

R0E556100MCU00

User's Manual

E100 Emulator MCU Unit for RX610 Group

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 Corp. without notice. Please review the latest information published by Renesas Electronics Corp. through various means, including the Renesas Electronics Corp. 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 or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving 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, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. 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 shall be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
6. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The intended 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; industrial robots; etc.

"High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or 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 damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user's manual or other Renesas Electronics document.

7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION ("Vulnerability Issues"). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. When using Renesas Electronics products, refer to the latest product information (data sheets, user's manuals, application notes, "General Notes for Handling and Using Semiconductor Devices" in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, 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 and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. Renesas Electronics products and technologies shall 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 shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1 October 2020)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan

www.renesas.com

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/.

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)

A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity.

Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.

2. Processing at power-on

The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.

3. Input of signal during power-off state

Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.

4. Handling of unused pins

Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.

5. Clock signals

After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.

6. Voltage application waveform at input pin

Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).

7. Prohibition of access to reserved addresses

Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.

8. Differences between products

Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Regulatory Compliance Notices

European Union regulatory notices

This product complies with the following EU Directives. (These directives are only valid in the European Union.)

CE Certifications:

- Electromagnetic Compatibility (EMC) Directive 2014/30/EU
EN 55032 Class A

WARNING: This is a Class A product. This equipment can cause radio frequency noise when used in the residential area. In such cases, the user/operator of the equipment may be required to take appropriate countermeasures under his responsibility.

EN 55024

- Information for traceability
 - Authorized representative
Name: Renesas Electronics Corporation
Address: TOYOSU FORESIA, 3-2-24, Toyosu, Koto-ku, Tokyo 135-0061, Japan
 - Person responsible for placing on the market
Name: Renesas Electronics Europe GmbH
Address: Arcadiastrasse 10, 40472 Dusseldorf, Germany
 - Trademark and Type name
Trademark: Renesas
Product name: E100 Emulator MCU Unit
Type name: R0E556100MCU00

Environmental Compliance and Certifications:

- Waste Electrical and Electronic Equipment (WEEE) Directive 2012/19/EU

United States Regulatory notices on Electromagnetic compatibility

FCC Certifications (United States Only):

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense.

This device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions: (1) this device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.

CAUTION: Changes or modifications not expressly approved by the party responsible for compliance could void the user's authority to operate the equipment.

Preface

The R0E556100MCU00 is a full-spec emulator for MCU of the RX610 Groups. This user's manual mainly describes specifications of the R0E556100MCU00 and how to setup it.

All components of the R0E556100MCU00 are listed under "Package Components" (page 17). If you have any questions about the R0E556100MCU00, contact your local distributor.

The manuals relevant to usage of the R0E556100MCU00 are listed below. You can download the latest manuals from the Renesas Tools homepage (<https://www.renesas.com/tools>).

Related manuals

Item	Manual
Accessory	R0E556100CFK00 User's Manual
Integrated development environment	High-performance Embedded Workshop User's Manual
C/C++ compiler and assembler	RX Family Compiler Package User's Manual

Important

Before using this product, be sure to read this user's manual carefully.

Keep this user's manual, and refer to it when you have questions about this product.

Emulator:

"Emulator" in this document collectively refers to the following products manufactured by Renesas Electronics Corporation.

- (1) E100 emulator main unit
- (2) MCU unit
- (3) Pitch converter board for connecting the user system

"Emulator" herein encompasses neither the customer's user system nor the host machine.

Purpose of use of the emulator:

This emulator is a device to support the development of systems that use the RX Family RX610 Group of Renesas 32-bit single-chip MCUs. It provides support for system development in both software and hardware.

Be sure to use this emulator correctly according to said purpose of use. Please avoid using this emulator other than for its intended purpose of use.

For those who use this emulator:

This emulator can only be used by those who have carefully read the user's manual and know how to use it.

Use of this emulator requires basic knowledge of electric circuits, logical circuits, and MCUs.

When using the emulator:

- (1) This product is a development-support unit for use in your program development and evaluation stages. When a program you have finished developing is to be incorporated in a mass-produced product, the judgment as to whether it can be put to practical use is entirely your own responsibility, and should be based on evaluation of the device on which it is installed and other experiments.
- (2) In no event shall Renesas Electronics Corporation be liable for any consequence arising from the use of this product.
- (3) Renesas Electronics Corporation strives to provide workarounds for and correct trouble with products malfunctions, with some free and some incurring charges. However, this does not necessarily mean that Renesas Electronics Corporation guarantees the provision of a workaround or correction under any circumstances.
- (4) The product covered by this document has been developed on the assumption that it will be used for program development and evaluation in laboratories. Therefore, it does not fall within the scope of applicability of the Electrical Appliance and Material Safety Law and protection against electromagnetic interference when used in Japan.
- (5) Renesas Electronics Corporation cannot predict all possible situations and possible cases of misuse that carry a potential for danger. Therefore, the warnings in this user's manual and the warning labels attached to the emulator do not necessarily cover all such possible situations and cases. The customer is responsible for correctly and safely using this emulator.
- (6) The product covered by this document has not been through the process of checking conformance with UL or other safety standards and IEC or other industry standards. This fact must be taken into account when the product is taken from Japan to some other country.

Usage restrictions:

The emulator has been developed as a means of supporting system development by users. Therefore, do not use it as an embedded device in other equipment. Also, do not use it to develop systems or equipment for use in the following fields.

- (1) Transportation and vehicular
- (2) Medical (equipment that has an involvement in human life)
- (3) Aerospace
- (4) Nuclear power control
- (5) Undersea repeaters

If you are considering the use of the emulator for one of the above purposes, please be sure to consult your local distributor.

About product changes:

We are constantly making efforts to improve the design and performance of this emulator. Therefore, the specification or design of this emulator, or this user's manual, may be changed without prior notice.

About rights:

- (1) We assume no responsibility for any damage or infringement on patent rights or any other rights arising from the use of any information, products or circuits presented in this user's manual.
- (2) The information or data in this user's manual does not implicitly or otherwise grant a license to patent rights or any other rights belonging to Renesas or to a third party.
- (3) This user's manual and this emulator are copyrighted, with all rights reserved by Renesas. This user's manual may not be copied, duplicated or reproduced, in whole or part, without prior written consent from Renesas.

About diagrams:

Some diagrams in this user's manual may differ from the objects they represent.

Precautions for Safety

Definitions of Signal Words

In both the user's manual and on the product itself, several icons are used to insure proper handling of this product and also to prevent injuries to you or other persons, or damage to your properties.

This chapter describes the precautions which should be taken in order to use this product safely and properly. Be sure to read this chapter before using this product.



This symbol represents a warning about safety. It is used to arouse caution about a potential danger that will possibly inflict an injury on persons. To avoid a possible injury or death, please be sure to observe the safety message that follows this symbol.



DANGER indicates an imminently dangerous situation that will cause death or heavy wound unless it is avoided. However, there are no instances of such danger for the product presented in this user's manual.



WARNING indicates a potentially dangerous situation that will cause death or heavy wound unless it is avoided.



CAUTION indicates a potentially dangerous situation that will cause a slight injury or a medium-degree injury unless it is avoided.

CAUTION

CAUTION with no safety warning symbols attached indicates a potentially dangerous situation that will cause property damage unless it is avoided.

IMPORTANT

This is used in operation procedures or explanatory descriptions to convey exceptional conditions or cautions to the user.

In addition to the five above, the following are also used as appropriate.

△means WARNING or CAUTION.

Example:



CAUTION AGAINST AN ELECTRIC SHOCK

⊘means PROHIBITION.

Example:



DISASSEMBLY PROHIBITED

●means A FORCIBLE ACTION.

Example:



UNPLUG THE POWER CABLE FROM THE RECEPTACLE.

WARNING

Warnings for AC Power Supply:



- If the AC power cable does not fit the receptacle, do not alter the AC power cable and do not plug it forcibly. Failure to comply may cause electric shock and/or fire.
- Do not touch the plug of the AC power cable when your hands are wet. This may cause electric shock.
- This product is connected signal ground with frame ground. If your developing product is transformless (not having isolation transformer of AC power), this may cause electric shock. Also, this may give an unrepairable damage to this product and your developing one.

While developing, connect AC power of the product to commercial power through isolation transformer in order to avoid these dangers.



- If other equipment is connected to the same branch circuit, care should be taken not to overload the circuit.
- When installing this equipment, insure that a reliable ground connection is maintained.



- If you smell a strange odor, hear an unusual sound, or see smoke coming from this product, then disconnect power immediately by unplugging the AC power cable from the outlet.

Do not use this as it is because of the danger of electric shock and/or fire. In this case, contact your local distributor.

- Before setting up this emulator and connecting it to other devices, turn off power or remove a power cable to prevent injury or product damage.

Warnings to Be Taken for This Product:



- Do not disassemble or modify this product. Personal injury due to electric shock may occur if this product is disassembled and modified. Disassembling and modifying the product will void your warranty.
- Make sure nothing falls into the cooling fan on the top panel, especially liquids, metal objects, or anything combustible.

Warning for Installation:



- Do not set this product in water or areas of high humidity. Make sure that the product does not get wet. Spilling water or some other liquid into the product may cause unrepairable damage.

Warning for Use Environment:



- This equipment is to be used in an environment with a maximum ambient temperature of 35°C. Care should be taken that this temperature is not exceeded.

CAUTION

Caution for AC Adapter:



- The DC plug of the AC adapter has the polarity shown below.



- Use an AC adapter which complies with the safety standards of the country where it is to be used.

Cautions to Be Taken for Turning On the Power:



- Turn ON/OFF the power of the emulator and user system as simultaneously as possible.
- When turning on the power again after shutting off the power, wait about 10 seconds.

Cautions to Be Taken for Handling This Product:



- Use caution when handling the main unit. Be careful not to apply a mechanical shock.
- Do not touch the connector pins of the emulator main unit and the target MCU connector pins directly. Static electricity may damage the internal circuits.
- Do not pull this emulator by the communications interface cable or the flexible cable. And, excessive flexing or force may break conductors.
- Do not flex the flexible cable excessively. The cable may cause a break.
- Do not use inch-size screws for this equipment. The screws used in this equipment are all ISO (meter-size) type screws. When replacing screws, use same type screws as equipped before.
- Do not tape the flexible cable or apply adhesives to secure the cable. The shielding material on the surface of the cable may come off.

Caution to Be Taken for System Malfunctions:



- If the emulator malfunctions because of interference like external noise, shut OFF the emulator once and then reactivate it.

Caution to Be Taken for Disposal:



- Penalties may be applicable for incorrect disposal of this waste, in accordance with your national legislation.

European Union regulatory notices:



The WEEE (Waste Electrical and Electronic Equipment) regulations put responsibilities on producers for the collection and recycling or disposal of electrical and electronic waste. Return of WEEE under these regulations is applicable in the European Union only. This equipment (including all accessories) is not intended for household use. After use the equipment cannot be disposed of as household waste, and the WEEE must be treated, recycled and disposed of in an environmentally sound manner.



Renesas Electronics Europe GmbH can take back end of life equipment, register for this service at "<http://www.renesas.eu/weee>".

Table of Contents

	Page
Preface.....	4
Important.....	5
Precautions for Safety.....	7
User Registration	15
Terminology	16
1. Outline.....	17
1.1 Package Components.....	17
1.2 Other Tool Products Required for Development.....	17
1.3 System Configuration.....	18
1.3.1 System Configuration	18
1.3.2 Names and Functions of Each Part of the Emulator.....	19
1.4 Specifications.....	21
1.5 Operating Environment	22
1.6 Specifications of the AC Adapter	23
2. Setup.....	24
2.1 Flowchart of Starting Up the Emulator	24
2.2 Installing the Included Software	26
2.3 Connecting the MCU Unit to and Disconnecting it from the E100 Emulator Main Unit	27
2.4 Connecting the Host Machine	29
2.5 Connecting the Emulator Power Supply	30
2.6 Turning On the Power.....	31
2.6.1 Checking the Connections of the Emulator System.....	31
2.6.2 Turning the Power On and Off	31
2.7 Self-Checking.....	32
2.8 Selecting the Clock Supply	33
2.8.1 Clock Source	33
2.8.2 Using an Internal Oscillator Circuit Board	34
2.8.3 Using the Oscillator Circuit on the User System	36
2.8.4 Using the Internal Generator Circuit.....	36
2.9 Connecting the User System	38
2.9.1 Connecting to a 144-Pin 0.5-mm Pitch Foot Pattern	39
3. Tutorial.....	40
3.1 Introduction	40
3.2 Starting the High-performance Embedded Workshop	41
3.3 Connecting the Emulator	42
3.4 Downloading the Tutorial Program	43
3.4.1 Downloading the Tutorial Program.....	43
3.4.2 Displaying the Source Program	44
3.5 Setting Software Breakpoints.....	45
3.6 Executing the Program	46
3.6.1 Resetting the CPU.....	46
3.6.2 Executing the Program.....	46

3.7	Checking Breakpoints	47
3.7.1	Checking Breakpoints	47
3.8	Altering Register Contents	48
3.9	Referencing Symbols	49
3.10	Checking Memory Contents.....	50
3.11	Referencing Variables.....	51
3.12	Showing Local Variables.....	53
3.13	Single-Stepping a Program.....	53
3.13.1	Executing the [Step In] Command	54
3.13.2	Executing the [Step Out] Command.....	55
3.13.3	Executing the [Step Over] Command.....	56
3.14	Forcibly Breaking a Program	57
3.15	Hardware Break Facility	58
3.15.1	Stopping a Program when It Executes a Specified Address.....	58
3.16	Stopping a Program when It Accesses Memory	59
3.17	Trace Facility	60
3.17.1	Showing the Trace Information Acquired by Fill until Stop	61
3.17.2	Showing the Trace Information Acquired by Fill around TP	64
3.17.3	Showing a Function Execution History.....	66
3.17.4	Filter Facility	68
3.18	Stack Trace Facility	70
3.19	What Next?	71
4.	Preparing to Debug.....	72
4.1	Starting the High-performance Embedded Workshop	72
4.2	Creating a New Workspace (Toolchain Unused).....	73
4.3	Creating a New Workspace (Toolchain Used).....	75
4.4	Opening an Existing Workspace	78
4.5	Connecting the Emulator	79
4.5.1	Connecting the Emulator.....	79
4.5.2	Reconnecting the Emulator	79
4.6	Disconnecting the Emulator	80
4.6.1	Disconnecting the Emulator	80
4.7	Quitting the High-performance Embedded Workshop	80
4.8	Setting Up the Debug.....	81
4.8.1	Specifying a Download Module.....	81
4.8.2	Setting Up Automatic Execution of Command Line Batch Files	82
5.	Debugging Functions	83
5.1	Setting Up the Emulation Environment	85
5.1.1	Setting Up the Emulator at Startup	85
5.1.2	Setting Up the Target MCU	86
5.1.3	Making the External Bus Related Setting	88
5.1.4	Setting Up the System.....	89
5.1.5	Setting Up Flash ROM Overwrite	91
5.1.6	Setting the Warning of Exceptional Events	92
5.1.7	Registering Information on the External Flash Memory	93
5.1.8	Showing Progress in Boot-up Processing.....	95
5.2	Downloading a Program	97
5.2.1	Downloading a Program.....	97
5.2.2	Showing the Source Code	97
5.2.3	Turning Columns in All Source Files Off	99
5.2.4	Turning Columns in One Source File Off	99
5.2.5	Showing Assembly Language Code	100

5.2.6	Correcting Assembly Language Codes.....	102
5.3	Displaying Memory Contents in Real Time.....	103
5.3.1	Displaying Memory Contents in Real Time	103
5.3.2	Switching the Displayed Bus	104
5.3.3	Setting RAM Monitor Update Intervals.....	104
5.3.4	Clearing RAM Monitor Access History	105
5.3.5	Clearing RAM Monitor Error Detection Data	105
5.4	Showing the Current Status	106
5.4.1	Showing the Emulator Status.....	106
5.4.2	Showing the Emulator State in the Status Bar	107
5.5	Periodically Reading Out and Showing the Emulator Status.....	108
5.5.1	Periodically Reading Out and Showing the Emulator Information	108
5.5.2	Selecting the Items to Be Displayed	109
5.6	Using Software Breakpoints.....	110
5.6.1	Using Software Breakpoints.....	110
5.6.2	Adding/Removing Software Breakpoints.....	110
5.6.3	Enabling/Disabling Software Breakpoints	111
5.7	Using Events	113
5.7.1	Using Events	113
5.7.2	Adding Events	114
5.7.3	Removing Events	119
5.7.4	Registering Events	121
5.7.5	Entering Events Each Time or Reusing Events	123
5.7.6	Applying Events.....	124
5.8	Setting Hardware Break Conditions.....	125
5.8.1	Setting Hardware Break Conditions	125
5.8.2	Setting Hardware Breakpoints	125
5.8.3	Saving/Loading the Set Contents of Hardware Breaks.....	128
5.9	Viewing Trace Information	129
5.9.1	Viewing Trace Information	129
5.9.2	Acquiring Trace Information	129
5.9.3	Setting Trace Information Acquisition Conditions	131
5.9.4	Setting Trace Modes	133
5.9.5	Setting Trace Points	135
5.9.6	Setting Capture/Do not Capture Conditions.....	139
5.9.7	Selecting the Content of Trace Acquisition	140
5.9.8	Showing Trace Results	141
5.9.9	Filtering Trace Information	143
5.9.10	Searching for Trace Records	144
5.9.11	Saving Trace Information to Files	145
5.9.12	Loading Trace Information from Files	146
5.9.13	Temporarily Stopping Trace Information Acquisition	146
5.9.14	Restarting Trace Information Acquisition	146
5.9.15	Switching Timestamp Display	146
5.9.16	Showing the History of Function Execution	147
5.9.17	Showing the History of Task Execution.....	148
5.10	Measuring Performance.....	149
5.10.1	Measuring Performance	149
5.10.2	Showing the Result of Performance Measurement	149
5.10.3	Setting Performance Measurement Conditions	150
5.10.4	Starting Performance Measurement	152
5.10.5	Clearing Performance Measurement Conditions	153
5.10.6	Clearing the Performance Measurement Result.....	153
5.10.7	About the Maximum Measurement Time of Performance	153

5.11	Acquiring Code Coverage	154
5.11.1	Acquiring Code Coverage	154
5.11.2	Opening the [Code Coverage] Window.....	154
5.11.3	Allocating Code Coverage Memory (Hardware Resource).....	155
5.11.4	Code Coverage in an Address Range	157
5.11.5	Adding Address Ranges	159
5.11.6	Changing Address Ranges	160
5.11.7	Removing Address Ranges	161
5.11.8	Code Coverage in a Source File	163
5.11.9	Adding Source Files	165
5.11.10	Removing Source Files	166
5.11.11	Showing Percentages and Graphs	167
5.11.12	Using the Sort Function.....	168
5.11.13	Searching for Unexecuted Lines	170
5.11.14	Clearing Code Coverage Information.....	171
5.11.15	Updating Coverage Information	171
5.11.16	Preventing Update of Coverage Information.....	171
5.11.17	Saving the Code Coverage Information to a File	172
5.11.18	Loading Code Coverage Information from a File	172
5.11.19	Coverage Information File Load Modes	173
5.11.20	Displaying Code Coverage Information in the [Editor] Window	175
5.12	Acquiring Data Coverage	176
5.12.1	Acquiring Data Coverage	176
5.12.2	Opening the [Data Coverage] Window.....	176
5.12.3	Allocating Data Coverage Memory (Hardware Resource).....	177
5.12.4	Data Coverage in an Address Range	179
5.12.5	Adding Address Ranges	180
5.12.6	Changing Address Ranges	181
5.12.7	Removing Address Ranges	182
5.12.8	Data Coverage in a Section	183
5.12.9	Adding Sections	184
5.12.10	Removing Sections	185
5.12.11	Data Coverage in a Task Stack	186
5.12.12	Clearing Data Coverage Information.....	187
5.12.13	Updating Coverage Information	187
5.12.14	Preventing Update of Coverage Information.....	187
5.12.15	Saving the Data Coverage Information to a File	188
5.12.16	Loading Data Coverage Information from a File	188
5.13	Viewing Realtime Profile Information	190
5.13.1	Viewing Realtime Profile Information	190
5.13.2	Setting Realtime Profile Measurement Modes.....	191
5.13.3	Measuring Function Profiles.....	192
5.13.4	Setting Function Profile Measurement Ranges	193
5.13.5	Saving Function Profile Measurement Ranges.....	194
5.13.6	Loading Function Profile Measurement Ranges	194
5.13.7	Measuring Task Profiles.....	195
5.13.8	Setting Task Profile Measurement Ranges	196
5.13.9	Saving Task Profile Measurement Tasks.....	197
5.13.10	Loading Task Profile Measurement Tasks.....	197
5.13.11	Clearing Realtime Profile Measurement Results	198
5.13.12	Saving Realtime Profile Measurement Results.....	198
5.13.13	Setting the Measurement Interval	198
5.13.14	Maximum Measurement Time of the Realtime Profile	199
5.14	Detecting Exceptional Events	200

5.14.1	Detecting Exceptional Events	200
5.14.2	Detecting an Access Protect Violation	200
5.14.3	Setting an Access Protected Area	202
5.14.4	Detecting Reading from Uninitialized Memory	206
5.14.5	Setting Areas where Reading from Uninitialized Memory is to be Detected	207
5.14.6	Detecting a Performance Overflow	210
5.14.7	Detecting a Realtime Profile Overflow	210
5.14.8	Detecting a Trace Memory Overflow.....	211
5.14.9	Detecting a Task Stack Access Violation.....	211
5.14.10	Setting a Task Stack Area.....	212
5.14.11	Detecting an OS Dispatch.....	215
5.15	Using the Start/Stop Function	216
5.15.1	Opening the [Start/Stop Function Setting] Dialog Box.....	216
5.15.2	Specifying the Work Address	216
5.15.3	Specifying the Routine to be Executed	216
5.15.4	Limitations on the Start/Stop Function	217
5.15.5	Limitations on the Statements Written in a Specified Routine	217
5.16	Using the Trigger Output Function.....	218
5.16.1	Using the External Trigger Cable for Output.....	218
5.16.2	Opening the [Trigger Output Conditions] Dialog Box.....	219
5.16.3	Manual Setting for Output through Trigger Pins 31 to 24	220
5.16.4	Setting for Output through Trigger Pins 20 to 16	221
5.16.5	Events	222
6.	Troubleshooting (Action in Case of an Error)	223
6.1	Flowchart for Remediation of Trouble.....	223
6.2	Error in Self-Checking.....	224
6.3	Errors Reported in Booting-Up of the Emulator	225
6.4	How to Request Support.....	228
7.	Hardware Specifications	229
7.1	Target MCU Specifications	229
7.2	Differences between the Actual MCU and Emulator	230
7.3	Connection Diagram	231
7.3.1	Connection Diagram for R0E556100MCU00.....	231
7.4	External Dimensions	232
7.4.1	External Dimensions of the E100 Emulator	232
7.4.2	External Dimensions of the Converter Board R0E556100CFK00	233
7.5	Notes on Using the MCU Unit.....	234
8.	Maintenance and Warranty	237
8.1	User Registration	237
8.2	Maintenance.....	237
8.3	Warranty.....	237
8.4	Repair Provisions	237
8.5	How to Make Request for Repair.....	238

User Registration

When you install debugger software, a text file for user registration is created on your PC. Fill it in and email it to your local distributor. If you have replaced an emulator main unit or emulation probe, rewrite an emulator name and serial number in the text file you filled in earlier to register your new hardware products.

Your registered information is used for only after-sale services, and not for any other purposes. Without user registration, you will not be able to receive maintenance services such as a notification of field changes or trouble information. So be sure to carry out the user registration.

For more information about user registration, please contact your local distributor.

Terminology

Some specific words used in this user's manual are defined below.

MCU unit (R0E556100MCU00)

This means the E100 emulator for the RX610 Group.

Emulator system

This means an emulator system built around the MCU unit (R0E556100MCU00). The emulator system is configured with an emulator main unit (R0E001000EMU00), MCU unit (R0E556100MCU00), emulator power supply, USB cable, emulator debugger and host machine.

Integrated development environment: High-performance Embedded Workshop

This tool provides powerful support for the development of embedded applications for Renesas microcomputers. It has an emulator debugger function allowing the emulator to be controlled from the host machine via an interface. Furthermore, it permits a range of operations from editing a project to building and debugging it to be performed within the same application. In addition, it supports version management.

Emulator debugger

This means a software tool that is started up from the High-performance Embedded Workshop, and controls the MCU unit and enables debugging.

Firmware

This means a control program stored in the emulator. This analyzes the contents of communications with the emulator debugger and controls the emulator hardware. To upgrade the firmware, download the program from the emulator debugger.

Host machine

This means a personal computer used to control the emulator.

Target MCU

This means the MCU to be debugged.

User system

This means a user's application system in which the MCU to be debugged is used.

User program

This means the program to be debugged.

Evaluation MCU

This means the MCU mounted on the emulator which is operated in a dedicated mode for use with tools.

#

This symbol indicates that a signal is active-low (e.g. RESET#).

1. Outline

This chapter describes the package components, the system configuration, and the specifications of the emulator functions and operating environment.

1.1 Package Components

The R0E556100MCU00 package consists of the following items. After you have unpacked the box, check if your R0E556100MCU00 contains all of these items.

Table 1.1 Package Components

Item		Quantity
R0E556100MCU00 MCU board		1
Oscillator module (12.5 MHz) mounted on the IC21 socket		1
R0E001000FLX10 flexible cable		2
R0E556100MCU00 Release Notes (English)		1
R0E556100MCU00 Release Notes (Japanese)		1
Repair Request Sheet (English)		1
Repair Request Sheet (Japanese)		1
CD-ROM	<ul style="list-style-type: none"> • RX610 E100 Emulator Software (RX610 E100 Emulator Debugger included) • User's Manual 	1

- Notes:
1. Please keep the R0E556100MCU00's packing box and cushioning materials at hand for later reuse in sending the product for repairs or for other purposes. Always use the original packing box and cushioning material when transporting the MCU unit.
 2. If you have any questions or are in doubt about any point regarding the packaged product, contact your local distributor.

1.2 Other Tool Products Required for Development

To proceed with the development of a program for RX600 Series RX610 Group MCUs, the products listed below are necessary in addition to those contained in the package and listed above. Procure them separately.

Table 1.2 Other Tool Products Required for Development

Product	Part No.
Emulator main unit E100	R0E001000EMU00
144-pin 0.5-mm pitch LQFP (PLQP0144KA-A; Previous code: 144P6Q-A)	R0E556100CFK00

Note: To purchase the product, contact your local distributor.

1.3 System Configuration

1.3.1 System Configuration

Figure 1.1 shows the configuration of the emulator system.

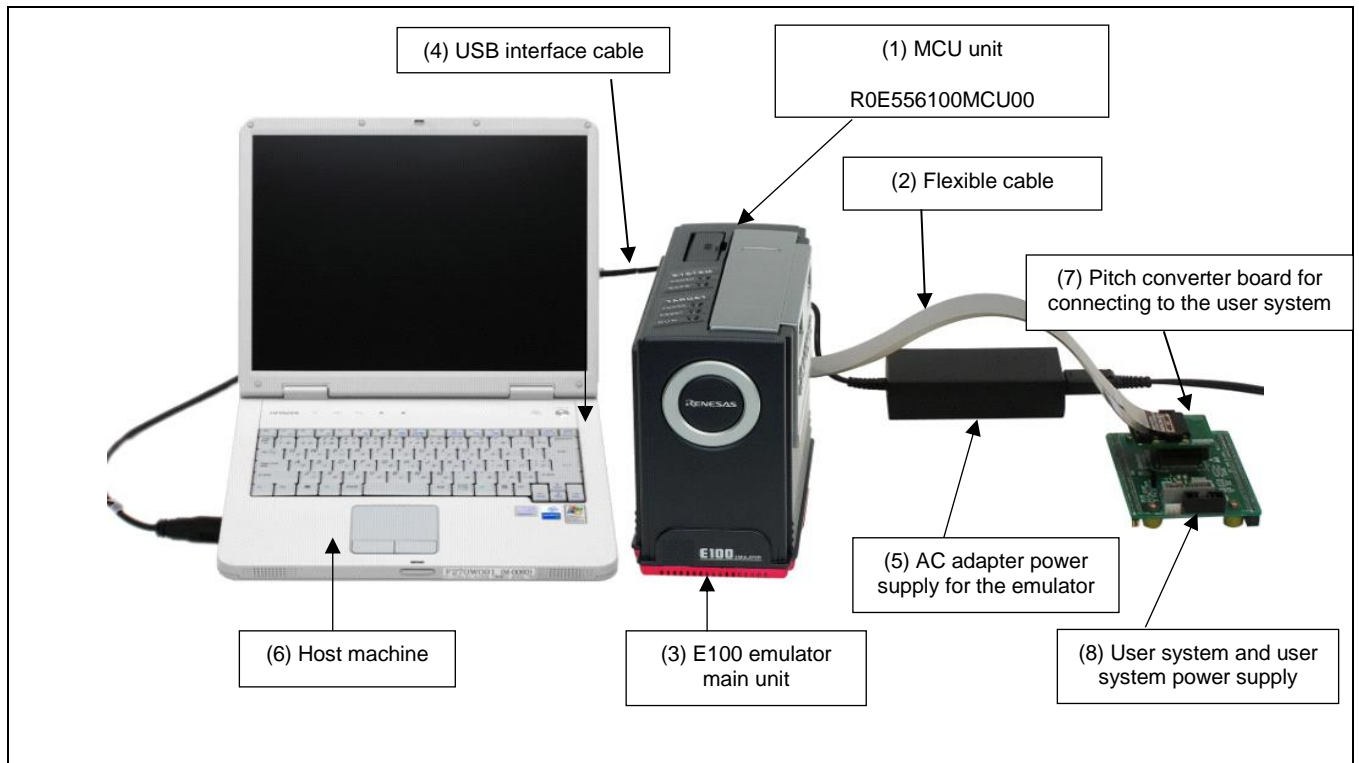


Figure 1.1 System Configuration

(1) MCU Unit R0E556100MCU00 (this product)

This is an MCU board for the RX610 Group MCUs with 2-MB ROM and contains an evaluation MCU.

(2) Flexible cable R0E001000FLX10 (included)

(3) E100 Emulator main unit R0E001000EMU00

This is the E100 emulator main unit.

(4) USB interface cable

This is an interface cable for the host machine and emulator.

(5) AC adapter supply for the emulator

(6) Host machine

A personal computer to control the emulator.

(7) Pitch converter board for connecting the user system R0E556100CFK00, etc.

(8) User system and user system power supply

User system is your application system. This emulator can be used without the user system.

The user system power supply is power supply for the user system. This emulator cannot supply power to the user system. Get a power supply separately.

1.3.2 Names and Functions of Each Part of the Emulator

Figure 1.2 shows the names of each part of the emulator.



Figure 1.2 Names of Each Part of the Emulator

- (1) Power switch
This is a switch to turn the emulator on and off.
- (2) USB cable connector
This is a connector for connecting the USB cable of the emulator.
- (3) Power connector
This is a connector for connecting the DC cable of the AC power adapter of the emulator.
- (4) External trigger connector
This is a connector to connect the external trigger cable of the emulator.
- (5) System status LEDs
The system status LEDs indicate the emulator E100's power supply, operating state of firmware, etc. Table 1.3 lists the definitions of each system status LED.

Table 1.3 Definitions of the System Status LEDs

Name	Status	Meaning
POWER	ON	Emulator system power is turned on.
	OFF	Emulator system power is turned off.
SAFE	ON	Emulator system is operating normally.
	Flashing	Emulator system cannot communicate with the host machine.
	Flashing (every 2 seconds)	The self-checking is in progress.
	OFF	Emulator system is not operating normally (system status error).

(6) Target status LEDs

The target status LEDs indicate operating state of the target MCU and power supply of the user system. Table 1.4 lists the definition of each target status LED.

Table 1.4 Definitions of the Target Status LEDs

Name	Status	Meaning
POWER	ON	Power is being supplied to the user system.
	OFF	Power is not being supplied to the user system.
RESET	ON	Target MCU is being reset, or reset signal of the user system is held low.
	OFF	Target MCU is not being reset.
RUN	ON	User program is being executed.
	OFF	User program has been halted.

IMPORTANT**Note on the Target Status POWER LED:**

If your MCU has two or more Vcc pins, the LED does not light up unless power is supplied to all the pins.

1.4 Specifications

Table 1.5 lists the specifications of the R0E556100MCU00.

Table 1.5 Specifications of the R0E556100MCU00

Item	Description
Applicable MCU	RX600 Series RX610 Group MCUs with 2-MB ROM
Applicable MCU mode	Single-chip mode, on-chip ROM disabled extended mode, on-chip ROM enabled extended mode
Maximum ROM/RAM capacity	<ol style="list-style-type: none"> Internal flash ROM: 2 MB + 32 KB 00100000h to 00107FFFh, FFE00000h to FFFFFFFFh Internal RAM: 128 KB 00000000h to 0001FFFFh
Maximum operating frequency	Power supply voltage: 3.0 to 3.6 V, 100 MHz (with PLL)
Software break	4096 points
Hardware break	16 points (execution PC, bus detection, interrupt, external trigger signal)
Combination, pass count	<ul style="list-style-type: none"> Cumulative AND/OR/simultaneous AND/status transition 255 pass counts
Detection of exceptional events	Violation of access protection/task stack access violation/OS dispatch/reading from uninitialized memory
Real-time tracing	288 bits × 4-M cycles (execution PC, operand address, operand data, operand status, DMAC/DTC address, DMAC/DTC data, DMAC/DTC status, target status, task ID, timestamp, 32 external trigger inputs)
Trace modes	Fill until stop/fill until full/fill around TP/repeat fill until stop/repeat fill until full
Extraction/deletion of trace data	<ul style="list-style-type: none"> Extracting or deleting data by specifying events Extracting data before and after trace points
Real-time RAM monitor	<ul style="list-style-type: none"> 16,384 bytes (512 bytes × 32 blocks) Data/last access
Time measurement	<ul style="list-style-type: none"> Execution time between program start and stop Maximum/minimum/average execution time and number of passes through eight specified sections Clock used to count times: 10 ns to 1.6 μs
Coverage measurement	2 MB (256 KB × 8 blocks)
Profile	1 MB (128 KB × 8 blocks)
Connection to user system	144-pin 0.5-mm pitch LQFP R0E556100CFK00
Emulator power supply	Supplied from an AC adapter (power supply voltage: 100 to 240 V, 50/60 Hz)

1.5 Operating Environment

Make sure to use this emulator in the operating environments listed in Tables 1.6 to 1.8.

Table 1.6 Operating Environmental Conditions

Item	Description
Operating temperature	5 to 35°C (no condensation)
Storage temperature	-10 to 60°C (no condensation)

Table 1.7 Operating Environment of the Host Machine (Windows® XP or Windows® 2000)

Item	Description
Host machine	IBM PC/AT compatible
OS	Windows® XP 32-bit editions*1, *3 Windows® 2000*1
CPU	Pentium 4 running at 1.6 GHz or more recommended
Interface	USB 2.0*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

Table 1.8 Operating Environment of the Host Machine (Windows Vista® or Windows® 7)

Item	Description
Host machine	IBM PC/AT compatible
OS	Windows Vista® 32-bit editions*1, *4 Windows® 7 32-bit editions*1, *5
CPU	Pentium 4 running at 3 GHz or Core 2 Duo running at 1 GHz or more recommended
Interface	USB 2.0*2
Memory	2 Gbytes 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

- 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.
 5. The 64-bit editions of Windows® 7 are not supported.

1.6 Specifications of the AC Adapter

The user must prepare an AC adapter and a power cable. Use an AC adapter which complies with the safety standards of the country where it is to be used. Table 1.9 lists the recommended specifications of the AC adapter.

Table 1.9 Recommended Specifications of the AC Adapter

Item	Description
AC input voltage range	AC 100 to 240 V, 50/60Hz single phase
Output power	36 W
DC output voltage, current	12.0 V, 3.0 A
DC output polarity	EIAJ TYPE IV, inner-side positive/outer-side negative

2. Setup

This chapter describes the preparation for using the MCU unit, the procedure for starting up the emulator and how to change settings.

2.1 Flowchart of Starting Up the Emulator

The procedure for starting up the emulator is shown in Figures 2.1 and 2.2. For details, refer to each section hereafter. If the emulator does not start up normally, refer to “Troubleshooting (Action in Case of an Error)”.

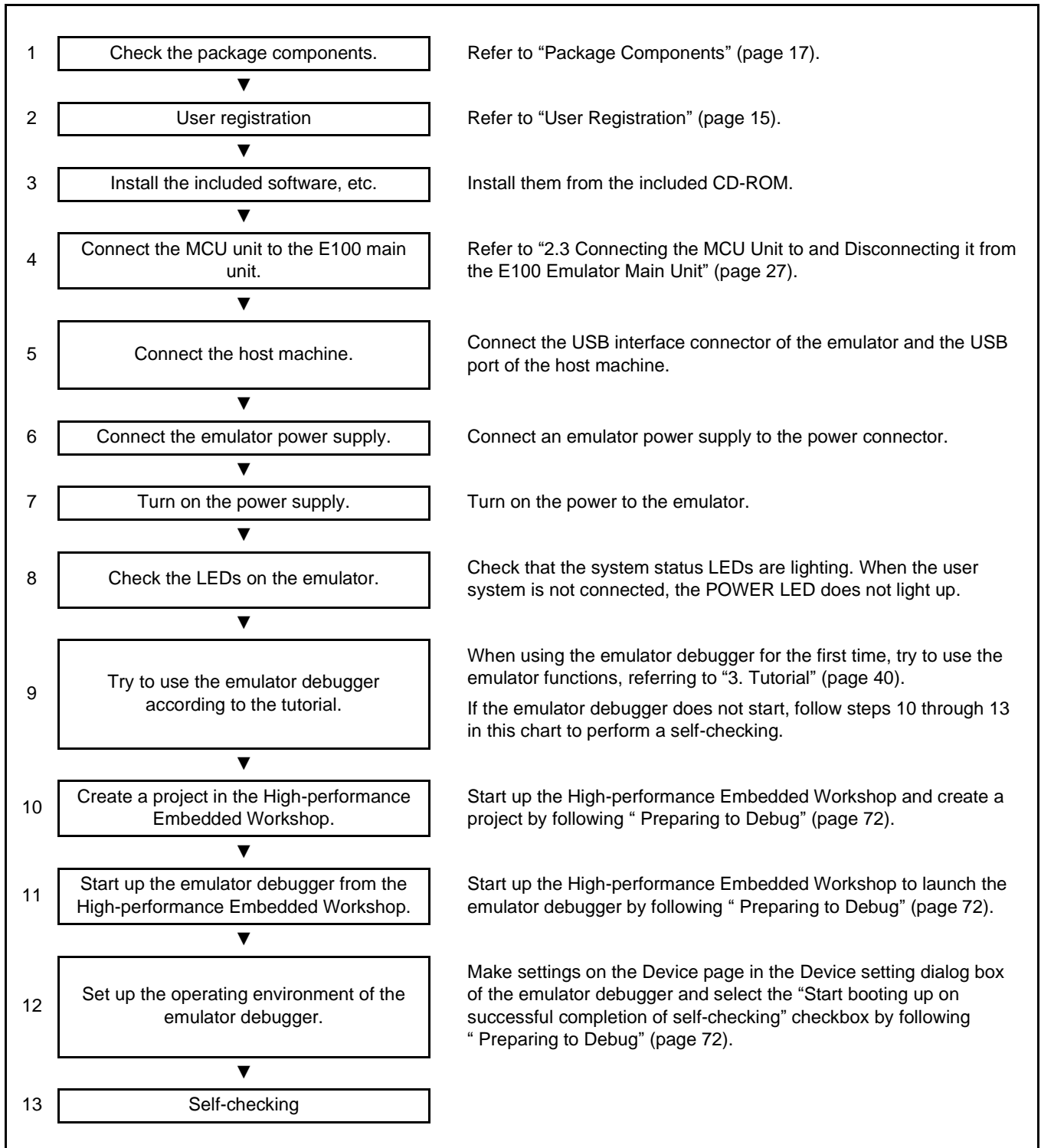


Figure 2.1 Flowchart of Starting up the Emulator (for the First Time)

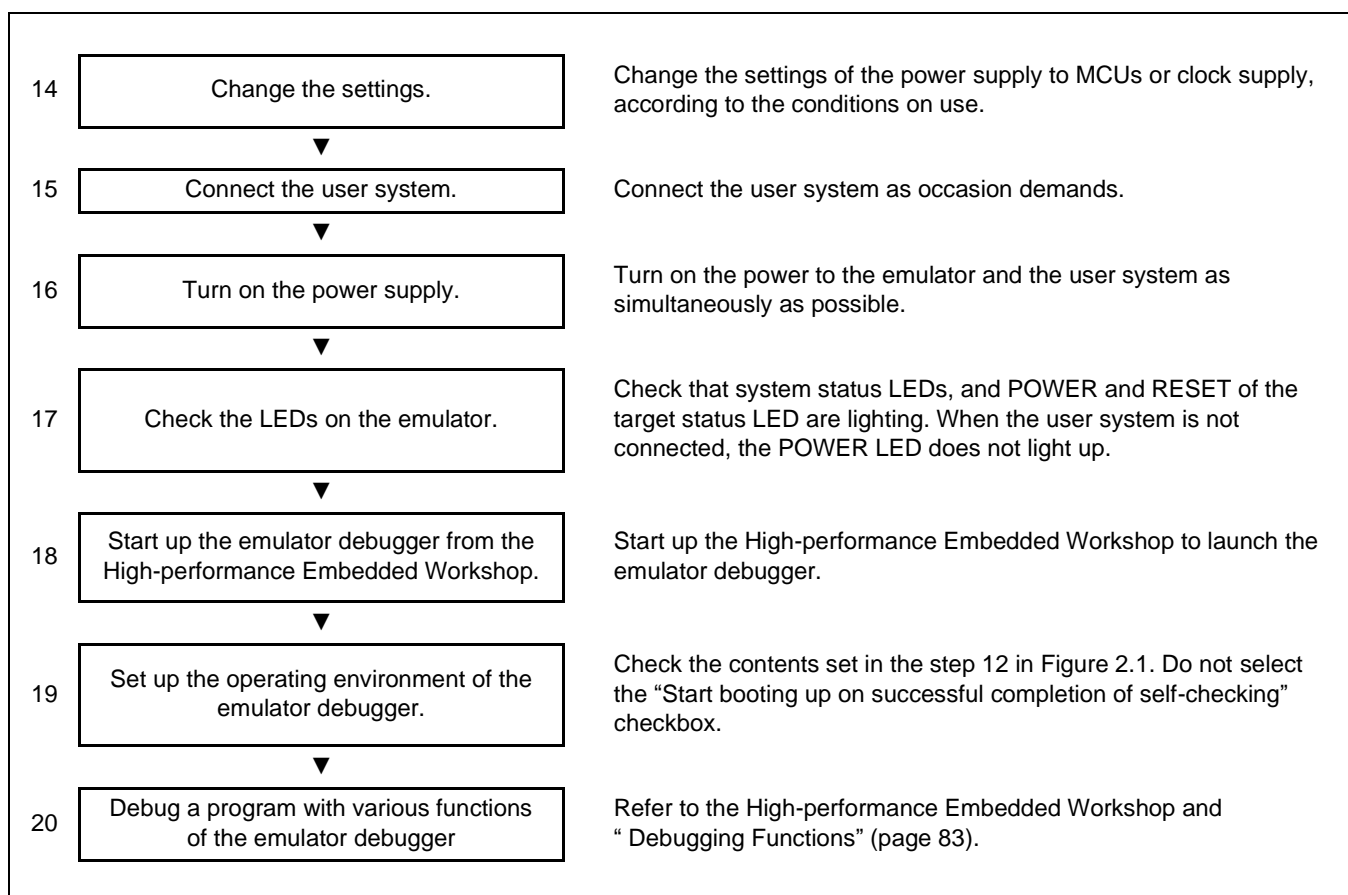


Figure 2.2 Flowchart of Starting up the Emulator (after the Self-Checking)

2.2 Installing the Included Software

If you have Windows® 7, Windows Vista®, Windows® XP or Windows® 2000 on the host machine, this installation must be executed by a user with administrator rights. Note that users without administrator rights cannot complete the installation.

Place the CD-ROM in the CD-ROM drive and follow the instructions to install the software.

2.3 Connecting the MCU Unit to and Disconnecting it from the E100 Emulator Main Unit

Figure 2.3 shows the procedure for connecting the MCU unit to the E100 emulator main unit.

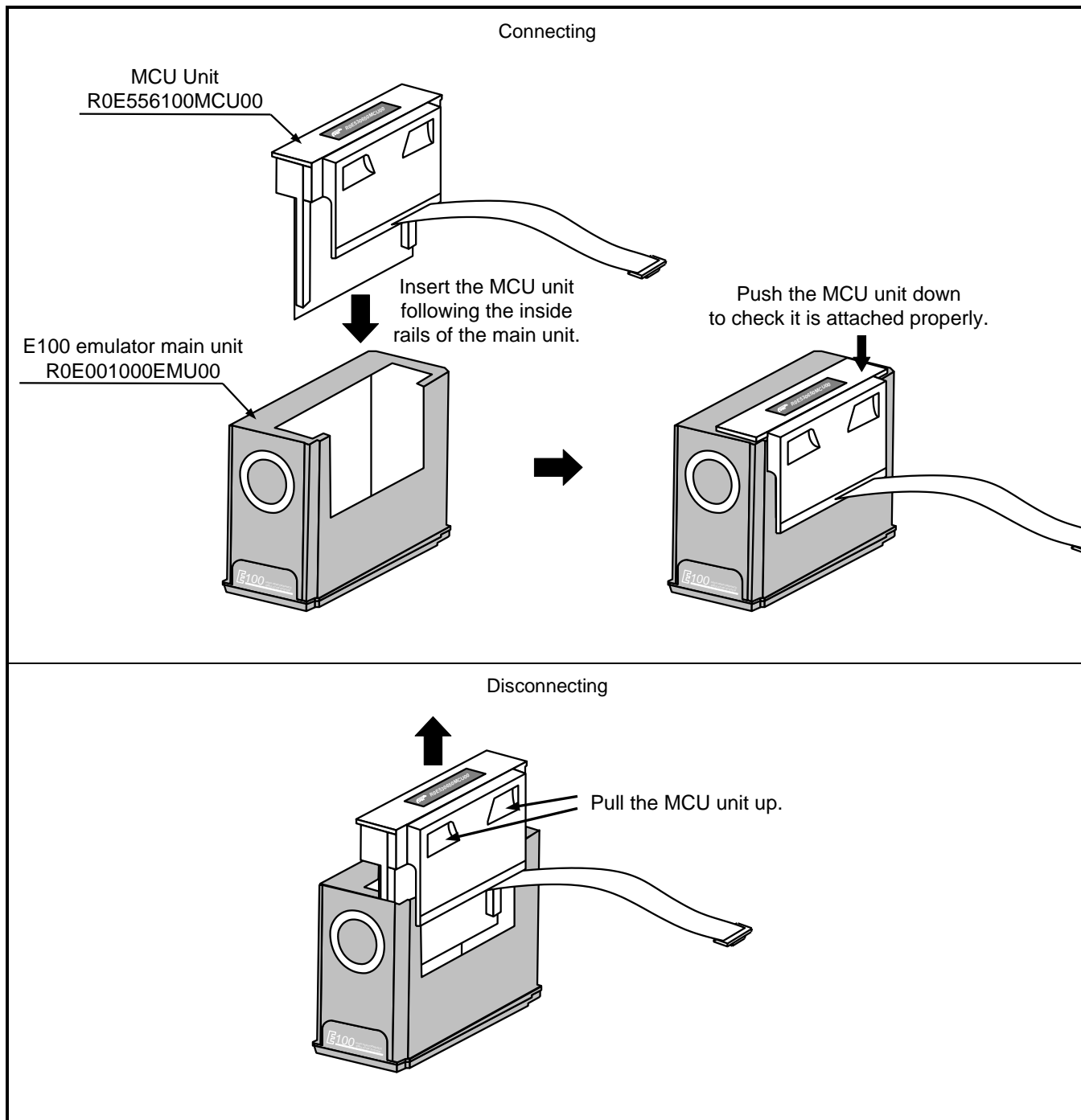


Figure 2.3 Connecting the MCU Unit to and Disconnecting it from the E100 Emulator Main Unit

 **CAUTION****Note on Connecting the MCU Unit to the E100 Emulator Main Unit:**

Always shut OFF power when connecting the MCU unit to the E100 emulator main unit. Otherwise, internal circuits may be damaged.

2.4 Connecting the Host Machine

USB interface is used for connecting the emulator to the host machine. The USB cable is connected to the USB cable connector of the emulator and the USB port of the host machine.

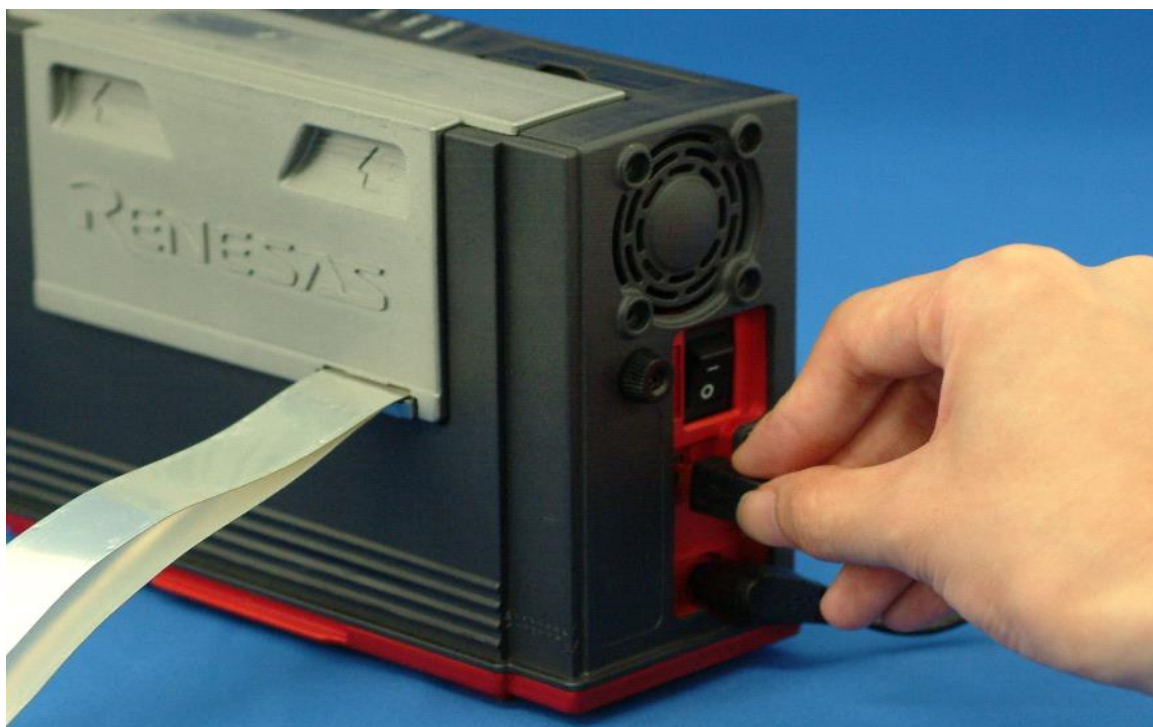


Figure 2.4 Connecting the Host Machine

2.5 Connecting the Emulator Power Supply

Power is supplied from an AC adapter to the emulator. The following shows how to connect the AC adapter.

- (1) Turn off the power of the emulator.
- (2) Connect the DC cable of the AC adapter to the emulator.
- (3) Connect the AC power cable to the AC adapter.
- (4) Connect the AC power cable to the outlet.



Figure 2.5 Connecting the Emulator Power Supply

CAUTION

Cautions for AC Power Cable:



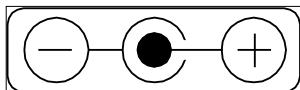
If the AC power cable does not fit the receptacle, do not alter the AC power cable and do not plug it forcibly. Failure to comply may cause electric shock and/or fire.

Do not touch the plug of the AC power cable when your hands are wet. This may cause electric shock.

Caution for AC Adapter:



The DC plug of the AC adapter has the polarity shown below.



Use an AC adapter which complies with the safety standards of the country where it is to be used.

2.6 Turning On the Power

2.6.1 Checking the Connections of the Emulator System

Before turning the power on, check the connection of the interface cable with the host machine, emulator, and user system.

2.6.2 Turning the Power On and Off

- Turn on/off the power of the emulator and user system as simultaneously as possible.
- When the SAFE LED of the system LEDs is flashing, check that the USB cable is connected to the host machine. When each of the target status LEDs is flashing, check that the MCU unit is connected.
- When turning on the power again after shutting off the power, wait for about 10 seconds.

IMPORTANT

Notes on Power Supply:

The emulator pin Vcc is connected to the user system in order to monitor user system voltage. For this reason, the emulator cannot supply power to the user system. Supply power to the user system separately.

The voltage of the user system should be as follows.

$$3.0 \text{ V} \leq V_{\text{cc}} \leq 3.6 \text{ V}$$

When you start the emulator without the user system, do not attach a converter board. When starting with a converter board, the MCU will be in a reset status.

When you start the emulator without the user system, take care that metallic pieces are not touched to the connector at the head of the flexible cable.

Do not leave either the emulator or user system powered on. The internal circuits may be damaged due to leakage current.

2.7 Self-Checking

Self-checking is to check if the emulator functions operate properly. To run the self-check function of the emulator, follow the procedure below. While the self-checking is in progress, the states of the LEDs will change as shown in Figure 2.6. In case of ERROR, because the states of the target status LEDs will change depending on the types of errors, check the system status LEDs.

- (1) If the user system is connected, disconnect the converter board and the user system.
- (2) Turn on the emulator.
- (3) Launch the emulator debugger, and select the "Start booting up on successful completion of self-checking" checkbox in the [Device setting] dialog box.
- (4) When you click OK, self-checking will start. If the normal result is displayed in about 60 seconds, self-checking has ended.

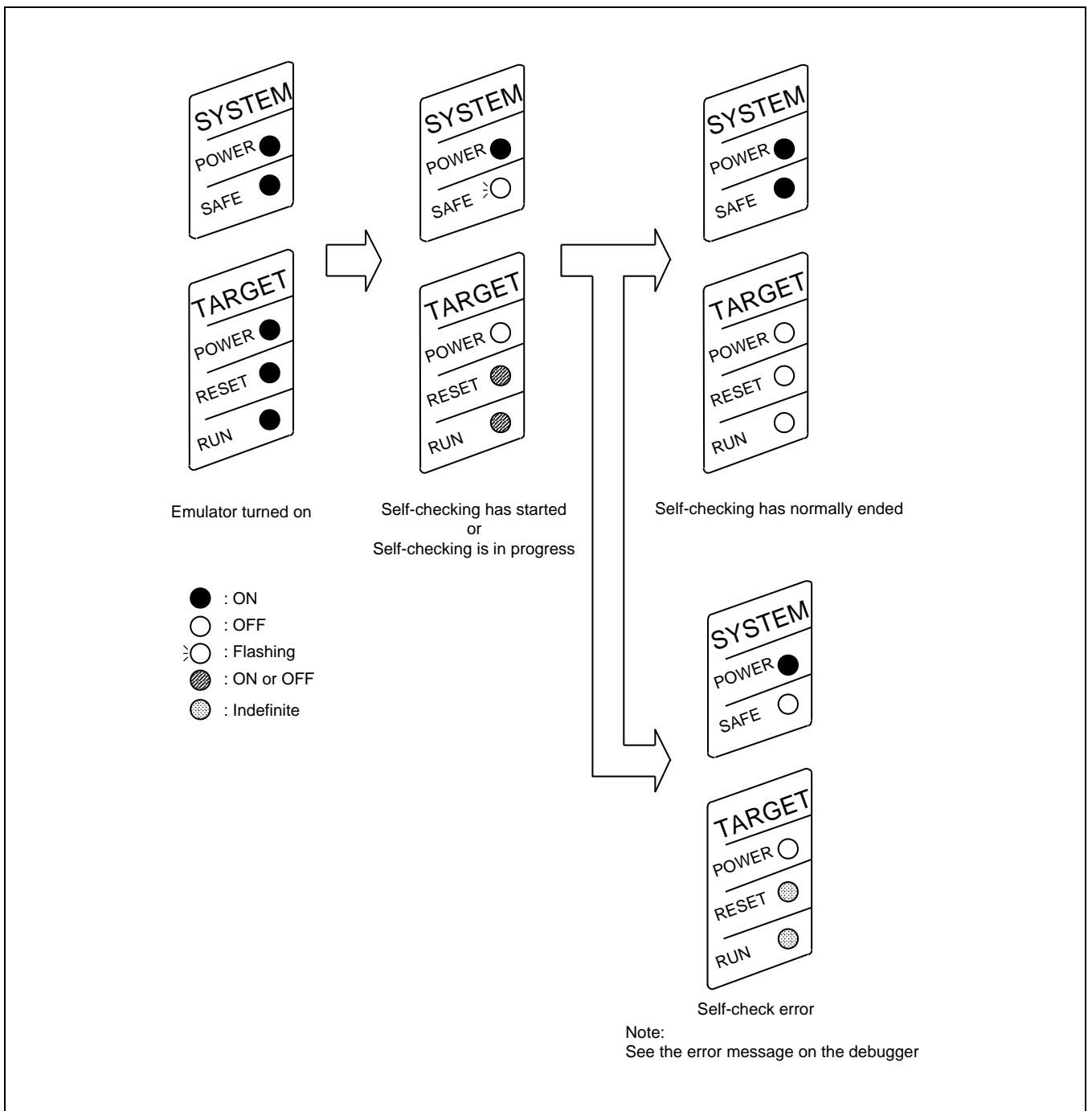


Figure 2.6 LED Displays during the Self-Checking

2.8 Selecting the Clock Supply

2.8.1 Clock Source

You can choose the clock source supplied to the evaluation MCU in the [Configuration properties] dialog box of the emulator debugger. Table 2.1 shows the clock sources and their default settings.

Table 2.1 Clock Supply to the MCU

Clock	Clock Selection in the Emulator Debugger	Description	Default Setting
Main (XTAL-EXTAL)	Emulator	IC21 mounting oscillator module	Yes
	User	Oscillator circuit on the user system	-
	Generate	Internal generator circuit (1.0 to 14.0 MHz)	-

IMPORTANT

Note on Changing the Clock Supply:

The clock supply can be set by the Configuration properties dialog box when starting up the emulator debugger or by an input of the emulator_clock command on the [Command Line] window.

2.8.2 Using an Internal Oscillator Circuit Board

Kinds of Oscillator Circuit Boards

An oscillator module (12.5 MHz) is mounted on the IC21 at factory setting. If you change the frequency, replace the oscillator module.

(1) Replacing the oscillator module

Remove the MCU unit from the E100 emulator main unit, and replace the oscillator module of the IC21 (see Figure 2.7).

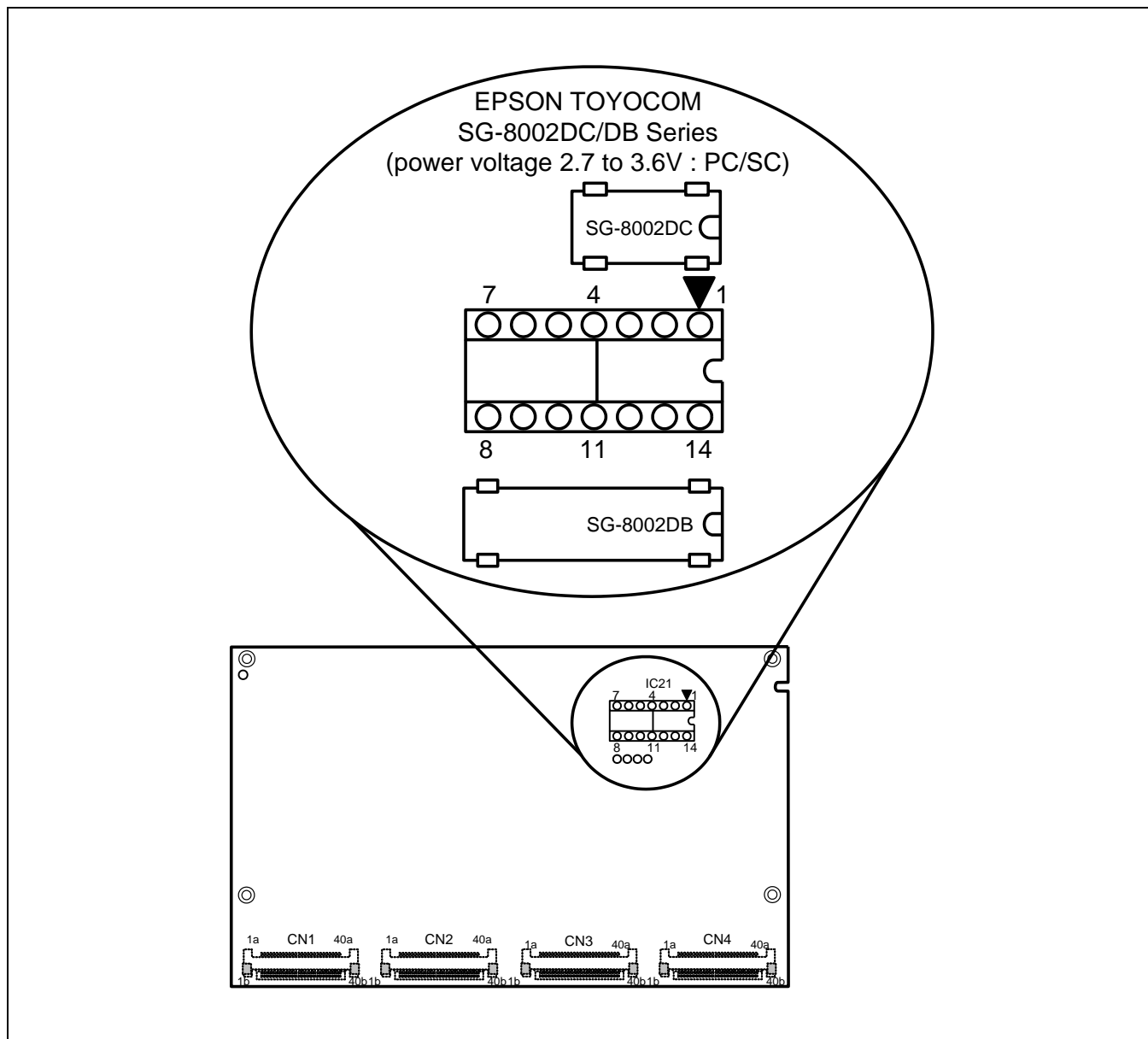


Figure 2.7 Replacing the Oscillator Module

 **CAUTION****Notes on Replacing the Oscillator Module and Oscillator Circuit Board:**

Always shut OFF power when replacing the oscillator module. Otherwise, internal circuits may be damaged.

When replacing the oscillator module, remove it with a tool such as an IC extractor so as not to damage the board. If the board is damaged, the pattern on the board may be cut and the emulator may not be able to operate.

2.8.3 Using the Oscillator Circuit on the User System

To operate this product with an external clock, construct the oscillator circuit as shown in Figure 2.8 in the user system and input the oscillator output at 50% duty (within the operating range of the evaluation MCU) into pin EXTAL. And pin XTAL should be open. Choose "User" in the emulator debugger to use this clock.

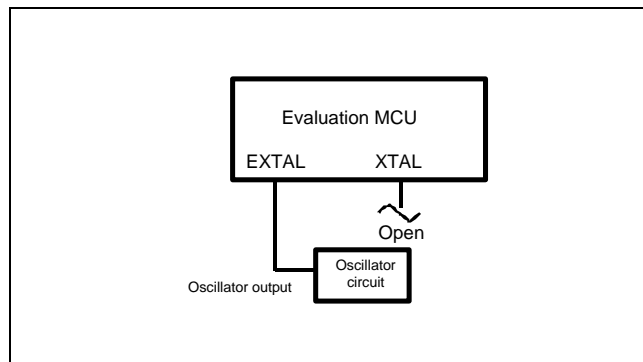


Figure 2.8 External Oscillator Circuit

Make note that in the oscillator circuit shown in Figure 2.9 where an oscillator is connected between pins XTAL and EXTAL, oscillation does not occur because a converter board and other devices are used between the evaluation MCU and the user system.

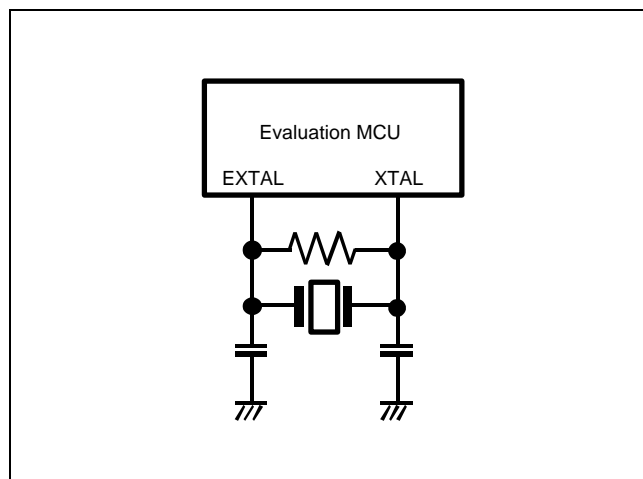


Figure 2.9 Circuit in which Oscillation does not Occur

2.8.4 Using the Internal Generator Circuit

The dedicated circuit in the E100 can generate any arbitrary frequency specified by the emulator debugger, and it can be supplied as a main clock. It does not depend on either the oscillator circuit board in the MCU unit or the oscillator circuit on the user system. If you want to debug programs without the user system or change a frequency temporarily, you can check its operation before purchasing an oscillator. If you want to use the internal generator circuit in the E100 as a main clock, choose "Generate" in the emulator debugger and specify a frequency you like to use this clock.

Although you can change a frequency between 1.0 and 99.9 MHz by 0.1 MHz for the E100, do not specify a value exceeding the maximum input frequency 14 MHz of the EXTAL of the MCU.

IMPORTANT

Notes on Using the Internal Generator Circuit:

The internal generator circuit is equipped for temporary debugging purposes. Temperature characteristics of frequencies are not guaranteed.

Be sure to evaluate your system with an oscillator whose frequency is the same as that of the oscillator module or oscillator circuit (emulator) for final evaluation purposes.

2.9 Connecting the User System

Figure 2.10 shows how to connect this product to your user system.

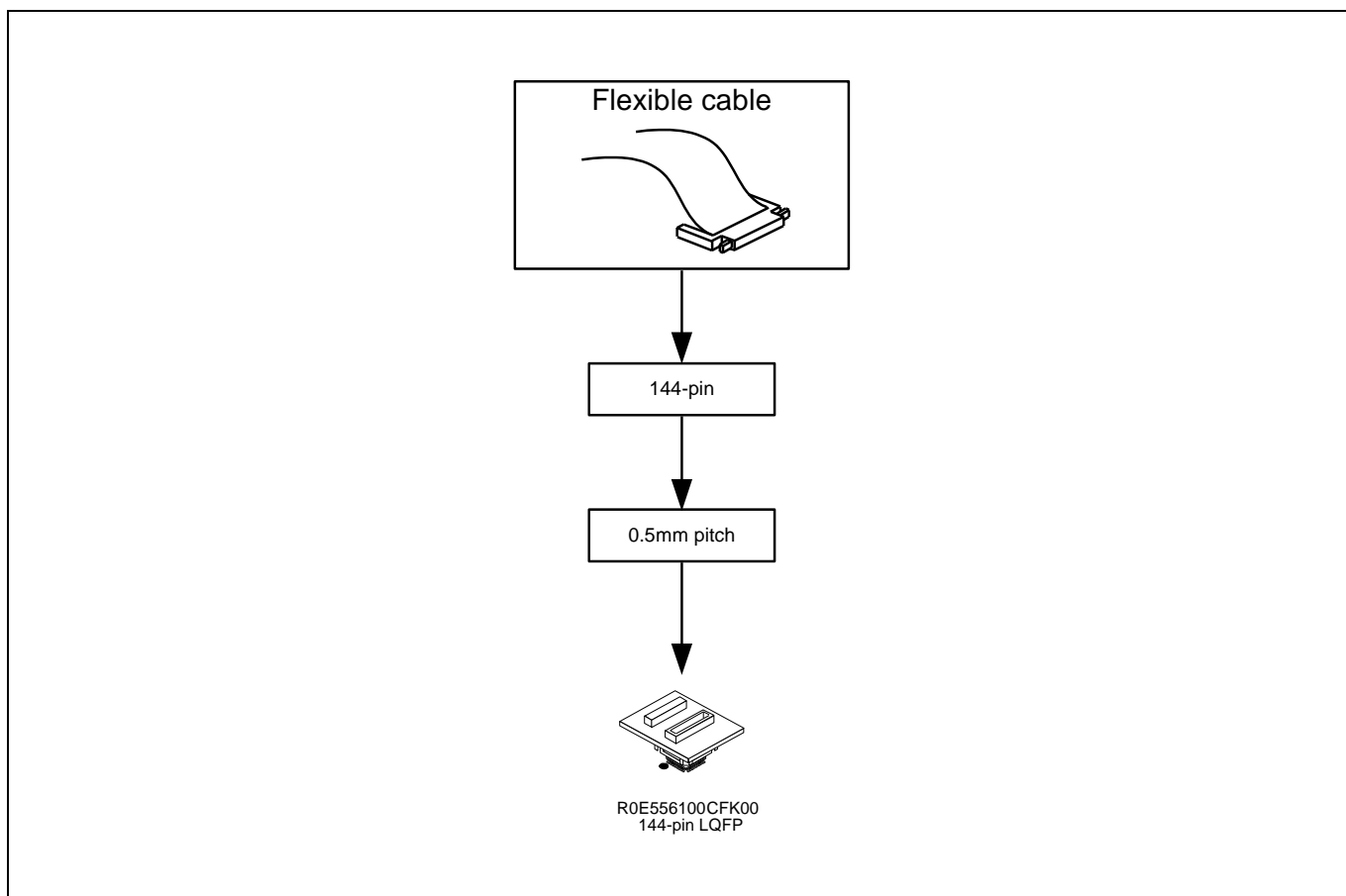


Figure 2.10 Connecting This Product to the User System

CAUTION

Note on Connecting the User System:



Take care not to attach a converter board in a wrong direction. It may cause a fatal damage to the emulator and user system.

