)	1644 to 1659 (66CH to 67BH)	1844 to 1859 (734H to 743H)	System area	—	—	—	—
460 (1CCH)	660 (294H)	860 (35CH)	1060 (424H)	1260 (4ECH)	1460 (5B4H)	1660 (67CH)	1860 (744H)	CH□ Digital value	0	Control	○	—
461 (1CDH)	661 (295H)	861 (35DH)	1061 (425H)	1261 (4EDH)	1461 (5B5H)	1661 (67DH)	1861 (745H)	System area	—	—	—	—
462 (1CEH)	662 (296H)	862 (35EH)	1062 (426H)	1262 (4EEH)	1462 (5B6H)	1662 (67EH)	1862 (746H)	CH□ Wave output start/ stop request	0	Control	×	—
463 to 479 (1CFH to 1DFH)	663 to 679 (297H to 2A7H)	863 to 879 (35FH to 36FH)	1063 to 1079 (427H to 437H)	1263 to 1279 (4EFH to 4FFH)	1463 to 1479 (5B7H to 5C7H)	1663 to 1679 (67FH to 68FH)	1863 to 1879 (747H to 757H)	System area	—	—	—	—

Address Decimal (Hexadecimal)								Name	Default value	Data type	Auto refresh	Necessity of Y9*1
CH1	CH2	CH3	CH4	CH5	CH6	CH7	CH8					
480 (1E0H)	680 (2A8H)	880 (370H)	1080 (438H)	1280 (500H)	1480 (5C8H)	1680 (690H)	1880 (758H)	CH□ Input value shift amount	0	Control	×	—
481 (1E1H)	681 (2A9H)	881 (371H)	1081 (439H)	1281 (501H)	1481 (5C9H)	1681 (691H)	1881 (759H)	System area	—	—	—	—
482 (1E2H)	682 (2AAH)	882 (372H)	1082 (43AH)	1282 (502H)	1482 (5CAH)	1682 (692H)	1882 (75AH)	CH□ Wave output step action movement amount	0	Control	×	—
483 to 499 (1E3H to 1F3H)	683 to 699 (2ABH to 2BBH)	883 to 899 (373H to 383H)	1083 to 1099 (43BH to 44BH)	1283 to 1299 (503H to 513H)	1483 to 1499 (5CBH to 5DBH)	1683 to 1699 (693H to 6A3H)	1883 to 1899 (75BH to 76BH)	System area	—	—	—	—
500 (1F4H)	700 (2BCH)	900 (384H)	1100 (44CH)	1300 (514H)	1500 (5DCH)	1700 (6A4H)	1900 (76CH)	CH□ D/A conversion enable/disable setting	1	Setting	×	○
501 (1F5H)	701 (2BDH)	901 (385H)	1101 (44DH)	1301 (515H)	1501 (5DDH)	1701 (6A5H)	1901 (76DH)	System area	—	—	—	—
502 (1F6H)	702 (2BEH)	902 (386H)	1102 (44EH)	1302 (516H)	1502 (5DEH)	1702 (6A6H)	1902 (76EH)	CH□ Scaling enable/ disable setting	1	Setting	×	○
503 (1F7H)	703 (2BFH)	903 (387H)	1103 (44FH)	1303 (517H)	1503 (5DFH)	1703 (6A7H)	1903 (76FH)	System area	—	—	—	—
504 (1F8H)	704 (2C0H)	904 (388H)	1104 (450H)	1304 (518H)	1504 (5E0H)	1704 (6A8H)	1904 (770H)	CH□ Scaling upper limit value	0	Setting	×	○
505 (1F9H)	705 (2C1H)	905 (389H)	1105 (451H)	1305 (519H)	1505 (5E1H)	1705 (6A9H)	1905 (771H)	System area	—	—	—	—
506 (1FAH)	706 (2C2H)	906 (38AH)	1106 (452H)	1306 (51AH)	1506 (5E2H)	1706 (6AAH)	1906 (772H)	CH□ Scaling lower limit value	0	Setting	×	○
507 (1FBH)	707 (2C3H)	907 (38BH)	1107 (453H)	1307 (51BH)	1507 (5E3H)	1707 (6ABH)	1907 (773H)	System area	—	—	—	—
508 (1FCH)	708 (2C4H)	908 (38CH)	1108 (454H)	1308 (51CH)	1508 (5E4H)	1708 (6ACH)	1908 (774H)	CH□ Alert output setting	1	Setting	×	○
509 (1FDH)	709 (2C5H)	909 (38DH)	1109 (455H)	1309 (51DH)	1509 (5E5H)	1709 (6ADH)	1909 (775H)	CH□ Rate control enable/ disable setting	1	Setting	×	○
510 (1FEH)	710 (2C6H)	910 (38EH)	1110 (456H)	1310 (51EH)	1510 (5E6H)	1710 (6AEH)	1910 (776H)	CH□ Alert output upper limit value	0	Setting	×	○
511 (1FFH)	711 (2C7H)	911 (38FH)	1111 (457H)	1311 (51FH)	1511 (5E7H)	1711 (6AFH)	1911 (777H)	System area	—	—	—	—
512 (200H)	712 (2C8H)	912 (390H)	1112 (458H)	1312 (520H)	1512 (5E8H)	1712 (6B0H)	1912 (778H)	CH□ Alert output lower limit value	0	Setting	×	○
513 (201H)	713 (2C9H)	913 (391H)	1113 (459H)	1313 (521H)	1513 (5E9H)	1713 (6B1H)	1913 (779H)	System area	—	—	—	—
514 (202H)	714 (2CAH)	914 (392H)	1114 (45AH)	1314 (522H)	1514 (5EAH)	1714 (6B2H)	1914 (77AH)	CH□ Increase digital limit value	64000	Setting	×	○
515 (203H)	715 (2CBH)	915 (393H)	1115 (45BH)	1315 (523H)	1515 (5EBH)	1715 (6B3H)	1915 (77BH)	System area	—	—	—	—
516 (204H)	716 (2CCH)	916 (394H)	1116 (45CH)	1316 (524H)	1516 (5ECH)	1716 (6B4H)	1916 (77CH)	CH□ Decrease digital limit value	64000	Setting	×	○
517 to 523 (205H to 20BH)	717 to 723 (2CDH to 2D3H)	917 to 923 (395H to 39BH)	1117 to 1123 (45DH to 463H)	1317 to 1323 (525H to 52BH)	1517 to 1523 (5EDH to 5F3H)	1717 to 1723 (6B5H to 6BBH)	1917 to 1923 (77DH to 783H)	System area	—	—	—	—
524 (20CH)	724 (2D4H)	924 (39CH)	1124 (464H)	1324 (52CH)	1524 (5F4H)	1724 (6BCH)	1924 (784H)	CH□ Output setting during wave output stop	1	Setting	×	—
525 (20DH)	725 (2D5H)	925 (39DH)	1125 (465H)	1325 (52DH)	1525 (5F5H)	1725 (6BDH)	1925 (785H)	CH□ Output value during wave output stop	0	Setting	×	—
526 (20EH)	726 (2D6H)	926 (39EH)	1126 (466H)	1326 (52EH)	1526 (5F6H)	1726 (6BEH)	1926 (786H)	CH□ Wave pattern start address setting (L)	10000	Setting	×	—
527 (20FH)	727 (2D7H)	927 (39FH)	1127 (467H)	1327 (52FH)	1527 (5F7H)	1727 (6BFH)	1927 (787H)	CH□ Wave pattern start address setting (H)		Setting	×	—

