User's Manual

R0E417250MCU00

User's Manual

E100 Emulator MCU Unit for H8SX/1700 Series

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

Renesas Electronics
www.renesas.com

Rev.3.01 Dec 2015

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
3. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.
5. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The recommended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.

   “Standard”: Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots etc.

   “High Quality”: Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; and safety equipment etc.

   Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (nuclear reactor control systems, military equipment etc.). You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application for which it is not intended. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.
6. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You should not use Renesas Electronics products or technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. When exporting the Renesas Electronics products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document, Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.
11. This document may not be reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

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

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

(2012.4)

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 55022 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 ser/operator of the equipment may be required to take appropriate countermeasures under his responsibility.

EN 55024

・Information for traceability

・Authorised representative

Name: Renesas Electronics Corporation
Address: Toyosu Foresia, 3-2-24, Toyosu, Koto-ku, Tokyo 135-0061, Japan

・Manufacturer

Name: Renesas System Design Co., Ltd.
Address: 5-20-1, Josuihon-cho, Kodaira-shi, Tokyo 187-8588, 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: R0E417250MCU00

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 R0E417250MCU00 is a full-spec emulator for MCUs of the H8SX/1700 Series. This user’s manual mainly describes specifications of the R0E417250MCU00 and how to set it up.

All components of the R0E417250MCU00 are listed under “1.1 Package Components” (page 17). If you have any questions about the R0E417250MCU00, contact your local distributor.

The manuals relevant to usage of the R0E417250MCU00 are listed below. You can download the latest manuals from the Renesas Tools homepage (http://www.renesas.com/tools).

Related manuals

Item	Manual
Accessory	R0E0100TNPFK00 User’s Manual
Integrated development environment	High-performance Embedded Workshop User’s Manual
C/C++ compiler and assembler	H8S, H8/300 Series C/C++ Compiler, Assembler, Optimizing Linkage Editor Compiler Package User’s Manual
	Notes on Usage of the C/C++ Compiler Package for H8SX, H8S, H8 Family and Corrections in the 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 H8SX family H8SX/1700 series 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**	**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**	**WARNING** indicates a potentially dangerous situation that will cause death or heavy wound unless it is avoided.
⚠ **CAUTION**	**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 attached 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.
- Use an AC power cable which complies with the safety standard of the country.
- 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

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.

Contents

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 (R0E417250MCU00)
This means the E100 emulator for the H8SX/1700 Series.

Emulator system
This means an emulator system built around the MCU unit (R0E417250MCU00). The emulator system is configured with an emulator main unit (R0E001000EMU00), MCU unit (R0E417250MCU00), emulator power supply, USB cable, emulator debugger and host machine.

Integrated development environment: High-performance Embedded Workshop
The High-performance Embedded Workshop (HEW) provides a GUI-based integrated development environment for the development and debugging of embedded applications for Renesas microcontrollers.

HEW, a powerful yet easy to use tool suite, features an industry standard user interface and is designed using a modular approach seamlessly incorporating device family-specific C/C++ compilers and the debugger elements for various debugging platforms including emulators and evaluation boards.

This provides the user with a single interface to fully exploit the advanced capabilities of the development tools for the entire development cycle from evaluation of a device through to completion of code development.

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 R0E417250MCU00 package consists of the following items. After you have unpacked the box, check if your R0E417250MCU00 contains all of these items.

Table 1.1 Package components

Item		Quantity
R0E417250MCU00 MCU board		1
Oscillator module (8 MHz) mounted on the IC21 socket		1
R0E001000FLX10 flexible cable		2
R0E417250MCU00 Release Notes (English)		1
R0E417250MCU00 Release Notes (Japanese)		1
Repair Request Sheet (English)		1
Repair Request Sheet (Japanese)		1
CD-ROM	- Emulator debugger (H8SX E100 emulator debugger) - User's Manual	1

* Please keep the R0E417250MCU00'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.

* 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 an H8SX-family H8SX/1700-series H8SX/1720-group MCU, 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
100-pin 0.5-mm pitch LQFP (PLQP0100KB-A Former code: LQFP-100)	R0E0100TNPFK00

* 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.

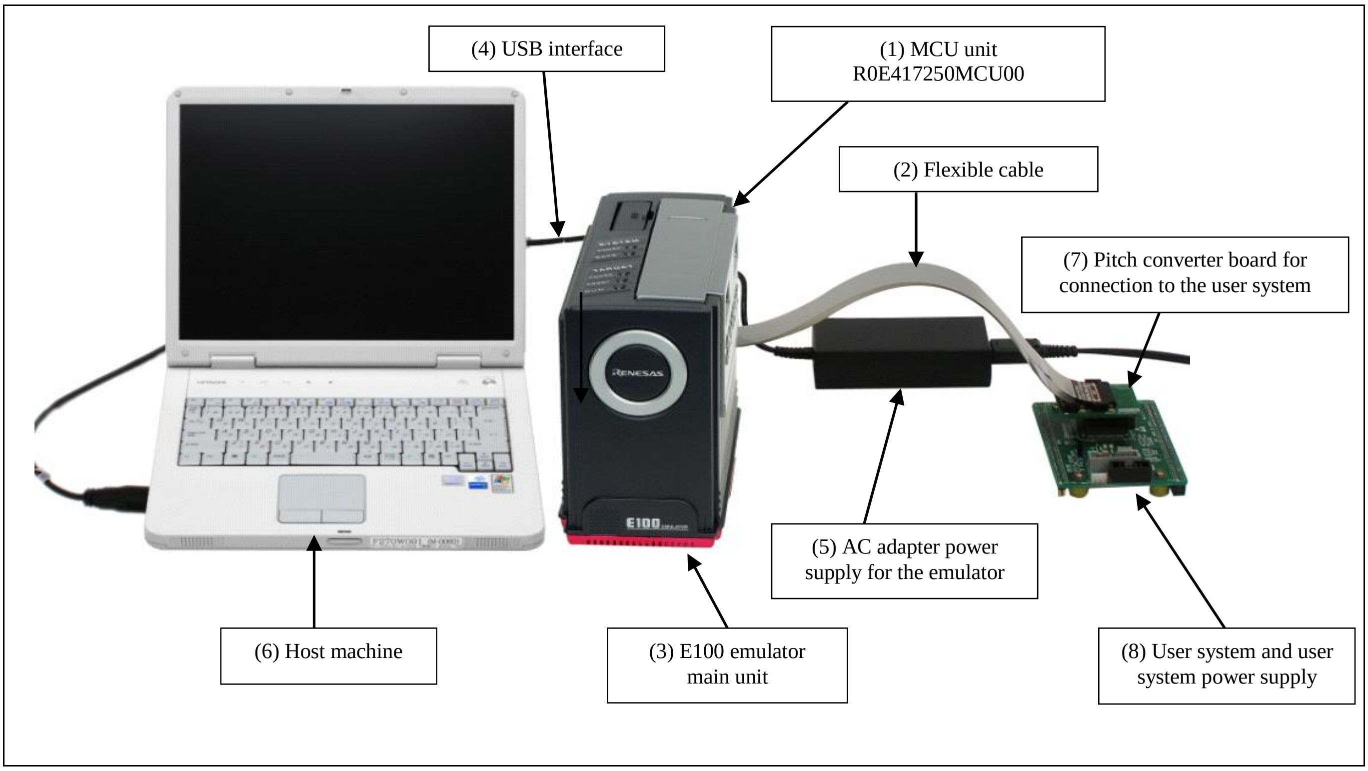


Figure 1.1 System configuration

(1) MCU unit R0E417250MCU00 (this product)
This is an MCU board for the H8SX/1700 Series MCUs 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 R0E0100TNPFK00 for connecting the user system
(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 the Emulator Parts

Figure 1.2 shows the names of the emulator parts.

Figure 1.2 Names of the emulator parts

(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 E100 emulator's power supply, operating state of firmware, etc. Table 1.3 lists the definitions of the system status LEDs.

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)	Self-checking is in progress.
	OFF	Emulator system is not operating normally (system status error).

(6) Target Status LEDs

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

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 R0E417250MCU00.

Table 1.5 Specifications of the R0E417250MCU00

Applicable MCU	H8SX-family H8SX/1700-series MCUs	
Applicable MCU mode	Single-chip mode, On-chip ROM enabled extended mode	
Maximum ROM/RAM capacity	1. Internal flash ROM: 1 Mbytes 2. Internal RAM: 64 Kbytes 3. Internal EEPROM: 32 Kbytes	
Maximum operating frequency	80 MHz	
Power supply voltage	3.0 to 3.6V, 4.5 to 5.5V	
Software break	4096 points	
Hardware break	16 points (Execution address, bus detection, interrupt, external trigger signal)	
Combination, pass count	- Cumulative AND/OR/simultaneous AND/state transition - 255 pass counts	
Detection of exceptional events	Violation of access protection/task stack access violation/OS dispatch/reading from a non-initialized area	
Real-time tracing	192 bits × 4 M cycles (Address, data, status, CPU status, bus 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	- Extracting or deleting data by specifying events or extracting the instruction that accesses the specified data - Extracting data before and after trace points	
Real-time RAM monitor	- 16,384 bytes (512 bytes × 32 blocks) - Data/last access	
Time measurement	- 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: Equal to the MCU clock or 10ns to 1.6 us	
Coverage measurement	C0: 2 Mbytes (256 Kbytes × 8 blocks) C1: 1 Mbyte (128 Kbytes × 8 blocks)	
Emulation memory	4 Mbytes (1 Mbyte × 4 blocks)	
Pseudo-generation of errors	Detect if external oscillation has stopped and correct ROM/RAM ECC errors	
Connection to user system	100-pin 0.5-mm pitch LQFP	R0E0100TNPFK00
Emulator power supply	Supplied from included 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 [*1]
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	768 Mbytes or larger (more than 10 times the file size of the load module) 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®)

Item	Description
Host machine	IBM PC/AT compatible [*1]
OS	Windows Vista® 32-bit editions [*1] [*4]
CPU	Pentium 4 running at 3GHz or Core 2 Duo running at 1GHz or more recommended
Interface	USB 2.0 [*2]
Memory	1.5 Gbytes or larger (more than 10 times the file size of the load module) 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.

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 "6. Troubleshooting " (page 212).

1	Check the package components.	Refer to "1.1 Package Components" (page 17).
2	User registration	Refer to "User Registration" (page 15).
3	Install the included software, etc.	Install them from the included CD-ROM.
4	Connect the MCU unit to the E100 main unit.	Refer to "2.3 Connecting the MCU Unit to and Disconnecting it from the E100 Emulator Main Unit" (page 26)
5	Connect the host machine.	Connect the USB interface connector of the emulator and the USB port of the host machine.
6	Connect the emulator power supply.	Connect an emulator power supply to the power connector.
7	Turn on the power supply.	Turn ON the power to the emulator.
8	Check the LEDs on the emulator.	Check that the system status LEDs are lighting.
9	Try to use the emulator debugger according to the tutorial.	When using the emulator debugger for the first time, try to use the emulator functions, referring to "3. Tutorial" (page 36). If the emulator debugger does not start, follow steps 10 through 13 in this chart to perform self-checking.
10	Create a project in the High-performance Embedded Workshop.	Start up the High-performance Embedded Workshop and create a project by following "4. Preparation for Debugging" (page 67).
11	Start up the emulator debugger from the High-performance Embedded Workshop.	Start up the High-performance Embedded Workshop to launch the emulator debugger by following "4. Preparation for Debugging" (page 67).
12	Set up the operating environment of the emulator debugger.	Make settings on the Device page in the Device setting dialog box of the emulator debugger and select the "Start booting up on successful completion of self-checking" checkbox by following "4. Preparation for Debugging" (page 67).
13	Self-checking	

Figure 2.1 Flowchart of starting up the emulator (for the first time)

Step	Action	Description
14	Change the settings.	Change the settings of the power supply to MCUs or clock supply, according to the conditions on use.
15	Connect the user system.	Connect the user system as occasion demands.
16	Turn on the power supply.	Turn ON the power to the emulator and the user system as simultaneously as possible.
17	Check the LEDs on the emulator.	Check that system status LEDs, and POWER and RESET of the target status LED are lighting. When the user system is not connected, the POWER LED does not light up.
18	Start up the emulator debugger from the High-performance Embedded Workshop.	Start up the High-performance Embedded Workshop to launch the emulator debugger.
19	Set up the operating environment of the emulator debugger.	Check the contents set in step 12 in Figure 2.1. Do not select the "Start booting up on successful completion of self-checking" checkbox.
20	Debug a program with various functions of the emulator debugger	Refer to the High-performance Embedded Workshop and "5. Debugging Functions" (page 78).

Figure 2.2 Flowchart of starting up the emulator (after self-checking)

2.2 Installing the Included Software

If you have Windows Vista, XP or 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.

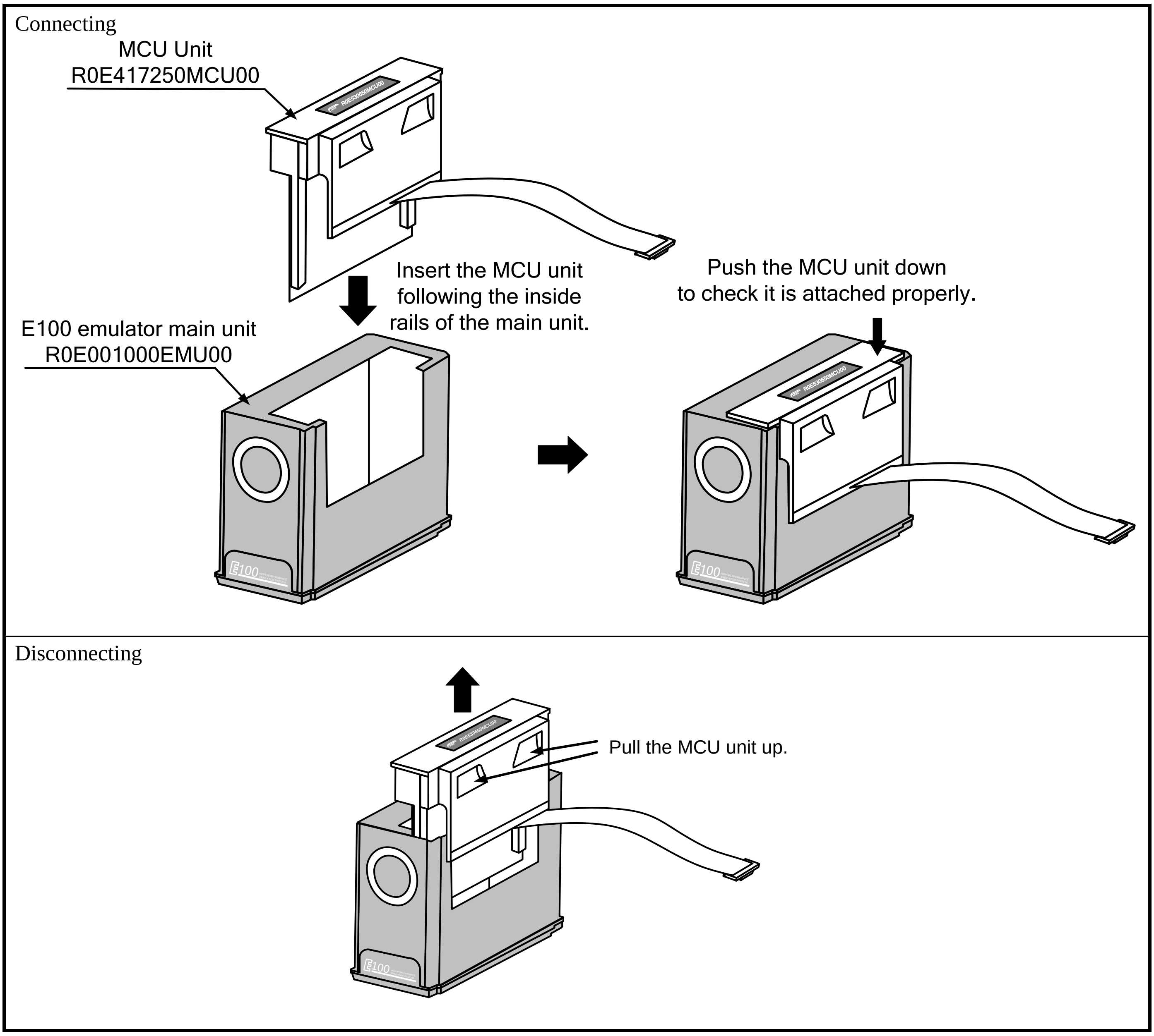


Figure 2.3 Connecting the MCU Unit to and Disconnecting it from the E100 Emulator Main Unit

⚠ 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 the included AC adapter to the emulator. The following shows how to connect the AC adapter.

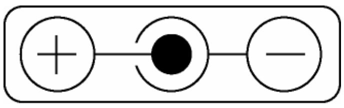
(1) Turn OFF the power of the emulator.
(2) Connect the DC cable of the AC adapter to the emulator.
(3) Connect the AC power cable to the AC adapter.
(4) Connect the AC power cable to the outlet.

Figure 2.5 Connecting the emulator power supply

⚠ CAUTION

Cautions for AC Adapter:

- Use only the AC adapter included in the E100 package.
- The included AC adapter is exclusively for the E100 emulator main unit. Do not use it for other products.
- Before installing this product or connecting it to other equipments, disconnect the AC power cable from the outlet to prevent injury or accident.
- The DC plug of the included AC adapter has the polarity shown below.



- The included AC adapter has no power switch. The AC adapter is always active while connected to the AC power cable.

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.
- Do not leave either the emulator or user system powered on. The internal circuits may be damaged due to leakage current.
- 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 V ≤ Vcc ≤ 3.6 V ,4.5 V ≤ Vcc ≤ 5.5 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 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 it.
(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.

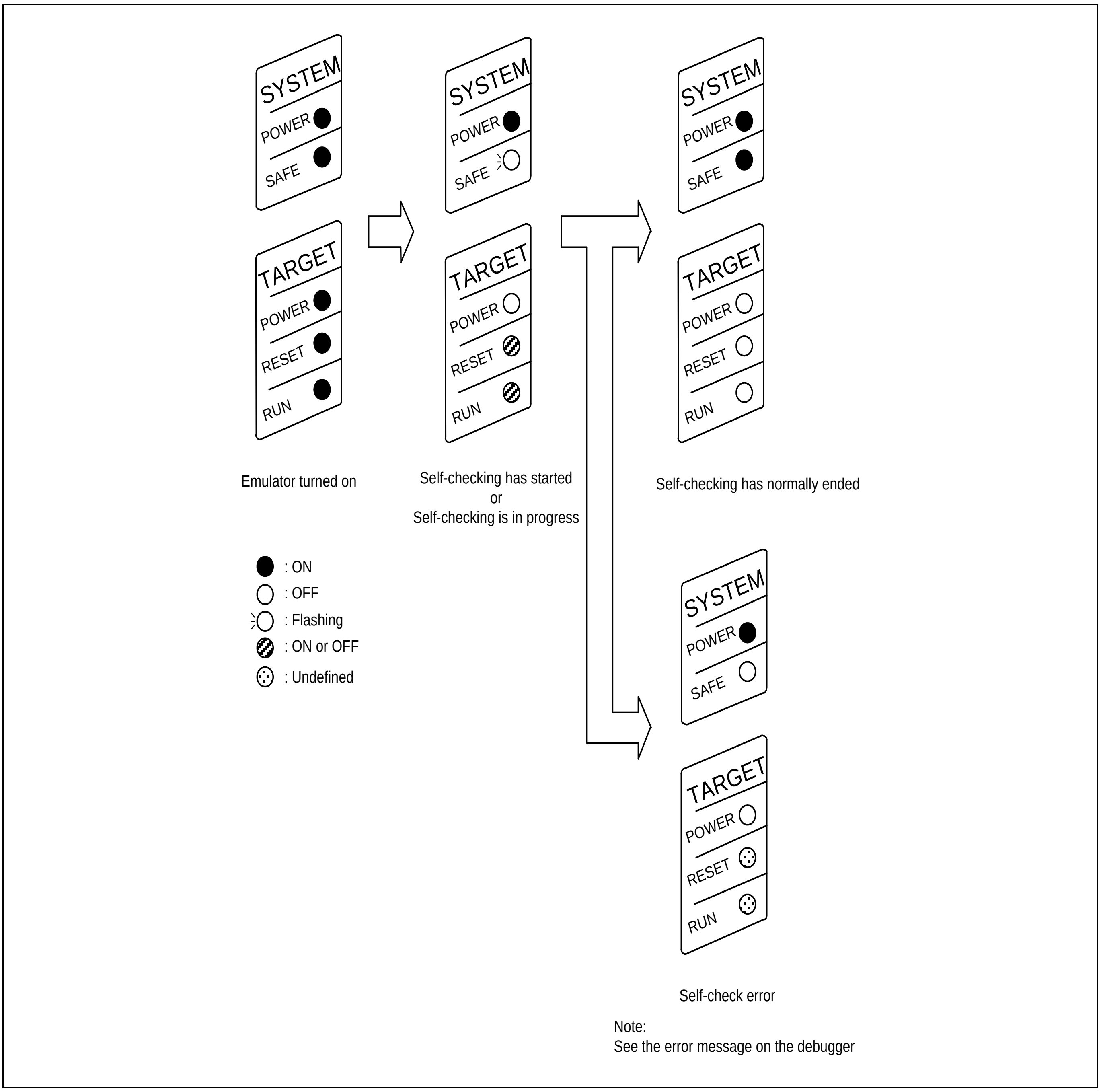


Figure 2.6 LEDs during self-checking

2.8 Selecting the Clock Supply

2.8.1 Clock Source

You can choose the clock source supplied to the evaluation MCU in the System page 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 (EXTAL)	Emulator	Oscillator module mounted on IC21	Yes
	User	Oscillator circuit on the user system	-
	Generate	Internal generator circuit (8.0 to 10.0 MHz)	-
Sub (OSC1 and OSC2)	Emulator	Internal oscillator circuit (32.768 kHz)	Yes

IMPORTANT

Notes on Changing the Clock Supply:

- The clock supply can be set on the System page of 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 (8 MHz) is mounted on IC21 at shipment of the R0E417250MCU00. If you wish to 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 on IC21 (see Figure 2.7).

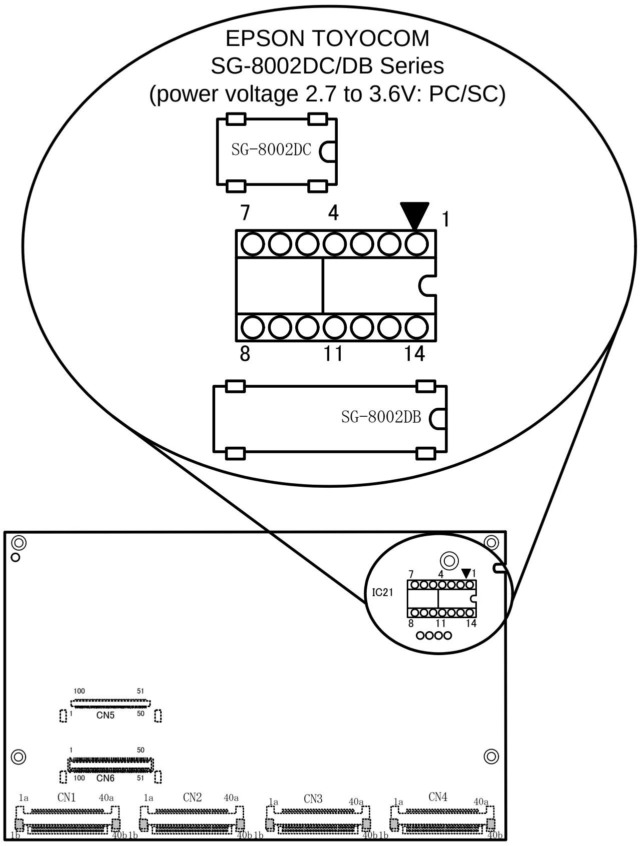


Figure 2.7 Replacing the oscillator module

⚠ CAUTION

Note 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 the MCU unit with an external clock source, 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. Pin XTAL, on the other hand, should be open. Choose "User" in the emulator debugger to use this clock source.

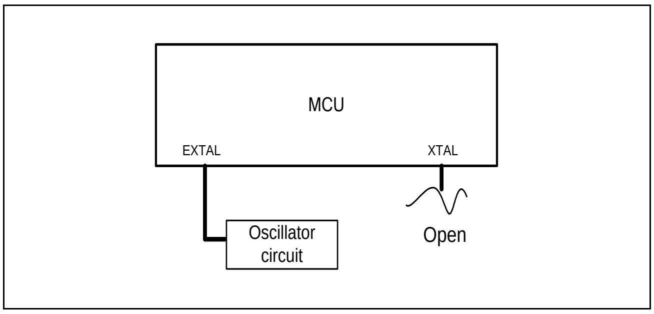


Figure 2.8 External oscillator circuit

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

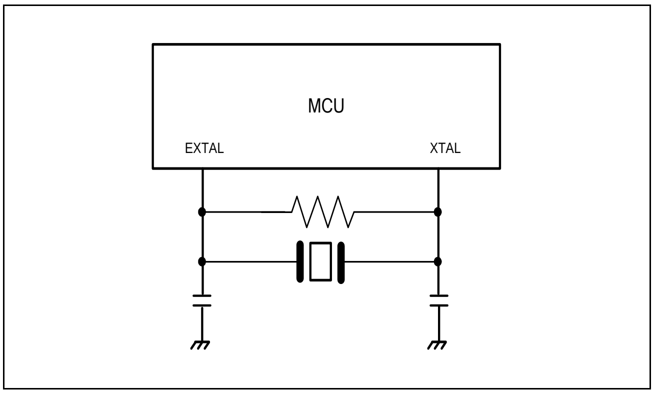


Figure 2.9 Circuit in which oscillation does not occur

2.8.4 Using the Internal Generator Circuit

The dedicated circuit in the E100 can generate clock source of any frequency specified in the emulator debugger, and it can be supplied as a main clock. It does not depend on 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 to generate a main clock, choose "Generate" in the emulator debugger and specify a frequency you like.

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 10 MHz for EXTAL of the MCU.

IMPORTANT

Note 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 (internal clock) for final evaluation purposes.

2.9 Connecting the User System

Figure 2.10 shows how to connect the MCU unit to your system.

