

# E1/E20 Emulator

Additional Document for User's Manual:

High-performance Embedded Workshop RX Debug

Target Devices  
RX Family

All information contained in these materials, including products and product specifications, represents information on the product at the time of publication and is subject to change by Renesas Electronics Corporation without notice. Please review the latest information published by Renesas Electronics Corporation through various means, including the Renesas Electronics Corporation website (<http://www.renesas.com>).

## 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.

## Contents

Section 1 Configuration of E1/E20 Emulator Manuals .....	7
Section 2 Preparations for Debugging .....	8
2.1 Operating Environment.....	8
2.2 Activating High-performance Embedded Workshop.....	9
2.3 Creating a New Workspace (with a Toolchain not in Use).....	10
2.4 Creating a New Workspace (with a Toolchain in Use).....	13
2.5 Selecting an Existing Workspace.....	16
2.6 Connecting the Emulator .....	17
2.6.1 Connecting the Emulator .....	17
2.6.2 Reconnecting the Emulator.....	17
2.7 Disconnecting the Emulator.....	17
2.7.1 Disconnecting the Emulator.....	17
2.8 Quitting the High-performance Embedded Workshop .....	17
2.9 Making Debugging-Related Settings .....	18
2.9.1 Specifying a Module for Downloading.....	18
2.9.2 Setting Up Automatic Execution of Command Line Batch Files .....	19
2.10 Procedure for Launching the E1/E20 Emulator Debugger .....	20
Section 3 Software Specifications.....	28
3.1 Specifications of the E1 and E20 Emulators.....	28
3.2 Differences between the MCU and the Emulator .....	33
3.3 Setting up the Debugger.....	35
3.3.1 [Initial Settings] Dialog Box.....	35
3.3.2 [Configuration Properties] Dialog Box.....	40
Section 4 Emulator Functions .....	48
4.1 Overview.....	48
4.2 Downloading a Program .....	49
4.3 Opening a Source File.....	50
4.3.1 Viewing the Source Code .....	50
4.3.2 Viewing the Assembly-Language Code.....	52
4.3.3 Modifying the Assembly-Language Code .....	53
4.4 Memory Access Functions.....	53
4.4.1 Memory Read/Write Function .....	53
4.5 Break Functions .....	55
4.5.1 Forced Break.....	55
4.5.2 S/W (Software) Break.....	55
4.5.3 On-Chip Break.....	55
4.6 Event Functions .....	56
4.6.1 Using Events.....	56
4.6.2 Adding Events.....	57
4.6.3 Removing Events.....	62
4.6.4 Registering Events .....	64
4.6.5 Removing a Registered Event.....	68
4.6.6 Creating Events for Each Instance of Usage or Reusing Events.....	69
4.6.7 Activating Events.....	70
4.7 On-Chip Breakpoints .....	71
4.7.1 Setting On-Chip Breakpoints.....	71

4.7.2	Saving/Loading On-Chip Break Settings.....	76
4.8	Trace Functions.....	77
4.8.1	Viewing Trace Information.....	78
4.8.2	Acquiring Trace Information.....	78
4.8.3	Results of Tracing.....	81
4.8.4	Popup Menu Options.....	85
4.8.5	Setting Trace Conditions.....	87
4.8.6	Saving Trace Information in Files.....	91
4.8.7	Loading Trace Information from Files.....	91
4.8.8	Temporarily Stopping Trace Acquisition.....	91
4.8.9	Restarting Trace Acquisition.....	91
4.9	Measuring Performance.....	92
4.9.1	Measuring Performance.....	92
4.9.2	Viewing the Results of Performance Measurement.....	92
4.9.3	Setting Performance Measurement Conditions.....	95
4.9.4	Starting Performance Measurement.....	100
4.9.5	Clearing Performance-Measurement Conditions.....	100
4.9.6	Clearing Results of Performance Measurement.....	101
4.9.7	Maximum Number of Rounds of Performance Measurement.....	101
4.9.8	Saving/Loading Performance-Measurement Settings.....	101
4.10	Viewing Memory Data in Real Time.....	102
4.10.1	RAM Monitoring.....	102
4.10.2	Allocating Blocks for RAM Monitoring.....	103
4.10.3	Viewing RAM Monitoring Information.....	105
4.10.4	When Some Trace Data are Lost.....	108
4.10.5	Preventing Loss of Data.....	109
4.11	Using the Start/Stop Function.....	111
4.11.1	Opening the [Start/Stop function setting] Dialog Box.....	111
4.11.2	Specifying the Routine to be Executed.....	111
4.11.3	Restrictions on the Start/Stop Function.....	111
4.11.4	Limitations on Statements within Specified Routines.....	112
4.12	Using the Debug Console.....	113
4.12.1	Opening the [DebugConsole] Window.....	113
4.12.2	Low-Level Interface Routines.....	113
4.13	Hot Plug-in Function.....	116
4.13.1	Startup Procedure.....	116
4.13.2	Precautions to Take when Using Hot Plug-in.....	121
4.14	Graph Functions.....	122
4.14.1	Displaying the [Graph] window.....	122
4.15	Stack Trace Function.....	125
4.16	Online Help.....	125
	Section 5 Tutorial.....	126
5.1	Introduction.....	126
5.2	Starting the High-performance Embedded Workshop.....	127
5.3	Booting Up the Emulator.....	127
5.4	Downloading the Tutorial Program.....	128
5.4.1	Downloading the Tutorial Program.....	128
5.4.2	Displaying the Source Program.....	129
5.5	Setting S/W breakpoints.....	130
5.6	Executing the Program.....	131
5.6.1	Resetting the CPU.....	131
5.6.2	Executing the Program.....	131

5.7	Checking Breakpoints.....	133
5.7.1	Checking Breakpoints.....	133
5.8	Altering Register Contents.....	134
5.9	Referring to Symbols.....	135
5.10	Checking Memory Contents.....	136
5.11	Referring to Variables.....	137
5.12	Showing Local Variables.....	139
5.13	Single-Stepping through a Program.....	140
5.13.1	Executing [Step In].....	140
5.13.2	Executing [Step Out].....	141
5.13.3	Executing [Step Over].....	142
5.14	Forcibly Breaking Program Execution.....	143
5.15	On-Chip Break Facility.....	144
5.15.1	Stopping a Program when It Executes the Instruction at a Specified Address.....	144
5.16	Stopping a Program when It Accesses Memory.....	145
5.17	Tracing Facility.....	146
5.17.1	Showing the Information Acquired in “Fill until Stop” Tracing.....	147
5.18	Stack Trace Facility.....	149
5.19	What Next?.....	150
Section 6 Notes on Usage.....		151
6.1	Memory.....	151
6.1.1	I/O Register Area.....	151
6.1.2	Internal Flash ROM Area.....	151
6.1.3	Downloading to the Internal Flash ROM.....	151
6.1.4	Re-Writing the Internal Flash ROM.....	151
6.1.5	FCU-RAM and FCU Firmware Areas.....	152
6.1.6	Working RAM Area for Use by the Debugger.....	152
6.2	[Memory] Window.....	152
6.2.1	Copy, Comparison, Find.....	152
6.2.2	Menu Items.....	152
6.3	Running Programs.....	152
6.3.1	[Go To Cursor].....	152
6.3.2	[Run Program].....	152
6.3.3	[Step Out].....	152
6.4	Reset.....	153
6.4.1	Contention between the Resets and the Operations by the Emulator System.....	153
6.4.2	MCU Reset When Using the Trace Function.....	153
6.4.3	MCU Reset When Using the Realtime RAM Monitor.....	153
6.4.4	Reset during the User Program Execution.....	153
6.5	[IO] Window.....	154
6.5.1	Customization of the I/O-Register Definition File.....	154
6.5.2	Verify.....	154
6.6	Tracing.....	155
6.6.1	Traceable Accesses.....	155
6.6.2	Trace Information.....	155
6.7	Events.....	156
6.7.1	Detectable Accesses.....	156
6.7.2	Event Combinations.....	156
6.7.3	Number of Passes.....	156
6.7.4	Data-Access Event with an Address Range Defined.....	156
6.7.5	Registering Events.....	156
6.7.6	Events for the WAIT Instruction.....	157
6.7.7	Event Resources.....	157

6.8	Break Functions .....	158
6.8.1	Notes on Setting Break Points .....	158
6.9	Realtime RAM Monitoring.....	159
6.9.1	Displaying the Block Boundary Memory .....	159
6.9.2	Uninitialized Area Detection .....	159
6.10	Performance Measurement .....	160
6.10.1	Reset During Performance Measurement .....	160
6.10.2	Limitation on Nesting for the Performance Measurement.....	160
6.10.3	Notes on Checking the [Measure the performance only once.] Checkbox .....	160
6.11	Downloading.....	161
6.11.1	About the Access Size .....	161
6.12	Downloading to the External Flash Memory .....	163
6.13	Execution Time.....	164
6.14	Start/Stop Function .....	164
6.15	Watch Function.....	164
6.16	Debug Console Function.....	164
6.17	FINE Interface .....	164
6.18	Option Settings Related Registers.....	165
6.18.1	About the Endian Select Registers (MDEB, MDES).....	165
6.18.2	About the Setting of Option Function Select Register 1 (OFS1) .....	165
6.19	Other .....	165
6.19.1	Register Values after the On-Chip Flash Memory Has been Programmed.....	165
6.19.2	DMAC and DTC.....	165
6.19.3	About the High-Speed Clock Oscillator (HOCO).....	166
6.19.4	About the Lock-Bits.....	166
6.19.5	About Disconnection of the Emulator .....	166
6.19.6	About the Operating Frequency .....	166
6.19.7	About the MCU Containing the USB Boot Program.....	167
6.19.8	About the Setting that Enables Clock Manipulation.....	167
6.19.9	About Access to the MPU Area.....	167
Appendix A	Menus .....	168
Appendix B	Notes on High-performance Embedded Workshop.....	171

## Section 1 Configuration of E1/E20 Emulator Manuals

The E1/E20 manual consists of multiple parts: the E1/E20 Emulator User's Manual and the additional documents for the user's manual for each MCU.

Be sure to read each part before using the E1/E20 emulator.

(1) E1/E20 emulator user's manual

The E1/E20 emulator user's manual has the following contents:

- Components of the emulators
- Emulator hardware specification
- Connection to the emulator and the host machine and user system

(2) E1/E20 Additional Documents for User's Manual (RX User System Design)

The E1/E20 Additional Documents for User's Manual (RX User System Design) describes information necessary for hardware design such as connection examples and interface circuits.

(3) E1/E20 Additional Document for User's Manual (High-performance Embedded Workshop RX Debug)

The E1/E20 Additional Document for User's Manual (High-performance Embedded Workshop RX Debug) describes the functions of the E1/E20 Emulator Debugger and the operating instructions.

- Software Specifications of the RX E1/E20 emulator debugger
- Instructions on how to use the RX E1/E20 emulator debugger
- A tutorial on procedures from starting up the RX E1/E20 emulator debugger to debugging operations
- Notes on using the RX E1/E20 emulator debugger

## Section 2 Preparations for Debugging

### 2.1 Operating Environment

Make sure to use this emulator in accordance with the operating environments of the host machine shown in Table 2.1 and Table 2.2.

**Table 2.1 Operating Environment (Windows® XP)**

PC Environment	
PC	IBM PC/AT compatible.
OS	Windows® XP 32-bit editions *1*3
CPU	Pentium 4 running at 1.6 GHz or more recommended.
Interface	USB2.0 *2
Memory	1 Gbyte plus 10 times the file size of the load module or larger recommended.
Pointing device such as mouse	Mouse or any other pointing device usable with the above OS that can be connected to the host machine.
CD drive	Needed to install the emulator debugger or refer to the user's manual.
Hard disk	Emulator debugger installation needs 600 MB or more free space. (In view of swap area, keep another free space which is more than twice the memory capacity. (More than four times the memory capacity recommended.))
Display resolution	1024 × 768 or greater recommended.

**Table 2.2 Operating Environment (Windows Vista® or Windows® 7)**

PC Environment	
PC	IBM PC/AT compatible.
OS	Windows Vista® 32-bit editions*1 *4 Windows® 7 32/64-bit editions*1
CPU	Pentium 4 running at 3 GHz or Core 2 Duo running at 1 GHz or more recommended.
Interface	USB2.0*2
Memory	2 Gbytes plus 10 times the file size of the load module or larger recommended (32-bit editions). 3 Gbytes plus 10 times the file size of the load module or larger recommended (64-bit editions).
Pointing device such as mouse	Mouse or any other pointing device usable with the above OS that can be connected to the host machine.
CD drive	Needed to install the emulator debugger or refer to the user's manual.
Hard disk	Emulator debugger installation needs 600 MB or more free space. (In view of swap area, keep another free space which is more than twice the memory capacity. (More than four times the memory capacity recommended.))
Display resolution	1024 × 768 or higher recommended.

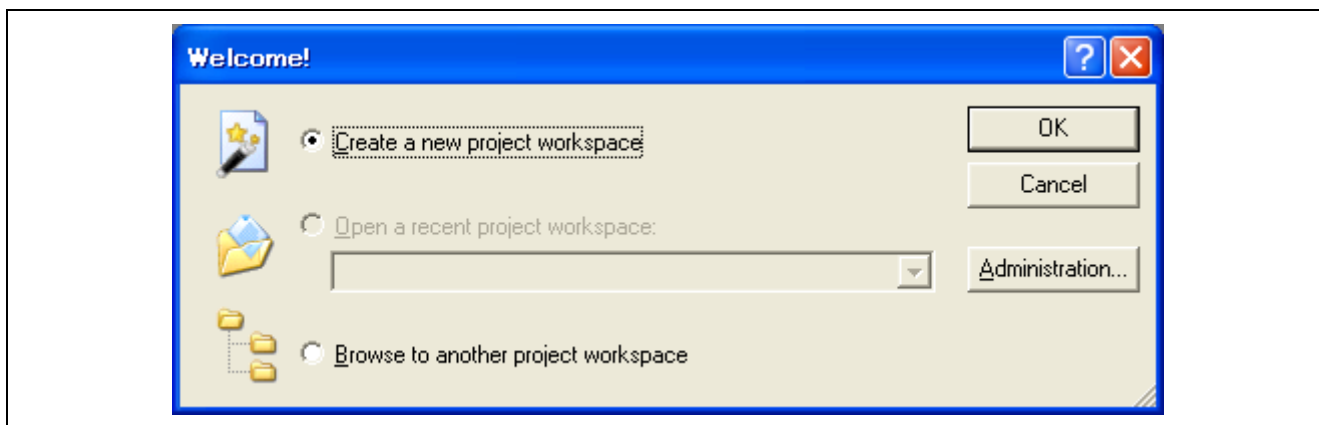
- Notes: 1. Windows and Windows Vista are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. All other company or product names are the property of their respective owners.
2. Operation with all combinations of host machine, USB device and USB hub is not guaranteed for the USB interface.
3. The 64-bit editions of Windows® XP are not supported.
4. The 64-bit editions of Windows Vista® are not supported.



## 2.2 Activating High-performance Embedded Workshop

To activate the High-performance Embedded Workshop, follow the procedure listed below.

- (1) Connect the emulator to the host machine and the user system, then turn on the power for the emulator and the user system.
- (2) Select [Renesas -> High-performance Embedded Workshop -> High-performance Embedded Workshop] from [Programs] in the [Start] menu.  
The [Welcome!] dialog box is displayed.



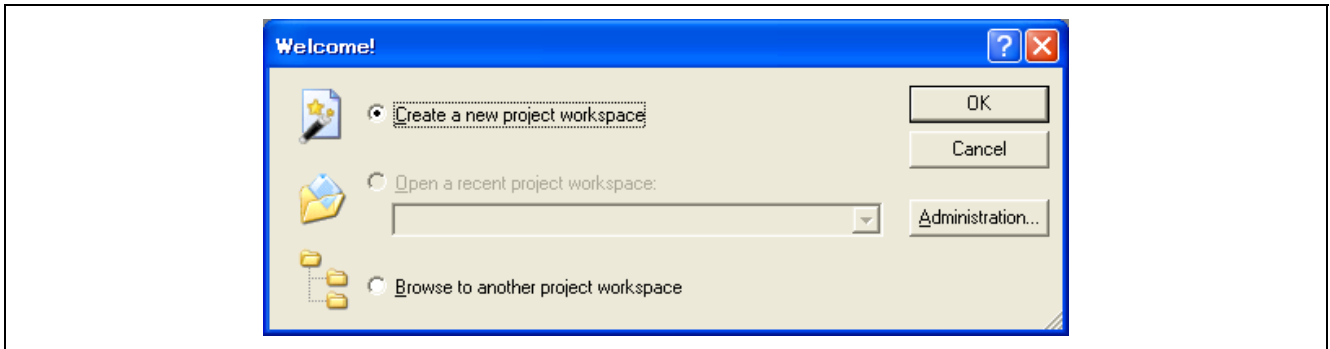
**Figure 2.1 [Welcome!] Dialog Box**

- |   |  |
|---|--|
| [Create a new project workspace] radio button:      | Creates a new workspace.   |
| [Open a recent project workspace] radio button:     | Uses an existing workspace and displays the history of the opened workspace. |
| [Browse to another project workspace] radio button: | Uses an existing workspace.  |

### 2.3 Creating a New Workspace (with a Toolchain not in Use)

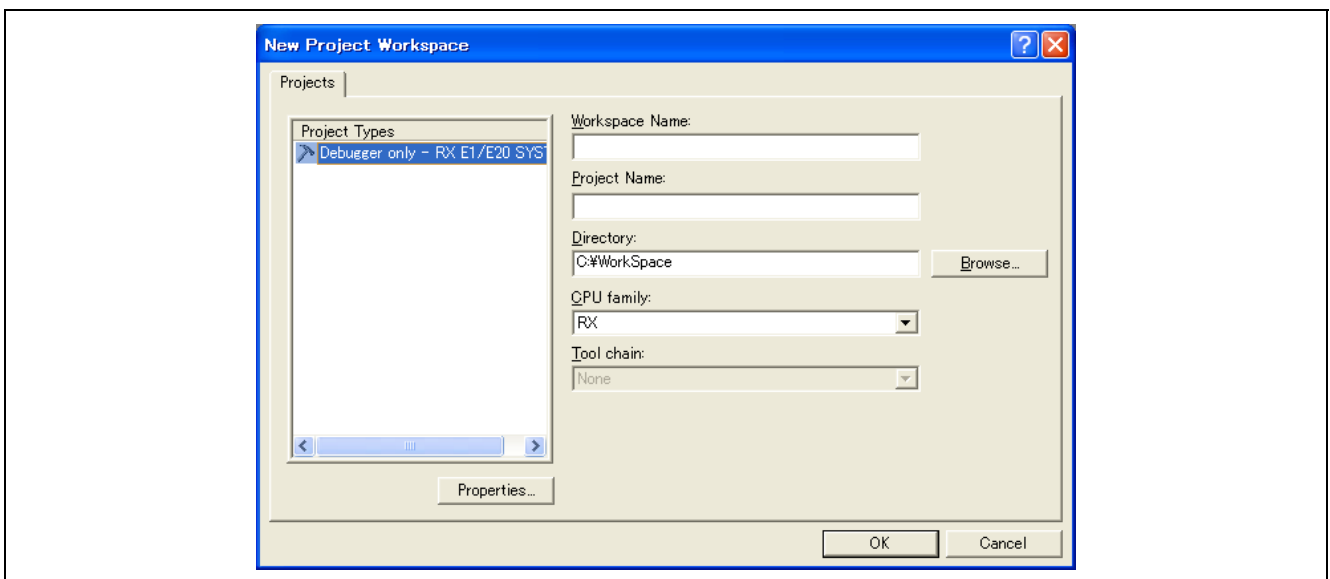
The procedures for creating a new project workspace differ according to whether or not a toolchain is in use. This product does not include a toolchain. You can only use a toolchain in an environment where a C/C++ compiler package has been installed. Follow the steps below to create a new workspace.

- (1) In the [Welcome!] dialog box that is displayed when the High-performance Embedded Workshop is activated, select [Create a new project workspace] radio button and click the [OK] button.



**Figure 2.2 [Welcome!] Dialog Box**

- (2) The Project Generator is started.

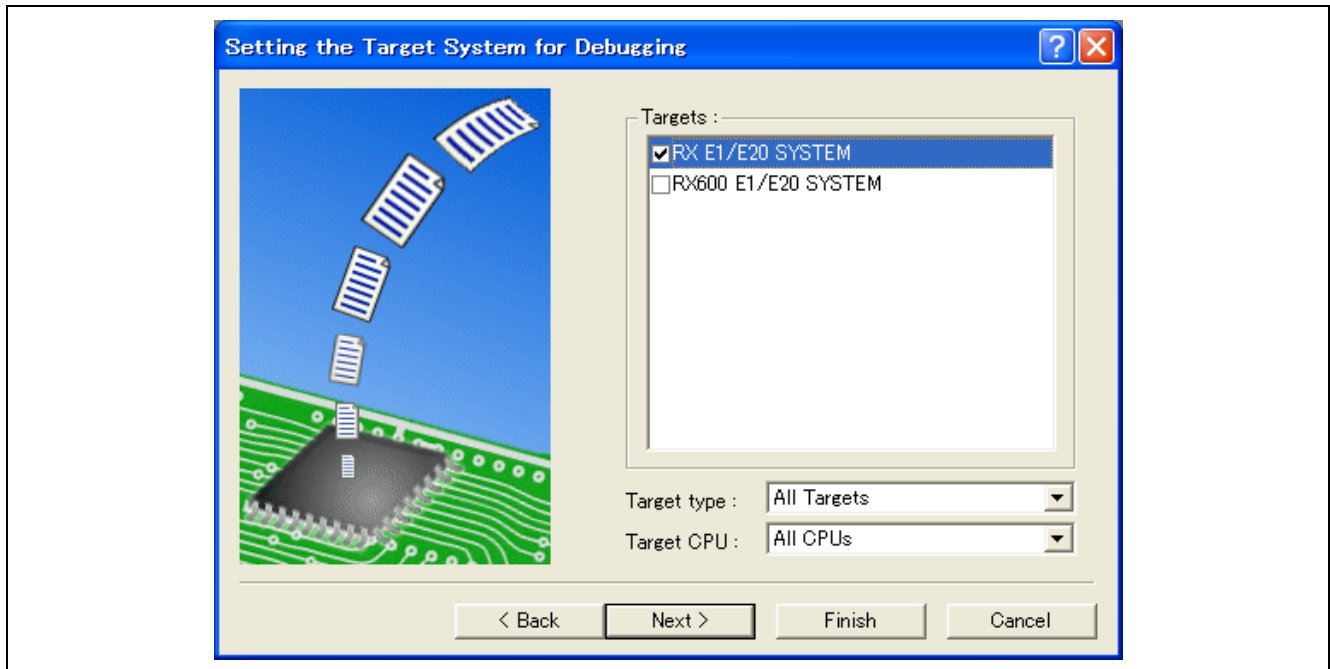


**Figure 2.3 [New Project Workspace] Dialog Box**

- [Workspace Name]: Enter a new workspace name.
- [Project Name]: Enter a project name. You do not need to enter any name if you wish this to be the same as the workspace name.
- [Directory]: Enter the directory in which you want the workspace to be created. Alternatively, click on the [Browse] button and select a workspace directory from the dialog box.
- [CPU family]: Select the CPU family of the MCU you are using.

Other list boxes are used for setting the toolchain; the fixed information is displayed when the toolchain has not been installed. Click on the [OK] button.

(3) Select the target for debugging.



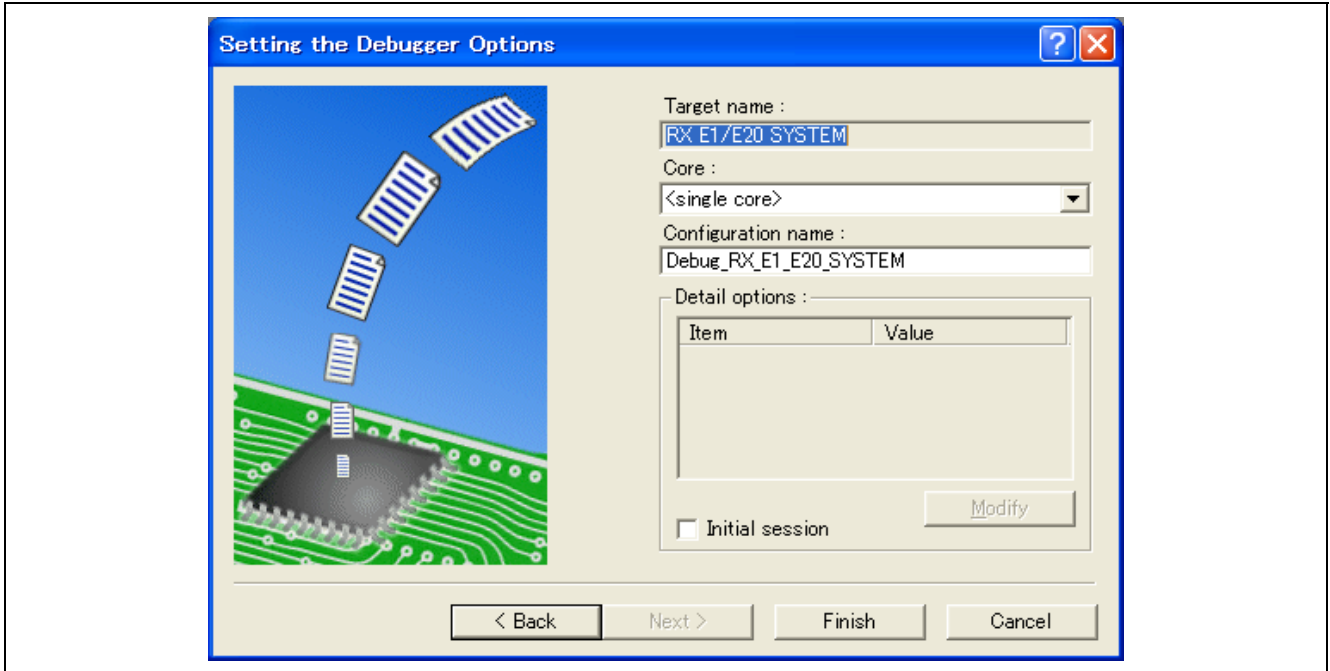
**Figure 2.4** [Setting the Target System for Debugging] Dialog Box

Select the checkbox “RX E1/E20 SYSTEM,” which is the target platform for the RX family, and click the [Next] button.

Note that the checkbox “RX600 E1/E20 SYSTEM” is provided for maintaining compatibility with the earlier versions. Do not select it.

(4) Set the configuration file name.

Configuration refers to a file in which information on the state of the High-performance Embedded Workshop for use with target software rather than emulators is saved.



**Figure 2.5 [Setting the Debugger Options] Dialog Box**

If you have selected two or more target platforms, click on the [Next] button and then set a configuration name for each of the selected target platforms.

When you have finished setting the configuration names, settings related to the emulator debugger have been completed.

Click on the [Finish] button, and the [Summary] dialog box will be displayed.

Clicking on the [OK] button in this dialog box starts the High-performance Embedded Workshop.

(5) After starting the High-performance Embedded Workshop, connect the emulator.

## 2.4 Creating a New Workspace (with a Toolchain in Use)

Follow the procedure below to create a new workspace.

- (1) In the [Welcome!] dialog box, select the [Create a new project workspace] radio button and click on the [OK] button.

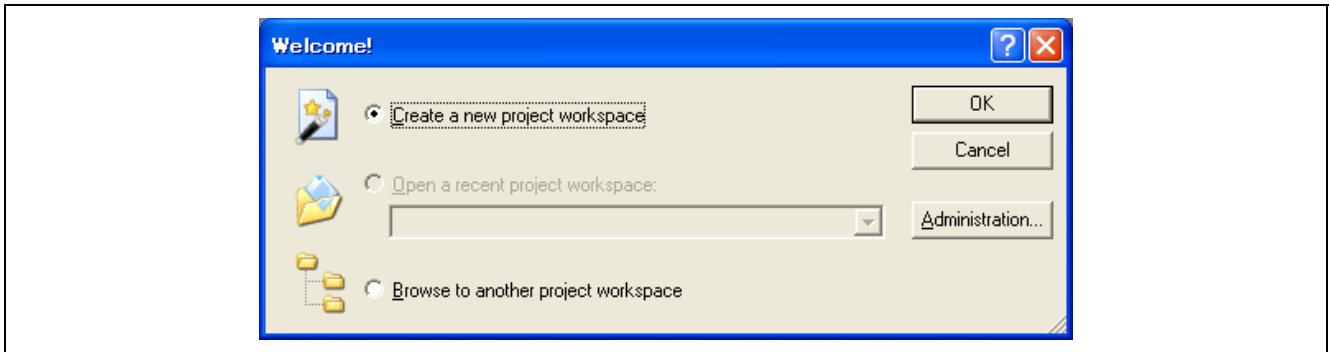


Figure 2.6 [Welcome!] Dialog Box

- (2) The Project Generator is started.

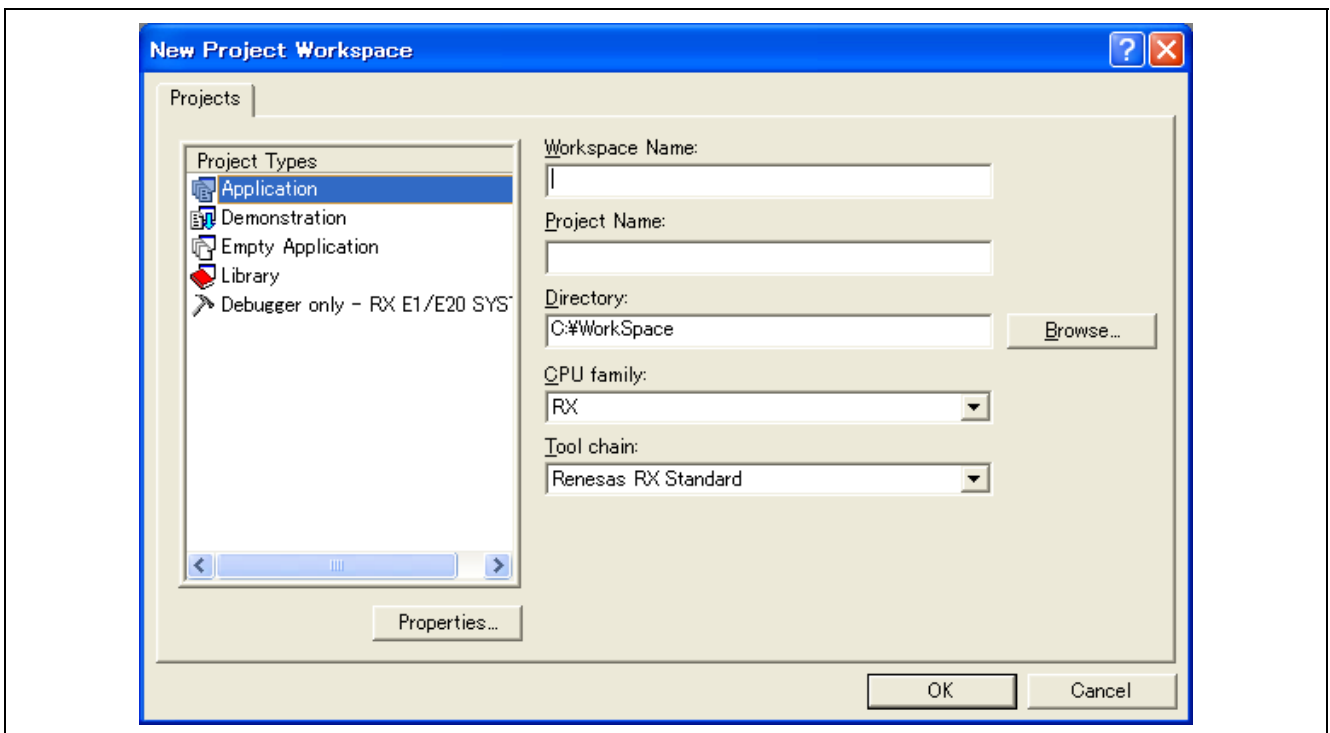


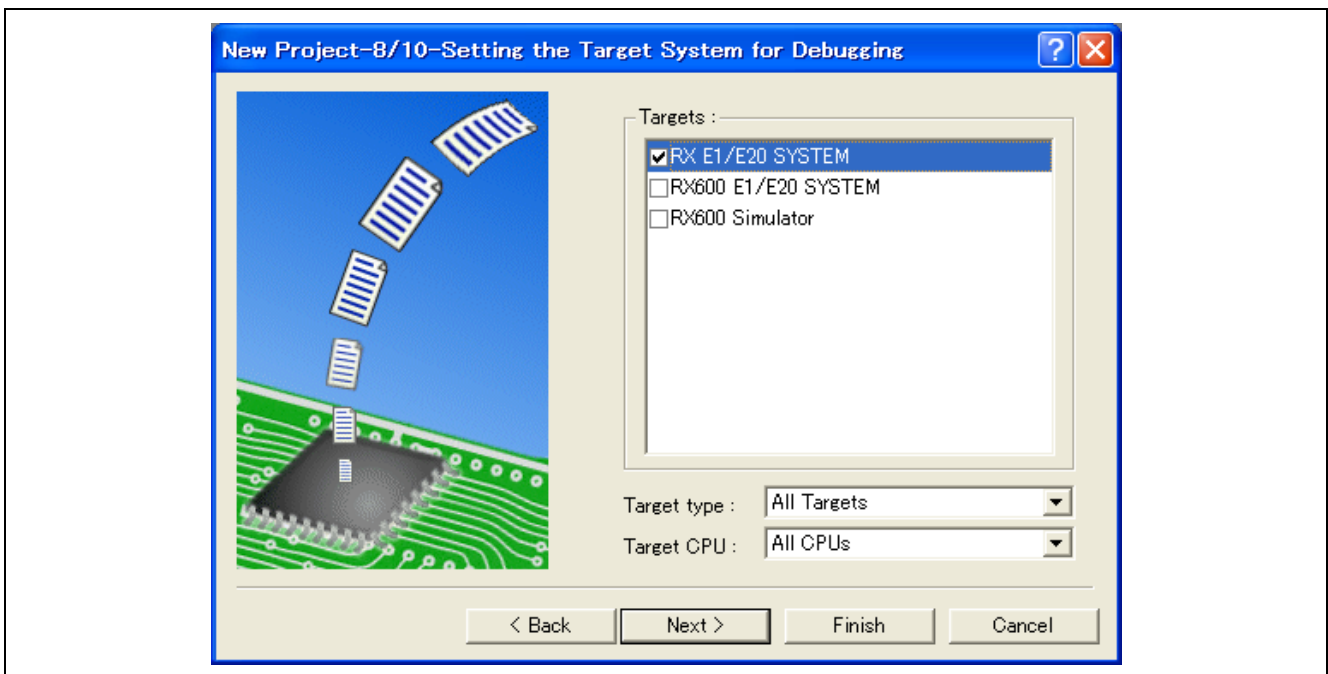
Figure 2.7 [New Project Workspace] Dialog Box

- [Workspace Name]: Enter a new workspace name.
- [Project Name]: Enter a project name. You do not need to enter any name if you wish this to be the same as the workspace name.
- [Directory]: Enter the directory in which you want the workspace to be created. Alternatively, click on the [Browse] button and select a workspace directory from the dialog box.
- [CPU family]: Select the CPU family of the MCU you are using.
- [Tool chain]: To use a toolchain, select the appropriate toolchain here. If you do not use any toolchain, select [None].

After making these settings, click on the [OK] button.

(3) Set the CPU and options for the toolchain and make other necessary settings.

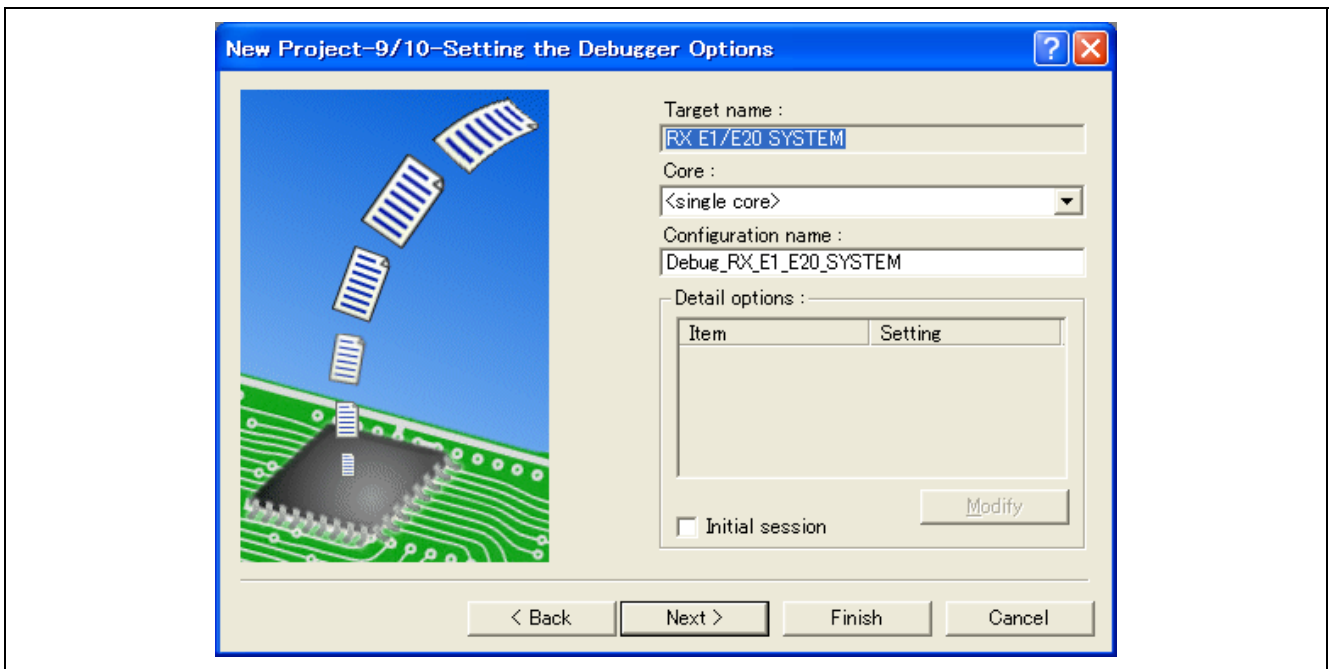
(4) Select the target for debugging.



**Figure 2.8 [New Project –8/10– Setting the Target System for Debugging] Dialog Box**

Select the target platform you wish to use by placing a check mark in its checkbox and click on the [Next] button. To use the debugger, select the checkbox “RX E1/E20 SYSTEM,” which is the target platform for the RX family. Note that the checkbox “RX600 E1/E20 SYSTEM” is provided for maintaining compatibility with the earlier versions. Do not select it.

(5) Set the configuration file name.



**Figure 2.9** [New Project -9/10- Setting the Debugger Options] Dialog Box

If you have selected two or more target platforms, click on the [Next] button and then set a configuration name for each of the selected target platforms.

When you have finished setting the configuration names, settings related to the emulator debugger have been completed.

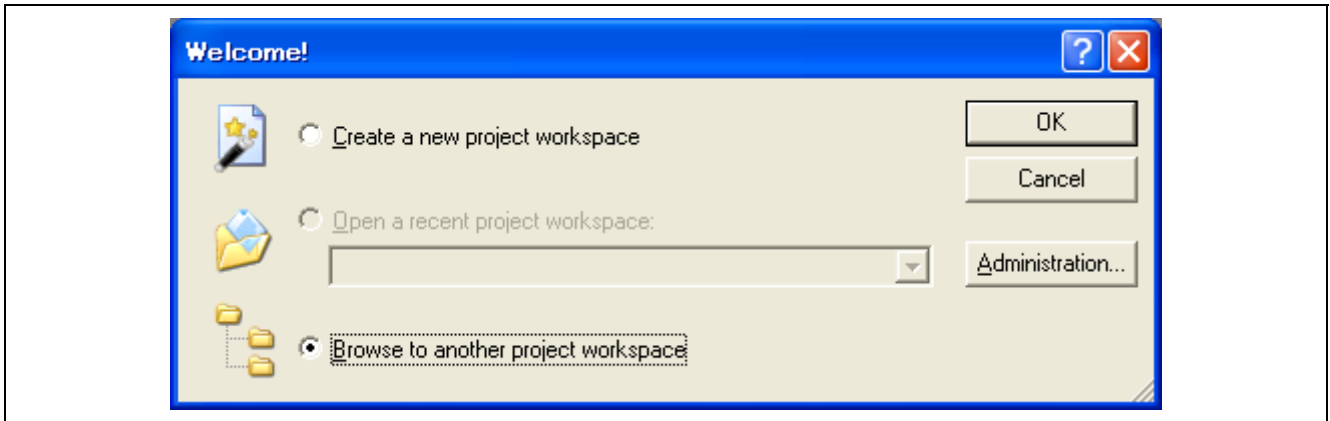
Click on the [Finish] button, and the [Summary] dialog box will be displayed. Clicking on the [OK] button in this dialog box starts the High-performance Embedded Workshop.

(6) After starting the High-performance Embedded Workshop, connect the emulator.

## 2.5 Selecting an Existing Workspace

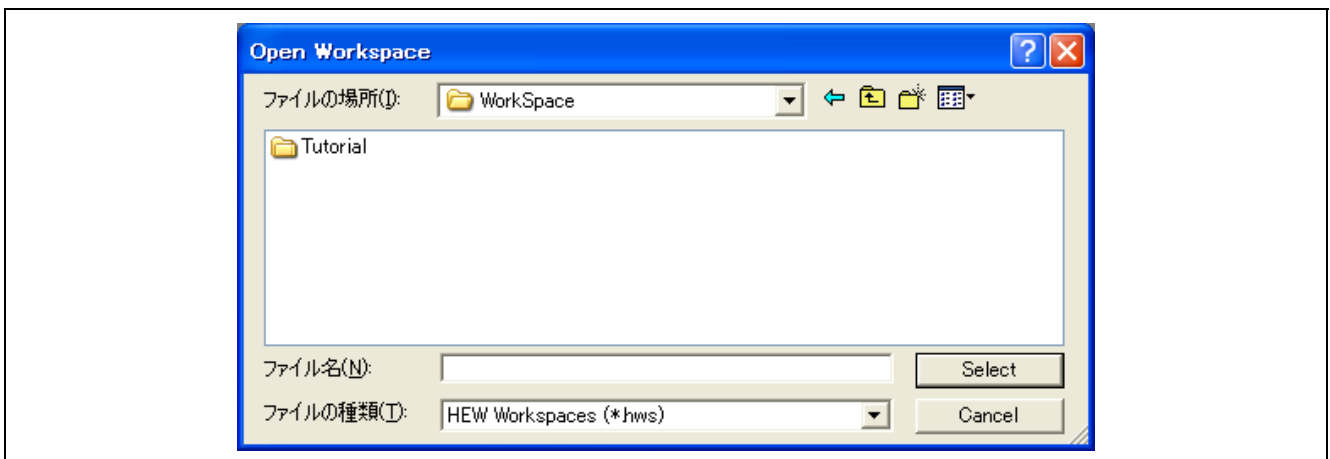
Follow the procedure below to open an existing workspace.

- (1) In the [Welcome!] dialog box, select the [Browse to another project workspace] radio button and click on the [OK] button.



**Figure 2.10 [Welcome!] Dialog Box**

- (2) The [Open Workspace] dialog box is displayed.



**Figure 2.11 [Open Workspace] Dialog Box**

Specify the directory in which the workspaces was created, select the workspace file (extension “.hws”), and click on the Select button.

- (3) The High-performance Embedded Workshop will start, and its state will be restored to the state at the time the selected workspace was saved.

If the emulator was connected at the time, the workspace is automatically connected to the emulator. If the emulator was not connected but you want to connect it, refer to section 2.6, Connecting the Emulator.



## 2.6 Connecting the Emulator

### 2.6.1 Connecting the Emulator

The following methods for connecting the emulator are available.

(a) Making the emulator settings in booting-up before connection


Choose [Debug Settings] from the [Debug] menu to open the [Debug Settings] dialog box. In this dialog box, you can select the target for debugging, and register modules for downloading and a command chain for automatic execution. When you select the target in the [Debug Settings] dialog box and then click on the [OK] button, the emulator will be connected.

(b) Loading a session file

Connect the emulator by simply switching the session file to one in which the setting for the emulator use has been registered.

### 2.6.2 Reconnecting the Emulator


While the emulator is disconnected, you can reconnect it in one of the ways described below.

- Choose [Connect] from the [Debug] menu.
- Click on the [Connect] toolbar button ().
- Enter the connect command in the [Command Line] window.

## 2.7 Disconnecting the Emulator

### 2.7.1 Disconnecting the Emulator

To disconnect the emulator while it is active, do so in one of the ways described below.

- Choose [Disconnect] from the [Debug] menu.
- Click on the [Disconnect] toolbar button ().
- Enter the disconnect command in the [Command Line] window.

## 2.8 Quitting the High-performance Embedded Workshop

Choosing [Exit] from the [File] menu closes the High-performance Embedded Workshop.

Before it closes, a message box will be displayed asking you whether you want to save the session. To save the session, click on the [Yes] button.

## 2.9 Making Debugging-Related Settings

Register download modules, set up automatic execution of command line batch files, and set download options, etc.

### 2.9.1 Specifying a Module for Downloading

Choose [Debug Settings] from the [Debug] menu to open the [Debug Settings] dialog box.

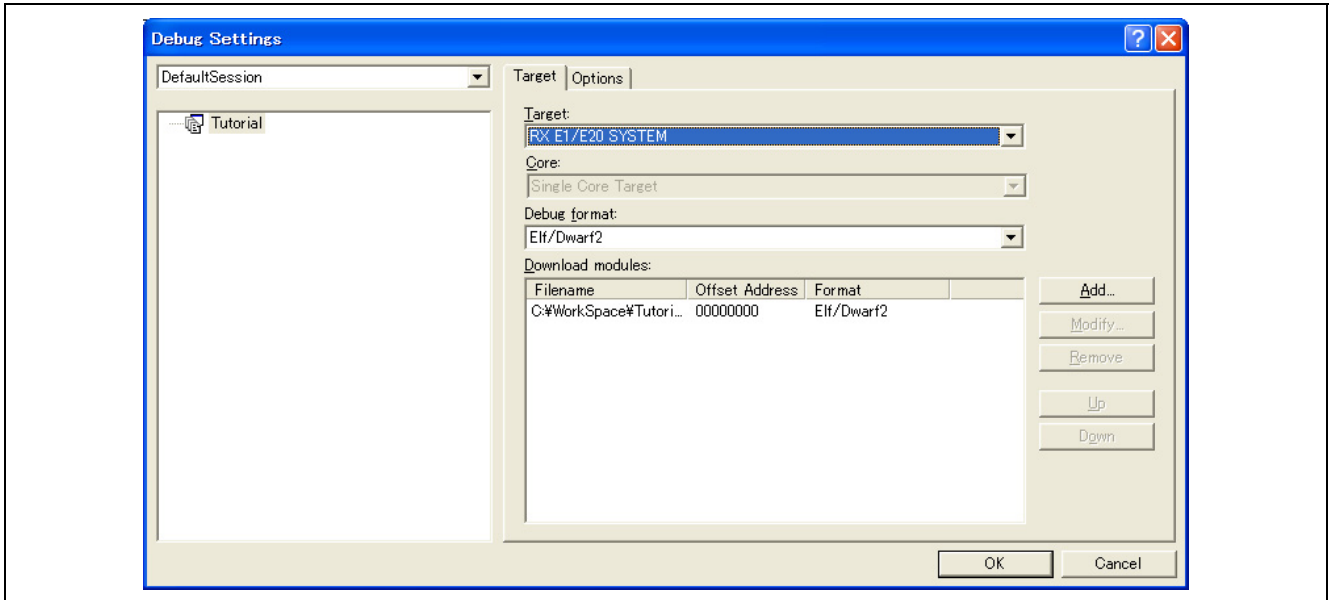


Figure 2.12 [Debug Settings] Dialog Box

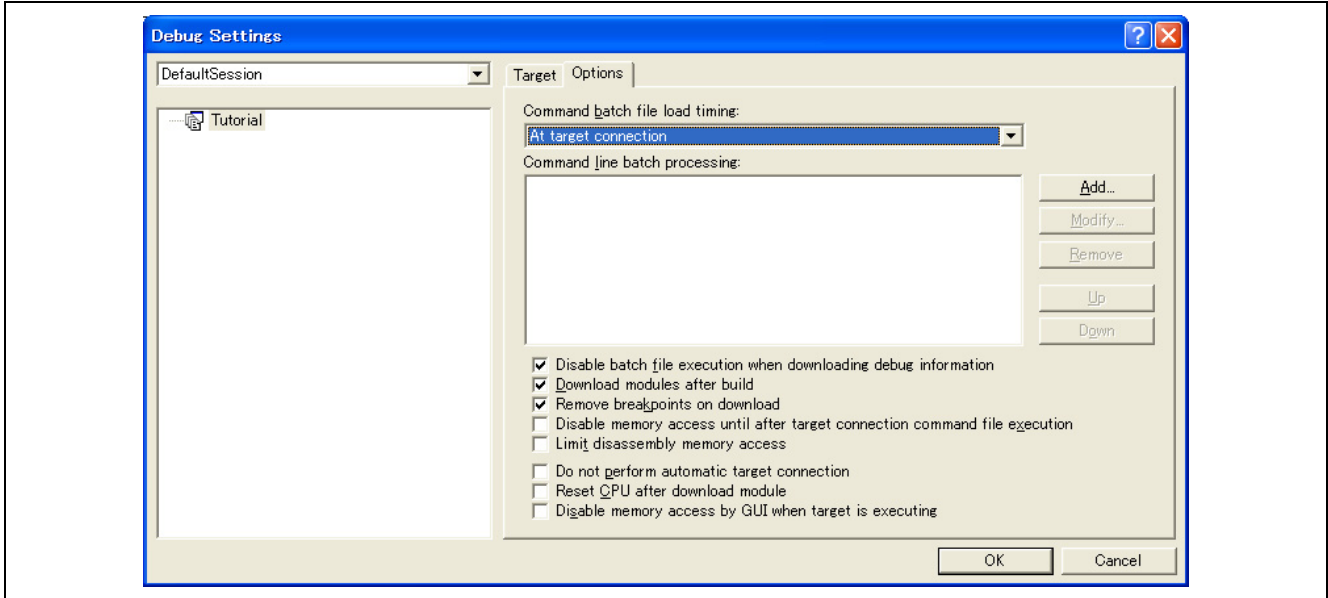
In the [Target] drop-down list box, select the name of the product you want to connect.

In the [Debug format] drop-down list box, select the format of the load module you want to download. Then register a module in the selected format in the [Download modules] list box.

Note: At this point in time, no programs have been downloaded yet. For details on how to download a program, refer to section 4.2, Downloading a Program.

### 2.9.2 Setting Up Automatic Execution of Command Line Batch Files

Click on the [Options] tab of the dialog box.



**Figure 2.13 [Debug Settings] Dialog Box**

Here, register a command chain to be automatically executed with the specified timing. Select your desired timing from among the following four choices:

- When the emulator is connected ([At target connection])
- Immediately before downloading ([Before download of modules])
- Immediately after downloading ([After download of modules])
- Immediately after a reset ([After reset])

In the [Command batch file load timing] drop-down list box, select the timing with which you want a command chain to be executed. Then add the command-batch files you wish to execute to the [Command Line Batch Processing] list box.

## 2.10 Procedure for Launching the E1/E20 Emulator Debugger

This section covers how to start up the High-performance Embedded Workshop and check the connection of the emulator and the MCU on the user system.

Here, use the workspace included as a tutorial for the product.

Take the following steps beforehand.

- Check that the power for the user system is off.
- Connect one end of the user-system interface cable to the user-side connector on the emulator and the other end to the connector on the user system.
- Connect the emulator and host machine via the USB interface cable.

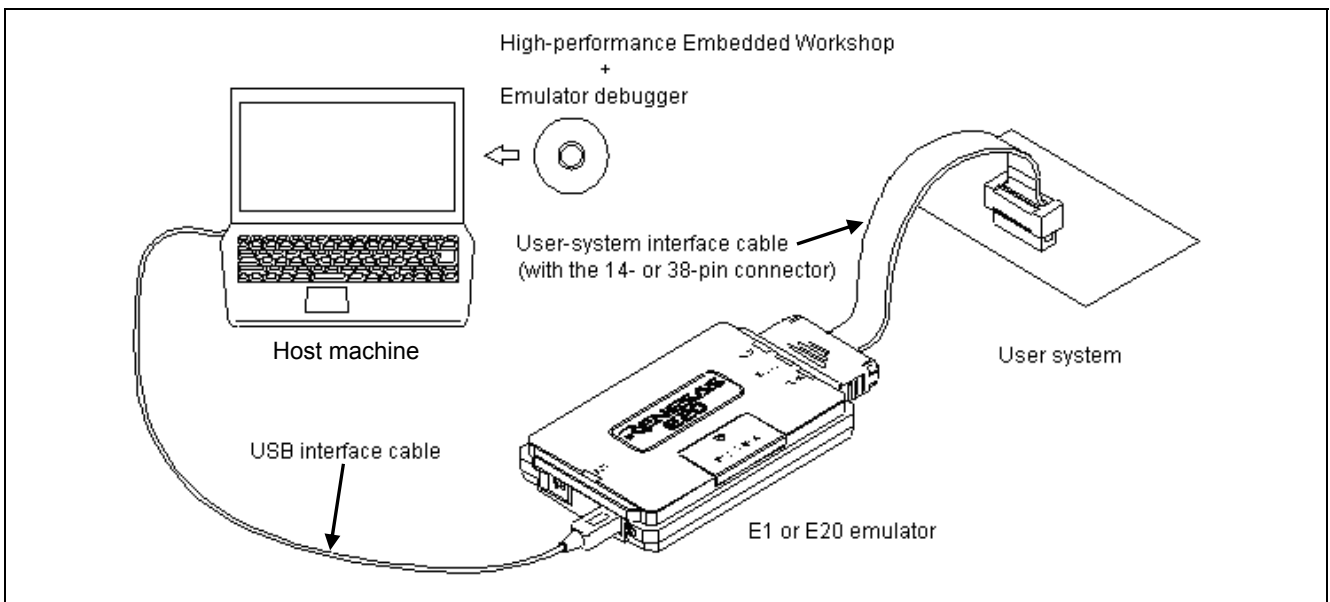


Figure 2.14 Configuration for System Check

- (1) Open the start menu and select [Programs -> Renesas -> High-performance Embedded Workshop -> High-performance Embedded Workshop].

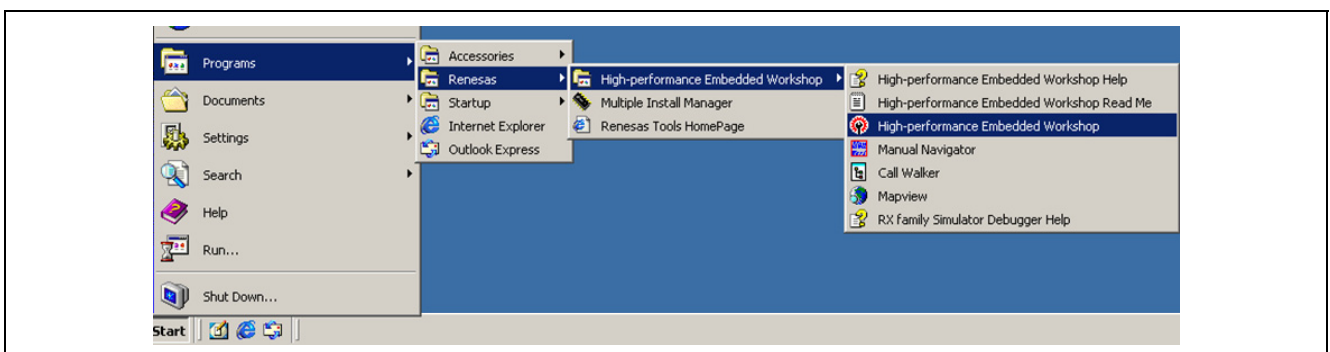
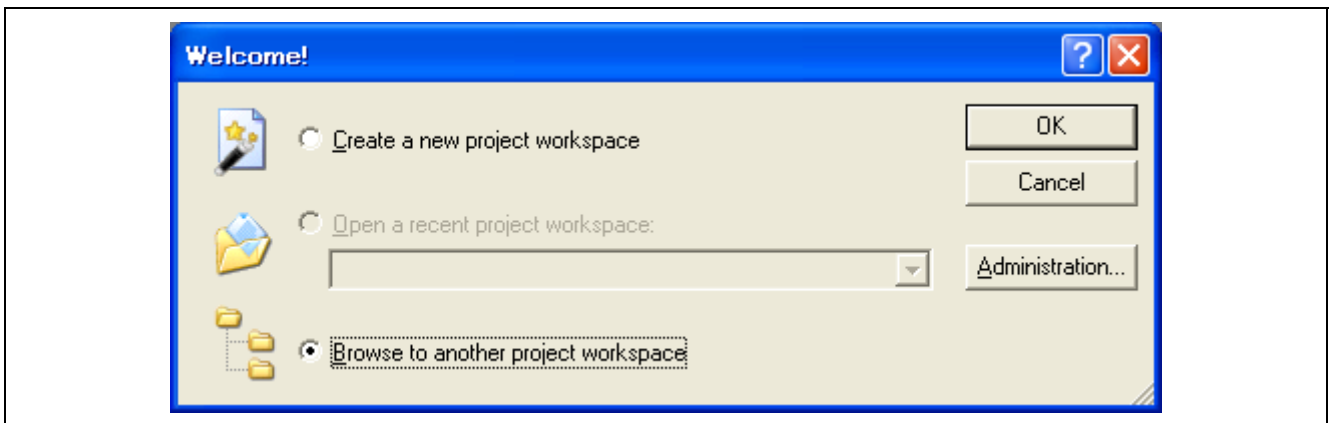


Figure 2.15 [Start] Menu

(2) The [Welcome!] dialog box is displayed.



**Figure 2.16 [Welcome!] Dialog Box**

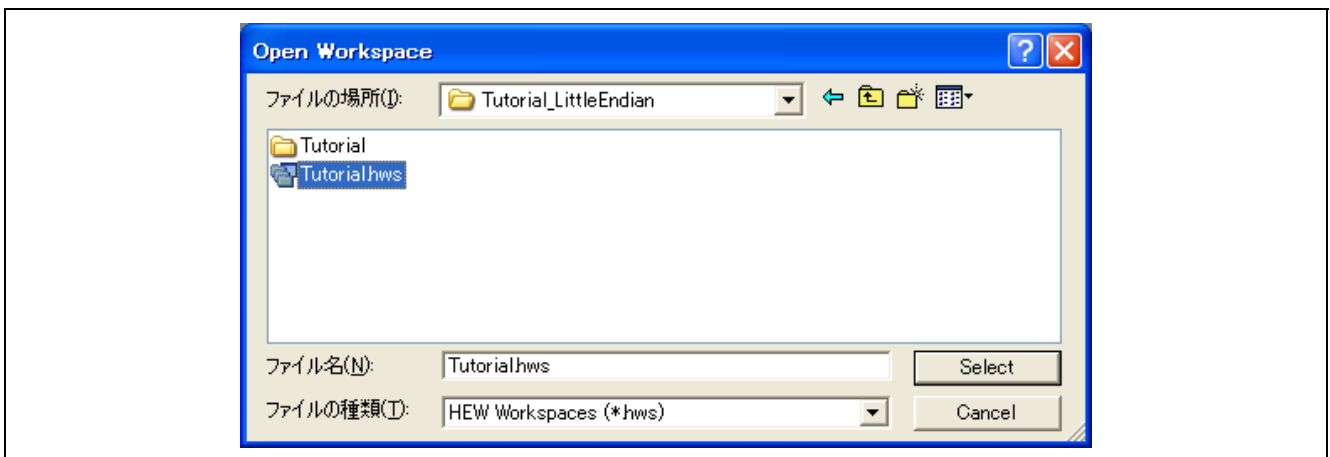
To use a workspace for the tutorial, select the [Browse to another project workspace] radio button and click on the [OK] button.

When the [Open Workspace] dialog box is opened, specify the following directory:

<Drive where the OS has been installed>: \WorkSpace\Tutorial\E1E20\RXxxx\Tutorial\_LittleEndian

Here, 'xxx' means the target product group.

After the directory has been specified, select the file "Tutorial.hws" and click on the [Select] button.



**Figure 2.17 [Open Workspace] Dialog Box**

(3) The [Initial Settings] dialog box is displayed.

In this dialog box, make the target MCU-related settings and startup and communication settings.

After setting up each item, click the [OK] button.

For details about the [Initial Settings] dialog box, refer to section 3.3.1, [Initial Settings] Dialog Box.

For hot plug-in, refer to section 4.13, Hot Plug-in Function.

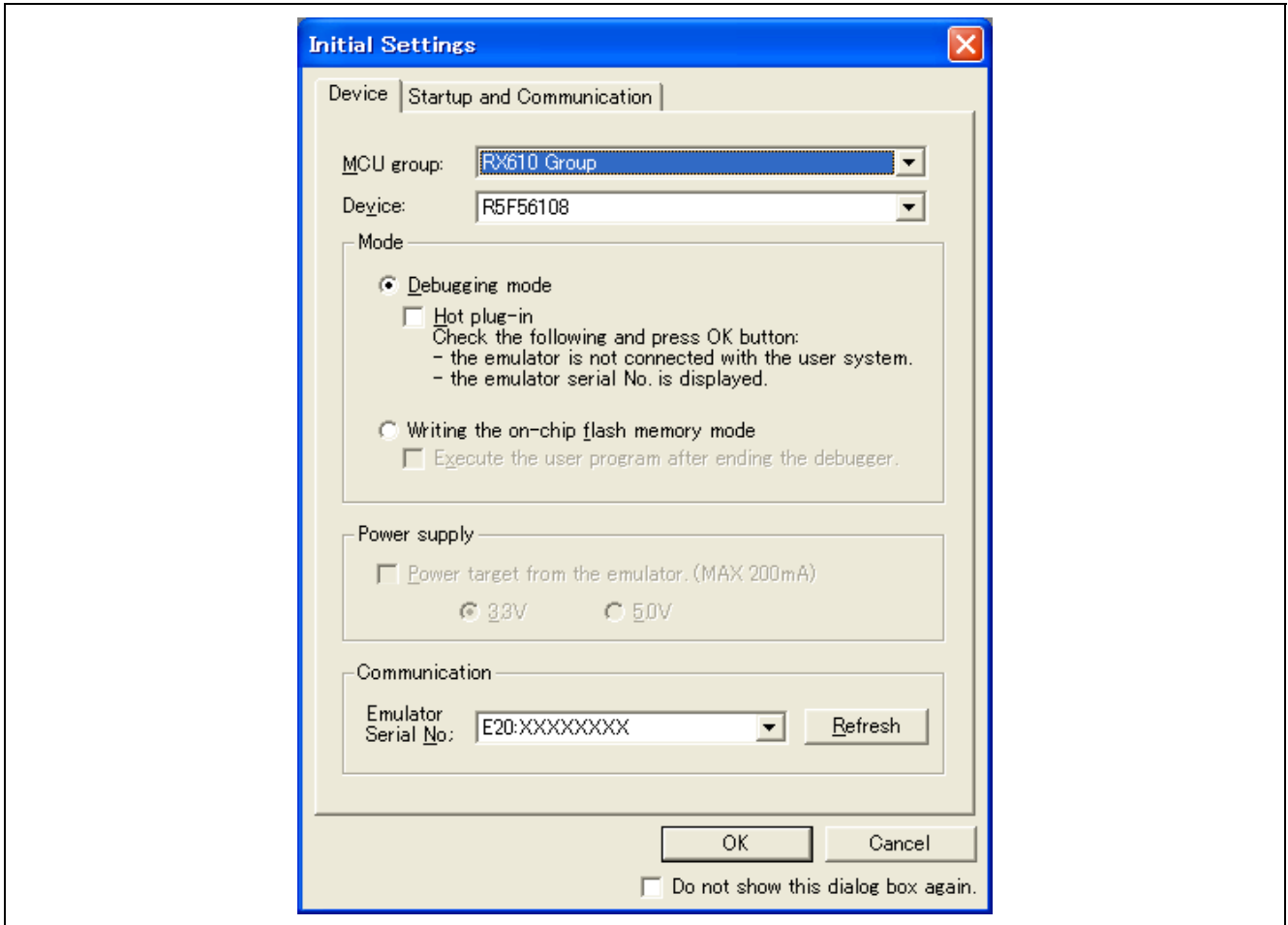


Figure 2.18 [Initial Settings] Dialog Box ([Device] Page)

Note: The items shown in the [Initial Settings] dialog box vary with the MCU.

The following two notes only apply to the E1 (because the E20 is not capable of supplying power to the user system).

[When the Emulator is to Supply Power to the User System]

If you have not selected the [Power target from the emulator] checkbox and there is no external power supply for the user system, the following dialog box appears.