2.9.1 Connecting to a 144-Pin 0.5-mm Pitch Foot Pattern

The following is a procedure of connecting to a 144-pin 0.5-mm pitch foot pattern on the user system using the R0E556100CFK00 (not included). For details on the R0E556100CFK00 (not included), refer to its user's manual.

- (1) Attach the NQPACK144SD-ND included with the R0E556100CFK00 to the user system.
- (2) Attach the YQPACK144SD included with the R0E556100CFK00 to the NQPACK144SD-ND and secure it with the YQ-GUIDES.
- (3) Attach the R0E556100CFK00 to the YQPACK144SD.
- (4) Attach the CN2 side of the R0E556100CFK00 to the CN2 side of the flexible cable.
- (5) Attach the CN1 side of the R0E556100CFK00 to the CN1 side of the flexible cable.

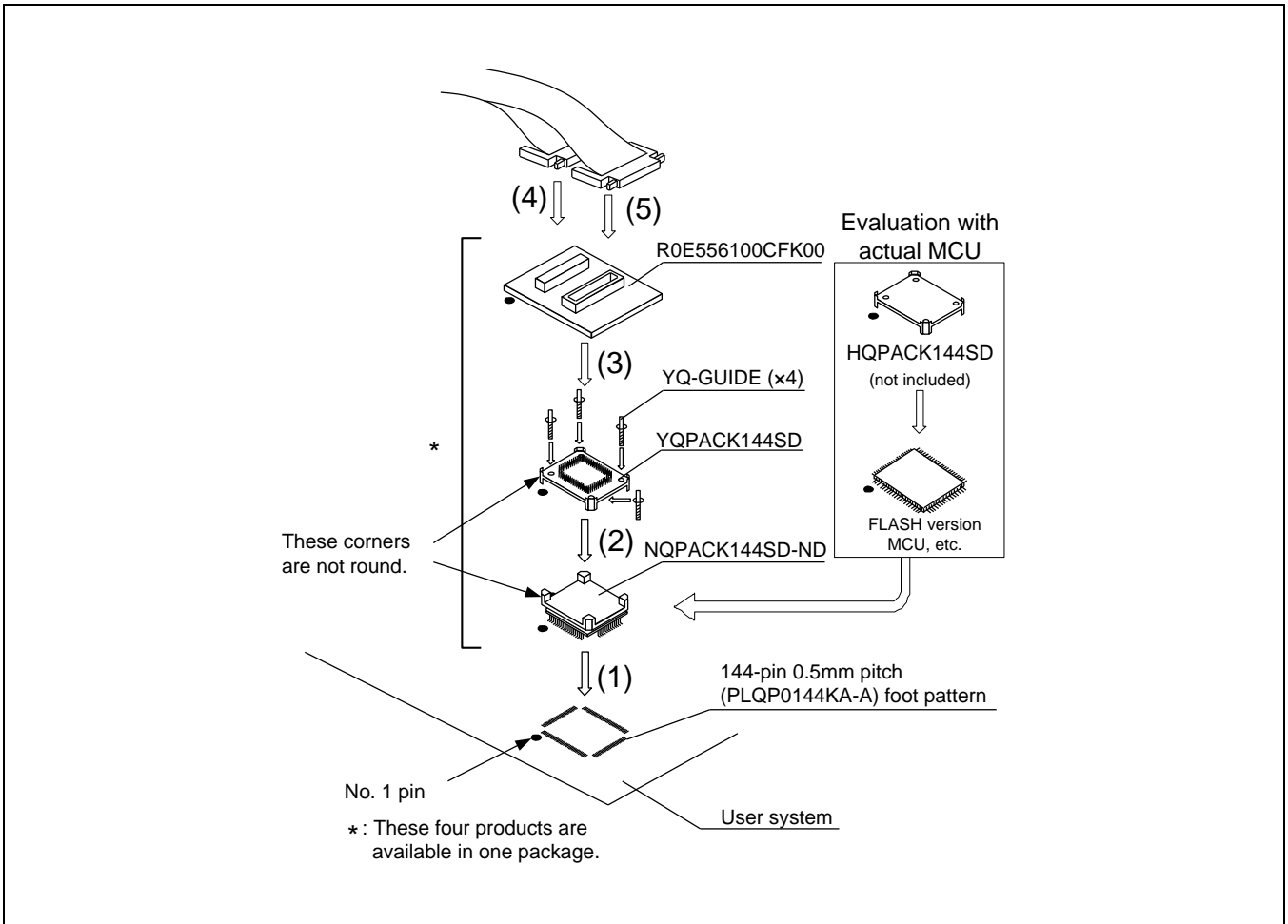


Figure 2.11 Connecting to a 144-pin 0.5-mm Pitch Foot Pattern

CAUTION

Notes on Connecting the User System:



Take care not to attach a converter board in a wrong direction. It may cause a fatal damage to the emulator and user system.

The connectors of the R0E556100CFK00 are guaranteed for only 50 insertion/removal iterations.

For purchasing the HQPACK144SD, contact the following:

Tokyo Eletech Corporation http://www.tetc.co.jp/e_index.htm

3. Tutorial

3.1 Introduction

The E100 emulator has a tutorial program available. This program is provided as a means for presenting to you the main features of the emulator, as will be explained in this document.

This tutorial program is written in C language, and is created to sort 10 pieces of random data in ascending/descending orders.

The following outlines the processing performed by the tutorial program.

The main function calls the tutorial function repeatedly in order to execute a sort process repeatedly.

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

The sort function inputs an array that contains the random data generated by the tutorial function and sorts the input data in ascending order.

The change function inputs an array that was sorted in ascending order by the sort function and sorts the input data in descending order.

The tutorial program is a program designed to help users to understand how to use the functions of the emulator and the emulator debugger. When developing user systems and user programs, refer to the user's manuals of the target MCUs.

CAUTION

If the tutorial program is recompiled, the addresses in a recompiled program may not be the same as those described in this chapter.

3.2 Starting the High-performance Embedded Workshop

Open a workspace following the procedure described in Section 4.4, "Opening an Existing Workspace".

For the directory, specify the one that is given below.

OS installed drive: \\WorkSpace\\Tutorial\\E100\\RX\\Tutorial_littleEndian

For the file, specify the one that is shown below.

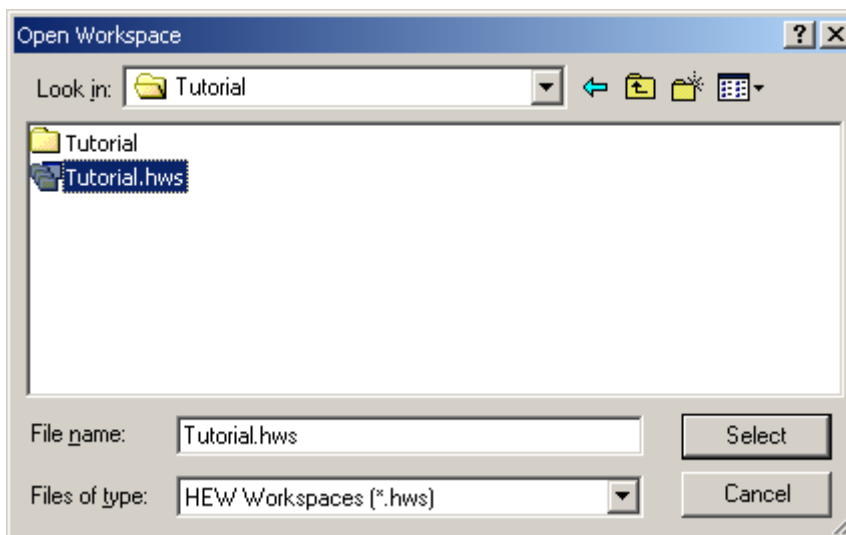


Figure 3.1 [Open Workspace] Dialog Box

3.3 Connecting the Emulator

When the debugger is connected to the emulator, a dialog box for setting up the debugger is displayed. In this dialog box, make initial settings of the debugger.

In the dialogs below, simply click the [OK] button leaving the initial values intact.

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

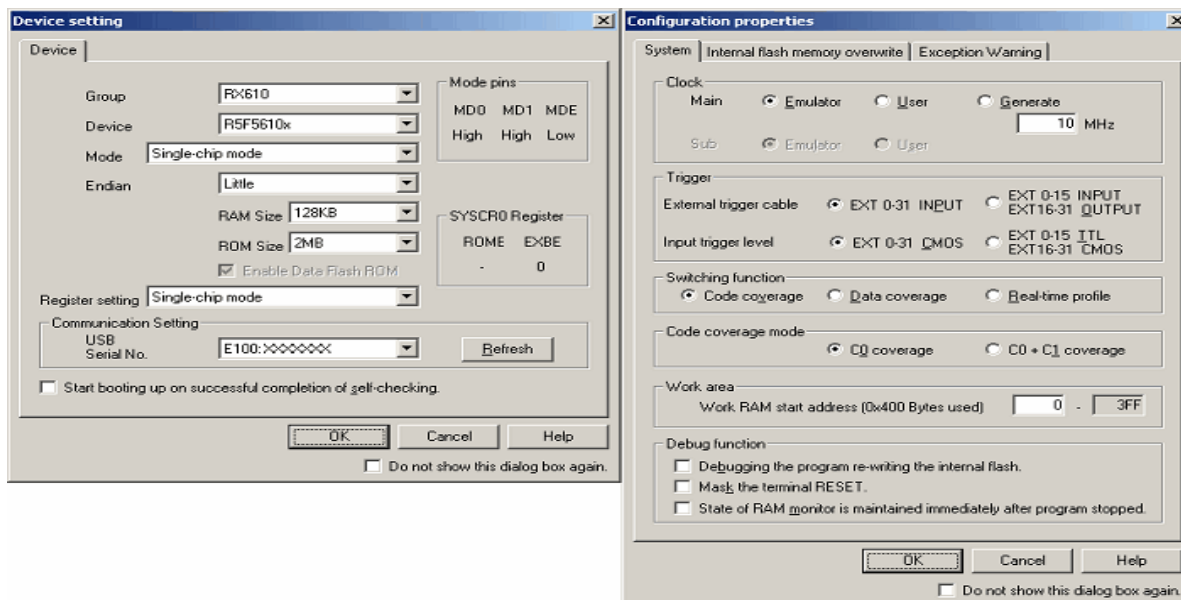


Figure 3.2 [Device Setting] and [Configuration Properties] Dialog Boxes

3.4 Downloading the Tutorial Program

3.4.1 Downloading the Tutorial Program

Download the object program you want to debug. Note, however, that the program to be downloaded and the address in the microcomputer to which downloaded differ with each microcomputer used. Read the strings, etc. on the screen as suitable for the microcomputer you are using.

Choose [Download] from [Tutorial.abs] of [Download modules].

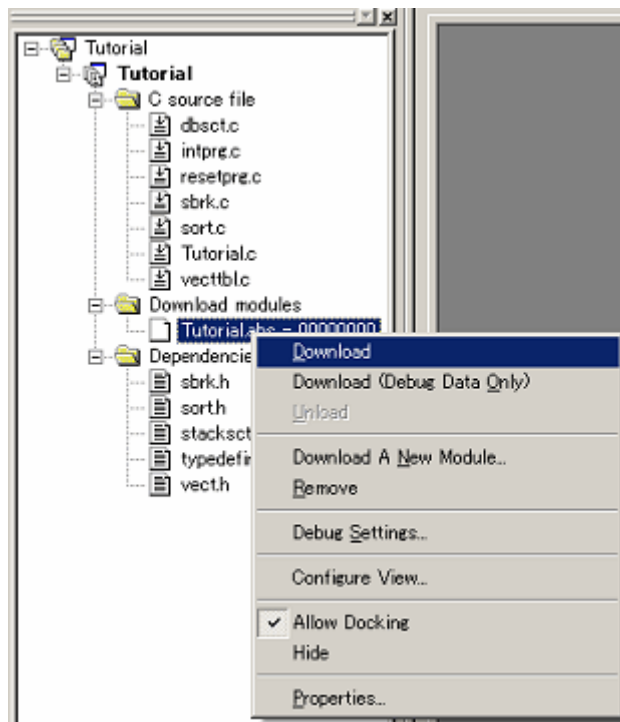


Figure 3.3 Download Display of the Tutorial Program

3.4.2 Displaying the Source Program

In the High-performance Embedded Workshop, you can debug a program at the source level.

Double-click [Tutorial.c] of [C source file].

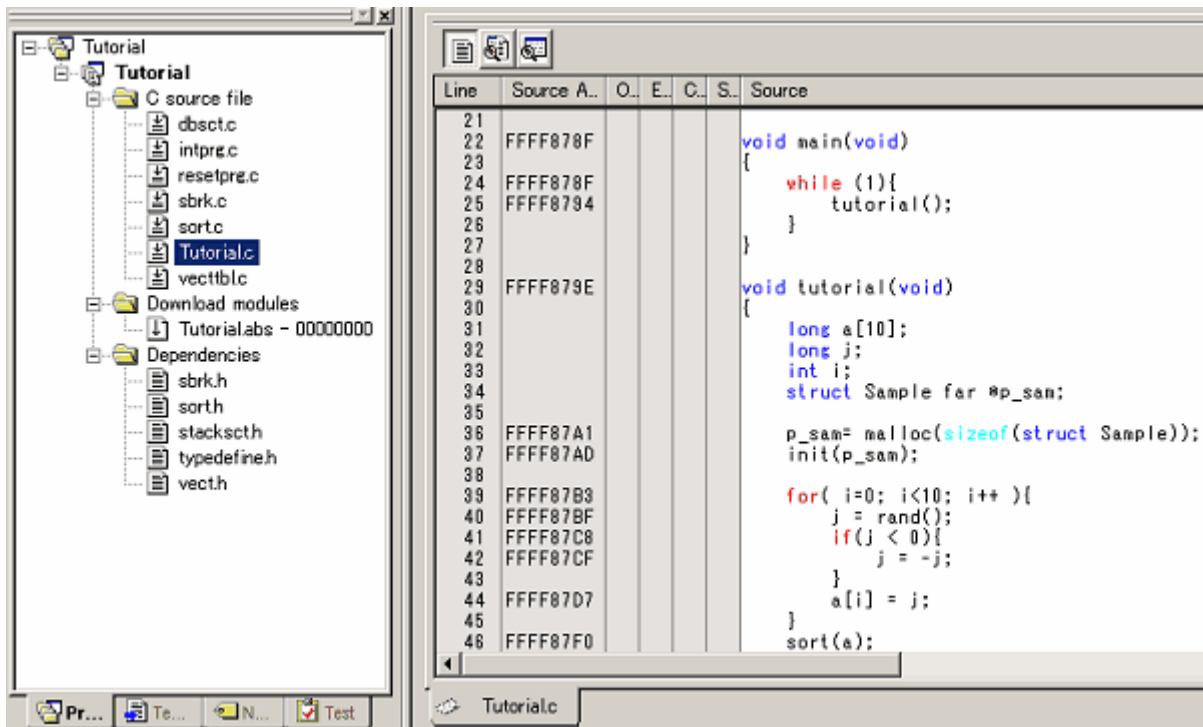


Figure 3.4 [Editor] Window (Displaying the Source Program)

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

The [Editor] window initially shows the beginning of a program. Using the scroll bar, you can look at another part of a program.

3.5 Setting Software Breakpoints

Software breakpoints are one of simple debug facilities.

The [Editor] window permits you to set software breakpoints easily. For example, you can set a software breakpoint at a place where the sort function is called.

Double-click a row in the [S/W Breakpoints] column corresponding to the source line that includes a sort function call.

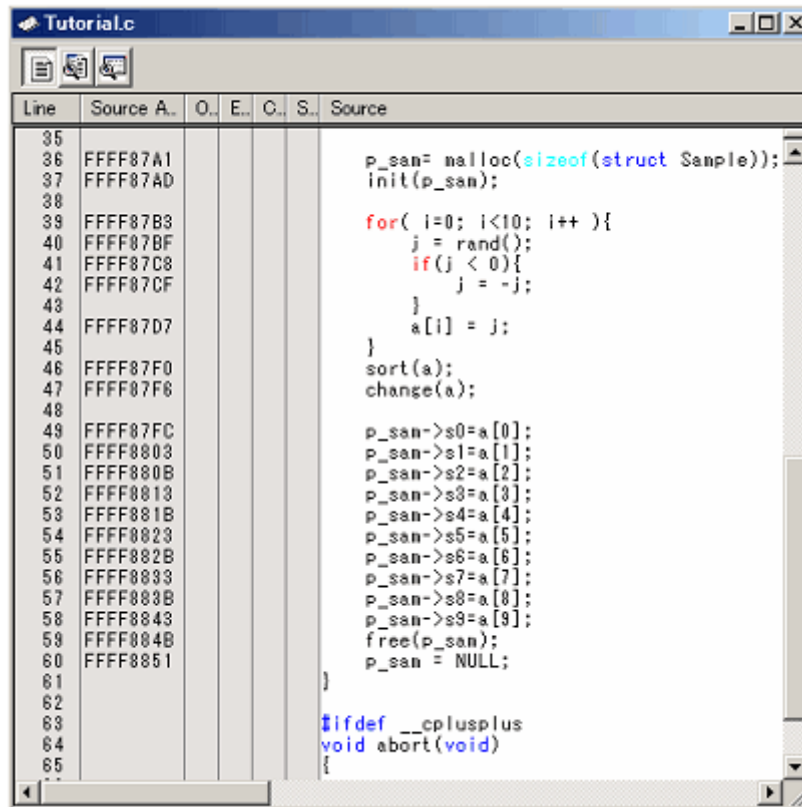



Figure 3.5 [Editor] Window (Setting a Software Breakpoint)

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


3.6 Executing the Program

The following describes how to run the program.

3.6.1 Resetting the CPU

To reset the CPU, choose [Reset CPU] from the [Debug] menu or click the [Reset CPU]  button in the toolbar.

3.6.2 Executing the Program

To execute the program, choose [Go] from the [Debug] menu or click the [Go] button  in the toolbar.

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 at which the program has stopped.

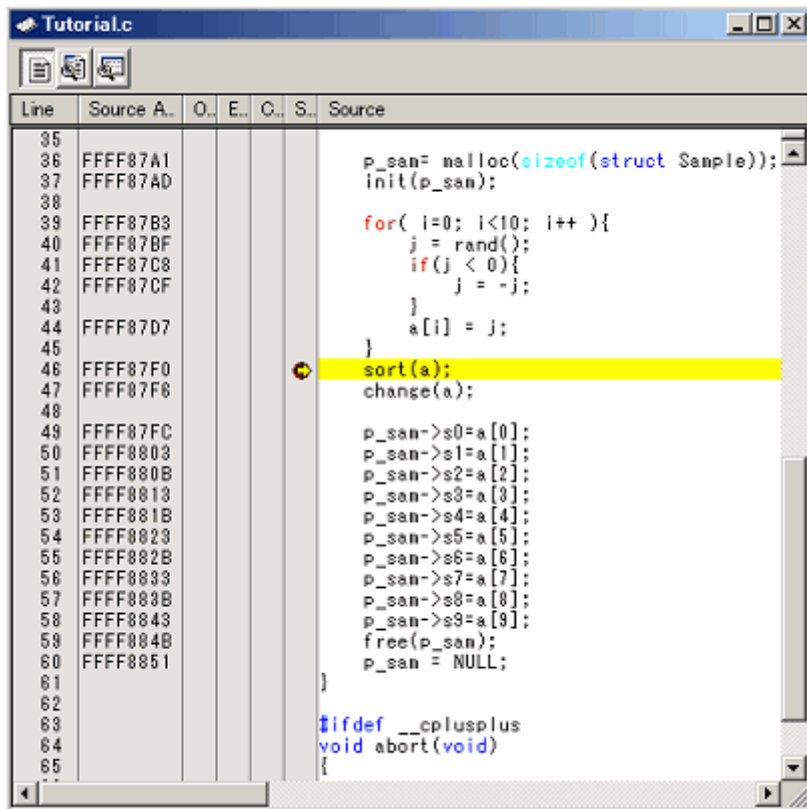



Figure 3.6 [Editor] Window (Program at a Break)

The [Status] window permits you to check the cause of the break that last occurred.

Choose [CPU -> Status] from the [View] menu or click the [View Status] toolbar button . When the [Status] window is displayed, open the [Target] sheet in it and check.

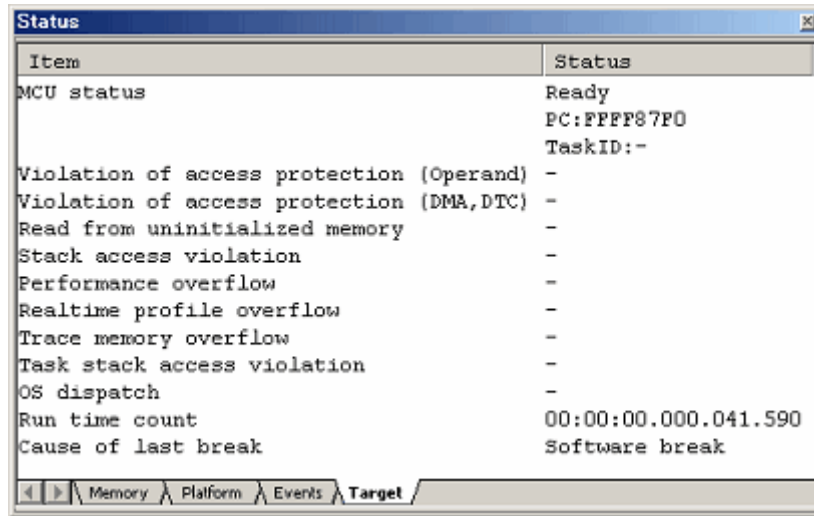


Figure 3.7 [Status] Window

CAUTION

The contents displayed in this window differ with each product. For details about the displayed contents of each product, refer to Chapter 5, “Debugging Functions,” or online help.

3.7 Checking Breakpoints

Use the [Breakpoints] dialog box to check all software breakpoints set.

3.7.1 Checking Breakpoints

Press the keys Ctrl + B on the keyboard of your PC. The [Breakpoints] dialog box shown below will be displayed.

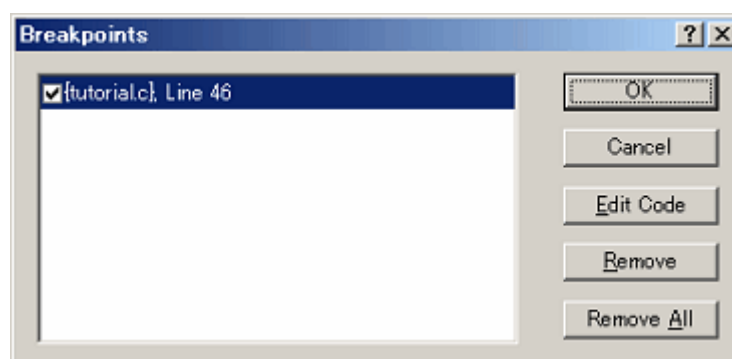



Figure 3.8 [Breakpoints] Dialog Box

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

3.8 Altering Register Contents

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

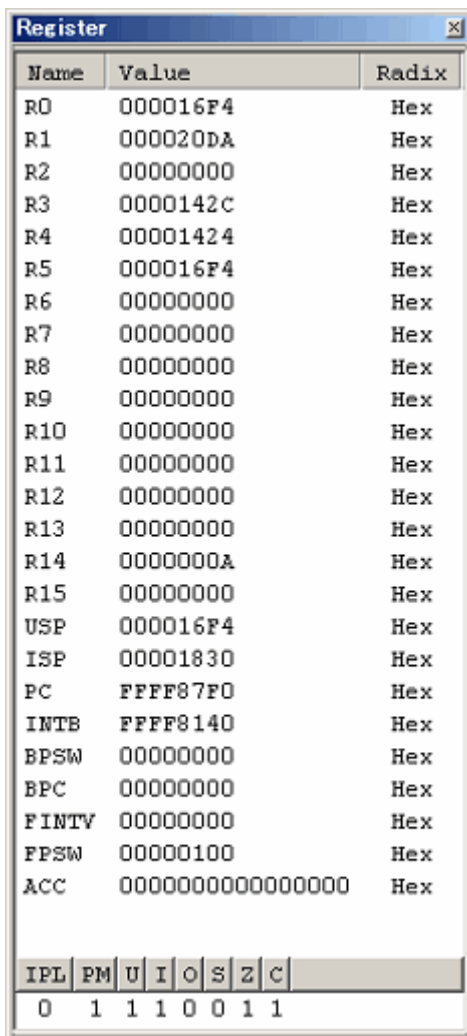


Figure 3.9 [Register] Window

The content of any register can be altered.

Double-click the line for the register you want to alter. The dialog box shown below will be displayed, so enter a new value with which you want to alter the register.

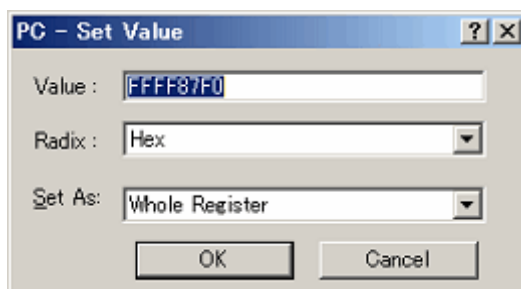

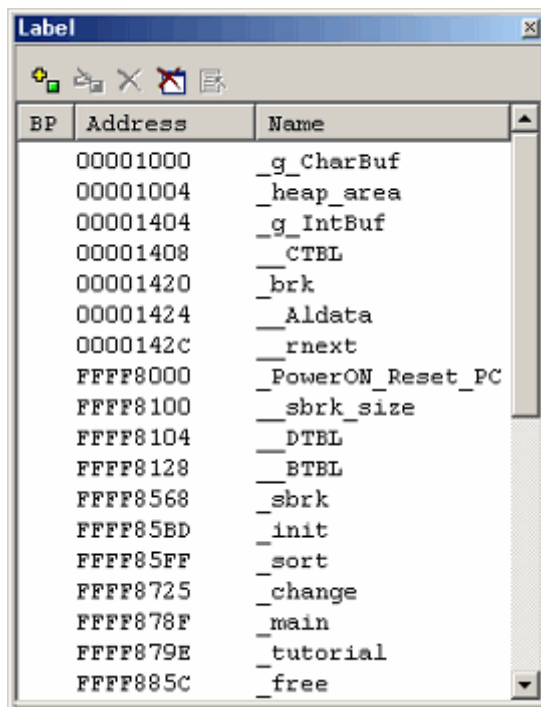


Figure 3.10 [Set Value] Dialog Box (PC)

3.9 Referencing Symbols

The [Label] window permits you to display the symbol information included in a module.

Choose [Symbols → Label] from the [View] menu or click the [Label] toolbar button . The [Label] window shown below will be displayed. Use this window to look at the symbol information included in a module.




BP	Address	Name
	00001000	_g_CharBuf
	00001004	_heap_area
	00001404	_g_IntBuf
	00001408	__CTBL
	00001420	_brk
	00001424	__Aldata
	0000142C	__rnext
	FFFF8000	_PowerON_Reset_PC
	FFFF8100	__sbrk_size
	FFFF8104	__DTBL
	FFFF8128	__BTBL
	FFFF8568	_sbrk
	FFFF85BD	_init
	FFFF85FF	_sort
	FFFF8725	_change
	FFFF878F	_main
	FFFF879E	_tutorial
	FFFF885C	_free

Figure 3.11 [Label] Window

3.10 Checking Memory Contents

Specifying a label name, you can check in the [Memory] window the content of memory where the label is registered. For example, you can check the content of memory corresponding to `_main` in byte unit, as shown below.

Choose [CPU -> Memory] from the [View] menu or click the [Memory] toolbar button  to display the [Display Address] dialog box.

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

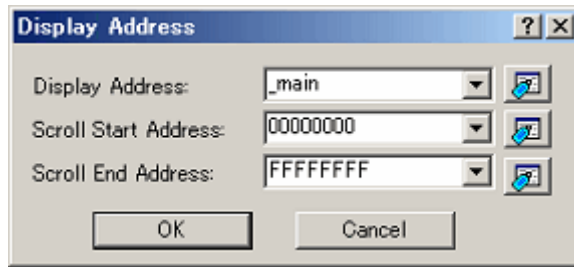


Figure 3.12 [Display Address] Dialog Box

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

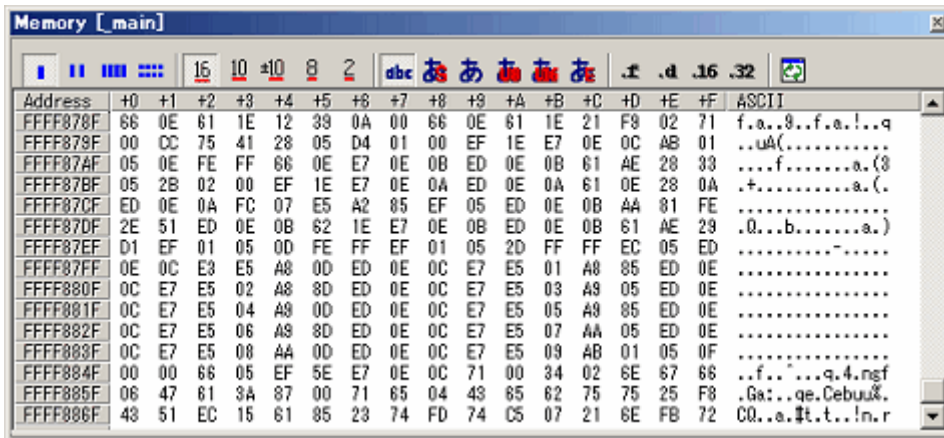


Figure 3.13 [Memory] Window

3.11 Referencing Variables

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

Click the left-hand side of the array 'a' displayed in the [Editor] window and place the cursor there. Select [Instant Watch] with the right mouse button. The dialog box shown below will be displayed.

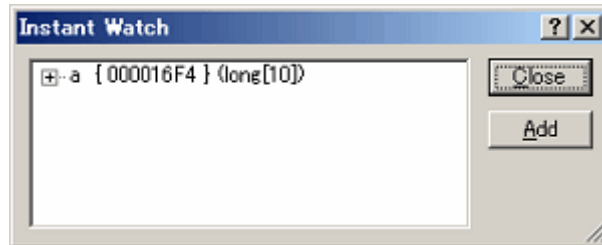


Figure 3.14 [Instant Watch] Dialog Box

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

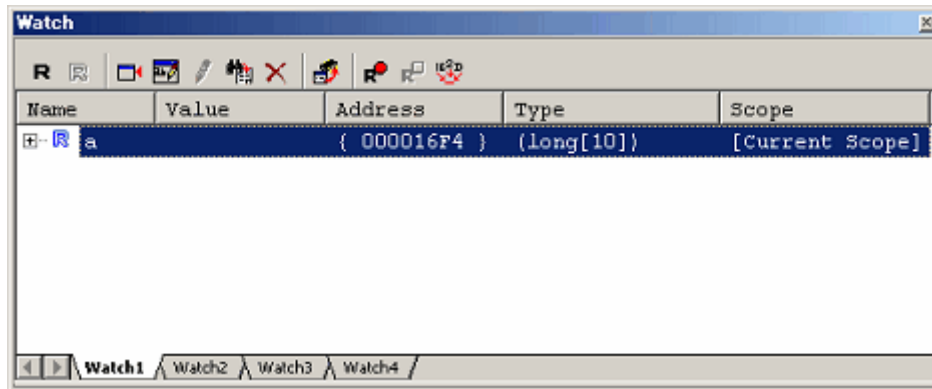


Figure 3.15 [Watch] Window (Array Display)

Or you can specify a variable name to add a variable to the [Watch] window. Click the right mouse button in the [Watch] window and choose [Add Watch] from the pop-up menu. The dialog box shown below will be displayed. Enter a variable 'i' in the [Variable or expression] edit box and click the [OK] button.

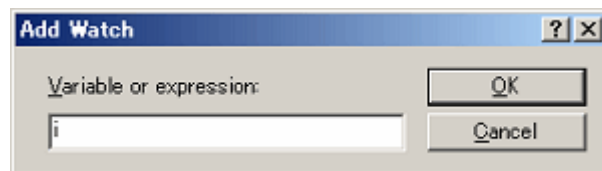


Figure 3.16 [Add Watch] Dialog Box

An int-type variable 'i' will be displayed in the [Watch] window.

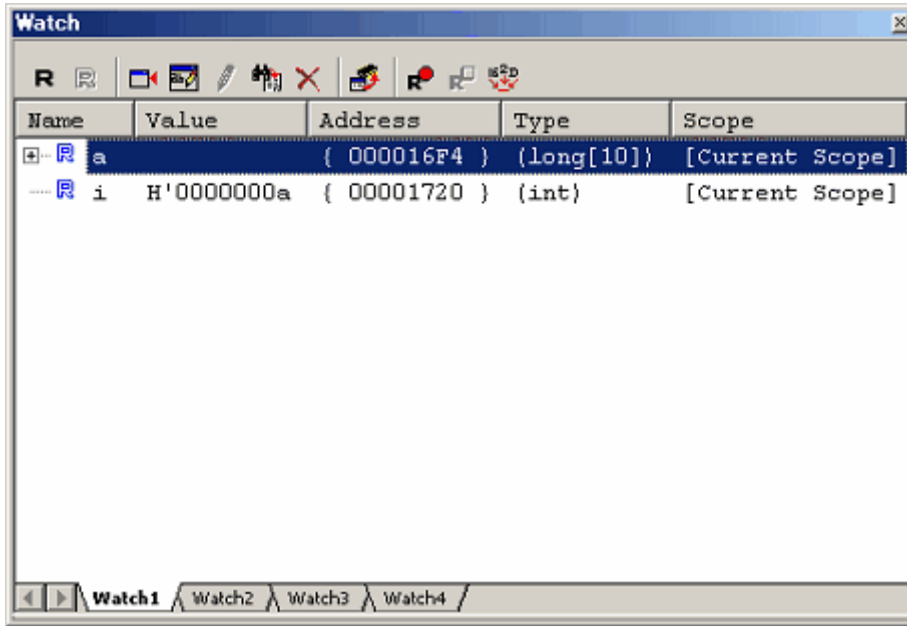


Figure 3.17 [Watch] Window (Showing a Variable)

Clicking the "+" mark shown to the left of the array 'a' in the [Watch] window, you can look at each element of the array 'a.'

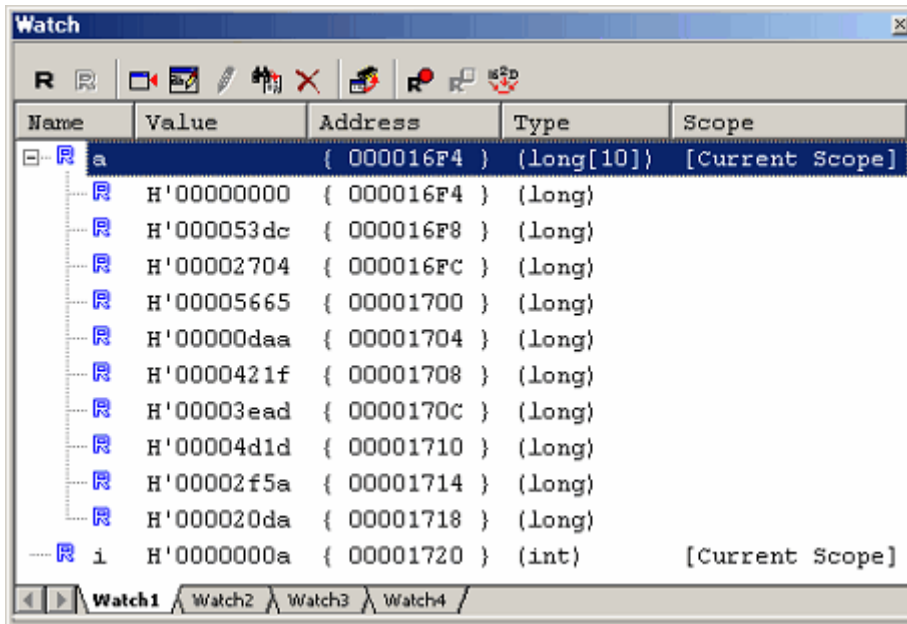



Figure 3.18 [Watch] Window (Showing Array Elements)

3.12 Showing Local Variables

Using the [Locals] window you can display the local variables included in a function. As an example, check the local variables of the tutorial function. This function declares four local variables 'a,' 'j,' 'i' and 'p_sam.'

Choose [Symbols -> Locals] from the View menu or click the Locals toolbar button  to display the Locals window.

The [Locals] window shows the local variables and the values of the function indicated by the current program counter (PC).

If no variables exist in the function, no information is displayed in the [Locals] window.

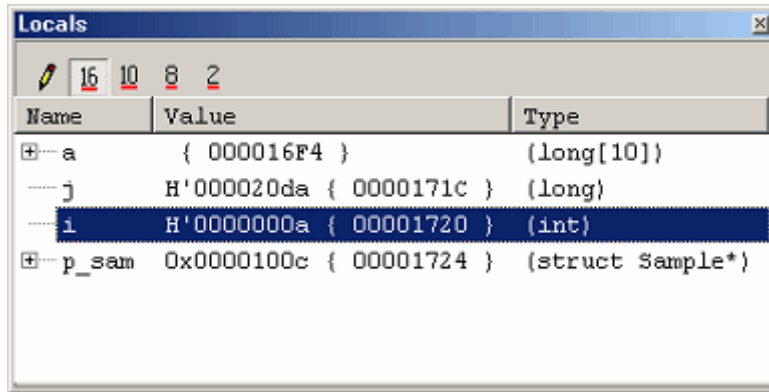


Figure 3.19 [Locals] Window

Click the "+" mark shown to the left of the array 'a' in the [Locals] window to display the elements comprising the array 'a'.

Before the sort function is executed, the array has random data placed in it.

3.13 Single-Stepping a Program

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

Table 3.1 Step Options

Item No.	Command	Description
1	Step In	Executes a program one statement at a time (including statements in a function).
2	Step Over	Executes a program one statement at a time by 'stepping over' a function call if any.
3	Step Out	After exiting a function, stops at the next statement of a program that called the function.
4	Step...	Single-step a program a specified number of times at a specified speed.

3.13.1 Executing the [Step In] Command

The [Step In] command 'steps in' a called function and stops at the first statement of the called function.

To enter the sort function, choose [Step In] from the [Debug] menu or click the [Step In] button in the toolbar.



Figure 3.20 [Step In] Button

Line	Source A.	O.	E.	C.	S.	Source
8	FFFF05BD					void init(struct Sample #p_san)
9						{
10	FFFF05C1					p_san->s0 = 0;
11	FFFF05C8					p_san->s1 = 0;
12	FFFF05CC					p_san->s2 = 0;
13	FFFF05D2					p_san->s3 = 0;
14	FFFF05D8					p_san->s4 = 0;
15	FFFF05DE					p_san->s5 = 0;
16	FFFF05E4					p_san->s6 = 0;
17	FFFF05EA					p_san->s7 = 0;
18	FFFF05F0					p_san->s8 = 0;
19	FFFF05F6					p_san->s9 = 0;
20						}
21						
22						//-----
23						int g_IntBuf;
24						char g_CharBuf;
25						//-----
26						
27	FFFF05FF					void sort(long *a)
28						{
29						long t;
30						int i, j, k, gap;
31						
32	FFFF0604					gap = 5;
33	FFFF0609					while(gap > 0){
34	FFFF0613					for(k=0; k<gap; k++){
35	FFFF0624					for(i=k+gap; i<10; i=i+gap){
36	FFFF0638					for(j=i-gap; j>=k; j=j-gap){
37	FFFF064B					g_IntBuf = j;
38	FFFF0658					if(a[j]>a[j+gap]){
39	FFFF0677					t = a[j];
40	FFFF0684					a[j] = a[j+gap];
41	FFFF069F					a[j+gap] = t;
42						}
43						else break;
44						}
45						}
46						}
47						gap = gap/2;
48	FFFF06EC					}
49	FFFF0705					g_CharBuf = (char)g_IntBuf & 0x00FF;
51						}

Figure 3.21 [Editor] Window (Step In)

The highlighting in the [Editor] window moves to the first statement of the sort function.

3.13.2 Executing the [Step Out] Command

The [Step Out] command exits a called function by executing it quickly and stops at the next statement of a program from which the function was called.

To exit the sort function, choose [Step Out] from the [Debug] menu or click the [Step Out] button in the toolbar.



Figure 3.22 [Step Out] Button

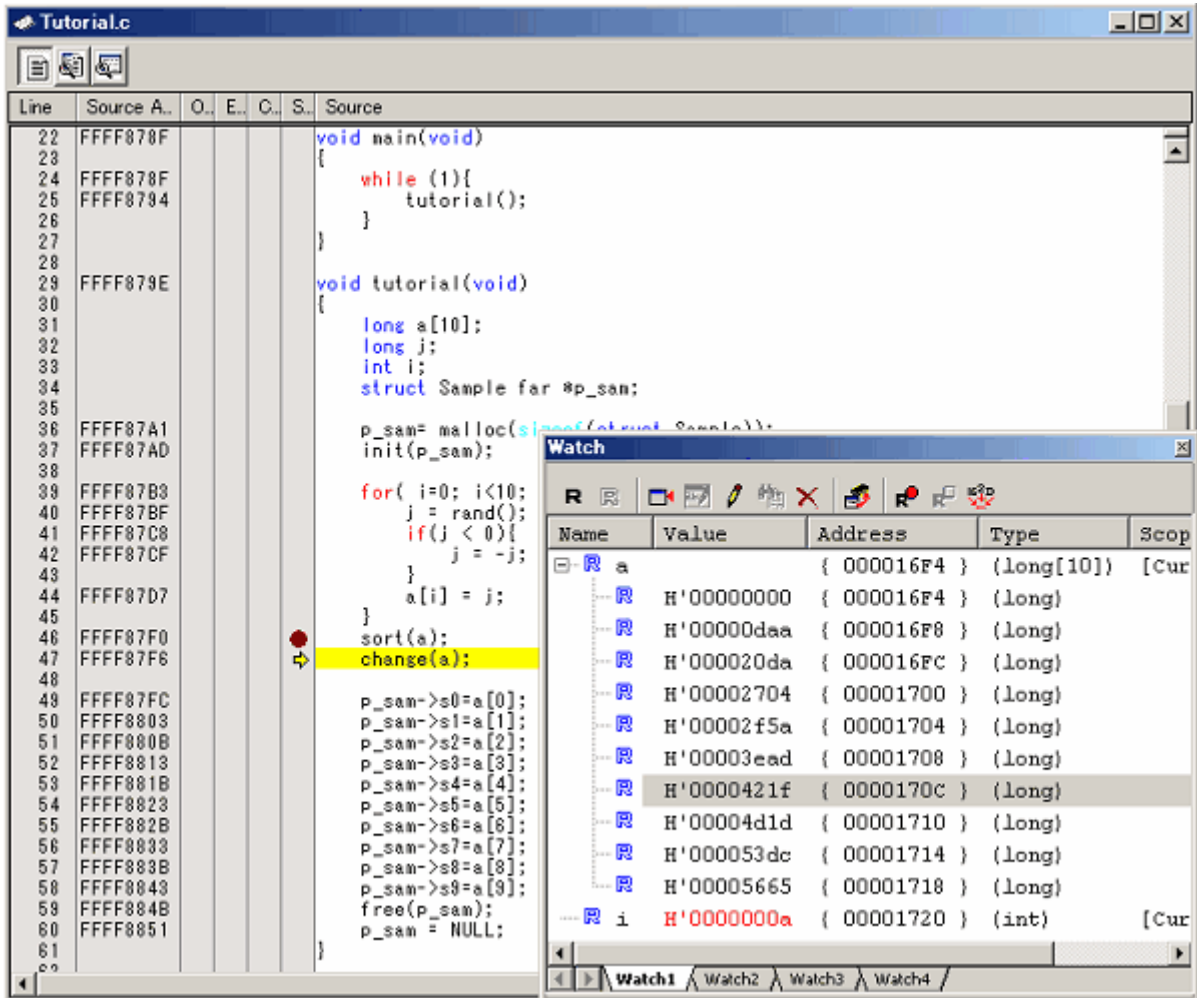


Figure 3.23 [Editor] Window (Step Out)

The data of the variable 'a' displayed in the [Watch] window will be sorted in ascending order.

3.13.3 Executing the [Step Over] Command

The [Step Over] command 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 a time, choose [Step Over] from the [Debug] menu or click the [Step Over] button in the toolbar.



Figure 3.24 [Step Over] Button

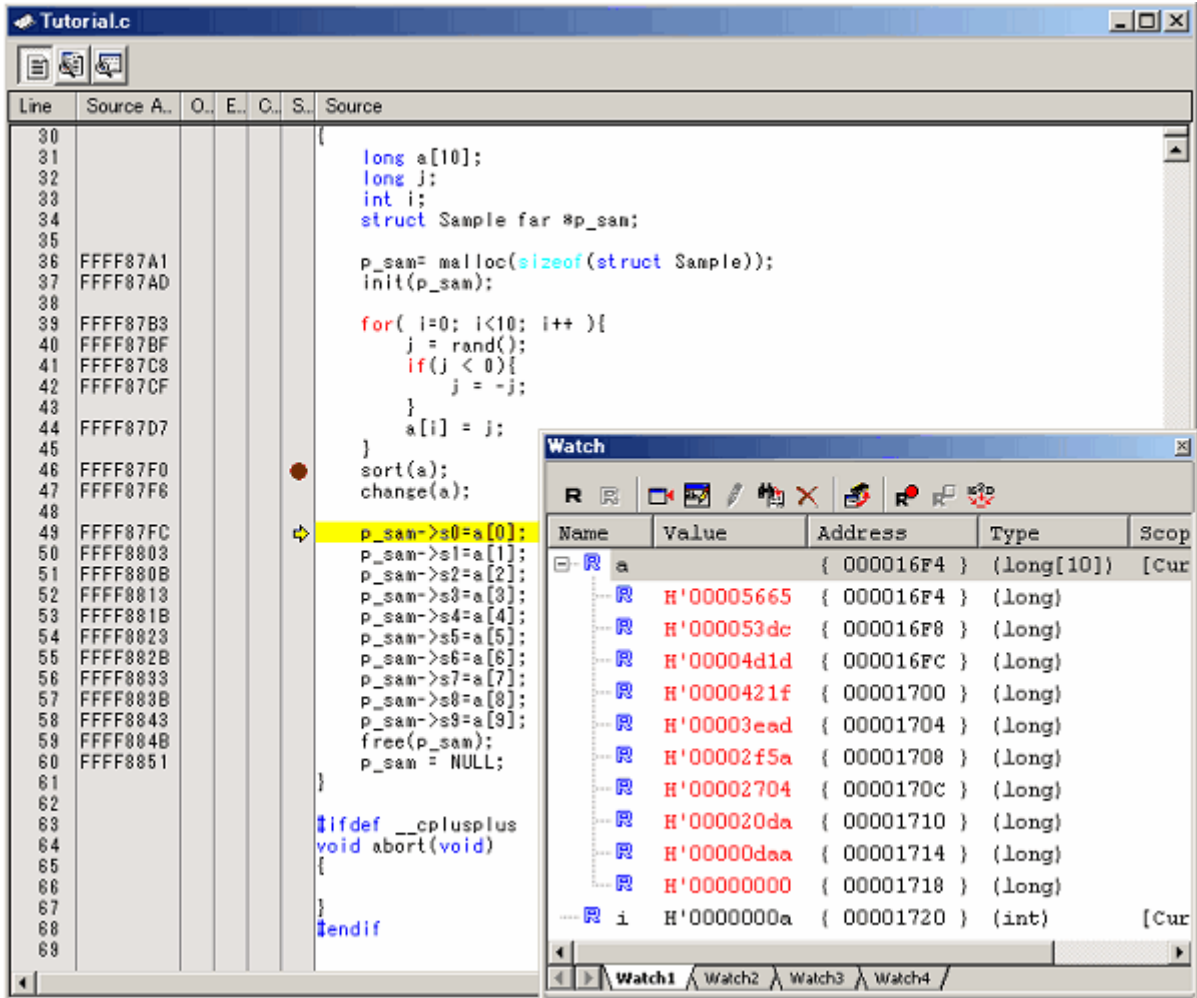


Figure 3.25 [Editor] Window (Step Over)

The data of the variable 'a' displayed in the [Watch] window will be sorted in descending order.

3.14 Forcibly Breaking a Program

The High-performance Embedded Workshop permits you to forcibly break a program.

Clear all breakpoints.

To execute the rest of the tutorial function, choose [Go] from the [Debug] menu or click the [Go] button in the toolbar.



Figure 3.26 [Go] Button

Since the program is executing an infinite loop process, choose [Stop Program] from the [Debug] menu or click the [Stop] button in the toolbar.



Figure 3.27 [Stop] Button

3.15 Hardware Break Facility

Hardware breaks cause the program to stop when it executes a specified address (instruction fetch) or reads or writes to a specified memory location (data access).

3.15.1 Stopping a Program when It Executes a Specified Address

The [Editor] window permits you to set an instruction fetch event easily. For example, you can set an instruction fetch event at a place where the sort function is called.

Double-click a row in the [Event] column corresponding to the source line that includes a sort function call.

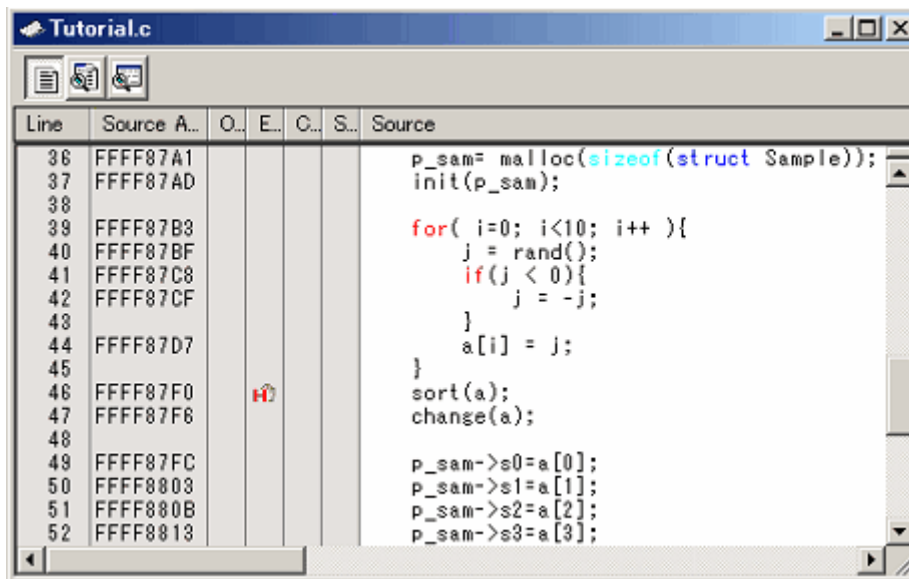




Figure 3.28 [Editor] Window (Setting a Hardware Breakpoint)

The source line that includes the sort function will be marked with , indicating that a hardware breakpoint that will cause a program to stop when it fetches an instruction has been set there.

When the program is executed, it stops running after passing the set hardware breakpoint.

Double-click  in the [Event] column again to clear the hardware breakpoint.

3.16 Stopping a Program when It Accesses Memory

To stop a program when it reads or writes a value to a global variable, set up a hardware break as described below.

Choose [Event -> Hardware Break] from the [View] menu to display the [Hardware Break] dialog box.

Open the [OR] page of the [Hardware Break] dialog box. In the [Editor] window, select a global variable that you want to be the object of a hardware break so that a program is made to stop when it reads or writes a value to the variable, and drag-and-drop the selected variable into the [OR] page.

Then click the [Apply] button.

When you run a program, it will stop running when a value is read or written to the global variable you have set.

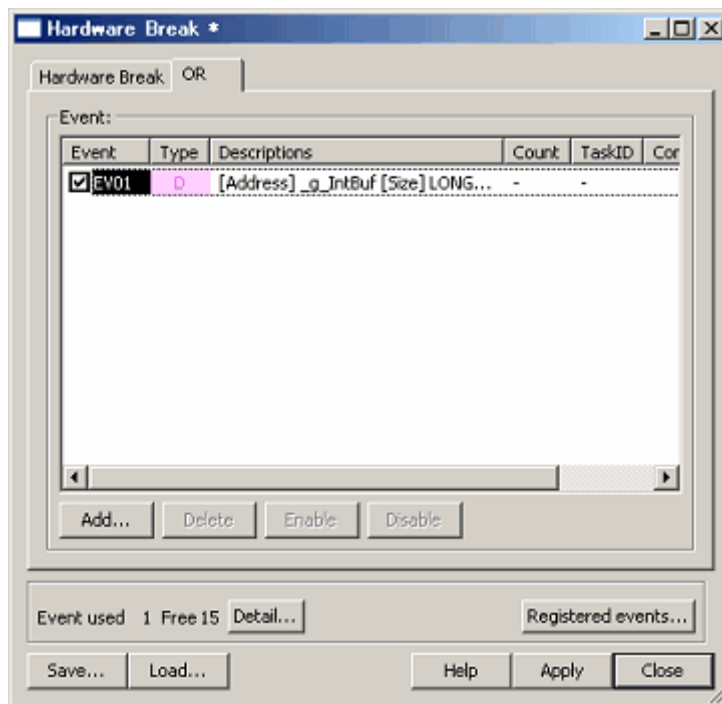



Figure 3.29 [Hardware Break] Dialog Box

- Notes:
1. Only global or static variables within the access size of MCU range (1, 2, and 4 bytes in case of RX610) can be registered.
 2. Static variables in functions cannot be registered..
 3. Local variables outside scope cannot be registered. Moreover, when the local variable is set, the break not intended outside scope might be generated.

3.17 Trace Facility

The trace facility of the E100 emulator has a special memory known as “trace memory” that can hold an execution record of up to 4-M bus cycles, which is always updated during program execution. The content of trace memory is displayed in the [Trace] window.

Choose [Code → Trace] from the [View] menu or click the [Trace] toolbar button .

The [Trace] window shown below will be displayed.

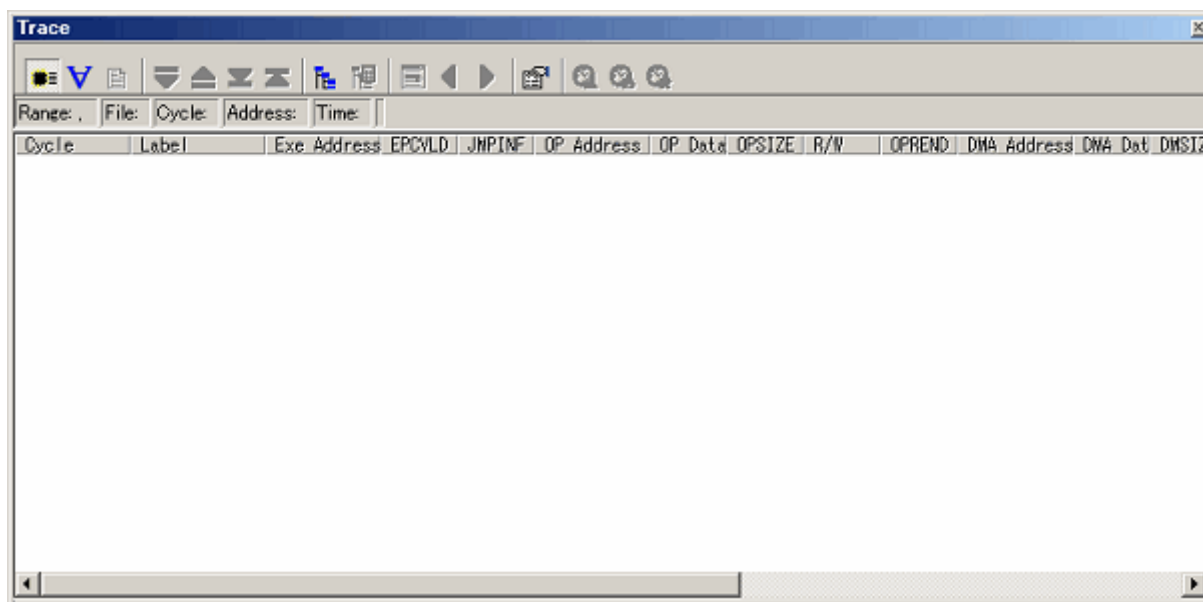


Figure 3.30 [Trace] Window

The following outlines the trace facility and describes how to set.

3.17.1 Showing the Trace Information Acquired by Fill until Stop

The free trace facility acquires trace information successively from when the user program starts running till when it breaks.

- (1) Clear all break conditions. Click the right mouse button anywhere in the [Trace] window and choose [Acquisition] from the pop-up menu that is displayed. The [Trace conditions] dialog box shown below will be displayed. Check to see that the selected trace mode is [Fill until stop]. Click the [Close] button.

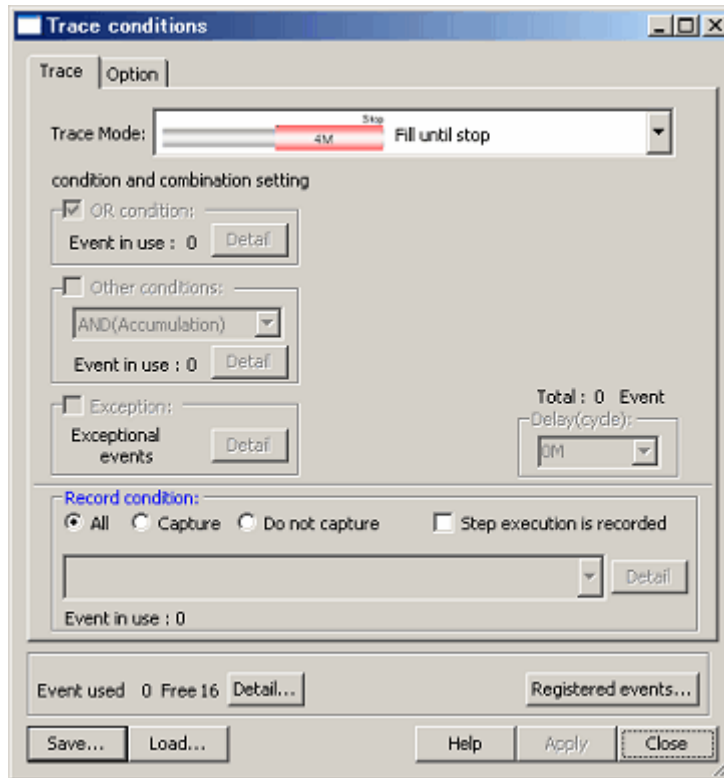


Figure 3.31 [Trace conditions] Dialog Box (Free Trace)

- (2) Set a software break in a line of the tutorial function where 'p_sam ->s0=a[0];' is written.

- (3) Choose [Reset Go] from the [Debug] menu. Processing will be halted by a break, and the trace information from start to break will be displayed in the [Trace] window.

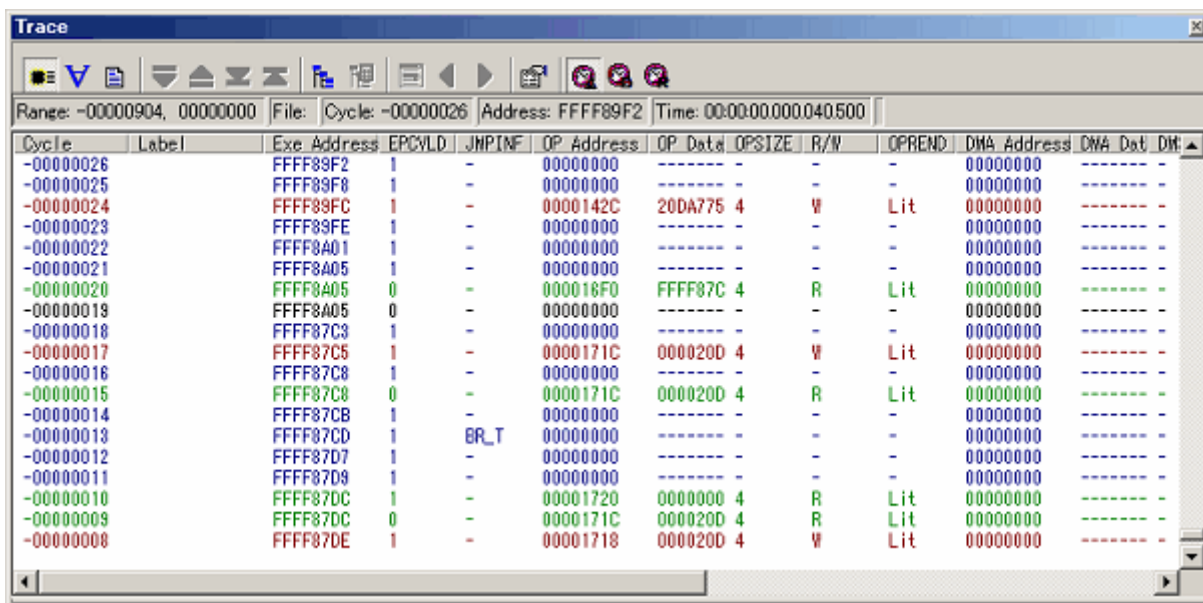


Figure 3.32 [Trace] Window (Free Trace)

- (4) A mixed display of bus, disassemble and/or source display is possible. Choosing [Display Mode -> DIS] from the pop-up menu, you can display trace information in a bus and disassemble mixed mode.

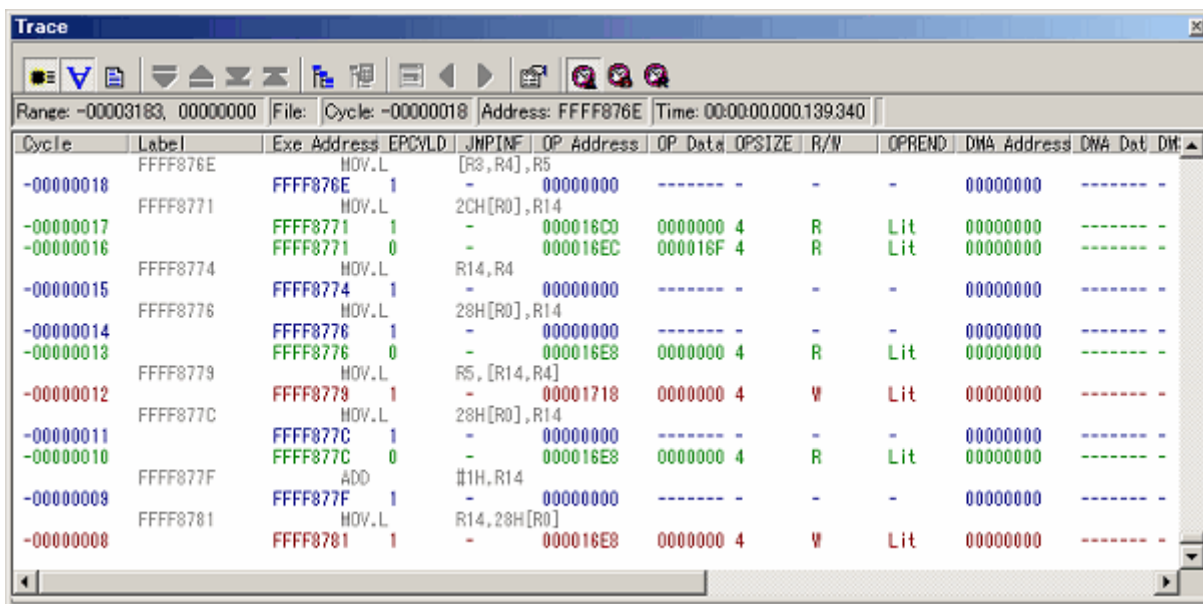


Figure 3.33 [Trace] Window (Bus and Disassemble Mixed Display)

(5) Furthermore, choosing [Display Mode → SRC] from the pop-up menu, you can display trace information in a bus, disassemble and source mixed mode.

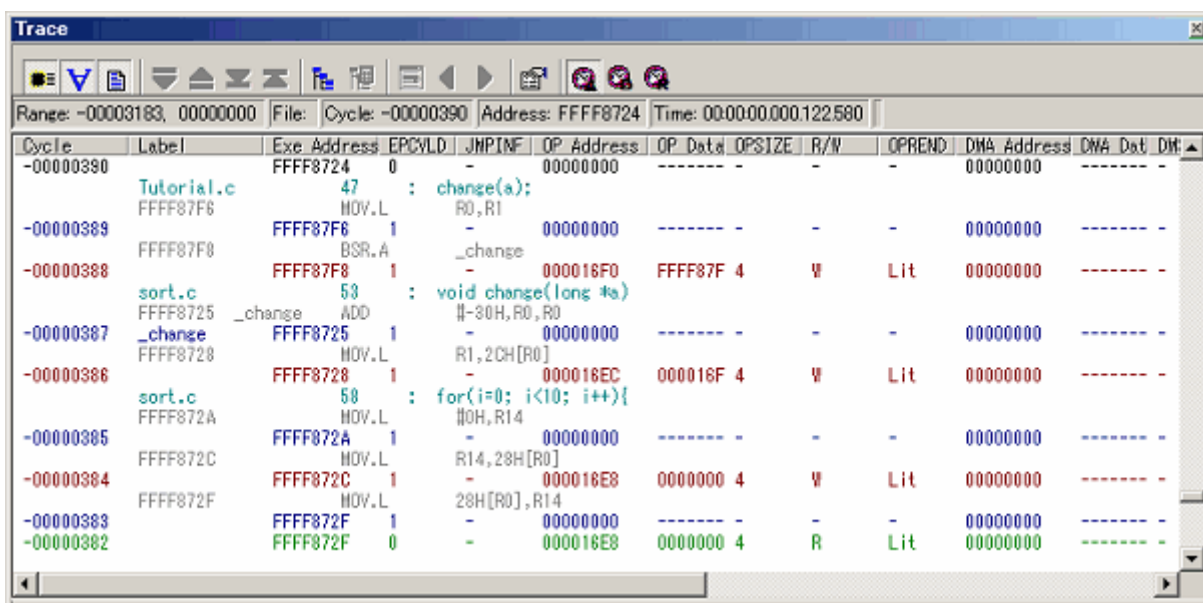


Figure 3.34 [Trace] Window (Bus, Disassemble and Source Mixed Display)

3.17.2 Showing the Trace Information Acquired by Fill around TP

The point & delay facility stops acquiring trace information a specified number of cycles after a trace point is encountered. This facility allows you to keep track of program flow from trace information without having to break the user program.

- (1) If any break conditions are set, clear all of them.
- (2) Choose [Fill around TP] for trace mode in the [Trace conditions] dialog box. In the [Delay (cycle)] column, specify [4M]. (Up to 4-M cycles of trace information from where a trace point is encountered will be acquired.)

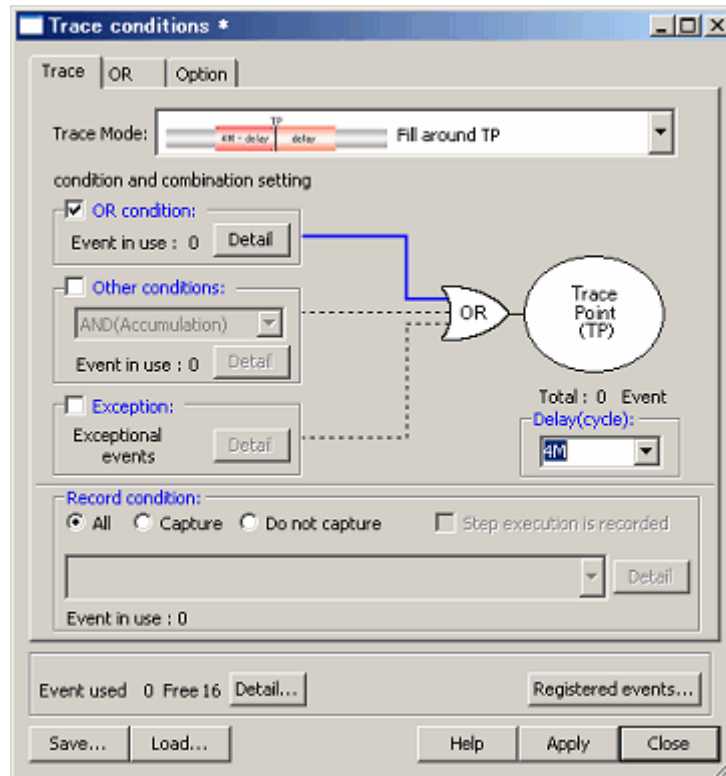


Figure 3.35 [Trace conditions] Dialog Box (Fill around TP)

- (3) Next, set a trace point at which the debugger starts acquiring trace information. Open the [OR] page of the [Trace conditions] dialog box. Select the main function in the [Editor] window and drag-and-drop it into the [OR] page. Click the [Apply] button and then the [Close] button.

Thus, the debugger will start acquiring trace information from when the main function is executed.

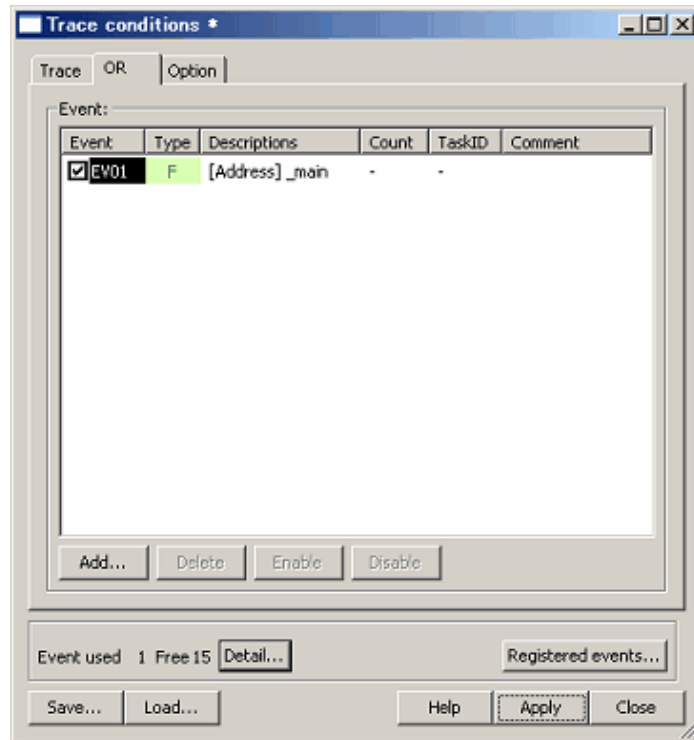


Figure 3.36 [Trace conditions] Dialog Box ([OR] Page)

- (4) Choose [Reset Go] from the [Debug] menu. A short time after a trace point is reached, the trace content shown below will be displayed in the [Trace] window.