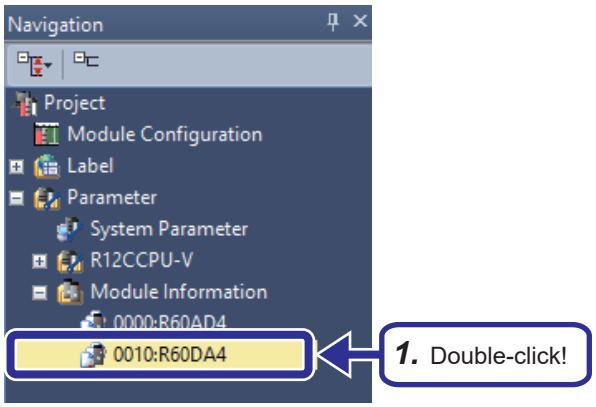
Address Decimal (Hexadecimal)								Name	Default value	Data type	Auto refresh	Necessity of Y9*1
CH1	CH2	CH3	CH4	CH5	CH6	CH7	CH8					
528 (210H)	728 (2D8H)	928 (3A0H)	1128 (468H)	1328 (530H)	1528 (5F8H)	1728 (6C0H)	1928 (788H)	CH□ Wave pattern data points setting (L)	0	Setting	×	—
529 (211H)	729 (2D9H)	929 (3A1H)	1129 (469H)	1329 (531H)	1529 (5F9H)	1729 (6C1H)	1929 (789H)	CH□ Wave pattern data points setting (H)		Setting	×	—
530 (212H)	730 (2DAH)	930 (3A2H)	1130 (46AH)	1330 (532H)	1530 (5FAH)	1730 (6C2H)	1930 (78AH)	CH□ Wave output repetition setting	1	Setting	×	—
531 (213H)	731 (2DBH)	931 (3A3H)	1131 (46BH)	1331 (533H)	1531 (5FBH)	1731 (6C3H)	1931 (78BH)	CH□ Constant for wave output conversion cycle	1	Setting	×	—
532 to 597 (214H to 255H)	732 to 797 (2DCH to 31DH)	932 to 997 (3A4H to 3E5H)	1132 to 1197 (46CH to 4ADH)	1332 to 1397 (534H to 575H)	1532 to 1597 (5FCH to 63DH)	1732 to 1797 (6C4H to 705H)	1932 to 1997 (78CH to 7CDH)	System area	—	—	—	—
598 (256H)	798 (31EH)	998 (3E6H)	1198 (4AEH)	1398 (576H)	1598 (63EH)	1798 (706H)	1998 (7CEH)	CH□ Range setting	0	Setting	×	○
599 (257H)	799 (31FH)	999 (3E7H)	1199 (4AFH)	1399 (577H)	1599 (63FH)	1799 (707H)	1999 (7CFH)	System area	—	—	—	—
2000 to 3599 (7D0H to E0FH)								System area	—	—	—	—

*1 Item enabled when 'Operating condition setting request' (Y9) turns on and off

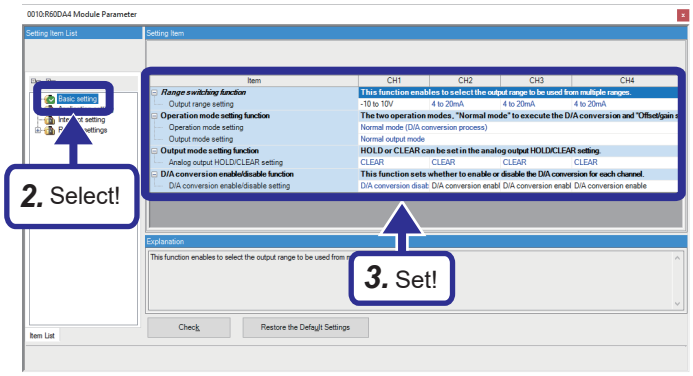
Setting intelligent function module data

This section describes how to set data of intelligent function modules.
By adding an intelligent function module to a project, data of an intelligent function module (such as parameters and switch settings) can be set.

Operating procedure

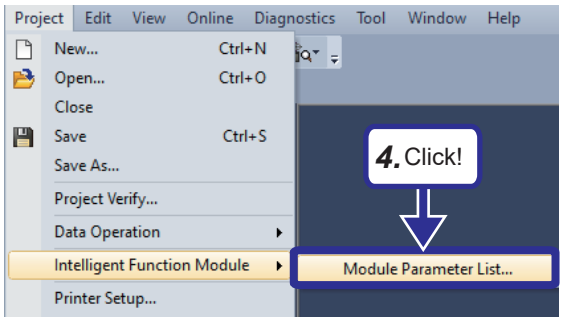


1. [Parameter] in the Navigation window ⇨ [Module Information] ⇨ double-click [0010: R60DA4].

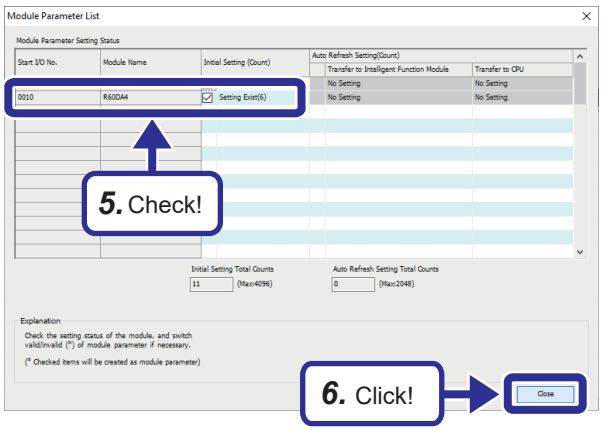


2. Select "Basic Settings" from the list of setting items.

3. Set "Basic Settings" as follows.
[Setting details]
Output range setting (CH1): -10 to 10V
D/A conversion enable/disable setting (CH1): D/A conversion enabled



4. [Project] on the menu ⇨ [Intelligent Function Module] ⇨ click [Module Parameter List].