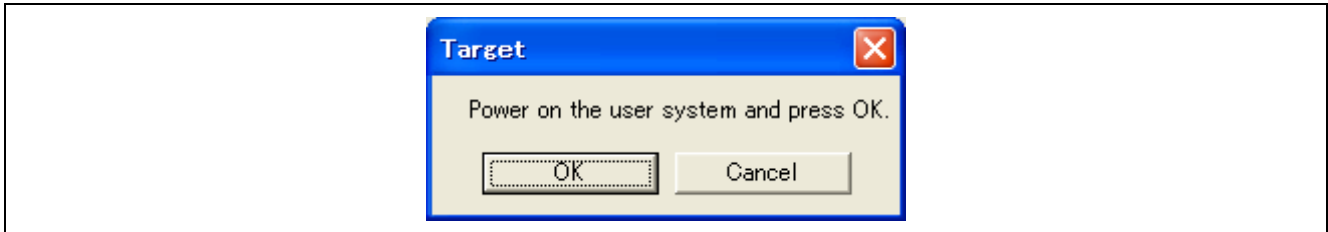


Figure 2.19 Lack of Power Supply

Clicking on the [OK] button opens the [Power Supply] dialog box. Select the checkbox and then specify the power supply voltage. Click on the [OK] button, and the emulator will start supplying power to the user system.

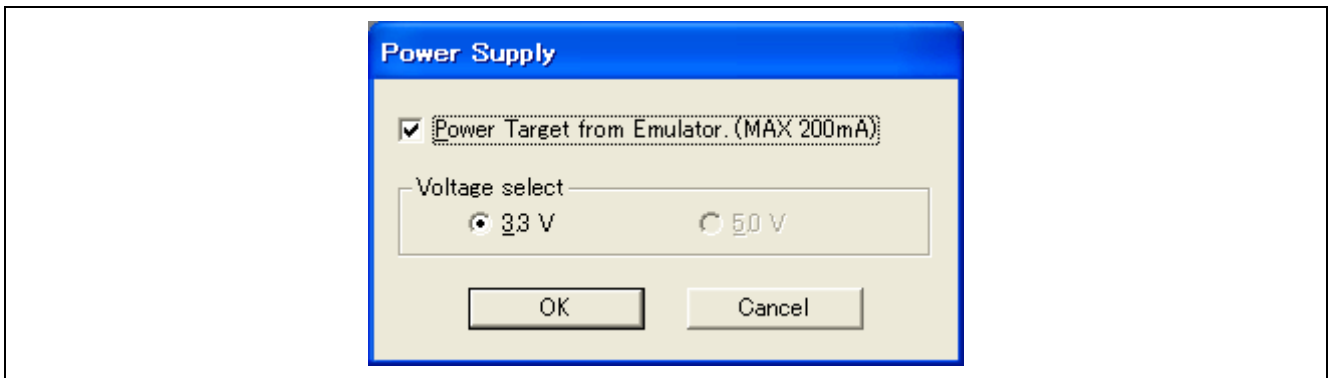


Figure 2.20 [Power Supply] Dialog Box

[When an External Power Supply is Used to Supply Power to the User System]

If you have selected the [Power target from the emulator] checkbox and an external power supply is supplying power to the user system, the following dialog box appears.



Figure 2.21 External Power for the User System Dialog Box

Clicking on the [OK] button opens the [Power Supply] dialog box. Deselect the [Power Target from Emulator]checkbox and click on the [OK] button to stop the emulator supplying power to the user system.

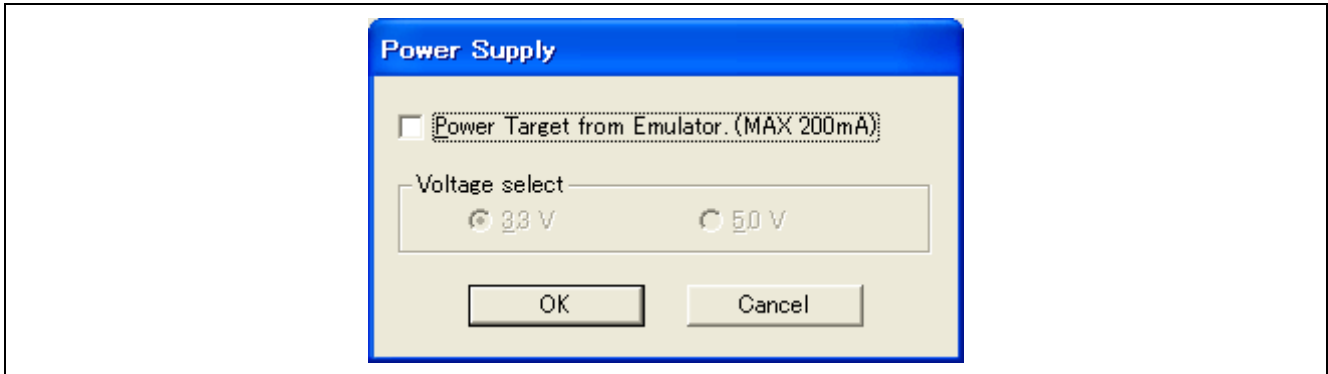


Figure 2.22 [Power Supply] Dialog Box

(4) Clicking on the [OK] button on the [Initial Settings] dialog box opens the [Connecting...] dialog box.

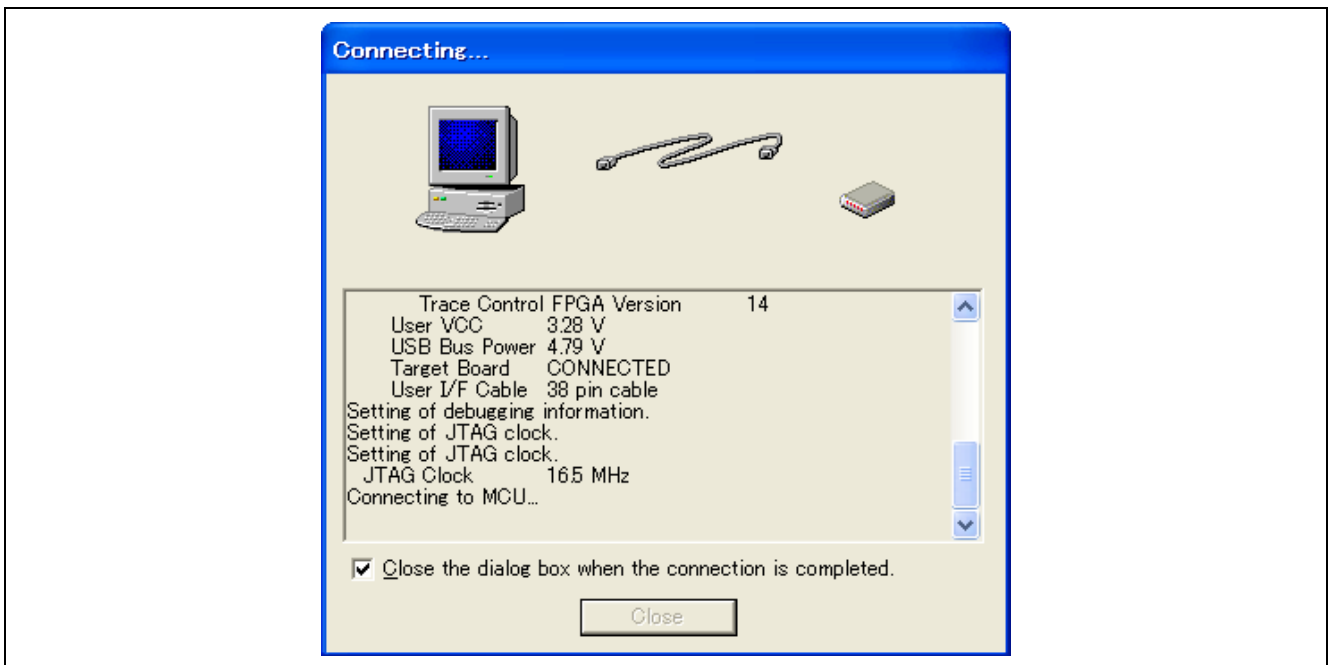


Figure 2.23 [Connecting...] Dialog Box



(5) When updating of the emulator firmware is required, the [Confirm Firmware] dialog box appears.

Clicking on the [OK] button starts updating of the emulator firmware. Once started, you cannot cancel the updating.



Figure 2.24 [Confirm Firmware] Dialog Box

The [Downloading] progress bar is displayed during the process of updating. Booting-up of the emulator resumes after the firmware has been updated.

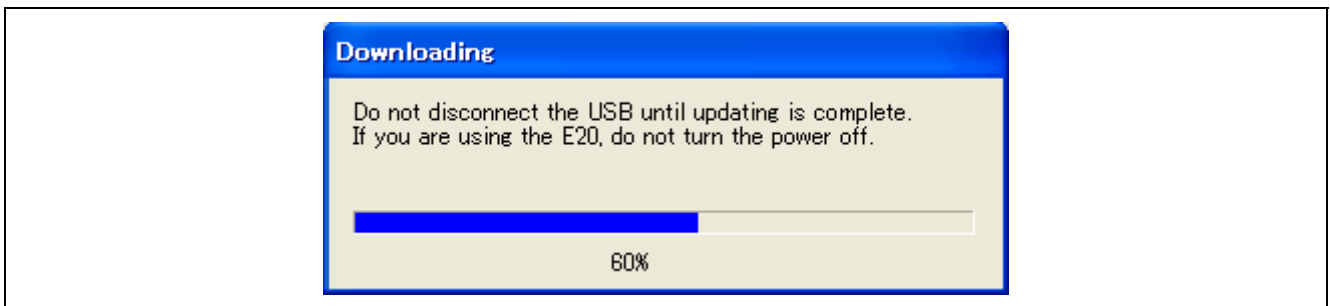
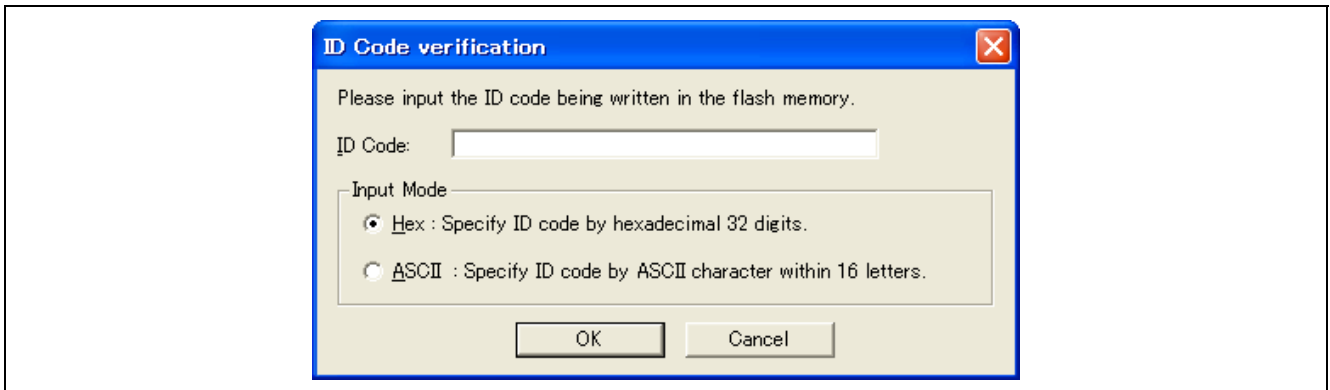


Figure 2.25 [Downloading] Progress Bar

## CAUTION

**The USB interface cable must not be disconnected until writing is complete. Early disconnection may damage the emulator.**

- (6) If an ID code has been set in the target MCU's on-chip flash memory, the [ID Code verification] dialog box appears. Enter the ID code and click on the [OK] button.

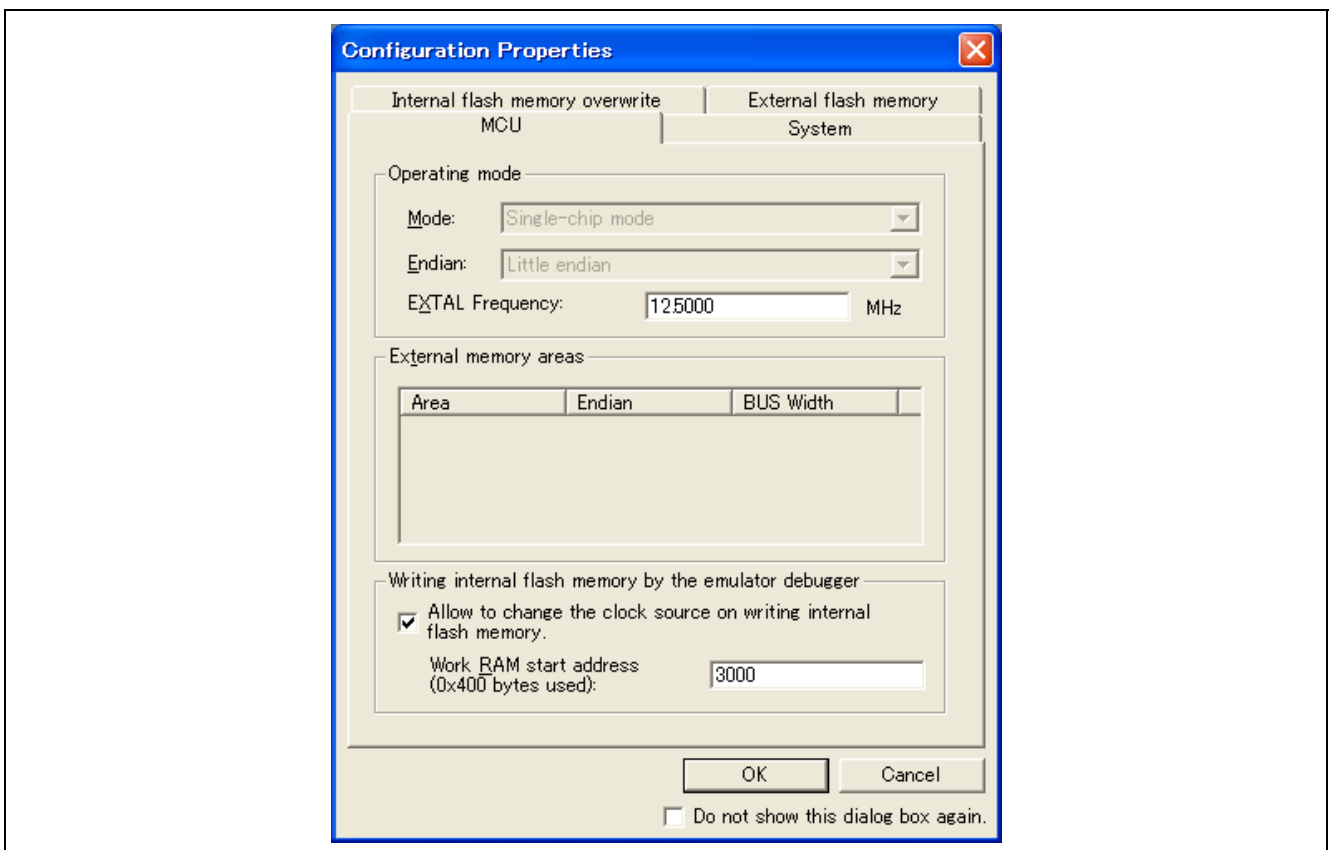


**Figure 2.26 [ID Code verification] Dialog Box**

- (7) The [Configuration Properties] dialog box appears.

Settings related to emulator operation and debugging are made in this dialog box. After setting up each item, click the [OK] button.

For details about the [Configuration Properties] dialog box, refer to section 3.3.2, [Configuration Properties] Dialog Box.



**Figure 2.27 [Configuration Properties] Dialog Box ([MCU] Page)**

Note: The items shown in the [Configuration Properties] dialog box vary with the MCU.

(8) When "Connected" is displayed in the [Output] window of the High-performance Embedded Workshop, the emulator initiation is completed.

If any other message is displayed in the [Output] window, see Table 2.3.

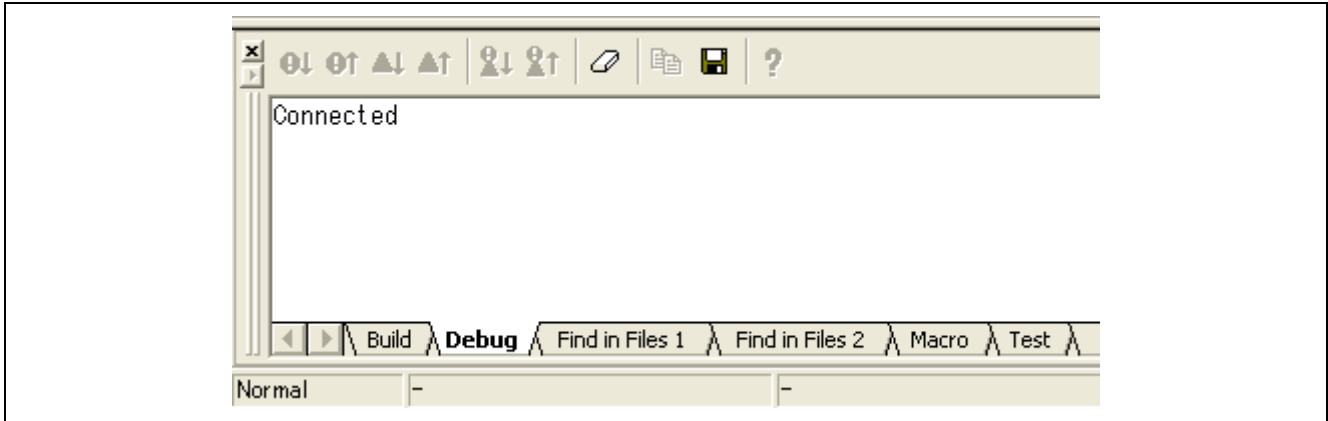


Figure 2.28 [Output] Window

Note: When the user program has already been downloaded to the flash memory, source-level debugging is not possible because there is no debugging information on the user program after the emulator has been activated. To perform source-level debugging, download a load-module file that includes debugging information after the emulator has been activated.

Table 2.3 Error Messages in the [Output] Window

Error Message	Description
The version of LEVEL0 is not corresponding to the debugger.	To use the emulator, the current debugger software needs to be updated. Download the updated version from our website.
The device ID code does not match the one for the selected device. Please check the device name.	The MCU name selected in the [Initial Settings] dialog box does not match the actual MCU in use. Check the MCU name again.
The power voltage has exceeded 5.9V. Please check the user system setting.	The voltage being supplied to the user system may be greater than the defined limit. Check the user-system settings.
A JTAG communication error. Please retry with reducing the JTAG clock.	Check the followings: <ul style="list-style-type: none"> <li>- The connection of the user-system interface cable may be incorrect.</li> <li>- The connection of the MCU pins may be incorrect.</li> <li>- The JTAG frequency may be too high or low.</li> </ul>
Failed to communicate with the mcu because its source clock of serial communication is changed. Please retry with the baud rate reduced.	Check the followings: <ul style="list-style-type: none"> <li>- The connection of the user-system interface cable may be incorrect.</li> <li>- The connection of the MCU pins may be incorrect.</li> <li>- The FINE baud rate may be too high or low.</li> </ul>

## Section 3 Software Specifications

### 3.1 Specifications of the E1 and E20 Emulators

Table 3.1 lists specifications of the E1 and E20 emulators that apply when you're using the RX610, RX621, RX62N, RX62T, RX62G, RX630, RX631, RX63N or RX63T Group. Table 3.2 lists specifications of the E1 and E20 emulators that apply when you're using the RX210, RX21A or RX220 Group.

Note that the number of events you can set, as well as the set conditions for the on-chip break, trace and performance functions vary with the target MCU. Functional specifications differing with each MCU series are summarized in Table 3.3.

**Table 3.1 Specifications of the E1 and E20 Emulators for the RX610, RX621, RX62N, RX62T, RX62G, RX630, RX631, RX63N and RX63T Groups**

Item	E1	E20	
Target MCUs	RX Family, RX600 Series RX610, RX621, RX62N, RX62T, RX62G RX630, RX631, RX63N and RX63T Groups	←	
Target operation mode* <sup>1</sup>	Single chip mode, on-chip ROM enabled extended mode, on-chip ROM disabled extended mode, and user boot mode	←	
S/W break	Maximum of 256 points	←	
Events (access by the DMAC or DTC is not detectable)	Number of events* <sup>2</sup>	Execution address: 8 points Data access: 4 points	←
	Number of passes* <sup>3</sup>	Maximum of 256	←
On-chip breakpoints	Pre-PC break* <sup>2</sup>	Maximum of 8 points	←
	Combination of events* <sup>2</sup>	OR, AND (cumulative), or sequential	←
	Other	Trace full break	←
Trace (access by the DMAC or DTC is not traceable)	Internal trace (acquisition of information on branches and data access for up to 256 branches or cycles)	<ul style="list-style-type: none"> <li>Internal trace (acquisition of information on branches and data access for up to 256 branches or cycles)</li> <li>External trace output (branch information on approx. 2M jumps or cycles and data access information acquired)</li> </ul>	

**Table 3.1 Specifications of the E1 and E20 Emulators for the RX610, RX621, RX62N, RX62T, RX62G, RX630, RX631, RX63N and RX63T Groups (cont)**

Item	E1	E20
Performance measurement	Numbers of cycles for execution for up to two sections or number of times executed (32-bit counter x 2 or 64-bit counter x 1)	←
Realtime RAM monitor (access by the DMAC or DTC cannot be monitored)	None	4 Kbytes (1 Kbyte x 4 blocks) Displays a history of data access through reading and writing
Debug console	Displays printf output in an output window	←
User interface	14-pin connector	38-pin connector or 14-pin connector**4
Interface with host machine	USB2.0 (full speed or high speed)*5	←
Connection to the user system	Connection by the provided user-system interface cable	←
Power supply for the emulator	No need (the host machine supplies power through the USB)	←
Power supply function*6	3.3 V or 5.0 V can be supplied to the user system (with current up to 200 mA)	None
Downloading to external flash memory	Available	←
Hot plug-in*7 *8	Available	←
Communication interface*9 *10	JTAG, FINE	←

- Notes:
1. The selectable target operation mode differs with each target MCU.
  2. Events are common for on-chip breakpoints and tracing. A given event cannot be used in more than one capacity at the same time.
  3. Number of passes can be specified for any single event.
  4. If the external trace-output and realtime RAM monitoring functions are not in use, the 14-pin connector can be used to connect the E20 emulator. In this case, the 38-pin to 14-pin conversion adapter is required.
  5. Connection to the host machine via USB1.1 is also possible.
  6. Do not use the power-supply function of the E1 emulator when it is being used to write a program that requires reliability. Separately supply power to the user system in accord with the specifications of the MCU. When writing a program for mass production processes, use the FDT.
  7. To use the hot plug-in function with the E1 emulator, please purchase the separately available hot-plug adapter (R0E000010ACB00).
  8. The hot plug-in function via FINE interface is not supported.
  9. The usable interface differs with each target MCU. For details about the interface, refer to the hardware manual for the MCU you're using. Note that the RX610, RX621, RX62N, RX62T and RX62G Groups cannot use the FINE interface.
  10. FINE interface is supported only for 2-wire system using FINEC and MD/FINED pins.

Table 3.2 Specifications of the E1 and E20 Emulators for the RX210, RX21A and RX220 Group

Item	E1	E20
Target MCUs	RX Family, RX600 Series RX210, RX21A and RX220 Group	←
Target operation mode	Single chip mode, on-chip ROM enabled extended mode, on-chip ROM disabled extended mode, and user boot mode	←
S/W break	Maximum of 256 points	←
Events (access by the DMAC or DTC is not detectable)	Number of events* <sup>1</sup>	Execution address: 4 points Data access: 2 points
	Number of passes	None
On-chip breakpoints	Pre-PC break* <sup>1</sup>	Maximum of 4 points
	Combination of events* <sup>1</sup>	OR, AND (cumulative), or sequential
	Other	Trace full break
Trace (access by the DMAC or DTC is not traceable)	Internal trace	←
	(acquisition of information on branches and data access for up to 64 branches or cycles)	
Performance measurement	Numbers of cycles for execution for one section (24-bit counter x 1)	←
Realtime RAM monitor (access by the DMAC or DTC cannot be monitored)	None	←
Debug console	Displays printf output in an output window	←
User interface	14-pin connector	← * <sup>4</sup>
Interface with host machine	USB2.0 (full speed or high speed)* <sup>2</sup>	←
Connection to the user system	Connection by the provided user-system interface cable	←
Power supply for the emulator	No need (the host machine supplies power through the USB)	←
Power supply function* <sup>3</sup>	3.3 V or 5.0 V can be supplied to the user system (with current up to 200 mA)	None
Downloading to external flash memory	Available	←
Hot plug-in	None	←
Communication interface* <sup>5</sup>	FINE	←

- Notes: 1. Events are common for on-chip breakpoints and tracing. A given event cannot be used in more than one capacity at the same time.
2. Connection to the host machine via USB1.1 is also possible.
3. Do not use the power-supply function of the E1 emulator when it is being used to write a program that requires reliability. Separately supply power to the user system in accord with the specifications of the MCU. When writing a program for mass production processes, use the FDT.
4. The 38-pin to 14-pin conversion adapter is required to use the E20 emulator.
5. FINE interface is supported for 1-wire system using MD/FINED pin.

Table 3.3 List of Functional Specifications by Target MCU

Function item		RX600 Series MCUs	RX200 Series MCUs
Event			
Detection Target		CPU bus only	←
Execution address	No. of points set	8 points	4 points
Data access	No. of points set	4 points	2 points
	Detection condition	Address condition specification: Address mask, Compare condition, Address range (1 point only)  Data condition specification: Read/Write, Access size, Data mask, Compare condition	Address condition specification: Address mask, Compare condition,  Data condition specification: Read/Write, Access size, Data mask, Compare condition
Pass count condition		Available (1 point only)	None
Combination of events	Condition	OR, AND (cumulative), or Sequential	←
	No. of points set	OR, AND (cumulative): Maximum of 8 points  Sequential: 7 steps (forward direction) + reset point	OR, AND (cumulative): Maximum of 6 points  Sequential: 3 steps (forward direction) + reset point
Trace			
Detection Target		CPU bus only	←
Display mode		BUS, DIS, SRC (mixed display possible)	←
External trace	Trace capacity	Selectable from 1, 2, 4, 8, 16 and 32 Mbytes	None
	Trace full break	Available	None
	Trace operation	Get trace, stop temporarily or restart during program execution	None
Internal trace	Trace capacity	Maximum of 256 branches	Maximum of 64 branches
	Trace full break	Available	Available
	Trace operation	None	None
Trace mode		Fill until full, Fill until stop	←
Trace output		CPU execution priority, Trace output priority, Do not output (internal trace buffer used)	Do not output (internal trace buffer used)
Trace type		Branch, Branch + Data, Data	Branch(Src), Branch, Branch(Src) + Time, Data, Data + Time
Trace data extraction	Condition	Combinatorial [OR] of data access events	←
	Advanced setting	Access extraction setting Exception, FPU, Bit operation, Logical operation, Arithmetical operation, String operation, Stack operation, Data transfer	None
Start condition		Start trace acquisition by event combination [OR /AND (cumulative) / sequential]	←
Stop condition		Terminate trace acquisition by event combination [OR]	←

Table 3.3 List of Functional Specifications by Target MCU (cont)

Function item	RX600 Series MCUs	RX200 Series MCUs
Performance		
Counter clock	ICLK	←
Counter channels	2 channels (32 bit) or 1 channel (64 bit)	1 channel (24 bit)
Measurement item	Execution cycle, Execution cycle (supervisor mode), Exception and interrupt cycle, Exception cycle, Interrupt cycle, Execution count, Exception and interrupt count, Exception count, Interrupt count, Event match count	Execution cycle
Display cycles as time span	Available	←
Measure performance only once	Available	←
Add events from function information	Available	←
64-bit counter	Available	None
Start condition	Start performance by event combination (OR)	←
Stop condition	Start performance by event combination (OR)	←



### 3.2 Differences between the MCU and the Emulator

(1) When the emulator system is initiated, it initializes the general registers and part of the control registers as shown in Table 3.4. The initial values of the MCU are undefined.

**Table 3.4 Register Initial Values at Emulator Link Up**

Register	Emulator at Link Up
R0 (SP)	00000000h
R1 to R15	00000000h
USP	00000000h
ISP	00000000h
PSW	00000000h
PC	Value of the SP in the power-on reset vector table
INTB	00000000h
BPSW	00000000h
FINTV	00000000h
FPSW*2	00000100h
ACC*1	00000000h

Notes: 1. Bits 15 to 0 in the ACC register are always read as 0. Attempts at writing to those bits will be ignored.  
2. The RX210 RX21A and RX220Group MCUs do not have the floating point status word (FPSW).

Notes: If register values are changed in the [Register] window, it is immediately before the user program starts running that the changes are actually reflected in the registers. The same applies when the REGISTER\_SET command is used to change register values.

If, after register values are changed in the [Register] window or with the REGISTER\_SET command, the CPU is reset without executing the user program even once, the changed register values have no effect.

#### (2) Low-Power States

When the emulator is used, release from power-down modes can be accomplished by a source for release or by pressing the [STOP] button, causing a break to occur.

#### (3) Reset Signals

If, while the user program remains halted, a reset is asserted by means of a pin reset, the reset signal is masked off during a JTAG connection. Note, however, that during a FINE connection, it is not masked and the MCU is reset.

Note: Do not break the user program when the RES#, or WAIT# signal is being low. A TIMEOUT error will occur. If the WAIT# signal is fixed to low during break, a TIMEOUT error will occur at memory access.

#### (4) Direct Memory Access Controller (DMAC)

The DMAC operates even during breaks in execution. Therefore, when a data transfer request is generated, the DMAC executes DMA transfer. Also, refer to section 6.19.2, DMAC and DTC.

#### (5) Using WDT

Counting by the watchdog timer is suspended during breaks in execution. When execution of the user program is restarted, the counting is also resumed.

## (6) Contention between Clock Generator Circuit-Related Register Modifications and Debug Functions

Do not change the values of clock generator circuit-related registers while the user program is under execution. For details about the clock generator circuit-related registers, refer to the hardware manual for the MCU you're using.

## (7) Occupancy of User Pins

Due to the following pin functions being multiplexed with pin functions for a user port, the user port is not available to the user during debugging. See the user manual for the MCU regarding multiplexing of pins for user ports.

- TRST#
- TCK
- TMS
- TDI
- TDO
- FINEC

Note that the RX621 and RX62N Group MCUs have such a characteristic that when its TDO-assigned pin is set for open-drain output, the pin becomes an open-drain output even though it is used as TDO. Therefore, do not set P26 for open-drain output during debugging.

When the E20 emulator is connected to the user system via the 38-pin connector, the following pins cannot be used. The pins can be used for their user-port functions when the 38-pin to 14-pin conversion adapter is in use.

- TRCLK
- TRSYNC
- TRDATA[3:0]

## (8) Notes on Maskable Interrupts

- Even if a user program is not being executed (including when run-time debugging is being performed), timers and other peripherals do not stop running. If a maskable interrupt is requested when the user program is not being executed (including when runtime debugging is being performed), the maskable interrupt request cannot be accepted, because the emulator disables interrupts. The interrupt request is accepted immediately after the user program execution is started.
- Take note that when the user program is not being executed (including when run-time debugging is being performed), a peripheral I/O interruption is not accepted.

### 3.3 Setting up the Debugger

On booting-up the emulator, the [Initial Settings] and [Configuration Properties] dialog boxes will open. Here, select the general options associated with the emulator. Note that the target MCU to be debugged, etc. can only be set once each time the emulator is booted-up.

#### 3.3.1 [Initial Settings] Dialog Box

This dialog box is used to set the target MCU. The settings remain valid the next time the emulator is booted up (except for the hot plug-in and the power-supply setting).

This dialog box can be re-opened by selecting [Setup -> Emulator -> Device setting] after the emulator has been booted up. Note, however, that settings cannot be changed after booting-up.

##### (1) Making Settings Related to the Emulator and the Target MCUs

On the [Device] page of the [Initial Settings] dialog box, specify the target MCU to be emulated and the emulation mode.

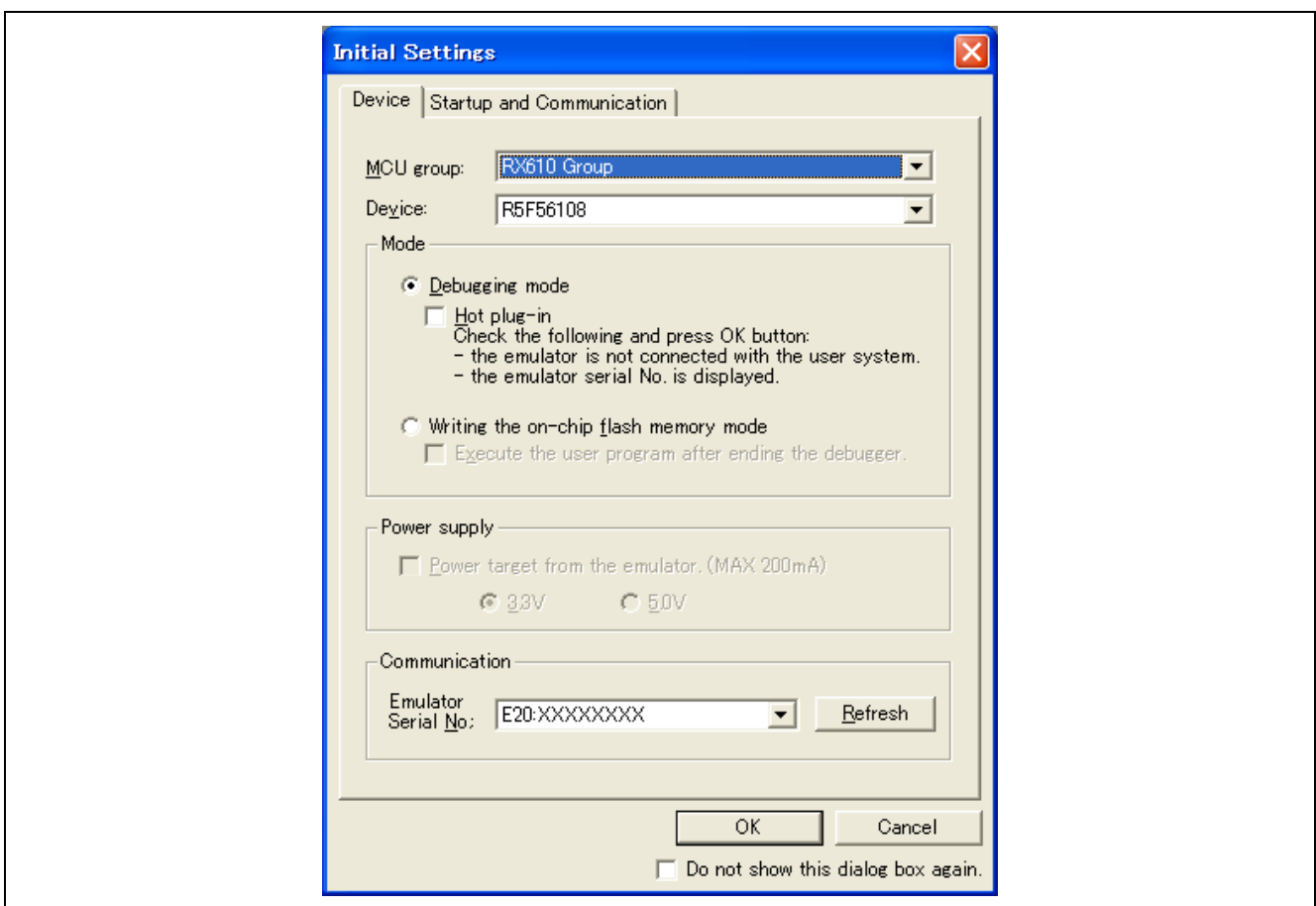


Figure 3.1 [Initial Settings] Dialog Box ([Device] Page)

Note: The items shown in the [Initial Settings] dialog box vary with the MCU.

The target MCU you have set here cannot be changed after the emulator is booted up. To change the target MCU, you need to cancel the process of booting-up and then reboot the emulator.

## (a) Selecting the Target MCU

Select the target MCU you want to emulate. First choose the target group from the [MCU group] pulldown menu and then the type name of the target MCU from the [Device] pulldown menu.

Note that the target MCUs you can select in the E1/E20 emulator debugger for the RX family are listed below.

**Table 3.5 List of Target MCUs**

MCU group	Device name	Communication interface
RX210 Group	R5F52103, R5F52104, R5F52105, R5F52106, R5F52107, R5F5210A, R5F5210B	
RX21A Group	R5F521A6, R5F521A7, R5F521A8	FINE
RX220 Group	R5F52201, R5F52203, R5F52205, R5F52206	FINE
RX610 Group	R5F56104, R5F56106, R5F56107, R5F56108	JTAG
RX621 Group	R5F56216, R5F56217, R5F56218	JTAG
RX62N Group	R5F562N7, R5F562N8	JTAG
RX62T Group	R5F562T6, R5F562T7, R5F562TA	JTAG
RX62G Group	R5F562G7, R5F562GA	JTAG
RX630 Group	R5F56307, R5F56308, R5F5630A, R5F5630B, R5F5630D, R5F5630E	JTAG, FINE
RX631 Group	R5F5631A, R5F5631B, R5F5631D, R5F5631E, R5F5631M, R5F5631N, R5F5631P	JTAG, FINE
RX63N Group	R5F563NA, R5F563NB, R5F563ND, R5F563NE	JTAG, FINE
RX63T Group	R5F563T4, R5F563T5, R5F563T6	JTAG, FINE

## (b) Selecting a Mode of Operation

Select the emulation mode from the options below.

[Debugging mode] or [Writing the on-chip flash memory mode].

[Debugging mode]

Select this mode to use the emulator as a debugger.

After a program has been downloaded, you cannot disconnect the emulator and operate the user system as a stand-alone unit. Writing of ID codes (intended to prohibit reading of the on-chip flash memory) to the on-chip flash memory is not possible in this mode.

- Select hot plug-in.

When the [Hot plug-in] checkbox is checked, it is possible to start debugging from the middle of user program execution. For details about hot plug-in, refer to section 4.13, Hot Plug-in Function.

[Writing the on-chip flash memory mode]

Select this mode to use the emulator as a flash-memory programmer.

Using the emulator for debugging is not possible in this mode. After a program has been downloaded, an ID code (intended to prohibit reading of the on-chip flash memory) is written to the on-chip flash memory.

Note that the checksum shows a value derived by adding up in bytes the whole data in three divided areas.

- Data flash area
- User boot area
- Program ROM area

(c) Selecting Whether or Not to Supply Power from the Emulator

Specify the method for supplying the power from the E1 emulator to the user system. To supply the power from the E1 emulator to the user system, check the [Power target from the emulator. (MAX 200mA)] checkbox and then select [3.3V] or [5.0V].

Note that the selectable power supply voltages vary with each target MCU. For details about the operating voltage of your MCU, refer to the hardware manual supplied with it.

Note also that this option is not selectable for the E20 because it isn't capable of supplying power to the user system.

---

**Voltage supplied from this product depends on the quality of the USB power supply of the host machine, and as such, precision is not guaranteed. When writing a program that requires reliability with this product, do not use the power supply function of this product. Supply power separately to the user system according to the allowable voltage for MCU writing. When writing a program for mass production processes, use the FDT. For details on the FDT, refer to <http://www.renesas.com/fdt>.**

---

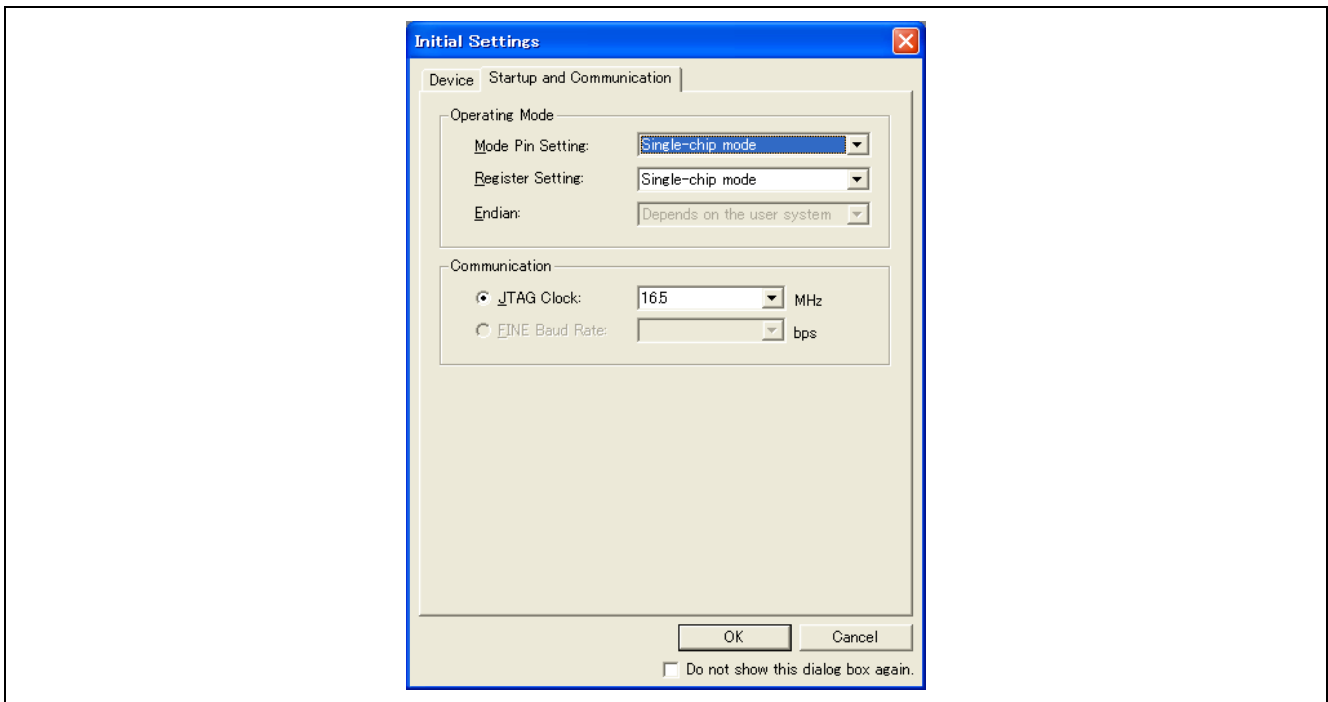
(d) Setting up Communications

You can select another target emulator for connection via USB.

The [Emulator Serial No.] list box shows unique identifying information on the emulator connected via USB. Clicking on the [Refresh] button updates the information.

## (2) Selecting the mode pin setting

In [Startup and Communication] page of the [Initial Settings] dialog box, make settings of the operation mode for the target MCU you have selected, and the communication interface.



**Figure 3.2 [Initial Settings] Dialog Box ([Startup and Communication] Page)**

The settings you have made here cannot be changed after the emulator is booted up.

To change the communication information, you need to cancel the process of booting-up and then reboot the emulator.

## (a) Selecting the Mode Pin Setting

Select the operation mode based on mode pins from the options below. Be sure to select one that matches the pin state of your system. The selectable option differs with each target MCU.

[Single-chip mode], [User boot mode]

## (b) Selecting the Register Setting

Select the option mode based on register settings from the options below. Be sure to select one that matches the operation mode that is set in the user program after startup. The selectable option differs with each target MCU.

[Single-chip mode], [On-chip ROM enabled extended mode], [On-chip ROM disabled extended mode]

## (c) Selecting the Endian

Select the endian from the options below. Note that if the target MCU you selected is a type that has the MDE pin, the options are grayed out and a remark “Depends on the user system” is displayed.

[Little endian], [Big endian]

## (d) Selecting the Interface

Select the communication interface from the options below. Be sure to select one that matches your user system. The selectable option differs with each target MCU. Refer to Table 3.5, "List of Target MCUs."

[JTAG Clock], [FINE Baud Rate]

## [JTAG Clock]

If the radio button for the JTAG clock is checked, it is possible to set the JTAG communication speed <sup>Note</sup> between the emulator and the user system. Select one from the options below (unit: MHz).

[16.5], [12.38], [6.188], [3.094], [1.547]

Note: Depending on the wired length of the JTAG signal and how it is wired on the user system, communication at the selected frequency of the JTAG clock may not be performed. Reducing this frequency may help to solve the problem.

## [FINE Baud Rate]

If the radio button for the FINE baud rate is checked, it is possible to set the FINE communication speed <sup>Note</sup> between the emulator and the user system. Select one from the options below (unit: bps).

[2000000], [750000], [500000], [250000]

Note: Depending on the wired length of the FINE signal and how it is wired on the user system, communication at the selected baud rate may not be performed. Reducing this baud rate may help to solve the problem.

### 3.3.2 [Configuration Properties] Dialog Box

This dialog box is used to make settings related to the emulator and debugging functions. The settings remain valid the next time the emulator is booted up.

This dialog box can be re-opened by selecting [Setup -> Emulator -> System] after the emulator has been booted up.

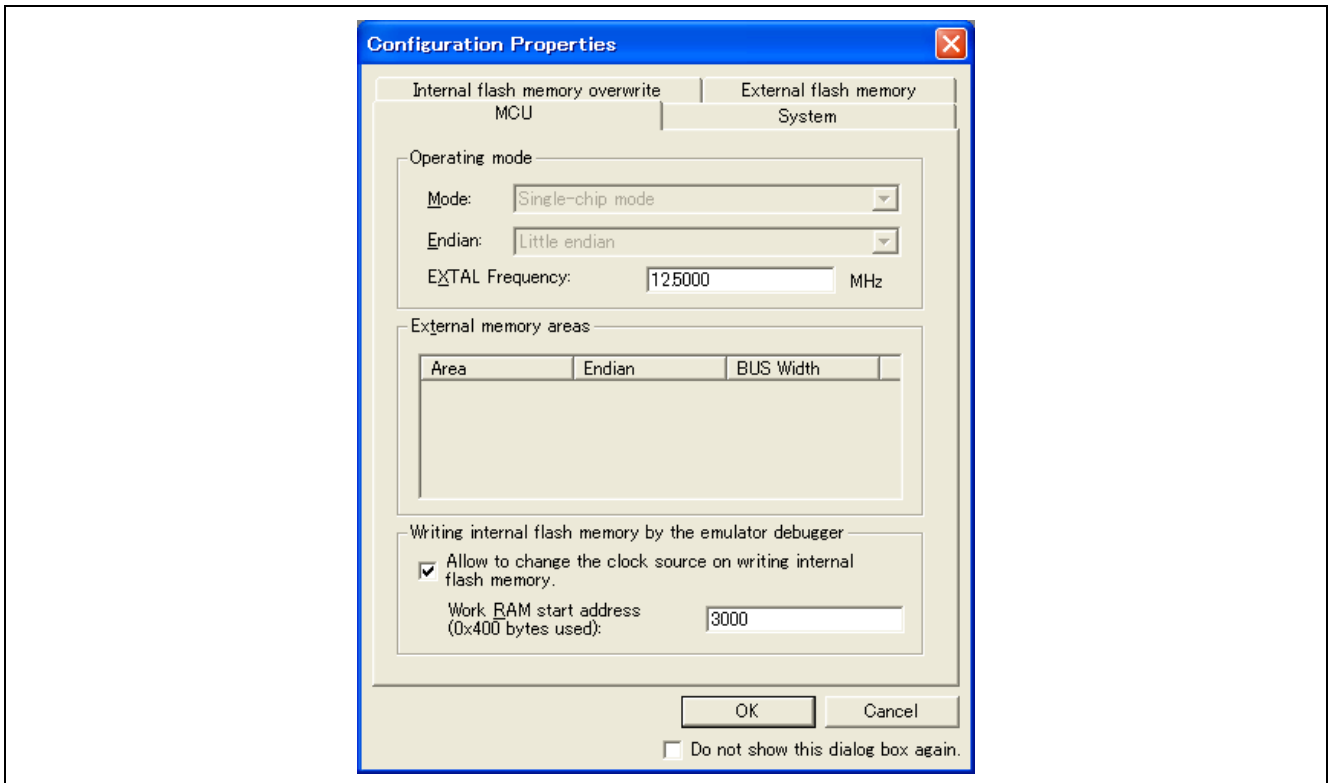
Settings for certain options in this dialog box can be changed after booting-up.

Those that can be changed are active while those that cannot are inactive (grayed out), but with their settings displayed.

#### (1) Setting Up the Emulator and Debugging Functions

On the [MCU] page of the [Configuration Properties] dialog box, make settings associated with the target MCU.

The [Configuration Properties] dialog box appears after the [Initial Settings] dialog box.



**Figure 3.3 [Configuration Properties] Dialog Box ([MCU] Page)**

Note: The items shown in the [Configuration Properties] dialog box vary with the MCU.

#### (a) Displaying the Operation Mode

An operation mode for the register setting you've selected on the [Startup and Communication] page of the [Initial Settings] dialog box is displayed.

#### (b) Displaying the Endian

Information on the endian setting you've selected on the [Startup and Communication] page of the [Initial Settings] dialog box is displayed.

Note that if the target MCU you selected is a type that has the MDE pin, the emulator reads out the user system's pin state and displays the retrieved information on endian setting.



## (c) Selecting the EXTAL Frequency

Specify the frequency of the clock source for supply to the target MCU.

The range of input values is 0.0001 to 99.9999. If any value exceeding this range is input, an error message is displayed.

Note that fractions of the input value below the 5th decimal point are truncated.

Also, if you use only the internal clock oscillation, there is no need to input the EXTAL frequency. In this case, uncheck the [EXTAL frequency] checkbox. Note that if the selected target MCU has no internal clock oscillations installed, this checkbox is not displayed.

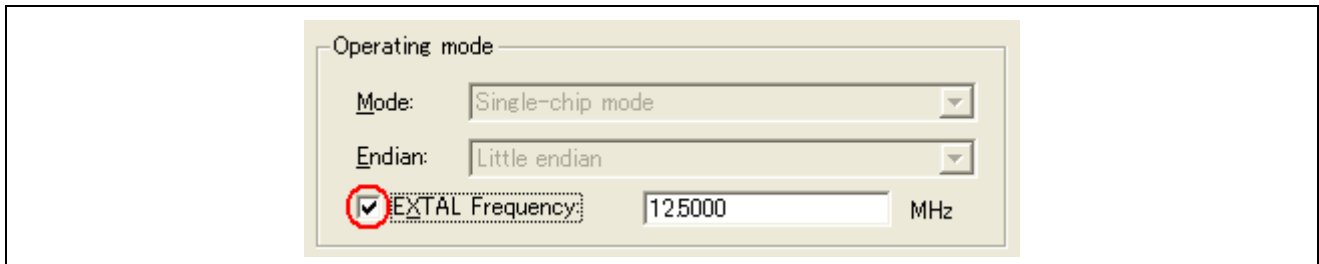


Figure 3.4 Setting the Operating Mode of the MCU

## (d) Selecting the External Memory Areas

Select the endian and bus width for external memory areas.

This setting is possible when the operation mode you selected by register setting is the on-chip ROM enabled extended mode or on-chip ROM disabled extended mode.

Double-clicking on a row for which settings are to be changed opens the [External Area] dialog box shown below. The selectable option differs with each target MCU.

Note that the external memory area differ with each target MCU. For details about the external area of your MCU, refer to the hardware manual supplied with it.

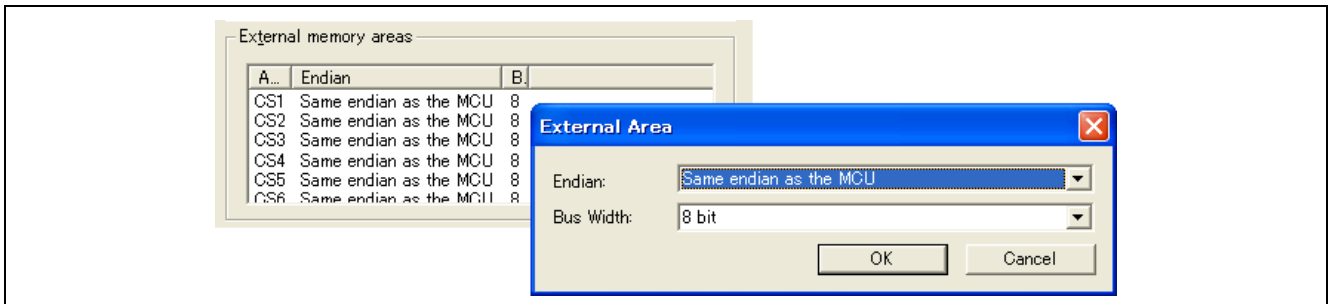


Figure 3.5 Setting the External Area

## (e) Enabling Clock Manipulation

Clock manipulation from the emulator is enabled when you rewrite the internal flash memory.

If, when this checkbox is checked, the clock setting at the time of internal flash memory rewriting is outside the operation-guaranteed range of the target MCU, the emulator changes clock settings to within the operation-guaranteed range as it performs rewriting.

Be aware, however, that if this checkbox is unchecked, the internal flash memory may not be rewritten normally, depending on clock settings. Also, refer to section , .

## (f) Specifying the Address of Working RAM

Specify the first address of the working RAM area for use by the debugger.

The specified amount of bytes in use beginning with the specified location address of the working RAM is used by the emulator-debugger firmware. Although the user program can also use this area (because memory data in this area will be saved in the host machine and then restored), do not specify any address in this area as the origin or destination of a transfer by the DMA or DTC.

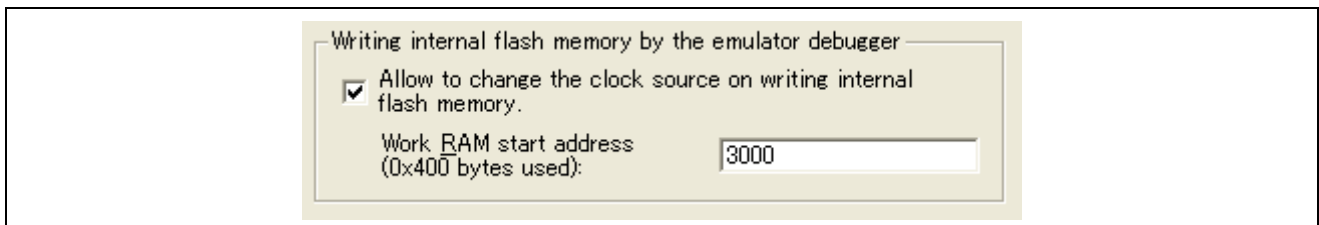
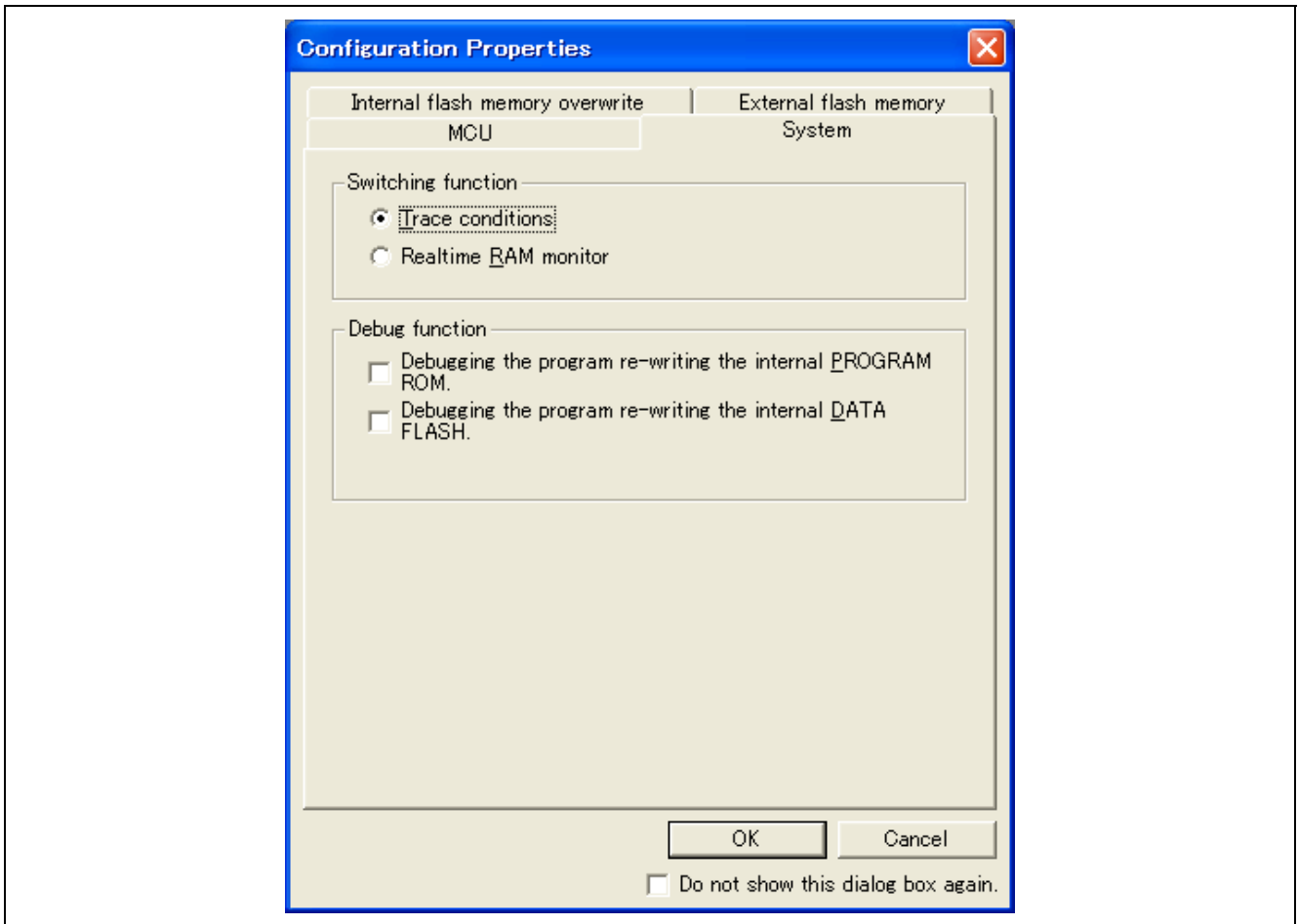


Figure 3.6 [Allow to change the clock source on writing internal flash memory] Checkbox

## (2) Setting Up the System

On the [System] page of the [Configuration Properties] dialog box, specify the configuration of the emulator system as a whole.



**Figure 3.7** [Configuration Properties] Dialog Box ([System] Page)

## (a) Selecting an Exclusive Function

Tracing and realtime RAM monitoring are mutually exclusive. Select either of the two functions.

The [Trace conditions] checkbox is selected by default.

Note that when you're using the E1 emulator, you cannot select the realtime RAM monitor function. (The checkbox is grayed out.)

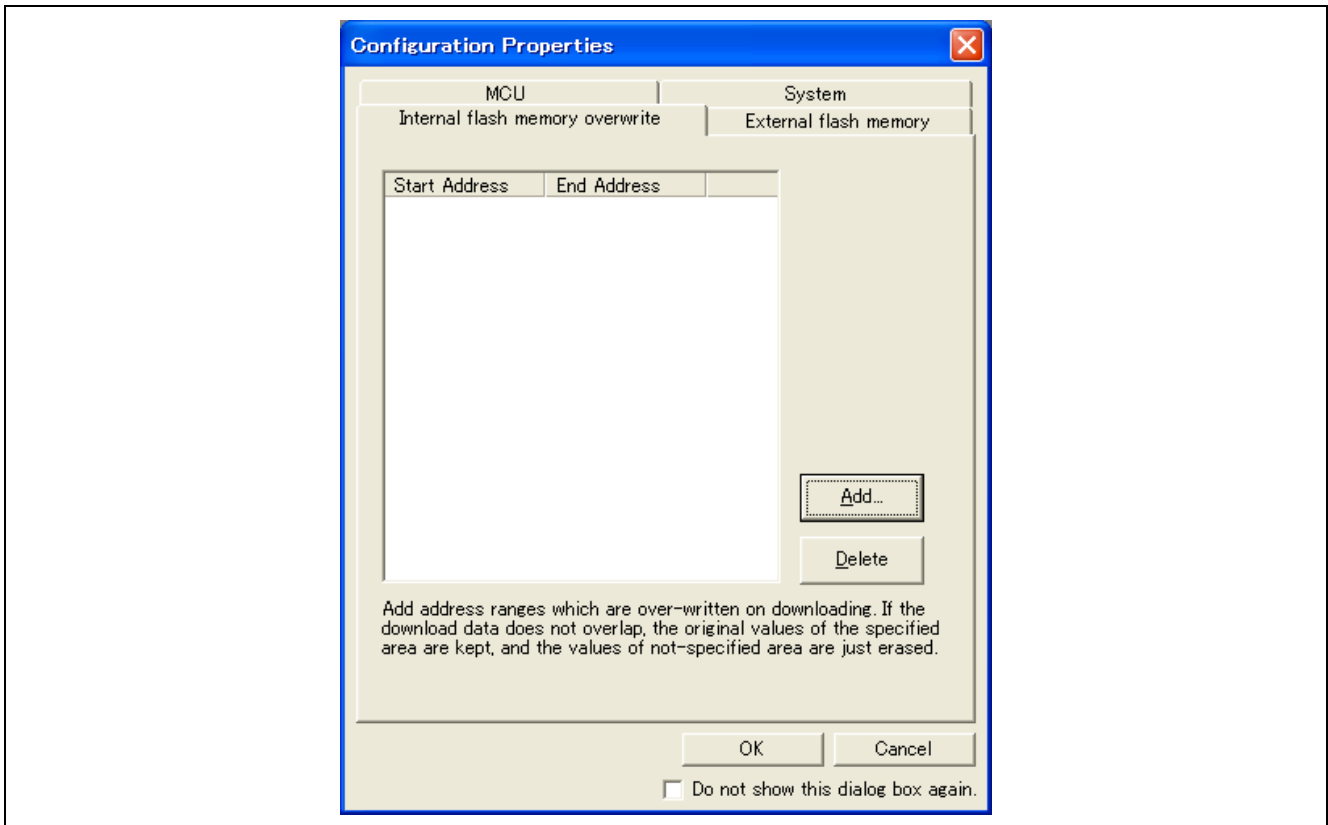
## (b) Using the Debug Function

- Debugging the program re-writing the internal PROGRAM ROM  
Check this checkbox when you debug the program that rewrites the internal program ROM area.
- Debugging the program re-writing the internal DATA FLASH  
Check this checkbox when you debug the program that rewrites the internal data flash area.

## (3) Setting for Overwriting Blocks of the Flash ROM

The [Internal flash memory overwrite] page of the [Configuration Properties] dialog box allows you to specify whether or not flash ROM should be overwritten.

Even after startup, you can open this dialog box and change the setting by selecting [Setup -> Emulator -> System].



**Figure 3.8 [Configuration Properties] Dialog Box ([Internal flash memory overwrite] Page)**

Specify the address range of the internal flash memory for registration.

Up to 4 ranges can be specified.

If the specified area has no data to download, the original values before download are kept.

If the non-specified area has no data to download, it retains the initial values of the flash memory.

## (a) Registering the Overwriting Range in the Internal Flash Memory

Clicking the [Add...] button opens the [Address Range] dialog box. Enter the range and click the [OK] button.

Information of the specified range is displayed in the list.

The address range you have entered is automatically expanded to the block boundary and reflected in the list.

Furthermore, if you've set the operation mode to "User boot mode" on the [Startup and Communication] page of the [Initial Settings] dialog box, you can register the user boot area also.

## (b) Deleting the Overwriting Range in the Internal Flash Memory

Select the information of the overwriting range that you wish to delete from the list and click the [Delete] button.

You also can use the Delete key, instead of the [Delete] button, to delete the information.

## (c) Changing the Range in the Internal Flash Memory

Select the information of the overwriting range on the list and double-click on it. The [Address Range] dialog box will be displayed.

In the [Address Range] dialog box, you can change the settings of the start address and the end address.