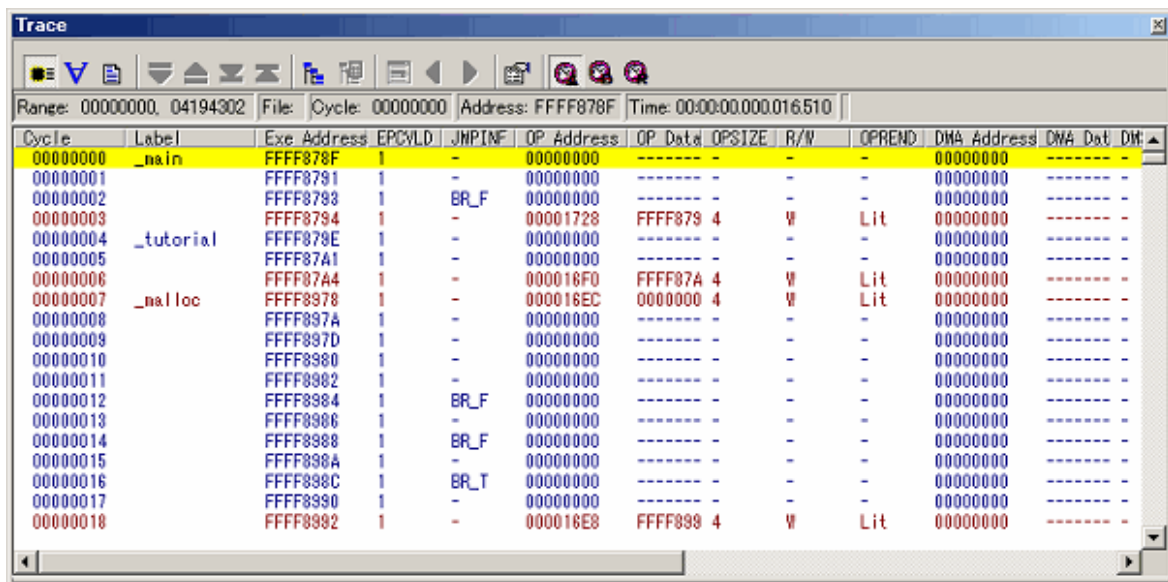


Figure 3.37 [Trace] Window (Fill around TP)

3.17.3 Showing a Function Execution History

A function execution history can be displayed from the acquired trace information.

- (1) Clear all break conditions. Click the right mouse button anywhere in the [Trace] window and choose [Acquisition] from the pop-up menu that is displayed. The [Trace conditions] dialog box will be displayed. Switch the trace mode to [Fill until stop] and click the [Apply] button. Then click the [Close] button.
- (2) Set a software break in a line of the tutorial function where 'p_sam->s0=a[0];' is written.
- (3) Choose [Reset Go] from the [Debug] menu. Processing will be halted by a break, and the trace information from start to break will be displayed in the [Trace] window.
- (4) Click the right mouse button anywhere in the [Trace] window and choose [Function Execution History -> Function Execution History] from the pop-up menu that is displayed.

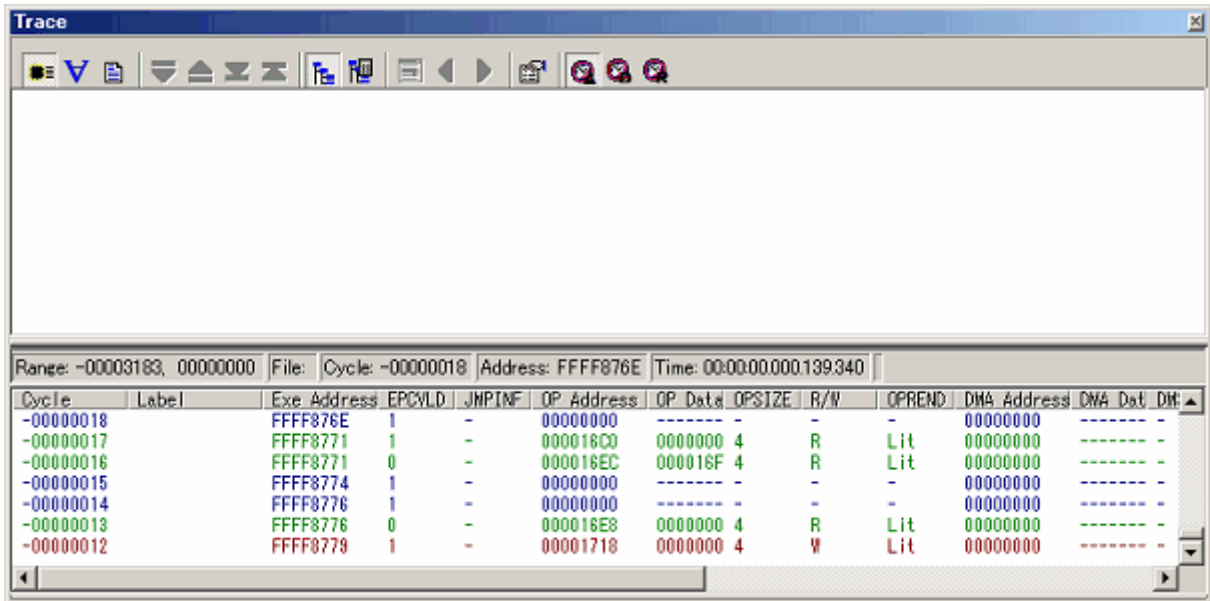


Figure 3.38 [Trace] Window (Function Execution History–Before Analysis)

- (5) Click the right mouse button anywhere in the displayed function execution history window and choose [Analyze Execution History] from the pop-up menu. A function execution history will be displayed in the upper pane of the [Trace] window.

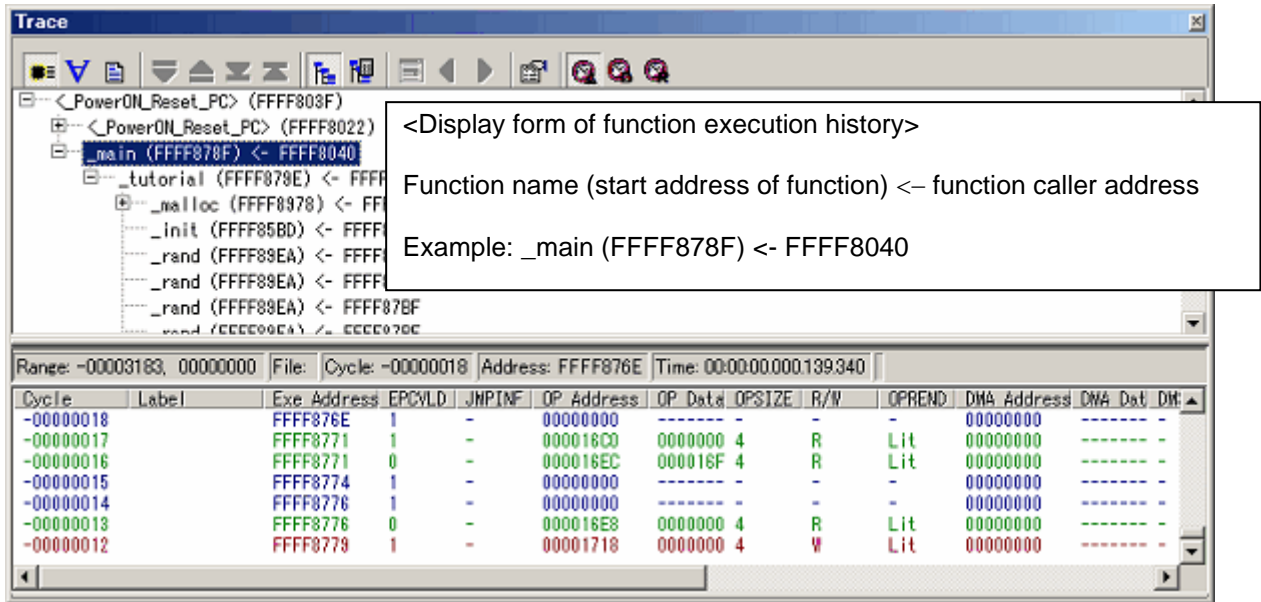


Figure 3.39 [Trace] Window (Function Execution History–After Analysis)

- (6) Double-click any function in the displayed function execution history, and the trace information corresponding to that function will be displayed in the lower pane of the [Trace] window.

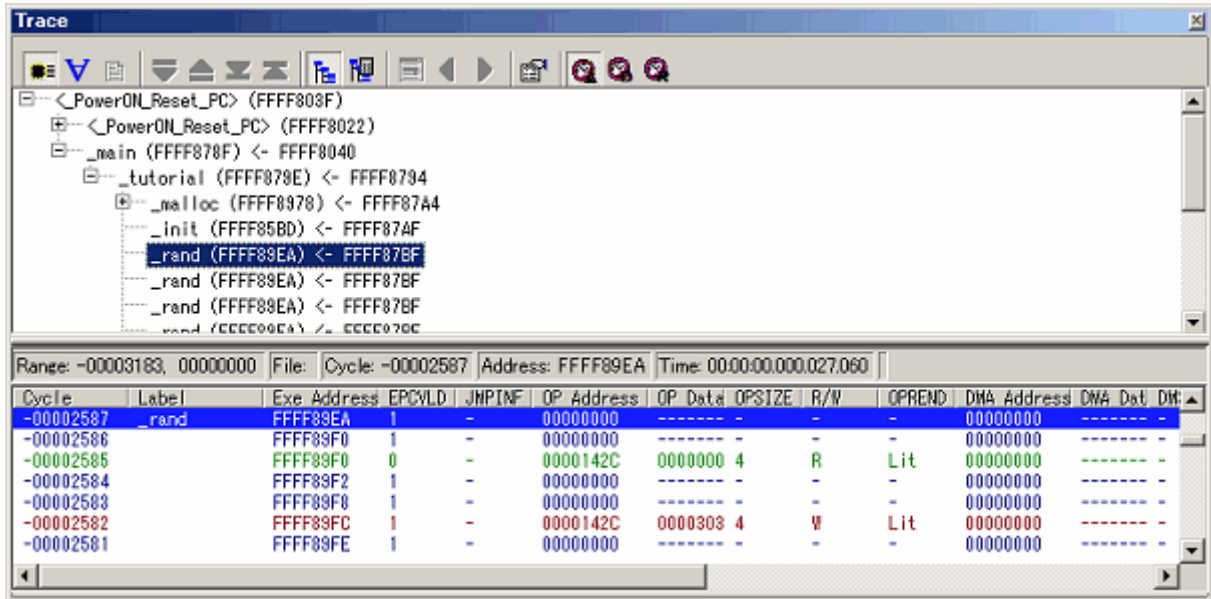


Figure 3.40 [Trace] Window (Function Execution History)

3.17.4 Filter Facility

Use the filter facility to extract only the necessary cycles from the acquired trace information.

The filter facility does this by filtering the trace information in software that was acquired by hardware.

Unlike the "Capture/Do not Capture conditions" where you set acquisition conditions before getting trace information, this facility allows you to change filter settings for the acquired trace information any number of times without having to reexecute. Therefore, the necessary information can be extracted easily.

- (1) Clear all break conditions. Click the right mouse button anywhere in the [Trace] window and choose [Acquisition] from the pop-up menu that is displayed. The [Trace conditions] dialog box will be displayed. Check to see that the selected trace mode is [Fill until stop]. Click the [Close] button.
- (2) Set a software break in a line of the tutorial function where 'p-sam->s0=a[0];' is written.
- (3) Choose [Reset Go] from the [Debug] menu. Processing will be halted by a break, and the trace information from start to break will be displayed in the [Trace] window.
- (4) Choose [Auto Filter] from the pop-up menu of the [Trace] window. The columns for which filtering can be applied will be marked by a button.

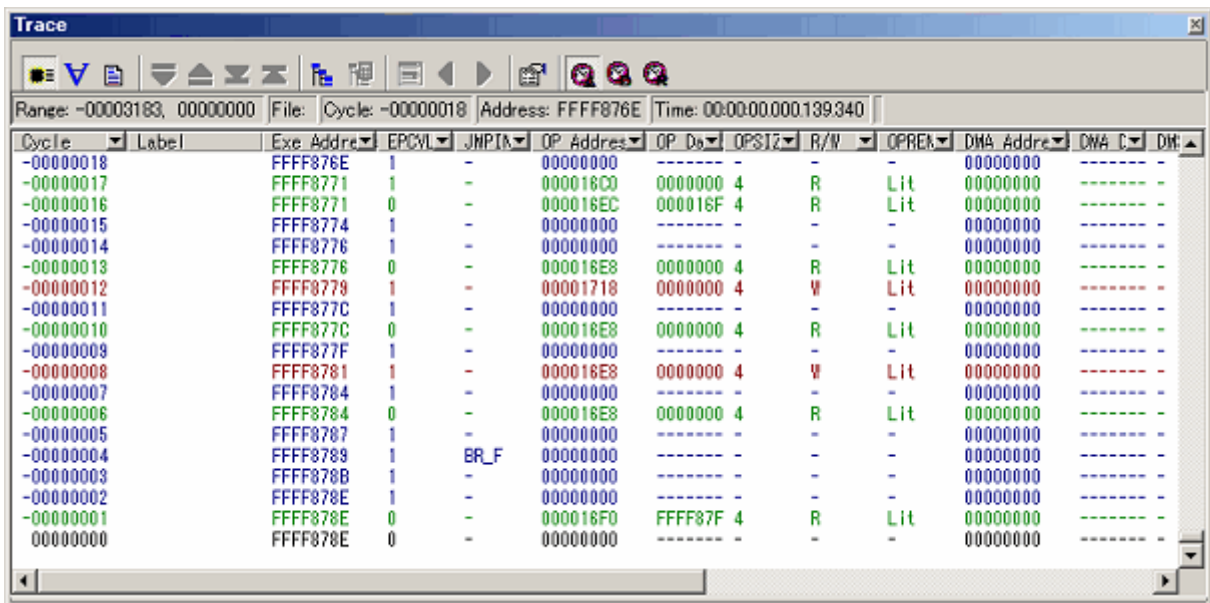



Figure 3.41 [Trace] Window (Auto Filter)

(5) Click the  button in the [R/W] column and choose [R] from the pop-up menu.

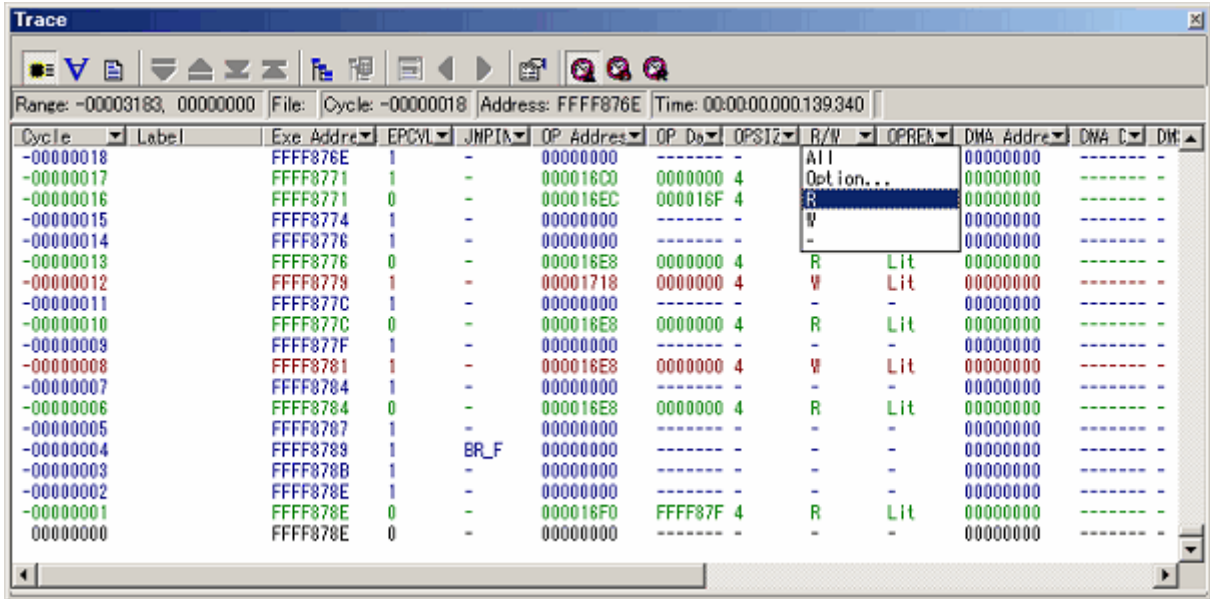


Figure 3.42 [Trace] Window (Auto Filter)

(6) That way, the trace information for only [R] in the [R/W] column can be displayed.

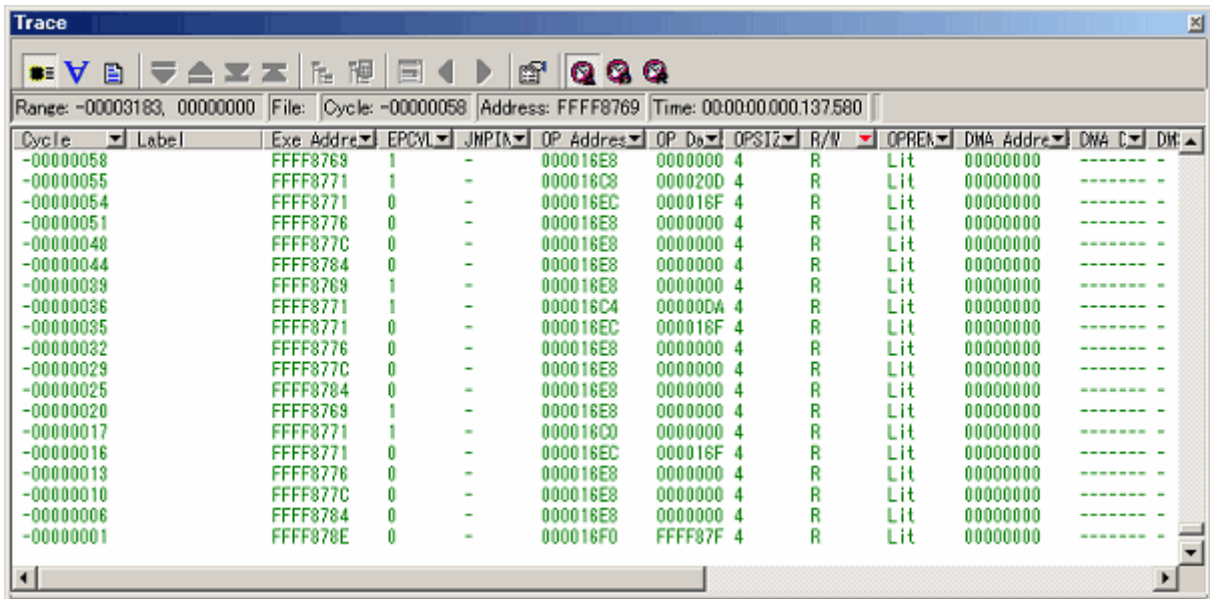


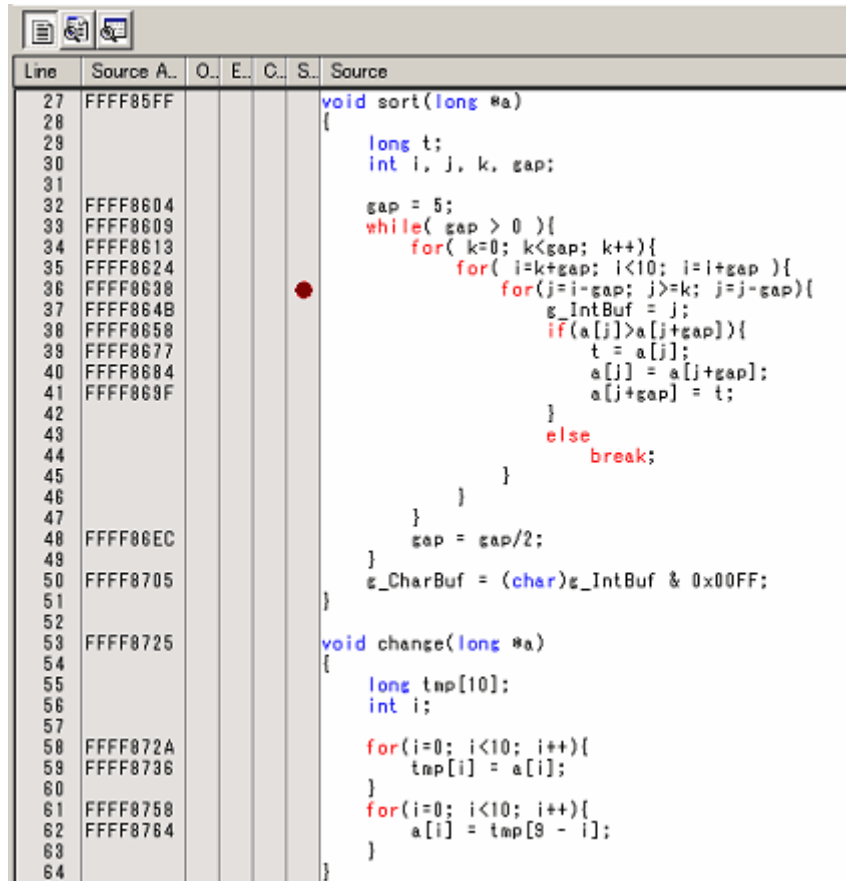
Figure 3.43 [Trace] Window (Auto Filter)

- Notes:
1. The filter function does not affect the trace memory, so that its content remains intact.
 2. The filter can be used when the selected trace mode is [Fill until stop], [Fill until full] or [Fill around TP].

3.18 Stack Trace Facility

Using stack information, it is possible to show which function is the caller to the function where the current PC exists.

Set a software breakpoint in any line of the sort function by double-clicking at its corresponding row in the [S/W Breakpoints] column.

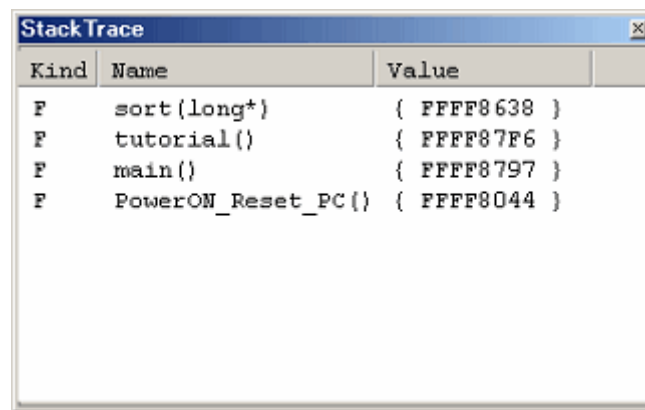


Line	Source A.	O.	E.	C.	S.	Source
27	FFFF85FF					void sort(long *a)
28						{
29						long t;
30						int i, j, k, gap;
31						
32	FFFF8604					gap = 5;
33	FFFF8609					while(gap > 0){
34	FFFF8613					for(k=0; k<gap; k++){
35	FFFF8624					for(i=k+gap; i<10; i=i+gap){
36	FFFF8638					for(j=i-gap; j>=k; j=j-gap){
37	FFFF864B					g_IntBuf = j;
38	FFFF8658					if(a[j]>a[j+gap]){
39	FFFF8677					t = a[j];
40	FFFF8684					a[j] = a[j+gap];
41	FFFF869F					a[j+gap] = t;
42						}
43						}
44						else break;
45						}
46						}
47						gap = gap/2;
48	FFFF86EC					}
49	FFFF8705					g_CharBuf = (char)g_IntBuf & 0x00FF;
50						}
51						
52						
53	FFFF8725					void change(long *a)
54						{
55						long tmp[10];
56						int i;
57						
58	FFFF872A					for(i=0; i<10; i++){
59	FFFF8736					tmp[i] = a[i];
60						}
61	FFFF8758					for(i=0; i<10; i++){
62	FFFF8764					a[i] = tmp[9 - i];
63						}
64						}

Figure 3.44 [Editor] Window (Setting a Software Breakpoint)

Choose [Reset Go] from the [Debug] menu.

After a break, choose [Code → Stack Trace] from the [View] menu to open the [Stack Trace] window.



Kind	Name	Value
F	sort(long*)	{ FFFF8638 }
F	tutorial()	{ FFFF87F6 }
F	main()	{ FFFF8797 }
F	PowerON_Reset_PC()	{ FFFF8044 }

Figure 3.45 [Stack Trace] Window

You will see that the current PC exists within the sort() function, and that the sort() function is called from the tutorial() function.

Clear the software breakpoint that you have set in a line of the sort function by double-clicking at its corresponding row in the [S/W Breakpoints] column again.

3.19 What Next?

In this tutorial, we have introduced to you several features of the E100 emulator and how to use the High-performance Embedded Workshop.

The emulation facility that the E100 emulator provides allows you to perform advanced debugging. Once the conditions that cause hardware or software problems to occur are exactly separated and identified by that debugging, you can examine those problems effectively.

4. Preparing to Debug

4.1 Starting the High-performance Embedded Workshop

Follow the procedure described below to start the High-performance Embedded Workshop.

- (1) Connect the host machine and the E100 Emulator and user system. Then turn on the power to the E100 Emulator and user system.
- (2) From [Programs] on the [Start] menu, choose [Renesas] -> [High-performance Embedded Workshop] -> [High-performance Embedded Workshop].

The [Welcome!] dialog box shown below will appear.

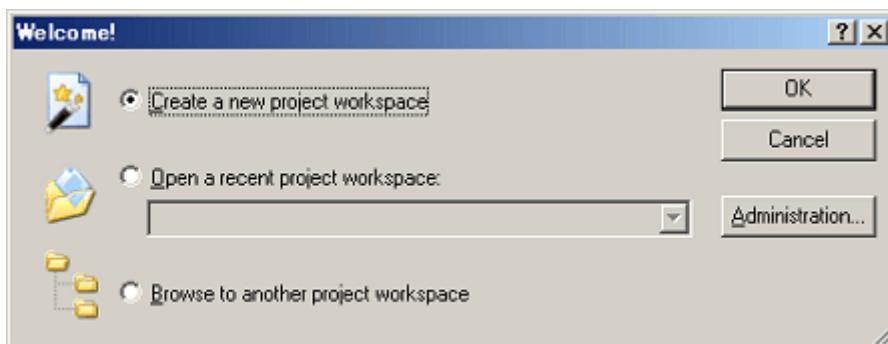


Figure 4.1 [Welcome!] Dialog Box

Select a startup method from the following.

- Create a new project workspace.
- Open a recently used project workspace.
Select this option when you use an existing workspace.
A history of the workspace you open will be displayed.
- Browse another project workspace.
Select this option when you use an existing workspace.
This is the option available to choose when the workspace you opened has no history recorded.

4.2 Creating a New Workspace (Toolchain Unused)

The procedure for creating a new project workspace differs depending on whether you use a toolchain or not.

The E100 emulator has no toolchains included in it. You can use a toolchain in an environment in which the C/C++ compiler package is installed.

Follow the procedure described below to create a new workspace.

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

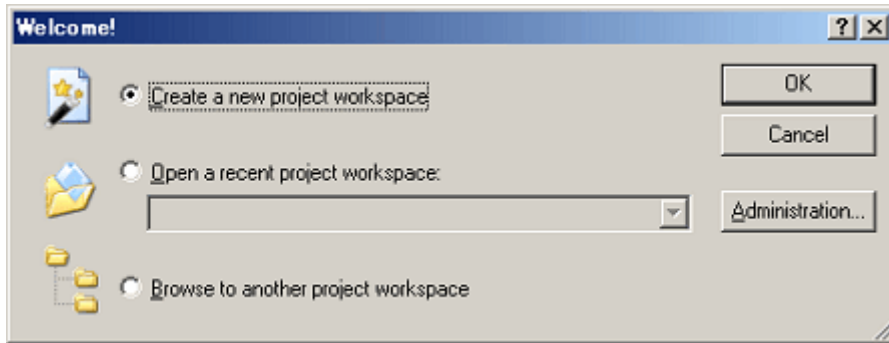


Figure 4.2 [Welcome!] Dialog Box

- (2) Project Generator will start.

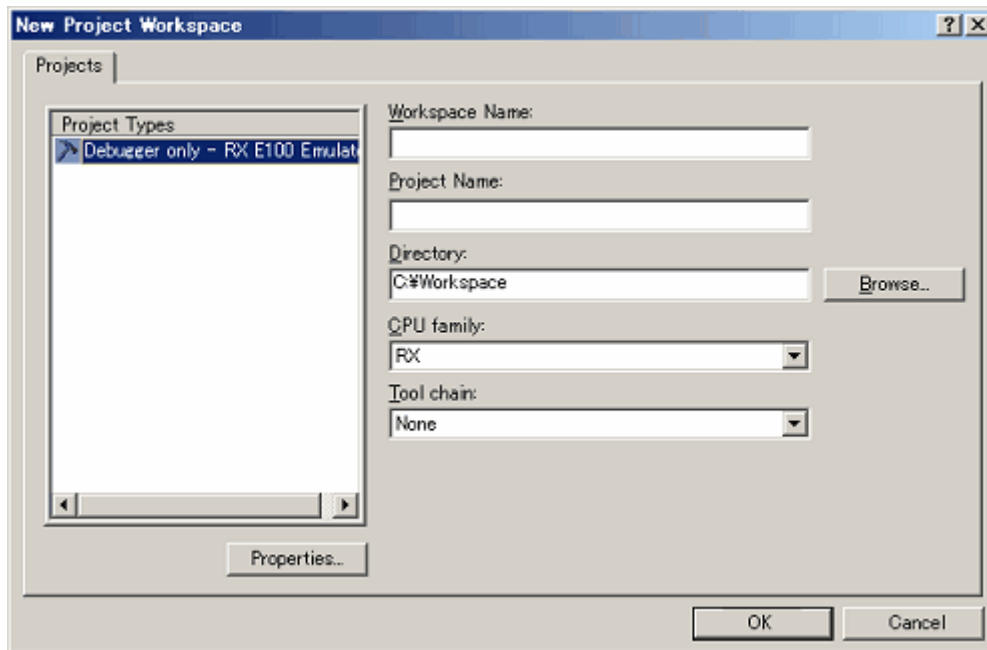


Figure 4.3 [New Project Workspace] Dialog Box

- Workspace Name: Enter a workspace name here.
- Project Name: Enter a project name here. If the same name as a workspace name is good, you do not need to enter it.
- Directory: Enter a directory in which you want a workspace to be created. Or you can click the [Browse] button and select a workspace directory from the ensuing list.
- CPU family: Select the CPU family of the MCU you are using.

The other list boxes are used for setting up a toolchain. If no toolchains are installed, the information specific to the CPU family is displayed here. Click the [OK] button.

(3) Select the debugger target.

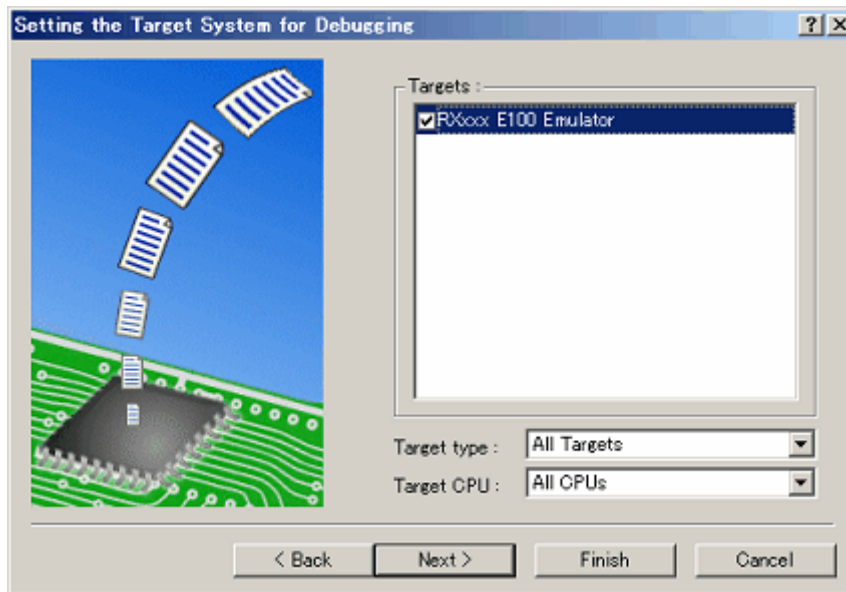


Figure 4.4 [Setting the Target System for Debugging] Dialog Box

Select the target platform you use by placing a check mark in its checkbox and click the [Next] button.

(4) Set a configuration name. Configuration refers to the file in which the High-performance Embedded Workshop status other than the emulator is saved.

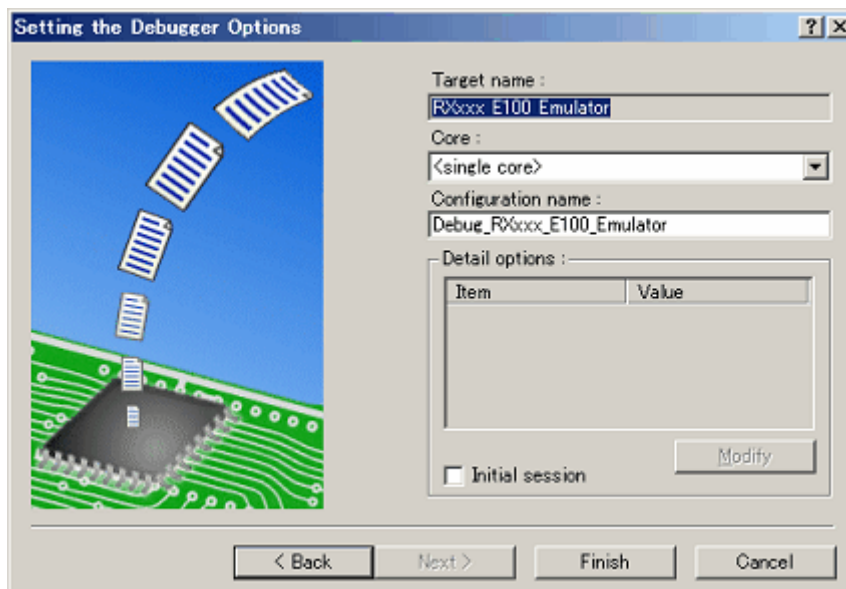


Figure 4.5 [Setting the Debugger Options] Dialog Box

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

When you have finished setting configuration names, emulator-related settings are completed.

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

(5) After starting the High-performance Embedded Workshop, connect the E100 emulator.

4.3 Creating a New Workspace (Toolchain Used)

Follow the procedure described below to create a new workspace.

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

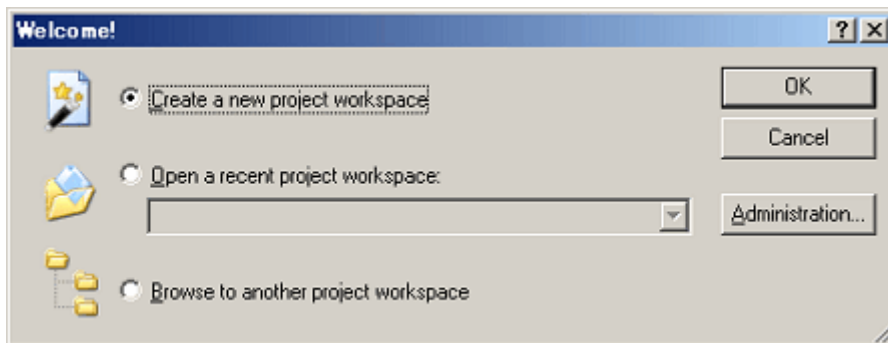


Figure 4.6 [Welcome!] Dialog Box

(2) Project Generator will start.

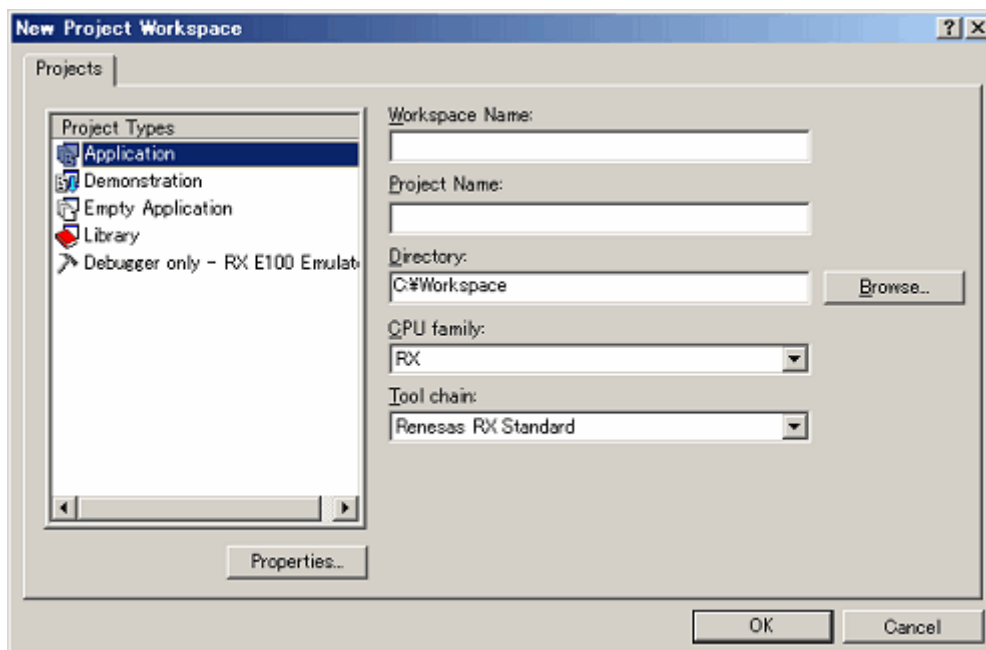


Figure 4.7 [New Project Workspace] Dialog Box

Workspace Name:	Enter a workspace name here.
Project Name:	Enter a project name here. If the same name as a workspace name is good, you do not need to enter it.
Directory:	Enter a directory in which you want a workspace to be created. Or you can click the [Browse] button and select a workspace directory from the ensuing list.
CPU family:	Select the CPU family of the MCU you are using.
Toolchain:	To use a toolchain, select the appropriate toolchain here. If you do not use, select [None].

The other list boxes are used for setting up a toolchain. If no toolchains are installed, the information specific to the CPU family is displayed here. Click the [OK] button.

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

(4) Select the debugger target.

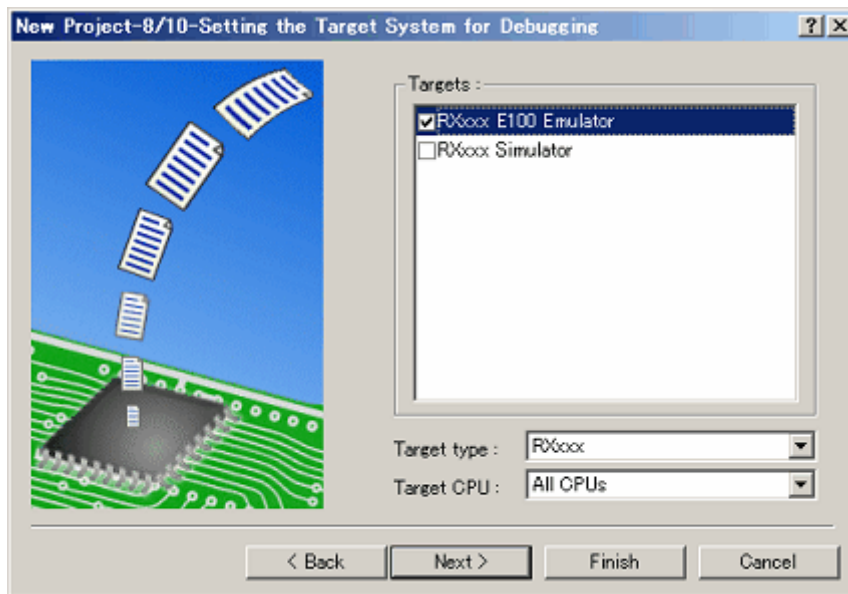


Figure 4.8 [Setting the Target System for Debugging] Dialog Box

Select the target platform you use by placing a check mark in its checkbox and click the [Next] button.

(5) Set a configuration name.

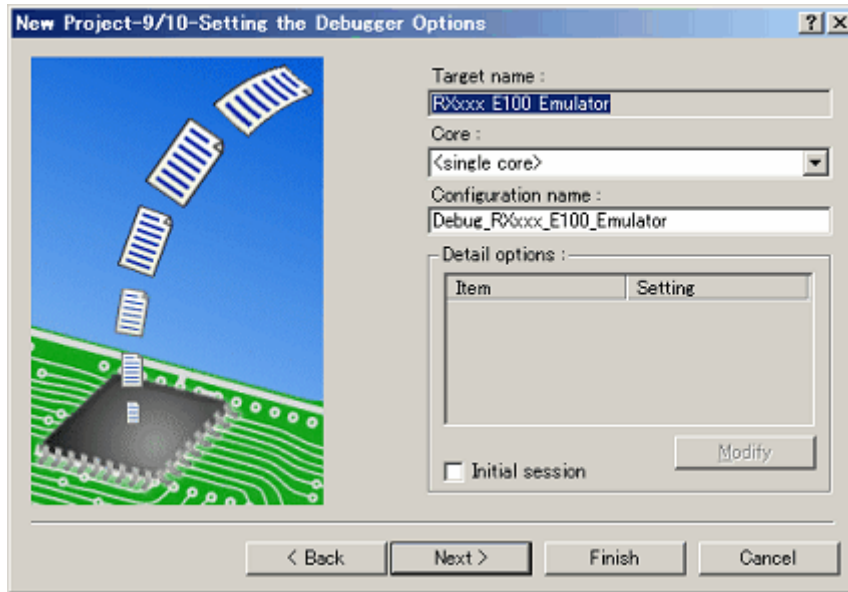


Figure 4.9 [Setting the Debugger Options] Dialog Box

If you have selected two or more target platforms, click the [Next] button and then set a configuration name for each target platform selected. When you have finished setting configuration names, emulator-related settings are completed. Click the [Finish] button, and the [Summary] dialog box will be displayed. Clicking the [OK] button in it starts the High-performance Embedded Workshop.

(6) After starting the High-performance Embedded Workshop, connect the E100 emulator.

4.4 Opening an Existing Workspace

Follow the procedure described below to open an existing workspace.

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

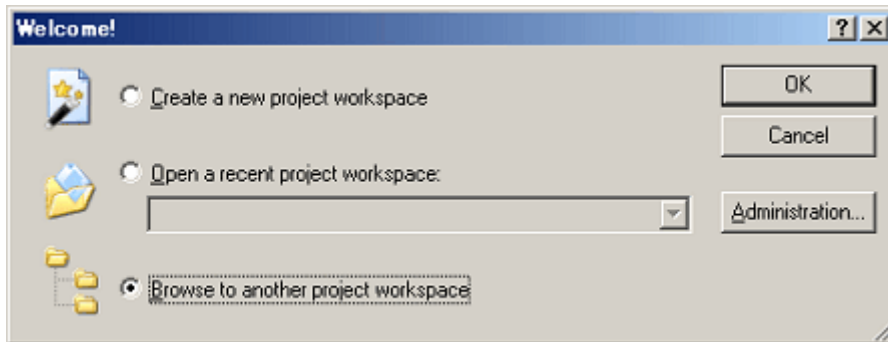


Figure 4.10 [Welcome!] Dialog Box

- (2) The [Open Workspace] dialog box shown below will appear.

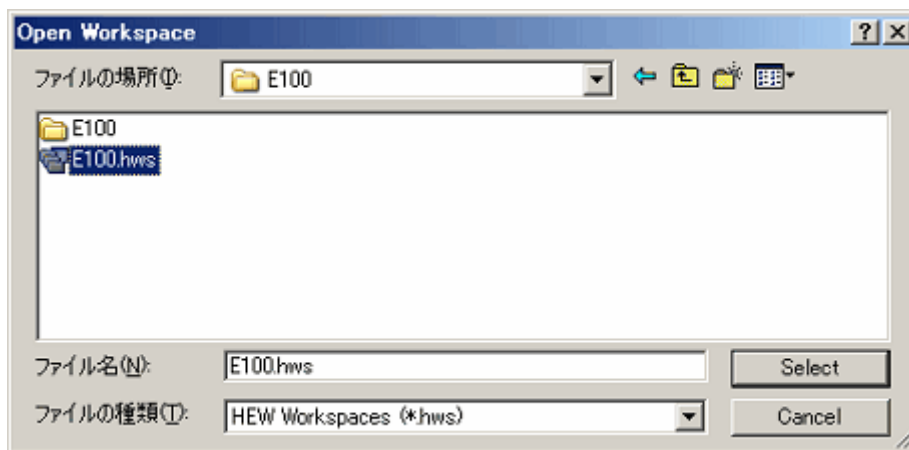


Figure 4.11 [Open Workspace] Dialog Box

Specify the directory in which workspaces are created, select a workspace file (extension “.hws”) and click the [Select] button.

- (3) The High-performance Embedded Workshop will start, and the state of the selected workspace in which it was saved will be restored. If the saved state of the selected workspace is one in which it was connected to the emulator, the workspace is automatically connected to the emulator. If the saved state of the selected workspace is one in which it was not connected to the emulator and you want to connect it, refer to “Connecting the Emulator”.

4.5 Connecting the Emulator

4.5.1 Connecting the Emulator

There are following methods for connecting the emulator.

(1) Setting up the emulator at startup before connecting

Choose [Debug Settings] from the [Debug] menu to open the [Debug Settings] dialog box. In this dialog box, you can register download modules and the command chain to be automatically executed. When you are finished filling in the [Debug Settings] dialog box, the emulator will be connected.


(2) Loading a session file

Switching to the session file that has emulator usage settings preregistered in it helps you connect the emulator easily.

4.5.2 Reconnecting the Emulator

While the emulator is disconnected, you can reconnect it following one of the procedures described below.

(1) Choose [Connect] from the [Debug] menu.


(2) Click the [Connect] toolbar button [].

(3) Enter the [connect] command in the [Command Line] window.

4.6 Disconnecting the Emulator

4.6.1 Disconnecting the Emulator

To disconnect the emulator while it is active, follow one of the procedures described below.

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

4.7 Quitting the High-performance Embedded Workshop

Choosing [Exit] from the [File] menu lets you close the High-performance Embedded Workshop itself.

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

4.8 Setting Up the Debug

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

4.8.1 Specifying a Download Module

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

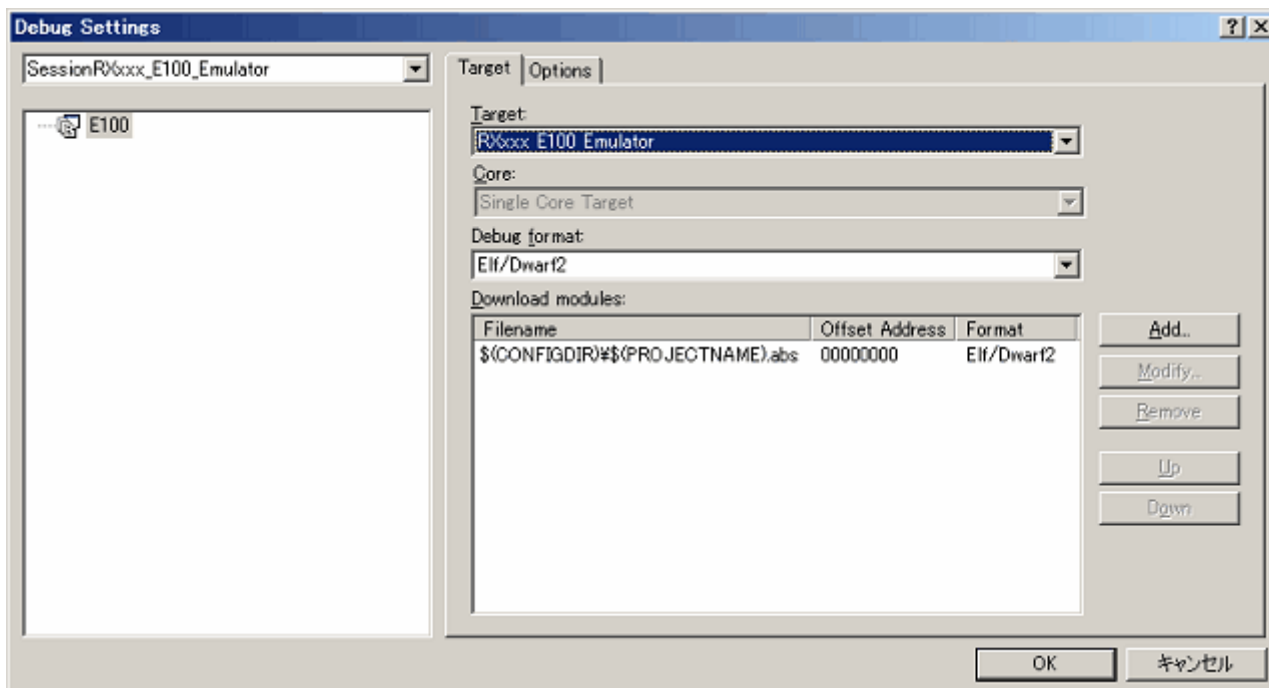


Figure 4.12 [Debug Settings] Dialog Box

In the [Target] drop-down list box, select the product name 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 the load module corresponding to the selected format in the [Download modules] list box.

CAUTION

In the [Download module] dialog box displayed when the [Add] or [Modify] button is clicked, set the access size to "1" (default). Access size set at startup will take precedence for access to external space.

CAUTION

At this point in time, no programs are downloaded yet.

For details on how to download, refer to "5.2.1 Downloading a Program".

4.8.2 Setting Up Automatic Execution of Command Line Batch Files

Click the [Options] tab of the dialog box.

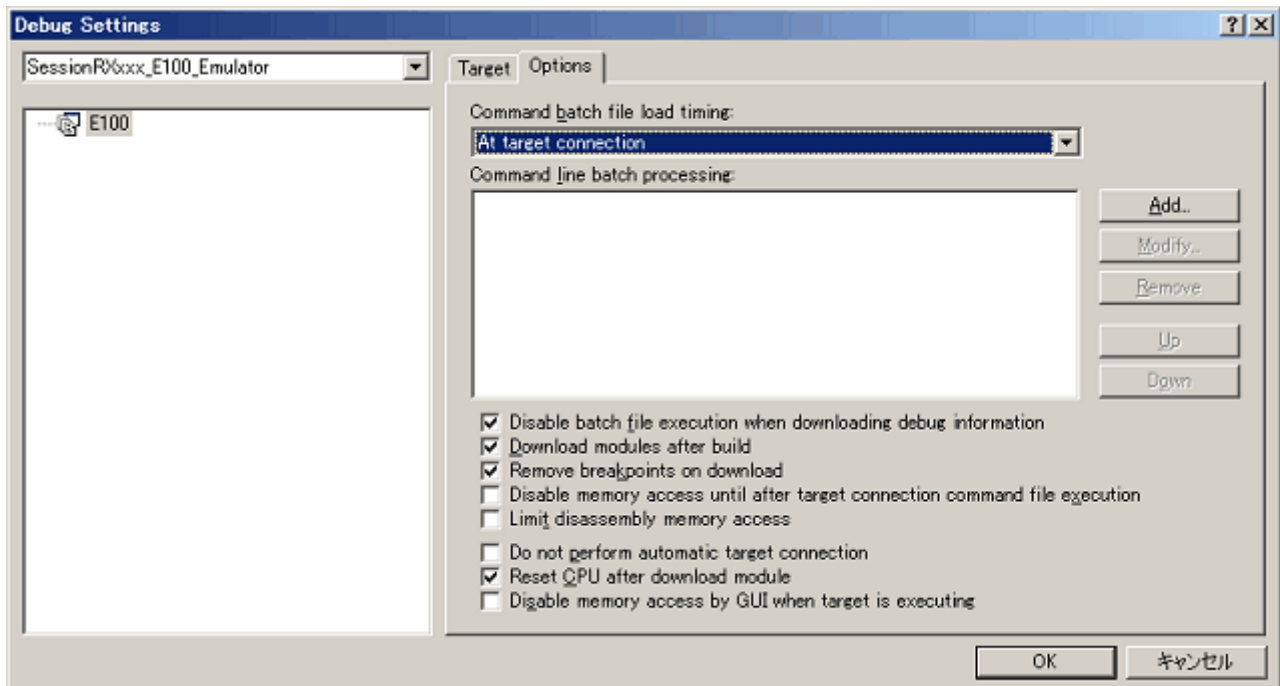


Figure 4.13 [Debug Settings] Dialog Box

Here, register a command chain that is automatically executed with specified timing.

Select your desired timing from the following four choices:

- When the emulator is connected
- Immediately before download
- Immediately after download
- Immediately 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.

5. Debugging Functions

The E100 emulator supports the functions listed in the table below.

Table 5.1 List of Debug Functions

Item No.	Item			Specification
1	Software break			4,096 points
2	On-chip break			8 points
3	Event	Number of event points		Maximum number of effective points: 16
		Content of event	Executed address detection	
			Data access detection	
			Interrupt generation/exit detection	
			External trigger detection	
Task ID	Can be set separately for each event			
		Number of times an event occurred		Maximum 255 times
4	Exception detection			Violation of access protection (operand access and DMAC/DTC)
				Read from uninitialized memory (operand access only)
				Performance overflow
				Realtime profile overflow
				Trace memory overflow
				Task stack access violation (operand access only)
				OS dispatch (operand access only)
5	Hardware break	Hardware breakpoints	Event combination	OR, AND (Accumulation), AND (Simultaneous), subroutine, sequential and state transition
			Exception detection	See item No. 4.
		Delay		Maximum 65,535 bus cycles

6	Trace	Trace size		Maximum 4-M cycles
		Trace mode	Fill until stop	Continues collecting until program stops running
			Fill until full	Stops collecting traces when trace memory is full
			Fill around TP	Stops collecting traces after retarded for delay cycles from when trace point is reached
			Repeat fill until stop	Collects a total of 512 cycles before and after trace point
			Repeat fill until full	Collects a total of 512 cycles before and after trace point
		Trace point	Event combination	OR, AND (Accumulation), AND (Simultaneous), subroutine, sequential and state transition
			Exception detection	See item No. 4
		Delay		Maximum 4M bus cycles
Trace Capture/Do not Capture		Capture/Do not Capture by event - Between two events - Duration of an event - Duration of an event occurring in a subroutine		
7	Performance	Content of measurement		Measures maximum, minimum and average execution time in up to 8 sections and pass counts
				Timeout detection
		Resolution		10 ns to 1.6 μ s
	Measurement mode	Event combination	Between two events, event period and Interrupt-disabled range between two events	
8	RAM monitor		512 bytes \times 32 blocks (operand access and DMAC/DTC) - Shows last read/write accesses performed - Includes uninitialized-area detection function (operand access only)	
9	Profile		128 Kbytes \times 8 blocks (1-Mbyte space)	
			Cumulative time and pass count overflow detection	
10	Coverage		C0 level code coverage	
			256 Kbytes \times 8 blocks (2-Mbyte space)	
			C0+C1 level code coverage	
			128 Kbytes \times 8 blocks (1-Mbyte space)	
			Address range and source file specification	
			Data coverage	
			64 Kbytes \times 8 blocks (512-Kbyte space)	
			Address range, section specification and task stack	

5.1 Setting Up the Emulation Environment

When the emulator is connected, the [Device setting] and [Configuration properties] dialog boxes are displayed. Here, select the general options associated with the emulator. Note that the target MCU to be debugged, etc. can be set only once at startup.

5.1.1 Setting Up the Emulator at Startup

When the emulator starts, the following three dialog boxes are displayed.

(1) [Device setting] dialog box

Use this dialog box to select the target MCU and establish communication.

This dialog box can be redisplayed by selecting [Emulator -> Device setting] from the [Setup] menu after starting the emulator. In this case, however, be aware that changes of the settings after starting the emulator are not reflected immediately and will be set as the initial value when reconnecting the emulator.

(2) [Configuration properties] dialog box

This dialog box is displayed after the [Device setting] dialog box. Use this dialog box to make settings related to the emulator and debug functions.

This dialog box can be re-opened by selecting [Emulator -> System] from the [Setup] menu after the emulator has been booted up. Some options in this dialog box can have their settings changed after startup. The changeable options are displayed as in active use, while the unchangeable options are inactive (grayed out), with their set contents only displayed.

(3) [Connecting] dialog box

This dialog box shows the progress of boot-up processing.

5.1.2 Setting Up the Target MCU

(1) Selecting the target MCU

On the [Device] page of the [Device setting] dialog box, specify the target MCU to be emulated.

For the RX610 Group, more than one product can be emulated by selecting the common type of device "R5F5610x".

If your device is not supported at the time the emulator software is released, select the customizable type of device "R5F5610x_CUSTOMIZE".

For details, refer to the hardware manual supplied with the product.

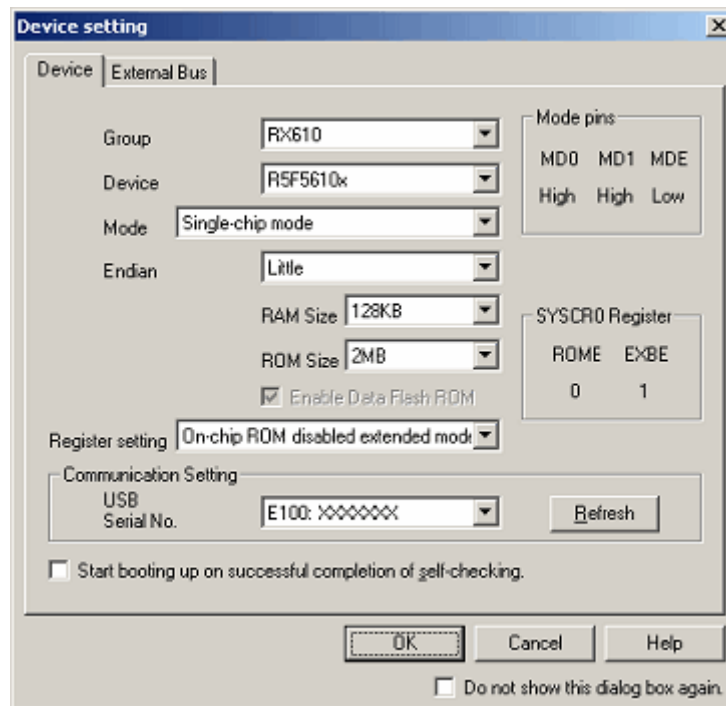


Figure 5.1 [Device setting] Dialog Box ([Device] Page)

The target MCU you have set here cannot be changed after the emulator is connected. To change the target MCU, you need to disconnect the emulator and connect it again.

(2) Selecting the operation mode

Select the operation mode on startup.

Depending on which operation mode is selected, the necessary mode pin setting levels are displayed to the right of the dialog box. If the pin settings of a connected target board do not match these levels, an error is assumed and the device cannot be started.

Select one from the following operation modes:

Single-chip mode (default) or user boot mode

CAUTION

The operation mode differs with each device selected.

(3) Selecting the endian

Select the endian in which mode data are arranged in the CPU.

There are two endian modes to choose from:

Big or little (default)

(4) Selecting the RAM size

Set the RAM size according to the target MCU to be emulated.

When any customizable type of device is selected, you can select one from the following RAM sizes:

64 KB or 128 KB (default)

(5) Selecting the ROM size

Set the ROM size according to the target MCU to be emulated.

When any customizable type of device is selected, you can select one from the following ROM sizes:

128 KB, 256 KB, 512 KB, 768 KB, 1 MB, 1.5 MB, or 2 MB (default)

(6) Enabling Data Flash ROM

This setting is possible when you've selected any customizable type of device.

Choose to enable or disable data flash ROM.

(7) Selecting the register setting

Specify the operation mode to be set in the target program after startup.

Depending on which mode is selected, the SYSCR0 register values that need to be set in the target program are displayed to the right of the dialog box.

Select one from the following operation modes:

Single-chip mode, on-chip ROM enabled extended mode, or on-chip ROM disabled extended mode

(8) Setting up communication

You can select another target emulator connected via USB.

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

(9) Performing self-checking

If you click on the [OK] button with the [Start booting up on successful completion of self-checking.] checkbox selected, hardware self-checking proceeds after connection to the emulator has been made according to the communication condition you have set.

The results are shown on completion of self-checking.

If the results are normal, boot-up processing continues. If an error is found, boot-up processing stops.

5.1.3 Making the External Bus Related Setting

In the [External Bus] page of the [Device setting] dialog box, set the endian and access size of each external bus area.

This page is displayable when you've set such a mode in the [Device] page of the [Device setting] dialog box that the external bus is enabled. Eight areas from CS0 to CS7 can be set.

For details about the external bus, see your hardware manual.

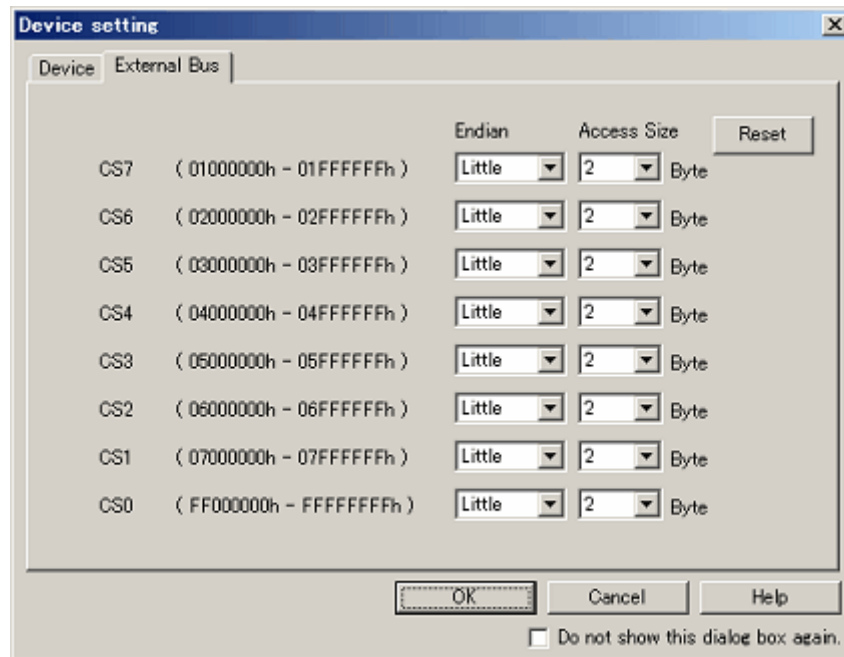


Figure 5.2 [Device setting] Dialog Box—[External Bus] Page

(1) Setting the endian and access size of each area, CS0 to CS7

Set the endians and access sizes of the respective areas individually.

If, in the [Device] page of the [Device setting] dialog box, you've selected the register setting "On-chip ROM enabled extended mode," CS0 is mapped to the internal ROM space and you cannot make settings for this area. Its endian depends on how the device is set up.

(2) [Reset] button

This button restores the endians and access sizes you've individually set to their initial values.

All areas assume the endians that you've set in the [Device] page of the [Device setting] dialog box and the access sizes default to "2".

5.1.4 Setting Up the System

On the [System] page of the [Configuration properties] dialog box, set up the entire emulator system.

This dialog box is displayed following the [Device setting] dialog box at startup.

Although this dialog box can be redisplayed after startup, you cannot change some settings in it. These settings can only be changed at startup.

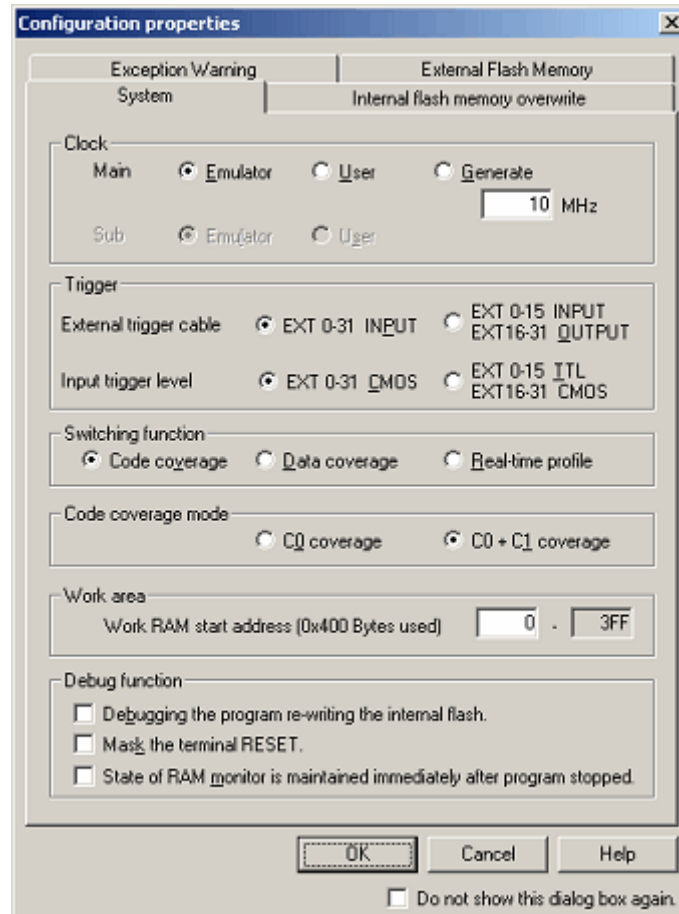


Figure 5.3 [Configuration properties] Dialog Box ([System] Page)

(1) Selecting the operating clock

In the [Clock] section on the [System] page, select the clock sources supplied to the main clock and sub-clock. The main clock can be selected from three choices: Emulator, User and Generate. (By default, 'Emulator' is selected.)

Select 'Emulator' when the main clock is supplied from an internal source or 'User' when the main clock is supplied from an external source. To use a user-defined clock, select 'Generate' and set the clock frequency to be used in the frequency input text box.

The clock frequency can be set in the range 1.0 to 99.9 MHz in 0.1 MHz increments. The clock frequency for 'Generate' can be set only once at startup.

Selection of the sub-clock is displayed only when sub-clocks are supported. It can be selected from 'Emulator' or 'User'. (By default, 'Emulator' is selected.)

CAUTION

The frequency accuracy for 'Generate' is $\pm 5\%$. Please make sure that final evaluation is performed using a resonator or oscillator module of the frequency used for the actual target board that is mounted on-board.

(2) Selecting the direction of external trigger cable

For [External trigger cable], select whether EXT pins 16–31 are directed for input or output. EXT pins 0–15 are fixed for input.

Select this option from the following:

- EXT 0–31 INPUT (default)
- EXT 0–15 INPUT, EXT 16–31 directed for OUTPUT

The setting of this option is reflected at only startup. If you set this option again in the present dialog box after startup, what you have set has no effect.

(3) Selecting a trigger input level

For Trigger input level, select CMOS level or TTL level. Select this option from the following:

- EXT 0–31 chosen to be CMOS (default)
- EXT 0–15 chosen to be TTL, EXT 16–31 chosen to be CMOS

(4) Selecting a switching function

The code coverage, data coverage and realtime profile functions cannot be used at the same time. Select one function from them.

Initially, code coverage is selected.

The setting of this option can be changed even after startup.

When the code coverage function is selected, measurements are performed at the coverage level selected in [Code coverage mode].

(5) Selecting a code coverage mode

Select a code coverage mode.

C0: Instruction coverage rate

C0 + C1: Instruction coverage rate + Branch coverage rate

You can measure up to 2 Mbytes when using the C0 level coverage, and up to 1 Mbyte when using the C0 + C1 level coverage

The initial value is C0 coverage.

The setting of this option is reflected at only startup.

This option is available only when 'Code coverage' is selected in [Switching function].

If you use the code coverage function, select the mode in this option at startup.

(6) Setting up the work area

Set up the work area that is temporarily used by the debugger while the program is stopped.

Be sure the work area is set up in an internal RAM space that is not accessed by the DMAC and DTC. In other than that, even in an area used as data, however, values may not be corrupted.

The memory size used as a work area differs with the device selected.

(7) Debugging the program re-writing the internal flash memory

Choose whether to debug the program while rewriting the internal flash ROM.

This option is selectable only on startup.

(8) Masking the terminal RESET

Select whether you want the input signal to the RESET pin of the target system to be masked.

(9) Maintaining the state of RAM monitor immediately after the program stopped

When the program stops while this option is selected, the emulator continues displaying the state of the RAM monitor in which it was immediately after the program stopped. Changes made to the data in the memory window, etc. while the program is stopped are not reflected in the [RamMonitor] window.

5.1.5 Setting Up Flash ROM Overwrite

On the [Internal flash memory overwrite] page of the [Configuration properties] dialog box, set up the overwriting of flash ROM blocks, block by block.

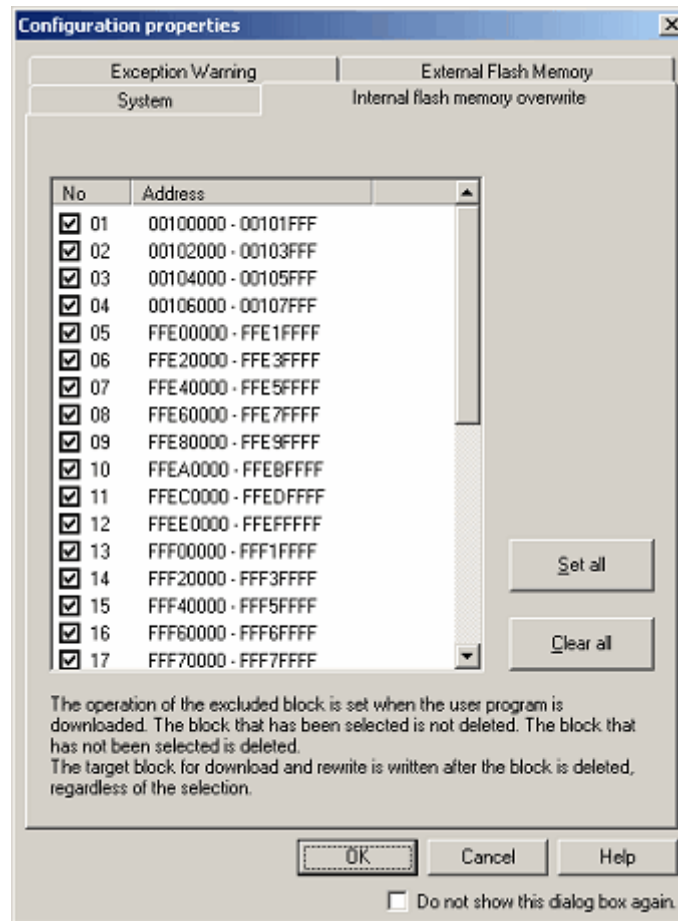


Figure 5.4 [Configuration properties] Dialog Box ([Internal flash memory overwrite] Page)

Block-by-block settings matched to the selected target MCU are automatically displayed in the list.

