Introduction

Welcome to the world of development environment CubeSuite+.
This tutorial introduces you to the integrated development environment provided by CubeSuite+ and the operation of CubeSuite+. By carrying out all the steps described in this tutorial, from creating a program to debugging of the microcontroller, you can easily experience the operation of CubeSuite+.
In this tutorial, you will use the E1 (on-chip debugging emulator) and MSRHQ176CP01 (target board: RH850/E1x evaluation board (made by Hitachi ULSI Systems Co., Ltd.)) to actually experience microcontroller system development using CubeSuite+.

Features of CubeSuite+

CubeSuite+ is an integrated development environment that provides an environment for developing microcontrollers from code generation, build, and debugging all in one tool.

Easy GUI customization

You can customize the screen as you like using such features as "docking", "floating", or "automatic hiding" to manipulate various panels of CubeSuite+ at will. CubeSuite+ now provides a feature to save the development environment in addition to the conventional feature to save the project environment. All of these features help you develop a microcontroller system more smoothly.

Easy preparation of development environment

Since the development environment for system development is integrated, it is easy to install the required tools. Because it is equipped with an automatic update function, you can update the software to its latest version (including documents) with a single click.

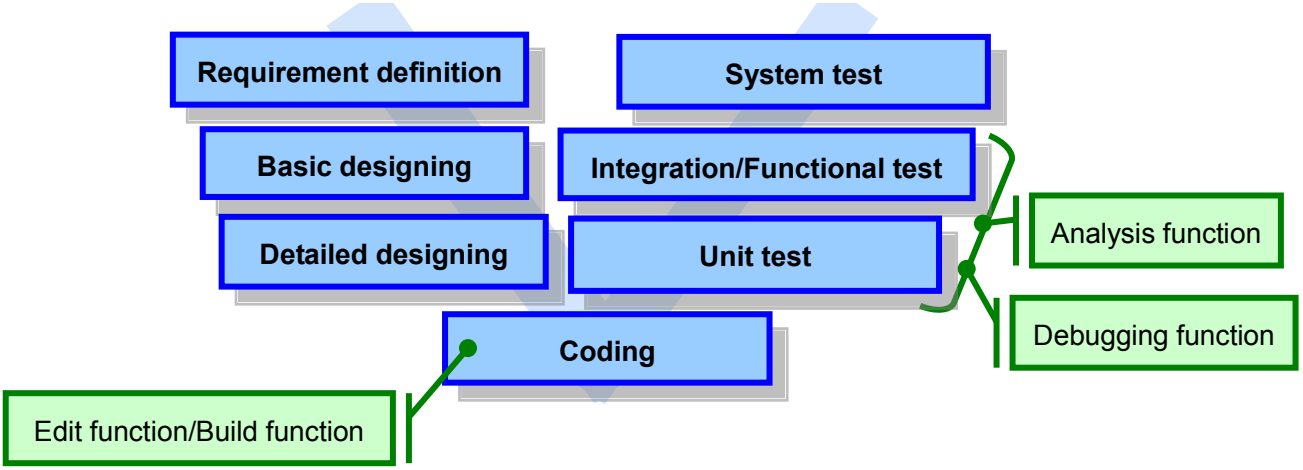
Reading this tutorial and the following document enables you to learn Overeview of programming for the RH850 multi-core.

Overview of programming for the RH850 multi-core(R20UT3069EJ)

Flow of Microcontroller System Development

This section describes a flow of system development using CubeSuite+.

General flow of system development (V-shaped model)



Following are the functions of CubeSuite+ corresponding to the flow of system development.

<Function>	<Description>
Edit function/ Build function	The edit function is for editing a program. After completing creation of a program, the built function is used to build the program.
Debugging function	This function enables you to debug the object code after downloading it to the target microcontroller.
Analysis function	This function helps you improve the execution performance and control the quality of a program by checking the analysis result.

Overview of Sample Program

This section describes an overview of the sample program and target board (MSRHQ176CP01).

1. Overview of sample program

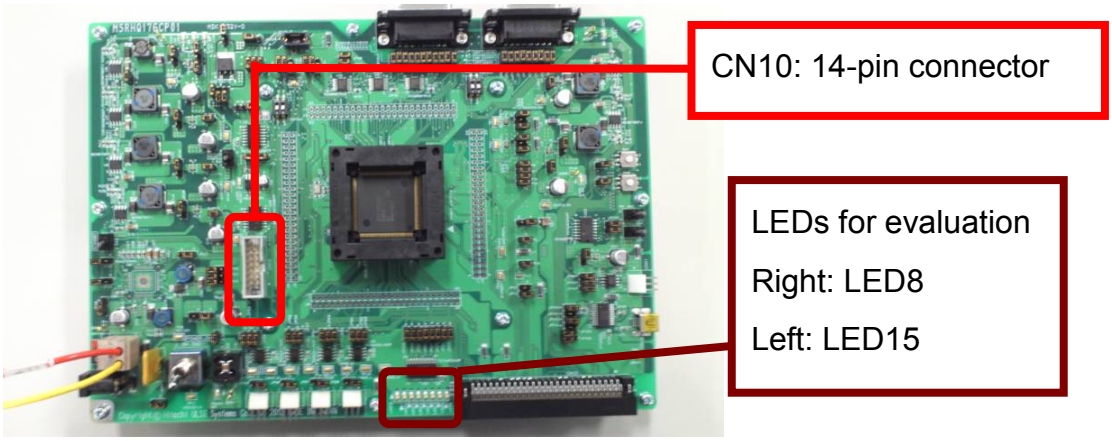
The program used here controls (turns on/off) a different LED for each core (CPU1 and PCU) of RH850/E1x.
For a detailed description of the program, refer to appendix, Description of Sample Programs.

CPU1 core: Controls LED9 and makes LED9 turn on and off.
PCU core: Controls LED10 and makes LED10 turn on and off.

2. Overview of target board (MSRHQ176CP01)

The following is an overview of MSRHQ176CP01 which is used as the target board.

MSRHQ176CP01



LED8 to LED15: Lights when P2_n (n = 0-7) of port group 2 is high.
CN10: Used at on-chip debugging or data writing

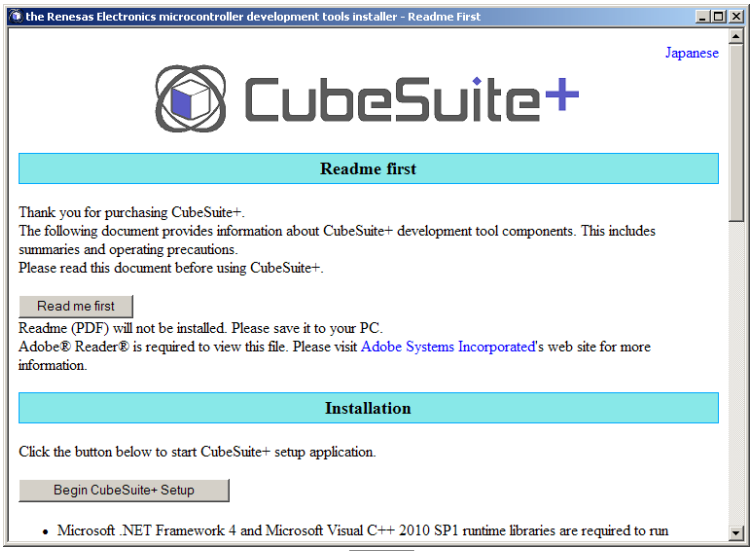
Installing

This section describes a procedure to install CubeSuite+.

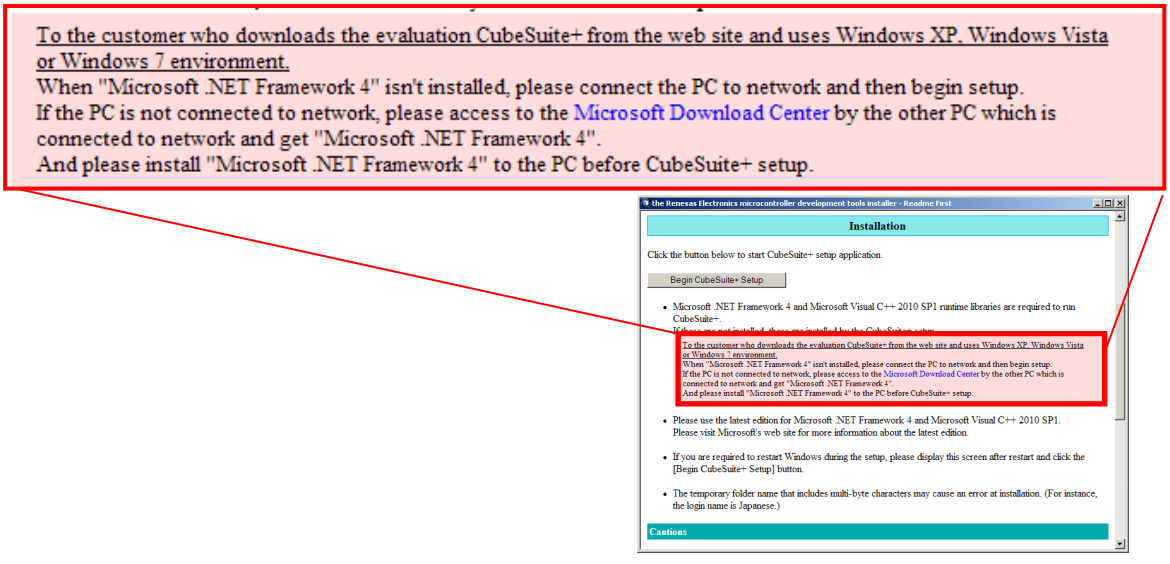
1. Installing Microsoft software products prior to installation

You must install .NET Framework and Visual C++ Runtime Library before installing CubeSuite+. If these software products have not been installed in the PC used, they will be installed at the time of the setup of CubeSuite+.

Insert CubeSuite+ product DVD into the drive of the PC.
The following screen appears automatically.



Install the required software products.

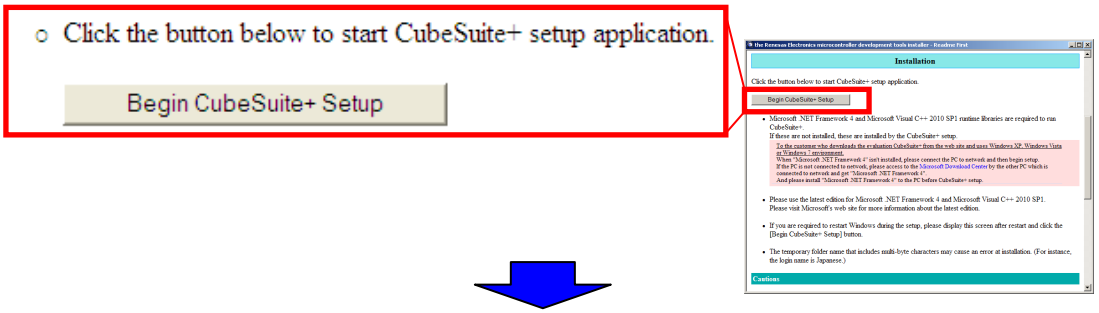


Installing

2. Running the integrated installer

CubeSuite+ products are installed by running the integrated installer.

Click [Begin CubeSuite+ Setup] and start the setup of CubeSuite+.



Make settings following the instructions provided by the installation wizard. As a final step, click the [Finish] button and complete installation.
* Restart the PC after installation.

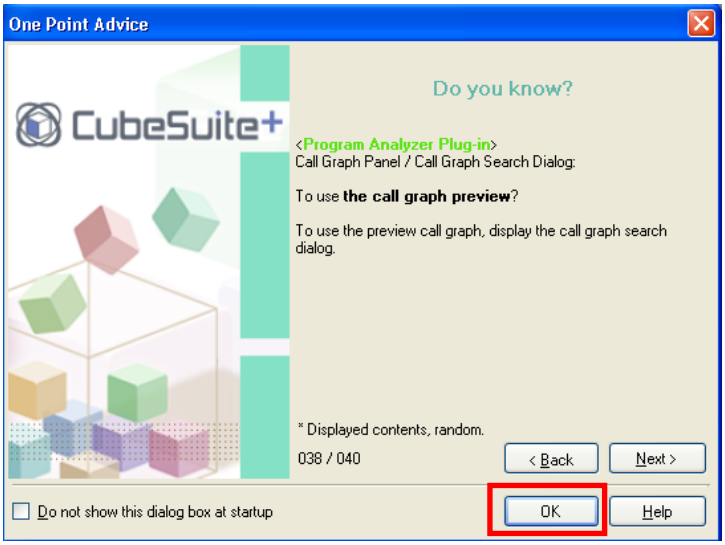
Starting CubeSuite+

This section describes the procedures from starting CubeSuite+ to creating a project.

1. Starting CubeSuite+

Start CubeSuite+ by selecting [Start] > [All Programs] > [Renesas Electronics CubeSuite+] > [CubeSuite+].

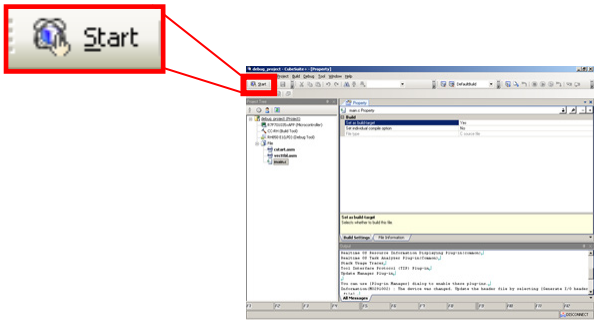
The One Point Advice dialog box opens when CubeSuite+ is started. Click the [Next] button if you want to read the content. Clicking the [OK] button displays the start screen of CubeSuite+.



Tip

About the Start panel

When you start to use CubeSuite+ to create a new project, click the "Start panel" button (see the figure below). The Start panel opens where you can easily create a new project or open the project you used recently or your favorite project. (The Start panel is displayed when you start CubeSuite+ for the first time after installation. If you have created a project, the latest project opens when you start CubeSuite+.)



Starting CubeSuite+

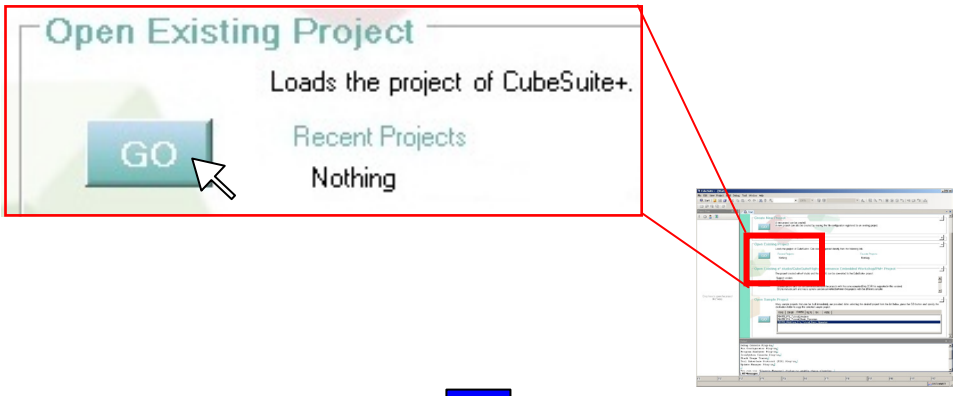
2. Loading the sample project

In this step, load the sample project.

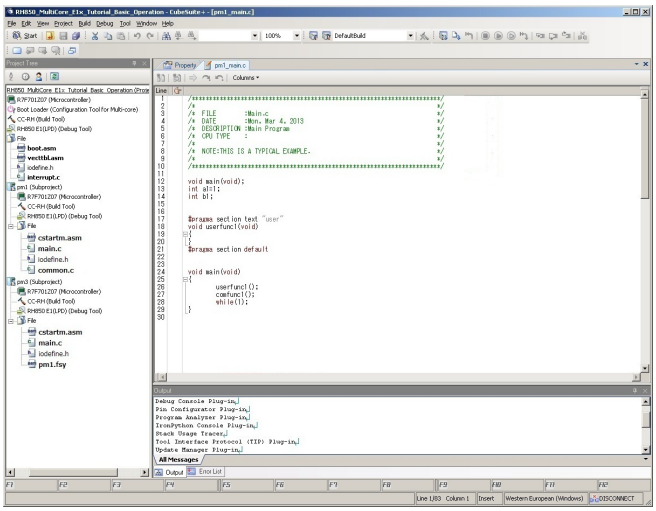
This document describes the step using a project that has been created according to the construction method for a project using CubeSuite+.
For details, refer to Tutorial for RH850 Multi-core Environment (Build).

Tutorial for RH850 Multi-core Environment(R20UT3070EJ)

In the "Open Existing Project" field, click the [GO] button, then select the created project (.mtpj).



Follow the on-screen instructions. A sample project opens as shown in the figure below.

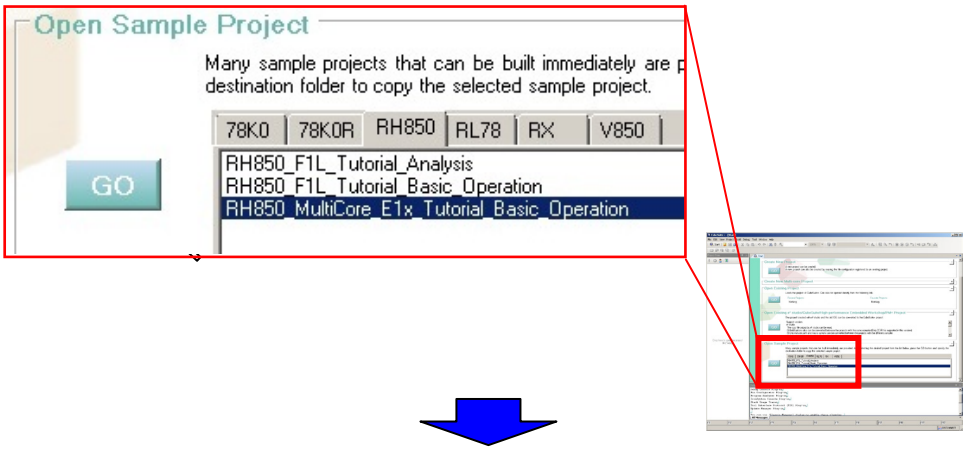


Tip

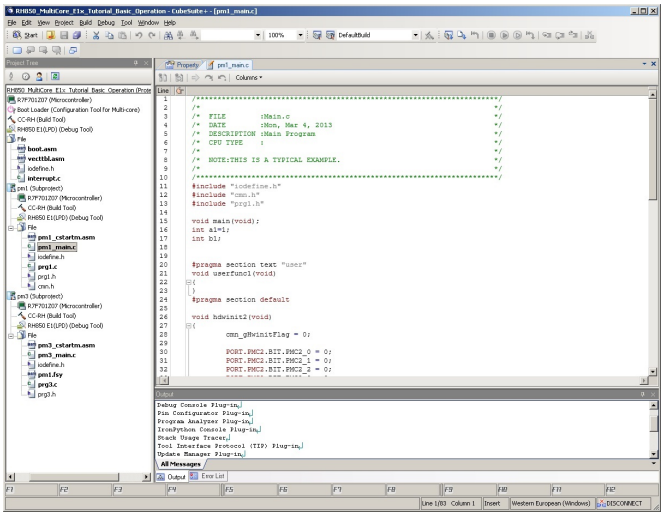
About sample projects

CubeSuite+ provides sample projects.
Sample projects are in a state after the "Editing the Program" operations in this document have been performed.
When using a sample project provided by CubeSuite+, load the sample project as shown below.