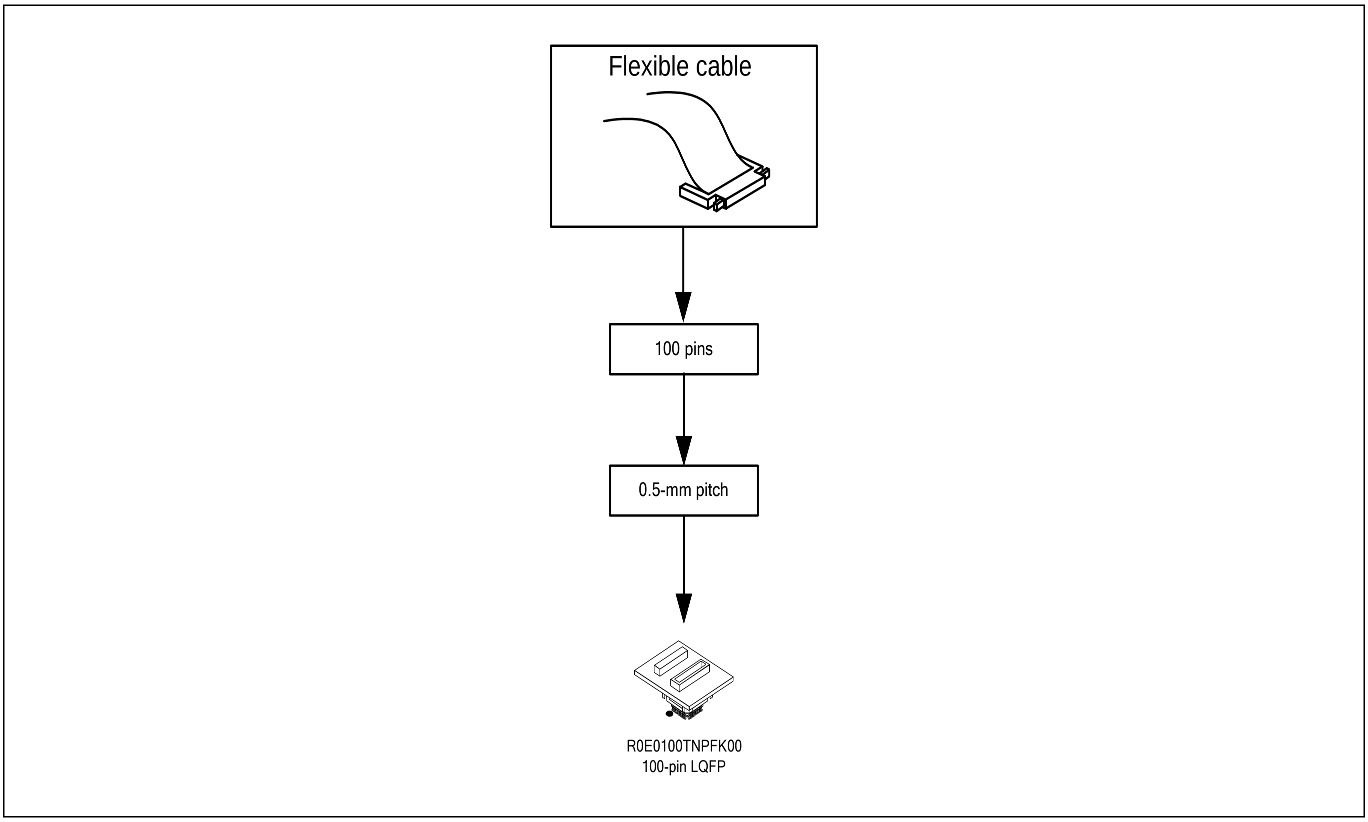


Figure 2.10 Connecting the MCU unit to the user system

⚠ 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 Connection to a 100-pin 0.5-mm Pitch Pad Pattern

The following is a procedure of connection to a 100-pin 0.5-mm pitch pad pattern on the user system using the R0E0100TNPFK00 (not included). For details on the R0E0100TNPFK00, refer to its user’s manual.

(1) Install the NQPACK100SD-ND, which comes with the R0E0100TNPFK00, on the user system.
(2) Connect the YQPACK100SD, which also comes with the R0E0100TNPFK00, to the NQPACK100SD-ND and secure it with the YQ-GUIDEs.
(3) Connect the R0E0100TNPFK00 to the YQPACK100SD.
(4) Connect CN2 of the R0E0100TNPFK00 to CN2 of the flexible cable.
(5) Connect CN1 of the R0E0100TNPFK00 to CN1 of the flexible cable.

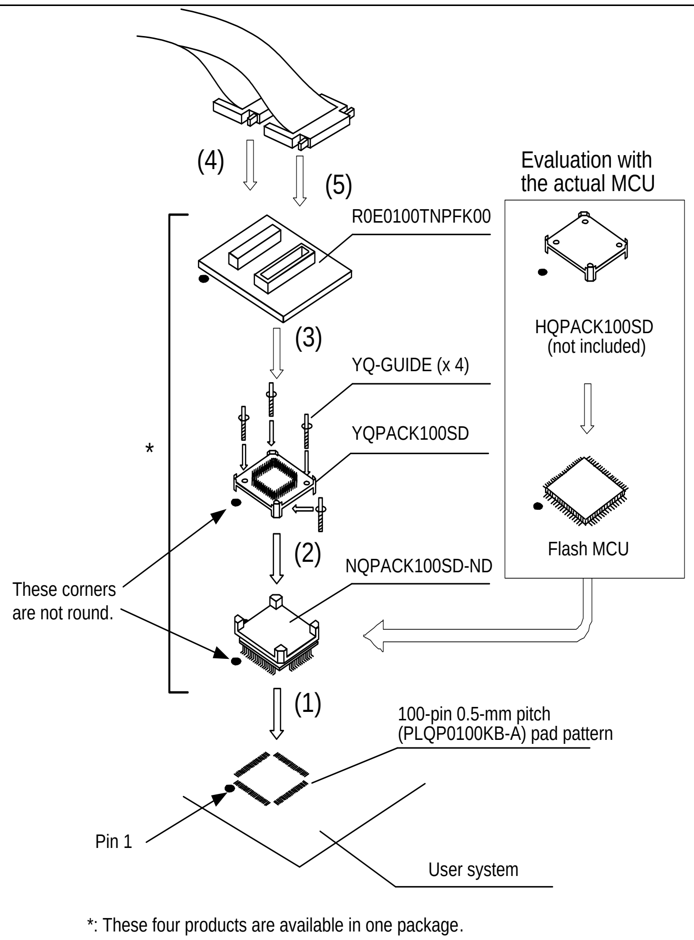


Figure 2.11 Connection to a 100-pin 0.5-mm pitch pad pattern

⚠ 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 R0E0100TNPFK00 are guaranteed for only 50 insertion/removal iterations.
- For purchasing the HQPACK100SD, contact the following:
  Tokyo Eletech Corporation http://www.tetc.co.jp/e_index.htm

3. Tutorial

3.1 Introduction

A tutorial program for the E100 emulator is provided as a means of presenting the emulator’s main features to you. The tutorial is described in this section.

The tutorial program was written in the C and C++ languages, and sorts random data (10 items) into ascending and descending order.
Processing by the tutorial program is as follows.

The main function repeatedly calls the tutorial function in order to repeatedly execute the process of sorting.
The tutorial function generates the random data to be sorted and calls the sort and change functions, in that order.
The sort function accepts input of an array that contains the random data generated by the tutorial function and sorts this data into ascending order.
The change function accepts input of the array that was sorted into ascending order by the sort function and sorts the data into descending order.

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

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

3.2 Starting the High-performance Embedded Workshop

Open a workspace by following the procedure described in Section 4.4, "Opening an Existing Workspace" (page 73).

Specify the directory given below.
(Drive where the OS is installed)\Workspace\Tutorial\E100\H8SX\Tutorial

Specify the file shown below.

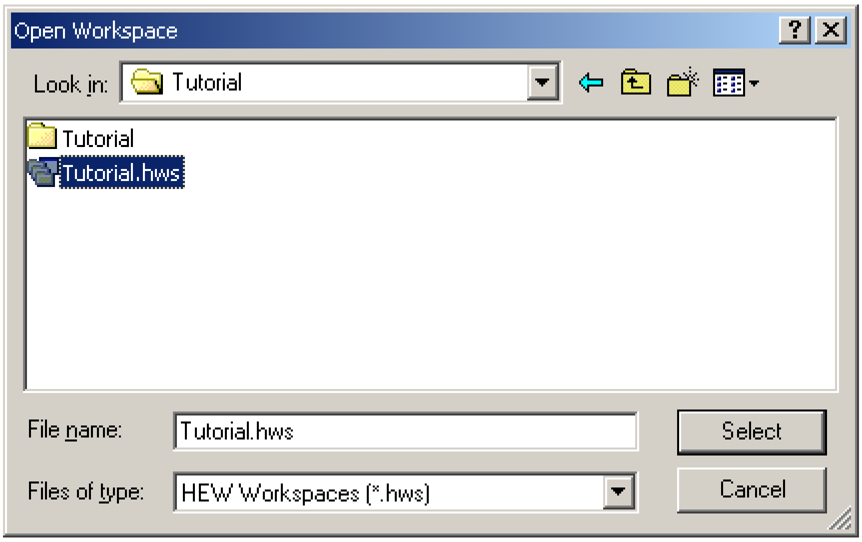


Figure 3.1 Open Workspace dialog box

3.3 Connecting the Emulator

When the debugger is connected to the emulator, a dialog box for setting up the debugger is displayed. Make initial settings of the debugger in this dialog box.
When you have finished setting up the debugger, you are ready to start debugging.

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 name of a program to be downloaded and the address where the program will be downloaded depend on the MCU in use. Accordingly, strings shown in the screen shots should be altered to those for the MCU in use.

Choose Download for Tutorial.abs under Download modules.

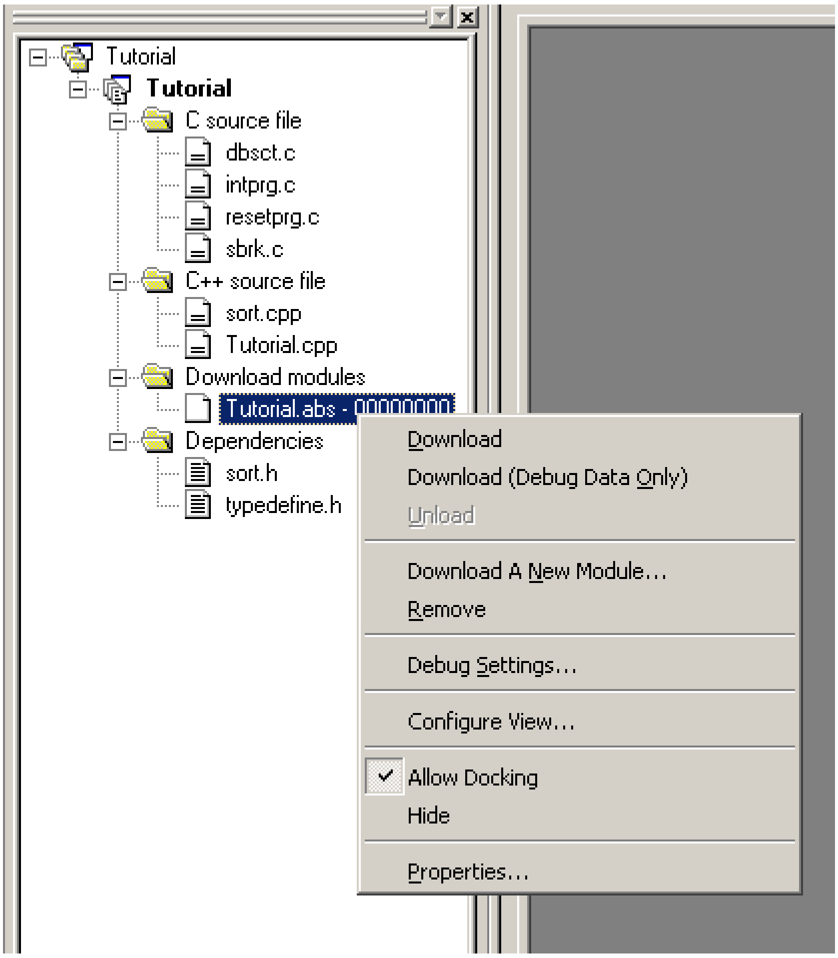


Figure 3.2 Downloading the tutorial program

3.4.2 Displaying the Source Program

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

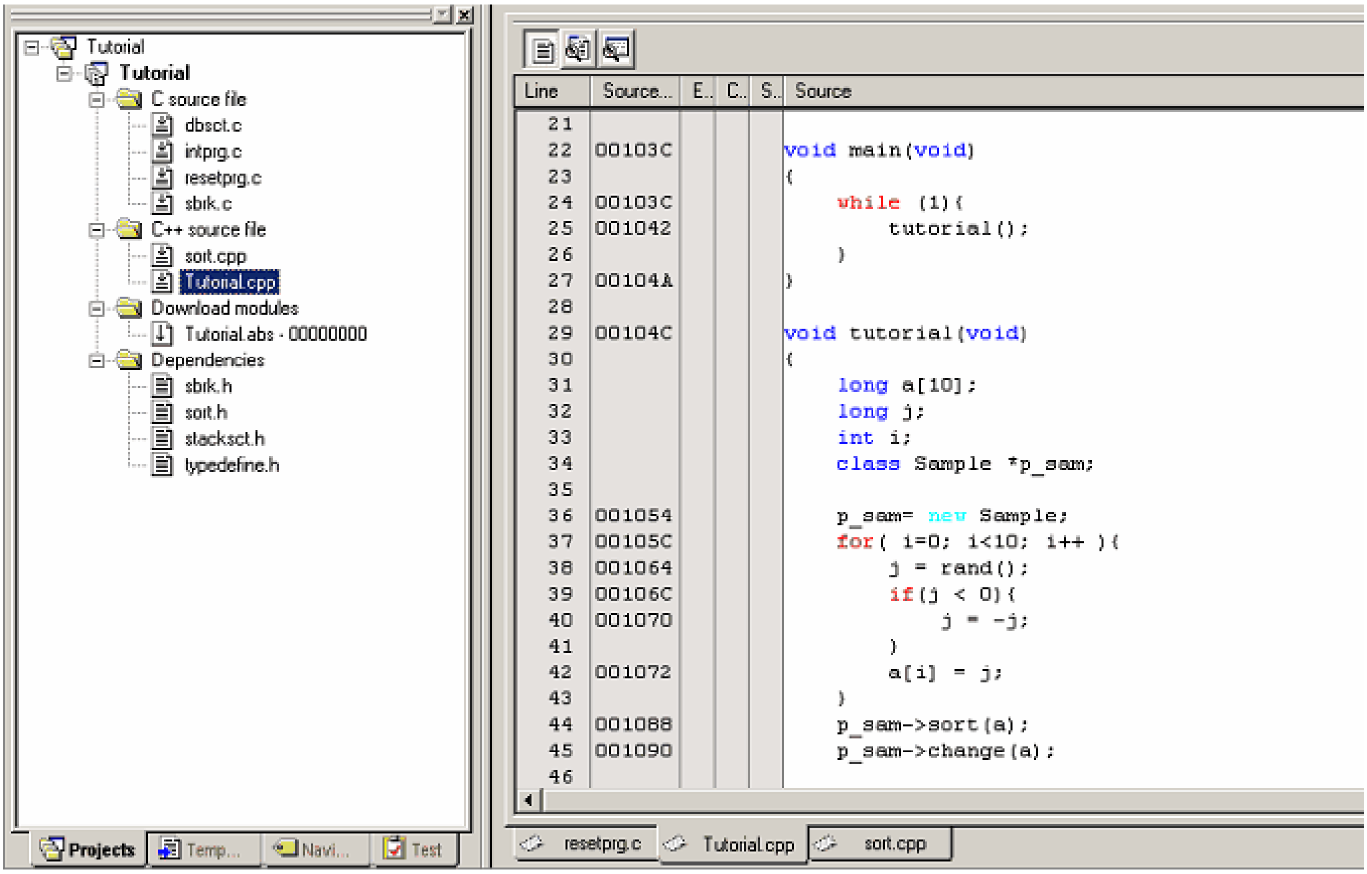


Figure 3.3 Editor window (displaying the source program)

If necessary, you can change the font and size to make the text more easily readable. For details, refer to the High-performance Embedded Workshop User's Manual.
The Editor window initially shows the beginning of the program. Use the scroll bar to view other parts of the program.

3.5 Setting Software Breakpoints

Setting of software breakpoints is one simple debugging facility.

Software breakpoints are easy to set in the Editor window. For example, you can set a software breakpoint at the line where the sort function is called.

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

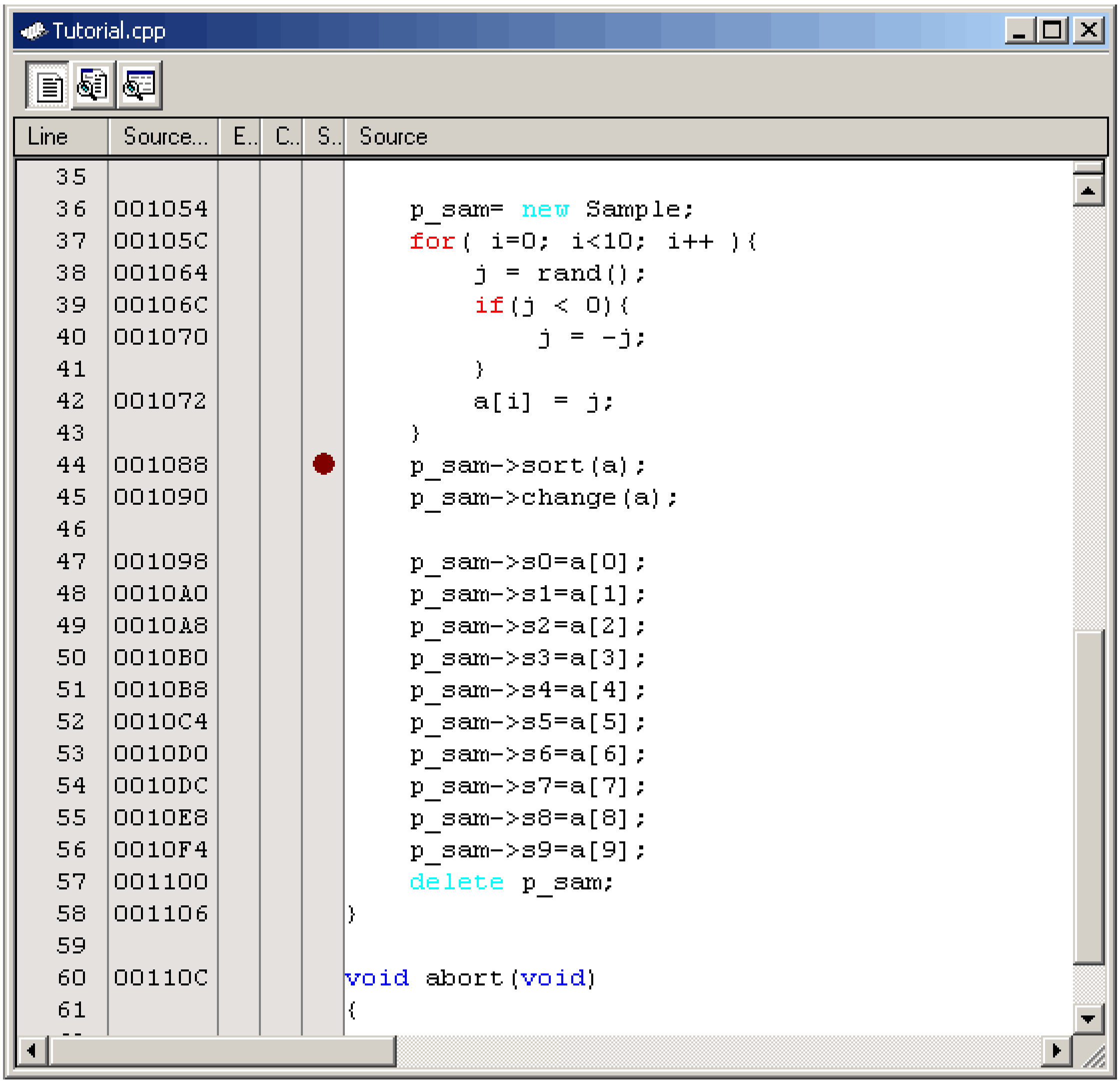


Figure 3.4 Editor window (setting a software breakpoint)

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 on the Reset CPU toolbar button [ ].

3.6.2 Executing the Program

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

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

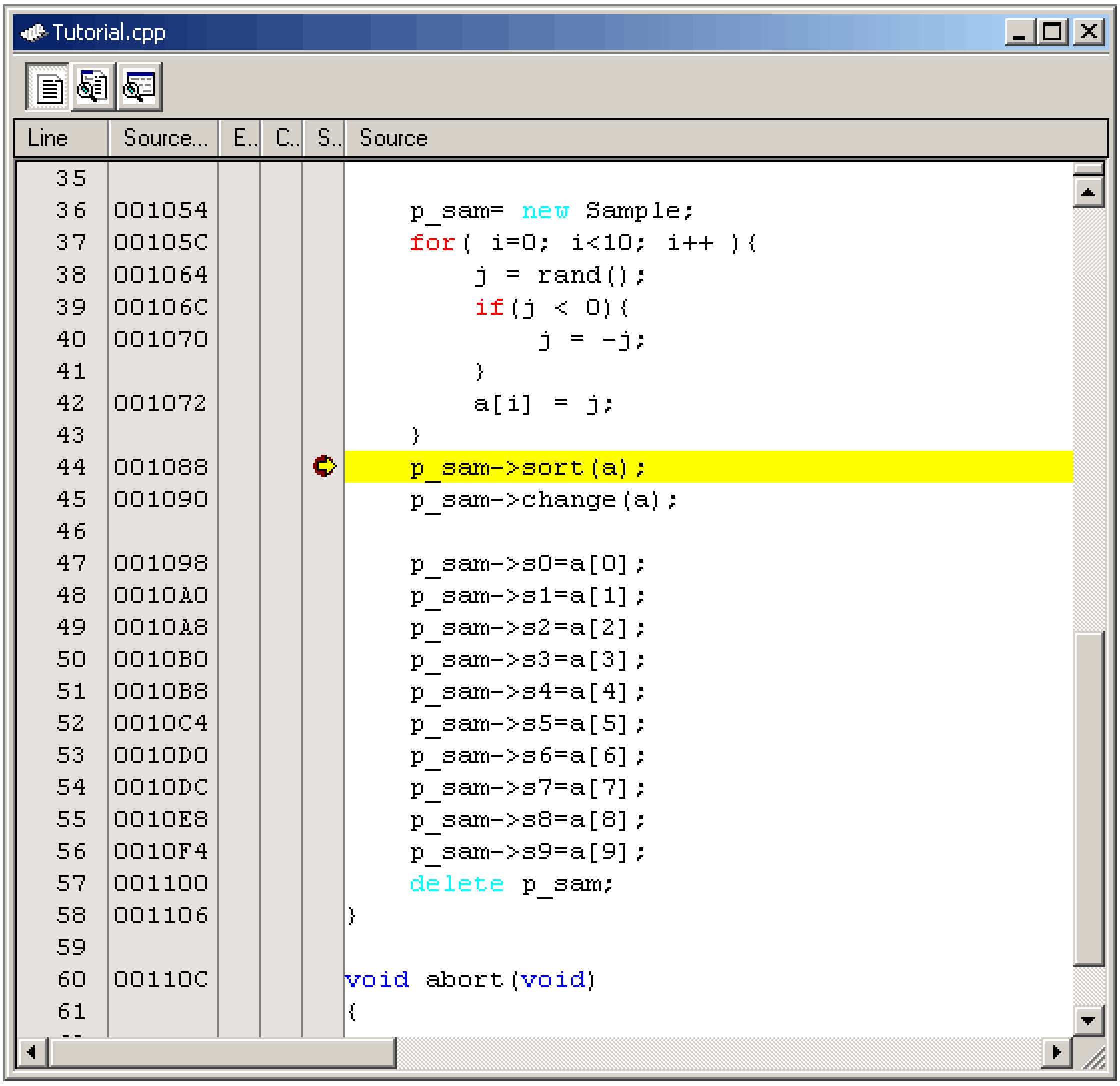


Figure 3.5 Editor window (break)

The Status window permits you to check the cause of the last break to have occurred.

Choose CPU –> Status from the View menu or click on the View Status toolbar button [ ]. When the Status window is displayed, open the Target sheet and check the cause of the break.

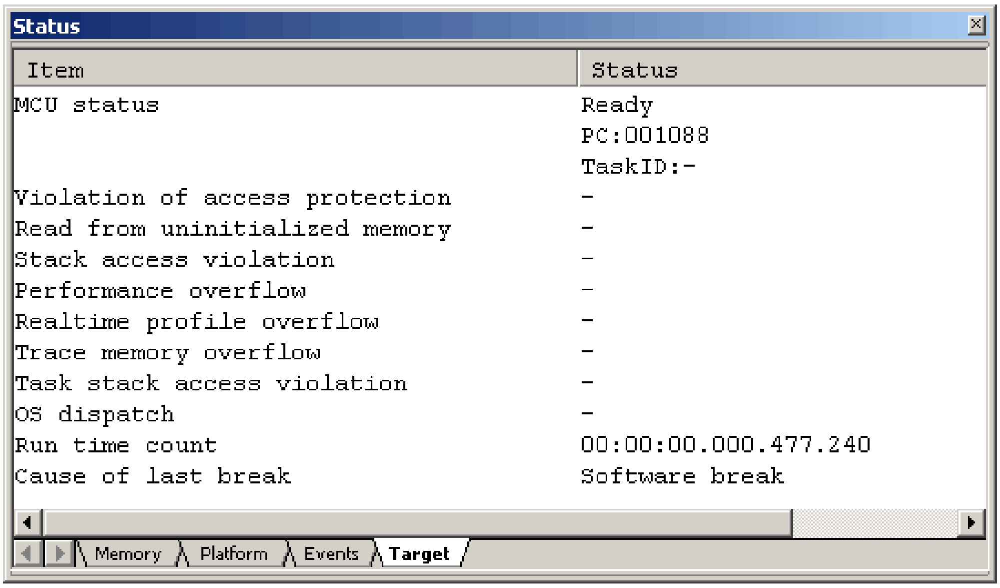


Figure 3.6 Status window

CAUTION

The contents displayed in this window differ with the product. For details of the contents displayed for particular products, refer to Chapter "5. Debugging Functions" (page 78) or the online help.

3.7 Checking Breakpoints

Use the Breakpoints dialog box to check all software breakpoints that have been 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.

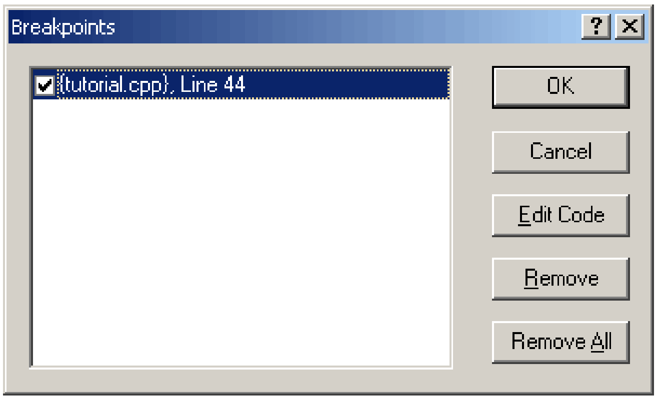
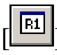


Figure 3.7 Breakpoints dialog box

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 on the Registers toolbar button []. The Register window shown below will be displayed.

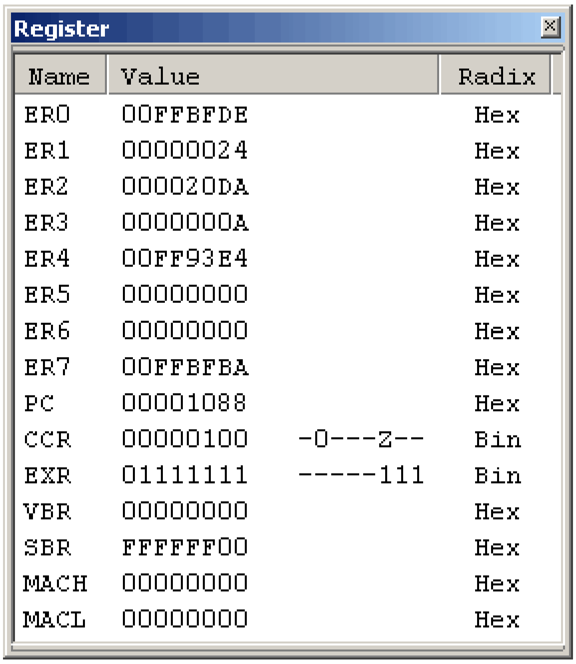


Figure 3.8 Register window

The contents of any register can be altered.