The blocks selected by placing a check mark in the respective checkboxes are overwritten (merged), without being erased, when a program is downloaded.

However, the target blocks to which downloaded and rewritten are erased before being written to, regardless of whether they are selected.

5.1.6 Setting the Warning of Exceptional Events

On the [Exception Warning] page of the [Configuration properties] dialog box, set whether or not to display warnings of exceptional events in the [Status] window and status bar balloon.

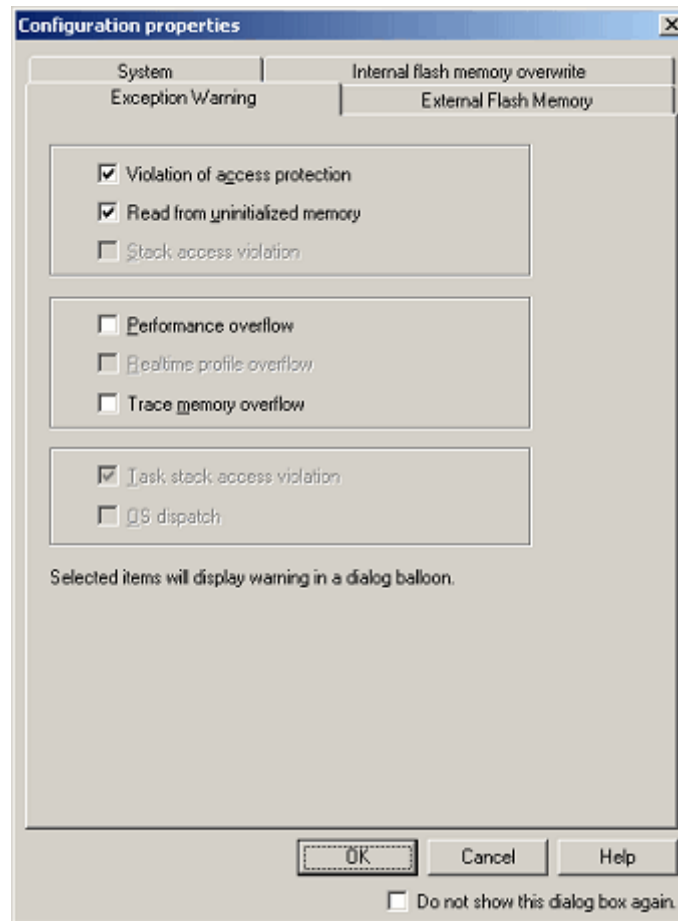


Figure 5.5 [Configuration properties] Dialog Box ([Exception Warning] Page)

The [Violation of access protection] and [Read from uninitialized memory] checkboxes are initially selected.

When a load module including OS has been downloaded, the [Task stack access violation] checkbox is also initially selected.

Other items are not selected.

If you deselect a checkbox, this item will be shown as '-' in the [Status] window.

CAUTION

The unusable functions are deactivated (grayed out) and cannot be selected.

5.1.7 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.

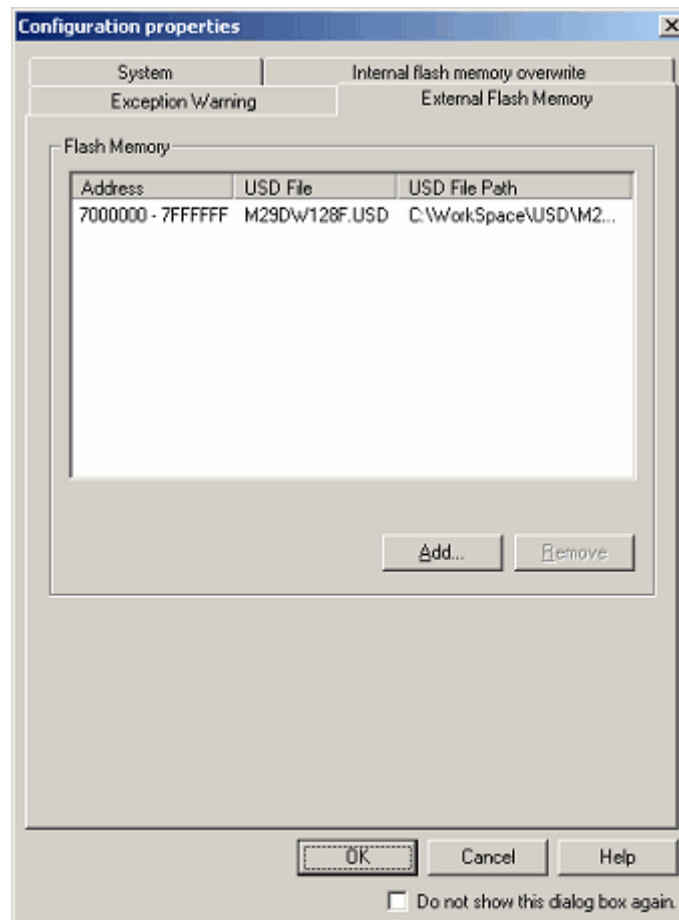


Figure 5.6 [Configuration properties] Dialog Box ([External Flash Memory] Page)

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.

(1) Registering Information on External Flash Memory

The [Add] button on the [External Flash Memory] page is only usable when on-chip ROM enabled extended mode or on-chip ROM disabled extended mode has been selected against [Mode] on the [MCU] page. 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] section of the [External Flash Memory] page.

(2) Deleting Information on the External Flash Memory

Select the USD file that you wish to delete and click on the [Remove] button.

(3) Setting for Overwriting Blocks of the External Flash Memory

Double-clicking on a USD file in the [Flash memory] section opens the [External Flash memory overwrite] dialog box, which allows you to specify whether or not individual sectors (blocks) of external flash memory should be overwritten.

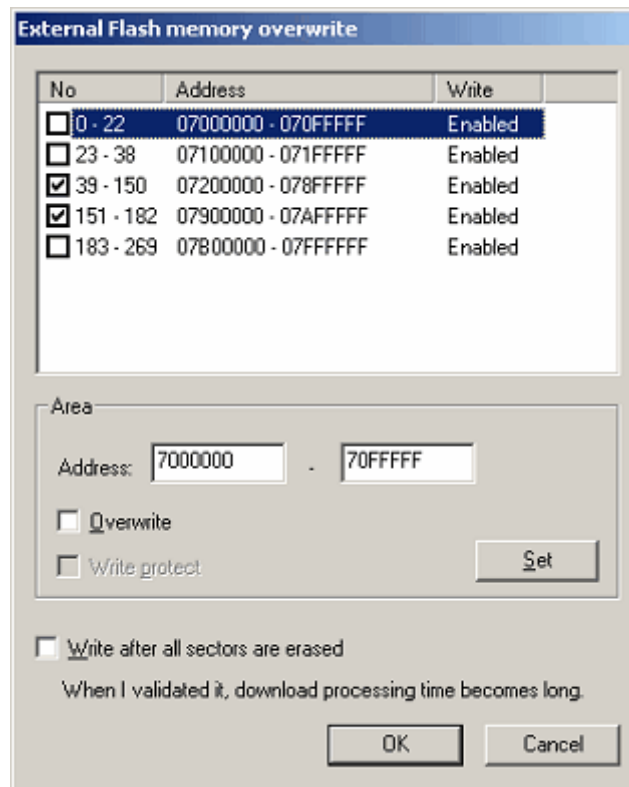


Figure 5.7 [External Flash memory overwrite] Dialog Box

Settings for all sectors defined in the selected USD file 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 [Write protect] checkbox is not selectable. If [Write after all sectors are erased] is selected, all sectors will be erased when the user program is downloaded.

5.1.8 Showing Progress in Boot-up Processing

You can confirm the progress of boot-up processing by checking the [Connecting] dialog box.

The [Connecting] dialog box continues displaying progress information from when boot-up processing starts till when it ends.

While the [Device setting] and [Configuration properties] dialog boxes are displayed, you cannot manipulate this dialog box.

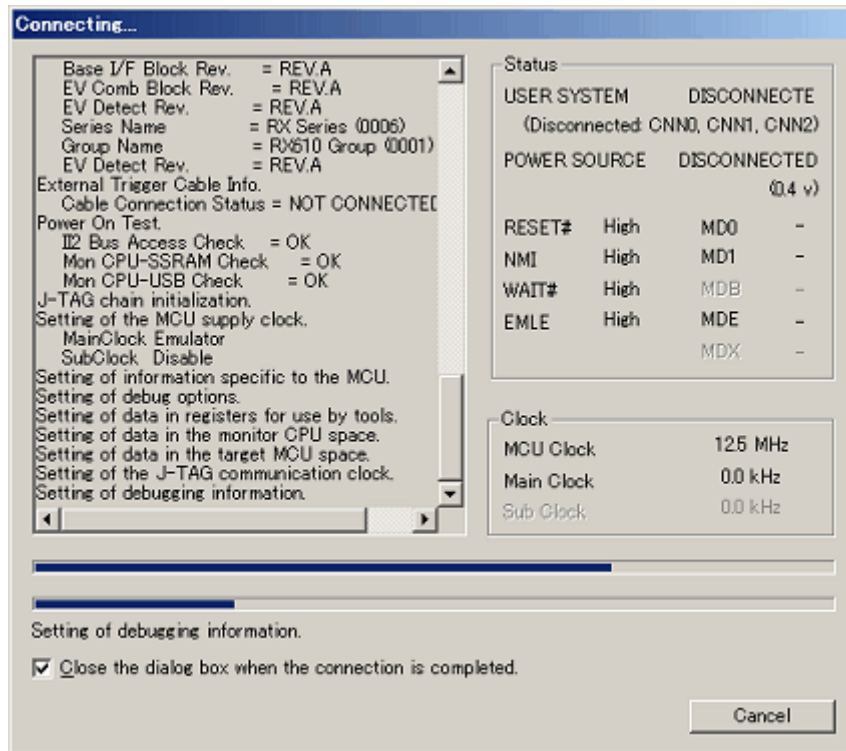


Figure 5.8 [Connecting] Dialog Box

(1) Showing the history of processing

The history display area on the left-hand side of the dialog box shows the history of completed processing. The contents shown here are recorded in a trouble report. To check the content of a trouble report, select [Technical Support -> Create Bug Report] from the [Help] menu.

(2) Showing the pin states

The pin states are displayed after the completion of the emulator's startup process.

The pin states of the actually connected target board are read out and displayed. If the set contents of the [Device setting] dialog box and these pin states do not match, a warning message is displayed in the history display area and a connection error is assumed.

If no target board is connected, indeterminate values are read out and a warning message is displayed in the history display area, in which case, however, the connection is completed.

(3) Showing the clocks

The clocks are displayed after the completion of the emulator's startup process.

Only the actually operating clocks are displayed here.

(4) Showing progress with progress bars

The upper progress bar shows the progress of the entire boot-up processing.

The lower progress bar shows the progress of each individual processing.

The content of the currently executed processing is displayed below the bar.

(5) Aborting a connection

Clicking the [Cancel] button aborts boot-up processing.

5.2 Downloading a Program

5.2.1 Downloading a Program

Download the load module to be debugged.

To download a program, choose [Download] from the [Debug] menu and select your desired load module from the ensuing list, or right-click a load module in [Download modules] of the [Workspace] window and then choose [Download] from the pop-up menu.

CAUTION

Before a program can be downloaded, you must have it registered as a load module in the High-performance Embedded Workshop. For details on how to register, refer to 4.8 "Setting Up the Debug".

5.2.2 Showing the Source Code

Follow the procedure described below to show the source code.

- Double-click a source file in the [Workspace] window.
- Right-click in the source file and choose [Open] from the pop-up menu.

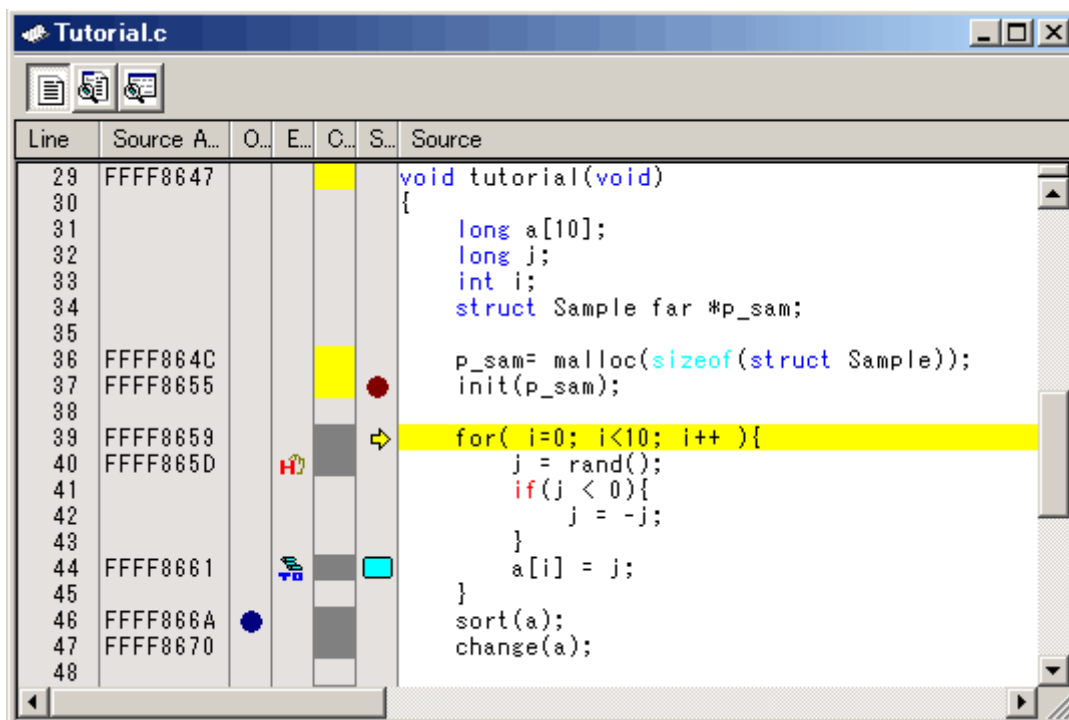


Figure 5.9 [Editor] Window

Shown at left edge of this window are the line information consisting of the following:

(1) [Line] column

Shows the line numbers corresponding to lines in the source file.


(2) [Source Address] column

When a program is downloaded, this column shows the addresses corresponding to lines in the current source file. This function will prove convenient when you determine where you want the PC value or breakpoint to be set.

(3) [On-chip Breakpoints] column

Following is displayed in the [On-chip Breakpoints] column.

Table 5.2 [On-Chip Breakpoints] Column Table



	On-chip breakpoint is set.
---	----------------------------

On-chip breakpoints can be inserted by double-clicking on the [On-chip Breakpoints] column.

(4) [Event] column

This column shows the following:

Table 5.3 [Event] Column List

	Hardware breakpoint is set.
	Trace point (fetch condition) is set.

A hardware breakpoint can be inserted by double-clicking in the [Event] column.

Trace points are displayed when fetch conditions are set.

[*] after the title on the title bar of the dialog boxes of [Hardware Break], [Trace conditions] or [Performance Analysis Conditions] shows that some setting is under editing. If you are doing some editing work, you cannot change the settings from the [Event] column of the [Editor] window.

(5) [Code Coverage] column




Shows C0 code coverage information graphically.

This column is displayed only when the code coverage function is enabled.

(6) [S/W Breakpoints] column

This column shows the following:

Table 5.4 [S/W Breakpoints] Column List

	Bookmark is set.
	Software break is set.
	PC position

Software breakpoints can be inserted by double-clicking on the [S/W Breakpoints] column.

5.2.3 Turning Columns in All Source Files Off

(1) From the [Editor] window

1. Right-click in the [Editor] window and choose [Define Column Format] from the pop-up menu.
2. The [Global Editor Column States] dialog box will be displayed.

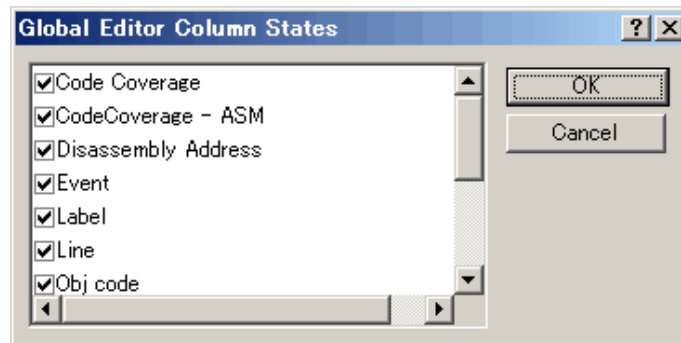


Figure 5.10 [Global Editor Column States] Dialog Box

3. Deselect the checkbox of the column you want to turn off. Click the [OK] button, and the new column settings you have set will take effect.

5.2.4 Turning Columns in One Source File Off

(1) From the [Editor] window

1. Right-click in the [Editor] window and choose [Columns] from the pop-up menu.
2. Cascaded menu items will be displayed. The currently enabled columns have a check mark attached to the left of the respective names.

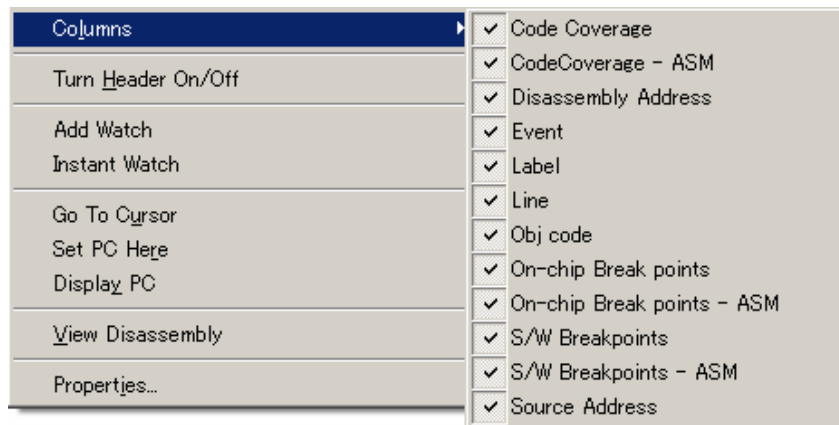


Figure 5.11 [Pop-Up Menu] Window

3. Clicking a column name lets you enable or disable the column alternately.


5.2.5 Showing Assembly Language Code

While a source file is open, click the right mouse button in the [Editor] window and choose [View Disassembly] from the pop-up menu. The [Disassembly] window will be displayed.

The display start address in the [Disassembly] window is the one that corresponds to the cursor position in the [Editor] window.

You also can use the [Disassembly View] button in the Editor window to display disassembled codes.

If no source files exist, one of the following methods may be used to display disassembled codes.

- Click the [Disassembly] toolbar button [].
- Choose [Disassembly] from the [View] menu.
- Use the “Ctrl + D” accelerator.

In this case, the [Disassembly] window opens at the current PC position.

Mixed mode display where all source lines are displayed beginning with that address is also supported as an option. To display disassembled codes in mixed mode, click the [Show in Mixed Mode] button.

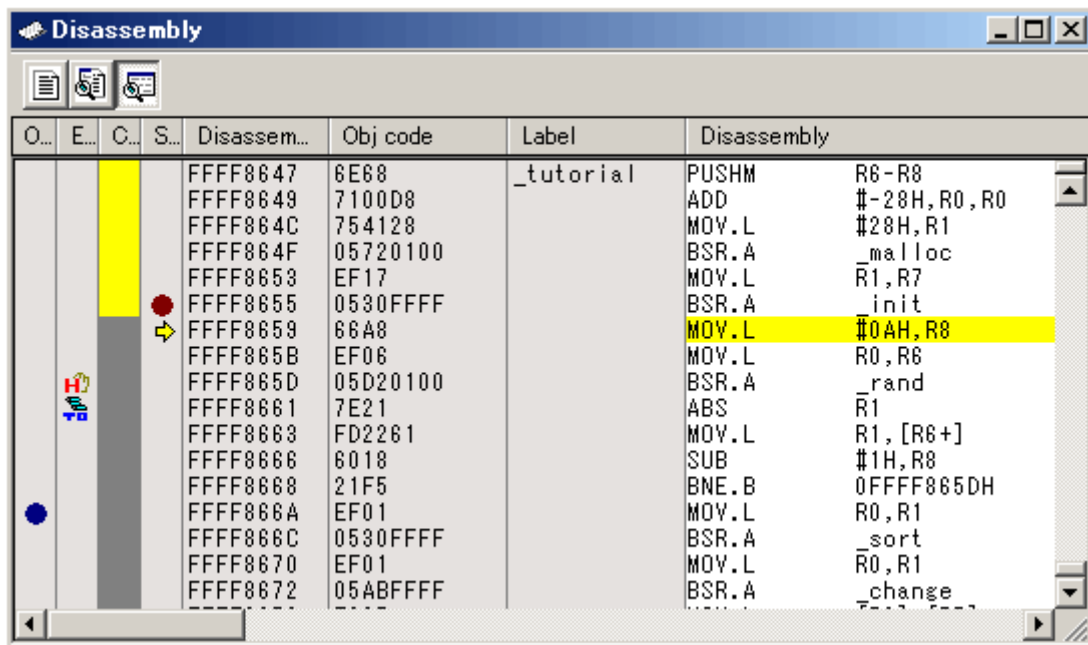



Figure 5.12 [Disassembly] Window

Shown at left edge of this window are the line information consisting of the following:

- (1) [On-chip Breakpoints] column

Following is displayed in the [On-chip Breakpoints] column.

Table 5.5 [On-chip Breakpoints] Column Table



	On-chip breakpoint is set.
---	----------------------------

On-chip breakpoints can be inserted by double-clicking on the [On-chip Breakpoints] column.

(2) [Event] column

This column shows the following:

Table 5.6 [Event] Column List

	Hardware breakpoint is set.
	Trace point (fetch condition) is set.

A hardware breakpoint can be inserted by double-clicking in the [Event] column.

Trace points are displayed when fetch conditions are set.

[*] after the title on the title bar of the dialog boxes of [Hardware Break], [Trace conditions] or [Performance Analysis Conditions] shows that some setting is under editing. If you are doing some editing work, you cannot change the settings from the [Event] column of the [Editor] window.

(3) [Code Coverage] column



Shows C0 code coverage information graphically.

This column is displayed only when the code coverage function is enabled.

(4) [S/W Breakpoints – ASM] column

This column shows the following:

Table 5.7 [Software Breakpoint – ASM] Column List

	Software break is set.
	PC position

Software breakpoints can be inserted by double-clicking on the [S/W Breakpoints] column.

(5) [Disassembly Address] column

Shows disassembly addresses. Double-clicking here brings up an [Address Specification] dialog box. In this dialog box, enter the address from which you want a disassembly display to start.

(6) [Obj code] column

Shows object codes.

(7) [Label]

Shows a label. This column is unusable unless any module is downloaded.

5.2.6 Correcting Assembly Language Codes

Double-click an instruction you want to correct in the [Disassembly] window or choose [Edit] from the pop-up menu, and the [Assembler] dialog box will be displayed. Use this dialog box to correct assembly language.

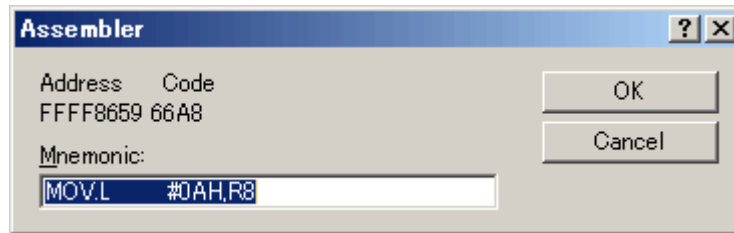


Figure 5.13 [Assembler] Dialog Box

The dialog box shows the address, instruction code and mnemonic of a selected instruction.

Enter a new instruction (or edit the old instruction) in the [Mnemonic] edit box. When done, hit the [Enter] key. The memory content will be overwritten with the new instruction code, and the pointer is moved to the next instruction.

Clicking the [OK] button overwrites the memory content with the new instruction code and closes the dialog box.

CAUTION

Assembly language codes are displayed from the current memory content. When you correct memory contents, new assembly language codes are displayed in the [Disassembly] window and the [Assembler] dialog box. However, the source file being displayed in the [Editor] window remains unchanged. The same applies when the source file includes assembler language.

5.3 Displaying Memory Contents in Real Time

5.3.1 Displaying Memory Contents in Real Time

To monitor memory contents while the user program is running, use the [RamMonitor] window.

The RAM monitor function permits the memory content and access status in an allocated monitor area to be recorded and inspected in real time without obstructing execution of the user program.

The [RamMonitor] window shows access statuses (read, write, uninitialized or uninspected) in different colors.

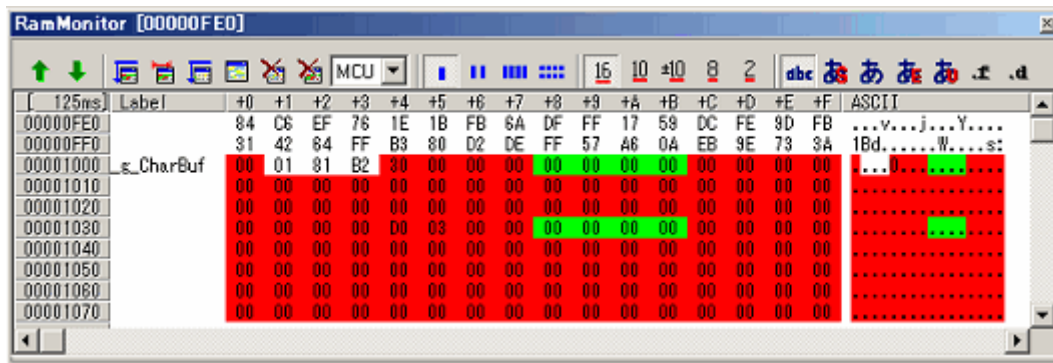


Figure 5.14 [RamMonitor] Window

(1) Allocating a RAM monitor area

A 16-Kbyte RAM monitor area is provided.

This RAM monitor area can be allocated to selected contiguous addresses or divided 32 blocks in 512-byte units.

With initial settings, a maximum 16 Kbytes of area from the beginning address of the internal RAM is allocated as a RAM monitor area.

(2) Monitor display

The access statuses are displayed in different background colors depending on access attributes, as listed below. (The background colors are customizable.)

The read and write accesses show the last accesses made.

Error detection can be displayed by choosing [Show Error Detection] from the pop-up menu. In this case, the read and write accesses are not displayed.

Table 5.8 Access Attribute and Background Color

		Access attribute	Background color
1		Read access	Green
2		Write access	Red
3	When errors detected	Uninitialized memory (an area not written to has been read)	Yellow
4		Uninspected memory (an initialized area has not been read)	Sky blue
5		No access	White

CAUTION

The contents shown in the [RamMonitor] window are the data acquired from bus accesses. Therefore, changes made by accessing memory via other than the user program, as when memory is rewritten directly from external I/O, are not reflected in the displayed memory content.

(3) Detecting reading from uninitialized memory

If a memory area that has not been written to is read, the emulator detects “reading from uninitialized memory” and outputs an error.

To view errors of this type, choose [Error Detection Display] from the pop-up menu.

Uninitialized memory areas are shown in yellow.

This error detection can be an exceptional event used as a condition of a hardware breakpoint or trace point (also refer to “Detecting exceptional events”).

(4) Detecting uninspected areas

If an initialized memory area has not been read, the emulator detects “an uninspected area” and outputs an error.

To view errors of this type, choose [Error Detection Display] from the pop-up menu.

Uninspected memory areas are shown in sky blue.

5.3.2 Switching the Displayed Bus

In the [RamMonitor] window, it is possible to switch the displayed bus information between the MCU bus and DMA bus.

If multiple instances of the [RamMonitor] window are open, do this switching separately for each window.

Follow the procedure below to switch the display.



- Use the list box in the [RAM Monitor] toolbar to select either one.
- Choose [Display Mode] from the popup menu of the [RamMonitor] window and select your desired mode from the cascaded menu items.

5.3.3 Setting RAM Monitor Update Intervals

Choose [Update Interval Setting] from the pop-up menu of the [RamMonitor] window. The [Update Interval Setting] dialog box shown below will appear.

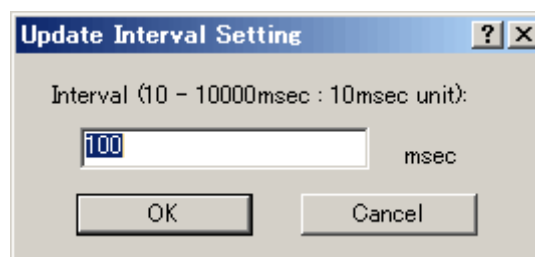


Figure 5.15 [Update Interval Setting] Dialog Box

The update interval can be specified separately for each window.

The initial value is 100 ms.

5.3.4 Clearing RAM Monitor Access History

Choose [Access Data Clear] from the pop-up menu of the [RamMonitor] window. The history of all accesses made to the RAM monitor area will be cleared.

CAUTION

If this function is executed while the user program is running, the user program's realtime capability may be lost because a memory dump occurs.


5.3.5 Clearing RAM Monitor Error Detection Data

Choose [Error Detection Data Clear] from the pop-up menu of the [RamMonitor] window. The detected data of all uninitialized memory and unreferred memory of the RAM monitor area will be cleared.

5.4 Showing the Current Status

5.4.1 Showing the Emulator Status

To know the current status of the emulator, display the [Status] window.

To open the [Status] window, choose [CPU -> Status] from the [View] menu, or click the [View Status] toolbar button .

This window does not update the displayed status during program execution.

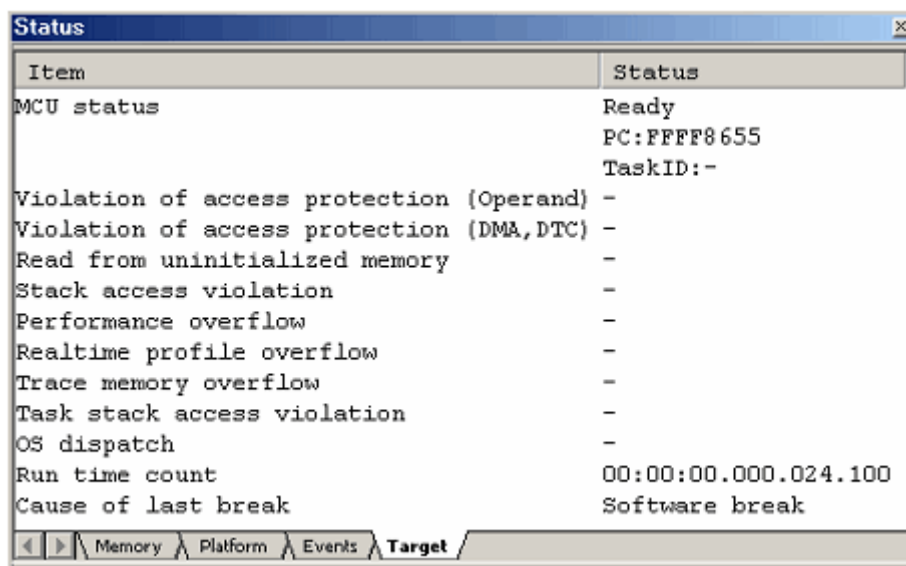


Figure 5.16 [Status] Window

The [Status] window has the following four sheets.

Table 5.9 Sheet List of the [Status] Window

Sheet Name	Description
Memory	Shows information relating to memory resources.
Platform	Shows information relating to the emulator and debugging.
Events	Shows information relating to events.
Target	Shows information relating to the target MCU.

5.4.2 Showing the Emulator State in the Status Bar

The state of the emulator can be displayed in the status bar.

By right clicking on the status bar, the items are shown. Check the items you want to show in the status bar.

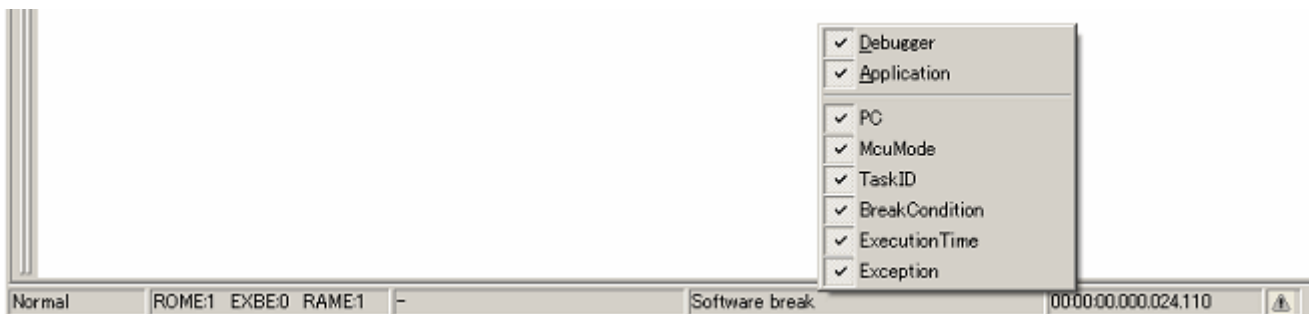


Figure 5.17 Status Bar

Table 5.10 Item List of the Emulator State Shown in the Status Bar

Item	Description
PC	PC value During execution: PC value Idle: Normal
McuMode	SYSCR0 register value (only when idle)
Task ID	Task ID, task entry label
BreakCondition	Break factors of the user program (only when idle)
ExecutionTime	Result of time measurement (only when idle)
Exception	State of the exceptional event

(1) When more than one break factors occur

When you click on the status bar area where the break factor is shown, a balloon is shown.

You can check the break factors being occurred in the balloon.

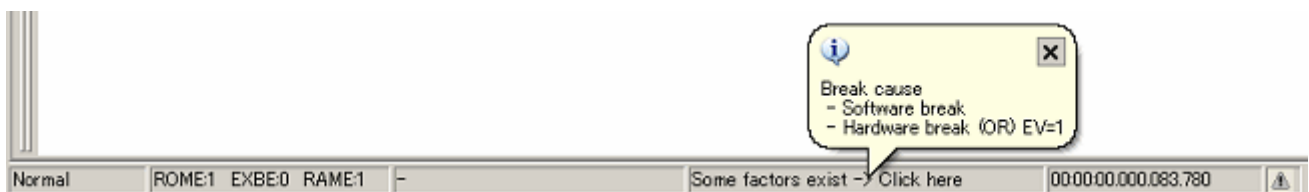


Figure 5.18 Example of Break Factors Display when Break Factors Occur

(2) When an exceptional event occurs

When an exceptional event occurs, a warning is displayed in a status bar balloon.

The exceptional event that is not checked on the [Exception Warning] page of the [Configuration properties] dialog box is not shown.




Figure 5.19 Example of Warning Display when Exceptional Events Occur

5.5 Periodically Reading Out and Showing the Emulator Status

5.5.1 Periodically Reading Out and Showing the Emulator Information

To know the changing emulator information whether the user program is running or remains idle, use the [Extended Monitor] window.

The extended monitor function only monitors the signals output from the user system or MCU, and does not affect execution of the user program.

To open the [Extended Monitor] window, choose [CPU -> Extended Monitor] from the [View] menu, or click the [Extended Monitor] toolbar button .

The displayed items are updated at an interval of about 1,000 ms during user program execution or about 5,000 ms during a break.

CAUTION

“CPU Clock” can be measured only when a user program is being executed.

Item	Value
User System Connection	DISCONNECT (Disconnect: CNNO, CNH1, CNH2)
User System Power Source	DISCONNECT (0.4 v)
User System RESET#	High
User System NMI	High
User System WAIT#	High
User System EMLE	Low
User System MD0	-
User System MD1	-
User System MDE	-
System Control Register	R0ME:1 EXBE:0 RAME:1
System Clock	-
Main Clock	Emulator 12.5 MHz
Sub Clock	-

Figure 5.20 [Extended Monitor] Window

5.5.2 Selecting the Items to Be Displayed

Choose [Properties] from the pop-up menu of the [Extended Monitor] window, and the [Extended Monitor Configuration] dialog box will be displayed.

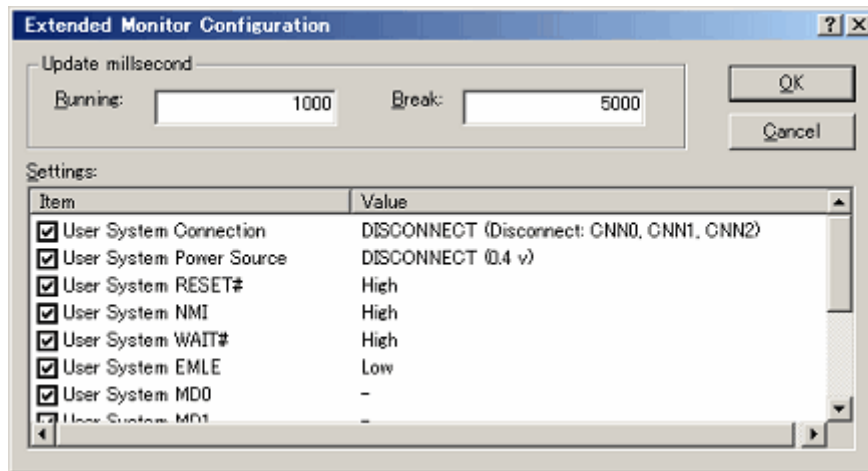


Figure 5.21 [Extended Monitor Configuration] Dialog Box

This dialog box permits you to set each item you want to be displayed in the [Extended Monitor] window.

5.6 Using Software Breakpoints

5.6.1 Using Software Breakpoints

A software break causes the user program to stop running by rewriting the instruction code at a specified address with a BRK instruction to generate a BRK interrupt. In that sense, this is a pre-execution break function.

4096 breakpoints can be set.

If multiple software breakpoints are set, the program breaks at any one of those breakpoints reached.

(1) When stopped at a software breakpoint

When the program you have created is run and the address you have set as a software breakpoint is reached, a message "Software Break" is displayed on the [Debug] sheet of the [Output] window, with the program made to stop there. At this time, the [Editor] or [Disassembly] window is updated, and the position at which the program has stopped is marked with an arrow [→] in the [S/W Breakpoints] column.

CAUTION

When a break occurs, the program stops immediately before executing the line or instruction at which a software breakpoint is set. If Go or Step is selected after the program has stopped at that software breakpoint, the program restarts from the line marked with an arrow.

5.6.2 Adding/Removing Software Breakpoints

Follow one of the following methods to add or remove software breakpoints.

- From the [Editor] or [Disassembly] window
- From the [Breakpoints] dialog box (only removing)
- From the command line

(1) From the [Editor] or [Disassembly] window

1. Check to see that the [Editor] or [Disassembly] window that is currently open includes the position at which you want to set a software breakpoint.
2. In the [S/W Breakpoints] column, double-click the line where you want the program to stop.

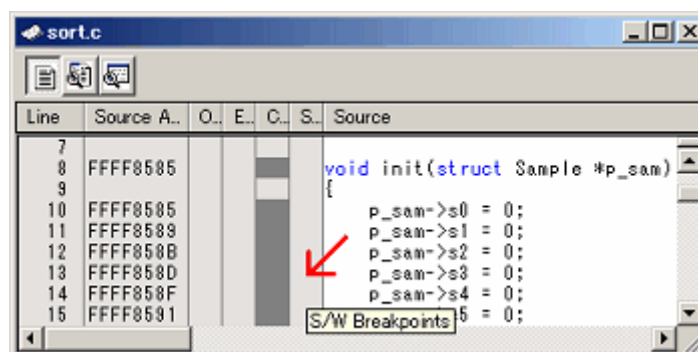


Figure 5.22 [Editor] Window

Or you use the method described below to set a breakpoint. Select [Toggle Breakpoint] from the pop-up menu or press the F9 key on the keyboard.

3. When a software breakpoint is set, a red circle [●] is displayed at the corresponding position in the [S/W Breakpoints] column of the [Editor] or [Disassembly] window.

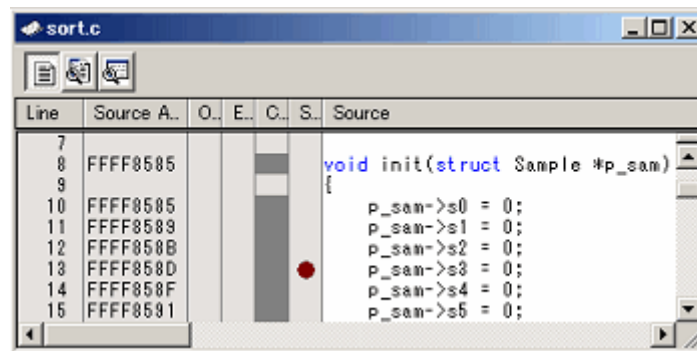


Figure 5.23 [Editor] Window

Double-clicking one more time removes the breakpoint.

5.6.3 Enabling/Disabling Software Breakpoints

Follow one of the following methods to enable or disable software breakpoints.

- From the [Editor] or [Disassembly] window
- From the [Breakpoints] dialog box
- From the command line

(1) From the [Editor] or [Disassembly] window

1. Place the cursor at the line where a software breakpoint exists and then select [Enable/Disable Breakpoint] from the pop-up menu. Or press the Ctrl and F9 keys together.

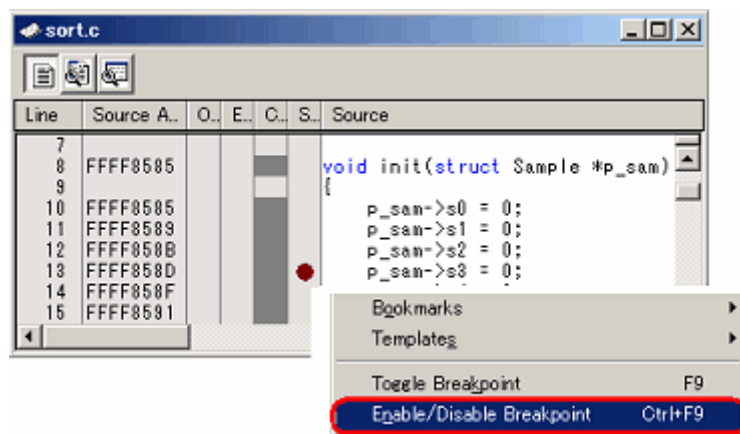


Figure 5.24 [Editor] Window and Pop-Up Menu

- The software breakpoint is enabled or disabled alternately.

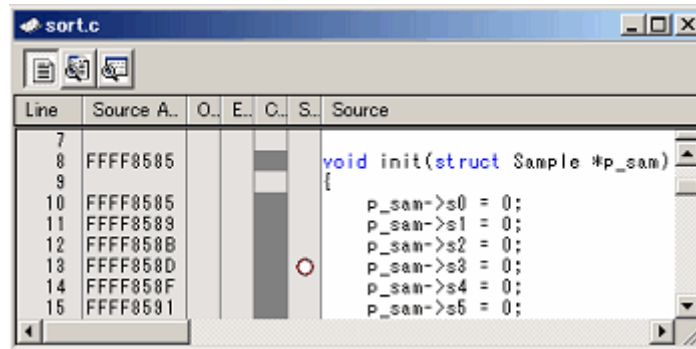


Figure 5.25 [Editor] Window

(2) From the [Breakpoints] dialog box

- Select [Source Breakpoints] from the [Edit] menu to bring up the [Breakpoints] dialog box. In this dialog box, you can alternately enable or disable a currently set breakpoint, as well as remove it.

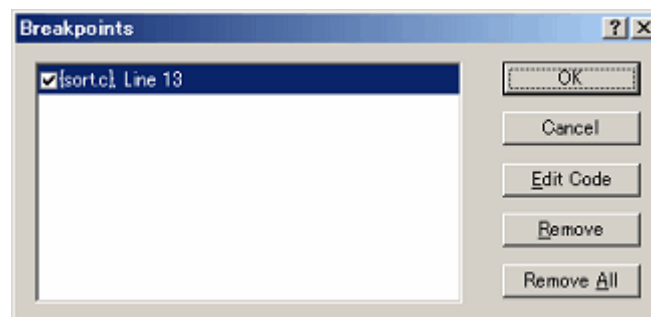


Figure 5.26 [Breakpoints] Dialog Box

5.7 Using Events

5.7.1 Using Events

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

The E100 emulator permits you to use the event you have set as a condition of the break, trace or performance function.

Events can be set at up to 16 points at the same time.

These 16 points can be located at any desired positions.

The events you have created can be registered for reuse at a later time.

(1) Types of events

There are following types of events.

Table 5.11 Event Types List

Instruction fetch	An event is detected when an instruction at the specified address is executed by CPU. An event is detected not in the cycles prefetched by an instruction queue but in the cycles executed by the CPU.
Data access	An event is detected when a specified address or specified address range is accessed under a specified condition.
Interrupt	Detection is made of an interrupt generation and interrupt termination.
Trigger input	An event is detected when the signal fed in from external trigger signal input cable is in a specified state.

(2) Event combination

One of the following combinatorial conditions can be specified using two or more events in combination.

Table 5.12 Event Combinations List

OR	Condition is met when any one of the specified events occurs.
AND (Accumulation)	Condition is met when all of the specified events occur irrespective of the time axis.
AND (Simultaneous)	Condition is met when all of the specified events occur at the same time.
Subroutine	Condition is met when a specified event occurs within a specified address range.
Sequential	Condition is met when a specified event occurs in a specified order.
State transitions	Condition is met when an event occurs under the condition specified in a state transition diagram.

5.7.2 Adding Events

Follow one of the following methods to add events.

- Create a new event
- Add by dragging and dropping from another window
- Add from the command line

(1) Creating a new event

[When creating an event from any setup dialog box]

1. Click the [Add] button or choose a line where you want to input and double-click.

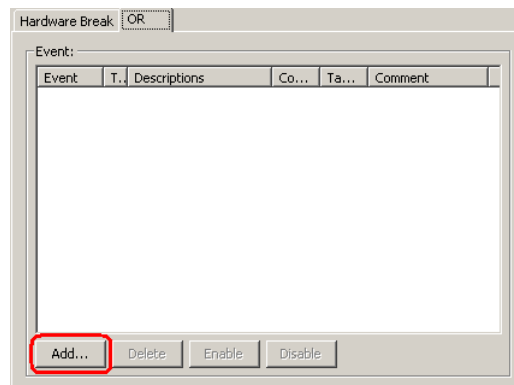


Figure 5.27 [Hardware Break] Dialog Box

2. The [Event] dialog box shown below will be displayed. In this dialog box, set detail event conditions and then click the [OK] button.

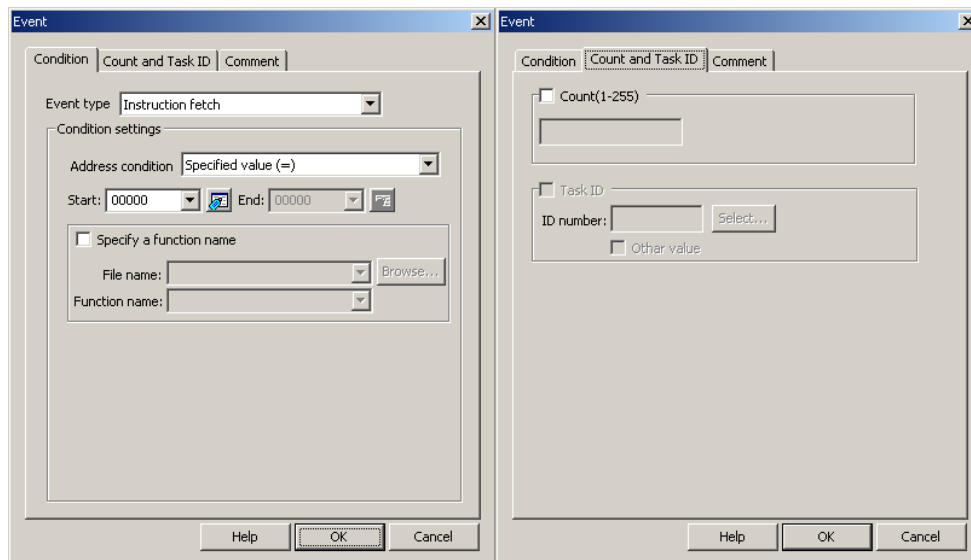


Figure 5.28 [Event] Dialog Box

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

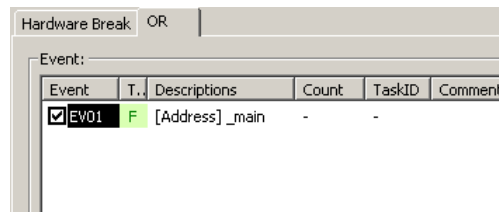


Figure 5.29 [Hardware Break] Dialog Box

4. If events exceed 16 points when you created an event, an error is displayed. If you created an event exceeding 16 points, the event you have added has no effect.

[When adding an event from the [Registered Events] dialog box]

1. Click the [Add] button in the [Registered Events] dialog box.

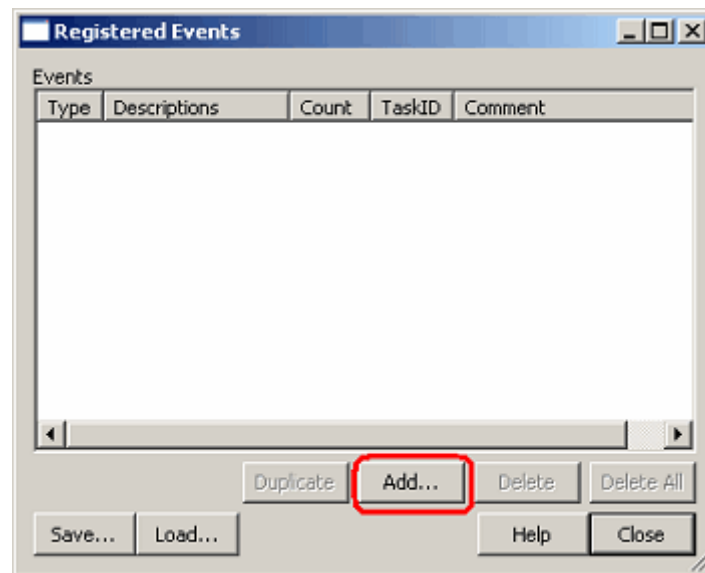


Figure 5.30 [Registered Events] Dialog Box

2. The [Event] dialog box shown below will be displayed. In this dialog box, set detail event conditions. Enter a comment if any necessary. Then click the [OK] button.

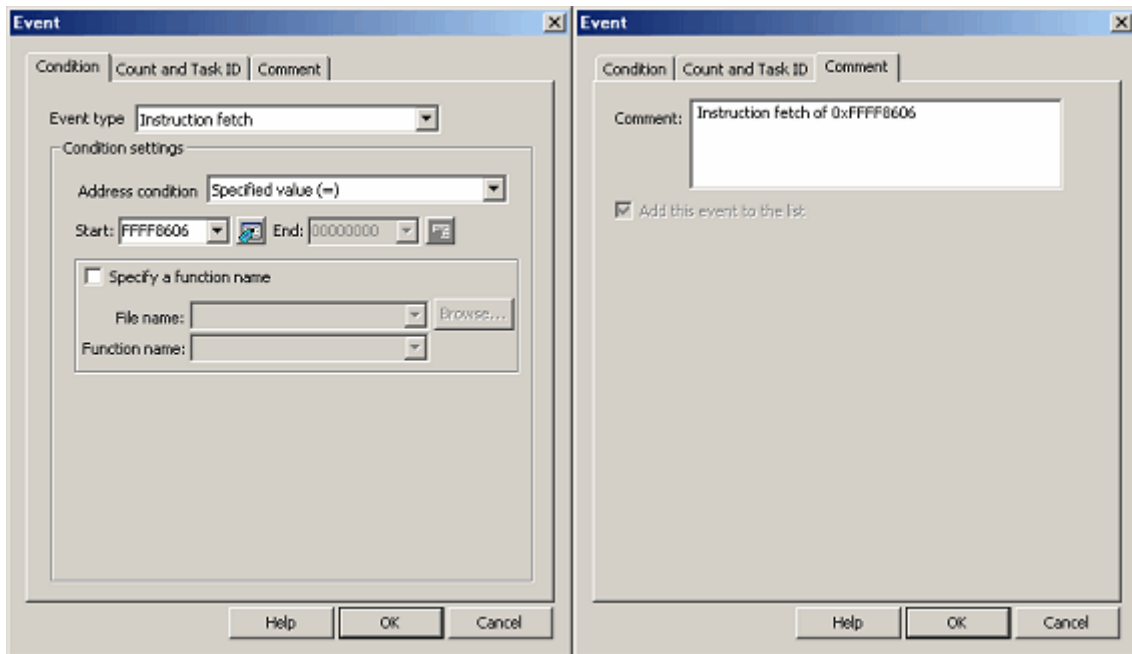


Figure 5.31 [Event] Dialog Box

3. An event will be added to the list of registered events.

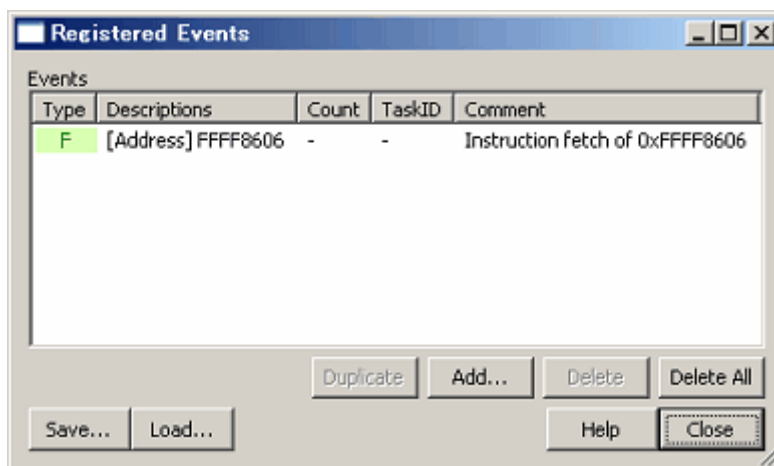


Figure 5.32 [Registered Events] Dialog Box

(2) Adding an event from the [Event] column of the [Editor] window

[When adding a hardware breakpoint]

1. Select [HW Break Point] from the pop-up menu displayed by double-clicking or right clicking anywhere in the [Event] column of the [Editor] window.

You can set a hardware breakpoint based on a fetch to that address as a condition. => Instruction fetch condition

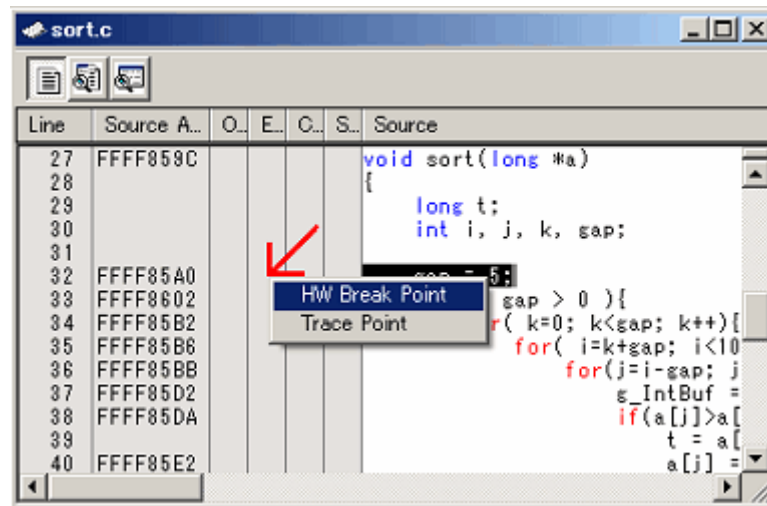


Figure 5.33 [Editor] Window

2. If there is room for event counts, the event you have added from the [Editor] window is added to the other events as an OR condition. If there is no room, an error message is displayed.

CAUTION

If you are doing some editing work in the [Hardware Break] dialog box, you cannot set hardware breaks from the [Event] column of the [Editor] window.

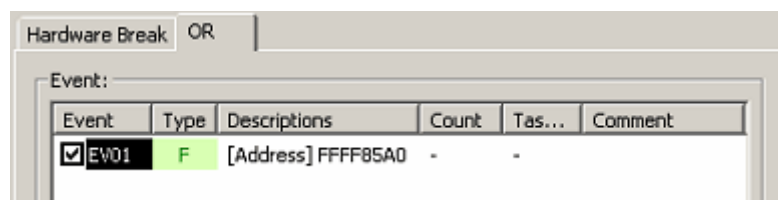


Figure 5.34 [Hardware Break] Dialog Box

[When adding a trace point]

1. Select [Trace Point] from the pop-up menu displayed by double-clicking or right clicking anywhere in the [Event] column of the [Editor] window.

You can set a trace point based on a fetch to that address as a condition. => Instruction fetch condition
Double-click the instruction fetch event in the [Event] column of the [Editor] window to delete it.

CAUTION

No trace points can be set from the [Event] column of the [Editor] window in the following cases.

- While editing the contents in the [Trace conditions] dialog box
- When selecting [Fill until stop] or [Fill until full] of the trace mode

(3) Adding events by dragging and dropping

[When dragging and dropping the variable and function names in the [Editor] window]

1. Dragging and dropping a variable name into the [Event] column, you can set an event based on an access to that variable as a condition. => Data access condition

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

Only global or static variables of 1 or 2 bytes in size can be registered as an event. Static variables in functions cannot be registered as an event.

2. Dragging and dropping a function name into the [Event] column, you can set an event based on an instruction fetch to the start address of that function as a condition.

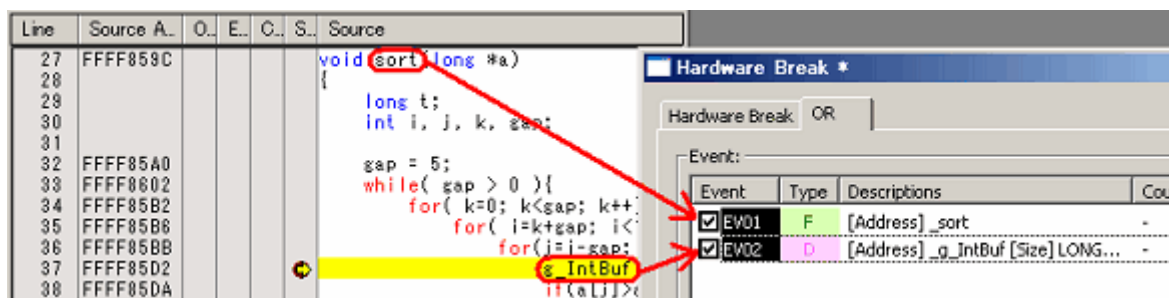


Figure 5.35 [Editor] Window and [Hardware Break] Dialog Box

[When dragging and dropping the address range in the [Memory] window]

Select memory content in the [Memory] window and drag and drop it into the [Event] column. That way, you can set a data access event based on the address range of the selected memory content as a condition. => Data access condition

[When dragging and dropping the label in the [Label] window]

You can set an event based on a fetch to that label as a condition. => Instruction fetch condition

5.7.3 Removing Events

Follow one of the following methods to remove events.

[When deleting an event from any setting dialog box]

1. To remove one point, select a line you want to remove in the event setting area and then click the [Delete] button. (You can use the keys Ctrl + Del instead of clicking the [Delete] button.)

The selected event will be removed from the event setting area.

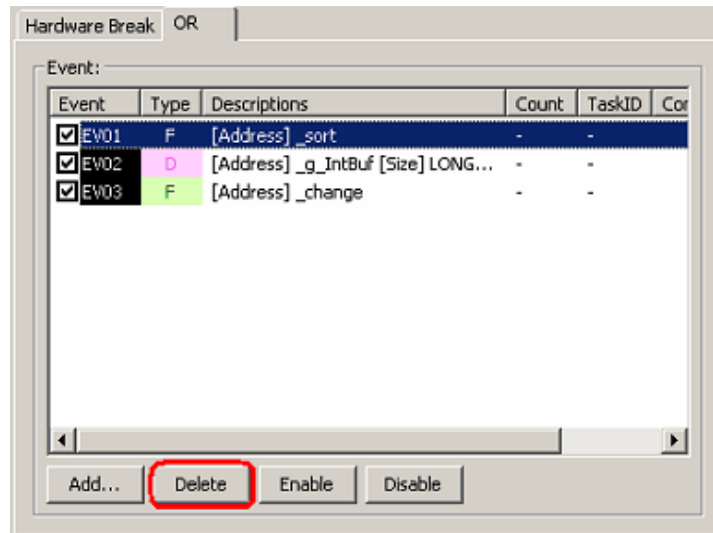


Figure 5.36 [Hardware Break] Dialog Box

2. To remove multiple events, hold down the Shift or the Ctrl key while you select lines you want to remove in the event setting area and then click the [Delete] button. (You can use the keys Ctrl + Del instead of clicking the [Delete] button.)

The selected events will be removed from the event setting area.

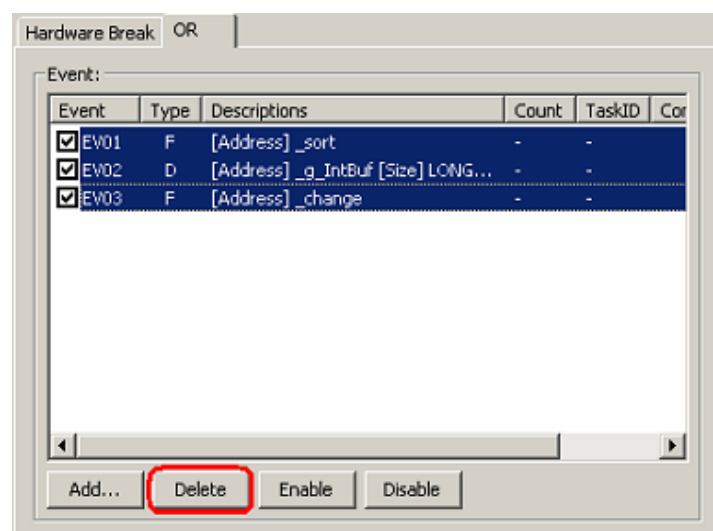


Figure 5.37 [Hardware Break] Dialog Box

[When deleting an event from the [Registered Events] dialog box]

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

The selected event will be removed from the list of registered events.

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

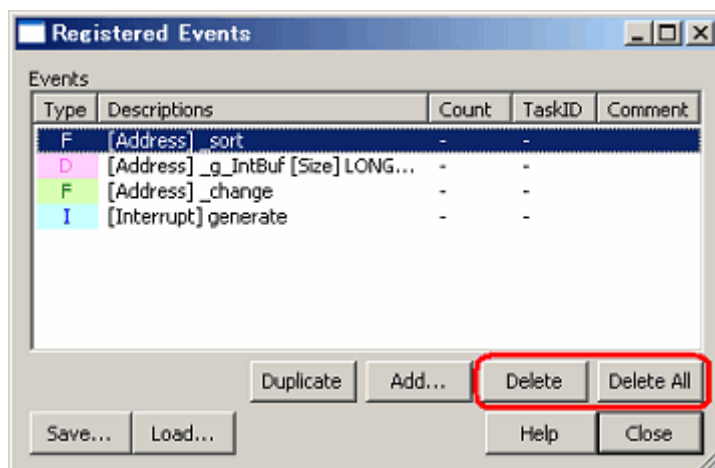


Figure 5.38 [Registered Events] Dialog Box

5.7.4 Registering Events

Registering an event refers to placing an event into the list of registered events. A registered event can be reused at a later time. Follow one of the following methods to register an event. Up to 256 events can be registered.

(1) Registering events

[When creating an event from the [Event] dialog box]

1. Display the [Comment] page of the [Event] dialog box and select the [Add this event to the list] checkbox. Then click the [OK] button.

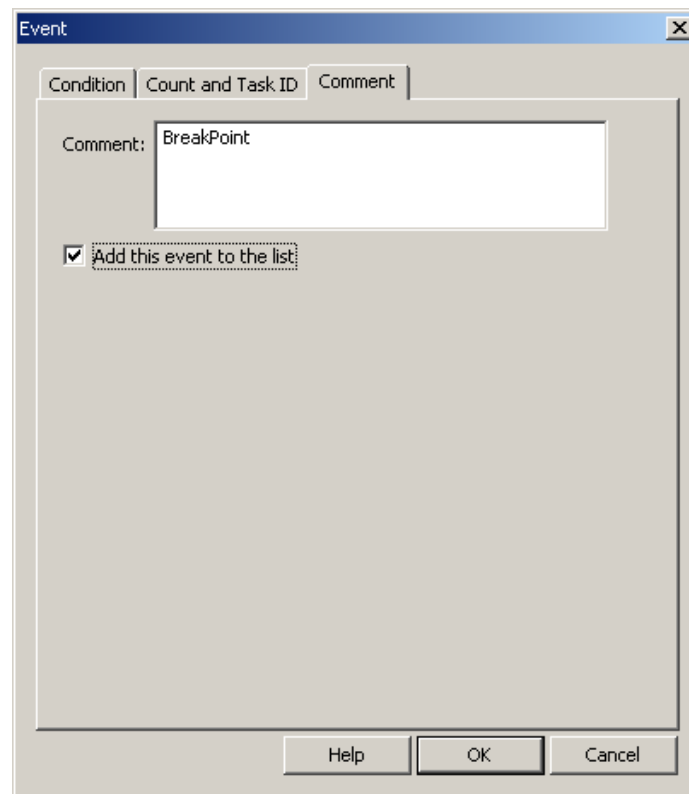


Figure 5.39 [Event] Dialog Box

2. An event is added at the specified position and registered in the [Registered Events] dialog box.

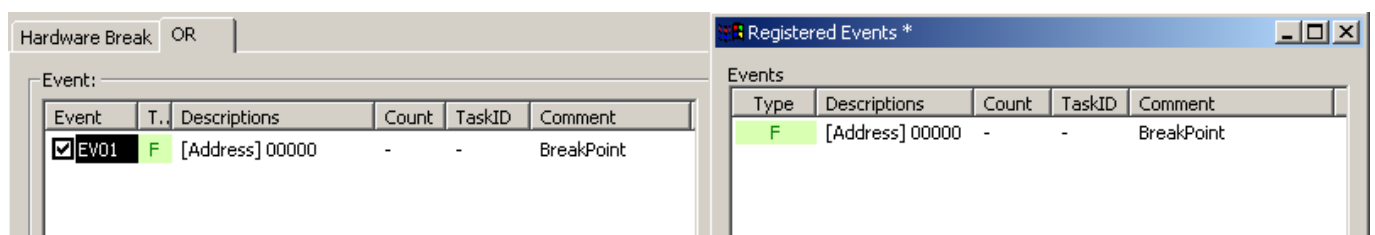


Figure 5.40 [Hardware Break] and [Registered Events] Dialog Boxes

[When registering an event by dragging and dropping]

The event you have created can be registered in the [Registered Events] dialog box by dragging and dropping it into the list.

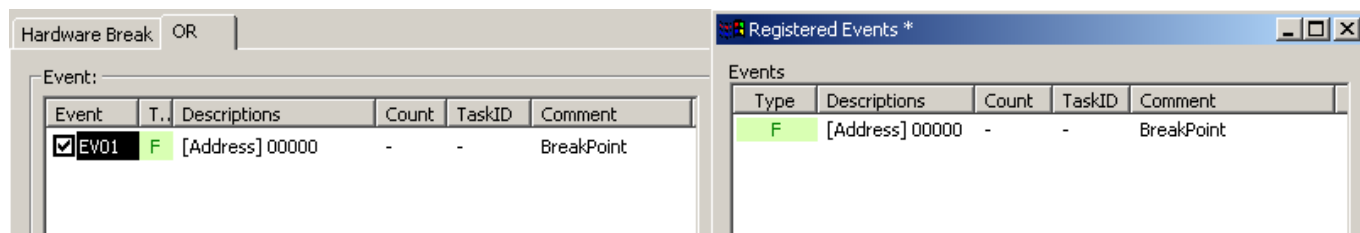


Figure 5.41 [Hardware Break] and [Registered Events] Dialog Boxes

[When registering an event from the [Registered Events] dialog box]

Click the [Add] button to create an event. The events you create here are added to the [Registered Events] dialog box.

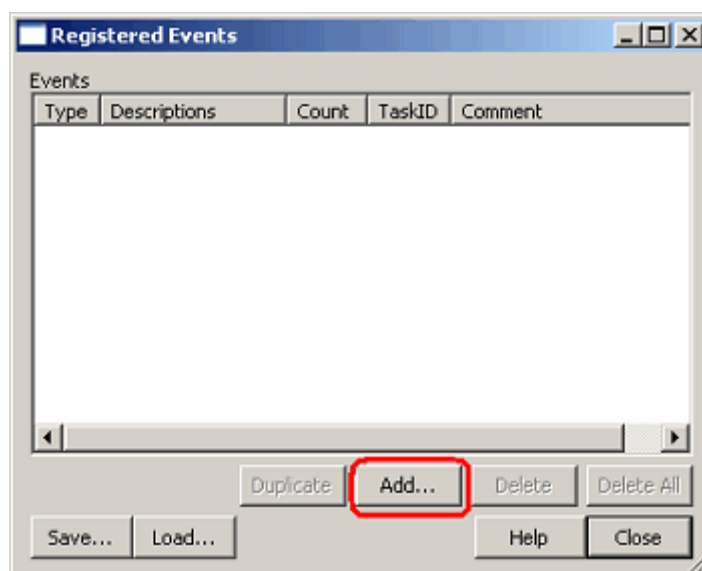


Figure 5.42 [Registered Events] Dialog Box

(2) Attaching comments

Attach a comment to the registered event as necessary. Check the [Registered Events] dialog box to know the registered contents and comments.

5.7.5 Entering Events Each Time or Reusing Events

There are following two methods to set events in any function concerned.

One method is to create events in the respective setting dialog boxes each time.

The other method is to choose one condition you want to use from the registered event list and drag and drop it into the condition area in which you want to set the event.

The former method is referred to here as entering events each time, and the latter as reusing events.