In the "Open Sample Project" field, from the [RH850] tab, select "RH850_Multicore_E1x_Tutorial_Basic_Operation", then click the [GO] button.



Follow the on-screen instructions. A sample project opens as shown in the figure below.

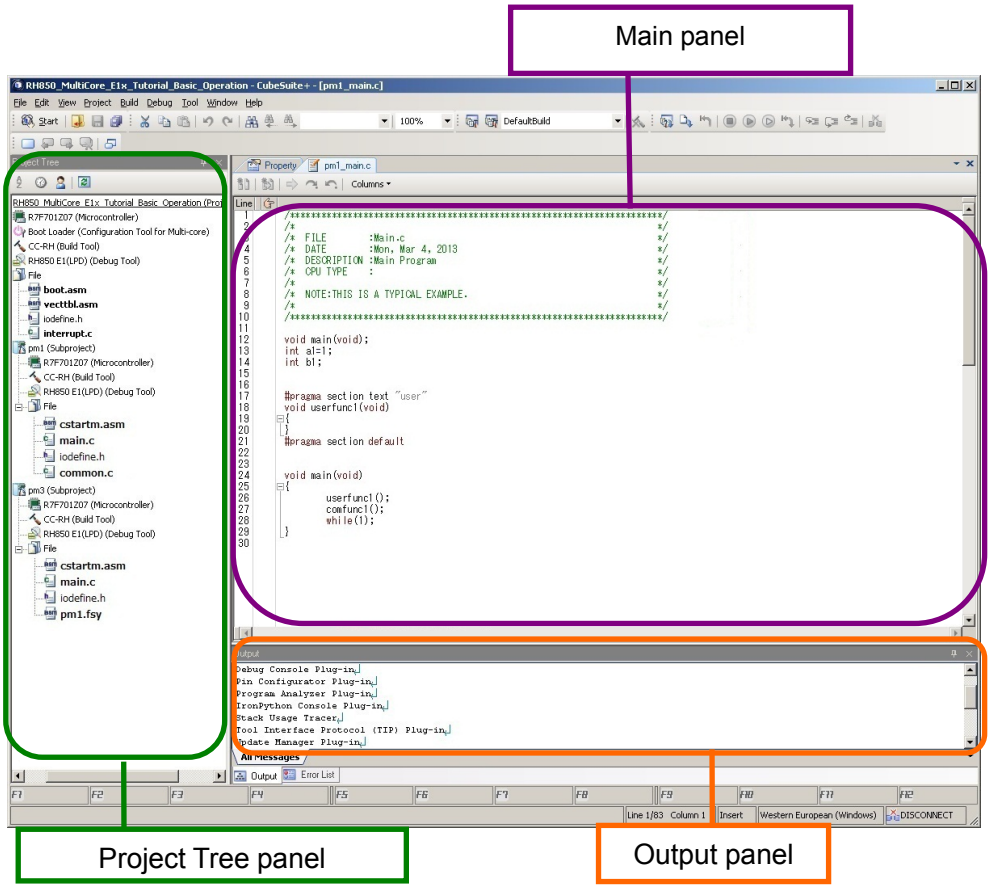


Manipulating Windows

In CubeSuite+, you can customize the windows at will. This section describes the configuration of windows and window customization functions such as "automatic hiding", "floating", and "docking".

1. Configuration of windows

The following figure shows the configuration of windows of CubeSuite+.



Project Tree panel: Displays the functions of CubeSuite+ corresponding to the flow of system development.
Main panel: Displays the panel (Editor panel, etc.) corresponding to the function selected in the Project Tree.
Output panel: Displays the output results.

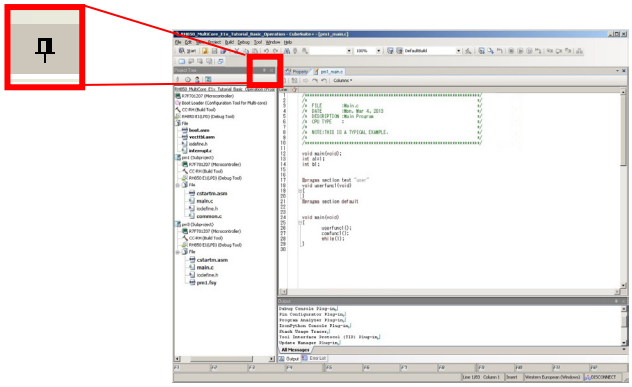
Manipulating Windows

2. Automatic hiding

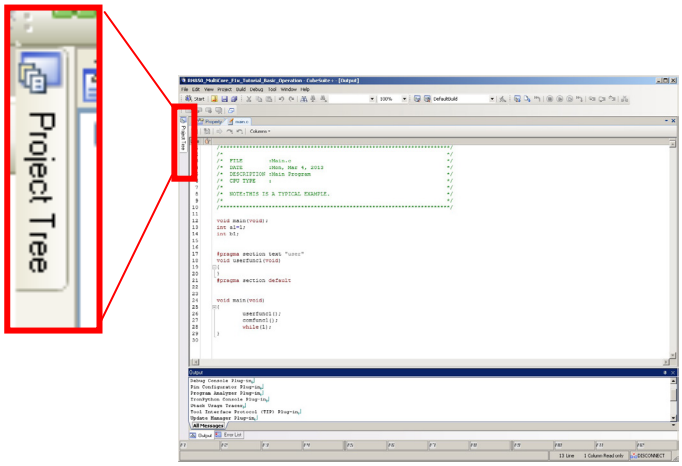
By clicking the Pin icon on the title bar of each panel, you can easily change the setting that determines whether or not to hide the panel automatically. By hiding the panels not necessary for operation, you can use the screen more effectively.

(a) Hiding the panel automatically (example: Project Tree)

Click the Pin icon in the Project Tree.

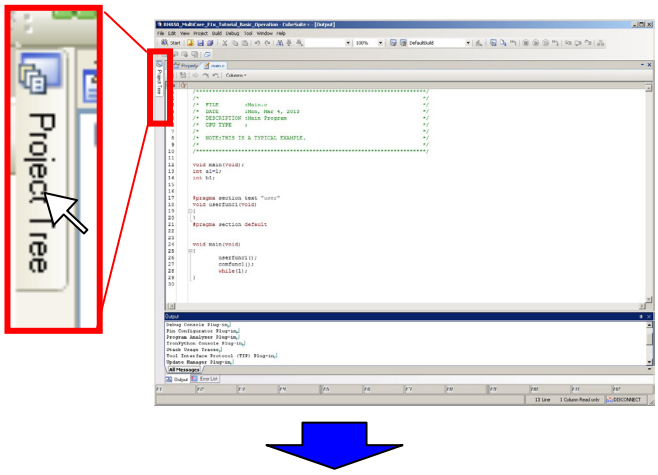


The Project Tree automatically disappears and the tab representing it appears.

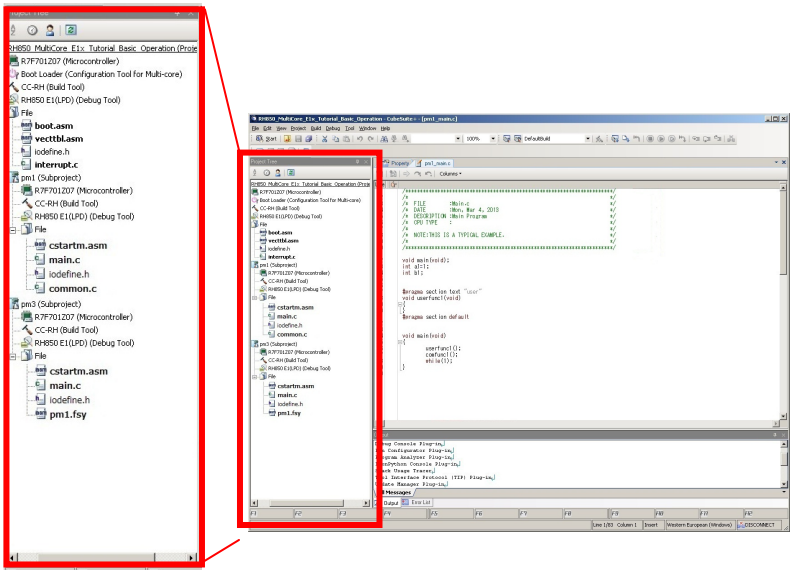


(b) Displaying the hidden panel (example: Project Tree)

Place the pointer on the [Project Tree] tab.



The Project Tree slides out.



Tip

Locations to hide the panels

You can hide the panel in three locations: one to the left of the window, one to the right of the window and one below the window. You can also hide multiple panels in the same location.

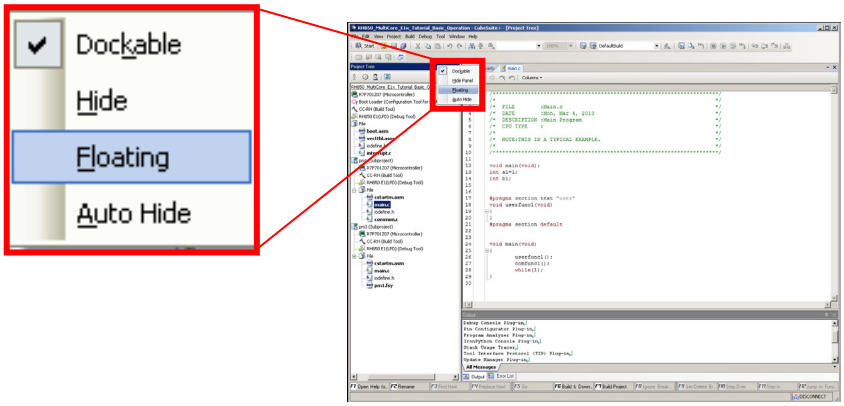
Manipulating Windows

3. Floating

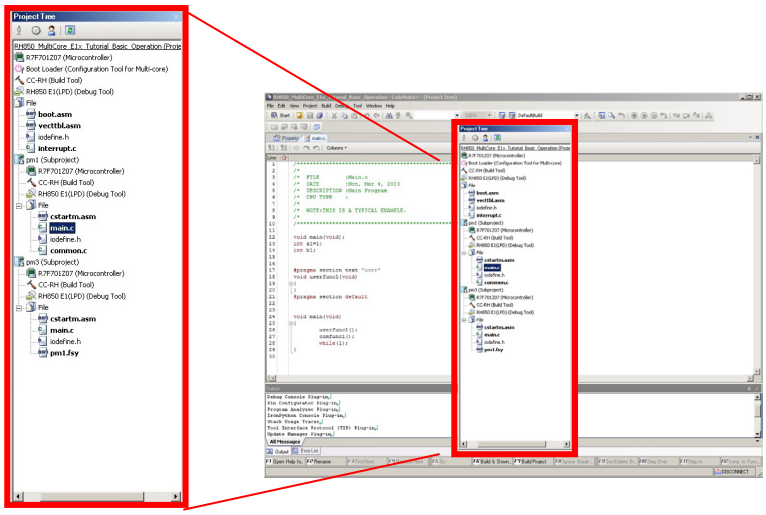
Right-clicking on the title bar and selecting [Floating] from the menu allows you to move the panel at will.

(a) Making the panel float (example: Project Tree)

Right-click on the title bar and select [Floating].



The panel enters the floating state.



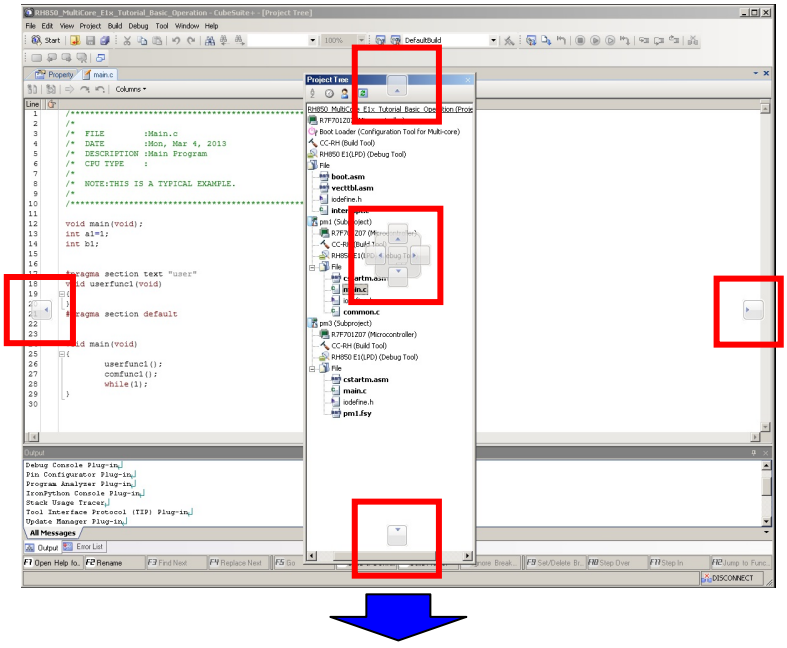
Manipulating Windows

4. Docking

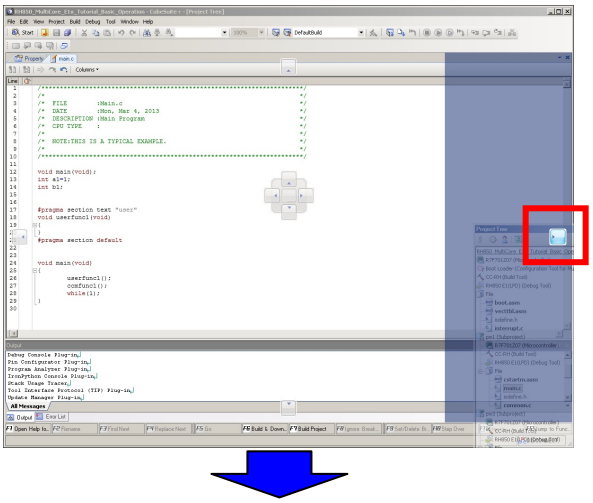
You can attach the floating panel to any of the four sides of another panel such as the main panel. You can easily change the position of the panel by dragging and dropping it to a desired location using a navigation icon.

(a) Moving the panel (example:Project Tree)

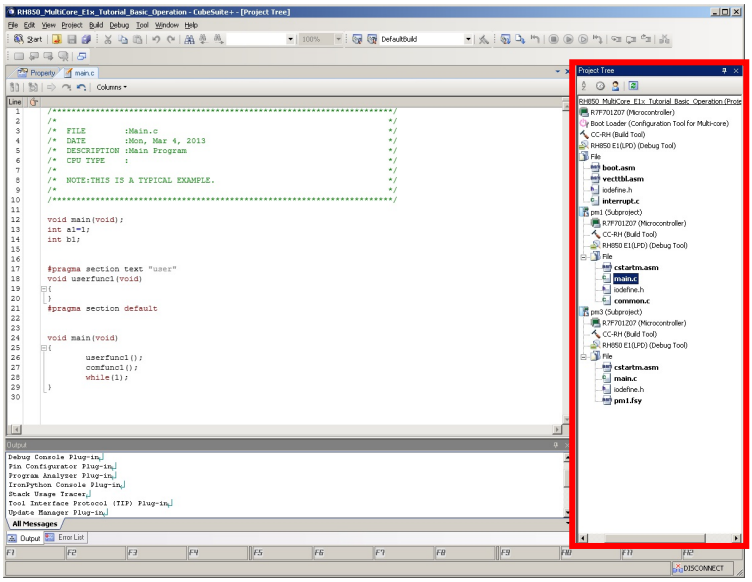
When you drag the floating panel, the navigation icon appears.



Place the pointer on the navigation icon located in the desired destination location and the destination area is highlighted in blue.



Drop the panel there and the Project Tree moves to the desired location (the figure below shows the example of attaching it to the right of the main panel).

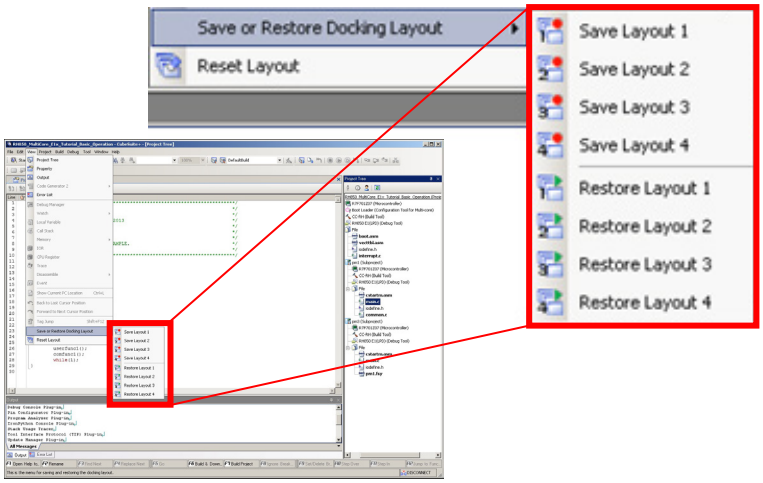


Tip

Saving and restoring layouts

You can save up to four panel layout (panel location information) states for before and after connecting to the debug tool. To do so, from the menu bar, select [View] -> [Save or Restore Docking Layout].

* Becomes a debugging-specific layout only when the debug tool is connected.



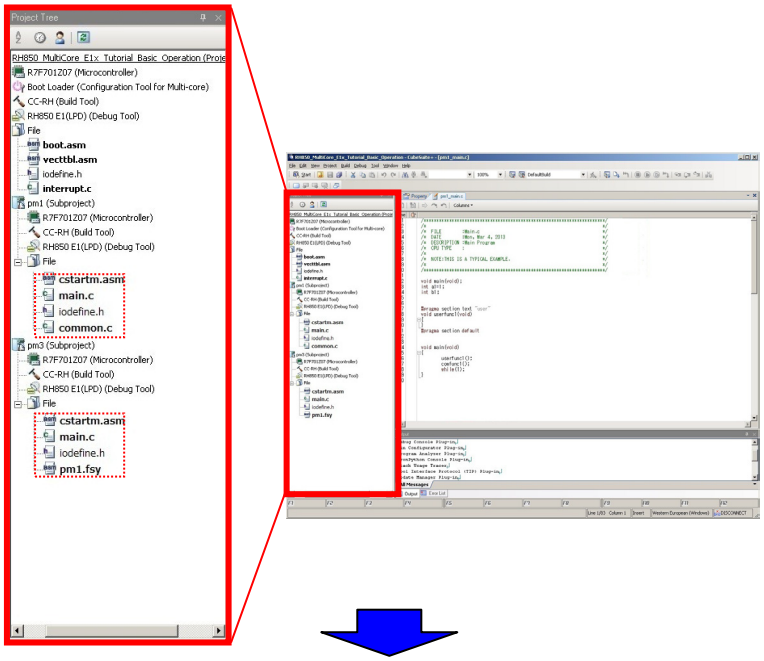
Editing the Program

In this section, you edit the user program.
First, the basic editing method is explained, and then you can edit the program through a simple copy and paste procedure. Edit the program following the steps listed below.