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

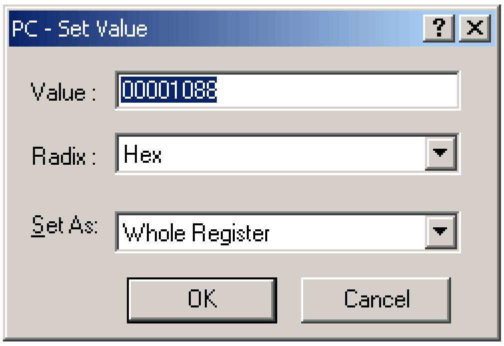


Figure 3.9 Set Value dialog box (PC)

3.9 Referring to Symbols

The Labels window permits you to view the symbolic information in a module.

Choose Symbol –> Labels from the View menu or click on the Labels toolbar button [ ]. The Labels window shown below will be displayed. Use this window to look at the symbolic information a module includes.

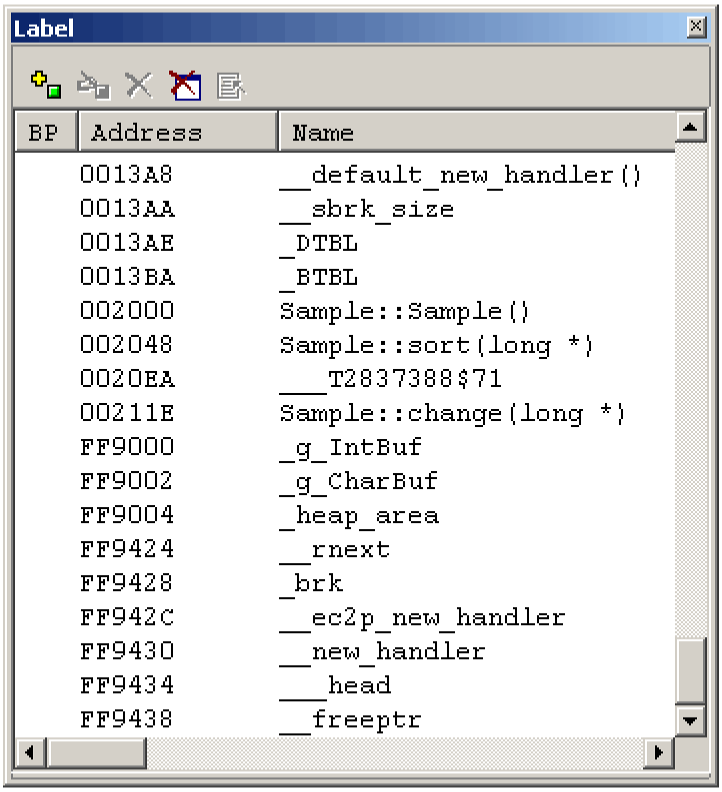


Figure 3.10 Label window

3.10 Checking Memory Contents

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

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

Enter "_main" in the edit box of the Display Address dialog box.

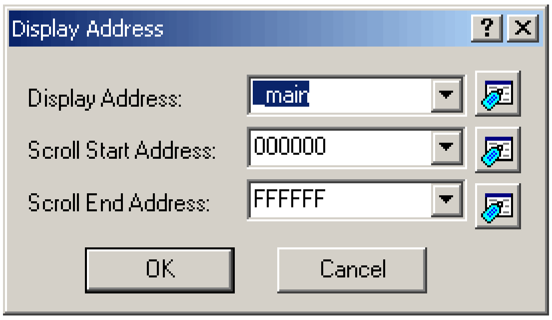


Figure 3.11 Display Address dialog box

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

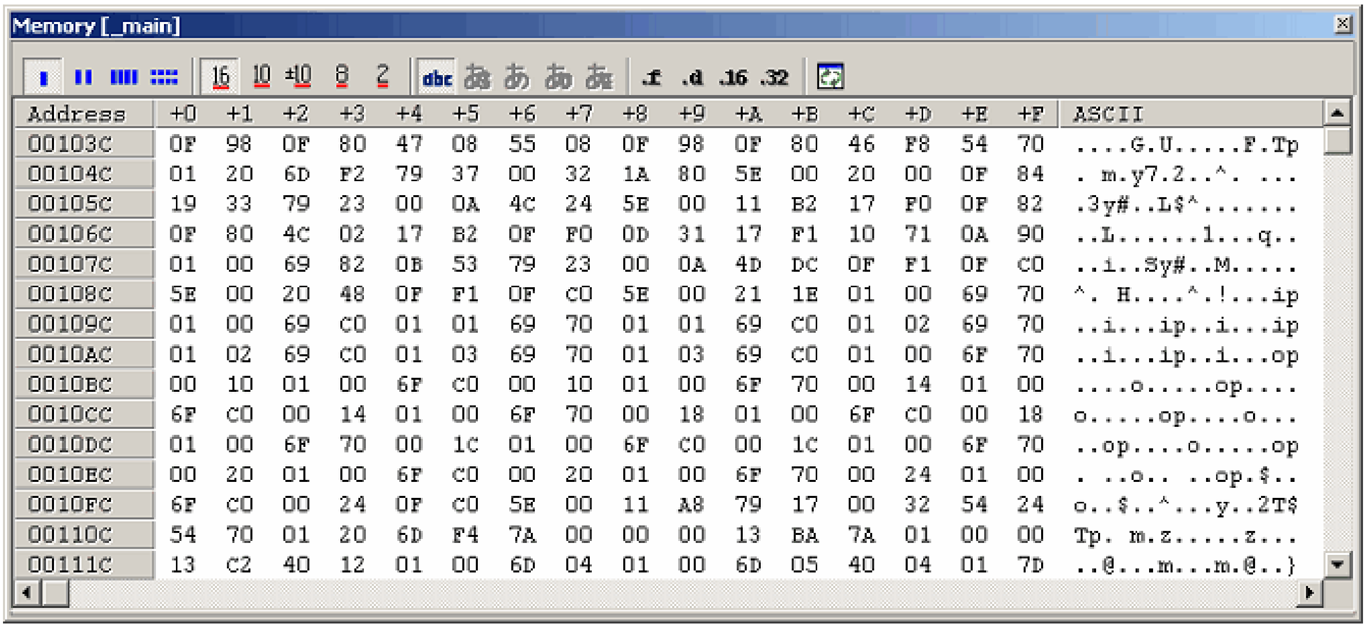


Figure 3.12 Memory window

3.11 Referring to Variables

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

Click on the left-hand side of the line containing the array 'a' in the Editor window to place the cursor there. Right-click and select Instant Watch. The dialog box shown below will be displayed.

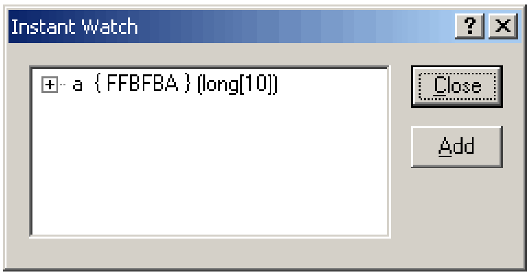


Figure 3.13 Instant Watch dialog box

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

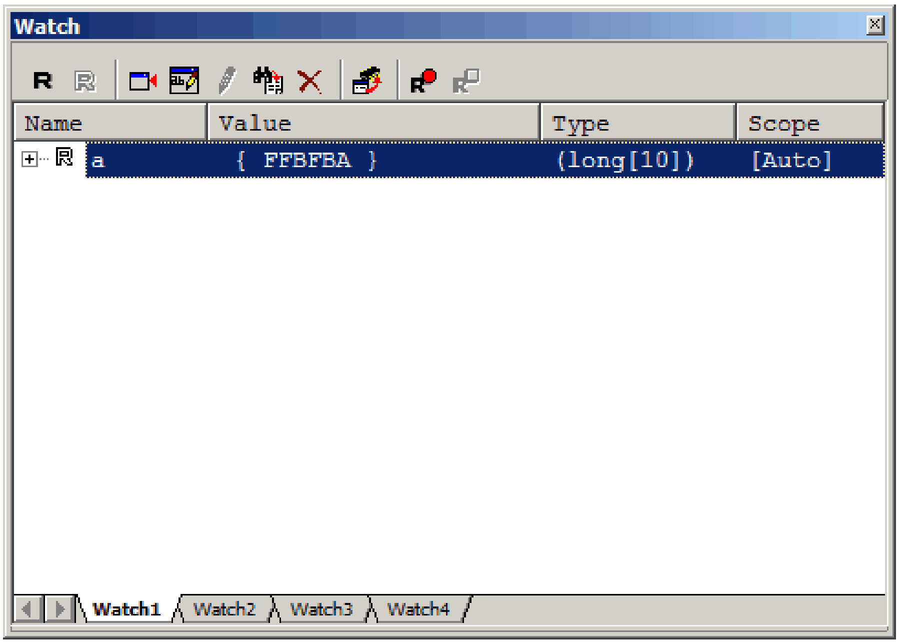


Figure 3.14 Watch window (array display)

Alternatively, you can specify a variable name to be added to the Watch window. Click the right mouse button in the Watch window and choose Add Watch from the popup menu. The dialog box shown below will be displayed.

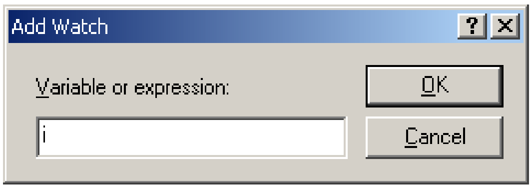


Figure 3.15 Add Watch dialog box

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

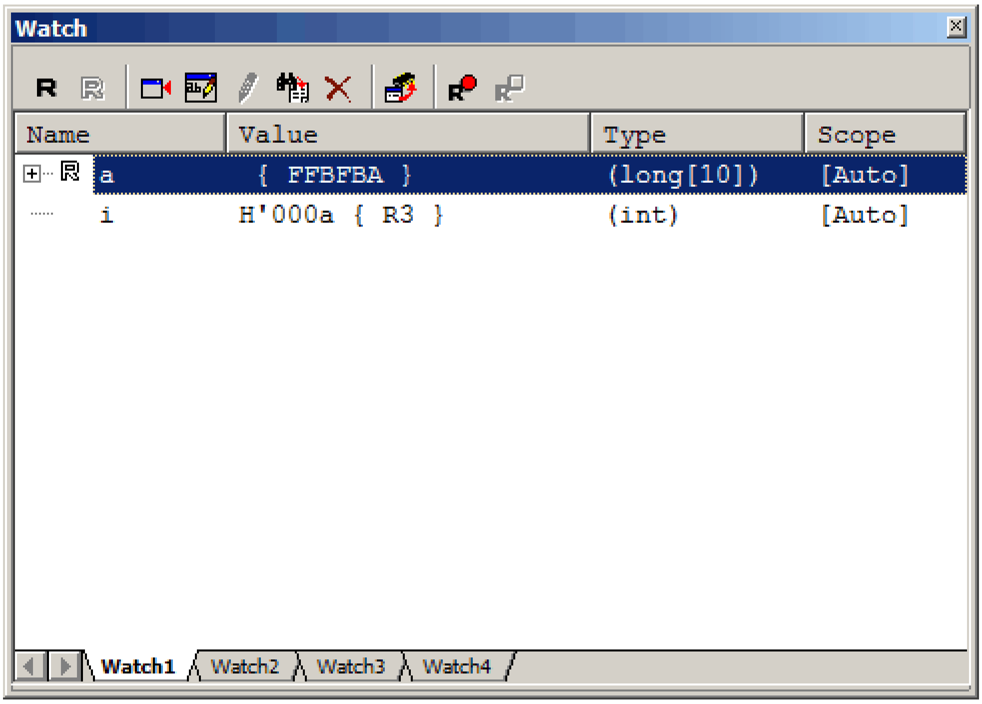


Figure 3.16 Watch window (showing a variable)

Click on the “+” mark shown to the left of the array ‘a’ in the Watch window. You can now look at the individual elements of the array ‘a.’

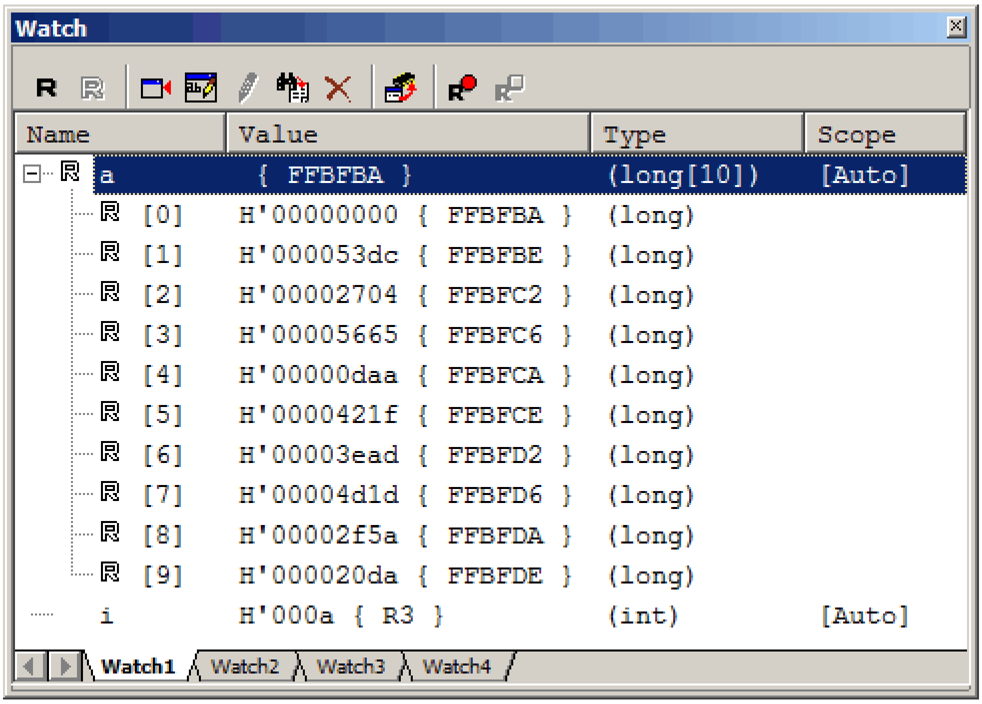


Figure 3.17 Watch window (showing array elements)

3.12 Showing Local Variables

By using the Local window, you can view the local variables included in a function. As an example, let’s check the local variables of the tutorial function. Four local variables are declared in this function: ‘a,’ ‘j,’ ‘i’ and ‘p_sam.’

Choose Symbol –> Locals from the View menu or click on the Locals toolbar button [ ] to display the Locals window.

The Locals window shows the values of local variables in the function indicated by the current value of the program counter (PC).

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

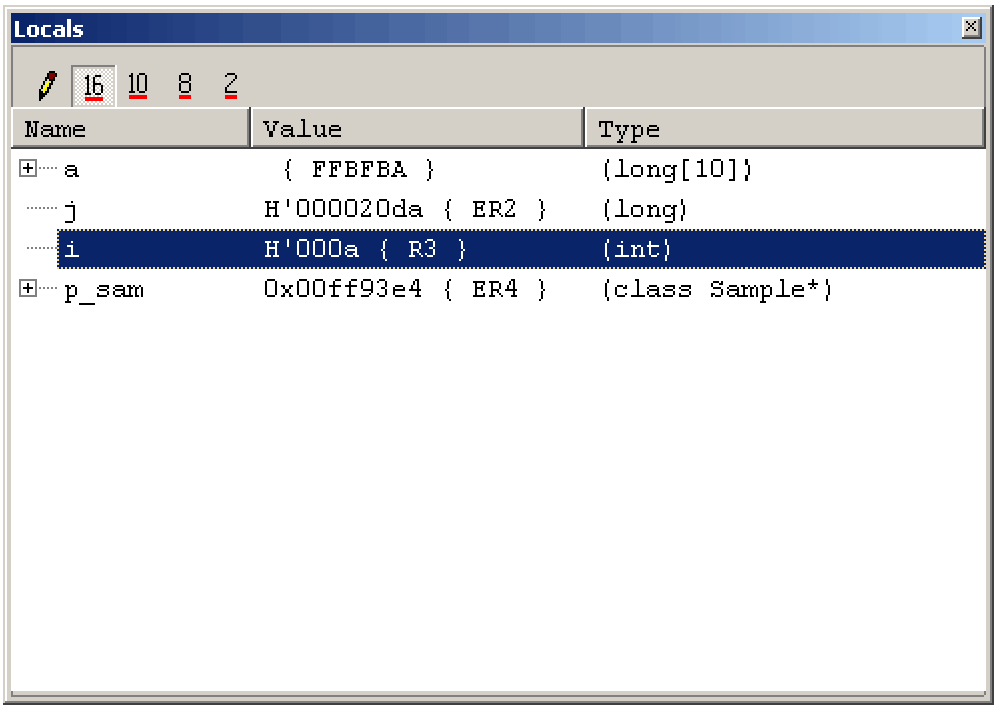


Figure 3.18 Locals window

Click on the “+” mark shown to the left of array a in the Locals window to display the elements comprising array a.

Confirm that the random data are being sorted into ascending order by inspecting the elements of array a before and after execution of the sort function.

3.13 Single-Stepping through a Program

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

Table 3.0.1 Step Options

Command	Description
Step In	Executes a program one statement at a time (including statements within functions).
Step Over	Executes a program one statement at a time by ‘stepping over’ function calls, if there are any.
Step Out	After exiting a function, stops at the next statement of a program that called the function.
Step...	Single-step a program a specified number of times at a specified speed.

3.13.1 Executing Step In Command

The Step In command ‘steps in’ to a called function and stops at the first statement of the function.
To enter the sort function, choose Step In from the Debug menu or click on the Step In toolbar button.

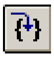


Figure 3.19 Step In button

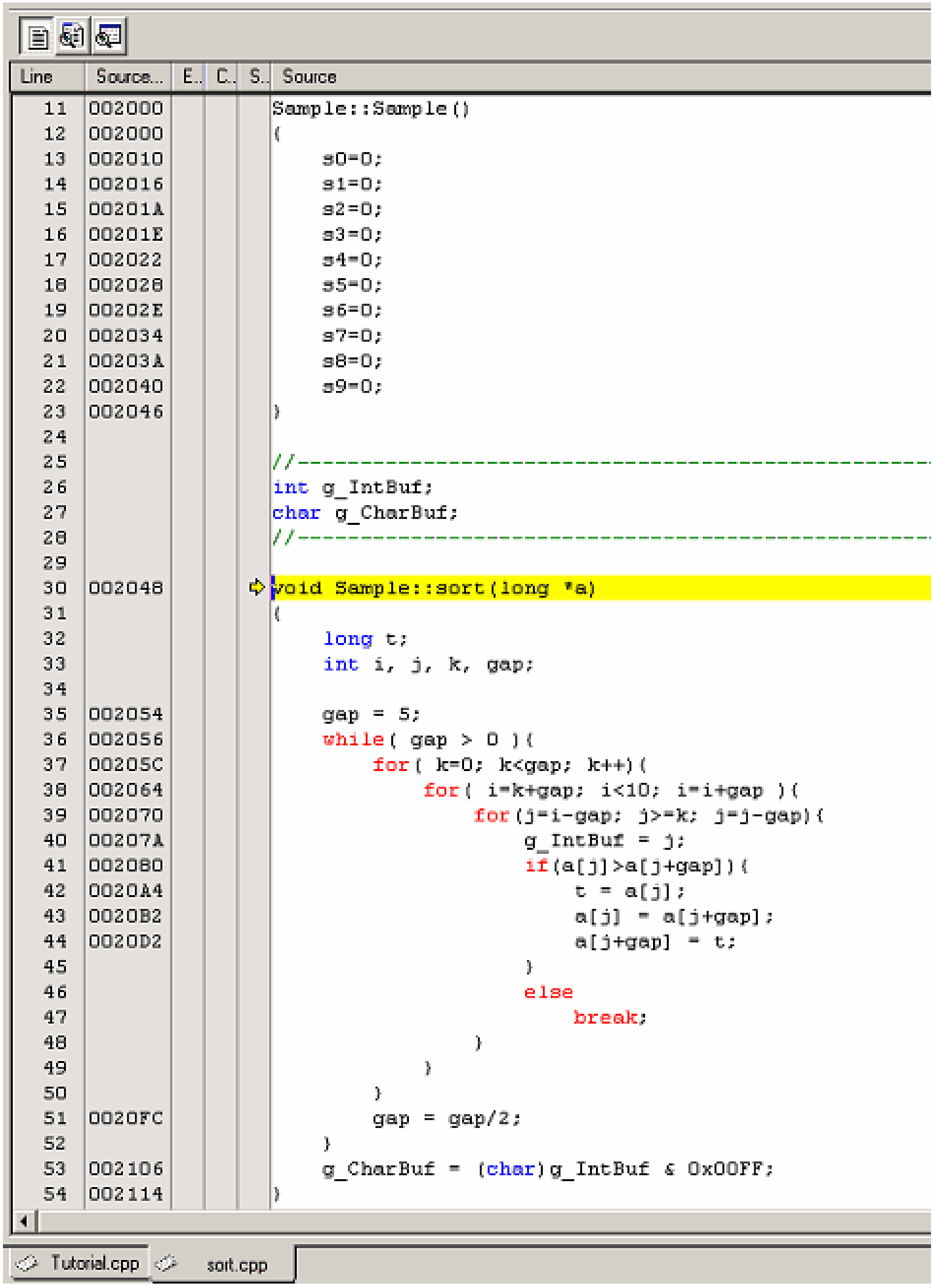


Figure 3.20 Editor window (Step In)

The highlight 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 takes execution out of a called function by completing its execution at once and only stopping at the next statement of the program from which the function was called.

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

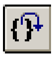


Figure 3.21 Step Out button

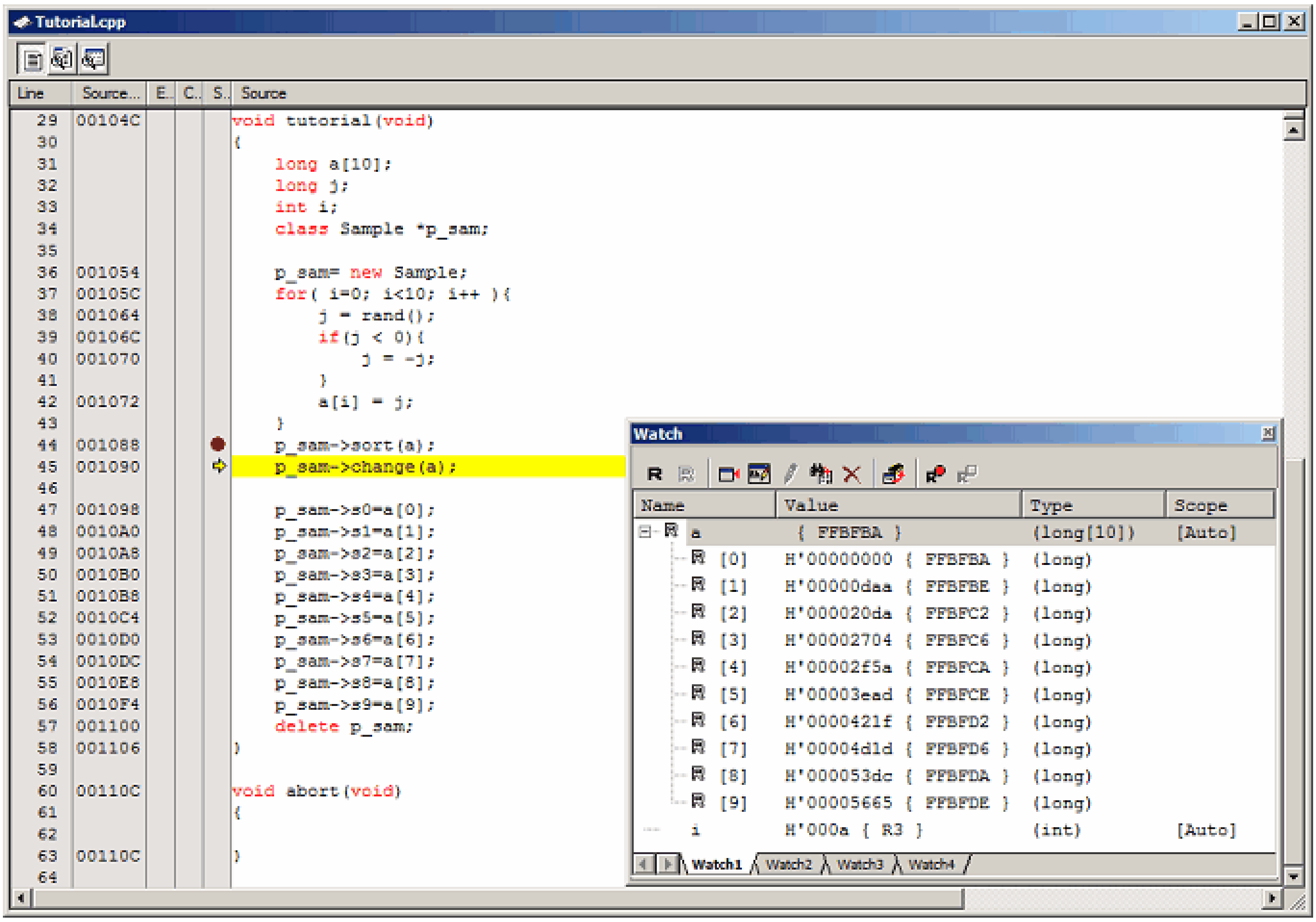


Figure 3.22 Editor window (Step Out)

The data of the variable 'a' displayed in the Watch window will have been sorted into 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 once, choose Step Over from the Debug menu or click on the Step Over toolbar button.