[Entering events each time]

This is the condition used only once. The event you created is used "without ever being registered".

After the event is used (i.e., changed or removed), its setting becomes nonexistent.

The events you create by only double-clicking in the [Event] column of the [Editor] window are the one that is entered each time.

[Reusing events]

Any event registered in the [Registered Events] dialog box can be reused by dragging and dropping it into the condition setting area of any function concerned.

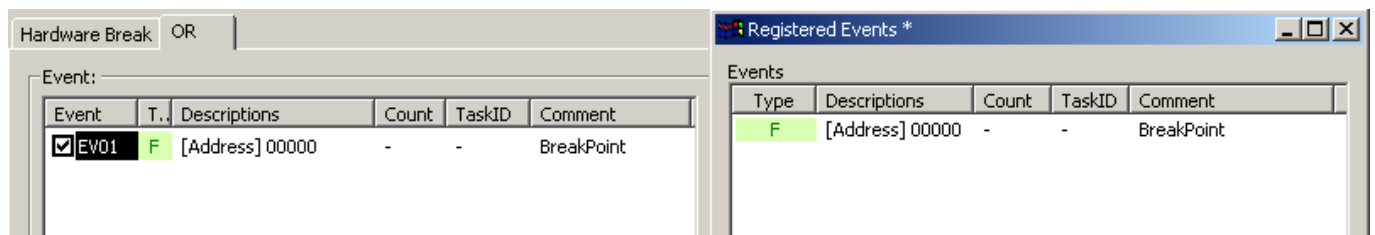


Figure 5.43 Schematic of Event Reuse

(1) Dragging and dropping into multiple functions

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

If the content of an event is altered after being dragged and dropped, the alteration you made is not reflected on the registered event list side.

(2) Registering duplicates in the registered event list

Even the events that have the same contents set can be registered in the list overlapping one another.

5.7.6 Applying Events

To enable the setting of an event after you have created it, click the [Apply] button. The content of what you have set has no effect until you click the [Apply] button.

[*] after the title on the title bar of the dialog boxes of [Hardware Break], [Trace conditions] or [Performance Analysis Conditions] shows that some setting is under editing. If you are doing some editing work, you cannot change the settings from the [Event] column of the [Editor] window or the command line.

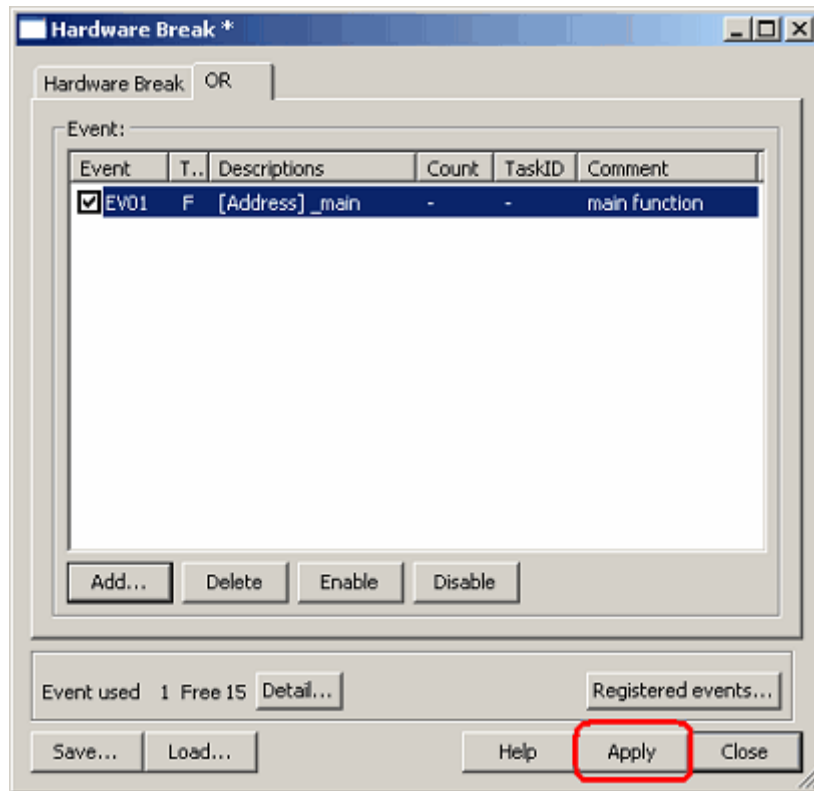


Figure 5.44 Applying the Setting

5.8 Setting Hardware Break Conditions

5.8.1 Setting Hardware Break Conditions

A hardware break causes the user program to stop running a specified number of cycles after a set event or phenomenon is detected (i.e., a hardware breakpoint is encountered). Up to 16 events can be specified as hardware breakpoint conditions.

5.8.2 Setting Hardware Breakpoints

(1) Setting Hardware Breakpoints

For hardware breakpoints, you can set an OR condition, other condition (AND (Accumulation), AND (Simultaneous), subroutine, sequential or state transition) and detection of exceptional events.

The OR condition, other condition and the detection of exceptional events can be set all at the same time, or only one at a time.

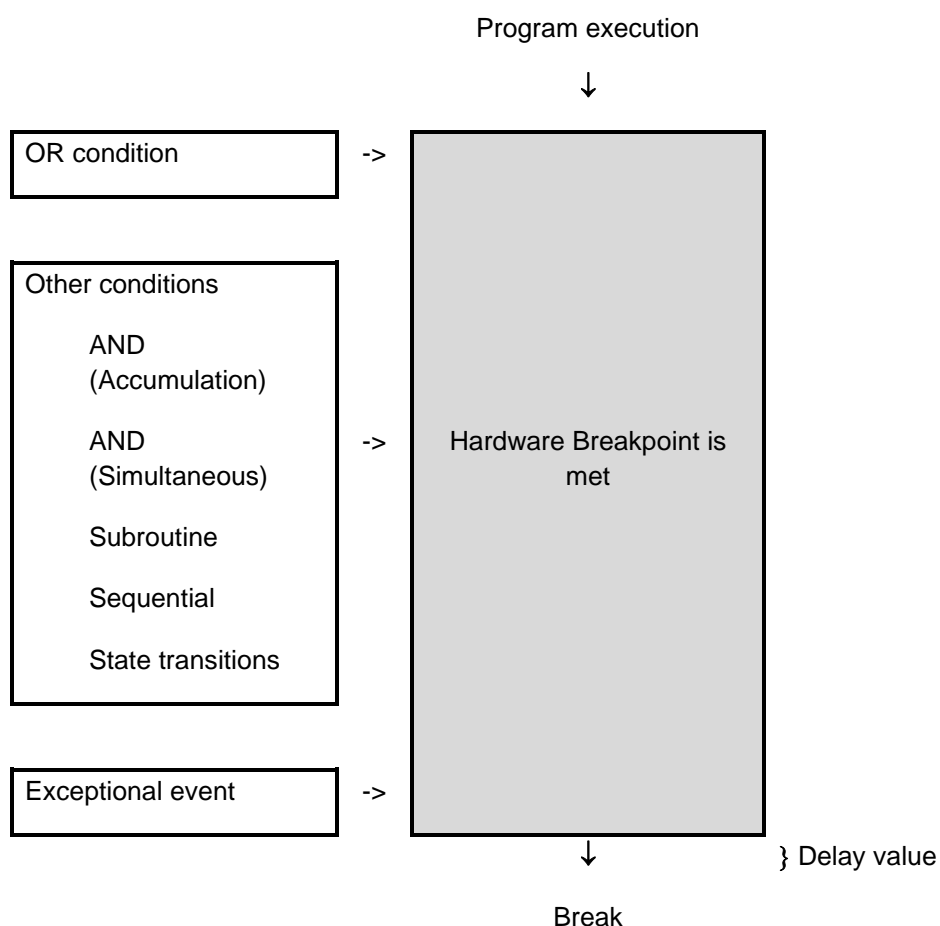


Figure 5.45 Outline of the Hardware Break

(2) Setting OR conditions

You can choose to enable or disable the OR condition. By default, the OR condition is enabled.

To disable the OR condition, deselect the checkbox to the left of "OR Condition".

If you add an event by double-clicking in the [Editor] window while the OR condition is disabled, the OR condition is automatically enabled.

If you reenables the OR condition when it is disabled, the previously set event is restored with its OR condition checkbox selected.

However, if a maximum of 16 events is exceeded when you have reenables for an event, the event is restored with its OR condition checkbox unselected (disabled).

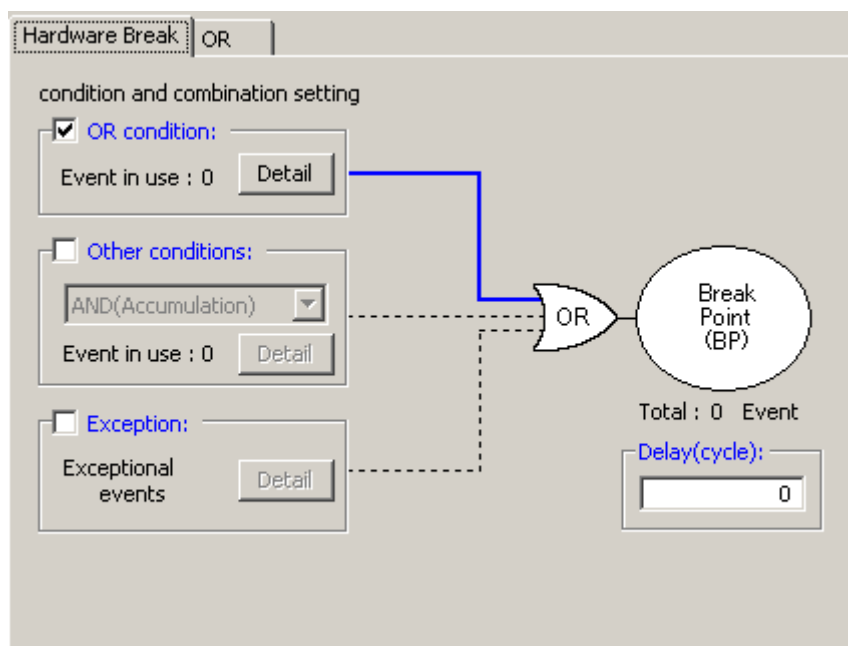


Figure 5.46 [Hardware Break] Dialog Box

Table 5.13 OR Condition

	Type	Description
1	OR condition	Breakpoint is encountered when any one of the set events occurs.

(3) Setting other conditions

You can select one of five choices available: AND (Accumulation), AND (Simultaneous), Subroutine, Sequential and State Transition. To set any condition, select the checkbox to the left of "Other Conditions". By default, other conditions are disabled (the checkbox to the left of "Other Conditions" is unselected).

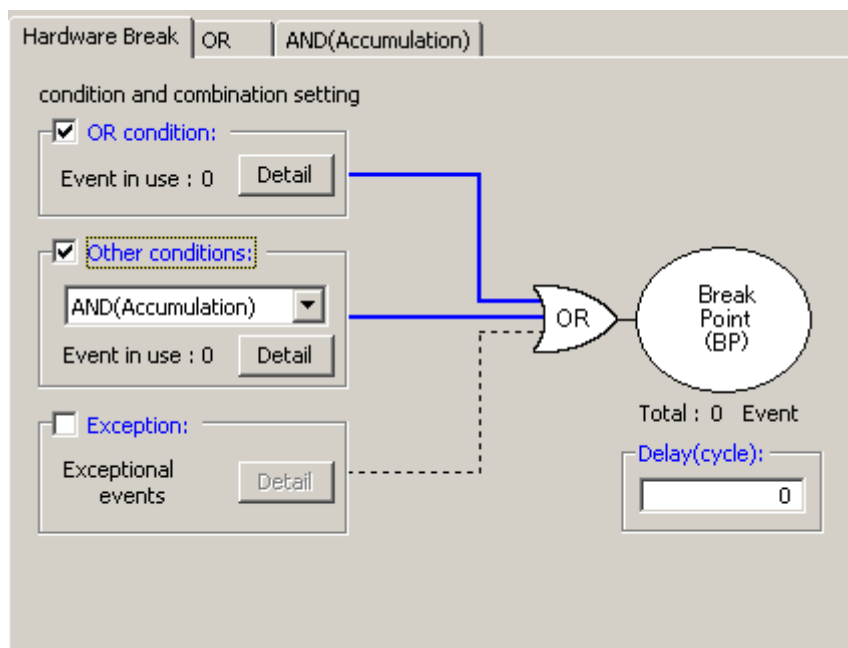


Figure 5.47 [Hardware Break] Dialog Box

Table 5.14 Other Conditions

	Type	Description
1	AND (Accumulation)	Breakpoint is encountered when all of the set events occur irrespective of the time axis.
2	AND (Simultaneous)	Breakpoint is encountered when all of the set events occur at the same time.
3	Subroutine	Breakpoint is encountered when a specified event occurs within a specified address range (subroutine or function).
4	Sequential	6 steps, (forward direction) + reset point Breakpoint is encountered when a set event occurs in a specified order.
5	State transitions	3 steps, 9 paths + reset point Breakpoint is encountered when a set event occurs in a specified order.

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

CAUTION

When a timeout condition is set for state transitions, the time to make transition from a set state to another then back to the original set state must be 10 μ s or more. Transition time of less than 10 μ s will result in an incorrect timeout detection.

(4) Detection of exceptional events

Specify whether you want detection of following exceptional events to be used as a breakpoint.

- Violation of access protection
- Read from uninitialized memory
- Stack access violation
- Performance overflow
- Realtime profile overflow
- Trace memory overflow
- Task stack access violation
- OS dispatch

(5) Specifying a delay value

The program breaks a specified number of cycles after a breakpoint is encountered. A breakpoint delay value can be set in the range from 0 to 65,535 bus cycles (default = 0).

5.8.3 Saving/Loading the Set Contents of Hardware Breaks

(1) Saving hardware break settings

Click the [Save] button of the [Hardware Break] dialog box. The [Save] dialog box will be displayed.

Specify a file name to which you want break settings to be saved. The file name extension is .hev. If omitted, the extension .hev is automatically attached.

(2) Loading the set contents of hardware breaks

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

When you load a file, the hardware break settings you had before you have loaded the file are discarded and the hardware breaks are reset with the loaded settings.

Click the [Apply] button of the [Hardware Break] dialog box to confirm the hardware break settings you have loaded.

5.9 Viewing Trace Information

5.9.1 Viewing Trace Information

A trace is the function to acquire bus information every cycle and store it in trace memory during user program execution. Using a trace you can track the flow of application execution or examine the points at which problems occurred.

The E100 emulator allows you to acquire up to 4-M bus cycles.

When program execution stops (for an exception break, forced halt or breakpoint), the contents stored in trace memory at the time the program has stopped are displayed as the trace result even when no trace points are encountered yet.

5.9.2 Acquiring Trace Information

The E100 emulator operates in such a way that when no trace information acquisition conditions are set, it by default traces all bus cycles to get trace information unconditionally. (Trace mode = Fill until stop)

In 'Fill until stop' mode, at the same time the user program starts running the emulator starts tracing bus cycles to get trace information, and 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 data table. The table has columns for Cycle, Label, Exe Address, EPCYLD, JNPTNF, OP Address, OP Data, OPSIZE, R/W, OPREND, DMA Address, DMA Data, and DM. The data rows show various bus cycles with addresses like FFFF89F2, FFFF89F8, FFFF89FC, FFFF8A01, FFFF8A05, FFFF8A05, FFFF87C3, FFFF87C5, FFFF87C8, FFFF87C8, FFFF87CB, FFFF87CD, FFFF87D7, FFFF87D9, FFFF87DC, FFFF87DC, and FFFF87DE. Some rows are highlighted in red, green, or blue.

Cycle	Label	Exe Address	EPCYLD	JNPTNF	OP Address	OP Data	OPSIZE	R/W	OPREND	DMA Address	DMA Data	DM
-0000026		FFFF89F2	1	-	00000000	-----	-	-	-	00000000	-----	-
-0000025		FFFF89F8	1	-	00000000	-----	-	-	-	00000000	-----	-
-0000024		FFFF89FC	1	-	0000142C	20DA775	4	W	Lit	00000000	-----	-
-0000023		FFFF89FE	1	-	00000000	-----	-	-	-	00000000	-----	-
-0000022		FFFF8A01	1	-	00000000	-----	-	-	-	00000000	-----	-
-0000021		FFFF8A05	1	-	00000000	-----	-	-	-	00000000	-----	-
-0000020		FFFF8A05	0	-	000016F0	FFFF87C	4	R	Lit	00000000	-----	-
-0000019		FFFF8A05	0	-	00000000	-----	-	-	-	00000000	-----	-
-0000018		FFFF87C3	1	-	00000000	-----	-	-	-	00000000	-----	-
-0000017		FFFF87C5	1	-	0000171C	000020D	4	W	Lit	00000000	-----	-
-0000016		FFFF87C8	1	-	00000000	-----	-	-	-	00000000	-----	-
-0000015		FFFF87C8	0	-	0000171C	000020D	4	R	Lit	00000000	-----	-
-0000014		FFFF87CB	1	-	00000000	-----	-	-	-	00000000	-----	-
-0000013		FFFF87CD	1	BR_T	00000000	-----	-	-	-	00000000	-----	-
-0000012		FFFF87D7	1	-	00000000	-----	-	-	-	00000000	-----	-
-0000011		FFFF87D9	1	-	00000000	-----	-	-	-	00000000	-----	-
-0000010		FFFF87DC	1	-	00001720	0000000	4	R	Lit	00000000	-----	-
-0000009		FFFF87DC	0	-	0000171C	000020D	4	R	Lit	00000000	-----	-
-0000008		FFFF87DE	1	-	00001718	000020D	4	W	Lit	00000000	-----	-

Figure 5.48 [Trace] Window

The following items of information are displayed. (This applies for bus display.)

Table 5.15 Display Items

Column	Description
Cycle	Cycle numbers stored in trace memory. The last cycle acquired is numbered 0, and the older cycles are assigned smaller numbers -1, -2, etc. sequentially retracing the past. If a delay count is set, the cycle in which a trace stop condition is met is numbered 0 and the cycles that were executed until the condition is met (cycles during a delay period) are assigned larger numbers +1, +2, etc. sequentially toward the last cycle acquired.
Label	Labels corresponding to addresses (displayed only when labels are set).
Exe Address	Shows the execution PC address.
EPCVLD	Shows the execution PC strobe signal. More specifically, it shows "1" when the execution PC address is valid or "0" when it is invalid.
JMPINF	Shows the execution status of a branch instruction. - Other than the following branches (e.g., BSR) or no branch BR_F Branched off by execution of a conditional branch instruction when the condition is not met BR_T Branched off by execution of a conditional branch instruction when the condition is met HW_INT Branched off by a hardware interrupt RTN Branched off by execution of RTE or RTFI instruction
OP Address	Shows the address of an operand access.
OP Data	Shows the data of an operand access.
OPSIZE	Shows the byte size of an operand access. 1 1-byte access 2 2-byte access 4 4-byte access 4_4 4-byte access, 1-byte operand (Four collective accesses) 4_2 4-byte access, 2-byte operand (Two collective accesses)
R/W	Shows the data bus state of an operand access. More specifically, it shows "R" for the read state, "W" for the write state, or "-" when no accesses are made.
OPREND	Shows the endian of an operand access. More specifically, it shows "Big" for the big endian and "Lit" for the little endian.
DMA Address	Shows the address of DMAC and DTC access.
DMA Data	Shows the data of DMAC and DTC access.
DMSIZE	Shows the byte size of DMAC and DTC access. 1 1-byte access 2 2-byte access 4 4-byte access
DMA_DTC	Shows whether the access made is a DMAC or a DTC access. More specifically, it shows "DMA" for the DMAC access or "DTC" for the DTC access.
DMA R/W	Shows the data bus state of DMAC and DTC access. More specifically, it shows "R" for the read state, "W" for the write state, or "-" when no accesses are made.
DEEND	Shows the endian of DMAC and DTC access. More specifically, it shows "Big" for the big endian and "Lit" for the little endian.

BUSACC	When memory access is performed by a debugger operation during user program execution, shows "0" during the emulator is occupying the MCU bus. [CAUTION] Execution of the user program is temporarily stopped during such access to memory.
EV	The event No. when a set event occurred. To show EV column, you need to select [Event number] on the [Option] page of the [Trace conditions] dialog box displayed from the menu of the [Trace] window.
TID	Task ID (when RTOS is used). Example display: A task ID (task entry label) is displayed such as 1 (_Task1). To show the TID column, you need to select the task ID on the [Option] page of the [Trace conditions] dialog box displayed from the menu of the [Trace] window.
EXT	Shows the signal fed in from the external trigger cable, indicated as "1" when the signal is high or "0" when the signal is low. To show the EXT column, you need to select the external trigger on the [Option] page of the [Trace conditions] dialog box displayed from the menu of the [Trace] window.
TimeStamp	Shows an elapsed time since the target program started. Each time the user program starts running, the timestamp starts counting from 0. [CAUTION] When the counter overflows, the time is not displayed correctly.

The unnecessary columns in the [Trace] window can be hidden. To hide a column, right-click in the header column and select the column you want to hide from the pop-up menu.

5.9.3 Setting Trace Information Acquisition Conditions

The trace buffer is limited in size, so that when the buffer is filled, the old trace data is overwritten with new data sequentially beginning with the oldest.

Setting trace information acquisition conditions, you can acquire only the useful trace information, making effective use of the trace buffer.

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

(1) Setting trace modes

First, select a trace mode.

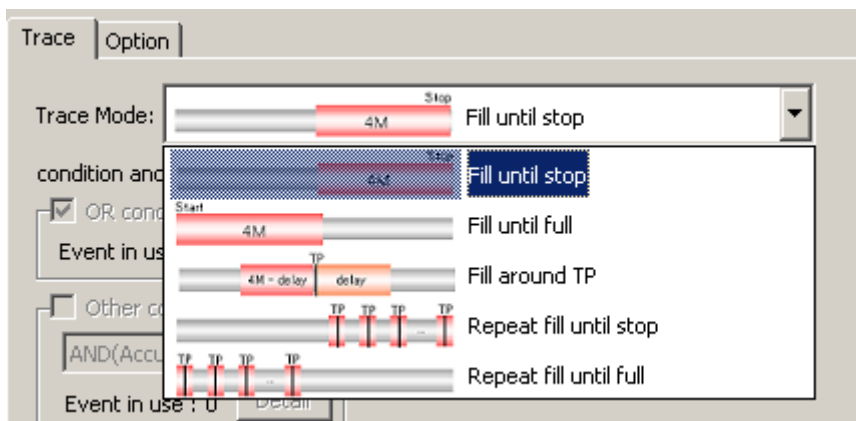


Figure 5.49 [Trace conditions] Dialog Box

(2) Setting trace points

If you selected [Fill around TP], [Repeat fill until stop] or [Repeat fill until full] for the trace mode, set a trace point.

For trace points, you can set event based on conditions and exceptional events.

For [Fill around TP], furthermore, you can set a delay value.

(3) Setting Capture/Do not Capture

If the selected trace mode is [Fill until stop], [Fill until full] or [Fill around TP], you can specify Capture/Do not Capture conditions in the [Record condition] group box.

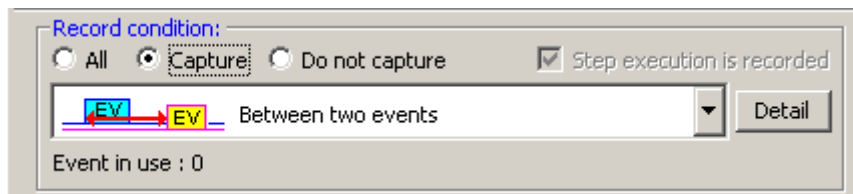


Figure 5.50 [Record condition] Group Box

You can choose to extract only the necessary portions of trace information specified by events or delete the unnecessary portions.

(4) Recording step execution

If the selected trace mode is [Fill until stop], you can record step execution. To record step execution, select the [Step execution is recorded] checkbox in the [Record condition] group box.

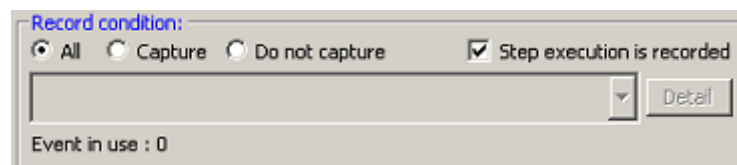


Figure 5.51 Recording Step Execution

The recordable modes of step execution are Step In, Step Over and Step Out.

(5) Setting trace acquisition methods

Use the [Option] page of the [Trace conditions] dialog box to set the acquisition method associated with the entire trace. By default, selecting trace acquisition is set for the event number.

5.9.4 Setting Trace Modes

(1) Setting trace modes

Following five trace modes are available.

Table 5.16 Trace Modes

	Stop Mode	Description
1	Fill until stop	Trace acquisition continues until the program stops running.
2	Fill until full	Trace acquisition stops when the trace memory is filled.
3	Fill around TP	Trace acquisition stops a specified number of cycles after a trace point is encountered. A delay value can be specified in a range of up to the maximum value of trace capacity.
4	Repeat fill until stop	Each time a trace point is encountered, acquisition is made for a total of 512 cycles* before and after that point, and acquisition continues that way until the program stops running.
5	Repeat fill until full	Each time a trace point is encountered, acquisition is made for a total of 512 cycles* before and after that point, and acquisition continues that way until the trace memory is filled.

CAUTION

Recording is made in units of total 512 cycles, consisting of 1 cycle at the line where a trace point is met and 255 cycles before that point and 256 cycles after that point.

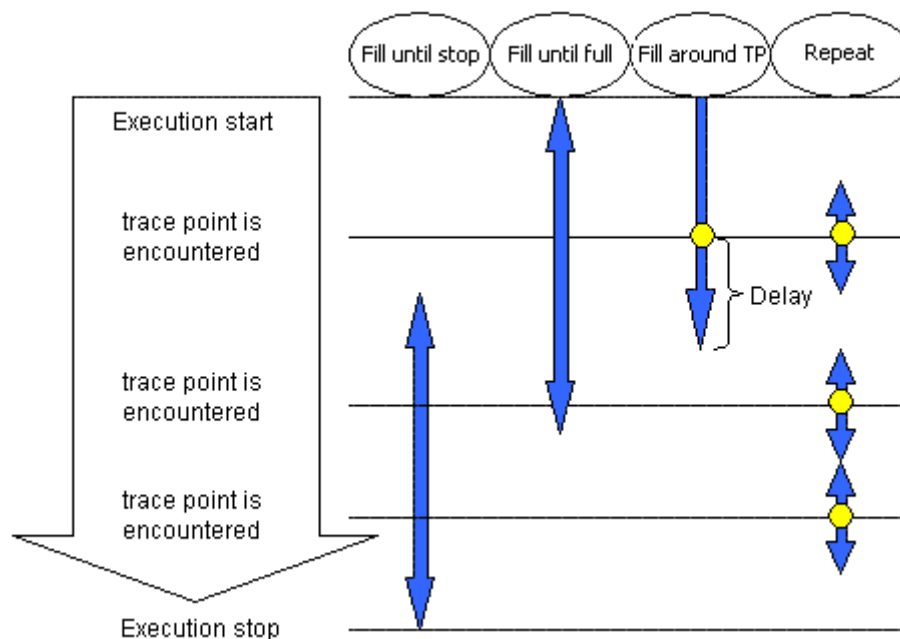


Figure 5.52 Differences between the Trace Modes

Specifiable conditions vary depending on trace mode, as summarized in the table below.

1. Fill until stop

The trace memory can hold up to 4-M bus cycles. When the buffer is filled, the oldest data of the acquired trace information is overwritten with new data. That way, the emulator continues acquiring trace information.

Table 5.17 Specifiable Conditions: Fill until stop

Trace Point Setting	Delay Specification	Capture/Do not Capture Condition Setting	Step Execution Recording
-	-	Possible	Possible

2. Fill until full

When the trace memory of the emulator main unit overflows during trace acquisition, the emulator stops acquiring trace information.

Table 5.18 Specifiable Conditions: Fill until full

Trace Point Setting	Delay Specification	Capture/Do not Capture Condition Setting	Step Execution Recording
-	-	Possible	-

3. Fill around TP

Trace acquisition is halted a specified number of cycles delayed after a trace point is encountered. In this mode, the user program continues running and only trace acquisition is halted. Sophisticated conditions can be set using a maximum of 16 event points. A delay value can be chosen to be 0, 1-M, 2-M, 3-M or 4-M cycles.

Table 5.19 Specifiable Conditions: Fill around TP

Trace Point Setting	Delay Specification	Capture/Do not Capture Condition Setting	Step Execution Recording
Possible	Possible	Possible	-

4. Repeat fill until stop

Each time a trace point is encountered, a total of 512 cycles before and after that point are acquired, and acquisition continues that way. Acquisition continues until it is halted by a break or forced stop. The positions where trace points are encountered can be checked in the [Trace] window.

Table 5.20 Specifiable Conditions: Repeat fill until stop

Trace Point Setting	Delay Specification	Capture/Do not Capture Condition Setting	Step Execution Recording
Possible	-	-	-

5. Repeat fill until full

Each time a trace point is encountered, a total of 512 cycles before and after that point are acquired, and acquisition continues that way. When the trace memory overflows, acquisition is halted. The positions where trace points are encountered can be checked in the [Trace] window.

Table 5.21 Specifiable Conditions: Repeat fill until full

Trace Point Setting	Delay Specification	Capture/Do not Capture Condition Setting	Step Execution Recording
Possible	-	-	-

CAUTION

If trace points are encountered in consecutive cycles in the repeat fill until stop or repeat fill until full mode, only one first cycle is highlighted in yellow as a trace point.

5.9.5 Setting Trace Points

(1) Setting trace points

For trace points, you can set an OR condition, other condition (AND (Accumulation), AND (Simultaneous), subroutine, sequential or state transition) and detection of exceptional events.

The OR condition, other condition and the detection of exceptional events can be set all at the same time, or only one at a time.

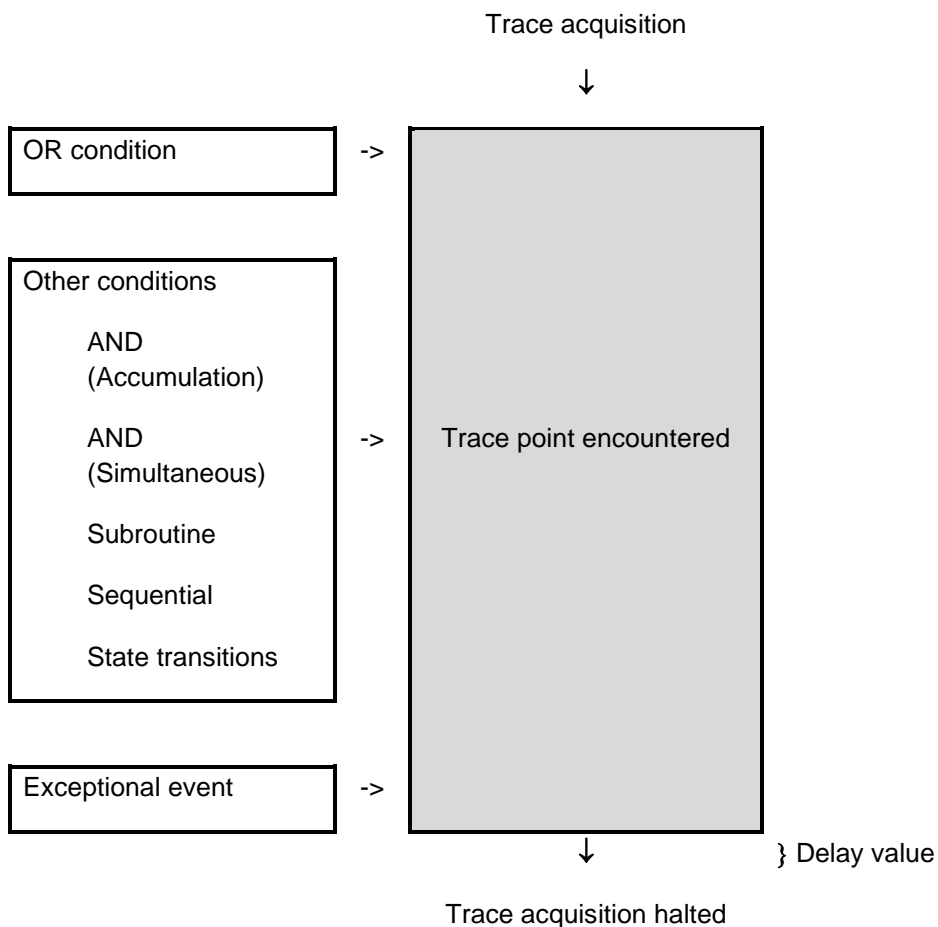


Figure 5.53 Outline of the Trace Point

(2) OR condition

You can choose to enable or disable [OR condition]. By default, [OR condition] is enabled.

If you reenables the OR condition when it is disabled, the previously set event is restored with its [OR condition] checkbox selected. However, if a maximum of 16 points is exceeded when you have reenables for an event, the event is restored with its [OR condition] checkbox unselected (disabled).

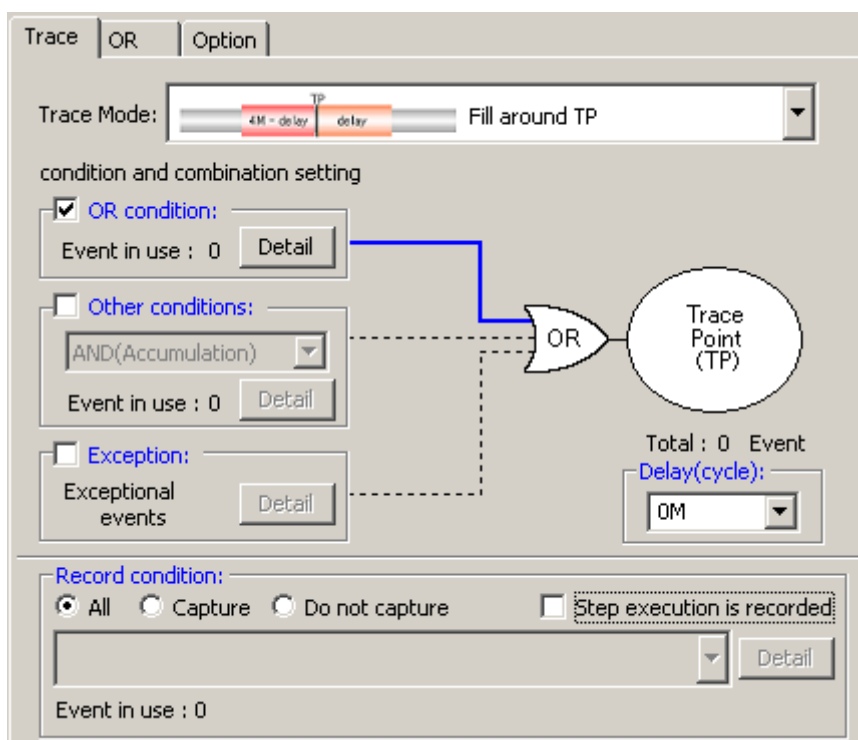


Figure 5.54 [Trace conditions] Dialog Box

Table 5.22 OR Condition

	Type	Description
1	OR condition	Trace point is encountered when any one of the set events occurs.

(3) Other conditions

You can select one of five choices available: AND (Accumulation), AND (Simultaneous), Subroutine, Sequential and State Transition. To set any condition, select the checkbox to the left of [Other conditions].

By default, other conditions are disabled (the checkbox to the left of [Other conditions] is unselected).

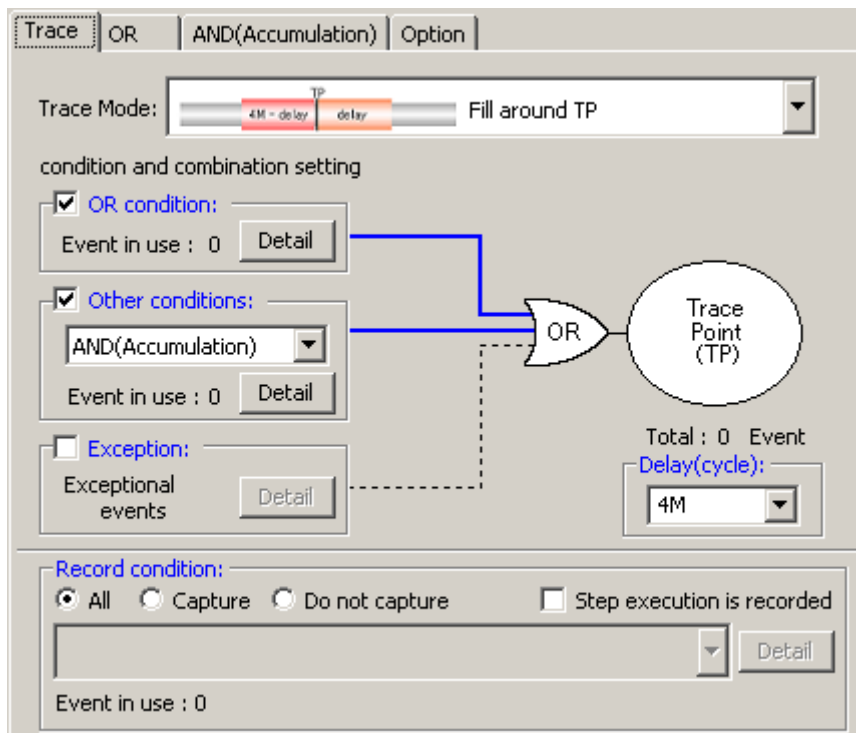


Figure 5.55 [Trace conditions] Dialog Box

Table 5.23 Other Conditions

	Type	Description
1	AND (Accumulation)	Trace point is encountered when all of the set events occur irrespective of the time axis.
2	AND (Simultaneous)	Trace point is encountered when all of the set events occur at the same time.
3	Subroutine	Trace point is encountered when a specified event occurs within a specified address range (subroutine or function).
4	Sequential	6 steps (forward direction) + reset point Trace point is encountered when a set event occurs in a specified order.
5	State transitions	3 steps, 9 paths + reset point Trace point is encountered when a set event occurs in a specified order.

CAUTION

When a timeout condition is set for state transitions, the time to make transition from a set state to another then back to the original set state must be 10 μ s or more. Transition time of less than 10 μ s will result in an incorrect timeout detection.

(4) Detection of exceptional events

Specify whether you want detection of following exceptional events to be used as a trace point.

- Violation of access protection
- Read from uninitialized memory
- Stack access violation
- Performance overflow
- Realtime profile overflow
- Task stack access violation
- OS dispatch

(5) Specifying a delay value

The program breaks a specified number of cycles delayed after a trace point is encountered.

A trace-point delay value can be selected from 0-M, 1-M, 2-M, 3-M or 4-M bus cycles (default: 0 M).

Select your desired delay value in the [Delay (cycle)] column.

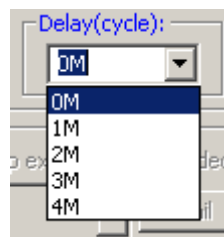


Figure 5.56 Setting a Delay Value

5.9.6 Setting Capture/Do not Capture Conditions



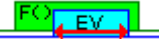

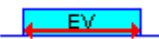

If the selected trace mode is [Fill until stop], [Fill until full] or [Fill around TP], you can specify Capture/Do not Capture conditions.

You can choose to extract only the necessary portions of trace information specified by events or delete the unnecessary portions.

(1) Capture/Do not Capture conditions

There are following types of conditions.

Table 5.24 Capture/Do not Capture Conditions

	Type		Description
Extraction		Between two events	Cycles extracted begin when a start event occurs and end with the next to last end event occurs (the cycle where an end event occurs is not extracted).
		Duration of an event	Only cycles where a specified event occurred are extracted.
		Duration of an event occurring in a subroutine	Only cycles where a specified event occurred in a specified address range (subroutine or function) are extracted.
Deletion		Between two events	Cycles extracted are deleted when a start event occurs and end with the next to last end event occurs (the cycle where an end event occurs is not extracted).
		Duration of an event	Only cycles where a specified event occurred are deleted.
		Duration of an event occurring in a subroutine	Only cycles where a specified event occurred in a specified address range (subroutine or function) are deleted.

Select the conditions you want to set from the list box that is displayed when you select [Capture] or [Do not capture] in the [Record condition] group box of the [Trace conditions] dialog box.

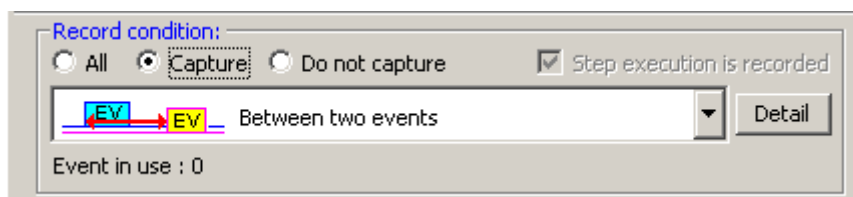


Figure 5.57 [Record condition] Group Box

Then click the [Detail] button. A page in which you can set events will be displayed.

CAUTION

When you specify extraction or deletion conditions, you cannot select DIS (disassemble display) and SRC (source display) from [Display Modes] in the [Trace] window.

5.9.7 Selecting the Content of Trace Acquisition

Select the content of trace information you want to be captured into trace memory. Use the [Option] page of the [Trace conditions] dialog box to make this selection.

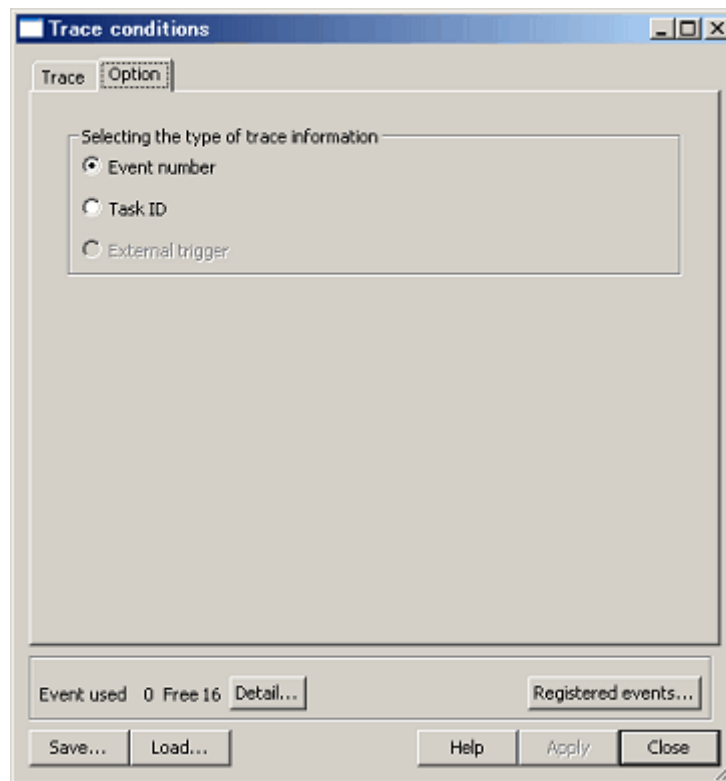


Figure 5.58 [Trace conditions] Dialog Box

Select which signal you want to be acquired from three choices available: Event number, Task ID or External trigger. By default, [Event number] is selected.

CAUTION

- If you want a history of task execution to be displayed in trace acquisitions carried out by running a realtime OS program, be sure to select [Task ID].
- The option "External trigger" becomes selectable when trigger cable is connected.

5.9.8 Showing Trace Results

To check trace results, view the [Trace] window. Trace results can be shown in one of the following display modes.

These display modes can be switched using [Display Modes] on the pop-up menu of the [Trace] window.

There are five trace result display modes: Bus Display, Disassembled Display, Source Display and Mixed Display.

(1) Bus Display Mode

In the pop-up menu, select [Display Modes -> BUS]. Bus information on each cycle traced are displayed. (Default display mode)

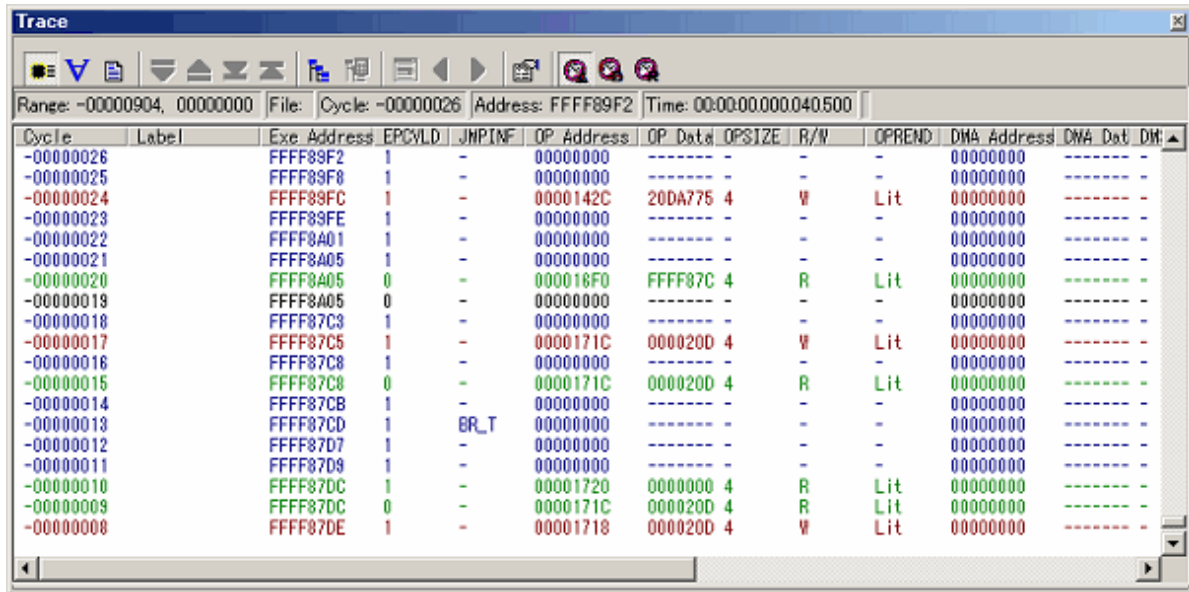


Figure 5.59 [Trace] Window

(2) Disassembled Display Mode

From the pop-up menu, choose [Display Modes -> DIS]. This display mode allows you to inspect the machine language instructions executed.

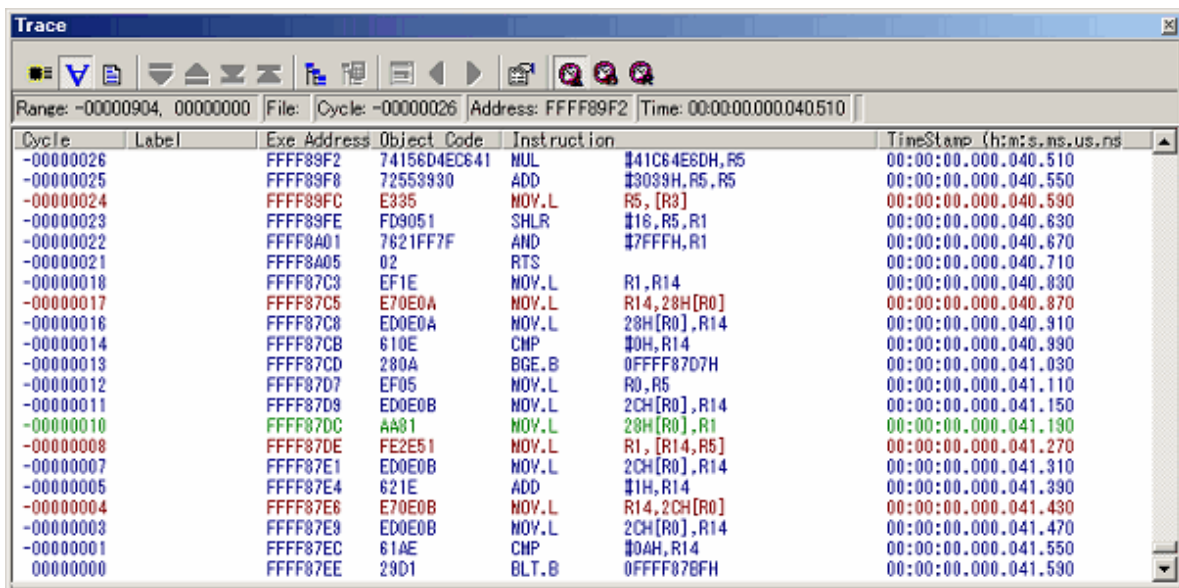


Figure 5.60 [Trace] Window

(3) Source Display Mode

From the pop-up menu, choose [Display Modes -> SRC]. This display mode allows you to inspect the source program's execution path.

The execution path can be verified by stepping through the source within trace data forward or backward from the current trace cycle.

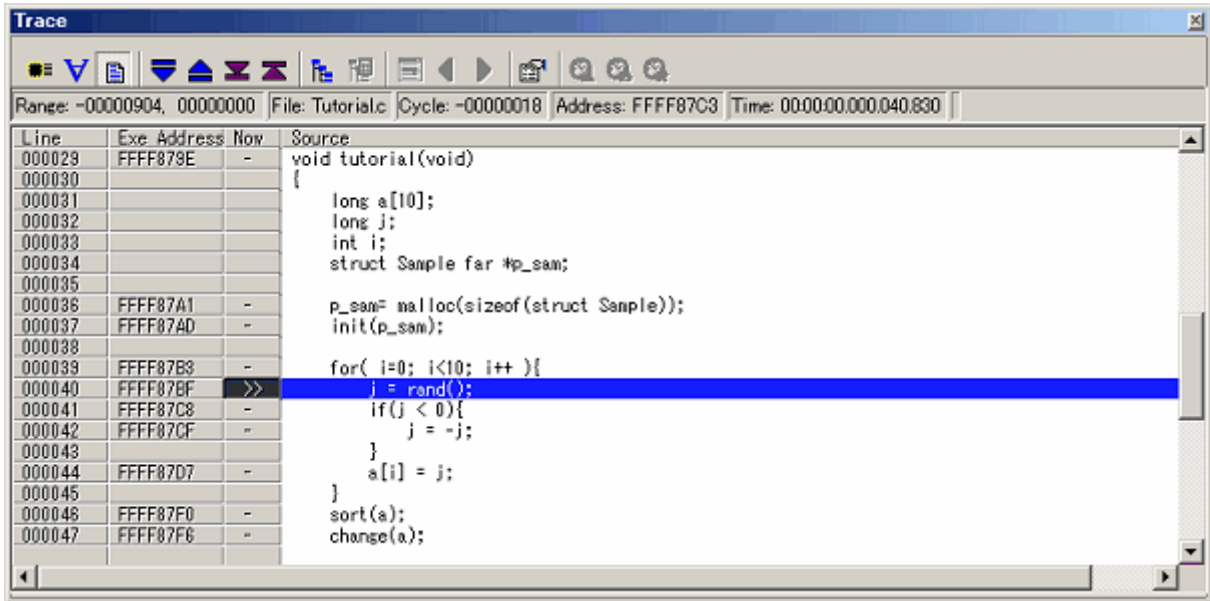


Figure 5.61 Source Display Screen

(4) Mixed Display Mode

This display mode provides a mixed display of bus, disassemble or source display.

After choosing [Display Modes -> BUS] from the pop-up menu, select [Display Modes -> DIS]. That way, you can produce a bus and disassemble mixed display.

In the same way, you can produce a bus and source, a disassemble and source or a bus, or disassemble and source mixed display.

To revert to a bus only display after viewing a bus and disassemble mixed display, choose [Display Modes -> DIS] from the pop-up menu again.

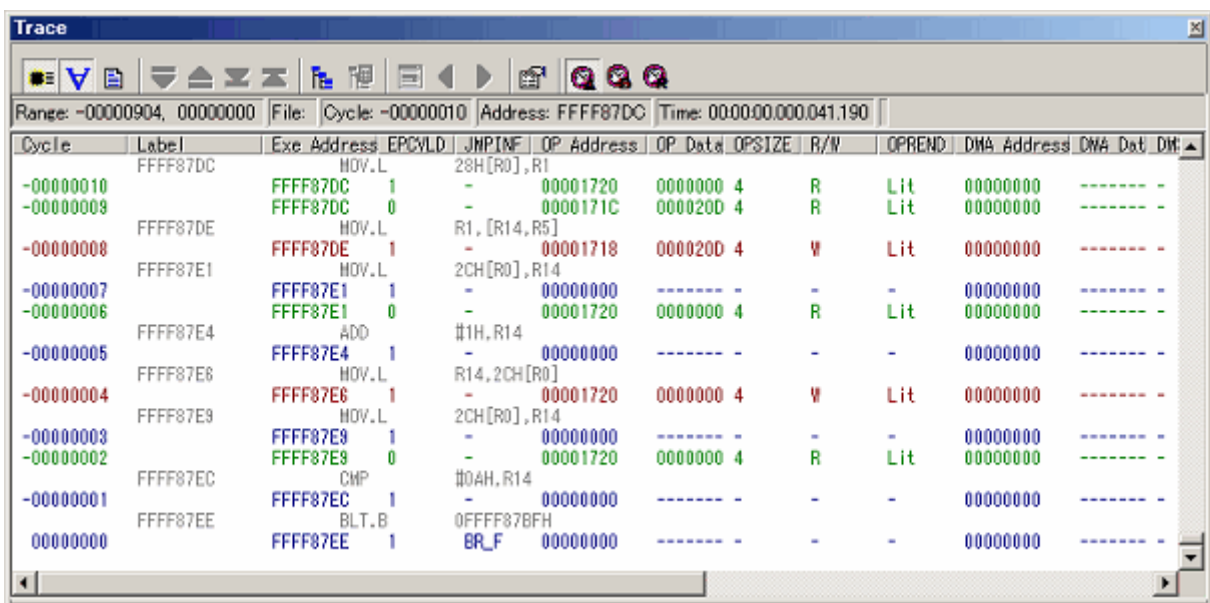


Figure 5.62 [Trace] Window

5.9.9 Filtering Trace Information

Use the filter function to extract only the necessary records from the acquired trace information. The filter function filters the trace information in software that was acquired by hardware.

Unlike the [Capture/Do not Capture] conditions where you set acquisition conditions before getting trace information, this function permits you to change filter settings for the acquired trace information any number of times.

Therefore, the necessary information can be extracted easily, with data analysis significantly facilitated.

The filter function does not affect the trace memory, so that its content remains intact.

The filter can be used when the selected trace mode is [Fill until stop], [Fill until full] or [Fill around TP] and the selected display mode is [Bus] or [Disassembled].

(1) Auto-filter function

To use the filter function, choose [Auto Filter] from the pop-up menu of the [Trace] window. When [Auto Filter] is turned on, each column of the [Trace] window is marked with an auto-filter arrow [▼].

Click any arrow [▼] and select the necessary condition from the ensuing drop-down list. That way, you can easily filter the records to get those that meet the condition. Selecting [Option] in the drop-down list brings up the [Option] dialog box. In this dialog box, you can set detail conditions.

Some columns for address and data etc. have only [Option] provided in the drop-down list because they do not have other inherent items. Selecting [All] returns you to a non-filter state.

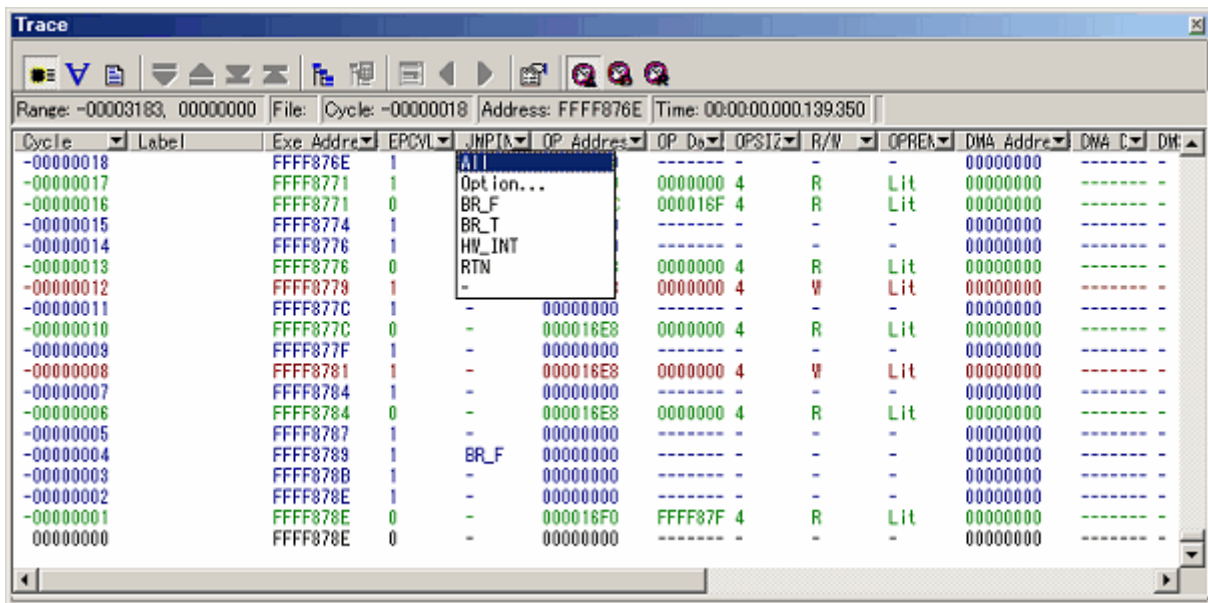


Figure 5.63 [Trace] Window

If after filtering records in bus display mode you switch to disassemble-only or source-only display, [Auto Filter] is deselected. Similarly, when after filtering records in disassembled display mode you switch to bus-only or source-only display, [Auto Filter] is deselected.

If there are multiple items you can specify in the [Option] dialog box, these items can be used as an OR condition with which to filter.

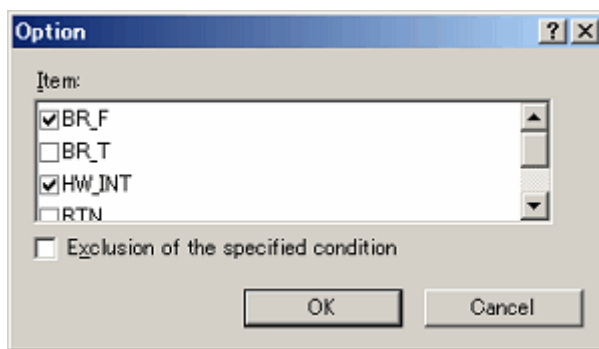



Figure 5.64 [Option] Dialog Box

5.9.10 Searching for Trace Records

You can search the acquired trace information for a specific trace record.

To search for trace records, use the [Find] dialog box. To open it, choose [Find -> Find] from the pop-up menu of the [Trace] window or click the [Find] button  in the toolbar.

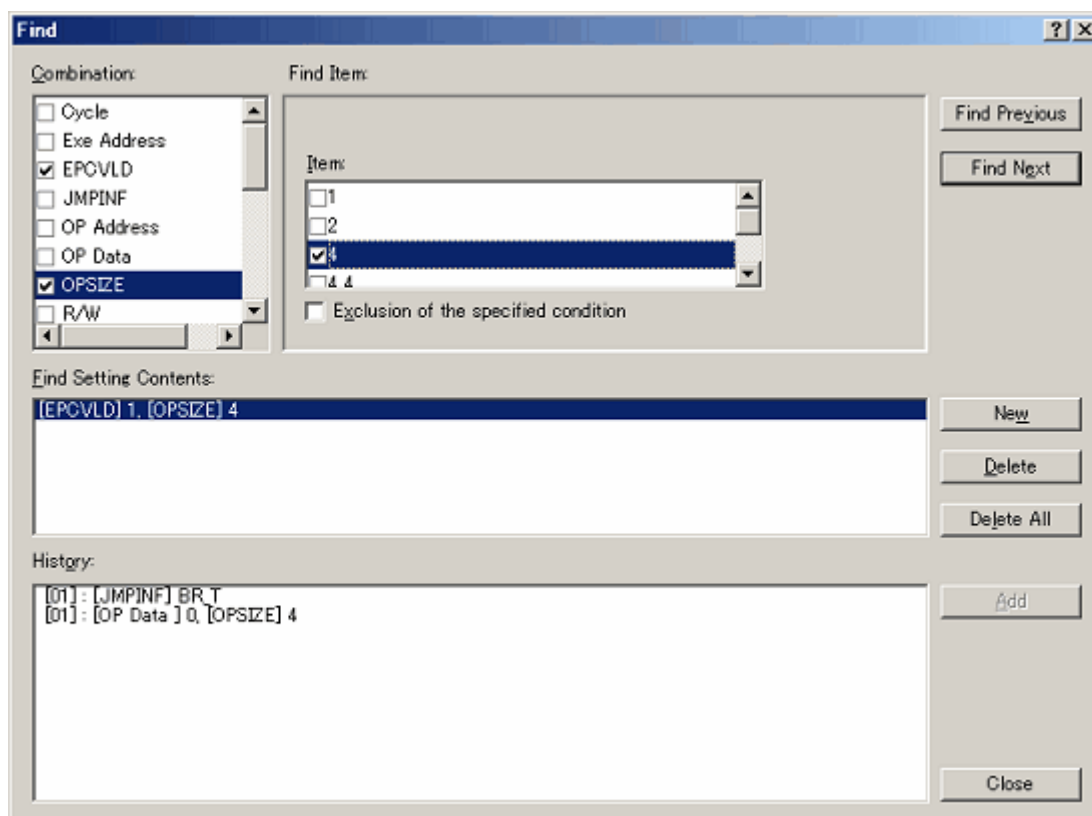


Figure 5.65 [Find] Dialog Box

Select the conditions you want to search for in the [Combination] column and select the checkboxes.

In the [Find Item] column, you can select the items that correspond to the selected conditions.

If you checked more than one condition in the [Combination] column, set items for each condition. The items you have set are searched for as multiple AND conditions.

The conditions you have set are shown in the [Find Setting Contents] list box.

After setting search conditions, click the [Find Previous] or [Find Next] button to start a search. Trace records are searched in forward or searched in reverse from the line you have clicked in the [Trace] window (the line highlighted in blue).

When a matching trace record is found by a search, the relevant line in the [Trace] window is highlighted. If no matching trace records are found, a message dialog box is displayed.

When an instance of the trace record was successfully found, choose [Find Previous] or [Find Next] from the pop-up menu. The next instance of the trace record will be searched for.

(1) Search history

The conditions once searched are left as a history in the history column while the High-performance Embedded Workshop remains active.

The next time you perform a search, choose the line you want to search from this history and click the [Add] button. That way, you can search trace information with that condition again.

The search history contains a history of up to 10 last searches performed.

(2) OR search

You can perform a search using two or more search conditions as OR conditions.

To set OR conditions, begin by setting the first condition (shown on the first line in the search content setting column) and then click the [New] button.

Then enter the second condition. At this time, the second condition is added to the second line in the search content setting column.


In this case, the conditions on the first and second lines in the search content setting column can be used as OR conditions for a search performed.

Up to 16 conditions (16 lines) can be set.

CAUTION

The conditions set on one and the same line in the search content setting column comprise AND conditions.

5.9.11 Saving Trace Information to Files

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

(1) Saving in binary format

To save trace information in 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 pop-up menu.


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

(2) Saving in text format

To save trace information in 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 pop-up menu.

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


5.9.12 Loading Trace Information from Files

To load trace information from a file, choose [File -> Load] from the pop-up menu or click the [Load] button  in the toolbar. Specify a trace information file saved in binary format. The current trace result is overwritten.

Before loading a file saved in binary format, switch to the trace mode in which mode you saved trace information. Do this switching in the [Trace conditions] dialog box that is displayed when you choose [Acquisition] from the pop-up menu of the [Trace] window.


If the current trace mode differs from the one in which mode you saved trace information, an error results. Trace information files saved in text format cannot be loaded into the [Trace] window.

5.9.13 Temporarily Stopping Trace Information Acquisition

To temporarily stop acquiring trace information during user program execution, choose [Trace -> Stop] from the pop-up menu of the [Trace] window or click the [Stop] button  in the toolbar.

Trace acquisition will be aborted, with the trace display updated. Use this function when you only want to stop acquiring trace information and check the trace information without stopping program execution.


5.9.14 Restarting Trace Information Acquisition

If after temporarily stopping acquisition of trace information during user program execution you want to start acquiring trace information again, choose [Trace -> Restart] from the pop-up menu of the [Trace] window or click the [Restart] button  in the toolbar.


5.9.15 Switching Timestamp Display

The timestamp displayed in the [Trace] window can be switched to absolute time, differential time or relative time. In the initial state, the timestamp is displayed in absolute time.


(1) Absolute time

From the pop-up menu, choose [Time -> Absolute Time] or click the [Absolute Time] button  in the toolbar. The timestamp will be displayed by an absolute time since program execution started.


(2) Differential time

From the pop-up menu, choose [Time -> Differences] or click the [Differences] button  in the toolbar. The timestamp will be displayed by a differential time from the preceding cycle.


(3) Relative time

From the pop-up menu, choose [Time -> Relative Time] or click the [Relative Time] button  in the toolbar. The timestamp will be displayed by a relative time from a specified cycle.

5.9.16 Showing the History of Function Execution

To show the history of function execution from the acquired trace information, choose [Function Execution History - > Function Execution History] from the pop-up menu or click the [Function Execution History] button  in the toolbar.

An upper pane of the window will be displayed. (Initially, this window is blank.)

When you choose [Analyze Execution History] from the pop-up menu or click the [Analyze Execution History] button  in the toolbar, the emulator starts analyzing the execution history from the end of the trace result and shows the result in a tree structure.

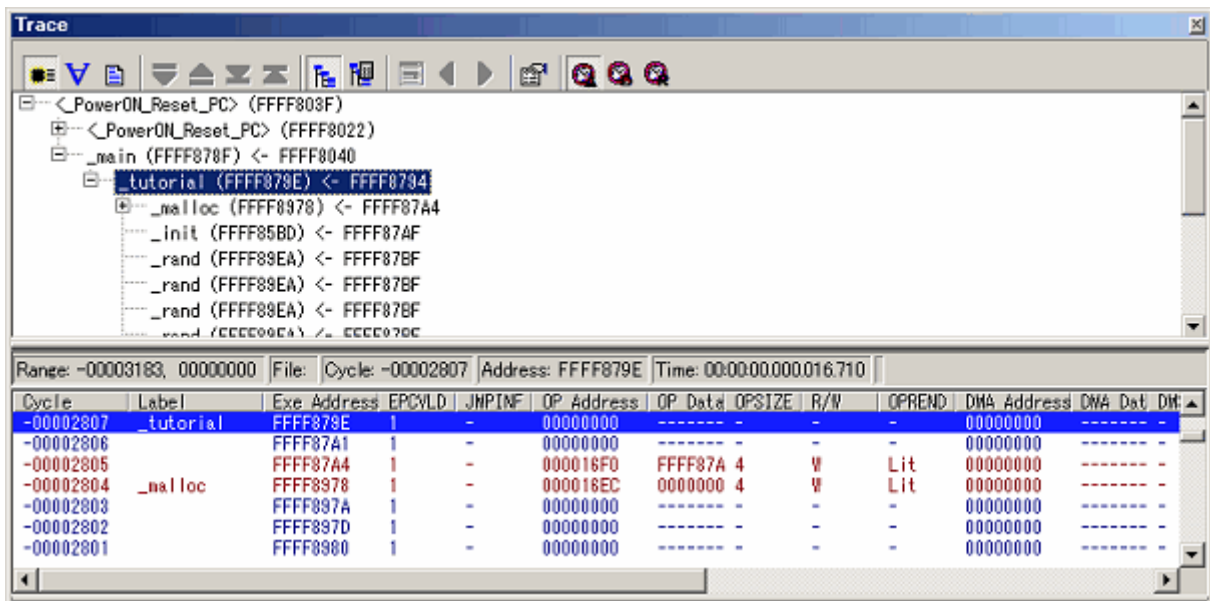


Figure 5.66 [Trace] Window

The lower pane of the window shows the trace result beginning with the cycle in which the function selected in the upper pane was called.

The lower pane of the window can show trace results in disassemble, source or mixed mode.

CAUTION


If trace extraction or deletion conditions are specified, the function execution history cannot be displayed.

If repeat (free) or repeat (full) mode is specified, the function execution history cannot be displayed.


5.9.17 Showing the History of Task Execution

The history of task execution can only be displayed when you are debugging a realtime OS program.

Furthermore, to show the history of task execution, you need to select [Task ID] on the [Option] page of the [Trace conditions] dialog box that is displayed when you choose [Acquisition] from the pop-up menu of the [Trace] window.

To show the history of function execution from the acquired trace information, choose [Show Function Execution History] from the pop-up menu click the [Show Function Execution History] button  in the toolbar.

An upper pane of the window will be displayed. (Initially, this window is blank.)

When you choose [Analyze Execution History] from the pop-up menu that is displayed when you right-click in the upper pane or click the [Analyze Execution History] button  in the toolbar, the emulator shows the history of task execution.

When showing the history of task execution, note that the functions called from within tasks are not displayed in a tree structure. Only the order in which the functions were executed is displayed.

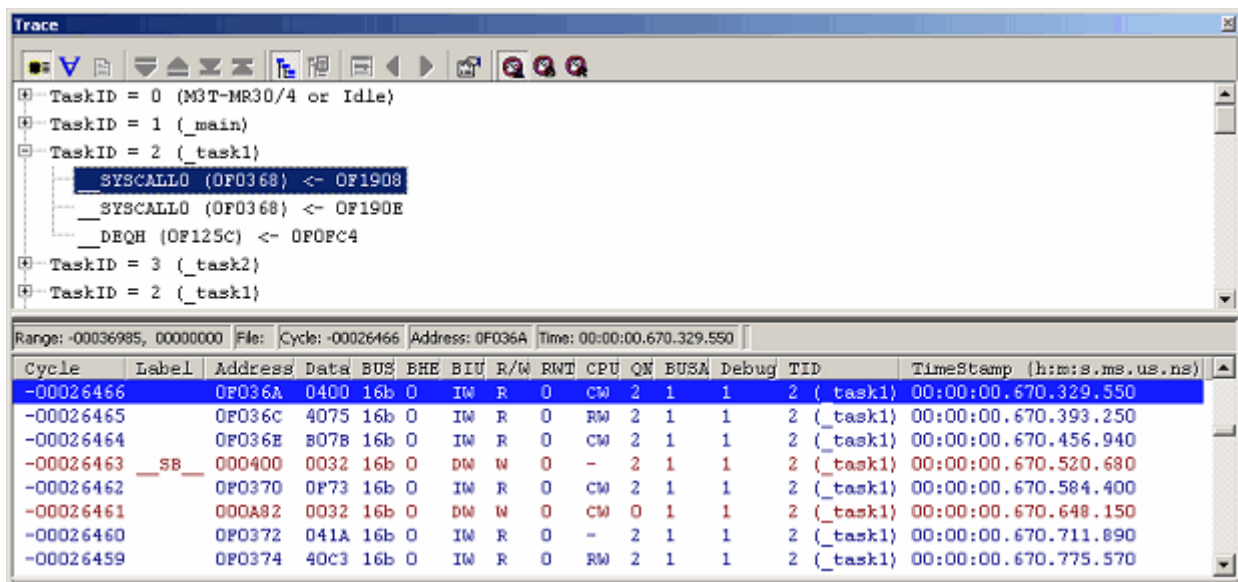


Figure 5.67 [Trace] Window

The lower pane of the window shows the trace result beginning with the cycle in which the task selected in the upper pane was called.