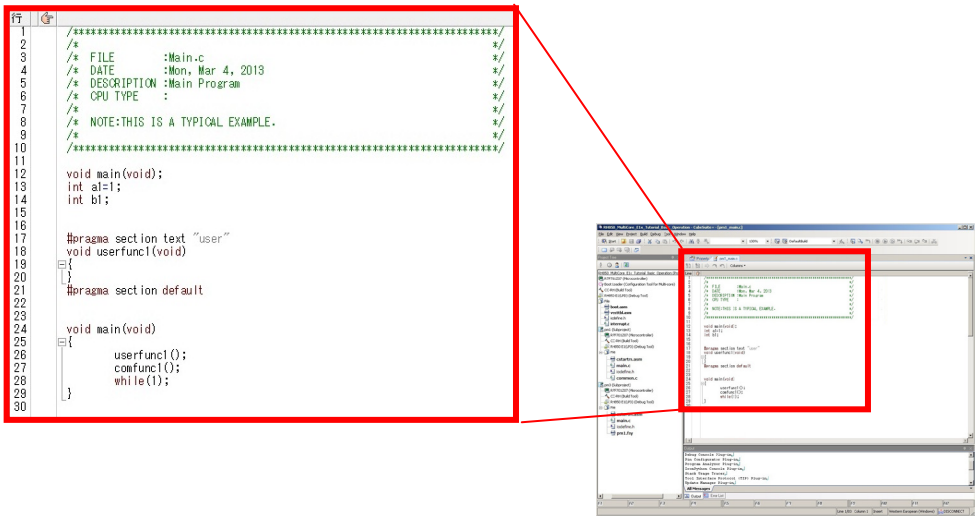
1. How to open a source code file

The following step describes how to open a source code file.

Find the source code file you want to edit in the Project Tree and double-click it.



This displays the source code in the main panel.

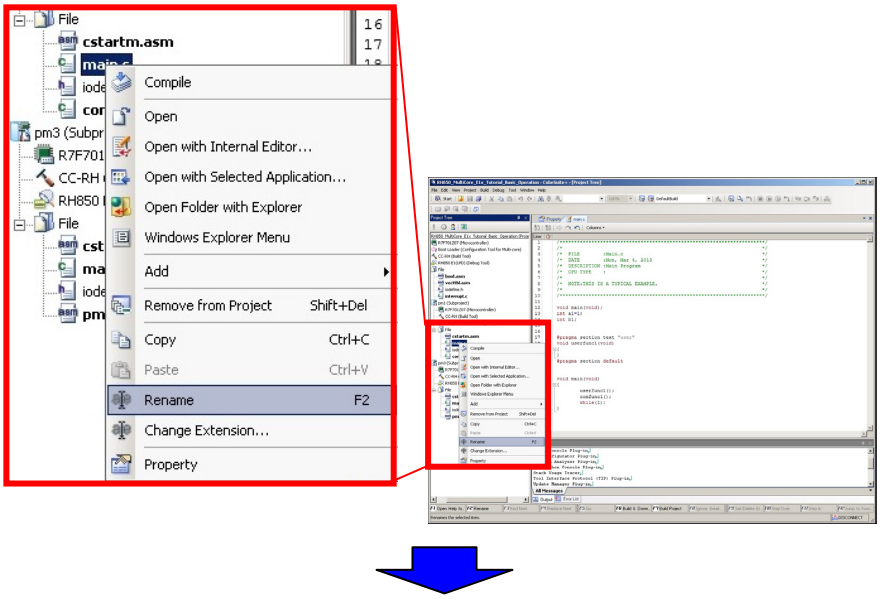


Editing the Program

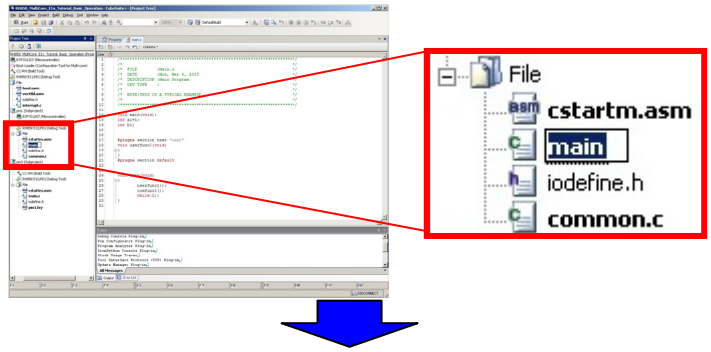
2. Renaming a file

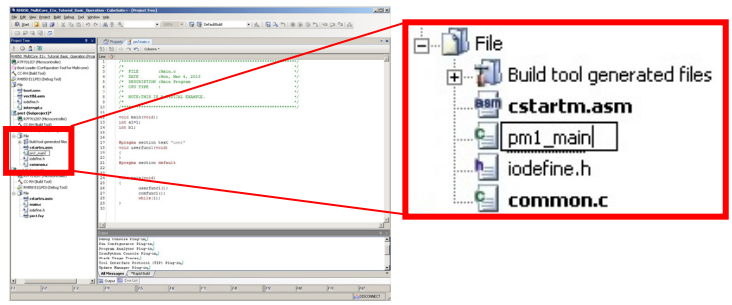
Rename the file following the steps listed below.

Select main.c of pm1 (subproject) in the Project Tree, right-click it to display the pop-up menu, then select "Rename" from the pop-up menu.

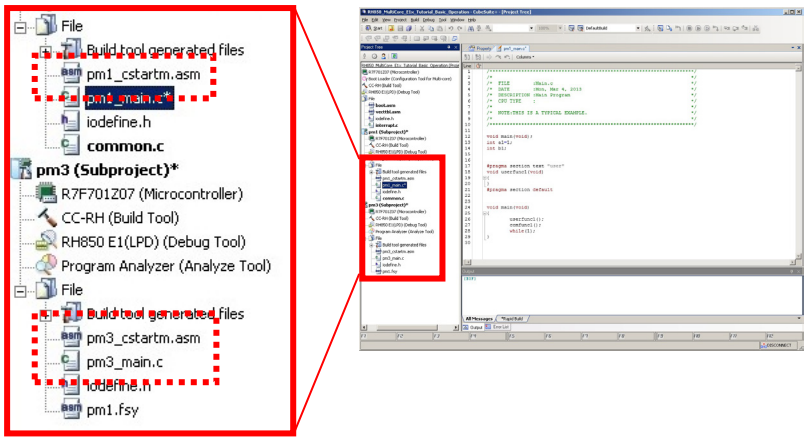


Since the file name can now be edited, change it to pm1_main.





In a similar manner, rename the following files:
cstartm.asm of pm1 (subproject) -> pm1_cstartm.asm
main.c of pm3 (subproject) -> pm3_main.c
cstartm.asm of pm3 (subproject) -> pm3_cstartm.asm



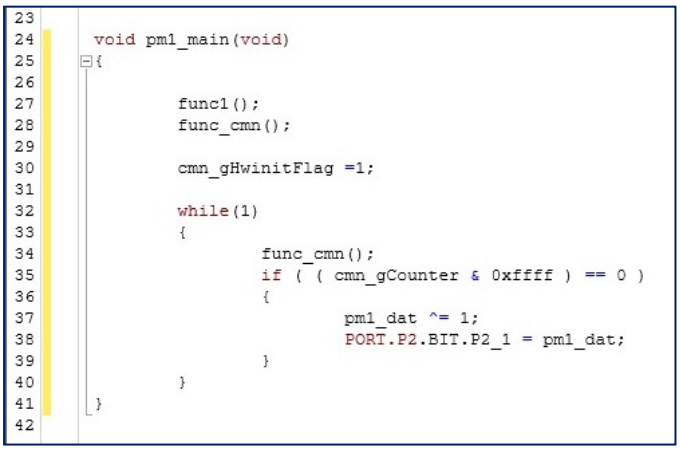
Editing the Program

3. Editing

Edit the program following the steps listed below.

Change the main() function name of pm1_main.c to pm1_main(), and copy the following code and paste it in the pm1_main() function.

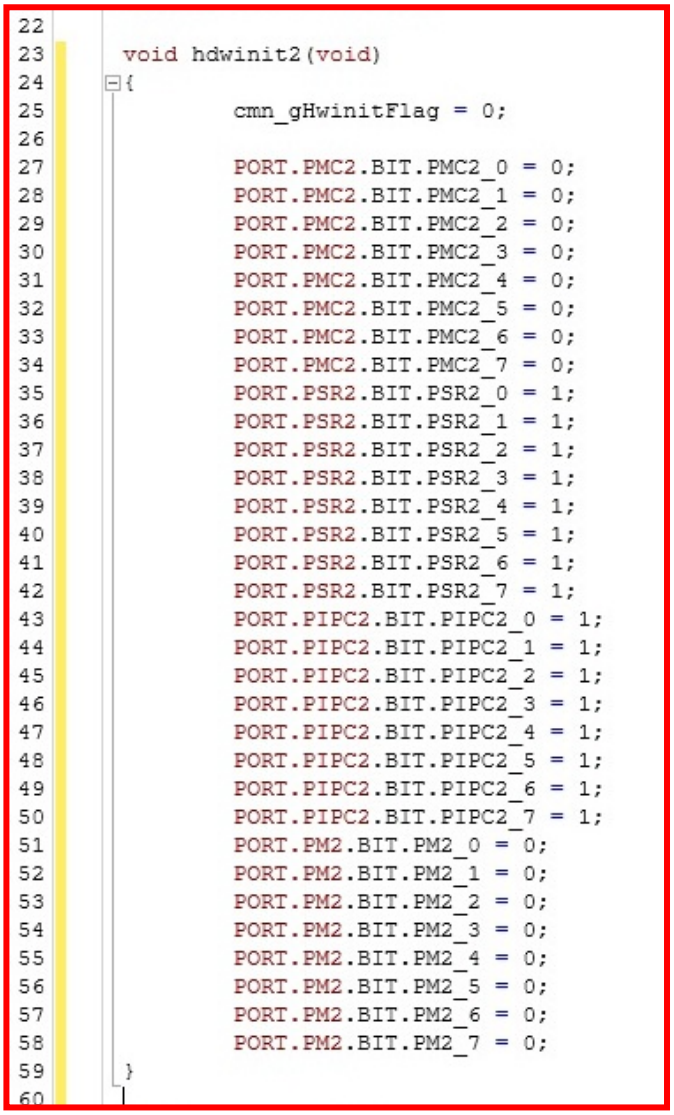
```
void pm1_main(void)
{
                        func1();
                        func_cmn();

                        cmn_gHwinitFlag =1;

                        while(1)
                        {
                                func_cmn();
                                if ( ( cmn_gCounter & 0xffff ) == 0 )
                                {
                                        pm1_dat ^= 1;
                                        PORT.P2.BIT.P2_0 = pm1_dat;
                                }
                        }
}
```



```
23
24      void pm1_main(void)
25      {
26
27              func1();
28              func_cmn();
29
30              cmn_gHwinitFlag =1;
31
32              while(1)
33              {
34                      func_cmn();
35                      if ( ( cmn_gCounter & 0xffff ) == 0 )
36                      {
37                              pm1_dat ^= 1;
38                              PORT.P2.BIT.P2_1 = pm1_dat;
39                      }
40              }
41      }
42
```

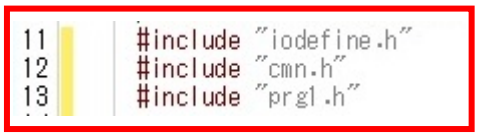
To add the hdwinit2() function to pm1_main.c, copy and paste the following code.

```
void hdwinit2(void)
{
    cmn_gHwinitFlag = 0;


    PORT.PMC2.BIT.PMC2_0 = 0;
    PORT.PMC2.BIT.PMC2_1 = 0;
    PORT.PMC2.BIT.PMC2_2 = 0;
    PORT.PMC2.BIT.PMC2_3 = 0;
    PORT.PMC2.BIT.PMC2_4 = 0;
    PORT.PMC2.BIT.PMC2_5 = 0;
    PORT.PMC2.BIT.PMC2_6 = 0;
    PORT.PMC2.BIT.PMC2_7 = 0;
    PORT.PSR2.BIT.PSR2_0 = 1;
    PORT.PSR2.BIT.PSR2_1 = 1;
    PORT.PSR2.BIT.PSR2_2 = 1;
    PORT.PSR2.BIT.PSR2_3 = 1;
    PORT.PSR2.BIT.PSR2_4 = 1;
    PORT.PSR2.BIT.PSR2_5 = 1;
    PORT.PSR2.BIT.PSR2_6 = 1;
    PORT.PSR2.BIT.PSR2_7 = 1;
    PORT.PIPC2.BIT.PIPC2_0 = 1;
    PORT.PIPC2.BIT.PIPC2_1 = 1;
    PORT.PIPC2.BIT.PIPC2_2 = 1;
    PORT.PIPC2.BIT.PIPC2_3 = 1;
    PORT.PIPC2.BIT.PIPC2_4 = 1;
    PORT.PIPC2.BIT.PIPC2_5 = 1;
    PORT.PIPC2.BIT.PIPC2_6 = 1;
    PORT.PIPC2.BIT.PIPC2_7 = 1;
    PORT.PM2.BIT.PM2_0 = 0;
    PORT.PM2.BIT.PM2_1 = 0;
    PORT.PM2.BIT.PM2_2 = 0;
    PORT.PM2.BIT.PM2_3 = 0;
    PORT.PM2.BIT.PM2_4 = 0;
    PORT.PM2.BIT.PM2_5 = 0;
    PORT.PM2.BIT.PM2_6 = 0;
    PORT.PM2.BIT.PM2_7 = 0;

}
```



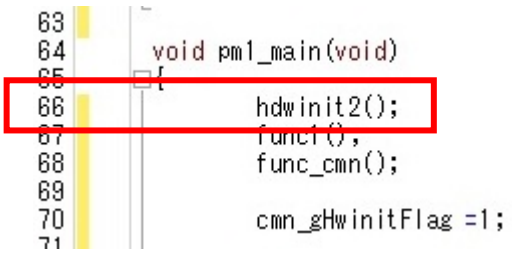
To add include statements to pm1_main.c, copy and paste the following code.

```
#include "iodefine.h"
#include "cmn.h"
#include "prg1.h"
```



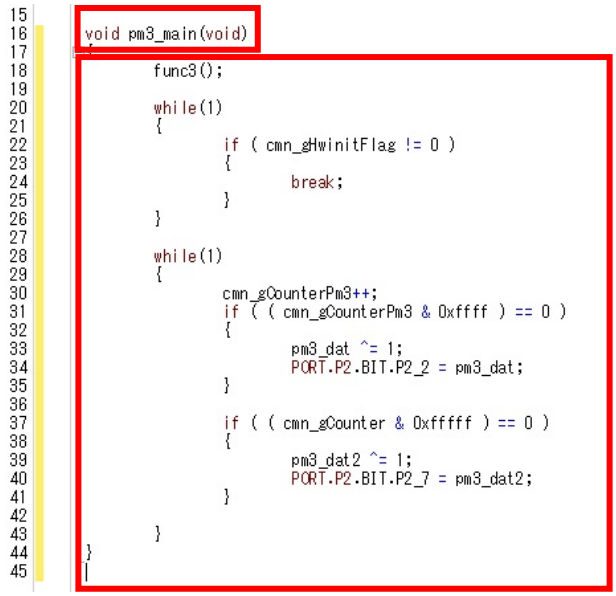
To add the hdwinit2() call to the pm1_main() function, copy and paste the following code.

```
hdwinit2();
```



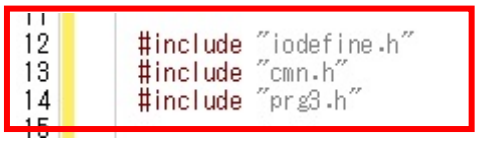
Change the main() function name of pm3_main.c to pm3_main(), and copy the following code and paste it in the pm3_main() function.

```
void pm3_main(void)
{
        func3();

        while(1)
        {
                if ( cmn_gHwinitFlag != 0 )
                {
                        break;
                }
        }

        while(1)
        {
                cmn_gCounterPm3++;
                if ( ( cmn_gCounterPm3 & 0xffff ) == 0 )
                {
                        pm3_dat ^= 1;
                        PORT.P2.BIT.P2_2 = pm3_dat;
                }
                if ( ( cmn_gCounter & 0xfffff ) == 0 )
                {
                pm3_dat2 ^= 1;
                PORT.P2.BIT.P2_7 = pm3_dat2;
                }

         }
}
```

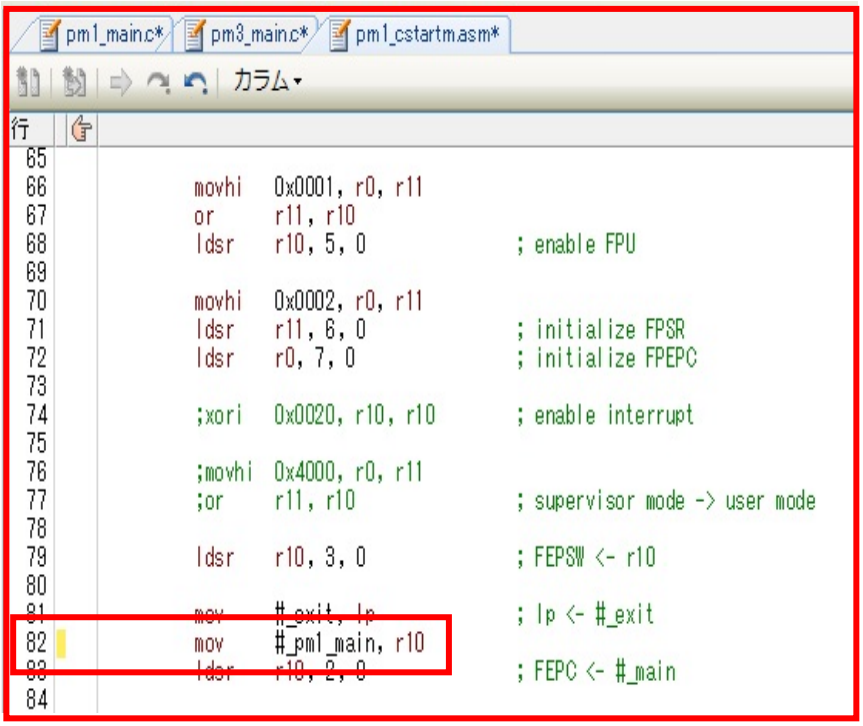


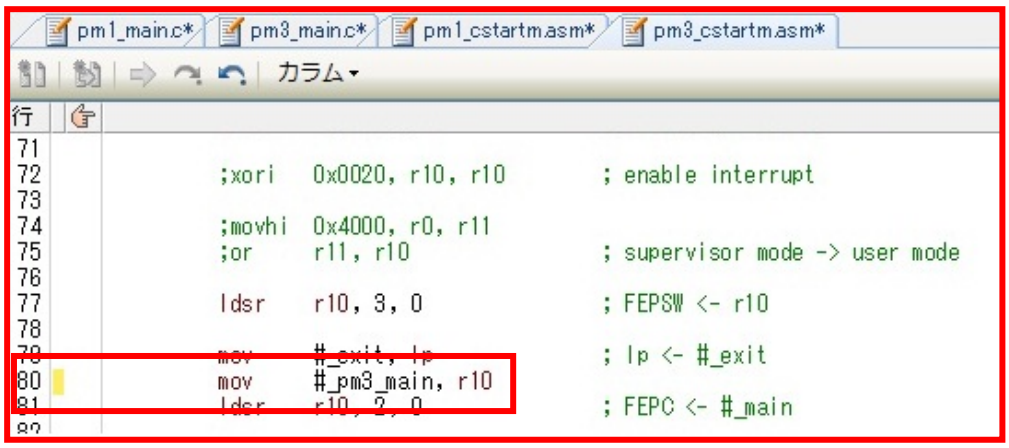
To add include statements to pm3_main.c, copy and paste the following code.

```
#include "iodefine.h"
#include "cmn.h"
#include "prg3.h"
```

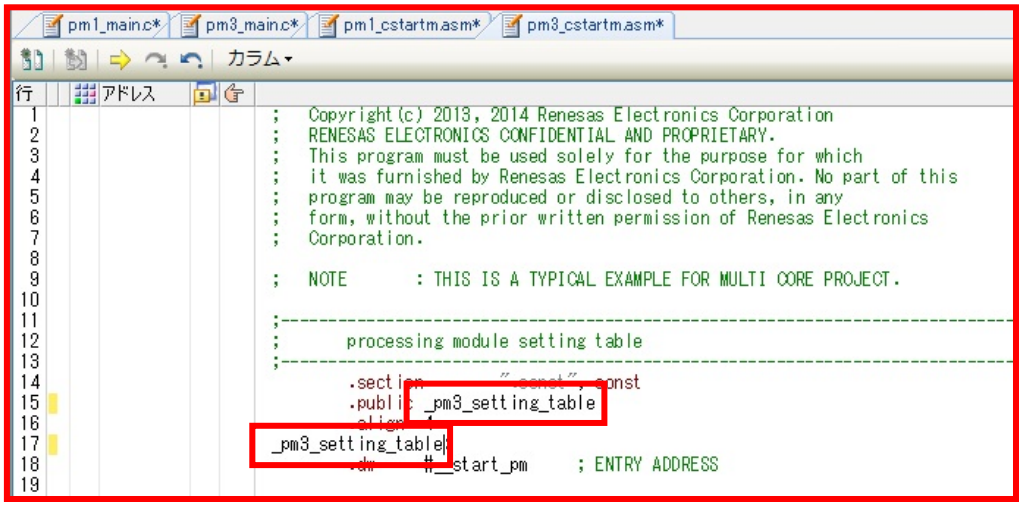


Change the branch destination of pm1_cstartm.asm to pm1_main() and the branch destination of pm3_cstartm.asm to pm3_main().





