Changes for the Better

MITSUBISHI ELECTRIC

Programmable Controllers
C Controller Quick Start Guide

Let's start C Controller!

Smart & Easy

A simpler and more sophisticated integrated-system platform is now available with the C Controller.

Device Debug - UserProject/Sample.c - CW Workbench
File Edit Source Refactor Navigate Search Project Target Run Window Help
Sample.c
Debug
Breakpoints
Build Console
Console
Name
Type
Value
Date
Q06UDHCPU
Q172DCPU
Q06CCPU
QD75MH4
QJ71GP21-SX
QG60
MITSUBISHI

```
    return;
}
/*******************************************************************/
/*  Module status monitor processing                              */
/*******************************************************************/
void        StatusCheckRtn()
{
    short   ret;
    long    path;
    long    plInfo[15];
    long    event[4];
    unsigned long   ulBufSize;
                                                                  */
    /*  Execute QBF_Open
    ret = QBF_Open(UNIT, &path);
    if(ret != 0){
        printf("QBF_Open error  ret = %hd ( %hxH )\n", ret, ret);
        return;
    }
                                                                  */
    /*  Register timer event of 500ms cycle.
    event[0] = 1;   /* Timer event count
    event[1] = 1;   /* Timer event No.1
    event[2] = 500; /* Timer event cycle(500ms)
    event[3] = 1;   /* Timer event synchronization type (1:individual)

    ret = QBF_EntryTimerEvent(path, event);
    if(ret != 0){
```

for a greener tomorrow

eco Changes

HOW TO READ THIS GUIDE

The following table lists symbols used in this guide with descriptions and examples.

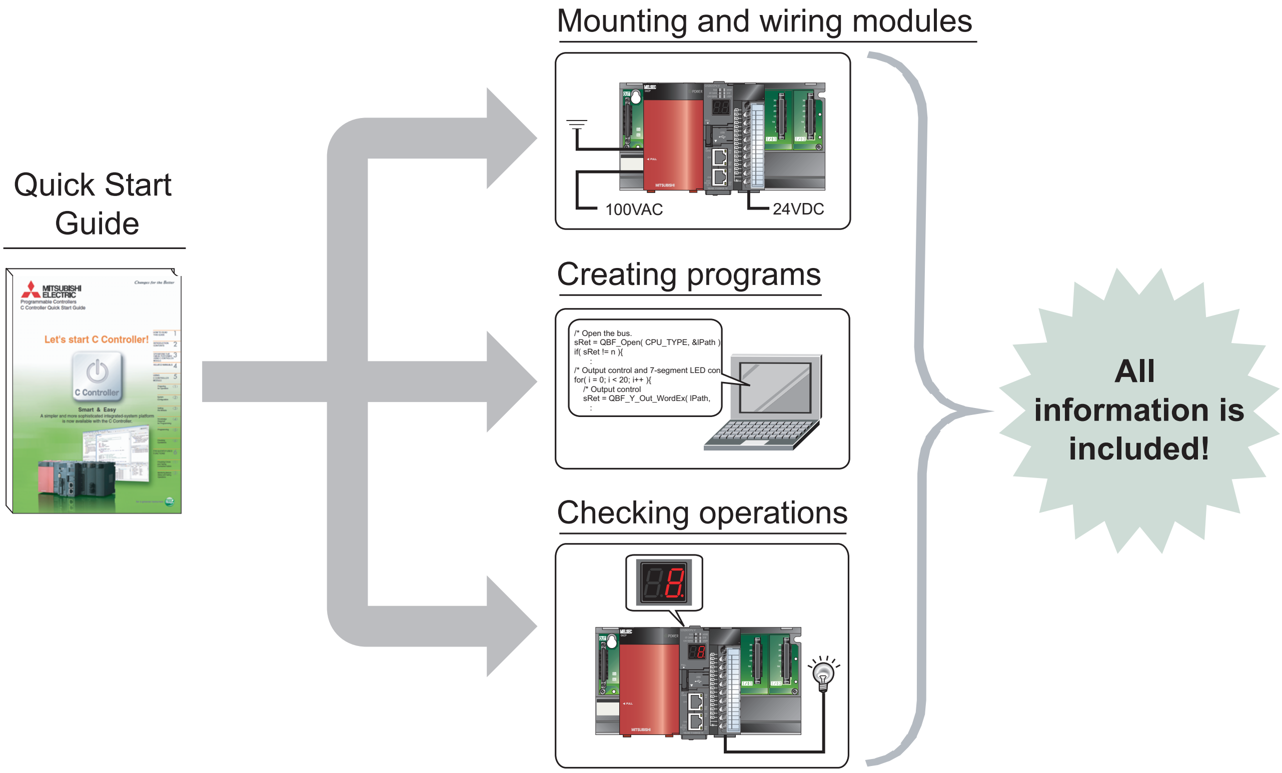
Symbol	Description	Example
Point	Shows information you need to know.	The C Controller module executes program operation regardless of the switch status (RUN/STOP).
Reference	Shows reference manuals and pages on which you can find the details.	Refer to the following. ☞ C Controller Module User's Manual (Hardware Design, Function Explanation) : SH-080766ENG
Terminology	Shows the explanations of terminology.	Buffer memory: The memory of an intelligent function module used to store data (such as setting values and monitored values) for communication with a C Controller module
⚠Caution	Shows descriptions that must be noted.	Power off the system before mounting a module.
[ ]	Menu names on the menu bar ([ ]→[ ] shows drop-down menus.)	Select [Project]→ [Properties].
▭	Buttons on the window	OK button
▭	Keys on the keyboard	F4 key

INTRODUCTION

This guide simply explains the basic operations of a C Controller module for the first-time users of the Mitsubishi programmable controller MELSEC-Q series C Controller module (hereafter abbreviated as C Controller module).

This guide is targeted for users who use the MELSEC-Q series for the first time and are in the following situations:

- Users with experience in C language or C++ language programming
- Users considering to replace the microcomputer board or the personal computer system with a C Controller system


Quick Start Guide
Let's start C Controller!
C Controller
Mounting and wiring modules
100VAC
24VDC
Creating programs
Checking operations
All information is included!

```
/* Open the bus.
sRet = QBF_Open( CPU_TYPE, &lPath )
if( sRet != n ){
       :
/* Output control and 7-segment LED con
for( i = 0; i < 20; i++ ){
    /* Output control
    sRet = QBF_Y_Out_WordEx( lPath,
       :
```

Reference

- ● Precautions

  For safe use of the C Controller module, read "SAFETY PRECAUTIONS" in the C Controller Module User's Manual.

⚠Caution

This guide explains operations using the system configuration in "<2> System Configuration" (P.15).

When designing/operating a system, refer to the manuals listed in the following.

☞ "RELATED MANUALS"(P.12)

CONTENTS

MEMO

2

OPERATIONS THAT CAN BE PERFORMED USING C CONTROLLER MODULE

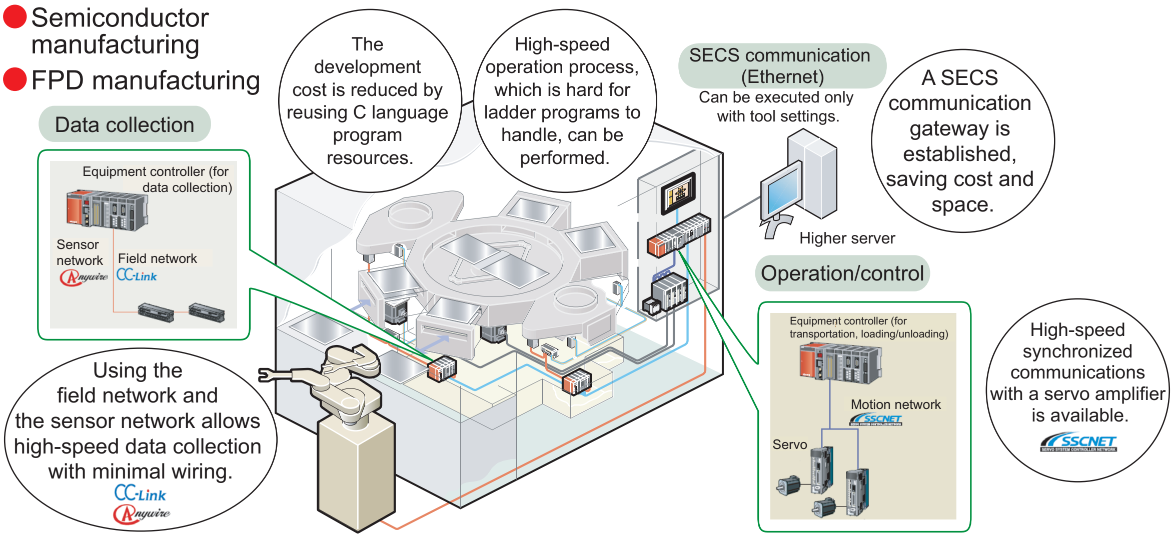
■ Sophisticated and high-speed processes and communications with the higher server

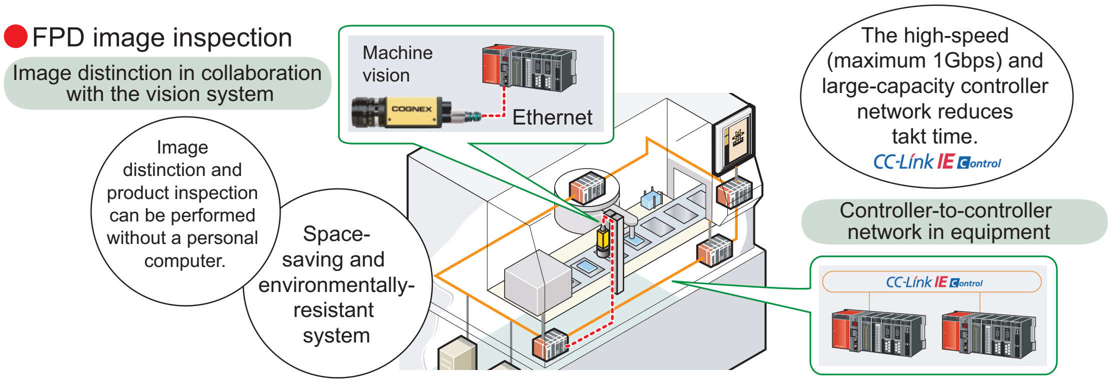
A C Controller module is a CPU module that supervises MELSEC-Q series modules and controls I/O devices using C language or C++ language program. This module also can:

- Reuse a C language or C++ language program developed under a microcomputer board and personal computer environment.
- Perform sophisticated and high-speed operation process, which is hard for ladder programs to handle, required in the fields such as manufacturing of semiconductor products, FPDs, and solar cells; and remote monitoring of public infrastructures (e.g. electricity, gas, and water systems).

The C Controller module easily achieves various functions using user programs.
Combined with partner products, the module can also perform the following functions.

- Program-free SECS communication commonly used for semiconductor manufacturing and direct communication with the higher server without a gateway personal computer can be executed through a SECS communication software package.
- In collaboration with a vision system, image distinction and product inspection can be performed without a personal computer.


● Semiconductor manufacturing
● FPD manufacturing
Data collection
Equipment controller (for data collection)
Sensor network
Field network
Using the field network and the sensor network allows high-speed data collection with minimal wiring.
The development cost is reduced by reusing C language program resources.
High-speed operation process, which is hard for ladder programs to handle, can be performed.
SECS communication (Ethernet)
Can be executed only with tool settings.
A SECS communication gateway is established, saving cost and space.
Higher server
Operation/control
Equipment controller (for transportation, loading/unloading)
Motion network
Servo
High-speed synchronized communications with a servo amplifier is available.


● FPD image inspection
Image distinction in collaboration with the vision system
Machine vision
Ethernet
Image distinction and product inspection can be performed without a personal computer.
Space-saving and environmentally-resistant system
The high-speed (maximum 1Gbps) and large-capacity controller network reduces takt time.
Controller-to-controller network in equipment

■ Various functions for real-time control

The C Controller module equips VxWorks (Wind River Systems, Inc.), real-time OS with many achievements and high reliability (The runtime license does not cost).

Since VxWorks supports a preemptive system*1, allowing real-time operation and sophisticated process that require an interrupt and punctuality, which may not be ensured under personal computer environment.

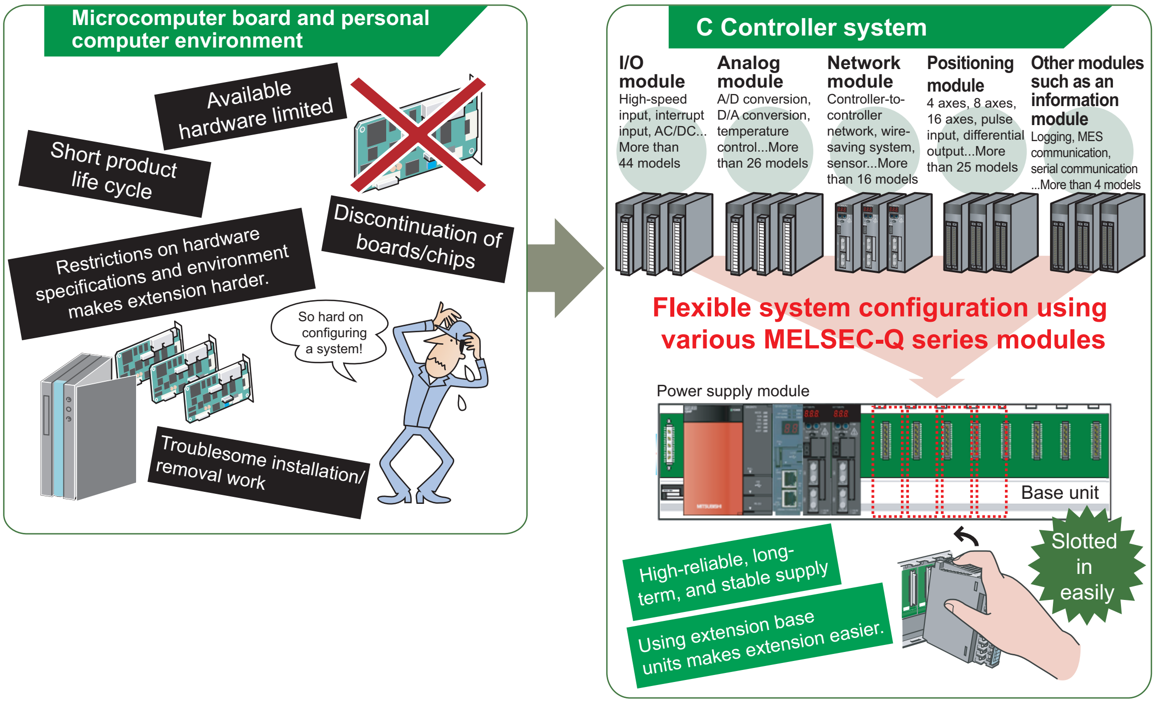
VxWorks also equips various functions, such as file access, drivers for the network functions, I/O and communication libraries, and therefore can be used for various purposes.

*1 A system that equally assigns execution time to multiple programs so that the processor (CPU) may not be dedicated to one program

■ Features

1. Flexible system configuration using various MELSEC-Q series modules

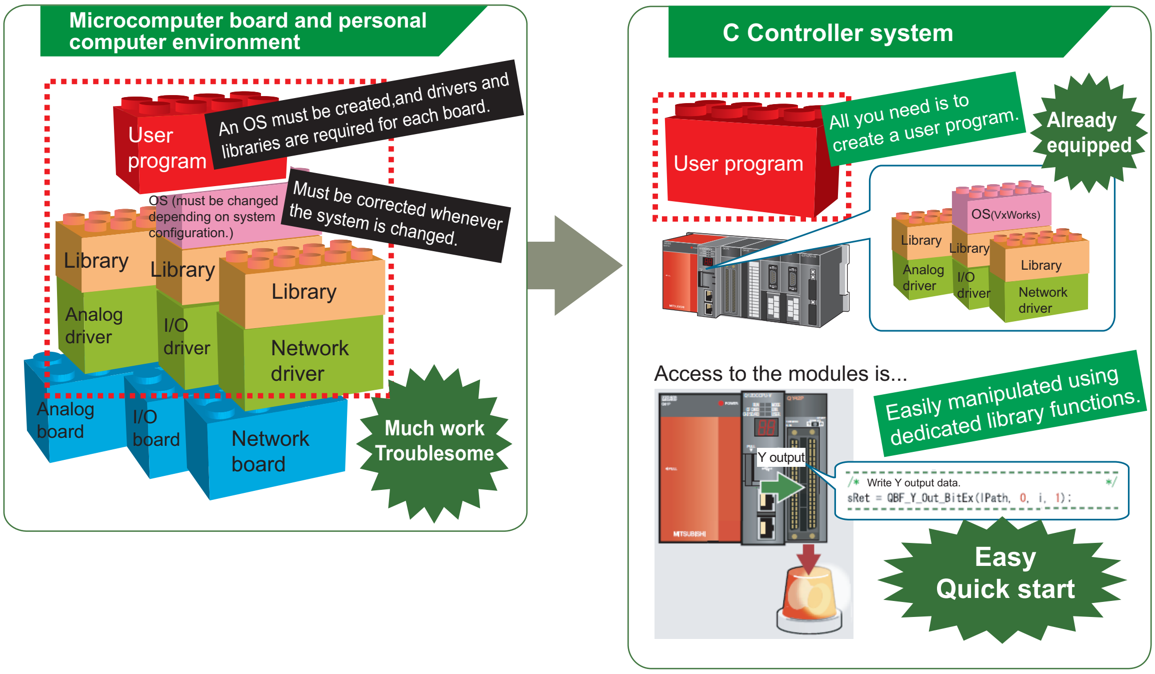
In a C Controller system, program resources can be reused and various MELSEC-Q series modules are available, making system configuration easier.


Microcomputer board and personal computer environment
Available hardware limited
Short product life cycle
Discontinuation of boards/chips
Restrictions on hardware specifications and environment makes extension harder.
So hard on configuring a system!
Troublesome installation/removal work
C Controller system
I/O module
High-speed input, interrupt input, AC/DC... More than 44 models
Analog module
A/D conversion, D/A conversion, temperature control...More than 26 models
Network module
Controller-to-controller network, wire-saving system, sensor...More than 16 models
Positioning module
4 axes, 8 axes, 16 axes, pulse input, differential output...More than 25 models
Other modules such as an information module
Logging, MES communication, serial communication ...More than 4 models
Flexible system configuration using various MELSEC-Q series modules
Power supply module
Base unit
Slotted in easily
High-reliable, long-term, and stable supply
Using extension base units makes extension easier.

2. Equipped OS, drivers, and libraries allow you to focus on developing user programs

Since OS and communication drivers have been equipped with a C Controller module, you are no longer bothered with troublesome work under microcomputer board and personal computer environment (OS porting, driver development, OS writing to ROM) and can focus on developing a user program.

The C Controller module can easily access MELSEC-Q series modules using library functions dedicated for a C Controller module (bus interface function, MELSEC communication function).

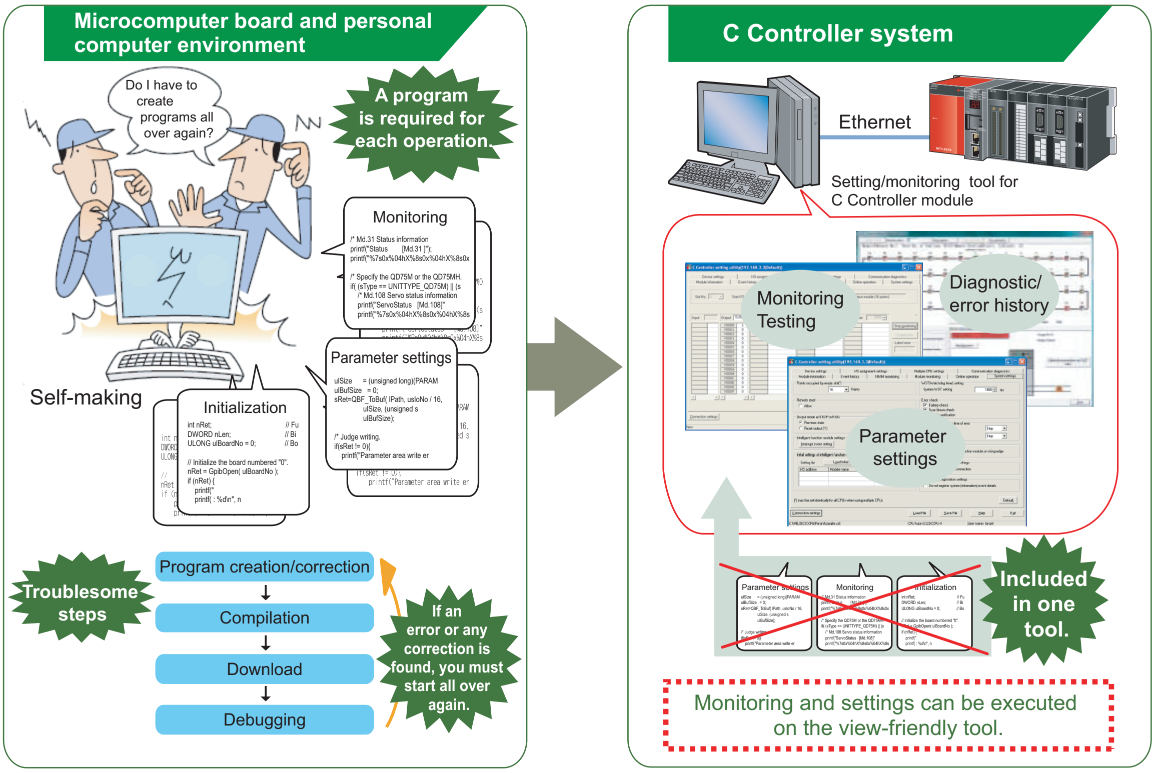

Microcomputer board and personal computer environment
User program
An OS must be created,and drivers and libraries are required for each board.
OS (must be changed depending on system configuration.)
Must be corrected whenever the system is changed.
Library
Library
Library
Analog driver
I/O driver
Network driver
Analog board
I/O board
Network board
Much work Troublesome
C Controller system
User program
All you need is to create a user program.
Already equipped
OS(VxWorks)
Library
Library
Library
Analog driver
I/O driver
Network driver
Access to the modules is...
Easily manipulated using dedicated library functions.
Y output
Easy Quick start

```
/*  Write Y output data.                                  */
sRet = QBF_Y_Out_BitEx(lPath, 0, i, 1);
```

3. Initialization, parameter settings, monitoring, and testing can be executed without a program

Complex programs for the initialization and the system settings of a C Controller module, and parameter settings of a network module are not required. The operations can be easily executed on view-friendly setting/monitoring tool for C Controller module.

Programs to check module status, errors occurred in a C Controller module and in a user program, cable disconnection, and communication status are also not required.