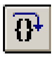


Figure 3.23 Step Over button

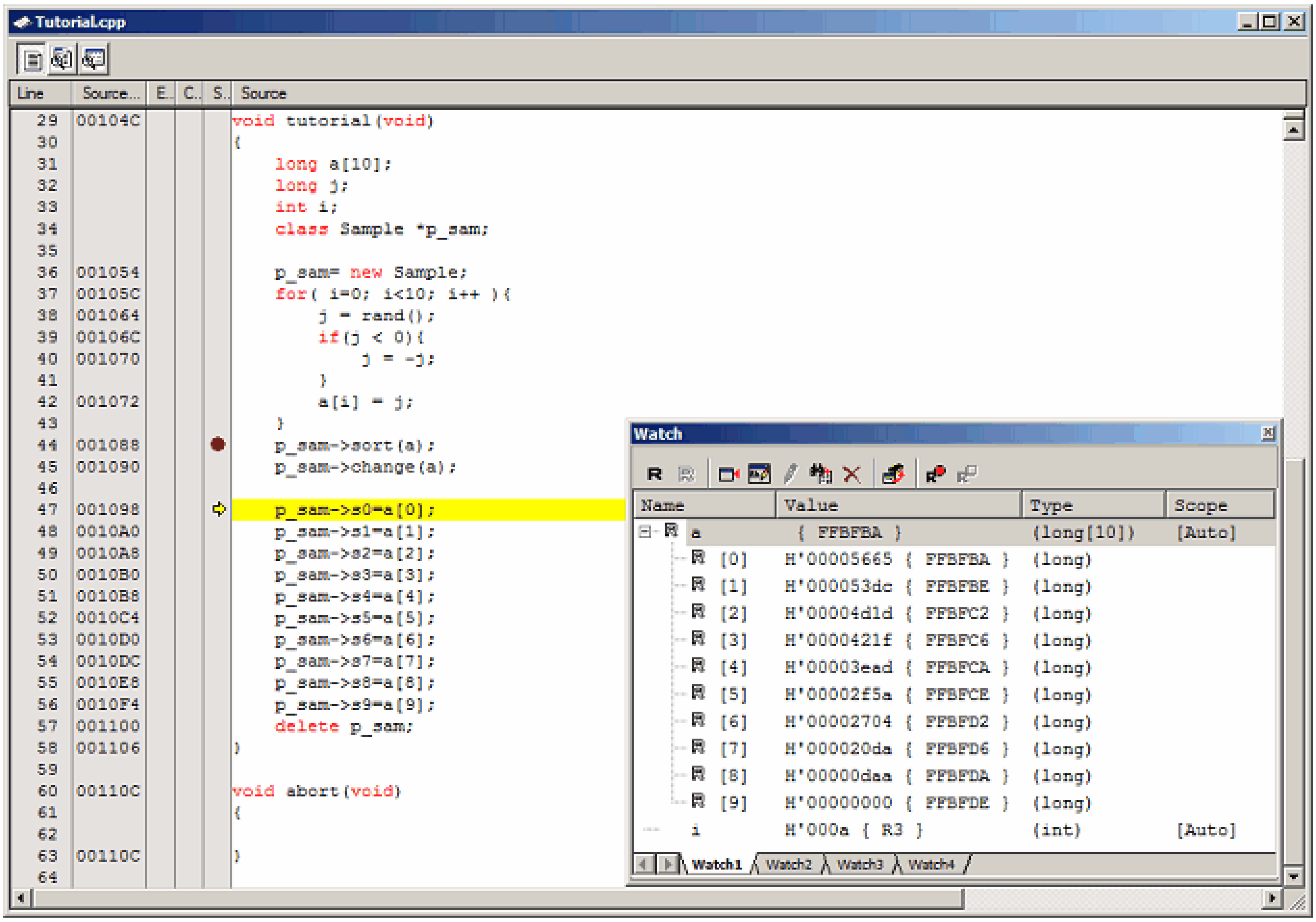


Figure 3.24 Editor window (Step Over)

The data of the variable 'a' displayed in the Watch window will have been sorted into descending order.

3.14 Forcibly Breaking Program Execution

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

Clear all breakpoints.

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

Figure 3.25 Go button

Since the program execution is now in an endless loop, choose Stop Program from the Debug menu or click on the Halt toolbar button.

Figure 3.26 Halt button

3.15 Hardware Break Facility

A hardware break causes the program to stop when it executes the instruction at a specified address (instruction fetch) or reads from or writes to a specified memory location (data access).

3.15.1 Stopping a Program when It Executes the Instruction at a Specified Address

It's easy to set an instruction fetch event in the Editor window. For example, you can set an instruction fetch event where the sort function is called.

Double-click in the row of the Event column which corresponds to the source line containing the call of the sort function.

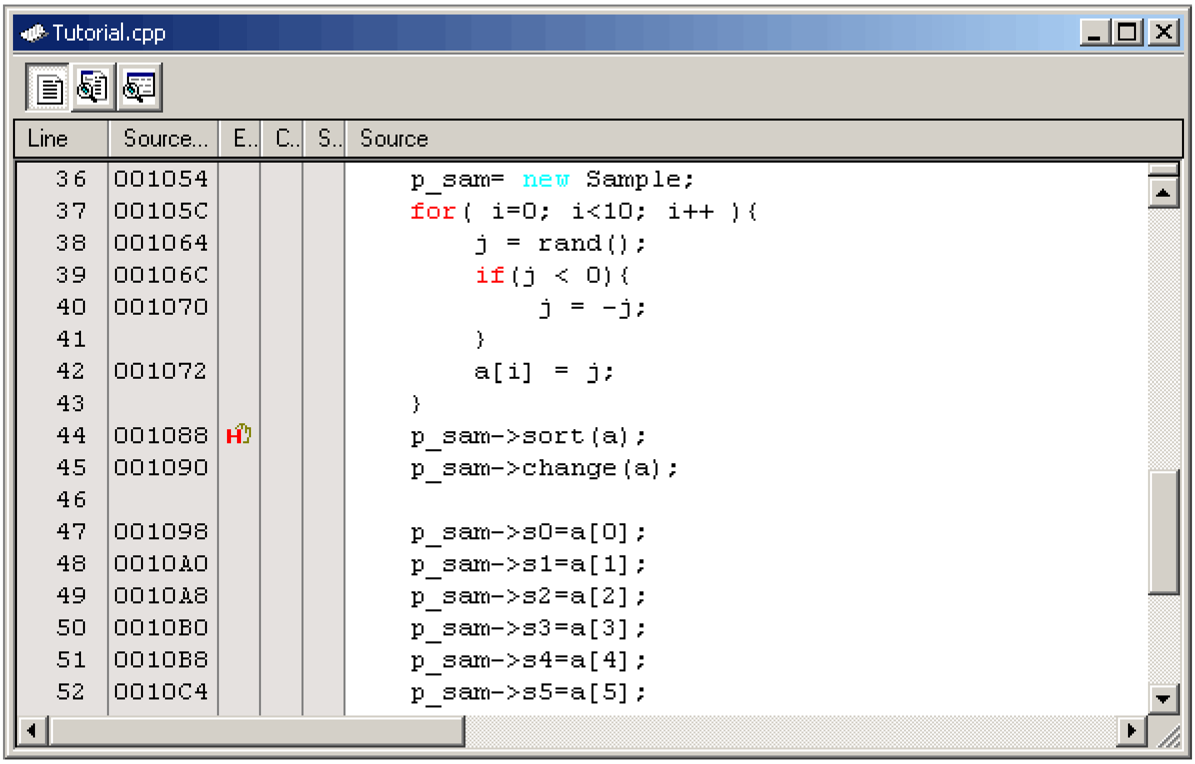


Figure 3.27 Editor window (setting a hardware breakpoint)

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

3.16 Stopping a Program when It Accesses Memory

To make a program stop when it reads or writes the value of a global variable, follow the procedure below.

Choose Event -> Hardware Break from the View menu to open the Hardware Break dialog box.

Open the OR page of the Hardware Break dialog box. Select a global variable in the Editor window, and drag-and-drop the selected variable into the OR page so that the program will stop when it reads or writes the value of that variable.

Then click on the Apply button.

The program will stop running when it reads or writes the value of the global variable you have set.

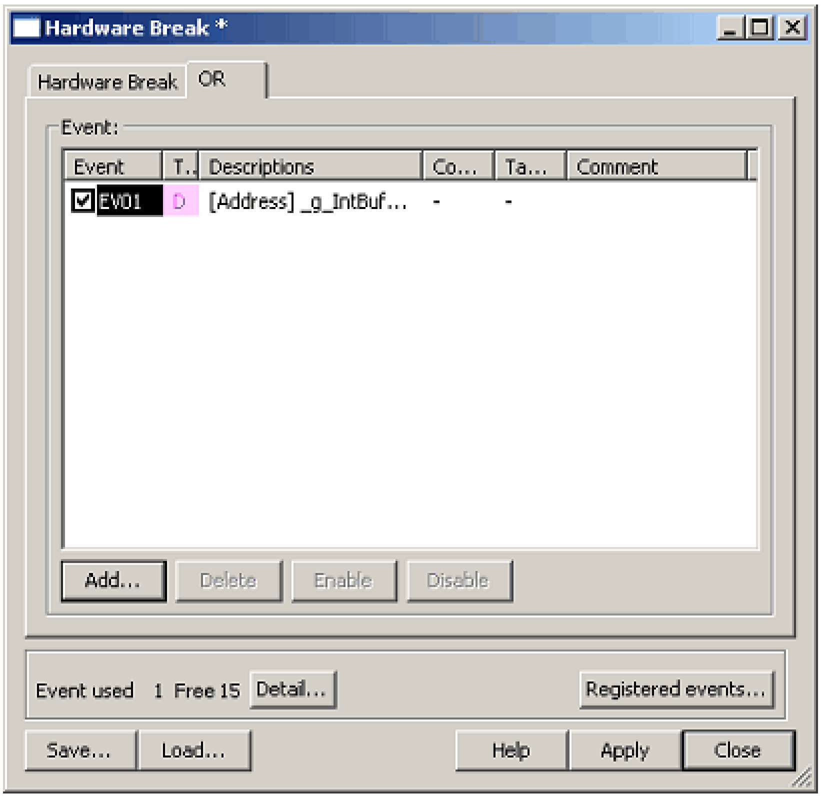


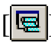
Figure 3.28 Hardware Break dialog box

Notes: (1) To be selectable, a global variable must be represented by 1, 2, or 4 bytes in memory.

(2) Local variables cannot be set as hardware-break conditions.

3.17 Tracing Facility

The tracing facility of the E100 emulator includes a special memory unit known as “trace memory” that can hold a record of the execution of up to 4-M bus cycles. This memory is constantly updated during program execution. The contents of trace memory are displayed in the Trace window.

Choose Code –> Trace from the View menu or click on the Trace toolbar button [].

The Trace window shown below will be displayed.

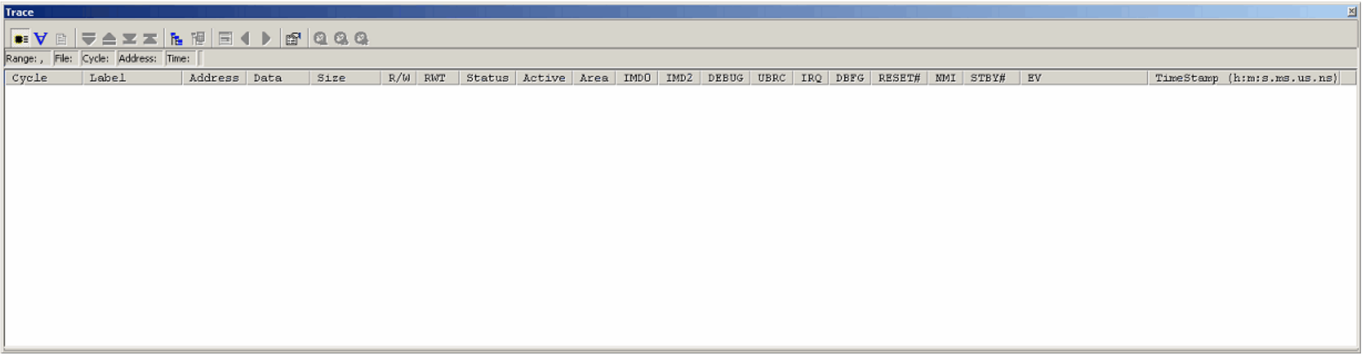


Figure 3.29 Trace window

The following section gives an outline of the tracing facility and how to set up the facility.

3.17.1 Showing the Information Acquired in "Fill Until Stop" Tracing

In "fill until stop" tracing, trace information is successively acquired from the start of user program execution until a break is encountered.

(1) Clear all break conditions. Click the right mouse button with the cursor anywhere in the Trace window and choose Acquisition from the popup menu. The Trace conditions dialog box shown below will be displayed. Check that the selected trace mode is Fill until stop. Click on the Close button.

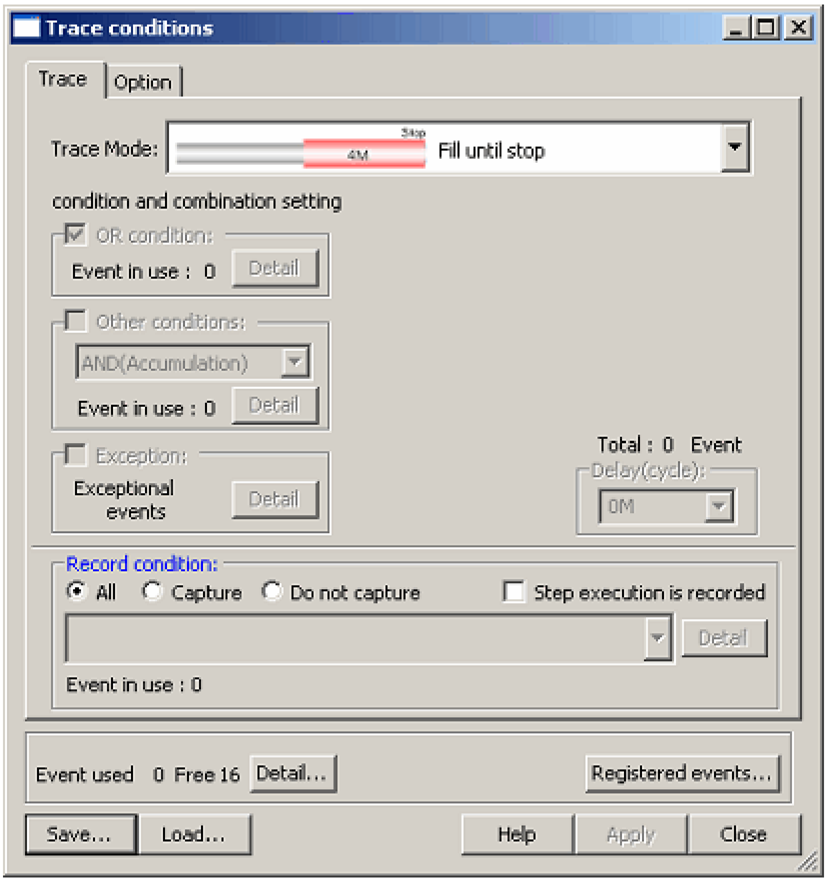


Figure 3.30 Trace conditions dialog box (fill until stop)

(2) Set a software break on the following line of the tutorial function: "p_sam ->s0=a[0];".

(3) Choose Reset Go from the Debug menu. Processing will be halted by the break, and the trace information acquired prior to the break will be displayed in the Trace window.

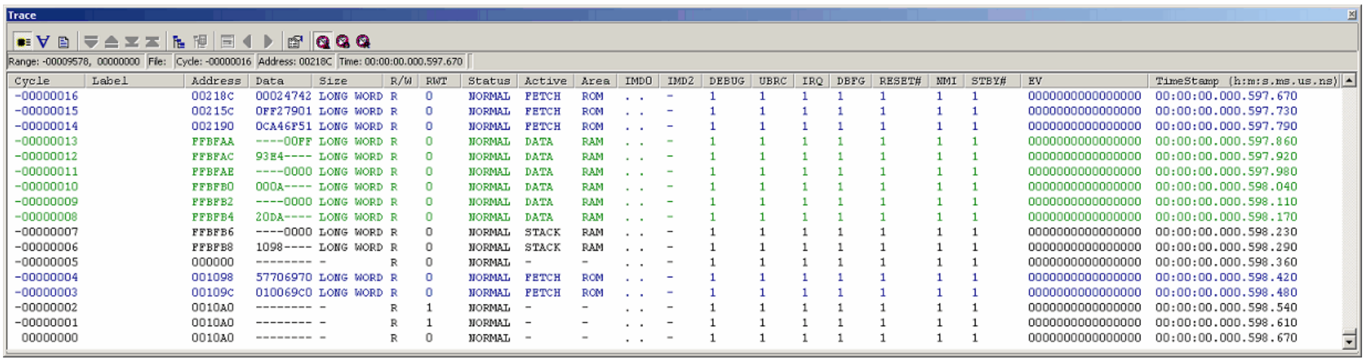


Figure 3.31 Trace window (fill-until-stop tracing)

(4) A mixed display of bus information and disassembly listing is possible. Choose Display Mode –> DIS from the popup menu to view trace information in mixed bus and disassembly mode.

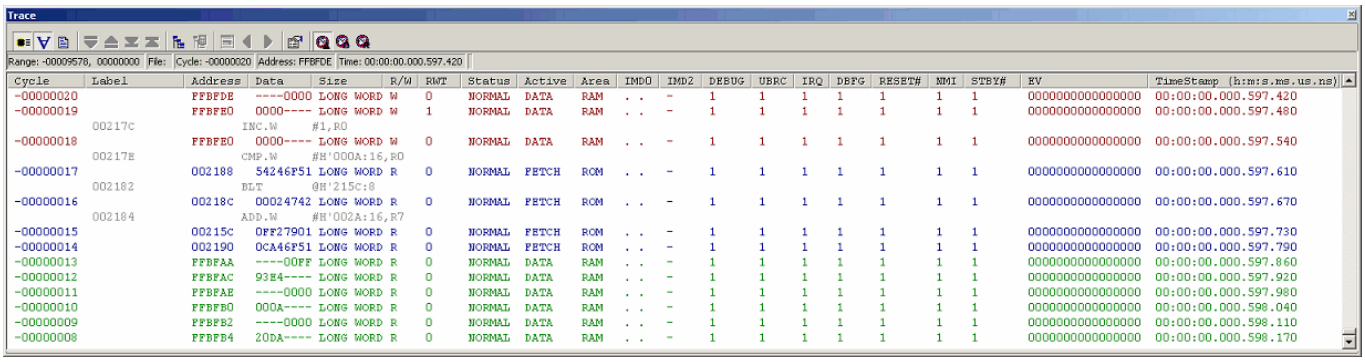


Figure 3.32 Trace window (mixed bus and disassembly mode)

(5) Choosing Display Mode –> SRC from the popup menu, on the other hand, shows a mixture of bus information, disassembly listing, and source code as the trace information.

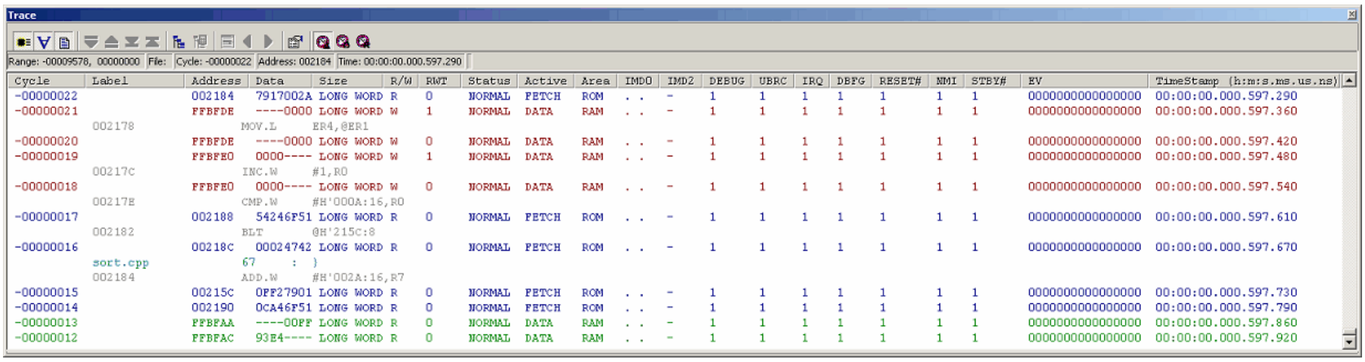


Figure 3.33 Trace window (mixed bus, disassembly, and source mode)

3.17.2 Showing the Information Acquired in "Fill around TP" Tracing

In "fill around TP" tracing, the acquisition of trace information is stopped a specified number of cycles after a trace point is encountered. This facility allows you to use trace information to keep track of program flow without having to break the user program.

(1) If any break conditions are set, clear all of them.

(2) Choose "Fill around TP" as the trace mode in the Trace conditions dialog box. In the Delay (cycle) section, specify 4M. (Up to 4-M cycles of trace information from where a trace point is encountered will be acquired.)

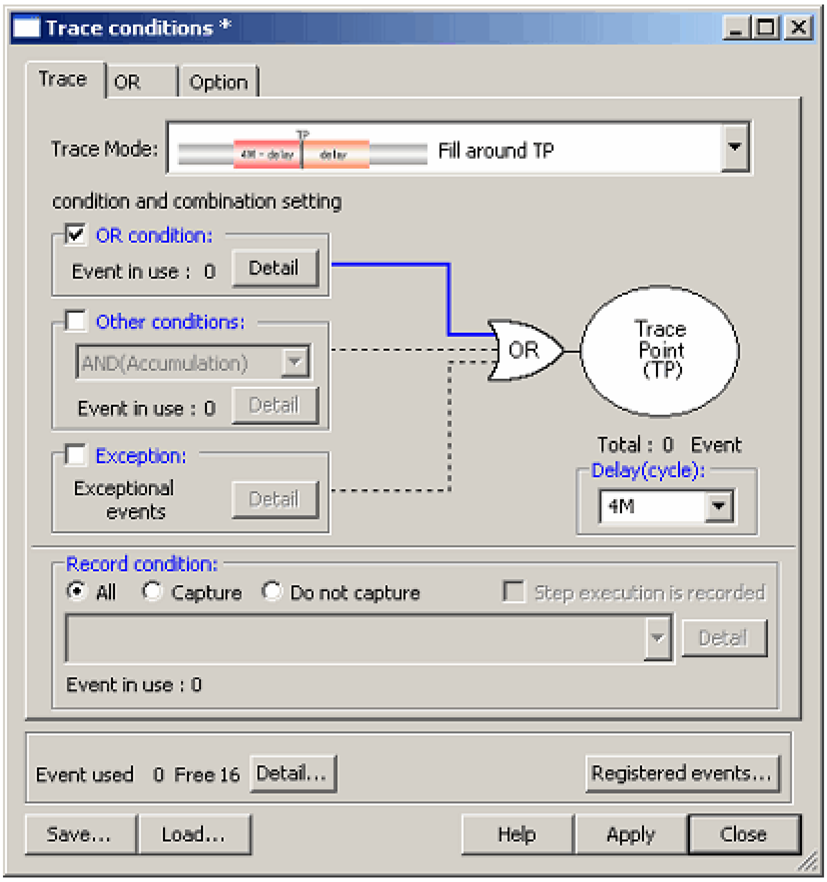


Figure 3.34 Trace conditions dialog box (Fill around TP)

(3) Next, set the trace point, i.e. the point where the debugger will start 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 onto the OR page. Click on the Apply button and then the Close button.

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

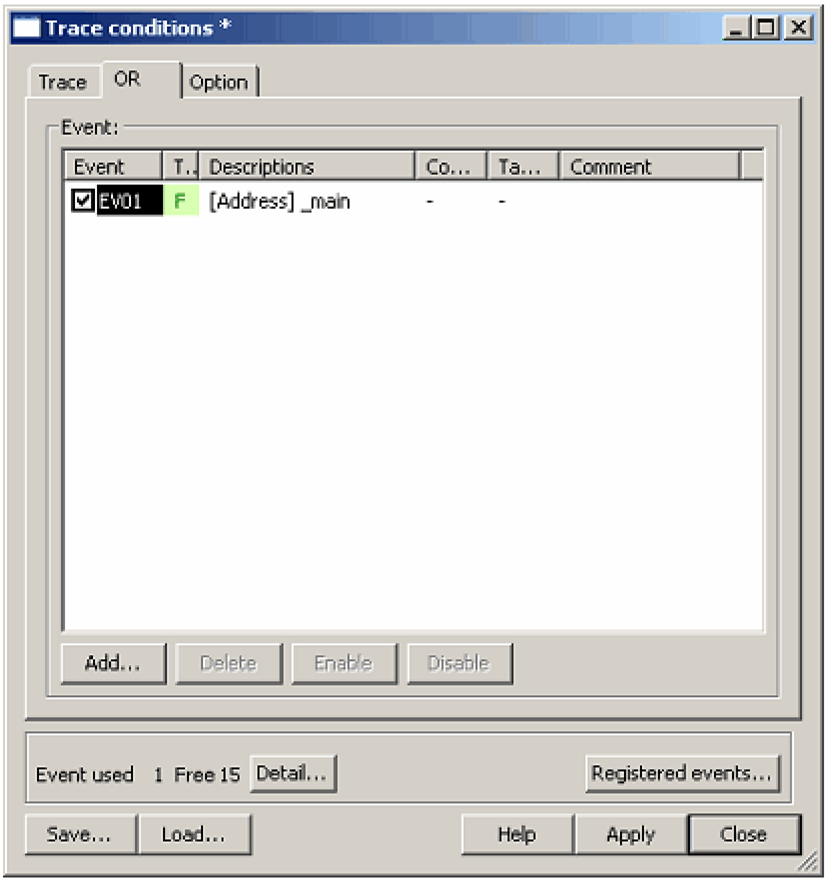
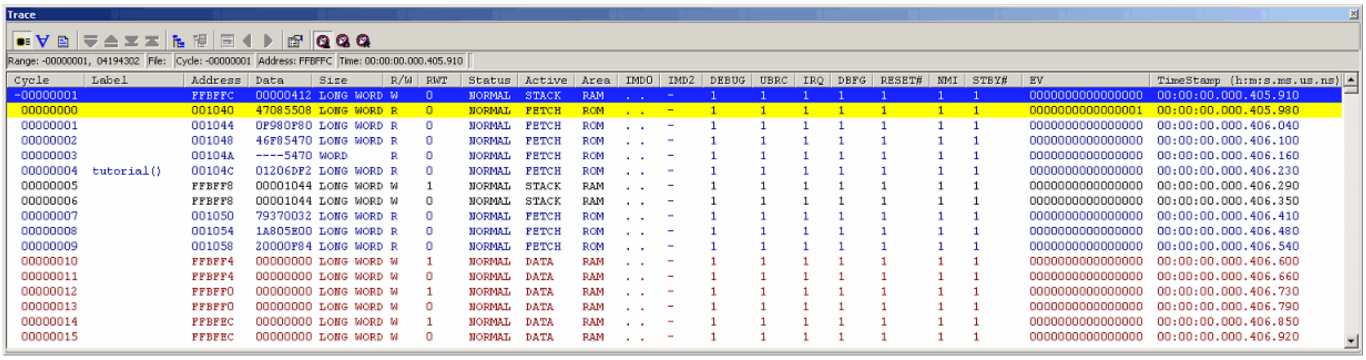


Figure 3.35 Trace conditions dialog box (OR page)

(4) Choose Reset Go from the Debug menu. As soon as the trace point is reached, trace information as shown below will start to be displayed in the Trace window.