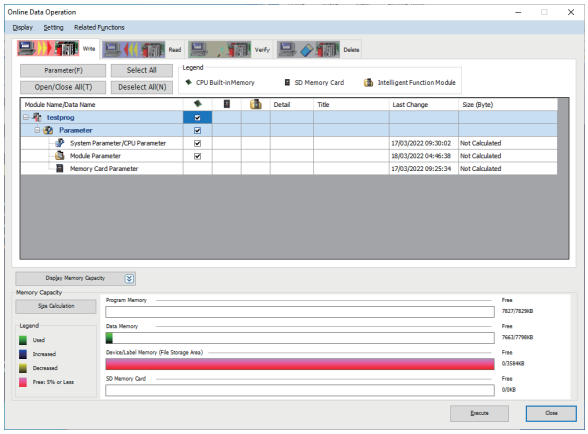
5. In the Module Parameter List window, check that the checkbox for "Setting Exist" for the initial setting for R60DA4 is selected.
6. Click the [Close] button.

5.7 Exercise 2 A/D conversion, D/A conversion

In this training, like in exercise 1, specify "C:\CCPU_CWW_Prj\enshu" as a workspace.

Writing parameters

Operating procedure



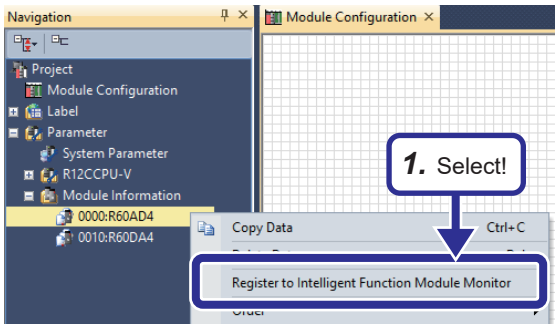
1. Refer to ☞ Page 46 Writing parameters to the C Controller module and write the parameters set in "Setting intelligent function module data" to the C Controller module.

Exercise 2.1 Checking the operation of the analog-digital converter module and digital-analog converter module

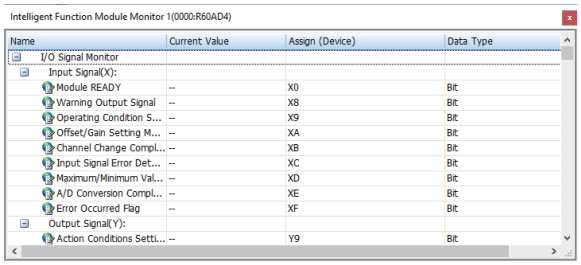
Using the intelligent function module monitor, check the operation of the analog-digital converter module R60AD4 and digital-analog converter module R60DA4.

Checking the operation of the R60AD

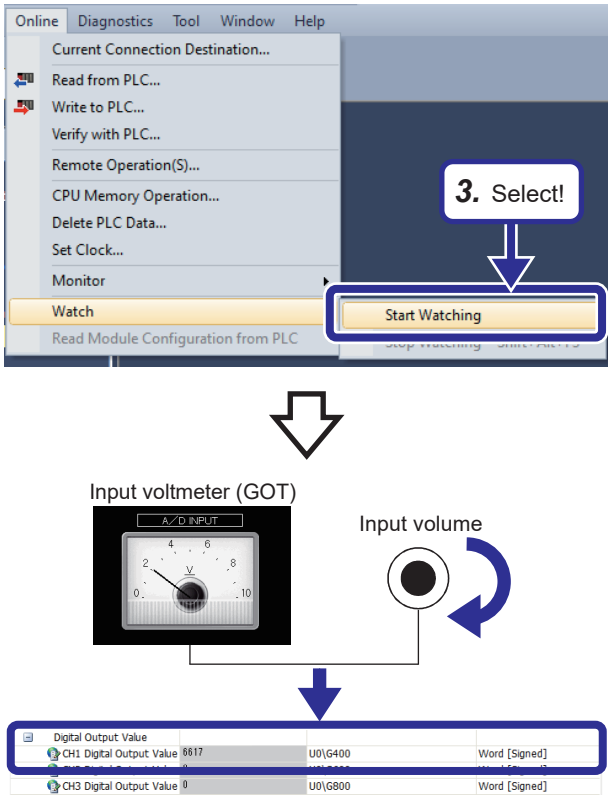
Operating procedure



1. Select R60AD4 in the project view, right-click the mouse, and select "Register to Intelligent Function Module Monitor".



2. The intelligent function module monitor window appears as a docking window, and 0000:R60AD is registered.

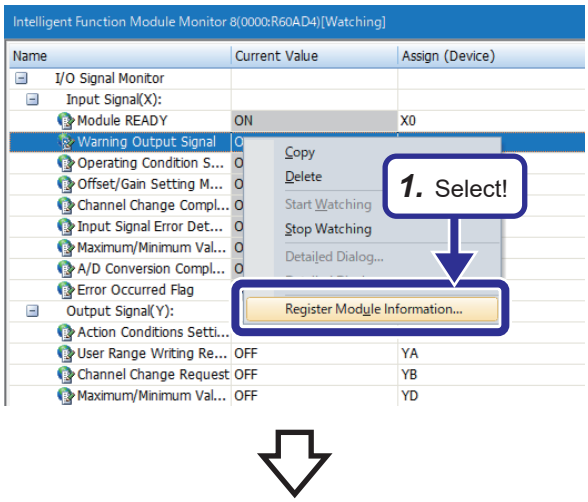


3. Select [Online] on the menu ⇨ [Watch] ⇨ [Start Watching] and start the intelligent function module monitor.

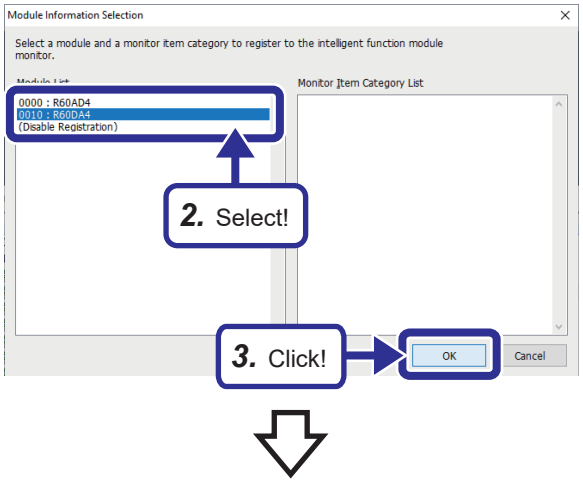
4. Check the current value by selecting the item "Buffer Memory Monitor" ⇨ "Digital output value" ⇨ "CH1 Digital output value". The R60AD4 receives an analog input value from the A/D INPUT of the demonstration machine at CH1, converts it through A/D conversion, and stores a digital output value to the address U0\G400 of its own buffer memory. This item indicates the current value of the above digital output value. Operate the input volume tab for the A/D INPUT of the demonstration machine to check that the digital output value changes according to the A/D conversion characteristics of the R60AD.