Microcomputer board and personal computer environment
Do I have to create programs all over again?
A program is required for each operation.
Self-making
Monitoring
Parameter settings
Initialization
Troublesome steps
Program creation/correction
Compilation
Download
Debugging
If an error or any correction is found, you must start all over again.
C Controller system
Ethernet
Setting/monitoring tool for C Controller module
Monitoring Testing
Diagnostic/ error history
Parameter settings
Parameter settings
Monitoring
Initialization
Included in one tool.
Monitoring and settings can be executed on the view-friendly tool.

```
/* Md.31 Status information
printf("Status       [Md.31 ]");
printf("%7s0x%04hX%8s0x%04hX%8s0x

/* Specify the QD75M or the QD75MH.
if( (sType == UNITTYPE_QD75M) || (s
   /* Md.108 Servo status information
   printf("ServoStatus   [Md.108]"
   printf("%7s0x%04hX%8s0x%04hX%8s
```

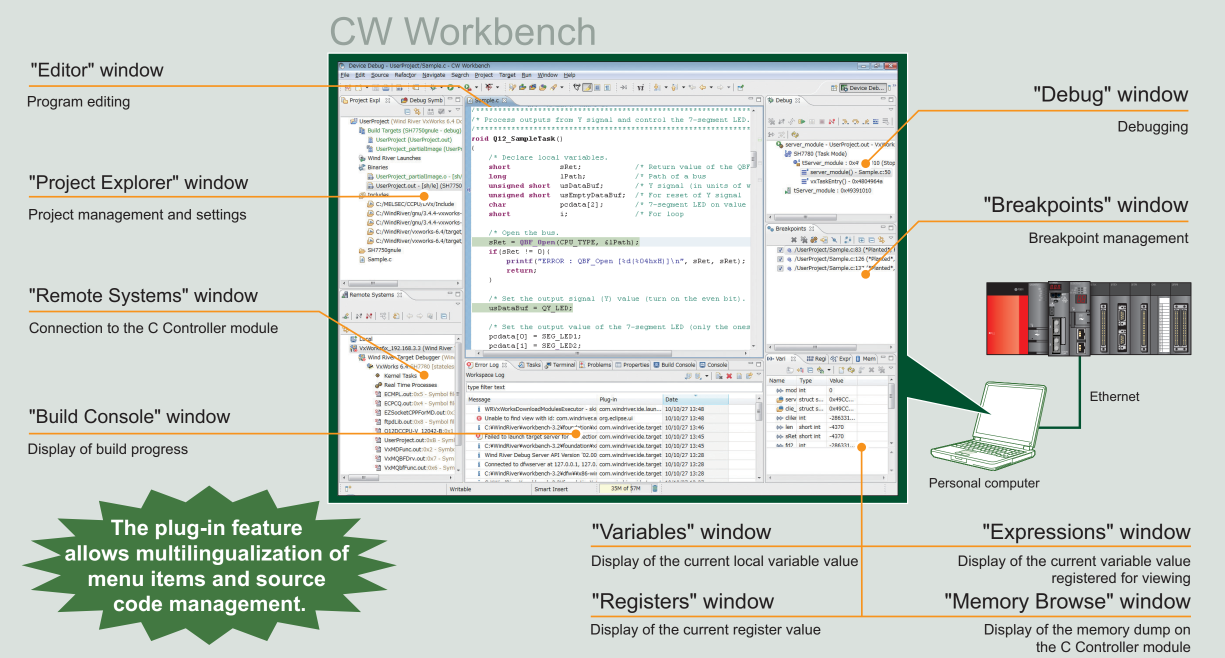
```
ulSize      = (unsigned long)(PARAM
ulBufSize   = 0;
sRet=QBF_ToBuf( IPath, usIoNo / 16,
             ulSize, (unsigned s
             ulBufSize);

/* Judge writing.
if(sRet != 0){
   printf("Parameter area write er
```

```
int nRet;                         // Fu
DWORD nLen;                       // Bi
ULONG ulBoardNo = 0;              // Bo

// Initialize the board numbered "0".
nRet = GpibOpen( ulBoardNo );
if (nRet) {
   printf("
   printf( : %d\n", n
```

4. Quick start using an integrated development environment, "CW Workbench"

An engineering tool for C Controller, "CW Workbench", equips basic functions such as program editing, generation of execution module, and debugging. A user program for a C Controller module is easily developed.

Eclipse-based CW Workbench allows function enhancement using a third-party plug-in software.



CW Workbench

"Editor" window

Program editing

"Project Explorer" window

Project management and settings

"Remote Systems" window

Connection to the C Controller module

"Build Console" window

Display of build progress

"Debug" window

Debugging

"Breakpoints" window

Breakpoint management

Ethernet

Personal computer

"Variables" window

Display of the current local variable value

"Registers" window

Display of the current register value

"Expressions" window

Display of the current variable value registered for viewing

"Memory Browse" window

Display of the memory dump on the C Controller module

The plug-in feature allows multilingualization of menu items and source code management.

```
/***************************************************************
/* Process outputs from Y signal and control the 7-segment LED.
/***************************************************************
void Q12_SampleTask()
{
    /* Declare local variables.
    short           sRet;           /* Return value of the QBF
    long            lPath;          /* Path of a bus
    unsigned short  usDataBuf;      /* Y signal (in units of w
    unsigned short  usEmptyDataBuf; /* For reset of Y signal
    char            pcdata[2];      /* 7-segment LED on value
    short           i;              /* For loop

    /* Open the bus.
    sRet = QBF_Open(CPU_TYPE, &lPath);
    if(sRet != 0){
        printf("ERROR : QBF_Open [%d(%04hxH)]\n", sRet, sRet);
        return;
    }

    /* Set the output signal (Y) value (turn on the even bit).
    usDataBuf = QY_LED;

    /* Set the output value of the 7-segment LED (only the ones
    pcdata[0] = SEG_LED1;
    pcdata[1] = SEG_LED2;
```

Workspace Log

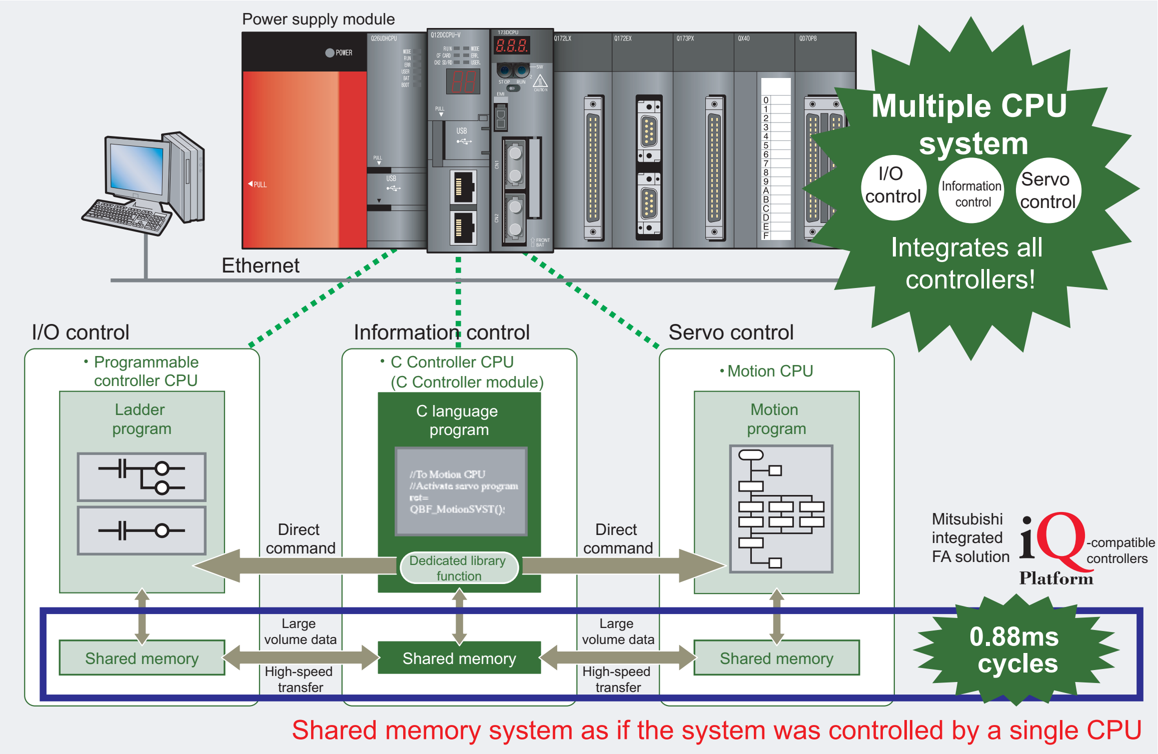
Message	Plug-in	Date
WRVxWorksDownloadModulesExecutor - ski	com.windriver.ide.laun...	10/10/27 13:48
Unable to find view with id: com.windriver.a	org.eclipse.ui	10/10/27 13:48
C:¥WindRiver¥workbench-3.2¥foundation¥x	com.windriver.ide.target	10/10/27 13:46
Failed to launch target server for ection	com.windriver.ide.target	10/10/27 13:45
C:¥WindRiver¥workbench-3.2¥foundation¥x	com.windriver.ide.target	10/10/27 13:45
Wind River Debug Server API Version '02.00	com.windriver.ide.target	10/10/27 13:28
Connected to dfwserver at 127.0.0.1, 127.0.	com.windriver.ide.target	10/10/27 13:28
C:¥WindRiver¥workbench-3.2¥dfw¥¥x86-wi	com.windriver.ide.target	10/10/27 13:28

5. High-speed and high-precise control by multiple CPU high speed bus transmission

Multiple CPU high speed bus transmission supports real-time sequential control synchronized with the operation cycle of the Motion CPU (0.88ms) and tracking control to keep up with the constant changes in the target value.

Additionally, large volume data up to 14K words can be transferred at high speed (0.88ms cycles) without a program, and data can be shared among CPUs.

By integrating the C Controller module with the CPU that serves as the nerve center of the factory, the entire system can be efficiently controlled and the load of computational processing can be distributed.



Shared memory system as if the system was controlled by a single CPU

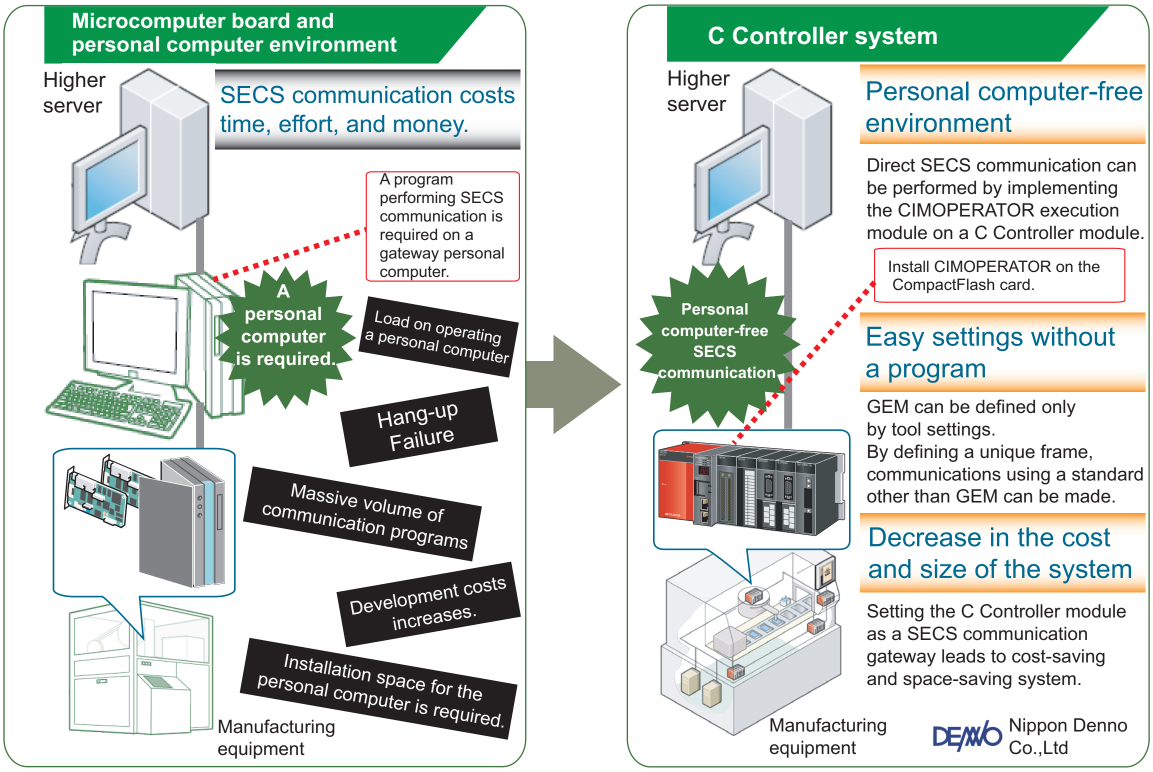
6. Wide application using partner products

In combination with the following partner products, higher functionality and easy information link can be achieved.

(1) Information link with SECS communication software package (CIMOPERATOR SECS+)

Introduction of CIMOPERATOR achieves personal computer-free and program-free SECS communication (GEM[*1]/non-GEM) with the higher server, enabling status management and information collection of manufacturing equipment.

[*1] One of the industry-standard communication protocol used in semiconductor manufacturing lines


Microcomputer board and personal computer environment
Higher server
A personal computer is required.
Manufacturing equipment
C Controller system
Higher server
Personal computer-free SECS communication
Manufacturing equipment
Nippon Denno Co.,Ltd

SECS communication costs time, effort, and money.

A program performing SECS communication is required on a gateway personal computer.

Load on operating a personal computer

Hang-up Failure

Massive volume of communication programs

Development costs increases.

Installation space for the personal computer is required.

Personal computer-free environment

Direct SECS communication can be performed by implementing the CIMOPERATOR execution module on a C Controller module.

Install CIMOPERATOR on the CompactFlash card.

Easy settings without a program

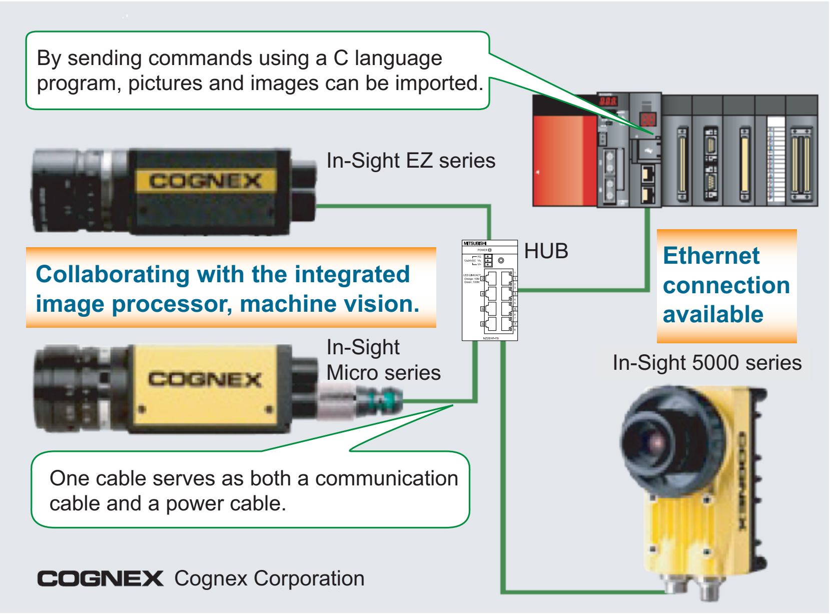
GEM can be defined only by tool settings.
By defining a unique frame, communications using a standard other than GEM can be made.

Decrease in the cost and size of the system

Setting the C Controller module as a SECS communication gateway leads to cost-saving and space-saving system.

(2) Collaboration with the vision system (COGNEX In-Sight EZ, In-Sight Micro, and In-Sight5000 series)

Collaboration of the COGNEX machine vision with the C Controller module can easily automate manufacturing processes including measurement, inspection, and distinction of products.



By sending commands using a C language program, pictures and images can be imported.

Collaborating with the integrated image processor, machine vision.

Ethernet connection available

One cable serves as both a communication cable and a power cable.

RELATED MANUALS

This guide explains the basic operations of a C Controller module.
To make maximum use of the C Controller module, refer to the following.

■ Learning about a C Controller module

● C Controller Module User's Manual (Hardware Design, Function Explanation)
....SH-080766ENG

This manual explains the system configuration, specifications, functions, handling, wiring, and troubleshooting of a C Controller module.

● C Controller Module User's Manual (Utility Operation, Programming)
....SH-080767ENG

This manual explains the installation and uninstallation of SW□PVC-CCPU, utility operations, and functions and programming using SW□PVC-CCPU.

■ Learning about CW Workbench

● CW Workbench Operating Manual SH-080982ENG

This manual explains the system configuration, installation and uninstallation, specifications, functions, and troubleshooting of CW Workbench.

USING C CONTROLLER MODULE

The C Controller module is installed with procedures as shown below.

<1> Preparing for Operation (P.14)
Preparing the necessary devices

<2> System Configuration (P.15)
Installing, wiring, and powering on the devices

1) System configuration example (P.15)
Showing the example of system configuration used in this guide
2) Mounting the modules (P.16)
Mounting the prepared modules on a base unit
3) Wiring the modules (P.17)
Wiring the power supply module and the output module
4) Checking the power supply module (P.19)
Powering on the system and checking module status

<3> Setting the Module (P.21)
Configuring settings to operate the C Controller module using C Controller setting utility

1) Initializing the C Controller module (P.21)
Preparing a standard RAM
2) Setting parameters (P.23)
Setting parameters for the C Controller module

<4> Knowledge Required for Programming (P.26)
Explaining the bus interface function

<5> Programming (P.29)
Creating a program using CW Workbench

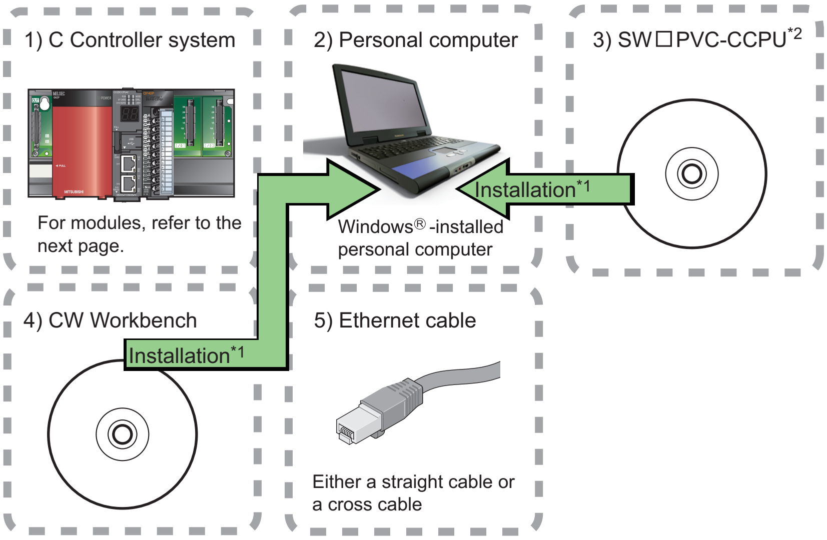
1) Creating a project (P.32)
Starting CW Workbench, creating projects, and configure settings
2) Creating a user program (P.36)
Creating a user program that controls a C Controller system.
3) Generating an execution module from the user program (P.37)
Converting (Building) the created program into an executable module
4) Connecting a C Controller module to CW Workbench (P.38)
Connecting a C Controller module to CW Workbench to perform debugging
5) Debugging the user program (P.40)
Checking operations of the created program
6) Registering an execution module (P.45)
Building the created program for operation and storing the program on the C Controller module

<6> Checking Operations (P.47)
Executing the program and checking operations

5
〈1〉
〈2〉
〈3〉
〈4〉
〈5〉
〈6〉

<1> Preparing for Operation

Prepare the necessary devices.



*1 Install SW□PVC-CCPU and CW Workbench on the same personal computer beforehand.

Reference

For installation of SW□PVC-CCPU, refer to the following.

☞ C Controller Module User's Manual (Utility Operation, Programming): SH-080767ENG

For installation of CW Workbench, refer to the following.

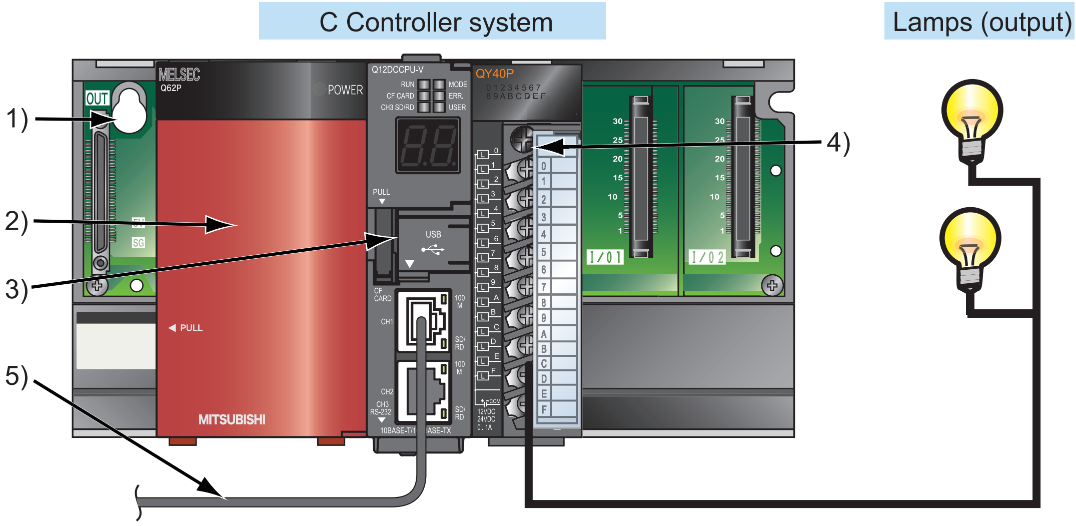
☞ CW Workbench Operating Manual: SH-080982ENG

*2 SW□PVC-CCPU is a setting/monitoring tool for C Controller module.

<2> System Configuration

1) System configuration example

This guide uses the following system configuration as an example.



*A wire to the power supply module is omitted.

No.	Name	Model	Description
1)	Base unit	Q33B	A unit on which a power supply module, a C Controller module, and I/O modules are mounted
2)	Power supply module	Q62P	Supplies power to modules such as a C Controller module and I/O modules.
3)	C Controller module	Q12DCCPU-V	Supervises the control process of a C Controller system.
4)	Output module	QY40P	-
5)	Cable (Ethernet cable)	An Ethernet cable meeting 10BASE-T/100BASE-TX standards	Connects the personal computer with SW☐PVC-CCPU and CW Workbench installed to the C Controller module.

2) Mounting the modules

Mount the prepared modules on a base unit.

When using the C Controller module for the first time, connect a battery connector.

⚠Caution

- ● Mount a battery before operation.
- ● Power off the system before mounting a module.

Point

● Mounting a battery to the C Controller module

1) Open the cover at the bottom of the C Controller module.

↓

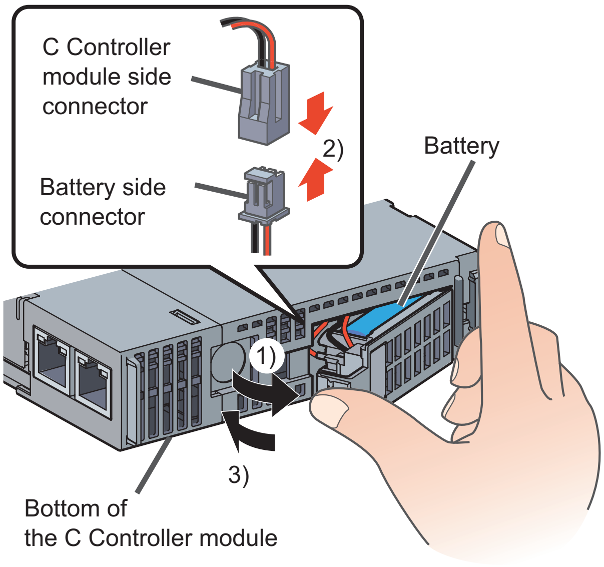
2) Insert the battery side connector into the C Controller module side connector in correct orientation.

↓

3) Close the cover at the bottom of the C Controller module.

↓

End


C Controller module side connector
2)
Battery side connector
Battery
1)
3)
Bottom of the C Controller module

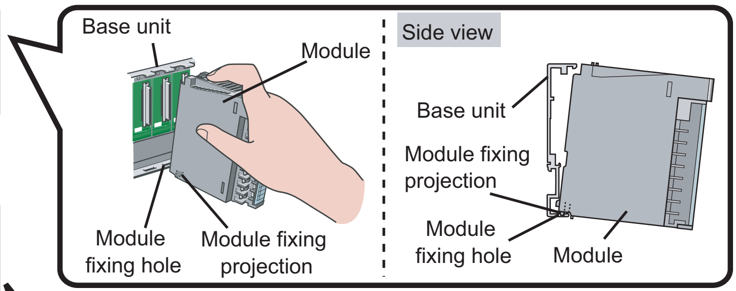
1) Insert the module fixing projection into the module fixing hole on the base unit.

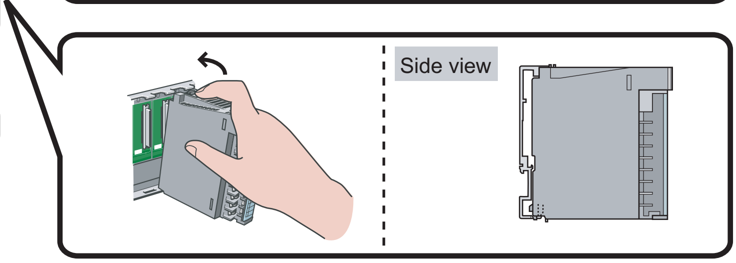
↓

2) Fully insert the module fixing projection in the arrow direction and press the module until the projection snaps into place.

↓

End





Reference

For how to remove a module, refer to the following.

☞ QCPU User's Manual (Hardware Design, Maintenance and Inspection): SH-080483ENG

3) Wiring the modules

Wire the power supply module.