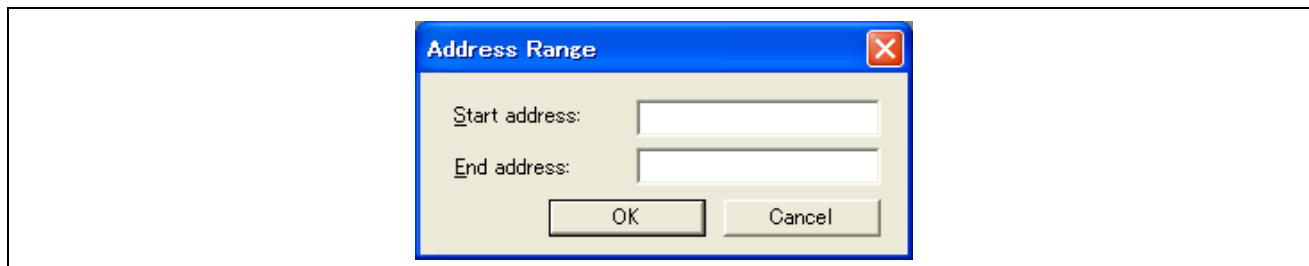
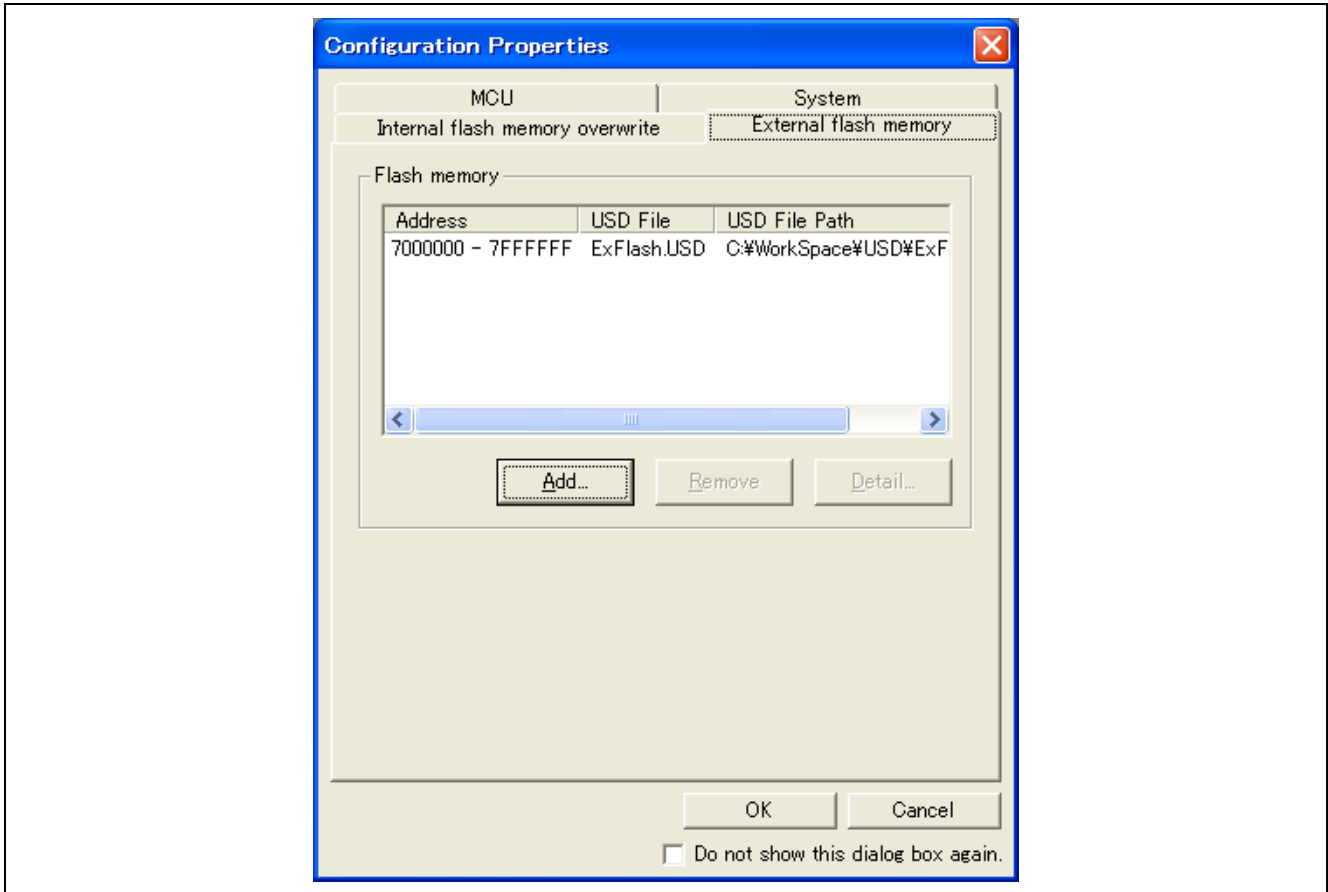


Figure 3.9 [Address Range] Dialog Box

## (4) Registering Information on the External Flash Memory

The [External flash memory] page of the [Configuration Properties] dialog box allows you to register information on external flash memory (that is, flash memory connected to the external-bus address space of the MCU). This registration is necessary when you wish to download a program to external flash memory.



**Figure 3.10** [Configuration Properties] Dialog Box ([External flash memory] Page)

External flash memory is not recognized until its USD file (which defines how the external flash memory is connected to the user system, etc.) is added to the [External flash memory] page of the [Configuration Properties] dialog box.

Up to four USD files can be added.

To create USD files, use the External Flash Definition Editor. For details on how to create USD files, refer to the user's manual for the External Flash Definition Editor, which is available at <http://www.renesas.com/efe>.

## (a) Registering Information on External Flash Memory

The [Add...] button on the [External flash memory] page is enabled when you've selected [On-chip ROM enabled extended mode] or [On-chip ROM disabled extended mode] for the operation mode on the [Startup and Communication] page of the [Initial Settings] dialog box.

Clicking on the [Add...] button opens a dialog box in which you can select USD files. When a USD file is selected, information from the file is displayed in the flash memory list in the [Flash memory] section of the [External flash memory] page.

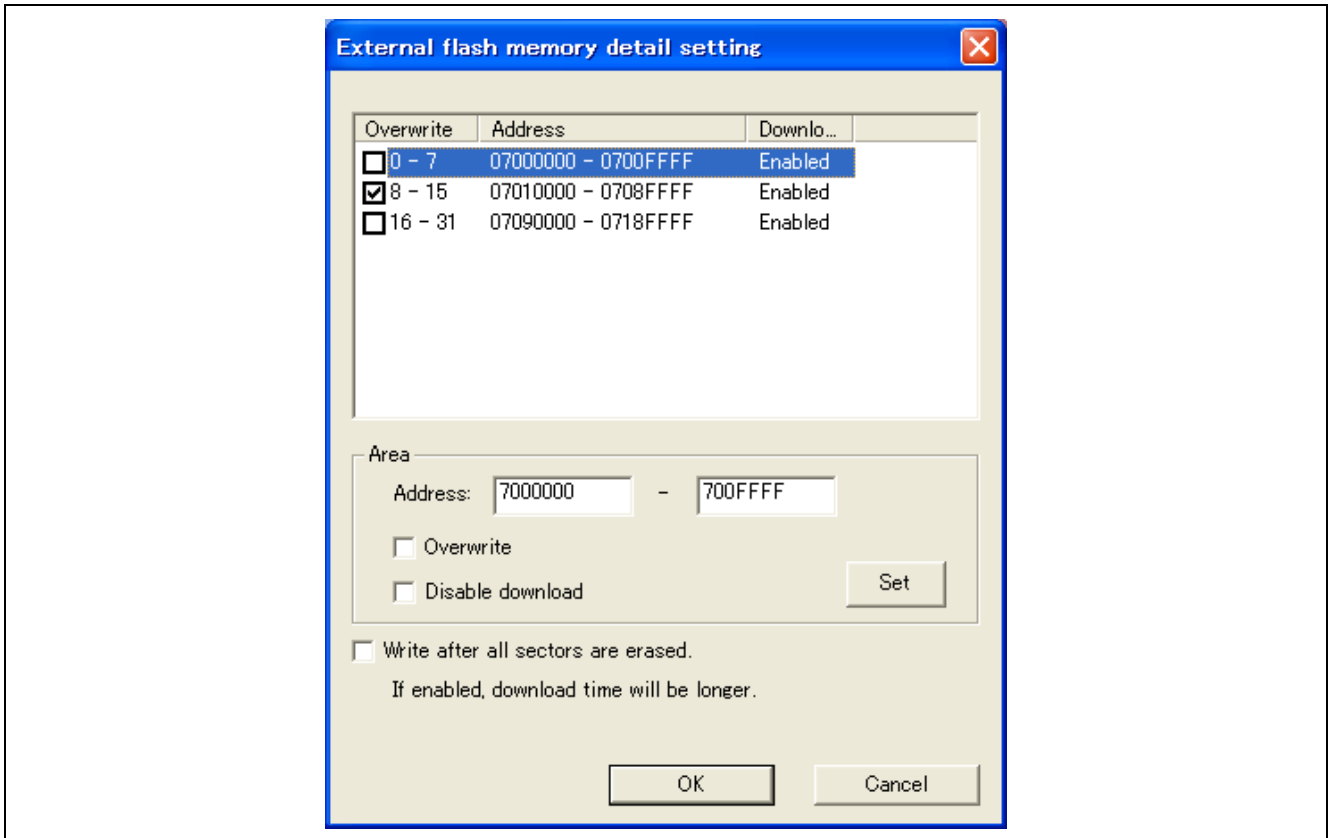
## (b) Deleting Information on the External Flash Memory

From the flash memory list, select the external flash memory information that you wish to delete and click on the [Remove] button.

## (c) Making Detailed Settings for the External Flash Memory

From the flash memory list, select the external flash memory information for which you want to make detail setting, and then click the [Detail...] button. The [External flash memory detail setting] dialog box is displayed. Double-clicking on the selected memory, instead of the [Detail...] button, also brings up this dialog box.

The [External flash memory detail setting] dialog box allows you to specify whether or not individual sectors (blocks) of external flash memory should be overwritten.



**Figure 3.11 [External flash memory overwrite] Dialog Box**

Settings for all sectors defined in the selected external flash memory information are automatically shown in the list. When a checkbox is selected, the sector will be overwritten (merged) when the user program is downloaded.

To alter the setting for one sector, enter the new first and last address in the [Address] edit box and click on [Set]. If [Overwrite] is selected, the selected sector will be overwritten without erasure when the user program is downloaded. The [Download] column of the list is marked as “Enabled.”

Also, if [Disable download] is checked, no data is downloaded to the sector you’ve set. The [Download] column of the list is marked as “Disabled.”

If [Write after all sectors are erased.] is selected, all sectors will be erased when the user program is downloaded.

## Section 4 Emulator Functions

This section describes the emulator functions.  
For the usage of each function, refer to Section 5, Tutorial.

### 4.1 Overview

Table 4.1 gives a functional overview of the emulator.

**Table 4.1 Emulator Functions**

Item	Function
User program execution	<ul style="list-style-type: none"> <li>Executes a program with the operating frequency within a range guaranteed by the MCU.</li> <li>Reset emulation</li> <li>Step functions:               <ul style="list-style-type: none"> <li>Single stepping (one step: one instruction)</li> <li>Source-level step (one step: one-line source)</li> <li>Step over (a break did not occur in a subroutine)</li> <li>Source-level step over (a break did not occur in a subroutine)</li> <li>Step out (when the PC points to a location within a subroutine, execution continues until it returns to the calling function)</li> </ul> </li> <li>[Go to Cursor]</li> <li>[Reset Go]</li> <li>[Free Go]</li> <li>Shows the reason for the most recent break</li> </ul>
Reset	<ul style="list-style-type: none"> <li>Issues a reset from the High-performance Embedded Workshop to the MCU during break.</li> </ul>
Tracing* <sup>2</sup>	<ul style="list-style-type: none"> <li>Internal trace (using the trace buffer in the MCU)</li> <li>External trace output:               <ul style="list-style-type: none"> <li>Trace-output priority mode (not in realtime)</li> <li>CPU-execution priority mode (in realtime but some of the acquired data may be lost)</li> </ul> </li> <li>Type of trace data:               <ul style="list-style-type: none"> <li>Branch (origin and destination) or data access</li> </ul> </li> <li>Searching</li> <li>Filtering</li> <li>Shows trace data as bus information, disassembled code, or source code</li> </ul>
Break* <sup>2</sup>	<ul style="list-style-type: none"> <li>S/W break</li> <li>On-chip break:               <ul style="list-style-type: none"> <li>Pre-PC break</li> <li>Event break (OR, cumulative AND, sequential, or number of passes)</li> <li>Trace full break (break to occur when the trace buffer becomes full)</li> </ul> </li> <li>Forced break</li> </ul>
Performance measurement* <sup>2</sup>	<ul style="list-style-type: none"> <li>Uses a counter in the MCU to measure the number of cycles that passes during point-to-point execution.</li> </ul>



Table 4.1 Emulator Functions (cont)

Item	Function
Memory access* <sup>2</sup>	<ul style="list-style-type: none"> <li>• Downloading to RAM</li> <li>• Downloading to the on-chip flash memory</li> <li>• Downloading to external flash memory*<sup>1</sup></li> <li>• Single-line assembly</li> <li>• Disassembly</li> <li>• Reading of memory</li> <li>• Writing to memory</li> <li>• Automatic updating of a display of selected variables during user program execution</li> <li>• Fill</li> <li>• Search</li> <li>• Move</li> <li>• Saving and loading data in memory</li> <li>• Verifying data in memory</li> <li>• Showing variables</li> <li>• Realtime RAM monitoring</li> <li>• Automatic updating of data in memory</li> </ul>
General/control register access	Reads or writes the general-purpose/control registers.
Internal I/O register access	Reads or writes the internal I/O registers.* <sup>1</sup>
Source-level debugging	Various source-level debugging functions
Command line	Supports command input. Batch processing is enabled when a file is created by arranging commands in input order.
Help	Describes the usage of each function or command syntax input from the command line window.
Hot plug-in	Supports the method for connecting the E1/E20 to the program under execution, allowing for debugging to start from the middle of program execution.

Notes: 1. The [IO] window displays the contents of registers defined in [xxxx.io]. You can edit the file to add or delete registers to the set for display in the [IO] window.

For the contents to be included in [xxxx.io], refer to reference 6, I/O File Format, in the High-performance Embedded Workshop User's Manual.

The following directory contains [xxxx.io] (xxxx means the name of the MCU group).

<High-performance Embedded Workshop folder>  
 \Tools\Renesas\DebugComp\Platform\E1E20\E20RX\IOFiles

2. Depending on the target MCU, some of the debug functions cannot be used or may differ in functional specifications.

## 4.2 Downloading a Program

Download the load module to be debugged.

To download a program, choose [Download] from the [Debug] menu and select a desired load module or right-click on a load module under [Download modules] of the [Workspace] window and then choose [Download] from the popup menu. Alternatively, double-click on the name of the load module.

Note: Before a program can be downloaded, you must have it registered as a load module in the High-performance Embedded Workshop.

## 4.3 Opening a Source File

### 4.3.1 Viewing the Source Code

Select your source file and click the [Open] button to make the High-performance Embedded Workshop open the file in the integrated editor. It is also possible to display your source files by double-clicking on them in the [Workspace] window.

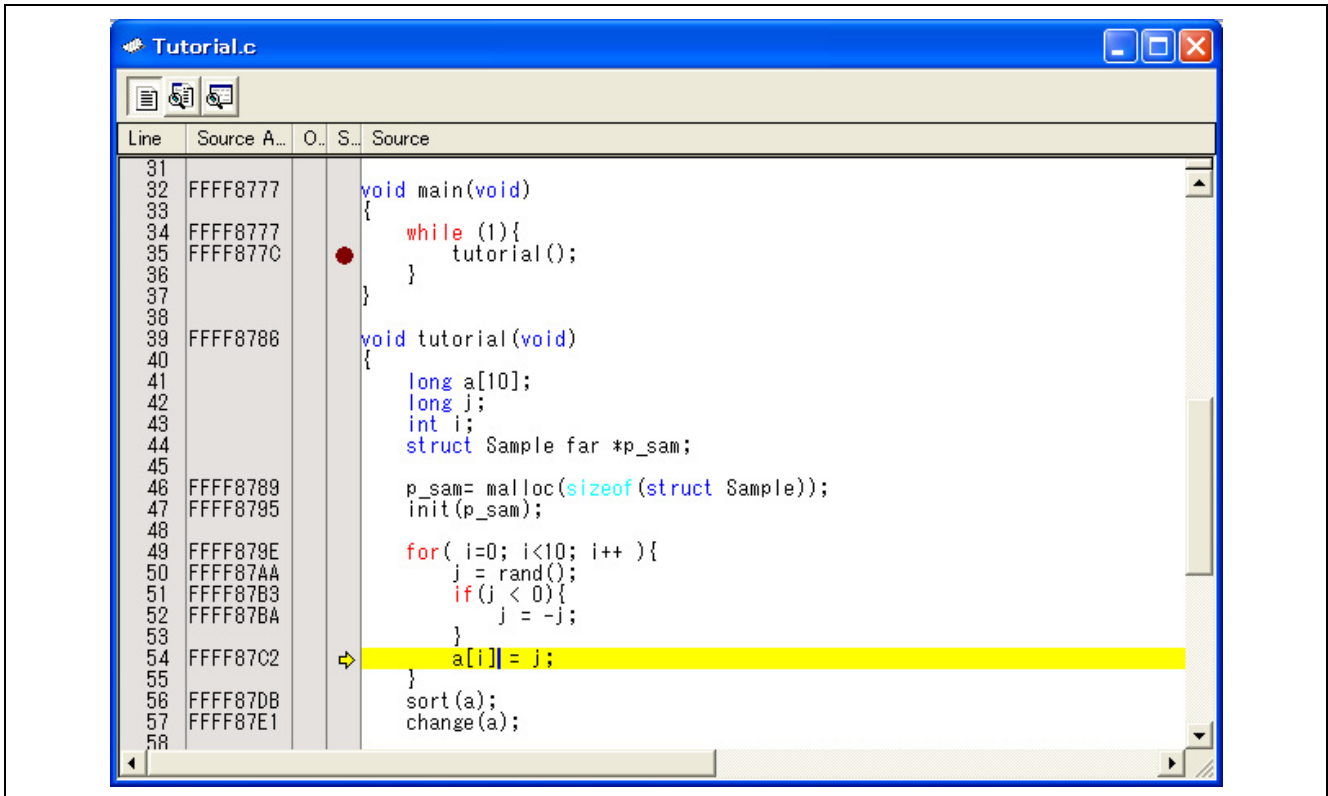


Figure 4.1 [Editor] Window

The columns listed below are to the left of the [Source] column.

The first column	([Source Address] column):	Address information
The second column	([On-Chip Breakpoint] column):	On-chip breakpoint information
The third column	([S/W Breakpoint] column):	PC, bookmark, and breakpoint information

#### [Source Address] column

When a program is downloaded, an address for the current source file is displayed on the [Source Address] column. These addresses are helpful when setting the PC value or breakpoints.




#### [On-Chip Breakpoint] column

The [On-Chip Breakpoint] column displays the following item:

- : Indicates the address condition for an on-chip break.  
The number of address conditions that can be set for on-chip breakpoints is the same as the overall number of events, which varies with the MCU.  
Double-clicking on the [On-Chip Breakpoint] column produces a bitmap symbol as shown above in the corresponding line.

#### [S/W Breakpoint] column

The [S/W Breakpoint] column displays the following items:

- : Bookmark
- : PC breakpoint
- : Program counter (PC)

- ☰ To switch off a column in all source files
  - (1) Right-click on the [Source] window or select the [Edit] menu.
  - (2) Click the [Define Column Format...] menu item.
  - (3) The [Global Editor Column States] dialog box is displayed.
  - (4) A checkbox indicates whether the column is enabled or not. If it is checked, the column is enabled. If the checkbox is gray, the column is enabled in some files and disabled in others. Deselect the checkbox of a column you want to switch off.
  - (5) Click on the [OK] button for the new column settings to take effect.

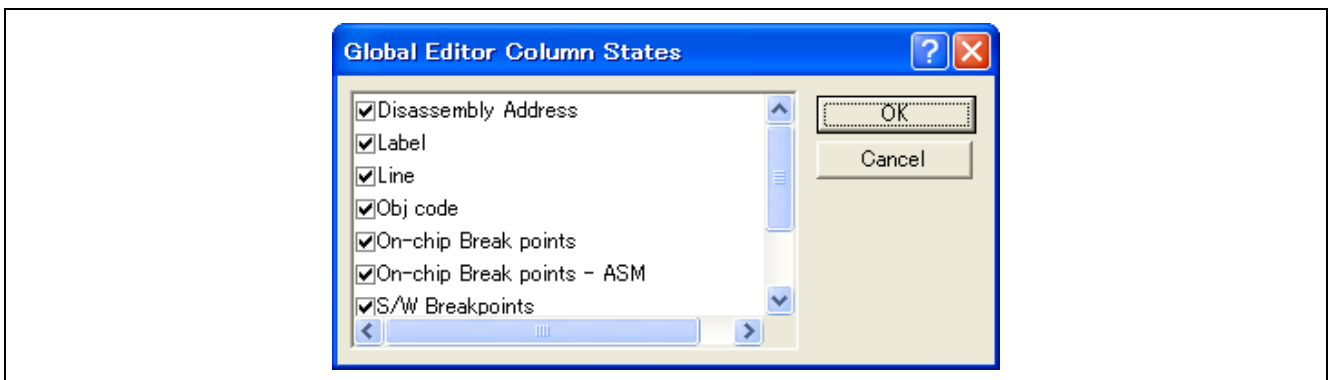



Figure 4.2 [Global Editor Column States] Dialog Box

☞ To switch off a column in one source file


- (1) Open the source file which contains the column you want to remove and right-click on the [Editor] window to open a popup menu.
- (2) Click on the [Columns] menu item to display a cascaded menu item. The columns are displayed in this popup menu. If a column is enabled, it has a tick mark next to its name. Clicking on the entry will toggle whether the column is displayed or not.

### 4.3.2 Viewing the Assembly-Language Code

Click the [Disassembly] toolbar button at the top of the window when a source file is opened to show the assembly-language code that corresponds to the current source file.

If you do not have a source file, but want to view code in the assembly-language level, either choose [View -> Disassembly...] or click on the [Disassembly] toolbar button .

The [Disassembly] window opens at the current PC location and shows [Address] and [Code] (optional) which show the disassembled mnemonics (with labels when available).

Selecting the [Mixed display] toolbar button  displays both the source and the code. The following shows an example in this case.

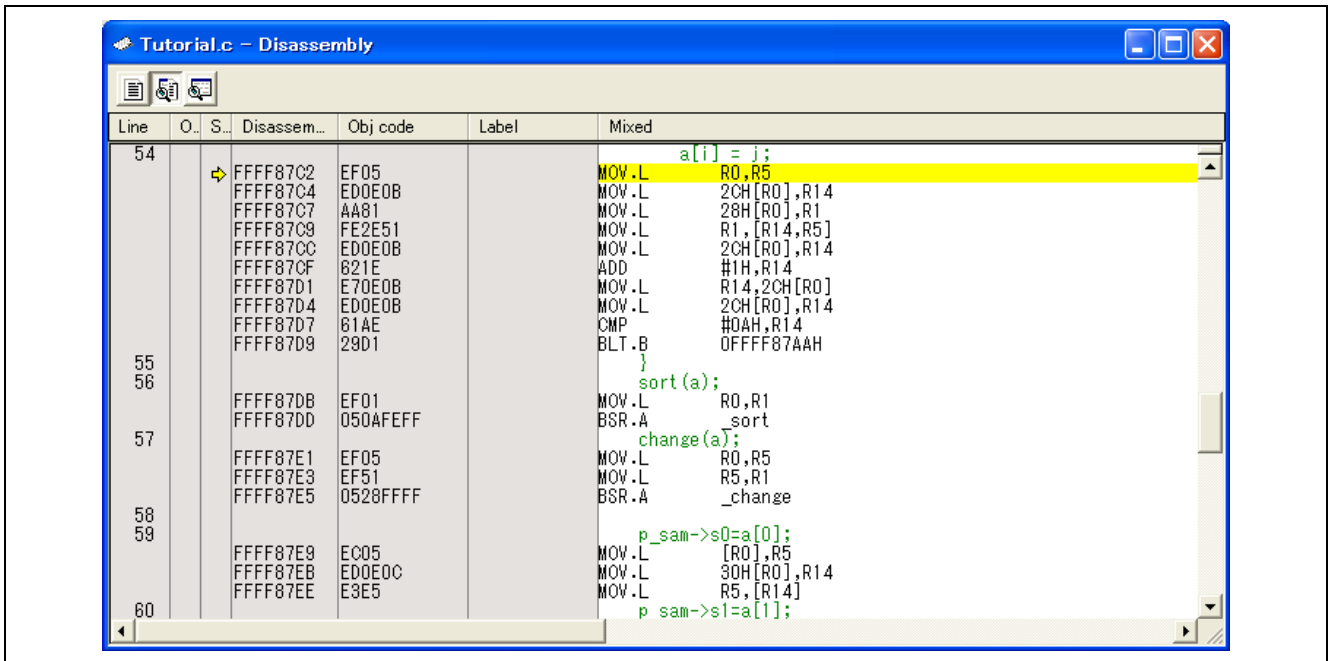
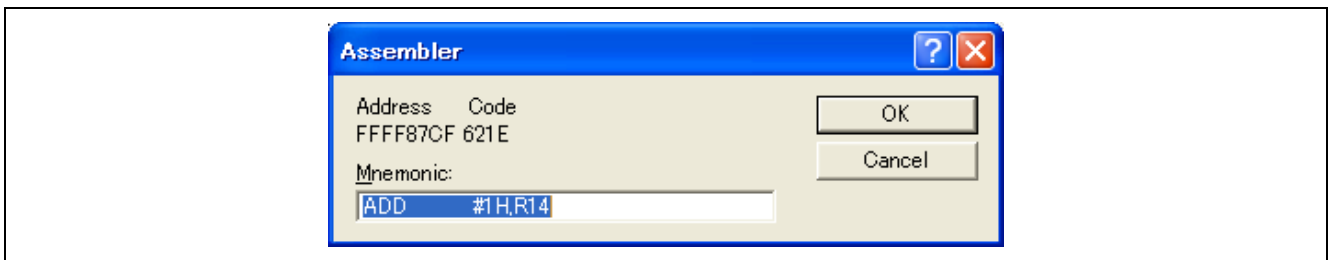


Figure 4.3 [Disassembly] Window

### 4.3.3 Modifying the Assembly-Language Code

You can modify the assembly-language code by double-clicking on the instruction that you want to change. The [Assembler] dialog box will be opened.



**Figure 4.4** [Assembler] Dialog Box

The address, machine code, and disassembled instruction are displayed. Enter the new instruction or edit the current instruction in the [Mnemonic] field. Pressing the [Enter] key will assemble the instruction into memory and move on to the next instruction. Clicking on the [OK] button will assemble the instruction into memory and close the dialog box. Clicking on the [Cancel] button or pressing the [Esc] key will close the dialog box.

**Note:** The assembly-language display is disassembled from the machine code on the actual memory. If the memory contents are changed, the dialog box (and the [Disassembly] window) will show the new assembly-language code, but the display content of the [Editor] window will not be changed. This is the same even if the source file contains assembly code.

## 4.4 Memory Access Functions

The emulator has the following memory access functions.

### 4.4.1 Memory Read/Write Function

- [Memory] window:

The memory contents are displayed in the window.

Only the amount specified when the [Memory] window is opened can be read.

Since there is no cache in the emulator, read cycles are always generated.

If the memory is written in the [Memory] window, read cycles in the range displayed in the [Memory] window will occur for updating the window.

When the [Memory] window is not to be updated, select [Lock Refresh] from the popup menu opened by right-clicking in the window.

- me command:

A command line function that reads or writes the specified amount of memory at the specified address.

#### (1) User Program Downloading Function

A load module registered in the workspace can be downloaded. Such module can be selected from [Download Module] in the [Debug] menu. Downloading is also possible by a popup menu that is opened by right-clicking on the mouse at the load module in the workspace. The user program is downloaded to the RAM or flash memory. This function also downloads information required for source-level debugging such as debugging information.

#### (2) Memory Data Uploading Function

The specified amount of memory from the specified address can be saved in a file. Select [Save] from the popup menu in the [Memory] window.

#### (3) Memory Data Downloading Function

The memory contents saved in a file can be downloaded. Select [Load] from the popup menu in the [Memory] window.

#### (4) Displaying Variables

The contents of a selected variable in the user program are displayed.

#### (5) Realtime RAM Monitoring

With some MCUs, contents of memory can be monitored while the user program is running. For details on the specifications of individual MCUs, refer to the online help.

#### (6) Automatic Updating of Data in Memory

The display of data in memory in the [Memory] window can be automatically updated while the user program is running. The interval for updating the display is also specifiable.

#### (7) Other Memory Operation Functions

Other functions are as follows:

- Memory fill
- Memory move
- Memory copy
- Memory save
- Memory verify
- Memory search
- Internal I/O display
- Displaying label and variable names and their contents

For details, refer to the online help.

## 4.5 Break Functions

The E1/E20 emulator has three kinds of break functions: a forced break, a S/W break, and an on-chip break function.

### 4.5.1 Forced Break

The forced break function is used to forcibly cause a break in execution of the user program.

### 4.5.2 S/W (Software) Break

The first instruction at a specified address is replaced with a special instruction (intended for debugging) that leads to a break in execution.

Setting a S/W breakpoint leads to replacement of the first instruction following the specified address by the special instruction and so involves writing to memory. Similarly, removing a S/W breakpoint involves writing to memory because the special instruction must now be replaced with the original instruction.

To set a S/W breakpoint, double-click on the [S/W Breakpoint] column in the [Editor] or [Disassembly] window of the High-performance Embedded Workshop.

### 4.5.3 On-Chip Break

The on-chip break function consists of three kinds of break: Pre-PC break, Event break (based on events, such as instruction-fetch event or operand access event), and Trace full break.

For the RX600 Series MCUs, on-chip breaks based on instruction-fetch events can be set for 8 points, and those based on operand access events for 4 points, allowing you to set event-combination breaks [OR and AND (cumulative)] for up to 8 points.

For the RX200 Series MCUs, on-chip breaks based on instruction-fetch events can be set for 4 points, and those based on operand access events for 2 points, allowing you to set event-combination breaks [OR and AND (cumulative)] for up to 6 points.

For the usage of on-chip break function, refer to section 4.7.1, Setting On-Chip Breakpoints.

#### (1) Pre-PC Break

Once a specified instruction-fetch event is encountered, a break occurs before the instruction at that address is executed. Pre-PC breakpoints can even be set while the user program is running.

To set a pre-PC breakpoint, double-click on the [On-Chip Breakpoint] column in the [Editor] or [Disassembly] window of the High-performance Embedded Workshop. For details on the [On-Chip Breakpoint] column, refer to section 4.2, Downloading a Program.

It is also possible to set pre-PC breakpoints via the [On-Chip Break] dialog box.

#### (2) Event Break

Several events can be used in combination. Instruction-fetch and data access (using operand-access events) are selectable for event conditions. The event combination can be [OR], [AND (cumulative)], [Sequential], or [Pass Count]. Event breakpoints can also be set while the user program is running.

It is also possible to set event breakpoints via the [On-Chip Break] dialog box.

Note that the RX200 Series MCUs do not support on-chip breaks based on number of passes.

#### (3) Trace Full Break

A break occurs when the trace buffer becomes full. Related settings cannot be changed while the user program is running.

Table 4.2 shows the list of break functions.

**Table 4.2 Break Functions**

Type of Break	Number of Points		Break Condition	Programming of Flash Memory	Can a Breakpoint be Set While the Program is Running?
	RX 600 Series MCUs	RX 200 Series MCUs			
S/W break	256	←	Address	Yes	No
Pre-PC break	8	4	Address	No	Yes
Event break (instruction fetch)	8	4	Address	No	Yes
Event break (data access)	4	2	Data access	No	Yes
Trace full break		—	Trace buffer full	No	No

## 4.6 Event Functions

### 4.6.1 Using Events

An event refers to a combination of phenomena that occur during program execution.

An event you have set can be used as a condition for the break, trace or performance-analysis function.

The number of events set and the setting conditions differ with each MCU. For functional specifications, refer to Table 3.3, “List of Functional Specifications by Target MCU.”

Events you create can be registered for reuse at a later time.

#### (a) Types of Events

Events are of the following types.

**Table 4.3 Event Types**

Execution address	The emulator detects execution of the instruction at the specified address by the CPU.  When an execution-address event is used for a pre-PC break, the event condition is satisfied immediately before execution of the instruction at the specified address. When an event is specified in any other way, the event condition is satisfied after the instruction at the specified address has been executed.
Data access	The emulator detects access to a specified address or specified address range.

#### (b) Event Combinations

The following types of combination can be specified for two or more events.

**Table 4.4 Types of Event Combination**

OR	The condition is met when any one of the specified events occurs.
AND (cumulative)	The condition is met when all of the specified events occur regardless of the timing.
Sequential	The condition is met when the specified events occur in a specified order.



### 4.6.2 Adding Events

Events are added in any of the following ways.

- Through the [On-Chip Break], [Trace conditions], or [Performance] dialog box
- By setting on-chip breakpoints in the [On-Chip Breakpoint] column of the [Editor] window (for on-chip break only)
- By dragging and dropping from another window
- From the command line

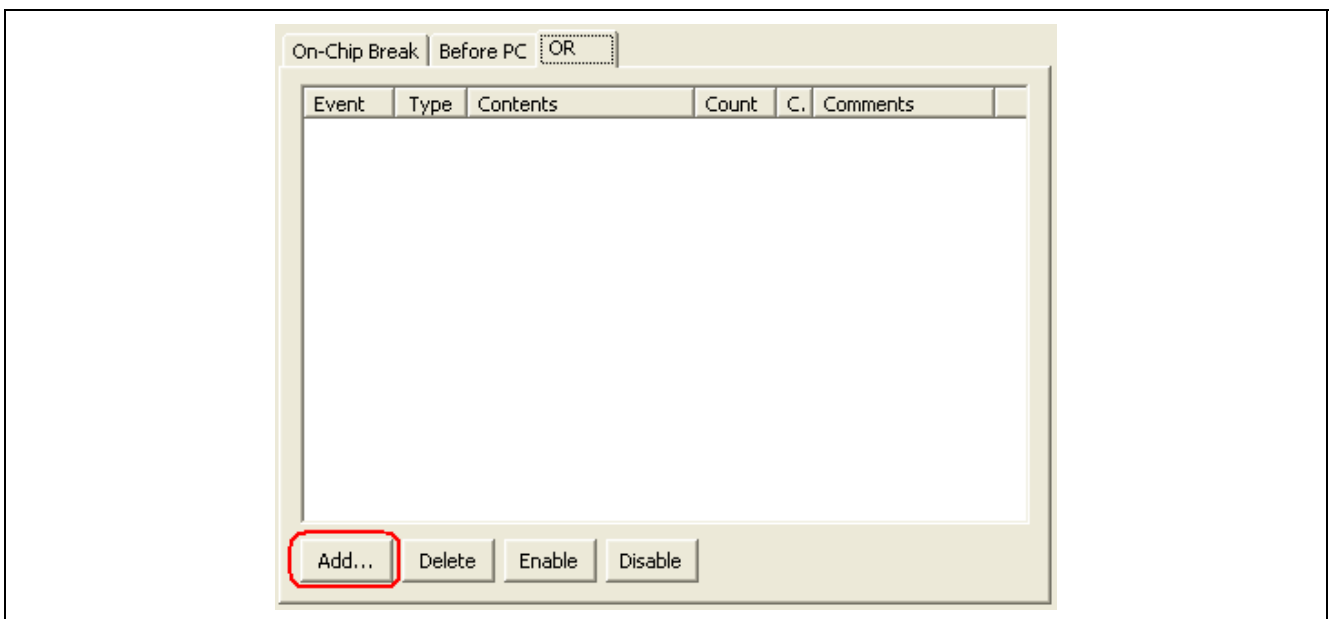
(a) Add an Event through the [On-Chip Break], [Trace conditions], or [Performance] Dialog Box

The following is an example of adding an event through the [On-Chip Break] dialog box.

1. Select [View -> Event -> On-chip Break] to open the [On-Chip Break] dialog box. Selecting an event combination on the [On-Chip Break] page opens the corresponding page where detailed settings can be made. We use an [OR] condition in this example, so select the [Event Break] checkbox and then the [OR] page in the [On-Chip Break] dialog box.

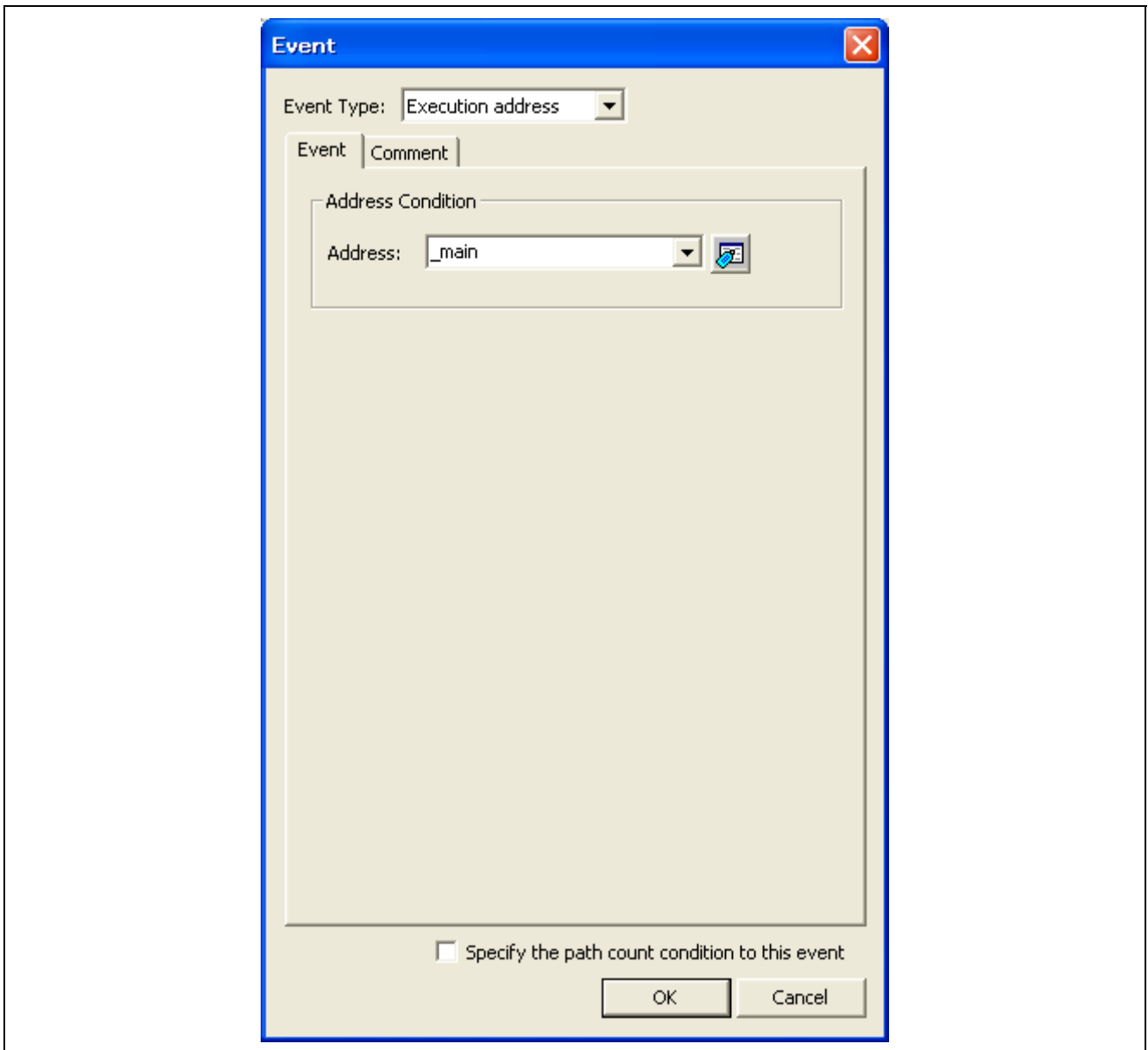
For details on the event types on the other tabbed page ([Before PC], [AND], and [Sequential]), refer to section 4.7.1, Setting On-Chip Breakpoints.

Click on the [Add] button or double-click on the line where the new event is to be added.



**Figure 4.5** [On-Chip Break] Dialog Box

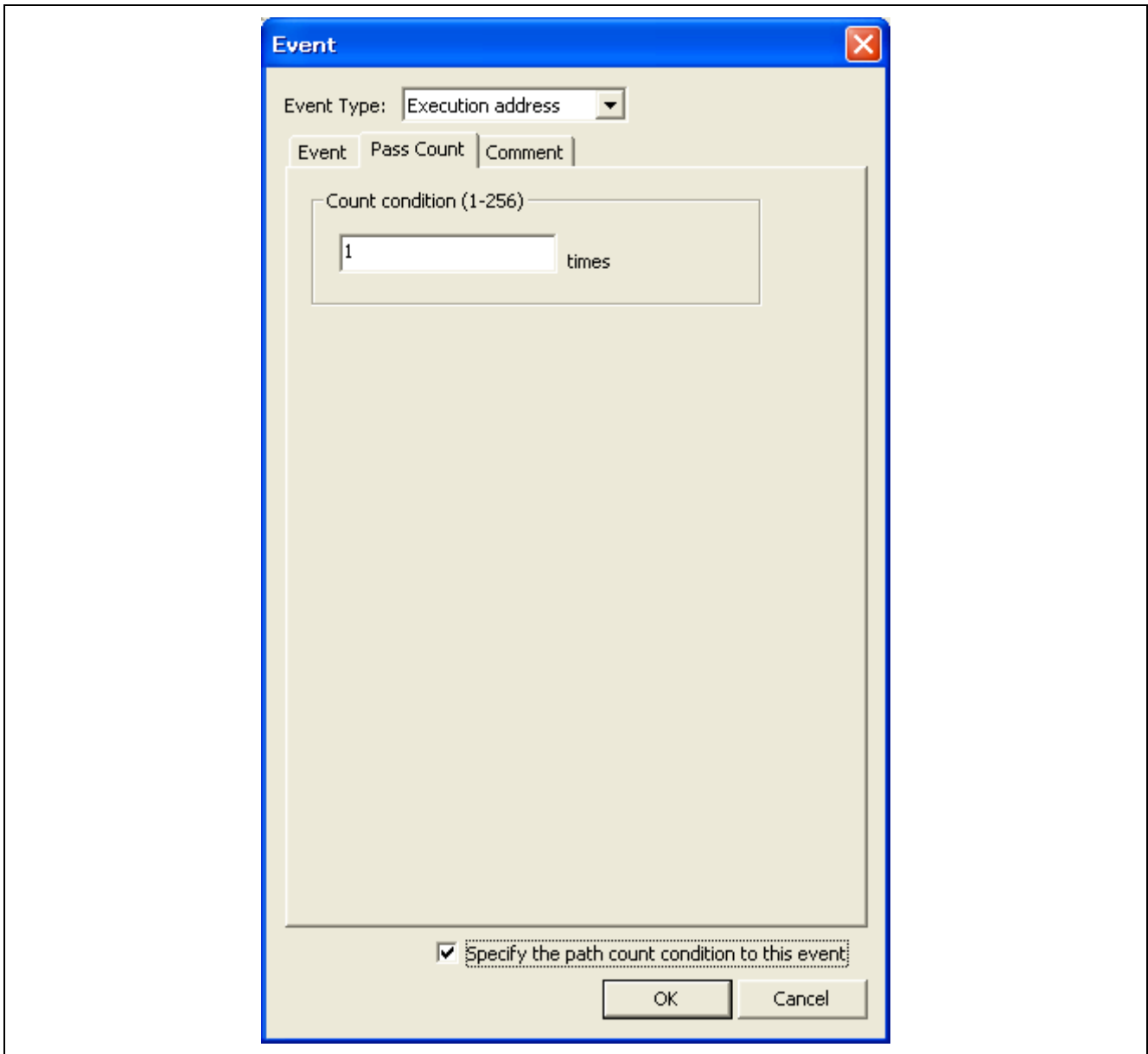
- The [Event] dialog box shown below will be displayed. Set details of the event condition in this dialog box. Then click on the [OK] button.



**Figure 4.6** Adding an Event

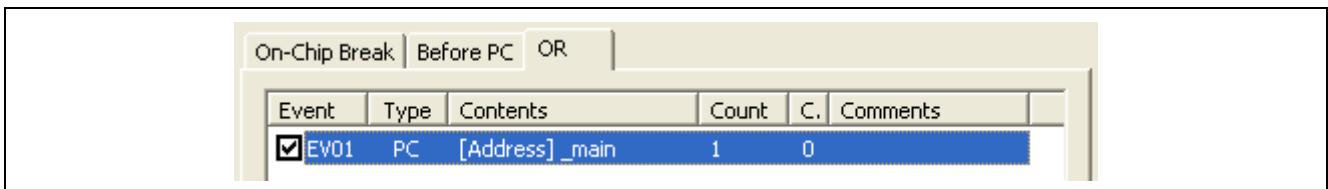
If the target MCU is one of the RX600 Series, you can set the condition for the number of event passes (1 to 256). Select the [Specify the pass count condition to this event] checkbox to include the number of event passes as part of the condition.

The [Pass Count] page will be added. Specify a number of times and click on [OK].



**Figure 4.7** Specifying a Number of Times

3. An event will be added at the specified position.



**Figure 4.8** Added Event

4. If you create an event that would make the total number of events exceed the limit, an error message is displayed. In this case, the event you have attempted to add is invalid.

(b) Adding an Event from the [On-chip Break points] Column of the [Editor] Window

[Adding an on-chip breakpoints]

1. Double-click on the [On-chip Break points] column of the [Editor] window.

This sets execution of the instruction at the corresponding address as the condition for an on-chip breakpoint, i.e. an execution-address condition.

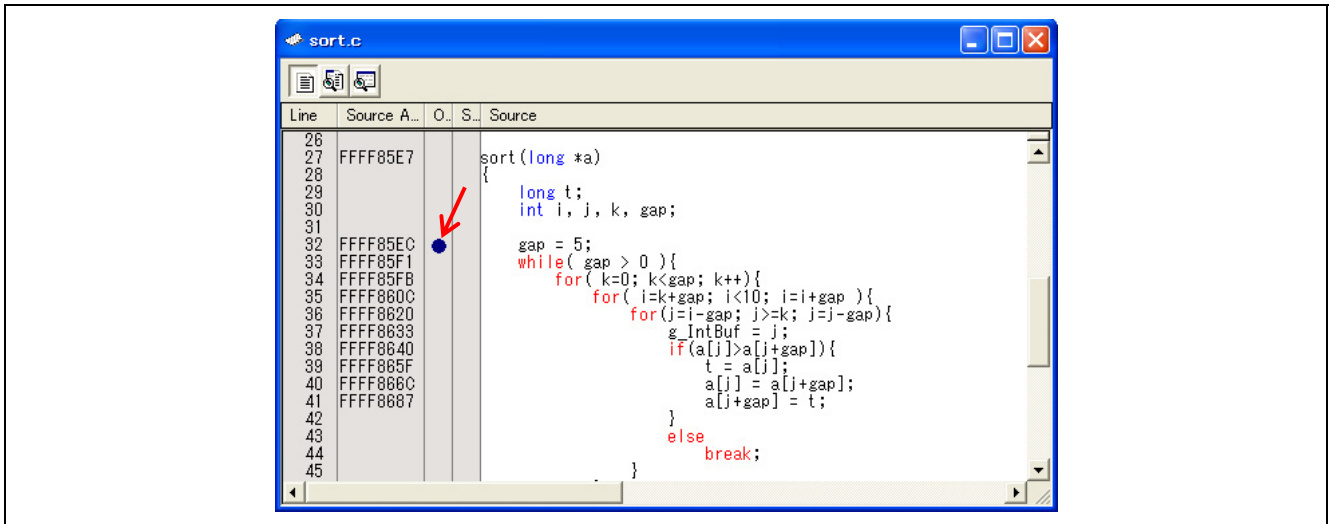


Figure 4.9 [Editor] Window

Note: If you have edited the contents of the [On-Chip Break] dialog box but not clicked on the [Apply] button (i.e. the settings have not been activated yet and [\*] is displayed after the title in the title bar), you cannot set an on-chip breakpoint from the [On-chip Break points] column of the [Editor] window.

## (c) Adding events by dragging and dropping

[Dragging and dropping a variable or function name in the [Editor] window]

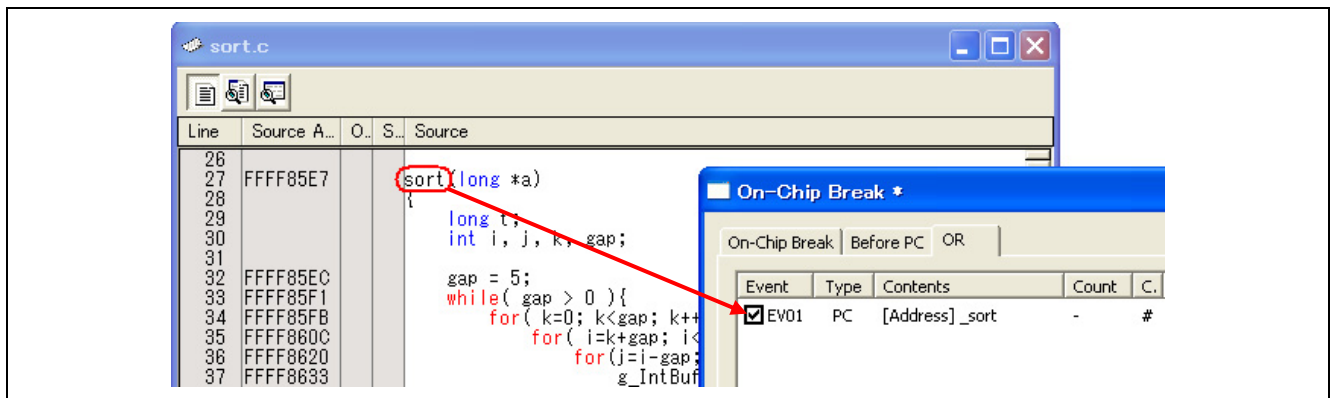
1. By dragging and dropping a variable name into the [Event] column of the [On-Chip Break] dialog box, you can set access to that variable as an event to be detected, i.e. a data-access condition.

At this time, the size of the variable is automatically set as a condition of the data access event.

Only global or static variables taking up 1, 2, or 4 bytes can be registered for event detection. Static variables in functions cannot be registered.

When a pre-PC breakpoint has been set (on the [Before PC] tabbed page), however, dragging and dropping of a variable name is not possible because the only event type selectable for a pre-PC break is “execution address”.

2. By dragging and dropping a function name into the [Event] column of the [On-Chip Break] dialog box, you can set execution of the instruction at the address where that function starts as an event to be detected, i.e. an execution-address condition.



**Figure 4.10 Dragging and Dropping into the [On-Chip Break] Dialog Box**

[Dragging and dropping an address range in the [Memory] window]

With the RX600 Series MCUs, select an address range in the [Memory] window and drag and drop it into the [Event] column of the [On-Chip Break] dialog box. In this way, you can set access to an address in the selected address range as a data access event to be detected, i.e. a data access condition.

When a pre-PC breakpoint has been set (on the [Before PC] tabbed page), however, dragging and dropping of an address range is not possible because the only event type selectable for a pre-PC break is “execution address”.

[Dragging and dropping a label in the [Label] window]

Select a label in the [Label] window and drag and drop it into the [Event] column of the [On-Chip Break] dialog box. In this way, you can set execution of the instruction at the label as an event to be detected, i.e. an execution-address condition.

### 4.6.3 Removing Events

The following way of removing events is available.

- Deleting an event from the [On-Chip Break], [Trace conditions], or [Performance] dialog box

(a) Deleting an Event Through the [On-Chip Break], [Trace conditions], or [Performance] Dialog Box

The following is an example of removing an event through the [On-Chip Break] dialog box.

1. To remove one point, select the line you want to remove in the [Event] list on the [OR] page of the [On-Chip Break] dialog box and then click on the [Delete] button (or use the keys Ctrl + Del instead of clicking on the button).

The selected event will be removed from the [Event] list.

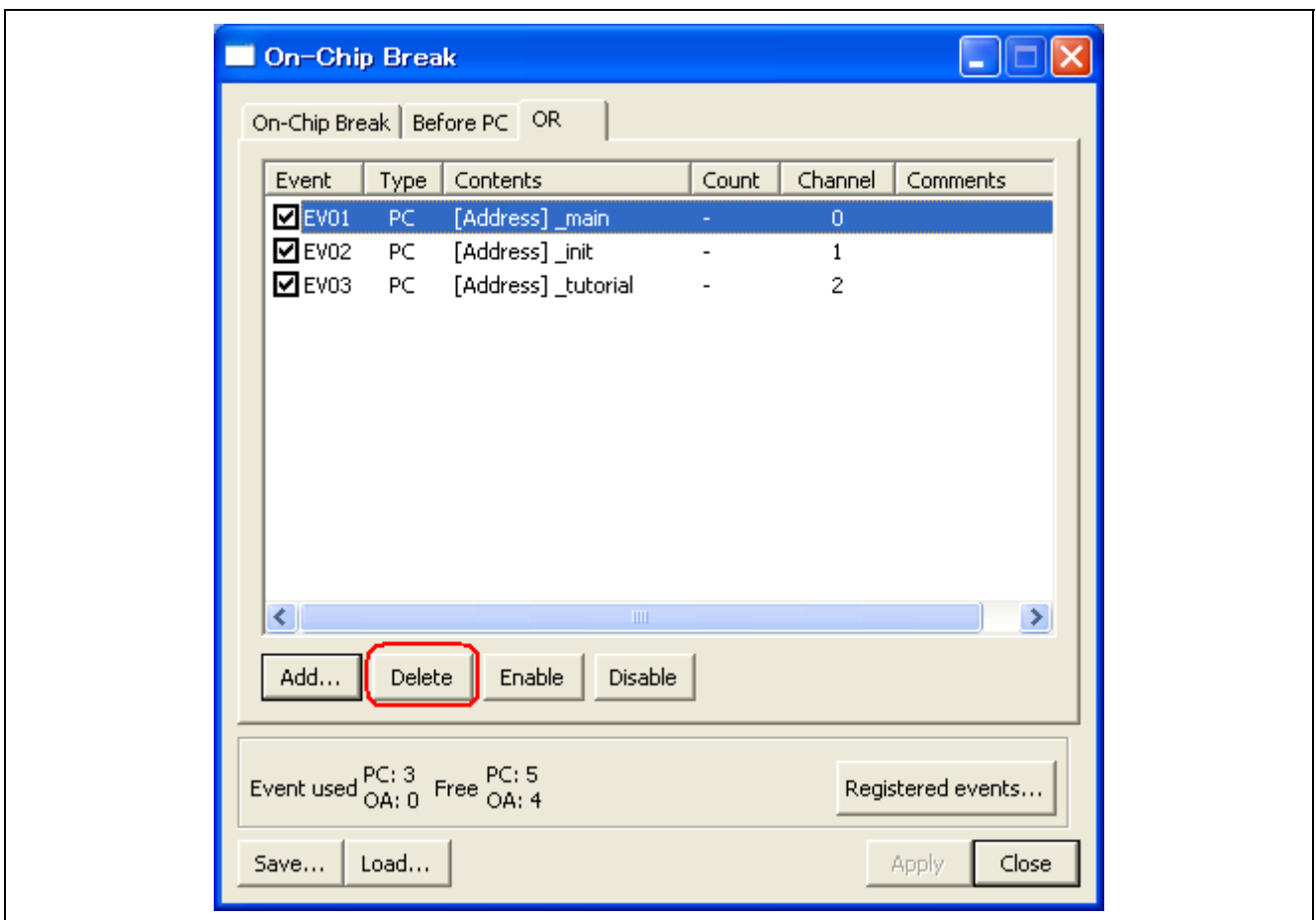


Figure 4.11 Removing an Event

2. To remove multiple events, hold down the Shift or the Ctrl key while you select lines you want to remove in the [Event] list in the [On-Chip Break] dialog box and then click on the [Delete] button (or use the keys Ctrl + Del instead of clicking on the button).

The selected events will be removed from the [Event] list.

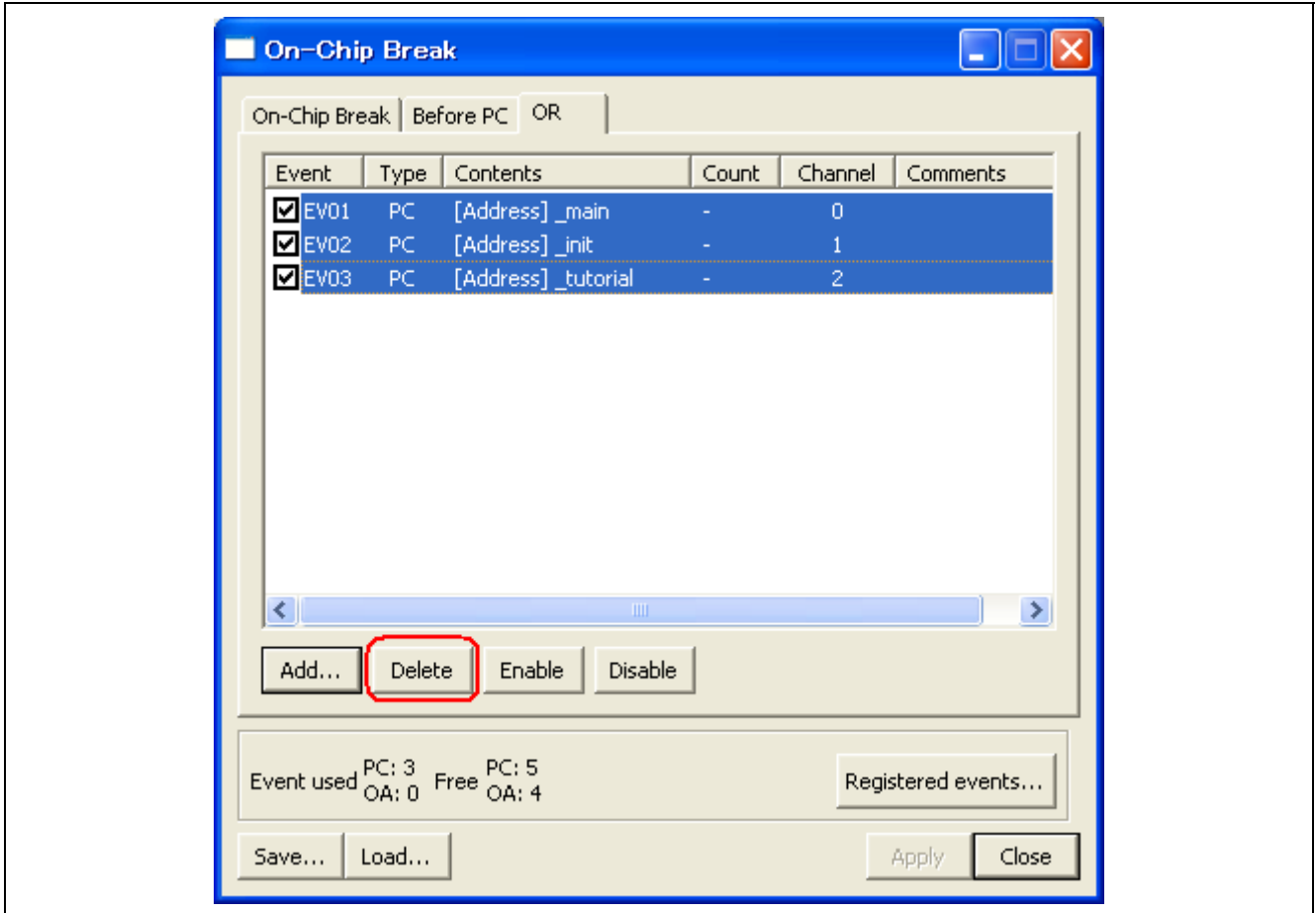


Figure 4.12 Removing Multiple Events

#### 4.6.4 Registering Events

“Registering an event” refers to placing an event in the list of registered events.

A registered event can be reused at a later time.

Select one of the following ways to register an event. Up to 256 events can be registered.

- Through the [On-Chip Break], [Trace conditions], or [Performance] dialog box
- By dragging and dropping from the [On-Chip Break], [Trace conditions], or [Performance] dialog box
- Through the [Registered Events] dialog box

##### (a) Registering Events Through the [On-Chip Break], [Trace conditions], or [Performance] Dialog Box

The following is an example of registering an event through the [On-Chip Break] dialog box.

1. Click on the [Add] button on the [OR] of the [On-Chip Break] dialog box. The [Event] dialog box will appear.

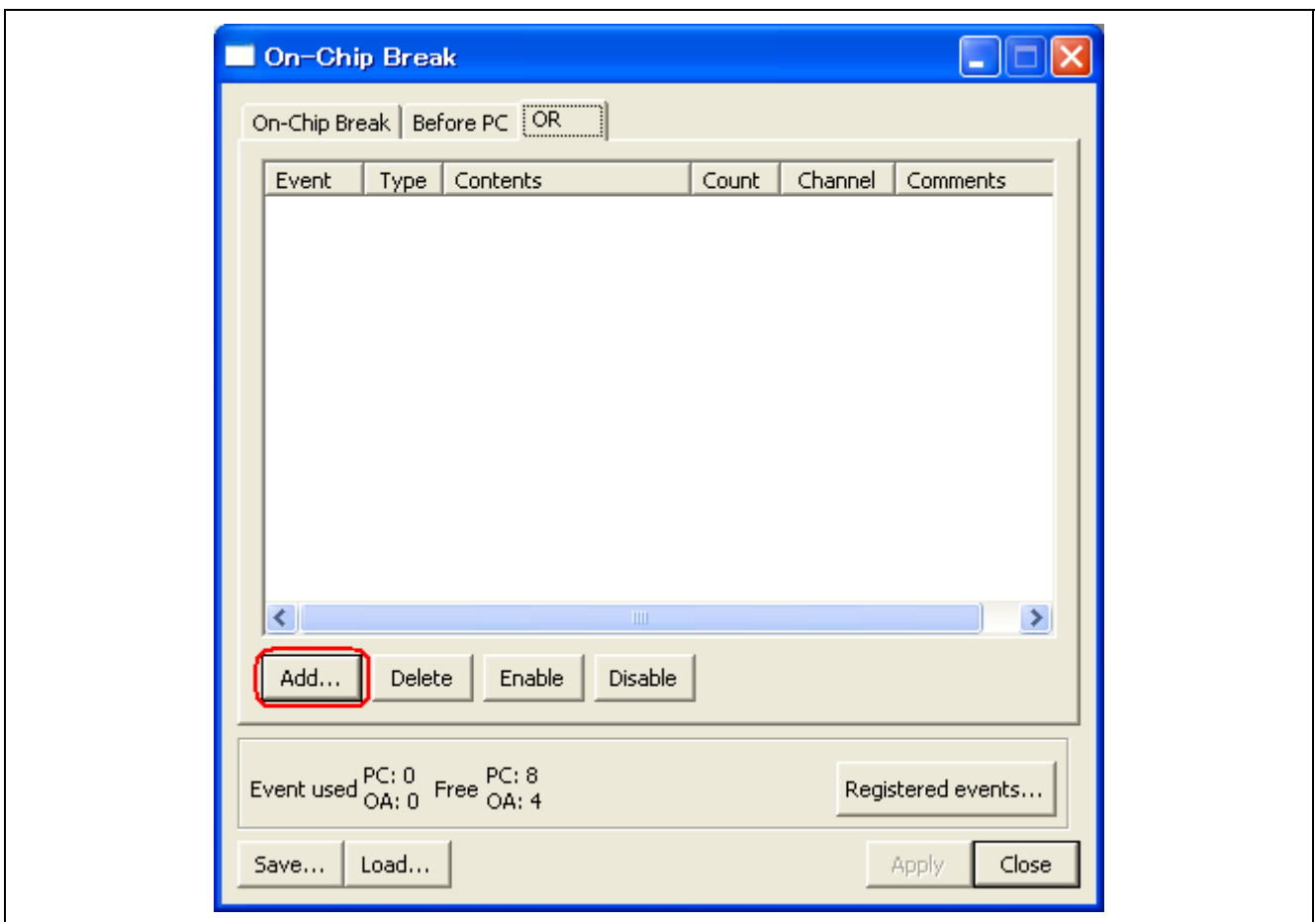


Figure 4.13 [On-Chip Break] Dialog Box



Open the [Comment] page and select the [Add this event to the list] checkbox.  
An explanatory comment for the event can be attached.  
You can check the [Registered Events] dialog box to see the contents of registered events and comments.  
Click on the [OK] button.