Change the label name "_pm1_setting_table" of pm3_cstartm.asm to "_pm3_setting_table".

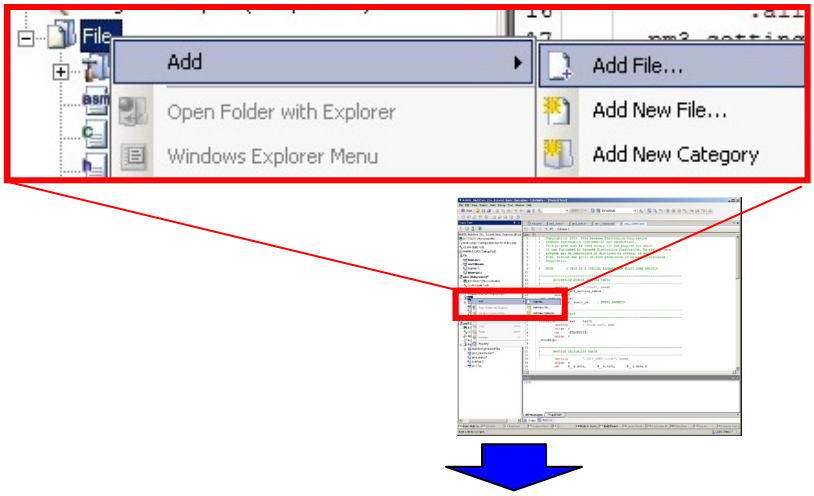

pm1_main.c*
pm3_main.c*
pm1_cstartm.asm*
pm3_cstartm.asm*
カラム
行
アドレス
; Copyright(c) 2013, 2014 Renesas Electronics Corporation
; RENESAS ELECTRONICS CONFIDENTIAL AND PROPRIETARY.
; This program must be used solely for the purpose for which
; it was furnished by Renesas Electronics Corporation. No part of this
; program may be reproduced or disclosed to others, in any
; form, without the prior written permission of Renesas Electronics
; Corporation.
; NOTE : THIS IS A TYPICAL EXAMPLE FOR MULTI CORE PROJECT.
; processing module setting table
.section "const", const
.public _pm3_setting_table
.align 4
_pm3_setting_table:
.dw #__start_pm ; ENTRY ADDRESS

Editing the Program

4. Adding files

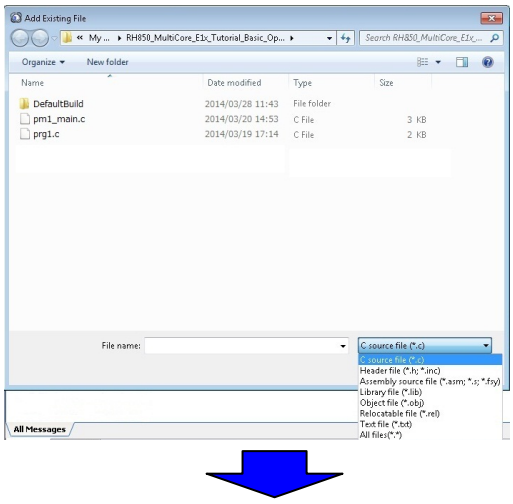
Add programs following the steps listed below.

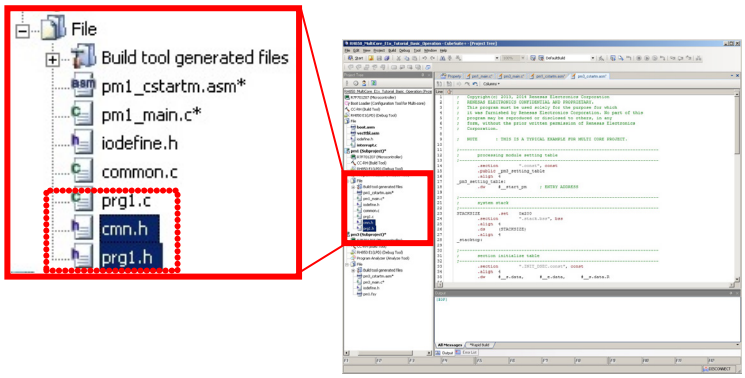
Select a file of pm1 (subproject) in the Project Tree, right-click it to display the pop-up menu, and then select "Add File..." from the pop-up menu.



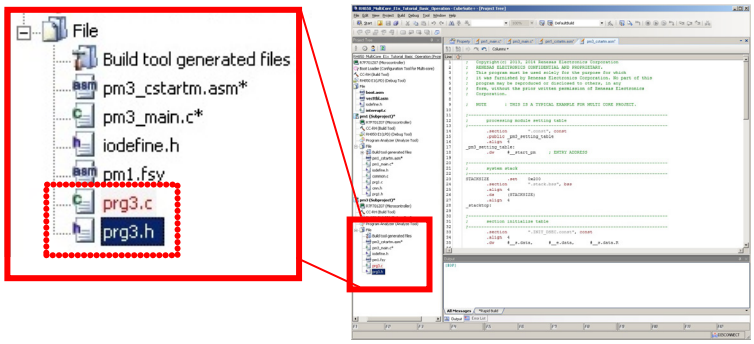
This opens the Add Existing File dialog box. Add the files in the following folder.

<Folder created when loading the sample project>¥
RH850_MultiCore_E1x_Tutorial_Basic_Operation¥pm1
- prg1.c
- prg1.h
- cmn.h





In a similar manner, add the following files to pm3 (subproject).
<Folder created when loading the sample project>¥
RH850_MultiCore_E1x_Tutorial_Basic_Operation¥pm3
- prg3.c
- prg3.h

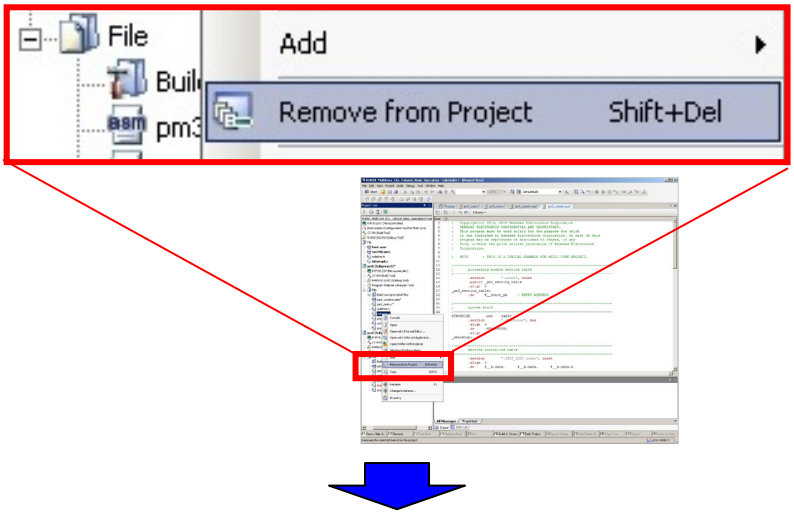


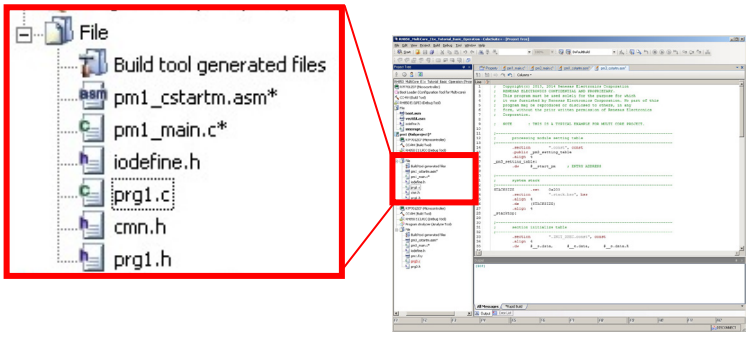
Editing the Program

5. Deleting files

Delete programs following the steps listed below.

Select common.c of pm1 (subproject) in the Project Tree, right-click it to display the pop-up menu, then select "Remove from Project" from the pop-up menu.



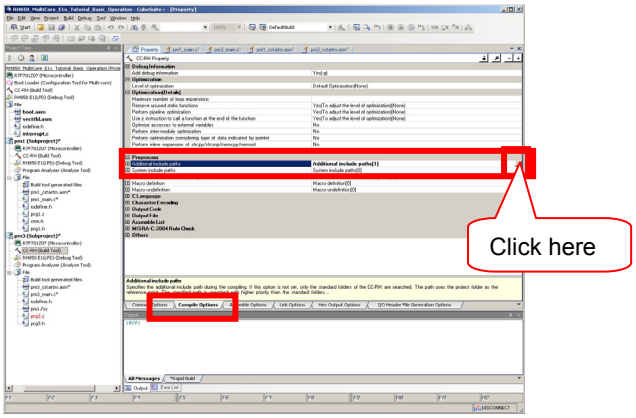


Editing the Program

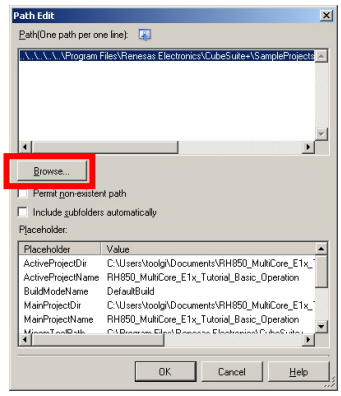
6. Changing the property of the compiler (CC-RH)

Change the property of CC-RH following the steps listed below.

Display the property of CC-RH of pm3 (subproject) in the Project Tree, then click the Add button to add an additional include path as a compile option.

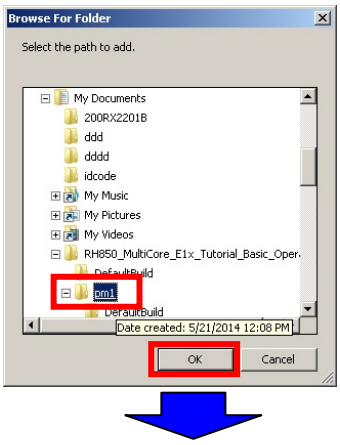


This opens the Path Edit dialog box. Click the Browse button.

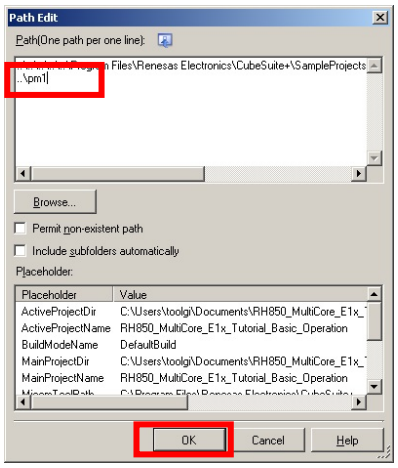


This opens the Browse For Folder dialog box. Select the pm1 folder shown below.

<Folder created when loading the sample project>¥
RH850_MultiCore_E1x_Tutorial_Basic_Operation¥pm1



After confirming that the folder has been added, click the OK button.

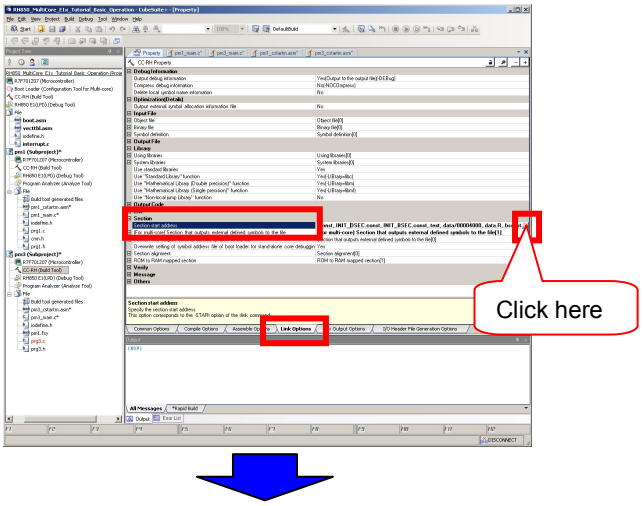


Editing the Program

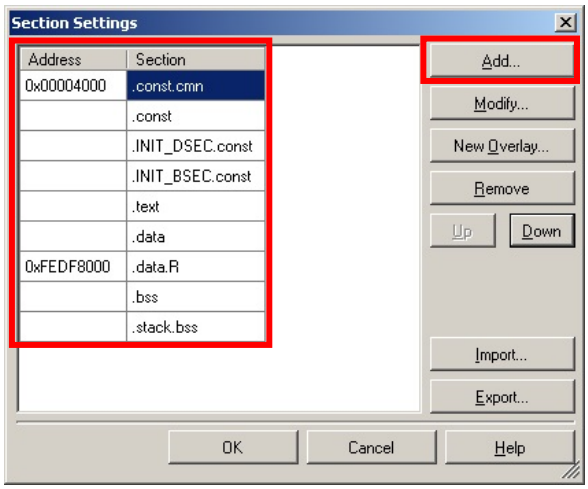
7. Editing the section start address

Change the section start address following the steps listed below.

Display the property of CC-RH of pm3 (subproject) in the Project Tree, then click the Edit button at the section start address of the section group as a link option.



This opens the Section Settings dialog box. Click the Add button to add the .const.cmn section and make settings and edit settings as shown below.



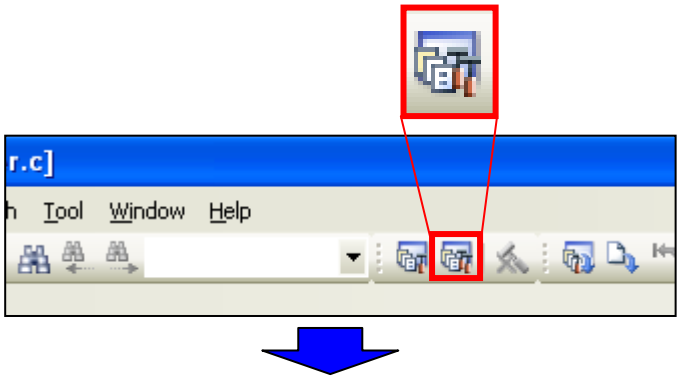
Rebuilding a Program

Rebuild the program of the loaded sample project.

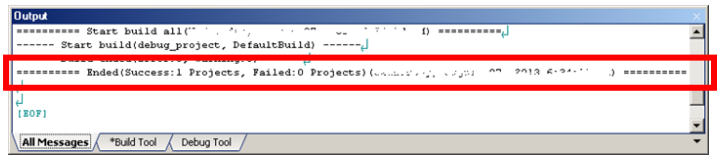
1. Building a project

In this step, rebuild the program of the loaded sample project.

Click the [Rebuild Project.] button.



Check to see if the rebuild process has been completed correctly. If the build process has been completed correctly, a load module file is created.



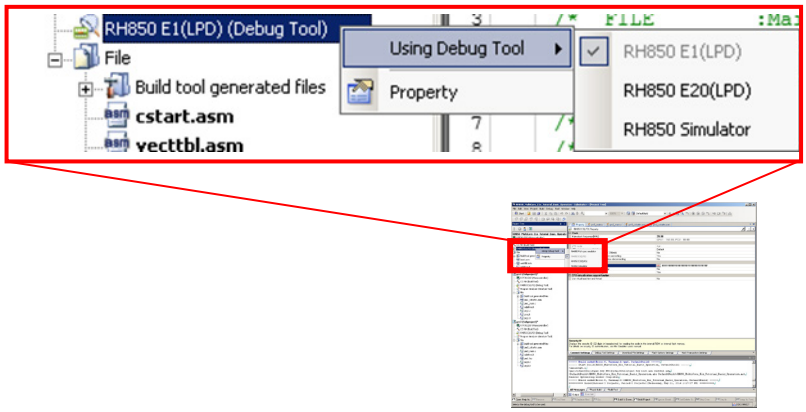
Connecting a Debugger and Downloading

In this section, you debug the program using the E1. First, make preparations for debugging.

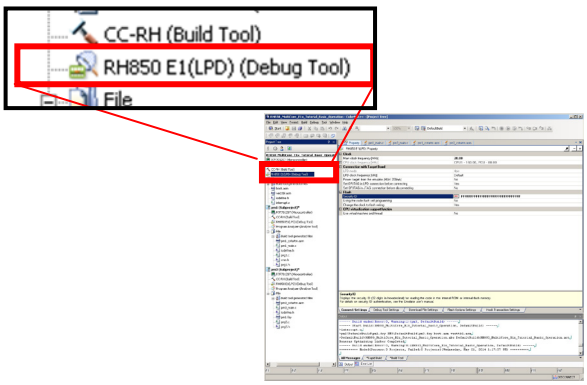
1. Selecting a debug tool

In this step, select [RH850 E1(LPD) (Debug Tool)] as a debug tool to be used.

Right-click on the Debug Tool in the Project Tree and select [Using Debug Tool] -> [RH850 E1(LPD)].