⚠Caution

Power off the system before wiring the module.

Reference

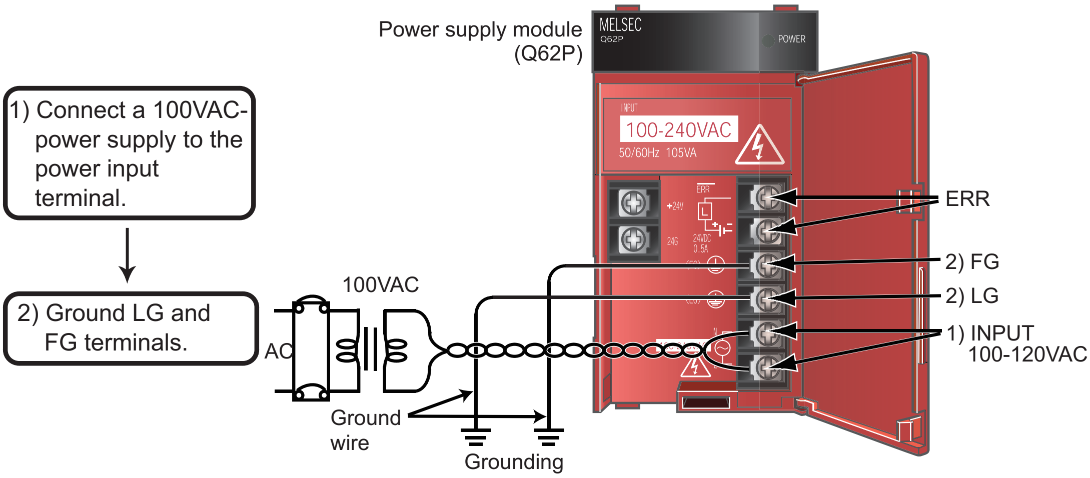
For wiring precautions, refer to the following.

☞ QCPU User's Manual (Hardware Design, Maintenance and Inspection): SH-080483ENG

1. Wiring the power supply module

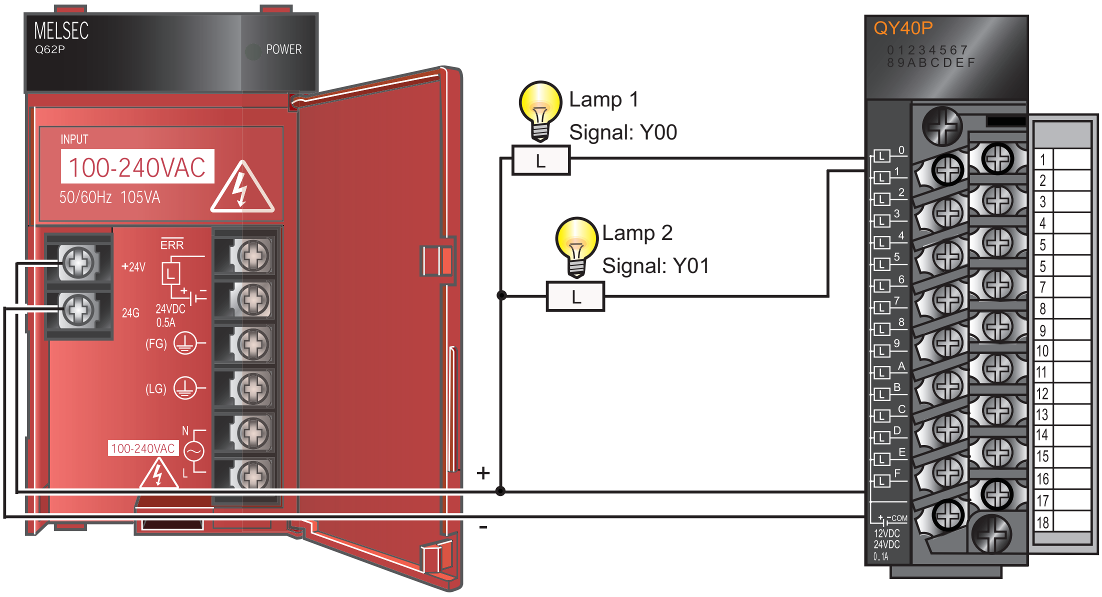
The following shows an example of wiring the power wire and the ground wire to the base unit.

Provide grounding to prevent electric shock and malfunction.



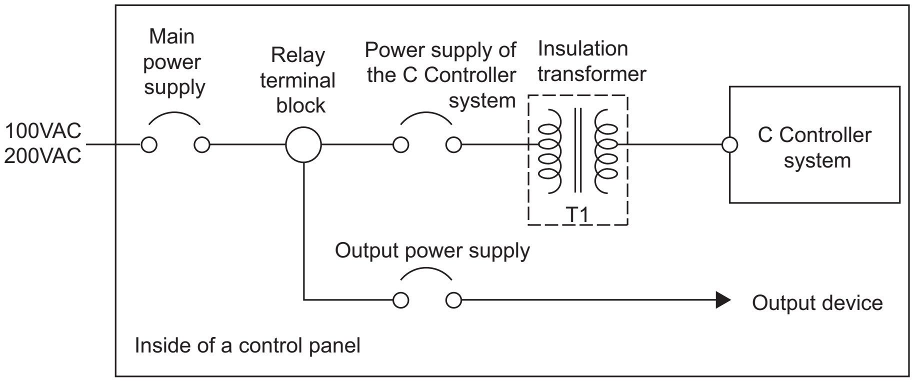
2. Wiring the output module

The following shows an example of wiring the output module (QY40P).



Point

Wire the power supply line of the output device and that of the C Controller system separately as shown below.



4) Checking the power supply module

Check that the power supply module runs normally after installing the system, mounting the modules, and wiring the system.

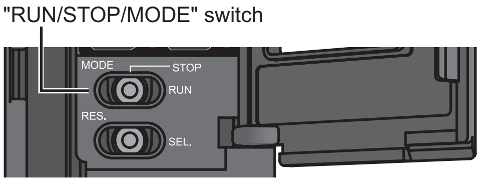
Operating procedure

1. Check the following before powering on the system.

- Wiring to the power supply module
- Power supply voltage

2. Set the C Controller module to STOP.

Open the cover on the front of the C Controller module and set the "RUN/STOP/MODE" switch to "STOP".

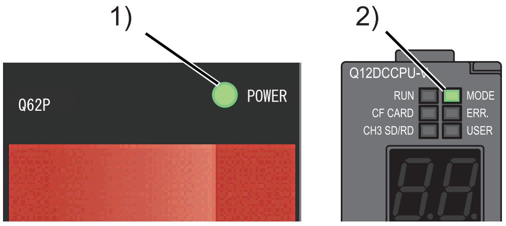


3. Power on the power supply module.

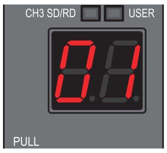
4. Check that the power supply module runs normally.

Check the front LED on each module.
The following lists the normal status of the LEDs.

1) Power supply module: The "POWER" LED lights in green.
2) C Controller module: The "MODE" LED lights in green.



When the C Controller module is the default (the standard RAM has not been initialized), the 7-segment LED displays a flashing "01". However, this does not mean a problem in this step.
The LED turns off after the module is initialized.



☞ "<3> Setting the Module" (P.21)

⇩

Construction of the system is ended.

Point

If the "POWER" LED of the power supply module remains off even after power-on, check that the power supply module is correctly wired and mounted.

Reference

If the "ERR." LED turns on or starts flashing, troubleshoot with reference to the following.

☞ C Controller Module User's Manual (Hardware Design, Function Explanation)
: SH-080766ENG

<3> Setting the Module

Configure settings to operate the C Controller module.

1) Initializing the C Controller module

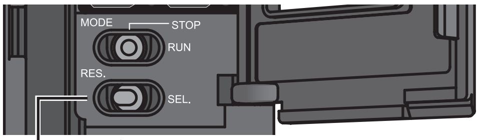
Prepare a standard RAM for the C Controller module.

⚠Caution

All files in the standard RAM are erased by module initialization.

Operating procedure

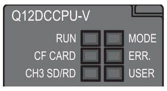
1) Open the cover on the module front and set the "RESET/SELECT" switch to "RESET".



Set the switch to "RESET".

"RESET/SELECT" switch

2) Check that the "MODE" LED is off.

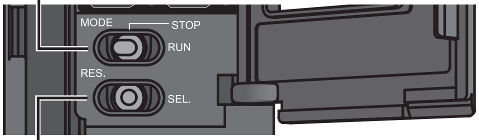


RUN : OFF
MODE : OFF
USER : OFF

3) Holding the "RUN/STOP/MODE" switch on the "MODE" position, set the "RESET/SELECT" switch to the center.

"RUN/STOP/MODE" switch

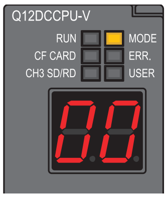
Hold the switch on the "MODE" position.



Return the switch to the center.

"RESET/SELECT" switch

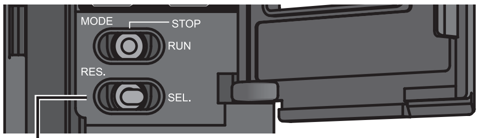
4) Check that the "MODE" LED lights in "orange", and the 7-segment LED displays "00".



RUN : OFF
MODE : Lights in orange.
USER : OFF

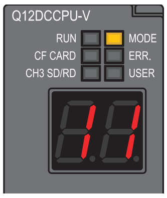
5) Release the "RUN/STOP/MODE" switch.
The switch returns to the "STOP" position.

6) Repeatedly set the "RESET/SELECT" switch to "SELECT" until the 7-segment LED displays "11" ("module initialization setting" mode).



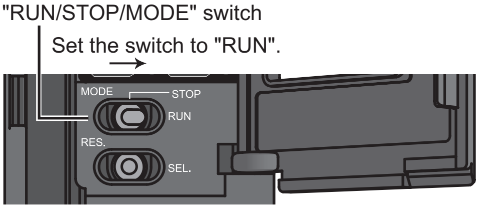
Repeatedly set the switch to "SELECT".

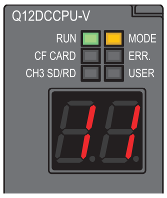
"RESET/SELECT" switch



RUN : OFF
MODE : Lights in orange.
USER : OFF

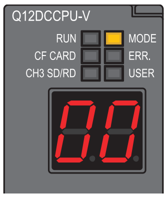
7) Set the "RUN/STOP/MODE" switch to "RUN" and initialize the module. The "RUN" LED will be flashing during initialization.





RUN : Flashes in green.
MODE: Lights in orange.
USER : OFF

8) Check that the "RUN" LED turns off, and the 7-segment LED displays "00". Reset the C Controller module.

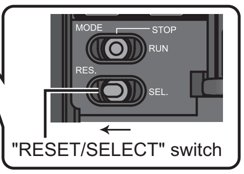


RUN : OFF
MODE: Lights in orange.
USER : OFF

Point

Resetting procedure

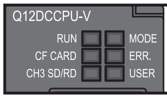
1) Set the "RESET/SELECT" switch on the front of the C Controller module to "RESET".



↓

2) Check that the "MODE" LED turns off.

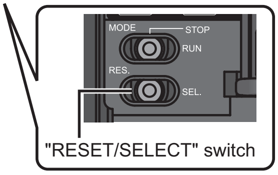
[Reset end status]



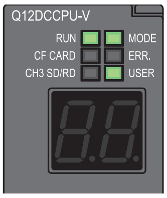
MODE: OFF

↓

3) Set the "RESET/SELECT" switch to the center.

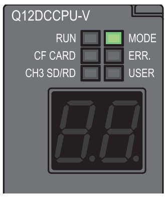


9) Resetting the C Controller module will format the standard RAM.
The "RUN" LED and the "USER" LED start flashing in green.



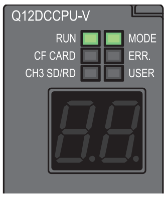
RUN : Flashes in green.
MODE: Lights in green.
USER : Flashes in green.

10) When the formatting is ended, the "RUN" LED and the "USER" LED end flashing, and the "MODE" LED starts flashing in green.



RUN : OFF
MODE: Flashes in green.
USER : OFF

11) Reset the C Controller module.
When the formatting is completed, the "RUN" LED and the "MODE" LED light in green.



RUN : Lights in green.
MODE: Lights in green.
USER : OFF

⚠Caution

Do not operate the switches using a sharp-pointed tool such as a driver.
Doing so may damage the switches.

2) Setting parameters

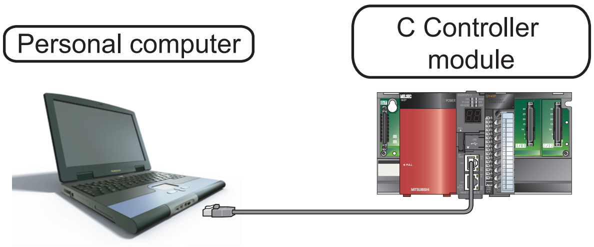
Set parameters for the C Controller module.

Terminology

Parameter: Setting data required for a C Controller system to operate.
Set modules and a network in a C Controller system using C Controller setting utility.

1. Connecting a C Controller module to a personal computer

Connect CH1 of the C Controller module to a personal computer using an Ethernet cable.



⚠Caution

The IP address of the C Controller module and that of the personal computer must be set to the same segment.
Since this guide uses the default IP address for the C Controller module (192.168.3.3), set the IP address for the personal computer to "192.168.3.* (*: other than 0, 3, and 255)".
Set the subnet mask for the personal computer to "255.255.255.0".

Reference

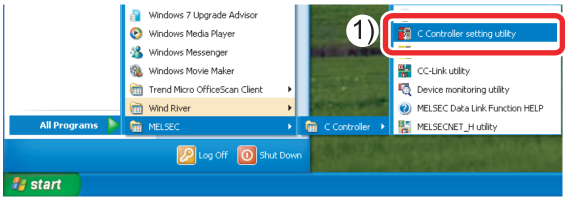
For how to change an IP address, refer to the following.

☞ C Controller Module User's Manual (Hardware Design, Function Explanation)
: SH-080766ENG

2. Starting C Controller setting utility

Operating procedure

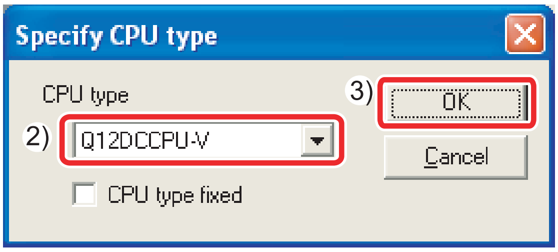
1) Select [start]→[All Programs]→[MELSEC]→[C Controller]→[C Controller setting utility].



The "Specify CPU type" window appears.

2) Select "Q12DCCPU-V".

3) Click the OK button.



Specify CPU type

CPU type: Q12DCCPU-V

□ CPU type fixed

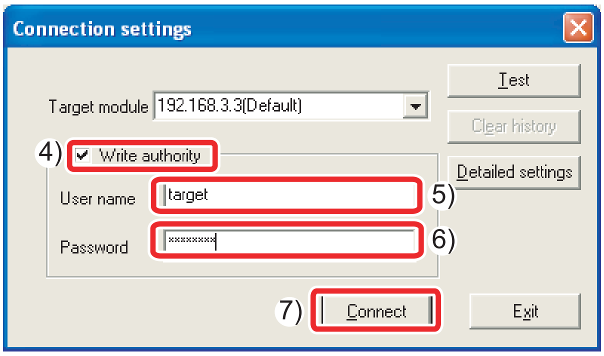
The "Connection settings" window appears.

4) Select the "Write authority" check box.

5) Enter "target".

6) Enter "password".

7) Click the Connect button.



Connection settings

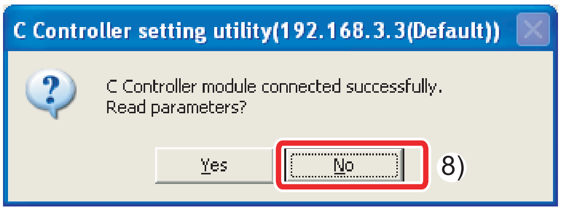
Target module: 192.168.3.3(Default)

☑ Write authority

User name: target

Password: xxxxxxxx

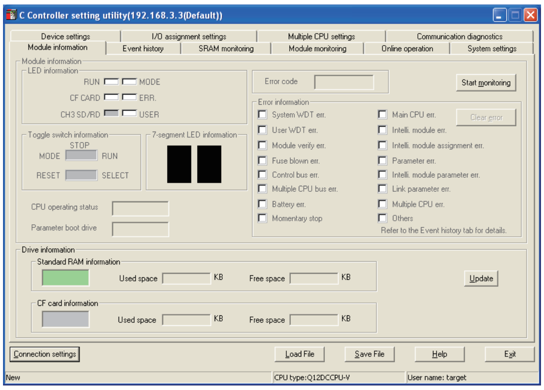
8) Click the No button.



C Controller setting utility(192.168.3.3(Default))

C Controller module connected successfully.
Read parameters?

The "C Controller setting utility" window appears.



C Controller setting utility(192.168.3.3(Default))

CPU type:Q12DCCPU-V

User name: target

3. Writing the parameters to the C Controller module

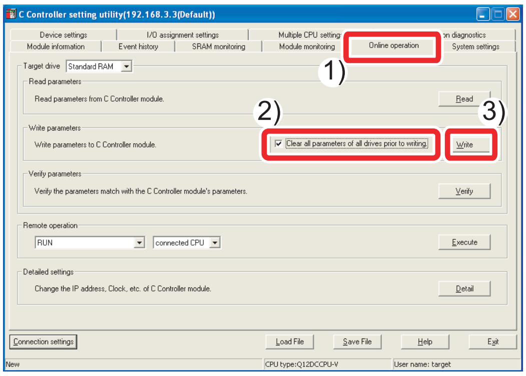
Write the parameters to the C Controller module using C Controller setting utility.

Operating procedure

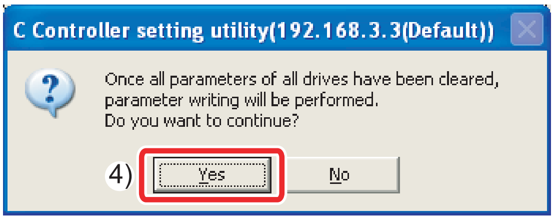
1) Select the "Online operation" tab in C Controller setting utility.

2) Select the "Clear all parameters of all drives prior to writing." check box.

3) Click the Write button.



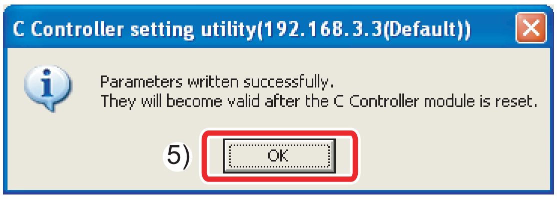
4) Click the Yes button.


C Controller setting utility(192.168.3.3(Default))

Once all parameters of all drives have been cleared, parameter writing will be performed.
Do you want to continue?

5) When the writing is completed, the following window appears.

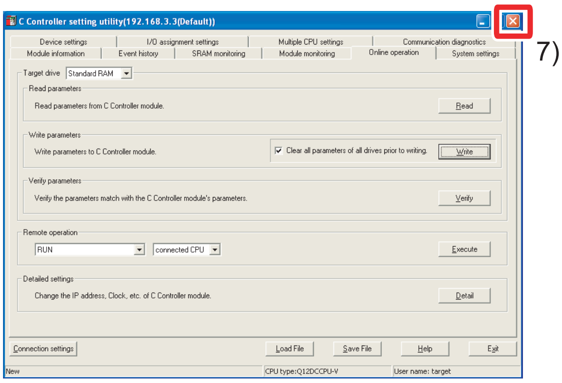
Click the OK button.


C Controller setting utility(192.168.3.3(Default))

Parameters written successfully.
They will become valid after the C Controller module is reset.

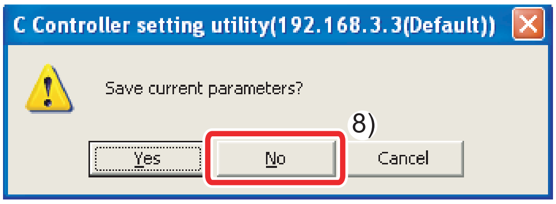
6) Reset the C Controller module.
After resetting the C Controller module, the written parameters will be valid.

7) Click the ☒ button to exit C Controller setting utility.



8) The following window appears.

Click the No button.


C Controller setting utility(192.168.3.3(Default))

Save current parameters?

<4> Knowledge Required for Programming

1. Bus interface functions

The bus interface function is a library function dedicated for a C Controller module.
Using this function in a user program allows a C Controller module to easily control MELSEC-Q series modules.

(1) Opening/closing a bus

To use the functions, open a bus at the start of the program and close the bus at the end of the program.

Functions to open/close a bus

Name	Function
QBF_Open	Opens a bus.
QBF_Close	Closes a bus.

Point

Open or close a bus (QBF_Open/QBF_Close functions) once at the start of a program and at the end of a program, respectively.
By using these functions only once, communication performance will be improved.

(2) I/O access

1-point access and 1-word access are available.

1) 1-point access: A function that treats 1-point data (ON/OFF of switches and lamps)

Example of 1-point access functions

Name	Function
QBF_X_In_BitEx	Reads an input signal (X) in units of one point.
QBF_Y_Out_BitEx	Outputs an output signal (Y) in units of one point.
QBF_Y_In_Bit_Ex	Reads an output signal (Y) in units of one point.

2) 1-word access: A function that treats 1-word (16 bits) data (numeric values, characters)

Example of 1-word access functions

Name	Function
QBF_X_In_WordEx	Reads an input signal (X) in units of words.
QBF_Y_Out_WordEx	Outputs an output signal (Y) in units of words.
QBF_Y_In_WordEx	Reads an output signal (Y) in units of words.

(3) User LED control

USER LED control and 7-segment LED control are available.

Example of user LED control functions

Name	Function
QBF_ControlLED	Controls the "USER" LED of a C Controller module.
QBF_Control7SegLED	Controls the 7-segment LED of a C Controller module.

Reference

Only the basic bus interface functions are explained in this section.
Bus interface function for controlling modules and the MELSEC communication function are also available.

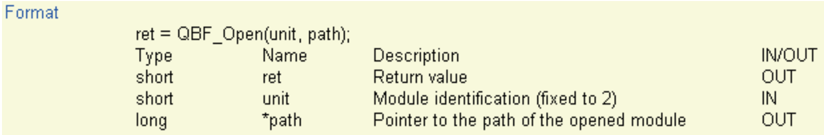
☞ Bus interface function help window and MELSEC communication function help window in SW□PVC-CCPU

☞ C Controller Module User's Manual (Utility Operation, Programming): SH-080767ENG

2. Bus interface functions used in this guide

Basic bus interface functions, output access and 7-segment LED control, are used in the program created in this guide.