The lower pane of the window can show trace results in disassemble, source or mixed mode.

CAUTION

If trace extraction or deletion conditions are specified, the task execution history cannot be displayed.

If repeat (free) or repeat (full) mode is specified, the task execution history cannot be displayed.

5.10 Measuring Performance

5.10.1 Measuring Performance


The performance function measures a maximum, minimum, average and total execution time and a pass count in each of up to eight specified sections of the user program and then shows a time ratio relative to the total execution time (Go-Break) numerically as percentage and graphically.

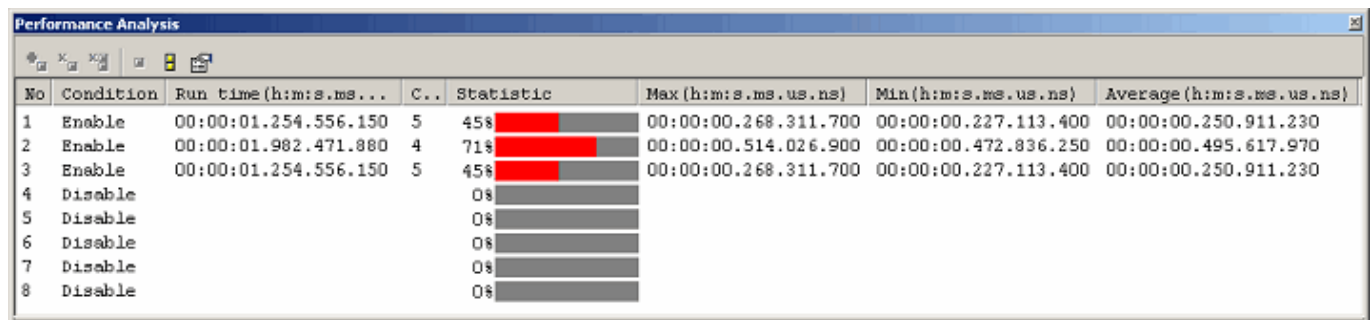
Since the performance function uses the emulator's performance measurement circuit to measure the execution time, it does not obstruct the execution of the user program.

Performance measurement conditions cannot be manipulated during program execution.

5.10.2 Showing the Result of Performance Measurement

Measurement results are displayed in the [Performance Analysis] window.

To open the [Performance Analysis] window, choose [Performance → Performance Analysis] from the [View] menu or click the [Performance Analysis] toolbar button [].



No	Condition	Run time (h:m:s.ms...)	C..	Statistic	Max (h:m:s.ms.us.ns)	Min (h:m:s.ms.us.ns)	Average (h:m:s.ms.us.ns)
1	Enable	00:00:01.254.556.150	5	45%	00:00:00.268.311.700	00:00:00.227.113.400	00:00:00.250.911.230
2	Enable	00:00:01.982.471.880	4	71%	00:00:00.514.026.900	00:00:00.472.836.250	00:00:00.495.617.970
3	Enable	00:00:01.254.556.150	5	45%	00:00:00.268.311.700	00:00:00.227.113.400	00:00:00.250.911.230
4	Disable			0%			
5	Disable			0%			
6	Disable			0%			
7	Disable			0%			
8	Disable			0%			

Figure 5.68 [Performance Analysis] Window

The [Performance Analysis] window shows a ratio of execution time conforming to the conditions you set in the immediately preceding program execution numerically as percentage and graphically.

The unnecessary columns in this window can be hidden.

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

To redisplay any hidden column, select that column from the pop-up menu again.

The contents displayed in this window are listed below.

Table 5.25 Columns and Contents

Column	Description
No	Numbers assigned to 1–8 measurement sections set in the [Performance Analysis Conditions] dialog box. Click [Settings] on the pop-up menu to open the [Performance Analysis Conditions] dialog box.
Condition	Indicated as [Enable] when measurement conditions are set in the [Performance Analysis Conditions] dialog box. Otherwise, indicated as [Disable].
Run time (h:m:s.ms.us.ns)	Cumulative execution time. It shows a cumulative time of measured execution time.
Count	Shows the number of times measured.
Statistic	Shows a ratio of cumulative execution time relative to Go–Break execution time. [Ratio calculation formula] (Cumulative execution time / Go–Break cumulative execution time) * 100
Max (h:m:s.ms.us.ns)	Maximum execution time per measurement performed
Min (h:m:s.ms.us.ns)	Minimum execution time per measurement performed
Average (h:m:s.ms.us.ns)	Average execution time per measurement performed

5.10.3 Setting Performance Measurement Conditions

In the [Performance] window, select a line of the section No. in which you want to set conditions and choose [Set] from the pop-up menu. The [Performance Analysis Conditions] dialog box will be displayed.

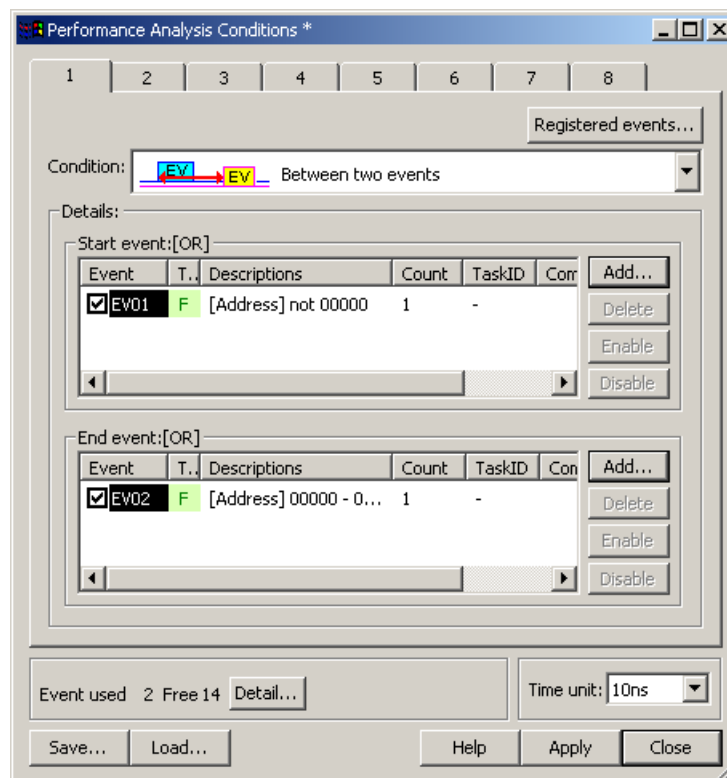


Figure 5.69 [Performance Analysis Conditions] Dialog Box

(1) Setting measurement conditions

A measurement condition can be selected from the following four modes. Select one measurement condition for one section. Use events to set a section. Event counts are fixed to 1. Even when an event count is set to other than 1, it is handled as 1.

Table 5.26 Measurement Condition Modes



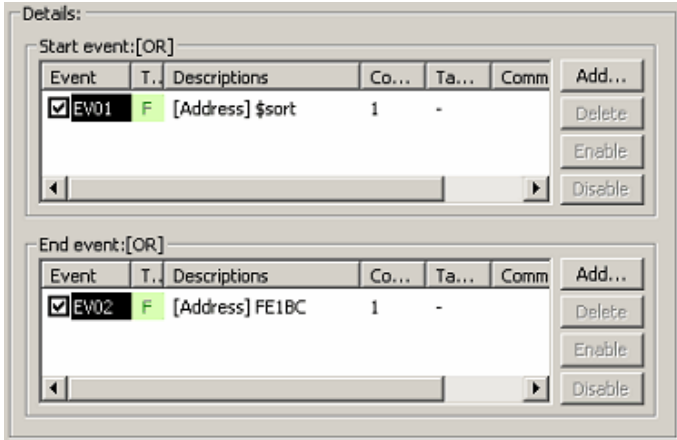

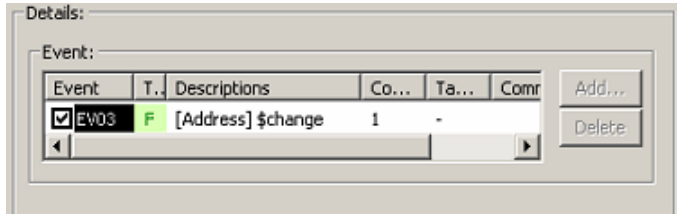
	<p>[Disabled] Not measured.</p>
	<p>[Between two events]</p> <div style="text-align: center;">  </div> <p style="text-align: center;">Figure 5.70 Between Two Events</p> <p>Measurement is taken of time from when a start event occurs to when an end event occurs.</p> <p>Specifically, measurement is taken of an execution time and execution count in the range set by a start event and an end event. The measurement of time starts when a start event occurs and is aborted when an end event occurs. The execution count is incremented by one each time a start event and an end event occur in pairs within the set range.</p> <p>Start event: One or multiple events can be set. End event: One or multiple events can be set.</p>
	<p>[Event cycle counting]</p> <div style="text-align: center;">  </div> <p style="text-align: center;">Figure 5.71 Event Cycle Counting</p> <p>Measurement is taken of periods in which an event occurs.</p> <p>Namely, measurement is taken of an event occurrence period and execution count. The time from when an event occurs to when the next event occurs is measured as one instance of measurement. The execution count is incremented by one each time an event occurs.</p> <p>Event: Only one event point can be set.</p>

Table 5.27 Measurement Condition Modes (Continued)

EV

[Interrupt-disabled range between two events]

Details:

Start event:[OR]

Event	T.	Descriptions	Co...	Ta...	Comm	Add...
<input checked="" type="checkbox"/> EV04	F	[Address] \$sort	1	-		Add... Delete Enable Disable

End event:[OR]

Event	T.	Descriptions	Co...	Ta...	Comm	Add...
<input checked="" type="checkbox"/> EV05	F	[Address] FE1BC	1	-		Add... Delete Enable Disable

Figure 5.72 Interrupt-Disabled Range between Two Events

Measurement is taken of an interrupt disabled section from when a start event occurs to when an end event occurs.

Specifically, measurement is taken of an interrupt disabled time and an interrupt disabled count within the range set by a start event and an end event. The measurement of time starts at the same time an interrupt is disabled and is aborted at the same time the interrupt is reenabled. The count is incremented by one each time an interrupt is disabled.

Start event: One or multiple events can be set.

End event: One or multiple events can be set.

[CAUTION]

To measure an execution time of a function (maximum, minimum or average execution time of a function), use [Between two events].

Set a fetch to the beginning address of the function as a start event and a fetch to the exit of the function (where return statement is written) as an end event. If there are more than one exit, set fetch conditions as an end event for each exit.

(2) Selecting the measurement interval

This setting is applied in common to all of 8 sections. The measurement interval can be selected from the following options:

10 ns (default), 20 ns, 40 ns, 80 ns, 160 ns, 1.6 μ s

The maximum measurement time varies with the measurement interval you set.

5.10.4 Starting Performance Measurement

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

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

When the user program is rerun without changing measurement conditions after being halted, the measured time in this instance is added to the previously measured value.

To perform a measurement over again, clear the measurement result before running the program.

5.10.5 Clearing Performance Measurement Conditions

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

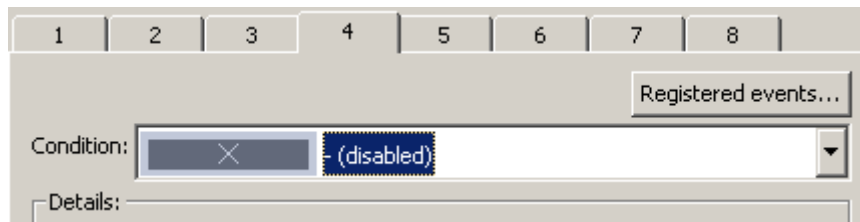


Figure 5.73 [Performance Analysis Conditions] Dialog Box

5.10.6 Clearing the Performance Measurement Result

Select the section you want to clear in the [Performance Analysis] window and then choose [Clear Data] from the pop-up menu. The measurement result of the selected section will be cleared. To clear all measurement results, choose [Clear All Data] from the pop-up menu.

5.10.7 About the Maximum Measurement Time of Performance

(1) Maximum measurement time

The timer used for performance measurement is comprised of a 40-bit counter.

The maximum measurement time varies with the measurement interval selected.

To select the measurement interval, use the [Measurement Unit] list box of the [Performance Analysis Conditions] dialog box.

The measurable maximum times are listed in the table below.

Table 5.28 Measurable Maximum Time

No.	Resolution	Measurable Maximum Time
1	10 ns	Approx. 3 hours, 03 minutes, 15 seconds
2	20 ns	Approx. 6 hours, 06 minutes, 30 seconds
3	40 ns	Approx. 12 hours, 13 minutes, 00 seconds
4	80 ns	Approx. 24 hours, 26 minutes, 00 seconds
5	160 ns	Approx. 48 hours, 52 minutes, 01 seconds
6	1.6 μ s	Approx. 488 hours, 40 minutes, 18 seconds

CAUTION

Note that performance measurement produces an error equal to ± 1 resolution (when resolution = 20 ns, ± 20 ns).

(2) Maximum measurement count

Execution counts are measured using a 32-bit counter. Measurement can be taken of up to a count of 4,294,967,295.

5.11 Acquiring Code Coverage

5.11.1 Acquiring Code Coverage

Code coverage is the function to indicate the 'digestion' degree of test, i.e., "to what degree tests have been carried out on software code (pass)."

Instruction execution information is displayed at C/C++ and assembly-language levels.

This function allows the emulator to acquire instruction execution information from a program without causing it to break. Therefore, the realtime operation of the user program will not be affected.

The coverage result is updated upon a break.

The E100 emulator supports C0 (instruction) coverage and C1 (branch) coverage.


Table 5.29 Code Coverage Definition

C0: Instruction coverage	All statements within the code are executed at least once.
C1: Branch coverage	All branches within the code are executed at least once.

The E100 emulator comes with up to a 2-Mbyte code coverage memory when using the C0 + C1 level coverage, and up to a 1-Mbyte code coverage memory when using the C1 level coverage.

With initial settings, the code coverage memory is allocated automatically to addresses in the ROM and RAM areas in this order.

5.11.2 Opening the [Code Coverage] Window

Choose [Code -> Code Coverage] from the [View] menu or click the [Code Coverage] toolbar button .

The [Code Coverage] window is initially empty.

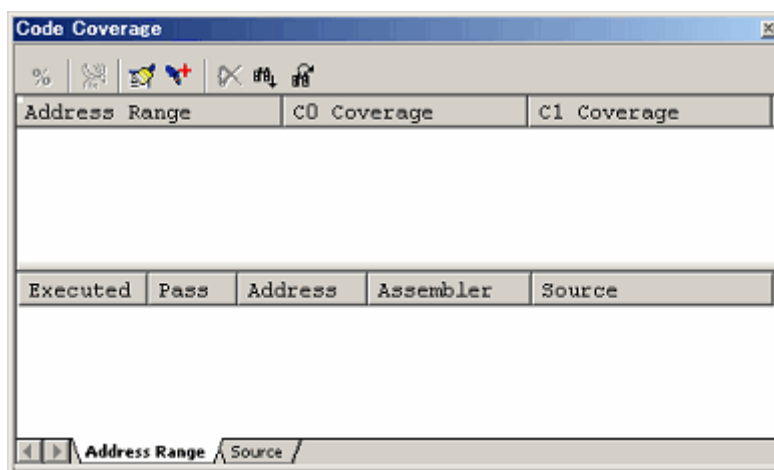


Figure 5.74 [Code Coverage] Window

(1) Measurement method

The [Code Coverage] window consists of two sheets.

Table 5.30 Sheets of the [Code Coverage] Window

Sheet Name	Description
Address Range	Measurement is performed on any address range.
Source	Measurement is performed on a specified source file

The respective sheets permit multiple ranges to be registered.

Up to two instances of the [Code Coverage] window can be opened at the same time.

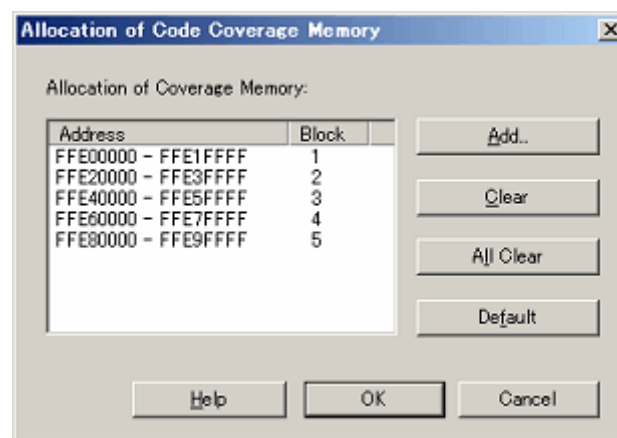
5.11.3 Allocating Code Coverage Memory (Hardware Resource)

(1) Memory allocation

Before code coverage can be measured, code coverage memory must be allocated to the addresses at which to be measured. Coverage data can be obtained from only the address range that has had memory allocated.

To allocate code coverage memory, use the [Allocation of Code Coverage Memory] dialog box.

To open it, choose [Hardware Settings] from the pop-up menu of the [Code Coverage] window.

**Figure 5.75 [Allocation of Code Coverage Memory] Dialog Box**

When using the C0 level coverage and C1 level coverage, you can specify any of 1–8 blocks (maximum 2 Mbytes) each beginning with the 256-Kbyte boundary and any of 1–8 blocks (maximum 1 Mbyte) each beginning with the 128-Kbyte boundary as a code coverage measurement area respectively.

Contiguous blocks or noncontiguous blocks, either one, can be set.

With default settings, blocks are automatically allocated to the addresses in the ROM and RAM areas in that order.

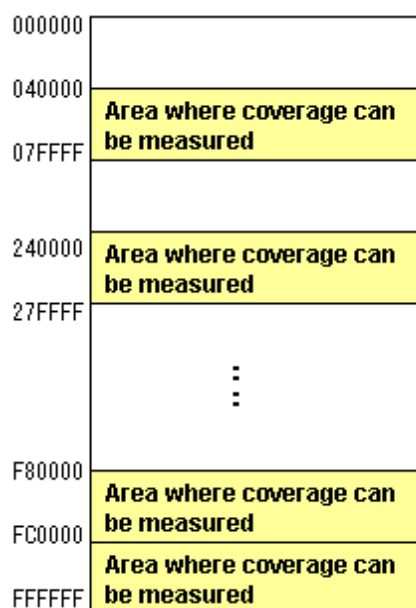


Figure 5.76 Schematic of Coverage Memory Allocation

(2) Changing memory allocation

If coverage memory allocation is changed, the coverage data acquired from the addresses before being changed is retrieved from coverage memory into a coverage-only buffer.

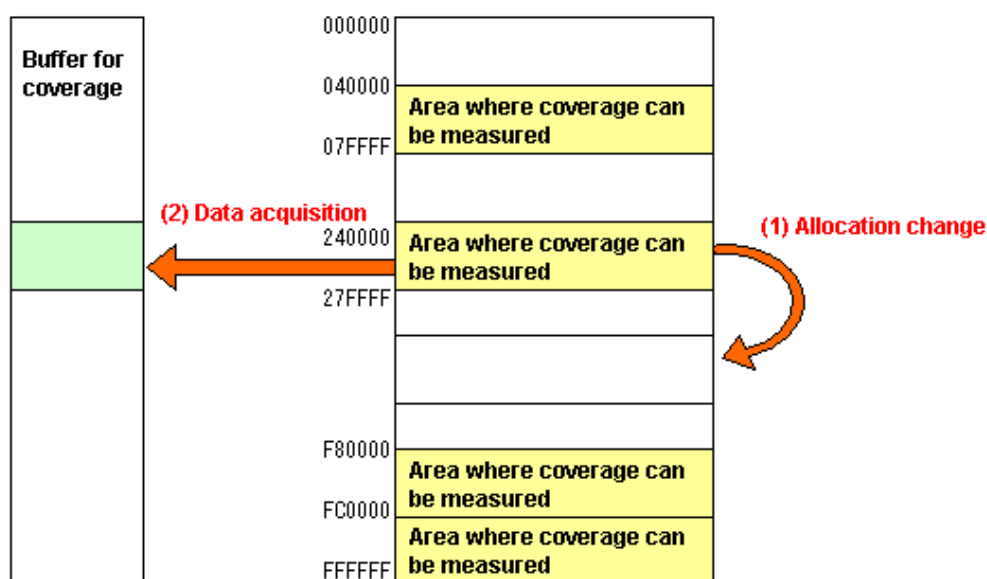


Figure 5.77 Schematic of Coverage Memory Allocation Change

The data accumulated in a coverage-only buffer is retained until the user clears it. However, data is not updated for the areas that have no coverage memory allocated.

The coverage information shown in the [Code Coverage] window includes the content of the coverage-only buffer.

5.11.4 Code Coverage in an Address Range

The [Address Range] sheet shows the code coverage information (C0 coverage and C1 coverage) acquired by the emulator from a user-specified address range.

Multiple address ranges can be registered.

An address range exceeding 2 Mbytes or even an area that has no coverage memory allocated can be specified. However, the coverage information on areas that have no coverage memory allocated is not updated.

Areas where coverage information is not updated are displayed in gray.

An example display is shown below.

The screenshot shows a window titled "Code Coverage" with a toolbar at the top. The main area is divided into two panes. The upper pane contains a table with three columns: "Address Range", "C0 Coverage", and "C1 Coverage". The lower pane contains a table with five columns: "Executed", "Pass", "Address", "Assembler", and "Source".

Address Range	C0 Coverage	C1 Coverage
FFFF85FF - FFFF8705	97.12%	77.78%
FFFF8725 - FFFF8764	76.92%	50.00%

Executed	Pass	Address	Assembler	Source
1	-	FFFF860C	CMP	...
1	T	FFFF860E	BGT.B	...
0	-	FFFF8610	BRA.W	...
1	-	FFFF8613	MOV.L	... for(k=0; k...
1	-	FFFF8615	MOV.L	...
1	-	FFFF8618	MOV.L	...

Figure 5.78 [Code Coverage] Window (Address Specification)

The [Code Coverage] window is vertically divided into two by the splitter.

The upper pane shows the address ranges to be measured, C0 coverage and C1 coverage.

Table 5.31 Contents Shown in the Upper Pane of the [Code Coverage] Window

[Address Range]	Address range in which coverage is measured
[C0 Coverage]	C0 coverage in percentage and graph
[C1 Coverage]	C1 coverage in percentage and graph

The lower pane shows detailed information of the address range selected in the upper pane (assembly-language level).

Table 5.32 Contents Shown in the Lower Pane of the [Code Coverage] Window

[Executed]	1: The instruction was executed 0: The instruction was not executed
[Pass]	Condition for execution of a conditional branch instruction T: The condition was satisfied. F: The condition was not satisfied. T/F: The condition was satisfied in one case and not satisfied in another. -: Instructions other than conditional branch instruction.
[Address]	Instruction address
[Assembler]	Disassembled program
[Source]	C/C++ or assembler source program

The acquired coverage information is accumulated in memory until the user clears it.

When you double click assembler code shown in the [Address Range] sheet, the corresponding source code is shown in the [Editor] window.

Be sure that source code is not displayed in the cases listed below.

- A source file that corresponds the assembler line does not exist.
- A source line that corresponds the assembler line does not exist.
- Where no debug information is included, such as where the assembler line is a library.

5.11.5 Adding Address Ranges

Follow the procedure described below to add address ranges.

(1) From the [Address Range] sheet of the [Code Coverage] window

1. Right-click in the upper pane of the [Address Range] sheet and choose [Add Range] from the pop-up menu.

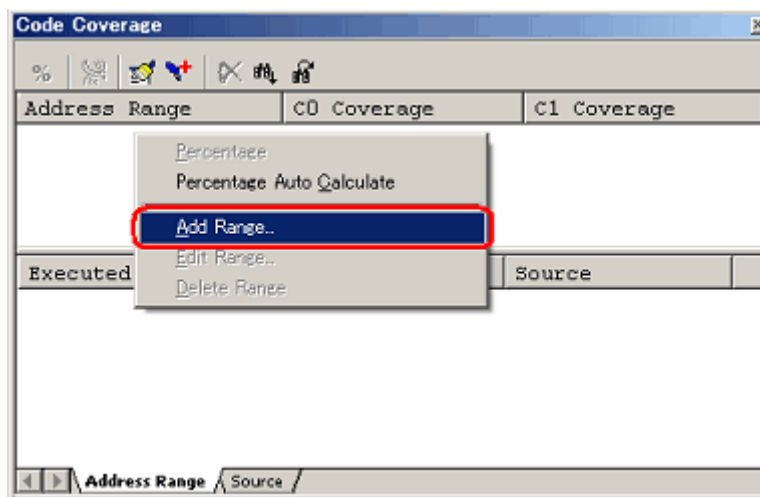


Figure 5.79 [Code Coverage] Window

2. In the [Add Address Range] dialog box that is displayed, enter an address range.

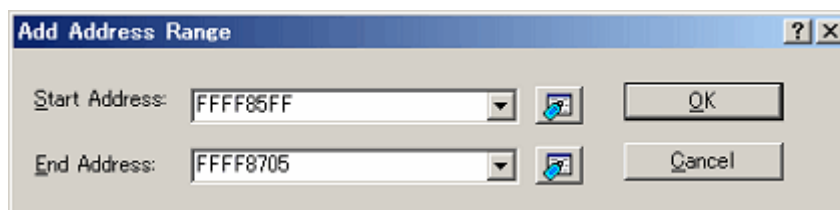


Figure 5.80 [Add Address Range] Dialog Box

3. The address range you have added will be displayed in the upper pane of the [Code Coverage] window.

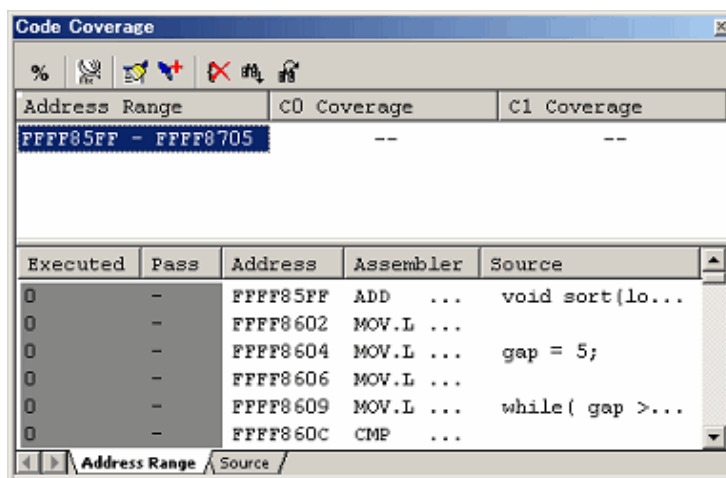


Figure 5.81 [Code Coverage] Window

5.11.6 Changing Address Ranges

Follow the procedure described below to change address ranges.

(1) From the [Address Range] sheet of the [Code Coverage] window

1. Select an address range you want to change in the [Address Range] sheet and while holding it selected, choose [Edit Range] from the pop-up menu.

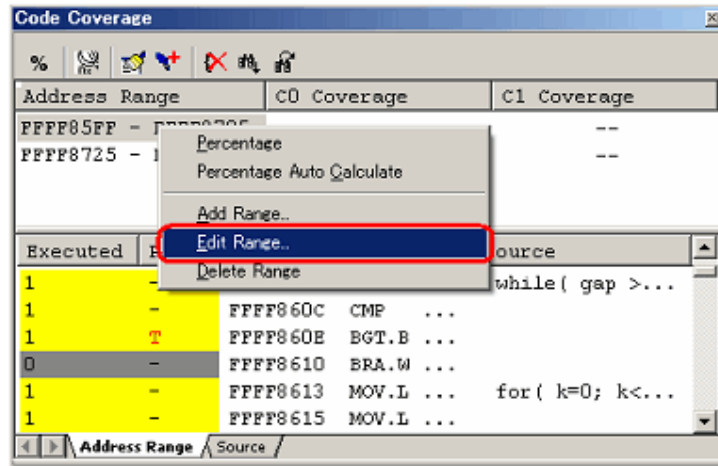


Figure 5.82 [Code Coverage] Window

2. In the [Edit Address Range] dialog box that is displayed, change the address range.

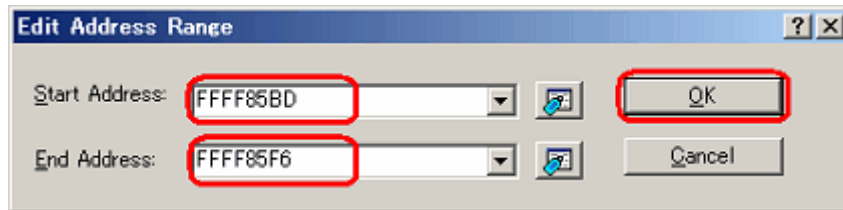


Figure 5.83 [Edit Address Range] Dialog Box

3. The address range you have changed will be displayed in the upper pane of the [Code Coverage] window.

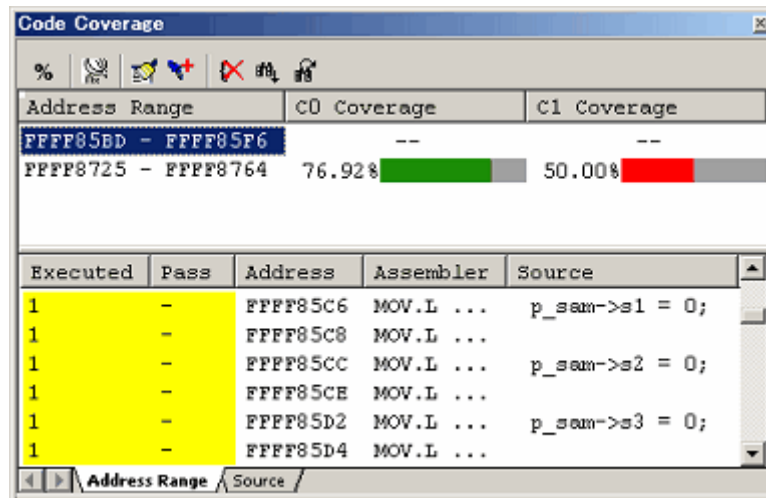


Figure 5.84 [Code Coverage] Window

5.11.7 Removing Address Ranges

Follow the procedure described below to remove address ranges.

(1) From the [Address Range] sheet of the [Code Coverage] window

1. Select an address range you want to remove in the [Address Range] sheet and while holding it selected, choose [Delete Range] from the pop-up menu.

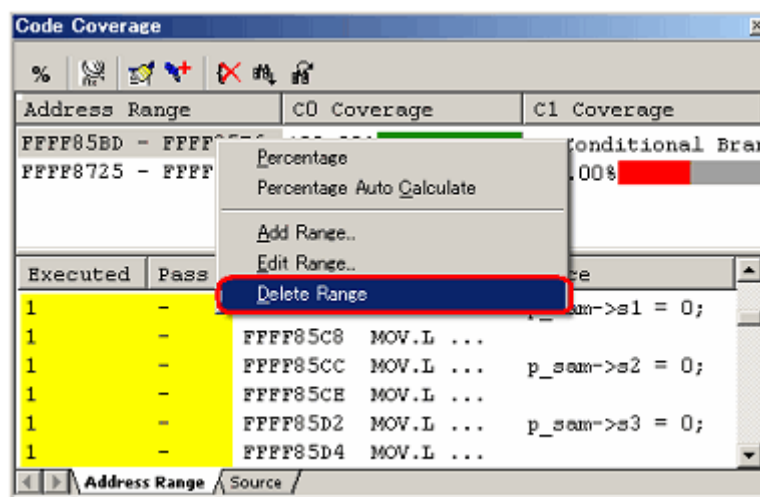


Figure 5.85 [Code Coverage] Window

2. A dialog box asking for your confirmation will be displayed.

Choose to save or not save coverage data. To save, specify a file name and then click the [OK] button. If you do not save, simply click the [OK] button.

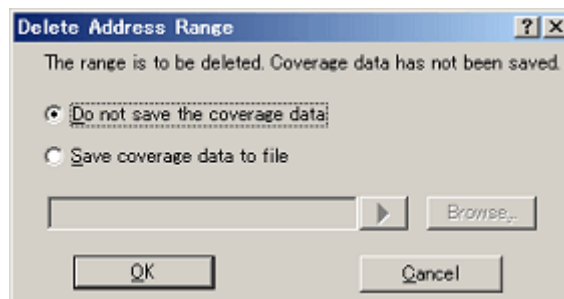


Figure 5.86 [Delete Address Range] Dialog Box

3. The address range you have selected will be removed.

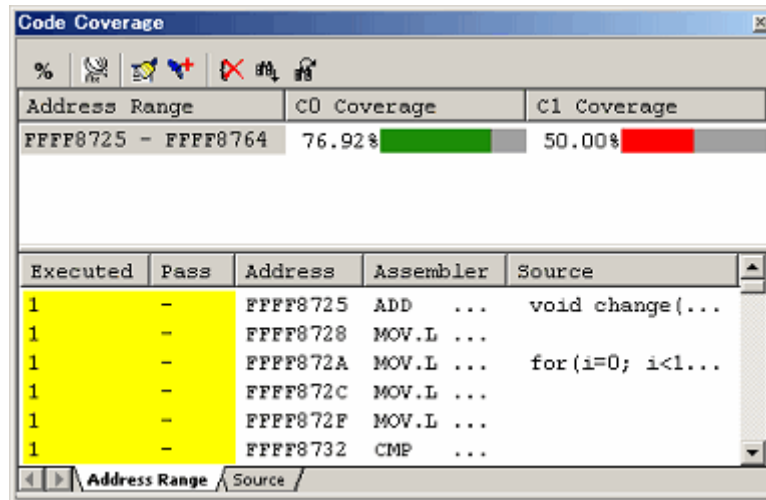


Figure 5.87 [Code Coverage] Window

5.11.8 Code Coverage in a Source File

The [Source] sheet shows the code coverage information (C0 coverage and C1 coverage) acquired by the emulator from a user-specified source file.

Multiple source files can be registered.

This is possible even when the source file memory range exceeds 2 Mbytes or when it includes an area that has no coverage memory allocated.

However, the coverage information on areas that have no coverage memory allocated is not updated.

Address lines where coverage information is not updated are displayed in gray.

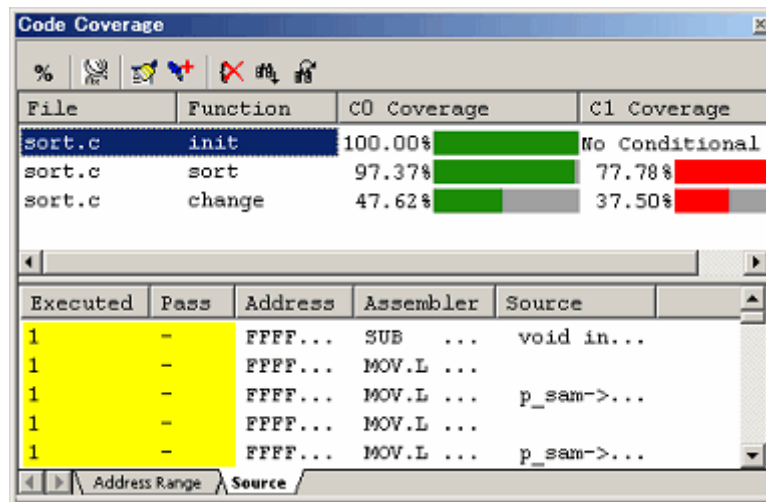


Figure 5.88 [Code Coverage] Window (Source File Specification)

The [Code Coverage] window is vertically divided into two by the splitter.

The upper pane shows the address ranges to be measured (file and function names), C0 coverage and C1 coverage.

Table 5.33 Contents Shown in the Upper Pane of the [Code Coverage] Window

[File]	File name
[Function]	Function name
[C0 Coverage]	C0 coverage in percentage and graph
[C1 Coverage]	C1 coverage in percentage and graph

The lower pane shows detailed information of the address range selected in the upper pane (assembly-language level).

Table 5.34 Contents Shown in the Lower Pane of the [Code Coverage] Window

[Executed]	1: Instructions was executed. 0: Instructions was not executed.
[Pass]	Condition for execution of a conditional branch instruction T: The condition was satisfied. F: The condition was not satisfied. T/F: The condition was satisfied in one case and not satisfied in another.
[Address]	Instruction address
[Assembler]	Disassembled program
[Source]	C/C++ or assembler source program

The acquired coverage information is accumulated in memory until the user clears it.

5.11.9 Adding Source Files

Follow the procedure described below to add source files.

(1) From the [Source] sheet of the [Code Coverage] window

1. Right-click in the upper pane of the [Source] sheet and choose [Add Range] from the pop-up menu.

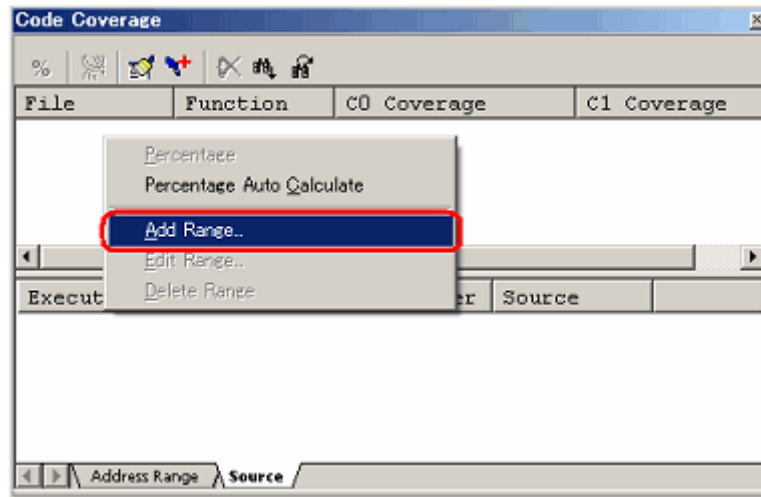


Figure 5.89 [Code Coverage] Window

2. In the [Add Source File] dialog box that is displayed, enter a file name.

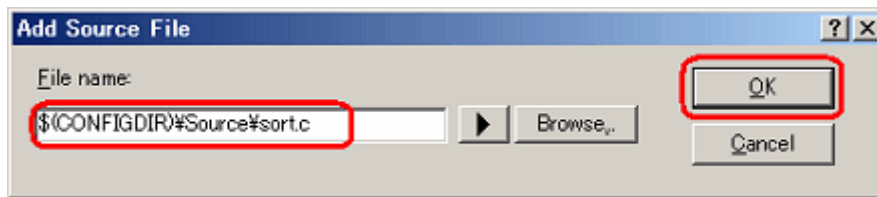


Figure 5.90 [Add Source File] Dialog Box

3. The source file you have added and the function names included in it will be displayed in the upper pane of the [Code Coverage] window.

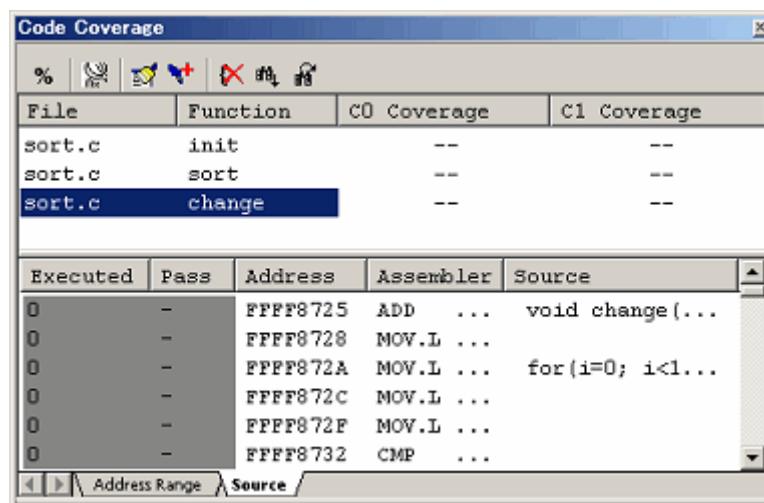


Figure 5.91 [Code Coverage] Window

5.11.10 Removing Source Files

Delete source files by the following methods.

(1) From the [Source] sheet of the [Code Coverage] window

1. Select a function you want to remove in the upper pane of the [Source] sheet and while holding it selected, choose [Delete Range] from the pop-up menu.

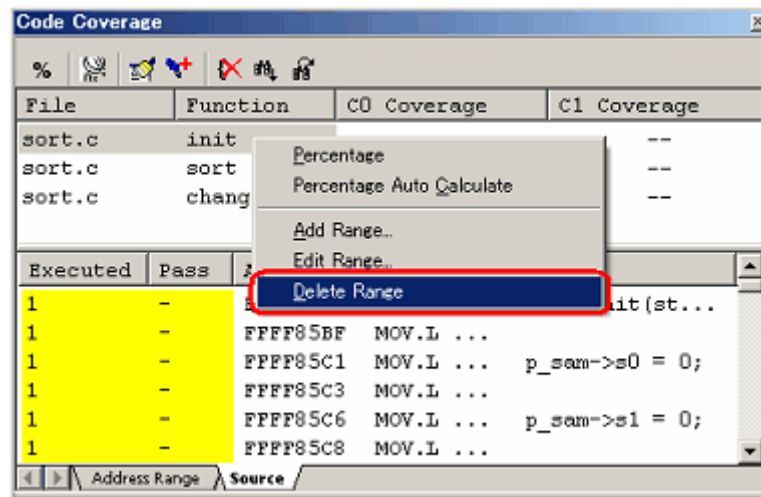


Figure 5.92 [Code Coverage] Window

2. A dialog box prompting for your confirmation will be displayed. Choose to save or not save coverage data. To save, specify a file name and then click the [OK] button. If you do not save, simply click the [OK] button.

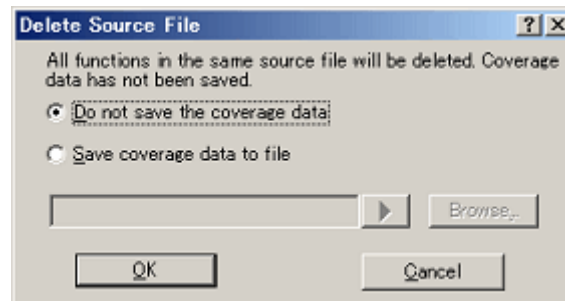


Figure 5.93 [Delete Source File] Dialog Box

- All functions included in the selected source file will be removed.

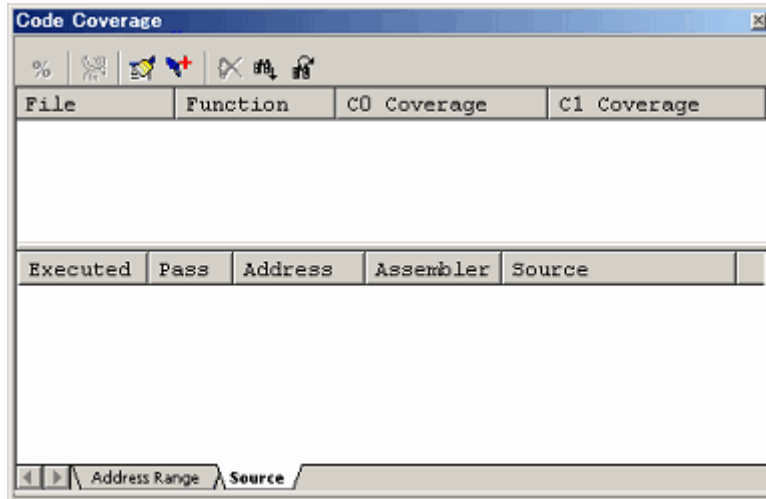


Figure 5.94 [Code Coverage] Window

5.11.11 Showing Percentages and Graphs

After the program has stopped, right-click in the upper pane of the [Code Coverage] window and choose [Percentage] from the pop-up menu. The emulator will start calculating C0 (instruction) coverage and C1 (branch) coverage for each address range.

When the calculation is completed, coverage information is displayed in the upper pane as percentages and graphs.

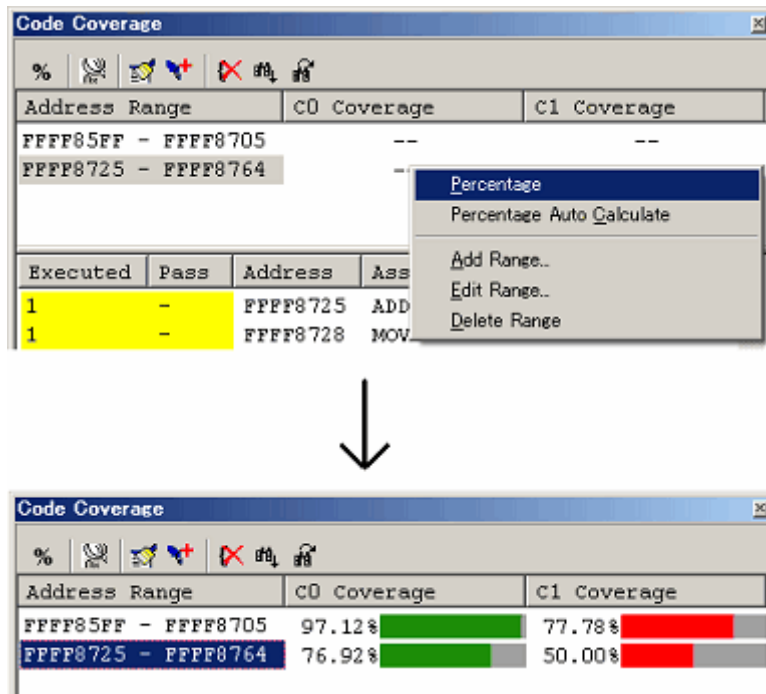


Figure 5.95 [Code Coverage] Window

5.11.12 Using the Sort Function

Clicking on a header column in the upper pane of the [Code Coverage] window allows the coverage data to be sorted.

(1) Clicking on the [File] column

The data can be sorted by the file name. Lines in the same file are sorted by function name.

Example:

File	Function	C0 Coverage
file1.cpp	func1	40% ■■■■
file1.cpp	func2	10% ■
file1.cpp	func3	80% ■■■■■■■■
file1.cpp	func4	70% ■■■■■■■■
file2.cpp	func1	20% ■■
file2.cpp	func2	60% ■■■■■■
file2.cpp	func3	90% ■■■■■■■■
file3.cpp	func1	0%
file3.cpp	func2	30% ■■■
file3.cpp	func3	10% ■

(2) Clicking on the [C0 Coverage] column

The data can be sorted by coverage rate.

First clicking on the column sorts the values in decreasing order. Clicking on the column again sorts the values in ascending order.

Example:

File	Function	C0 Coverage
file2.cpp	func3	90% ■■■■■■■■
file1.cpp	func3	80% ■■■■■■■■
file1.cpp	func4	70% ■■■■■■■■
file2.cpp	func2	60% ■■■■■■
file1.cpp	func1	40% ■■■■
file3.cpp	func2	30% ■■■
file2.cpp	func1	20% ■■
file1.cpp	func2	10% ■

file3.cpp func3 10% ■

file3.cpp func1 0%

(3) Clicking on the [C0 Coverage] and [File] columns in this order


The data for each file is sorted by the coverage rate in descending order.

Example:

File	Function	C0 Coverage

file1.cpp	func3	80% ■■■■■■■■
file1.cpp	func4	70% ■■■■■■■■
file1.cpp	func1	40% ■■■■
file1.cpp	func2	10% ■
file2.cpp	func3	90% ■■■■■■■■
file2.cpp	func2	60% ■■■■■■
file2.cpp	func1	20% ■■
file3.cpp	func2	30% ■■■
file3.cpp	func3	10% ■
file3.cpp	func1	0%

5.11.13 Searching for Unexecuted Lines

Search for unexecuted lines in a selected address range or function. When you click the Find button  in the toolbar, the [Find] dialog box shown below appears.

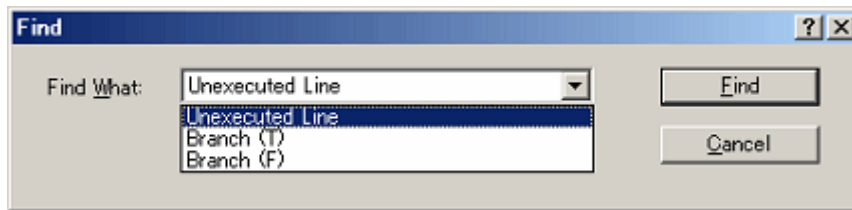



Figure 5.96 [Find] Dialog Box

Following three search options are available.

Table 5.35 Search Options

Unexecuted Line	Instructions not executed yet
Branch (T)	Branch instructions only tested as TRUE.
Branch (F)	Branch instructions only tested as FALSE.

Clicking the [Find] button starts a search.

Clicking the [Find Next] button () in the toolbar resumes the search under the same conditions.

When a matching instruction is found, the line of that instruction is highlighted.

When no matching instructions are found, a message is displayed.

5.11.14 Clearing Code Coverage Information

(1) Clearing the code coverage information of the specified range

Selecting [Clear Coverage Range] from the pop-up menu opens the [Clear Address Range] dialog box.

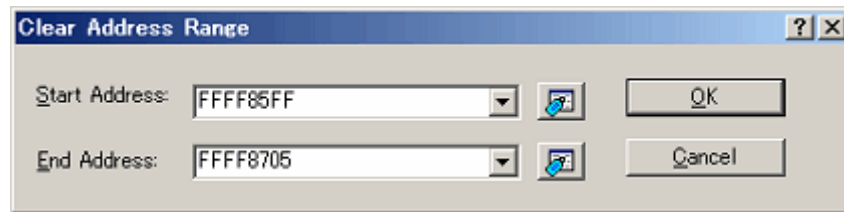


Figure 5.97 [Clear Address Range] Dialog Box

Enter the start and end addresses of the range to be cleared. Clicking the [OK] button clears the coverage information of the selected range.

(2) Clearing all the code coverage information

Selecting [Clear the Entire Coverage] from the pop-up menu clears all the code coverage information.

5.11.15 Updating Coverage Information

Selecting [Refresh] from the pop-up menu updates the content of the [Code Coverage] window.

If [Lock Refresh] has been selected, the information is not automatically updated when the program breaks. To view the latest information, therefore, you need to update manually.

5.11.16 Preventing Update of Coverage Information

Selecting [Lock Refresh] from the pop-up menu prevents update of the [Code Coverage] window while the user program execution is stopped.

5.11.17 Saving the Code Coverage Information to a File

You can save the code coverage information of the currently selected sheet to a file.

Selecting [Save Data] from the pop-up menu opens the [Save Coverage Data] dialog box.

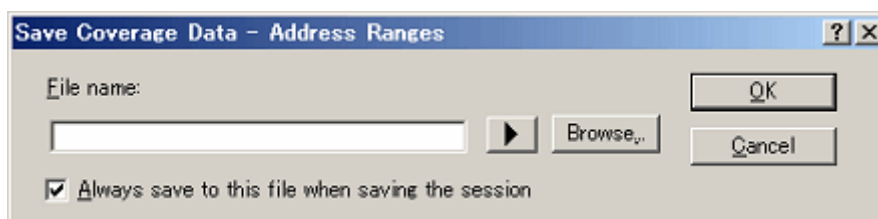


Figure 5.98 [Save Coverage Data] Dialog Box

Enter a file name in which you want the information to be saved.

If the file extension is omitted, “.cov” will automatically be added as the file extension.

If you specify an existing file name, the file is overwritten.

5.11.18 Loading Code Coverage Information from a File

You can load a code coverage information file.

Selecting [Load Data] from the pop-up menu opens the [Load Coverage Data] dialog box.

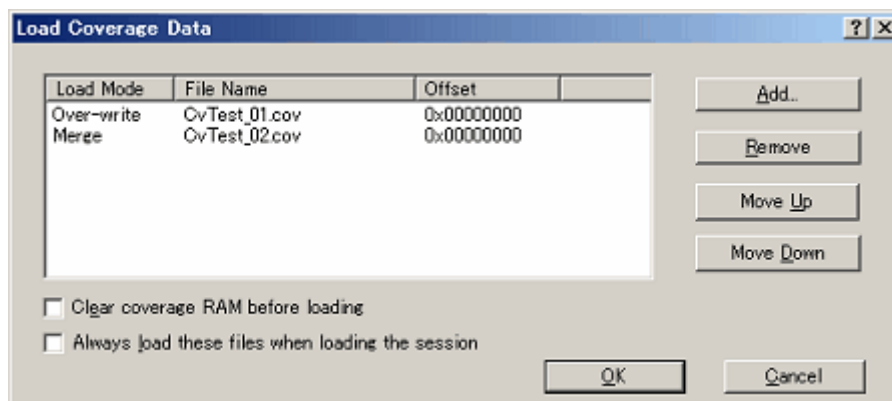


Figure 5.99 [Load Coverage Data] Dialog Box

Clicking on the [Add] button opens the [Add Coverage Files] dialog box shown below.

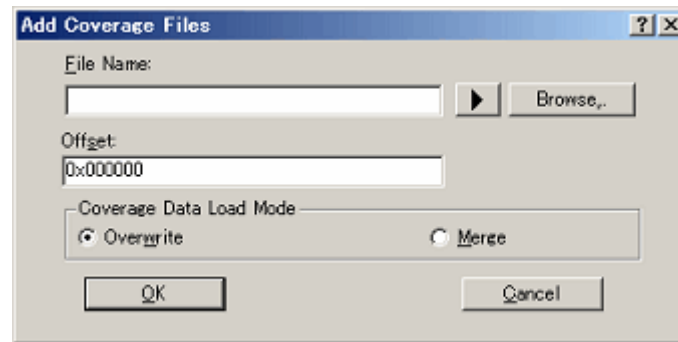


Figure 5.100 [Add Coverage Files] Dialog Box

Use this dialog box to specify a coverage information file you want to load. You can also specify a load mode and offset for each file you load.

The only file extension available is “.cov”. An error message will appear if any other file extension is entered.

The files you add will be listed in the [Load Coverage Data] dialog box. The files will be loaded in the order in which they are listed. If necessary, use the [Move Up] or [Move Down] button to change the order.

CAUTION

If the coverage information file you are loading is of a source file type, you cannot specify an offset.

5.11.19 Coverage Information File Load Modes

Coverage information file load modes are schematically shown below.

(1) When [Overwrite] has been selected

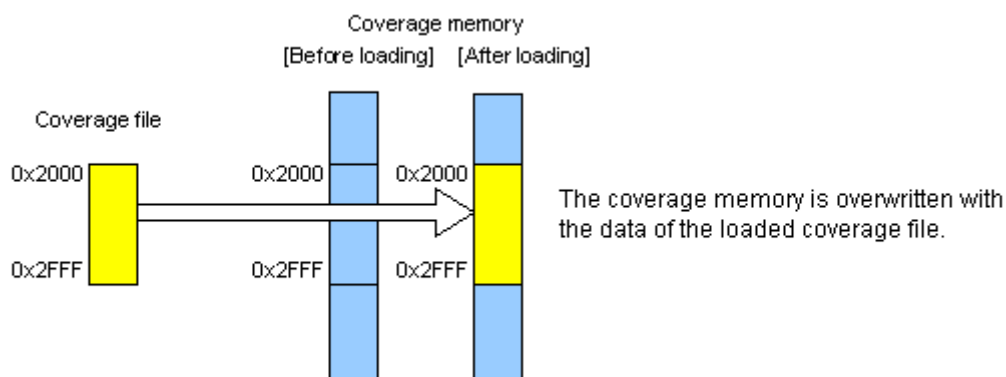


Figure 5.101 Schematic of the Overwrite Mode

(2) When [Merge] has been selected

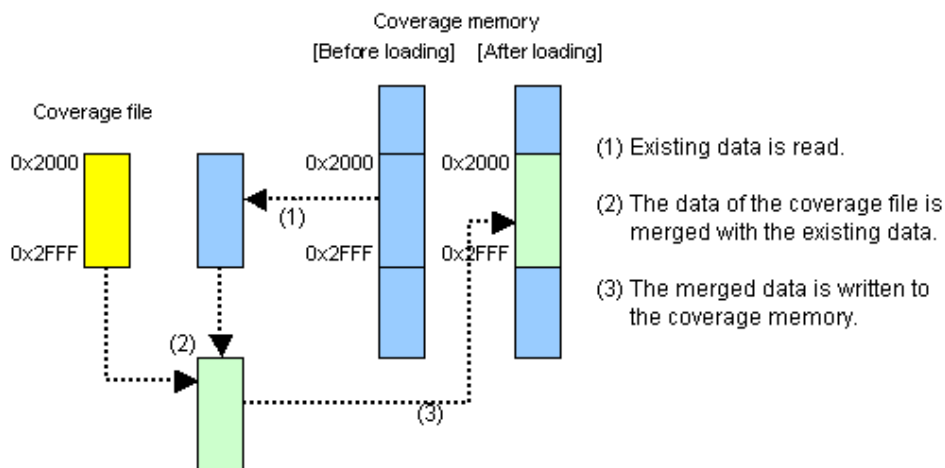


Figure 5.102 Schematic of the Merge Mode

(3) Application example of merge mode

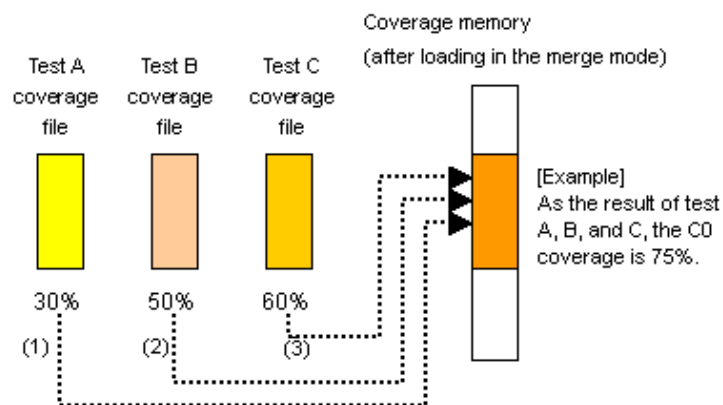


Figure 5.103 Schematic of Merge-Mode Application

[Procedure]

(1) Open the [Load Coverage Data] dialog box.

To begin with, select the [Clear coverage RAM before loading] checkbox.

(2) Add a coverage file for test A in the merge mode.

(3) Add a coverage file for test B in the merge mode.

(4) Add a coverage file for test C in the merge mode.

(5) Click the [OK] button.

You have now finished merging three files.

By calculating percentages in the [Code Coverage] window, you can view the coverage (in percentage) of those tests as a whole.

Furthermore, the merged data can be saved to a file, so that you can manage those data as a single file.

5.11.20 Displaying Code Coverage Information in the [Editor] Window

When the [Editor] window is open in the source mode, the results of coverage are displayed in its [Code Coverage] column.

Part of the [Code Coverage] column that corresponds to a source line where an instruction has been executed is highlighted in yellow.

If the user changes any setting regarding the coverage information in the [Code Coverage] window, the content of the corresponding [Code Coverage] column will also be updated.

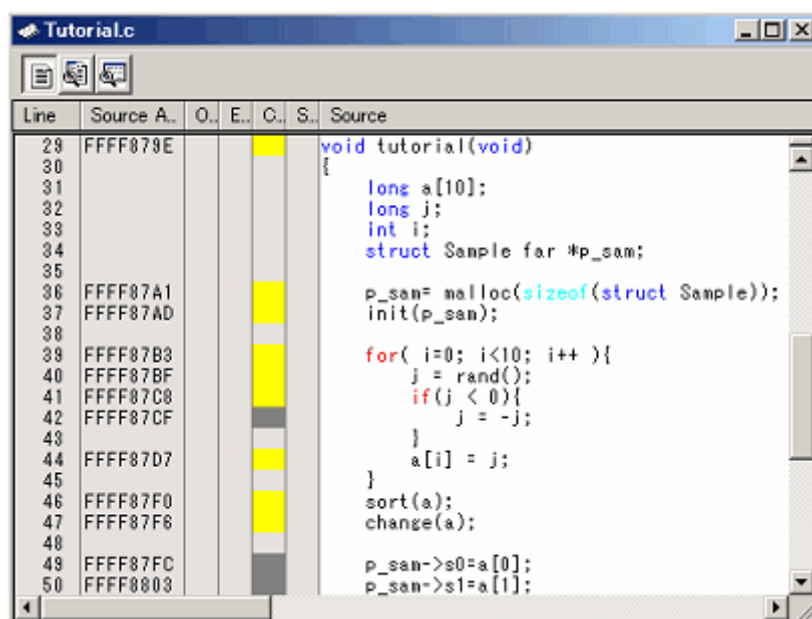


Figure 5.104 Example of Code Coverage Results

5.12 Acquiring Data Coverage

5.12.1 Acquiring Data Coverage

The E100 emulator has code coverage, data coverage and realtime profile functions usable exclusively to each other.

To use the data coverage function, choose [Data Coverage] in the [Exclusive Functions] section on the [System] page of the [Configuration properties] dialog box.

Data coverage is the function to indicate “what kinds of accesses have been made” to the data area.


This function allows the emulator to acquire access information per byte without causing a program to break. Therefore, the realtime operation of the user program will not be affected.

The coverage result is updated upon a break.

The E100 emulator comes with 512 Kbytes of data coverage memory.

With initial settings, the data coverage memory is automatically assigned at addresses in the ROM and RAM areas in this order.

5.12.2 Opening the [Data Coverage] Window

Choose [Code -> Data Coverage] from the [View] menu or click the [Data Coverage] toolbar button .

The [Data Coverage] window is initially empty.

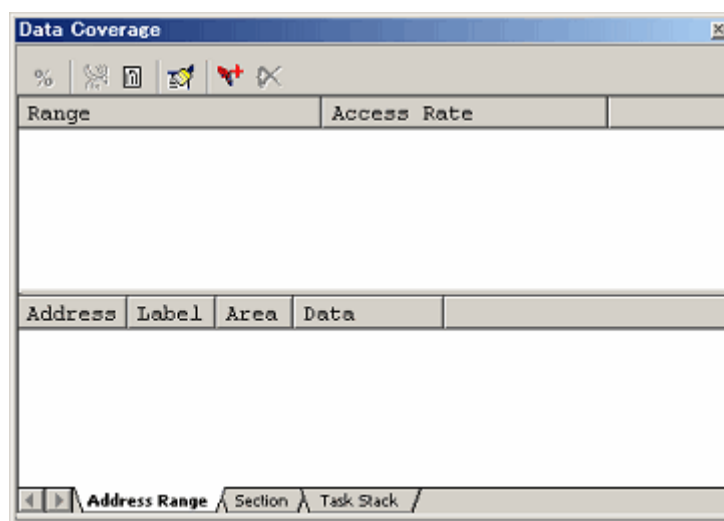


Figure 5.105 [Data Coverage] Window

(1) Measurement method

The [Data Coverage] window consists of three sheets.

Table 5.36 Sheets of the [Data Coverage] Window

Sheet	Description
Address Range	Measurement is performed on any address range.
Section	Measurement is performed on a specified section.
Task Stack	Measurement is made of all task stack areas.

The respective sheets permit multiple ranges to be registered.

The [Task Stack] sheet supports only automatic registration.

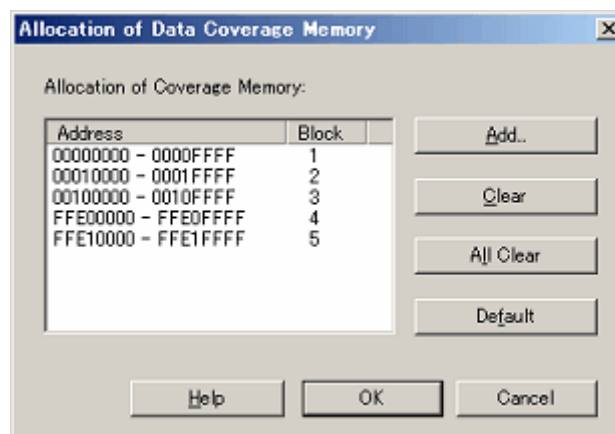
Up to three instances of the [Data Coverage] window can be opened at the same time.

5.12.3 Allocating Data Coverage Memory (Hardware Resource)

(1) Memory allocation

Before data coverage can be measured, data coverage memory must be assigned to the target address range. Coverage data can be obtained from only the address range that has had memory allocated.

To allocate data coverage memory, use the [Allocation of Data Coverage Memory] dialog box. To open this dialog box, select [Hardware Settings...] from the pop-up menu of the [Data Coverage] window.

**Figure 5.106 [Allocation of Data Coverage Memory] Dialog Box**

You can specify any of blocks 1 to 8 (up to 512 Kbytes) each beginning with a 64-Kbyte boundary as data coverage measurement areas.

Either contiguous blocks or non-contiguous blocks can be assigned.

With default settings, blocks are automatically allocated to the addresses in the ROM and RAM areas in that order.

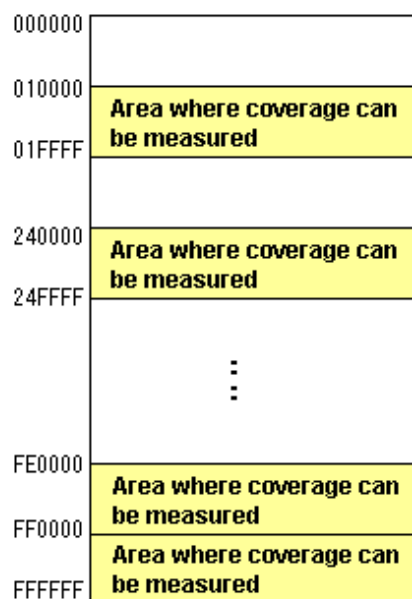


Figure 5.107 Schematic of Data Coverage Memory Allocation

(2) Changing memory allocation

If coverage memory allocation is changed, the coverage data acquired from the addresses before being changed is retrieved from coverage memory into a coverage-only buffer.

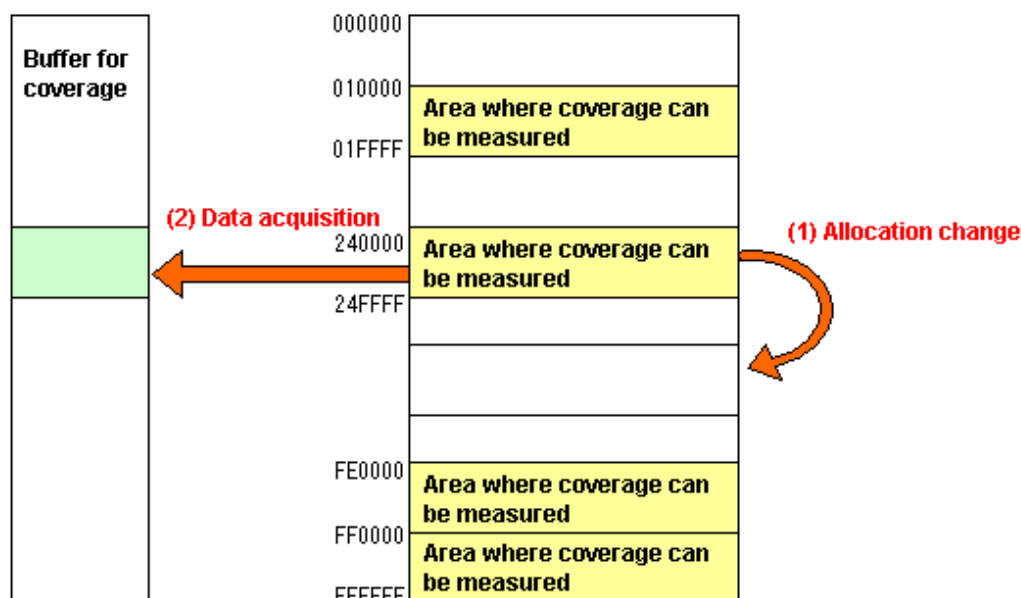


Figure 5.108 Schematic of Data Coverage Memory Allocation Change

The acquired coverage information is accumulated in the coverage buffer until the user clears it. However, the coverage information on areas that have no coverage memory allocated is not updated.

The coverage information shown in the [Data Coverage] window includes the content of the coverage-only buffer.

5.12.4 Data Coverage in an Address Range

The E100 emulator shows the access information it acquired from a user-specified address range.

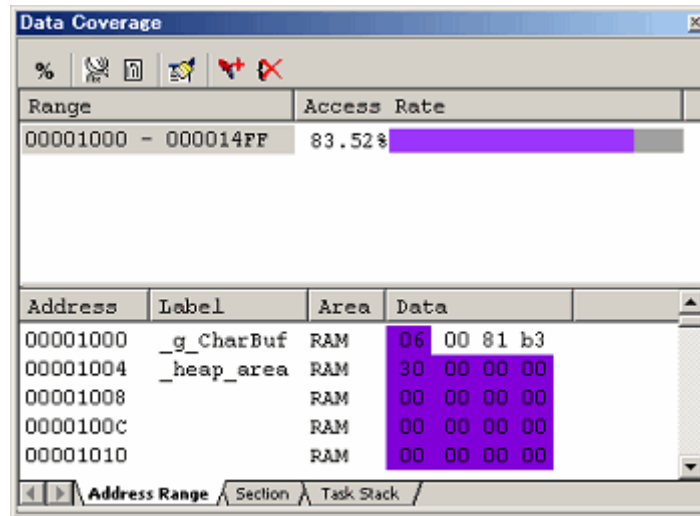


Figure 5.109 [Data Coverage] Window (Address Specification)

The [Data Coverage] window is vertically divided into two by the splitter.