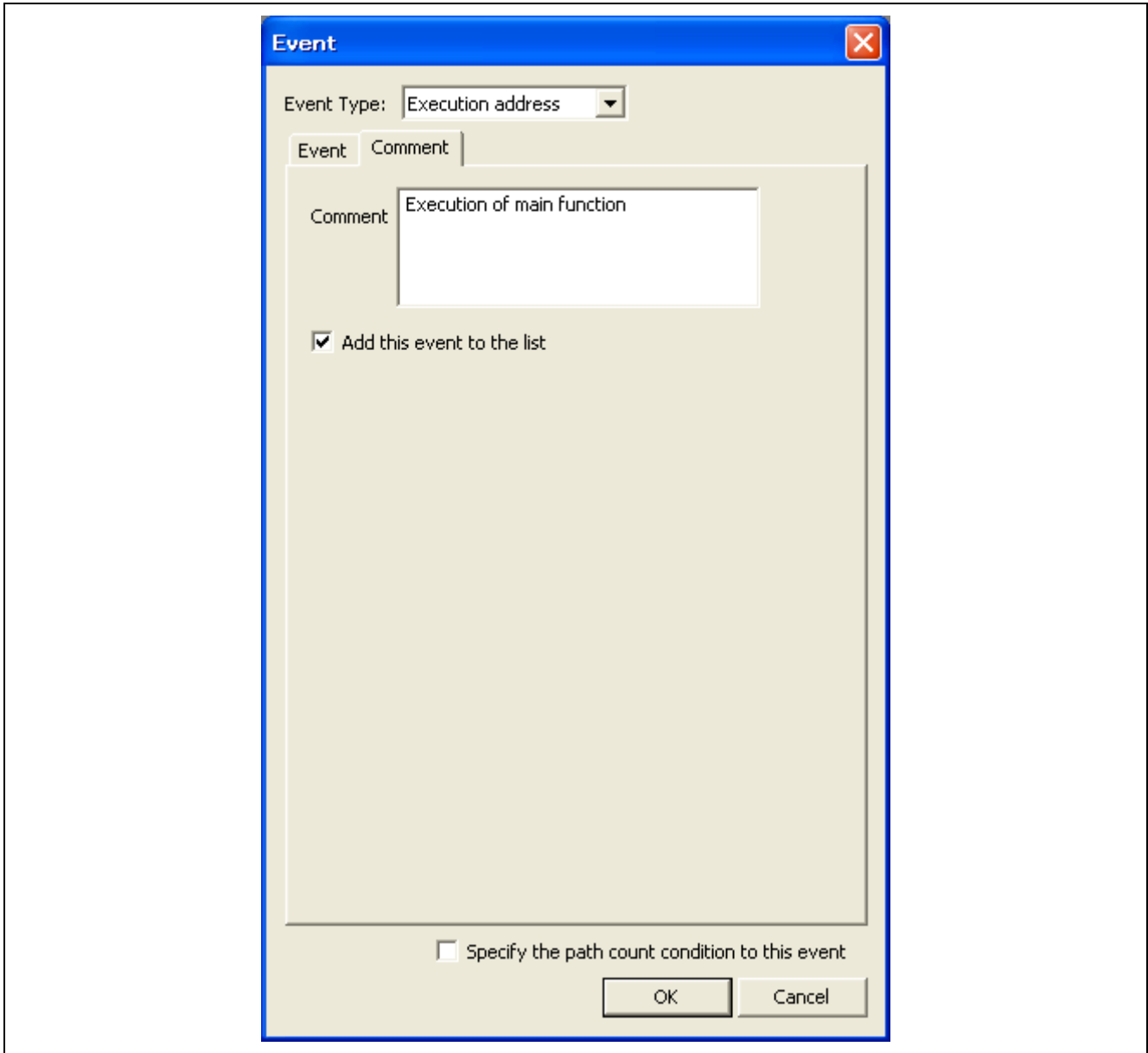


Figure 4.14 Registering an Event

- The event is added at the specified position and registered in the [Registered Events] dialog box at the same time.

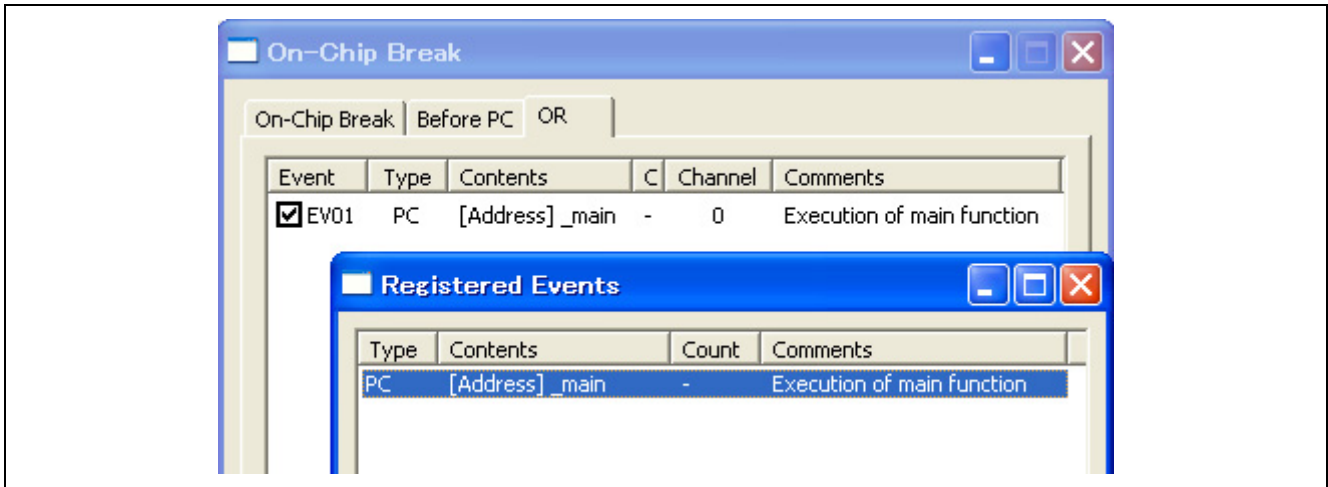


Figure 4.15 Registered Events

To open the [Registered Events] dialog box, click on the [Registered events...] button at the bottom of the [On-Chip Break] dialog box.

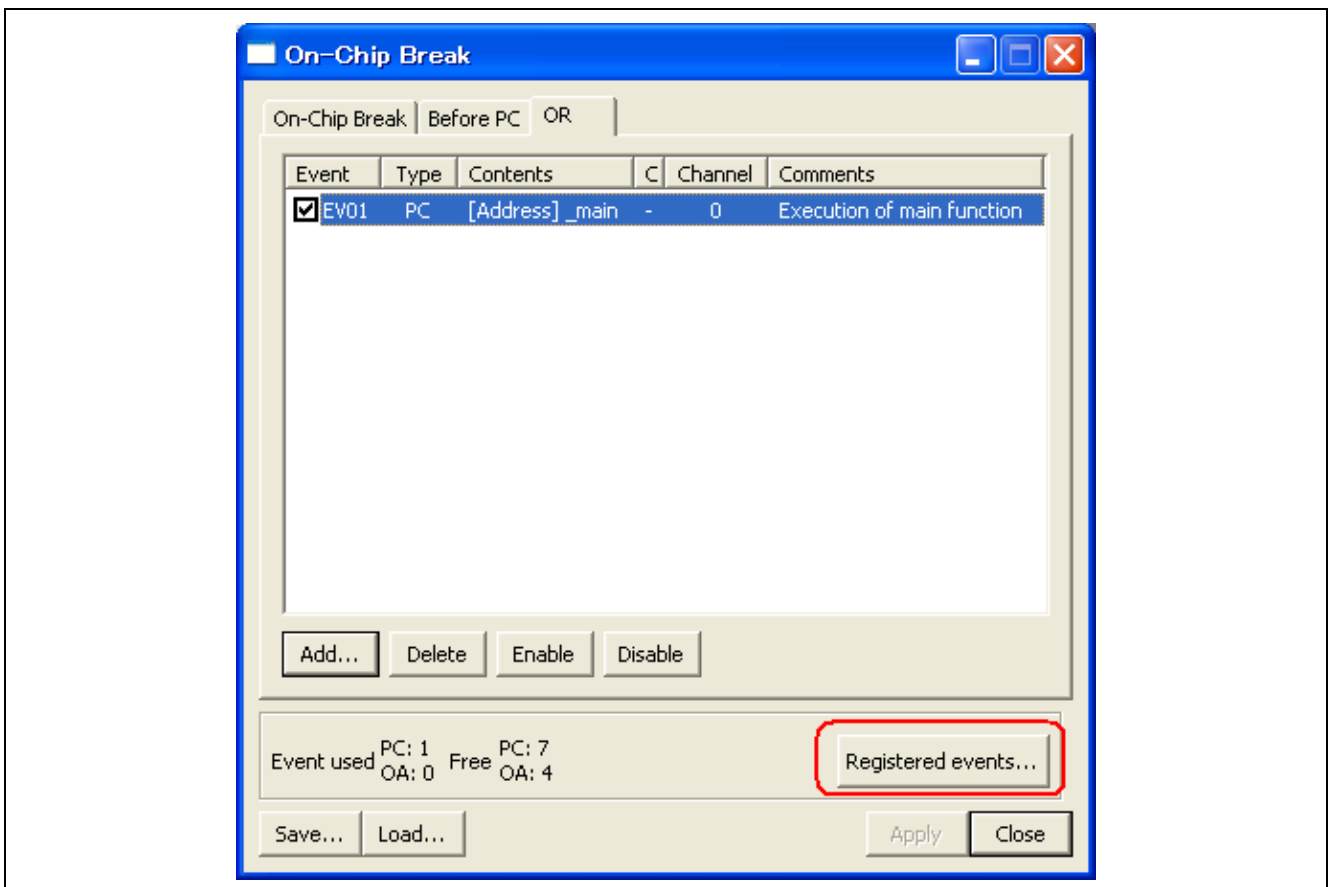


Figure 4.16 Opening the [Registered Events] Dialog Box

(b) Registering Events by Dragging and Dropping

An event you have created in the [On-Chip Break] dialog box can be registered in the [Registered Events] dialog box by dragging and dropping it into the list.

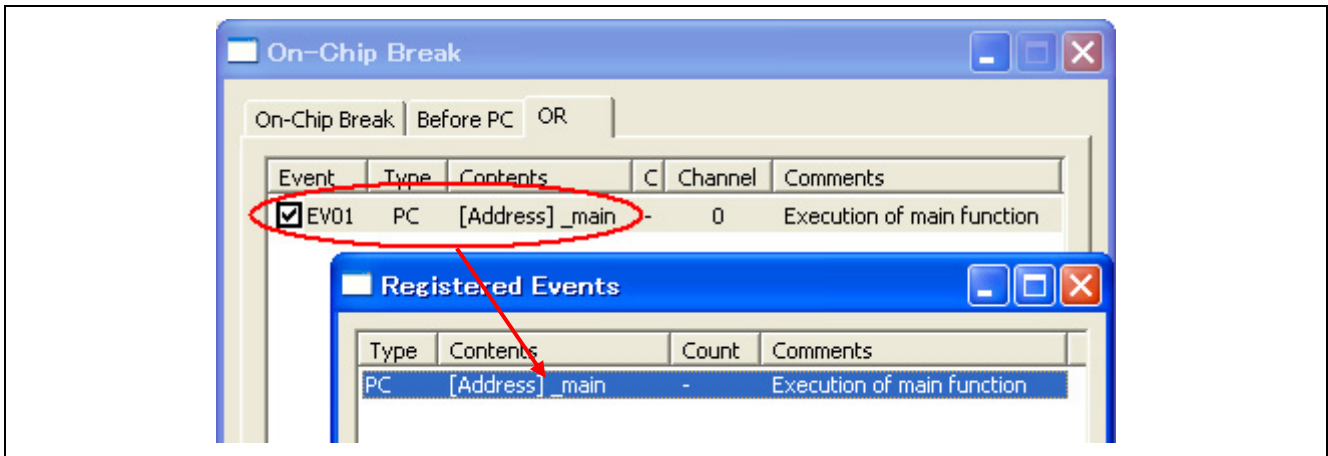


Figure 4.17 Dragging and Dropping into the [Registered Events] Dialog Box

(c) Registering an Event in the [Registered Events] Dialog Box

Click on the [Add] button to open the [Event] dialog box, in which you can create events. Any event you create here is added to the [Registered Events] dialog box.

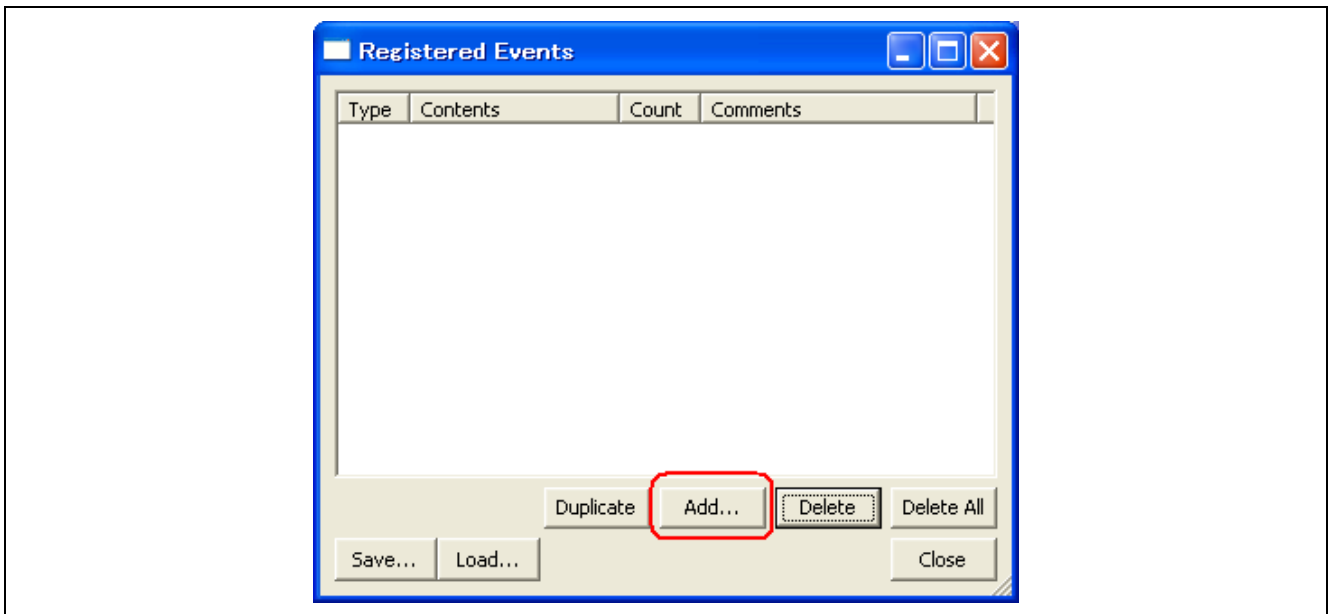


Figure 4.18 [Registered Events] Dialog Box

#### 4.6.5 Removing a Registered Event

To remove an event that has been registered, click on the [Registered events...] button at the bottom of the [On-Chip Break] dialog box. The [Registered Events] dialog box opens.

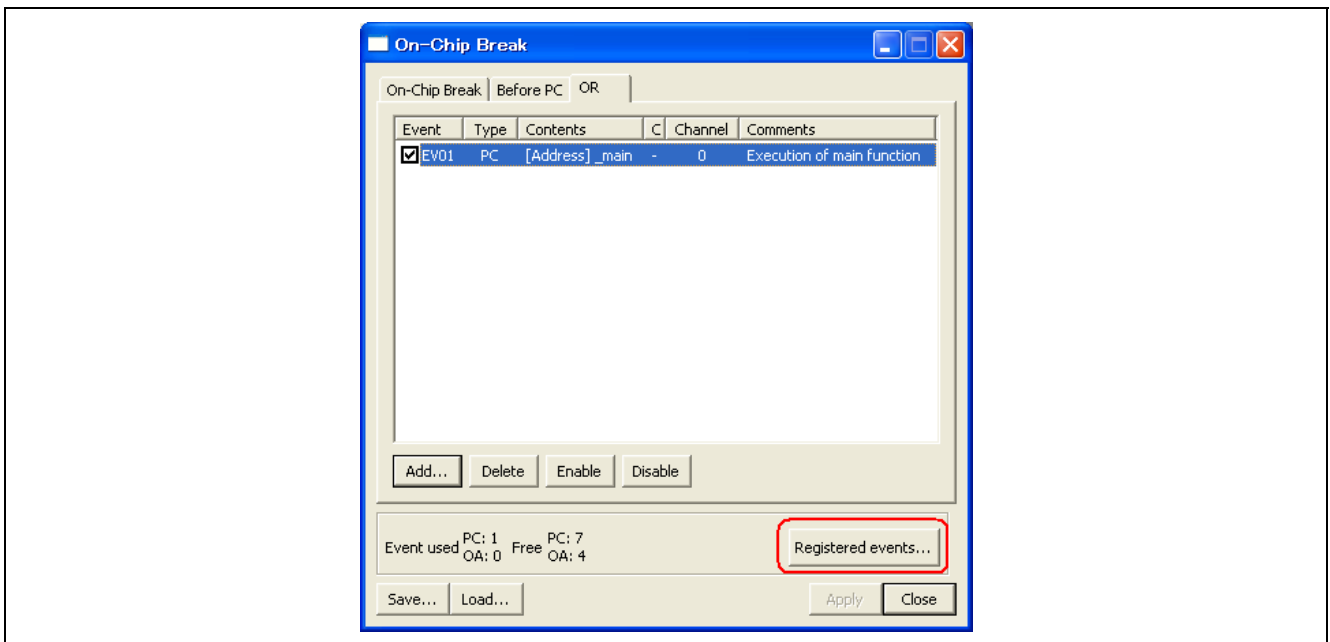


Figure 4.19 Opening the [Registered Events] Dialog Box

To remove one event, select the line you want to remove in the [Registered Events] dialog box and click on the [Delete] button (or use the keys Ctrl + Del instead of clicking on the button).

The selected event will be removed from the [Registered Events] dialog box.

To remove all events, click on the [Delete All] button.

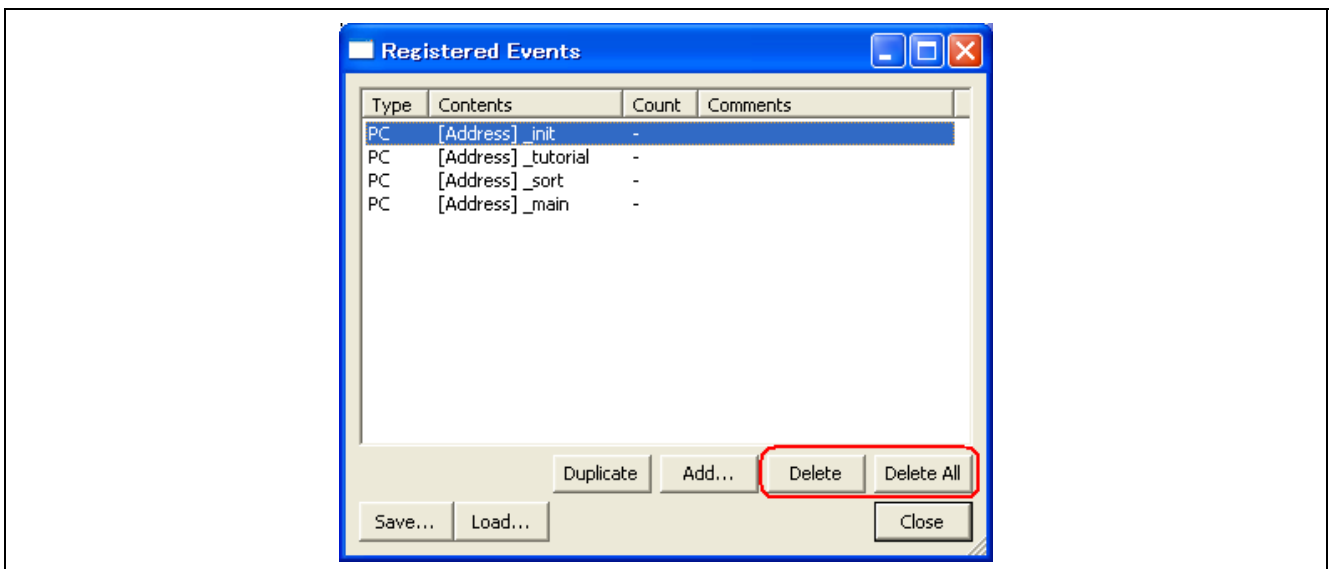


Figure 4.20 Removing an Event

#### 4.6.6 Creating Events for Each Instance of Usage or Reusing Events

The following two approaches are available for setting events in the [On-Chip Break], [Trace conditions], or [Performance] dialog box.

One is to create events in the dialog box each time they are to be used. The other is to choose a condition from the [Registered Events] dialog box and drag and drop it into the [Event] list in the [On-Chip Break], [Trace conditions], or [Performance] dialog box.

Here, we refer to the former as creating events per usage and the latter as reusing events.

- **Creating events per usage**

Select this method if you intend to use a specific condition only once.

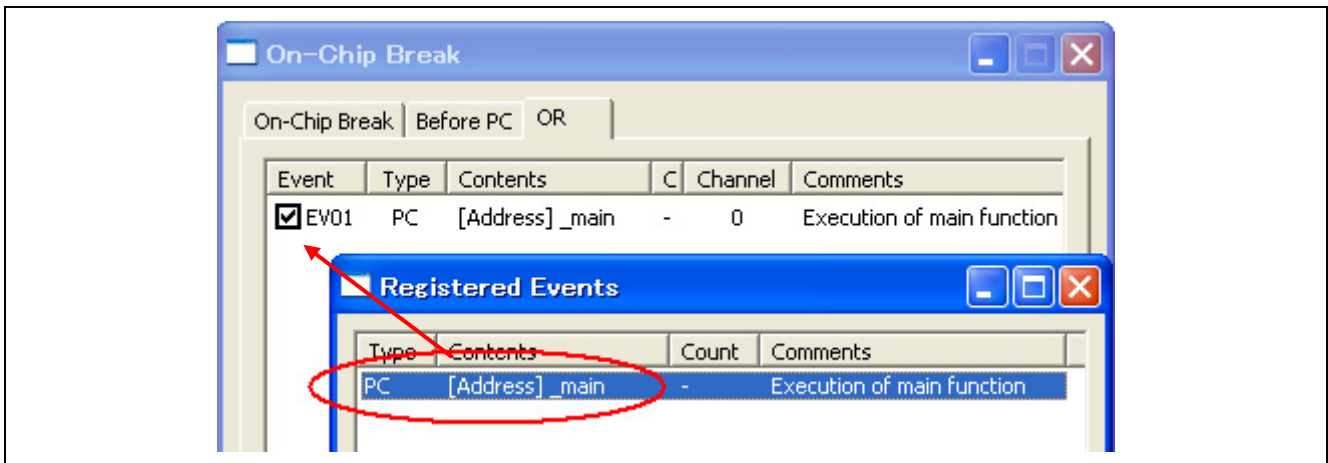
The event you have created is used without ever being registered.

Once the event is no longer in use (i.e., it has been changed or deleted), its setting is nonexistent.

Any event created by a simple operation such as double-clicking in the [Event] column of the [Editor] window constitutes an event created per usage.

- **Reusing events**

Any event registered in the [Registered Events] dialog box can be reused by dragging and dropping it into the [Event] list in the [On-Chip Break], [Trace conditions], or [Performance] dialog box.



**Figure 4.21 Reusing an Event**

(a) Dragging and dropping an event into multiple dialog boxes

An event in the [Registered Events] dialog box can be dragged and dropped into multiple dialog boxes.

If a condition of an event is altered after the event has been dragged and dropped, the alteration is not reflected in the setting of the original event in the [Registered Events] dialog box.

(b) Registering duplicates in the [Registered Events] dialog box

Even duplicate events that have the same conditions can be registered in the [Registered Events] dialog box.

#### 4.6.7 Activating Events

To activate the settings for events that you have created, click on the [Apply] button.

Settings you make do not become effective until you click on the [Apply] button.

[\*] after the title on the title bar of the [On-Chip Break], [Trace conditions], or [Performance] dialog box indicates that some setting is being edited. While you are editing an event, you cannot change the settings via the [Event] column of the [Editor] window or the command line.

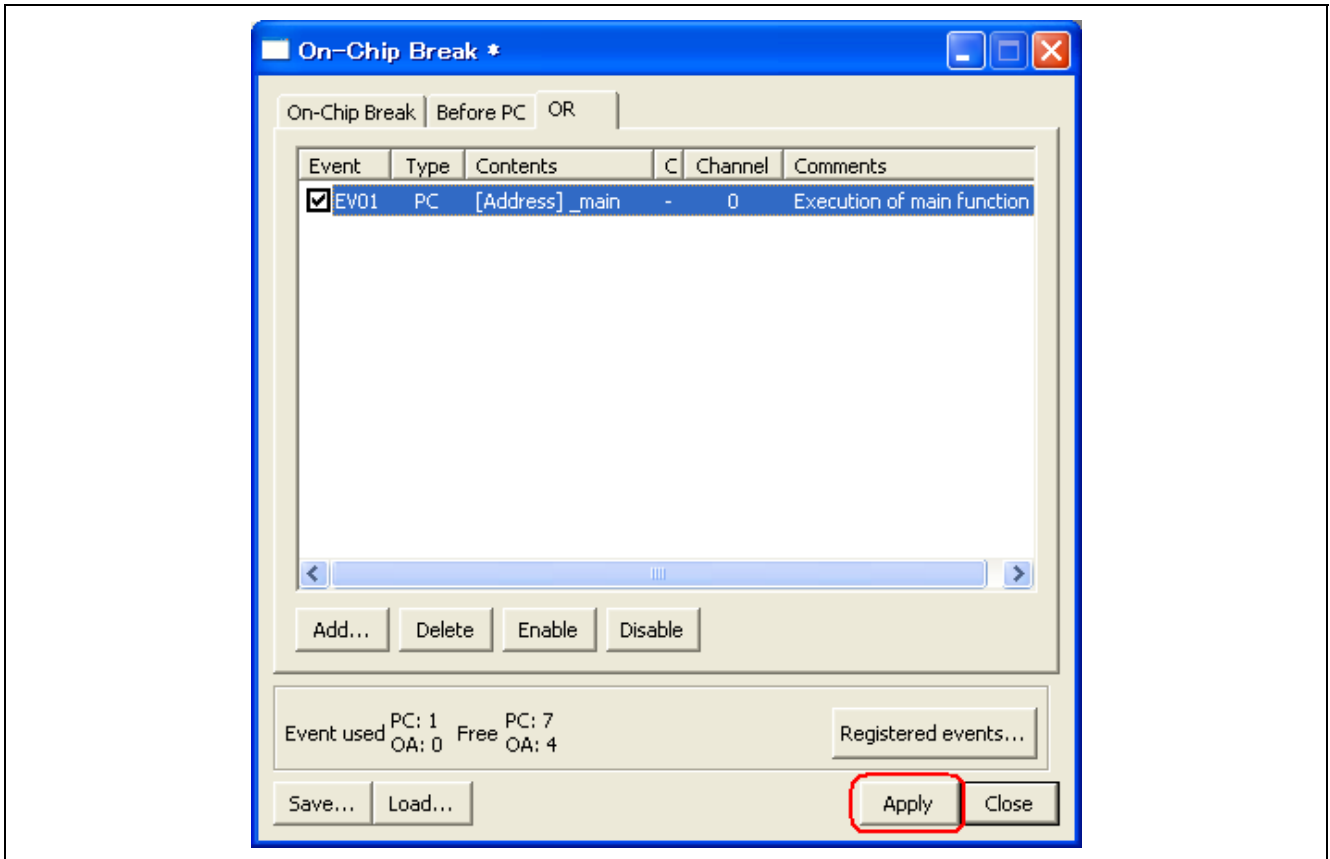


Figure 4.22 Applying the Settings

## 4.7 On-Chip Breakpoints

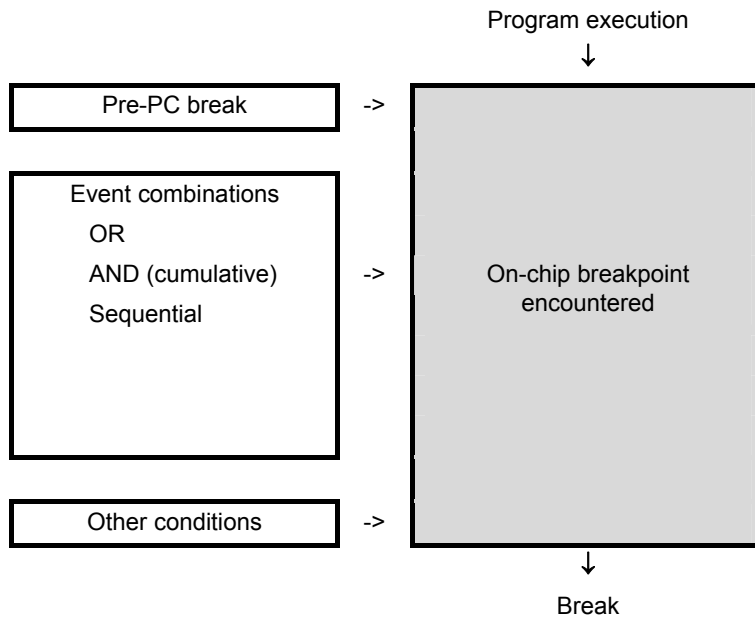
An on-chip break causes the user program to stop running when a specific event is detected.

### 4.7.1 Setting On-Chip Breakpoints

#### (a) Setting an On-Chip Breakpoint

For an on-chip breakpoint, you can set a pre-PC break condition, event combination (OR, AND (cumulative), or sequential) and other conditions.

You can specify all or only one from among the pre-PC break condition, event combination (OR, AND (cumulative), or sequential) and other conditions.



**Figure 4.23 On-Chip Break in Outline**

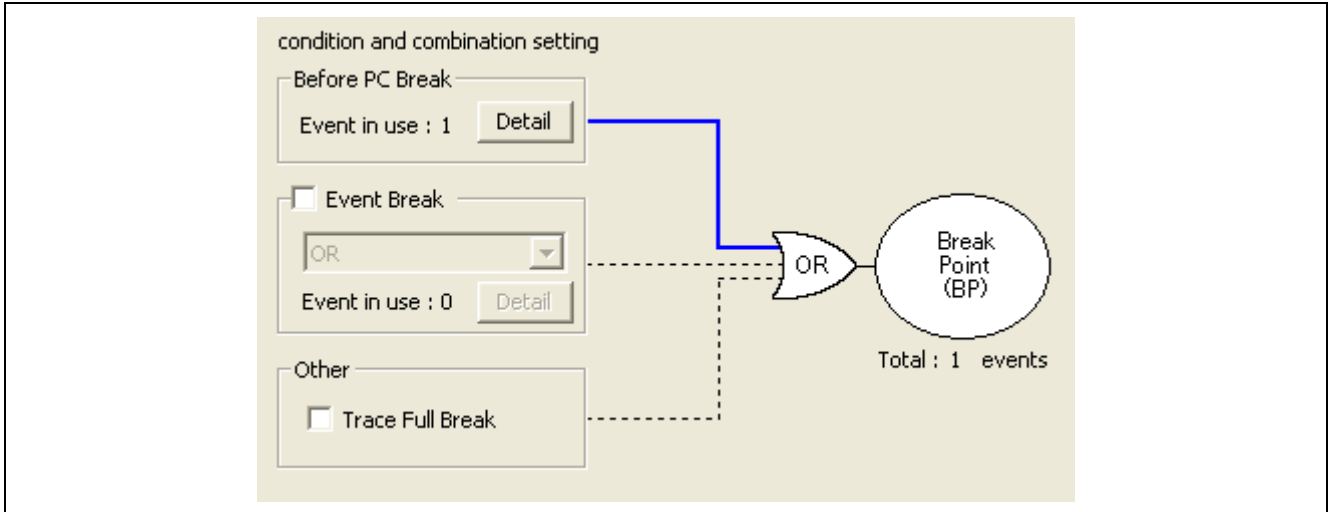
To set an on-chip breakpoint, open the [On-Chip Break] dialog box by selecting [View -> Event -> On-Chip Break].

(b) Setting a Pre-PC Breakpoint

[Before PC Break] is always enabled.

If you wish to disable [Before PC Break], delete or disable the registered event.

The break in execution will be immediately before execution of the instruction at the specified address.

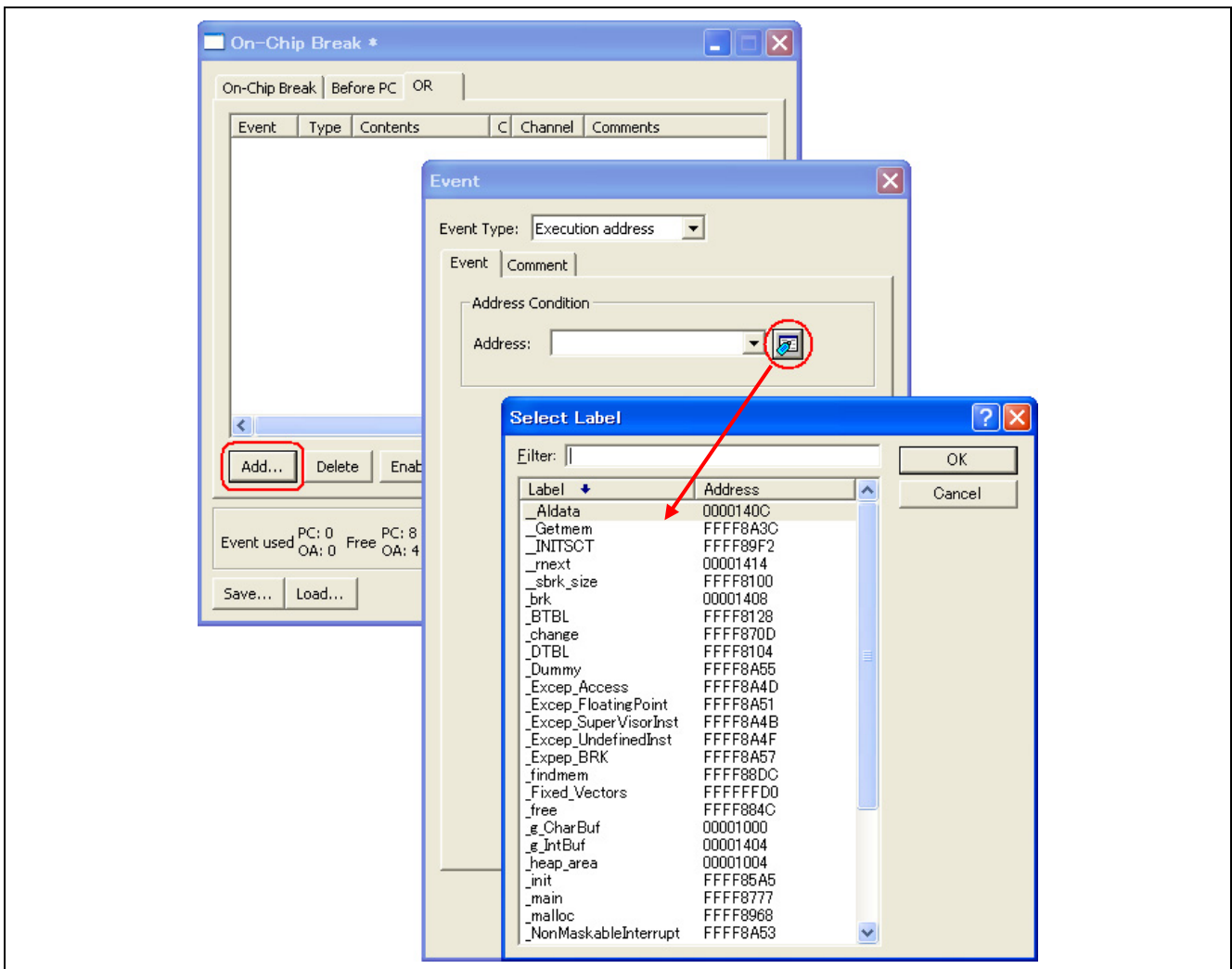


**Figure 4.24** [On-Chip Break] Dialog Box (Pre-PC Break)

Clicking on the [Detail] button for [Before PC Break] opens the [Before PC] page.



Clicking on the [Add] button on the [Before PC] page opens the [Event] dialog box, in which you can specify events for use as pre-PC breakpoints (i.e. execution-address conditions). It is also possible to specify a label as an event. The only event type selectable for a pre-PC break is “execution address”.



**Figure 4.25** Setting Pre-PC Break Conditions

[\*] after the title in the title bar of the [On-Chip Break] dialog box indicates that the settings have not been activated yet. When you click on the [Apply] button to activate the settings, [\*] in the title bar will disappear.

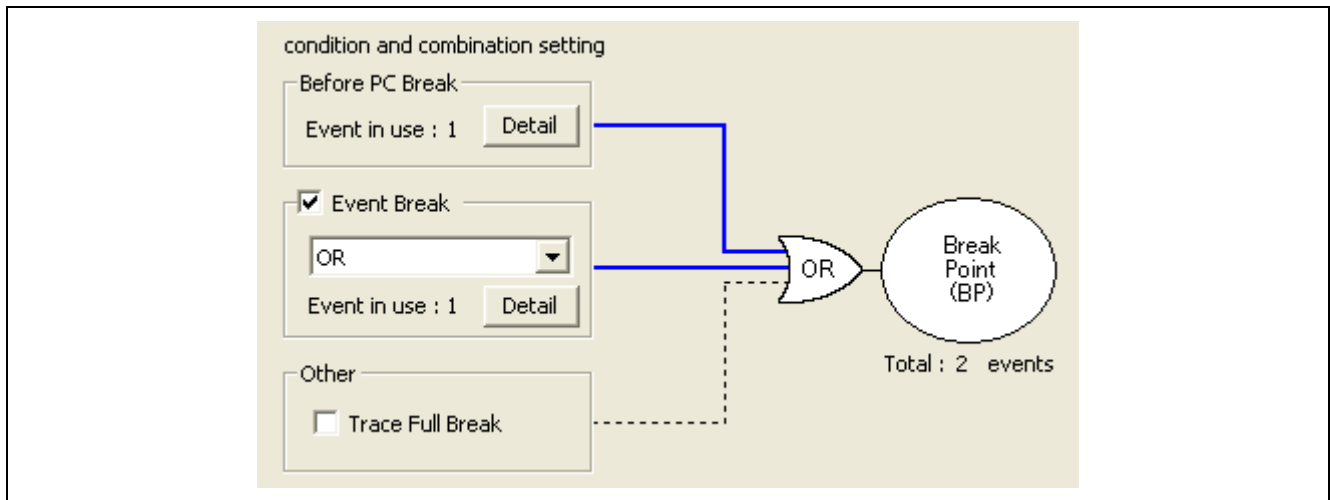
## (c) Setting an Event Break

You can select [OR], [AND (simultaneous)], or [Sequential].

To enable an event break, select the checkbox to the left of [Event Break].

To disable an event break, remove the checkmark from the checkbox to the left of the [Event Break].

[Event Break] is disabled by default (the checkbox to the left of [Event Break] is not selected).



**Figure 4.26** [On-Chip Break] Dialog Box (Event Break)

**Table 4.5** List of Event Break Items

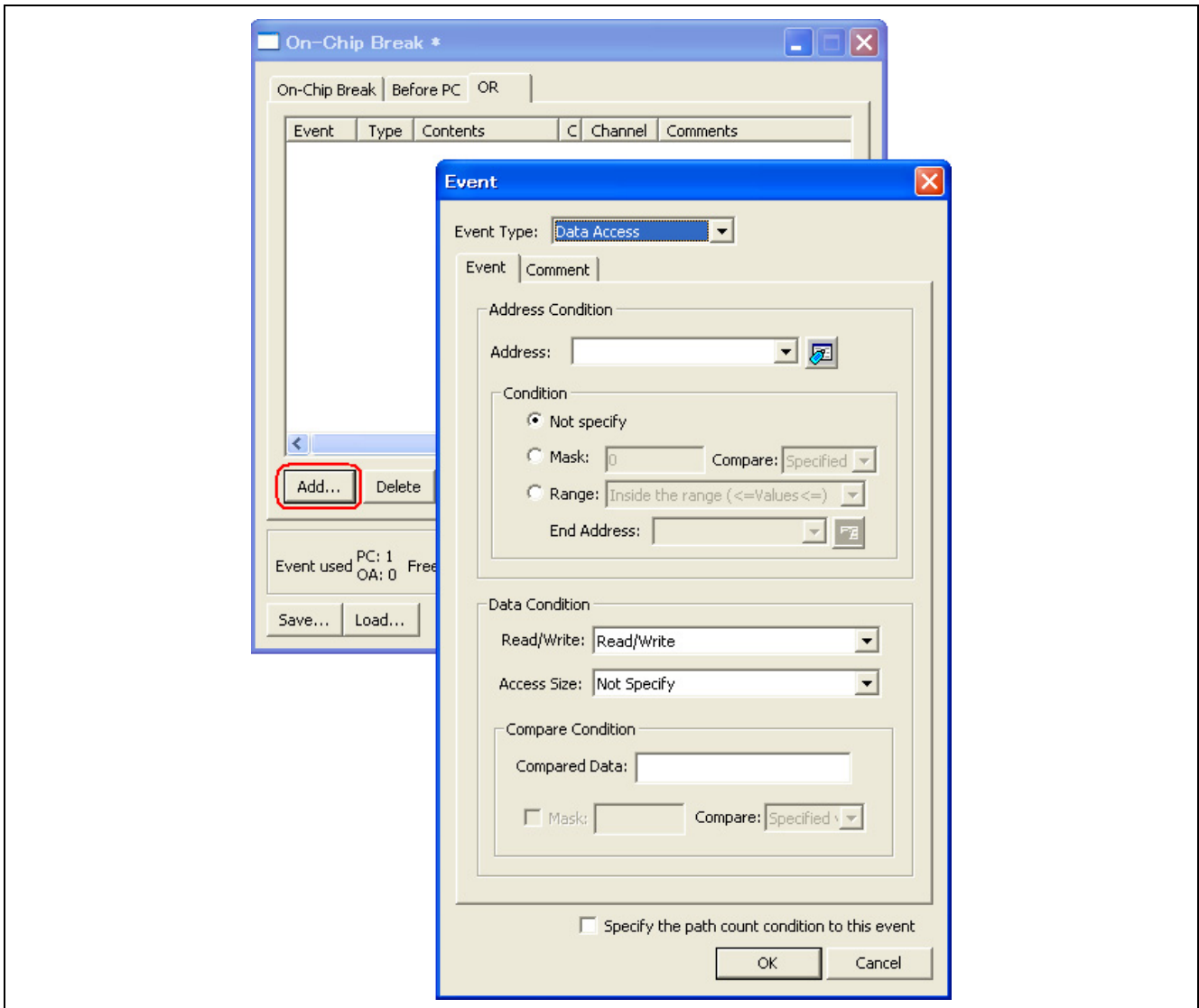
Type	Description
OR	When any one of the set events occurs, the break condition is met.
AND (cumulative)	When all of the set events occur regardless of the timing, the break condition is met.
Sequential	When the set events occur in a specified order, the break condition is met. RX600 Series MCUs: 7 steps (forward direction) + reset point (R)* <sup>1</sup> RX200 Series MCUs: 3 steps (forward direction) + reset point (R) * <sup>1</sup>

Note: 1. Reset point (R): When the event that is set at a reset point occurs, all of the events that have occurred until that time are cleared.

The events shown in the list for each condition can be deleted by the keys Ctrl + Del.

Clicking on the [Detail] button for [Event Break] opens the page that corresponds to the selected condition (e.g. the [OR] page when [OR] has been selected).

Clicking on the [Add] button on the page opens the [Event] dialog box, in which you can specify events for use as event breakpoints (i.e. execution-address or data-access conditions). It is also possible to specify a label as an event.



**Figure 4.27 Setting Event Break Conditions**

[\*] after the title in the title bar of the [On-Chip Break] dialog box indicates that the settings have not been activated yet. When you click on the [Apply] button to activate the settings, [\*] in the title bar will disappear.

(d) Setting Other Conditions

Specify whether you want the following condition to be used as a breakpoint.

- Trace full break

#### 4.7.2 Saving/Loading On-Chip Break Settings

##### (a) Saving On-Chip Break Settings

Click on the [Save] button of the [On-Chip Break] dialog box. The [Save As] dialog box will be displayed.

Specify the name of the file where you want the on-chip break settings to be saved. The file-name extension is ".hev". If this is omitted, the extension ".hev" is automatically appended.

##### (b) Loading On-Chip Break Settings

Click on the [Load] button of the [On-Chip Break] dialog box.

The [Open] dialog box will be displayed.

Specify the name of the file you want to load.

When you load a file, the previous on-chip break settings are discarded and the new settings appear in the dialog box.

Click on the [Apply] button of the [On-Chip Break] dialog box to activate the new on-chip break settings you have loaded.

## 4.8 Trace Functions

The E1/E20 emulator has two kinds of trace functions available to use: an internal trace and an external trace, as shown in Table 4.6.

**Table 4.6 Trace Functions**

Function	Internal Trace	External Output Trace
Branch trace	Supported	Supported
Data trace	Supported	Supported

Table 4.7 shows the products that the external output trace function can be used.

**Table 4.7 Name of the Product and External Output Trace Function**

Name of the Product (Type Number)	External Output Trace Function
E1 (R0E000010KCE00)	Not supported
E20 (R0E000200KCT00)	Supported

Internal trace or external output trace can be set in the [Trace conditions] dialog box in the [Trace] window.

### (1) Internal Trace Function

The internal trace function uses the trace buffer in the MCU.

Instruction-fetch and operand-access events can be used to acquire CPU-bus trace data on branches and data access.

For the RX600 Series MCUs, it is possible to obtain source/destination address information and data access information for up to 256 jumps or cycles.

For the RX200 Series MCUs, it is possible to obtain source/destination address information and data access information for up to 64 jumps or cycles. However, before data access information can be obtained, it is necessary that address conditions be set by the trace extraction function.

The trace results thus obtained can be displayed at the bus, disassembly or C source level in the trace window.

For functional specifications, refer to Table 3.3, "List of Functional Specifications by Target MCU."

### (2) External Trace-Output Function

The E20 includes a facility for acquiring trace data through the trace pins of MCUs. The E20 is capable of recording large amounts of trace data output in this way. Note, however, that the external trace-output function is only available when the E20 is connected to the 38-pin connector on the user board. The same types of trace data can be acquired through the external trace-output function as through the internal trace function, the only difference being that the maximum amount of trace data becomes 2-M branches or cycles.

There are two modes for external trace output: trace-output priority (full tracing) and CPU-execution priority (realtime trace).

#### (a) Trace-Output Priority Mode (Full Tracing)

All trace data will be output; none will be lost.

If the amount of trace data for input to the trace circuit starts to exceed the output capacity, the emulator forcibly suspends the CPU and bus so that no further data are input to the circuit. After all trace data that have been processed in the MCU are output, the emulator restarts the CPU and bus.

For this reason, operation in the trace-output priority mode is not necessarily realtime.

## (b) CPU-Execution Priority Mode (Realtime Trace Mode)

Trace data will be output in realtime.

If the amount of trace data input to the trace circuit starts to exceed the output capacity, some data will be discarded because realtime operation is given priority.

#### 4.8.1 Viewing Trace Information

Tracing means the acquisition of bus information per cycle and storage of this information in trace memory during user program execution.

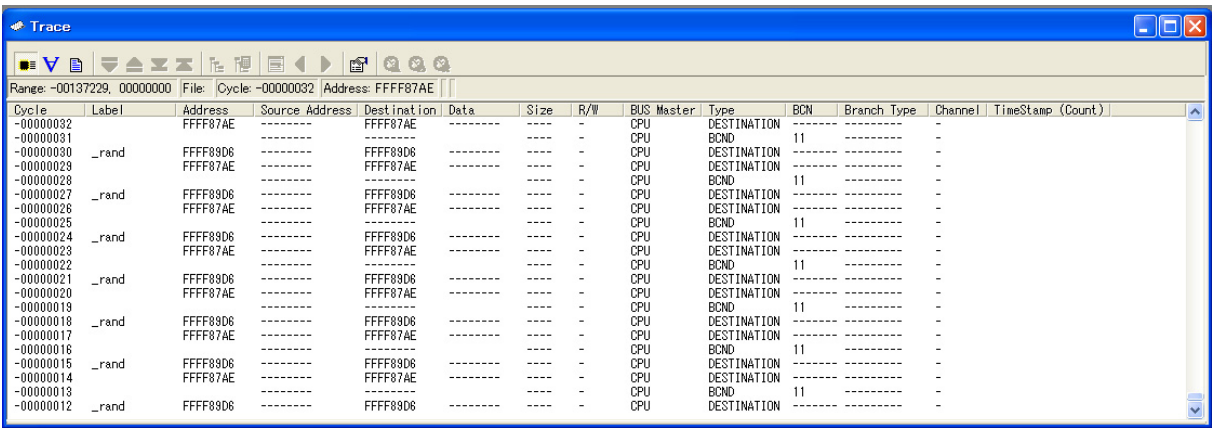
You can use tracing to track the flow of application execution or to search for and examine the points where problems arise.

#### 4.8.2 Acquiring Trace Information

In cases where no trace acquisition conditions are set, the default behavior of the emulator debugger is to acquire information on all bus cycles unconditionally (trace mode = [Fill until stop]).

In "fill until stop" mode, the emulator starts trace acquisition as soon as the user program starts running. When the user program stops, the emulator stops tracing.

The acquired trace information is displayed in the [Trace] window.



The screenshot shows a window titled "Trace" with a toolbar and a table of bus cycle data. The table has the following columns: Cycle, Label, Address, Source Address, Destination, Data, Size, R/W, BUS Master, Type, BCN, Branch Type, Channel, and TimeStamp (Count). The data rows show alternating cycles of CPU and BCND (Branch Condition Not Detected) with various addresses and destinations.

Cycle	Label	Address	Source Address	Destination	Data	Size	R/W	BUS Master	Type	BCN	Branch Type	Channel	TimeStamp (Count)
-00000032		FFFF87AE	-----	FFFF87AE	-----	----	-	CPU	DESTINATION	-----	-----	-	-
-00000031		FFFF87AE	-----	FFFF87AE	-----	----	-	CPU	BCND	11	-----	-	-
-00000030	_rand	FFFF89D6	-----	FFFF89D6	-----	----	-	CPU	DESTINATION	-----	-----	-	-
-00000029		FFFF87AE	-----	FFFF87AE	-----	----	-	CPU	DESTINATION	-----	-----	-	-
-00000028		FFFF87AE	-----	FFFF87AE	-----	----	-	CPU	BCND	11	-----	-	-
-00000027	_rand	FFFF89D6	-----	FFFF89D6	-----	----	-	CPU	DESTINATION	-----	-----	-	-
-00000026		FFFF87AE	-----	FFFF87AE	-----	----	-	CPU	DESTINATION	-----	-----	-	-
-00000025		FFFF87AE	-----	FFFF87AE	-----	----	-	CPU	BCND	11	-----	-	-
-00000024	_rand	FFFF89D6	-----	FFFF89D6	-----	----	-	CPU	DESTINATION	-----	-----	-	-
-00000023		FFFF87AE	-----	FFFF87AE	-----	----	-	CPU	DESTINATION	-----	-----	-	-
-00000022		FFFF87AE	-----	FFFF87AE	-----	----	-	CPU	BCND	11	-----	-	-
-00000021	_rand	FFFF89D6	-----	FFFF89D6	-----	----	-	CPU	DESTINATION	-----	-----	-	-
-00000020		FFFF87AE	-----	FFFF87AE	-----	----	-	CPU	DESTINATION	-----	-----	-	-
-00000019		FFFF87AE	-----	FFFF87AE	-----	----	-	CPU	BCND	11	-----	-	-
-00000018	_rand	FFFF89D6	-----	FFFF89D6	-----	----	-	CPU	DESTINATION	-----	-----	-	-
-00000017		FFFF87AE	-----	FFFF87AE	-----	----	-	CPU	DESTINATION	-----	-----	-	-
-00000016		FFFF87AE	-----	FFFF87AE	-----	----	-	CPU	BCND	11	-----	-	-
-00000015	_rand	FFFF89D6	-----	FFFF89D6	-----	----	-	CPU	DESTINATION	-----	-----	-	-
-00000014		FFFF87AE	-----	FFFF87AE	-----	----	-	CPU	DESTINATION	-----	-----	-	-
-00000013		FFFF87AE	-----	FFFF87AE	-----	----	-	CPU	BCND	11	-----	-	-
-00000012	_rand	FFFF89D6	-----	FFFF89D6	-----	----	-	CPU	DESTINATION	-----	-----	-	-

Figure 4.28 [Trace] Window

The following items are displayed (in bus display mode).

**Table 4.8** Items Shown in the [Trace] Window

Column	Description
Cycle	Number of the cycle within trace memory. By default, the number of the last cycle to have been acquired is 0, and earlier cycles are assigned progressively lower numbers in sequence, i.e. -1, -2, etc.
Label	Label corresponding to the address (displayed only when a label has been set)
Address	Address value. If there is no corresponding address, the entry is blank.
Source Address	Source address of a branch. If the cycle does not include such a value, the entry is “—”.
Destination Address	Destination address of a branch. If the cycle does not include such a value, the entry is “—”.
Data	Byte, word, or longword data displayed as hexadecimal values. If there was no access over the data bus in the given cycle, the entry is “—” or blank.
Size	Unit of access. BYTE: Byte WORD: Word LONG: Longword If there is no size value for the given cycle, the entry is “—”.
R/W	Data bus state. W: Writing (indicated by brick red) R: Reading (indicated by green) Cases where data on the bus were produced by neither writing nor reading are indicated by black. If there is no R/W value for the given cycle, the entry is “—”.
BUS Master	Bus master that triggered tracing. CPU: CPU access If there is no bus master for the given cycle, the entry is “—”.
Type	Type of trace information acquired. BCND: Conditional branch BRANCH: Branch source DESTINATION: Branch destination BRANCH/DESTINATION: Branch source and destination MEMORY: Operand access STANDBY: Standby information LOST: The corresponding trace information was lost. If there is no type value for the given cycle, the entry is “—”.

Table 4.8 Items Shown in the [Trace] Window (cont)

Column	Description
BCN	Whether or not the condition for execution of a conditional branch instruction was satisfied. 1: The condition was satisfied. 0: The condition was not satisfied. Each line can show the information on up to 15 branches. If there is no BCN value for the given cycle, the entry is “—”.
Branch Type	Reserved area. The entry is “—”.
Channel	Indicates an event match that occurred due to operand access. None: No operand access or the operand access did not match an event condition. 0: Event match on channel 0 1: Event match on channel 1 2: Event match on channel 2 3: Event match on channel 3 If there is no channel value for the given cycle, the entry is “—”.
Time Stamp (Count)	Displays TimeStamp. For the RX600 Series MCUs, it is displayed in decimal, from 0 up to 19 bits. For the RX200 Series MCUs, it is displayed in decimal, from 0 up to 23 bits. Overflows cannot be detected. The TimeStamp value of traces differ in count sources with each MCU.  [RX610, RX621, RX62N, RX62T, RX62G Group MCUs] If $EXTAL \times 8 \leq 100$ MHz TimeStamp count frequency = $EXTAL \times 8$ If $EXTAL \times 8 > 100$ MHz TimeStamp count frequency = $EXTAL \times 4$  [RX630, RX631, RX63N, RX63T Group MCUs] When using EXTAL: Selected clock source / 2 or Selected clock source x 1 (when division ratio of 1:1 is set by SCKCR.ICK) When not using EXTAL: Selected clock source x 1  [RX210, RX21A, RX220 Group MCUs] ICLK (performance counters used)

Columns of the [Trace] window can be hidden if you do not require them.

To hide a column, right-click in the header column and select the column you want to hide from the popup menu.



### 4.8.3 Results of Tracing

The [Trace] window shows the results of tracing.

Trace results can be shown in one of the following display modes: bus, disassembly, source, or mixed.

The display can be switched by changing the selection of [Display Mode] in the popup menu of the [Trace] window.

#### (a) Bus Display Mode

Select [Display -> BUS] in the popup menu.

Per-cycle bus information is shown. The contents vary with the MCU and emulator system in use. A mixed display of bus, disassembly, source, and data-access information is also possible.

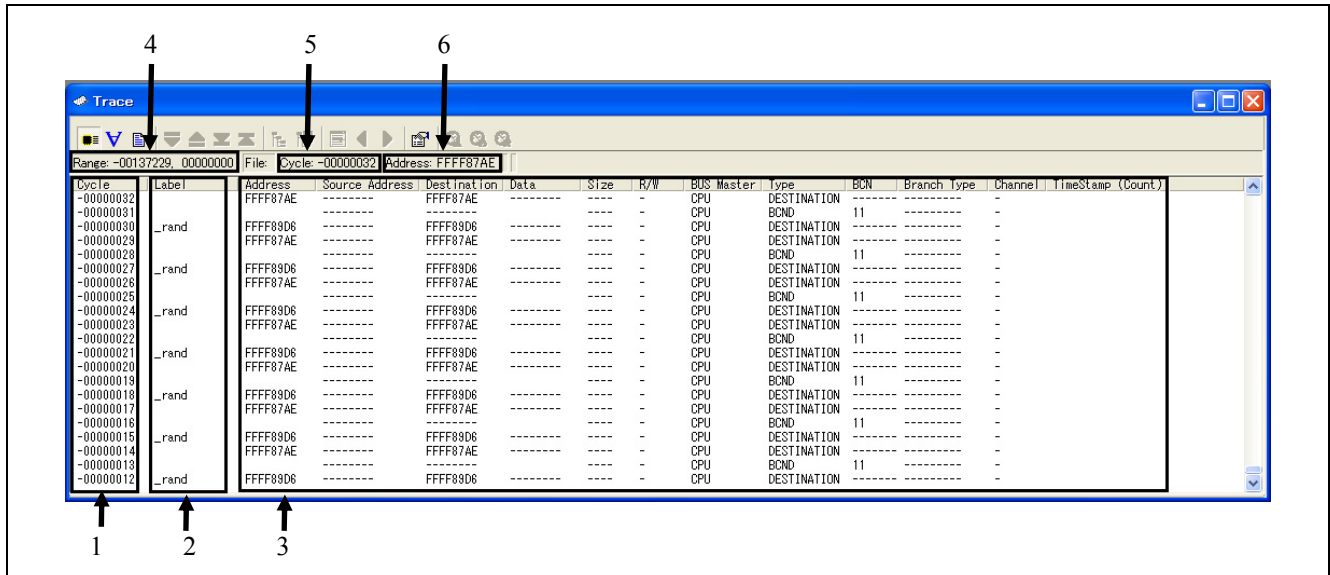


Figure 4.29 [Trace] Window in Bus Display Mode

1. Cycle number. Double-clicking on this area opens a further dialog box in which you can change the first cycle to be shown in the [Trace] window.
2. Label for the information on the address bus.
3. Bus information
4. Range of trace information that has been acquired
5. Cycle number of the first line
6. Address of the first line

## (b) Disassembly Display Mode

Select [Display -> DIS] in the popup menu.

Executed machine instructions are shown. A mixed display of disassembly, source, and data-access information is also possible.

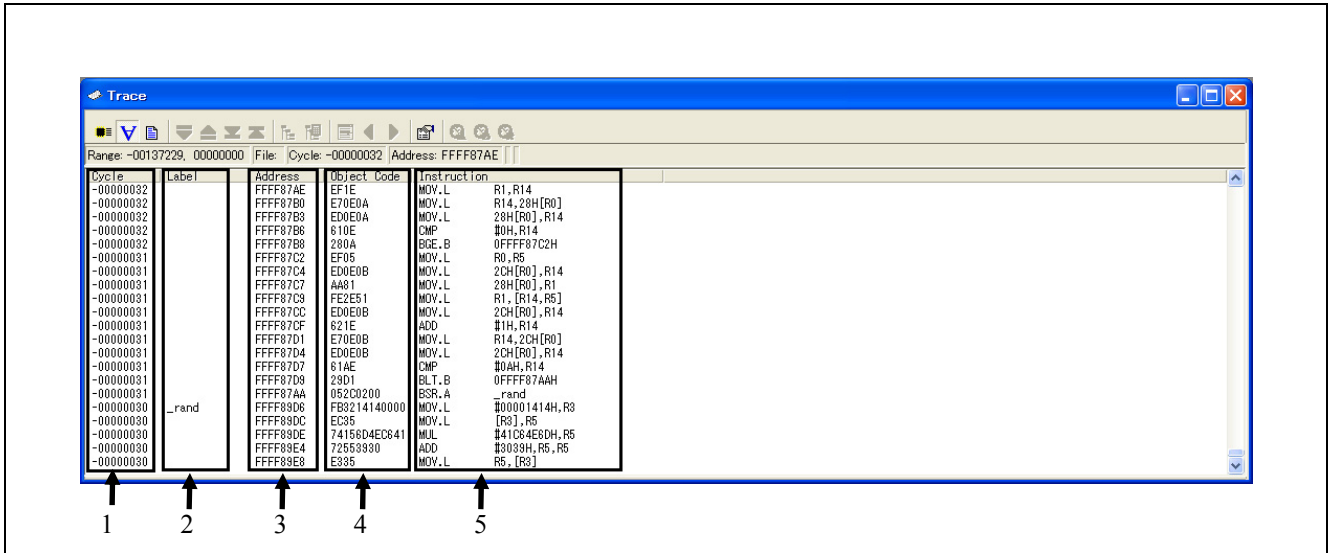


Figure 4.30 [Trace] Window in Disassembly Display Mode

1. Cycle number. Double-clicking on this area opens a further dialog box in which you can change the first cycle to be shown in the [Trace] window.
2. Label corresponding to the address of the instruction.
3. Address of the instruction. Double-clicking on this area opens a further dialog box in which you can search for a desired address.
4. Object code for the instruction
5. Instruction

## (c) Source Display Mode

Select [Display -> SRC] in the popup menu.

You can check the flow of execution by stepping forwards and backwards through the source code from the current trace cycle.

The result is displayed in the window when tracing stops and cleared when tracing resumes.

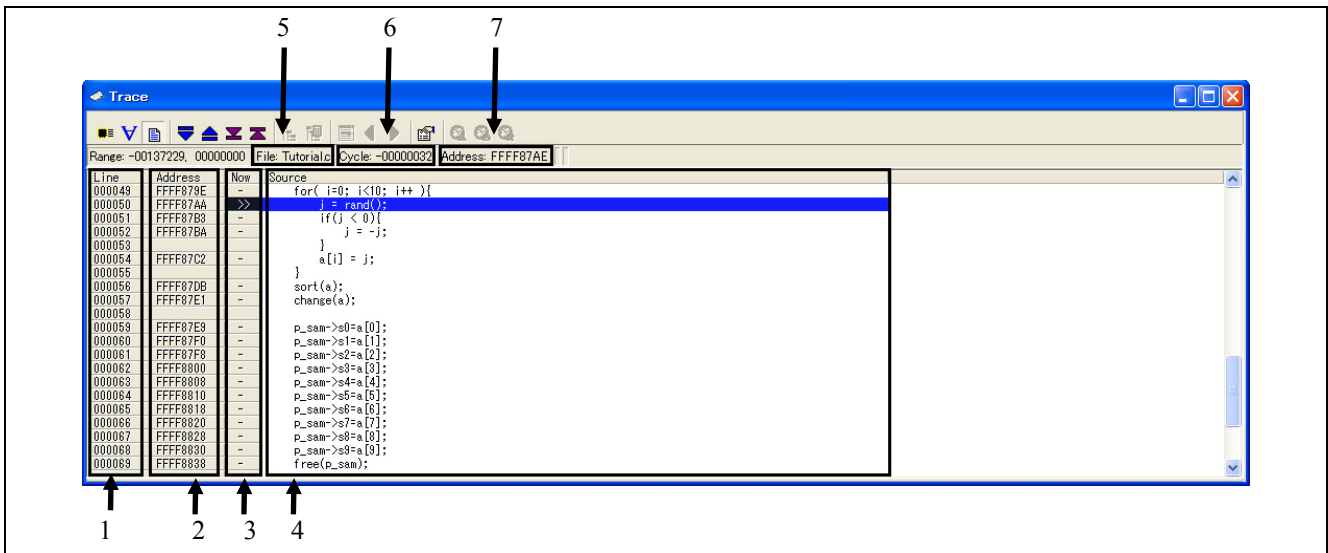


Figure 4.31 [Trace] Window in Source Display Mode

1. Line number in the current source file.
2. Address corresponding to the line of source code. Double-clicking on this area opens a further dialog box in which you can search for a desired address.
3. ">>" indicates the instruction corresponding to the cycle currently being referred to and "-" indicates that an address corresponds to the line.
4. Source code
5. Name of the current source file
6. Currently referenced cycle
7. Address of the cycle currently being referred to

## (d) Mixed Display Modes

Two or all of the modes can be selected at the same time, providing mixed displays of bus, disassembly, and source information.

After choosing [Display Mode -> BUS] from the popup menu, select [Display Mode -> DIS]. This produces a mixed display of bus and disassembly modes.

In the same way, you can produce mixed displays of bus–source, disassembly–source, or bus–disassembly–source.

To revert to bus mode after viewing a bus–disassembly mixed display, reselect [Display Mode -> DIS] from the popup menu.