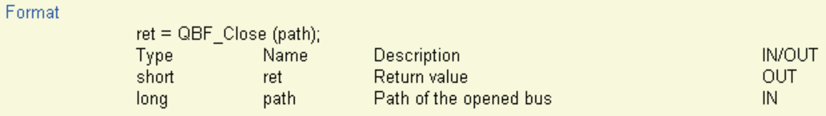
- Opening/closing a bus: QBF_Open/QBF_Close functions



Format

ret = QBF_Open(unit, path);

Type	Name	Description	IN/OUT
short	ret	Return value	OUT
short	unit	Module identification (fixed to 2)	IN
long	*path	Pointer to the path of the opened module	OUT

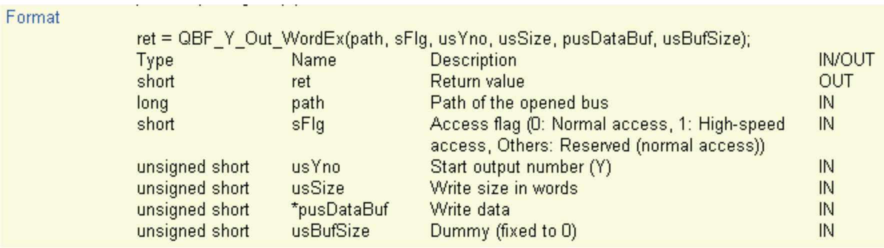


Format

ret = QBF_Close (path);

Type	Name	Description	IN/OUT
short	ret	Return value	OUT
long	path	Path of the opened bus	IN

- Output access: QBF_Y_Out_WordEx function

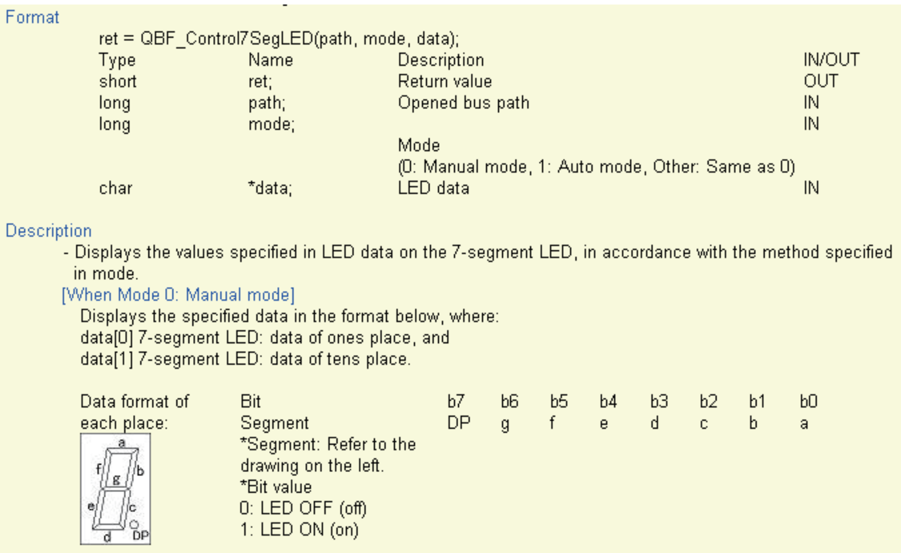


Format

ret = QBF_Y_Out_WordEx(path, sFlg, usYno, usSize, pusDataBuf, usBufSize);

Type	Name	Description	IN/OUT
short	ret	Return value	OUT
long	path	Path of the opened bus	IN
short	sFlg	Access flag (0: Normal access, 1: High-speed access, Others: Reserved (normal access))	IN
unsigned short	usYno	Start output number (Y)	IN
unsigned short	usSize	Write size in words	IN
unsigned short	*pusDataBuf	Write data	IN
unsigned short	usBufSize	Dummy (fixed to 0)	IN

- 7-segment LED control: QBF_Control7SegLED function



Format

ret = QBF_Control7SegLED(path, mode, data);

Type	Name	Description	IN/OUT
short	ret;	Return value	OUT
long	path;	Opened bus path	IN
long	mode;	Mode (0: Manual mode, 1: Auto mode, Other: Same as 0)	IN
char	*data;	LED data	IN

Description

- Displays the values specified in LED data on the 7-segment LED, in accordance with the method specified in mode.

[When Mode 0: Manual mode]

Displays the specified data in the format below, where:
data[0] 7-segment LED: data of ones place, and
data[1] 7-segment LED: data of tens place.

Data format of each place:	Bit	b7	b6	b5	b4	b3	b2	b1	b0
	Segment	DP	g	f	e	d	c	b	a

*Segment: Refer to the drawing on the left.
*Bit value
0: LED OFF (off)
1: LED ON (on)

Reference

The following data types are available for C language and C++ language programming used on a C Controller module.

Data type	Bit width	Designation
byte	8	Unsigned integer
char	8	Character string
unsigned char	8	Unsigned character string
short	16	Signed short integer
unsigned short	16	Unsigned short integer
int	32	Signed (long) integer
long	32	
unsigned long	32	Unsigned (long) integer
float	32	Single-precision real number
double	64	Double-precision real number
void	-	-

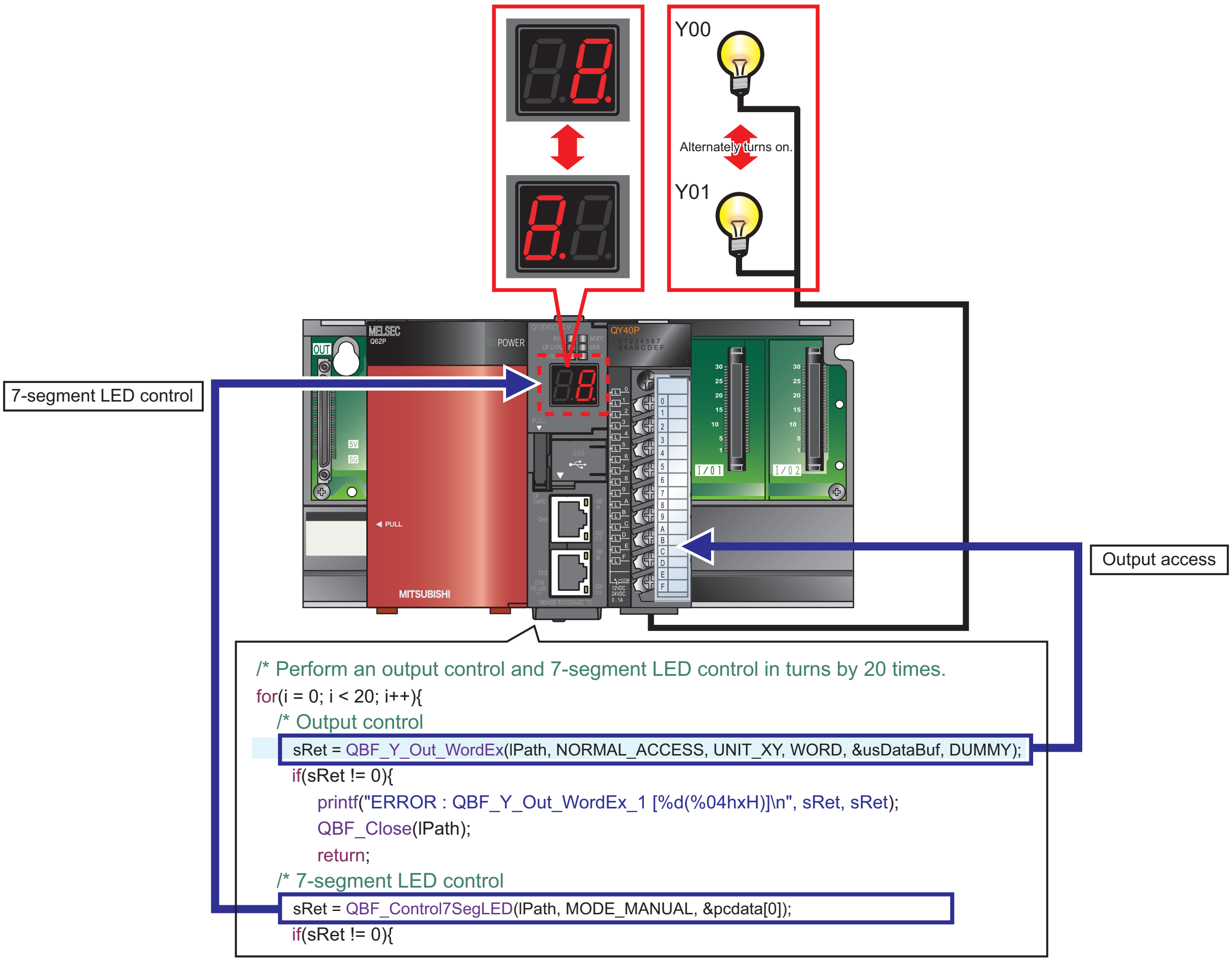
<5> Programming

Create a program in which lamps connected to an output module and the 7-segment LED on the front of the C Controller module flash.

1. Program example and control description

Create a program that performs the following control.

When the C Controller module is set to RUN, output lamps Y00 and Y01 alternately turn on. Synchronizing with the on status of the output lamps, the tens place and ones place of the 7-segment LED alternately turn on.



```
/* Perform an output control and 7-segment LED control in turns by 20 times.
for(i = 0; i < 20; i++){
   /* Output control
     sRet = QBF_Y_Out_WordEx(lPath, NORMAL_ACCESS, UNIT_XY, WORD, &usDataBuf, DUMMY);
     if(sRet != 0){
         printf("ERROR : QBF_Y_Out_WordEx_1 [%d(%04hxH)]\n", sRet, sRet);
         QBF_Close(lPath);
         return;
   /* 7-segment LED control
     sRet = QBF_Control7SegLED(lPath, MODE_MANUAL, &pcdata[0]);
     if(sRet != 0){
```

2. Source code

The following describes source codes.

```
/**************************************************************************************************************/
/* Function header                                                                                             */
/**************************************************************************************************************/
#include <vxworks.h>            /* VxWorks function header                                                     */
#include <taskLib.h>            /* VxWorks function header                                                     */
#include <stdio.h>              /* Standard function header                                                    */
#include "QbfFunc.h"            /* Bus interface function header                                               */
```

Declare the file that defined a function list for use of the library function.

```
/**************************************************************************************************************/
/* Definition                                                                                                  */
/**************************************************************************************************************/
/* For debugging                                                                                               */
#define UNIT_XY                 0x0000      /* Start I/O number of the module                                  */
#define QY_LED                  0x5555      /* Initial output value of Y signal (even bit: on)                 */
#define SEG_LED1                0xFF        /* Initial output value of 7-segment LED (ones place)              */
#define SEG_LED2                0x00        /* Initial output value of 7-segment LED (tens place)              */

/**************************************************************************************************************/
/* For QBF function                                                                                            */
#define CPU_TYPE                2           /* CPU identification flag (CCPU: 2)                               */
#define WORD                    1           /* 1-word specification                                            */
#define NORMAL_ACCESS           0           /* General access specification                                    */
#define DUMMY                   0           /* Dummy                                                           */
#define MODE_MANUAL             0           /* 7-segment LED control mode                                      */
```

Define values used for the control.

```
/**************************************************************************************************************/
/* Process outputs from Y signal and control the 7-segment LED.                                                */
/**************************************************************************************************************/
void Q12_SampleTask()
{
   /* Declare local variables.                                                                                 */
   short            sRet;                   /* Return value of the QBF function                               */
   long             lPath;                  /* Path of a bus                                                  */
   unsigned short   usDataBuf;              /* Y signal (in units of words)                                   */
   unsigned short   usEmptyDataBuf;         /* For reset of Y signal                                          */
   char             pcdata[2];              /* 7-segment LED on value                                         */
   short            i;                      /* For loop                                                       */

   /* Open the bus.                                                                                            */
   sRet = QBF_Open(CPU_TYPE, &lPath);
   if(sRet != 0){
      printf("ERROR : QBF_Open [%d(%04hxH)]\n", sRet, sRet);
      return;
   }
```

Enable the bus interface function at the start of the program.

```
   /* Set the output signal (Y) value (turn on the even bit).                                                  */
   usDataBuf = QY_LED;

   /* Set the output value of the 7-segment LED (only the ones places are all lit).                            */
   pcdata[0] = SEG_LED1;
   pcdata[1] = SEG_LED2;

   /* Perform an output control and 7-segment LED control in turns by 20 times.                               */
   for(i = 0; i < 20; i++){
      /* Output control                                                                                        */
      sRet = QBF_Y_Out_WordEx(lPath, NORMAL_ACCESS, UNIT_XY, WORD, &usDataBuf, DUMMY);
      if(sRet != 0){
         printf("ERROR : QBF_Y_Out_WordEx_1 [%d(%04hxH)]\n", sRet, sRet);
         QBF_Close(lPath);
         return;
      }
```

Control the output module using the bus interface function.

```
      /* 7-segment LED control                                                                                 */
      sRet = QBF_Control7SegLED(lPath, MODE_MANUAL, &pcdata[0]);
      if(sRet != 0){
         printf("ERROR : QBF_Control7SegLED_1 [%d(%04hxH)]\n", sRet, sRet);
         QBF_Close(lPath);
         return;
      }
```

Control the 7-segment LED using the bus interface function.

```
      /* Invert the output signal (Y) value (turn on the bits in order of odd bit -> even bit ->...).          */
      usDataBuf = ~usDataBuf;
```

```
        /* Invert the output values of the 7-segment LED (turn on in order of all ones places -> all tens places...).  */
        pcdata[0] = ~pcdata[0];
        pcdata[1] = ~pcdata[1];

        /* Wait.                                                                                                      */
        taskDelay(40);
    }
```

```
    /* Reset the Y signal.                                                                                            */
    usEmptyDataBuf = 0x00;
    sRet = QBF_Y_Out_WordEx(lPath, NORMAL_ACCESS, UNIT_XY, WORD,
         &usEmptyDataBuf, DUMMY);
    if(sRet != 0){
        printf("ERROR : QBF_Y_Out_WordEx_2 [%d(%04hxH)]\n", sRet, sRet);
        QBF_Close(lPath);
        return;
    }

    /* Reset the 7-segment LED.                                                                                       */
    pcdata[0] = 0x00;
    pcdata[1] = 0x00;
    sRet = QBF_Control7SegLED(lPath, MODE_MANUAL, &pcdata[0]);
    if(sRet != 0){
        printf("ERROR : QBF_Control7SegLED_2 [%d(%04hxH)]\n", sRet, sRet);
        QBF_Close(lPath);
        return;
    }
```

Turn off both outputs from the output module and the 7-segment LED.

```
    /* Close the bus.                                                                                                 */
    QBF_Close(lPath);
    return;
}
```

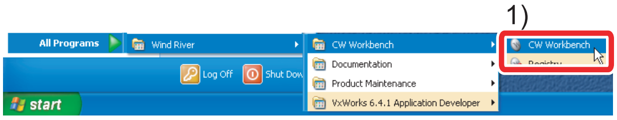
Disable the bus interface function at the end of the program.

1) Creating a project

1. Starting CW Workbench

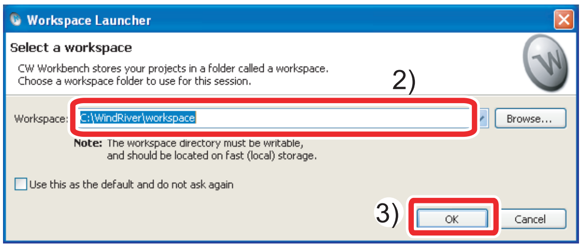
Operating procedure

1) Select [start]→[All Programs]→[Wind River]→[CW Workbench]→[CW Workbench].



2) Enter the storage location of the workspace.
In this procedure, enter "C:\WindRiver\workspace".

3) Click the OK button.


Workspace Launcher

Select a workspace

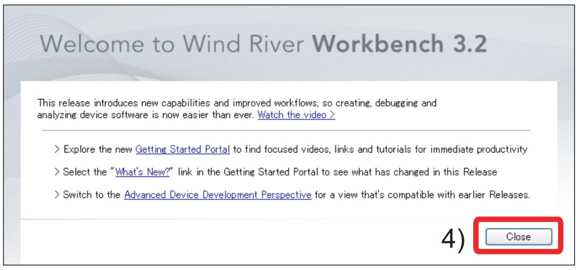
CW Workbench stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: C:\WindRiver\workspace

Note: The workspace directory must be writable,
and should be located on fast (local) storage.

□ Use this as the default and do not ask again

4) Click the Close button.


Welcome to Wind River **Workbench 3.2**

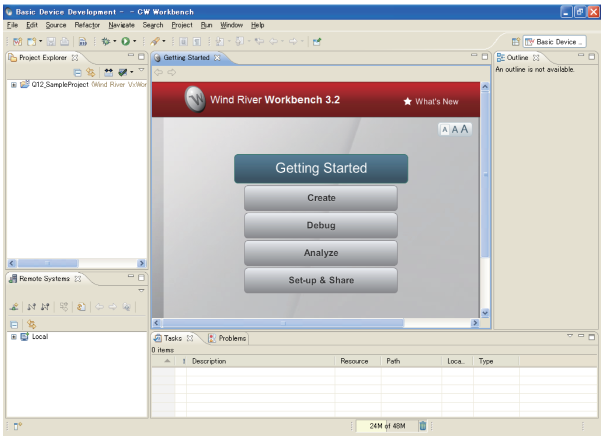
This release introduces new capabilities and improved workflows, so creating, debugging and analyzing device software is now easier than ever. Watch the video >

> Explore the new Getting Started Portal to find focused videos, links and tutorials for immediate productivity

> Select the "What's New?" link in the Getting Started Portal to see what has changed in this Release

> Switch to the Advanced Device Development Perspective for a view that's compatible with earlier Releases.

The main window of CW Workbench appears.



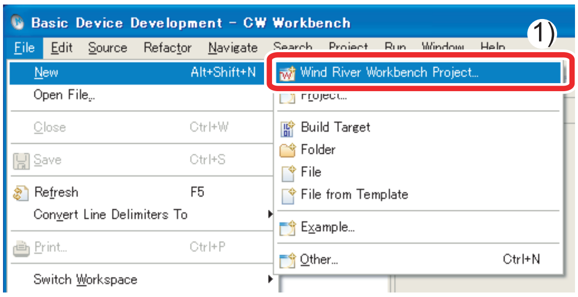
Reference

- The default window sizes and icon positions on CW Workbench depends on a personal computer. If a window size differs from that shown in this guide, adjust the size.
- To default an enlarged/deleted window, select [Window]→ [New Window].

2. Creating a project

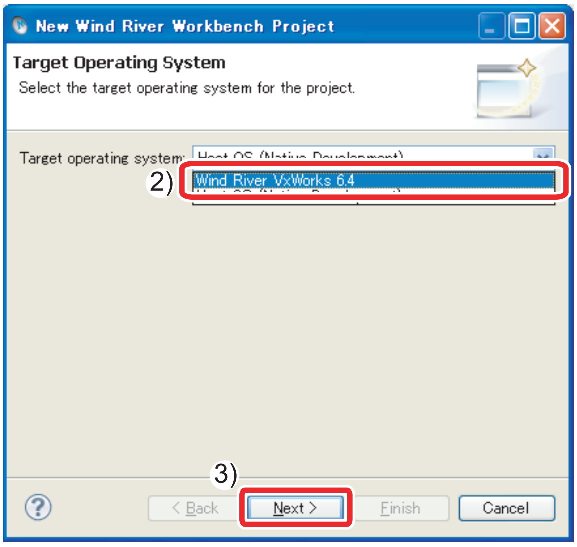
Operating procedure

1) Select [File]→[New]→[Wind River Workbench Project...].



2) Select "Wind River VxWorks6.4".

3) Click the Next > button.


New Wind River Workbench Project

Target Operating System

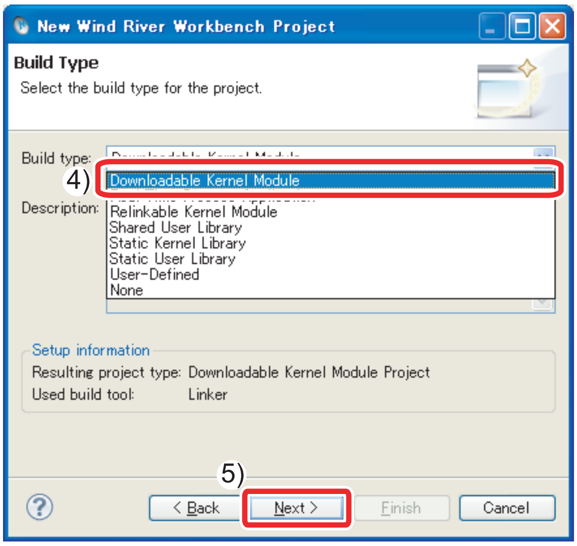
Select the target operating system for the project.

Target operating system: Host OS (Native Development)

Wind River VxWorks 6.4

4) Select "Downloadable Kernel Module".

5) Click the Next > button.


New Wind River Workbench Project

Build Type

Select the build type for the project.

Build type: Downloadable Kernel Module

Description:

Downloadable Kernel Module
Relinkable Kernel Module
Shared User Library
Static Kernel Library
Static User Library
User-Defined
None

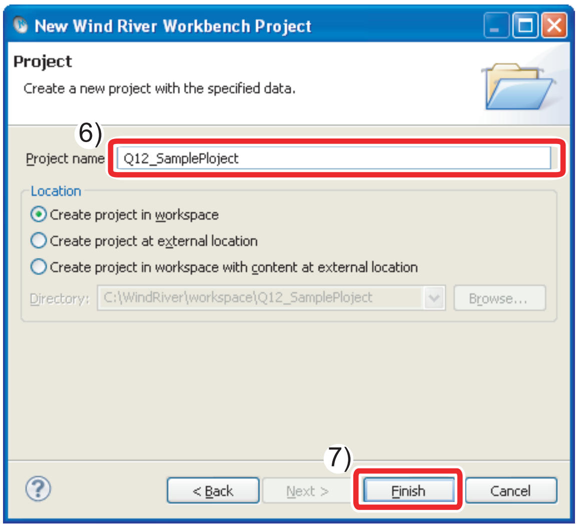
Setup information

Resulting project type: Downloadable Kernel Module Project

Used build tool: Linker

6) Enter a project name.
In this procedure, enter "Q12_SampleProject".

7) Click the Finish button.


New Wind River Workbench Project

Project

Create a new project with the specified data.

Project name Q12_SamplePloject

Location

● Create project in workspace
○ Create project at external location
○ Create project in workspace with content at external location

Directory: C:\WindRiver\workspace\Q12_SamplePloject Browse...

The project has been created.

3. Creating a project property

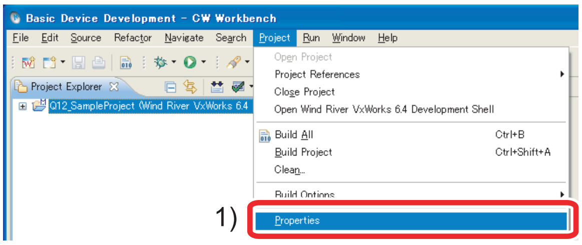
Configure settings to convert (build) the created project into a module that can be executed on a C Controller module.

Terminology

Build: An operation that compiles source codes according to a processor and links the code to the include file.

(1) Setting the processor

1) Select the created project in the "Project Explorer" window, and click [Project]→[Properties].



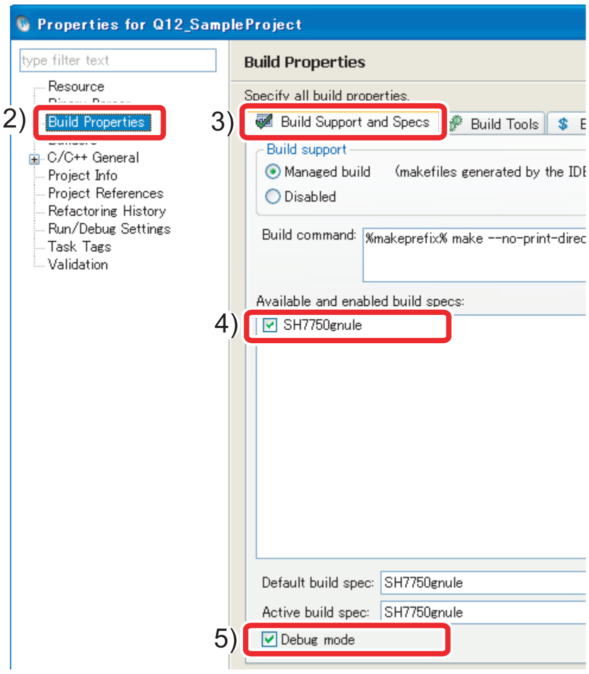
Basic Device Development - CW Workbench
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
Q12_SampleProject (Wind River VxWorks 6.4
Open Project
Project References
Close Project
Open Wind River VxWorks 6.4 Development Shell
Build All Ctrl+B
Build Project Ctrl+Shift+A
Clean...
Build Options
1)
Properties

2) Select "Build Properties" from the tree view to the left in the window.

3) Click the "Build Support and Specs" tab.

4) Select the "SH7750gnule" check box only in "Available and enabled build specs:".

5) Select the "Debug mode" check box.