Checking the operation of the R60DA

Operating procedure



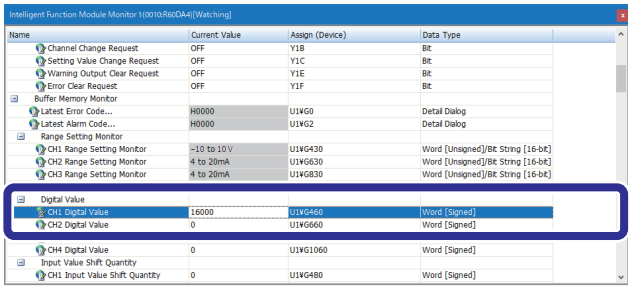
1. Right-click the intelligent function module monitor and select "Register Module Information".



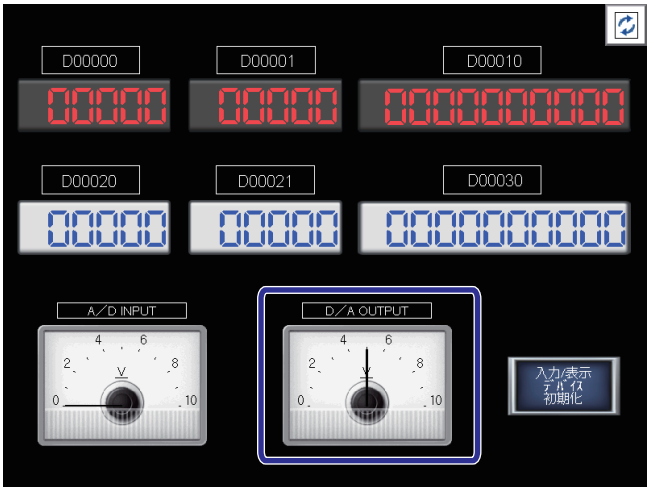
2. When the module information selection dialog box appears, select "0010: R60DA4" in the module list.

3. Click the [OK] button.

4. The module registered for the intelligent function module monitor is changed from R60AD4 to R60DA4.



5. Double-click the input value by selecting the item "Buffer Memory Monitor" ⇨ "Digital value" ⇨ "CH1 Digital output value" and enter 16000.



6. In accordance with the D/A conversion characteristics, R60DA4 converts the CH1 digital value stored in the address U1\G460 of the buffer memory into an analog value, and outputs it to outside as an analog output. Check with the D/A OUTPUT meter of the demonstration machine that in accordance with the D/A conversion characteristics, the digital value 16000 has been converted as a voltage of approx. 5V and output.

Exercise 2.2 Loading an A/D conversion output value into the HMI

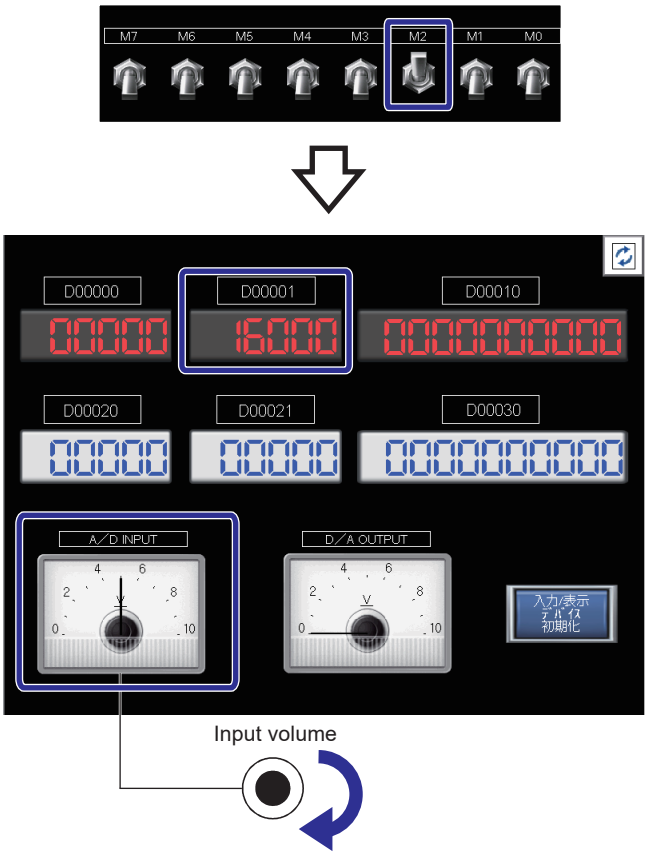
The A/D INPUT of the demonstration machine is converted through A/D conversion using the R60AD4, and its digital output value is loaded and displayed in the display device D1 of the HMI.

Operating procedure

1. Start up CW Workbench and newly create a project named "enshu2". (☞ Page 58 Creating a project)
   A workspace does not need to be set again because the workspace is the same as the one set in exercise 1.
2. Copy the program enshu2_2.c to the project folder C:\CCPU_CWW_Prj\enshu\enshu2 to add it to the project. (☞ Page 66 Procedure for adding a program)
3. Rebuild enshu2_2.c in debug mode. (☞ Page 67 Generating a module for execution)
4. Connect the C Controller module and CW Workbench. (☞ Page 72 Connecting and disconnecting)
5. Debug the created program to check if it operates correctly. (☞ Page 73 Debugging the user program)
6. Rebuild the program enshu2_2.c that underwent debugging by canceling the debug mode, and store the created user program on the C Controller module. (☞ Page 80 Registering a module for execution)
7. Create a script and store it in the C Controller module. (☞ Page 80 Registering a module for execution)
   [Script details]
   Load a program: ld (1, 0, "/0/enshu2.out")
   Generate a task: sp (enshu2_2)
8. Reset the C Controller module and set the switch on the front to the RUN position.
9. Operate the switch M2 of the demonstration machine. A value obtained by converting an input voltage from the demonstration machine into a digital value is displayed in the display device D1 of the HMI. Turn the input volume tab for the A/D INPUT to check that the value on D1 changes when the input voltage is changed. (☞ Page 123 Checking the operation)
10. After the operation is checked completely, disconnect CW Workbench from the C Controller module (☞ Page 72 Connecting and disconnecting) once, and delete the user program and script stored in the C Controller module.

Checking the operation

Operating procedure



1. Reset the C Controller module and set the switch on the front to the RUN position.
2. Operate the switch M2 of the demonstration machine.
3. Turn the input volume tab for A/D INPUT, and the meter value for the A/D INPUT changes and a value obtained by converting an input voltage into a digital value is displayed in the display device D1.