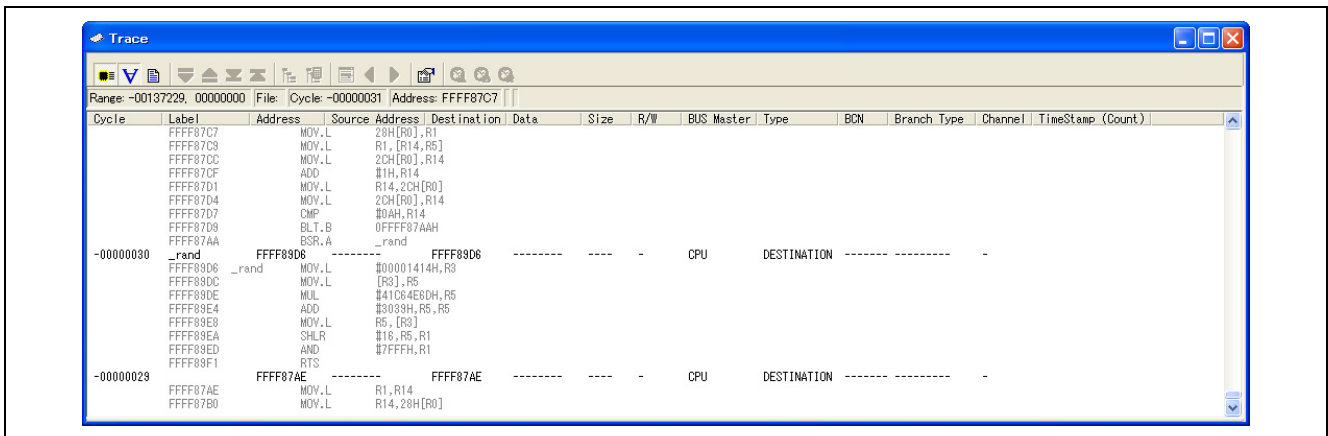


Figure 4.32 [Trace] Window for a Bus–Disassembly Mixed Display

## 4.8.4 Popup Menu Options

Right-clicking in the [Trace] window opens a popup menu containing the options listed below. Most of these options are also available as toolbar buttons.

**Table 4.9**      **Popup Menu Options**

Menu	Option	Description
Display Mode	BUS	Shows bus information.
	DIS	Shows disassembly information.
	SRC	Shows source information.
Trace	Step Forward	Steps forward through the source code from the current cycle (only usable in source mode)
	Step Backward	Steps back through the source code from the current cycle (only usable in source mode)
	Come Forward	Runs a program in the forward direction until a specified cursor position is reached (only usable in source mode)
	Come Backward	Runs a program in the reverse direction until a specified cursor position is reached (only usable in source mode)
	Stop* <sup>1</sup>	Only usable when trace data are being acquired during user program execution. This option stops tracing and updates the display of trace information (and is not usable in the internal trace mode).
	Restart* <sup>1</sup>	Restarts trace acquisition. It works only when trace acquisition has been halted temporarily during the user program execution. Not usable in the internal trace mode.
Find	Find...	Opens the [Find] dialog box, which is used to search for specific trace data.
	Find Previous	Searches for a preceding cycle that matches the pattern specified for [Find].
	Find Next	Searches for a subsequent cycle that matches the pattern specified for [Find].
Auto Filter	Enables or disables automatic filtering	
Acquisition...	Opens the [Trace conditions] dialog box, in which you can set trace conditions.	
Layout...	Opens the [Layout] dialog box, in which you can select whether each of the columns should be shown or hidden.	

Table 4.9 Popup Menu Options (cont)

Menu	Option	Description
File	Edit Source	Opens the current source file in the [Editor] window.
	Display Source	Select a source file you want to view in source display mode. A file selection dialog box will be opened.
	Save	Saves trace data in a file. You can select binary or text as the format in which the data will be saved. When you select binary format, data for all cycles are saved. When you select text format, the contents of the [Trace] window are saved. In text format, you can also save data for a specified range of cycles.  Furthermore, trace data can be saved in tab-separated format (if bus, disassembly and source information coexist as in a mixed mode display, the bus display is tab-separated and disassembly and source displays are space separated).
	Load	Loads trace data that have been saved. Only files saved in binary format can be loaded.
Toolbar display	Shows or hides the toolbar.	
Customize toolbar	Used to customize toolbar buttons.	
Allow Docking	Docks the window.	
Hide	Hides the window.	

Note: 1. Since the RX200 Series MCUs support only internal traces, this option menu cannot be used.

### 4.8.5 Setting Trace Conditions

Since the size of the trace buffer is limited, the oldest trace data is overwritten with new data after the buffer has become full.

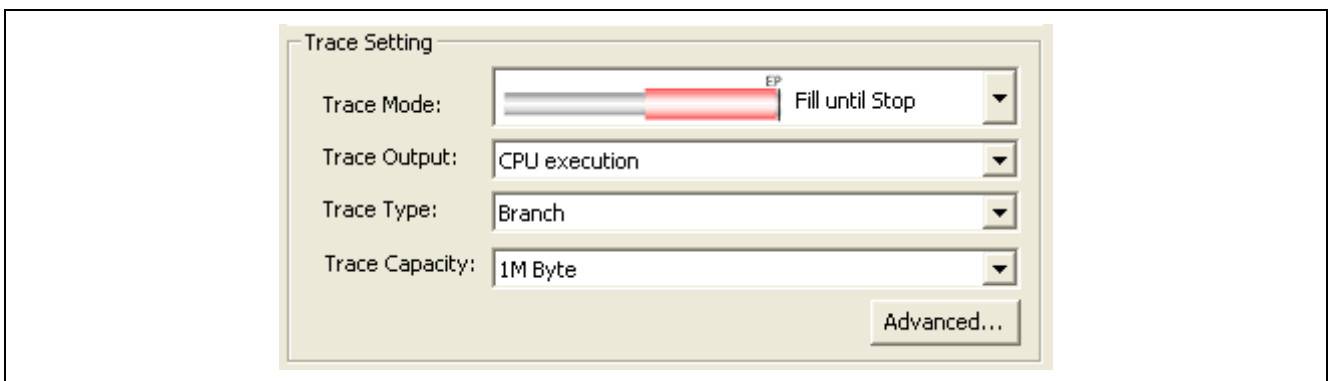
You can set trace conditions to restrict the acquired trace information to that which is useful, thus more effectively using the trace buffer.

To set trace conditions, use the [Trace conditions] dialog box that is displayed when you choose [Acquisition] from the popup menu of the [Trace] window.

Note that the trace capacity and the trace conditions you can set differ with each MCU. Refer to Table 3.3, “List of Functional Specifications by Target MCU.”

#### (a) Selecting the Trace Mode

Start by selecting the trace mode.



**Figure 4.33** [Trace conditions] Dialog box

The following four types of trace condition can be set in the [Trace Setting] section.

**Table 4.10** Trace Conditions

Parameter	Description
Trace Mode	Select the trace mode.
Fill until stop	The acquisition of trace data continues until the program stops or an event selected to stop tracing occurs.
Fill until full	The acquisition of trace data stops when the trace buffer becomes full.
Trace Output* <sup>1</sup>	Select the trace-output mode.
CPU execution	CPU execution is given priority. Some trace data may be lost.
Trace output	Tracing is given priority. CPU execution stops for the output of trace data, so this affects the realtime operation.
Do not output	The trace buffer in the MCU will be used (i.e. no trace data are output).

Table 4.10 Trace Conditions (cont)

Parameter	Description
Trace Type	Select the type of trace data.
Branch	[RX600 Series MCUs]
Branch + Data	Branch Traces source and destination address information on branch processes that occurred during program execution.
Data	Branch + Data Branch and data-access information is acquired.
Branch(Src)	Data Traces data information on events that occurred during program execution.
Branch(Src) + Time	
Data + Time	
	[RX200 Series MCUs]
	Branch Traces source and destination address information on branch processes that occurred during program execution.
	Data* <sup>2</sup> Traces data information on events that occurred during program execution.
	Branch(Src) Traces only the source address information on branch processes that occurred during program execution.
	Branch(Src) + Time Traces source address information and timestamp on branch processes that occurred during program execution.
	Data + Time* <sup>2</sup> Traces data information and timestamp on events that occurred during program execution.
Trace Capacity* <sup>3</sup>	Select the capacity of the trace buffer.
1, 2, 4, 8, 16, or 32 Mbyte	1, 2, 4, 8, 16, or 32 Mbytes

- Notes:
1. For the RX200 Series MCUs, this parameter is grayed out and a remark "Do not output" is displayed.
  2. For the RX200 Series MCUs, only the data accesses that you set in trace extraction conditions can be traced.
  3. For the RX200 Series MCUs, this parameter is grayed out.

Clicking on the [Advanced...] button on the [Trace] page opens the [Advanced Setting] dialog box, in which items to be shown in the [Trace] window can be selected. Select the checkboxes for the types of information that you wish to view in the [Trace] window.

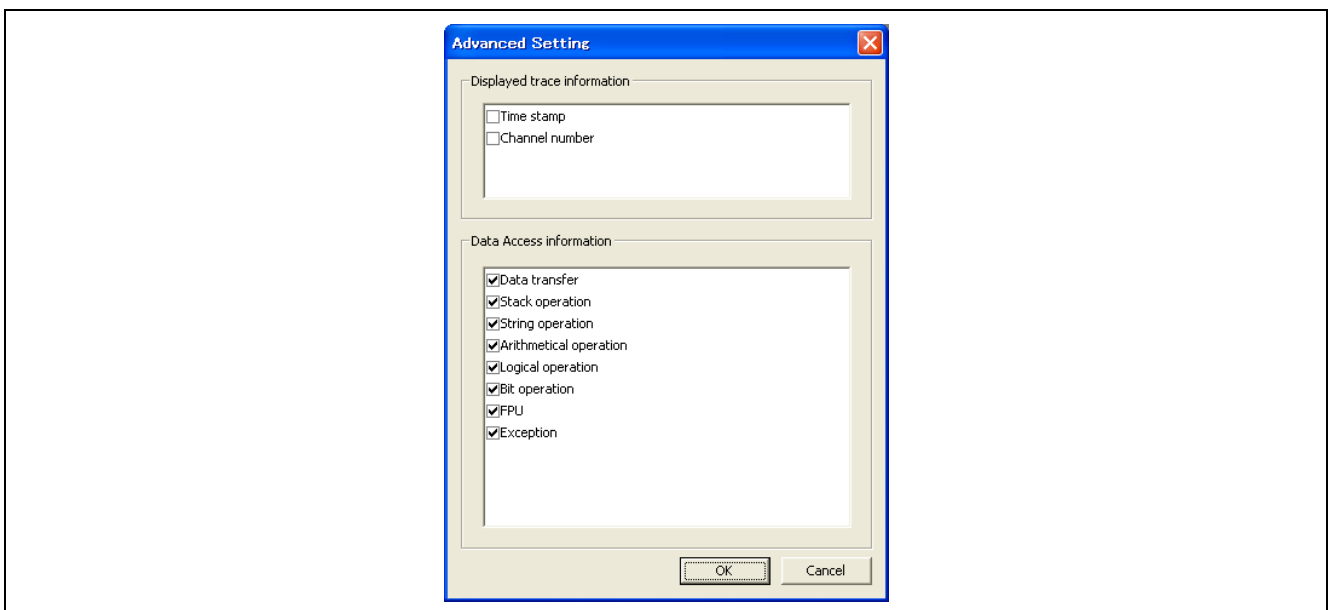


Figure 4.34 [Advanced Settings] Dialog box



(b) Setting Trace Conditions

You can set the conditions for starting and stopping trace acquisition, as well as the condition for extracting information from the trace acquisition. When no trace conditions are set, program execution is traced from when it started till when it stops.

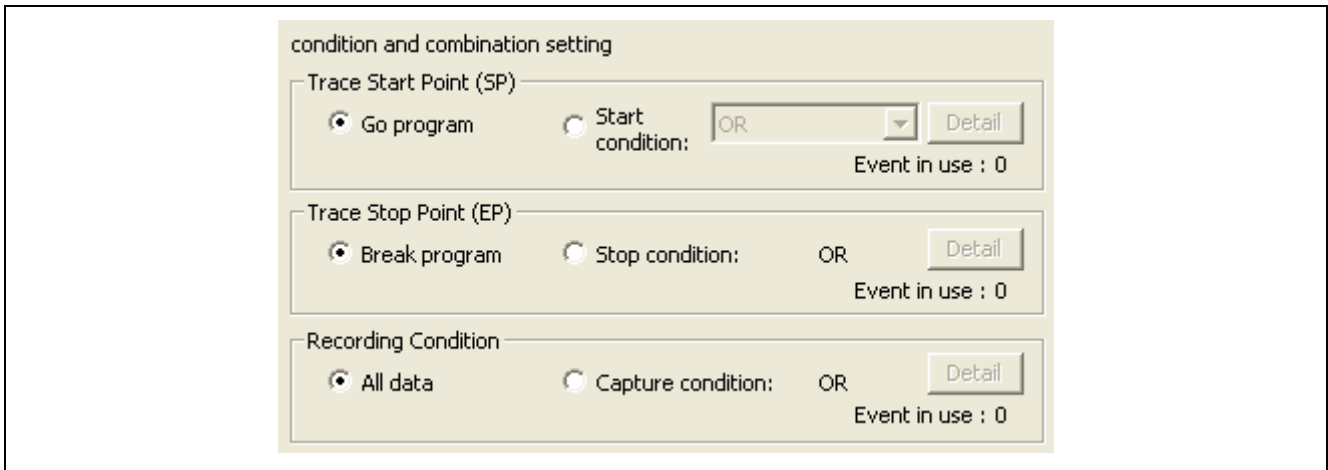


Figure 4.35 [Trace conditions] Dialog box

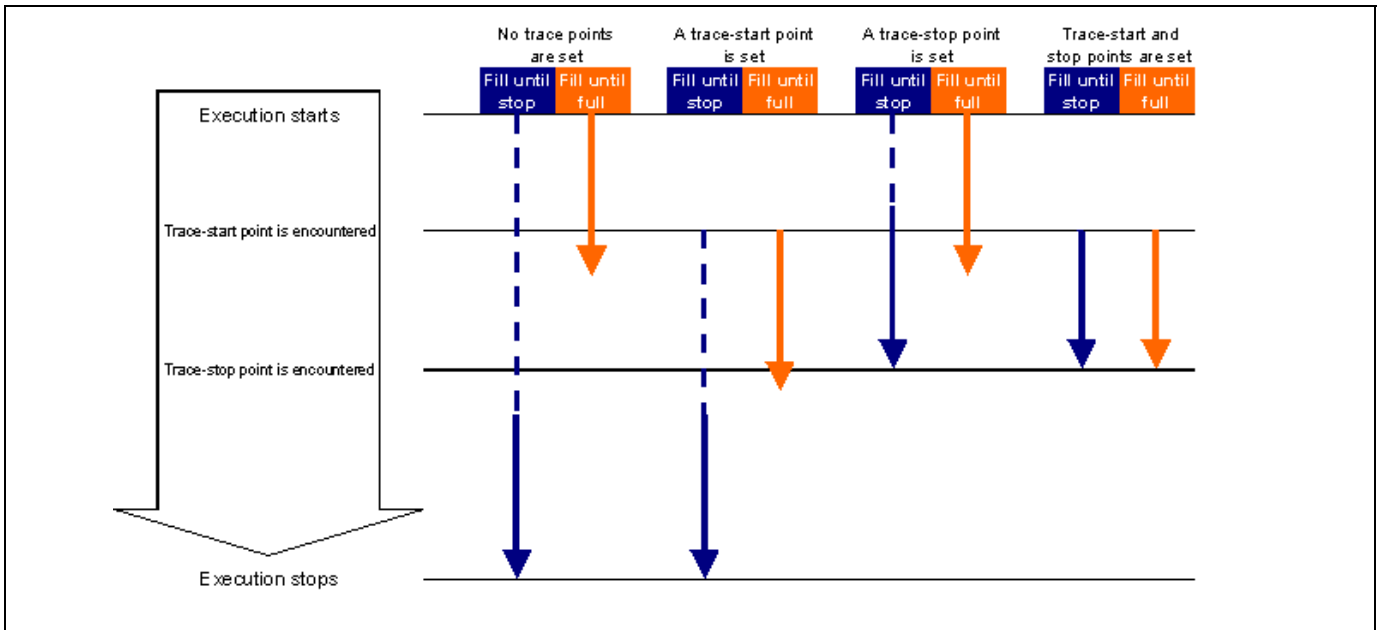
Trace conditions are of the three types described in the table below.

Table 4.11 Types of Trace Conditions

Item	Description
Trace Start Point (SP)	The trigger to start trace acquisition can be specified by a combination of events. If no trace acquisition start conditions are set, trace acquisition starts at the same time the program starts.
	OR The trace acquisition start condition is met when any of the set events occurs.
	AND (cumulative) The trace acquisition start condition is met when all of the set events occur irrespective of the time base.
	Sequential The trace acquisition start condition is met when the set events occur in a specified order. RX600 Series MCUs: 7 steps (forward direction) + reset point (R)* <sup>1</sup> RX200 Series MCUs: 3 steps (forward direction) + reset point (R) * <sup>1</sup>
Trace Stop Point (EP)	The trigger to stop trace acquisition can be specified by an event ("OR" only). If no trace acquisition stop conditions are set, trace acquisition stops at the same time the program stops.
Recording Condition* <sup>2</sup>	The trigger to extract trace information can be specified by an event ("OR" only).

Notes: 1. Reset point (R): When the event that is set at a reset point occurs, all of the events that have occurred up to that time are cleared.  
2. For the RX200 Series MCUs, this is grayed out when trace condition (OR) is specified.

The selected trace mode (“fill until stop” or “fill until full”) and whether or not trace conditions have been set determine the period of tracing. The differences are shown below.



**Figure 4.36** Differences between Trace Modes and Trace Condition Settings

#### (c) Saving/Loading Trace Settings

##### 1. Saving trace settings

Click on the [Save] button of the [Trace conditions] dialog box. The [Save] dialog box will be displayed.


Specify the name of the file where you want the trace settings to be saved. The file-name extension is ".tev". If this is omitted, the extension ".tev" is automatically appended.

##### 2. Loading trace settings

Click on the [Load] button of the [Trace conditions] dialog box. The [Load] dialog box will be displayed. Specify the name of the file you want to load.

When you load a file, the previous trace settings are discarded and the new settings appear in the dialog box. Click on the [Apply] button of the [Trace conditions] dialog box to activate the new trace settings you have loaded.

#### 4.8.6 Saving Trace Information in Files

To save trace information in a file, choose [File -> Save] from the popup menu or click on the [Save] toolbar button (). The trace information displayed in the [Trace] window is saved in a binary or text format.

##### (a) Saving in the Binary Format

To save trace information in the binary format, choose [Trace Data File: Memory Image (\*.rtt)] in the [Save As Type] list box of the dialog box that is displayed when you choose [File -> Save] from the popup menu.


When information is saved in the binary format, information for all cycles is saved. This type of file can be loaded back into the [Trace] window.

##### (b) Saving in the Text Format

To save trace information in the text format, choose [Text Files: Save Only (\*.txt)] in the [Save As Type] list box of the dialog box that is displayed when you choose [File -> Save] from the popup menu.


When information is saved in the text format, saving of information for a range of cycles can be specified. This type of file can only be saved and cannot be loaded back into the [Trace] window.

#### 4.8.7 Loading Trace Information from Files

To load trace information from a file, choose [File -> Load] from the popup menu or click on the [Load] toolbar button (). Specify a trace information file that was saved in the binary format. The current results of tracing are overwritten. Before loading a file saved in the binary format, switch to the trace mode in which the saved trace information was acquired. This switching should be performed in the [Trace conditions] dialog box that is displayed when you choose [Acquisition] from the popup menu of the [Trace] window.


Trace information files saved in the text format cannot be loaded back into the [Trace] window.

#### 4.8.8 Temporarily Stopping Trace Acquisition

To temporarily stop the acquisition of trace information during user program execution, choose [Trace -> Stop] from the popup menu of the [Trace] window or click on the [Stop] toolbar button ().

Trace acquisition will be stopped, with the trace display updated. Use this function when you only want to stop acquisition and check the trace information but not to stop program execution.

#### 4.8.9 Restarting Trace Acquisition

If you want to restart trace acquisition after it has temporarily been stopped during user program execution, choose [Trace -> Restart] from the popup menu of the [Trace] window or click on the [Restart] toolbar button ().

## 4.9 Measuring Performance

### 4.9.1 Measuring Performance

The performance measurement facility of the emulator is capable of measuring the total times execution takes and the number of passes for each of up to two specified sections of the user program, and of showing the time and number of cycles taken by execution.

The performance-measurement counter in the MCU is used to measure performance in overall execution (i.e. from [Go] to [Stop]) or between a start event and an end event. It is also possible to restart performance measurement while the user program is running. Instruction-fetch events, data-access events, and combination of both are specifiable as the start event and end event.

Since this facility uses the emulator's performance measurement circuit to measure the execution time, it does not impede execution of the user program.

Performance measurement conditions cannot be manipulated during program execution.

For the RX600 Series MCUs, select one of the following 11 items as the target for measurement: [Not use], [Execution cycle], [Execution cycle (supervisor mode)], [Exception and interrupt cycle], [Exception cycle], [Interrupt cycle], [Execution count], [Exception and interrupt count], [Exception count], [Interrupt count], and [Event match count].

You can also choose whether two 32-bit counters (ch. 0 and ch. 1) should be used separately or handled in combination as a 64-bit counter.

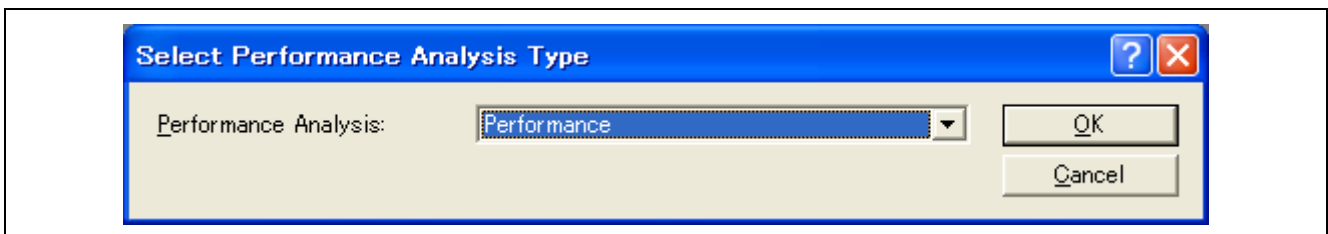
For the RX200 Series MCUs, measurement is possible for only elapsed cycles, with a 24-bit counter x 1 ch supported.

For functional specifications, refer to Table 3.3, "List of Functional Specifications by Target MCU."

### 4.9.2 Viewing the Results of Performance Measurement

Results of measurement are displayed in the [Performance Analysis] window.

To open the [Performance Analysis] window, choose [View -> Performance -> Performance Analysis] or click on the [Performance Analysis] toolbar button (E). The [Select Performance Analysis Type] dialog box appears. Click on the [OK] button.



**Figure 4.37** [Select Performance Analysis Type] Dialog Box

The [Performance Analysis] window appears.

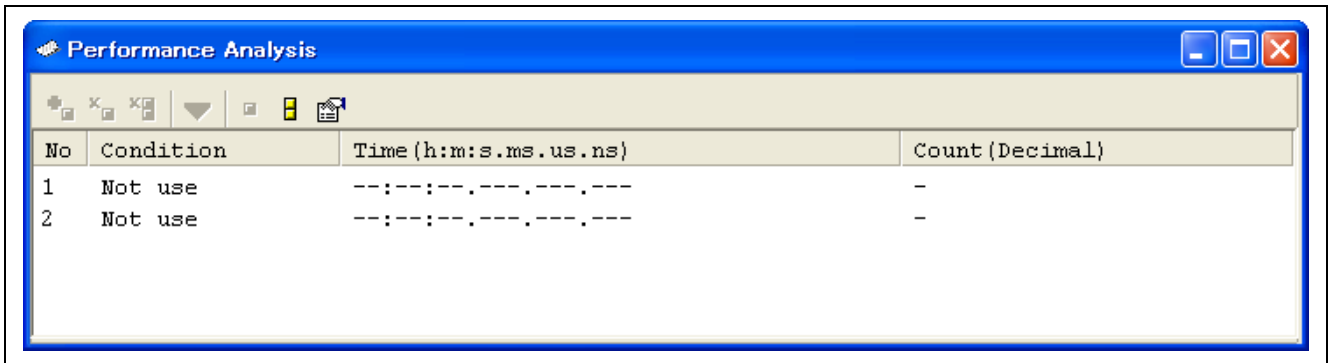
Right-click on the window and select [Set] from the popup menu to open the [Performance] dialog box, in which you can select events for use and set other conditions for performance measurement.

After setting performance-measurement conditions in the [Performance] dialog box, click on the [OK] button to execute the user program. When execution stops, the [Performance Analysis] window will show the execution time and number of cycles taken by code satisfying the conditions you have set.

Any unnecessary columns in the [Performance Analysis] window can be hidden.

To hide any column, right-click in the header column and select the column you want to hide from the popup menu.

To view any hidden column, reselect that column from the popup menu again.



No	Condition	Time (h:m:s.ms.us.ns)	Count (Decimal)
1	Not use	--:--:--.--.--.--	-
2	Not use	--:--:--.--.--.--	-

Figure 4.38 [Performance Analysis] Window

The columns in the [Performance Analysis] window are listed below.

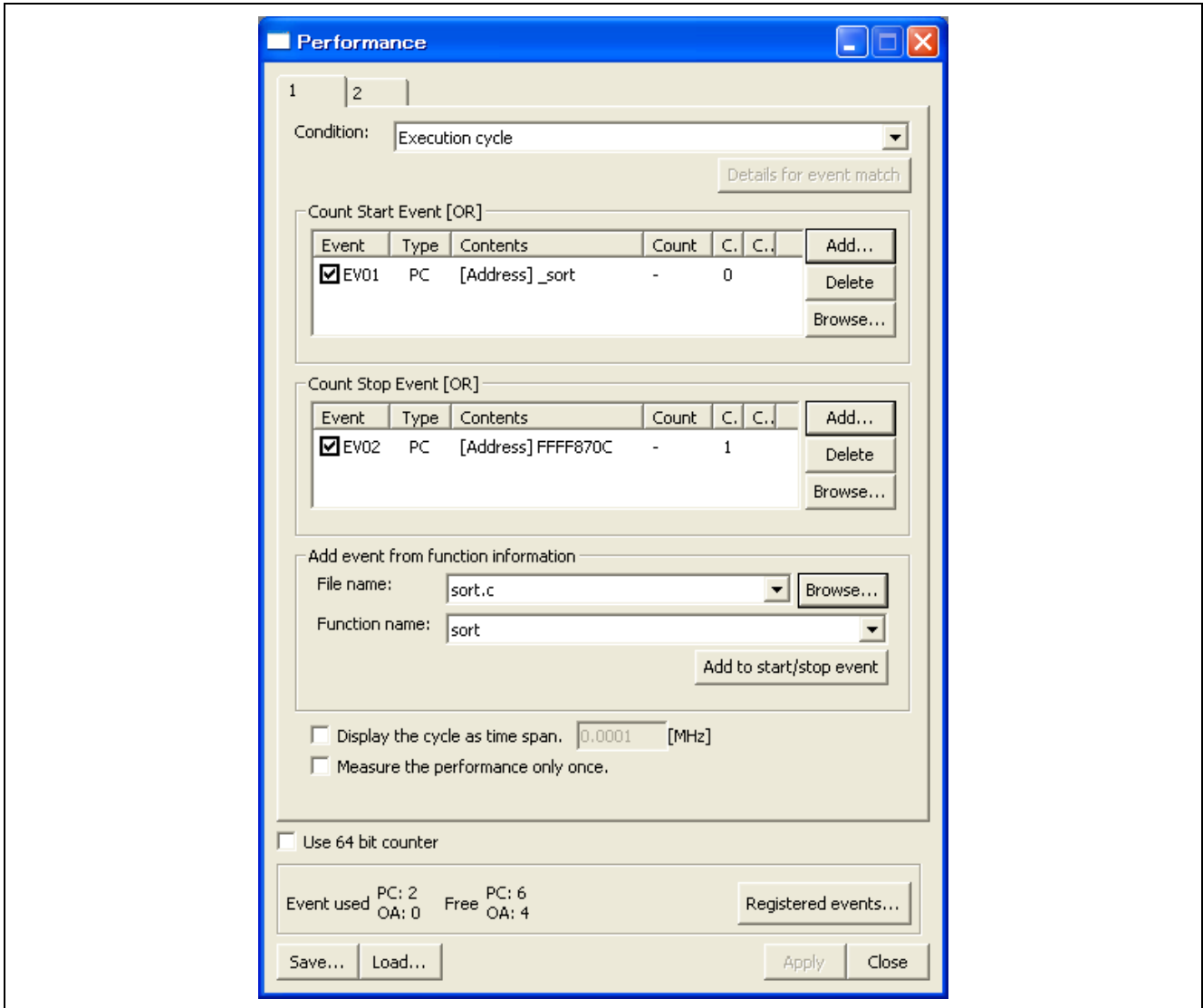
**Table 4.12 Columns in the [Performance Analysis] Window**

Column	Description
No	<p>Number set up in the [Performance] dialog box.</p> <p>[RX600 Series MCUs]</p> <p>Displays 1 and 2 when a 32-bit counter is used for each of two ranges or 1 when the two 32-bit counters are handled as a 64-bit counter for measurement of performance in a single address range.</p> <p>[RX200 Series MCUs]</p> <p>Displays only 1 with a 24-bit counter.</p>
Condition	<p>Target for measurement selected in the [Performance] dialog box.</p> <p>[RX600 Series MCUs]</p> <ul style="list-style-type: none"> <li>• Not in use: [Not use]</li> <li>• Number of cycles that have elapsed: [Execution cycle]</li> <li>• Number of cycles that have elapsed in supervisor mode: [Execution cycle (supervisor mode)]</li> <li>• Number of cycles of processing for interrupts or other exceptions: [Exception and interrupt cycle]</li> <li>• Number of cycles of processing for exceptions: [Exception cycle]</li> <li>• Number of cycles of processing for interrupts: [Interrupt cycle]</li> <li>• Number of valid instructions issued: [Execution count]</li> <li>• Number of interrupts and other exceptions accepted: [Exception and interrupt count]</li> <li>• Number of exceptions accepted: [Exception count]</li> <li>• Number of interrupts accepted: [Interrupt count]</li> <li>• Number of event-condition matches: [Event match count]</li> </ul> <p>[RX200 Series MCUs]</p> <ul style="list-style-type: none"> <li>• Number of cycles that have elapsed: [Execution cycle]</li> </ul>
Time (h:m:s.ms.us.ns)	<p>Cumulative total of measured execution times.</p> <p>[RX600 Series MCUs]</p> <ul style="list-style-type: none"> <li>• When the target for measurement is [Execution cycle], [Execution cycle (supervisor mode)], [Exception and interrupt cycle], [Exception cycle], or [Interrupt cycle]: Displays count in terms of time. Cumulative time calculated from the frequency specified by the user and the value in the [Count] column. Note that if the calculated value is 65,536 hours or more, this is displayed as 0h0m0s0ms0us0ns.</li> <li>• When the target for measurement is [Execution count], [Exception and interrupt count], [Exception count], [Interrupt count], or [Event match count]: Displays "---:---:---:---:---:---".</li> </ul> <p>[RX200 Series MCUs]</p> <ul style="list-style-type: none"> <li>• Displays count in terms of time. Cumulative time calculated from the frequency specified by the user and the value in the [Count] column. Note that if the calculated value is 65,536 hours or more, this is displayed as 0h0m0s0ms0us0ns.</li> </ul>
Count(Decimal)	<p>Decimal value indicating the number of times measurement for the section has proceeded. When the counter has overflowed, "overflow" is displayed.</p>

### 4.9.3 Setting Performance Measurement Conditions

In the [Performance Analysis] window, select the line of a section number to use for the condition and choose [Set] from the popup menu.

The [Performance] dialog box will be displayed.



**Figure 4.39 [Performance] Dialog Box**

[\*] after the title in the title bar of the [Performance] dialog box indicates that the settings have not been activated yet. When you click on the [Apply] button to activate the settings, [\*] in the title bar will disappear.

(a) Setting Measurement Conditions

For the RX600 Series MCUs, measurement of performance in up to two address ranges is possible by using a 32-bit counter for each range.

Alternatively, the two 32-bit counters can be handled as a 64-bit counter for measurement of performance in one address range. When the 64-bit counter for a single range is selected, the settings for the other range become invalid.

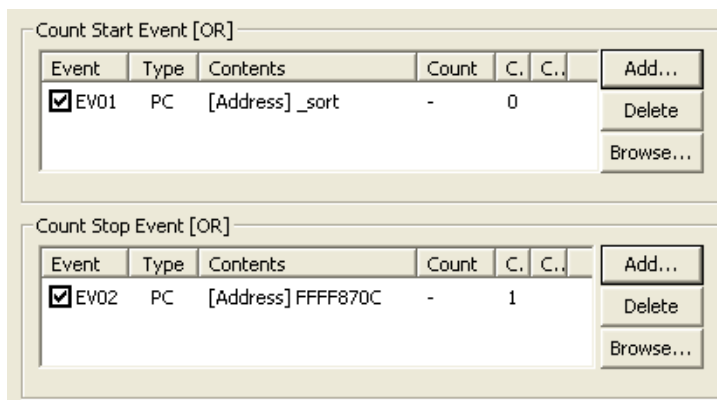
For the RX200 Series MCUs, measurement is possible in only 1 section using a 24-bit counter.

Select one measurement mode for one range.

Events are used to define these ranges.

**Table 4.13 Measurement Modes**

Target for Measurement	Description
None: [Not use]	Measurement is disabled.
Number of cycles: [Execution cycle] [Execution cycle (supervisor mode)] [Exception and interrupt cycle] [Exception cycle] [Interrupt cycle]	Between two events
Number of times specified processing is performed: [Execution count] [Exception and interrupt count] [Exception count] [Interrupt count]	



**Figure 4.40 Between Two Events**

Measurement is performed between the start event and the end event.

Specifically, the number of cycles or number of times in the range between the start event and the end event is measured.

Measurement of the number of cycles starts when the start event occurs and is suspended when the end event occurs.

The number of times is measured as the number of passes between the start event and end event that specify the range.

[Count Start Event]: One or multiple events can be set. If no events are set, measurement starts with execution of the program.

[Count Stop Event]: One or multiple events can be set. If no events are set, measurement stops with execution of the program.



Table 4.13 Measurement Modes (cont)

Target for Measurement	Description
Number of times the event has occurred: [Event match count]	Number of times the event has occurred

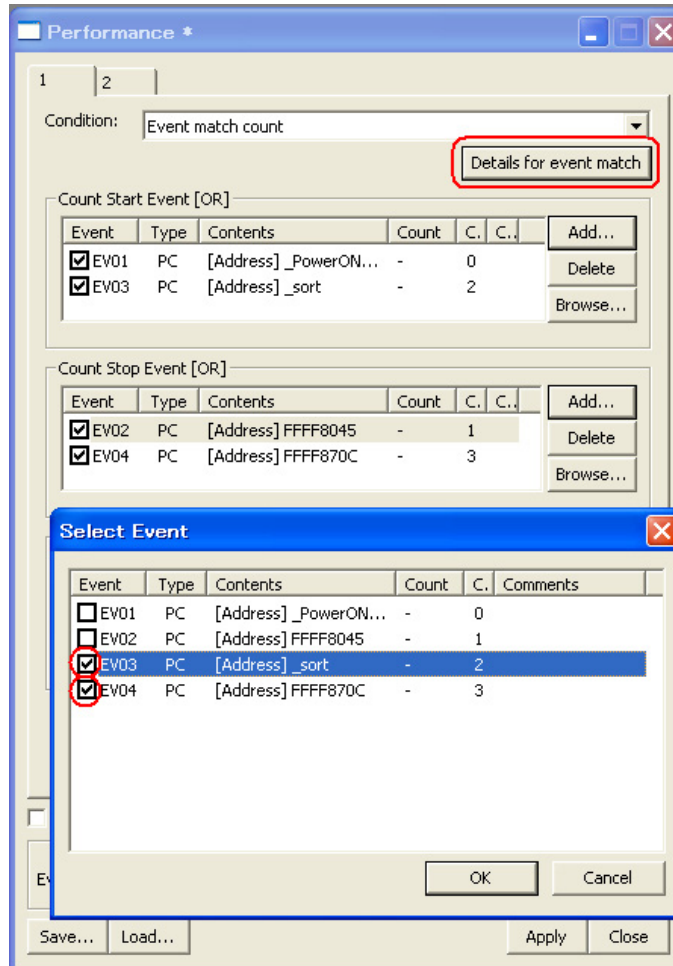


Figure 4.41 Number of Times the Event Has Occurred

The number of times a specific event has occurred is measured between the start event and the end event.

[Count Start Event]: One or multiple events can be set.

[Count Stop Event]: One or multiple events can be set.

The event-match counting conditions must be selected from among events specified as trace, on-chip break, or performance-measurement conditions.

Click on the [Details for event match] button to open the [Select Event] dialog box and specify the event for which the number of times is to be measured.

Table 4.13 Measurement Modes (cont)

Target for Measurement	Description
Number of times the event has occurred: [Event match count]	<p>Events specified in event-match count measurements are placed in a shared state.</p> <p>The events in a shared state cannot have their enable or disable settings changed.</p> <p>Some of such events may be placed out of the shared state by an event-related operation, becoming an independent event for event-match count measurements.</p> <p>To remove an independent event used in event-match count measurements, uncheck the specified event's checkbox in the [Select Event] dialog box and be sure to change its settings in the [Performance] dialog box to update its state.</p> <p>The specified event goes out of the screen at the same time its checkbox is unchecked above.</p> <p>Note that cancellations thereafter are unaccepted.</p> <p>The [Select Event] dialog box may have multiple events with the same name displayed in it.</p>

Note: To measure the execution time of a function, use [Between two events]. Specify execution of the instruction at the first address of the function as the start event and execution of the instruction at the exit point of the function (point corresponding to the line containing the function's return statement) as the end event. If there is more than one exit point, set a fetch condition that covers each of them as the end event.

## (b) Measurement Item

Table 4.14 shows the items selectable for measurement through the performance measurement function. The selected name is displayed under [Condition] in the [Performance Analysis] window.

**Table 4.14 Measurement List Item**

Condition	Selected Item
Not use	No item for performance analysis has been set.
Execution cycle	The number of cycles (ICLK) elapsing within the specified interval has been set as the item for measurement.
Execution cycle (supervisor mode)	The number of cycles (ICLK) of execution in supervisor mode has been set as the item for measurement.
Exception and interrupt cycle	The number of cycles (ICLK) for the execution of interrupt processing and other exception processing has been set as the item for measurement.
Exception cycle	The number of cycles (ICLK) for the execution of exception processing has been set as the item for measurement.
Interrupt cycle	The number of cycles (ICLK) for the execution of interrupt processing has been set as the item for measurement.
Execution count	The number of instruction for which execution is completed has been set as the item for measurement.
Exception and interrupt count	The number of accepted interrupts and other exceptions has been set as the item for measurement.
Exception count	The number of accepted exceptions has been set as the item for measurement.
Interrupt count	The number of accepted interrupts has been set as the item for measurement.
Event match count	The number of times the event condition is matched has been set as the item for measurement.

The exception and interrupt cycle on the performance measurement have the following exception event of the MCU.  
[Exception]

- Undefined instruction exception
- Privileged instruction exception
- Floating-point exceptions
- Unconditional trap (INT instruction BRK instruction)

[Interrupt]

- Non-maskable interrupt
- Interrupts (maskable)
- Reset

(c) Adding Events from Data on Functions\*<sup>1</sup>

The start and end events can be selected from data on functions in a source file. Click on the [Browse] button and select a source file. Then select a function and click on the [Add to start/stop event] button. The first and last addresses of the specified function will then be executed as the start and end events.

Note: 1. The start and end events are the first and last addresses of the specified function. Measurement does not proceed correctly if the last address is not the exit (generally an RTS instruction) of the function. When the location where the function ends does not correspond to the exit from the function, manually specify the exit as the end event.

## (d) Displaying Time Information

If “cycle” is selected for [Condition] as the measurement item in the [Performance] dialog box and the [Display the cycle as time span] checkbox is checked, the time information calculated from the frequency you input in the frequency edit box and the count value resulting from performance measurements is displayed.

The range of input values is 0.0001 to 999.999. If any value exceeding this range is input, an error message is displayed.

Note that input values are effective in up to 6 digits, so that when the integer part consists of 2 digits or less, fractions below 4 decimal places are truncated, and that if the integer part consists of 3 digits, fractions below 3 decimal places are truncated.

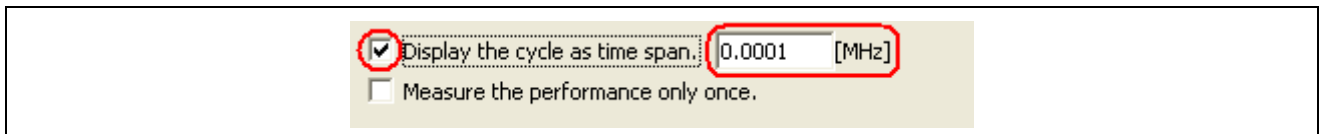


Figure 4.42 Displaying Time Information

## 4.9.4 Starting Performance Measurement

When the user program is run, performance measurement is automatically started according to the conditions set on performance measurement.

When the user program is halted, the results of measurement are displayed in the [Performance Analysis] window.

When execution of the user program is halted and then restarted without changing the conditions of measurement, the newly measured times are added to the previous values.

To perform the measurements afresh, clear the results of measurement by selecting [Clear Data] from the popup menu before running the program.

Also, if the [Measure the performance only once] checkbox is checked, the measured time of re-executions is not added to the previous measured time, and measurement results are displayed separately for each execution performed.

## 4.9.5 Clearing Performance-Measurement Conditions

Select the measurement condition you want to clear in the [Performance Analysis] window and then choose [Set] from the popup menu to display the [Performance] dialog box. In the [Performance] dialog box, disable the condition you want to clear.

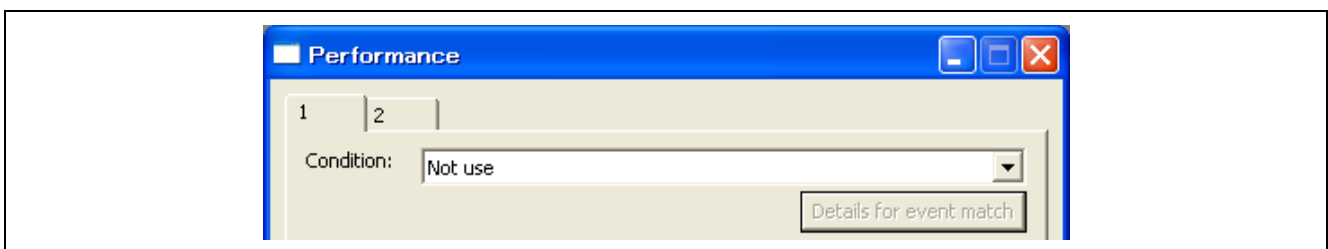


Figure 4.43 Clearing Performance-Measurement Conditions

#### 4.9.6 Clearing Results of Performance Measurement

In the [Performance Analysis] window, select the section corresponding to the results you want to clear and then choose [Clear Data] from the popup menu. The results of measurement for the selected section will be cleared.  
To clear all results of measurement, choose [Clear All Data] from the popup menu.

#### 4.9.7 Maximum Number of Rounds of Performance Measurement

For the RX600 Series MCUs, two 32-bit counters are used for performance measurement. While the 32-bit counters are in use for measurements in two address ranges, the maximum number of rounds for measurement is 4,294,967,295.

When the counters are handled as a 64-bit counter, on the other hand, the maximum number of rounds for measurement is 18,446,744,073,709,551,615.

For the RX200 Series MCUs, a 24-bit counter is used for performance measurement. The maximum number of rounds for measurement is 16,777,215.

If the counters overflow during measurement, "overflow" appears in the [Performance Analysis] window.

To handle the counters as a single 64-bit counter, select the [Use 64-bit counter] checkbox in the [Performance] dialog box.

#### 4.9.8 Saving/Loading Performance-Measurement Settings

##### (a) Saving Performance-Measurement Settings

Click on the [Save] button of the [Performance] dialog box. The [Save As] dialog box will be displayed.

Specify the name of the file where you want the performance-measurement settings to be saved. The file-name extension is ".pev". If this is omitted, the extension ".pev" is automatically appended.

##### (b) Loading Performance-Measurement Settings

Click on the [Load] button of the [Performance] dialog box. The [Open] dialog box will be displayed. Specify the name of the file you want to load.

When you load a file, the previous performance-measurement settings are discarded and the new settings appear in the dialog box.

Click on the [Apply] button of the [Performance] dialog box to activate the new performance-measurement settings you have loaded.

## 4.10 Viewing Memory Data in Real Time

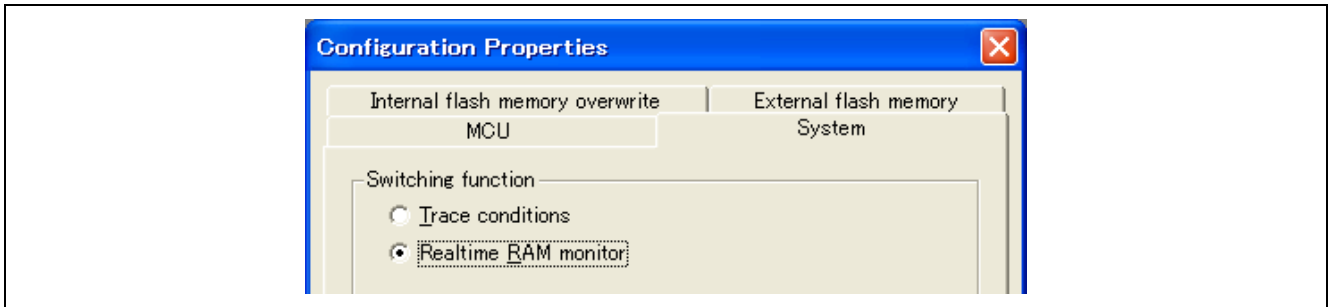
### 4.10.1 RAM Monitoring

Use the [RAM Monitor] window to monitor data in memory while the user program is running.

The RAM monitoring function permits recording and inspection of the data in an area of memory for which monitoring has been assigned and the states of access in real time without obstructing execution of the user program.

The [RAM Monitor] window shows the access states (read, written, data lost, or error detection) in different colors.

RAM monitoring is only available with the E20 because this facility utilizes the external trace-output function. To enable RAM monitoring, select [Realtime RAM monitor] under [Switching function] in the [Configuration Properties] dialog box.



**Figure 4.44** [Configuration Properties] Dialog Box

However, some trace functions are not usable while [Realtime RAM Monitor] is selected. The following table shows the restrictions on the trace functions.

**Table 4.15** Restrictions on Trace Functions

Function	Restriction
Events	Events are only specifiable for [Extract]. Events for [Trace Start] and [Trace Stop] are not selectable.
[Trace Mode]	The selection is fixed to [Fill until stop].
[Trace Type]	The selection is fixed to [Data].
[Trace Capacity]	The selection is fixed to [1 MB].
[Displayed trace information] in the [Advanced Setting] dialog box	[Time stamp], [Stack pointer], and [Channel number] are not selectable.

### 4.10.2 Allocating Blocks for RAM Monitoring

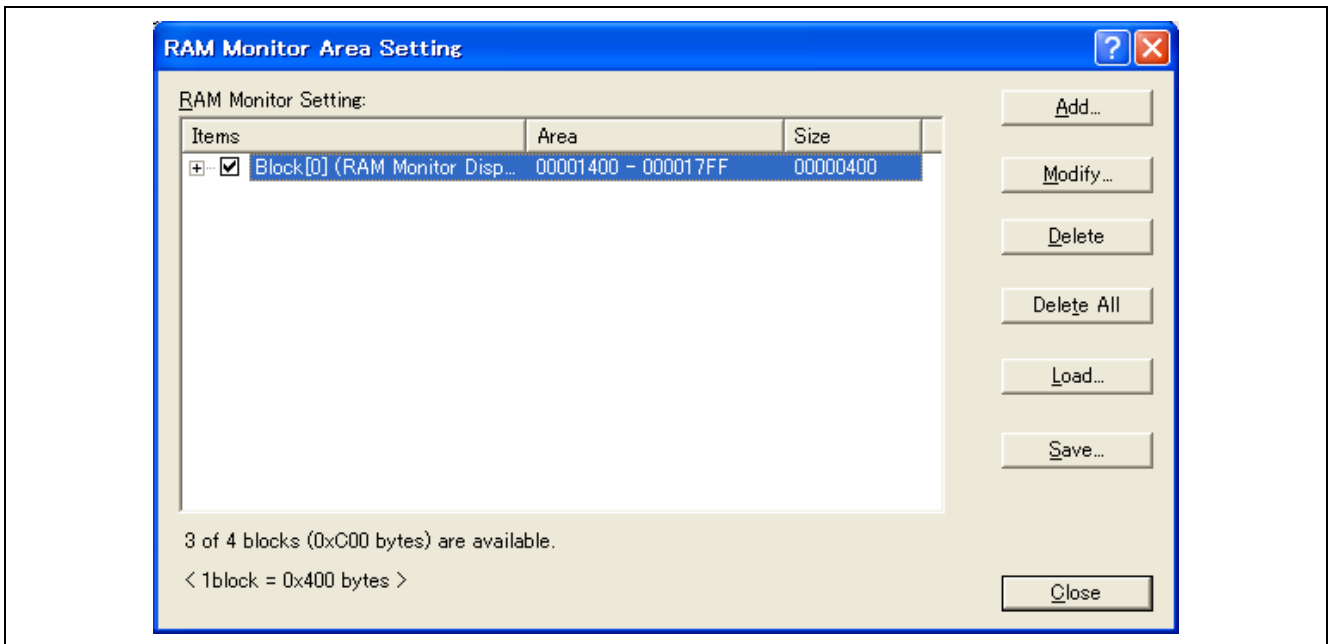
Monitoring of up to 4096 bytes of RAM is possible.

This can be a desired contiguous address range or up to four blocks of 1024 bytes.

During monitoring of the selected blocks of RAM, trace data on all access to memory are output and some trace data may be lost. To minimize the range to be monitored, you can use events to extract trace data for output. For details, refer to section 4.10.5, Preventing Loss of Data.

Blocks for RAM monitoring are allocated via the [RAM Monitor Area Setting] dialog box.

Select [View -> CPU -> RAM Monitor] to open the [RAM Monitor] window. Clicking on the [RAM monitor area setting] toolbar button opens the [RAM Monitor Area Setting] dialog box.



**Figure 4.45** [RAM Monitor Area Setting] Dialog Box

(a) Adding Blocks for RAM Monitoring

Clicking on [Add] opens a dialog box in which you can specify a range of addresses to indicate a block for RAM monitoring.

Each block is 1024 bytes in size. Even if you specify a range that is smaller than 1024 bytes, the emulator allocates the 1024-byte space beginning from the specified address.

If the specified range of addresses is greater than 1024 bytes, on the other hand, the emulator automatically allocates two or more blocks as required.

Note that if this process is executed during user program execution, the realtime capability of the RAM monitor is lost because a memory read cycle occurs.

(b) Modifying the Addresses of Blocks

Select the block that you wish to modify and click on [Modify].

This opens a dialog box in which you can modify the addresses that indicate the block.

Note that if this process is executed during user program execution, the realtime capability of the RAM monitor is lost because a memory read cycle occurs.

(c) Deleting Blocks

Select the block that you wish to delete and click on [Delete].

To delete all blocks listed in the [RAM Monitor Area Setting] dialog box, click on [Delete All].

(d) Saving the Settings of Memory for RAM Monitoring

Clicking on [Save] opens the [Save As] dialog box.

Specify the name of the file where you wish the settings to be saved.

The filename extension is ".rbi", and this is automatically appended even if it was omitted in the dialog box.

(e) Loading the Settings of RAM-Monitoring Blocks

Clicking on [Load] opens the [Open] dialog box.

Specify the name of the file.

When a new file is loaded, the previous settings for RAM monitoring are discarded.

Clicking on [Apply] activates the new settings that have been loaded.



### 4.10.3 Viewing RAM Monitoring Information

To open the [RAM Monitor] window, select [View -> CPU -> RAM Monitor] or click on the [CPU] toolbar button ( ).

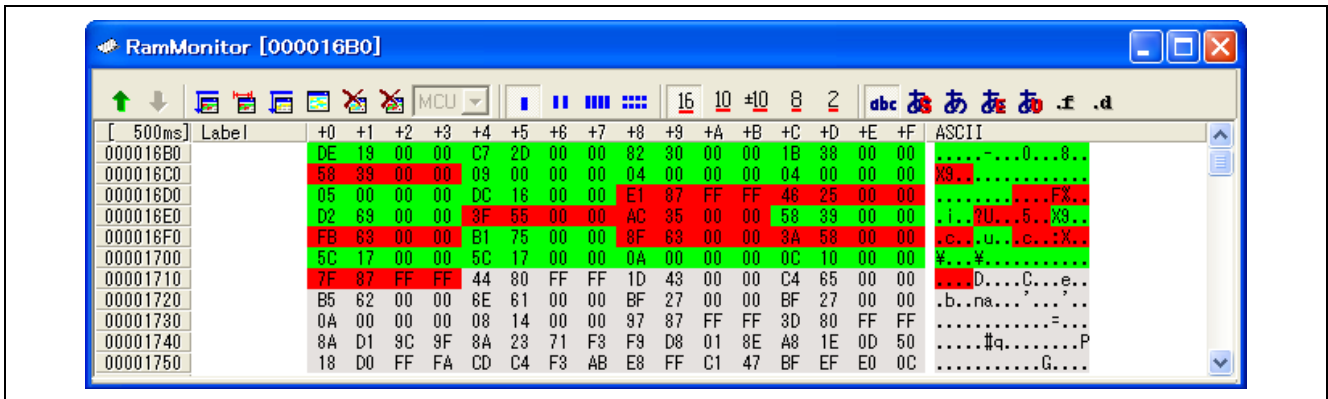


Figure 4.46 [RAM Monitor] Window

Access states are indicated by different background colors according to the access attribute as listed below. The access attributes "read" and "written" indicate the last access to each memory location.

The contents of the [RAM Monitor] window are acquired from bus access by the CPU. Therefore, changes made to memory areas that are not readable or writable by the user program (e.g. by another device reading from and writing to external memory) or by the DTC or DMA are not reflected in the [RAM Monitor] window.

If the data displayed in blocks for RAM monitoring are not in 1-byte units of length, the memory access attribute for the data may differ from byte to byte. If access attributes for a single datum differ in this way, the datum is enclosed in parentheses when displayed in the window. The background color in this case indicates the access attribute for the first byte of that data.

(0001) E4BF 84BA 399F 68AD


The display updating interval may be longer than the specified interval due to operating conditions (those listed below).

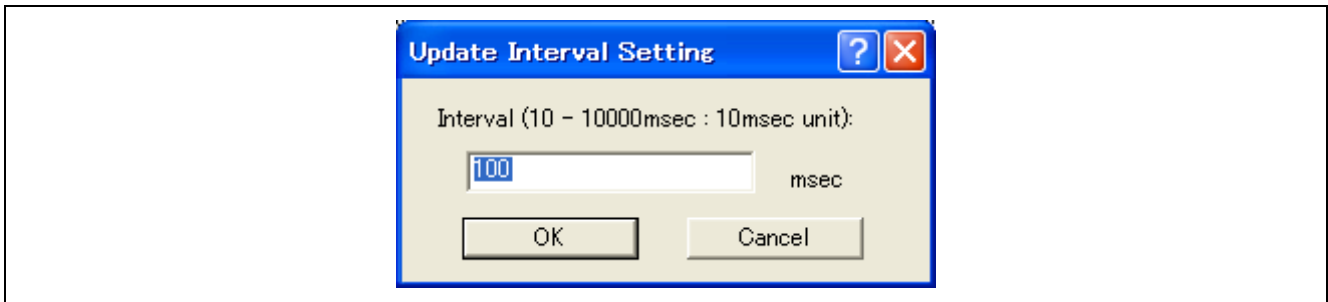
- Performance of and load on the host machine
- Communications interface
- Window size (memory display range) and the number of windows being displayed

Table 4.16 Access Attribute and Background Color

	Access Attribute	Background Color
Read		Green
Written		Red
When error detected	Uninitialized memory (Not-write-accessed area was accessed for read)	Yellow
	Unreferenced memory (Write-accessed area was not accessed for read)	Sky blue
No access		White
No access since some data were lost		Gray

## (a) Setting the Update Interval for RAM Monitoring


Choose [Update Interval Setting] from the popup menu of the [RAM Monitor] window, or click on the tool bar button (  ). The [Update Interval Setting] dialog box shown below will appear.



**Figure 4.47** [Update Interval Setting] Dialog Box

A separate update interval can be specified per [RAM Monitor] window. The initial value is 100 ms.

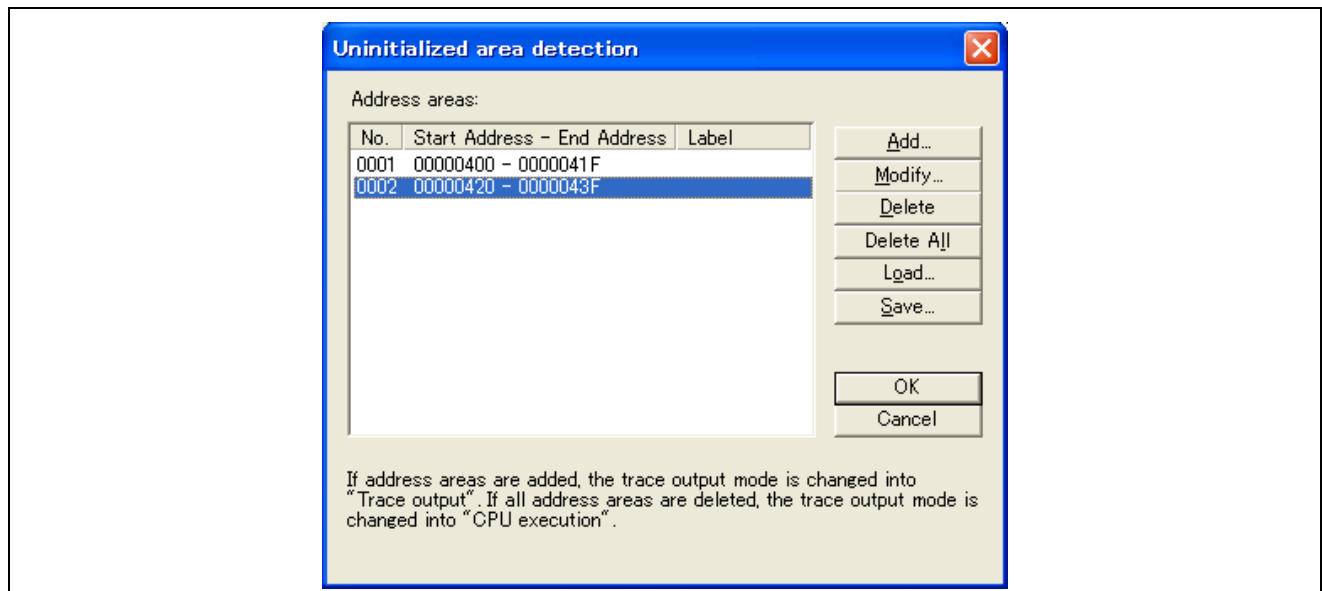
## (b) Setting uninitialized area detection

Selecting [Uninitialization Detection Setting...] from the popup menu of the [RAM monitor] window or clicking on the toolbar button (  ) brings up the [Uninitialized area detection] dialog box.

If a memory area registered for uninitialized area detection in this dialog box is accessed for read while it has not been write-accessed, the area is determined to be “uninitialized” and highlighted in yellow for error output. Also, if a registered memory area is not accessed for read even once after it has been write-accessed, the area is determined to be “unreferenced” and highlighted in sky blue for error output.

The memory areas registered for uninitialized area detection are also displayed as “Uninitialization” in the registered list of the [RAM Monitor Area Setting] dialog box.

Note that no areas can be set for uninitialized area detection while the user program is under execution.



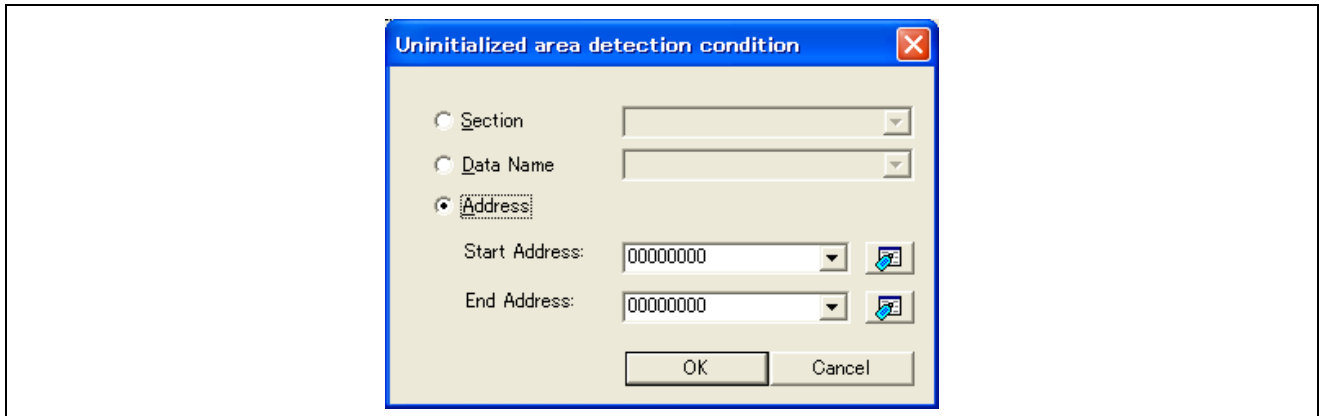
**Figure 4.48** [Uninitialized area detection] Dialog Box

- Adding an error detection area

To register a new error detection area, click the [Add...] button.

The [Uninitialized area detection condition] dialog box is displayed.

Register an area for which you want errors detected by specifying a section name, data name or an address range.



**Figure 4.49** [Uninitialized area detection condition] Dialog Box

- Altering an error detection area

Select a registered error detection area and click on the [Modify...] button.

The [Uninitialized area detection condition] dialog box appears.

- Deleting an error detection area

Select a registered error detection area and click on the [Delete] button.

To delete all of the registered error detection areas, click the [Delete All] button.

- Loading the set content of an error detection area

Click on the [Load...] button of the [Uninitialized area detection] dialog box.

The [Open] dialog box appears.

Specify a file name that you want to load.

When this file is loaded, previous settings of an error detection area are destroyed and the area has its settings reset with the loaded content.


- Saving the set content of an error detection area

Click the [Save...] button of the [Uninitialized area detection] dialog box.

The [Save as] dialog box appears.

Specify a file name to which you want to save. The file name extension is “.uni.” If it is omitted, the extension “.uni” is added.

## (c) Displaying Error Detection

Select [Error Detection Display] from the popup menu of the [RamMonitor] window or click the toolbar button (  ). The detection ranges that you registered in the [Uninitialized area detection] dialog box are displayed in extracted form in the [Ram Monitor] window. At this time, the access types, read or write, are not displayed.

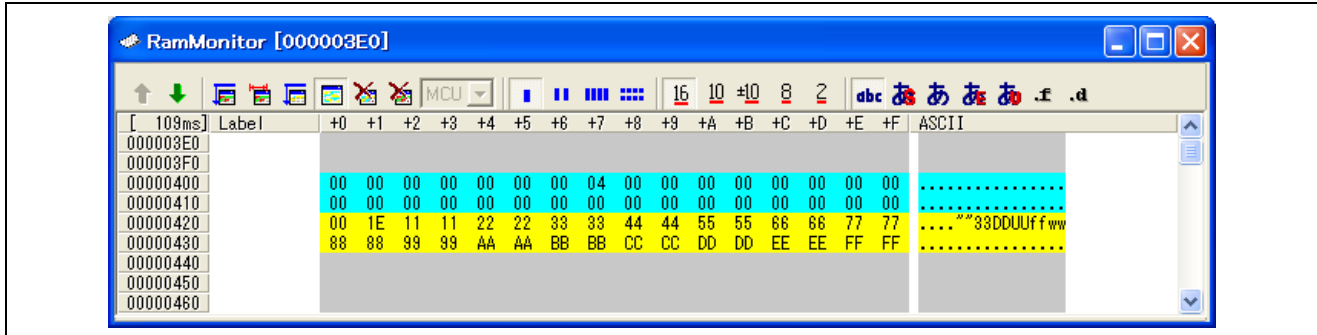




Figure 4.50 [Ram Monitor] Window (Error Detection Display)

## (d) Clearing Error Detection Information

Selecting [Error Detection Clear] from the popup menu of the [RamMonitor] window or clicking the toolbar button (  ) clears information on all of the detected uninitialized memory and unreferenced memory in the RAM monitor area.