Properties for Q12_SampleProject
type filter text
Resource
2)
Build Properties
C/C++ General
Project Info
Project References
Refactoring History
Run/Debug Settings
Task Tags
Validation
Build Properties
Specify all build properties.
3)
Build Support and Specs
Build Tools
Build support
● Managed build (makefiles generated by the IDE
○ Disabled
Build command: %makeprefix% make --no-print-direc
Available and enabled build specs:
4)
☑ SH7750gnule
Default build spec: SH7750gnule
Active build spec: SH7750gnule
5)
☑ Debug mode

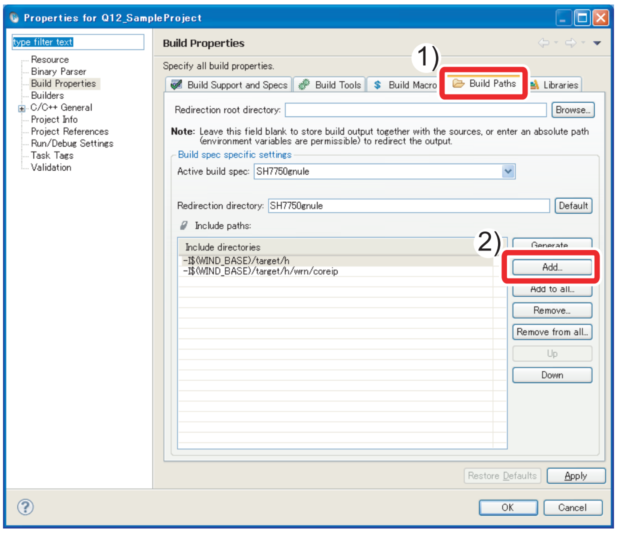
Point

Clear the "Debug mode" check box for the actual system operation.

(2) Setting a include file

1) Click the "Build Paths" tab.

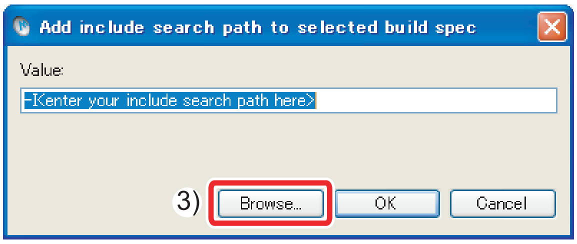
2) Click the Add... button.



Note: Leave this field blank to store build output together with the sources, or enter an absolute path (environment variables are permissible) to redirect the output.

-I$(WIND_BASE)/target/h
-I$(WIND_BASE)/target/h/wrn/coreip

3) Click the Browse... button.

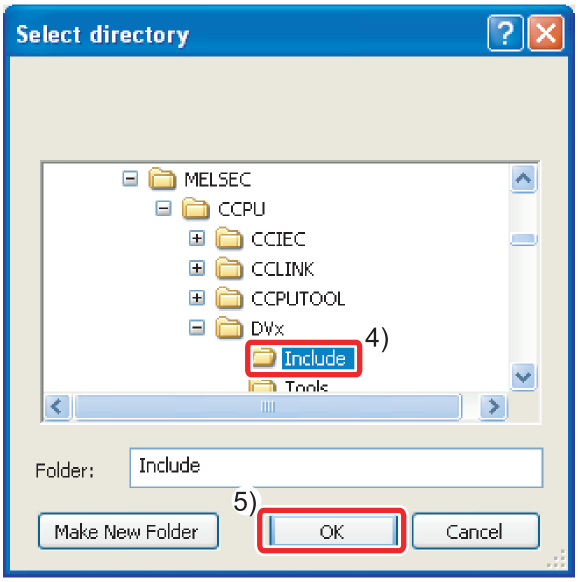


Add include search path to selected build spec
Value:
-I<enter your include search path here>
3)
Browse...
OK
Cancel

4) Select the include folder dedicated for the C Controller module in the "Select directory" window.

In this procedure, the folder is the one when SW□PVC-CCPU has been installed on "C:\MELSEC".

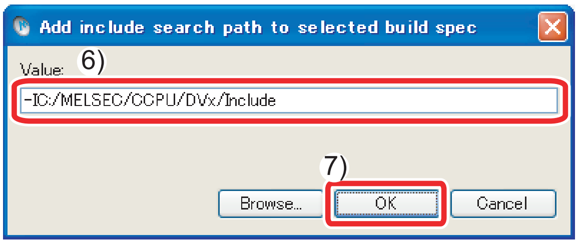
5) Click the OK button.



Folder: Include

6) Check that the folder specified in the "Select directory" window has been selected.

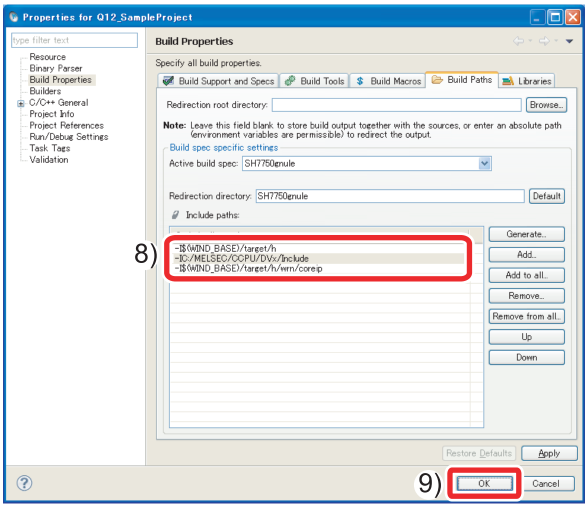
7) Click the OK button.



Value: -IC:/MELSEC/CCPU/DVx/Include

8) Check that the added include path is displayed in the "Include paths:" area.

9) Click the OK button.



Specify all build properties.

Redirection root directory:

Note: Leave this field blank to store build output together with the sources, or enter an absolute path (environment variables are permissible) to redirect the output.

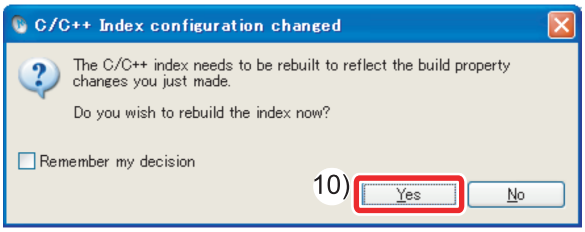
Active build spec: SH7750gnule

Redirection directory: SH7750gnule

Include paths:

```
-I$(WIND_BASE)/target/h
-IC:/MELSEC/CCPU/DVx/Include
-I$(WIND_BASE)/target/h/wrn/coreip
```

10) If the following message appears after clicking the OK button, click the Yes button.



The C/C++ index needs to be rebuilt to reflect the build property changes you just made.

Do you wish to rebuild the index now?

□ Remember my decision

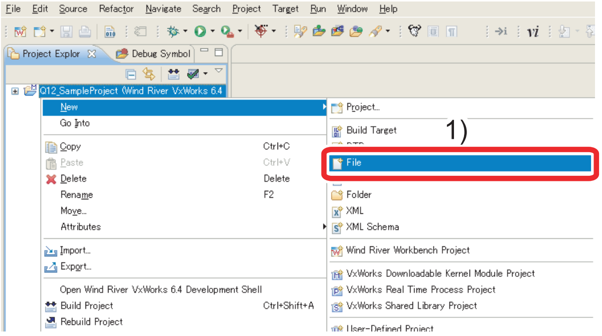
The project property has been set.

2) Creating a user program

Create a user program that controls a C Controller system.

Operating procedure

1) Right-click the created project in the "Project Explorer" window, and click [New]→[File].

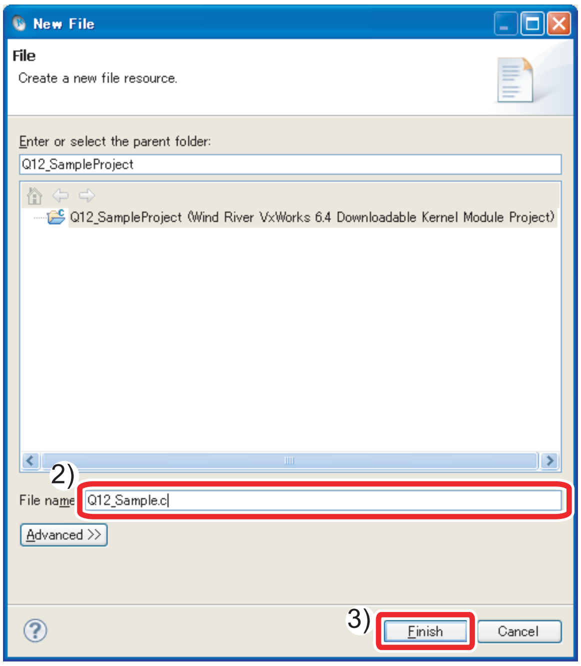

File Edit Source Refactor Navigate Search Project Target Run Window Help
Project Explor
Debug Symbol
Q12_SampleProject (Wind River VxWorks 6.4
New
Go Into
Copy Ctrl+C
Paste Ctrl+V
Delete Delete
Rename F2
Move...
Attributes
Import...
Export...
Open Wind River VxWorks 6.4 Development Shell
Build Project Ctrl+Shift+A
Rebuild Project
Project...
Build Target
File
Folder
XML
XML Schema
Wind River Workbench Project
VxWorks Downloadable Kernel Module Project
VxWorks Real Time Process Project
VxWorks Shared Library Project
User-Defined Project
1)

2) Enter a source file name to be created in "File name:".
Enter "Q12_Sample.c" in this procedure.

Point

Enter a file name with extension.
Do not use two-byte characters for a file name. If used, a compilation error occurs in compilation.

3) Click the Finish button.


New File
File
Create a new file resource.
Enter or select the parent folder:
Q12_SampleProject (Wind River VxWorks 6.4 Downloadable Kernel Module Project)
2)
File name
Advanced >>
3)
Finish
Cancel

Enter or select the parent folder: Q12_SampleProject

File name: Q12_Sample.c

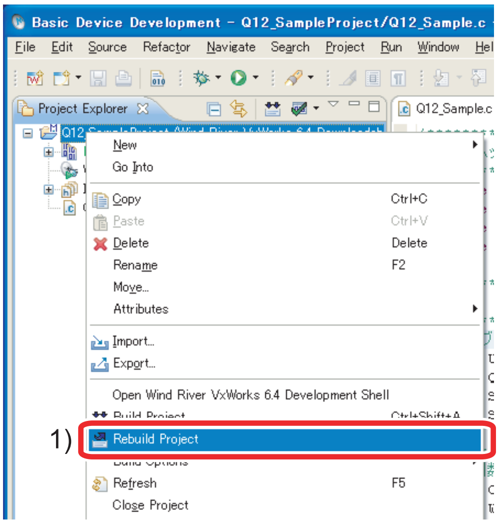
4) Describe "Source code"(P.30) to access the output module and to control the 7-segment LED in the "Editor" window.

3) Generating an execution module from the user program

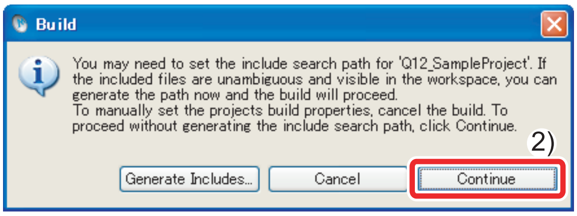
Convert (Build) the created program into a module that can be executed on a C Controller module.

Operating procedure

1) Right-click the created project in the "Project Explorer" window, and click [Rebuild Project].


Basic Device Development - Q12_SampleProject/Q12_Sample.c
File Edit Source Refactor Navigate Search Project Run Window Hel
Project Explorer
Q12_Sample.c
New
Go Into
Copy Ctrl+C
Paste Ctrl+V
Delete Delete
Rename F2
Move...
Attributes
Import...
Export...
Open Wind River VxWorks 6.4 Development Shell
Build Project Ctrl+Shift+A
1)
Rebuild Project
Build Options
Refresh F5
Close Project

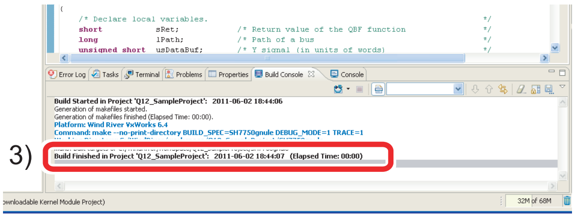
2) If the message shown below appears, click the Continue button.



You may need to set the include search path for 'Q12_SampleProject'. If the included files are unambiguous and visible in the workspace, you can generate the path now and the build will proceed.
To manually set the projects build properties, cancel the build. To proceed without generating the include search path, click Continue.

The project starts to be built. The progress is displayed in the "Build Console" window.

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



```
{
    /* Declare local variables.                                          */
    short           sRet;           /* Return value of the QBF function  */
    long            lPath;          /* Path of a bus                     */
    unsigned short  usDataBuf;      /* Y signal (in units of words)      */
```

```
Build Started in Project 'Q12_SampleProject':  2011-06-02 18:44:06
Generation of makefiles started.
Generation of makefiles finished (Elapsed Time: 00:00).
Platform: Wind River VxWorks 6.4
Command: make --no-print-directory BUILD_SPEC=SH7750gnule DEBUG_MODE=1 TRACE=1
Build Finished in Project 'Q12_SampleProject':  2011-06-02 18:44:07  (Elapsed Time: 00:00)
```

"Build Finished..." indicates the completion of creation and build of the user program.

Point

If "Build Finished..." is not displayed and an error occurs, check the error and correct the program.
After the correction, perform the operation again from "3) Generating an execution module from the user program"(P.37).

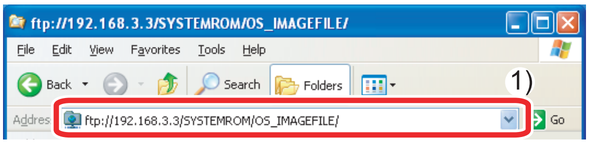
4) Connecting a C Controller module to CW Workbench

Connect a C Controller module to CW Workbench to perform debugging using CW Workbench.

Operating procedure

1) To acquire a VxWorks image file from the C Controller module, start Explorer and enter the following address in the address area.

ftp://192.168.3.3/SYSTEMROM/OS_IMAGEFILE/


ftp://192.168.3.3/SYSTEMROM/OS_IMAGEFILE/
File Edit View Favorites Tools Help
Back
Search
Folders
1)
Address
Go

Address: ftp://192.168.3.3/SYSTEMROM/OS_IMAGEFILE/

The "Log On As" window appears.

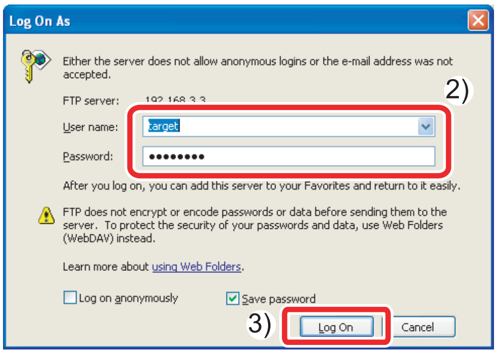
Point

To communicate between the C Controller module and the personal computer, specify the same VxWorks image file for both.

2) Enter the following user name and password in the "Log On As" window.

- User name : target
- Password : password

3) Click the Log On button.



Either the server does not allow anonymous logins or the e-mail address was not accepted.

FTP server: 192.168.3.3

User name: target

Password: ●●●●●●●●●

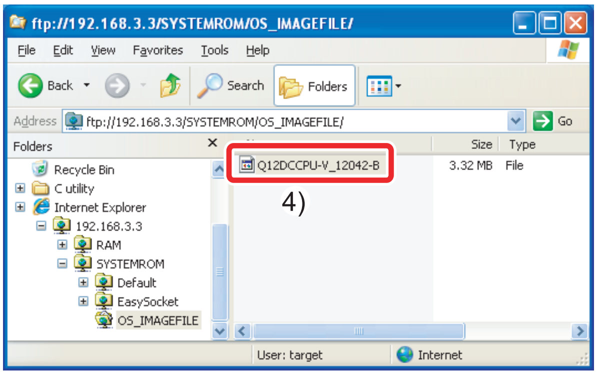
After you log on, you can add this server to your Favorites and return to it easily.

FTP does not encrypt or encode passwords or data before sending them to the server. To protect the security of your passwords and data, use Web Folders (WebDAV) instead.

Learn more about using Web Folders.

□ Log on anonymously ☑ Save password

4) Copy the VxWorks image file stored on the C Controller module to "C:\MELSEC\CCPU\DVx\Tools".



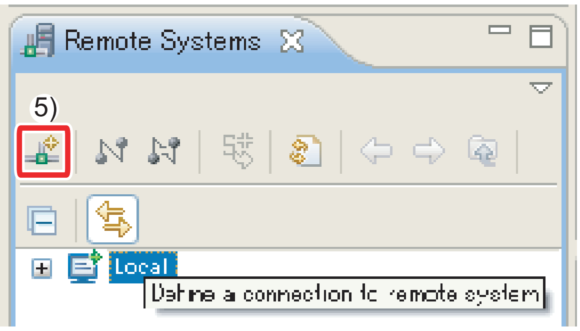
Address: ftp://192.168.3.3/SYSTEMROM/OS_IMAGEFILE/

Name	Size	Type
Q12DCCPU-V_12042-B	3.32 MB	File

Point

The "C:\MELSEC\CCPU\DVx\Tools" folder is created when SW☐PVC-CCPU has been installed on "C:\MELSEC".

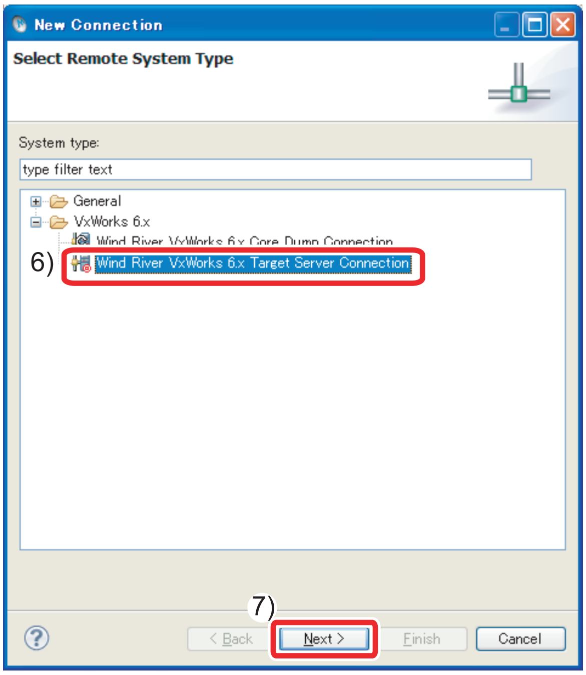
5) Click in the "Remote Systems" window.



The "New Connection" window appears.

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

7) Click the Next > button.

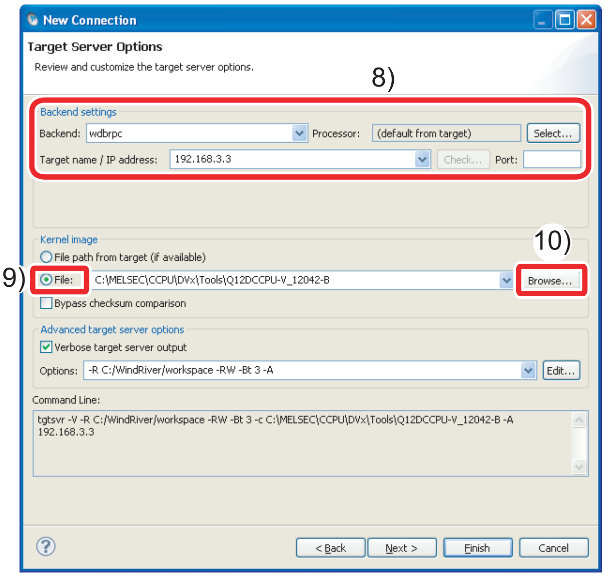

New Connection
Select Remote System Type
System type:
type filter text
General
VxWorks 6.x
Wind River VxWorks 6.x Core Dump Connection
6)
Wind River VxWorks 6.x Target Server Connection
7)
< Back
Next >
Finish
Cancel

8) Set the following items in "Backend settings".

- Backend : wdbrpc
- Processor : SH7780 (Click the Select... button and select the processor.)
- IP address : 192.168.3.3 (default)
- Port : Blank

9) Select the "File" radio button in "Kernel image".

10) Click the Browse... button.



New Connection

Target Server Options

Review and customize the target server options.

Backend settings

Backend: wdbrpc Processor: (default from target) Select...

Target name / IP address: 192.168.3.3 Check... Port:

Kernel image

○ File path from target (if available)

● File: C:\MELSEC\CCPU\DVx\Tools\Q12DCCPU-V_12042-B Browse...

□ Bypass checksum comparison

Advanced target server options

☑ Verbose target server output

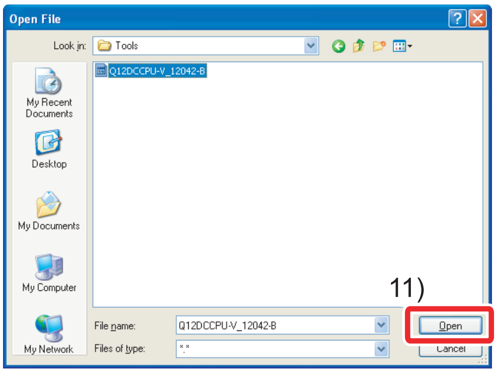
Options: -R C:/WindRiver/workspace -RW -Bt 3 -A Edit...

Command Line:

```
tgtsvr -V -R C:/WindRiver/workspace -RW -Bt 3 -c C:\MELSEC\CCPU\DVx\Tools\Q12DCCPU-V_12042-B -A
192.168.3.3
```

The "Open File" window appears.

11) Select the VxWorks image file copied in the step 4) (C:\MELSEC\CCPU\DVx\Tools) from the tree view, and click the Open button.



Open File

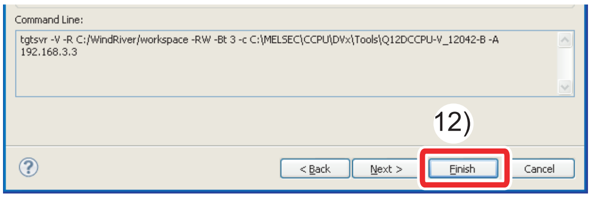
Look in: Tools

Q12DCCPU-V_12042-B

File name: Q12DCCPU-V_12042-B

Files of type: *.*

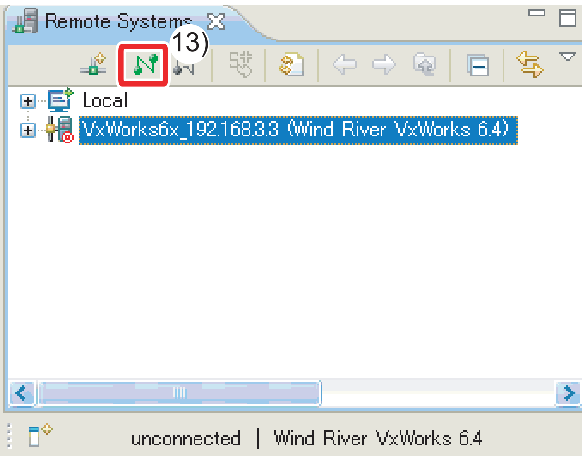
12) Click the Finish button.



Command Line:

```
tgtsvr -V -R C:/WindRiver/workspace -RW -Bt 3 -c C:\MELSEC\CCPU\DVx\Tools\Q12DCCPU-V_12042-B -A
192.168.3.3
```

13) Select the target server added in the "Remote Systems" window, and click .



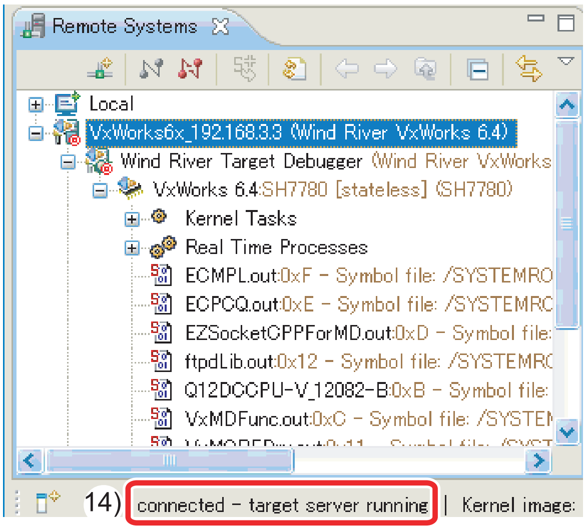
Remote Systems

Local

VxWorks6x_192.168.3.3 (Wind River VxWorks 6.4)

unconnected | Wind River VxWorks 6.4

14) After is clicked, the connection is completed when "connected - target server running" is displayed at the bottom of the "Remote Systems" window.