Source code

The following shows the source code of the program enshu2_2.

```
/*****************************************************************************************
* Exercise 2.2
*
* The R60AD4 converts analog input from the A/D INPUT of the demonstration machine
* into a digital value.
* This is a sample program in which the converted digital value is read out
* from the R60AD4, and displayed in the GOT display device D1 when M2 is turned on.
*****************************************************************************************/

#include <vxworks.h>
#include <tasklib.h>
#include <stdio.h>
#include <CCPUFunc.h>

#define ACCESS_FLG 1 /* input/output access flag */
#define ENABLE_SWITCH 0x00 /* analog conversion acquisition Enable */
#define AD_ADDR 0x00 /* start input number of the R60ADN */
#define AD_CONV_OUT 400 /* R60ADN buffer memory address (Ch1 digital output value) */
#define DEV_ADDR 1 /* start device number for the GOT display device */
#define WORD 1 /* data size */
#define DUMMY 0 /* CCPU function dummy */
#define Dev_CCPU_D 13 /* device type */
#define Dev_CCPU_M 4 /* device type */

/*****************************************************************************************
* Funcion declaration
*****************************************************************************************/
void enshu2_2();


/*****************************************************************************************
* Function name : enshu2_2()
* Version: 1.01
* The digital value, which is converted by the R60AD4 from the analog input from
* the A/D INPUT of the demonstration machine,
* is displayed in the GOT display device D1 only when M2 is turned on.
*****************************************************************************************/
void enshu2_2()
{
    short sRet; /* CCPU function return value */
    unsigned short usData; /* CCPU function M input value */
    unsigned short usDataBuf; /* CCPU function X input value / Y output value */
    unsigned short usDataBuf4AD; /* for storing the A/D conversion value */

    while( 1 )
    {
        /* Check whether M2 is on or off */
        sRet = CCPU_ReadDevice(Dev_CCPU_M, ENABLE_SWITCH, WORD, &usData, WORD);
        if(sRet != 0)
            goto finalization;

        if(usData == 4)
        {
           /* Read the R60ADN input information */
           sRet = CCPU_X_In_WordEx(ACCESS_FLG, AD_ADDR, WORD, &usDataBuf, WORD);
           if(sRet != 0)
              goto finalization;

           /* When X00 and X0E are on */
           if( ( ( usDataBuf & 0x0001 ) != 0 ) && ( ( usDataBuf & 0x4000 ) != 0 ) )
           {
                /* Read the buffer memory of the intelligent function module */
                sRet = CCPU_FromBuf(( AD_ADDR / 0x10 ), AD_CONV_OUT, WORD, &usDataBuf4AD, WORD );
```

```
                if( sRet != 0 )
                    goto finalization;

                sRet = CCPU_WriteDevice( Dev_CCPU_D, DEV_ADDR, WORD, &usDataBuf4AD, DUMMY );
                if( sRet != 0 )
                    goto finalization;
            }
        }
        else
        {
            /* Clear the GOT display device to 0 */
            usDataBuf4AD = 0x0000;
            sRet = CCPU_WriteDevice( Dev_CCPU_D, DEV_ADDR, WORD, &usDataBuf4AD, DUMMY );
            if( sRet != 0 )
                goto finalization;
            }

            /* 1-tick wait */
            taskDelay( 1 );
    }

finalization:
    /* Clear the GOT display device to 0 */
    usDataBuf4AD = 0x0000;
    sRet = CCPU_WriteDevice( Dev_CCPU_D, DEV_ADDR, WORD, &usDataBuf4AD, DUMMY );

    return;
}
```

Exercise 2.3 Outputting a value in the HMI through D/A conversion

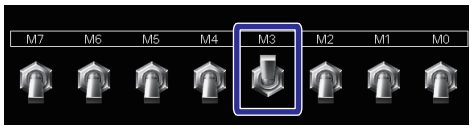
The R60DA4 converts a value of the input device D21 of the demonstration machine as a digital value through D/A conversion, and outputs a value as a D/A OUTPUT voltage output when M3 is turned on.

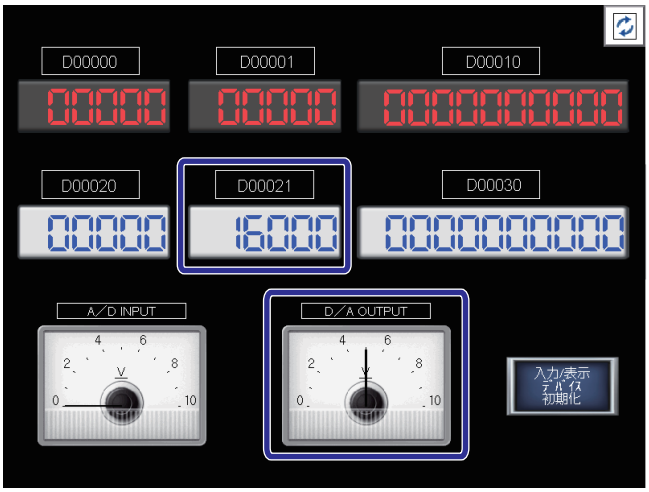
Operating procedure

1. Copy the program enshu2_3.c to the project folder C:\CCPU_CWW_Prj\enshu\enshu2 to add it to the project. (☞ Page 66 Procedure for adding a program)
2. Rebuild enshu2_3.c in debug mode. (☞ Page 67 Generating a module for execution)
3. Connect the C Controller module and CW Workbench. (☞ Page 72 Connecting and disconnecting)
4. Debug the created program to check if it operates correctly. (☞ Page 73 Debugging the user program)
5. Rebuild the program enshu2_3.c that underwent debugging by canceling the debug mode, and store the created user program on the C Controller module. (☞ Page 80 Registering a module for execution)
6. Create a script and store it in the C Controller module. (☞ Page 80 Registering a module for execution)
   [Script details]
   Load a program: ld (1, 0, "/0/enshu2.out")
   Generate a task: sp (enshu2_3)
7. Reset the C Controller module and set the switch on the front to the RUN position.
8. Operate the switch M3 of the demonstration machine. Check that the value set by the input device D21 of the HMI is converted through D/A conversion and output to the D/A OUTPUT. When the value of the input device D21 is 32001 or more, D/A conversion is not performed, and the lamp Y17F turns on. (☞ Page 126 Checking the operation)
9. After the operation is checked completely, disconnect CW Workbench from the C Controller module (☞ Page 72 Connecting and disconnecting) once, and delete the user program and script stored in the C Controller module.