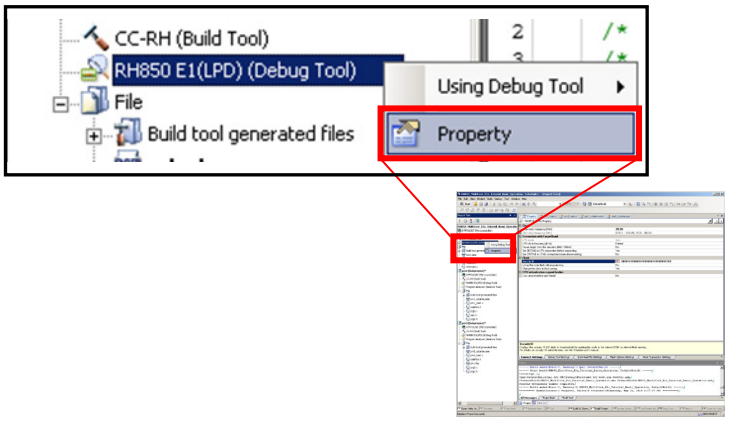
[RH850 E1(LPD) (Debug Tool)] is selected as a debug tool.



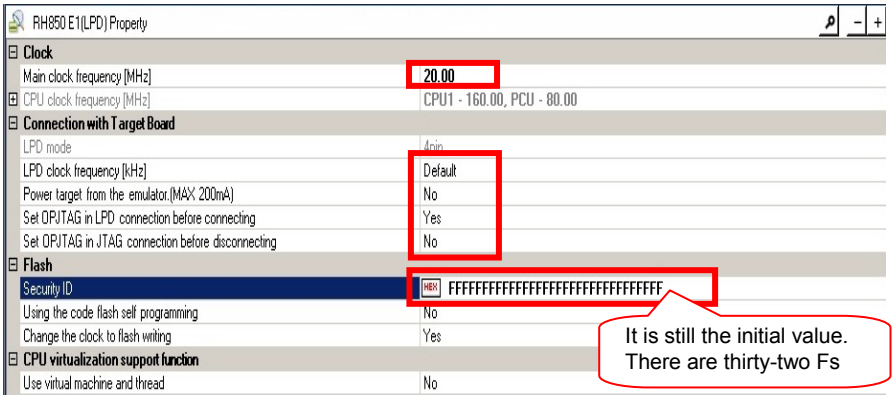
Connecting a Debugger and Downloading

2. Setting E1 for connection to the target board

Right-click the debug tool in the Project Tree to select [Property].



Make settings as shown below in the [Connection with Target Board] tab.



Tip

About the security ID

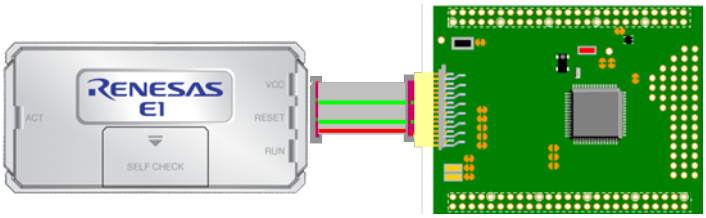
The 128-bit ID code can be written to the microcontroller so that the flash memory contents are not read by an unauthorized user. If the code that is input by the user when the debugger is started does not match the ID code written to the microcontroller, flash memory cannot be accessed. Settings should be made by a flash programmer. When a blank product (all flash memory contents are erased) is used, only F should be input for the security ID.

Connecting a Debugger and Downloading

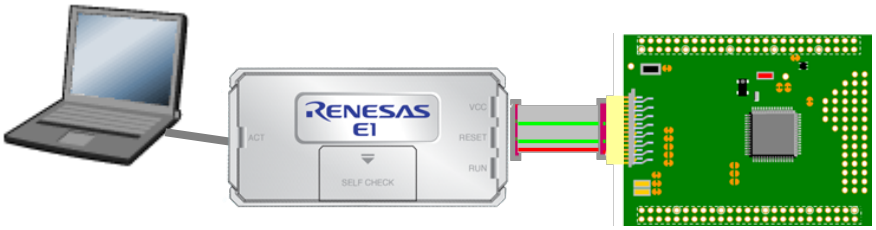
3. Connecting E1

In this step, on-chip debugging is performed using E1.

Connect E1 to the RH850/E1x board (MSRHQ176CP01). Align pin 1 of the connector.



Connect E1 to the PC. ("Found New Hardware Wizard" appears when E1 is connected for the first time. Select "Install software automatically" and install the USB driver following the instructions.)



Turn on the power of MSRHQ176CP01.

Tip

About option bytes

In flash memory, there is an extended area (option bytes) for holding data specified by the user for various purposes. In the RH850/E1x-FCC1 microcontroller, not only are settings for the debugging interface made, but settings for WDT-related features and the operating mode and startup area of the microcontroller are to be made. When the program of this tutorial is used, set the OPBT0 register to H'53FFFFED and the OPBT2 register to H'BFFFFFFF.

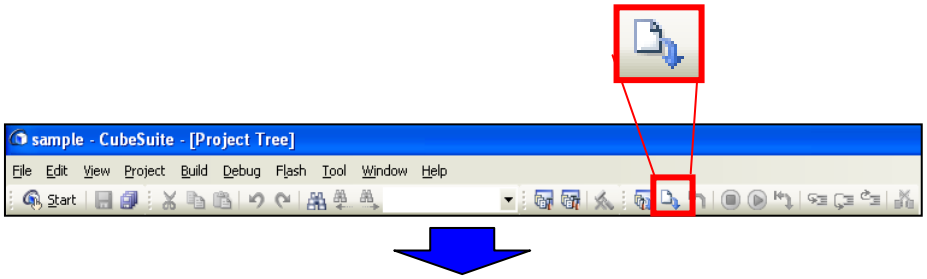
Connecting a Debugger and Downloading

4. Downloading a load module file to E1

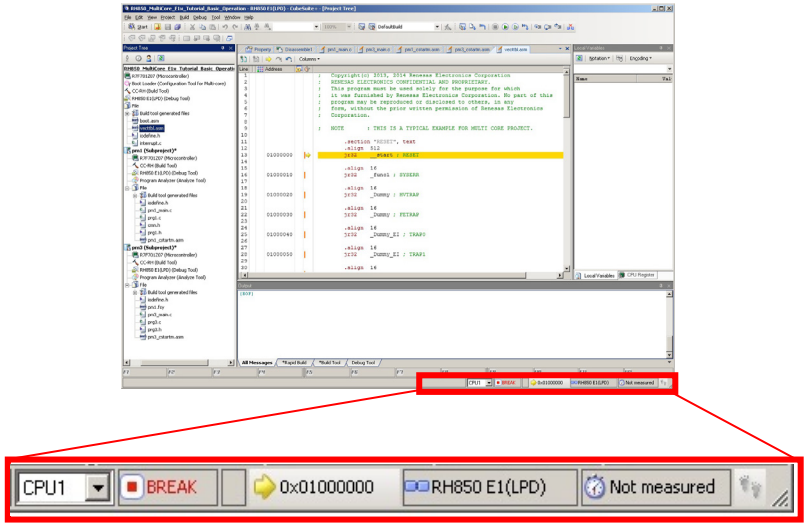
In this step, download the load module file generated by the build process to the target microcontroller.

When download is complete, the program can be executed.

Click the Download button in the menu.



The status bar of the main window changes as shown below.

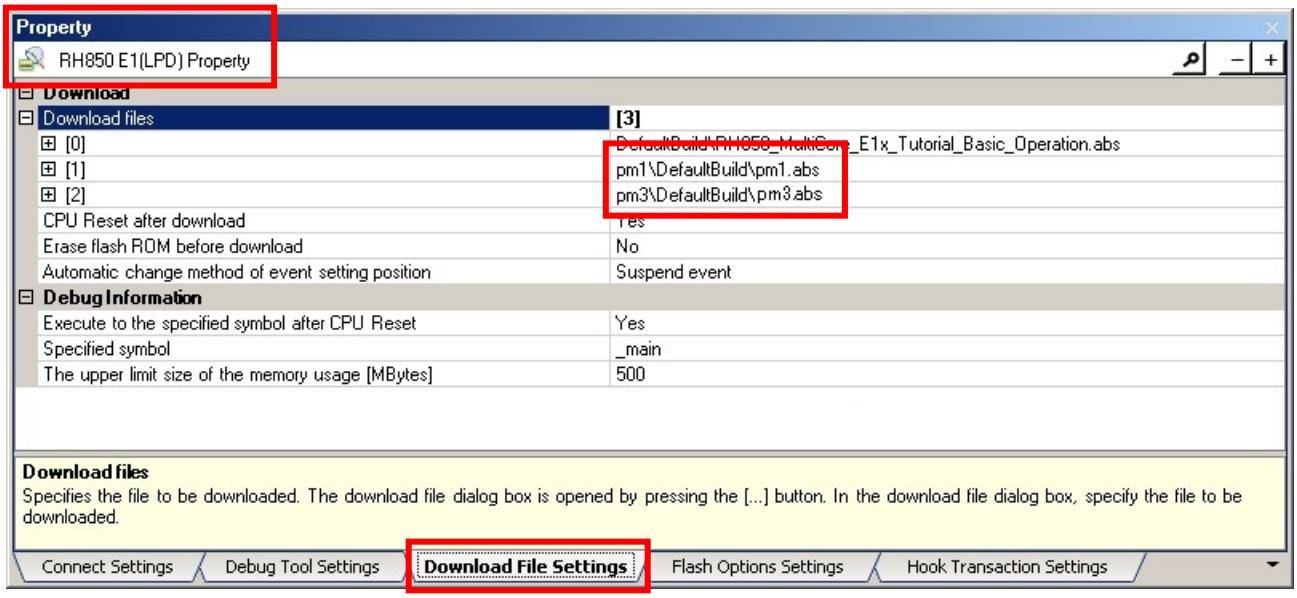


Tip

Registering load module files

Load module files generated in subprojects need to be registered as load module files subject to download.

Load module files can be registered in the [Download File Settings] tab of the Property panel of the debug tool.


Property
RH850 E1(LPD) Property
Download
Download files
[3]
[0]
[1]
[2]
pm1\DefaultBuild\pm1.abs
pm3\DefaultBuild\pm3.abs
CPU Reset after download
Yes
Erase flash ROM before download
No
Automatic change method of event setting position
Suspend event
Debug Information
Execute to the specified symbol after CPU Reset
Yes
Specified symbol
_main
The upper limit size of the memory usage [MBytes]
500
Download files
Specifies the file to be downloaded. The download file dialog box is opened by pressing the [...] button. In the download file dialog box, specify the file to be downloaded.
Connect Settings
Debug Tool Settings
Download File Settings
Flash Options Settings
Hook Transaction Settings

Switching Cores

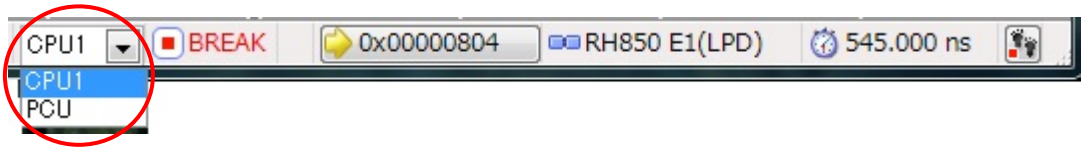
Now that download of the load module file to the target is complete, let's switch the core to be debugged.

1. Switching the core

There are two methods for switching the core to be debugged.

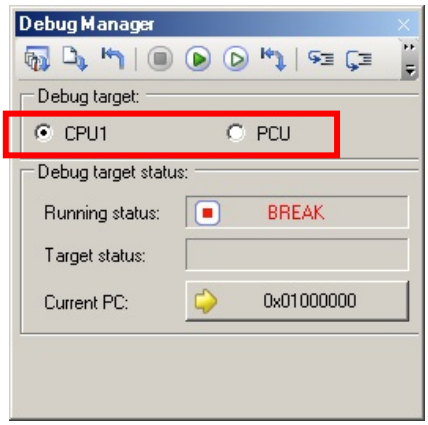
a. Switch the core from the status bar

The core can be switched using the drop-down list on the status bar of the main window.



b. Switch the core from the Debug Manager panel

Selecting [View] menu -> [Debug Manager] opens the Debug Manager panel. The core can be switched in the Debug Manager panel.



Here, the state of CPU1 being selected should not be changed.

Tip

Core to be debugged

Running or stopping of the program cannot be performed by only one core running or stopping the program.
Running or stopping of the program must be performed by both cores operating in a coordinated manner.
When CPU1 is set as the core to be debugged, referencing or changing memory by CubeSuite+ is effective only for CPU1. To perform operations for the other core, the core to be debugged has to be switched.

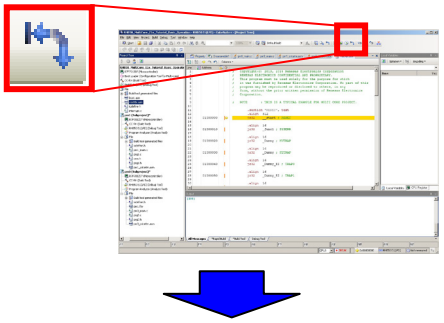
Running and Stopping the Program

Now that download of the load module file to the target is complete, the program can be executed. First, run and stop the program.

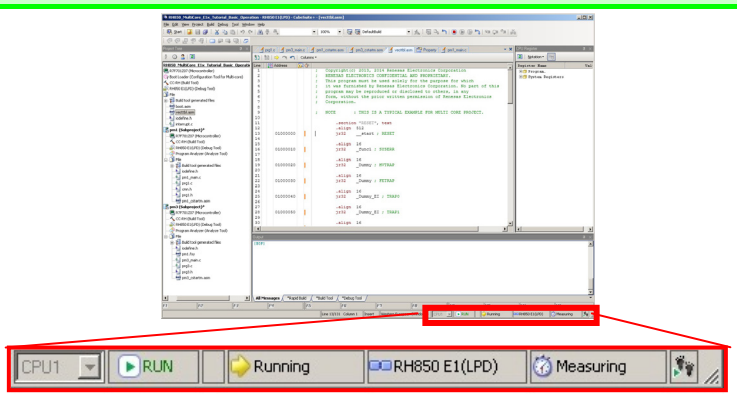
1. Running the program

Run the program after resetting the CPU.

Click the [Restart] button in the menu.



This runs the program and displays [RUN] in the status bar.



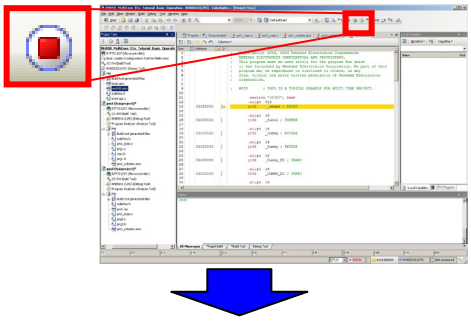
LED9 and LED10 on MSRHQ176CP01 light alternately.

Running and Stopping the Program

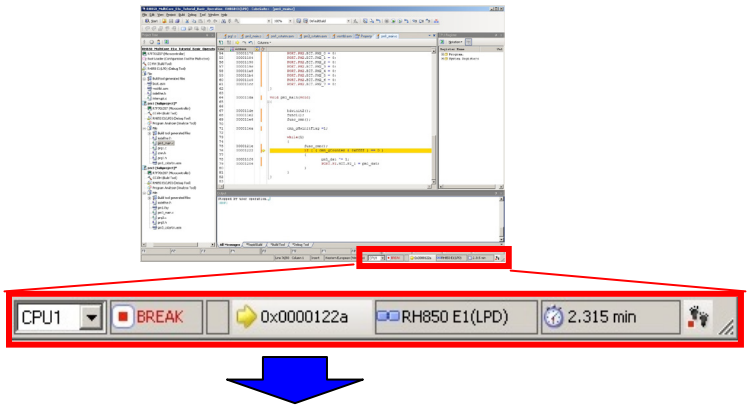
2. Stopping the program

In this step, stop the program.

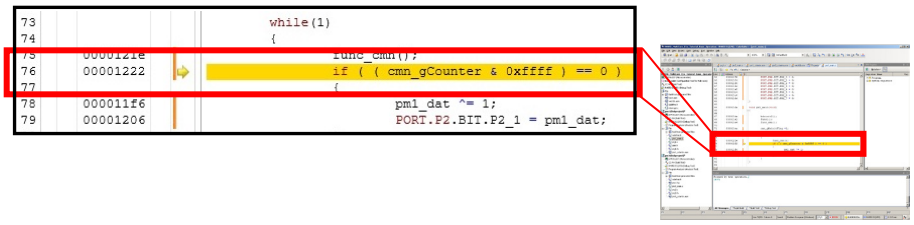
Click the Stop button.



This stops the program and displays [BREAK] in the status bar.



The source code line where the program stopped (current position of the program counter (PC)) is highlighted in yellow.

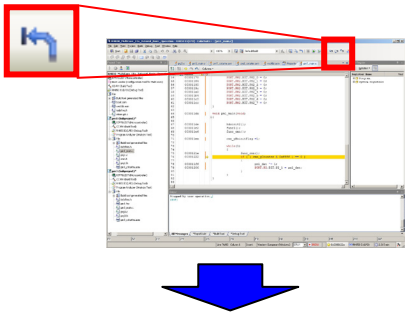


Running and Stopping the Program

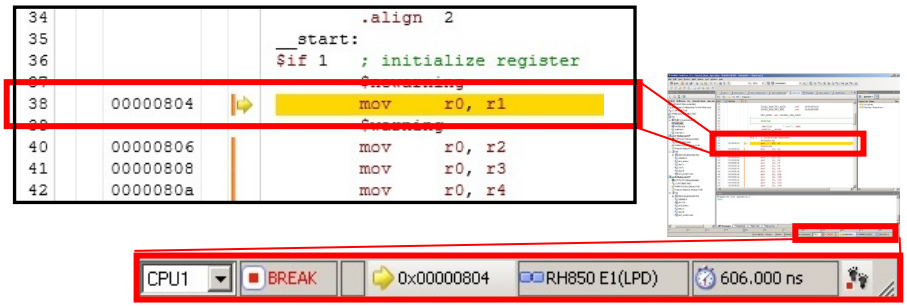
3. Resetting the program

In step 1, you have reset and run the program using a single button, but you can also carry out the reset operation independently.

Click the Reset button.



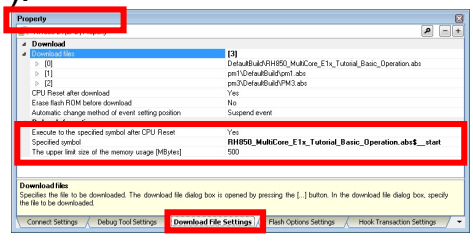
The program is reset and the program counter (PC) goes back to the start of the program.



Tip

About the program counter

The program counter (PC) is a control register that holds information on the next program address. When the RH850/E1x-FCC1 microcontroller generates a reset signal, 00000000H is set in the PC for user mode and 01000000H is set in the PC for user boot mode. In the program of this tutorial, since the program is set to run up to the __start function after a reset, the program enters the break state after executing the __start function. Such kind of behavior can be changed in the [Download File Settings] tab of the Property panel of RH850 E1(LPD).