If this process is executed during user program execution, the realtime capability of the RAM monitor is lost because a memory read cycle occurs.

## (e) Clearing RAM monitoring access history

Choose [Access Data Clear] from the popup menu of the [RAM Monitor] window or click on the tool bar button (  ). The history of all access to the RAM monitoring area will be cleared.

If this process is executed during user program execution, the realtime capability of the RAM monitor is lost because a memory read cycle occurs.

#### 4.10.4 When Some Trace Data are Lost

If some trace data are lost, the information shown in the [RAM Monitor] window does not match the actual data in memory. The emulator cannot identify which part is missing.

When data are missing, the whole area that is displayable in the window is grayed-out (this indicates that some data have been lost). Display subsequently resumes when data are again output without loss.

If data loss occurs within a period that is shorter than the interval for updating of the [RAM Monitor] window, the window may constantly be grayed out. In this case, refer to section 4.10.5, Preventing Loss of Data.

### 4.10.5 Preventing Loss of Data

The two ways of preventing loss of data accessed during RAM monitoring are described below.

#### (a) Trace-Output Priority Mode

Select [View -> Event -> Trace Conditions] to open the [Trace conditions] dialog box and select [Trace output] (trace-output priority mode) for [Trace Output]. None of the data will be lost during RAM monitoring, but operation in the [Trace output] mode is not necessarily realtime because the emulator gives priority to the output of trace data and thus may temporarily suspend the CPU.

#### (b) Extraction of Trace Data

With the settings made in section 4.10.2, Allocating Blocks for RAM Monitoring, trace data on all access to memory are output. For this reason, when data from any location are lost (even if the location is not included in a block specified for RAM monitoring), the [RAM Monitor] window will indicate the loss.

To reduce the possibility of data loss, use events to extract trace data for output because this can minimize the range to be monitored. Note that a range of data that does not fit in the monitor range of the [RAM Monitor] window is displayed in gray.

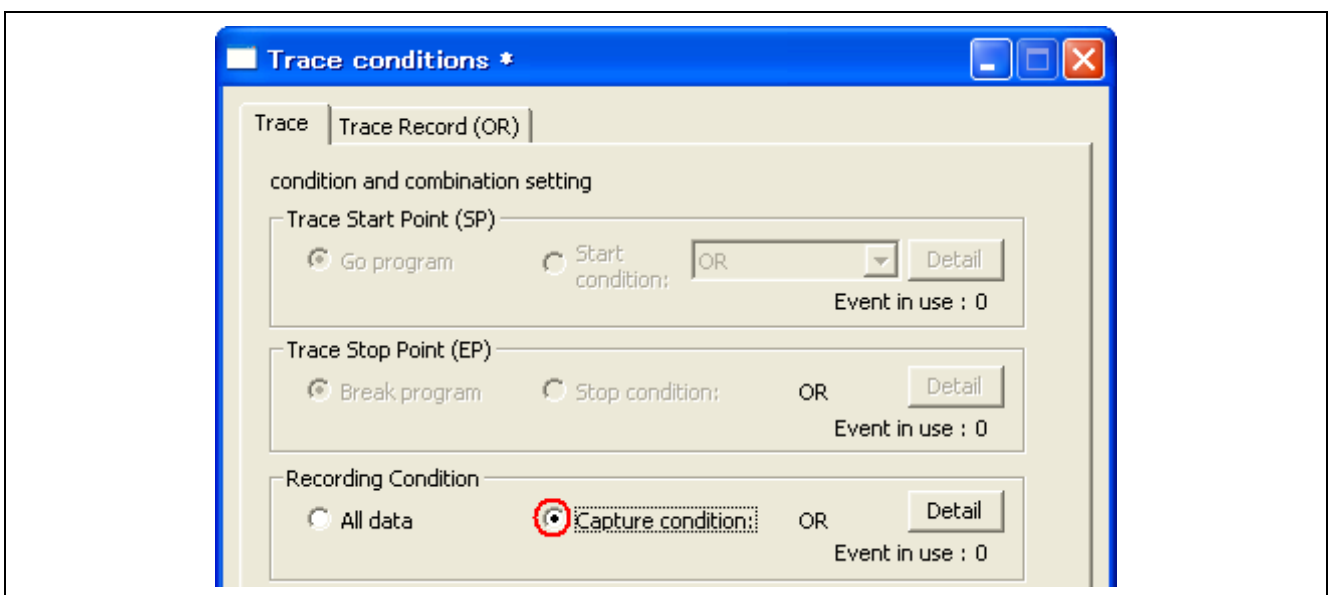
While the realtime characteristic of operation is not affected, the range of RAM monitoring is smaller. No further measures for prevention are available, however, if data access to the specified range is still so frequent that data are being lost.

Conditions for the extraction of trace data can be set in the [Trace conditions] dialog box that is opened by selecting [View -> Event -> Trace Conditions]. An example is given on the following page.

Example:

To monitor data in the range from 0x1700 to 0x171F, add the range from 0x1400 to 0x17FF as a block for RAM monitoring by the method described in section 4.10.2, Allocating Blocks for RAM Monitoring. Then specify the extraction of trace data so that only trace data for the range from 0x1700 to 0x171F will be output.

(1) Select the [Capture condition] radio button in the [Trace conditions] dialog box.



**Figure 4.51** Extraction of Trace Data

- Open the [Trace Record (OR)] tabbed page and click on the [Add...] button. The [Event] dialog box appears. Make the setting shown below and click on the [OK] button, then click on the [Apply] button to activate the new setting.

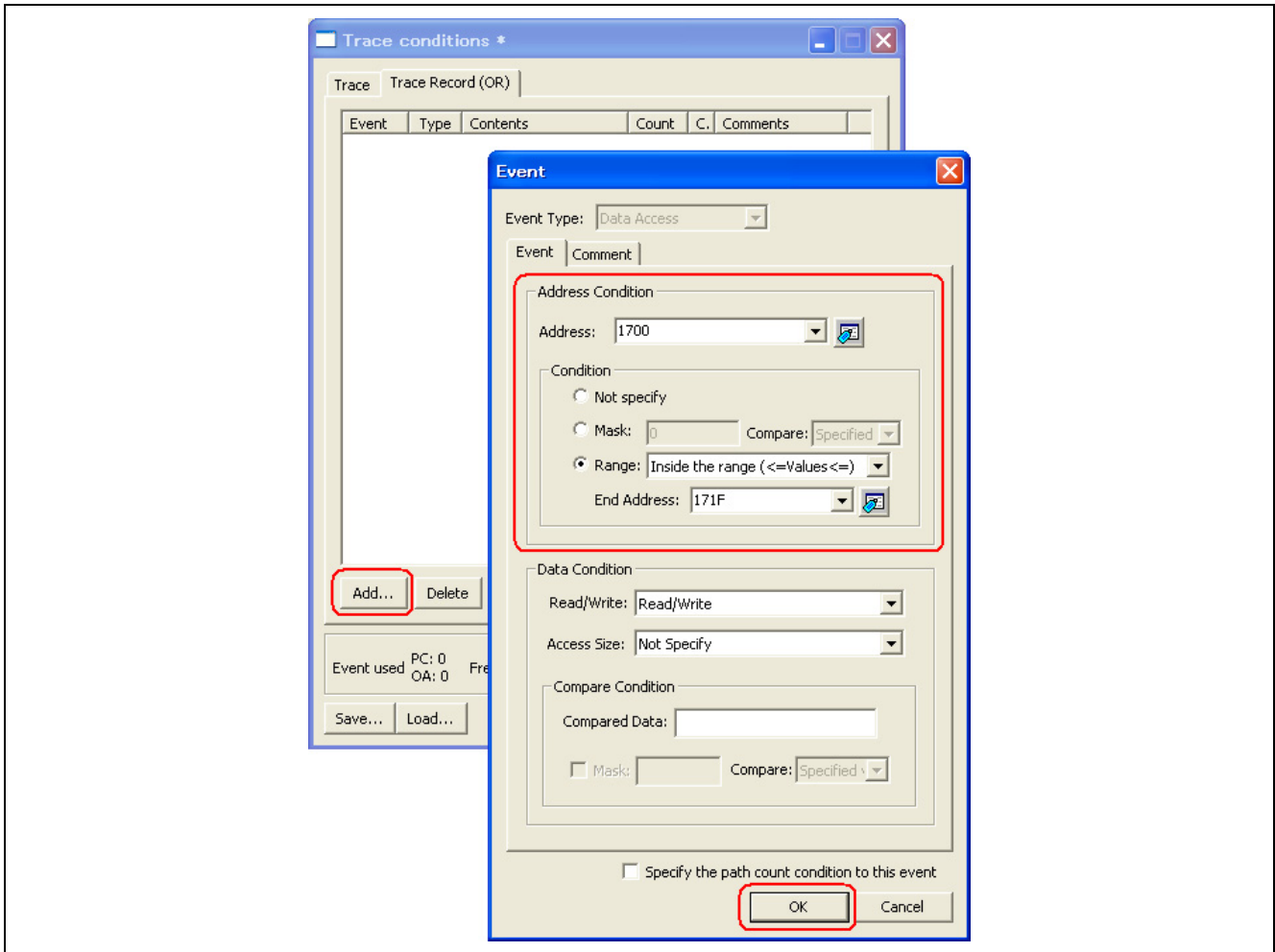


Figure 4.52 Setting an Address Condition

- The trace data in the monitor range (addresses 0x1700 to 0x171F) specified in trace extraction conditions is displayed in the [RAM Monitor] window.

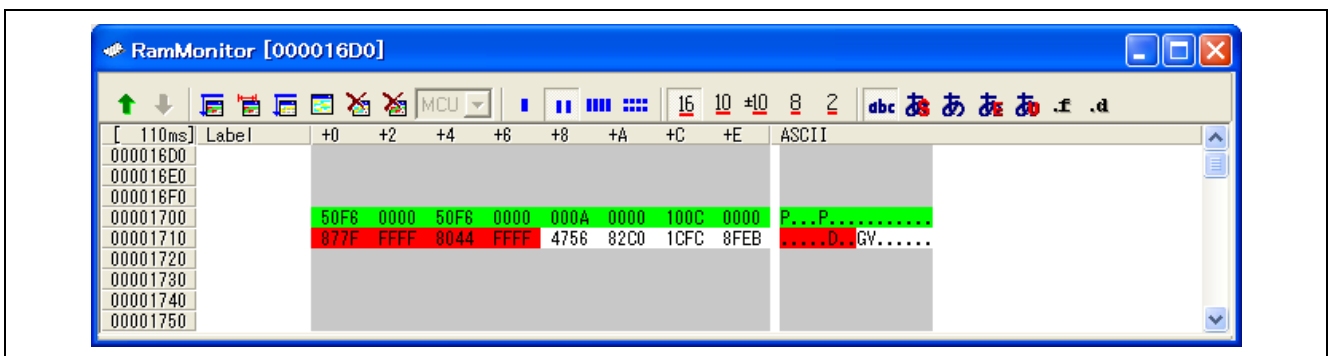


Figure 4.53 [RAM Monitor] Window (Monitor Range Specified)

## 4.11 Using the Start/Stop Function

The emulator can be made to execute specific routines of the user program immediately before starting and immediately after halting program execution. This function is useful if you wish to control a user system in synchronization with starting and stopping of user program execution.

### 4.11.1 Opening the [Start/Stop function setting] Dialog Box

The routines to be executed immediately before starting and after halting execution of the user program are specified in the [Start/Stop function setting] dialog box.

To open the [Start/Stop function setting] dialog box, choose [Setup -> Emulator -> Start/Stop function setting...] from the menu.

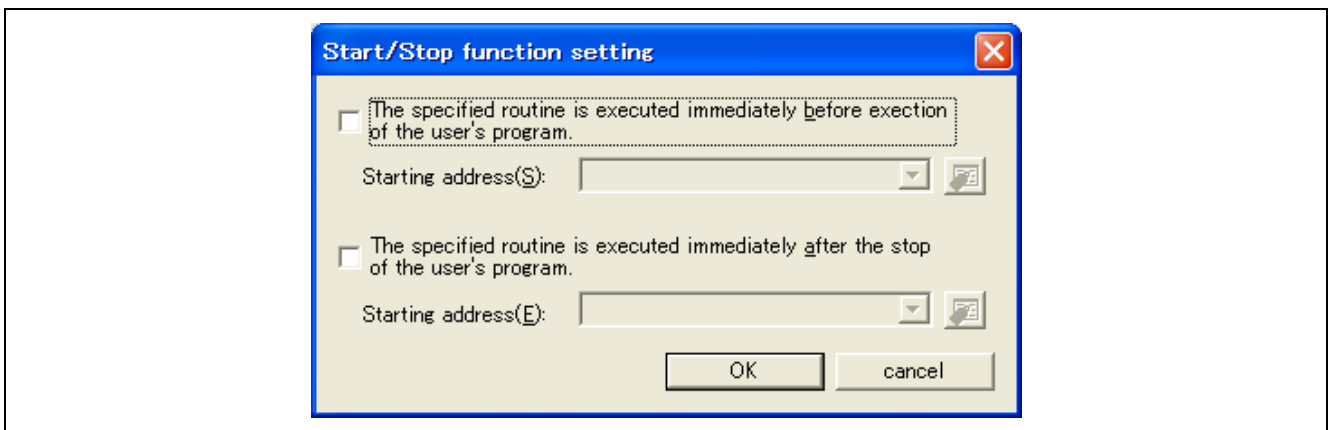


Figure 4.54 [Start/Stop function setting] Dialog Box

### 4.11.2 Specifying the Routine to be Executed

The routines to run immediately before starting and after halting execution of the user program are specified separately. When the [The specified routine is executed immediately before execution of the user's program] checkbox is selected, the routine specified in the [Starting address] combo box, which is below the checkbox, is executed immediately before execution of the user program starts.

When the [The specified routine is executed immediately after the stop of the user's program] checkbox is selected, the routine specified in the [Starting address] combo box, which is below the checkbox, is executed immediately after execution of the user program stops.

### 4.11.3 Restrictions on the Start/Stop Function

The start/stop function is subject to the following restrictions.

- The debugging functions listed below are not to be used while the start/stop function is in use.
  - (a) Memory setting and downloading into the program area of a routine specified as a start/stop function.
  - (b) Breakpoint setting in the program area of a specified routine
- While either of the specified routines is running, the four bytes of memory indicated by the interrupt stack pointer are in use by the emulator.
- In the specified routines, the following restrictions apply to registers and flags.

**Table 4.17 Restrictions on Registers and Flags**

Register and Flag Names	Restrictions
ISP register	When execution of a specified routine is ended, the register must be returned to its value at the time the routine started.
Flag U	While a specified routine is running, switching to user mode is prohibited.
Flag I	No interrupts are allowed during execution of a specified routine.
Flag PM	While a specified routine is running, switching to user mode is prohibited.

- When either of the specified routines is running, the debugging functions listed below have no effect.
  - (a) Tracing
  - (b) Break-related facilities
  - (c) RAM monitoring
  - (d) Performance analysis
  - (e) Setting events within the specified routines
- While either of the specified routines is running, non-maskable interrupts are always disabled.
- The table below shows which state the MCU will be in when the user program starts running after execution of a routine specified as a start function.

**Table 4.18 MCU Status at Start of the User Program**

MCU Resource	Status
MCU general-purpose registers	These registers are in the same state as when the user program last stopped or in states determined by user settings in the [Register] window. Changes made to the contents of registers by the specified routine are not reflected.
Memory in MCU space	Access to memory after execution of the specified routine is reflected.
MCU peripheral functions	Operation of the MCU peripheral functions after execution of the specified routine is continued.

#### 4.11.4 Limitations on Statements within Specified Routines

Statements within specified routines are subject to the limitations described below.

- If a specified routine uses a stack, the stack must always be the interrupt stack.
- The processing of a specified routine must end with a return-from-subroutine instruction.
- Ensure that a round of processing by a specified routine is complete within 100 ms. If, for example, the clock is turned off and left inactive within a specified routine, the emulator may become unable to control program execution.
- The values stored in the registers at the time a specified routine starts running are undefined. Ensure that each specified routine initializes the register values.
- The emulator executes the specified routines in supervisor mode. Do not switch the mode to user mode.
- For JTAG interface, the emulator executes the user program within approximately 20 ms after the start function. (CPU clock: 100 MHz, JTAG clock: 16.5 MHz)
- For JTAG interface, the emulator executes the stop function no sooner than 20 ms after the end of execution of the user program. (CPU clock: 100 MHz, JTAG clock: 16.5 MHz)
- For FINE interface, the emulator executes the user program within approximately 30 ms after the start function. (CPU clock: 100 MHz, FINE Baud Rate: 2000000 bps)
- For FINE interface, the emulator executes the stop function no sooner than 40 ms after the end of execution of the user program. (CPU clock: 100 MHz, FINE Baud Rate: 2000000 bps)




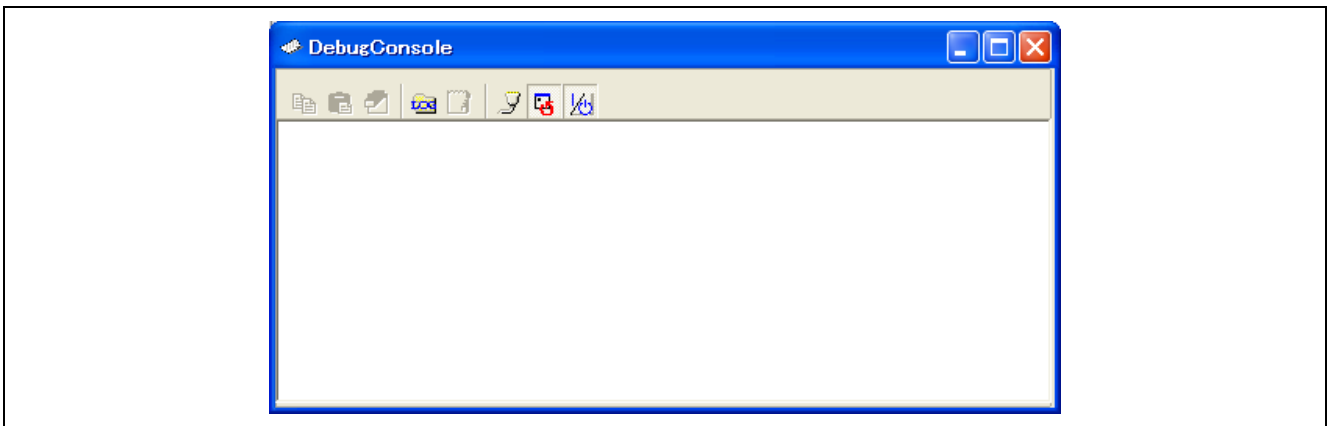
## 4.12 Using the Debug Console

The debug console supports standard input and output by the user program. Standard input and output through the [DebugConsole] window is obtained by replacing low-level interface routines (`_charput` and `_charget`) in the user program.

Note: When you use the input/output function, be sure to leave the [DebugConsole] window open.

### 4.12.1 Opening the [DebugConsole] Window

To open the [DebugConsole] window, select [View -> CPU -> DebugConsole] or click on the [DebugConsole] button .



**Figure 4.55** [Debug Console] Window

Standard output from the user program is displayed in this window. Keyboard input in the window becomes standard input to the user program.

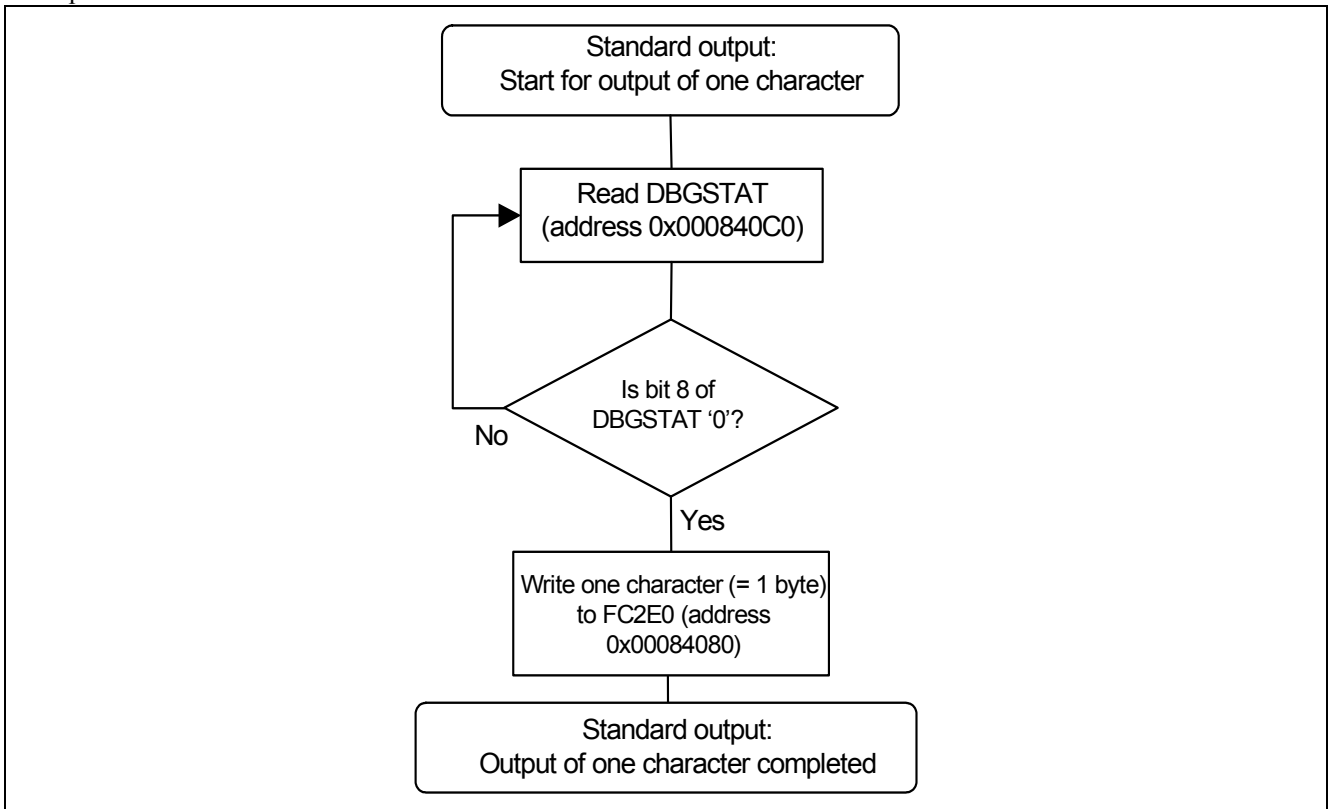
### 4.12.2 Low-Level Interface Routines

If you wish to use standard input and output in a C/C++ program, you need to create a low-level interface routine. To utilize the debug console, functions `charput` and `charget` (or `_charput` and `_charget`), which respectively handle the output and input of a single character and are called from within a low-level interface routine, must be replaced with special code intended for the debug console.

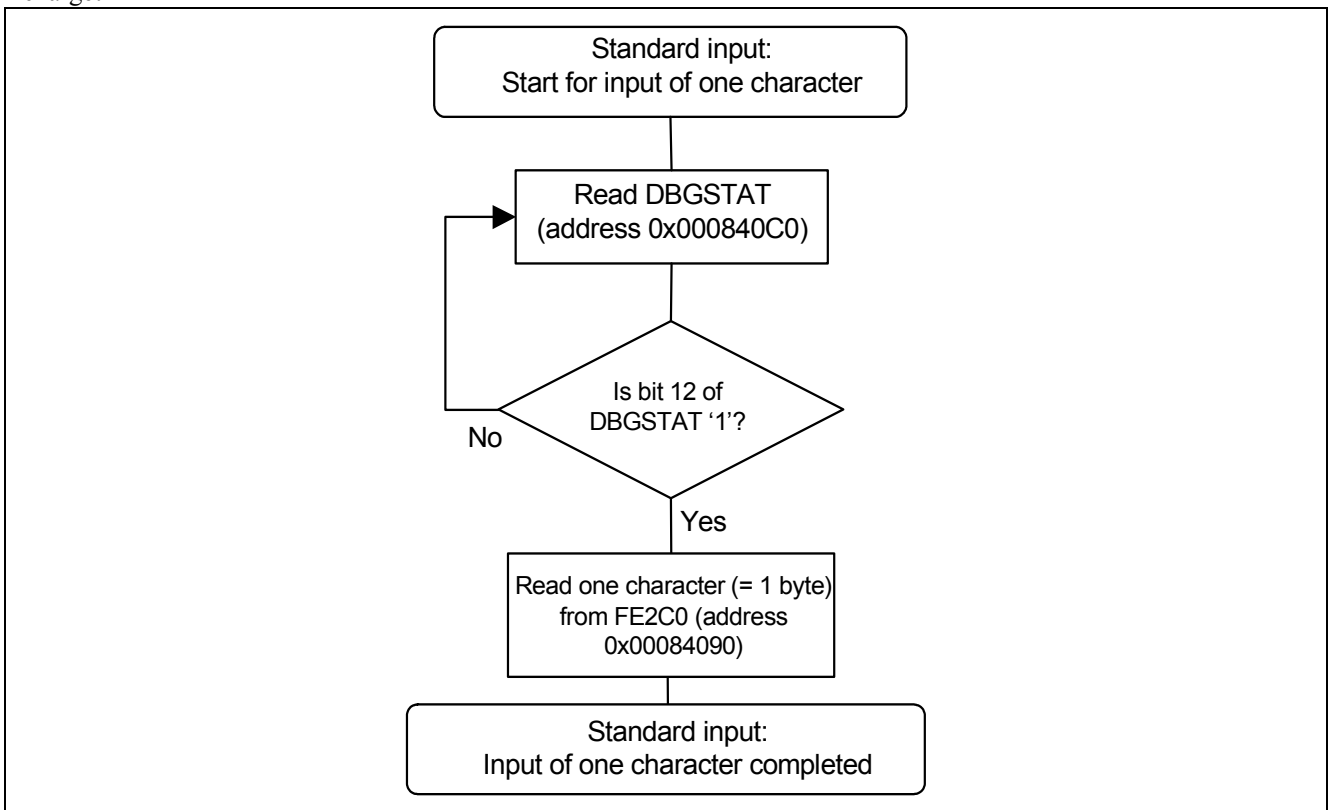
If you wish to modify the respective functions, they must handle output to and input from a specific address. The flow of processing and a sample program are given on the following pages.

Note: Any program that implements the given flow of processing, including the sample program, is specifically intended for the debug console provided by the emulator. User programs must not include such code unless the emulator is connected to the user system.

charput



charget



Sample Program for the Debug Console

```

;-----
;
; FILE      :lowlvl.src
; DATE      :Wed, Jul 01, 2009
; DESCRIPTION :Program of Low level
; CPU TYPE  :RX
;
;-----
                .GLB      _charput
                .GLB      _charget

FC2E0      .EQU 00084080h
FE2C0      .EQU 00084090h
DBGSTAT    .EQU 000840C0h
RXFLOEN    .EQU 00001000h
TXFLOEN    .EQU 00000100h
.SECTION P, CODE
;-----
;  _charput:
;-----
_charput:
.STACK      _charput = 00000000h
__C2ESTART: MOV.L  #TXFLOEN, R3
            MOV.L  #DBGSTAT, R4
__TXLOOP:   MOV.L  [R4], R5
            AND   R3, R5
            BNZ   __TXLOOP
__WRITEFC2E0: MOV.L  #FC2E0, R2
            MOV.L  R1, [R2]
__CHARPUTEXIT: RTS
;-----
;  _charget:
;-----
_charget:
                .STACK  _charget = 00000000h
__E2CSTART:  MOV.L  #RXFLOEN, R3
            MOV.L  #DBGSTAT, R4
__RXLOOP:    MOV.L  [R4], R5
            AND   R3, R5
            BZ    __RXLOOP
__READFE2C0: MOV.L  #FE2C0, R2
            MOV.L  [R2], R1
__CHARGETEXIT: RTS
;-----
.END

```

### 4.13 Hot Plug-in Function

Hot plug-in permits the E1/E20 emulator to be connected to the user program while it is being executed, allowing you to debug the program under execution. To use hot plug-in with the E1 emulator, however, you need to connect the separately available "hot-plug adapter" (R0E000010ACB00) between the E1 emulator and the user system. The current status of which emulator product supports the hot plug-in function is shown in the table below.

**Table 4.19 Emulator Products and Support for Hot Plug-in Function**

Product name (Product type name)	Use of the hot plug-in function
E1 (R0E000010KCE00)	Unusable.
E1 (R0E000010KCE00) + hot-plug adapter (R0E000010ACB00)	Usable.
E20 (R0E000200KCT00)	Usable.

#### 4.13.1 Startup Procedure

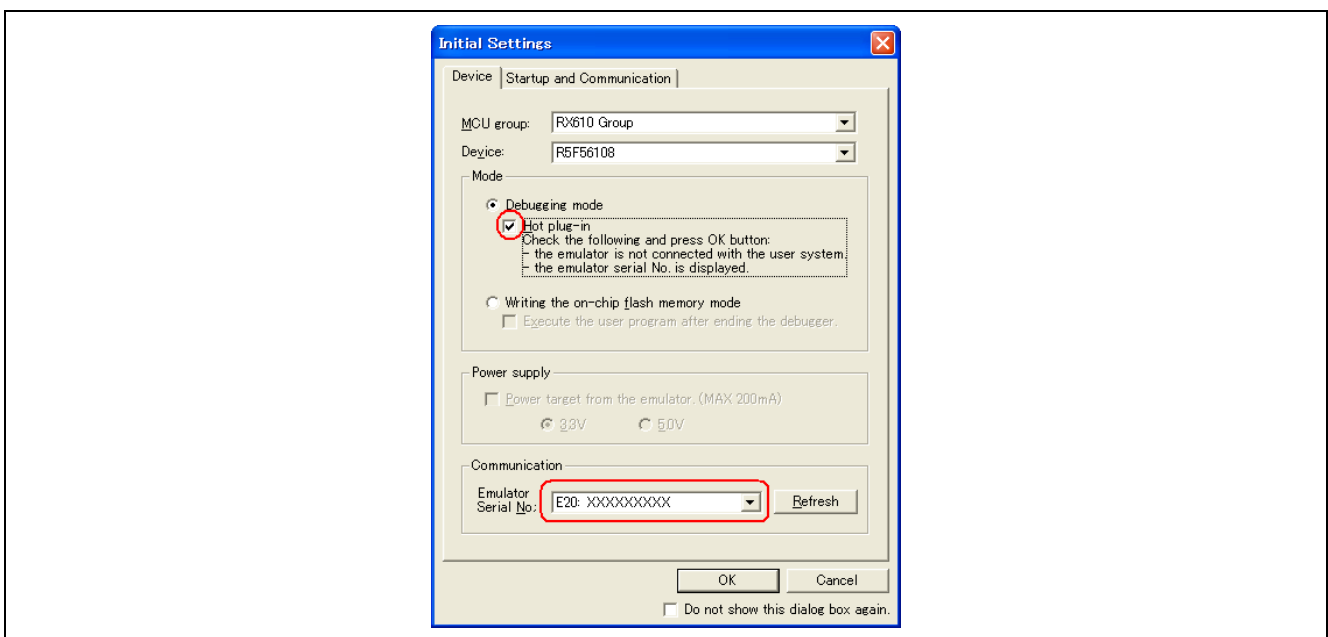
Here is the startup procedure to be followed when using hot plug-in. Do not connect the user system and the emulator until you are instructed to do so.

- (1) While the user system and the emulator are not connected yet, choose [Programs -> Renesas -> High-performance Embedded Workshop -> High-performance Embedded Workshop] from the Start menu and then select the workspace you're using.
- (2) The [Initial Settings] dialog box is displayed. Select [Debugging mode] for [Mode] and check the [Hot plug-in] checkbox.

After confirming that the user system and the emulator are not connected yet and that [Emulator Serial No.] for [Communication] is displayed, click the [OK] button.

Note that the [Hot plug-in] checkbox you've checked has no effect the next time you start.

For other settings, refer to section 3.3.1, [Initial Settings] Dialog Box.



**Figure 4.56 [Initial Settings] Dialog Box when Using Hot Plug-in (Device) Page)**

Note: The contents displayed in the [Initial Settings] dialog box may differ with each MCU used.

(3) The [Connect] dialog box is displayed. Connect the user system and the emulator following the procedure described in the user’s manual of your emulator. Note that if the firmware needs to be rewritten, a rewrite is executed before this dialog box is displayed. For details about firmware rewrite, refer to paragraph 5. of section 2.10, [Confirm Firmware] dialog box.

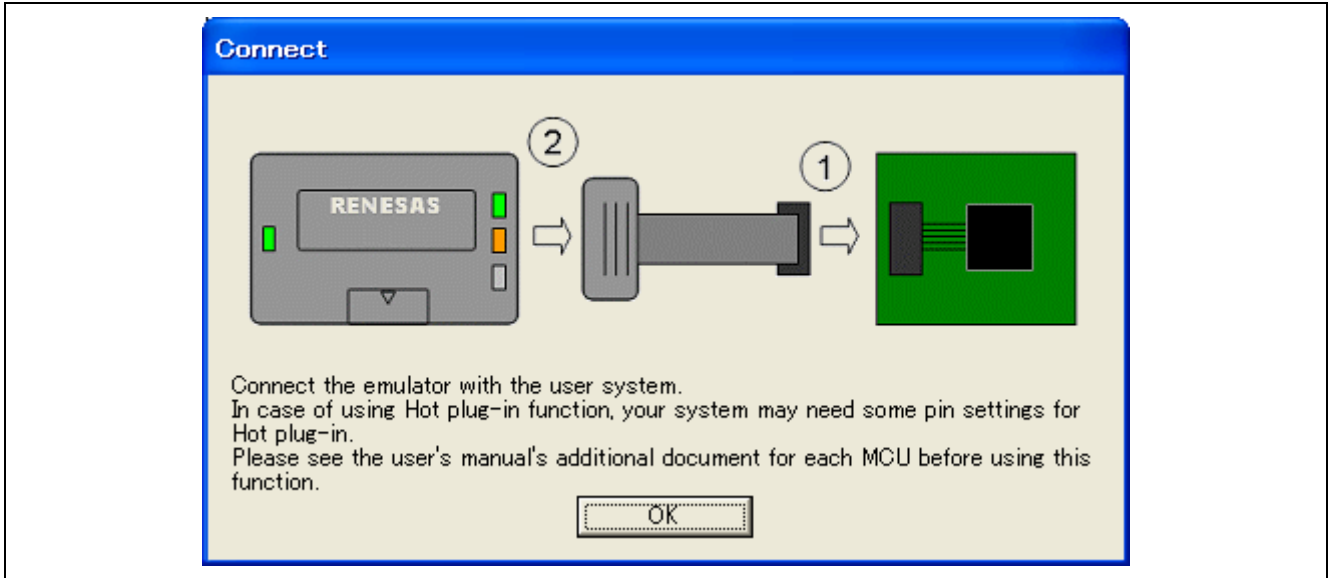


Figure 4.57 [Connect] Dialog Box (E20 Emulator)

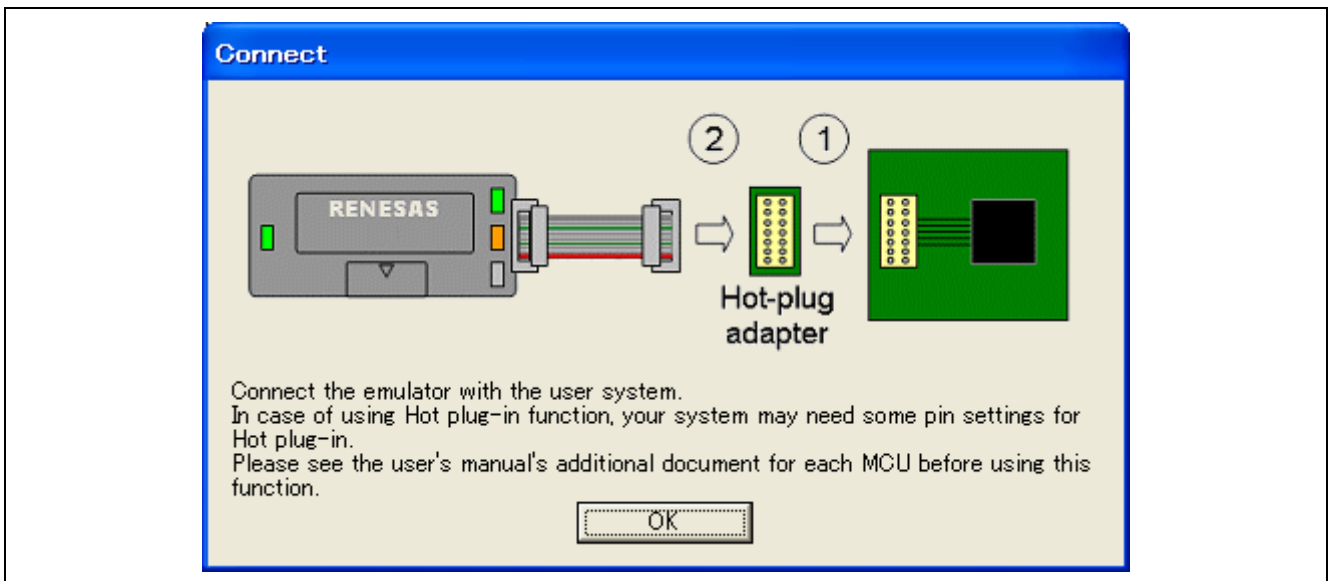
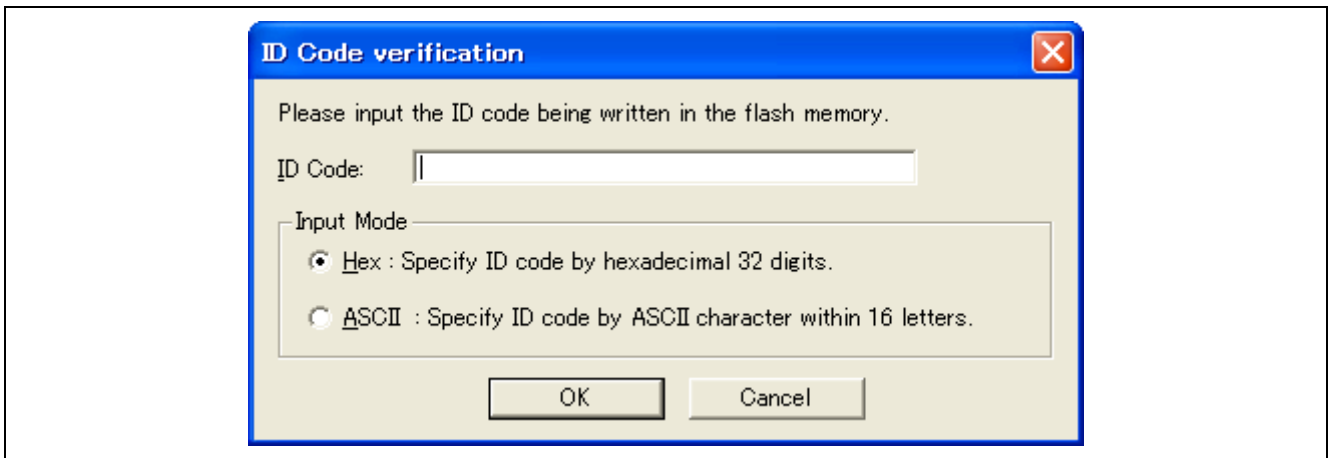


Figure 4.58 [Connect] Dialog Box (E1 Emulator)

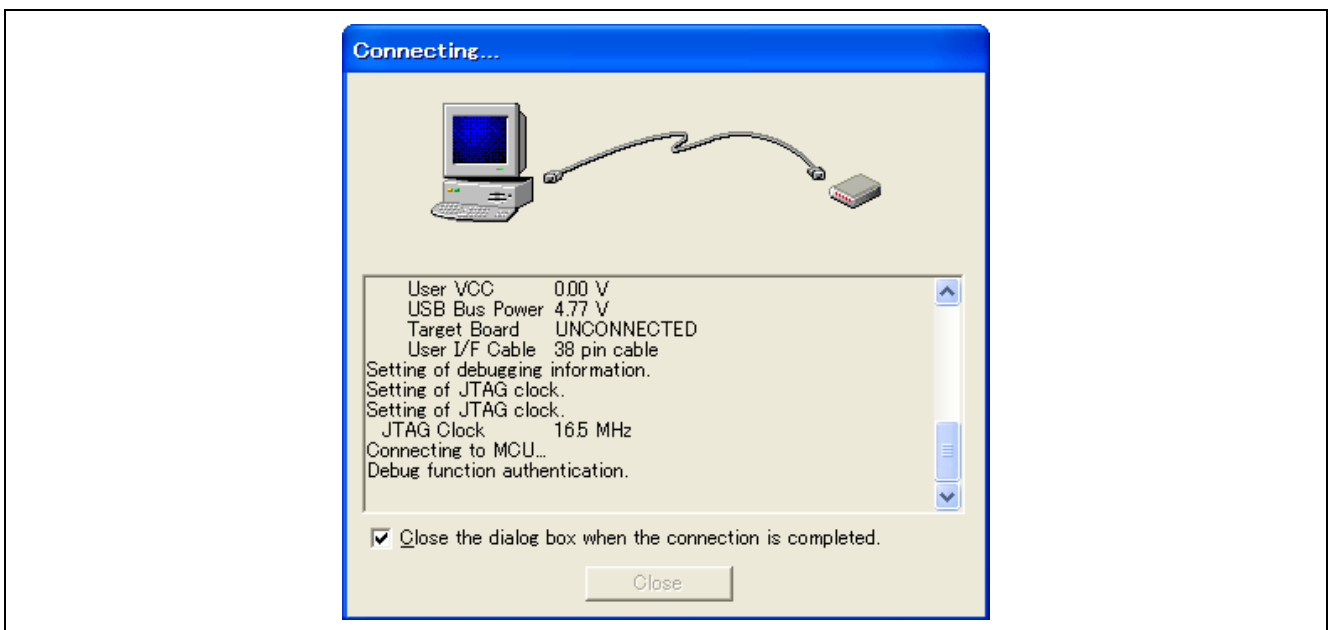
- (4) The [ID Code verification] dialog box is displayed. For the sake of security of the target MCU's internal flash memory, supply the ID code you've set\*<sup>1</sup>. If you did not set ID code, enter "0xFF" for a specified number of digits according to the input mode (Hex or ASCII).



**Figure 4.59** [ID Code verification] Dialog Box when Using Hot Plug-in

Note: 1. For information on ID code, refer to the hardware manual of your MCU.

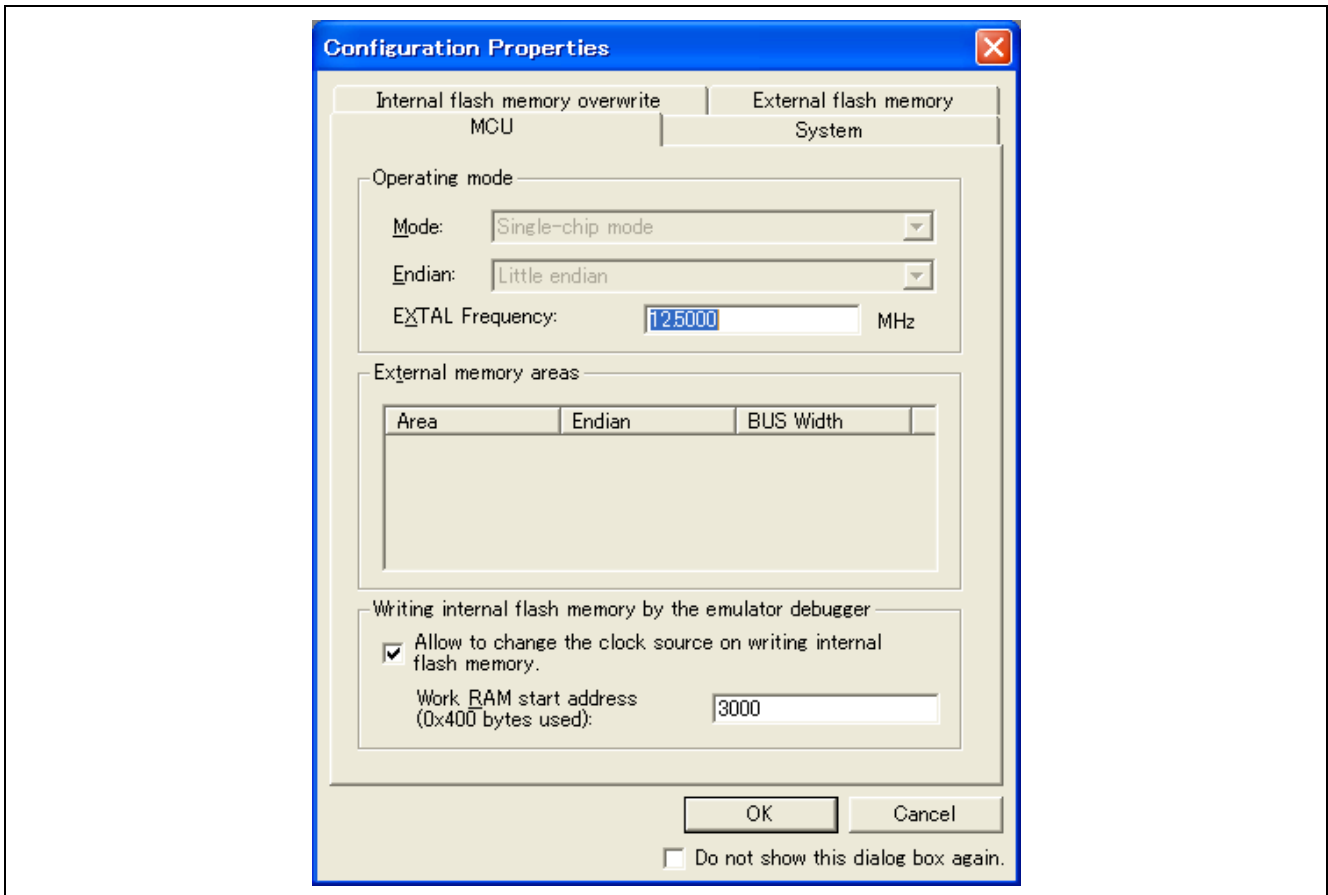
- (5) The [Connecting...] dialog box is displayed.



**Figure 4.60** [Connecting...] Dialog Box when Using Hot Plug-in

(6) The [Configuration Properties] dialog box is displayed. Make the necessary settings for the emulator and debug function, then click the [OK] button.

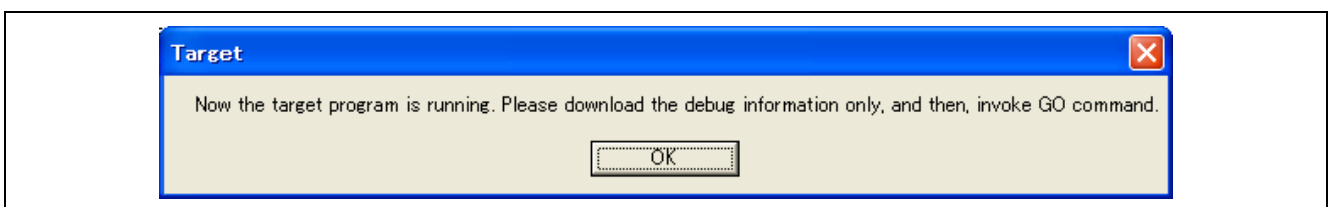
For details about these settings, refer to section 3.3.2, [Configuration Properties] Dialog Box.



**Figure 4.61** [Configuration Properties] Dialog Box when Using Hot Plug-in ([MCU] Page)

Note: The contents displayed in the [Configuration Properties] dialog box may differ with each MCU used.

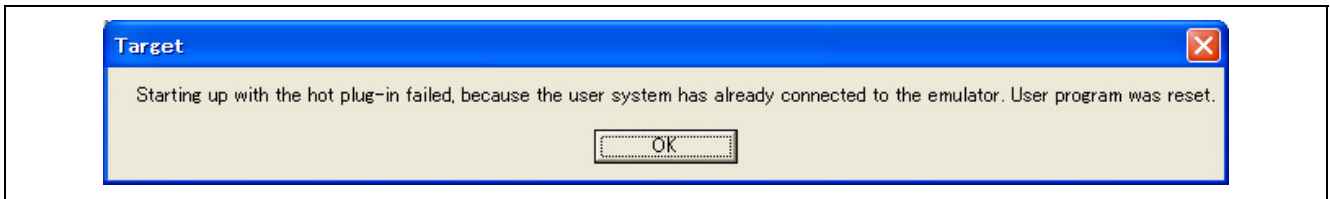
(7) You'll see that "Connected" is displayed in the [Output] window of High-performance Embedded Workshop and the message box shown below is displayed. Click the [OK] button to download only the debug information of the download module file and then click the [Go] button.



**Figure 4.62** Message Box Displayed at Completion of Hot Plug-in Connection

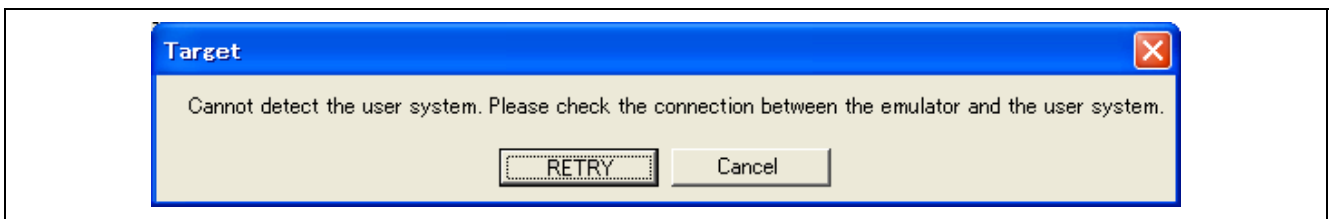
Notes:

1. If you check the [Hot plug-in] checkbox and click the [OK] button in the [Initial Settings] dialog box while the emulator and the user system are connected, the following error message is displayed.



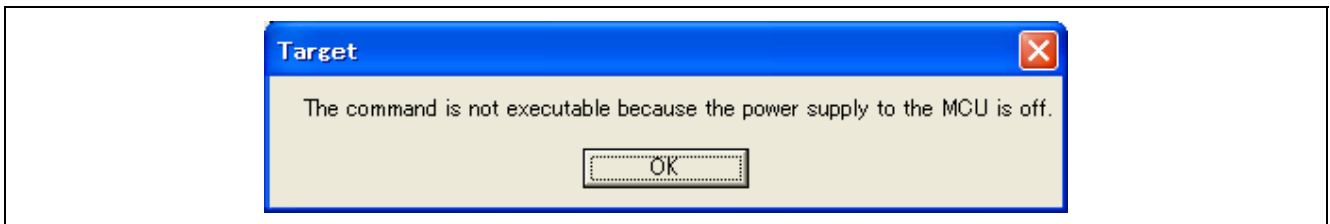
**Figure 4.63** [User System Connection Error] Dialog Box

2. If you click the [OK] button in the [Connect] dialog box while the emulator and the user system are not connected, the following error message is displayed.



**Figure 4.64** [User System Connection Error] Dialog Box

3. If the power to the user system is no supplied from an external source, the following error message is displayed.



**Figure 4.65** [Command Processing Error] Dialog Box



#### 4.13.2 Precautions to Take when Using Hot Plug-in


There are following precautions to take when using hot plug-in.

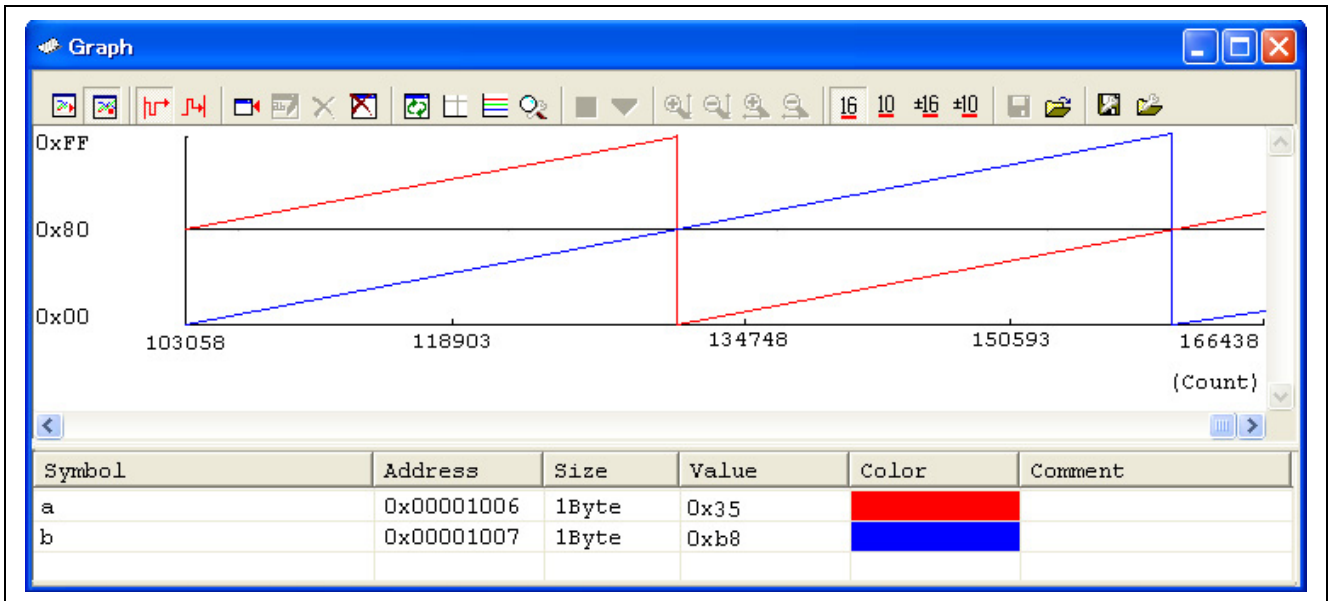
- Until you click the [Go] button after the completion of a hot plug-in connection, do not change event settings, set breakpoints, or use other debug functions. Also, be aware that the S/W breakpoints saved in session information are not set in a hot plug-in connection.
  - Until you click the [Go] button after the completion of a hot plug-in connection, do not perform a memory comparison in the [Command Line] or [Memory] window. If a memory comparison is performed in either window, correct memory comparison results will not be displayed.
  - From when you first click the [Go] button after the completion of a hot plug-in connection till when the program stops, you cannot use the trace functions.
  - Even when you use the step-in or step-over command to run the program after the completion of a hot plug-in connection, the program behaves in the same way as it would when you pressed the [Go] button.
  - Immediately after you started with hot plug-in, what is displayed in the [Register] window and the actual register values differ.
  - When you start with hot plug-in, you cannot use the realtime RAM monitor. (It is grayed out in the [Configuration Properties] dialog box.)
  - The emulator stops the user program temporarily in approximately 800 $\mu$ s to check the ID code at hot plug-in (CPU clock: 100 MHz, JTAG clock: 16.5 MHz).
  - Hot plug-in activation via FINE interface is not supported.
  - For hot plug-ins using the RX630, RX631, RX63N or RX63T Group MCU, always make sure that the endian setting value of the endian select register (MDEB, MDES) written in the MCU and the endian you set in the [Initial Settings] dialog box are matched. For details about the endian select registers (MDEB, MDES), refer to the hardware manual for the MCU you're using.
- Hot plug-ins cannot be used while a flash rewrite program is running.

## 4.14 Graph Functions

To view the changes in memory contents in a graph, use the [Graph] window.  
The [Graph] window shows changes in the values of registered symbols graphically.



### 4.14.1 Displaying the [Graph] window

To open the [Graph] window, select [View -> Graphic -> Graph] or click the [Graph] toolbar button .



**Figure 4.66 [Graph] Window**

#### (1) Selecting the Mode

There are two modes in the [Graph] window: Sampling Mode and Trace Mode. To switch between the mode, right-click in the upper pane of the [Graph] window and choose [Mode -> Sampling], or [Mode -> Trace] from the pop-up menu. Alternatively, you can click on the toolbar buttons   in the [Graph] window.

In the Sampling Mode, values are measured and displayed each time the set sampling period elapses during the execution of the program. In the Trace Mode, information of the trace memory is analyzed and displayed.

The Trace mode has two modes: Full and Free. In Full Mode, the information of the trace memory is analyzed and displayed when the trace buffer is filled or the program is stopped. In Free Mode, the information of the trace memory is analyzed and displayed when the program is stopped.

The Trace Mode requires time to analyze the trace memory. For this reason, not all the trace memory is analyzed at the moment the graph is displayed.

When the graph is scrolled to the right or contracted, the area shown is analyzed.

The registered symbols are not common between the Sampling Mode and the Trace Mode. They must be registered for individual modes.


Note: In the Sampling Mode and a Free Mode of Trace, when exceeding the maximum amount of data that can be held, the oldest data is erased first.

The Sampling Mode has up to 10 registered symbols for each [Graph] window. The Trace Mode has up to 4 registered symbols in sum total for all [Graph] windows.

If the switching function on the [System] page of the [Configuration Properties] dialog box has had [Realtime RAM monitor] selected, the Trace Mode cannot be used.

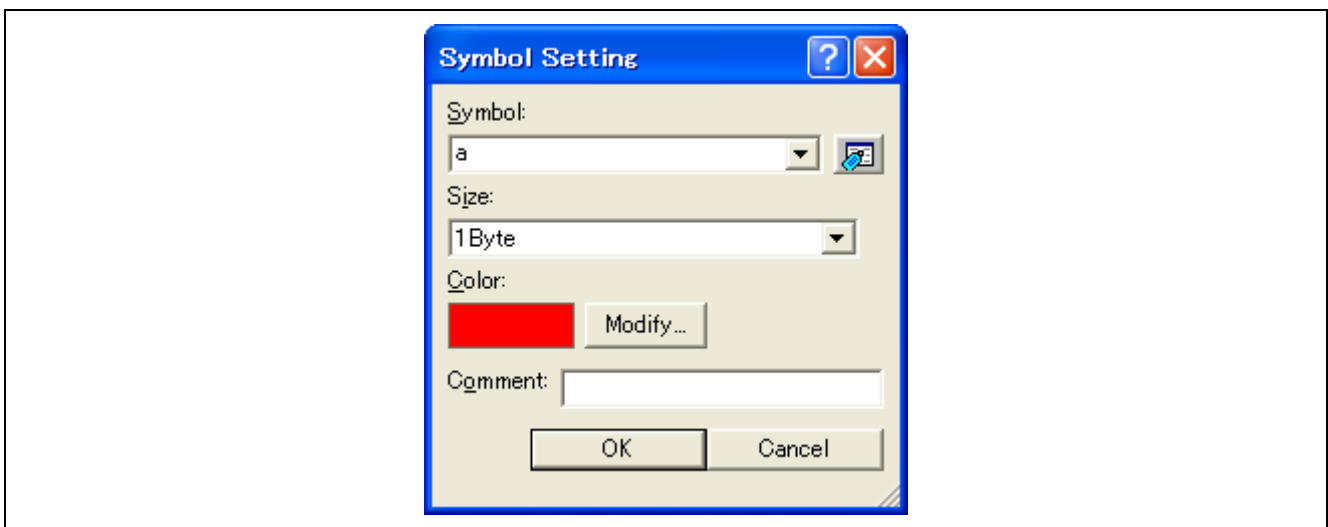
If the switching function on the [System] page of the [Configuration Properties] dialog box has had [Trace conditions] selected and the Sampling Mode is enabled, values are retrieved from regular memory, and not from the RAM monitor.

### (2) Registering a Symbol

Choosing [Add Symbol...] from the pop-up menu of the [Graph] window or clicking the toolbar button  in the [Graph] window opens the following dialog box.


You can also register a symbol by dragging and dropping a character string that can be evaluated from the Editor window into the lower pane of the [Graph] window.

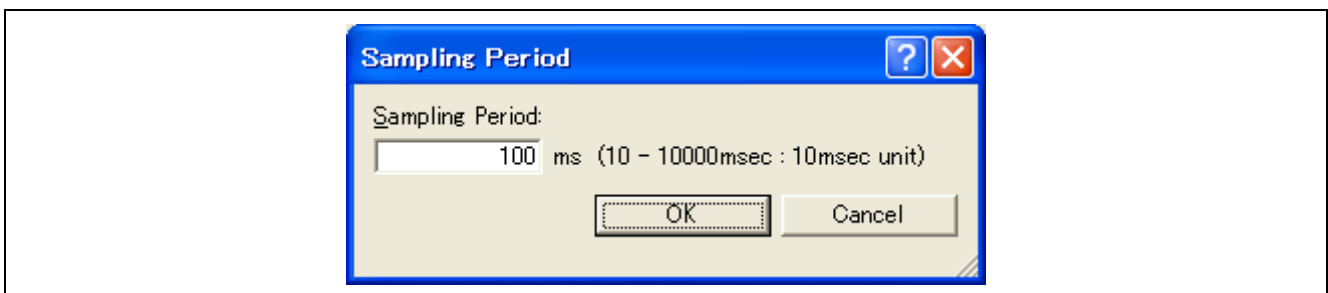
If a symbol is registered in the Trace Mode, part of the trace function becomes unusable for it.



**Figure 4.67** [Symbol Setting] Dialog Box

### (3) Setting a Sampling Period

When the [Graph] window is in the Sampling Mode, set the cycle for data measurement. Choose [Sampling Period...] from the pop-up menu of the [Graph] window or click the toolbar button  in the [Graph] window to open the following dialog box.




**Figure 4.68** [Sampling Period] Dialog Box

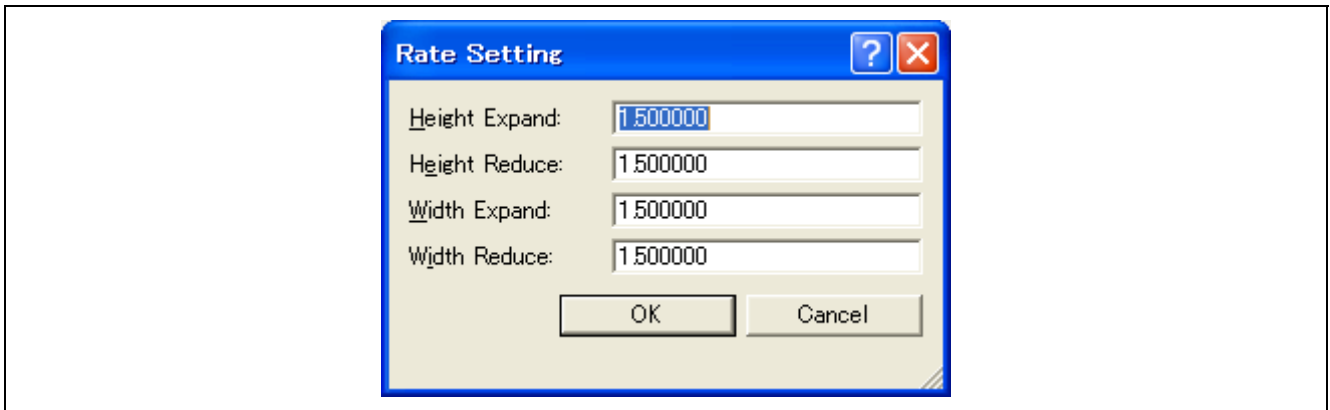
The sampling period can be specified separately for each window.

A sampling period from 10 ms to 10000 ms can be specified in 10 ms units. The initial value is 100 ms.

## (4) Setting the Expansion/Reduction Rates

Set a graph expansion rate or graph reduction rate with which you want the graph to be enlarged or contracted.

Choosing [Ration Setting...] from the pop-up menu of the [Graph] window or clicking the toolbar button  in the [Graph] window opens the following dialog box.



**Figure 4.69** [Rate Setting] Dialog Box


Enter a graph expansion or graph reduction rate with which you want the graph to be enlarged or contracted vertically or horizontally.

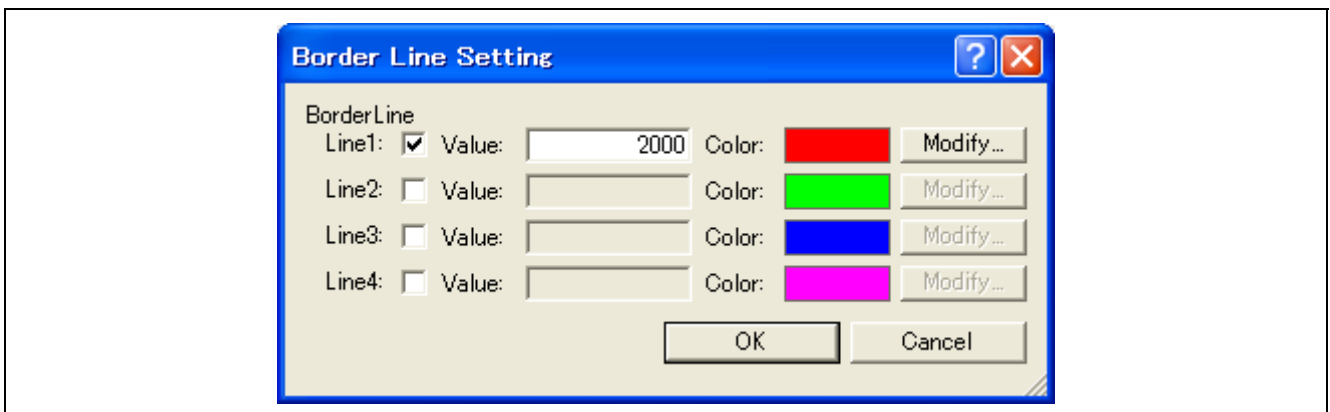
The initial value is 1.5.