Cycle	Label	Address	Data	Size	R/W	RWT	Status	Active	Area	IMD0	IMD2	DEBUG	UBRC	IRQ	DBFG	RESET#	NMI	STBY#	EV	TimeStamp (h:m:s.ms.us.ns)
-00000001		FFBFFC	00000412	LONG WORD	W	0	NORMAL	STACK	RAM	. .	-	1	1	1	1	1	1	1	0000000000000000	00:00:00.000.405.910
00000000		001040	47085508	LONG WORD	R	0	NORMAL	FETCH	ROM	. .	-	1	1	1	1	1	1	1	0000000000000001	00:00:00.000.405.980
00000001		001044	0F980F80	LONG WORD	R	0	NORMAL	FETCH	ROM	. .	-	1	1	1	1	1	1	1	0000000000000000	00:00:00.000.406.040
00000002		001048	46F85470	LONG WORD	R	0	NORMAL	FETCH	ROM	. .	-	1	1	1	1	1	1	1	0000000000000000	00:00:00.000.406.100
00000003		00104A	----5470	WORD	R	0	NORMAL	FETCH	ROM	. .	-	1	1	1	1	1	1	1	0000000000000000	00:00:00.000.406.160
00000004	tutorial()	00104C	01206DF2	LONG WORD	R	0	NORMAL	FETCH	ROM	. .	-	1	1	1	1	1	1	1	0000000000000000	00:00:00.000.406.230
00000005		FFBFF8	00001044	LONG WORD	W	1	NORMAL	STACK	RAM	. .	-	1	1	1	1	1	1	1	0000000000000000	00:00:00.000.406.290
00000006		FFBFF8	00001044	LONG WORD	W	0	NORMAL	STACK	RAM	. .	-	1	1	1	1	1	1	1	0000000000000000	00:00:00.000.406.350
00000007		001050	79370032	LONG WORD	R	0	NORMAL	FETCH	ROM	. .	-	1	1	1	1	1	1	1	0000000000000000	00:00:00.000.406.410
00000008		001054	1A805E00	LONG WORD	R	0	NORMAL	FETCH	ROM	. .	-	1	1	1	1	1	1	1	0000000000000000	00:00:00.000.406.480
00000009		001058	20000F84	LONG WORD	R	0	NORMAL	FETCH	ROM	. .	-	1	1	1	1	1	1	1	0000000000000000	00:00:00.000.406.540
00000010		FFBFF4	00000000	LONG WORD	W	1	NORMAL	DATA	RAM	. .	-	1	1	1	1	1	1	1	0000000000000000	00:00:00.000.406.600
00000011		FFBFF4	00000000	LONG WORD	W	0	NORMAL	DATA	RAM	. .	-	1	1	1	1	1	1	1	0000000000000000	00:00:00.000.406.660
00000012		FFBFF0	00000000	LONG WORD	W	1	NORMAL	DATA	RAM	. .	-	1	1	1	1	1	1	1	0000000000000000	00:00:00.000.406.730
00000013		FFBFF0	00000000	LONG WORD	W	0	NORMAL	DATA	RAM	. .	-	1	1	1	1	1	1	1	0000000000000000	00:00:00.000.406.790
00000014		FFBFEC	00000000	LONG WORD	W	1	NORMAL	DATA	RAM	. .	-	1	1	1	1	1	1	1	0000000000000000	00:00:00.000.406.850
00000015		FFBFEC	00000000	LONG WORD	W	0	NORMAL	DATA	RAM	. .	-	1	1	1	1	1	1	1	0000000000000000	00:00:00.000.406.920

Figure 3.36 Trace window (Fill around TP)

3.17.3 Showing a History of Function Execution

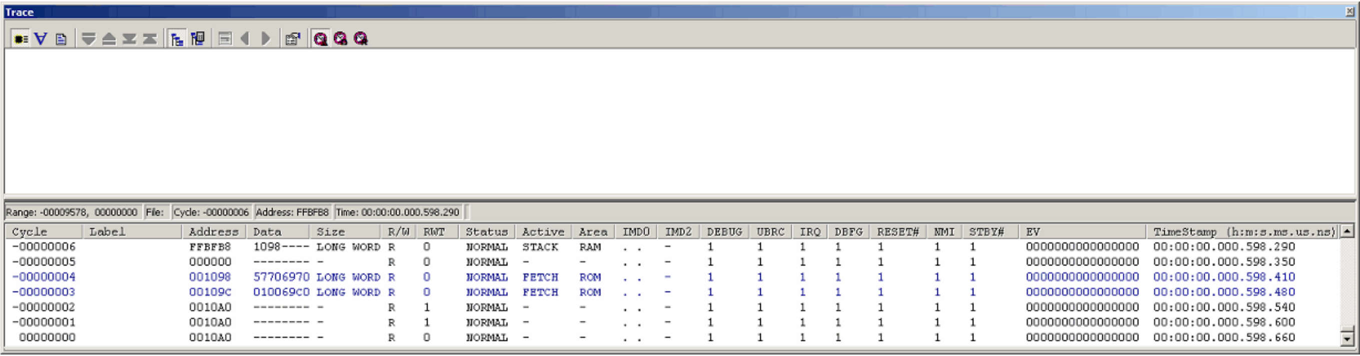
You can extract the history of executed functions from the acquired trace information.

(1) Clear all break conditions. Click the right mouse button with the cursor anywhere in the Trace window and choose Acquisition from the popup menu. The Trace conditions dialog box will open. Switch to fill-until-stop tracing and click on the Apply button and then the Close button.

(2) Set a software break on the following line of the tutorial function: "p_sam–>s0=a[0];".

(3) Choose Reset Go from the Debug menu. Processing will be halted by the break, and the trace information acquired prior to the break will be displayed in the Trace window.

(4) Click the right mouse button with the cursor anywhere in the Trace window and choose Function Execution History -> Function Execution History from the popup menu.



Cycle	Label	Address	Data	Size	R/W	RWT	Status	Active	Area	IMD0	IMD2	DEBUG	UBRC	IRQ	DBFG	RESET#	NMI	STBY#	EV	TimeStamp (h:m:s.ms.us.ns)
-00000006		FFBFB8	1098----	LONG WORD	R	0	NORMAL	STACK	RAM	. .	-	1	1	1	1	1	1	1	0000000000000000	00:00:00.000.598.290
-00000005		000000	--------	-	R	0	NORMAL	-	-	. .	-	1	1	1	1	1	1	1	0000000000000000	00:00:00.000.598.350
-00000004		001098	57706970	LONG WORD	R	0	NORMAL	FETCH	ROM	. .	-	1	1	1	1	1	1	1	0000000000000000	00:00:00.000.598.410
-00000003		00109C	010069C0	LONG WORD	R	0	NORMAL	FETCH	ROM	. .	-	1	1	1	1	1	1	1	0000000000000000	00:00:00.000.598.480
-00000002		0010A0	--------	-	R	1	NORMAL	-	-	. .	-	1	1	1	1	1	1	1	0000000000000000	00:00:00.000.598.540
-00000001		0010A0	--------	-	R	1	NORMAL	-	-	. .	-	1	1	1	1	1	1	1	0000000000000000	00:00:00.000.598.600
00000000		0010A0	--------	-	R	0	NORMAL	-	-	. .	-	1	1	1	1	1	1	1	0000000000000000	00:00:00.000.598.660

Figure 3.37 Trace window (function execution history–before analysis)

(5) Click the right mouse button with the cursor anywhere in the lower pane of the Trace window and choose Analyze Execution History from the popup menu. The history of function execution will be displayed in the upper pane.

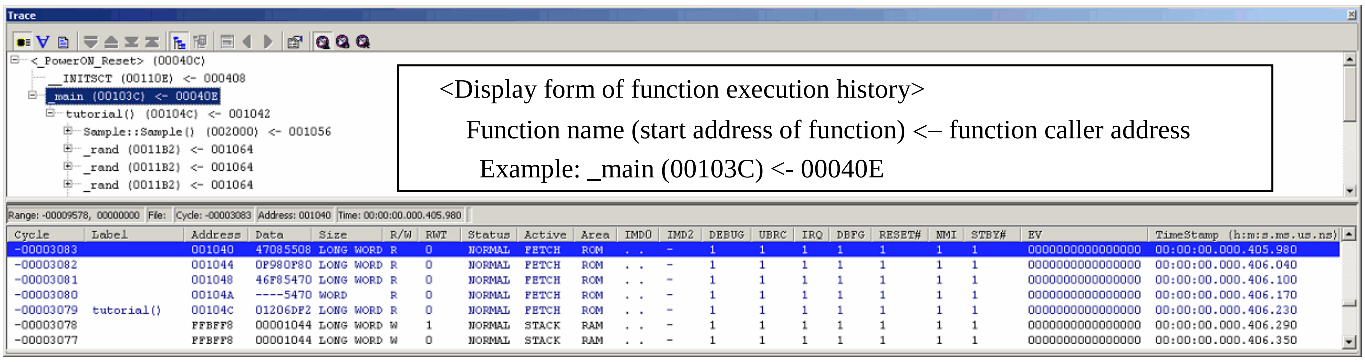


Figure 3.38 Trace window (function execution history–after analysis)

(6) Double-click on a function in the upper pane to view the trace information corresponding to that function in the lower pane.

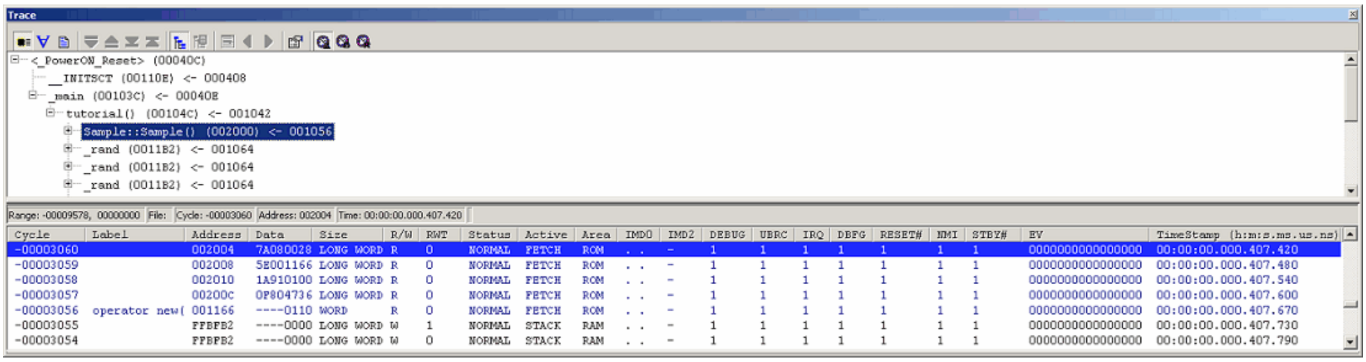


Figure 3.39 Trace window (function execution history)

3.17.4 Filtering Facility

Use the filtering facility to extract specific cycles from the acquired trace information.
This is achieved by software filtering of the trace information that was acquired by hardware.
Unlike the "Capture/Do not Capture conditions" where you set conditions for acquisition before getting the trace information, this facility allows you to change filter settings for the acquired trace information any number of times without having to reexecute the program. This makes it easy to extract the information you need.

(1) Clear all break conditions. Click the right mouse button with the cursor anywhere in the Trace window and choose Acquisition from the popup menu that is displayed. The Trace conditions dialog box will be displayed. Check that the selected trace mode is Fill until stop. Click the Close button.

(2) Set a software break on the following line of the tutorial function: "p_sam–>s0=a[0];".

(3) Choose Reset Go from the Debug menu. Processing will be halted by the break, and the trace information acquired prior to the break will be displayed in the Trace window.

(4) Choose Auto Filter from the popup menu of the Trace window. The columns for which filtering can be applied will be marked by a [▼] button.

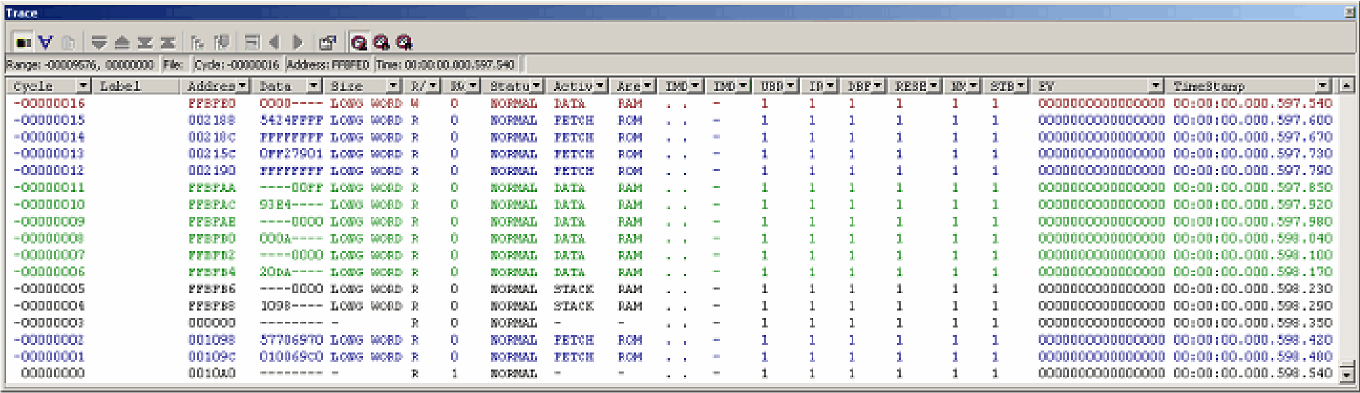


Figure 3.40 Trace window (Auto Filter)

(5) Click on the [ ] button in the R/W column and choose R.

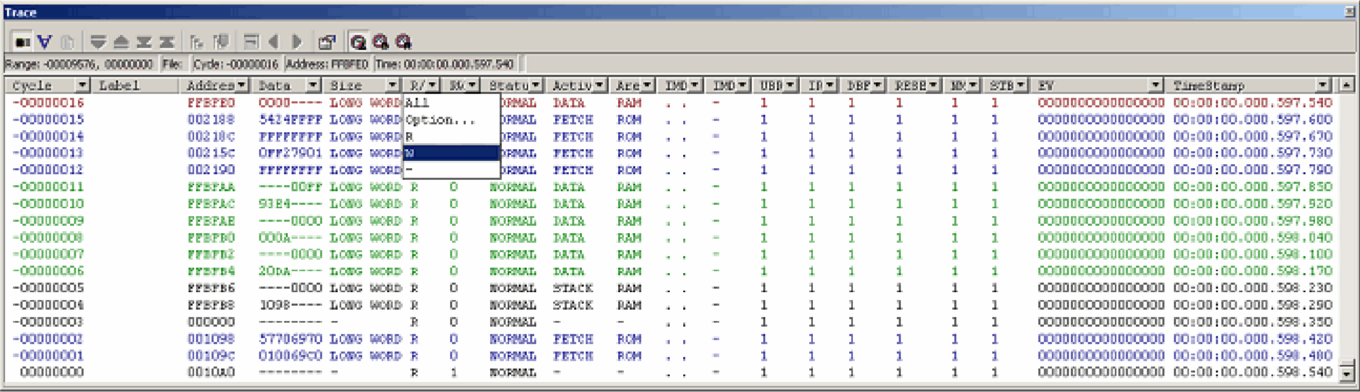


Figure 3.41 Trace window (Auto Filter)

(6) The Trace window now only shows trace information for cycles that have R in the R/W column.

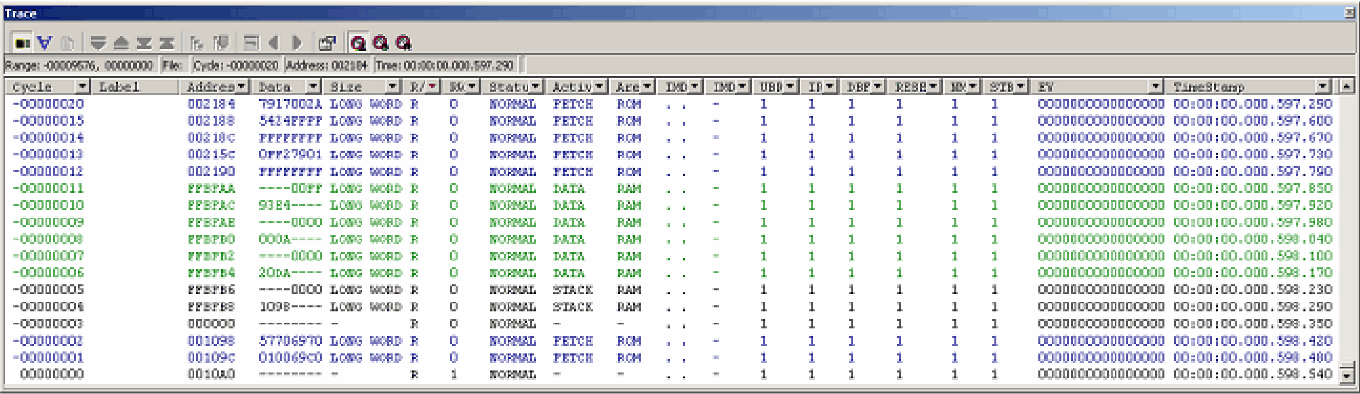


Figure 3.42 Trace window (Auto Filter)

Notes:

(1) Filtering does not affect the trace memory, so that its contents remain intact.

(2) Filtering is available for trace information regardless of whether the setting is fill until stop, fill until full or fill around TP.

3.18 Stack Trace Facility

Stack information can be used to find out which function called the function corresponding to the current PC value.

Set a software breakpoint in any line of the tutorial function by double-clicking on the corresponding row in the S/W Breakpoints column.

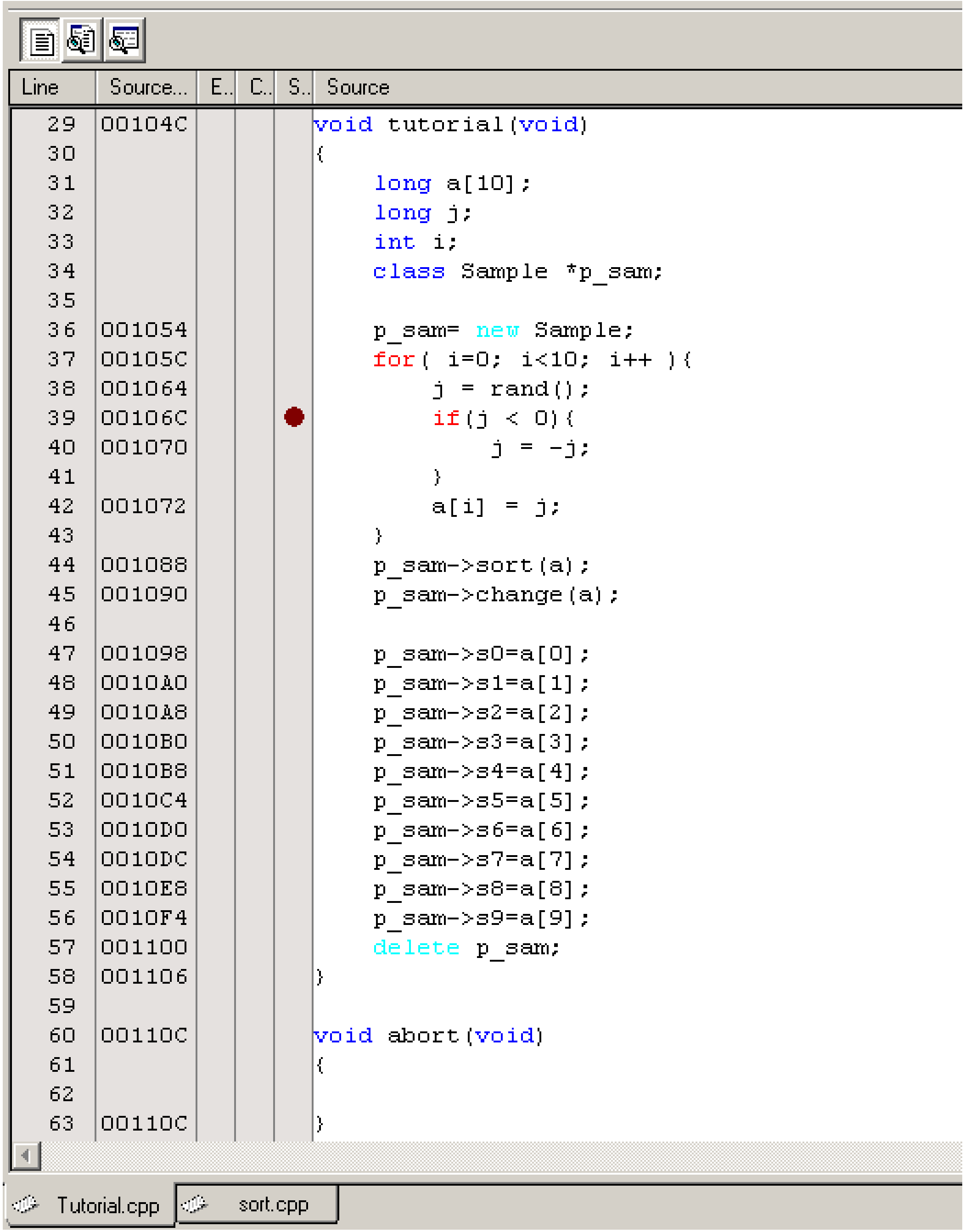


Figure 3.43 Editor window (setting a software breakpoint)

Choose Reset Go from the Debug menu.

After the break, choose Code –> Stack Trace from the View menu to open the Stack Trace window.

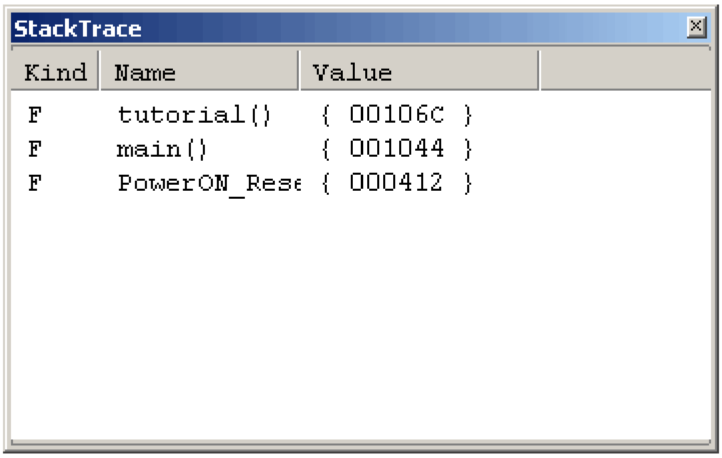


Figure 3.44 Stack Trace window

You will see that the current PC value is within the tutorial() function, and that the tutorial() function was called from the main() function.

Clear the software breakpoint that you set on a line of the tutorial function by again double-clicking on the corresponding row in the S/W Breakpoints column.

3.19 What Next?

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

The emulation facilities of the E100 emulator provide for advanced debugging. You can apply them to precisely distinguish the causes of problems in hardware and software and, once these have been identified, to effectively examine the problems.

4. Preparation for Debugging

4.1 Starting the High-performance Embedded Workshop

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

(1) Connect the host machine, E100 emulator, and user system. Then turn on 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.

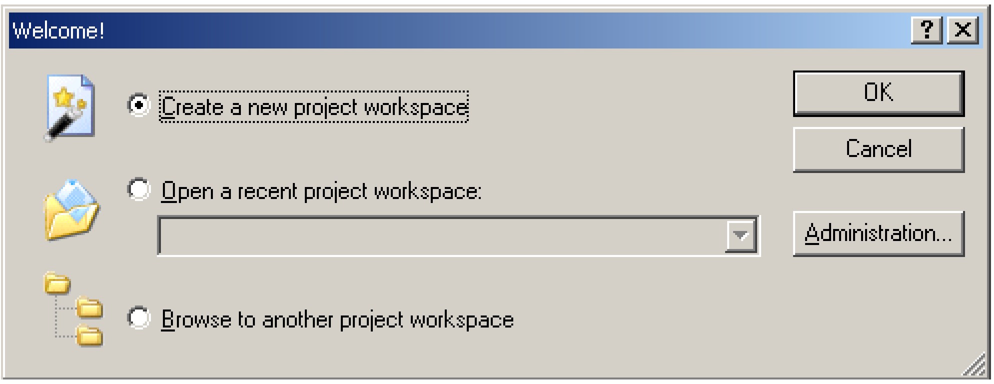


Figure 4.1 Welcome! dialog box

Select the startup method from among the following.

- Create a new project workspace
- Open a recently used project workspace
  Select this option when you use an existing workspace.
  The names of recently opened workspaces will be displayed.
- Browse for another project workspace
  Select this option when you intend to use an existing workspace.
  This option is available when there is no record of a recently opened workspace.

4.2 Creating a New Workspace (Toolchain Unused)

The procedure for creating a new project workspace differs according to whether you are using a toolchain or not.

Toolchains are not included with the E100 emulator. Toolchains can be used in environment in which the C/C++ compiler package is installed.

Follow the procedure below to create a new workspace.

(1) In the Welcome! dialog box, select the radio button with the caption “Create a new project workspace” and click on the OK button.

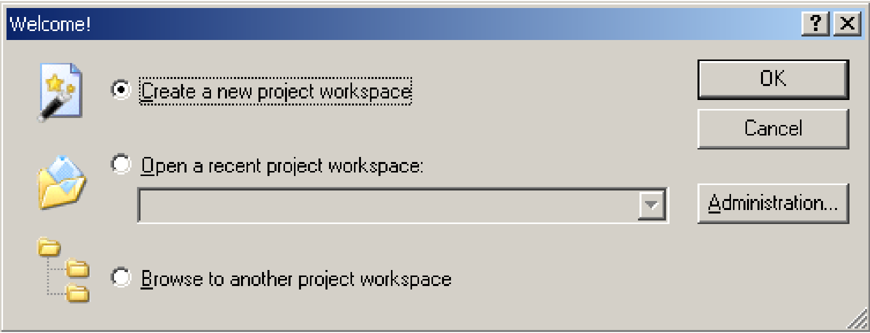


Figure 4.2 Welcome! dialog box

(2) The Project Generator will start.

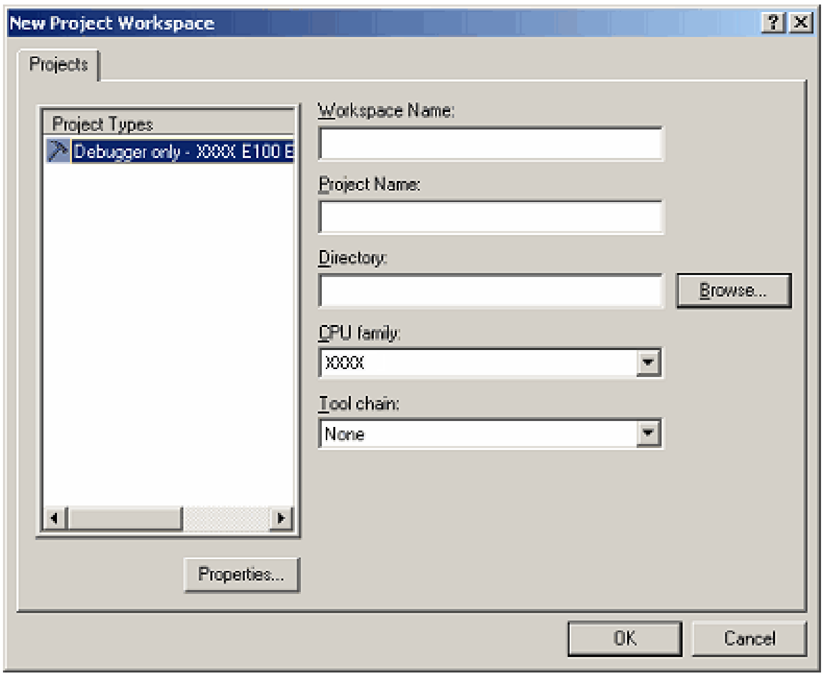


Figure 4.3 New Project Workspace dialog box

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

The other list boxes are used for setting up a toolchain. If no toolchains are installed, fixed information is displayed here. Click on the OK button.

(3) Select the target for debugging.