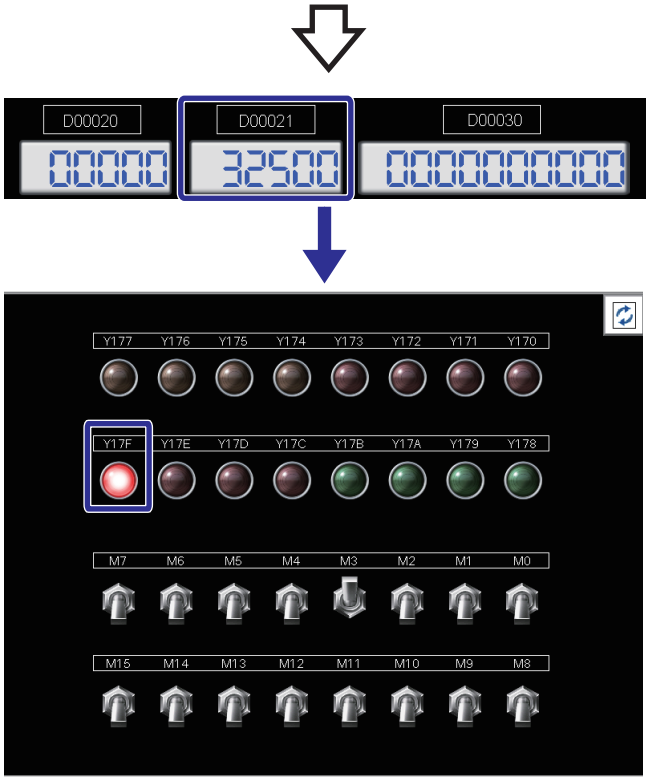
Checking the operation

Operating procedure





1. Reset the C Controller module and set the switch on the front to the RUN position.
2. Operate the switch M3 of the demonstration machine.
3. The value set by the input device D21 of the HMI is output to the D/A OUTPUT.



4. When the value of the input device D21 is 32001 or more, D/A conversion is not performed, and the lamp Y17F turns on.

Source code

The following shows the source code of the program enshu2_3.

```
/*****************************************************************************************
* Exercise 2.3
*
* This program aquires a value of the input device D21 as a digital input value and
* converts it into an analog value when M3 is turned on, then outputs the converted
* value to the D/A OUTPUT of the demonstration machine.
* When the value of the input device is 32001 or more, D/A conversion is not performed,
* and Y17F turns on.
*****************************************************************************************/

#include <vxworks.h>
#include <tasklib.h>
#include <stdio.h>
#include <CCPUFunc.h>

#define ACCESS_FLG 1
/* input/output access flag*/
#define ENABLE_SWITCH 0x00
/* analog conversion acquisition Enable */
#define DIGITAL_OVER 0x17F
/* out of setting range indication lamp for D/A conversion digital value*/
#define DA_OUT_ENABLE 0x11
/* R60DAN CH1 analog output enable*/
#define DA_UNIT_READY 0x10
/* R60DAN unit READY*/
#define DEV_ADDR 21
/* start I/O number of the input device*/
#define DA_UNIT_ADDR 0x10
/* R60DAN start I/O number*/
#define DA_INPUT_BUF 460
/* R60DAN CH1 digital value*/
#define DWORD 2
/* data size*/
#define WORD 1
/* data size*/
#define DUMMY 0
/* CCPU function dummy*/
#define BIT_ON 1
#define BIT_OFF 0
#define Dev_CCPU_D 13
/* device type */
#define Dev_CCPU_M 4
/* device type */

/*****************************************************************************************
* Functon declaration
*****************************************************************************************/
void enshu2_3();

/*****************************************************************************************
* Function name : enshu2_3()
*
* When M3 is turned on, the R60DA4 converts the setting value of the input device D21
* as the digital input value into an analog value.
* The converted value are output to the D/A OUTPUT of the demonstration machine.
* When the value of the input device is 32001 or more, Y17F turns on.

* *****************************************************************************************/
void enshu2_3()
{
    short sRet; /* CCPU function return value*/
    unsigned short usData; /* read of CCPU function X input value in units of bits*/
    unsigned short usData2; /* read of CCPU function M input value*/
    unsigned short usDataBuf4DEV; /* the input device value for D/A conversion*/
    long plStatusBuf; /* storage location of the operating status*/

    while(1)
    {
        /* Operating status acquisition of the C Controller module*/
            sRet = CCPU_GetCpuStatus(&plStatusBuf, DWORD);
            if(sRet != 0)
                goto finalization;

        if(plStatusBuf == 0)
        {
```

```
/* Acquisition of the Q62DAN unit READY signal*/
sRet = CCPU_X_In_BitEx(ACCESS_FLG, DA_UNIT_READY, &usData);
if(sRet != 0)
    goto finalization;

if(usData == 1)
{
    /* Check whether M3 is on or off*/
    sRet = CCPU_ReadDevice(Dev_CCPU_M, ENABLE_SWITCH, WORD, &usData2, WORD);
    if(sRet != 0)
        goto finalization;

    if(usData2 == 8)
    {
        /* Acquisition of the input device value*/
        sRet = CCPU_ReadDevice(Dev_CCPU_D, DEV_ADDR, WORD, &usDataBuf4DEV, WORD);
        if(sRet != 0)
            goto finalization;

         if(usDataBuf4DEV <= 32000)
        {
            sRet = CCPU_Y_Out_BitEx(ACCESS_FLG, DIGITAL_OVER, BIT_OFF);
            if(sRet != 0)
                goto finalization;

            sRet = CCPU_ToBuf((DA_UNIT_ADDR / 0x10), DA_INPUT_BUF, WORD,
                   &usDataBuf4DEV, DUMMY);
            if(sRet != 0)
                goto finalization;

            /* Turn on the D/A conversion output enable/disable signal */
            sRet = CCPU_Y_Out_BitEx(ACCESS_FLG, DA_OUT_ENABLE, BIT_ON);
            if(sRet != 0)
                goto finalization;
        }
        else
        {
            /* Turn off the D/A conversion output enable/disable signal */
            sRet = CCPU_Y_Out_BitEx(ACCESS_FLG, DA_OUT_ENABLE, BIT_OFF);
            if(sRet != 0)
                goto finalization;

            sRet = CCPU_Y_Out_BitEx(ACCESS_FLG, DIGITAL_OVER, BIT_ON);
            if(sRet != 0)
                goto finalization;
        }
    }
    else
    {
        /* Turn off the D/A conversion output enable/disable signal */
        sRet = CCPU_Y_Out_BitEx(ACCESS_FLG, DA_OUT_ENABLE, BIT_OFF);
        if(sRet != 0)
            goto finalization;
    }
}
else
{
    /* Turn off the D/A conversion output enable/disable signal */
    sRet = CCPU_Y_Out_BitEx(ACCESS_FLG, DA_OUT_ENABLE, BIT_OFF);
    if(sRet != 0)
        goto finalization;
```

```
        }

        /* 1-tick wait*/
         taskDelay(1);

    }

    }

finalization:
    /* Clear the GOT input device to 0*/
    usDataBuf4DEV = 0x0000;
    sRet = CCPU_WriteDevice(Dev_CCPU_D, DEV_ADDR, WORD, &usDataBuf4DEV, DUMMY);

    return;
}
```

APPENDICES

Appendix 1 Security Functions

These functions serve to protect the user property stored in a personal computer and the user property inside the C Controller module in the MELSEC iQ-R series system against threats such as theft, tampering, faulty operation, and unauthorized execution due to the unauthorized access by an outsider.
Use an appropriate security function according to the purpose.

Point

Each security function is just one means of preventing unauthorized access (such as destruction of programs or data) from external devices, but does not prevent it completely. To maintain the safety of the C Controller system against unauthorized access from external devices, include measures in addition to the security functions. Mitsubishi Electric Corporation cannot be held responsible for any problems involving system trouble that may occur as a result of unauthorized access.
The following are examples of measures against unauthorized access.