The upper pane shows the address ranges to be measured and access rates.

Table 5.37 Contents Shown in the Upper Pane of the [Data Coverage] Window

[Range]	Address range in which coverage is measured
[Access Rate]	Access rate in percentage and graph

The lower pane shows detailed information of the address range selected in the upper pane.

Table 5.38 Contents Shown in the Lower Pane of the [Data Coverage] Window

[Address]	Address value
[Label]	Label name
[Area]	Memory area (ROM, RAM, IO) This column is blank when the area is unused.
[Data]	Memory data The accessed data has its background displayed in purple.

If located outside the coverage memory allocated area, address lines are displayed in gray. Although the existing coverage information of those addresses is retained, the coverage information will not be updated by program execution.

The acquired coverage information is accumulated in memory until the user clears it.

5.12.5 Adding Address Ranges

Follow the procedure described below to add address ranges.

(1) From the [Address Range] sheet of the [Data Coverage] window

1. Right-click in the upper pane of the [Address Range] sheet and choose [Add Range] from the pop-up menu.

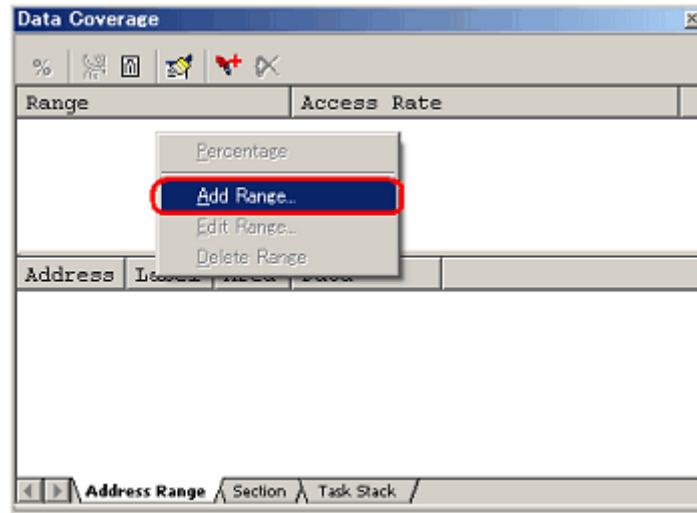


Figure 5.110 [Data Coverage] Window

2. In the [Add Address Ranges] dialog box that is displayed, enter an address range.

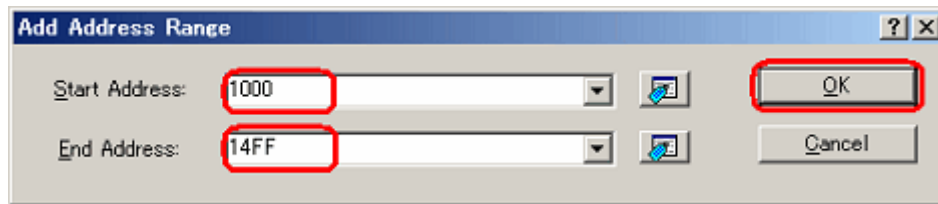


Figure 5.111 [Add Address Range] Dialog Box

3. The address range you have added will be displayed in the upper pane of the [Data Coverage] window.

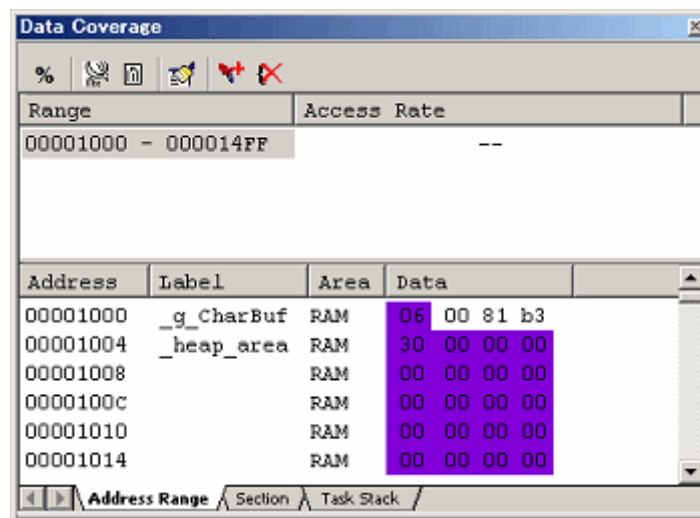


Figure 5.112 [Data Coverage] Window

5.12.6 Changing Address Ranges

Follow the procedure described below to change address ranges.

(1) From the [Address Range] sheet of the [Data Coverage] window

1. Select an address range you want to change in the [Address Range] sheet and while holding it selected, choose [Edit Range] from the pop-up menu.

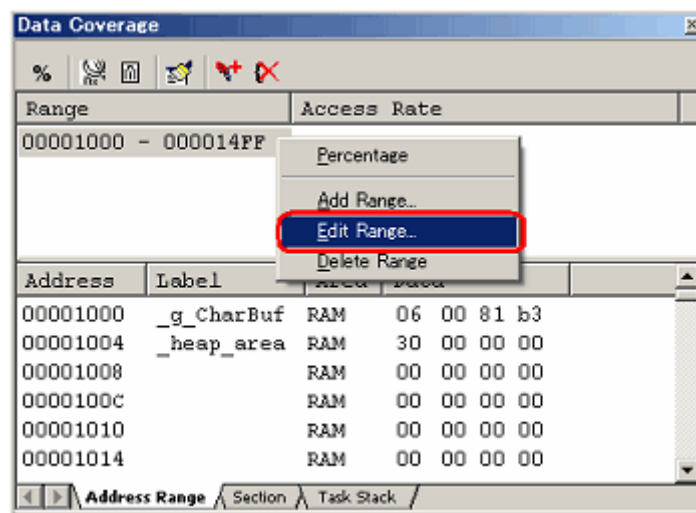


Figure 5.113 [Data Coverage] Window

2. In the [Edit Address Range] dialog box that is displayed, change the address range.

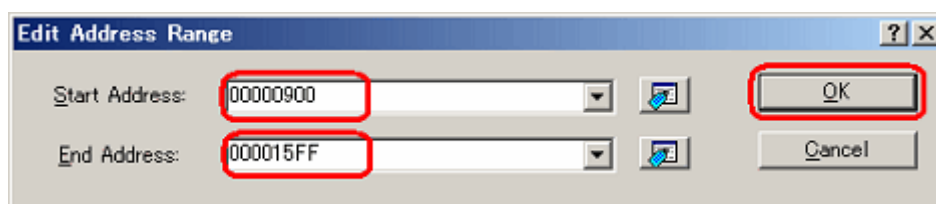


Figure 5.114 [Edit Address Range] Dialog Box

3. The address range you have changed will be displayed in the upper pane of the [Data Coverage] window.

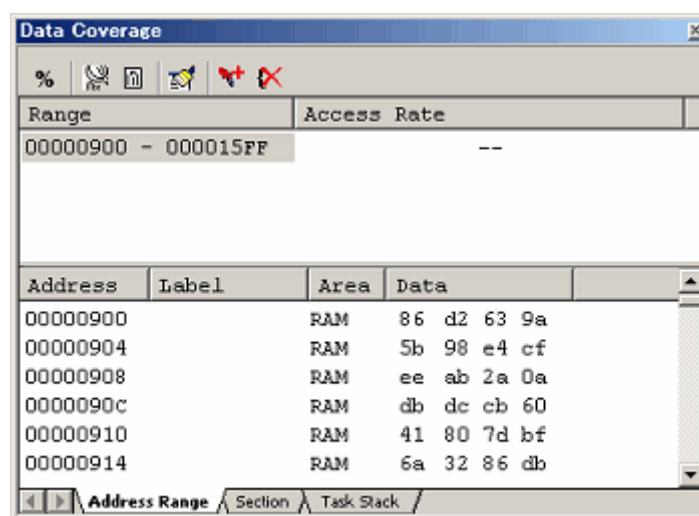


Figure 5.115 [Data Coverage] Window

5.12.7 Removing Address Ranges

Follow the procedure described below to remove address ranges.

(1) From the [Address Range] sheet of the [Data Coverage] window

1. Select an address range you want to remove in the [Address Range] sheet and while holding it selected, choose [Delete Range] from the pop-up menu.

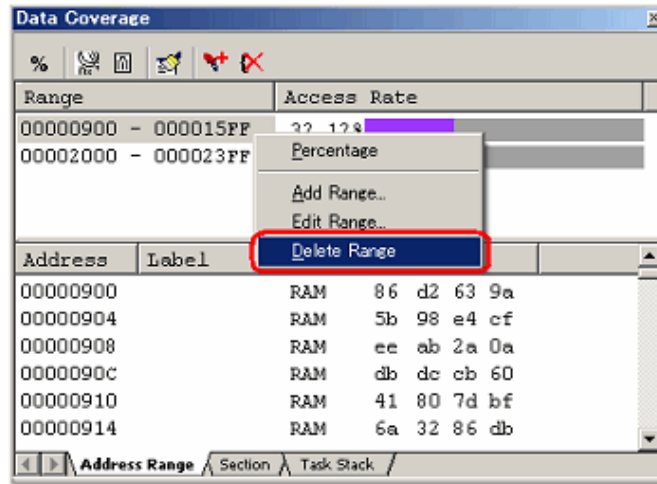


Figure 5.116 [Data Coverage] Window

2. A dialog box prompting for your confirmation will be displayed. Choose to save or not save coverage data. To save, click the [Yes] button. If you do not save, click the [No] button.

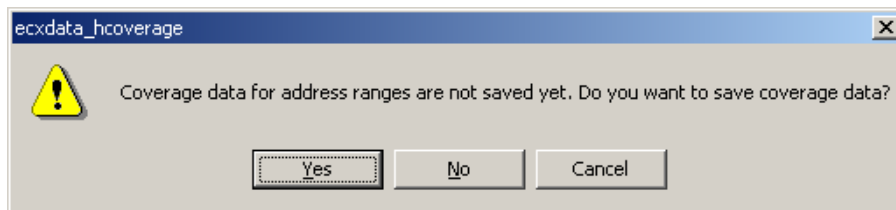


Figure 5.117 Dialog Box for Choosing to Save or Not Save Coverage Data

3. The address range you have selected will be removed.

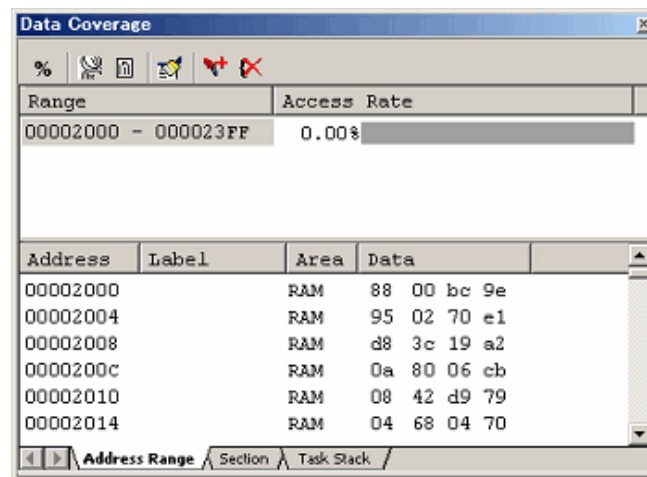


Figure 5.118 [Data Coverage] Window

5.12.8 Data Coverage in a Section

The E100 emulator shows the access information it acquired from a user-specified section.

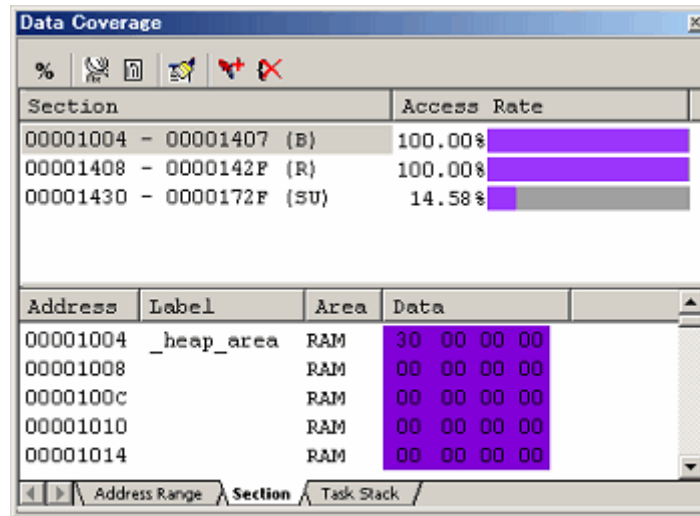


Figure 5.119 [Data Coverage] Window (Section Name Specification)

The [Data Coverage] window is vertically divided into two by the splitter.

The upper pane shows the address ranges (section names) to be measured and access rates.

Table 5.39 Contents Shown in the Upper Pane of the [Data Coverage] Window

[Section]	Address range (section) in which coverage is measured
[Access Rate]	Access rate in percentage and graph

The lower pane shows detailed information of the address range selected in the upper pane.

Table 5.40 Contents Shown in the Lower Pane of the [Data Coverage] Window

[Address]	Address value
[Label]	Label name
[Area]	Memory area (FlashROM, RAM, SFR) This column is blank when the area is unused.
[Data]	Memory data The accessed data has its background displayed in purple.

If located outside the coverage memory allocated area, address lines are displayed in gray. Although the existing coverage information of those addresses is retained, the coverage information will not be updated by program execution.

The acquired coverage information is accumulated in memory until the user clears it.

5.12.9 Adding Sections

Follow the procedure described below to add sections.

(1) From the [Section] sheet of the [Data Coverage] window

1. Right-click in the upper pane of the [Section] sheet and choose [Add Range] from the pop-up menu.

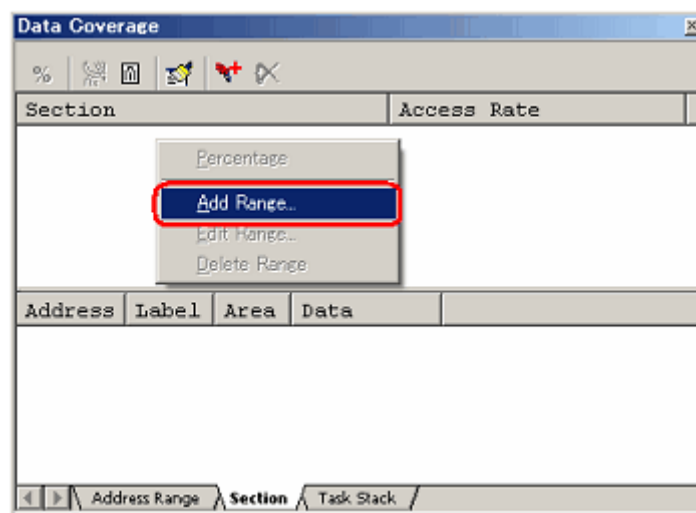


Figure 5.120 [Data Coverage] Window

2. In the [Add A Section] dialog box that is displayed, enter a section name.



Figure 5.121 [Add A Section] Dialog Box

3. The address range (section name) you have added will be displayed in the upper pane of the [Data Coverage] window.

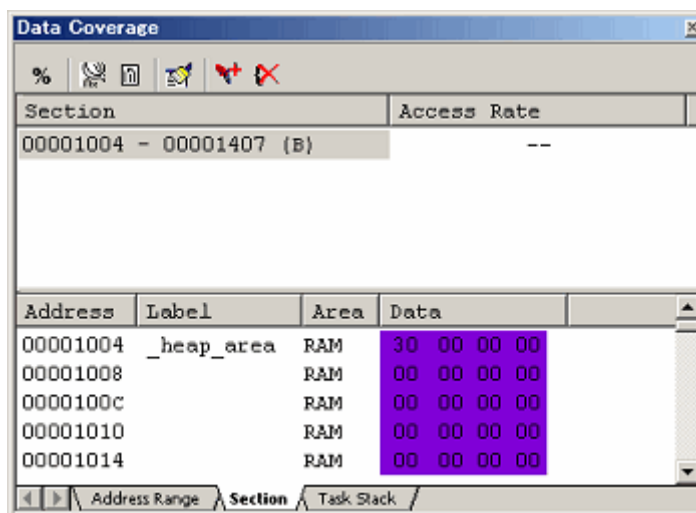


Figure 5.122 [Data Coverage] Window

5.12.10 Removing Sections

Follow the procedure described below to remove sections.

(1) From the [Section] sheet of the [Data Coverage] window

1. Select a section name you want to remove in the [Section] sheet and while holding it selected, choose [Delete Range] from the pop-up menu.

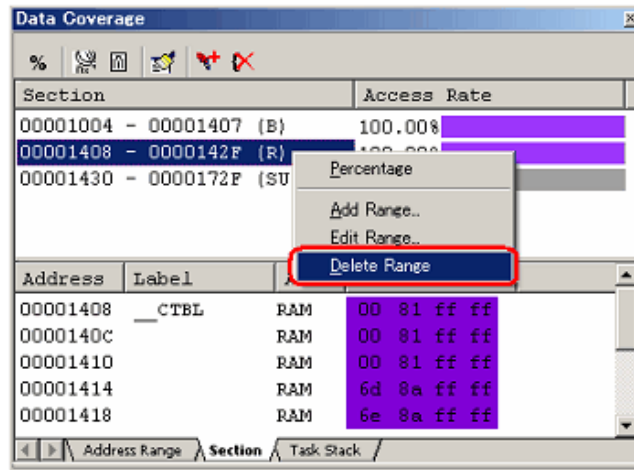


Figure 5.123 [Data Coverage] Window

2. A dialog box prompting for your confirmation will be displayed. Choose to save or not save coverage data. To save, click the [Yes] button and specify a file name. If you do not save, click the [No] button.

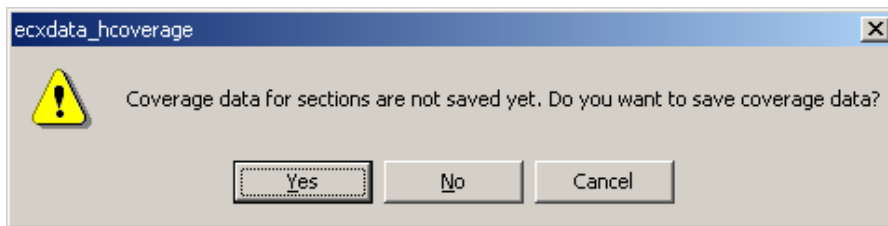


Figure 5.124 Dialog Box for Choosing to Save or Not Save Coverage Data

3. The section name you have selected will be removed.

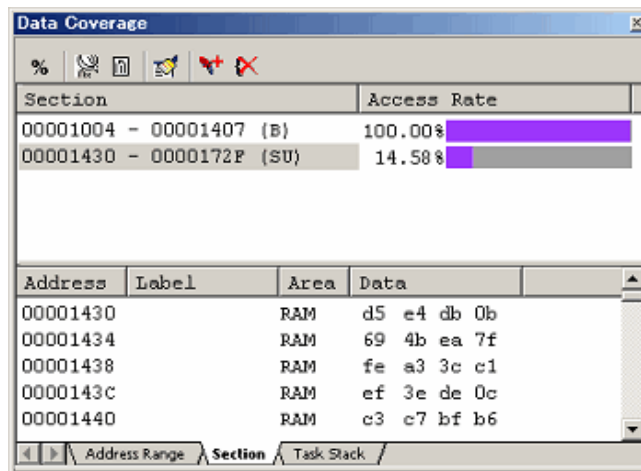


Figure 5.125 [Data Coverage] Window

5.12.11 Data Coverage in a Task Stack

The [Task Stack] sheet shows the access information acquired from a task stack.

Task stacks are automatically registered.

You cannot add, remove or change any task.

If tasks are changed pursuant to alterations of the user program, for example, the window is automatically updated.

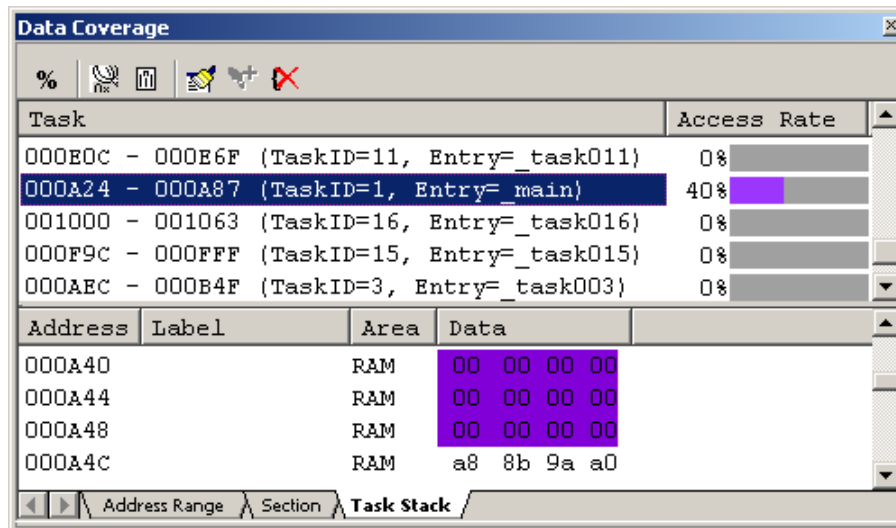


Figure 5.126 [Data Coverage] Window (Task Stack Specification)

The [Data Coverage] window is vertically divided into two by the splitter.

The upper pane shows the automatically registered task stacks and access rates.

Table 5.41 Contents Shown in the Upper Pane of the [Data Coverage] Window

[Task]	Task stacks (task ID, task entry label)
[Access Rate]	Access rate in percentage and graph

The lower pane shows detailed information of the task stack selected in the upper pane.

Table 5.42 Contents Shown in the Lower Pane of the [Data Coverage] Window

[Address]	Address value
[Label]	Label name
[Area]	Memory area (ROM, RAM, IO) This column is blank when the area is unused.
[Data]	Memory data The accessed data has its background displayed in purple.

If located outside the coverage memory allocated area, address lines are displayed in gray. Although the existing coverage information of those addresses is retained, the coverage information will not be updated by program execution.

The acquired coverage information is accumulated in memory until the user clears it.

5.12.12 Clearing Data Coverage Information

(1) Clearing the data coverage information of the specified range

Selecting [Clear Coverage Range] from the pop-up menu on the [Address Range] or [Section] sheet opens the [Clear Coverage Range] dialog box.

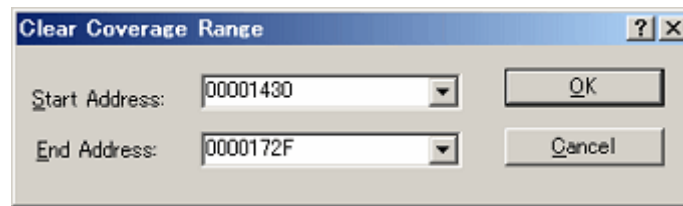


Figure 5.127 [Clear Coverage Range] Dialog Box

Enter the start and end addresses of the range to be cleared. Clicking the [OK] button clears the coverage information of the selected range.

(2) Clearing all data coverage information

Selecting [Clear the Entire Coverage] from the pop-up menu clears all the data coverage information.

5.12.13 Updating Coverage Information

Selecting [Refresh] from the pop-up menu updates the content of the [Data Coverage] window.

If [Lock Refresh] has been selected, the information is not automatically updated when the program breaks. To view the latest information, therefore, you need to update manually.

5.12.14 Preventing Update of Coverage Information

Selecting [Lock Refresh] from the pop-up menu prevents update of the [Data Coverage] window while the user program execution is stopped.

5.12.15 Saving the Data Coverage Information to a File

You can save the data coverage information of the currently selected sheet to a file.

Selecting [Save Data] from the pop-up menu opens the [Save Data] dialog box.

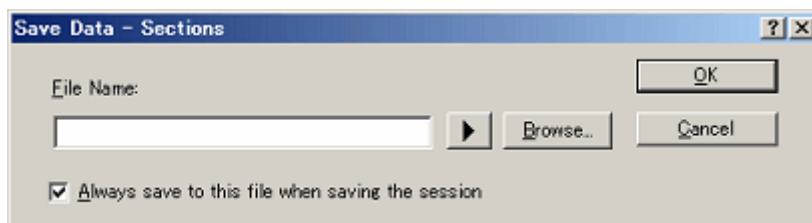


Figure 5.128 [Save Data] Dialog Box

Enter a file name in which you want the information to be saved. If a file extension is omitted, “.cdv” will automatically be added as the file extension. If you specify an existing file name, the file is overwritten.

5.12.16 Loading Data Coverage Information from a File

You can load a data coverage information file.

Selecting [Load Data] from the pop-up menu opens the [Load Coverage Data] dialog box.

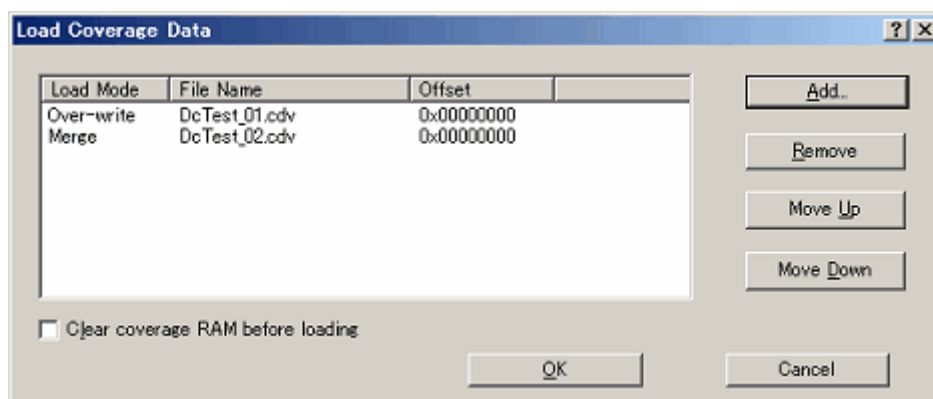


Figure 5.129 [Load Coverage Data] Dialog Box

Clicking on the [Add] button opens the [Add coverage data file] dialog box shown below.

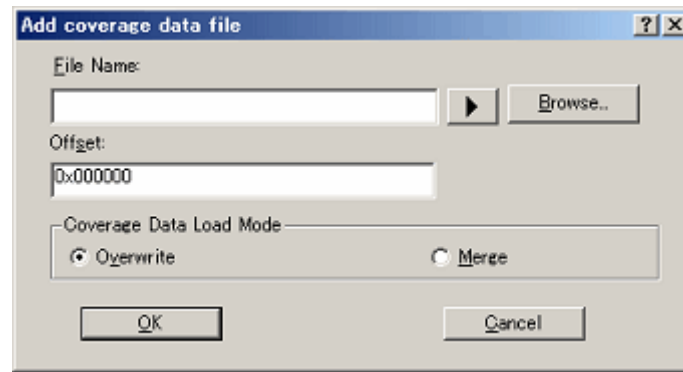


Figure 5.130 [Add coverage data file] Dialog Box

Use this dialog box to specify a coverage information file you want to load. You can also specify a load mode and offset for each file you load.

The only file extension available is ".cdv". An error message will appear if any other file extension is entered.

The files you add will be listed in the [Load Coverage Data] dialog box. The files will be loaded in the order in which they are listed. If necessary, use the [Move Up] or [Move Down] button to change the order.

5.13 Viewing Realtime Profile Information

5.13.1 Viewing Realtime Profile Information

The E100 emulator has code coverage, data coverage and realtime profile functions usable exclusively to each other.

To use the realtime profiling function, choose [Realtime Profile] in the [Switching function] section on the [System] page of the [Configuration properties] dialog box.

Realtime profiling is the function to measure execution performance within an area allocated to addresses in the profile range, one function or one task at a time. It will help you find the locations and causes of performance degradation in an application program.

Measurements are carried out without obstructing user program execution.

The measurement results are updated when the program breaks.

(1) Function profile

Execution performance is measured one function at a time.

The [Realtime Profile] window shows function names, the start addresses of functions, function sizes, counts and the cumulative execution time, execution rate and average execution time of functions.

The function profile of the E100 emulator does not include the execution time of subroutines in its cumulative display of function execution time.

CAUTION

The function profile is subject to the following limitations:

(a) About the areas to be measured

The E100 emulator can acquire profile information on all functions in areas up to 8 blocks, each in 128-KB units.

Each block you set can be comprised of a contiguous or noncontiguous address area.

No functions can be set that are outside the range of block addresses. In that case, the functions or tasks are displayed in gray.

(b) Limit to the number of functions

Measurement can be taken of up to $8\text{ K} - 1$ (= 8,191) functions.

If the number of functions measured exceeds $8\text{ K} - 1$ (= 8,191), the extra functions are excluded from the subject of measurement. In that case, those functions are displayed in gray. (Function names, address and function sizes are displayed in gray.)

(c) In-line expansion

The functions that are expanded in-line for optimization by the compiler are not displayed in the [Realtime Profile] window.

(d) Recursive functions

Although the execution time of recursive functions can be measured correctly, they are executed only once.

- (e) Relationship between Go execution start address and break address within a measurement range and the measurable range

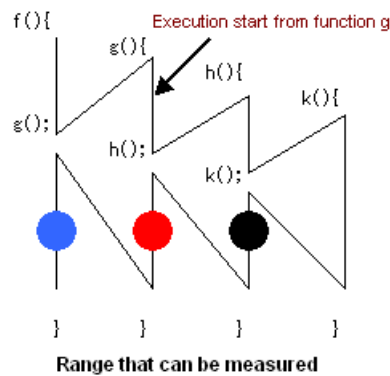


Figure 5.131 Measurable Range

Regardless of whether the program breaks at the position of a block dot [●], red dot [●], or blue dot [●], the following measurements are always performed.

Measurable range: Execution time of the function g and the number of times it is executed, the execution time of k and the number of times it is executed

Even after the program returned to a high-order function, execution counts of the function from which program execution started cannot be measured.

- (f) Function measurement

To measure functions accurately, you need to be in a function to be measured for 100 ns or more after entering the function. Otherwise, the execution time and count may not be measured properly.

- (g) Debug information option

To get execution time and execution count of functions, you need to specify a source file that includes the functions for measurement or an option that outputs debug information to the library during compiling. When not specifying the debug information option, you cannot measure execution time and execution counts of the function.

- (h) Maximum execution time and minimum execution time

With the realtime profile, you cannot measure the maximum and minimum execution time of a function. To measure the maximum and minimum execution time of a function, use the [Performance Analysis] window.

- (2) Task profile

Execution performance is measured one task at a time.

The [Realtime Profile] window shows task IDs, counts and the cumulative execution time, execution rate and average execution time of tasks.

5.13.2 Setting Realtime Profile Measurement Modes

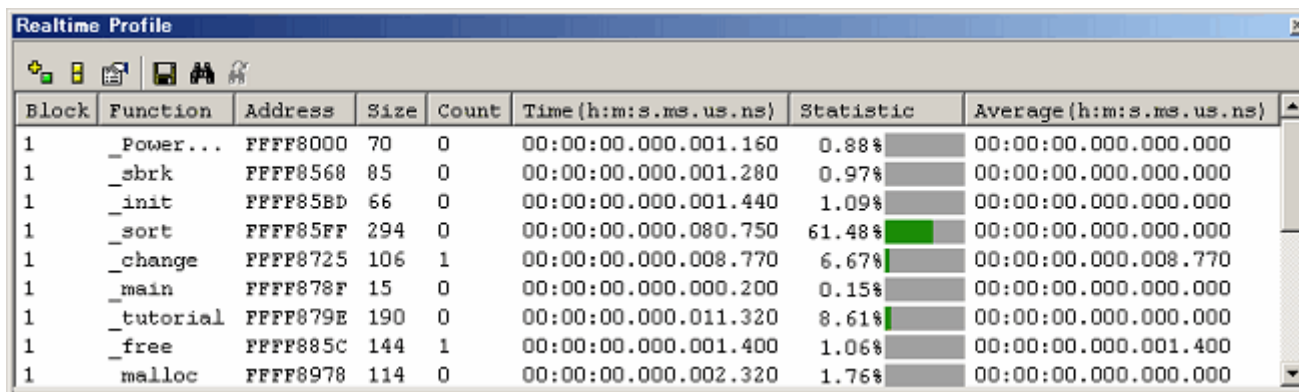
Choose [Set Ranges] from the pop-up menu that is displayed when you right-click in the present window.

The [Realtime Profile Setting] dialog box will be displayed. In the [Profile Mode] list box of this dialog box, you can select [Function profile] or [Task profile].

When profile modes are changed, all measurement results are cleared.

5.13.3 Measuring Function Profiles

You can measure execution performance one function at a time.



Block	Function	Address	Size	Count	Time (h:m:s.ms.us.ns)	Statistic	Average (h:m:s.ms.us.ns)
1	_Power...	FFFF8000	70	0	00:00:00.000.001.160	0.88%	00:00:00.000.000.000
1	_sbrk	FFFF8568	85	0	00:00:00.000.001.280	0.97%	00:00:00.000.000.000
1	_init	FFFF858D	66	0	00:00:00.000.001.440	1.09%	00:00:00.000.000.000
1	_sort	FFFF85FF	294	0	00:00:00.000.080.750	61.48%	00:00:00.000.000.000
1	_change	FFFF8725	106	1	00:00:00.000.008.770	6.67%	00:00:00.000.008.770
1	_main	FFFF878F	15	0	00:00:00.000.000.200	0.15%	00:00:00.000.000.000
1	_tutorial	FFFF879E	190	0	00:00:00.000.011.320	8.61%	00:00:00.000.000.000
1	_free	FFFF885C	144	1	00:00:00.000.001.400	1.06%	00:00:00.000.001.400
1	_malloc	FFFF8978	114	0	00:00:00.000.002.320	1.76%	00:00:00.000.000.000

Figure 5.132 [Realtime Profile] Window (Function Profile)

The following shows detailed information in each column.

Table 5.43 Details on Each Column

Block	Block number
Function	Function name
Address	Start address of function
Size	Function size
Count	Number of times a function is called
Time	Cumulative time of function execution The timestamp is displayed in the form shown below. Hours:minutes:seconds.milliseconds.microseconds.nanoseconds
Statistic	Ratio of function Time to Go-Break execution time
Average	Average execution time per measurement performed

If located outside the profile memory allocated area, address lines are displayed in gray.

The acquired profile measurement results are accumulated in memory until the user clears them.

5.13.4 Setting Function Profile Measurement Ranges

Choose [Set Ranges] from the pop-up menu that is displayed when you right-click in the present window.

The [Realtime Profile Setting] dialog box will be displayed. In this dialog box, set a profile measurement range.

[Function mode]

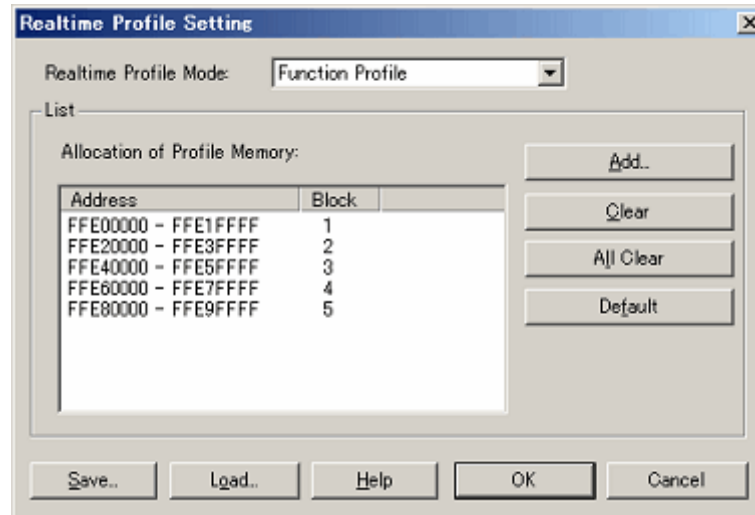


Figure 5.133 [Realtime Profile Setting] Dialog Box

(1) Memory allocation

Before function profiles can be measured, profile memory must be allocated to the addresses at which to be measured. Profile data can be obtained from only the address range that has had memory allocated.

You can specify any of blocks 1 to 8 (up to 1 Mbyte) each beginning with a 128-Kbyte boundary as profile measurement areas.

Either contiguous or non-contiguous blocks can be assigned.

With initial settings, the profile memory is automatically assigned at addresses in the ROM and RAM areas.

(2) Automatic function detection

When profile memory is assigned at addresses, the E100 emulator automatically detects functions included that address range and registers those functions in the window.

5.13.5 Saving Function Profile Measurement Ranges

You can save the current task mode and function profile measurement range (memory allocation state).

Click the [Save] button of the [Realtime Profile Setting] dialog box, and the [Save As] dialog box will be displayed.

Enter a file name in which you want function profile measurement ranges to be saved.

If the file extension is omitted, “.rpf” will automatically be added as the file extension.

If you specify an existing file name, a message is displayed asking you to confirm whether you want the file to be overwritten.

5.13.6 Loading Function Profile Measurement Ranges

You can load a function profile measurement range.

Click the [Load] button of the [Realtime Profile Setting] dialog box, and the [Open] dialog box will be displayed.

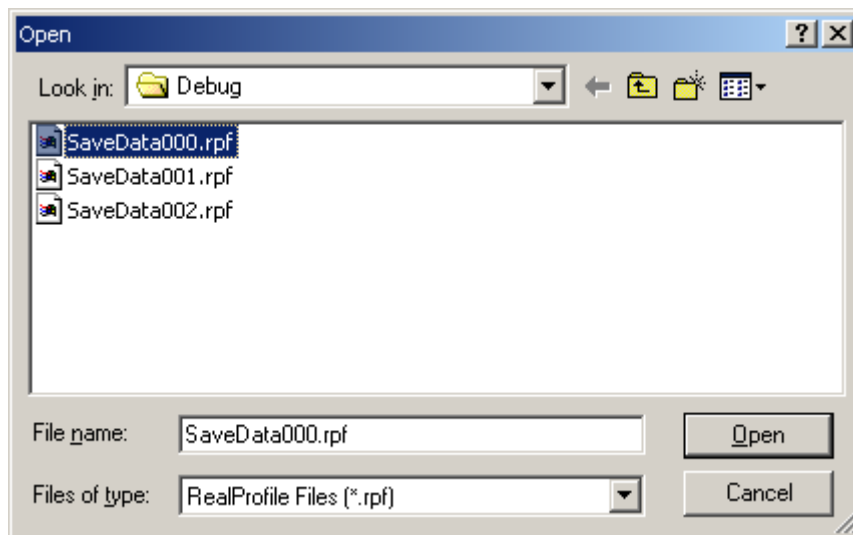


Figure 5.134 [Open] Dialog Box

Enter a file name you want to load.

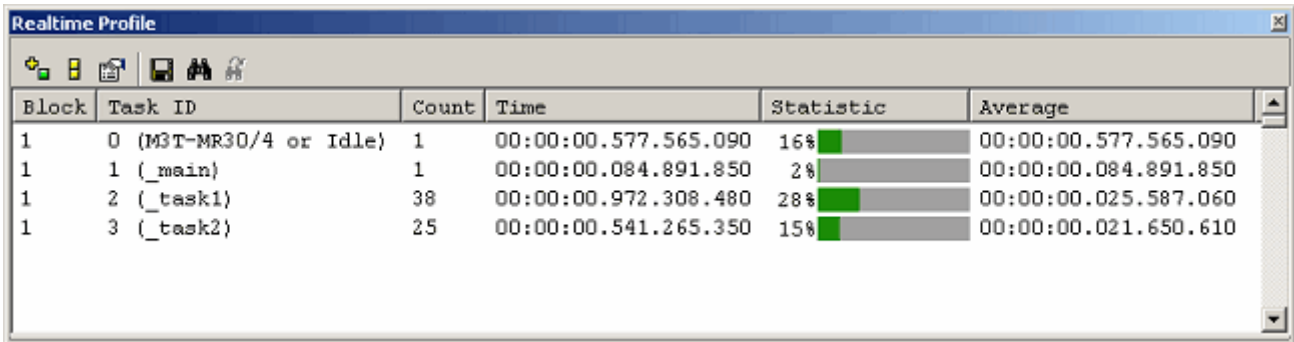
The only file extension available is “.rpf”. An error message will appear if any other file extension is entered.

When a file load is complete, the list in the [Realtime Profile Setting] dialog box is updated.

If task profile information is included in the loaded file, modes in the [Realtime Profile Setting] dialog box are switched to task mode.

5.13.7 Measuring Task Profiles

You can measure execution performance one task at a time.



Block	Task ID	Count	Time	Statistic	Average
1	0 (M3T-MR30/4 or Idle)	1	00:00:00.577.565.090	16%	00:00:00.577.565.090
1	1 (_main)	1	00:00:00.084.891.850	2%	00:00:00.084.891.850
1	2 (_task1)	38	00:00:00.972.308.480	28%	00:00:00.025.587.060
1	3 (_task2)	25	00:00:00.541.265.350	15%	00:00:00.021.650.610

Figure 5.135 [Realtime Profile] Window (Task Profile)

The following shows detailed information in each column.

Table 5.44 Details on Each Column

Block	Block number
Task ID	Task ID, entry address
Count	Number of times a task is called
Time	Cumulative time of task execution The timestamp is displayed in the form shown below. Hours:minutes:seconds.milliseconds.microseconds.nanoseconds
Statistic	Ratio of task Time to Go-Break execution time
Average	Average execution time per measurement performed

Disabled tasks are displayed in gray.

The acquired profile measurement results are accumulated in memory until the user clears them.

5.13.8 Setting Task Profile Measurement Ranges

Choose [Set Range] from the pop-up menu that is displayed when you right-click in the present window.

The [Realtime Profile Setting] dialog box will be displayed. In this dialog box, set a profile measurement range.

[Task mode]

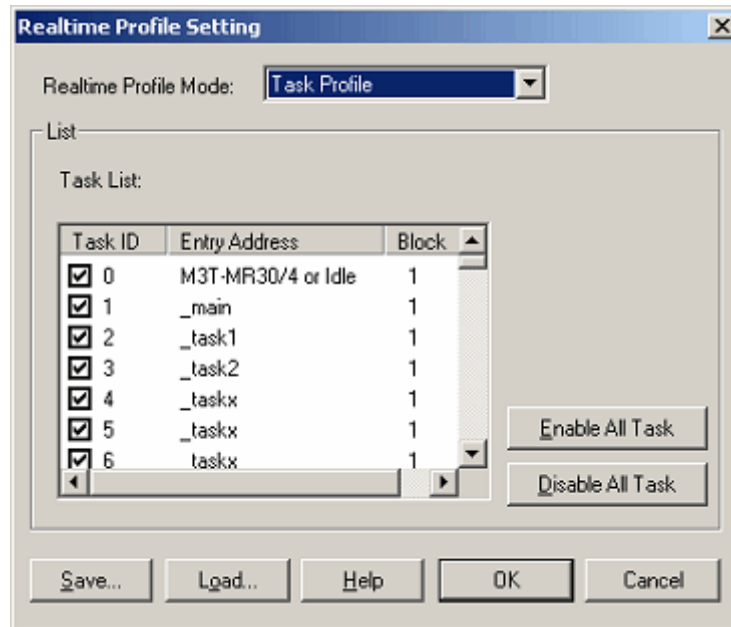


Figure 5.136 [Realtime Profile Setting] Dialog Box

(1) Automatic task detection

If you have downloaded a load module that has the OS included in it, the E100 emulator automatically detects a task list.

(2) Selecting tasks

Select the checkbox of a task ID you want to measure. (By default, all checkboxes are selected.)

The selected tasks will automatically be assigned block numbers (1–8).

CAUTION

If measurement blocks are lacking, block numbers become blank, so that no more task IDs can be registered. In that case, deselect the checkboxes of unnecessary task IDs.

5.13.9 Saving Task Profile Measurement Tasks

You can save the current task mode and measurement tasks (task IDs and enabled/disabled states).

Click the [Save] button of the [Realtime Profile Setting] dialog box, and the [Save As] dialog box will be displayed.

Enter a file name in which you want task profile measurement tasks to be saved.

If the file extension is omitted, “.rpf” will automatically be added as the file extension.

If you specify an existing file name, a message is displayed asking you to confirm whether you want the file to be overwritten.

5.13.10 Loading Task Profile Measurement Tasks

You can load task profile measurement tasks.

Click the [Load] button of the [Realtime Profile Setting] dialog box, and the [Open] dialog box will be displayed.

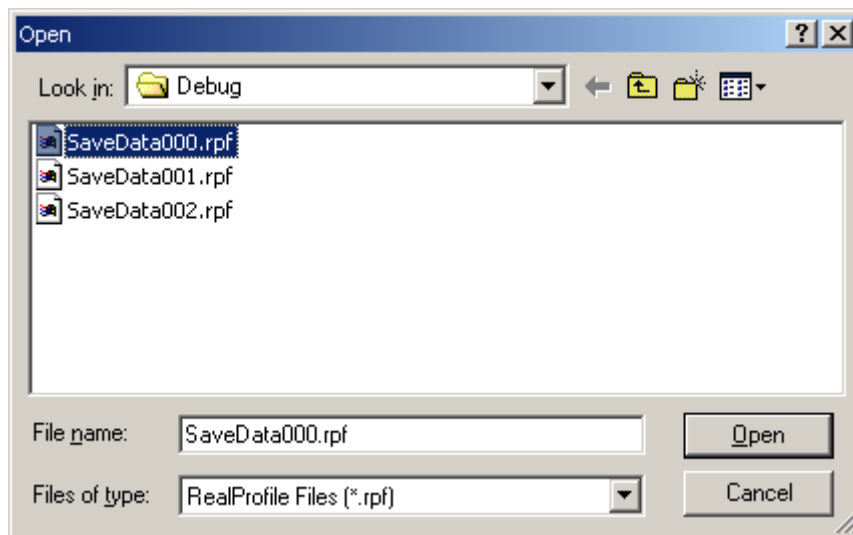


Figure 5.137 [Open] Dialog Box

Enter a file name you want to load.

The only file extension available is “.rpf”. An error message will appear if any other file extension is entered.

When a file load is complete, the list (task list) in the [Realtime Profile Setting] dialog box is updated.

If any loaded task IDs are nonexistent, although they are displayed once in the list (task list) in the [Realtime Profile Setting] dialog box, it is only the existing task IDs that are registered as measurement tasks when you click the OK button.

Reopening the [Realtime Profile Setting] dialog box, you can check the currently registered measurement tasks.

If function profile information is included in the loaded file, modes in the [Realtime Profile Setting] dialog box are switched to function mode.

5.13.11 Clearing Realtime Profile Measurement Results

Choose [Clear] from the pop-up menu of the [Realtime Profile] window, and all measurement results will be cleared.

Unless you choose to [Clear], measurement results are accumulated in memory.

5.13.12 Saving Realtime Profile Measurement Results

You can save the current realtime profile measurement results in text format.

Choose [Save To File] from the pop-up menu of the [Realtime Profile] window, and the [Save As] dialog box will be displayed.

Enter a file name in which you want the measurement results to be saved.

If the file extension is omitted, ".txt" will automatically be added as the file extension.

If you specify an existing file name, a message is displayed asking you to confirm whether you want the file to be overwritten.

5.13.13 Setting the Measurement Interval

Choose [Properties] from the pop-up menu that is displayed when you right-click in the present window.

The [Properties] dialog box will be displayed.

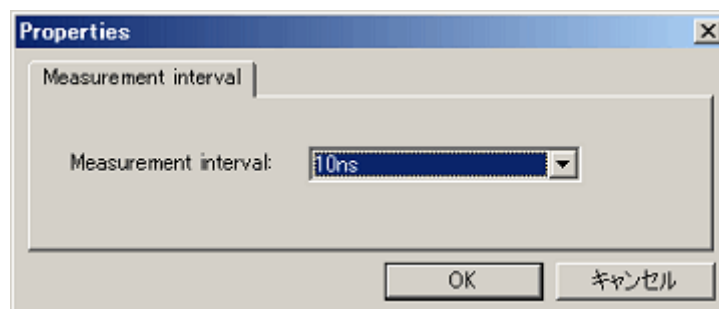


Figure 5.138 [Properties] Dialog Box

The measurement interval can be selected from the following options:

10 ns, 20 ns, 40 ns, 80 ns, 160 ns, 1.6 μ s

CAUTION

When the currently set measurement interval is changed, the measurement results accumulated are cleared.

5.13.14 Maximum Measurement Time of the Realtime Profile

(1) Maximum measurement time

The timer used for performance measurement is comprised of a 40-bit counter. The maximum measurement time varies with the measurement interval selected.

To select a measurement interval, specify it in the [Measurement interval] drop-down list of the [Properties] dialog box.

The measurable maximum times are listed below.

Table 5.45 Maximum Measurement Time

No.	Resolution	Maximum Measurement Time
1	10 ns	Approx. 3 hours, 03 minutes, 15 seconds
2	20 ns	Approx. 6 hours, 06 minutes, 30 seconds
3	40 ns	Approx. 12 hours, 13 minutes, 00 seconds
4	80 ns	Approx. 24 hours, 26 minutes, 00 seconds
5	160 ns	Approx. 48 hours, 52 minutes, 01 seconds
6	1.6 us	Approx. 488 hours, 40 minutes, 18 seconds

CAUTION

Note that performance measurement produces an error of \pm "2 resolution + 100 ns" (when resolution = 20 ns, \pm 140 ns) whenever entering functions. If the resolution is 20 ns and you enter functions 10 times, \pm 1400 ns error occurs.

(2) Maximum measurement count

Execution counts of the realtime profile are measured using a 16-bit counter. Measurement can be taken of up to a count of 65,535.

5.14 Detecting Exceptional Events

5.14.1 Detecting Exceptional Events

The E100 emulator permits you to detect various exceptional events that have occurred during user program execution.

Exceptional events include an abnormal behavior of the user program, as well as an overflow of the measurement counter of any function involved, etc. Detection of a specified exceptional event can be set as a condition of a breakpoint or trace point.

(1) Exceptional events

The E100 emulator detects the exceptional events listed below.

- Violation of access protection: An error is detected when an access other than a specified access attribute was attempted.
- Read from uninitialized memory: An error is detected when uninitialized area (not write accessed) was accessed for read.
- Performance overflow: An error is detected when the time measurement counter for a section has overflowed.
- Realtime profile overflow: An error is detected when the maximum measurable time or maximum measurable number of passes is exceeded during profile measurement of a function (or a task).
- Trace memory overflow: An error is detected when the trace memory has overflowed.
- Task stack access violation: An error is detected when one task attempts writing to the task stack of another task.
- OS dispatch: An error is detected if a task dispatch has occurred.

5.14.2 Detecting an Access Protect Violation

Violations of access protection such as writing to a ROM area or access to an unused area (for reading, writing, or execution of an instruction) can be detected as an error.

For CPU bus and DMA bus accesses, this can be set separately for each.

(1) Access attributes

For any CPU bus and DMA bus areas, the following attributes can be specified in word units, respectively.

Read/Write: Accessible for both read/write

Read Only: Accessible for read only

Write Only: Accessible for write only

Disable: Access prohibited

Disable (OS): Any access except from OS is prohibited (this attribute is automatically assigned for CPU bus detection only when a program including an OS is downloaded).

(2) Protected areas

Any area in the entire memory space may be access protected.

Initially on emulator startup, all areas assume the access attribute "Read/Write" (readable and writable).

(3) Methods for setting protection

There are following two methods of specification:

- Automatic setting by section information in a download module
- Specifying the access attribute of any area individually

(4) Detection method

An access protect violation is detected by the emulator's internal resources (blocks 1–16).

The blocks are automatically allocated by the emulator's exclusive algorithm.

CAUTION

Since the emulator's internal resources are limited, not all blocks can be access protected. In that case, reduce the amount of used blocks by "removing blocks" before setting protection again.

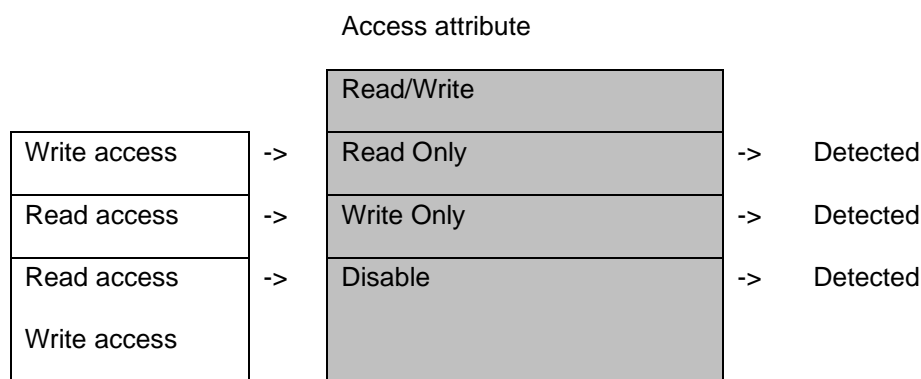


Figure 5.139 NG Patterns of Detection Methods

(5) Actions taken when an access protect violation is detected

The following actions can be set:

- Displaying a warning
Selecting the [Access Protect Violation] checkbox on the [Exception Warning] page of the [Configuration properties] dialog box, you can display a warning in the [Status] window and in a status bar balloon.
- Making the detection of an access protect violation a condition of a hardware breakpoint
- Making the detection of an access protect violation a condition of a trace point

5.14.3 Setting an Access Protected Area

Follow the procedure described below to set an access protected area.

(1) From the [Hardware Break] dialog box

1. Select the [Exception] checkbox on the [Hardware Break] sheet and then click the [Detail] button.

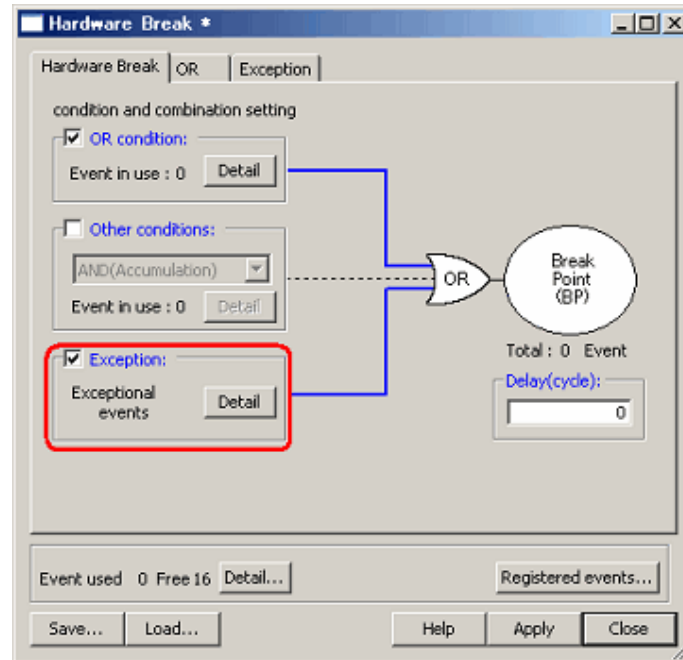


Figure 5.140 [Hardware Break] Dialog Box

2. The [Exception] page shown below will appear. Click the [Detail] button to the right of the [Violation of access protection] checkbox.

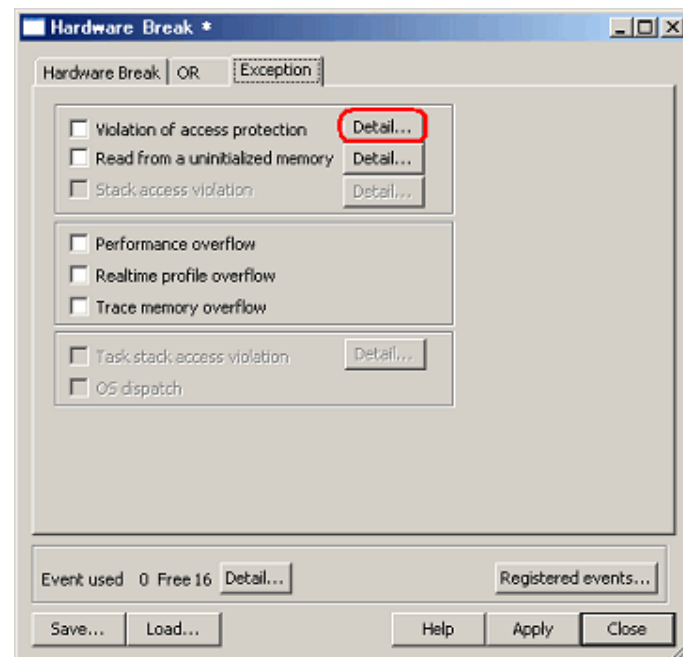


Figure 5.141 [Hardware Break] Dialog Box

- The [Violation of access protection] dialog box shown below will be displayed.
To have the access attributes automatically set according to the section information in a download module when a program is downloaded, select the checkbox labeled "Automatically set address areas at downloading".

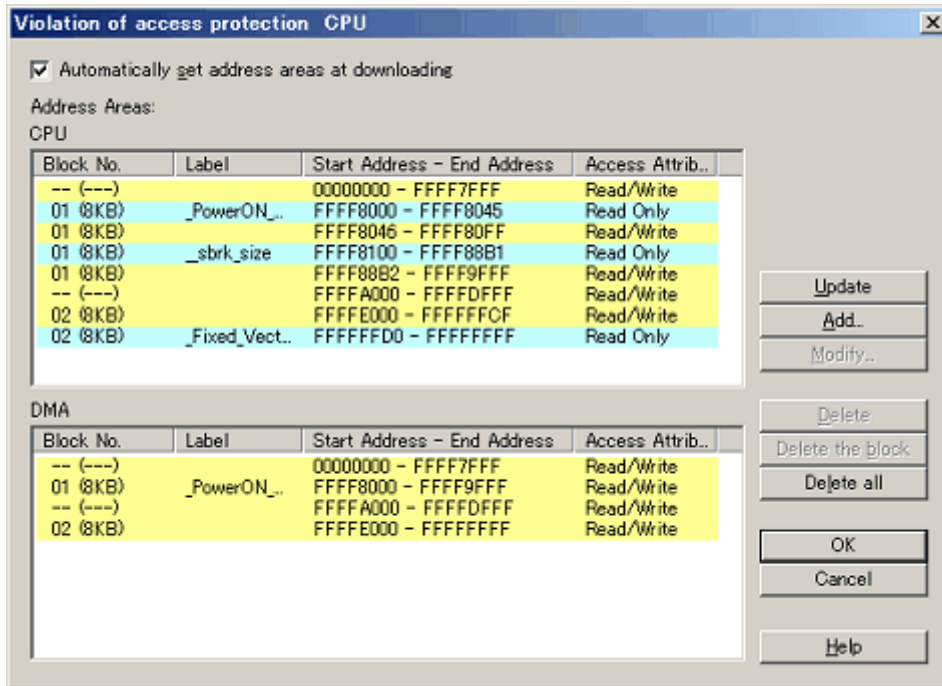


Figure 5.142 [Violation of access protection] Dialog Box

- Click the [Update] button, and the access attributes of the CPU bus will be updated according to the section information in a download module. In this case, information on the DMA bus becomes "Read/Write" in all the areas. You need to set information for the DMA bus manually clicking on the [Add] or [Modify] button.
- To add an access attribute manually, click anywhere in the list display area of the CPU bus or DMA bus and then click the [Add] button. When you click in the list display area, the caption part of the dialog box is updated, showing which bus area you are adding to.
Clicking the [Add] button brings up the [Access protection condition] dialog box. When it is displayed, specify the address range and access attribute you want.

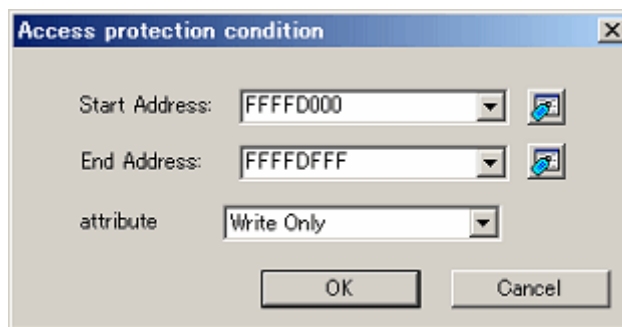


Figure 5.143 [Access protection condition] Dialog Box

- The protected area you have added will be displayed in the [Address Areas] list of the [Violation of access protection] dialog box.

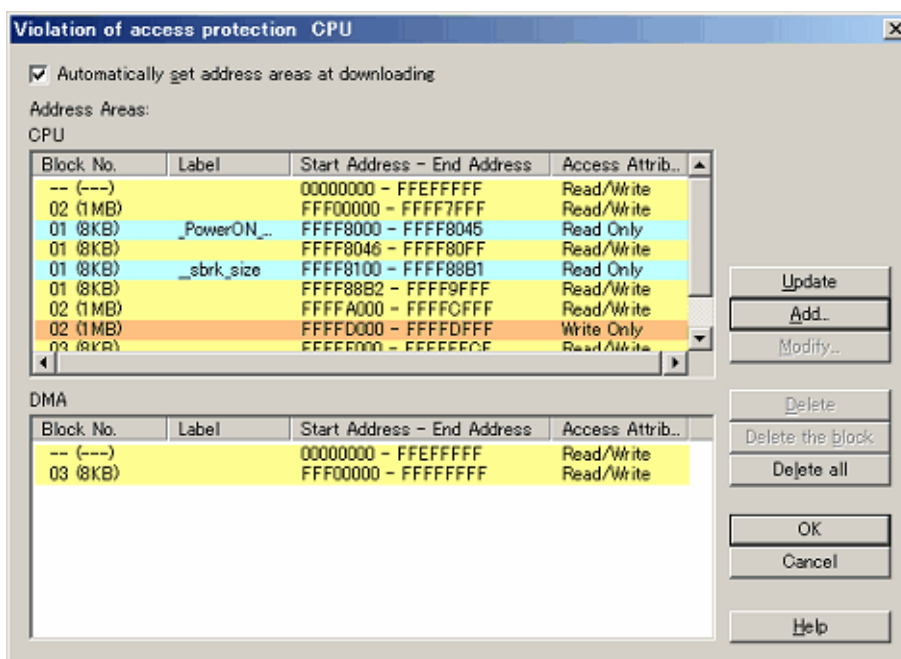


Figure 5.144 [Violation of access protection] Dialog Box

(2) From the [Trace conditions] dialog box

- In the [Trace Mode] drop-down list of the [Trace] sheet, select [Fill around TP]. Select the [Exception] checkbox and then click the [Detail] button.

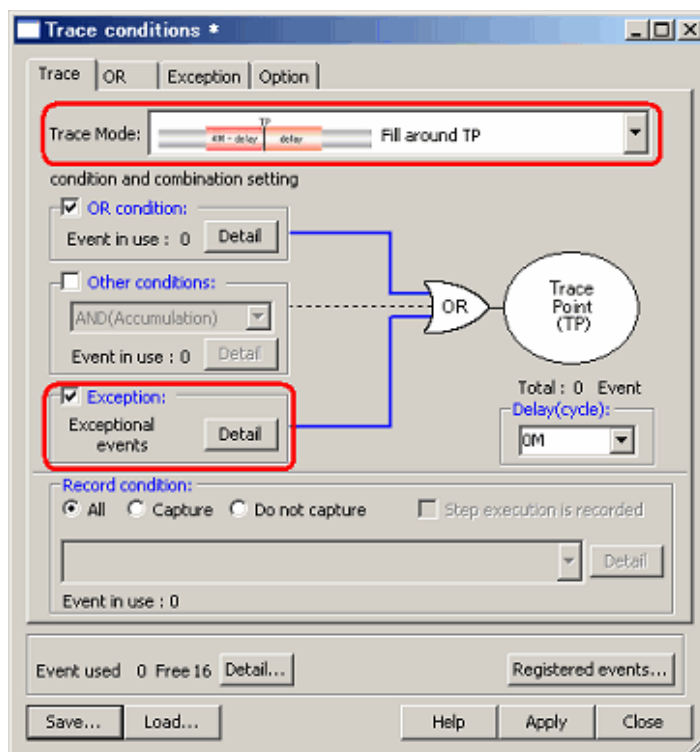


Figure 5.145 [Trace conditions] Dialog Box

- The [Exception] page shown below will appear. Click the [Detail] button to the right of the [Violation of access protection] checkbox.

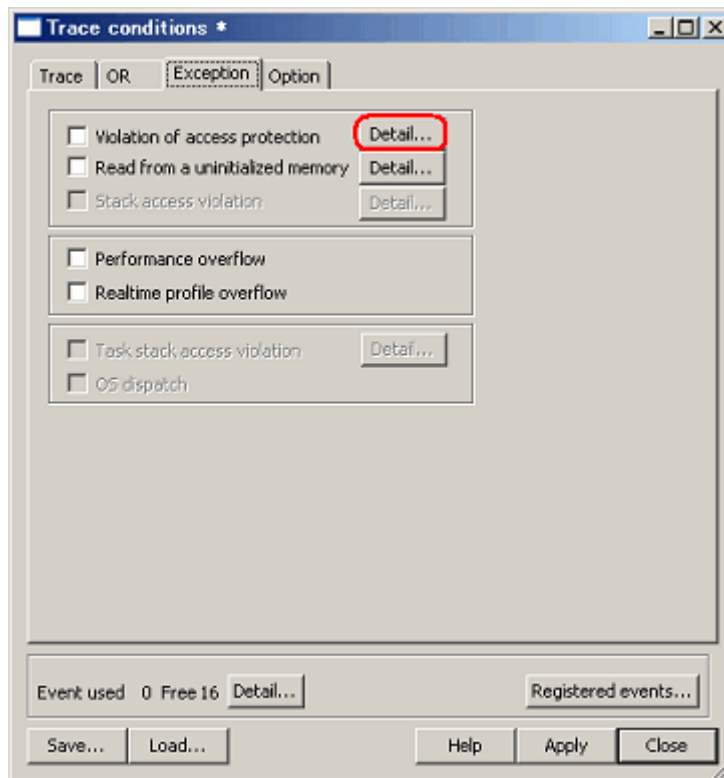


Figure 5.146 [Trace conditions] Dialog Box

- The [Violation of access protection] dialog box will be displayed.
The rest is the same as you opened it from the [Hardware Break] dialog box.

5.14.4 Detecting Reading from Uninitialized Memory

Reading from uninitialized memory, i.e. cases of reading from a memory location to which nothing has been written, can be detected as an error.

In the emulator, blocks 0-31 (maximum 16 Kbytes) can be specified as areas where reading from uninitialized memory is to be detected.

CAUTION

Detection of reading from uninitialized memory is applied to operand accesses only.

(1) Detection method

Reading from uninitialized memory is detected by the RAM monitor function.

When an area where reading from uninitialized memory is to be detected is set, the emulator automatically allocates a RAM monitor range, thereby enabling the detection.

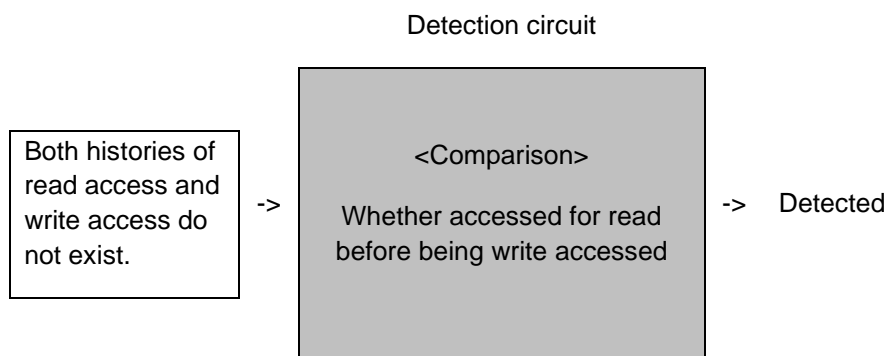


Figure 5.147 Outline of Detecting Reading from Uninitialized Memory

(2) Actions taken when reading from uninitialized memory is detected

The following actions can be set:

- Displaying a warning
Selecting the [Read from uninitialized memory] checkbox on the [Exception Warning] page of the [Configuration properties] dialog box, you can display a warning in the [Status] window and in a status bar balloon.
- Setting the detection of reading from uninitialized memory as a condition of a hardware breakpoint
- Setting the detection of reading from uninitialized memory as a condition of a trace point

5.14.5 Setting Areas where Reading from Uninitialized Memory is to be Detected

Follow the procedure below to set areas where reading from uninitialized memory is to be detected.

(1) In the [Hardware Break] dialog box

1. Check the [Exception] checkbox on the [Hardware Break] sheet and then click the [Detail] button.

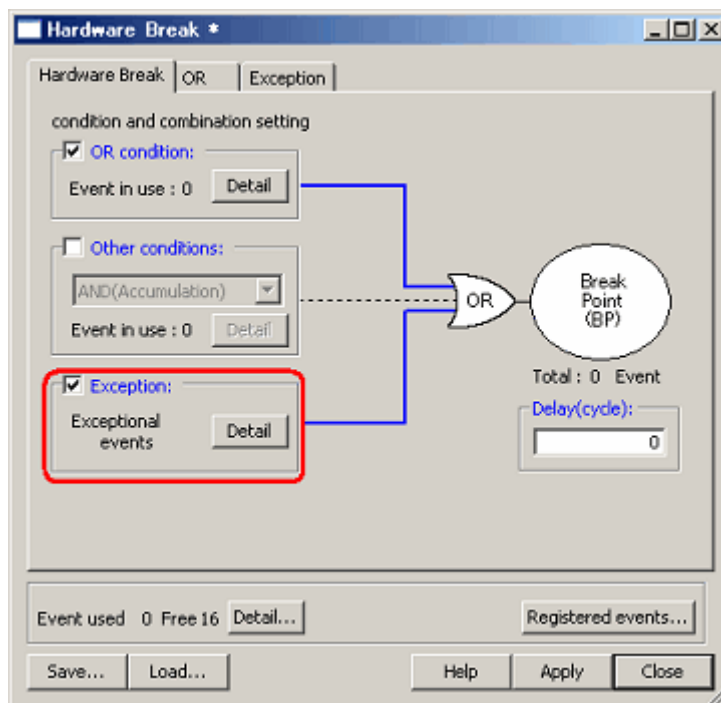


Figure 5.148 [Hardware Break] Dialog Box

2. The [Exception] page will open.

Click the [Detail] button to the right of the [Read from a uninitialized memory] checkbox.