For all the items in this dialog box, if a value smaller than 1 is specified, 1 is set. If a number with a decimal point is specified, the value may be approximated due to a rounding error.

## (5) Setting the borderline

Up to four borderlines can be set here.

Choosing [Borderline Setting...] from the pop-up menu of the [Graph] window or clicking the toolbar button  in the [Graph] window opens the following dialog box.



**Figure 4.70** [Border Line Setting] Dialog Box

Set whether or not to display the borderline for Line1 to line4 separately.

For the border you have selected to display, enter the value or the expression where the border is to be displayed, and select the color.

If the Radix setting of the [Graph] window is [Signed Dec] or [Signed Hex], a negative value can be specified.

#### 4.15 Stack Trace Function

The emulator uses the information on the stack to display the names of functions in the sequence of calls that led to the function to which the program counter is currently pointing. This function can be used only when the load module that has the Elf/Dwarf2-type debugging information is loaded. For the usage of this function, refer to section 5.18, Stack Trace Facility.

#### 4.16 Online Help

An online help explains the usage of each function or the command syntax that can be entered from the command line window.

Select [Emulator Help] from the [Help] menu to view the emulator help.

## Section 5 Tutorial

### 5.1 Introduction

A tutorial program for the E20 emulator is provided as a means of presenting the emulator's main features to you. The tutorial is described in this section.

The tutorial program was written in the C language, and sorts random data (10 items) into ascending and descending order.

Processing by the tutorial program is as follows.

The main function repeatedly calls the tutorial function in order to repeatedly execute the process of sorting.

The tutorial function generates the random data to be sorted and calls the sort and change functions, in that order.

The sort function accepts input of an array that contains the random data generated by the tutorial function and sorts this data into ascending order.

The change function accepts input of the array that was sorted into ascending order by the sort function and sorts the data into descending order.

The tutorial program is designed to help users to understand how to use the functions of the emulator and emulator debugger. When developing a user system or user program, refer to the user's manual for the target MCU.

- Notes:
1. This tutorial program is for operation in little-endian environments. If you need to run this program in a big endian environment, recompile the program.
  2. If the tutorial program is recompiled, the addresses in the recompiled program may not be the same as those described in this section.
  3. Filename extensions for the files to be used and the address where the tutorial program starts differs with the MCU. Therefore, actual information shown on the screen may differ from the information given in this section.

## 5.2 Starting the High-performance Embedded Workshop

Open a workspace by following the procedure described in section 2.10, Procedure for Launching the E1/E20 Emulator Debugger.

Specify the directory given below.

(Drive where the OS is installed)\Workspace\Tutorial\E1E20\RX600\Tutorial\_LittleEndian

Specify the file shown below.

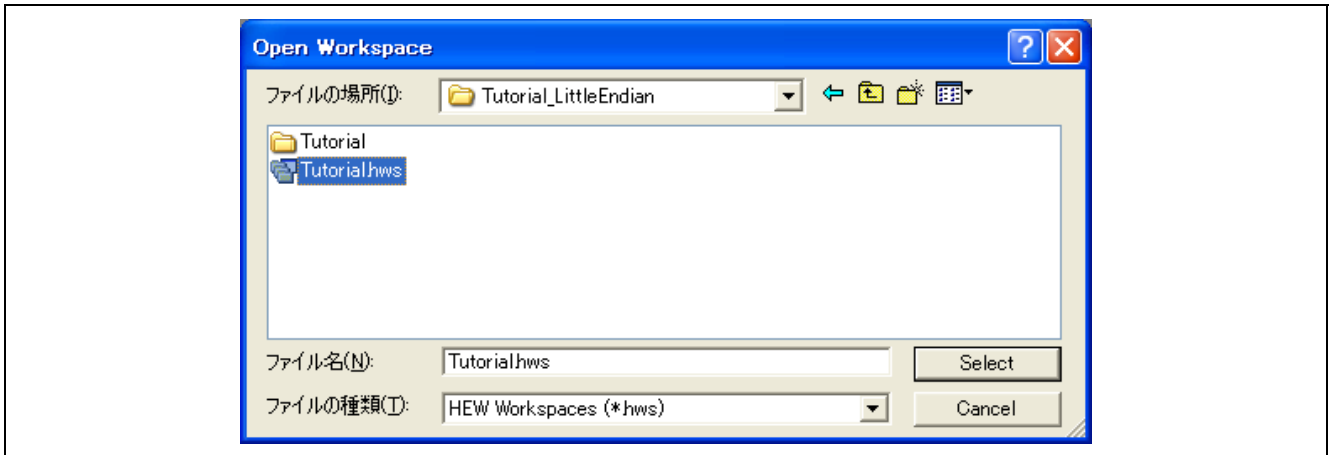


Figure 5.1 [Open Workspace] Dialog Box

## 5.3 Booting Up the Emulator

On booting up the emulator, a dialog box for setting up the debugger is displayed. Make initial settings of the debugger in this dialog box.

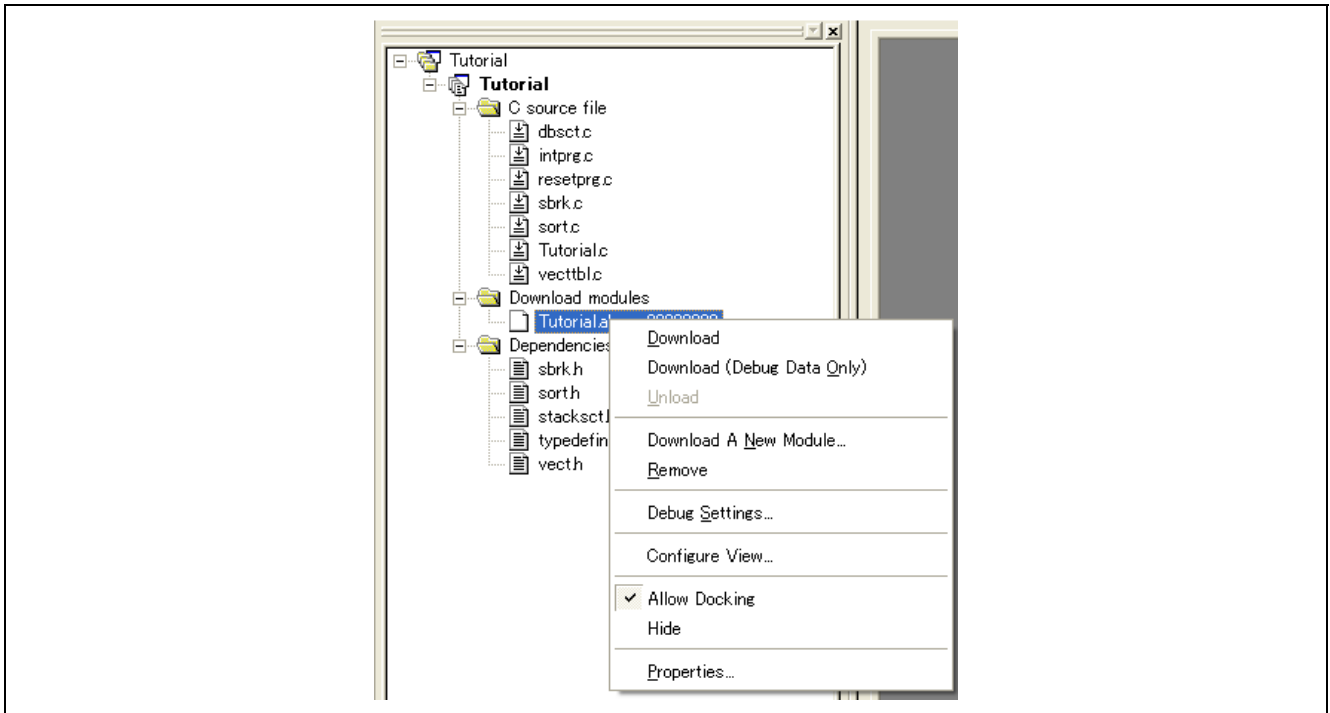
When you have finished setting up the debugger, you are ready to start debugging.

## 5.4 Downloading the Tutorial Program

### 5.4.1 Downloading the Tutorial Program

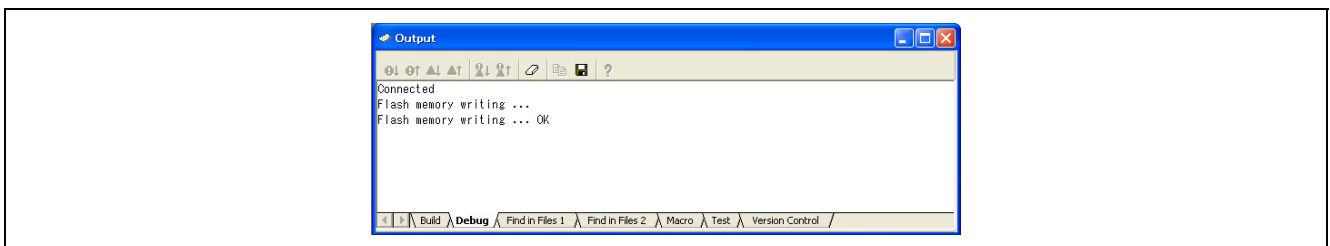
Download the object program you want to debug. Note, however, that the name of a program to be downloaded and the address where the program will be downloaded depend on the MCU in use. Accordingly, strings shown in the screen shots should be altered to those for the MCU in use.

Choose [Download] for [Tutorial.abs] under [Download modules].



**Figure 5.2** Downloading the Tutorial Program

After the above operation, a download to the internal flash memory begins. When the download is completed normally, a message is displayed to that effect in the [Output] window. An arrow is displayed in the [Tutorial.abs] icon of [Download modules].



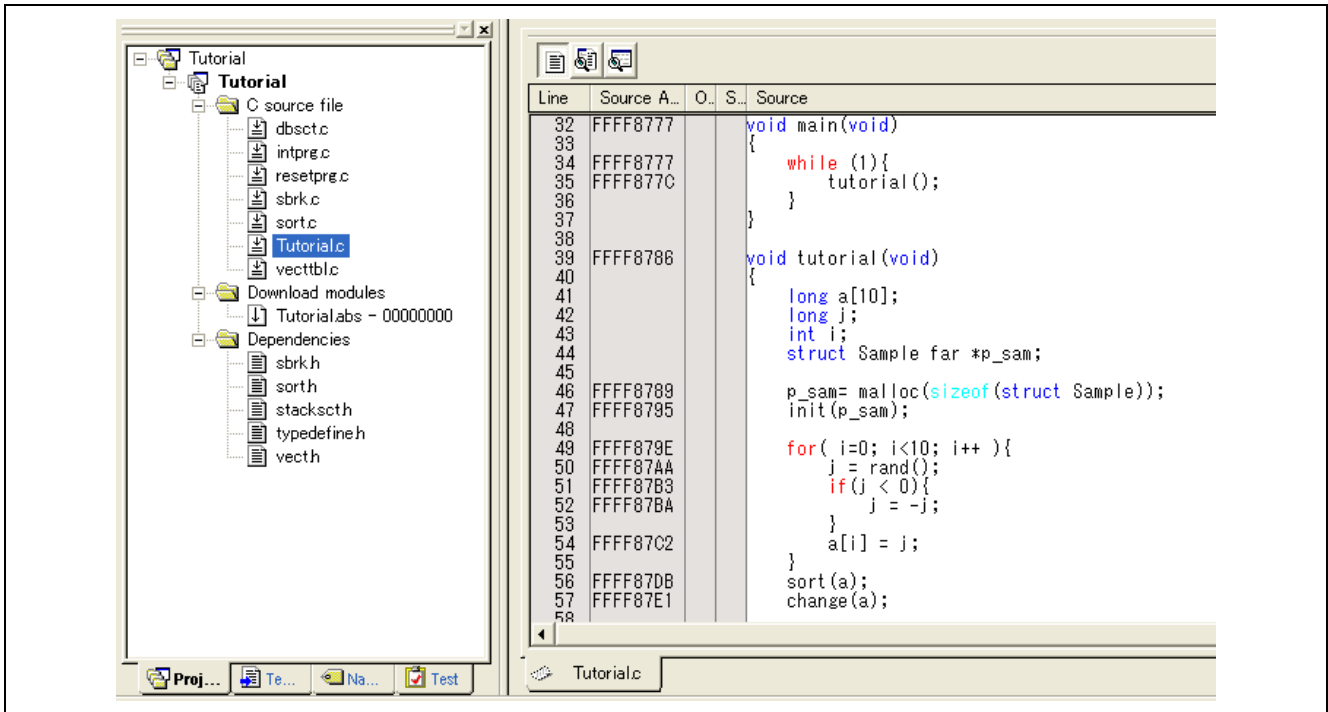
**Figure 5.3** Completion of Download of the Tutorial Program

Note that if you've checked the [Reset CPU after download] checkbox on the [Options] page of the [Debug Settings] dialog box that is displayed when you select [Debug Settings] from the [Debug] menu, the CPU is reset each time you download.



### 5.4.2 Displaying the Source Program

In the High-performance Embedded Workshop you can debug programs at the source level. Double-click on the C source file [Tutorial.c].



**Figure 5.4 [Editor] Window (Displaying the Source Program)**

If necessary, you can change the font and size to make the text more easily readable. For details, refer to the High-performance Embedded Workshop User's Manual.

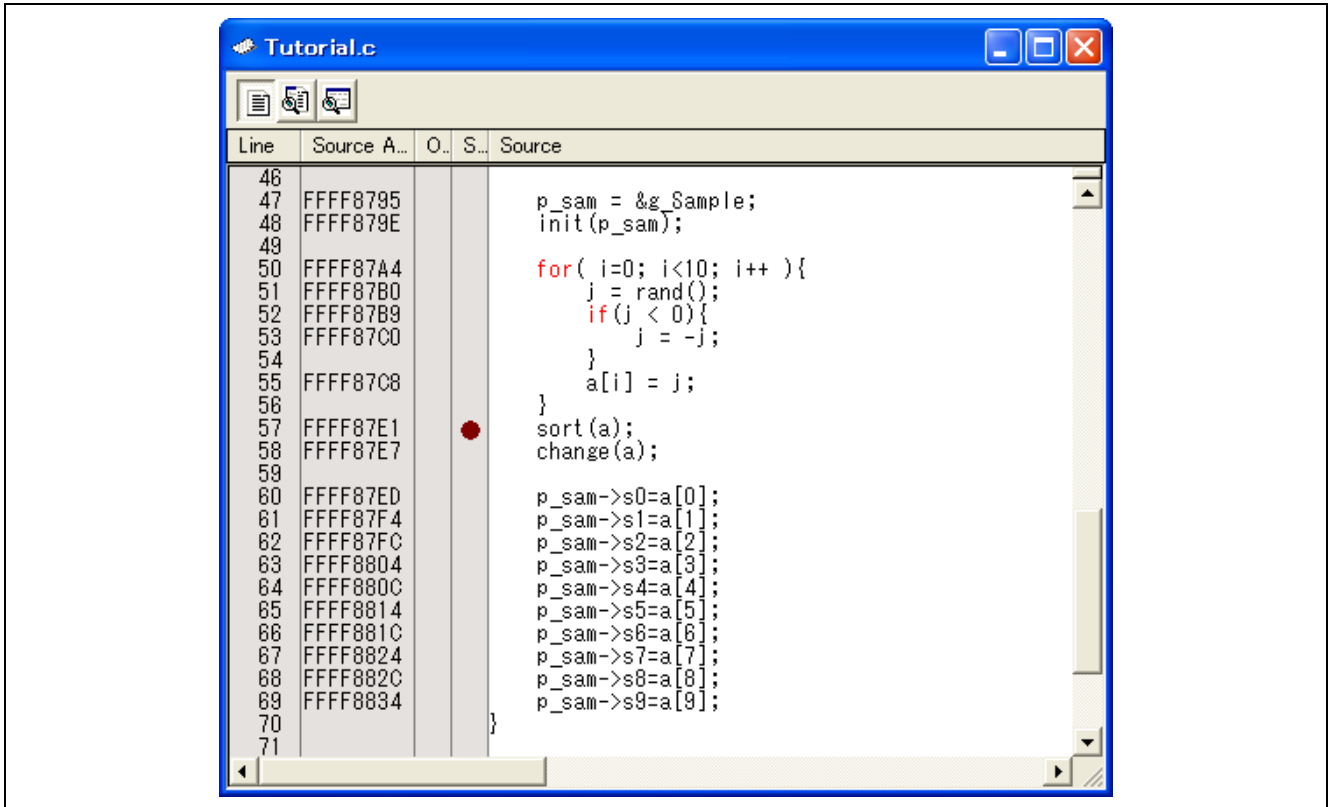
The [Editor] window initially shows the beginning of the program. Use the scroll bar to view other parts of the program.

## 5.5 Setting S/W breakpoints

Setting of S/W breakpoints is one simple debugging facility.

S/W breakpoints are easy to set in the [Editor] window. For example, you can set a S/W breakpoint at the line where the sort function is called.

Double-click in the row of the [S/W Breakpoints] column which corresponds to the source line containing the call of the sort function.



**Figure 5.5** [Editor] Window (Setting a S/W breakpoint)

The source line that includes the sort function will be marked with a red circle, indicating that a S/W breakpoint has been set there.

## 5.6 Executing the Program

The following describes how to run the program.

### 5.6.1 Resetting the CPU

To reset the CPU, choose [Reset CPU] from the [Debug] menu or click on the [Reset CPU] toolbar button (⏮).

### 5.6.2 Executing the Program

To execute the program, choose [Go] from the [Debug] menu or click on the [Go] toolbar button (▶).

The program will be executed continuously until a breakpoint is reached. An arrow will be displayed in the [S/W Breakpoints] column to indicate the position where the program stopped.

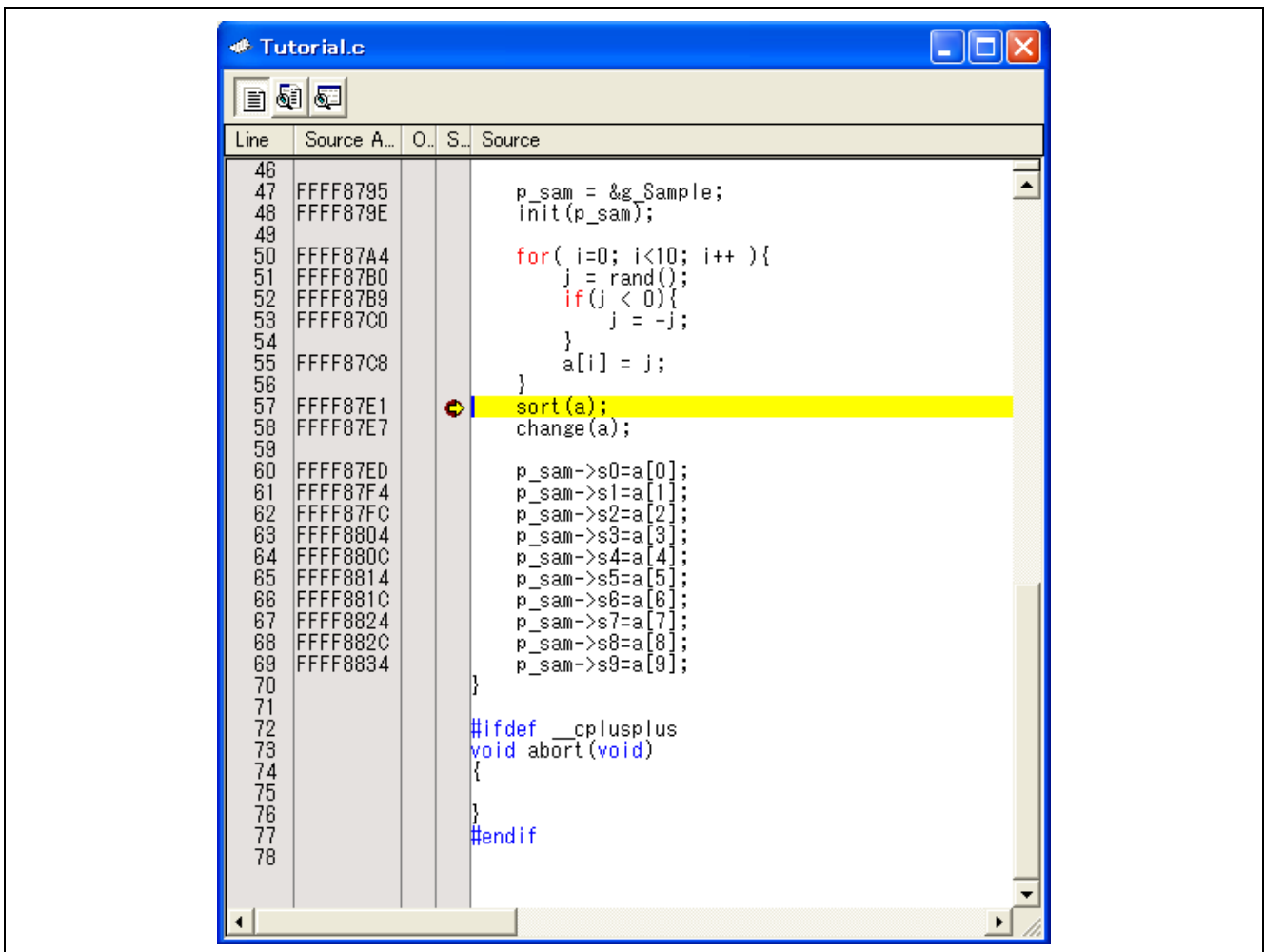

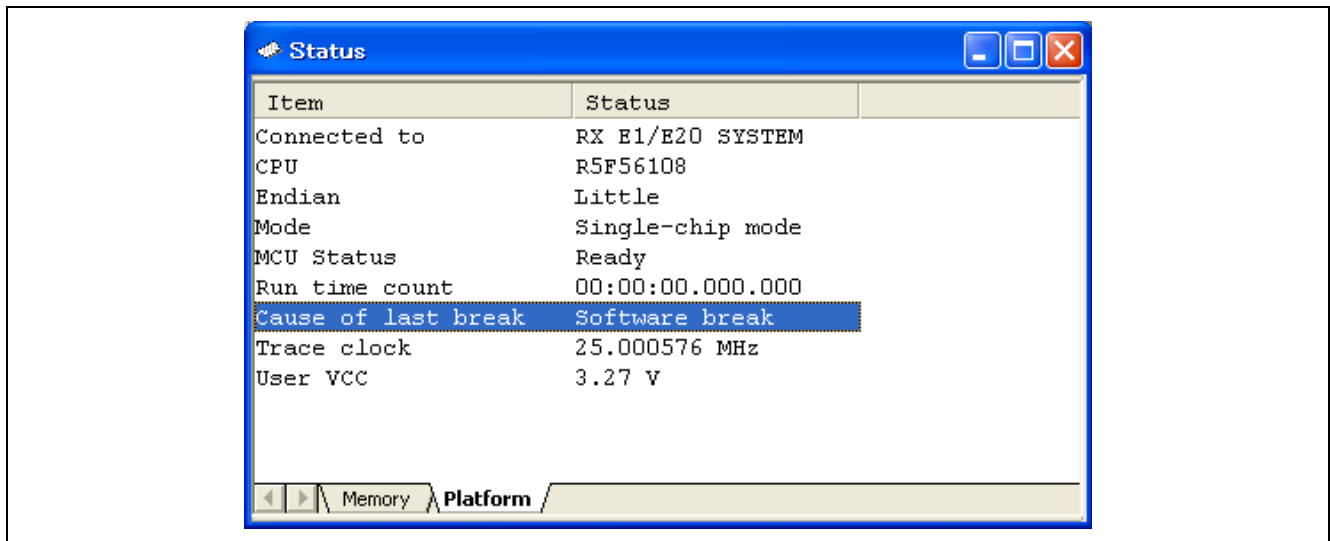


Figure 5.6 [Editor] Window (Break)

The [Status] window permits you to check the cause of the last break to have occurred. Choose [View -> CPU -> Status] or click on the [View Status] toolbar button (  ). When the [Status] window is displayed, open the [Platform] sheet and check the cause of the break.



**Figure 5.7** [Status] Window

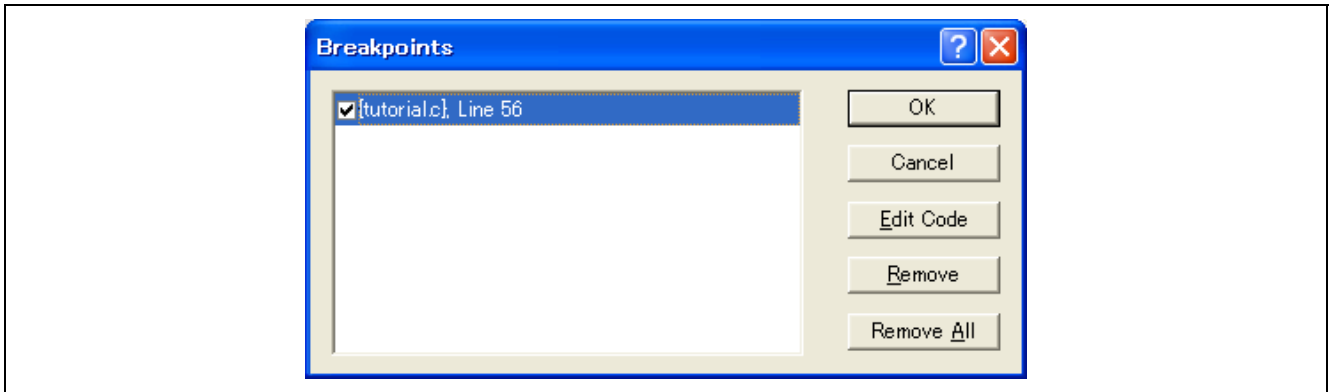
Note: The items shown in the [Status] window vary with the MCU. For details, refer to the online help.

## 5.7 Checking Breakpoints

Use the [Breakpoints] dialog box to check all S/W breakpoints that have been set.

### 5.7.1 Checking Breakpoints

Choose [Edit -> Source Breakpoints...] to open the [Breakpoints] dialog box.




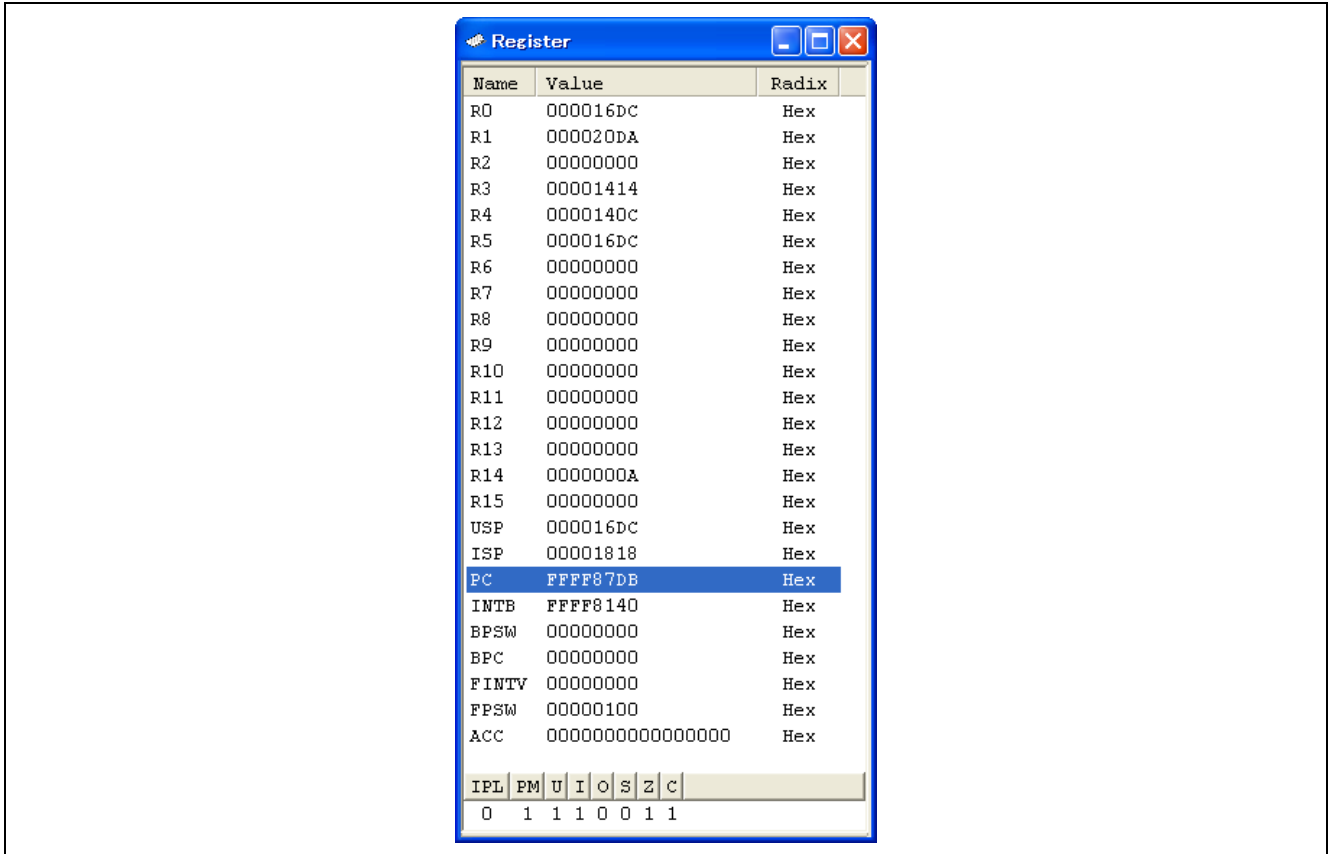
**Figure 5.8** [Breakpoints] Dialog Box

Use this dialog box to remove a breakpoint or enable or disable a breakpoint.

Note: The [Breakpoints] dialog box cannot be opened if no S/W breakpoints have been set.

## 5.8 Altering Register Contents

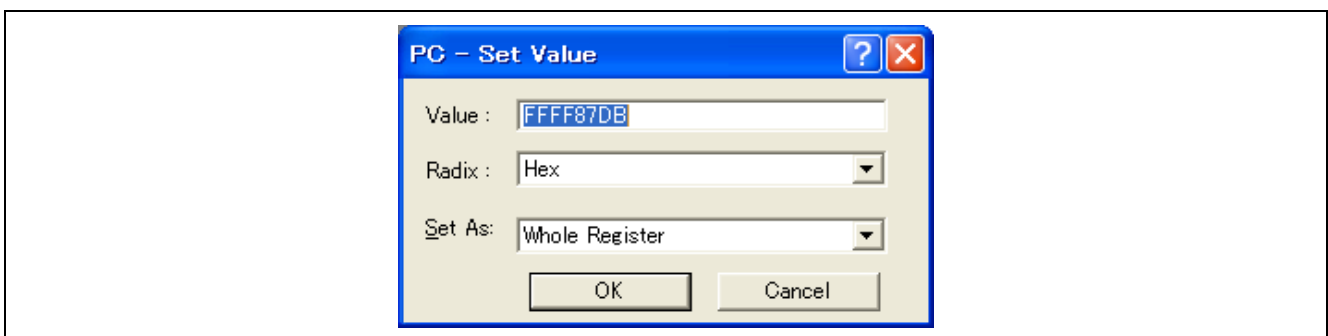
Choose [View -> CPU -> Registers] or click on the [Registers] toolbar button () . The [Register] window shown below will be displayed.



**Figure 5.9** [Register] Window

The contents of any register can be altered.


Double-click on the line for the register you want to alter. The dialog box shown below is displayed, allowing you to enter the new value for the register.



**Figure 5.10** [Set Value] Dialog Box (PC)

Note: The register items displayed in the [Register] window differ with each MCU used. For details about the register items, refer to the hardware manual for the MCU you're using.

## 5.9 Referring to Symbols

The [Labels] window permits you to view the symbolic information in a module. Choose [View -> Symbols -> Labels] or click on the [Labels] toolbar button (  ). The [Labels] window shown below will be displayed. Use this window to look at the symbolic information a module includes.

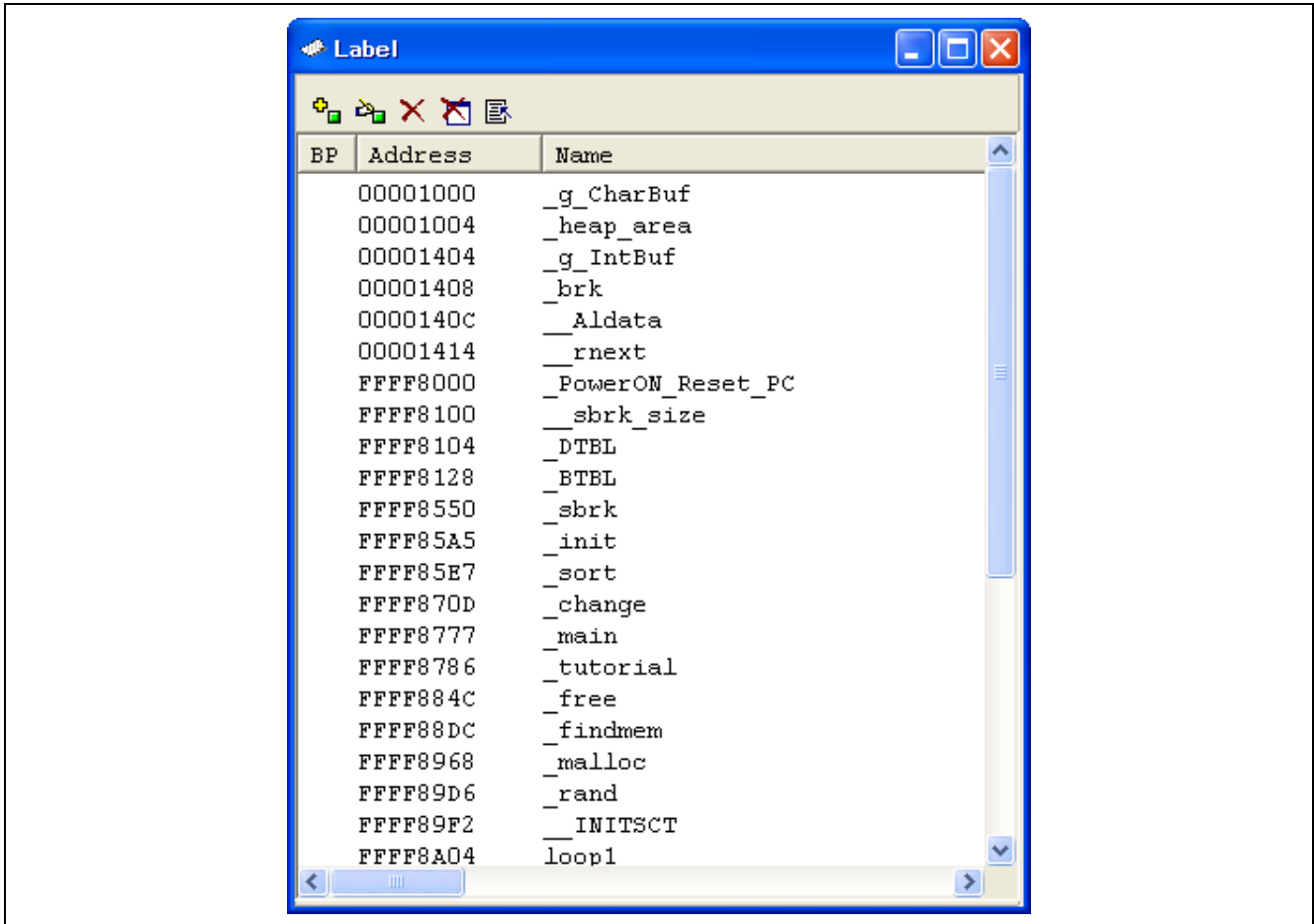



Figure 5.11 [Label] Window

### 5.10 Checking Memory Contents

After you have specified a label name, you can use the [Memory] window to check the contents of memory where that label is registered. For example, you can check the contents of memory corresponding to `_main` in byte units, as shown below.

Choose [View -> CPU -> Memory] or click on the [Memory] toolbar button  to open the [Display Address] dialog box.

Enter “`_main`” in the edit box of the [Display Address] dialog box.

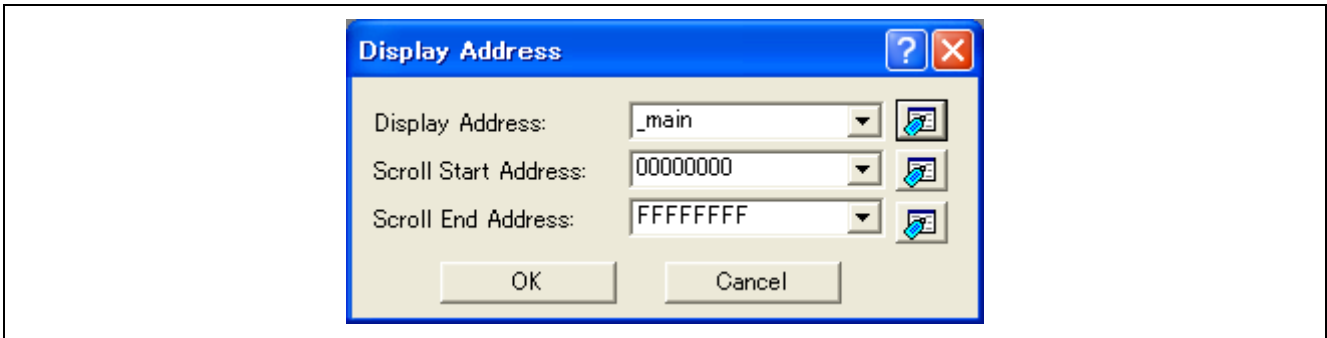


Figure 5.12 [Display Address] Dialog Box

Click on the [OK] button. The [Memory] window will be displayed, showing a specified memory area.

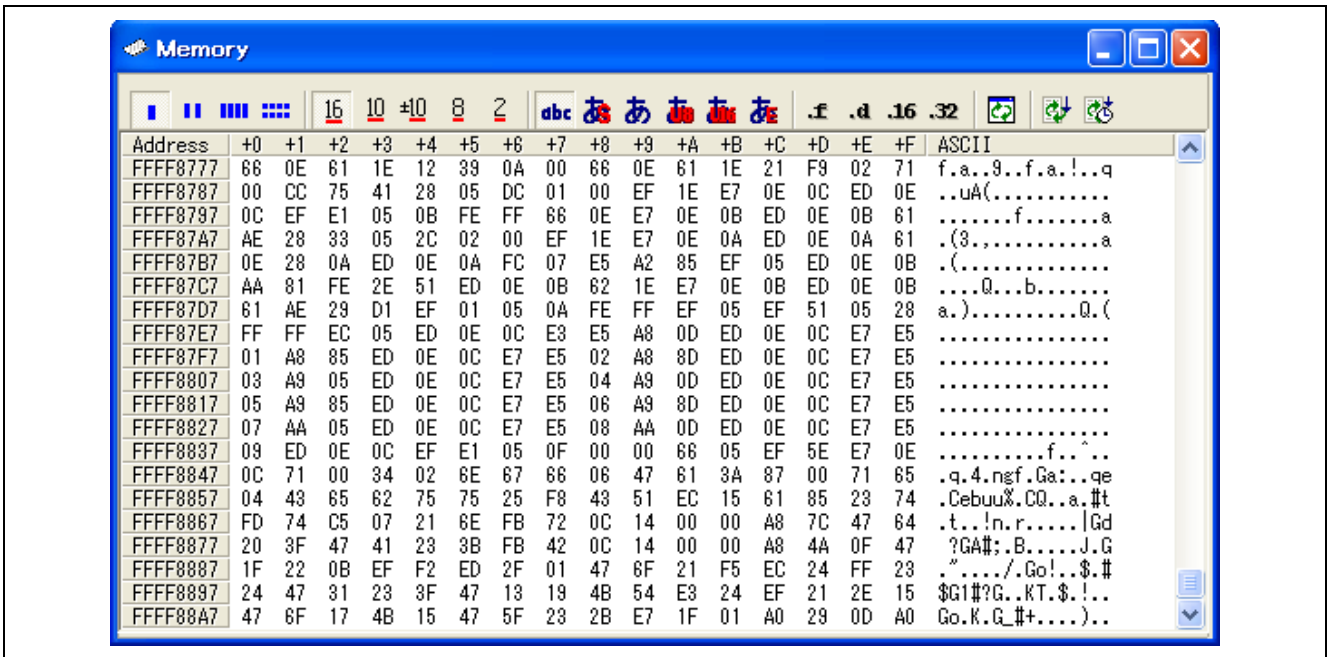


Figure 5.13 [Memory] Window



### 5.11 Referring to Variables

When single-stepping through a program, you can see how the values of the variables used in the program change as you step through source lines or instructions. For example, by following the procedure described below, you can look at the long-type array 'a' that is declared at the beginning of the program.

Click on the left-hand side of the line containing the array 'a' in the [Editor] window to place the cursor there.

Right-click and select [Instant Watch].

The dialog box shown below will be displayed.

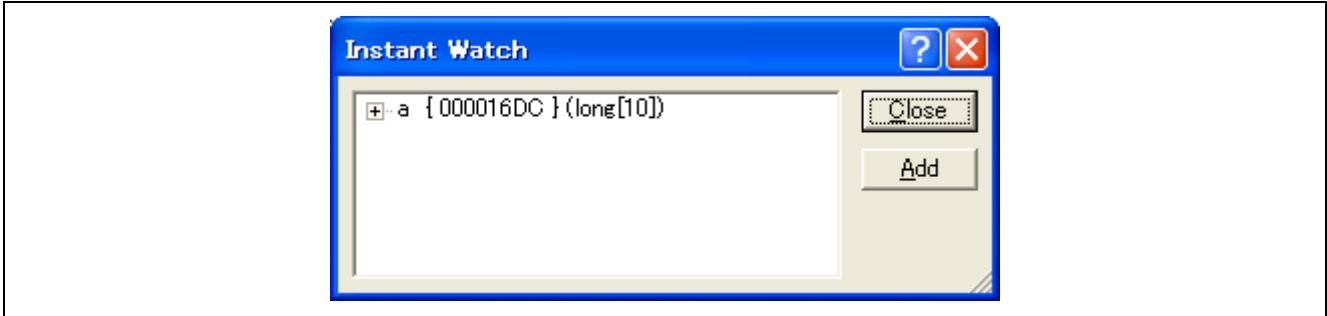


Figure 5.14 [Instant Watch] Dialog Box

Click on the [Add] button to add the variable to the [Watch] window.

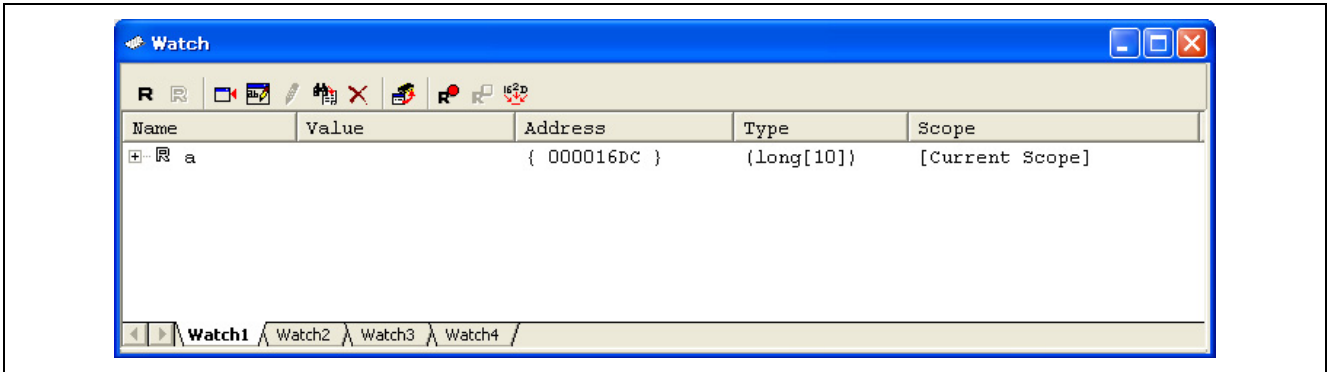


Figure 5.15 [Watch] Window (Array Display)

Alternatively, you can specify a variable name to be added to the [Watch] window.

Right-click in the [Watch] window and choose [Add Watch] from the popup menu. The dialog box shown below will be displayed.

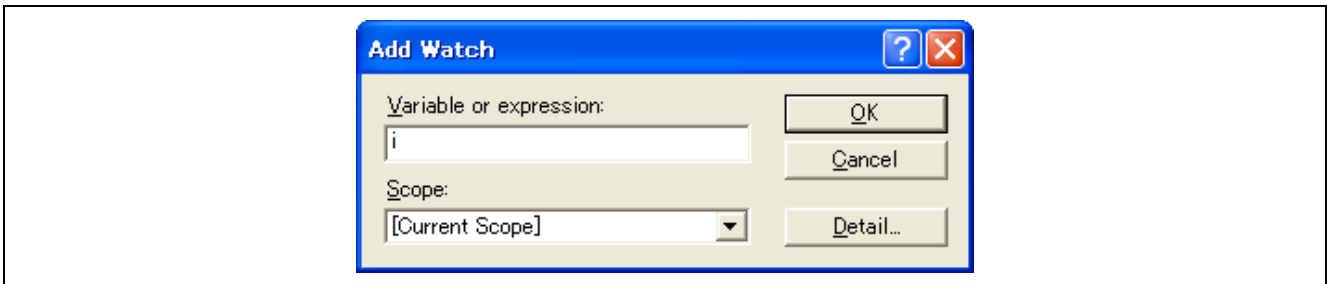


Figure 5.16 [Add Watch] Dialog Box

Enter variable 'i' in the [Variable or expression] edit box and click on the [OK] button. The int-type variable 'i' will be displayed in the [Watch] window.

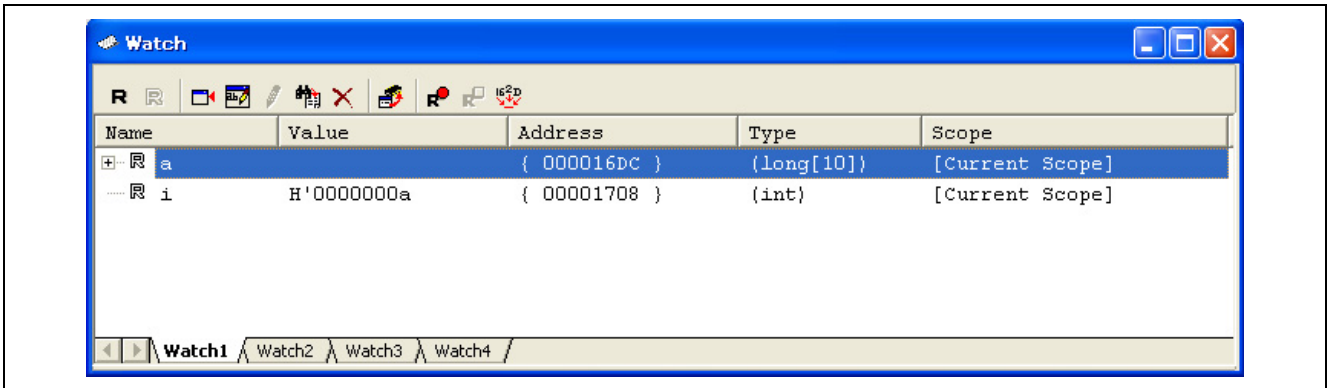


Figure 5.17 [Watch] Window (Showing a Variable)

Click on the "+" mark shown to the left of the array 'a' in the [Watch] window. You can now look at the individual elements of the array 'a.'

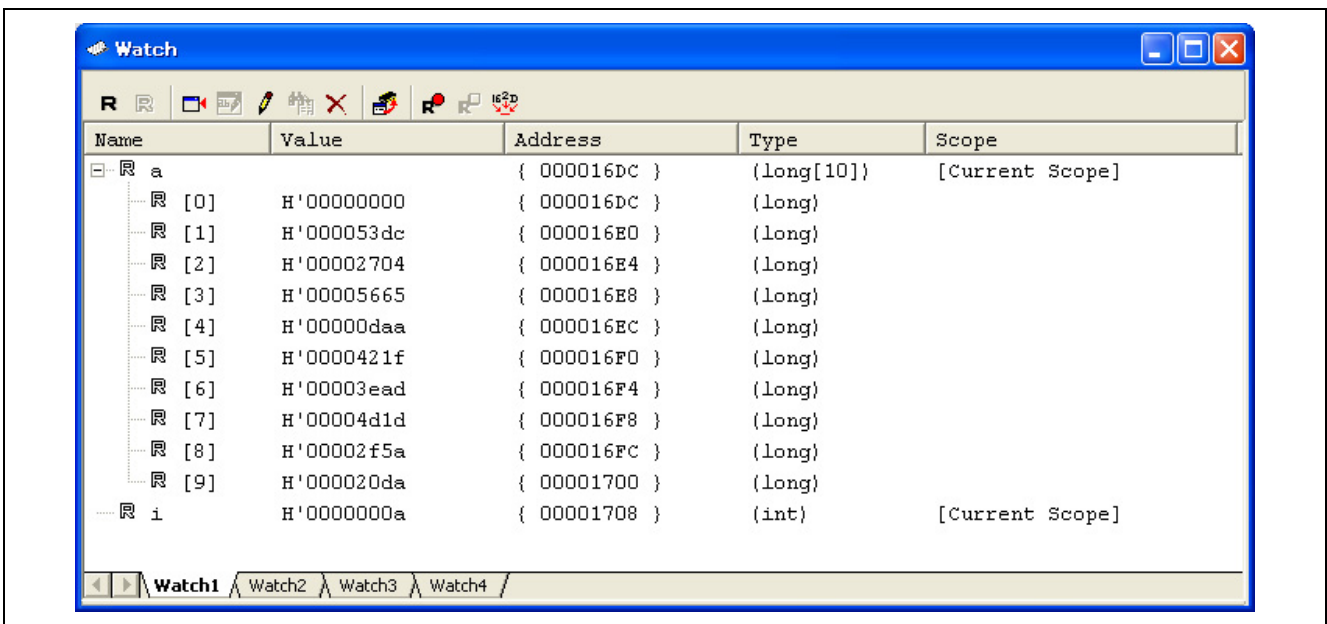

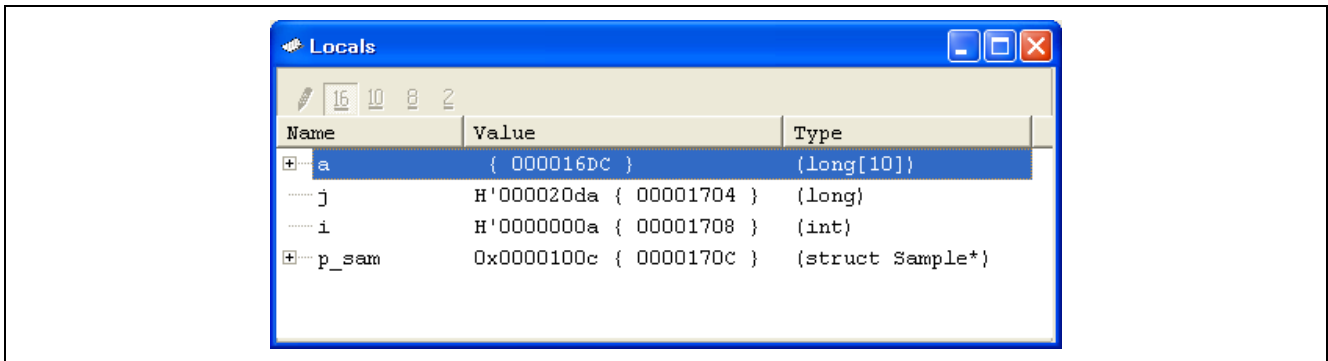


Figure 5.18 [Watch] Window (Showing Array Elements)

## 5.12 Showing Local Variables

By using the [Local] window, you can view the local variables included in a function. As an example, let's check the local variables of the tutorial function. Four local variables are declared in this function: 'a,' 'j,' 'i' and 'p\_sam.' Choose [View -> Symbols -> Local] or click on the [Locals] toolbar button (  ) to display the [Locals] window. The [Locals] window shows the values of local variables in the function indicated by the current value of the program counter (PC). If no variables exist in the function, no information is displayed in the [Locals] window.



**Figure 5.19** [Locals] Window

Click on the “+” mark shown to the left of array “p\_sam” in the [Locals] window to display the elements of class instance p\_sam.

Confirm that the random data are being sorted into descending order by inspecting the elements of class instance “p\_sam” before and after execution of the sort function.

### 5.13 Single-Stepping through a Program

The High-performance Embedded Workshop provides various step commands that will prove useful in debugging programs.

**Table 5.1 Step Options**

Command	Description
Step In	Executes a program one statement at a time (including statements within functions).
Step Over	Executes a program one statement at a time by 'stepping over' function calls, if there are any.
Step Out	After exiting a function, stops at the next statement of a program that called the function.
Step...	Single-step a program a specified number of times at a specified speed.

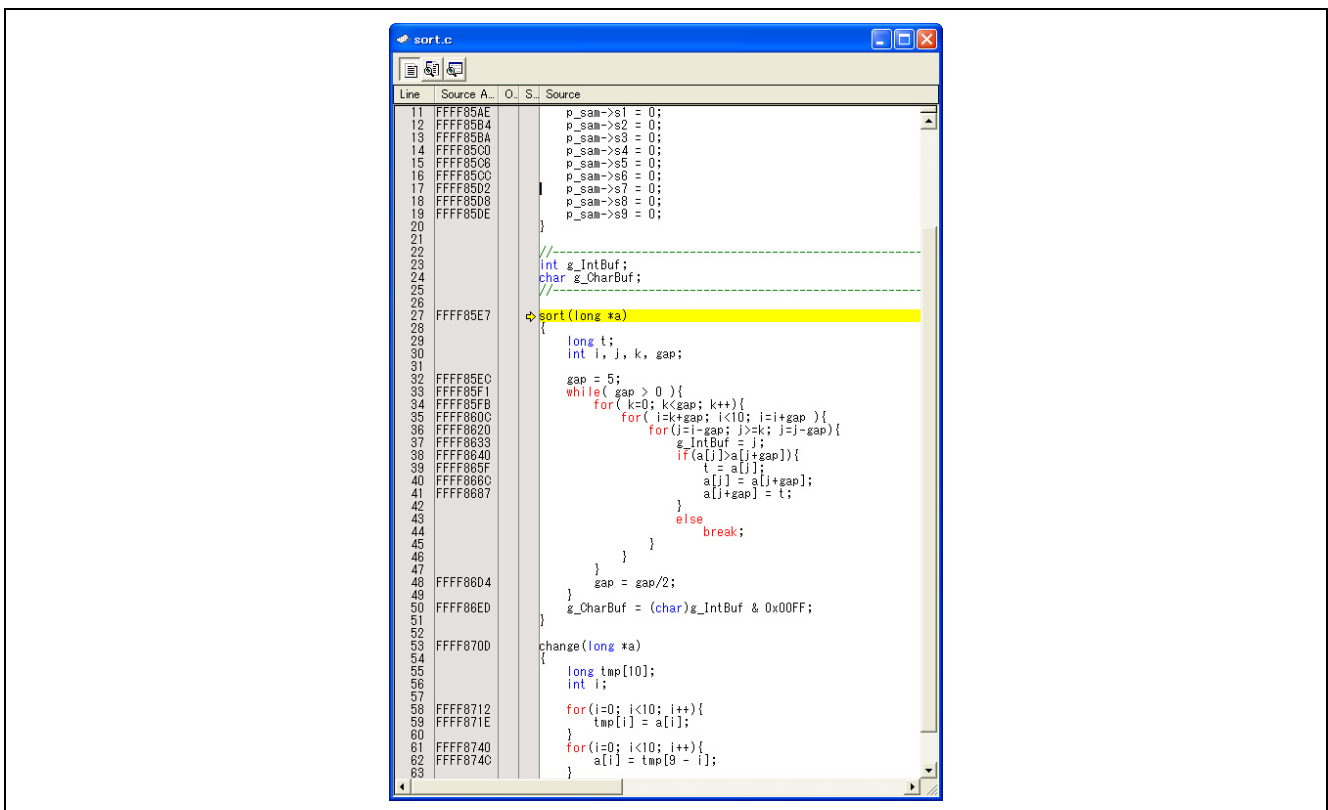
#### 5.13.1 Executing [Step In]

[Step In] 'steps in' to a called function and stops at the first statement of the function.

To enter the sort function, choose [Step In] from the [Debug] menu or click on the [Step In] toolbar button.



**Figure 5.20 [Step In] Button**



**Figure 5.21 [Editor] Window (Step In)**

The highlight in the [Editor] window moves to the first statement of the sort function.

5.13.2 Executing [Step Out]

[Step Out] takes execution out of a called function by completing its execution at once and only stopping at the next statement of the program from which the function was called.

To exit from the sort function, choose [Step Out] from the [Debug] menu or click on the [Step Out] toolbar button.



Figure 5.22 [Step Out] Button

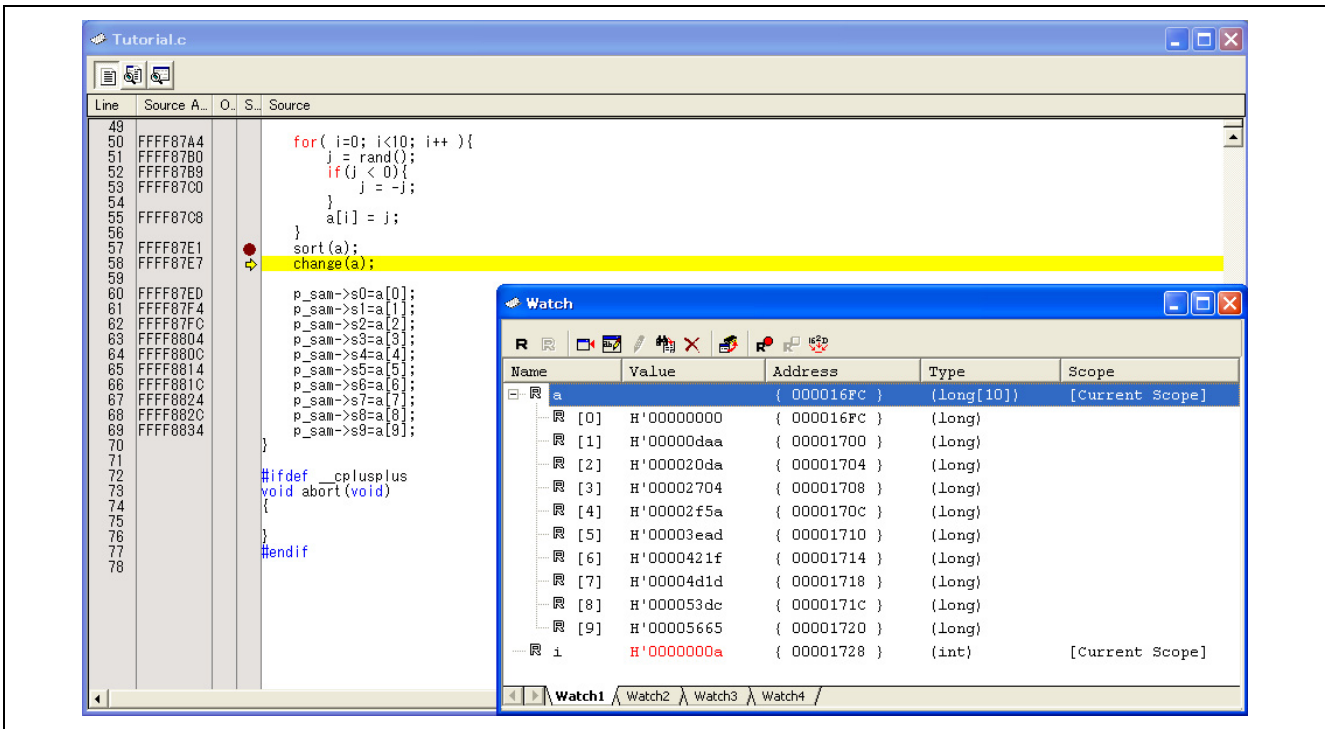


Figure 5.23 [Editor] Window (Step Out)

The data of the variable 'a' displayed in the [Watch] window will have been sorted into ascending order.

### 5.13.3 Executing [Step Over]

[Step Over] executes the whole of a function call as one step and then stops at the next statement of the main program. To execute all statements in the change function at once, choose [Step Over] from the [Debug] menu or click on the [Step Over] toolbar button.



Figure 5.24 [Step Over] Button

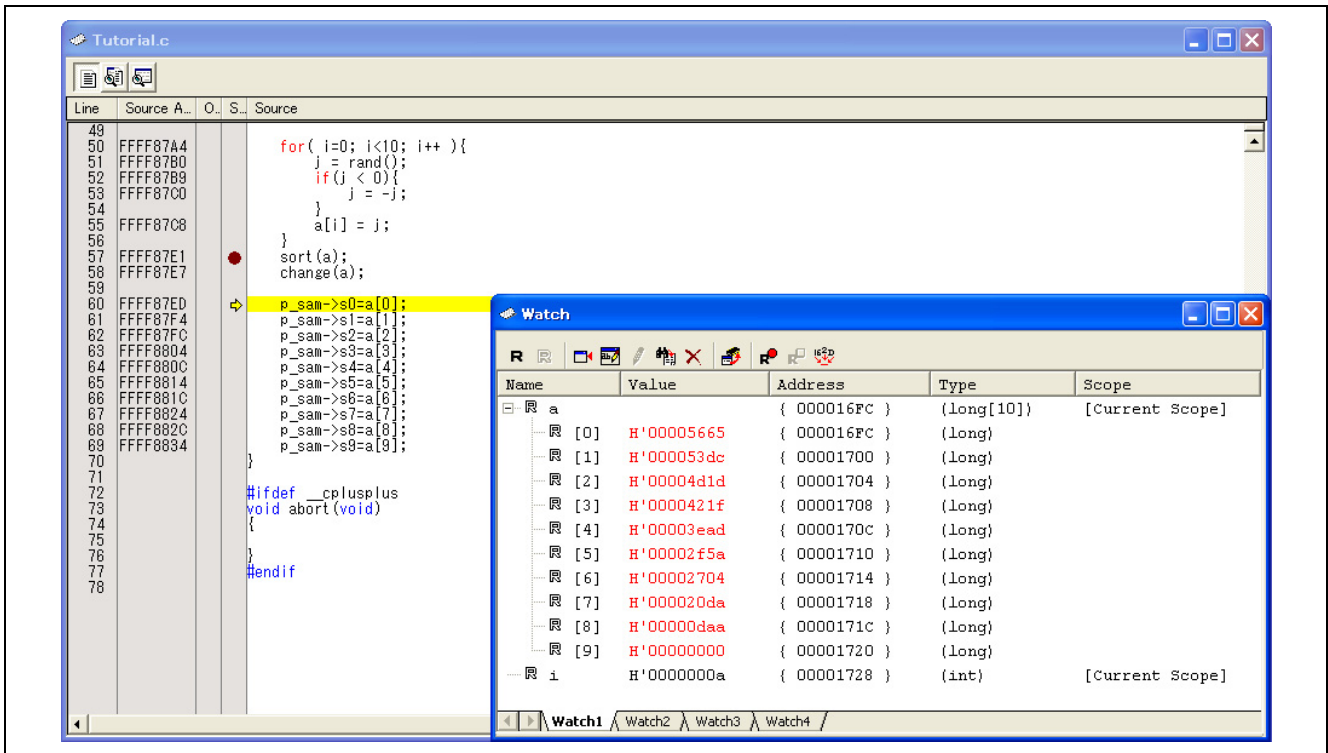


Figure 5.25 [Editor] Window (Step Over)

The data of the variable 'a' displayed in the [Watch] window will have been sorted into descending order.

## 5.14 Forcibly Breaking Program Execution

The High-performance Embedded Workshop permits you to forcibly break program execution.

Clear all breakpoints.

To execute the rest of the tutorial function, choose [Go] from the [Debug] menu or click the on [Go] toolbar button.



**Figure 5.26 [Go] Button**

Since the program execution is now in an endless loop, choose [Stop Program] from the [Debug] menu or click on the [Halt] toolbar button.