- Install a firewall.
- Install a personal computer as a relay station to control the relaying of send/receive data using an application program.
- Install an external device that can control access rights as a relay station. (For details on external devices that can control access rights, contact the network service provider or equipment dealer.)

Individual identification information

The individual identification information of the C Controller module can be read with the C Controller module dedicated function (CCPU_GetIDInfo). By implementing the activation function in a user program, a user program that does not operate in a C Controller module with different individual identification information can be created.
For details on C Controller module dedicated functions, refer to the following.
MELSEC iQ-R C Controller Programming Manual

A

File access restriction

A file attribute can be set to a file stored in the following types of memory. When a file attribute is set, access to the target file can be restricted, and alteration and file leakage by an unauthorized user can be prevented.

- Program memory
- Data memory
- SD memory card
- USB Mass Storage Class compatible device

Point

- When an SD memory card and USB Mass Storage Class compatible device are mounted on a device other than the C Controller module (a peripheral such as a personal computer), an access restricted file can be manipulated. When access restriction is set to a file in an SD memory card and USB Mass Storage Class compatible device, take measures so that the SD memory card and USB Mass Storage Class compatible device cannot be removed from the C Controller module freely.
- Access restriction cannot be applied to folders.

Setting file access restriction

Use the attrib() command to change the attribute of a file to be handled by the C Controller module. To change the file attribute, the security password is required.

■Setting a file attribute

Use the attrib() command to set a file attribute to the file to be restricted.

The following table shows the file attributes the C Controller module can handle.

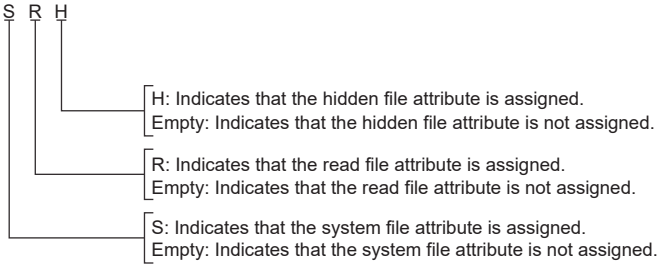
Attribute		Description
S	System file attribute	This attribute helps prohibit file operation.
R	Read only attribute*1	This attribute helps prohibit a file from being deleted or written to.
H	Hidden file attribute*2	A file is hidden when files are displayed in list form using the ls command and when FTP connection is performed.

*1 Not covered by the file access restriction function. However, when this file attribute is set, deleting or writing to a file can be prohibited.

*2 A file can be manipulated when it is opened by specifying the file name. To prohibit manipulation of a file, the system file attribute must be set.

■Checking the file attribute

Use the attrib() command to check the set file attribute.



Checking the file access restriction status

The file access restriction status can be checked with the Shell command or the C Controller module dedicated function (CCPU_GetFileSecurity).

Point

The file access restriction status cannot be checked from the script file (STARTUP.CMD).

Clearing/re-setting file access restriction

The file access restriction status is changed using the Shell command, script file (STARTUP.CMD), or user program. In addition, to change the file access restriction, the security password set with CW Configurator is required.

■Changing the system file attribute

For operation to be performed on a file with the system file attribute, use the C Controller module dedicated function (CCPU_ChangeFileSecurity) to temporarily clear the file access restriction. The cleared setting is configured again by setting file access restriction with the C Controller module dedicated function (CCPU_ChangeFileSecurity) or re-setting the C Controller module.

Point

- To access a file with the system file attribute in the script file "STARTUP.CMD", clear the access restriction inside the script file. In that case, to prevent password leakage, assign the system file attribute to the script file.
- To prevent password leakage, do not use a file with the system file attribute in the script file (STARTUP.CMD) in an SD memory card.

Precautions

■When safety must be maintained from unauthorized access from outside

To maintain the safety of the C Controller system from unauthorized access from outside, include measures taken by the user.

To prevent security password leakage, set a password with the following in mind.

- Avoid setting a password using only simple one-byte alphanumeric characters.
- Set a complex password that also includes symbols.

■Characters that can be used for a security password

One-byte alphanumeric characters and symbols can be used. (Passwords are case-sensitive.)

■If the security password is forgotten

Initialize the C Controller module.

For details on the initialization procedure, refer to the following.

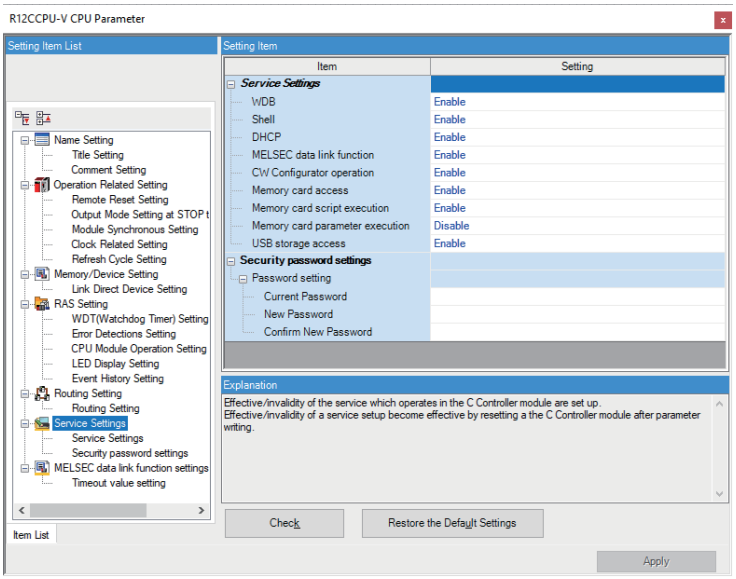
MELSEC iQ-R C Controller Module User's Manual (Startup)

Setting a service

Set a service that operates in the C Controller module.

By restricting the operating service, unauthorized access from another user can be prevented.

Window



Item		Description	Setting range	Default
Setting a service	WDB	Required when connecting CW Workbench.	• Disable • Enable	Enable
	Shell	Required when executing a command.		
	DHCP	Required when using the function that automatically assigns the network settings.		
	MELSEC communication function	Required for MELSEC communications.		
	CW Configurator operation	Required when operating CW Configurator. When this service is disabled, the following operations become unavailable. • Writing to the C Controller module • Reading from the C Controller module • Verifying with the C Controller module • Deleting data from the C Controller module • Operating CPU memory (initialization)		
	Memory card access	Required when accessing a memory card.		
	Memory card script execution	Required when executing the script file "STARTUP.CMD" stored in the memory card.		
	Memory card parameter execution*1	Required when executing parameters stored in the memory card.		Disable
	USB storage access	Required when accessing a USB device.		Enable
Security password setting	Password setting	Set a security password.	8 to 16 characters (one-byte alphanumeric characters, symbols)	password

*1 To change a service, write the parameters to the data memory.
When the parameters are written to the memory card, the service settings are not changed.