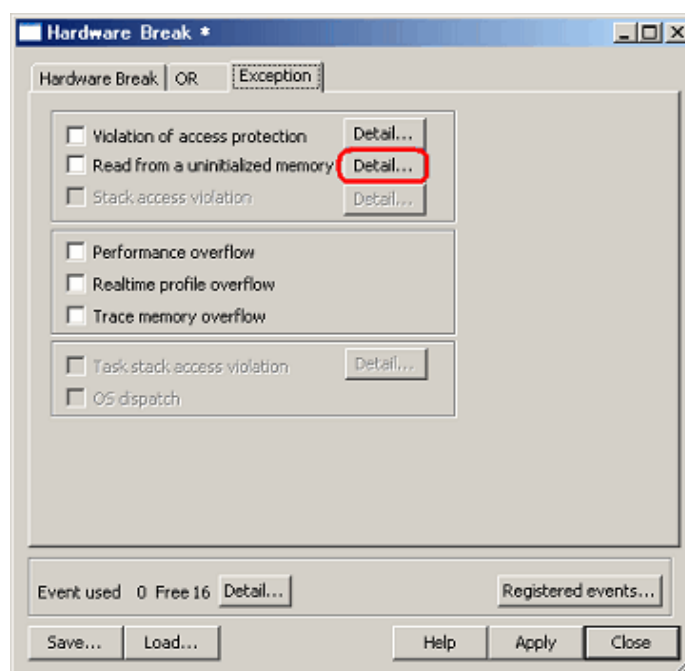


Figure 5.149 [Hardware Break] Dialog Box

3. The [Uninitialized area detection] dialog box will be displayed.

At startup, the emulator automatically sets a detection range beginning with the entry address of the RAM space and allocates a RAM monitor area at the same time. To detect uninitialized memory at addresses outside the range, you need to set the addresses manually. In this dialog box, the detection start point besides the setting of the range of detection can be set. Select the [Program Start] or [Address] radio button.

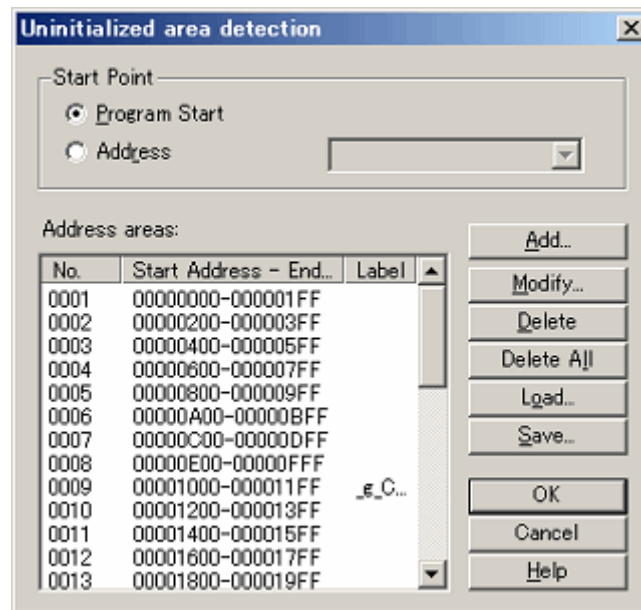


Figure 5.150 [Uninitialized area detection] Dialog Box

4. To manually add an area where reading from uninitialized memory is to be detected, click the [Add] button. The [Uninitialized area detection condition] dialog box will appear. Specify any address range you want. There are three options for address input to choose from: [Section], [Data Name], or [Address].

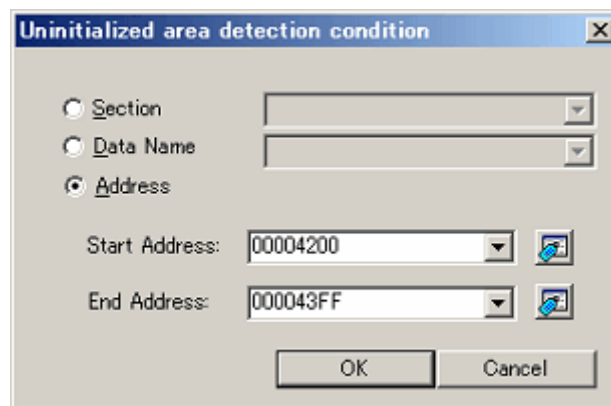


Figure 5.151 [Uninitialized area detection condition] Dialog Box

(2) In the [Trace conditions] dialog box

1. From the [Trace Mode] dropdown list on the [Trace] sheet, select [Fill around TP]. Select the [Exception] checkbox and then click the [Detail] button.

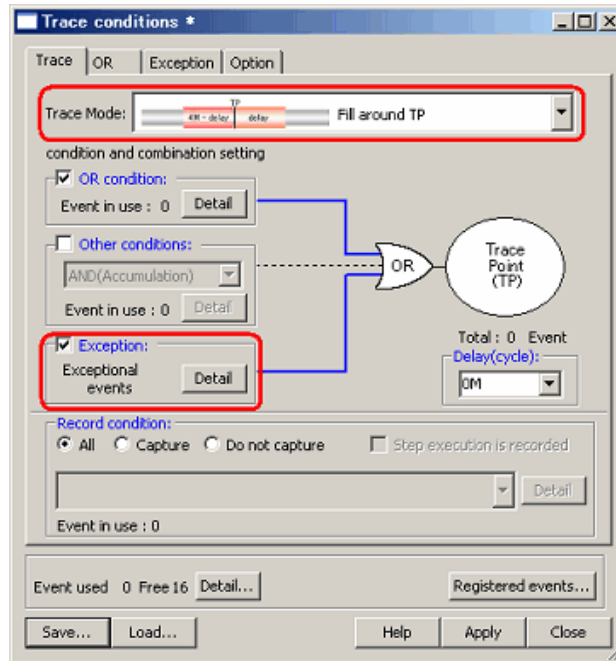


Figure 5.152 [Trace conditions] Dialog Box

2. The [Exception] page shown below will appear. Click the [Detail] button to the right of the [Read from a uninitialized memory] checkbox.

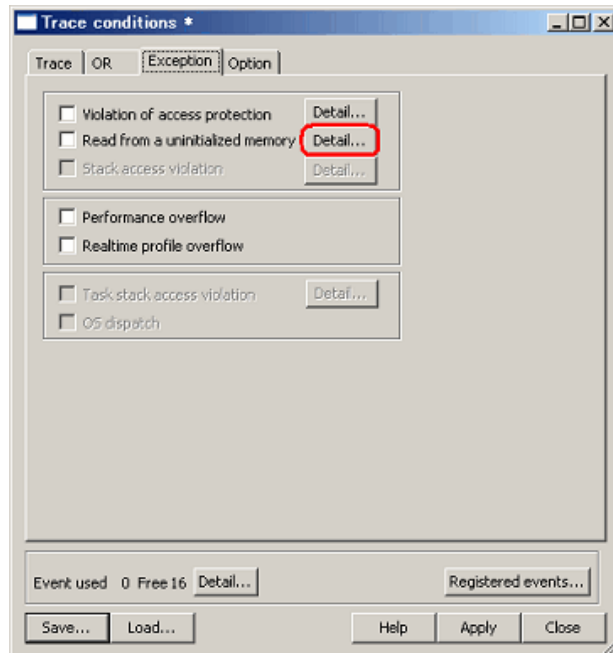


Figure 5.153 [Trace conditions] Dialog Box

3. The [Uninitialized area detection] dialog box will be displayed.
The rest is the same as you did when you opened the [Hardware Break] dialog box.

5.14.6 Detecting a Performance Overflow

A time in performance measurement coming to exceed the maximum value can be detected as an error.

Timeout case in a performance measurement is referred to as a performance overflow.

(1) Actions taken when a performance overflow is detected

The following actions can be set:

- Displaying a warning
A warning is displayed in the [Performance] window.
The result display line of a program section in which a timeout phenomenon occurred is marked with a string "overflow".
Selecting the [Performance Overflow] checkbox on the [Exception Warning] page of the [Configuration properties] dialog box, you can display a warning in the [Status] window and in a status bar balloon.
- Setting the detection of a performance overflow as a condition of a hardware breakpoint
- Setting the detection of a performance overflow as a condition of a trace point

5.14.7 Detecting a Realtime Profile Overflow

A time or number of passes in realtime profile measurement coming to exceed the maximum value can be detected as an error.

Timeout and count-out (count expired) cases in a realtime profile are collectively referred to as a realtime profile overflow.

(1) Actions taken when a realtime profile overflow is detected

The following actions can be set:

- Displaying a warning
A warning is displayed in the [Realtime Profile] window.
The function or result display line of a task in which a timeout or count-out phenomenon occurred is marked with a string "overflow".
Selecting the [Realtime Profile Overflow] checkbox on the [Exception Warning] page of the [Configuration properties] dialog box, you can display a warning in the [Status] window and in a status bar balloon.
- Setting the detection of a realtime profile overflow as a condition of a hardware breakpoint
- Setting the detection of a realtime profile overflow as a condition of a trace point

5.14.8 Detecting a Trace Memory Overflow

Overflows of the trace memory (4-M cycles) can be detected as errors.

(1) Actions taken when a trace memory overflow is detected

The following actions can be set:

- Displaying a warning
Selecting the [Trace memory overflow] checkbox on the [Exception Warning] page of the [Configuration properties] dialog box, you can display a warning in the [Status] window and in a status bar balloon.
- Setting the detection of a trace memory overflow as a condition of a hardware breakpoint

5.14.9 Detecting a Task Stack Access Violation

This facility is only available when a load module that includes an OS has been downloaded. The emulator detects an error when one task attempts writing to the task stack for another task.

CAUTION

Detection of task stack access violation is applied to operand accesses only.

(1) Initial settings at startup

At startup, the [Automatically set address areas at downloading] checkbox is selected (flagged by a check mark). However, because address information is nonexistent, the function does not work until a program is downloaded.

(2) Actions taken when a task stack access violation is detected

The following actions can be set:

- Displaying a warning
Selecting the [Task stack access violation] checkbox on the [Exception Warning] page of the [Configuration properties] dialog box, you can display a warning in the [Status] window and in a status bar balloon.
- Setting the detection of a task stack access violation as a condition of a hardware breakpoint
- Setting the detection of a task stack access violation as a condition of a trace point

5.14.10 Setting a Task Stack Area

Follow the procedure described below to set a task stack area.

(1) From the [Hardware Break] dialog box

1. Select the [Exception] checkbox on the [Hardware Break] sheet and then click the [Detail] button.

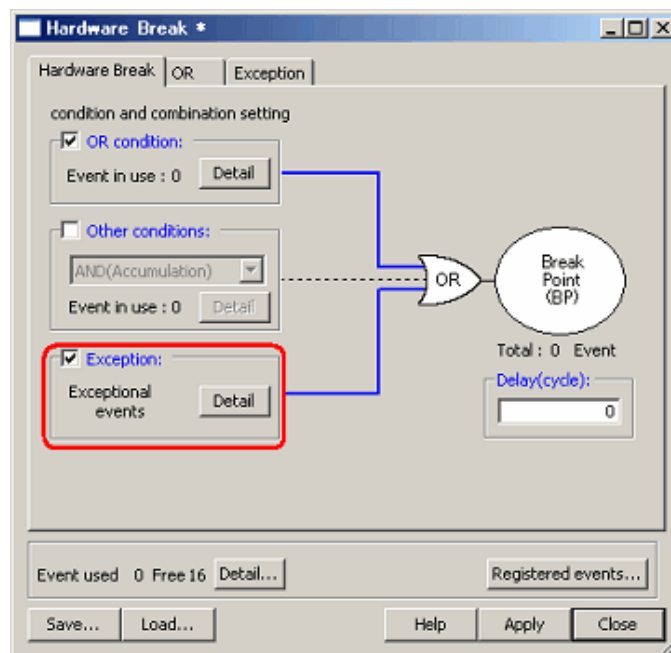


Figure 5.154 [Hardware Break] Dialog Box

2. The [Exception] page shown below will appear. Click the [Detail] button to the right of the [Task stack access violation] checkbox.

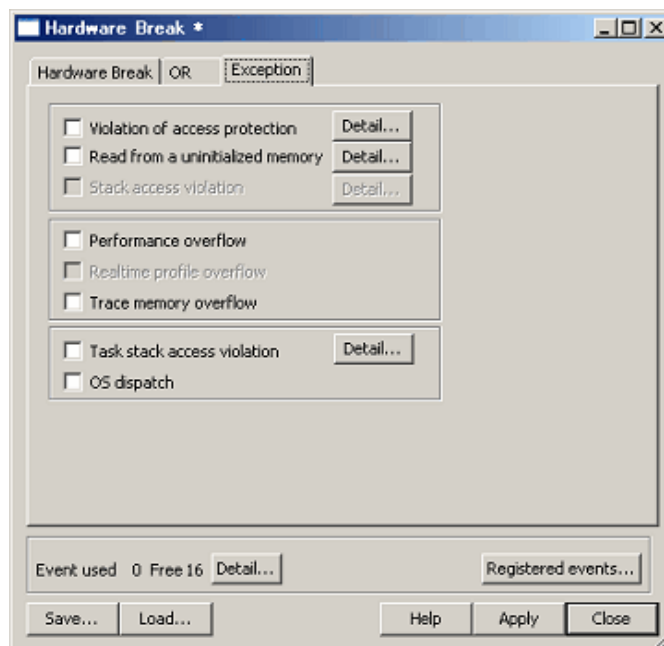


Figure 5.155 [Hardware Break] Dialog Box

- 3. The [Violation of task stack access] dialog box shown below will be displayed. To have the task stack ranges automatically set when a program is downloaded, select the [Automatically set address areas at downloading] checkbox.

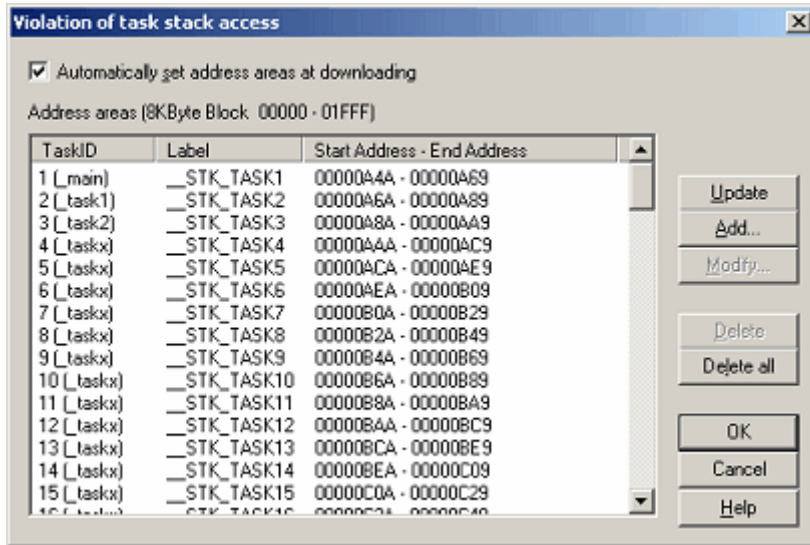


Figure 5.156 [Violation of task stack access] Dialog Box

- 4. Click the [Update] button, and the task stack ranges will be automatically set.
- 5. To add a task stack range manually, click the [Add] button. The [Task stack access condition] dialog box shown below will appear. Specify any task ID and the address range of a task stack.

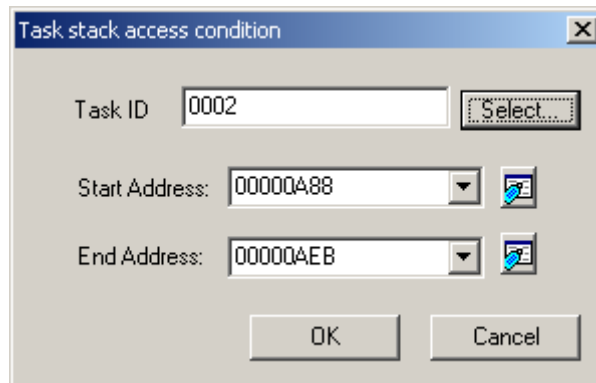


Figure 5.157 [Task stack access condition] Dialog Box

- 6. The task stack ranges you have added will be displayed in the [Address areas] list of the [Violation of task stack access] dialog box.

(2) From the [Trace conditions] dialog box

1. In the [Trace Mode] drop-down list of the [Trace] sheet, select [Fill around TP]. Select the [Exception] checkbox and then click the [Detail] button.

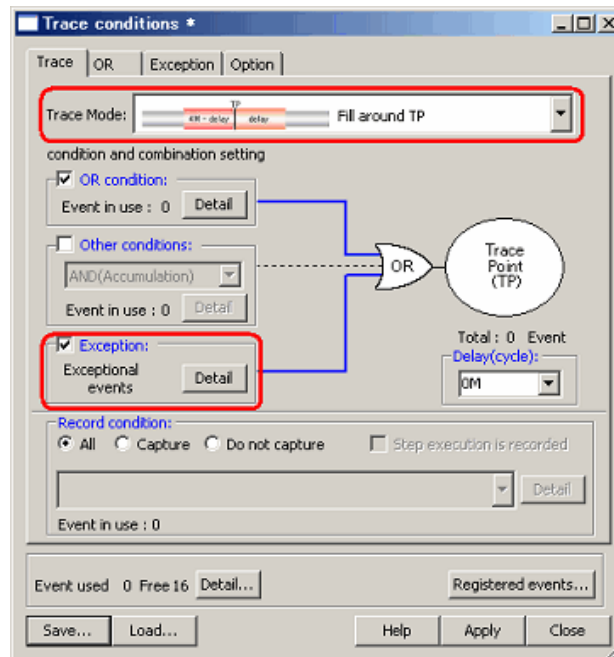


Figure 5.158 [Trace conditions] Dialog Box

2. The [Exception] page shown below will appear. Click the [Detail] button to the right of the [Task stack access violation] checkbox.

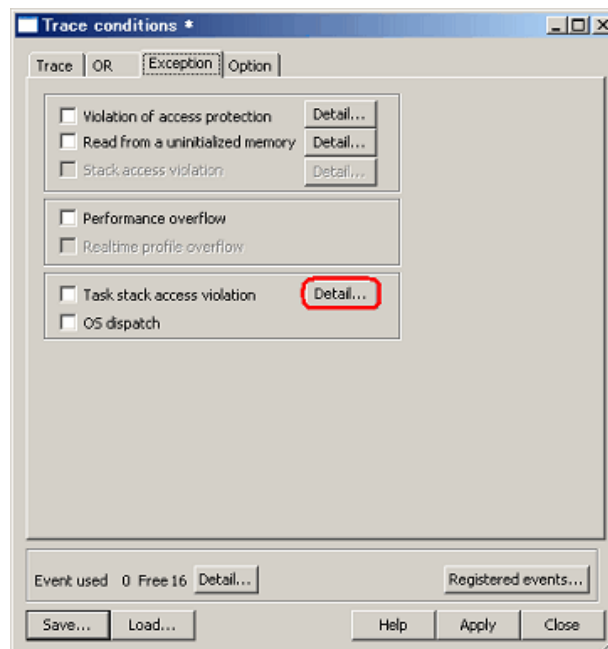


Figure 5.159 [Trace conditions] Dialog Box

3. The [Violation of task stack access] dialog box will be displayed. The rest is the same as you opened it from the [Hardware Break] dialog box.

5.14.11 Detecting an OS Dispatch

This facility is only available when a load module that includes an OS has been downloaded. The emulator detects the generation of task dispatch as an error.

CAUTION

Detection of OS dispatch is applied to operand accesses only.

(1) Actions taken when an OS dispatch is detected

The following actions can be set:

- Displaying a warning
Selecting the [OS dispatch] checkbox on the [Exception Warning] page of the [Configuration properties] dialog box, you can display a warning in the [Status] window and in a status bar balloon.
- Setting the detection of an OS dispatch as a condition of a hardware breakpoint
- Setting the detection of an OS dispatch as a condition of a trace point

5.15 Using the Start/Stop Function

The emulator executes the specified routine of the user program immediately before starting and immediately after halting program execution. This function is used to control the user system in synchronization with execution and halting of the user program.

5.15.1 Opening the [Start/Stop Function Setting] Dialog Box

The routine executed immediately before starting and immediately after halting the user program execution is 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.

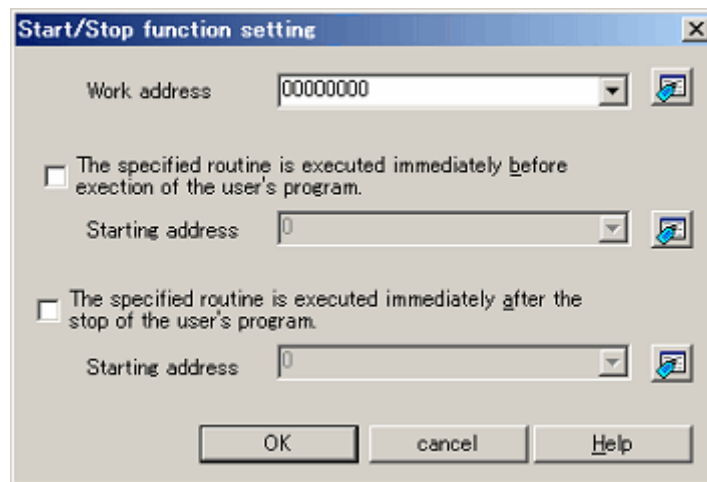


Figure 5.160 [Start/Stop function setting] Dialog Box

5.15.2 Specifying the Work Address

Use this command to specify the address of a work area (stack area) for use by a routine to run before the user program execution is started or after user program execution is stopped.

CAUTION

The specified address must be in the RAM area and not used by the user program.

5.15.3 Specifying the Routine to be Executed

It is possible to specify the respective routines immediately before starting and immediately after halting the user program execution.

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 this checkbox, is executed immediately before starting user program execution.

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 this checkbox, is executed immediately after halting user program execution.

5.15.4 Limitations on the Start/Stop Function

The Start/Stop function is subject to the following limitations.

- While the Start/Stop function is in use, do not use the debug functions listed below.
 - (a) Memory setting and download into the program area of a specified routine
 - (b) Breakpoint setting in the program area of a specified routine
- While a specified routine is executed, the 4-byte value, consisting of 4 addresses lower than the specified work address by 1 to 4 bytes respectively, is used under control on the emulator side.
- The general-purpose registers and flags used in a specified routine are subject to the following limitations.

Table 5.46 Limitations on the Registers and Flags

Register/Flag Name	Limitations
ISP register	When a specified routine has ended, the value of this register must be restored to one that it had when the specified routine started.
U flag	Do not switch to user mode while a specified routine is executed.
I flag	Interrupts are disabled while a specified routine is executed.
PM flag	Do not switch to user mode while a specified routine is executed.

- When a specified routine is executed, the debug functions listed below have no effect.
 - (a) Trace function
 - (b) Break-related functions
 - (c) RAM monitor function
- When a specified routine is executed, non-maskable interrupts are always disabled.
- The table below shows which state the MCU will be in when the user program starts running after a specified routine is executed.

Table 5.47 MCU Status at Start of the User Program

MCU Resource	Status
MCU general-purpose registers	These registers are in the state in which they were when the user program last stopped or the MCU registers that were set in the register window by the user. The register contents changed after a specified routine is executed are not reflected.
Memory in MCU space	Memory accesses attempted after a specified routine is executed are reflected.
MCU peripheral functions	Operation of the MCU peripheral functions after execution of the specified routine is continued.

5.15.5 Limitations on the Statements Written in a Specified Routine

The statements written in a specified routine are subject to the limitations described below.

- If a stack needs to be used in a specified routine, always be sure to use the interrupt stack.
- To terminate the processing of a specified routine, write a return subroutine instruction.
- Make sure that one session of processing performed by a specified routine is terminated within 10 ms. If, for example, the clock is turned off and kept inactive within a specified routine, then the emulator may become unable to control program execution.
- The values stored in the registers at the time a specified routine starts running are indeterminate. Be sure that the register values are initialized within a specified routine.
- The specified routine starts running in supervisor mode. Do not switch the mode to user mode.

5.16 Using the Trigger Output Function

The trigger output function allows output of signals through an external trigger cable. Trigger pin numbers 31 to 16 can be used for output. Note, however, that operation of a trigger pin depends on its pin number. Table 5.48 lists the trigger pin numbers and how they operate.

Table 5.48 Trigger Pin Numbers and Operation

No.	Operation
31 to 24	These pins constantly output a signal; either high or low can be selected.
23	A high-level signal is output when a breakpoint is encountered.
22	A high-level signal is output when a trace point is encountered.
21	A high-level signal is output when specific trace data is extracted or discarded.
20 to 16	An event can be specified for each of the signals and a high-level signal is output when that event occurs.

Output is at the power voltage level of the target system. If the MCU in use has two power supplies, the level on VCC1 will be applicable.

5.16.1 Using the External Trigger Cable for Output

You can specify input and output through the external trigger cable on the [System] page of the [Configuration properties] dialog box. Select the [EXT 0-15 INPUT EXT16-31 OUTPUT] radio button for [External trigger cable].

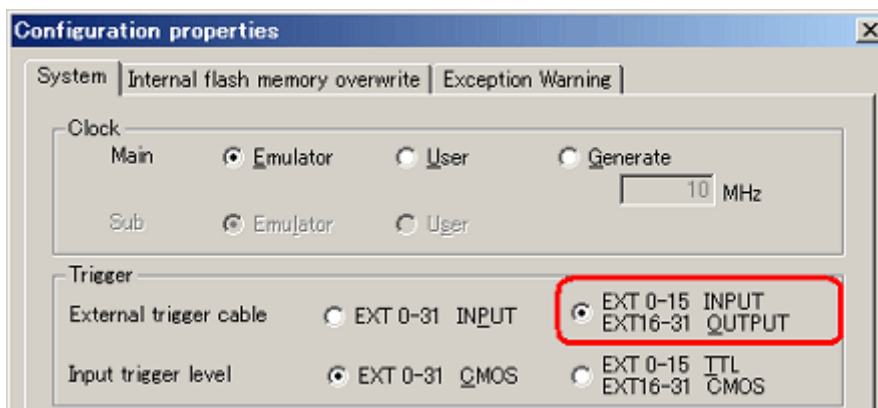



Figure 5.161 [Configuration properties] Dialog Box ([System] Page)

5.16.2 Opening the [Trigger Output Conditions] Dialog Box

Choose [Event -> Trigger Output Conditions] from the [View] menu, or click on the [Trigger Output Conditions] toolbar button [].

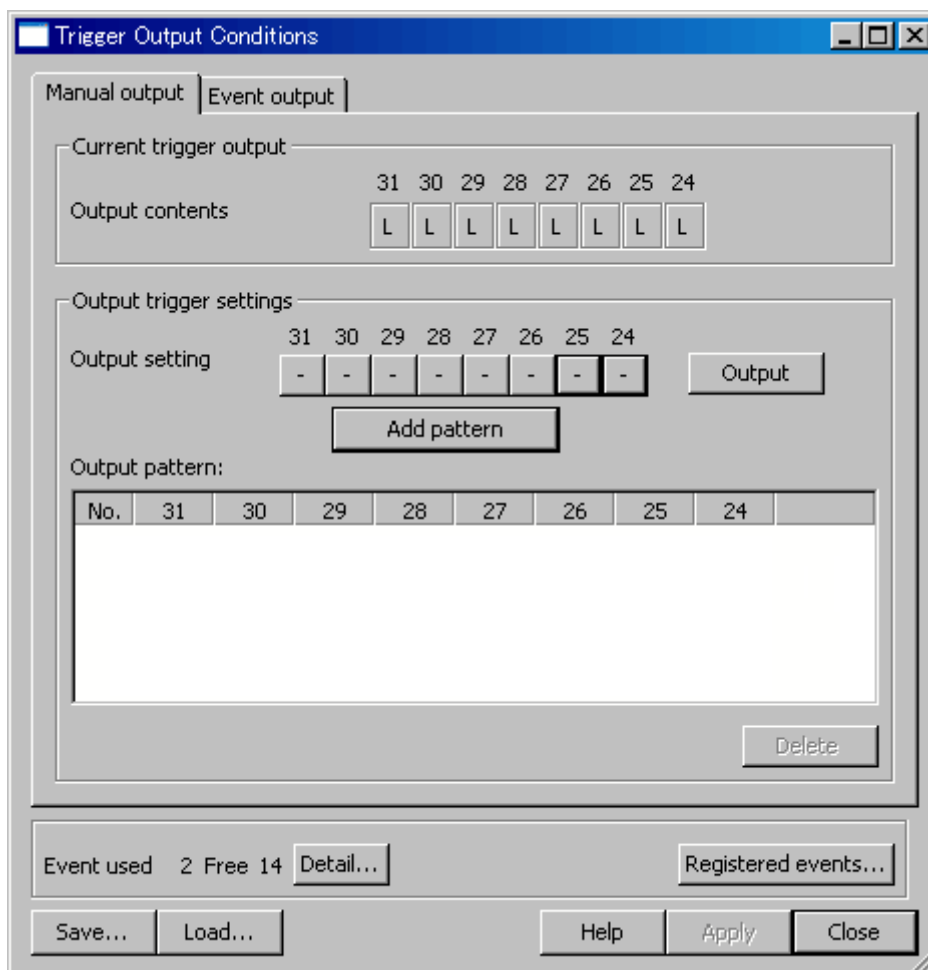


Figure 5.162 [Trigger Output Conditions] Dialog Box

Note that you cannot open the [Trigger Output Conditions] dialog box in either of the following cases.

- [EXT 0-31 INPUT] has been selected on the [System] page of the [Configuration properties] dialog box.
- An external trigger cable is not connected.

5.16.3 Manual Setting for Output through Trigger Pins 31 to 24

Make the manual settings for output through trigger pins 31 to 24 on the [Manual output] page.

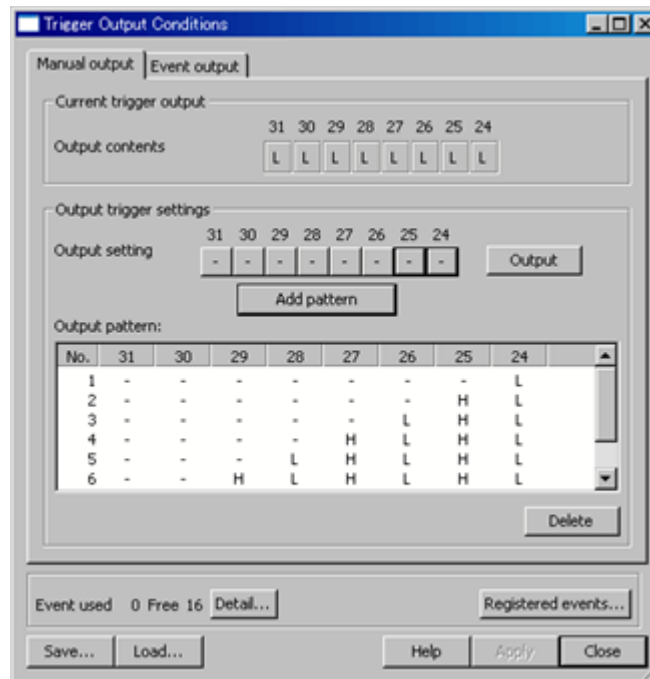


Figure 5.163 [Trigger Output Conditions] Dialog Box ([Manual output] Page)

(1) Display of output states: [Output contents]

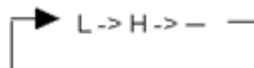
[Output contents] indicates the current signal levels on trigger pins 31 to 24.

H: High

L: Low

(2) [Output setting]

[Output setting] indicates the levels of signals to be output through trigger pins 31 to 24. Clicking on one of these buttons changes the state of the corresponding pin in the following order.



L: Low

H: High

-: The previous setting is retained.

When the [Trigger Output Conditions] dialog box is opened, the states of all signals in the [Output setting] section are always indicated as '-', whether the previous setting was L or H.

(3) Starting output of signals

Click on the [Output] button to validate the settings and start output of signals.

(4) Saving output patterns

- You can save the settings on trigger pins 31 to 24 and reflect a saved setting as [Output setting]. This simplifies operations.
- After making settings for [Output setting], click on the [Add pattern] button. The new setting will be added as the last line in the [Output pattern] list.
- Up to 256 patterns can be added.
- Double-clicking on a line in the [Output pattern] list reflects the information on the line as [Output setting].
- The order of the lines (patterns) can be changed by dragging and dropping.
- To delete a pattern, select the line and click on the [Delete] button.

5.16.4 Setting for Output through Trigger Pins 20 to 16

The [Event output] page allows manual setting for output through trigger pins 20 to 16.

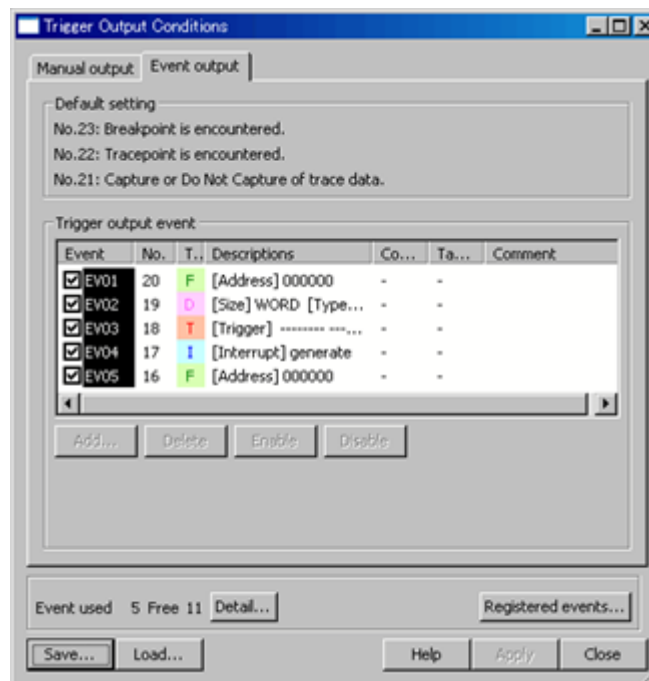


Figure 5.164 [Trigger Output Conditions] Dialog Box ([Event output] Page)

(1) Default setting

[Default setting] indicates the trigger output conditions on pins 23 to 21. These pins are always enabled. Signals are output through these pins when the respective conditions are satisfied. Table 5.49 gives details on how the conditions control output.

Table 5.49 Trigger Output Conditions and Output

No.	Condition	Output
23	A breakpoint is encountered	Continued output of a high-level signal is started.
22	A trace point is encountered	A high-level signal is output only during cycles in which the trace-point condition is satisfied.
21	Specific trace data is extracted or deleted	A high-level signal is output only in cycles where trace data is being extracted or discarded.

(2) Trigger output event

You can specify an event for trigger pins 20 to 16. A high-level signal will only be output while the event is occurring.

CAUTION

The actual trigger output follows event detection after some delay. The number of cycles of delay varies with the product. The delay for trigger output in the R0E556100MCU00 is 8 cycles.

5.16.5 Events

For details on the setting of events, see section "5.7 Using Events" (page 113).

6. Troubleshooting (Action in Case of an Error)

6.1 Flowchart for Remediation of Trouble

Figure 6.1 shows the flowchart for remediation of trouble arising between activation of the power supply to the emulator system and the emulator debugger starting up. Go through the checks with the user system disconnected. For the latest FAQs, visit the Renesas Tools Homepage.

<https://www.renesas.com/tools>

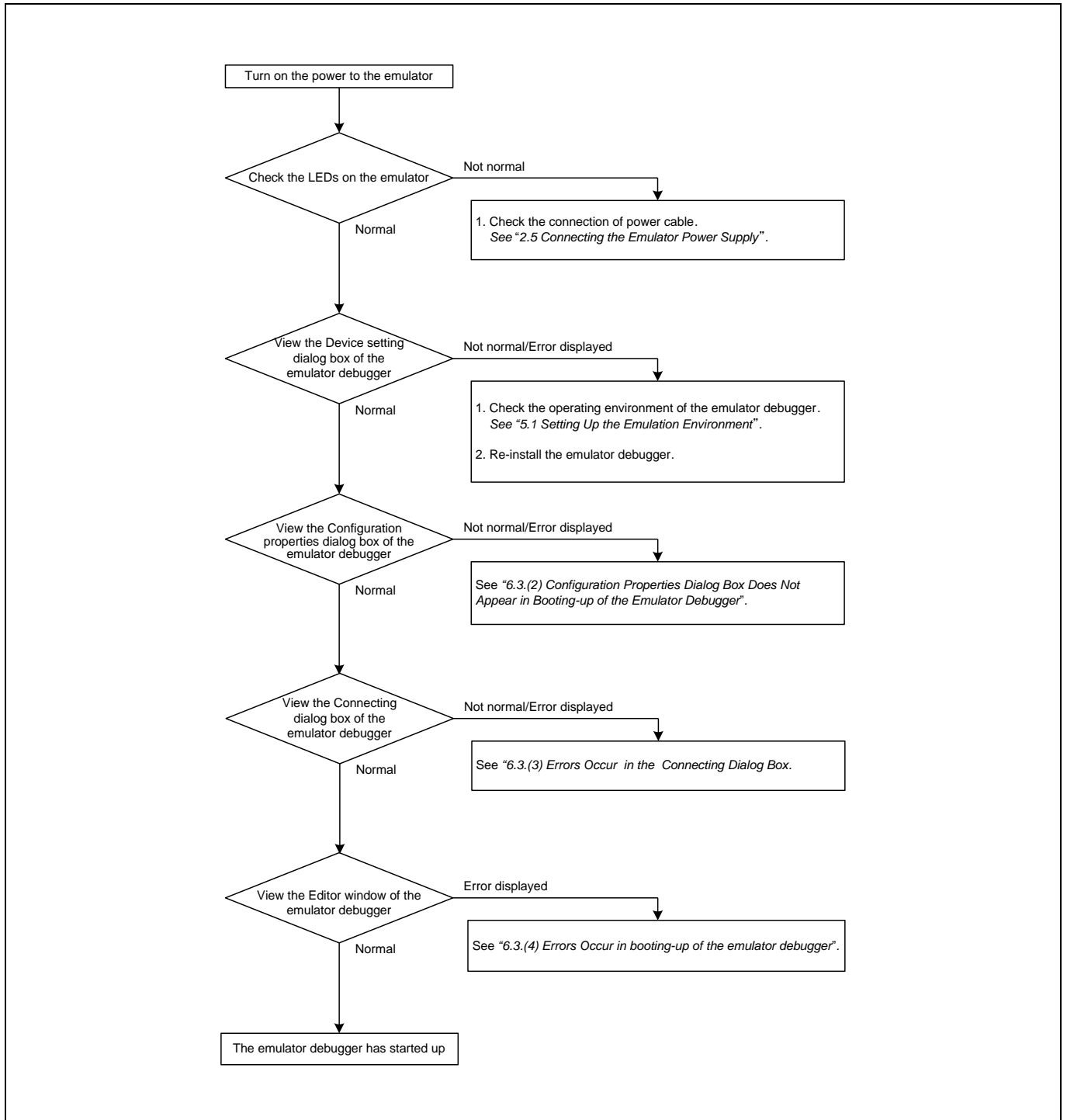


Figure 6.1 Flowchart for Remediation of Trouble

6.2 Error in Self-Checking

When an error occurs in self-checking, check the following items.

- (1) Re-check the connection between the E100 emulator main unit and the MCU unit.
- (2) Download the proper firmware again.
- (3) Check the error log from self-checking by the debugger software, and refer to the instructions given therein (see Figure 6.2).

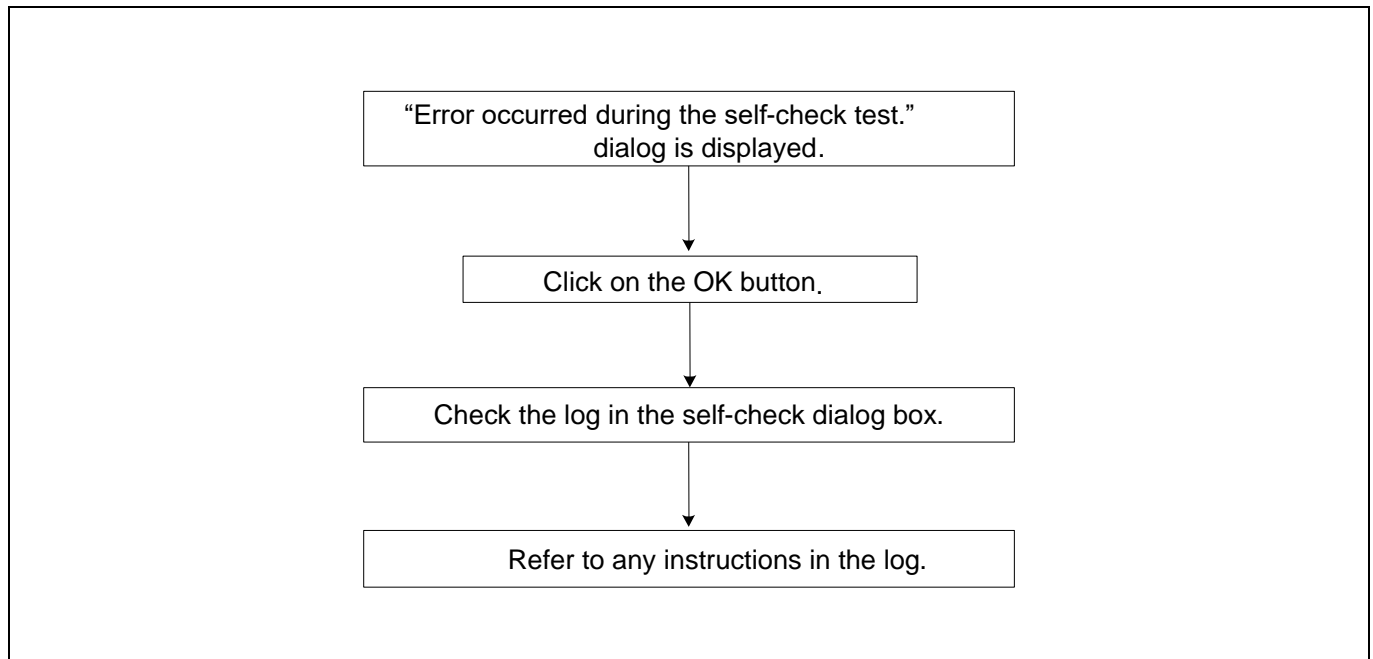


Figure 6.2 Flowchart for Checking of an Error in Self-Checking

IMPORTANT

Notes on the Self-checking:

Disconnect the MCU unit from a converter board and the user system before you start self-checking.

If the results of self-checking are not normal (excluding status errors of the target system), the product may have been damaged. Contact your local distributor.

6.3 Errors Reported in Booting-Up of the Emulator

(1) States of the LEDs on the E100 are incorrect

Table 6.1 Points to Check for Errors Indicated by Incorrect States of the LEDs on the E100

Error	Connection to the User System	Point to Check
SAFE LED remains lit.	-	Check that the USB cable is connected. See "Connecting the Host Machine" (page 29).
SAFE LED does not light up.	-	Re-check the connection between the E100 and the MCU unit. See "2.3 Connecting the MCU Unit to and Disconnecting it from the E100 Emulator Main Unit" (page 27).
Target Status POWER LED does not light up.	Connected	Check that power (Vcc) is being correctly supplied to the user system and that the user system is properly grounded (GND).
Target Status RESET LED does not go out.	Connected	(1) Check that the reset pin of the user system is being pulled up. (2) When using the emulator without the user system, check to see if the converter board is disconnected from the emulator.

(2) Configuration Properties Dialog Box Does Not Appear in Booting-up of the Emulator Debugger

Table 6.2 Points to Check for Errors in Booting-Up of the Emulator Debugger (1)

Error	Point to Check
Communication initialize error	Check all emulator debugger settings and the connection of the interface cable.
A communication error.	See "Preparing to Debug" (page 72).

(3) Error Occurs in the Connecting Dialog Box

Table 6.3 Points to Check for Errors in Booting-Up of the Emulator Debugger (2)

Error	Point to Check
MCU unit is not connected.	Re-check the connection between the E100 and the MCU unit. See "2.3 Connecting the MCU Unit to and Disconnecting it from the E100 Emulator Main Unit" (page 27).
The system configuration of the E100 emulator is not corresponding to the content of the E100.ENV file.	The combination between the emulator software and the MCU unit is not correct. Refer to the release notes of the emulator software, and confirm the combination between the emulator software and the MCU unit.
A timeout error. The MCU is in the reset state. Is system reset issued?	Check the oscillation of the oscillator module mounted on the MCU unit, and confirm that the oscillator module is properly mounted.
A timeout error. The MCU's internal clock is halted. Is system reset issued?	
A timeout error. No clock signal is supplied to the MCU. Is system reset issued?	
A timeout error. The power supply to the MCU is off. Is system reset issued?	Check that power is being correctly supplied to the user system and that the user system is properly grounded.

(4) Errors Occur in Booting-Up of the Emulator Debugger

Table 6.4 Points to Check for Errors in Booting-Up of the Emulator Debugger (3)

Error	Point to Check
A timeout error.	(1) Check that NQPACK etc. mounted on the user system is soldered properly. (2) Check that the connector is installed properly to the user system.

(5) Errors occur when a program is downloaded, written to the MCU's internal flash ROM, or the program is run.

Table 6.5 Points to Check for Errors with the Emulator Debugger

Error	Point to Check
Because the access violation etc. to on-chip ROM area are generated, the flash rewriting cannot be executed.	The access violation is committed because the MCU's internal flash ROM is not correctly set up. Check the set contents of registers.
The access violation to on-chip ROM was detected, and the flash rewriting was not able to be executed normally.	While a transfer from the MCU's internal flash ROM by a DMAC or DTC request was in progress, a flash rewrite from the emulator was executed and an access violation occurred in that process, causing the FCU to get into a command locked state. Wait until the transfer from the DMAC or DTC is complete before you download a program, write to the MCU's internal flash ROM, and run the program.

6.4 How to Request Support

After checking the items under "6. Troubleshooting (Action in Case of an Error)", fill in the text file which is downloadable from the following URL, then send the information to your local distributor.

<https://www.renesas.com/contact>

For a prompt response, please fill in the following information:

(1) Operating environment

- Operating voltage: _____ [V]
- Operating frequency: _____ [MHz]
- Clock supply to the MCU: Internal oscillator/External oscillator

(2) Condition

- The emulator debugger starts up/does not start up
- The error is detected/not detected in the self-checking
- Frequency of errors: always/frequency (_____)

(3) Details of request for support

7. Hardware Specifications

This chapter describes specifications of the MCU unit.

7.1 Target MCU Specifications

Table 7.1 lists the specifications of target MCUs which can be debugged with the MCU unit.

Table 7.1 Specifications of Target MCUs for R0E556100MCU00

Item	Description
Applicable MCU	RX600 Series RX610 Group MCUs with 2-MB ROM or less
Evaluation MCU	R5E56108 ROM size : 2 MB + 32 KB, RAM size: 128 KB
Applicable MCU mode	Single-chip mode, on-chip ROM enabled extended mode, on-chip ROM disabled extended mode
Maximum ROM/RAM capacity	1. Internal flash ROM: 2 MB + 32 KB 00100000h to 00107FFFh, FFE00000h to FFFFFFFFh 2. Internal RAM: 128 KB 00000000h to 0001FFFFh
Power supply voltage	Vcc: 3.0 to 3.6 V
Operating voltage/frequency	Power supply voltage: 3.0 to 3.6 V, 100 MHz (with PLL)

7.2 Differences between the Actual MCU and Emulator

Differences between the actual MCU and emulator are shown below. When debugging the MCU using the MCU unit, be careful about the following precautions.

IMPORTANT

Note on Differences between the Actual MCU and Emulator:

Operations of the emulator system differ from those of actual MCUs as listed below.

(1) Reset condition

Set the time for starting up (0.2 Vcc to 0.8 Vcc) 1 μ s or less.

(2) Initial values of internal resource data of an MCU at power-on

(3) Capacities of the internal memories (ROM and RAM)

The evaluation MCU of this product has RAM of 128 KB and flash ROM of 2 MB.

(4) Oscillator circuit

In the oscillator circuit where an oscillator is connected between pins XTAL and EXTAL, oscillation does not occur because a converter board is used between the evaluation MCU and the user system.

(5) A/D converter

The characteristics of the A/D converter differ from those of actual MCU because there are a converter board and other devices between the evaluation MCU and the user system.

Note on RESET# Input:

A low input to pin RESET# from the user system is accepted only when a user program is being executed (only while the RUN status LED on the E100 upper panel is lit).

Note on the Watchdog Timer:

While the user program is stopped, the timer counter of the watchdog timer is also halted. When the user program restarts, the timer counter of the watchdog timer also starts counting from where it stopped.

Notes on Maskable Interrupts:

Even if a user program is not being executed, the evaluation MCU executes a debug control program. Therefore, timers and other components do not stop running. If a maskable interrupt is requested when the user program is not being executed, 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, a peripheral I/O interrupt is not accepted.

Note on the DMAC and DTC:

With this product, the user program is stopped with a loop program to a specific address. Therefore, if a DMAC and DTC requests are generated while the user program is stopped, DMAC and DTC are executed. However, make note of the fact that DMAC and DTC while the program is stopped may not be performed correctly. Also note that the registers have been changed to generate DMAC and DTC transfer as explained here even when the user program is stopped.

Note on Final Evaluation:

Be sure to evaluate your system with an evaluation MCU. Before starting mask production, evaluate your system and make final confirmation with a CS (commercial sample) version MCU.

7.3 Connection Diagram

7.3.1 Connection Diagram for R0E556100MCU00

Figure 7.1 shows a partial circuit diagram of the connections of R0E556100MCU00. This diagram mainly shows the circuitry to be connected to the user system. Other circuitry, such as that for the emulator's control system, has been omitted.

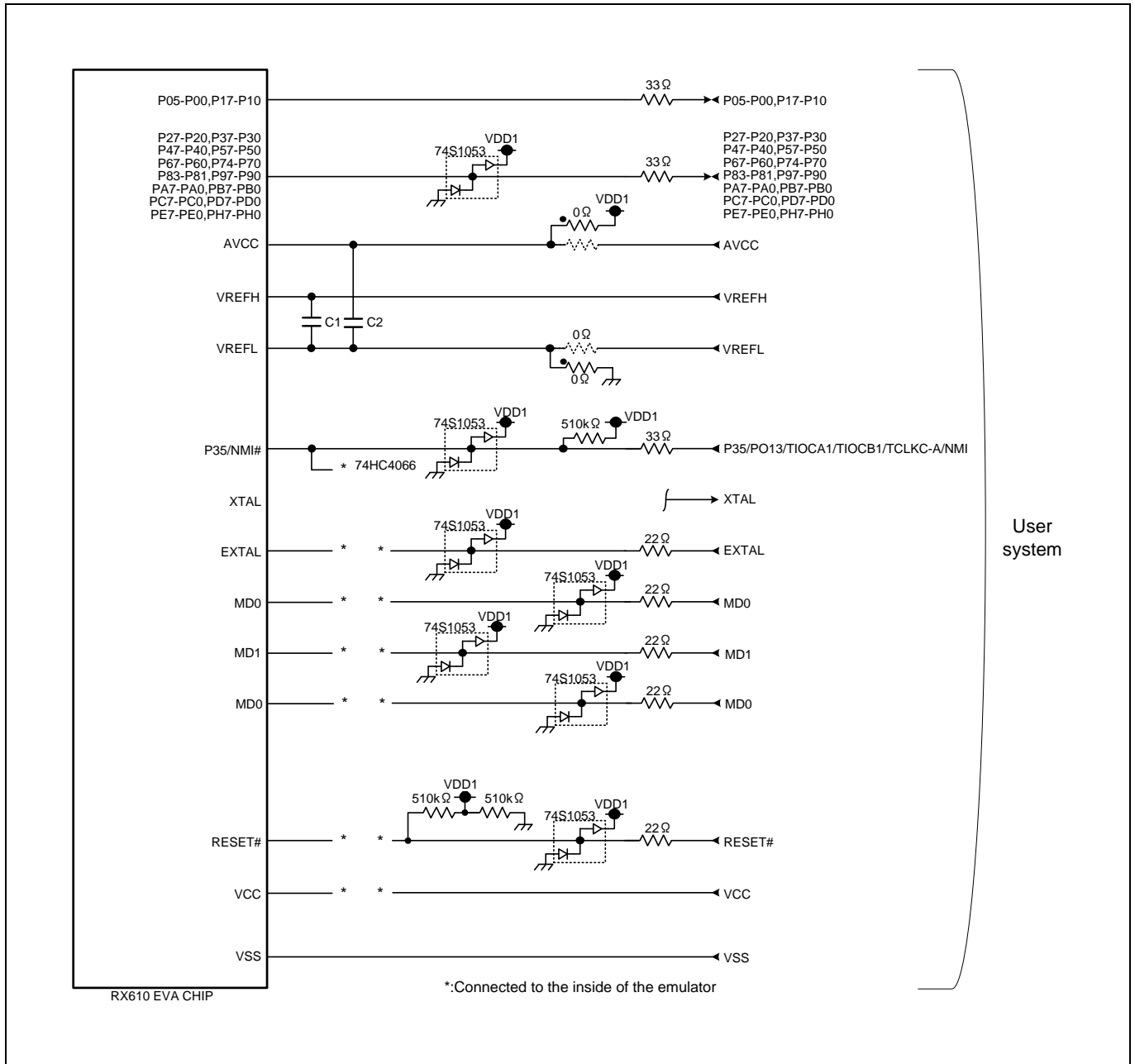


Figure 7.1 Connection Diagram for R0E556100MCU00 (Partial)

7.4 External Dimensions

7.4.1 External Dimensions of the E100 Emulator

Figure 7.2 shows external dimensions of the E100 emulator.

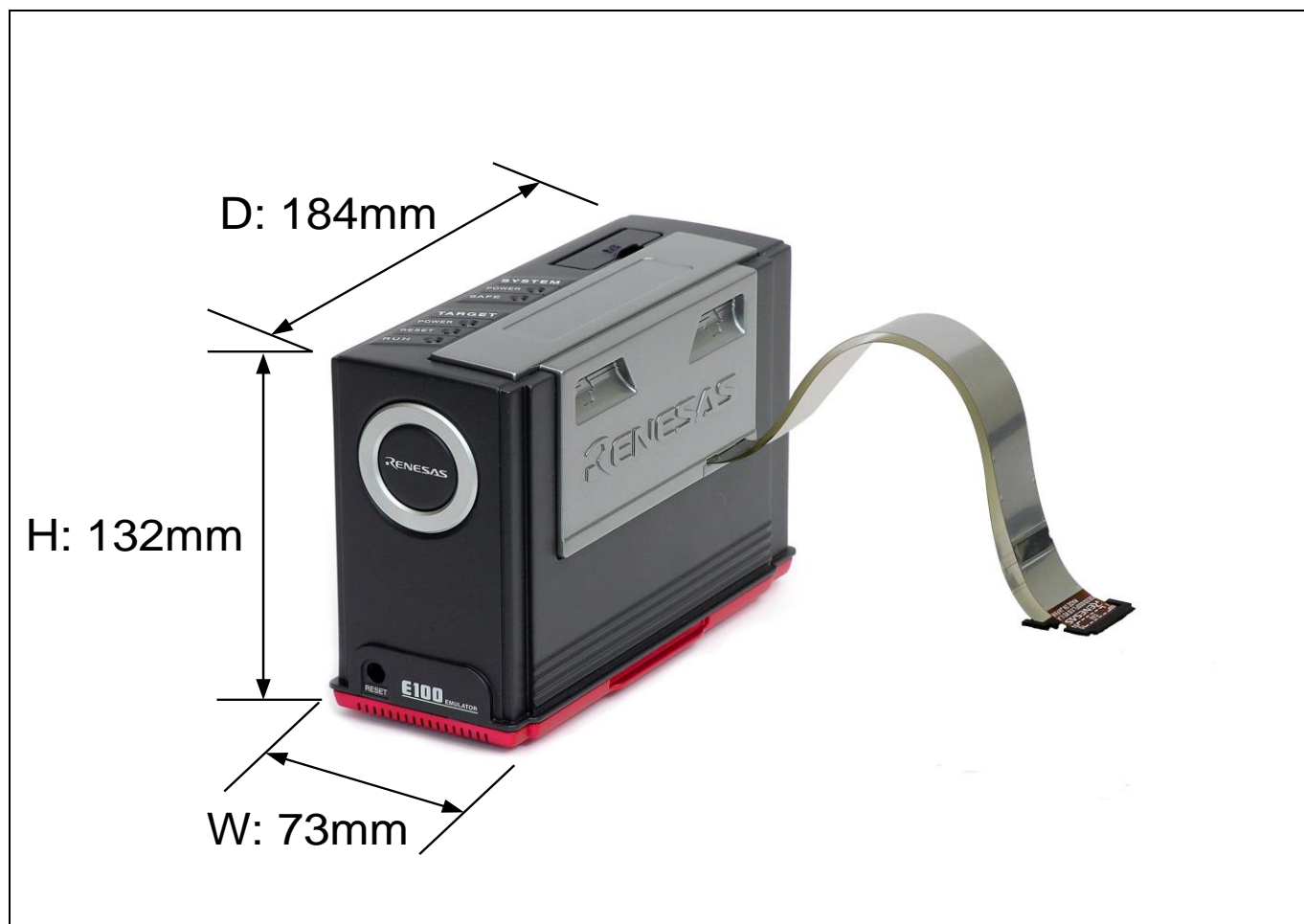


Figure 7.2 External Dimensions of the E100 Emulator

7.4.2 External Dimensions of the Converter Board R0E556100CFK00

Figure 7.3 shows external dimensions and a sample foot pattern of the converter board R0E556100CFK00 for a 144-pin 0.5-mm pitch LQFP.

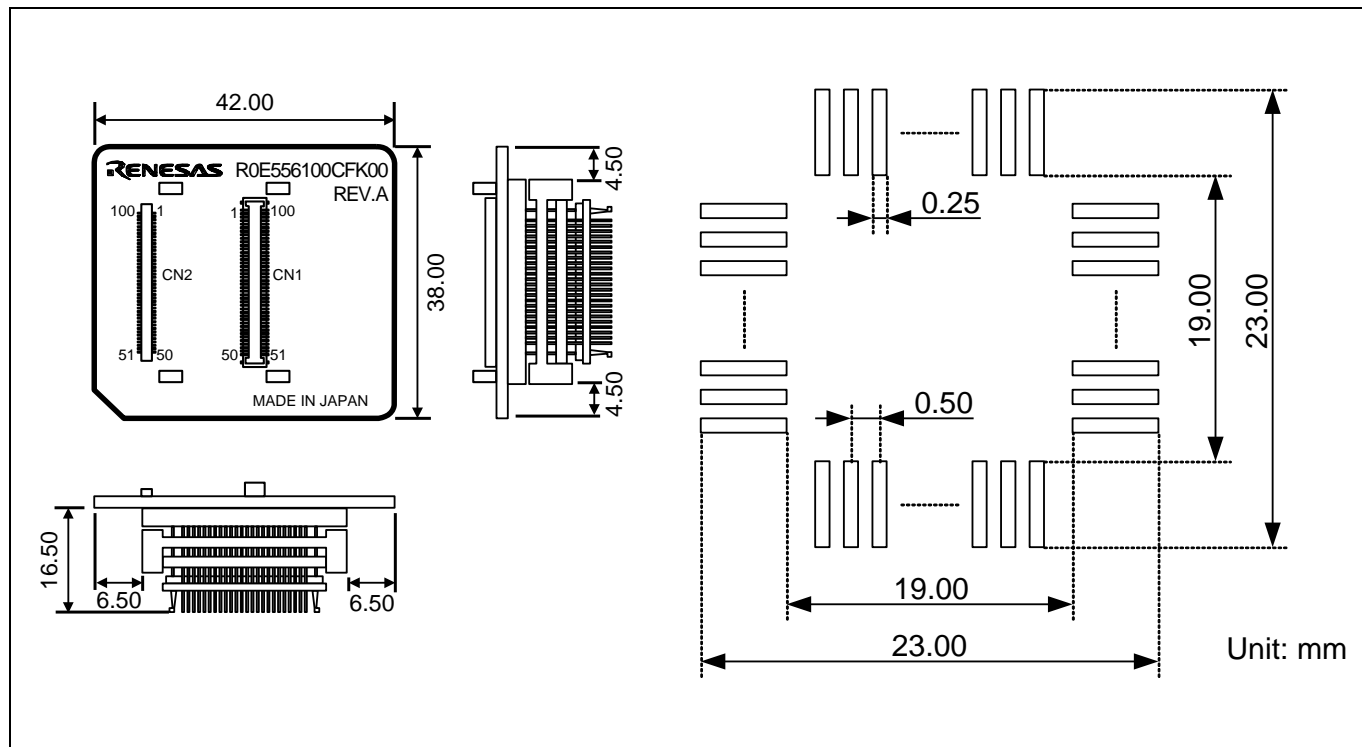


Figure 7.3 External Dimensions and a Sample Foot Pattern of R0E556100CFK00

7.5 Notes on Using the MCU Unit

Notes on using the MCU unit are listed below. When you debug an MCU using the MCU unit, be careful about the following precautions.

IMPORTANT

Note on the Version of the Emulator Debugger:

Be sure to use this product with the following emulator debugger.

- RX610 E100 Emulator Software V.1.00 Release 00 or later

Notes on Downloading Firmware:

Before using this product for the first time, it is necessary to download the dedicated firmware (emulator's control software installed in the flash memory in the E100). If you need to download at debugger startup, a message will appear. Download the firmware following the message.

Do not shut off the power while downloading the firmware. If this happens, the product will not start up properly. If the power is shut off unexpectedly, re-download the firmware.

Disconnect the MCU unit from the user system before you start downloading the firmware.

Notes on Self-Checking:

If self-checking does not result normally (excluding user system errors), the product may be damaged. Then contact your local distributor.

Disconnect the MCU unit from the user system before you start self-checking.

Note on Quitting the Emulator Debugger:

To restart the emulator debugger, always shut off the emulator power supply and then turn on it again.

Note on Display of MCU Status:

"Status" you can view in the [Connecting] dialog box of the emulator debugger shows pin levels of the user system. Make sure that proper pin levels are selected according to the mode you use.

Note on Contention between Reset and Emulator Control:

The emulator system may become uncontrollable if there is a contention between resets (pin reset, watchdog timer reset, etc.) and operations by the emulator system (memory reference in the [Memory] window, etc.). When "A timeout error. The MCU is in the reset state. Is system reset issued?" is displayed, click on the [Yes] button to issue a system reset.

Note on On-Chip ROM Disabled Extended Mode:

After a transition to on-chip ROM disabled extended mode, the pin state and internal state reset by pin reset, watchdog timer reset, etc. are the same as those reset by pin reset.

IMPORTANT

Note on Clock Supply to the MCU:

A clock supplied to the evaluation MCU is selected by the [System] page in the [Configuration properties] dialog box of the emulator debugger.

(1) When "Emulator" is selected:

A clock generated by the oscillator circuit board on the MCU unit is supplied. It is continually supplied regardless of the status of the user system clock and that of the user program execution.

(2) When "User" is selected:

A clock generated by the oscillator in the user system is supplied. It depends on the status of the oscillation (on/off) of the user system.

(3) When "Generate" is selected:

A clock generated by the dedicated circuit in the E100 is supplied. It is continually supplied regardless of the status of the user system clock and that of the user program execution.

Note on Low-Power States:

Other than a stop request, you can make no command requests.

Note on the Watchdog Function:

If the reset circuit of the user system has a watchdog timer, disable it when using the emulator.

Note on Access Prohibited Area:

You cannot use internally reserved areas. If you attempt to write to this area in the Memory or other windows, an error message is displayed and values read will be undefined.

Note on Data Flash:

During the execution of the user program, no reference or rewrite to the data flash is possible in the [Memory] windows, etc. If you attempt to write to this area, an error message is displayed. If you attempt to read from the block where read is disabled by the data flash read enable register, a warning is displayed and the values read will be undefined.

When a program is downloaded or rewrite is executed in the [Memory] window, etc. data are written in 256-byte units by the emulator debugger. The area in which the data are written is no longer an erased area.

Note on Breaks:

The following break functions are available in the emulator debugger.

(1) Software break

This is a debugging function which generates an interruption by changing an instruction at a specified address to a dedicated instruction for use with the emulator to break a program immediately before the system executes an instruction at a specified address. The instruction at the preset address will not be executed.

(2) On-chip break

This is a debugging function which uses the on-chip break function of the evaluation MCU to break a program immediately before the system executes an instruction at a specified address. The instruction at the preset address will not be executed.

(3) Hardware break

This is a debugging function which breaks a program by setting the detection of an execution of an instruction at a specified address as a break event. The program will break after the instruction at the specified address is executed.

(4) Exceptional event

This is a debugging function which stops a program by an abnormal operation of the user program or overflow of each function's measurement counter, etc.

IMPORTANT

Notes on Software Breaks:

When setting a software breakpoint in the RAM space, set the registers as follows.

- (1) Set the RAM enable bit of the system control register 1 to enabled.
- (2) Set the RAM0 module stop bit and RAM1 module stop bit of the module stop control register C to RAM run.

Notes on Power Supply to the User System:

Pin Vcc is connected to the user system to observe the voltage. Therefore, the power is not supplied to the user system from the emulator. Design your system so that the user system is powered separately.

The voltage of the user system should be as follows.

$$3.0\text{ V} \leq V_{cc} \leq 3.6\text{ V}$$

Notes on Internal Flash ROM of the MCU:

Because the number of write/erase cycles of the internal flash ROM of the MCU is limited, it must be replaced at the end of its service-life.

If the following errors occur frequently when downloading a program, contact your local distributor.

- (1) An error has occurred in erasing of the flash ROM in the target MCU.
- (2) An error has occurred in programming of the flash ROM in the target MCU.
- (3) An error has occurred in verification of the flash ROM in the target MCU.

Notes on the DMAC and DTC:

When a DMAC or DTC request is generated to the MCU's internal flash ROM (source from which transferred) while the user program is stopped, do not perform the following operations:

- (1) Setting a software break in the MCU's internal flash ROM area
- (2) Writing from the [Memory] or [Command Line] window to the MCU's internal flash ROM area
- (3) Downloading into the MCU's internal flash ROM area

Do not access the work area set up on the [System] page of the [Configuration properties] dialog box by DMAC and DTC, otherwise the emulator debugger becomes unable to control the emulator.

Notes on Debug in ROM P/E and Data Flash P/E Modes:

Do not stop the program while operating in the ROM P/E or data flash P/E mode.

And do not single step an instruction shifting to ROM P/E or data flash P/E mode.

This is because, when in the ROM P/E or data flash P/E mode, the emulator debugger becomes unable to control the emulator.

To check the data after executing write/erase operations, stop the program in other than the rewrite control program area and use the [Memory] window, etc. to check.

Notes on Downloading to the External Flash Memory:

Only flash memory with 4096 or fewer sectors can be registered.

Rewrite to the external flash memory can be executed only by downloading and not by using the [Memory] window.

If you are using external RAM as work RAM when downloading to the external flash memory, set the external RAM to the same endian as the CPU.

The emulator debugger accesses the areas registered in the [External Flash Memory] page of the [Configuration properties] dialog box as an external flash memory. To download a program into the internal flash ROM to any addresses from FFE00000h to FFFFFFFFh in on-chip ROM disabled extended mode, delete the external flash area overlapped with the area in the internal flash ROM from the [External Flash Memory] page.

8. Maintenance and Warranty

This chapter covers basic maintenance, warranty information, provisions for repair and the procedures for requesting a repair.

8.1 User Registration

When you purchase our product, be sure to register as a user. For user registration, refer to "User Registration" (page 15) of this user's manual.

8.2 Maintenance

- (1) If dust or dirt collects anywhere on your emulation system, wipe it off with a dry soft cloth. Do not use thinner or other solvents because these chemicals can cause the equipment's surface coating to separate.
- (2) When you do not use the MCU unit for a long period, for safety purposes, disconnect the power cable from the power supply.

8.3 Warranty

If your product becomes faulty within one year after purchase while being used under conditions of observance of the "IMPORTANT" and "Precautions for Safety" notes in this user's manual, we will repair or replace your faulty product free of charge. Note, however, that if your product's fault is due to any of the following causes, an extra charge will apply to our repair or replacement of the product.

- Misuse, abuse, or use under extraordinary conditions
- Unauthorized repair, remodeling, maintenance, and so on
- Inadequate user system or improper use of the user system
- Fires, earthquakes, and other unexpected disasters

In the above cases, contact your local distributor. If your product is being leased, consult the leasing company or the owner.

8.4 Repair Provisions

(1) Repairs not covered by warranty

Problems arising in products for which more than one year has elapsed since purchase are not covered by warranty.

(2) Replacement not covered by warranty

If your product's fault falls into any of the following categories, the fault will be corrected by replacing the entire product instead of repairing it, or you will be advised to purchase a new product, depending on the severity of the fault.

- Faulty or broken mechanical portions
- Flaws, separation, or rust in coated or plated portions
- Flaws or cracks in plastic portions
- Faults or breakage caused by improper use or unauthorized repair or modification
- Heavily damaged electric circuits due to overvoltage, overcurrent or shorting of power supply
- Cracks in the printed circuit board or burnt-down patterns
- A wide range of faults that make replacement less expensive than repair
- Faults that are not locatable or identifiable

(3) Expiration of the repair period

When a period of one year has elapsed after production of a given model ceased, repairing products of that model may become impossible.

(4) Carriage fees for sending your product to be repaired

Carriage fees for sending your product to us for repair are at your own expense.

8.5 How to Make Request for Repair

If your product is found faulty, fill in a Repair Request Sheet downloadable from the following URL. And email the sheet and send the product to your local distributor.

<https://www.renesas.com/repair>

CAUTION

Note on Transporting the Product:



When sending your product for repair, use the packing box and cushioning material supplied with the MCU unit when it was delivered to you and specify caution in handling (handling as precision equipment). If packing of your product is not complete, it may be damaged during transportation. When you pack your product in a bag, make sure to use the conductive plastic bag supplied with the MCU unit (usually a blue bag). If you use a different bag, it may lead to further trouble with your product due to static electricity.

Revision History	R0E556100MCU00 User's Manual
------------------	------------------------------

Rev.	Date	Description	
		Page	Summary
1.02	Dec.01.15	3	Regulatory Compliance Notices was changed.
1.03	Sep.01.21	—	The statement of inclusion of an AC adapter was removed.
		4	Regulatory Compliance Notices was changed.
		23	The specifications of the AC adapter were added.

R0E556100MCU00 User's Manual
E100 Emulator Main Unit for RX610 Group

Publication Date: Rev.1.03 Sep.01.21

Published by: Renesas Electronics Corporation