Remote Systems

Local

VxWorks6x_192.168.3.3 (Wind River VxWorks 6.4)

Wind River Target Debugger (Wind River VxWorks

VxWorks 6.4:SH7780 [stateless] (SH7780)

Kernel Tasks

Real Time Processes

ECMPL.out:0xF - Symbol file: /SYSTEMRO

ECPCQ.out:0xE - Symbol file: /SYSTEMRC

EZSocketCPPForMD.out:0xD - Symbol file:

ftpdLib.out:0x12 - Symbol file: /SYSTEMRC

Q12DCCPU-V_12082-B:0xB - Symbol file:

VxMDFunc.out:0xC - Symbol file: /SYSTEM

connected - target server running | Kernel image:

Point

If "connected - target server running" is not displayed, check that the C Controller module is normally powered on, and perform the operation again from "4) Connecting a C Controller module to CW Workbench"(P.38).

5) Debugging the user program

Check that the created program correctly operates.

1. Downloading the user program on the C Controller module

To debug the user program, download the execution module on the memory in the C Controller module.

Downloading a user program allows users to execute the program without a script file.

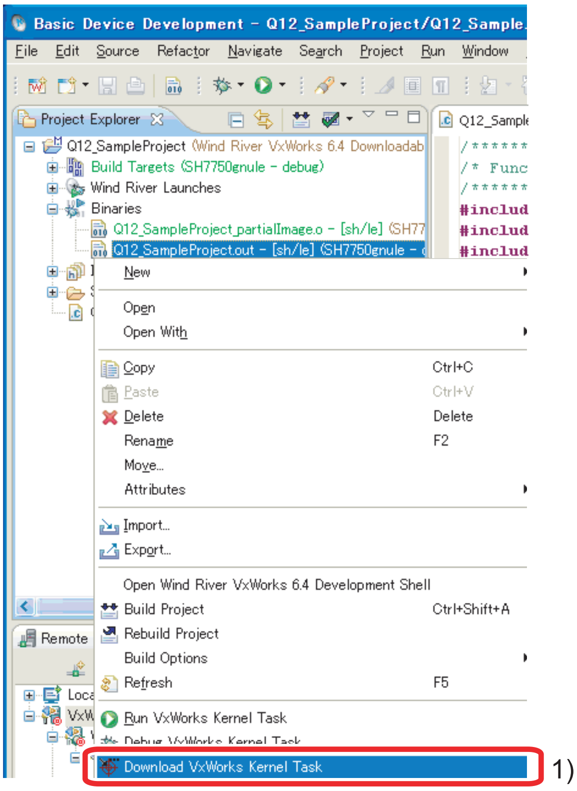
Terminology

Script file: A file that describes the download location and the startup procedure of the user program that starts at the start of a C Controller module

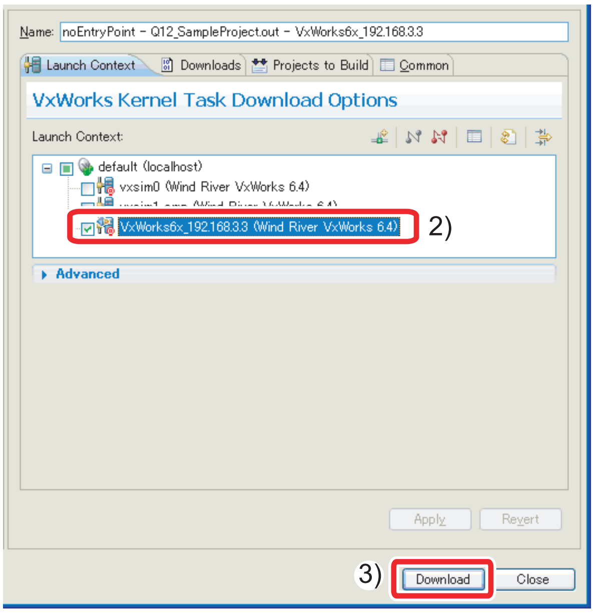
1) Right-click the created module file "Q12_SampleProject.out" in the "Project Explorer" window, and click [Download VxWorks Kernel Task].

2) Select the "VxWorks6x_192.168.3.3 (Wind River VxWorks 6.4)" check box only in "Launch Context:".

3) Click the Download button.

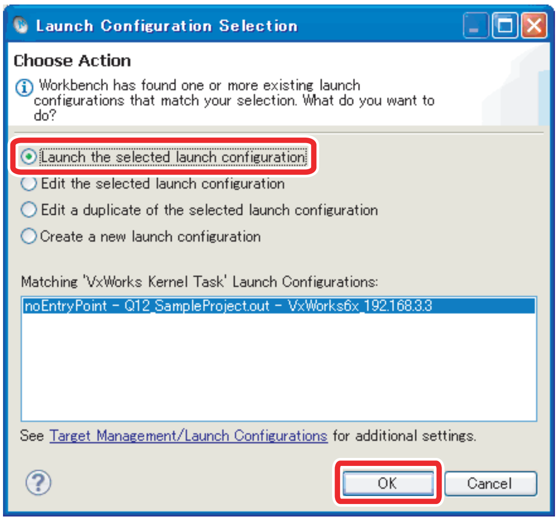

Basic Device Development - Q12_SampleProject/Q12_Sample
File Edit Source Refactor Navigate Search Project Run Window
Project Explorer
Q12_SampleProject (Wind River VxWorks 6.4 Downloadab
Build Targets (SH7750gnule - debug)
Wind River Launches
Binaries
Q12_SampleProject_partialImage.o - [sh/le] (SH77
Q12_SampleProject.out - [sh/le] (SH7750gnule -
Q12_Sampl
#includ
New
Open
Open With
Copy Ctrl+C
Paste Ctrl+V
Delete Delete
Rename F2
Move...
Attributes
Import...
Export...
Open Wind River VxWorks 6.4 Development Shell
Build Project Ctrl+Shift+A
Rebuild Project
Build Options
Refresh F5
Run VxWorks Kernel Task
Debug VxWorks Kernel Task
Download VxWorks Kernel Task
Remote
1)

The "Download Configurations" window appears.


Name: noEntryPoint - Q12_SampleProject.out - VxWorks6x_192.168.3.3

Point

The "Launch Configuration Selection" window appears on and after the second operation of the step 2).
Select the "Launch the selected launch configuration" radio button and click the OK button.



Launch Configuration Selection

Choose Action

Workbench has found one or more existing launch configurations that match your selection. What do you want to do?

● Launch the selected launch configuration
○ Edit the selected launch configuration
○ Edit a duplicate of the selected launch configuration
○ Create a new launch configuration

Matching 'VxWorks Kernel Task' Launch Configurations:

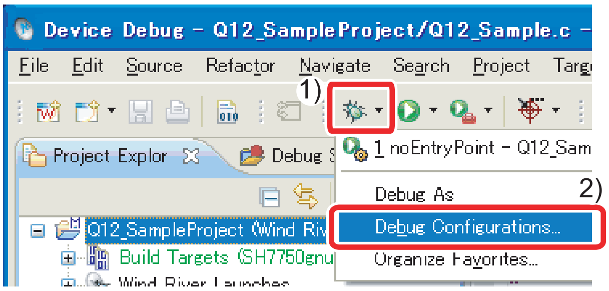
noEntryPoint - Q12_SampleProject.out - VxWorks6x_192.168.3.3

See Target Management/Launch Configurations for additional settings.

2. Debugging the user program

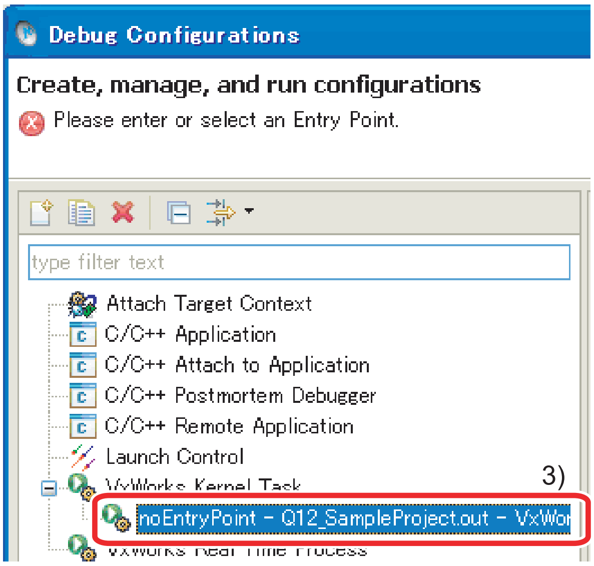
1) Select the created project in the "Project Explorer" window, and click ▼ on the right side of the debug icon on the toolbar.

2) Click [Debug Configurations...].



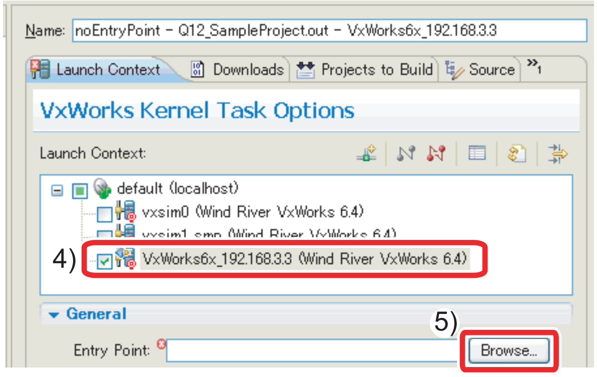
The "Debug Configurations" window appears.

3) Click the downloaded module "Q12_SampleProject.out" from "VxWorks Kernel Task".


Debug Configurations
Create, manage, and run configurations
Please enter or select an Entry Point.
type filter text
Attach Target Context
C/C++ Application
C/C++ Attach to Application
C/C++ Postmortem Debugger
C/C++ Remote Application
Launch Control
VxWorks Kernel Task
3)
noEntryPoint - Q12_SampleProject.out - VxWor
VxWorks Real Time Process

4) Select the target server indicating connection to the C Controller module.

5) Click the Browse... button.



Name: noEntryPoint - Q12_SampleProject.out - VxWorks6x_192.168.3.3

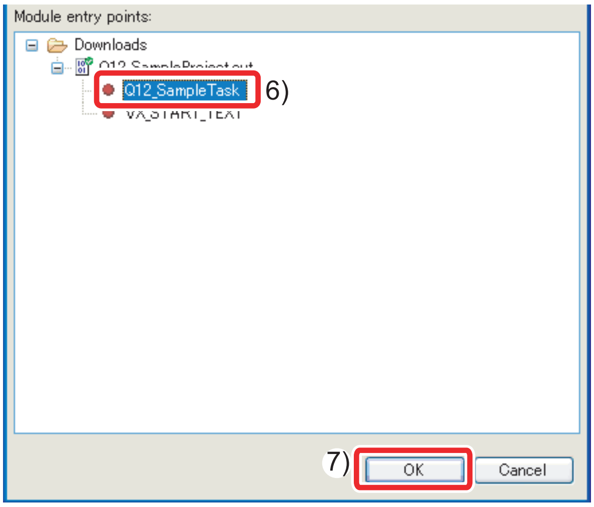
☑ VxWorks6x_192.168.3.3 (Wind River VxWorks 6.4)

Entry Point:

The "Entry Points" window appears.

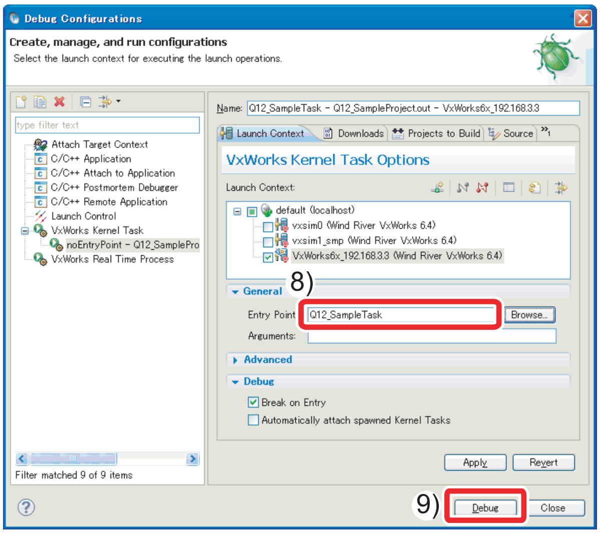
6) Select the function that starts debugging (Q12_SampleTask).

7) Click the OK button.



8) Check that the function name selected in the step 6) has been selected in "Entry Point:".

9) Click the Debug button.


Debug Configurations
Create, manage, and run configurations
Select the launch context for executing the launch operations.
type filter text
Attach Target Context
C/C++ Application
C/C++ Attach to Application
C/C++ Postmortem Debugger
C/C++ Remote Application
Launch Control
VxWorks Kernel Task
noEntryPoint - Q12_SamplePro
VxWorks Real Time Process
Filter matched 9 of 9 items
Launch Context
Downloads
Projects to Build
Source
VxWorks Kernel Task Options
Launch Context:
default (localhost)
vxsim0 (Wind River VxWorks 6.4)
vxsim1_smp (Wind River VxWorks 6.4)
General
8)
Browse...
Advanced
Debug
Apply
Revert
9)
Debug
Close

Name: Q12_SampleTask - Q12_SampleProject.out - VxWorks6x_192.168.3.3

☑ VxWorks6x_192.168.3.3 (Wind River VxWorks 6.4)

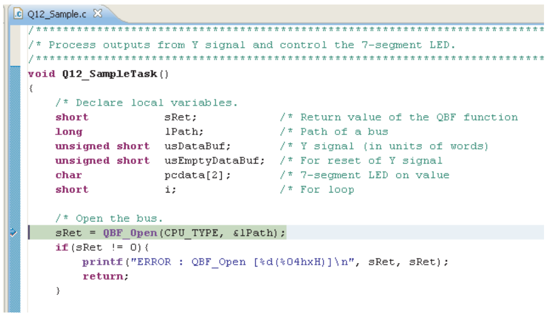
Entry Point: Q12_SampleTask

Arguments:

☑ Break on Entry

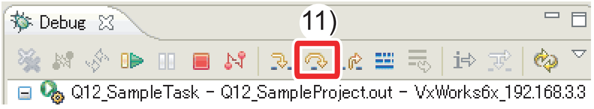
□ Automatically attach spawned Kernel Tasks

10) Debugging starts. Program execution stops at the start of the function specified in "Entry Point:".

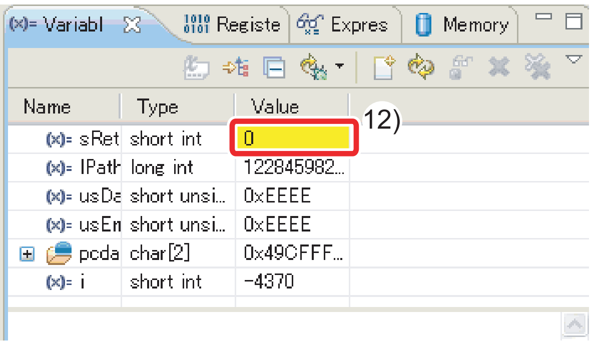


```
/**********************************************************************
/* Process outputs from Y signal and control the 7-segment LED.
/**********************************************************************
void Q12_SampleTask()
{
    /* Declare local variables.
    short           sRet;           /* Return value of the QBF function
    long            lPath;          /* Path of a bus
    unsigned short  usDataBuf;      /* Y signal (in units of words)
    unsigned short  usEmptyDataBuf; /* For reset of Y signal
    char            pcdata[2];      /* 7-segment LED on value
    short           i;              /* For loop

    /* Open the bus.
    sRet = QBF_Open(CPU_TYPE, &lPath);
    if(sRet != 0){
        printf("ERROR : QBF_Open [%d(%04hxH)]\n", sRet, sRet);
        return;
    }
```

11) Click in the "Debug" window to perform debugging by one step.

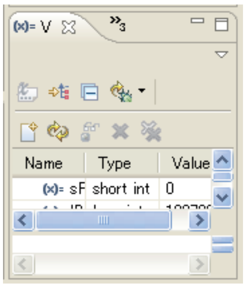

Debug
11)
Q12_SampleTask - Q12_SampleProject.out - VxWorks6x_192.168.3.3

12) By clicking a tab on the bottom right of the "Variables" window*1. variable values can be checked and changed.
In this step, check that "sRet", return value of the "QBF_Open" function, is "0" (normal value).



Name	Type	Value
sRet	short int	0
lPath	long int	122845982...
usDa	short unsi...	0xEEEE
usEm	short unsi...	0xEEEE
pcda	char[2]	0x49CFFF...
i	short int	-4370

*1 Depending on a personal computer, the "Variables" window appears as shown below. Adjust the window size.



In the steps 11) and 12), debug the entire program.

Reference

If the return value of the bus interface function is other than "0", troubleshoot with reference to the following.

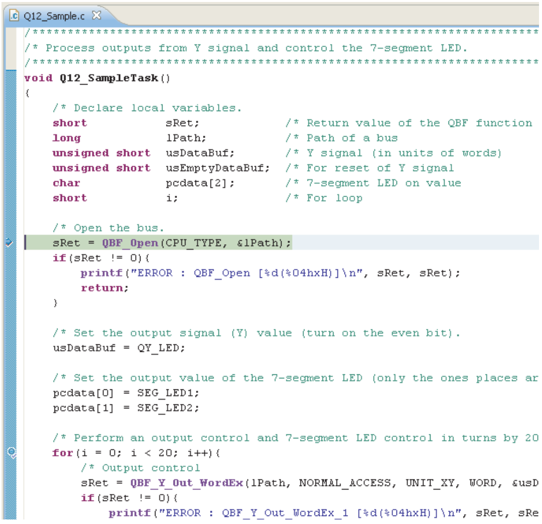
☞ Bus interface function help window in SW☐PVC-CCPU

☞ C Controller Module User's Manual (Hardware Design, Function Explanation): SH-080766ENG

<Debugging using breakpoint>

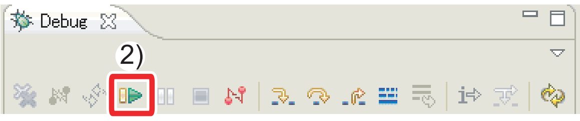
As well as debugging in units of one step described in the step 11) shown to the left, debugging using a breakpoint is available.

1) Double-click the left edge of a source file window and insert a breakpoint.

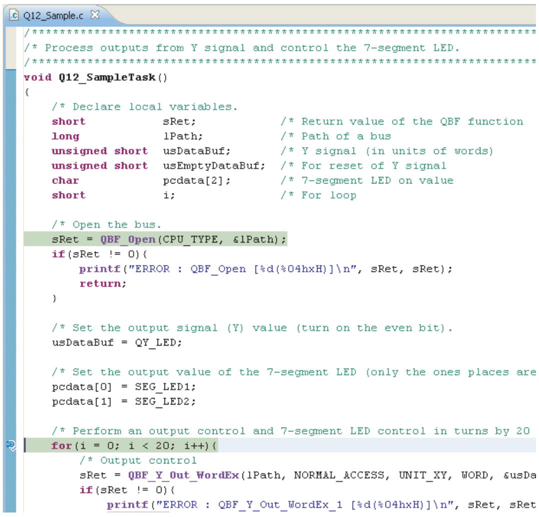


```
/***********************************************************************
/* Process outputs from Y signal and control the 7-segment LED.
/***********************************************************************
void Q12_SampleTask()
{
    /* Declare local variables.
    short           sRet;           /* Return value of the QBF function
    long            lPath;          /* Path of a bus
    unsigned short  usDataBuf;      /* Y signal (in units of words)
    unsigned short  usEmptyDataBuf; /* For reset of Y signal
    char            pcdata[2];      /* 7-segment LED on value
    short           i;              /* For loop

    /* Open the bus.
    sRet = QBF_Open(CPU_TYPE, &lPath);
    if(sRet != 0){
        printf("ERROR : QBF_Open [%d(%04hxH)]\n", sRet, sRet);
        return;
    }

    /* Set the output signal (Y) value (turn on the even bit).
    usDataBuf = QY_LED;

    /* Set the output value of the 7-segment LED (only the ones places ar
    pcdata[0] = SEG_LED1;
    pcdata[1] = SEG_LED2;

    /* Perform an output control and 7-segment LED control in turns by 20
    for(i = 0; i < 20; i++){
        /* Output control
        sRet = QBF_Y_Out_WordEx(lPath, NORMAL_ACCESS, UNIT_XY, WORD, &usD
        if(sRet != 0){
            printf("ERROR : QBF_Y_Out_WordEx_1 [%d(%04hxH)]\n", sRet, sRe
```

2) Click .



The program is executed at the position specified by the breakpoint.



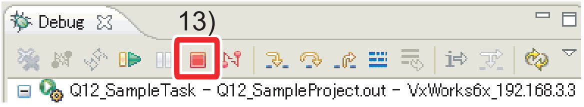
```
/***********************************************************************
/* Process outputs from Y signal and control the 7-segment LED.
/***********************************************************************
void Q12_SampleTask()
{
    /* Declare local variables.
    short           sRet;           /* Return value of the QBF function
    long            lPath;          /* Path of a bus
    unsigned short  usDataBuf;      /* Y signal (in units of words)
    unsigned short  usEmptyDataBuf; /* For reset of Y signal
    char            pcdata[2];      /* 7-segment LED on value
    short           i;              /* For loop

    /* Open the bus.
    sRet = QBF_Open(CPU_TYPE, &lPath);
    if(sRet != 0){
        printf("ERROR : QBF_Open [%d(%04hxH)]\n", sRet, sRet);
        return;
    }

    /* Set the output signal (Y) value (turn on the even bit).
    usDataBuf = QY_LED;

    /* Set the output value of the 7-segment LED (only the ones places are
    pcdata[0] = SEG_LED1;
    pcdata[1] = SEG_LED2;

    /* Perform an output control and 7-segment LED control in turns by 20
    for(i = 0; i < 20; i++){
        /* Output control
        sRet = QBF_Y_Out_WordEx(lPath, NORMAL_ACCESS, UNIT_XY, WORD, &usDa
        if(sRet != 0){
            printf("ERROR : QBF_Y_Out_WordEx_1 [%d(%04hxH)]\n", sRet, sRet
```

Reference

The descriptions of icons are as follows:

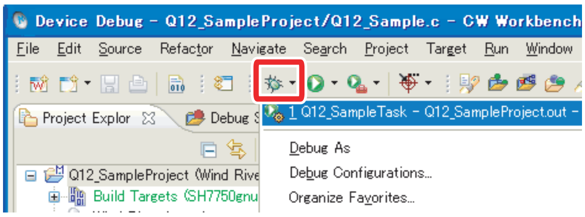
- : Step Into
  Steps into the called function and stops at the first line of the function.
- : Step Over
  Executes the current line of the function and then stops at the next line of the function.
- : Continues execution until the current function has returned to its caller.
- ▶ : Executes a program.
- ⏸ : Stops a program.
- 🟥 : Ends debugging.

13) Click 🟥 in the "Debug" window to terminate the debugging session.


Debug
13)
Q12_SampleTask - Q12_SampleProject.out - VxWorks6x_192.168.3.3

Point

To start debugging again, click ▼ on the right side of ⚙ on the toolbar and select the created debug configuration at the top of the pop-up menu.
The steps 1) to 10) can be skipped.


Device Debug - Q12_SampleProject/Q12_Sample.c - CW Workbench
File Edit Source Refactor Navigate Search Project Target Run Window
Project Explor
Debug
1 Q12_SampleTask - Q12_SampleProject.out -
Debug As
Debug Configurations...
Organize Favorites...
Q12_SampleProject (Wind Rive
Build Targets (SH7750gnu

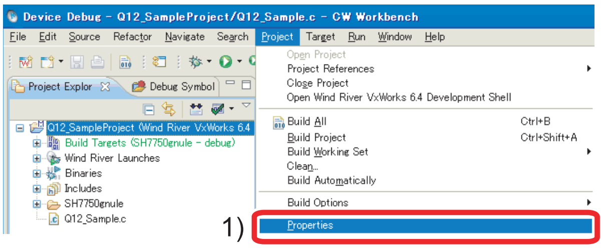
6) Registering an execution module

Build the created program for operation and store the created module on the C Controller module.

Operating procedure

1. Building the user program

1) Select the created project in the "Project Explorer" window, and click [Project]→[Properties].