Password setting

■Current password

Enter the current security password.

■New password, password for confirmation

To change the security password, enter the new password into "New Password" and "Confirm New Password".

Precautions

■CW Configurator operation

When CW Configurator operation is disabled, parameters cannot be set. To enable the service, initialize the C Controller module.

■Memory card parameter execution

When CW Configurator operation is disabled, parameters cannot be set. To enable the service, initialize the C Controller module.

Lockout

Disables the password authentication for a certain duration of time after a certain number of failed authentication attempts. Brute-force attacks by malicious users can be prevented.

Lockout time

The following table lists the durations of the lockout times.

Number of password input errors*1	Lockout time
1st time to 5th time	0 minutes
6th time	1 minute
7th time	5 minutes
8th time	15 minutes
9th time or later	60 minutes

*1 When the correct password is entered, the number of password input errors is reset.

Point

- While lockout is active, a password input error is not counted. For that reason, even if a 7th input error occurs within one minute after the occurrence of the 6th error, the one minute lockout time is not extended.
- Lockout is not executed if security is set with the C Controller module dedicated function (CCPU_ChangeFileSecurity).

Appendix 2 Modules for Realizing Higher-Speed Analog I/O Conversion

This section describes modules for realizing higher-speed analog I/O conversion.

High-speed analog input module R60ADH4

The high-speed analog input module can perform A/D conversion at high-speed: 1μs/CH, medium speed: 10μs/CH, and low speed: 20μs/CH per sampling period in normal mode. In addition, the simultaneous conversion mode, which makes simultaneous conversion at four channels per module possible, is available and supports sampling at the fastest speed of 5μs/4CH. Also available is the continuous logging function, which can log digital output values for four channels simultaneously and transfer logged data to the CPU module continuously without stopping logging.

High-speed analog output module R60DAH4

For the high-speed analog output module, high-speed conversion (1μs) improves the analog output response performance. This realizes highly responsive feedback control in systems such as drive systems that perform speed control with analog commands.

Appendix 3 CW Workbench

CW Workbench is a product for developing user programs that run on the C Controller module and C intelligent function module.
It is an OEM product of Wind River Systems in the United States. As a subset product of Wind River Workbench 3.3, it implements only the functions minimally required for user program development such as coding, building, and debugging.

Features

CW Workbench has the following features.

Integrated development environment

CW Workbench is an integrated development environment that enables project management, source code editing, building, and debugging, helping develop user programs for the C Controller module and C intelligent function module efficiently.

Same specifications as the specifications of Wind River Workbench 3.3

For the display and operation of the functions common to CW Workbench and Wind River Workbench 3.3, the specifications are the same.

Windows® supported for operating systems for personal computers

The following operating systems for personal computers are supported: Windows®XP, Windows® 7, Windows® 8, Windows® 8.1, and Windows® 10.
In addition, some 64 bit versions of operating systems are supported as well.

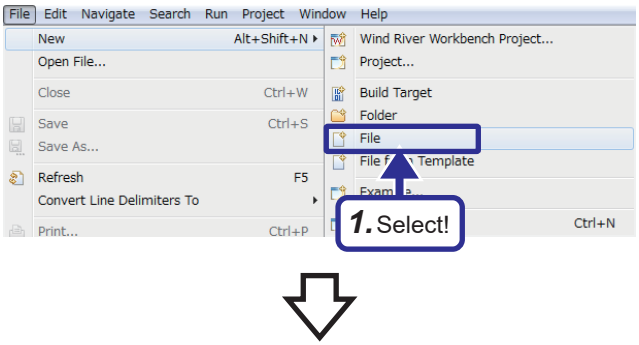
Extension of a function with plugin software

Third-party plugin software can be added easily, so that a function can be extended with ease.

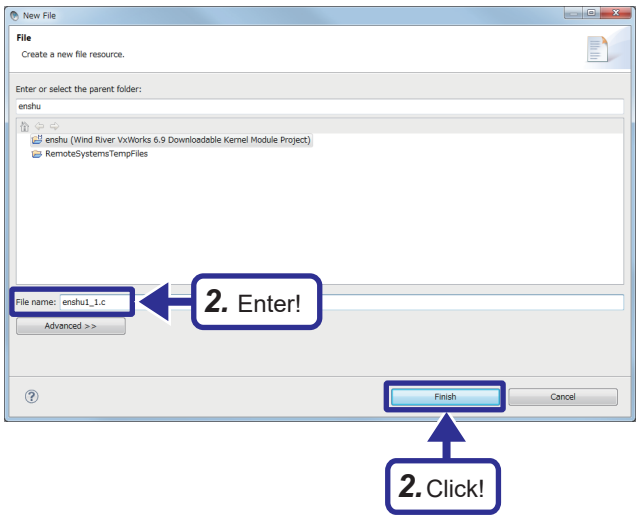
Creating a new user program

Create a user program for controlling the C Controller system.

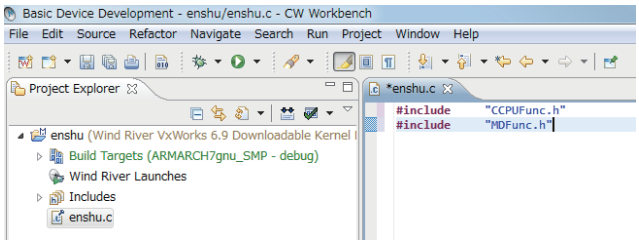
Operating procedure



1. In the "Project Explorer" window, select a project to which a new file will be added, and select [File] ⇨ [New] ⇨ [File].



2. Enter a file name for "File name" and click the [Finish] button.
For the file name to be entered for "File name", enter up to the file extension (.c, .h, .cpp, .hpp).



3. Edit the source file in the "Editor" window, and include files.
C Controller module dedicated function: "CCPUFunc.h"
MELSEC communication function: "MDFunc.h"
Data analysis function/statistical analysis function: "DANLFunc.h"

Point

- If a character that cannot be used on Windows® is entered for "File name", the error text is displayed on the header section of the window, and the [Finish] button is disabled.
- Do not use the characters below for "File name". If a file containing any of the following characters is compiled, a compile error occurs.
  #, $, &, ‘, (,), :, =, `, two-byte character, one-byte katakana character

Programmable Controllers Training Manual
C Controller Basic Course

MODEL	SCHOOL-R REDUNDANT-E
MODEL CODE	
SH(NA)-082520ENG-A(2203)MEE	

MITSUBISHI ELECTRIC CORPORATION

HEAD OFFICE : TOKYO BUILDING, 2-7-3 MARUNOUCHI, CHIYODA-KU, TOKYO 100-8310, JAPAN
NAGOYA WORKS : 1-14 , YADA-MINAMI 5-CHOME , HIGASHI-KU, NAGOYA , JAPAN