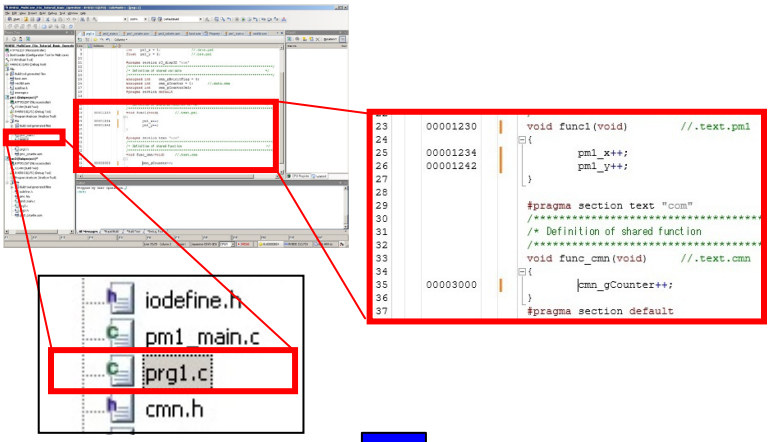


Referring to a Variable

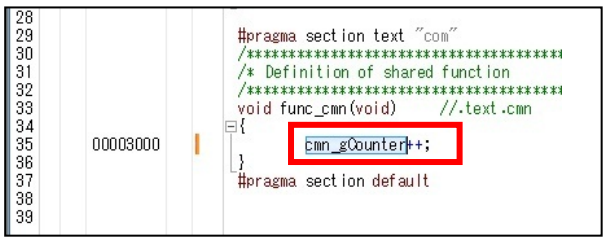
Watch function

By registering a variable as a watch-expression, it is possible to display the value of that variable. Here, register the two shared variables (variables which are located in a shared area that can be referenced from both the CPU1 core and PCU core) "cmn_gCounter" and "cmn_gCounterPm3" as watch-expressions and confirm that the values of the variables are incremented. cmn_gCounter and cmn_gCounterPm3 are variables that are incremented by the CPU1 core and PCU core, respectively, while the program is running. It is possible to confirm that the count values at a break differ because the execution speed of the CPU1 core and PCU core are different.

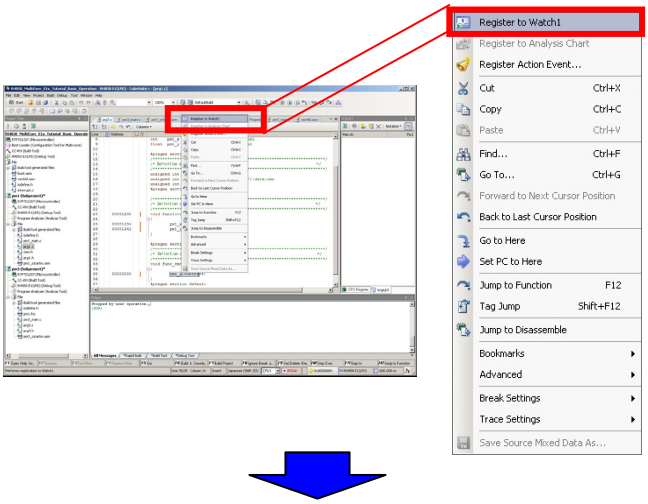
In the Project Tree, double click "prg1.c" to display the source code file.



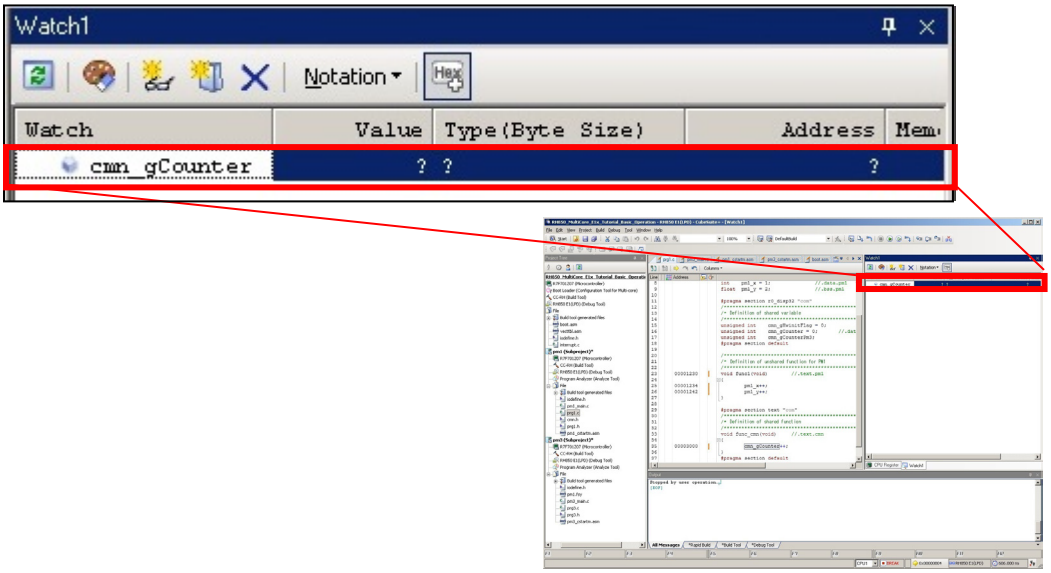
Select variable "cnm_gCounter" in the source code.



While "cnm_gCounter" is selected, right-click the mouse and select [Register to Watch1] from the menu.

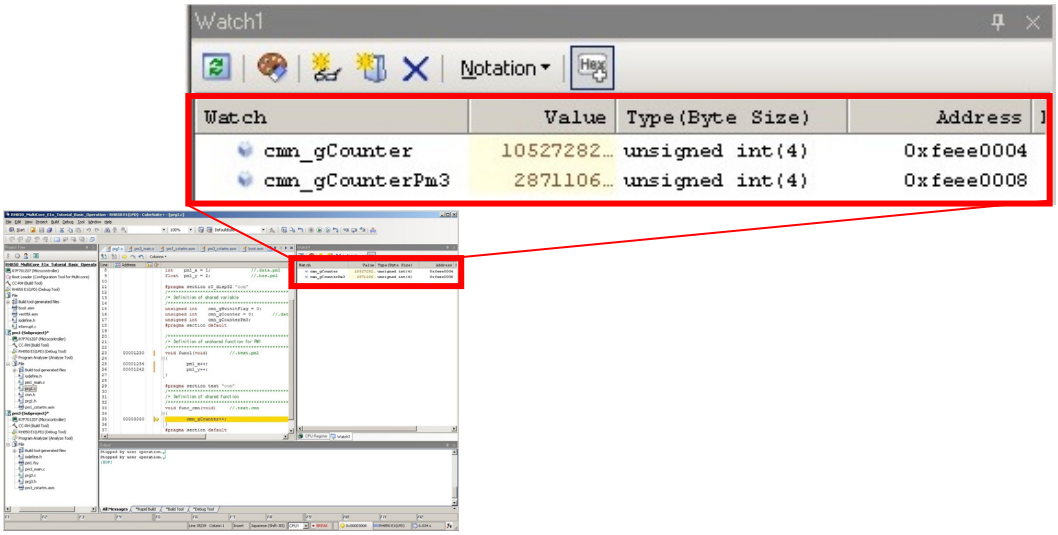


This opens the Watch panel where you can confirm that the variable is now registered. The current value of "cnm_gCounter" is ?.



In a similar manner, register "cnm_gCounterPm3" in the Watch panel from pm3_main.c.

Click the [Restart] button in the menu. After several seconds have passed, click the [Stop] button.



The difference in the operating frequency of the CPU1 core and PCU core cause the execution count of cmn_gCounter and cmn_gCounterPm3 to differ. The difference in the execution count can be confirmed by the fact that LED9 turns on and off at a high speed whereas LED10 turns on and off at a lower speed than LED9.

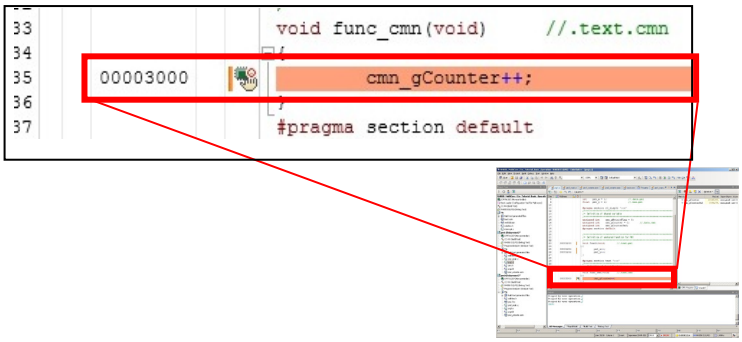
Setting a Breakpoint

Setting a breakpoint

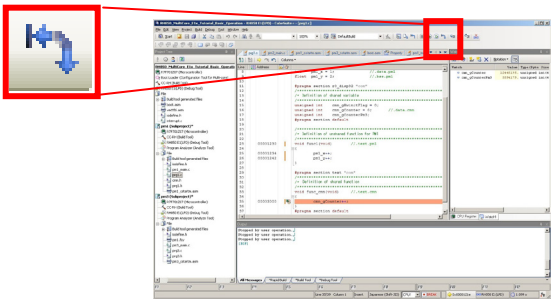
If you want to stop the program at a specific position intentionally in the source code, you can break the program before executing the instruction at the specified address ("before execution" break) by setting a breakpoint.

Let's see how the variable (cnm_gCounter) that was registered as a watch-expression changes by running the program and causing it to break.

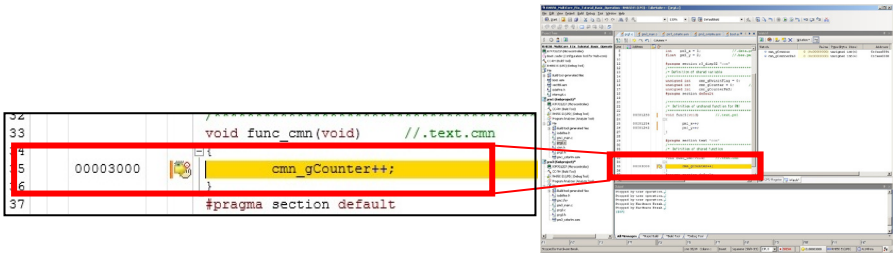
Click the empty column to the left of the desired line in the source code as shown below. A hardware break is set and the line is highlighted in red.



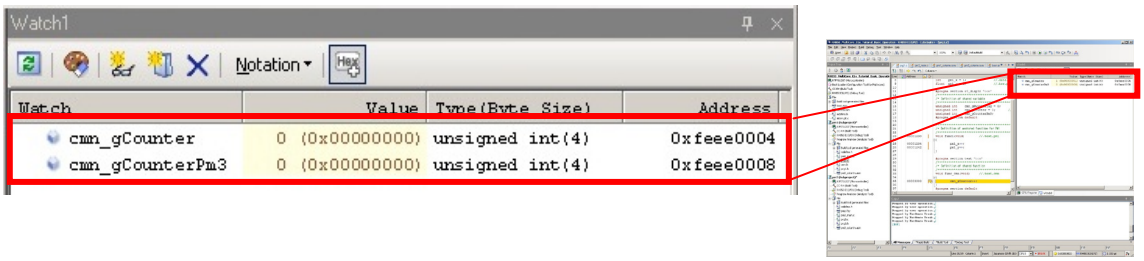
Click the [Restart] button in the menu.



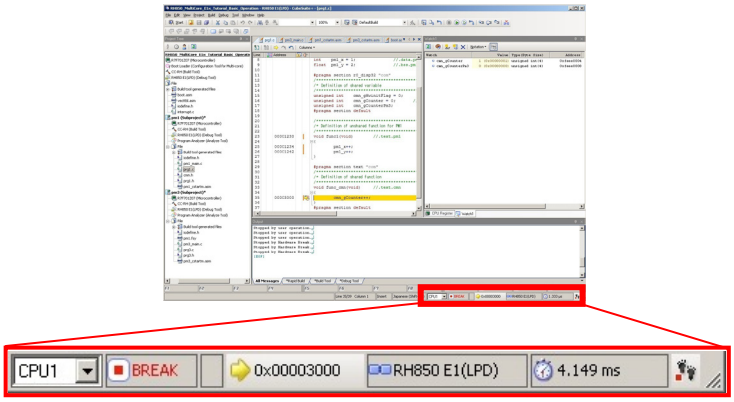
The program breaks at the line where the break has been set and the breakpoint line is highlighted in yellow.



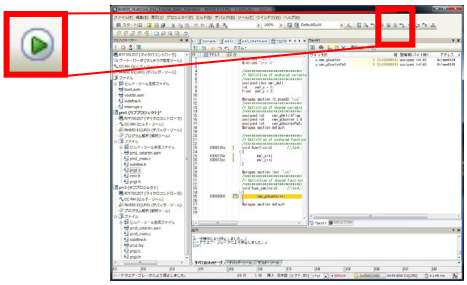
See the Watch panel and confirm that the value of [cnm_gCounter] is counted up to 0x0.



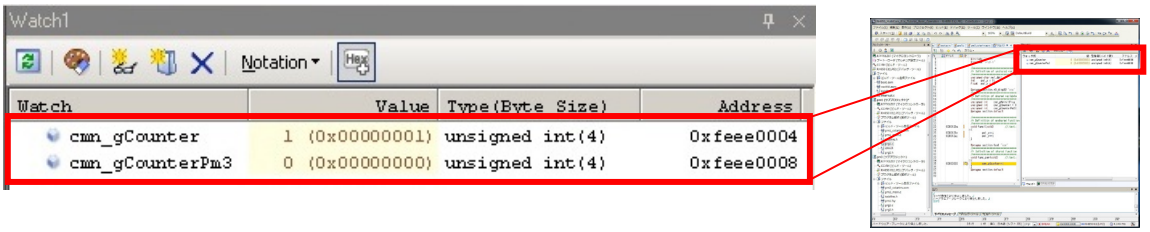
The status bar shows [BREAK] and the PC value at the break.



Click the [Go] button in the menu.



The program breaks again at the line where the break has been set. See the Watch panel and confirm that the value of [cnm_gCounter] is counted up to 0x1.

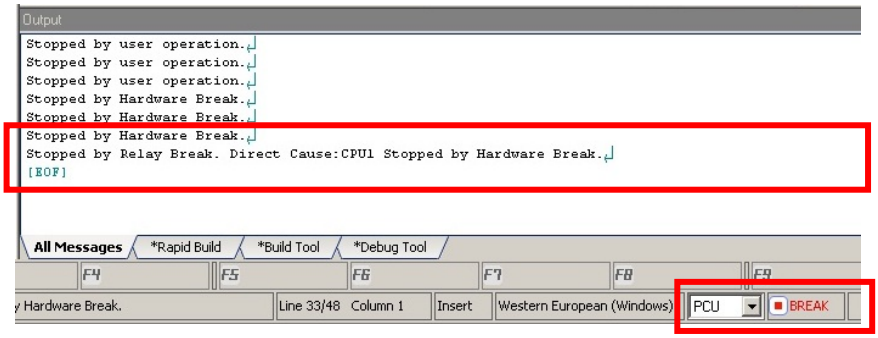


Tip

Break in a multi-core device

Normally, the position where a break occurred in the program is displayed when a break occurs. In a multi-core device, if a break was caused by a break source of another core (core not being debugged), a break occurs in the target core (core being debugged) at an address where no break condition has been set. The break source can be checked in the Output panel.

In the example below, the Output panel shows that a break (relay break) in another core is the break source.



Acquiring the Execution History

Acquiring the execution history

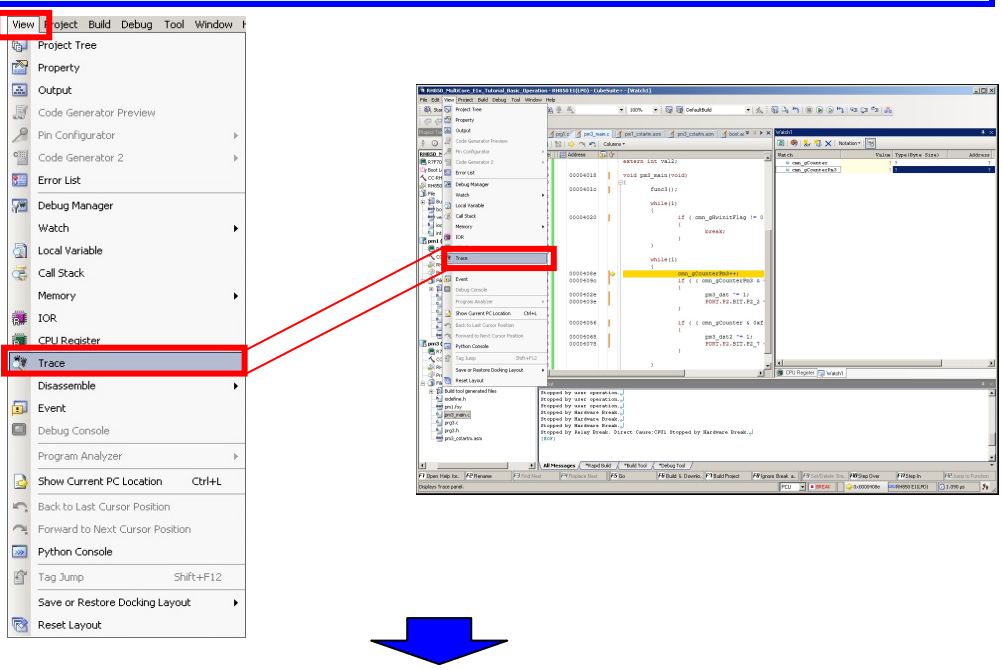
In general, the execution history of the program is called trace information. If a program gets out of control, it is extremely difficult to investigate the cause only from the memory contents or stack information after the runaway occurred. However, using the trace function and analyzing the acquired trace information enables the process until the runaway occurred to be directly investigated.

Collecting the Execution History

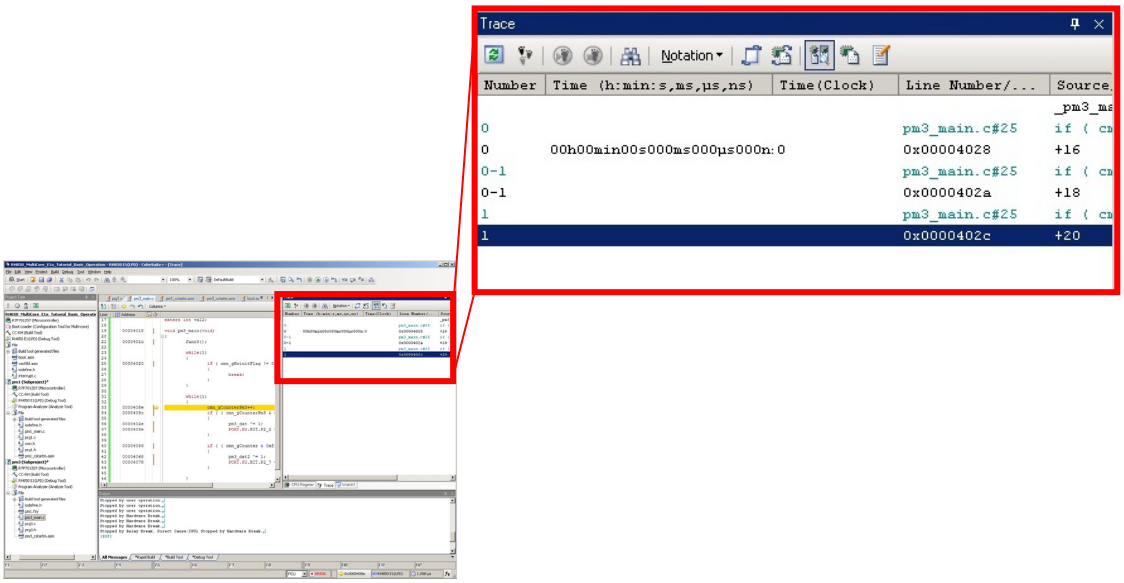
Setting trace operation

When the trace function starts recording, the execution process of the program currently running is recorded in trace memory (when program execution stops, the trace function also stops automatically).
Settings related to tracing need to be made in advance to use the trace function.