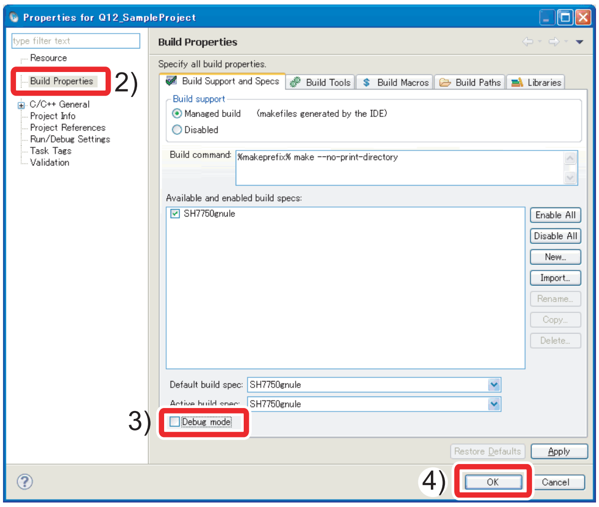


Device Debug - Q12_SampleProject/Q12_Sample.c - CW Workbench
File Edit Source Refactor Navigate Search Project Target Run Window Help
Project Explor
Debug Symbol
Q12_SampleProject (Wind River VxWorks 6.4
Build Targets (SH7750gnule - debug)
Wind River Launches
Binaries
Includes
SH7750gnule
Q12_Sample.c
Open Project
Project References
Close Project
Open Wind River VxWorks 6.4 Development Shell
Build All Ctrl+B
Build Project Ctrl+Shift+A
Build Working Set
Clean...
Build Automatically
Build Options
Properties
1)

2) Select "Build Properties" from the tree view to the left in the window.

3) Clear the "Debug mode" check box.

4) Click the OK button.



Specify all build properties.

Build support
● Managed build (makefiles generated by the IDE)
○ Disabled

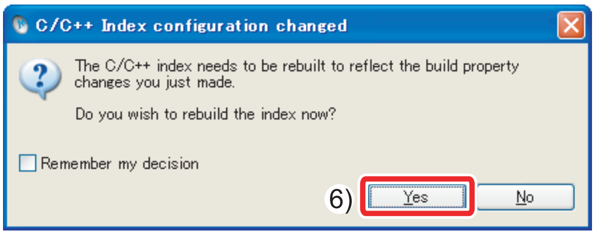
Build command: %makeprefix% make --no-print-directory

Available and enabled build specs:
☑ SH7750gnule

Default build spec: SH7750gnule
Active build spec: SH7750gnule
□ Debug mode

5) Build the program following the procedure shown in "3) Generating an execution module from the user program"(P.37).

6) If the following message appears, click the Yes button.



The C/C++ index needs to be rebuilt to reflect the build property changes you just made.

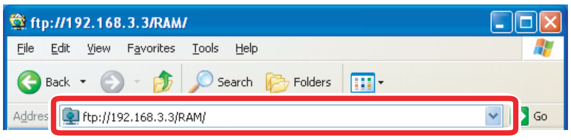
Do you wish to rebuild the index now?

□ Remember my decision

2. Storing the user program

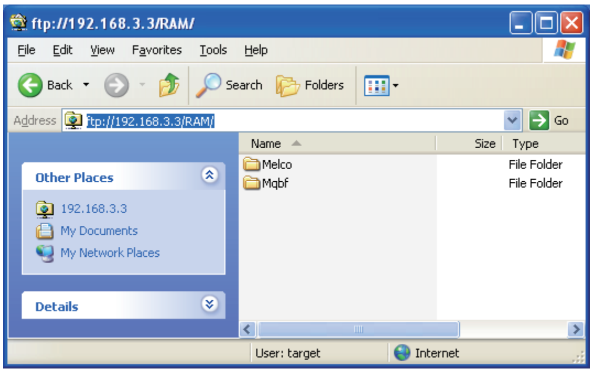
1) Start Explorer and enter the following address in the address area for the C Controller module.

ftp://192.168.3.3/RAM



Address: ftp://192.168.3.3/RAM/

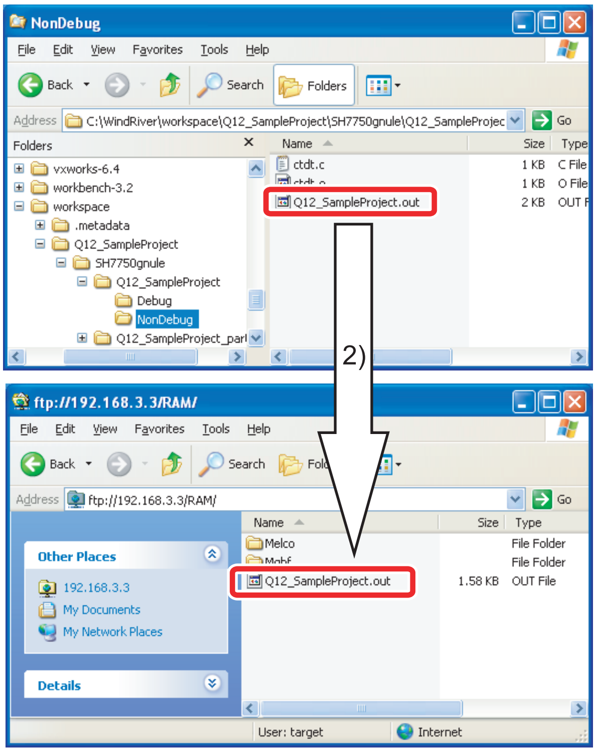
After login to the C Controller module, the address is displayed as shown below.



Address: ftp://192.168.3.3/RAM/

Name	Size	Type
Melco		File Folder
Mqbf		File Folder

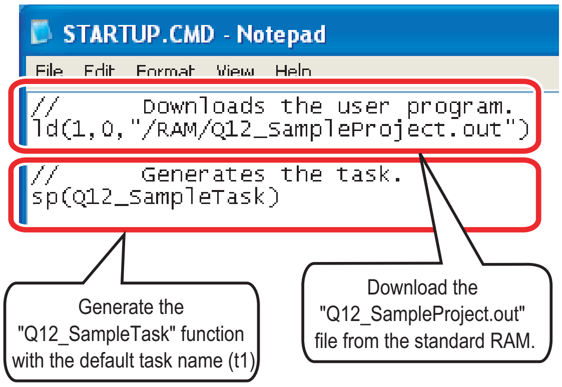
2) Copy the created user program "Q12_SampleProject.out" on the standard RAM for the C Controller module by drag and drop.
The user program created in this guide is stored on the following:
C:\WindRiver\workspace\Q12_SampleProject\SH7750gnule\Q12_SampleProject\NonDebug



3. Creating and storing a script file

Create a script file that automatically downloads the execution module at the start of the C Controller module.

1) Open a text file and describe a script file that downloads the user program and generates the task as shown below.



```
//      Downloads the user program.
ld(1,0,"/RAM/Q12_SampleProject.out")
//      Generates the task.
sp(Q12_SampleTask)
```

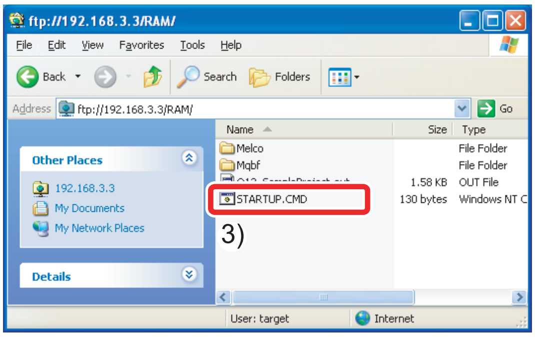
Generate the "Q12_SampleTask" function with the default task name (t1)

Download the "Q12_SampleProject.out" file from the standard RAM.

2) Name the file as "STARTUP.CMD" and save the file.

3) Copy the created script file on the standard RAM of the C Controller module.

ftp://192.168.3.3/RAM



The script file has been created and stored.

Point

A user program and a script file can be stored on the CompactFlash card as well. When a script file is stored both the standard RAM and the CompactFlash card, one on the CompactFlash card is started by priority.

<6> Checking Operations

Execute the program registered with the C Controller module and check operations.

Use the "RUN/STOP/MODE" and "RESET/SELECT" switches on the front of the C Controller module.

[Functions of the "RUN/STOP/MODE" switch]

- RUN : Enables outputs (Y) and writing to the buffer memory from a user program
- STOP : Disables outputs (Y) and writing to the buffer memory from a user program
- MODE : Used for the hardware self-diagnostic function

[Functions of the "RESET/SELECT" switch]

- RESET : Resets hardware and programs.
- SELECT : Used for the hardware self-diagnostic function

Point

The C Controller module executes program operation regardless of the switch status (RUN/STOP).

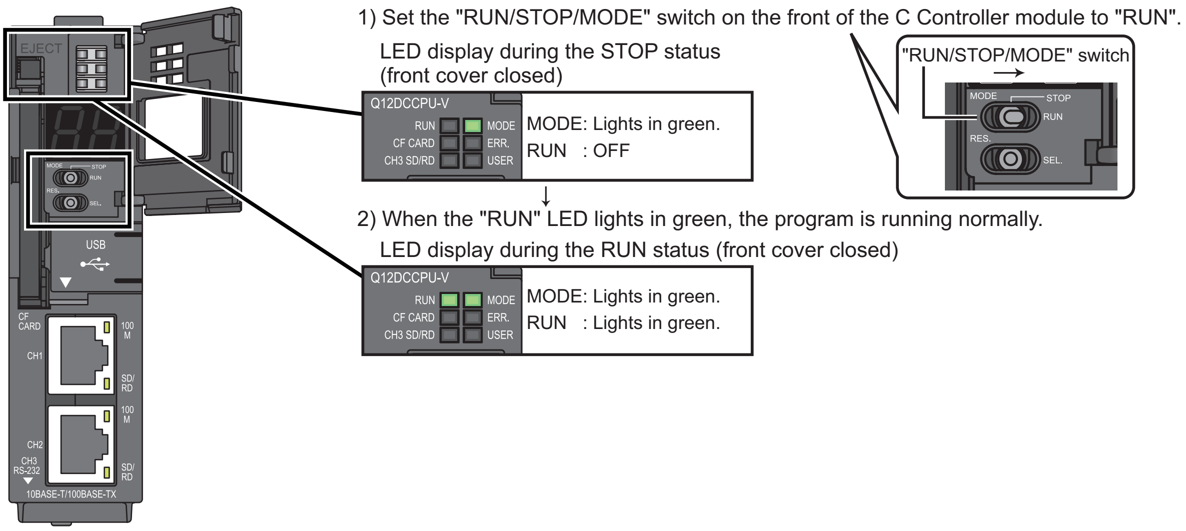
Reference

For details on the "RUN/STOP/MODE" and "RESET/SELECT" switches, refer to the following.

☞ C Controller Module User's Manual (Hardware Design, Function Explanation)
: SH-080766ENG

Operating procedure

1. Enable outputs (Y) from the user program.



1) Set the "RUN/STOP/MODE" switch on the front of the C Controller module to "RUN".

LED display during the STOP status
(front cover closed)

MODE: Lights in green.
RUN : OFF

2) When the "RUN" LED lights in green, the program is running normally.

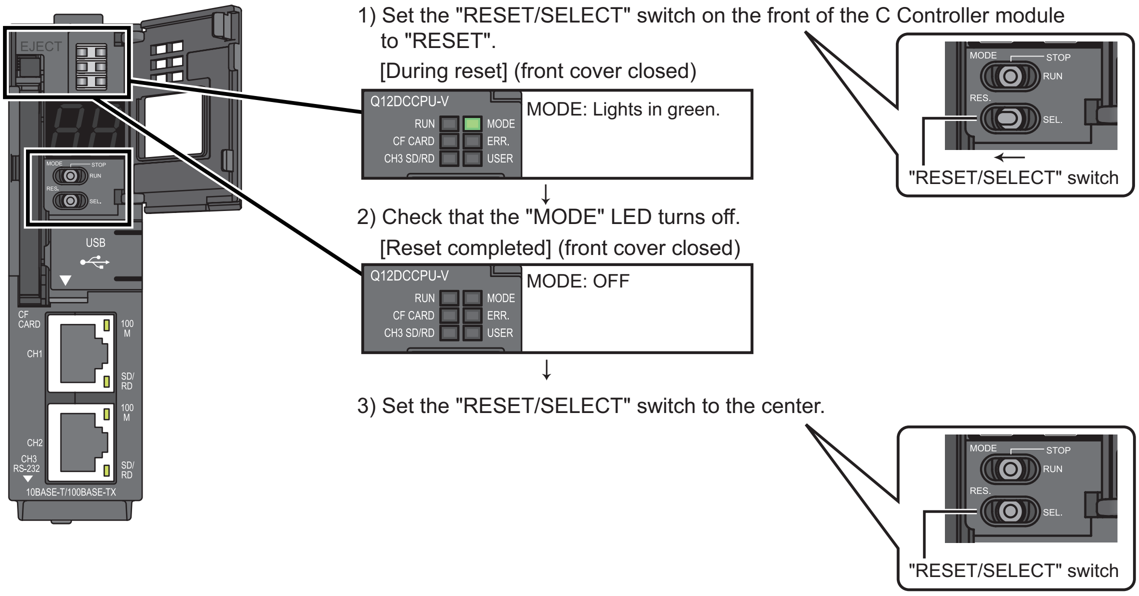
LED display during the RUN status (front cover closed)

MODE: Lights in green.
RUN : Lights in green.

Point

To disable outputs (Y) from the user program, set the "RUN/STOP/MODE" switch to "STOP".

2. Reset the C Controller module.



1) Set the "RESET/SELECT" switch on the front of the C Controller module to "RESET".

[During reset] (front cover closed)

MODE: Lights in green.

2) Check that the "MODE" LED turns off.

[Reset completed] (front cover closed)

MODE: OFF

3) Set the "RESET/SELECT" switch to the center.

Reference

If the "ERR." LED turns on or starts flashing, troubleshoot with reference to the following.

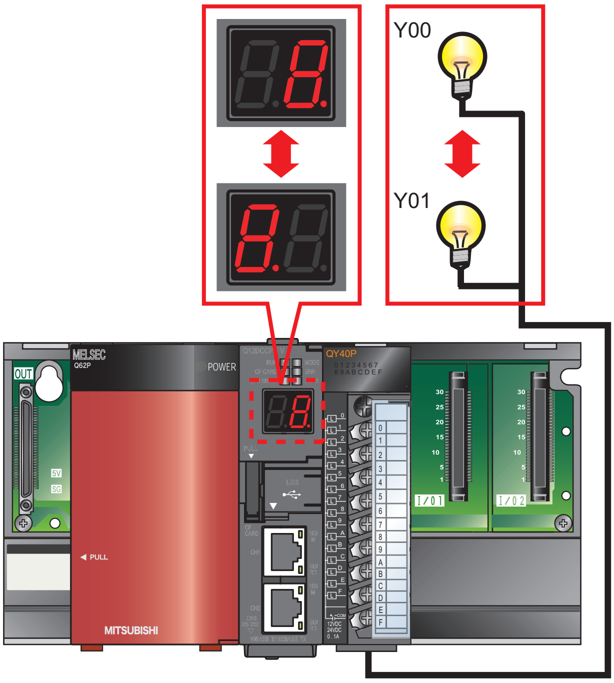
☞ C Controller Module User's Manual (Hardware Design, Function Explanation)
: SH-080766ENG

3. Use the 7-segment LED and lamps to check operations.

The 7-segment LED on the front of the C Controller module and output lamps operate as follows:

1) The tens place and ones place of the 7-segment LED alternately turn on by 20 times.

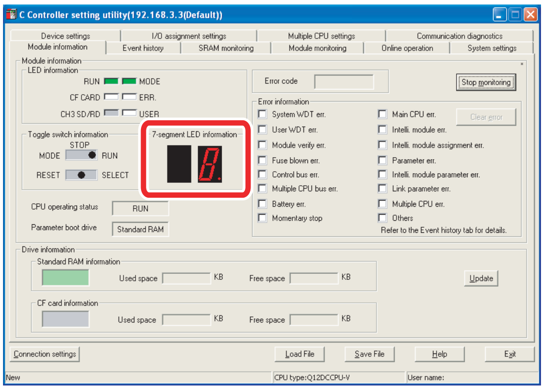
2) Synchronizing with the 7-segment LED, output lamps Y00 and Y01 alternately turn on.



3) To check the operations again, reset the C Controller module.

Reference

Status of the 7-segment LED and the output lamps also can be checked on C Controller setting utility. (P.52)



Module information

LED information: RUN 🟩 🟩 MODE; CF CARD □ □ ERR.; CH3 SD/RD □ □ USER

Toggle switch information: MODE STOP / RUN; RESET / SELECT

7-segment LED information

CPU operating status: RUN

Parameter boot drive: Standard RAM

Error code:

Error information

☐ System WDT err. ☐ Main CPU err.
☐ User WDT err. ☐ Intelli. module err.
☐ Module verify err. ☐ Intelli. module assignment err.
☐ Fuse blown err. ☐ Parameter err.
☐ Control bus err. ☐ Intelli. module parameter err.
☐ Multiple CPU bus err. ☐ Link parameter err.
☐ Battery err. ☐ Multiple CPU err.
☐ Momentary stop ☐ Others

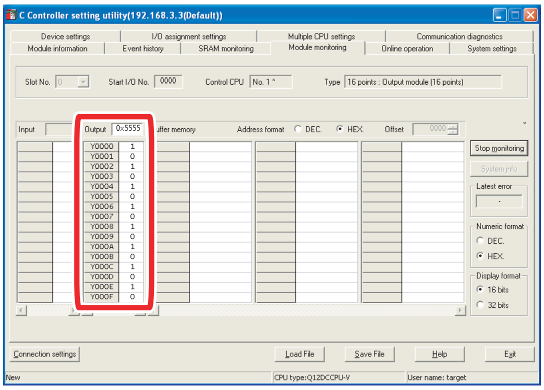
Refer to the Event history tab for details.

Drive information

Standard RAM information: Used space KB Free space KB

CF card information: Used space KB Free space KB

New | CPU type:Q12DCCPU-V | User name:



Slot No. 0 Start I/O No. 0000 Control CPU No. 1 * Type 16 points : Output module (16 points)

Input | Output 0x5555 | uffer memory | Address format ○ DEC. ● HEX. | Offset 0000

Y0000	1
Y0001	0
Y0002	1
Y0003	0
Y0004	1
Y0005	0
Y0006	1
Y0007	0
Y0008	1
Y0009	0
Y000A	1
Y000B	0
Y000C	1
Y000D	0
Y000E	1
Y000F	0

Latest error: -

Numeric format: ○ DEC. ● HEX.

Display format: ● 16 bits ○ 32 bits

New | CPU type:Q12DCCPU-V | User name: target

FREQUENTLY-USED FUNCTIONS

This chapter describes functions frequently used for the start-up and the maintenance after operation of a C Controller system.

<1> Checking Errors and Taking Corrective Action

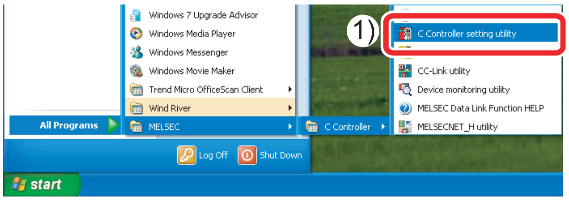
An error can be checked and the corrective action can be taken using C Controller setting utility.

1) How to check an error and take corrective action

Operating procedure

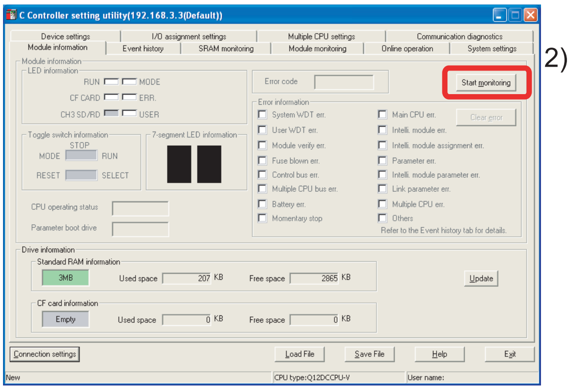
1. Checking for error <Module information>

1) Select[start]→[All Programs]→[MELSEC]→[C Controller]→[C Controller setting utility].



C Controller setting utility starts.

2) Click the Start monitoring button on the "Module information" tab.



Standard RAM information: 3MB Used space 207 KB Free space 2865 KB

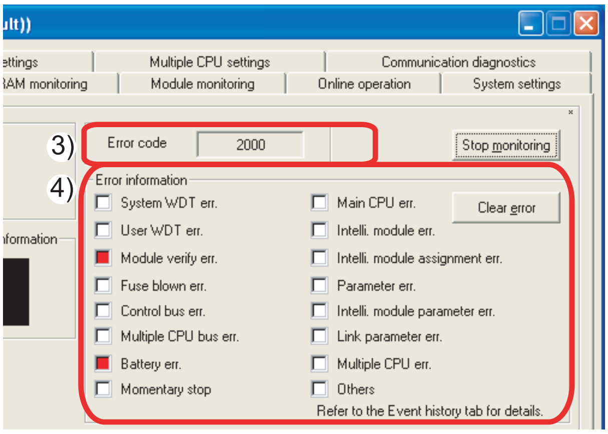
CF card information: Empty Used space 0 KB Free space 0 KB

New CPU type:Q12DCCPU-V User name:

3) An error code is displayed in the window.

4) The check boxes of the current errors color in red (🟥).

The error code is kept updated during monitoring.



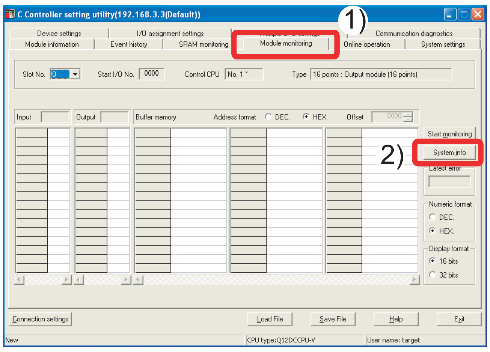
Error code 2000

□ System WDT err.
□ User WDT err.
🟥 Module verify err.
□ Fuse blown err.
□ Control bus err.
□ Multiple CPU bus err.
🟥 Battery err.
□ Momentary stop
□ Main CPU err.
□ Intelli. module err.
□ Intelli. module assignment err.
□ Parameter err.
□ Intelli. module parameter err.
□ Link parameter err.
□ Multiple CPU err.
□ Others

2. Checking the error cause and the corrective action <Module monitoring>

1) Click the "Module monitoring" tab.

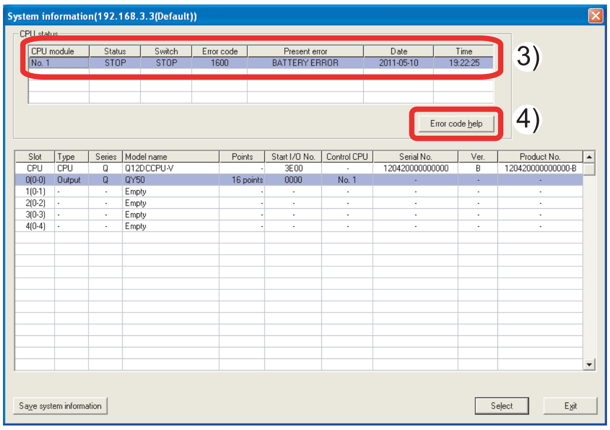
2) Click the System info button.



The "System information" window appears.

3) The current error is displayed in the window.

4) Click the Error code help button.


System information(192.168.3.3(Default))

CPU status

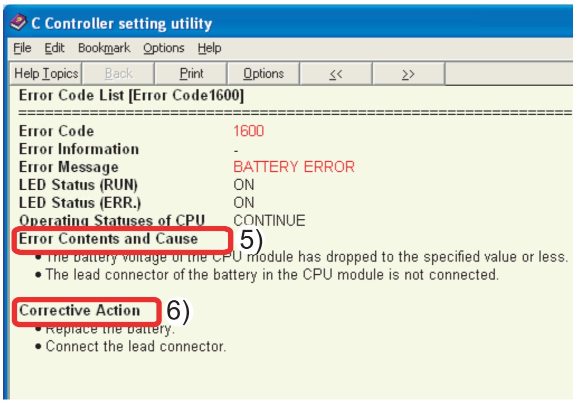
CPU module	Status	Switch	Error code	Present error	Date	Time
No. 1	STOP	STOP	1600	BATTERY ERROR	2011-05-10	19:22:25

Slot	Type	Series	Model name	Points	Start I/O No.	Control CPU	Serial No.	Ver.	Product No.
CPU	CPU	Q	Q12DCCPU-V	-	3E00	-	120420000000000	B	120420000000000-B
0(0-0)	Output	Q	QY50	16 points	0000	No. 1	-	-	-
1(0-1)	-	-	Empty	-	-	-	-	-	-
2(0-2)	-	-	Empty	-	-	-	-	-	-
3(0-3)	-	-	Empty	-	-	-	-	-	-
4(0-4)	-	-	Empty	-	-	-	-	-	-

The help window for the current error appears.