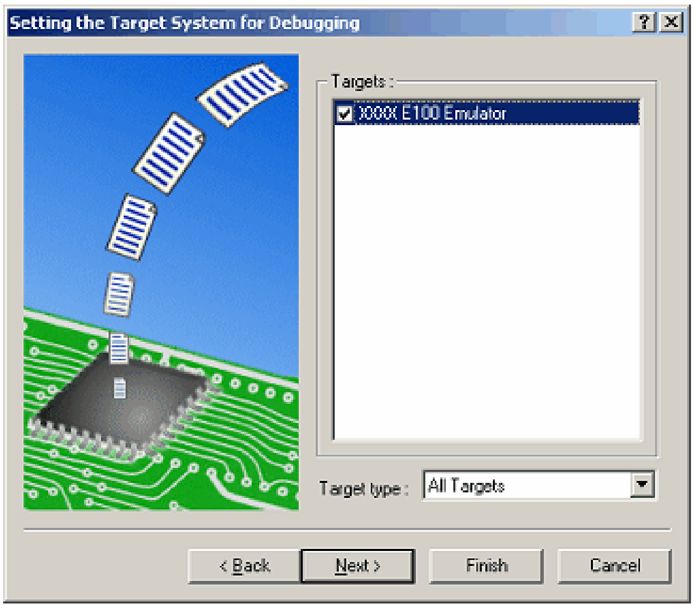


Figure 4.4 Setting the Target System for Debugging dialog box

Select the target platform you wish to use by placing a check mark in its checkbox and click on the Next button.

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

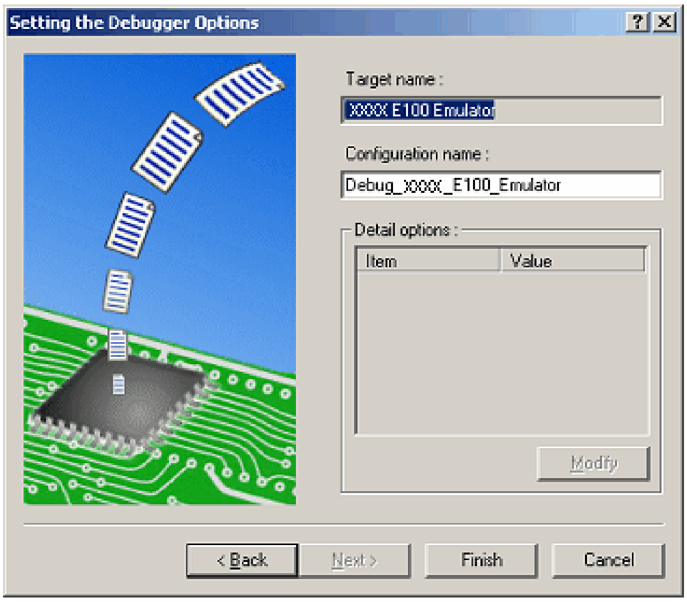


Figure 4.5 Setting the Debugger Options dialog box

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

When you have finished setting the configuration names, emulator-related settings are completed.
Click on the Finish button, and the Summary dialog box will be displayed. Clicking on the OK button in this dialog box starts the High-performance Embedded Workshop.

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

4.3 Creating a New Workspace (with a Toolchain in Use)

Follow the procedure below to create a new workspace.

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

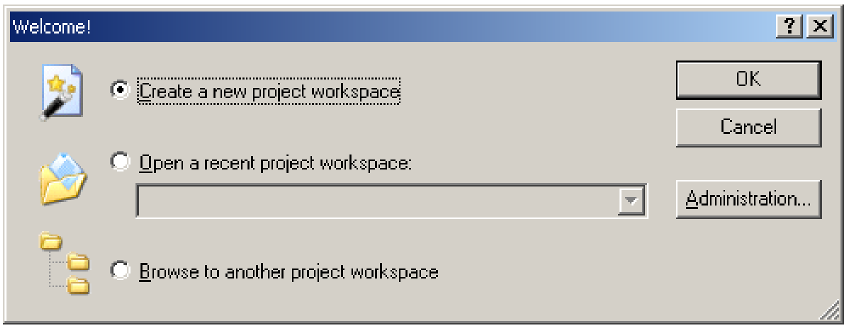


Figure 4.6 Welcome! dialog box

(2) The Project Generator will start.

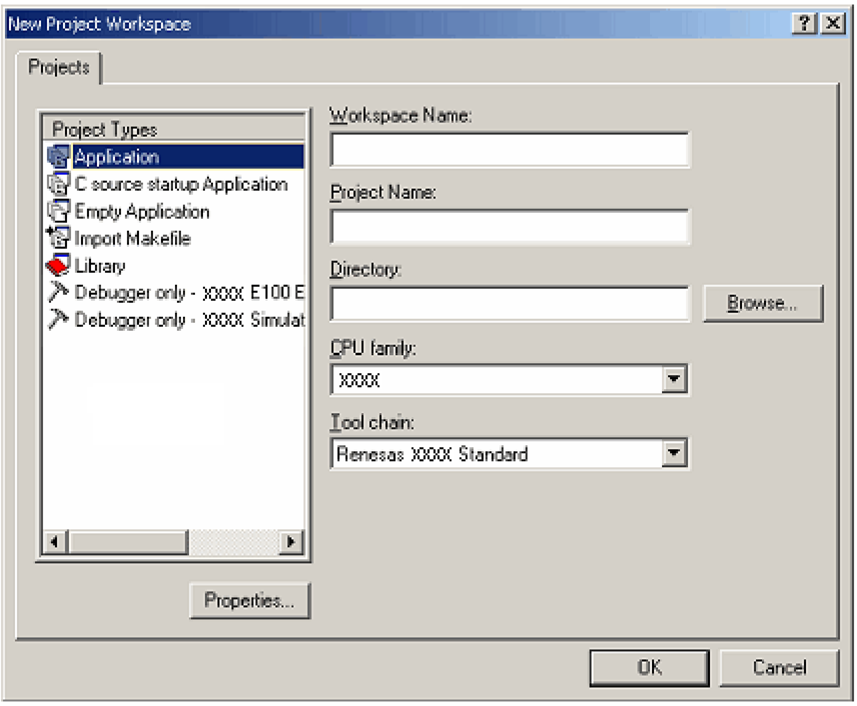


Figure 4.7 New Project Workspace dialog box

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

The other list boxes are used for setting up a toolchain. If no toolchains are installed, fixed information is displayed here. Click on the OK button.

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

(4) Select the target for debugging.

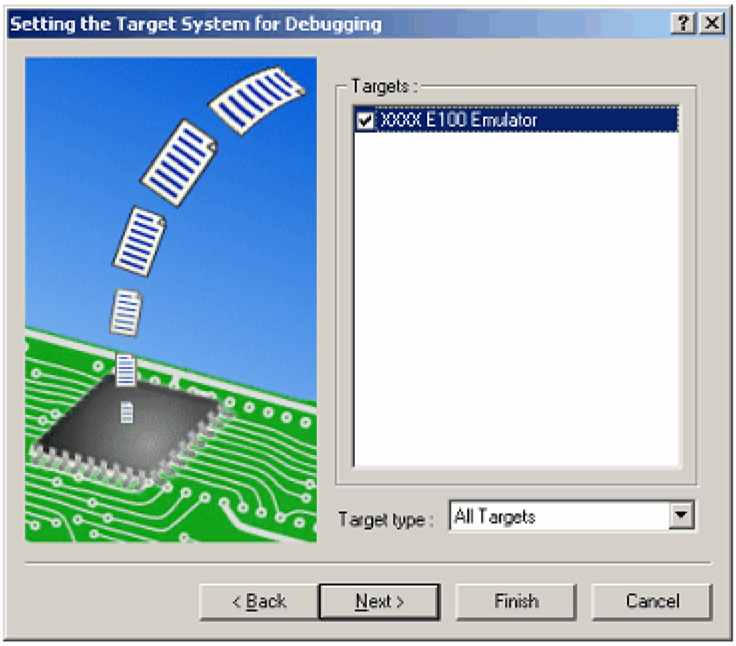


Figure 4.8 Setting the Target System for Debugging dialog box

Select the target platform you wish to use by placing a check mark in its checkbox and click on the Next button.

(5) Set a configuration name.

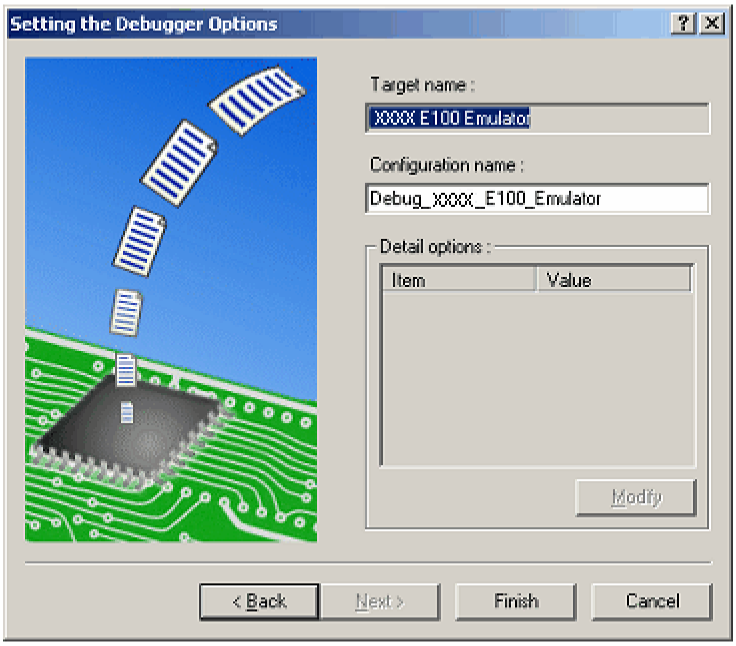


Figure 4.9 Setting the Debugger Options dialog box

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

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

4.4 Opening an Existing Workspace

Follow the procedure below to open an existing workspace.

(1) In the Welcome! dialog box, select the radio button with the caption "Browse to another project workspace" and click on the OK button.

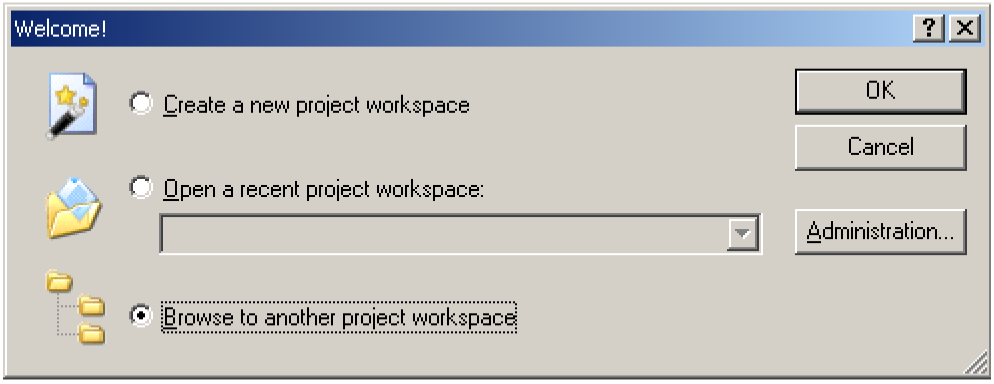


Figure 4.10 Welcome! dialog box

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

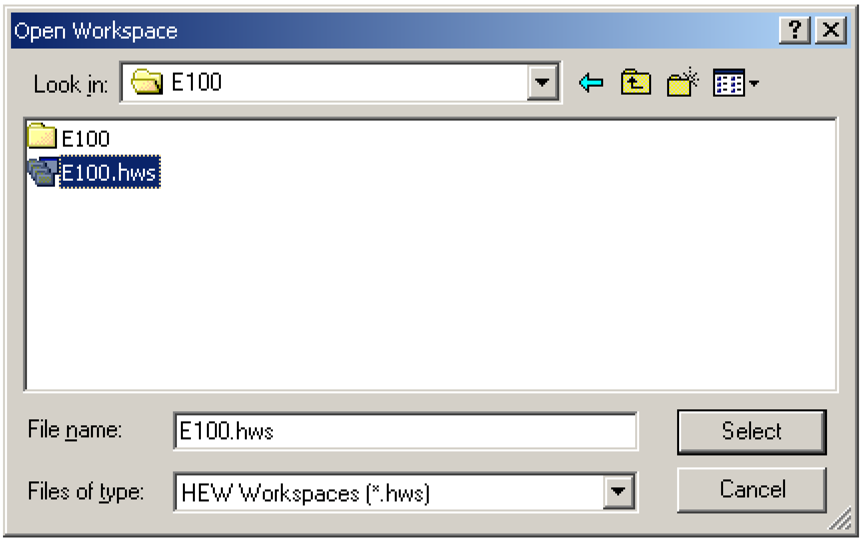


Figure 4.11 Open Workspace dialog box

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

(3) The High-performance Embedded Workshop will start, and its state will be restored to the state at the time the selected workspace was saved. If the emulator was connected at the time, the workspace is automatically connected to the emulator. If the emulator was not connected but you want to connect it, refer to "4.5 Connecting the Emulator" (page 74).

4.5 Connecting the Emulator

4.5.1 Connecting the Emulator

The following methods for connecting the emulator are available.

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

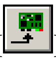
Choose Debug Settings from the Debug menu to open the Debug Settings dialog box. In this dialog box, you can register download modules and the command chain to be automatically executed. When you are finished filling in the Debug Settings dialog box, the emulator will be connected.

(2) Loading a session file

Switching to a session file in which settings for emulator usage have been made in advance simplifies the procedure of connecting the emulator.

4.5.2 Reconnecting the Emulator

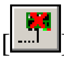
While the emulator is disconnected, you can reconnect it in one of the ways described below.

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

4.6 Disconnecting the Emulator

4.6.1 Disconnecting the Emulator

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

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

4.7 Quitting the High-performance Embedded Workshop

Choosing Exit from the File menu closes the High-performance Embedded Workshop.
Before it closes, a message box will be displayed asking you whether you want to save the session. To save the session, click on the Yes button.

4.8 Making Debugging-Related Settings

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

4.8.1 Specifying a Module for Downloading

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

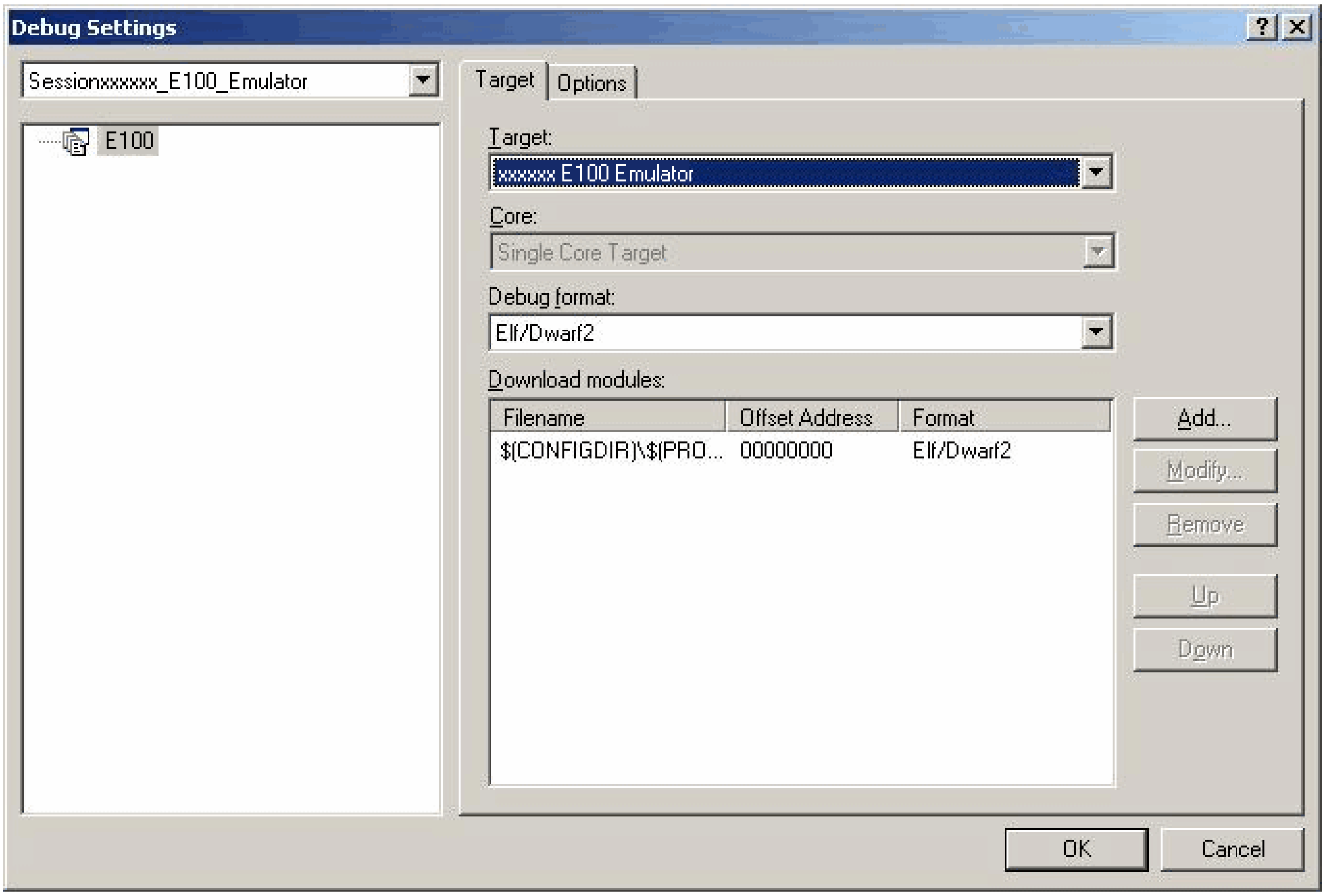


Figure 4.12 Debug Settings dialog box

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

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

CAUTION

At this point in time, no programs have been downloaded yet.

For details on how to download a program, refer to "5.2 Downloading a Program" (page 92).

4.8.2 Setting Up Automatic Execution of Command Line Batch Files

Click on the Options tab of the dialog box.

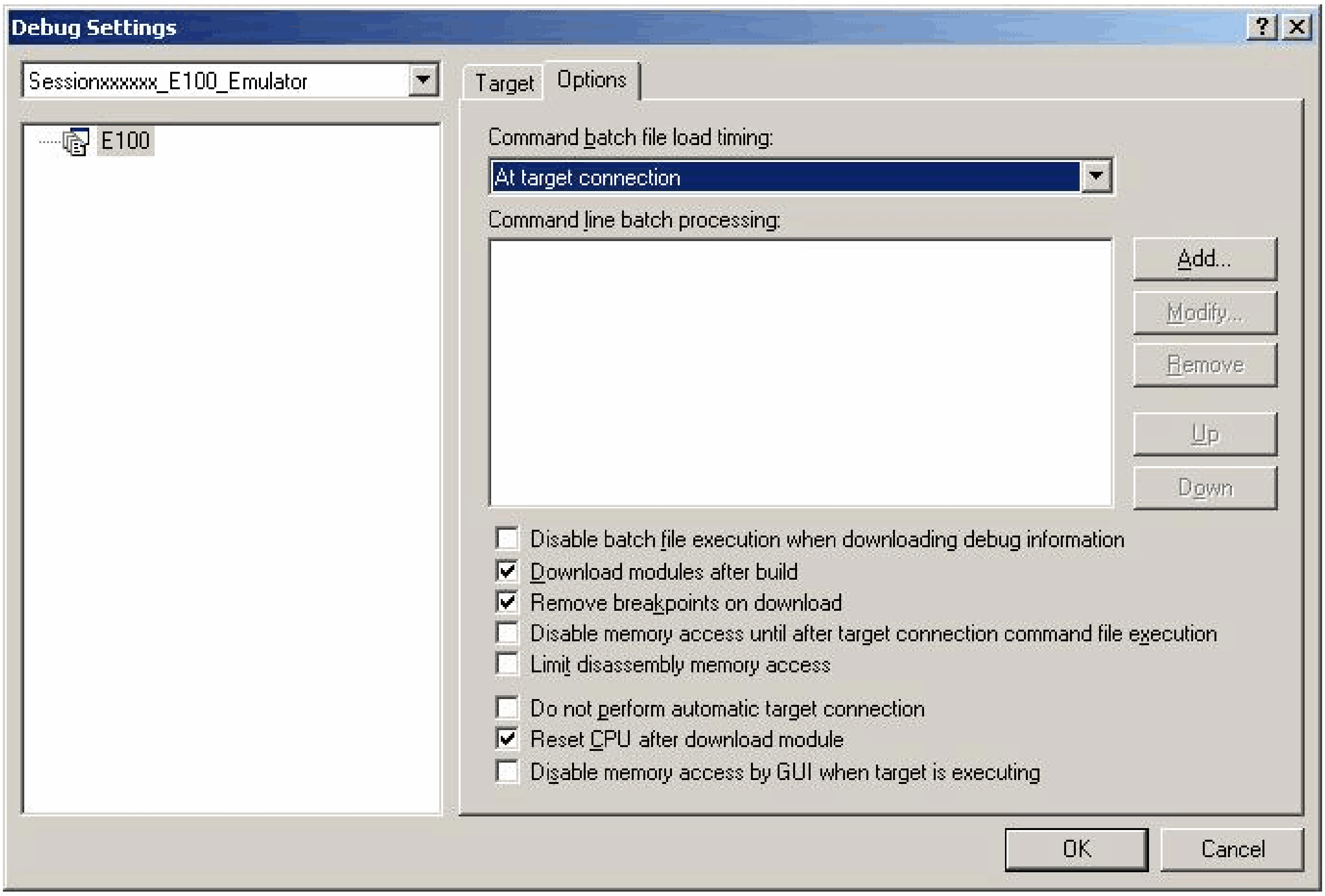


Figure 4.13 Debug Settings dialog box

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

- When the emulator is connected
- Immediately before downloading
- Immediately after downloading
- Immediately after a 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 Debugging Functions

Item No.	Item			Specification
1	Software break			4,096 points
2	Event	Number of event points		Maximum number of effective points: 16
		Event type		Executed address detection
				Data access detection
				Interrupt generation or exit detection
				External trigger detection
		Task ID		Can be set separately for each event
		Condition for number of times an event has occurred		Up to 255 times
3	Exception detection			Violation of access protection
				Reading from non-initialized memory areas
				Stack access violation
				Performance-measurement overflow
				Realtime profile overflow
				Trace memory overflow
				Task stack access violation
				OS dispatch
4	Hardware break	Hardware breakpoints	Event combination	OR, AND (cumulative), AND (simultaneous), subroutine, sequential and state transition
			Exception detection	See item No. 3
		Delay		Maximum 65,535 bus cycles
5	Trace	Trace size		Maximum 4-M cycles
		Trace mode	Fill until stop	Trace acquisition continues until the program stops running.
			Fill until full	Trace acquisition stops when trace memory becomes full.
			Fill around TP	Trace acquisition proceeds for a delay in cycles after the trace point has been reached.
			Repeat fill until stop	Information for a total of 512 cycles before and after each trace point are acquired, and this continues until the program stops.
			Repeat fill until full	Information for a total of 512 cycles before and after each trace point are acquired, and this continues until trace memory is full.
		Trace point	Event combination	OR, AND (cumulative), AND (simultaneous), subroutine, sequential and state transition
			Exception detection	See item No. 3
		Delay		Up to 4-M bus cycles
		Extraction/deletion of trace data		Extracting or deleting data by specifying events - Between two events - Duration of an event - Duration of an event occurring in a subroutine
				Instruction accessing specific data
6	Performance	Content of measurement		Measures maximum, minimum and average execution time, and number of passes, for up to 8 sections
				Timeout detection
		Resolution		10 ns to 1.6 µs
		Measurement mode	Event combination	Between two events, Period of an event and Interrupt-disabled range between two events
7	RAM monitor			512 bytes × 32 blocks - Shows last read/write accesses performed - Includes a facility to detect reading from non-initialized areas
8	Profile			128 Kbytes × 8 blocks (1-Mbyte space)
				Cumulative time and number of passes overflow detection

Table 5.1 List of Debug Functions (cont)

Item No.	Item	Specification
9	Coverage	C0-level code coverage 256 Kbytes x 8 blocks (2-Mbyte space)
		C0 + C1-level code coverage 128 Kbytes x 8 blocks (1-Mbyte space) Address range and source file
		Data coverage 64 Kbytes × 8 blocks (512-Kbyte space) Address range, section, and task stack

5.1 Setting Up the Emulation Enviroment

When the emulator is connected, the Device setting and the 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 only be set once each time the emulator is booted-up.

5.1.1 Emulator Settings During Booting up

While the emulator is booting up, the following three dialog boxes are opened in sequence.

(1) Device setting dialog box

Use this dialog box to select the target MCU and establish communication.
This dialog box can be re-opened by selecting Emulator -> Device setting from the Setup menu after the emulator has been booted up. In this case, however, be aware that changes of setting made after boot-up will not be reflected immediately but will be set as initial values when the emulator is reconnected.

(2) Configuration properties dialog box

This dialog box is opened after the Device setting dialog box. Use this dialog box to make settings related to the emulator and debugger functions.
This dialog box can be re-opened by selecting Emulator -> System from the Setup menu after the emulator has been booted up. Settings for certain options in this dialog box can be changed after boot-up. Those that can be changed are active while those that cannot are inactive (grayed out), but with their settings 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 details, refer to the hardware manual supplied with each product.

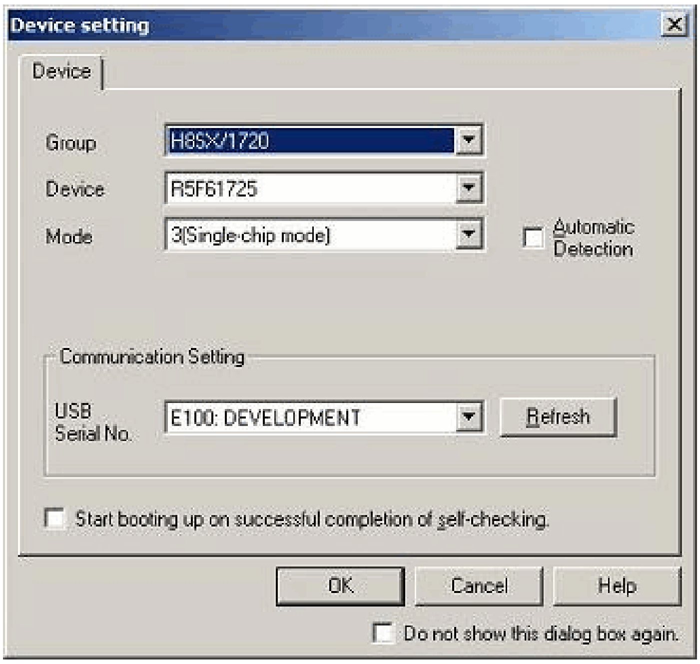


Figure 5.1 Device setting dialog box (Device page)

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

(2) Selecting an operation mode

Select an operating mode.
For details, refer to the hardware manual for the MCU in use.

(3) Automatic detection of operating mode

Select the checkbox if you want automatic detection of the operating mode after connection of the emulator has been established. When any of the following states is detected, the emulator shows an error message and releases the connection.

- The user system is not connected.
- The user system is not supplied with power.
- The setting of the MD2 to MD0 pins of the user system corresponds to an operating mode that is not supported by the emulator.

Even if the [Automatic Detection] checkbox is not selected and the user system is being supplied with power, the emulator checks the states of the MD2 to MD0 pins against the operating mode selection made by the user. When they do not match or the operating mode indicated by the MD2 to MD0 pins is not available, the emulator shows a message and makes the connection for the operating mode selected by the user.