**Figure 5.27 [Halt] Button**

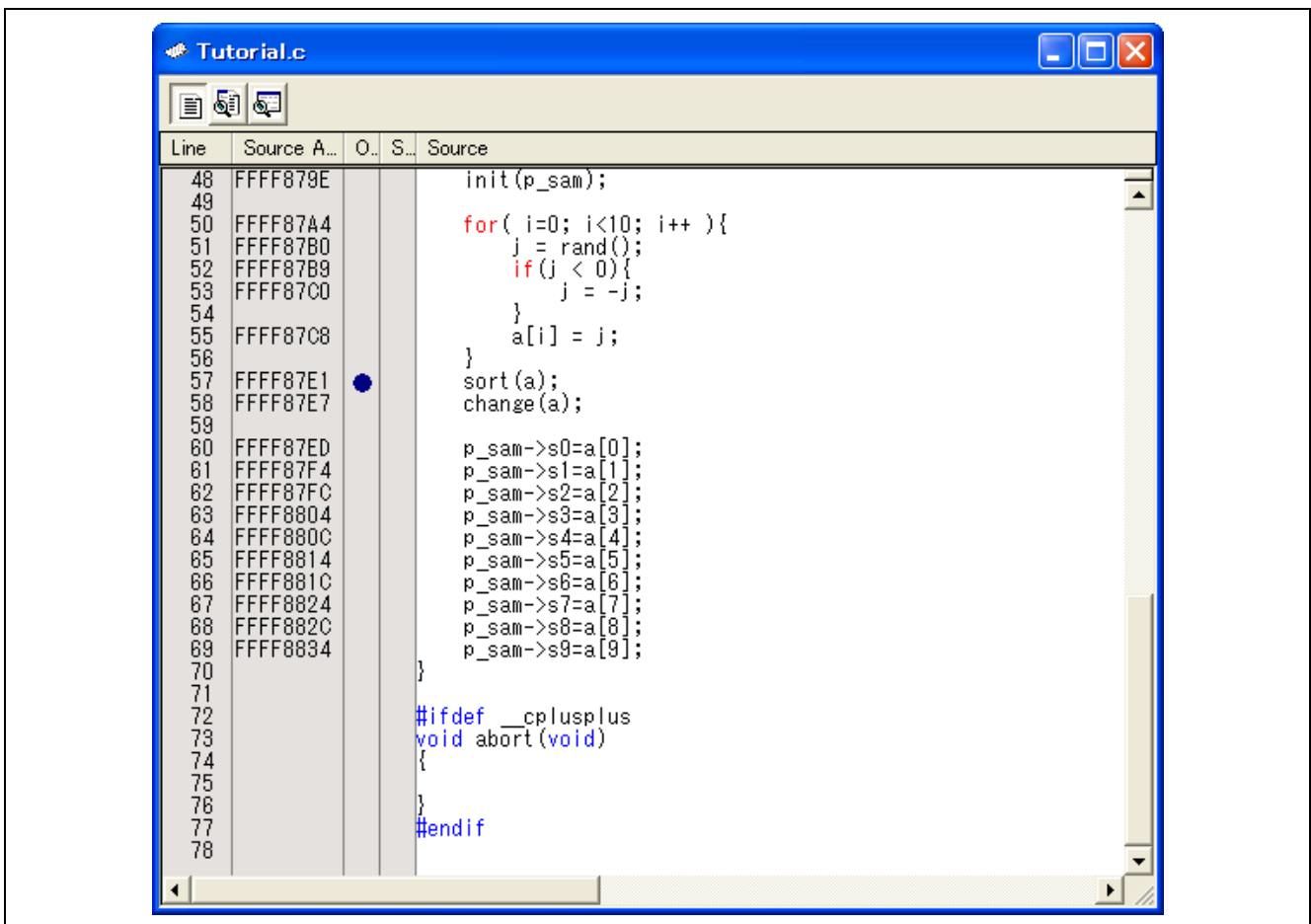
## 5.15 On-Chip Break Facility

The on-chip break facility is available if it is supported by the MCU. An on-chip break causes the program to stop when it executes the instruction at a specified address (execution address) or reads from or writes to a specified memory location (data access).

### 5.15.1 Stopping a Program when It Executes the Instruction at a Specified Address

It's easy to set an execution-address event as an on-chip breakpoint in the [Editor] window. For example, you can set an on-chip breakpoint where the sort function is called.

Double-click in the row of the [On-Chip Breakpoint] column which corresponds to the source line containing the call of the sort function.



**Figure 5.28** [Editor] Window (Setting an On-Chip Breakpoint)

The source line that includes the sort function will be marked with a filled blue circle, indicating that an on-chip breakpoint that will cause a program to stop when it fetches an instruction has been set there.



## 5.16 Stopping a Program when It Accesses Memory

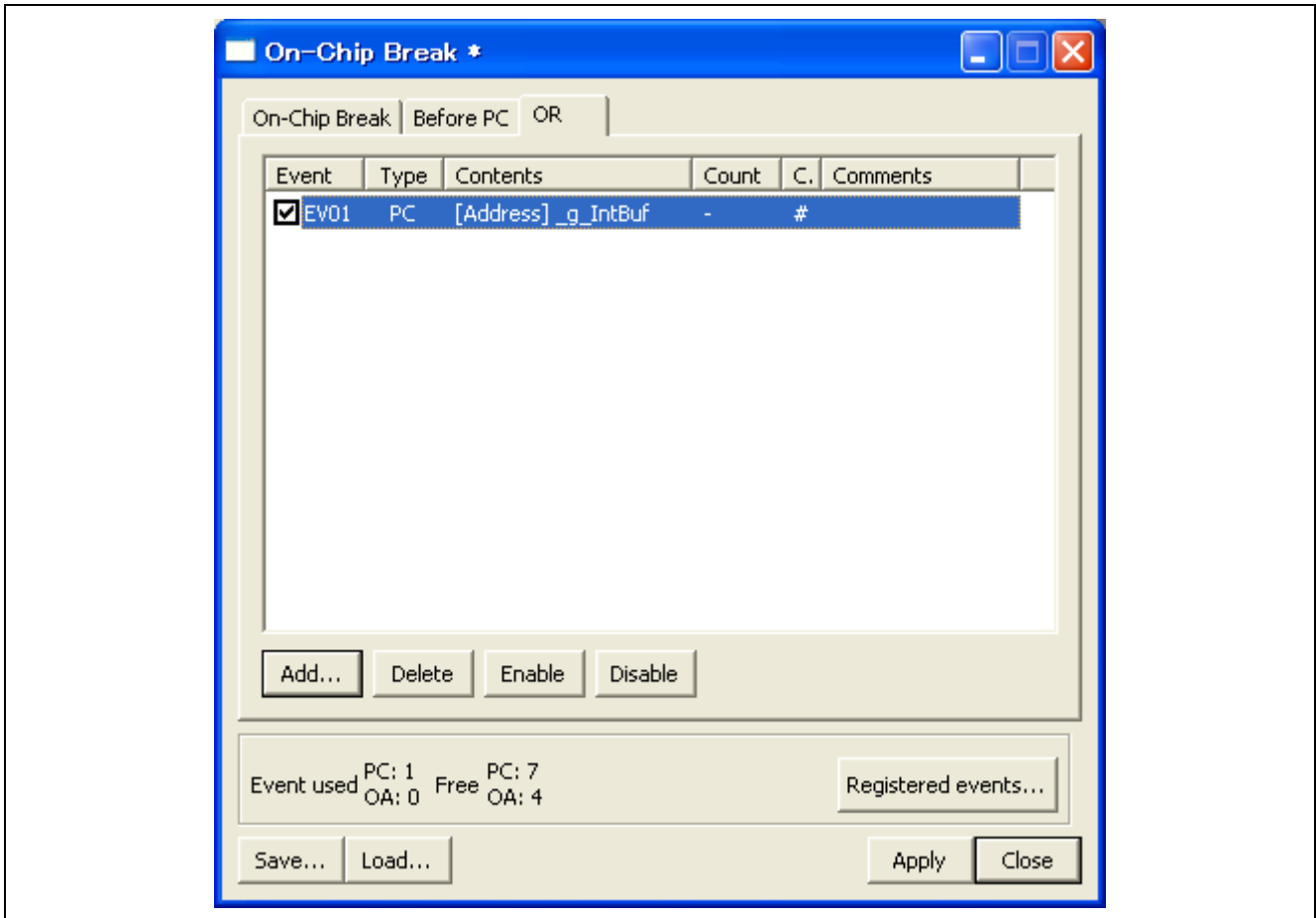
To make a program stop when it reads or writes the value of a global variable, follow the procedure below. Choose [View -> Event -> On-chip Break] to open the [On-Chip Break] dialog box.

Select the [OR] page of the [On-Chip Break] dialog box.

Select a global variable in the [Editor] window, and drag-and-drop the selected variable into the [OR] page so that the program will stop when it reads or writes the value of that variable.

Then click on the [Apply] button.

The program will stop running when it reads or writes the value of the global variable you have set.



**Figure 5.29 [On-Chip Break] Dialog Box**


- Notes:
1. Only global variables of 1 or 2 bytes in size can be registered.
  2. Local variables cannot be set as on-chip break conditions.

## 5.17 Tracing Facility

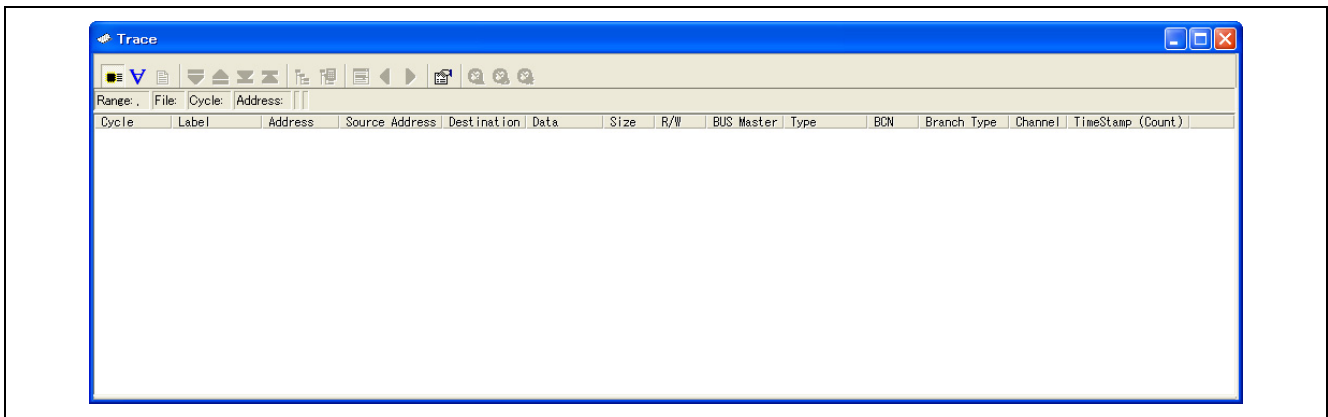
The two types of tracing facility listed below are available if they are supported by the MCU.

- Internal trace  
Tracing of this type uses the internal trace buffer in the MCU. The branch and data access information is displayed in the [Trace] window. The contents of the displayed information and the number of cycles that can be acquired vary with the MCU.
- External trace  
Tracing of this type is only supported by the E20 emulator, and is only available when the MCU in use supports external-trace output and its trace pins are connected to the E20 emulator. The contents of the displayed information and the number of cycles that can be acquired vary with the MCU.

The following is an example of settings for the RX600 Series MCUs.

Choose [View -> Code -> Trace] or click on the [Trace] toolbar button (  ).

The [Trace] window will be displayed.



**Figure 5.30** [Trace] Window

The following section gives an outline of the tracing facility and how to set up the facility.

### 5.17.1 Showing the Information Acquired in “Fill until Stop” Tracing

In “fill until stop” tracing, trace information is successively acquired from the start of user program execution until a break is encountered.

- (1) Clear all break conditions. Click the right mouse button with the cursor anywhere in the [Trace] window and choose [Acquisition] from the popup menu. The [Trace conditions] dialog box shown below will be displayed. Check that the selected trace mode is [Fill until stop]. Click on the [Close] button.

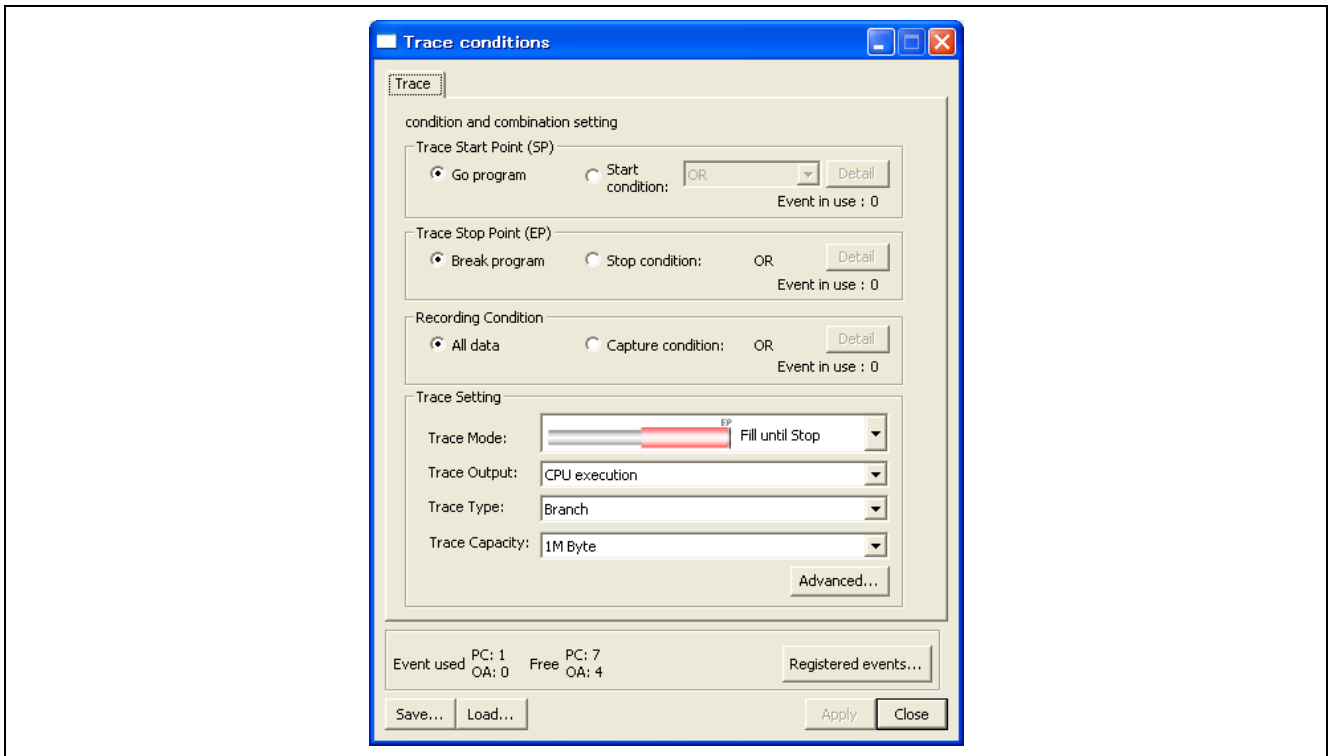


Figure 5.31 [Trace conditions] Dialog Box (Fill until Stop)

- (2) Set a S/W breakpoint on the line of the tutorial function: “p\_sam ->s0=a[0];”.

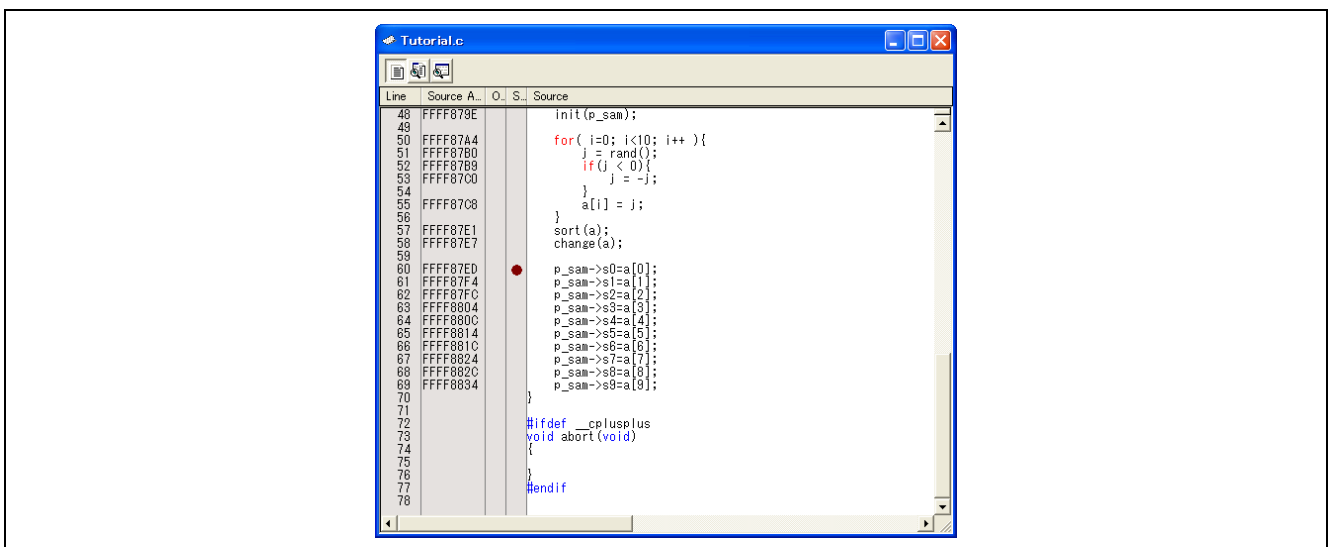


Figure 5.32 [Editor] Window (S/W breaks Set in tutorial Function)

(3) Choose [Reset Go] from the [Debug] menu. Processing will be halted by the break, and the trace information acquired prior to the break will be displayed in the [Trace] window.

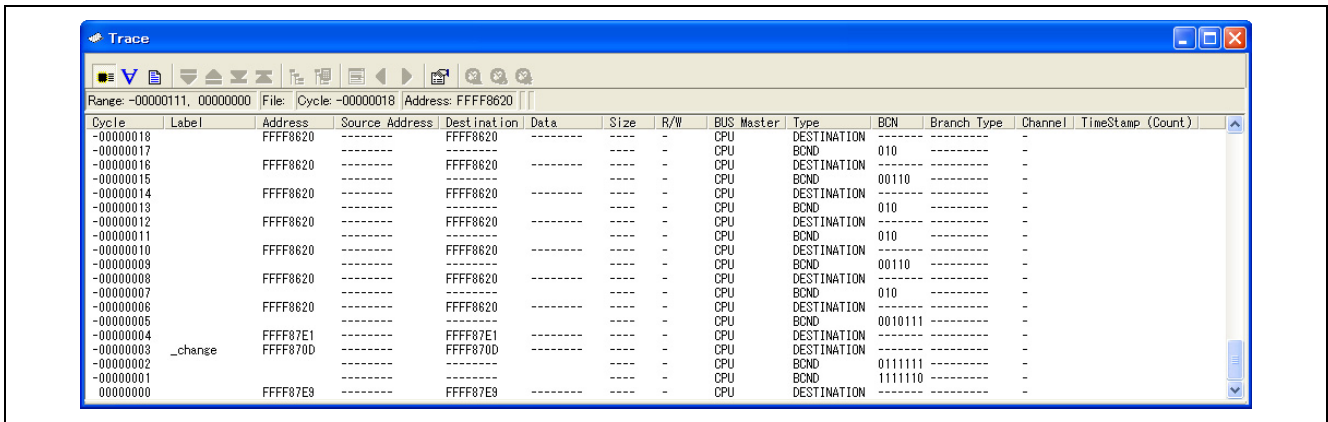


Figure 5.33 [Trace] Window (“Fill until Stop” Tracing)

(4) A mixed display of bus information and disassembly listing is possible. Choose [Display Mode → DIS] from the popup menu to view trace information in mixed bus and disassembly mode.

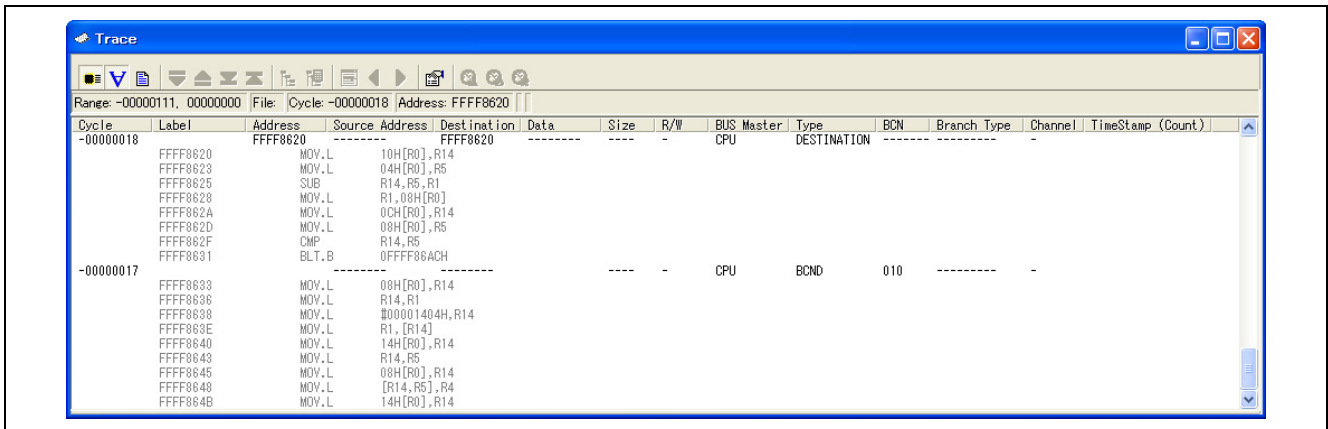


Figure 5.34 [Trace] Window (Mixed Bus and Disassembly Mode)

(5) Choosing [Display Mode → SRC] from the popup menu, on the other hand, shows a mixture of bus information, disassembly listing, and source code as the trace information.

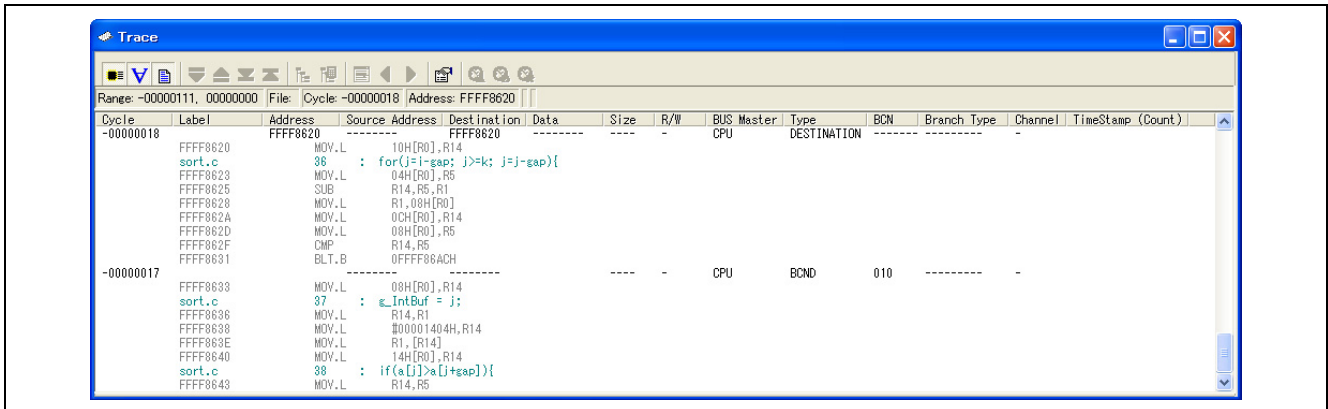
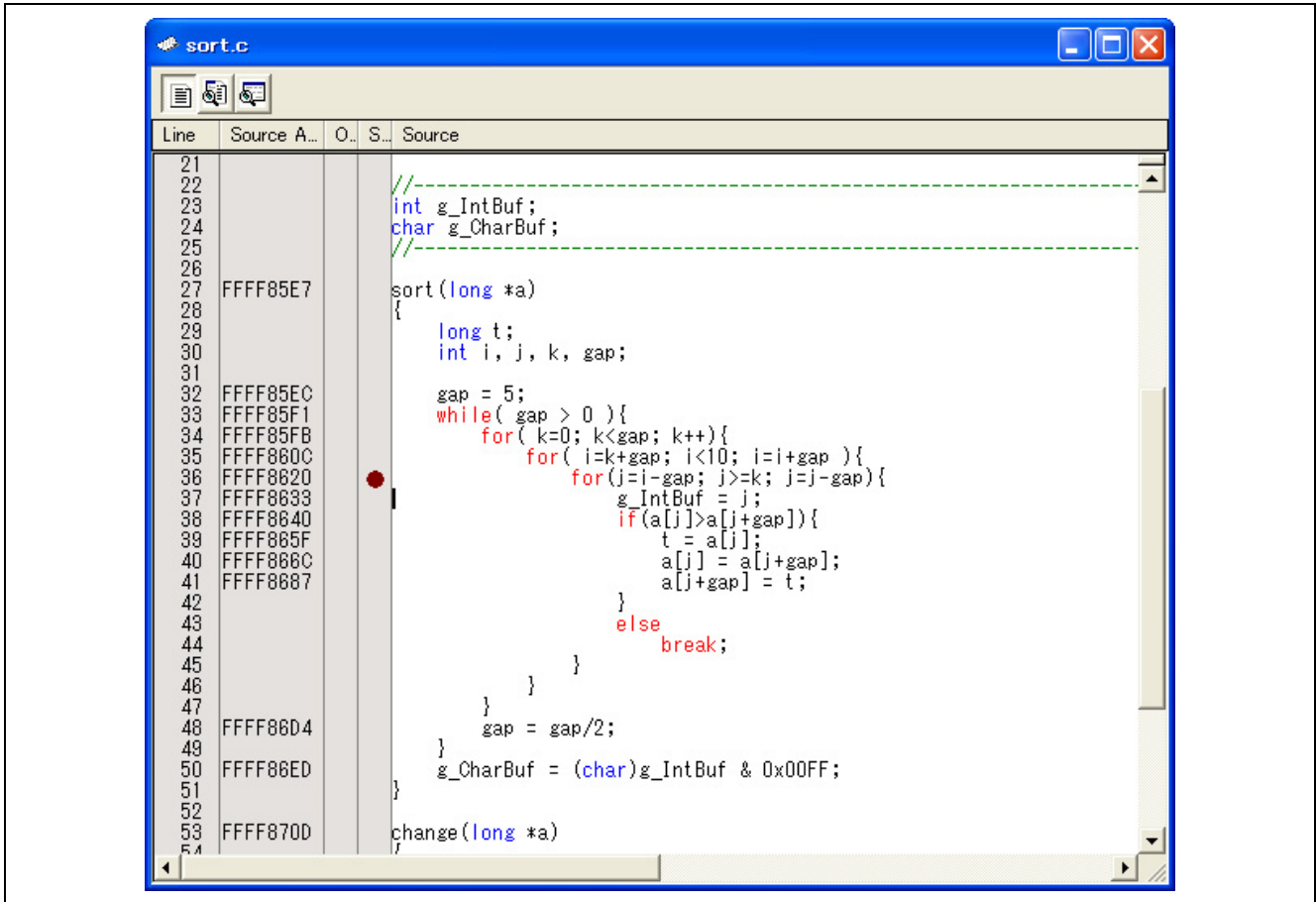


Figure 5.35 [Trace] Window (Mixed Bus, Disassembly, and Source Mode)

## 5.18 Stack Trace Facility

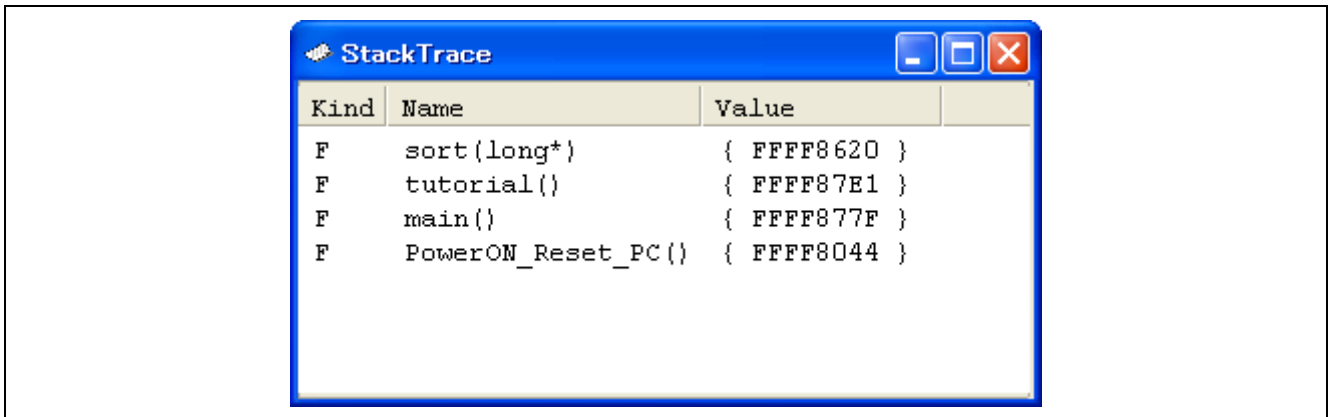
Stack information can be used to find out which function called the function corresponding to the current PC value. Set a S/W breakpoint in any line of the tutorial function by double-clicking on the corresponding row in the [S/W Breakpoints] column.



**Figure 5.36** [Editor] Window (Setting a S/W breakpoint)

Choose [Reset Go] from the [Debug] menu.

After the break, choose [View -> Code -> Stack Trace] to open the [Stack Trace] window.



**Figure 5.37** [Stack Trace] Window

You will see that the current PC value is within the sort() function, and that the sort() function was called from the tutorial() function.

Clear the S/W breakpoint that you set on a line of the tutorial function by again double-clicking on the corresponding row in the [S/W Breakpoints] column.

## 5.19 What Next?

In this tutorial, we have introduced to you several features of the emulator and usage of the High-performance Embedded Workshop.

The emulation facilities of the emulator provide for advanced debugging. You can apply them to precisely distinguish the causes of problems in hardware and software and, once these have been identified, to effectively examine the problems.

## Section 6 Notes on Usage

### 6.1 Memory

#### 6.1.1 I/O Register Area

To view or set the data in the I/O register area by such as memory window, access must be in the units defined in the hardware manual. Therefore, using the I/O window is recommended.

#### 6.1.2 Internal Flash ROM Area

- When memory data in the internal flash ROM area are modified by an operation other than downloading (e.g. editing through the [Memory] window or line assembly), the new values are written to the data flash area the next time the user program starts running.
- An attempt at reading from the data flash area when it has been erased leads to undefined values according to the specifications of the MCU are being displayed. If programming of the data flash area through the debugger is attempted, data are written in 256-byte units and none of them is an undefined value.
- The blocks not registered on the [Internal flash memory overwrite] page are erased before a download is performed.
- When you're going to download multiple files but their respective areas overlap in the same block of flash memory, be sure to register that block on the [Internal flash memory overwrite] page.
- MCUs that are connected to the emulator and used in debugging are placed under stress by repeated programming of flash memory during emulation. Do not use MCUs that were used in debugging in mass-production for end users.

#### 6.1.3 Downloading to the Internal Flash ROM

Some "access sizes" in the [Download Module] dialog box are not supported. Make sure the specified access size is "1", "2", or "4".

#### 6.1.4 Re-Writing the Internal Flash ROM

- If, after rewriting the internal flash ROM (program ROM area) in the user program, you want to inspect the rewritten content in the memory window, etc., check the [Debugging the program re-writing the internal PROGRAM ROM] checkbox on the [System] page of the [Configuration Properties] dialog box. If this is not selected, reference to re-written contents is not possible. For the data flash area, check the [Debugging the program re-writing the internal DATA FLASH] checkbox in the same way as for the above.
- During execution of the user program, do not refer to contents of the internal flash ROM (program area and data flash area) which have been re-written with the user program. Values read out from re-written areas will not be correct.

### 6.1.5 FCU-RAM and FCU Firmware Areas

Do not use the debugger to alter the contents of the FCU-RAM area. The debugger is incapable of rewriting contents of the FCU firmware area.

### 6.1.6 Working RAM Area for Use by the Debugger

The specified amount of bytes in use beginning with the working RAM address specified on the [MCU] page of the [Configuration Properties] dialog box is used by the firmware when rewriting the internal flash memory. Although the user program can also use this area (because memory data in this area will be saved in the host machine and then restored), do not specify any address in this area as the origin or destination of a transfer by the DMA or DTC. An area in the on-chip RAM must be designated as the working RAM area. The user program must not include any process that will disable the on-chip RAM.

## 6.2 [Memory] Window

### 6.2.1 Copy, Comparison, Find

Do not specify 8 bytes as the data size for the copy, comparison, and find functions. If this size is specified, the emulator will not operate correctly.

Also, these functions have a maximum range of 16 Mbytes.

### 6.2.2 Menu Items

Of the functions in the [Memory] window, the menu items that are grayed out do not work.

## 6.3 Running Programs

### 6.3.1 [Go To Cursor]

- When [Go To Cursor] is executed, the program does not stop at any breakpoints that have been set.

[RX600 Series MCUs]

- If a total of 8 or more events of executed-address type have been set in trace condition or performance condition settings, the "Go To Cursor" command cannot be used.

[RX200 Series MCUs]

- If a total of 4 or more events of executed-address type have been set in trace condition or performance condition settings, the "Go To Cursor" command cannot be used.

### 6.3.2 [Run Program]

Only the first of the temporary PC breakpoints set for [Run Program] is valid.

### 6.3.3 [Step Out]

Stepping out of a function that uses a jump instruction (i.e. not a subroutine-return instruction) to return to the origin of the call is not possible.



## 6.4 Reset

### 6.4.1 Contention between the Resets and the Operations by the Emulator System

An error message "A timeout error. The MCU is in the reset state. Is system reset issued?" is displayed in cases of contention between a reset (through a pin, from the watchdog timer, etc.) and operations by the emulator system (memory reference in the [Memory] window, etc.).

If the [Yes] button is clicked, the emulator is initialized and the user program stops. After a system reset is issued, the trace record is initialized too. Or, if the [No] button is clicked, neither the emulator is initialized nor the user program stops.

Note that after you've clicked the [Yes] button, you can continue with debugging.

### 6.4.2 MCU Reset When Using the Trace Function

If a reset through a pin, from the watchdog timer, etc. is issued while a user program is running, the correct recording of trace information both before and after the reset will not be possible.

### 6.4.3 MCU Reset When Using the Realtime RAM Monitor

If a reset through a pin, from the watchdog timer, etc. is issued while a user program is running, the results of subsequent RAM monitoring cannot be guaranteed (values displayed may be incorrect).

### 6.4.4 Reset during the User Program Execution

If a pin reset or an internal reset occurs under either of the following conditions, refer to Table 6.1, showing the notes on pin resets, or Table 6.2, showing notes on internal resets. The points to note depend on the operation mode of the MCU and communication interface of the emulator.

- While the user program is being executed in the on-chip ROM disabled extended mode or user boot mode
- While the user program is being executed via FINE communication interface

**Table 6.1 Notes when a Pin Reset has occurred**

Groups	Interface	Operation mode	Notes when a pin reset has occurred during user program execution
RX610, RX621, RX62N	JTAG	On-chip ROM disabled extended	The reset is canceled by the emulator. Therefore, the reset timing here differs from when the actual MCU is operating singly.
RX630, RX631, RX63N, RX63T	JTAG	User boot or On-chip ROM disabled extended	
RX630, RX631, RX63N, RX63T, RX210, RX21A, RX220	FINE	Any mode	
RX210, RX21A, RX220	FINE	User boot or On-chip ROM disabled extended	When a pin reset has occurred during the execution of the user system, the performance counter values and the acquired trace data are initialized.

Table 6.2 Notes when an Internal Reset has occurred

Groups	Interface	Operation mode	Notes when an internal reset has occurred during user program execution
RX610, RX621, RX62N, RX630, RX631, RX63N, RX63T	JTAG	On-chip ROM disabled extended	Debugging can be performed after the reset is canceled and the MCU operation mode is set to the on-chip ROM disabled extended mode in the user program.
RX630, RX631, RX63N, RX63T	JTAG	User boot	If an internal reset occurs, it becomes impossible to control from the emulator.
RX630, RX631, RX63N, RX63T, RX210, RX21A, RX220	FINE	Any mode	Do not generate an internal reset such as those generated by the watchdog timer.

## 6.5 [IO] Window

### 6.5.1 Customization of the I/O-Register Definition File

The internal I/O registers can be accessed from the [IO] window.

Furthermore, since there will be changes to MCU specifications, change the I/O-register definition file to match the description in the hardware manual in cases where the addresses of I/O registers in the I/O-register definition file differ from those given in the hardware manual for the target MCU. I/O registers are customizable in accord with the format of the I/O-register definition file.

### 6.5.2 Verify

In the [IO] window, the verify function of the input value is disabled.

## 6.6 Tracing

### 6.6.1 Traceable Accesses

- Only access via the CPU bus is traceable. No record of access by the DMAC or DTC is included in trace data.

### 6.6.2 Trace Information

- If conditions for tracing to start and stop have been specified, trace information is only acquired during each period between satisfaction of the conditions at the respective points. In such cases, the disassembled code displayed in the [Trace] window will not be correct.
- The disassembly display of the [Trace] window can show up to 1,024 instructions for a one-branch interval. If there are 1,025 or more instructions in a branch interval, the disassembly display is disabled.

#### [RX600 Series MCUs]

- Trace information of type 'BCND' is not displayed until "BRANCH", "DESTINATION", or "BRANCH/DESTINATION" is output. In other words, the [Trace] window shows no trace information if the type of all trace cycles is "BCND".

- 

#### [RX200 Series MCUs]

- If the data type that is set for trace acquisition condition is "Data" (data access) or "Data + time" (data access + time), be sure to set address conditions in the trace extraction function. Otherwise, trace information on data accesses is not acquired.  
The trace information on data accesses has only the accesses to specified addresses recorded in it. Note that the Address column of the trace window shows the start address values that were set in the trace extraction function.
- TimeStamp in the trace window is implemented using the counter for performance measurement. Therefore, if the counter start/stop condition by a combination of events is set in the [Performance] dialog, expected timestamps are not displayed.

## 6.7 Events

### 6.7.1 Detectable Accesses

- Only access via the CPU bus is detectable. Conditions for access by the DMAC or DTC cannot be included in events.

### 6.7.2 Event Combinations

- If 'AND' or 'Sequential' has been selected as an event break condition, only 'OR' is selectable as the condition to start tracing. If 'AND' or 'Sequential' has been selected as the condition to start tracing, only 'OR' is selectable as the event break condition.

[RX600 Series MCUs]

- A data-access event specifiable in a sequential condition is for only event numbers 1 to 3 or the reset event.
- In a sequential condition, a data-access event for which an address range has been defined is only specifiable for event number 1.
- The number of passes condition ([Pass Count]) is not specifiable for the reset event in a sequential condition.

[RX200 Series MCUs]

- A data-access event specifiable in a sequential condition is for only event numbers 1 and 2.

### 6.7.3 Number of Passes

[RX600 Series MCUs]

- The number of passes condition ([Pass Count]) is only specifiable for one event.

### 6.7.4 Data-Access Event with an Address Range Defined

[RX600 Series MCUs]

- A data-access event for which an address range condition has been defined is only specifiable for one event.

### 6.7.5 Registering Events

- If conditions on data access are registered for the [Before PC] event list, the settings will be ignored.
- If execution of the instruction at an address is registered for the [Trace Record] event list, the setting will be ignored.

[RX200 Series MCUs]

- If an event is registered by dragging-and-dropping an address range of the [Memory] window, no data-access event can be set on condition that access is made to the address range. Therefore, an event that has had only the start address set is added in an invalid state to the event list.

To register an event by dragging-and-dropping from the [Memory] window, select only one data in size of 1, 2 or 4 bytes.

### 6.7.6 Events for the WAIT Instruction

[RX630, RX631, RX63N, RX63T, RX210, RX21A, and RX220 Group MCUs]

- If a break occurs for an executed-address type of event that is set in the WAIT instruction or an instruction immediately preceding it, a time-out error may occur. To cause the target program to break in the WAIT instruction or an instruction immediately preceding it, use an on-chip breakpoint.

### 6.7.7 Event Resources

- Among the break resources displayed in the [Output] window or [Status] window, when the program is stopped at an on-chip break point (OR), the last number of an event resource displayed as “Event break at” (such as “PC0”) indicates the channel allocated to the event. Use the [On-Chip Break] dialog box to refer to the channel allocated to the event.

The event resource displayed by the “Event break at” above differs from the event number specified in <event> by the EVENT SET command.

## 6.8 Break Functions

### 6.8.1 Notes on Setting Break Points

- S/W break points replace the instructions at the corresponding addresses. They cannot be specified in the area other than the on-chip ROM and on-chip RAM.
- During stepped execution, specifications of S/W break points and on-chip breakpoints are invalid.
- When execution is restarted from the address where it stopped at a breakpoint (S/W breakpoints or pre-PC-execution breakpoints), the actual instruction at the address must be executed as a single step before further execution continues. Operation is thus not in real time.

Note that any event that occurred in this single-step execution is cleared when the program is executed thereafter. Therefore, if a trace or performance start event trigger is set in this step, neither trace acquisition nor performance measurement begins until the event occurs again. The same applies to break-and-trace start combinational events, so that any event that occurred in a step is not added to AND (cumulative), nor does it cause a state transition.

- On-chip breakpoints cannot be set while the emulator is in a power-down mode.
- If an on-chip breakpoint has been set through the [On-chip Break points] column of the [Editor] window in a power-down mode, the [Output] window shows either of the messages given below.

In the sleep mode:

"The internal clock is halted because the MCU is in the sleep mode."

In the all-module clock stop, software standby, or deep software standby mode:

"The internal clock is halted because the MCU is in the standby mode."

This makes setting of on-chip breakpoints through the [On-chip Break points] column impossible. For release from this state, open and then close the [On-Chip Break] dialog box (you do not have to change the settings in the dialog box).

## 6.9 Realtime RAM Monitoring

- Only access via the CPU bus can be monitored. No record of access by the DMAC or DTC is displayed.
- If a reset such as a pin reset or watchdog timer reset is issued while a user program is running, the results of subsequent RAM monitoring cannot be guaranteed (values displayed may be incorrect).
- The realtime RAM monitoring function rewrites the window values based on trace data, so that even when values are rewritten in the [I/O] or the [Memory] window, regardless of whether the user program is running or not, the changed values are not reflected in the [RAM Monitor] window.
- If a monitor range is set by the trace extraction function, the watch function cannot be used for the data that does not fit in the monitor range (those displayed in gray).

### 6.9.1 Displaying the Block Boundary Memory

- To display memory for one block overlapping a boundary or an area consisting of two or more consecutive blocks in the [RAM monitor] window, be sure that the display start address is a multiple of the display size. If these address and size do not match when you display memory, the displayed memory content is grayed.
- If any block preceding the one you registered in RAM monitor area settings is unregistered (grayed out), memory contents cannot be monitored correctly in the registered block, even when one of the following accesses is attempted in big endian mode.

Assuming that the start address of a registered block is address 'n'

- (1) Access in 2 and 4 bytes to the address  $n - 1$
- (2) Access in 4 bytes to the address  $n - 2$
- (3) Access in 4 bytes to the address  $n - 3$

### 6.9.2 Uninitialized Area Detection

- If a detection area is registered in the [Uninitialized area detection] dialog box, be aware that the output mode in the [Trace conditions] dialog box is switched to "Trace output."  
Note that if all of the registered detection areas are deleted, it switches back to "CPU execution."  
Also, if the output mode is manually changed after registering a detection area, the changed setting has priority when detection is performed.
- The areas set for uninitialized area detection and the RAM monitor area are the common shared resources.  
Therefore, if a block area not registered in the RAM monitor is registered for uninitialized area detection, this block is registered in the list of RAM monitor area settings, too.

## 6.10 Performance Measurement

### 6.10.1 Reset During Performance Measurement

Performance measurement proceeds even if the MCU is reset. However, the results may not be correct because the clock settings for measurement have been initialized.

### 6.10.2 Limitation on Nesting for the Performance Measurement

[RX600 Series MCUs]

The number of times interrupts or exceptions occur and the number of RTE or RTFI instructions must be the same, so obtaining this balance is a precondition. If the number of times interrupts or exceptions occur and the number of RTE or RTFI instructions are not the same, that is, balance has not been obtained, correct measurement of the number of cycles for processing is not possible.

Also, data can only be retained for 16 levels of nesting. Therefore, if nesting of interrupts or exceptions reaches the 17th level, correct measurement is not possible.

### 6.10.3 Notes on Checking the [Measure the performance only once.] Checkbox

[RX600 Series MCUs]

- While the measurement listed below is performed, even if the start event and end event occur, if the condition for measurement is not satisfied even once, the results of measurement will not be displayed.
  - Execution count
  - Exception and interrupt count
  - Exception count
  - Interrupt count
  - Event match count

Examples: No exceptions or interrupts occur within the range for measurement.

No event matches occur within the range for measurement.

- Measurement will be suspended when the start event for performance measurement occurs twice even though the end event has not occurred.



## 6.11 Downloading

### 6.11.1 About the Access Size

When downloading the modules, the access size can be specified. This leads to download (write) the modules to the memory by the specified access size.

Note, however, that the specifiable access size for the download module differs with each area.

Although the on-chip ROM/RAM areas accept an access size of "1", "2", or "4" (do not specify "8"), only the access size "1" can be selected for the download module that includes data for the external flash ROM area.

Note also that for the access to external RAM areas, the bus width specified for each of external areas in the [Configuration Properties] dialog box at startup has priority.

If you need to specify an access size of "2", set the bus width for the external RAM area to "16 bit" in the [Configuration Properties] dialog box at startup, divide the download module for the external flash ROM area into separate modules, and specify "2" for the module that does not include the external flash ROM area.

Also, as for the external flash ROM, the access size specified in the USD file will be used.

For details, refer to the manual of the External Flash Definition Editor described in the URL below.

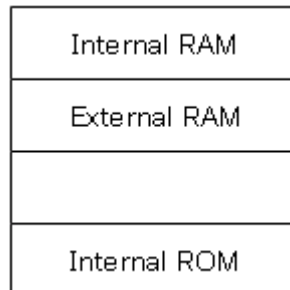
<http://www.renesas.com/efe>

Example: When setting the access size as "2" to the external RAM in the following program.

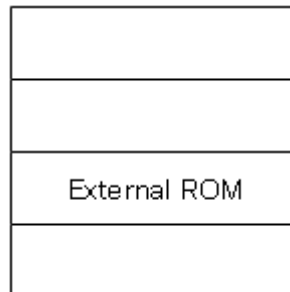
Internal RAM
External RAM
External ROM
Internal ROM

To download the modules, sort them into two modules (a) and (b) as shown below.

(a) When "2" is specified for the access size (For the external RAM access sizes, the bus width "16 bits" specified at startup has priority.):



(b) When "1" is specified for the access size:



## 6.12 Downloading to the External Flash Memory

- Only flash memory with 4096 or fewer sectors can be registered. If flash memory with more sectors is connected, programming cannot be guaranteed.
- A script specified by the USD file corresponds to the executable memory\_fill command of the High-performance Embedded Workshop debugger. However, note that some options are not available.

The format to be used for script is as follows.

mf<start><end><data>[<mode>]

<start> start address

<end> end address

<data> data value

<mode> BYTE (one byte)

<mode> WORD (two byte)

<mode> LONG (four byte)

Abbreviation: (same as BYTE)

- S/W breakpoints cannot be set in external address spaces.
- Data in external flash memory cannot be changed by using the [Memory] window or command line.
- Select "1" as the size for access to download modules.
- When downloading to the external flash memory, if you are using external RAM as working RAM, set it to the same endian as the CPU.
- If your device does support a lock command, select an option to clear lock-bits when creating a USD file. By doing this, the emulator can skip the process of checking for unnecessary lock-bits.

### 6.13 Execution Time

The execution time displayed in the status bar and the [Status] window is the time measured while the program is being executed. Values smaller than 100  $\mu$ s are discarded. The execution time will be incorrect when [Step In], [Step Over], or [Step Out] is performed.

### 6.14 Start/Stop Function

If, in the Start/Stop function mechanism, a transfer from the Start function to the user program does not occur or the Stop function does not stop, an error message "A timeout error has occurred in START/STOP function processing. Is system reset issued?" is displayed.

If the [Yes] button is clicked, the emulator is initialized and the user program stops. After a system reset is issued, the trace record is initialized too. Or, if the [No] button is clicked, the emulator is not initialized but the user program stops. Note that after you've clicked the [Yes] button, you can continue with debugging.

### 6.15 Watch Function

If a symbol is registered while the user program is under execution, the [Value] column of the [Watch] window shows a caution "Not available now," with the symbol unregistered. To register a symbol, be sure that the user program is halted.

### 6.16 Debug Console Function

To use the debug console function, do not switch the system clock in the charput and charget functions of the user program.

Switching of the system clock may adversely affect the communication between the emulator and MCU, hampering normal transmission or reception of data.

### 6.17 FINE Interface

- The external trace-output and realtime RAM monitoring functions via FINE interface are not supported.
- Hot plug-ins via FINE interface are not supported.

[RX630, RX631, RX63N, and RX63T Group MCUs]

- The FINE interface supports only a two-wire system making use of FINEC and MD/FINED pins. One-wire systems are not supported.

[RX210, RX21A, and RX220 Group MCUs]

- The FINE interface supports a one-wire system making use of MD/FINED pin.

## 6.18 Option Settings Related Registers

### 6.18.1 About the Endian Select Registers (MDEB, MDES)

[RX630, RX631, RX63N, RX63T, RX210, RX21A, and RX220 Group MCUs]

The endian select registers (MDEB, MDES) are such that endian information you set on the [Startup and Communication] page of the [Initial Settings] dialog box is written to the register corresponding to the operation mode in which to be debugged.

When debugging in the single-chip mode, you cannot rewrite the endian select register S (MDES) in the [Memory] window, etc. When debugging in the user boot mode, you cannot rewrite the endian select register B (MDEB) in the [Memory] window, etc.

### 6.18.2 About the Setting of Option Function Select Register 1 (OFS1)

[RX630, RX631, RX63N, RX63T, RX210, RX21A, and RX220 Group MCUs]

- When option function select register 1 (OFS1) is modified by an operation other than downloading (e.g. editing through the [Memory] window or line assembly), the following steps are necessary to make the new setting take effect.

- (1) Modify OFS1.
- (2) Start user program execution (write access to OFS1 occurs; for details, refer to section 6.1.2).
- (3) Reset the CPU.

When OFS1 is modified by downloading, the new setting takes effect when the CPU is reset after the downloading.

- Option function select register 1 (OFS1) allows you to enable or disable a voltage monitoring 0 reset and set a voltage detection level. This register is located in the flash memory, so be aware that depending on settings, it cannot be controlled by the E1/E20 emulator.

When the flash ROM contents are modified (by downloading or through the [Memory] window), if option function select register 1 (OFS1) is set to a condition that generates a voltage monitoring 0 reset, an error will occur. For details about the option function select register 1 (OFS1), refer to the hardware manual for the MCU you're using.

## 6.19 Other

### 6.19.1 Register Values after the On-Chip Flash Memory Has been Programmed

When programming of the on-chip flash memory takes place (e.g. downloading of a program, setting a S/W breakpoint, or modifying a value in the [Memory] window), the values of the flash-memory-related registers are also rewritten by the debugger.

### 6.19.2 DMAC and DTC

Do not attempt the following operations if a DMAC or DTC request with the MCU's internal flash ROM as the origin is generated while execution of the user program is stopped.

- Setting a S/W breakpoint in the MCU's internal flash ROM area
- Writing to the MCU's internal flash ROM area from the [Memory] window or [Command line] window
- Downloading to the MCU's internal flash ROM area

### 6.19.3 About the High-Speed Clock Oscillator (HOCO)

[RX210, RX21A, and RX220 Group MCUs]

- The emulator uses a device's internal high-speed clock oscillator (hereafter the HOCO) to achieve communications with the device.  
Therefore, the HOCO is always in an oscillating state no matter how the HOCO-related registers are set.
- If there is a contention between switching of the HOCO frequency and memory access, the memory access operation is not guaranteed.

### 6.19.4 About the Lock-Bits

#### (a) About Clearing of the Lock-Bit at the Time of Download

If the user program is to be downloaded into a flash ROM area for which a lock-bit is set, the emulator clears the lock-bit of the block that contains download data before it starts downloading.

#### (b) About Restoring of the Lock-Bit after Completion of a Download

Operation differs depending on how the [Debugging the program re-writing the internal PROGRAM ROM] checkbox of the [Configuration Properties] dialog box is set.

[When the checkbox is checked]

The block that contains download data for a program ROM area and those that contain no download data and are not registered in [Internal flash memory overwrite] as the ones to be overwritten have their lock-bits set back again after a download is completed.

[When the checkbox is unchecked]

The lock-bits remain cleared and are not reset after a download is completed.

### 6.19.5 About Disconnection of the Emulator

To disconnect the emulator, follow one of the methods described in section 2.7, Disconnecting the Emulator. Always be sure to disconnect the emulator this way.

If the debugger was forcibly terminated, the emulator cannot be started normally the next time you start. In that case, turn the power to the emulator off briefly and then on again. Do this, for the E20 emulator, by switching over the power switch, and for the E1 emulator, by removing the USB interface cable temporarily and then inserting again.

### 6.19.6 About the Operating Frequency

[RX630, RX631, RX63N, RX63T, RX210, RX21A, and RX220 Group MCUs]

The minimum operating frequency of the MCUs which can be debugged with the E1 and E20 emulators is 32.768 kHz.

### 6.19.7 About the MCU Containing the USB Boot Program

[RX630, RX631, RX63N, and RX63T Group MCUs]

To activate the debugger in user boot mode when using an MCU that contains the USB boot program, erase the USB boot program through the Flash Development Toolkit or Renesas Flash Programmer.

If the debugger is activated in user boot mode while the MCU contains the USB boot program, an activation error will occur.

### 6.19.8 About the Setting that Enables Clock Manipulation

[RX630, RX631, RX63N, RX63T, RX210, RX21A, and RX220 Group MCUs]

When clock manipulation is enabled through the [Configuration Properties] dialog box of the debugger, if internal flash ROM is rewritten by the debugger while the FlashIF clock (FCLK) of the target MCU is outside of the guaranteed operating range (that is, while operating with LOCO or subclock), the E1/E20 emulator will switch the clock source.

After rewriting to the internal flash ROM is completed, the clock will be restored to the previous clock source.

Note that the operating frequency of the peripheral clock will change during on-chip flash memory rewriting because the clock source is switched.

The clock manipulation enabling setting takes effect when the internal flash ROM is rewritten after program execution or step execution. Note that the clock source is forcibly switched regardless of the clock manipulation enabling setting if FCLK is outside of the guaranteed operating range immediately after the debugger is activated or when the [CPUReset] button is clicked.

### 6.19.9 About Access to the MPU Area

[MCU that has an MPU area]

The MPU area can be referred to and written to only while program execution is stopped.

If an attempt is made to access the MPU area during program execution,  
the read data will be shown as 0x00, or  
an error will occur if write access is attempted.

The following operations cannot be done in the MPU area.

- Line assembly
- Memory search
- Memory comparison
- Memory copy

## Appendix A Menus

Tabel A. 1 shows GUI menus.

**Tabel A. 1 GUI Menus**
























Menu	Option	Shortcut	Toolbar Button	Remarks	
View	Difference			Opens the [Difference] window.	
	Map			Opens the [Map Section Information] window.	
	Command Line	Ctrl + L		Opens the [Command Line] window.	
	TCL toolkit	Ctrl + Shift + K		Opens the [Console] window.	
	Workspace	Alt + K		Opens the [Workspace] window.	
	Output	Alt + O		Opens the [Output] window.	
	Status bar	Alt + A		Shows or hides the status bar.	
	Disassembly	Ctrl + D		Opens the [Disassembly] window.	
	CPU	Registers	Ctrl + R		Opens the [Register] window.
		Memory...	Ctrl + M		Opens the [Memory] window.
IO		Ctrl + I		Opens the [IO] window.	
Status		Ctrl + U		Opens the [Status] window.	
RAM Monitor				Opens the [RAM Monitor] window.	
Debug Console				Opens the [DebugConsole] window.	
Event		On-chip Break			Opens the [On-Chip Break] dialog box.
	Trace Conditions			Opens the [Trace conditions] dialog box.	
	Performance			Opens the [Performance] dialog box.	
Symbol	Labels	Ctrl + Shift + A		Opens the [Labels] window.	
	Watch	Ctrl + W		Opens the [Watch] window.	
	Locals	Ctrl + Shift + W		Opens the [Locals] window.	
Graphic	Image...	Ctrl + Shift + G		Opens the [Image Properties] dialog box.	
	Waveform...	Ctrl + Shift + V		Opens the [Waveform Properties] dialog box.	
	Graph			Opens the [Graph] window.	



Table A.1 GUI Menus (cont)























Menu	Option	Shortcut	Toolbar Button	Remarks	
View (cont)	Code	Trace	Ctrl + T		Opens the [Trace] window.
		Stack Trace	Ctrl + K		Opens the [Stack Trace] window.
	Performance	Performance Analysis	Ctrl + Shift + P		Opens the [Performance Analysis] window.
Debug	Synchronized Debugging...				Opens the [Synchronized Debug] dialog box, in which settings for synchronized debugging can be made.
	Debug Sessions...				Opens the [Debug Sessions] dialog box for listing, adding, or removing debug sessions.
	Debug Settings...				Opens the [Debug Settings] dialog box to set the debugging conditions or load modules.
	Reset CPU				Resets the target MCU and sets the PC to the reset vector address.
	Go	F5			Starts running the user program from the location currently indicated by the PC.
	Reset Go	Shift + F5			Resets the target MCU and executes the user program from the reset vector address.
	Free Go				Starts running the user program with all breakpoints ignored.
	Go To Cursor				Starts running the user program from the address currently indicated by the PC until the PC reaches the address indicated by the current text cursor position.
	Set PC To Cursor				Sets the PC to the address at the row of the text cursor.
	Run...				Launches the [Run Program] dialog box allowing the user to enter the PC or PC breakpoint during executing the user program.
	Display PC	Ctrl + Shift + Y			Displays the current PC value in the [Editor] window.
	Step In	F11			Executes a block of user program before breaking.
	Step Over	F10			Executes a block of user program before breaking. If a subroutine call is reached, then the subroutine will not be entered.
	Step Out	Shift + F11			Executes the user program to reach the end of the current function.

Table A.1 GUI Menus (cont)

Menu	Option	Shortcut	Toolbar Button	Remarks	
Debug (cont)	Step...			Launches the [Step Program] dialog box allowing the user to modify the settings for stepping.	
	Step Mode	Auto		Steps only one source line when the [Editor] window is active. When the [Disassembly] window is active, stepping is executed in a unit of assembly instructions.	
		Assembly		Executes stepping in a unit of assembly instructions.	
		Source		Steps only one source line.	
	Halt Program	Esc		Stops the execution of the user program.	
	Initialize			Disconnects and then restarts the debugging platform.	
	Connect			Connects the debugging platform.	
	Disconnect			Disconnects the debugging platform.	
	Save Memory...			Saves the specified range of data from memory data in a file (.bin, .hex, or .mot).	
	Verify Memory...			Verifies file contents against data in memory.	
Download Modules			Downloads the object program.		
Unload Modules			Unloads the object program.		
Setup	Radix	Hexadecimal		Uses a hexadecimal for displaying a radix in which the numerical values will be displayed and entered by default.	
		Decimal		Uses a decimal for displaying a radix in which the numerical values will be displayed and entered by default.	
		Octal		Uses an octal for displaying a radix in which the numerical values will be displayed and entered by default.	
		Binary		Uses a binary for displaying a radix in which the numerical values will be displayed and entered by default.	
	Emulator	Device setting...			Opens the [Initial Settings] dialog box, in which settings for the target MCU can be made.
		System...			Opens the [Configuration Properties] dialog box allowing the user to modify the debugging platform settings.
		Start/stop Function Setting...			Opens the [Start/Stop function setting] dialog box.

## Appendix B Notes on High-performance Embedded Workshop

### (1) Note on Moving Source File Position after Creating Load Module

When the source file is moved after creating the load module, the [Open] dialog box may be displayed to specify the source file during the debugging of the created load module. Select the corresponding source file and click on the [Open] button.

### (2) Source-Level Execution

- Source file

Do not display source files that do not correspond to the load module in the program window. For a file having the same name as the source file that corresponds to the load module, only its addresses are displayed in the program window. The file cannot be operated in the program window.

- Step

Even standard C libraries are executed. To return to a higher-level function, use [Step Out]. You can also select not to step into addresses where no debugging information exists. Open the [Options] dialog box via [Setup -> Options] and select the [Only step in when debug information is available] checkbox on the [Debug] page.

In a for statement or a while statement, executing a single step does not move execution to the next line. To move to the next line, execute two steps.

### (3) Operation During Accessing Files

Do not perform other operations during downloading the load module, operating [Verify Memory] or [Save Memory] in the [Memory] window, or saving in the [Trace] window because this will not allow correct file accessing to be performed.

### (4) Watch

- Local variables at optimization

Depending on the generated object code, local variables in a source file that is compiled with the optimization option enabled will not be displayed correctly. Check the generated object code by displaying the [Disassembly] window.

If the allocation area of the specified local variable does not exist, displays as follows.

Example:           The variable name is asc.

asc    Not available now.

- Variable name specification

When a name other than a variable name, such as a symbol name or function name, is specified, no data is displayed.

Example:           The function name is main.

main   Not available now.

## (5) Command Line Interface

## • Batch file

To display the message “Not currently available” while executing a batch file, enter the sleep command. Adjust the sleep time length which differs depending on the operating environment.

Example: To display “Not currently available” during memory\_fill execution:

```
sleep d'3000  
  
memory_fill 0 ffff 0
```

## • File specification by commands

The current directory may be altered by file specifications in commands. It is recommended to use absolute paths are recommended to be used to specify the files in a command file so that the current directory alteration is not affected.

Example: FILE\_LOAD C:\Workspace\Tutorial\E1E20\RX600\Tutorial\Tutorial\_LittleEndian  
\Debug\_RXxxx\_E1\_E20\_SYSTEM\tutorial.abs

## (6) Loading of Motorola S-type Files

This HEW does not support Motorola S-type files with only the CR code (0Dh) at the end of each record. Load Motorola S-type files with the CR and LF codes (0D0Ah) at the end of each record.

## (7) Note on [Register] Window Operation During Program Execution

The register value cannot be changed in the [Register] window during program execution. Even if the changed value is displayed, the register contents are not changed actually.

## (8) Note on RUN-TIME Display

Although the execution time for the user program is displayed on the status bar and in the [Status] window, values are rounded down to the nearest 100  $\mu$ s (i.e. values less than 100  $\mu$ s are discarded) since an internal 32-bit counter of the emulator is used for this measurement. Values are also not accurate during single stepping, [Step Over], or [Step Out].

## (9) Memory Test Function

This product does not support the memory test function, which is activated by selecting [Test...] from the [Memory] menu.

## (10) “Writing the On-Chip Flash Memory” Mode

- When MCUs are to be continuously programmed, be sure to turn the target system on or off.
- Debugging functions other than downloading of a program are not usable in the “writing the on-chip flash memory” mode.

## (11) Disassembled Display

The [Editor] window in the disassembly mode displays undefined codes as ‘???’.

---

E1/E20 Emulator

Additional Document for User's Manual:

High-performance Embedded Workshop RX Debug

Publication Date: Rev.1.00 Sep 01, 2012

Published by: Renesas Electronics Corporation

---

**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.**  
2880 Scott Boulevard Santa Clara, CA 95050-2554, 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: +44-1628-651-700, Fax: +44-1628-651-804

**Renesas Electronics Europe GmbH**  
Arcadiastrasse 10, 40472 Düsseldorf, Germany  
Tel: +49-211-65030, Fax: +49-211-6503-1327

**Renesas Electronics (China) Co., Ltd.**  
7th Floor, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100083, P.R.China  
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

**Renesas Electronics (Shanghai) Co., Ltd.**  
Unit 204, 205, AZIA Center, No.1233 Lujiazui Ring Rd., Pudong District, Shanghai 200120, China  
Tel: +86-21-5877-1818, Fax: +86-21-6887-7858 / -7898

**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: +852-2886-9318, Fax: +852-2886-9022/9044

**Renesas Electronics Taiwan Co., Ltd.**  
13F, No. 363, Fu Shing North Road, Taipei, 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.**  
11F., Samik Lavied'or Bldg., 720-2 Yeoksam-Dong, Kangnam-Ku, Seoul 135-080, Korea  
Tel: +82-2-558-3737, Fax: +82-2-558-5141

E1/E20 Emulator  
Additional Document for User's Manual:  
High-performance Embedded Workshop RX Debug