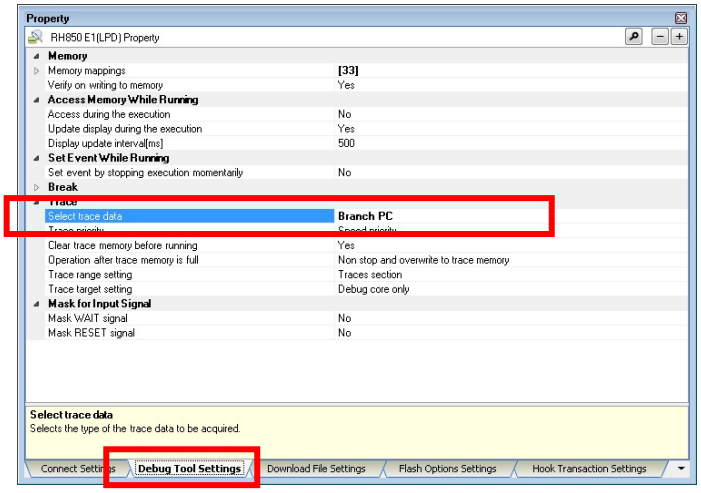
Select [Trace] from the [View] menu.



This opens the Trace panel.

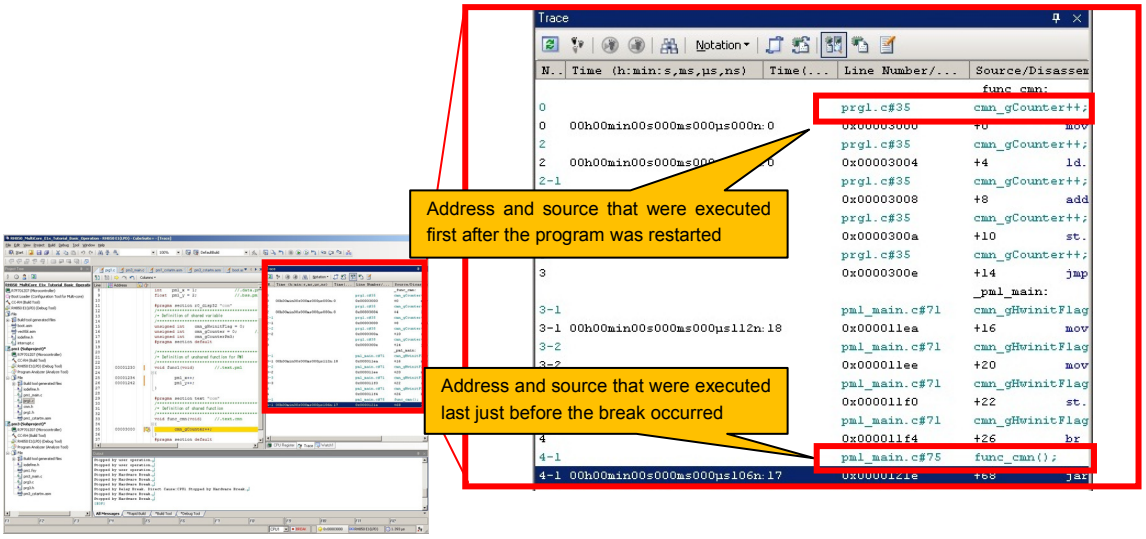


Settings for tracing can be made in the [Trace] category on the [Debug Tool Settings] tab of the Property panel.
Select the [Debug Tool Settings] tab and make the settings as shown below.



Click the [Go] button in the menu.

A break occurred and the execution history is displayed in the Trace window.

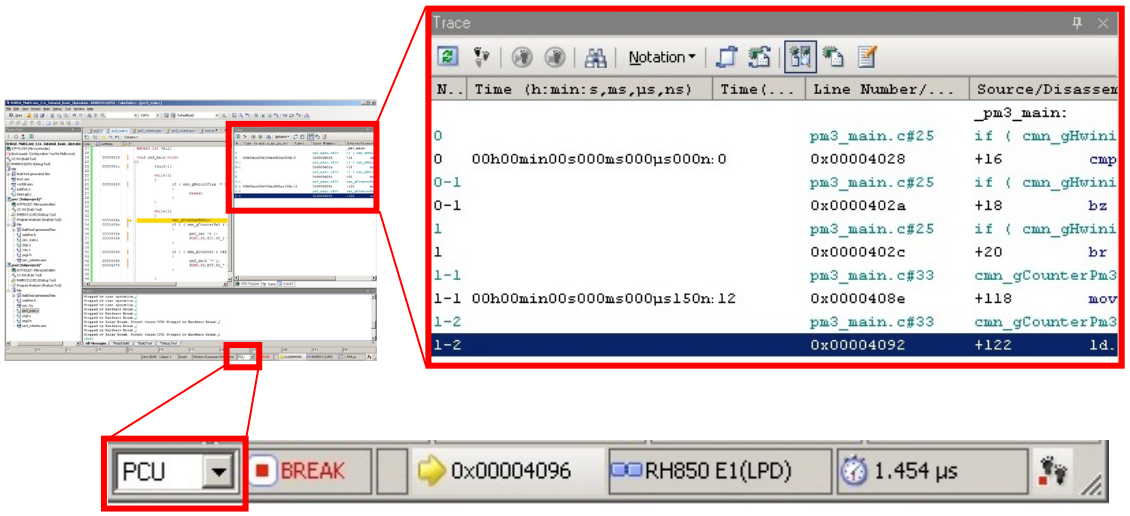


Tip

Tracing in a multi-core device

When trace data is acquired with the core to be debugged set to CPU1, only information for the CPU1 side can be observed. In order to acquire trace data for the PCU side, execute the program again after switching the core to be debugged to PCU.

After trace data has been acquired with the core to be debugged set to CPU1, even though the core to be debugged is switched to PCU, trace data for the PCU side cannot be observed.



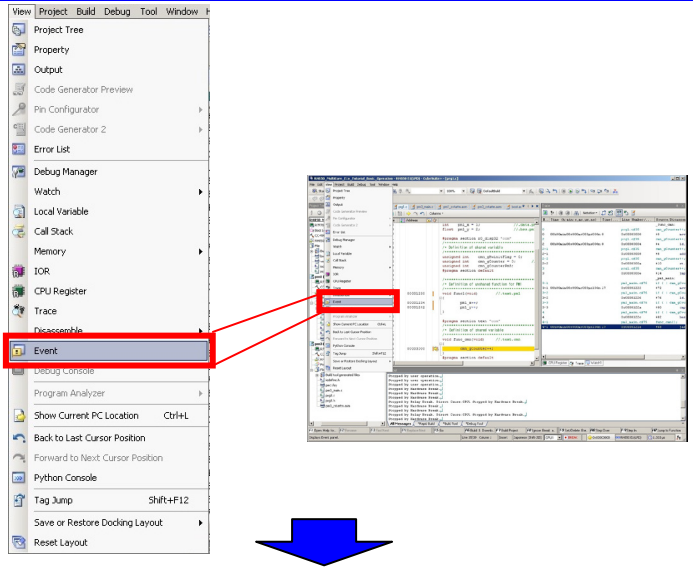
Canceling a Break

Canceling a Break

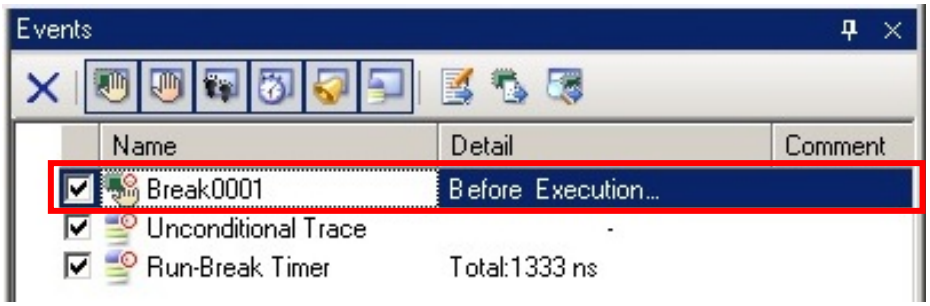
In this section, you cancel the break which was set in the previous section. The previously set break is set as a hardware break. A hardware break is registered as an event. Deleting the event will cancel the hardware break.

An event is an operation of the microcontroller such as fetch, read, and write. The event can be used as an action trigger to enable debugging functions such as setting a breakpoint or tracing. The hardware break that was set in the previous section was an event causing the program to break when a particular address is fetched (before it is executed).

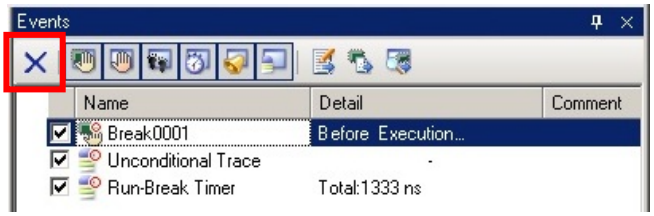
Select [Event] from the [View] menu.



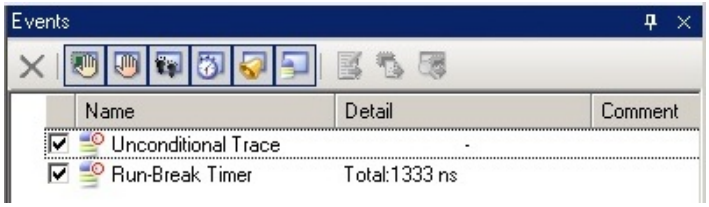
This opens the Event panel. Select [Break0001].



Left-click the Delete button. This cancels the access break.



Confirm that [Break0001] has been deleted from the Event panel.

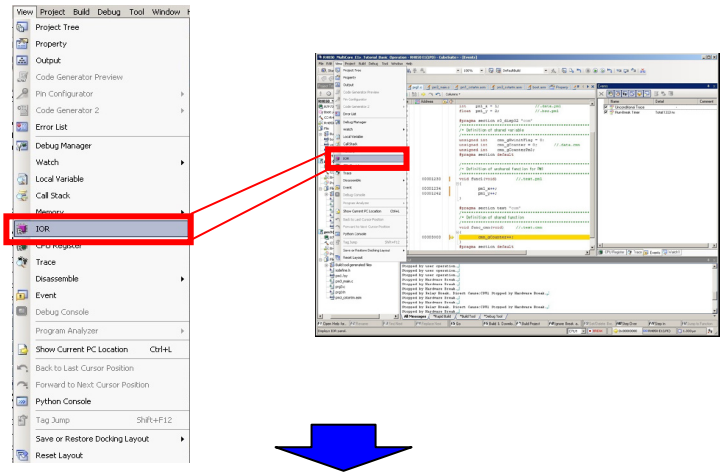


Displaying Special Function Registers (IORs)

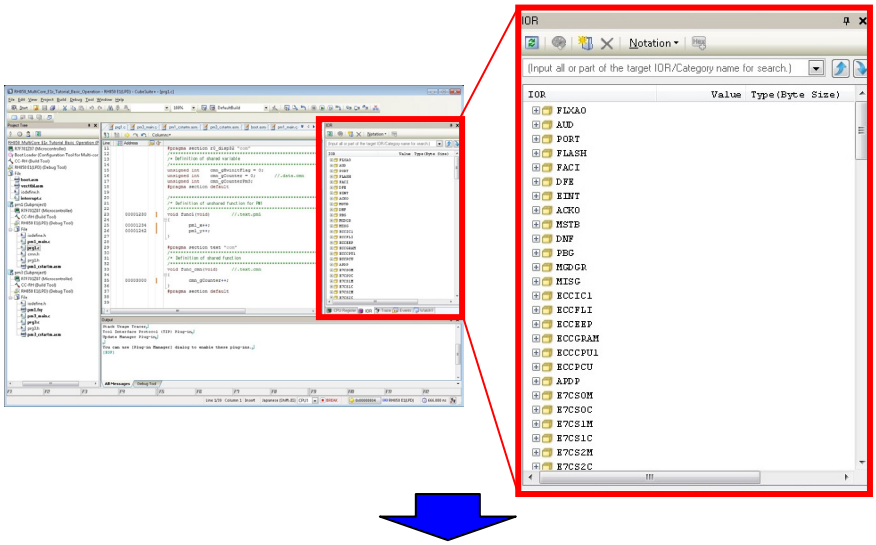
Displaying IORs

In this section, you observe the values of registers that implement peripheral functions built in the microcontroller. First, let's make the panel float so that it is easier to view.

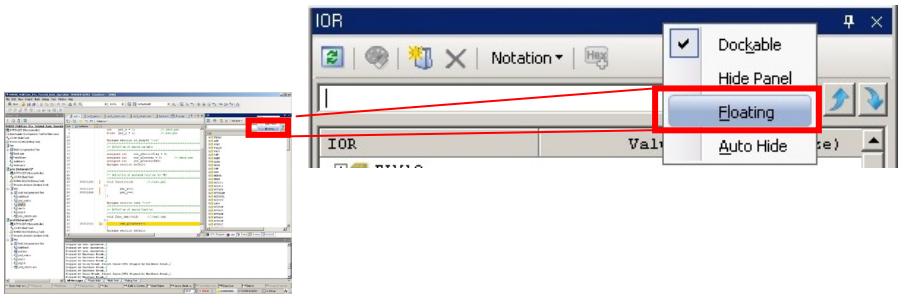
Select [IOR] from the [View] menu.



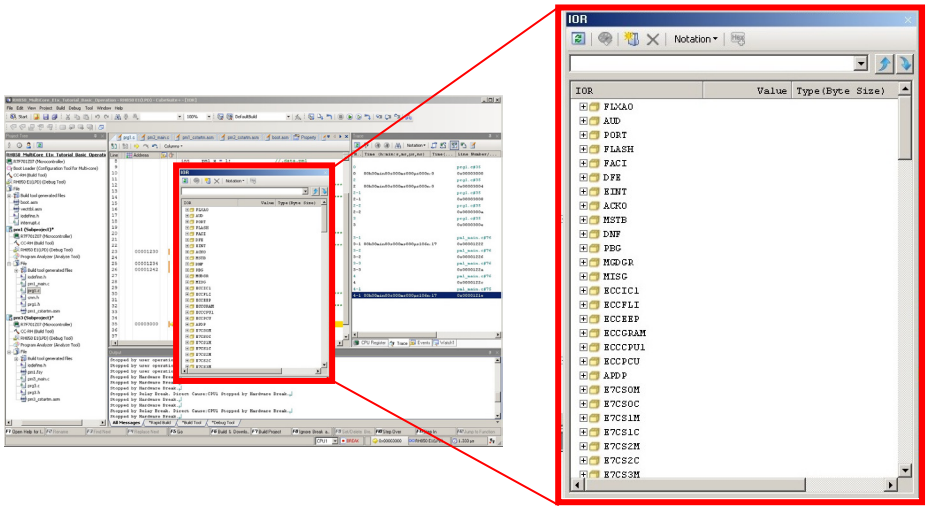
This opens the IOR panel.



Right-click on the title bar of the IOR panel and check [Floating].



The IOR panel enters the floating state and it becomes easier to view.



Tip

About display of IOR Bits

The IOR panel does not support displaying of IOR bits. Therefore, in order to check the bits of an IOR, that IOR has to be registered in a Watch panel so it can be referenced.

Select [Add New Watch] from the context menu in the desired Watch panel and input a watch-expression. To specify register bits, enter as shown below.

AAA0.BBB.CCC

<Module name>.<Register name>.<Bit name>

[Example]

Watch-expression for registering the P2_1 bit in the P2 register of a (general I/O) port in a Watch panel:

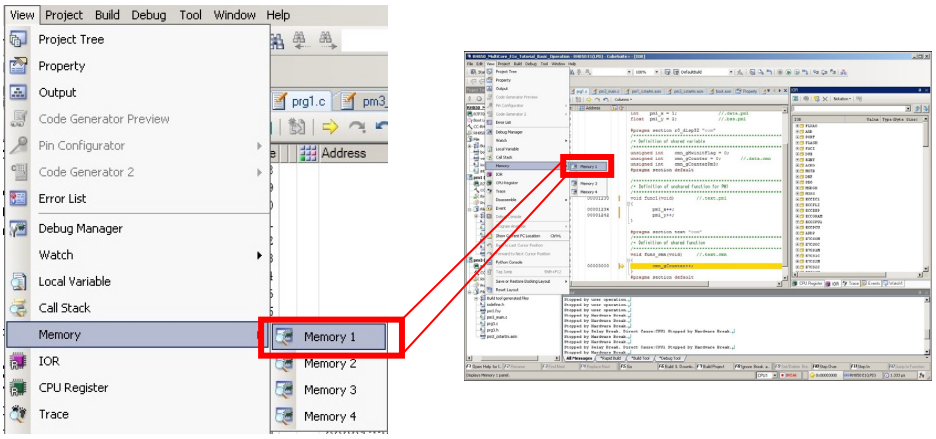
PORT.P2.P2_1

Displaying Memory

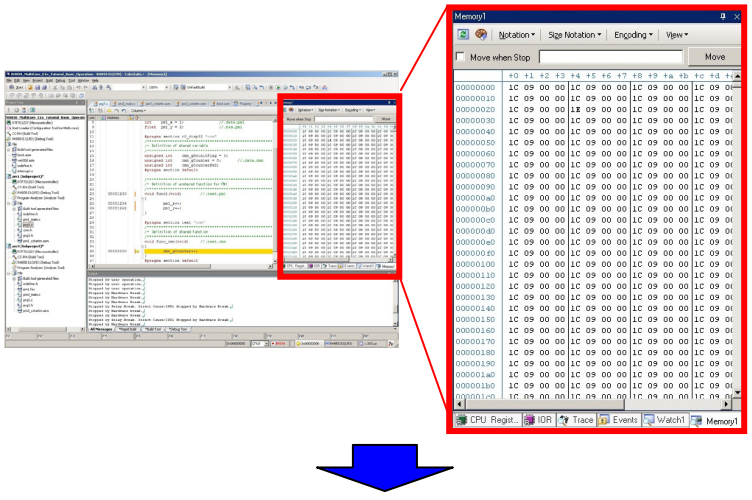
Displaying the Memory panel

The Memory panel displays the state of memory. In this example, two of the four Memory panels are displayed. If [Memory 1] and [Memory 2] are displayed at the same time, they are tabbed by default, making it impossible to view both panels at same time. Let's dock these two panels so that they can be viewed together.