(4) Setting up communications

You can select another target emulator for connection via USB.
The 'USB Serial No.' list box shows unique identifying information on the E100 emulator connected via USB. Clicking on the Refresh button updates the information.

(5) 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 according to the communications 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 Setting Up the System

On the System page of the Configuration Properties dialog box, specify the configuration of the emulator system as a whole. During the boot-up process, this dialog box appears after the Device setting dialog box.

Although it is possible to open this dialog box even after the emulator has been booted up, some items (e.g. target MCU and clock selection) will be grayed-out since they cannot be changed.

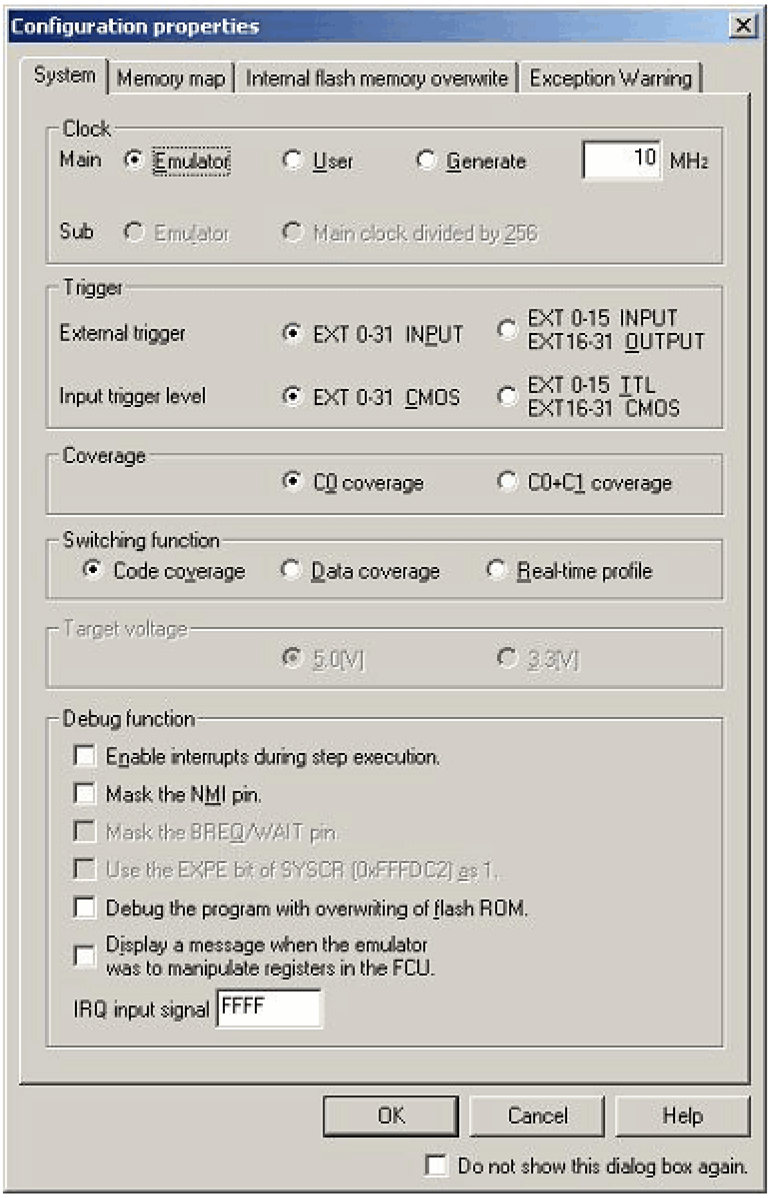


Figure 5.2 Configuration properties dialog box (System page)

(1) Selecting the input clock

In the Clock section on the System page, select the sources of the clock signals supplied for the main clock and subclock.
The main clock can be selected from among three choices: Emulator, User and Generate (by default, Emulator is selected).
Select Emulator when the main clock is supplied from an internal source and User when the main clock is supplied from an external source. To use a user-defined clock, select Generate and enter the clock frequency in the text box.
The clock frequency can be set in the range from 1.0 to 99.9 MHz in 0.1-MHz units. The clock frequency for Generate can be set only once each time the emulator is booted-up.

Subclock options are only selectable for MCUs that support a subclock function. 'Emulator' or 'Main clock divided by 256' can be selected (by default, Emulator is selected).

CAUTION
The frequency accuracy for Generate is ±5%. Please make sure that final evaluation is performed with a resonator or oscillator module mounted to generate the actual frequency for use on the target board.

(2) Selecting the direction of the external trigger cable

For the external trigger cable, select the direction of EXT pins 16–31 as input or output. EXT pins 0–15 are fixed as inputs.
Select either of the following options:

- EXT 0–31 INPUT (initial value)
- EXT 0–15 INPUT, EXT 16–31 OUTPUT

(3) Selecting a trigger input level

Select CMOS level or TTL level as the trigger input level. Select either of the following options:

- EXT 0–31 CMOS (initial value)
- EXT 0–15 TTL, EXT 16–31 CMOS

(4) Selecting a code-coverage mode

Select a code-coverage mode.
C0: Instruction coverage rate
C0 + C1: Instruction coverage rate and branch coverage rate
Up to 2 Mbytes of coverage information can be measured in C0-level coverage, while up to 1 Mbyte of coverage information can be measured in C1-level coverage. C0 coverage is selected by default.
This option can only be set in booting-up of the emulator and is only available when Code coverage has been selected in the Switching function section. If you wish to use the code coverage function after the emulator has started up, use this option to select a mode in advance.

(5) Selecting a switching function

The code coverage, data coverage and realtime profile functions cannot be used at the same time. Select one from among these functions.
Code coverage is selected by default.
The setting of this option can be changed even after the emulator has been booted up.
When the code coverage function is selected, measurements are performed at the coverage level selected under Coverage.

(6) Selecting the power voltage of the target system

Select the voltage level of the user system. This option can only be set in booting-up of the emulator and is only available when the MCU in use supports both 5.0 and 3.3 V. If the MCU only supports either 5.0 or 3.3 V, the emulator automatically selects the given voltage level.

(7) Enabling interrupts during stepped execution

Select whether interrupts should be enabled or disabled from the start of stepping until an instruction is executed. Interrupts are always accepted while a subroutine is being invoked by step-over or step-out execution.

(8) Masking the NMI pin

Select whether you want masking of input signals to the NMI pin of the target system.

(9) Masking the BREQ/WAIT pin

Select whether you want the input signal to the BREQ/WAIT pin of the target system to be masked.

(10) Usage with the EXPE bit of SYSCR (0xFFFDC2) as 1

Select this checkbox if you wish to enable access to external space while the MCU is in single-chip mode.
This option is only available for MCUs that have an EXPE bit in SYSCR. When the MCU is not in single-chip mode, this option is ignored.

(11) Debugging with overwriting of flash memory

Select this checkbox if you wish to allow rewriting of the contents of the internal ROM or EEPROM during debugging.

CAUTION
For details on debugging with overwriting of flash memory, see “Debugging with Overwriting of Flash Memory” in section 7.5 Notes on Using the MCU Unit (page 222).

(12) Display a message when the emulator is to manipulate registers in the FCU

Select the checkbox if you wish to view a message when the emulator manipulates registers in the FCU.

(13) Enable or disable IRQ input signals

Specify the hexadecimal number for the pattern of bits that corresponds to the user IRQ signals (IRQn) that you wish to monitor.
0: Disables monitoring of IRQn
1: Enables monitoring of IRQn
n = 15 to 0.
The default value is FFFF.

5.1.4 Setting up the Memory Map

The Memory map page of the Configuration properties dialog box allows the user to assign emulation memory to the internal ROM or external space.

You can allocate two areas in the internal ROM (in 128-Kbyte units) and four areas in external space (in 1-Mbyte units).

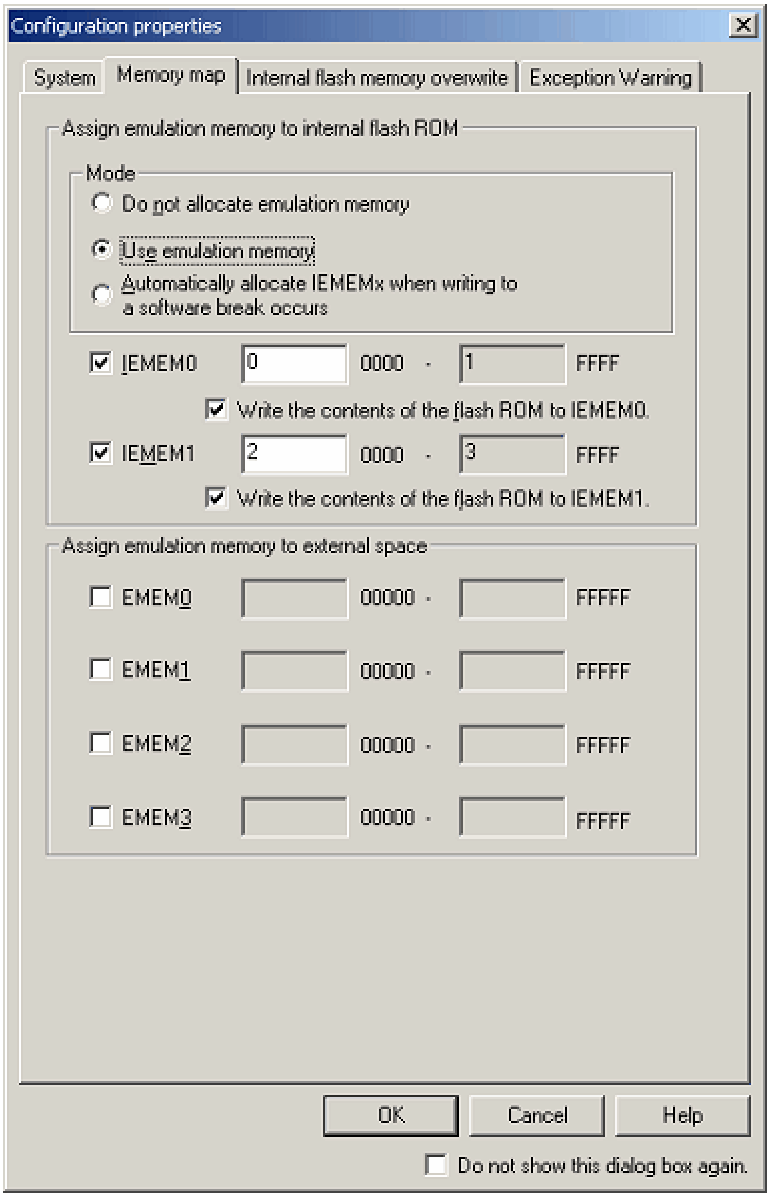


Figure 5.3 Configuration properties dialog box (Memory map page)

(1) Assign emulation memory to internal flash ROM

Up to two blocks can be allocated as emulation memory. Select the checkboxes for the areas you wish to use and specify the addresses where the blocks start and end. Note, however, that the 16 lower-order bits of the addresses are fixed because the blocks are only specifiable in 128-Kbyte units.

By default, up to two areas of emulation memory will be allocated from the first address of the internal ROM.

When emulation memory is released from allocation as internal ROM, the current data in the emulation memory will be written to the internal ROM.

(2) Selecting a mode

Select one of the following modes for assigning emulation memory to the internal ROM.
- Do not allocate emulation memory
- Use emulation memory
- Automatically allocate IEMEMx when writing to a software break occurs.

[Do not allocate emulation memory]:
The emulator will not use emulation memory allocated as internal ROM.

[Use emulation memory]:
The emulator will use emulation memory allocated as internal ROM.

[Automatically allocate IEMEMx when writing to a software break occurs]:
Emulation memory will automatically be allocated as internal ROM when an attempt is made to set a software break at a location in the internal ROM area. If you have manually assigned IEMEM0 and IEMEM1 as emulation memory to internal flash ROM, however, automatic assignment of emulation memory will not be performed.
Even when emulation memory has been allocated automatically, the allocated memory is treated as if it had been manually allocated: deleting a software breakpoint does not release the corresponding memory.

(3) Write the contents of the flash ROM areas to IEMEMx

Selecting a checkbox makes the assignment of emulation memory to internal ROM areas include writing the contents of the internal flash-ROM area to the corresponding emulation memory block. If a program is downloaded to internal ROM, the program will also be downloaded into the emulation memory block.

(4) Assign emulation memory to external space

Up to four blocks of emulation memory can be allocated to external space. Select the checkboxes for the areas you wish to use and specify the addresses where they start and end. Note that the 20 lower-order bits of the addresses are fixed because the blocks are only specifiable in 1-Mbyte units. Assignment of emulation memory to an external space is only possible at the time the emulator is booted-up.

5.1.5 Setting for Overwriting Blocks of the Flash ROM

The Internal flash memory overwrite page of the Configuration properties dialog box allows you to specify whether or not individual blocks of flash ROM should be overwritten.

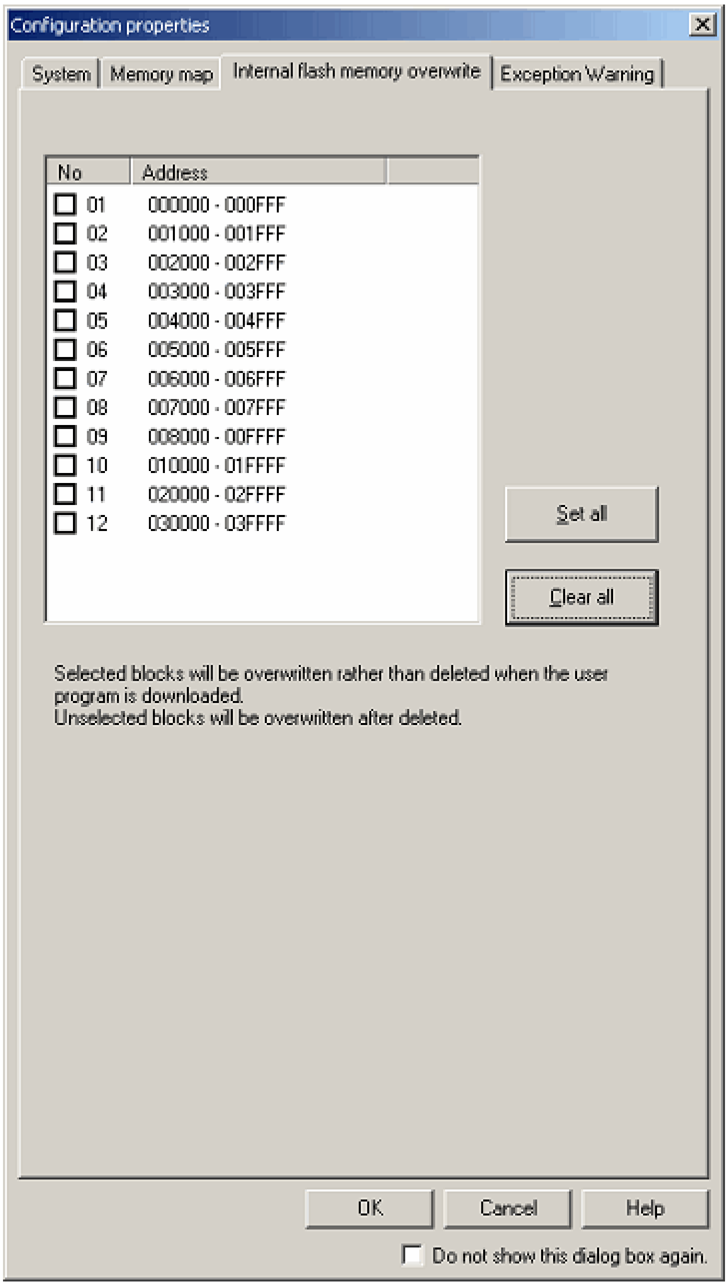


Figure 5.4 Configuration properties dialog box (Internal flash memory overwrite page)

Settings for all blocks are automatically shown in the list according to the information on the target MCU. When a checkbox is selected, the block will be overwritten rather than deleted when the user program is downloaded.

This is the same even when emulation memory has been allocated as internal ROM.

5.1.6 Settings to Request Notification of Exceptional Events

The Exception Warning page of the Configuration properties dialog box allows you to select whether or not to display warnings in the Status window and as a balloon on the status bar when exceptional events occur.

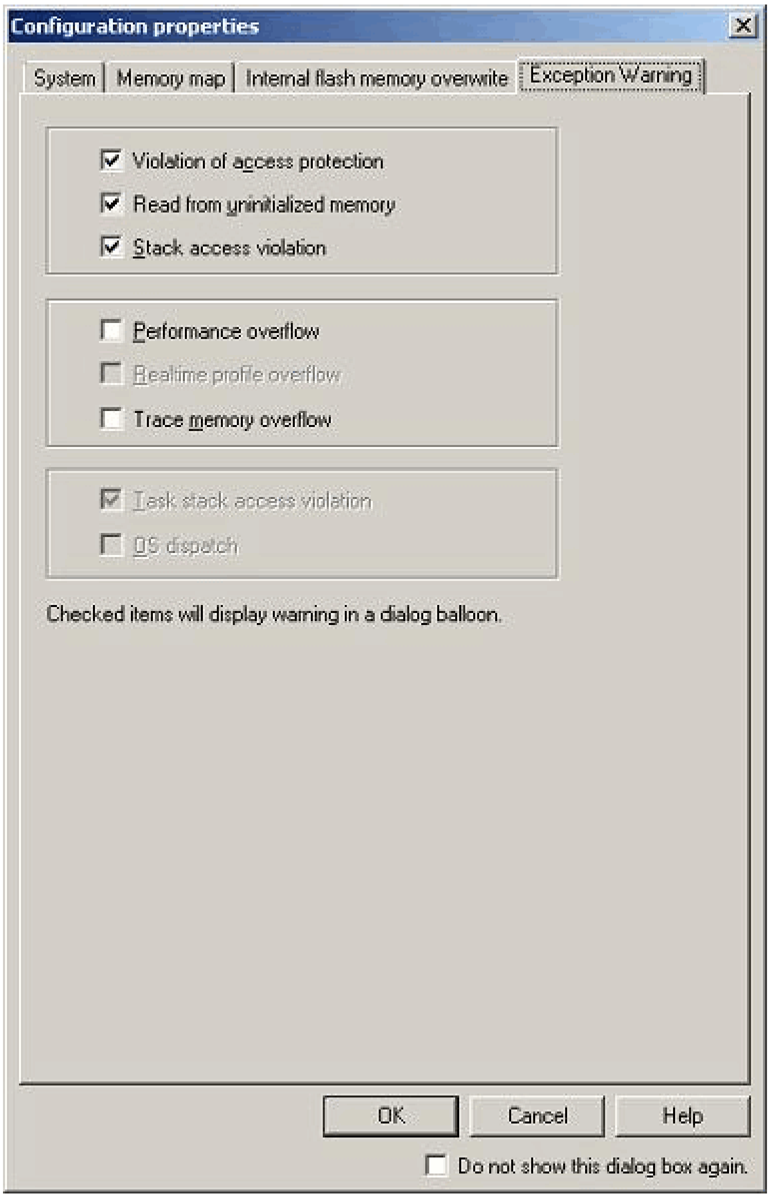


Figure 5.5 Configuration properties dialog box (Exception Warning page)

The 'Violation of access protection' , 'Read from uninitialized memory' and 'Stack access violation' checkboxes are initially selected.

When a load module that includes an OS has been downloaded, the 'Task stack access violation' checkbox is also initially selected.

Other items are non-selected by default.

If you deselect a checkbox, the corresponding item will appear as '-' in the Status window.

5.1.7 Viewing the Progress of Boot-Up Processing

You can check the progress of boot-up processing in the Connecting dialog box.
This dialog box appears when boot-up processing is started and remains open until it is completed.
As long as display of the Device setting and the Configuration properties dialog boxes continues, you cannot manipulate this dialog box.

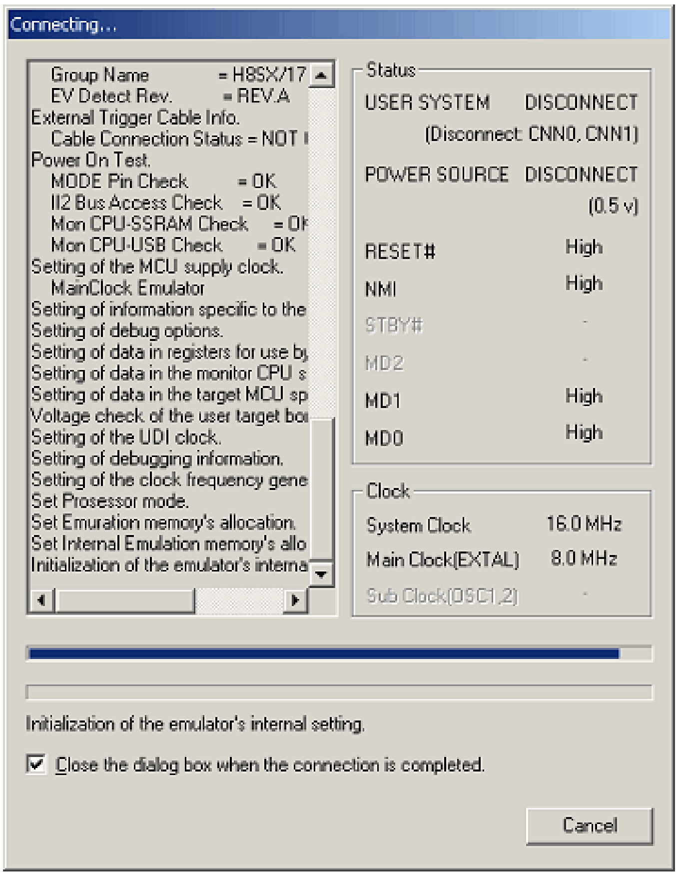


Figure 5.6 Connecting dialog box

(1) Description of progress

The progress history box on the left-hand side of the dialog box shows the history of progress.
The information shown here is saved in a bug report. To check the contents of the bug report, select Technical Support -> Create Bug Report from the Help menu.

(2) Display of pin states

The pin states are updated when you close the Configuration properties dialog box.
A warning will be shown in the progress history box if the pin states do not match the settings made in the Device setting dialog box.

(3) Display of states of clock signals

This information will be updated on completion of processing for the clock settings.
Only information on the clock signals that are actually operating is shown here.

(4) State of progress as progress bars

The upper progress bar shows the state of progress through the overall process of booting up.
The lower progress bar shows the state of progress through the current part of the process of booting up.
The name of the current part of the overall process is shown under the progress bar.

(5) Canceling the connection

Click on the Cancel button to cancel the process of booting up.

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 a desired load module or right-click on a load module under Download modules of the Workspace window and then choose Download from the popup menu.

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 load modules, refer to "4.8 Making Debugging-Related Settings" (page 76).

5.2.2 Viewing the Source Code

Select either of the following ways to view the source code.

- Double-click on the name of the source file in the Workspace window.
- Right-click on the name of the source file and choose Open from the popup menu.

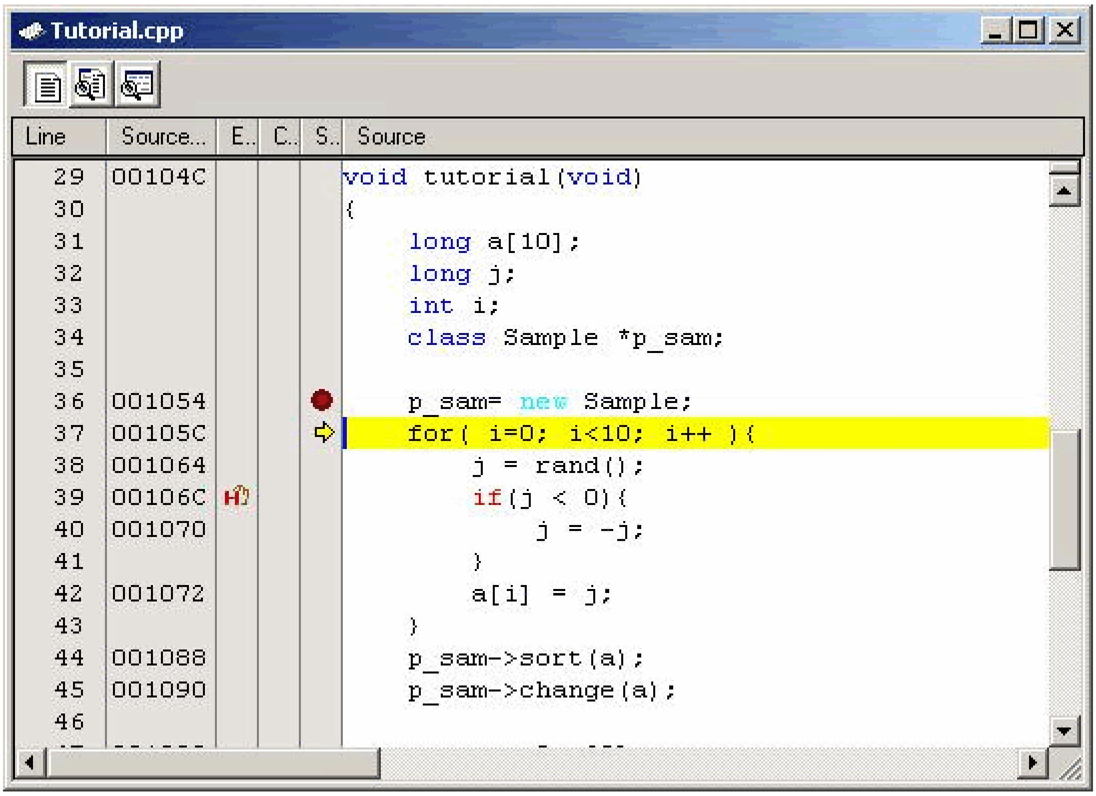


Figure 5.7 Editor window

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

(1) Line column

This column shows the line numbers of lines in the source file.

(2) Source Address column

When a program is downloaded, this column shows the addresses that correspond to the lines of the current source file. This function is convenient for determining values for the PC and where to set breakpoints.

(3) Event column

This column shows the following:

Table 5.2 Icons in the Event column

	Hardware breakpoint
	Trace point (fetch condition)

A hardware breakpoint can be set by double-clicking in the Event column.
A trace point is only displayed when a fetch condition has been set.
[*] after the title on the title bar of the Hardware break, Trace conditions and Performance Analysis Conditions dialog boxes shows that a setting is being edited. You cannot change the settings from the Event column of the Editor window while editing is in progress.

(4) Code Coverage column

This column graphically shows the C0 code coverage information.

(5) S/W Breakpoints column

This column shows the following:

Table 5.3 Icons in the S/W Breakpoints column

	Bookmark
	Software break
	PC position

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 popup menu.

2. The Global Editor Column States dialog box will be displayed.

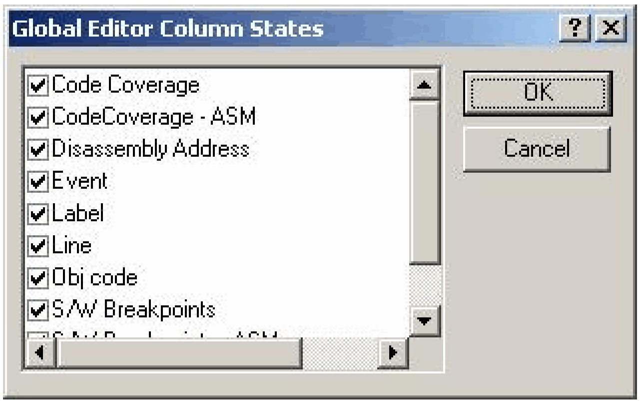


Figure 5.8 Global Editor Column States dialog box

3. Deselect the checkboxes of columns you want to turn off. Click the OK button, and the new settings you have made will take effect.