5) Find the possible cause from "Error Contents and Cause".

6) Take corrective action with reference to "Corrective Action".


C Controller setting utility

Error Code List [Error Code1600]

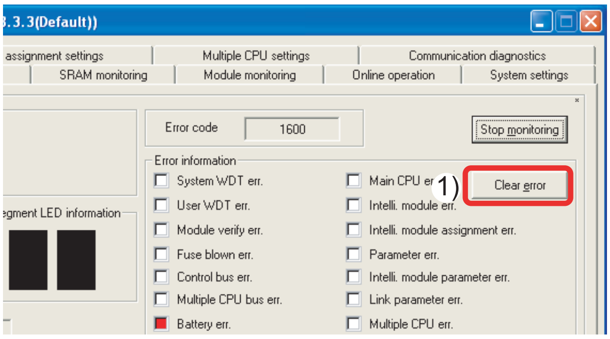
Error Code 1600
Error Information -
Error Message BATTERY ERROR
LED Status (RUN) ON
LED Status (ERR.) ON
Operating Statuses of CPU CONTINUE
Error Contents and Cause
- The battery voltage of the CPU module has dropped to the specified value or less.
- The lead connector of the battery in the CPU module is not connected.

Corrective Action
- Replace the battery.
- Connect the lead connector.

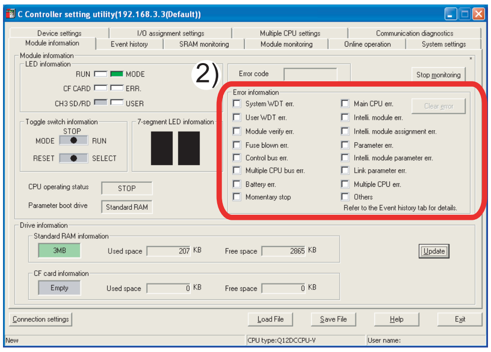
3. Clearing the error after taking the corrective action

(1) When the "ERR." LED of the C Controller module is on

1) Click the Clear error button in the "Module information" tab.



2) Check that the error has been cleared.



(2) When the "ERR." LED of the C Controller module is flashing

After taking the corrective action, reset the C Controller module.

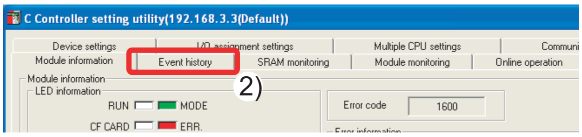
2) Checking error history

Errors occurred up to the present and the error details can be checked.
When and what kind of error occurs can be checked, useful in error analysis.

Operating procedure

1) Start C Controller setting utility.

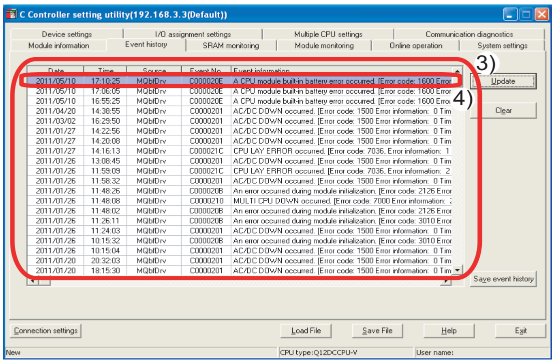
2) Click the "Event history" tab.



Error code: 1600

3) Error history and the error details are displayed.

4) To see more details of an error, double-click the error.



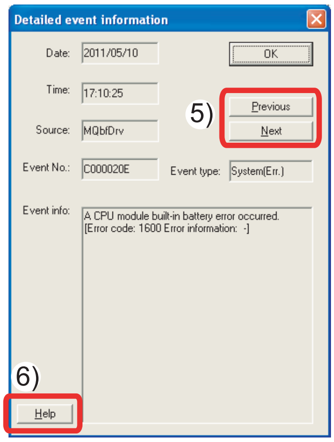
Date	Time	Source	Event No.	Event information
2011/05/10	17:10:25	MQbfDrv	C000020E	A CPU module built-in battery error occurred. [Error code: 1600 Error
2011/05/10	17:06:05	MQbfDrv	C000020E	A CPU module built-in battery error occurred. [Error code: 1600 Error
2011/05/10	16:55:25	MQbfDrv	C000020E	A CPU module built-in battery error occurred. [Error code: 1600 Error
2011/04/20	14:38:55	MQbfDrv	C0000201	AC/DC DOWN occurred. [Error code: 1500 Error information: 0 Tim
2011/03/02	16:29:50	MQbfDrv	C0000201	AC/DC DOWN occurred. [Error code: 1500 Error information: 0 Tim
2011/01/27	14:22:56	MQbfDrv	C0000201	AC/DC DOWN occurred. [Error code: 1500 Error information: 0 Tim
2011/01/27	14:20:08	MQbfDrv	C0000201	AC/DC DOWN occurred. [Error code: 1500 Error information: 0 Tim
2011/01/27	14:16:13	MQbfDrv	C000021C	CPU LAY ERROR occurred. [Error code: 7036, Error information: 1
2011/01/26	13:08:45	MQbfDrv	C0000201	AC/DC DOWN occurred. [Error code: 1500 Error information: 0 Tim
2011/01/26	11:59:09	MQbfDrv	C000021C	CPU LAY ERROR occurred. [Error code: 7036, Error information: 2
2011/01/26	11:58:32	MQbfDrv	C0000201	AC/DC DOWN occurred. [Error code: 1500 Error information: 0 Tim
2011/01/26	11:48:26	MQbfDrv	C000020B	An error occurred during module initialization. [Error code: 2126 Error
2011/01/26	11:48:08	MQbfDrv	C0000210	MULTI CPU DOWN occurred. [Error code: 7000 Error information: 2
2011/01/26	11:48:02	MQbfDrv	C000020B	An error occurred during module initialization. [Error code: 2126 Error
2011/01/26	11:26:11	MQbfDrv	C000020B	An error occurred during module initialization. [Error code: 3010 Error
2011/01/26	11:24:03	MQbfDrv	C0000201	AC/DC DOWN occurred. [Error code: 1500 Error information: 0 Tim
2011/01/26	10:15:32	MQbfDrv	C000020B	An error occurred during module initialization. [Error code: 3010 Error
2011/01/26	10:15:04	MQbfDrv	C0000201	AC/DC DOWN occurred. [Error code: 1500 Error information: 0 Tim
2011/01/20	20:32:03	MQbfDrv	C0000201	AC/DC DOWN occurred. [Error code: 1500 Error information: 0 Tim
2011/01/20	18:15:30	MQbfDrv	C0000201	AC/DC DOWN occurred. [Error code: 1500 Error information: 0 Tim

New | CPU type:Q12DCCPU-V | User name:

The "Detailed event information" window appears.

5) Clicking the Previous or the Next button will display the details of the previous or the following error.

6) Clicking the Help button will open the help window on the error.



Date: 2011/05/10

Time: 17:10:25

Source: MQbfDrv

Event No.: C000020E

Event type: System(Err.)

Event info: A CPU module built-in battery error occurred.
[Error code: 1600 Error information: -]

<2> Monitoring Module Status and Testing Operations

Module I/O status and buffer memory status can be checked through C Controller setting utility. I/O status can be checked and operations can be tested at start-up and maintenance.

1) Checking module I/O status and buffer memory status

The input (X) and output (Y) status of the module and buffer memory status can be monitored.

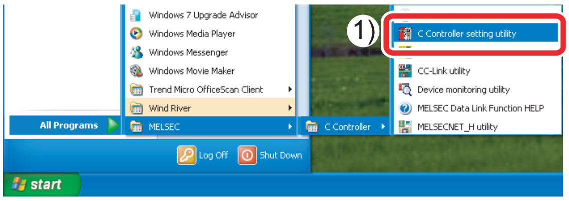
Terminology

Buffer memory: The memory of an intelligent function module (module such as A/D conversion module and D/A conversion module having a function other than input and output) used to store data (such as setting values and monitored values) for communication with a C Controller module

Operating procedure

1. Start C Controller setting utility.

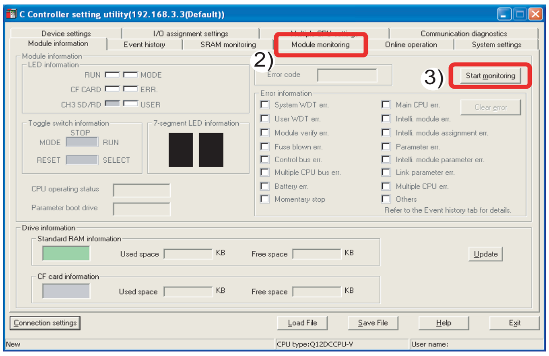
1) Select [start]→[All Programs]→[MELSEC]→[C Controller]→[C Controller setting utility].



C Controller setting utility starts.

2) Click the "Module monitoring" tab.

3) Click the Start monitoring button.


C Controller setting utility(192.168.3.3(Default))

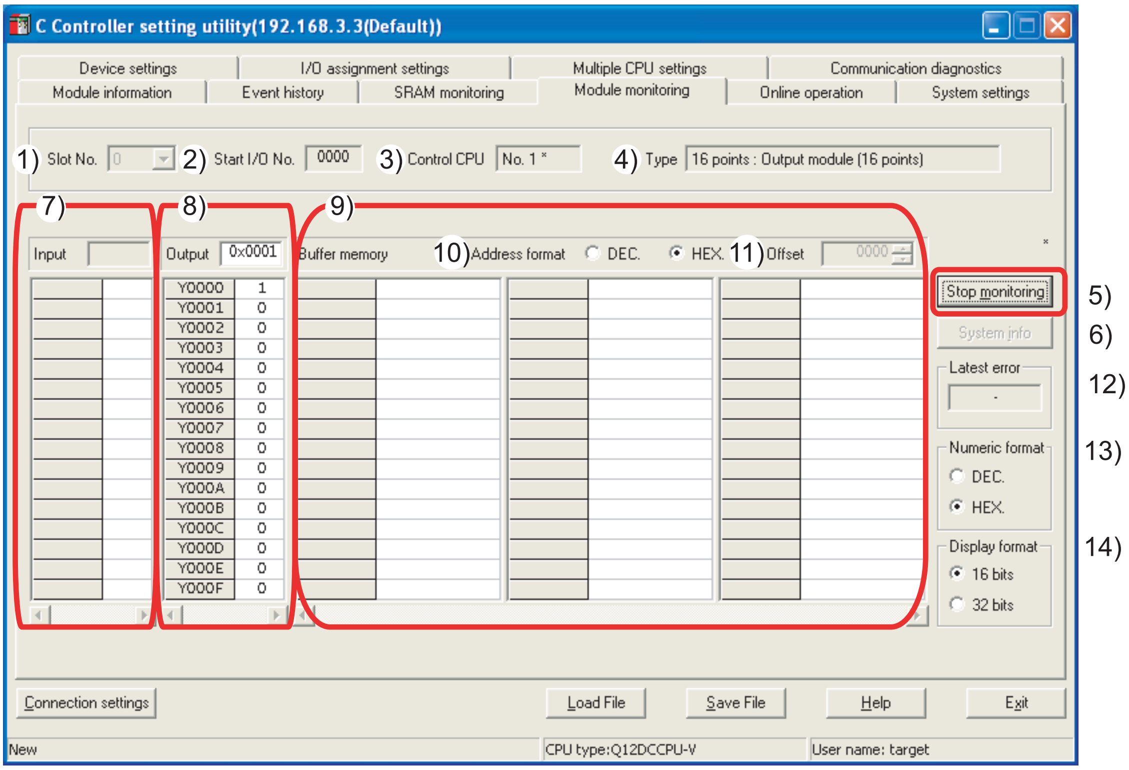
□ System WDT err.
□ User WDT err.
□ Module verify err.
□ Fuse blown err.
□ Control bus err.
□ Multiple CPU bus err.
□ Battery err.
□ Momentary stop
□ Main CPU err.
□ Intelli. module err.
□ Intelli. module assignment err.
□ Parameter err.
□ Intelli. module parameter err.
□ Link parameter err.
□ Multiple CPU err.
□ Others

Refer to the Event history tab for details.

New | CPU type:Q12DCCPU-V | User name:

The "Module monitoring" window appears.

2. Check the "Module monitoring" window.


C Controller setting utility(192.168.3.3(Default))

1) Slot No. 0
2) Start I/O No. 0000
3) Control CPU No. 1 *
4) Type 16 points : Output module (16 points)

7) Input

8) Output 0x0001

Y0000	1
Y0001	0
Y0002	0
Y0003	0
Y0004	0
Y0005	0
Y0006	0
Y0007	0
Y0008	0
Y0009	0
Y000A	0
Y000B	0
Y000C	0
Y000D	0
Y000E	0
Y000F	0

9) Buffer memory

10) Address format ○ DEC. ● HEX.

11) Offset 0000

5) Stop monitoring

6) System info

12) Latest error -

13) Numeric format ○ DEC. ● HEX.

14) Display format ● 16 bits ○ 32 bits

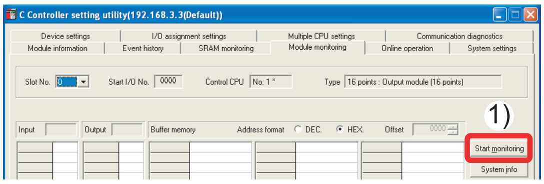
New | CPU type:Q12DCCPU-V | User name: target

No.	Name	Description
1)	Slot No.	Specify a slot No. to be monitored.
2)	Start I/O No.	Displays the start I/O No. of the module mounted on the slot specified in 1).
3)	Control CPU	Displays the CPU No. that controls the module mounted on the slot specified in 1). When C Controller setting utility is connected to the C Controller module that serves as a control CPU, "*" appears on the right of the CPU No.
4)	Type	Displays the number of I/O points and the type of a module when a module other than a CPU module is mounted on the slot specified in 1).
5)	Start monitoring button, Stop monitoring button	Starts or stops monitoring of the C Controller module. "*" flashes in the upper right of this button during monitoring.
6)	System info button	Displays the "System information" window.
7)	Input	Monitors the input (X) of the module mounted on the slot specified in 1). 0: OFF 1: ON
8)	Output	Monitors the output (Y) of the module mounted on the slot specified in 1). 0: OFF 1: ON
9)	Buffer memory	Monitors a buffer memory when an intelligent function module is mounted on the slot specified in 1).
10)	Address format	Select a numeric format for "Offset".
11)	Offset	Specify the address of a buffer memory area to be monitored.
12)	Latest error	Displays the error code of the latest error occurred in an intelligent function module.
13)	Numeric format	Select a numeric format for a buffer memory or a CPU shared memory.
14)	Display format	Select a display format for a buffer memory or a CPU shared memory.

2) Testing operations by forced output

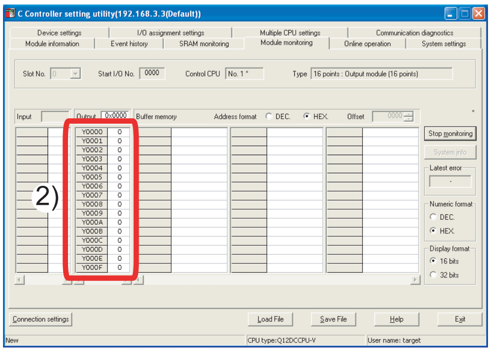
Module operations can be tested by forced output from an output (Y).
The following describes the procedure for forced output.

1) Click the Start monitoring button in the "Module monitoring" window.



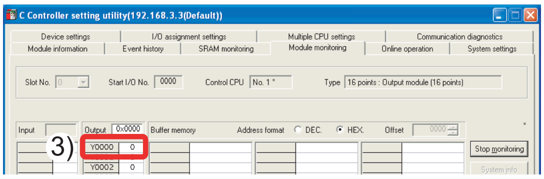
Slot No. 0 Start I/O No. 0000 Control CPU No. 1 Type 16 points : Output module (16 points)

2) Check the output status.

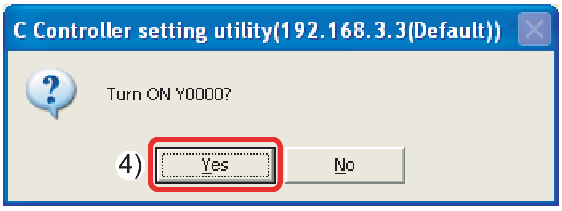


Y0000	0
Y0001	0
Y0002	0
Y0003	0
Y0004	0
Y0005	0
Y0006	0
Y0007	0
Y0008	0
Y0009	0
Y000A	0
Y000B	0
Y000C	0
Y000D	0
Y000E	0
Y000F	0

3) Double-click the output (Y) from which forced output is executed.

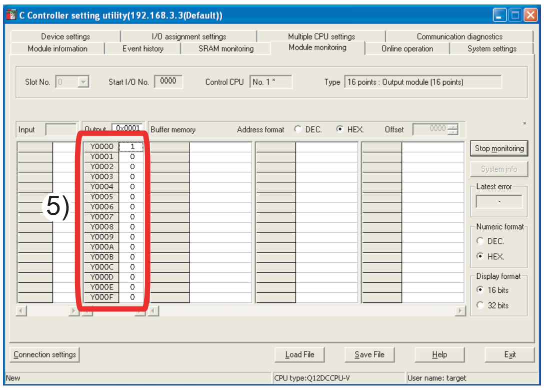


4) The confirmation window appears.
Clicking the Yes button will execute forced output from the output (Y).



Turn ON Y0000?

5) Check the output status.

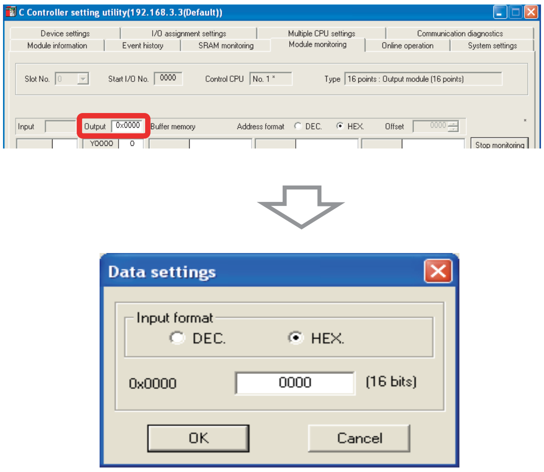


Y0000	1
Y0001	0
Y0002	0
Y0003	0
Y0004	0
Y0005	0
Y0006	0
Y0007	0
Y0008	0
Y0009	0
Y000A	0
Y000B	0
Y000C	0
Y000D	0
Y000E	0
Y000F	0

The LED of the output module turns on.

Reference

Double-clicking an item under "Output" will open the "Data settings" window. Outputs (Y) can be batch-selected by specifying a numeric value.



0x0000 0000 (16 bits)

Point

An operation test by forced write to a buffer memory can be executed in the same manner.

iQ Platform

Mitsubishi Electric
Programmable Controllers

Precautions before use

This publication explains the typical features and functions of the products herein and does not provide restrictions and other information related to usage and module combinations. Before using the products, always read the product user manuals. Mitsubishi Electric will not be held liable for damage caused by factors found not to be the cause of Mitsubishi Electric; opportunity loss or lost profits caused by faults in Mitsubishi Electric products; damage, secondary damage, or accident compensation, whether foreseeable or not, caused by special factors; damage to products other than Mitsubishi Electric products; and to other duties.

⚠ For safe use

- To use the products given in this publication properly, always read the relevant manuals before use.
- The products have been manufactured as general-purpose parts for general industries, and have not been designed or manufactured to be incorporated in a device or system used in purposes related to human life.
- Before using the products for special purposes such as nuclear power, electric power, aerospace, medicine or passenger movement vehicles, consult with Mitsubishi.
- The products have been manufactured under strict quality control. However, when installing the products where major accidents or losses could occur if the products fail, install appropriate backup or fail-safe functions in the system.

Country/Region	Sales office	Tel/Fax
USA	Mitsubishi Electric Automation Inc. 500 Corporate Woods Parkway Vernon Hills, IL 60061, USA	Tel : +1-847-478-2100 Fax : +1-847-478-2253
Brazil	MELCO-TEC Rep. Com.e Assessoria Tecnica Ltda. Av Paulista, 1439-Cj. 72 Cerqueira Cesar CEP 01311-200, Sao Paulo, SP, CEP:01311-200, Brazil	Tel : +55-11-3146-2200 Fax : +55-11-3146-2217
Germany	Mitsubishi Electric Europe B.V. German Branch Gothaer Strasse 8 D-40880 Ratingen, Germany	Tel : +49-2102-486-0 Fax : +49-2102-486-1120
UK	Mitsubishi Electric Europe B.V. UK Branch Travellers Lane, Hatfield, Hertfordshire., AL10 8XB, UK	Tel : +44-1707-276100 Fax : +44-1707-278695
Italy	Mitsubishi Electric Europe B.V. Italian Branch Viale Colleoni 7-20041 Agrate Brianza (Milano), Italy	Tel : +39-039-60531 Fax : +39-039-6053312
Spain	Mitsubishi Electric Europe B.V. Spanish Branch Carretera de Rubi 76-80 E-08190 Sant Cugat del Valles(Barcelona), Spain	Tel : +34-93-565-3131 Fax : +34-93-589-2948
France	Mitsubishi Electric Europe B.V. French Branch 25,Boulevard des Bouvets, F-92741 Nanterre Cedex, France	Tel : +33-1-5568-5568 Fax : +33-1-5568-5757
Czech Republic	Mitsubishi Electric Europe B.V.-o.s.-Czech office Avenir Business Park, Radlická 714/113a CZ-158 00 Praha 5	Tel : +420-251-551-470 Fax : +420-251-551-471
Poland	Mitsubishi Electric Europe B.V. Polish Branch ul. Krakowska 50 32-083 Balice, Poland	Tel : +48-12-630-47-00 Fax : +48-12-630-47-01
Russia	Mitsubishi Electric Europe B.V. Russian branch St.Petersburg office Sverdlovskaya emb., bld "Sch", BC "Benua", office 720; 195027, St.Petersburg, Russia	Tel : +7-812-633-3497 Fax : +7-812-633-3499
South Africa	Circuit Breaker Industries Ltd. 9 Derrick Road, Spartan, Gauteng PO Box 100, Kempton Park 1620, South Africa	Tel : +27-11-977-0770 Fax : +27-11-977-0761
China	Mitsubishi Electric Automaiton (China) Ltd. No.1386 Hongqiao Road,Mitsubishi Electric Automation Center Shanghai China	Tel : +86-21-2322-3030 Fax : +86-21-2322-3000
Taiwan	Setsuyo Enterprise Co., Ltd. 6F., No.105, Wugong 3 rd, Wugu Dist, New Taipei City 24889, Taiwan, R.O.C.	Tel : +886-2-2299-2499 Fax : +886-2-2299-2509
Korea	Mitsubishi Electric Automation Korea Co., Ltd. 1480-6, Gayang-dong, Gangseo-ku Seoul 157-200, Korea	Tel : +82-2-3660-9530 Fax : +82-2-3664-8372
Singapore	Mitsubishi Electric Asia Pte, Ltd. 307 Alexandra Road #05-01/02, Mitsubishi Electric Bulding Singapore 159943	Tel : +65-6470-2480 Fax : +65-6476-7439
Thailand	Mitsubishi Electric Automation (Thailand) Co., Ltd. Bang-Chan Industrial Estate No.111 Soi Serithai 54, T.Kannayao, A.Kannayao, Bangkok 10230 Thailand	Tel : +66-2-906-3238 Fax : +66-2-906-3239
Indonesia	P.T. Autoteknindo Sumber Makmur Muara Karang Selatan Block A/Utara No.1 Kav. No.11 Kawasan Industri/Pergudangan Jakarta-Utara 14440, P.O Box5045 Jakarta 11050, Indonesia	Tel : +62-21-663-0833 Fax : +62-21-663-0832
India	Mitsubishi Electric India Pvt. Ltd. 2nd Floor, DLF Building No.9B, DLF Cyber City Phase III, Gurgaon 122002, Haryana, India	Tel : +91-124-4630300 Fax : +91-124-4630399
Australia	Mitsubishi Electric Australia Pty.Ltd. 348 Victoria Road, Rydalmere, N.S.W 2116, Australia	Tel : +61-2-9684-7777 Fax : +61-2-9684-7245

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

New publication, effective Jun. 2011.
Specifications are subject to change without notice.