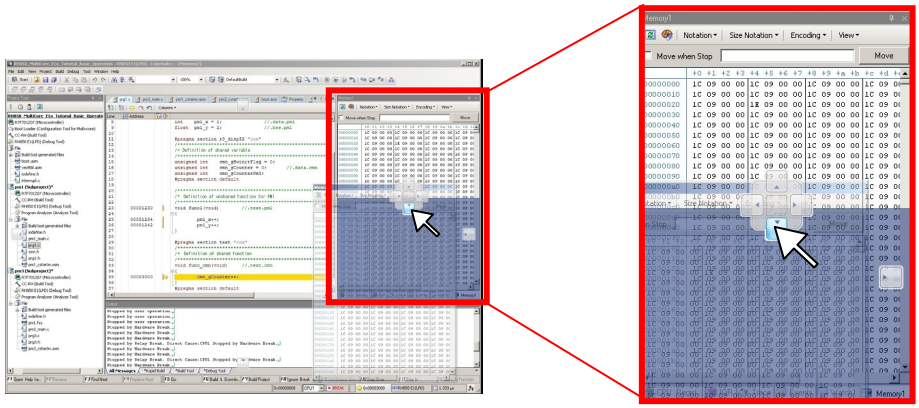
Select [Memory 1] from the [View] menu.



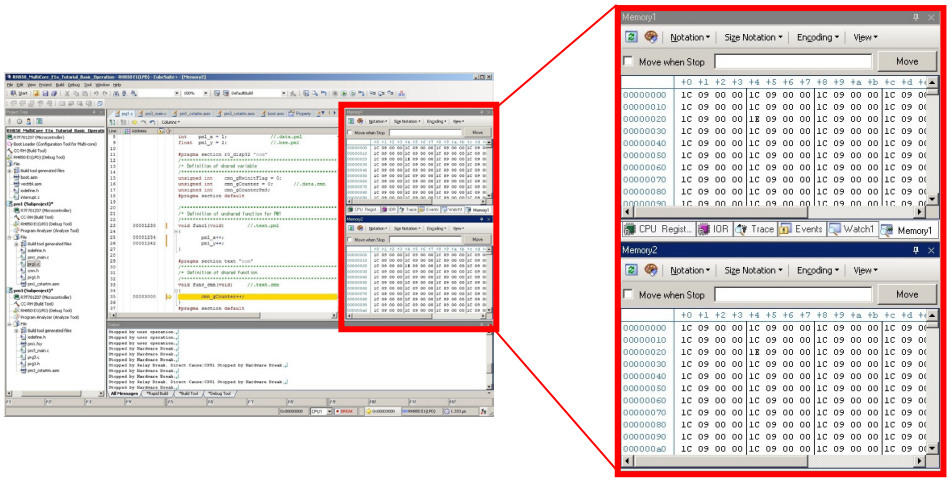
The Memory 1 panel is displayed.
(Similarly, display the Memory 2 panel.)



Make the Memory 2 panel enter the floating state and dock it to the Output panel.



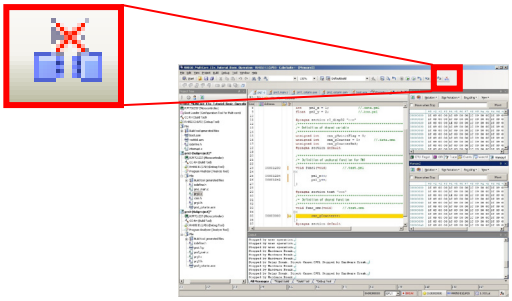
It becomes possible to view the Memory 1 and Memory 2 panels at the same time.



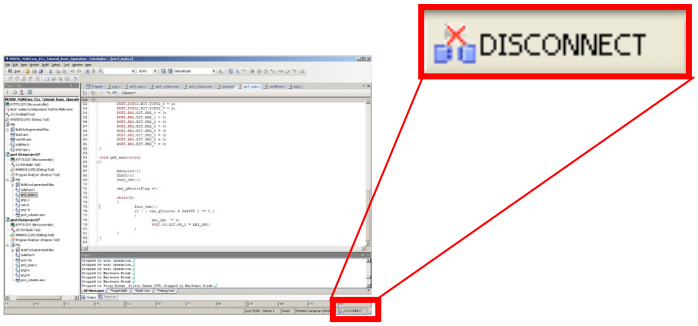
Disconnecting a Debug Tool

When finishing debugging, disconnect the debug tool.

Click the "Disconnect from Debug Tool" button.



The status bar in the main window shows [DISCONNECT] as shown below and debugging finishes.



Tip

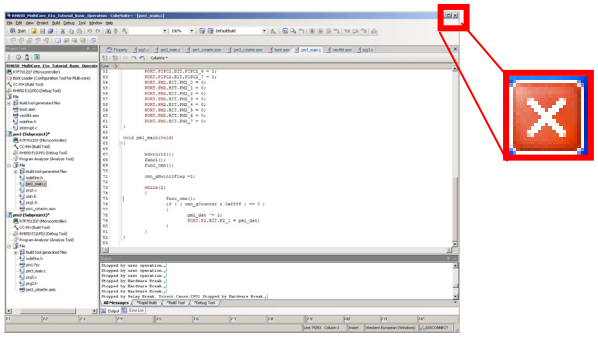
Downloading a program

If a program is changed after being downloaded to the target, you have to perform the build process again and download the program. If a program is changed after downloading, the right side of the line number turns yellow (or green) and breakpoints cannot be set.

Termination Procedure

This section describes the termination procedure.

Click the Close button in the main window.

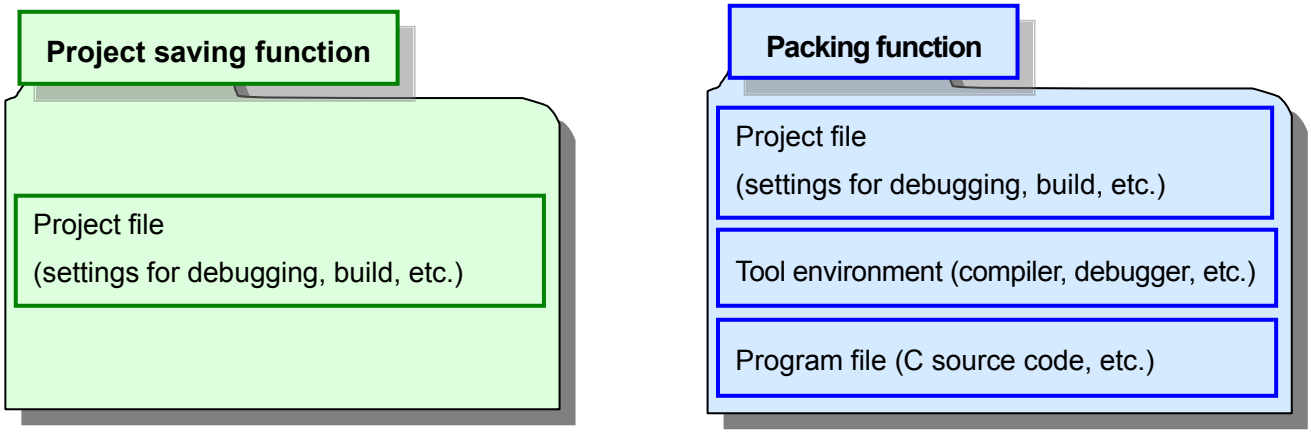


This closes the main window and exits CubeSuite+. If the environment is not saved, a window that prompts you to select whether or not to save it will appear, so follow the instructions.

Tip

Saving the development environment (project saving function and packing function)

CubeSuite+ provides two functions (project saving function and packing function) for saving the development environment. Each of these functions is used to save the contents shown in the figure below. Either saving function can be chosen depending on the development phase.



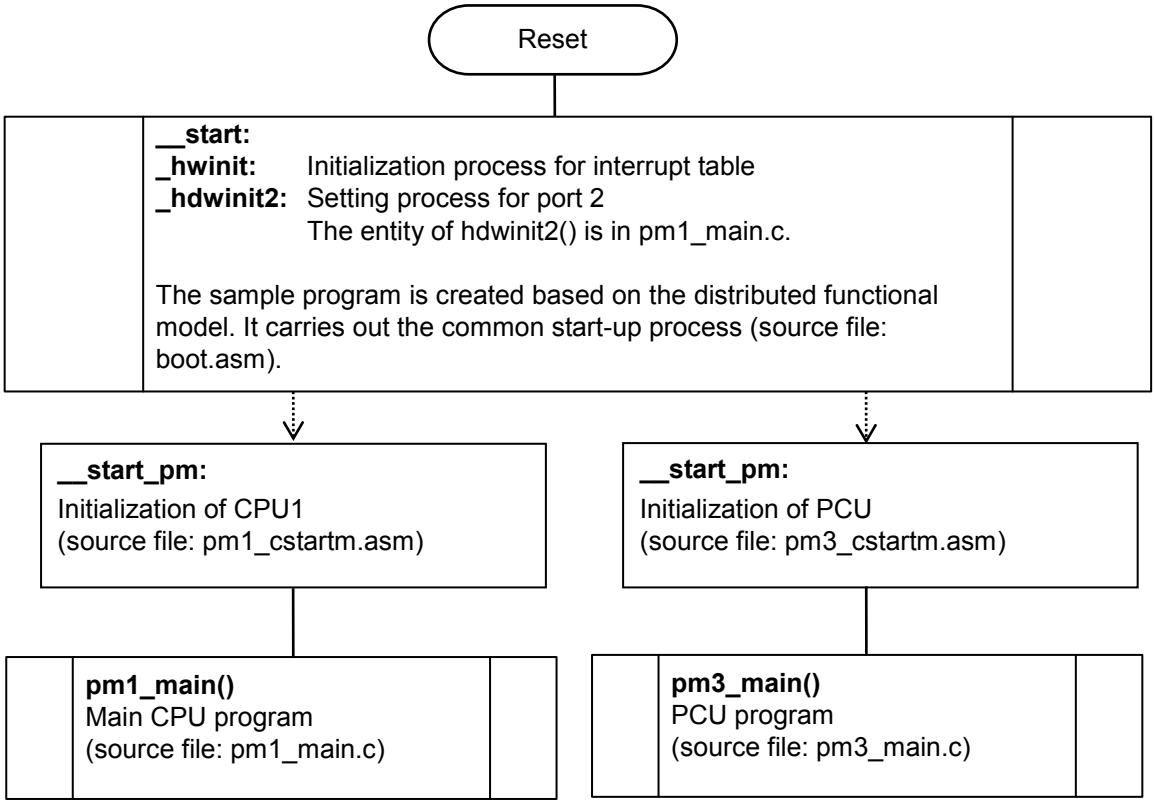
About Flash Programming

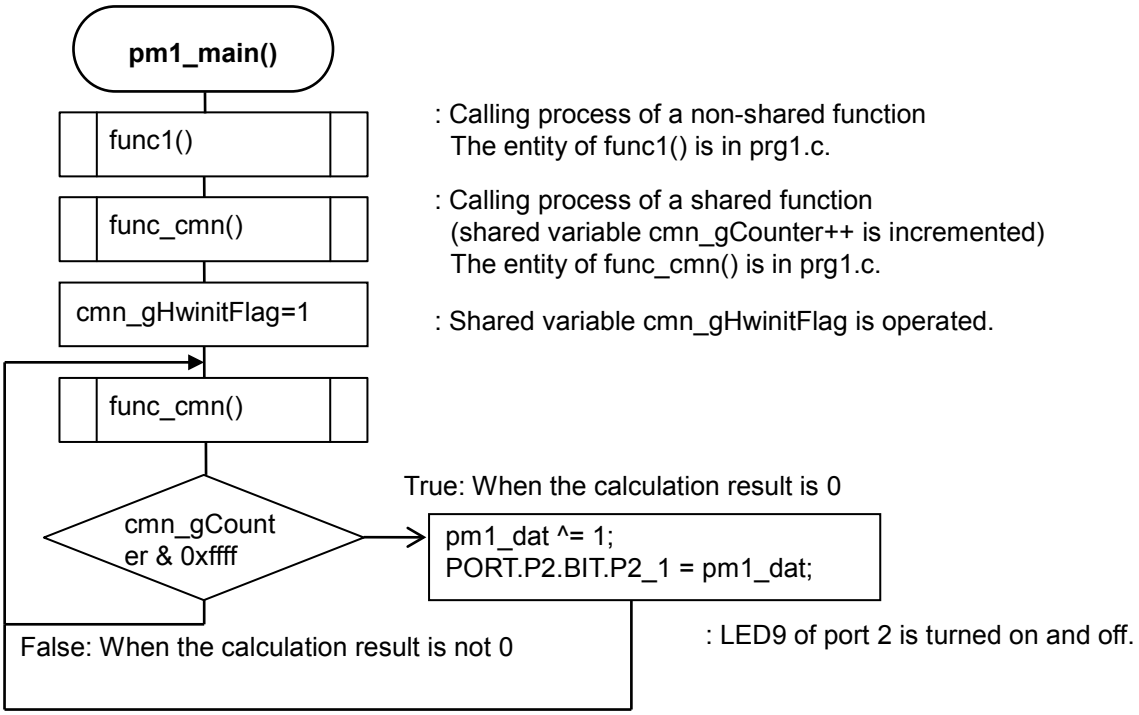
When using the E1 emulator to write a .hex file to the microcontroller, use Renesas Flash Programmer (RFP).

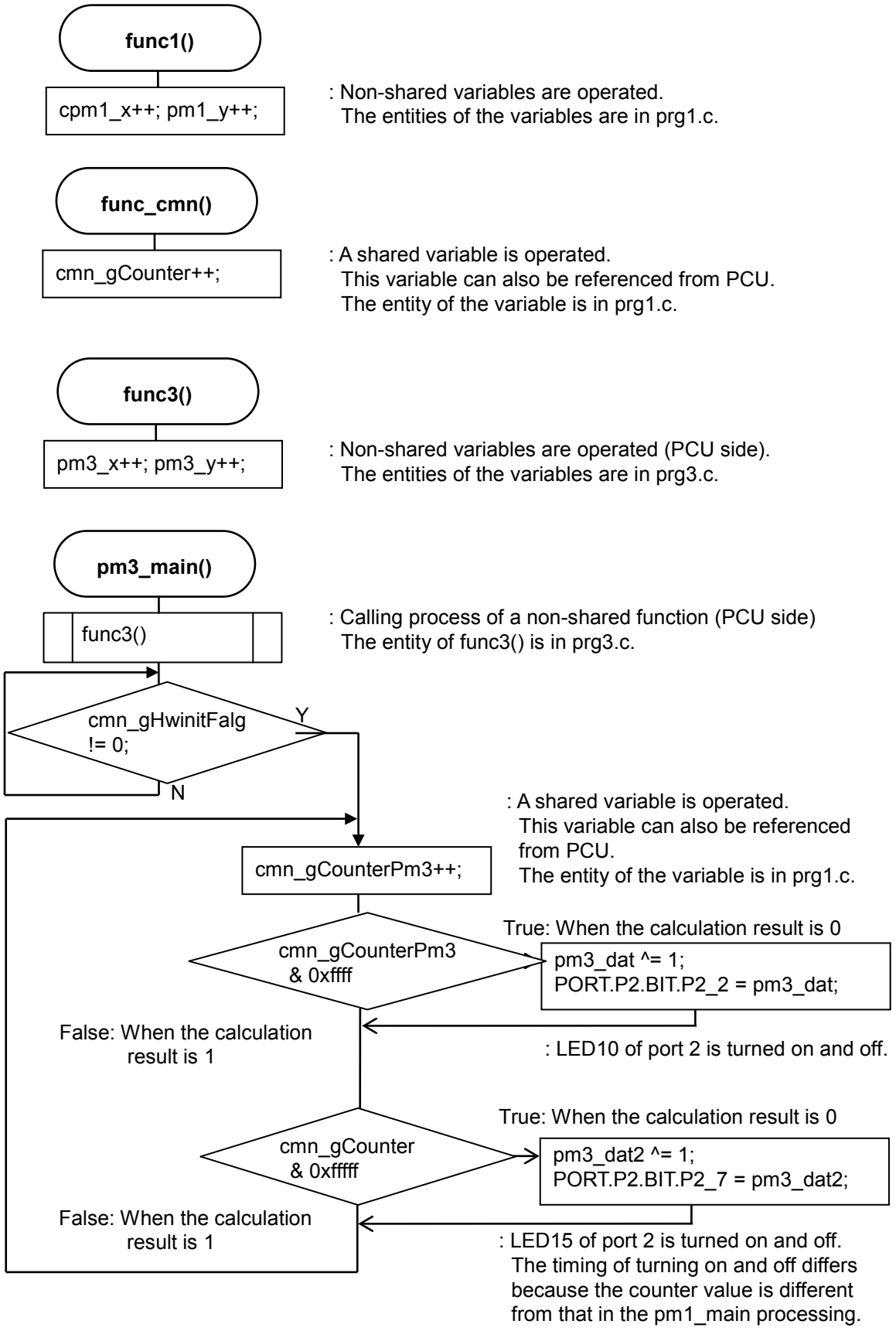
- Start Renesas Flash Programmer (RFP) by selecting [Start] -> [All Programs] -> [Renesas Electronics Utilities] -> [Programming tools].
- Refer to the user's manual for the usage method.

Description of Sample Programs

The flow diagrams of sample programs are shown below.






func1()
cpm1_x++; pm1_y++;
: Non-shared variables are operated.
The entities of the variables are in prg1.c.
func_cmn()
cmn_gCounter++;
: A shared variable is operated.
This variable can also be referenced from PCU.
The entity of the variable is in prg1.c.
func3()
pm3_x++; pm3_y++;
: Non-shared variables are operated (PCU side).
The entities of the variables are in prg3.c.
pm3_main()
func3()
: Calling process of a non-shared function (PCU side)
The entity of func3() is in prg3.c.
cmn_gHwinitFalg
!= 0;
Y
N
: A shared variable is operated.
This variable can also be referenced
from PCU.
The entity of the variable is in prg1.c.
cmn_gCounterPm3++;
True: When the calculation result is 0
cmn_gCounterPm3
& 0xffff
pm3_dat ^= 1;
PORT.P2.BIT.P2_2 = pm3_dat;
False: When the calculation
result is 1
: LED10 of port 2 is turned on and off.
True: When the calculation result is 0
cmn_gCounter
& 0xfffff
pm3_dat2 ^= 1;
PORT.P2.BIT.P2_7 = pm3_dat2;
False: When the calculation
result is 1
: LED15 of port 2 is turned on and off.
The timing of turning on and off differs
because the counter value is different
from that in the pm1_main processing.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
3. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; and safety equipment etc.
   Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (nuclear reactor control systems, military equipment etc.). You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application for which it is not intended. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.
6. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You should not use Renesas Electronics products or technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. When exporting the Renesas Electronics products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document, Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.
11. This document may not be reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

RENESAS

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
2801 Scott Boulevard Santa Clara, CA 95050-2549, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
1101 Nicholson Road, Newmarket, Ontario L3Y 9C3, Canada
Tel: +1-905-898-5441, Fax: +1-905-898-3220

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
Room 1709, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100191, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, P. R. China 200333
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

Renesas Electronics Hong Kong Limited
Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022/9044

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre, Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

Renesas Electronics Malaysia Sdn.Bhd.
Unit 906, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics Korea Co., Ltd.
12F., 234 Teheran-ro, Gangnam-Ku, Seoul, 135-920, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 4.0