5.2.4 Turning columns off for one source file

(1) From the Editor window

1. Right-click in the Editor window and choose Columns from the popup menu.

2. A cascaded menu will be displayed. A check mark is to the left of the names of currently enabled columns.

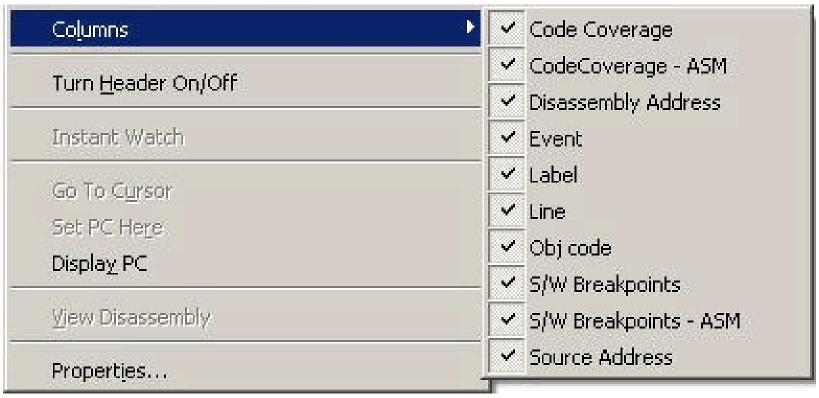


Figure 5.9 Popup menu window

3. Clicking on a column name toggles the setting between enabling and disabling of the column.

5.2.5 Viewing Assembly Language Code

While a source file is open, click the right mouse button in the Editor window and choose View Disassembly from the popup menu. The Disassembly window will be displayed.
The first address shown in the Disassembly window corresponds to the cursor position in the Editor window.
You can also use the View Disassembly button in the Editor window to view code produced by disassembly.
If there is no source file, you can still view the disassembly by one of the following methods.

– Click on the Disassembly toolbar button [ ].

– Choose Disassembly from the View menu.
– Use the “Ctrl + D” shortcut keys.

In this case, the Disassembly window opens with a listing from the position currently indicated by the PC.

The emulator also supports a mixed mode as an optional way to show all source lines from the address where disassembly started. To view disassembly code in mixed mode, click the View mixed mode button.

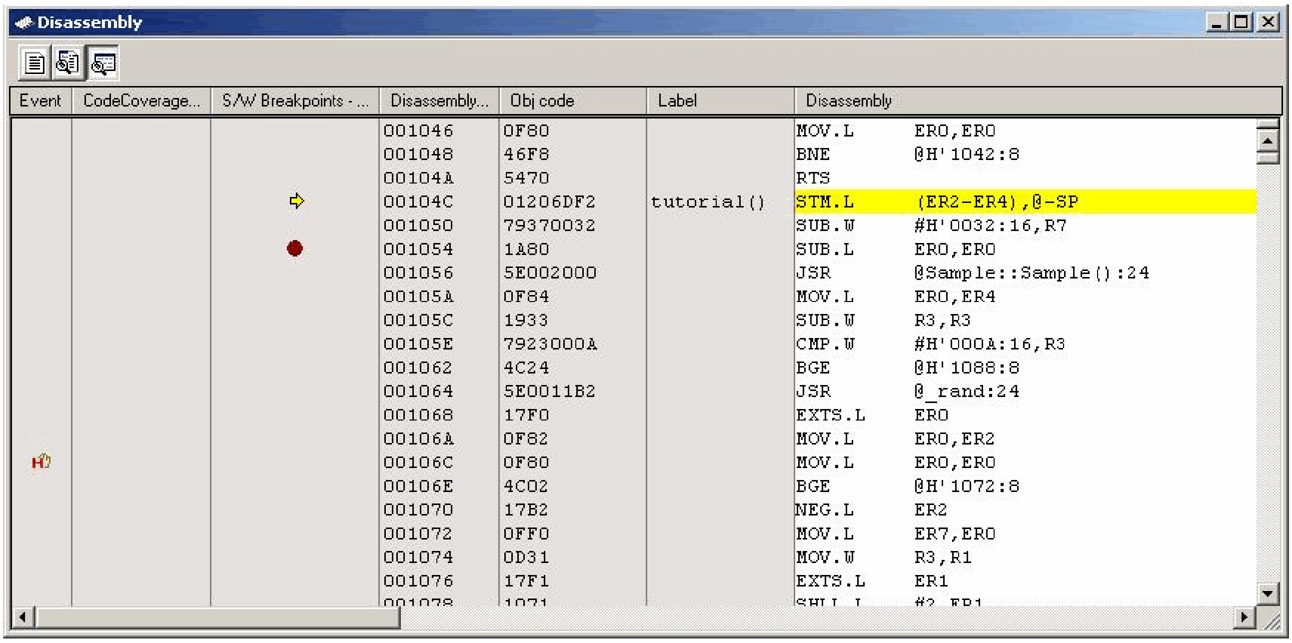


Figure 5.10 Disassembly window

The columns listed below are to the left of the Disassembly column.

(1) Event column

This column shows the following:

Table 5.4 Icons in the Event column

H	Hardware breakpoint
	Trace point (fetch condition)

A hardware breakpoint can be set by double-clicking in the Event column.
A trace point is only displayed when a fetch condition has been set.

(2) Code Coverage - ASM column

This column graphically shows the C0 code coverage information.

(3) S/W Breakpoints - ASM column

This column shows the following:

Table 5.5 Icons in the S/W Breakpoints – ASM column

	Software break
	PC position

(4) Disassembly Address column

This column shows the address of the machine code corresponding to the disassembly. Double-clicking in this column brings up the Set Address dialog box. Enter the address where you want the display of disassembly code to start in this dialog box.

(5) Obj code column

This column shows the object code.

(6) Label

This column shows labels. This column is not usable if no module has been downloaded.

5.2.6 Correcting Assembly Language Code

Double-click on the instruction you want to correct in the Disassembly window or choose Edit from the popup menu. The Assembler dialog box will open. Use this dialog box to correct the assembly-language code.

Figure 5.11 Assembler dialog box

The dialog box shows the address, instruction code and mnemonic of the selected instruction.
Enter a new instruction (or edit the old instruction) in the Mnemonic edit box. When you have finished, hit the Enter key. The value in memory is overwritten by the new instruction code, and the pointer is moved to the next instruction.
Click on the OK button to overwrite the current value in memory with the new instruction code and close the dialog box.

CAUTION
The assembly-language code shown in the Disassembly window and the Assembler dialog box is based on the data currently in memory. When you modify data in memory, the new assembly-language code is shown in the Disassembly window and the Assembler dialog box. However, the source file being displayed in the Editor window remains unchanged, even if it includes assembly-language code.

5.3 Viewing Memory Data in Real Time

5.3.1 Viewing Memory Data in Real Time

Use the RAM Monitor window to monitor data in memory while the user program is running.
The RAM monitoring function permits recording and inspection of the data in an area of memory for which monitoring has been assigned and the states of access in real time without obstructing execution of the user program.
The RAM Monitor window shows the access states (read, written, non-initialized or not inspected) in different colors.

(1) Allocating an area for RAM monitoring

A 16-Kbyte RAM monitoring area is provided.
This RAM monitoring area can be allocated to a desired contiguous address range or up to 32 blocks of 512 bytes.
By default, a maximum of 16 Kbytes of space from the first address of the internal RAM is allocated as the RAM monitoring area.

(2) Monitor display

Access states are indicated by different background colors according to the access attribute as listed below (the background colors are customizable).
The access attributes "read" and "written" indicate the last access to each memory location.
To view detected errors, choose Error Detection Display from the popup menu. In this case, the information on reading and writing is not displayed.

Table 5.6 Access attribute and background color

Access attribute		Background color
Read		Green
Written		Red
Error detected	Non-initialized memory (the location has been read but nothing has been written to it yet)	Yellow
	Non-inspected memory (a value has been written to the location but it has not been read)	Sky blue
No access		White

CAUTION
The contents of the RAM Monitor window are acquired from bus access. Therefore, changes made to memory by access that was not through the user program (e.g. writing to memory directly from external I/O) are not reflected in the RAM Monitor window.

(3) Detecting reading from non-initialized areas

If a memory location is read but nothing has been written to that location, the emulator detects "a non-initialized area" and indicates the error.
To view errors of this type, choose Error Detection Display from the popup menu.
Non-initialized memory locations are shown against a yellow background.
Errors of this type can be detected as exceptional events and used as conditions of hardware breakpoints and trace points (also refer to "5.14 Detecting Exceptional Events" (page 187)).

(4) Detecting non-inspected areas

If a memory location has been initialized but has not been read, the emulator detects this as "a non-inspected area" and indicates the error.
To view errors of this type, choose Error Detection Display from the popup menu.
Non-inspected memory locations are shown against a sky blue background.

5.3.2 Setting the Update Interval for RAM Monitoring

Choose Update Interval Setting from the popup menu of the RAM Monitor window. The Update Interval Setting dialog box shown below will appear.

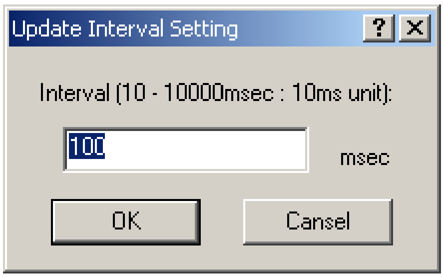


Figure 5.12 Update Interval Setting dialog box

A separate Update Interval can be specified per RAM Monitor window.
The initial value is 100 ms.

5.3.3 Clearing RAM Monitoring Access History

Choose Access Data Clear from the popup menu of the RAM Monitor window. The history of all access to the RAM monitoring area will be cleared.

CAUTION
If clearing proceeds while the user program is being executed, the realtime characteristic of execution may be lost because clearing produces a memory dump.

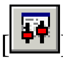
5.3.4 Clearing RAM Monitoring Error Detection Data

Choose Error Detection Data Clear from the popup menu of the RAM Monitor window. All information on the detected errors in the RAM monitor area will be cleared.

5.4 Viewing the Current Status

5.4.1 Viewing the Emulator Status

To find out the current status of the emulator, open the Status window.

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

The information shown in this window is not updated while the program is running.

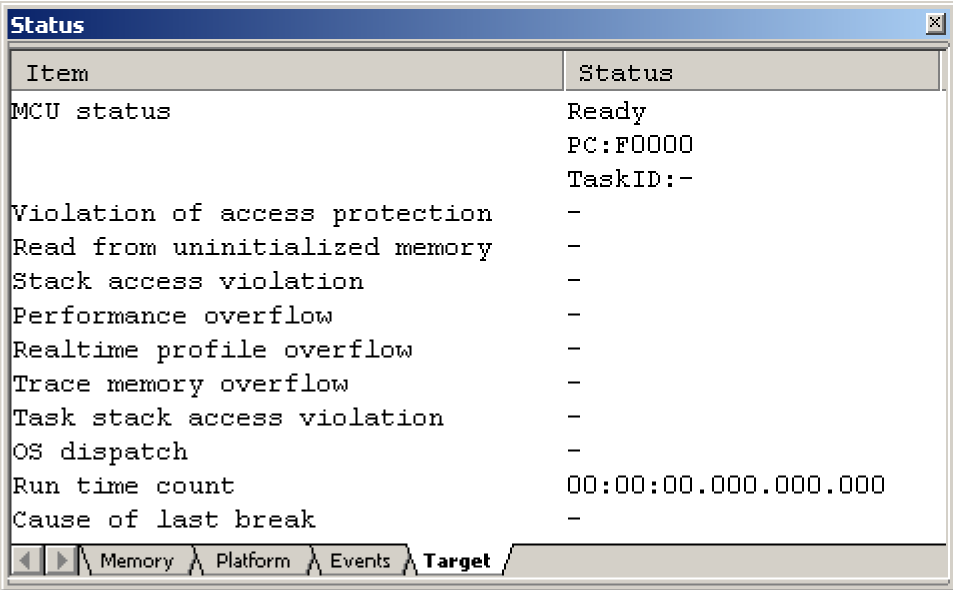


Figure 5.13 Status window

The Status window has the following four sheets.

Table 5.7 Sheets of the Status window

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

5.4.2 Viewing the Emulator Status in the Status Bar

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

Right clicking on the status bar brings up a list of the available items. Check the items you want to view in the status bar.

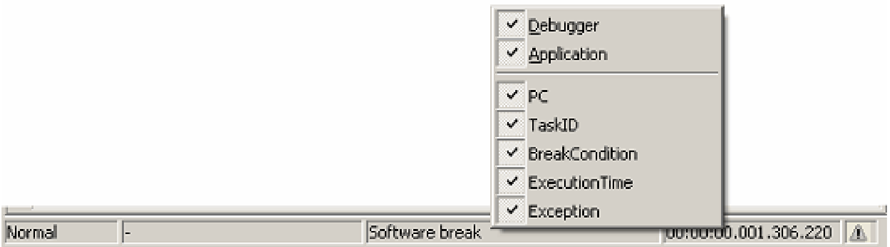


Figure 5.14 Status bar

Table 5.8 Items regarding emulator status shown in the status bar

Item	Description
PC	PC value During execution: PC value During a break: Normal
Task ID	Task ID, task entry label
BreakCondition	Source of a break in the user program
ExecutionTime	Result of time measurement
Exception	Whether or not an exceptional event has occurred

(1) When more than one break source is present

When you click on the status bar indicating the source of a break ("Some factors exist" when there is more than one), a balloon appears.

Read the contents of the balloon to check the source of the break.

Figure 5.15 Checking the source of a break

(2) When an exceptional event has occurred

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

However, exceptional events of types that are not selected on the Exception Warning page of the Configuration Properties dialog box are not shown.

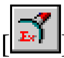
Figure 5.16 Example of warning display when exceptional events have occurred

5.5 Periodically Reading Out and Showing the Emulator Status

5.5.1 Periodically Reading Out and Showing the Emulator Information

To find out about changes in emulator information whether the user program is running or idle, use the Extended Monitor window.

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

To open the Extended Monitor window, choose CPU -> Extended Monitor from the View menu, or click on 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

"Main Clock (System Clock)" can only be measured while the user program is running.

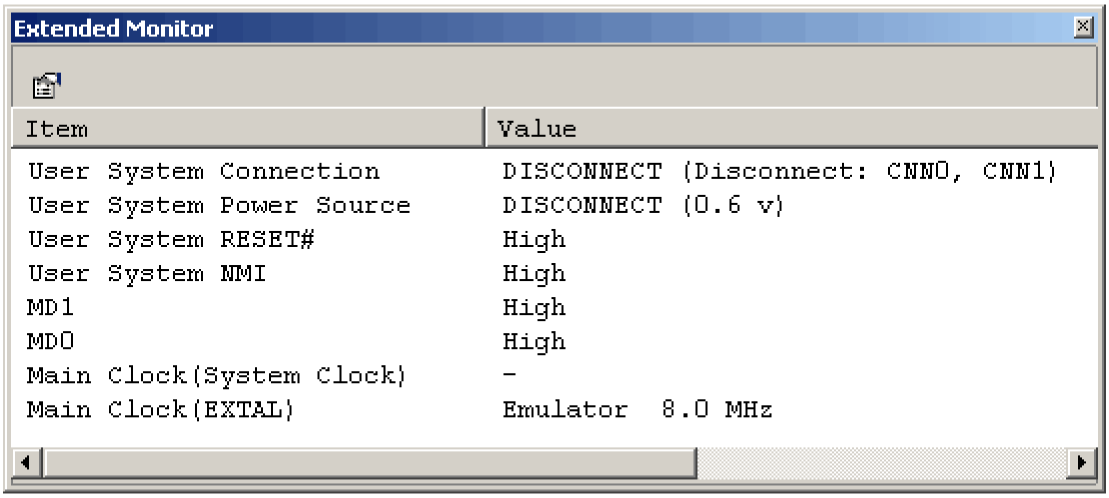


Figure 5.17 Extended Monitor window

5.5.2 Selecting the Items to Be Displayed

Choose Properties from the popup menu of the Extended Monitor window. The Extended Monitor Configuration dialog box will be displayed.

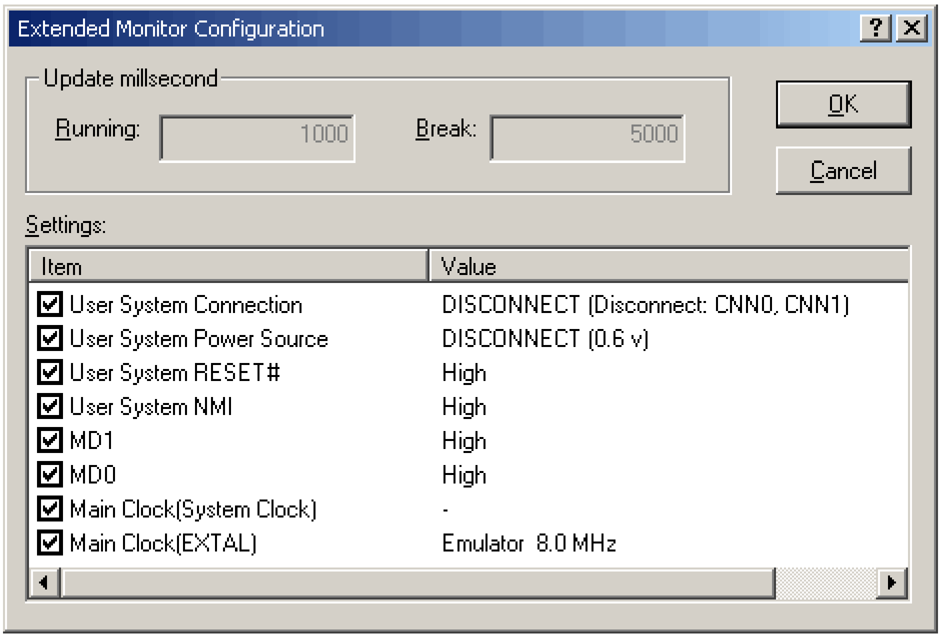


Figure 5.18 Extended Monitor Configuration dialog box

This dialog box allows you to select items to be shown in the Extended Monitor window.

5.6 Using Software Breakpoints

5.6.1 Using Software Breakpoints

In a software break, the instruction code at a specified address is replaced with a BRK instruction, which causes the user program to stop running by generating a BRK interrupt. In that sense, this is a pre-execution break function.
Up to 4096 breakpoints can be set.
If multiple software breakpoints are set, program execution breaks when it arrives at any of the breakpoints reached.

(1) When stopped at a software breakpoint

When the program you have created is run and arrives at an address you have set as a software breakpoint, the program stops and the message “Software Break” is displayed on the Debug sheet of the Output window. At this time, the Editor or Disassembly window is updated, and the position where 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 the software breakpoint is set. If Go or Step is selected after the program has stopped at the breakpoint, the program restarts from the line marked with an arrow.

5.6.2 Adding and Removing Software Breakpoints

Select either of the following ways to add or remove software breakpoints.

- From the Editor or Disassembly window
- From the Breakpoints dialog box (only for removal)
- From the command line

(1) From the Editor or Disassembly window

1. Check that the Editor or Disassembly window that is currently open shows the position at which you want to set a software breakpoint.

2. In the S/W Breakpoints column, double-click on the line where you want the program to stop.

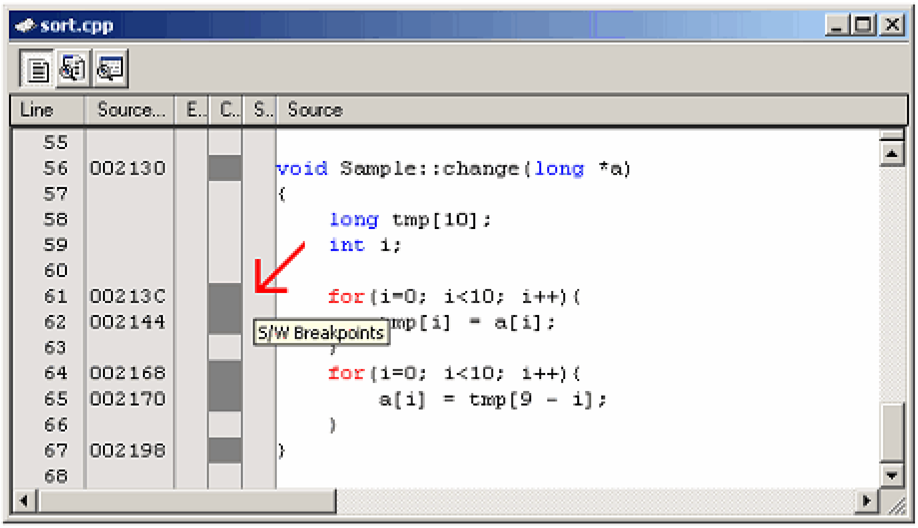


Figure 5.19 Editor window

Alternatively, you can select Toggle Breakpoint from the popup menu or press the F9 key.

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.

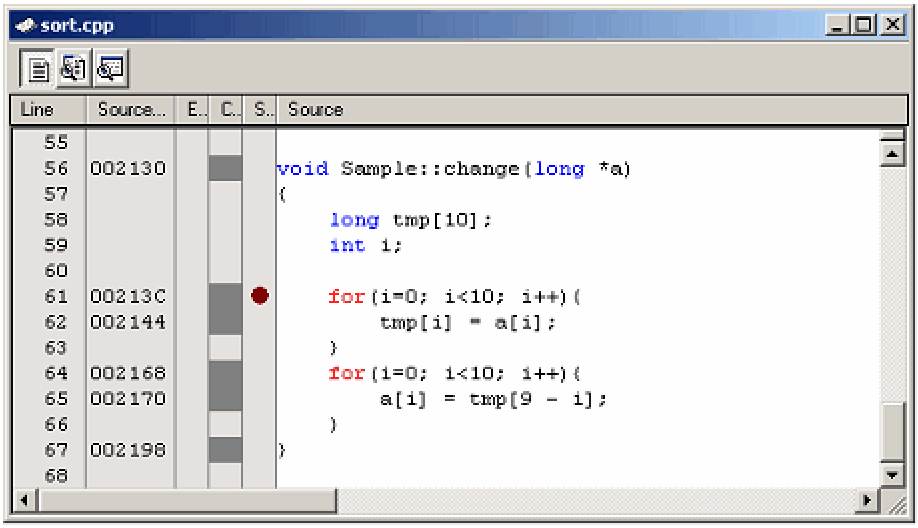


Figure 5.20 Editor window

Double-clicking one more time removes the breakpoint.

5.6.3 Enabling and Disabling Software Breakpoints

Select one of the following ways 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 popup menu. Alternatively, press the Ctrl and F9 keys at the same time.

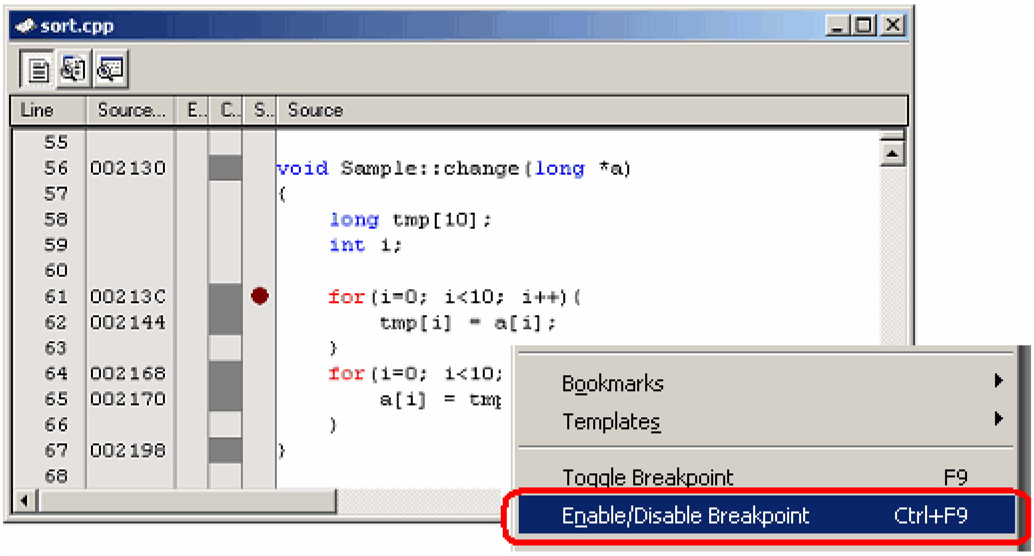


Figure 5.21 Editor window and popup menu

2. The software breakpoint is alternately enabled or disabled.

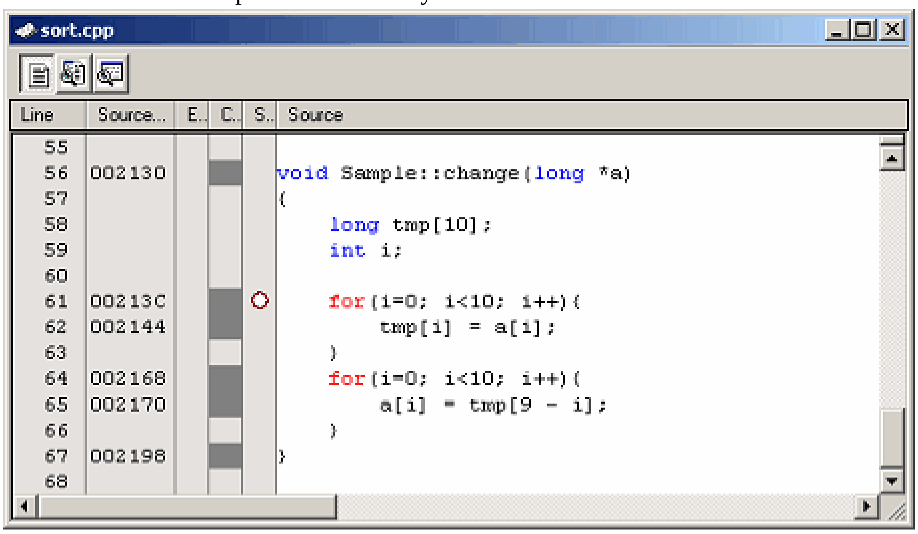


Figure 5.22 Editor window

(2) From the Breakpoints dialog box

1. Select Source Breakpoints from the Edit menu to bring up the Breakpoints dialog box. In this dialog box, you can alternately enable, disable, or remove a currently set breakpoint.

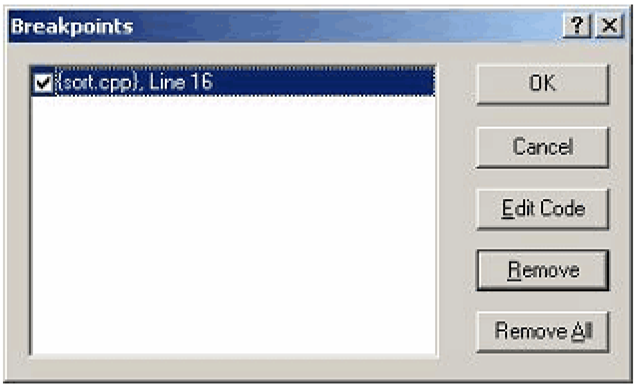


Figure 5.23 Breakpoints dialog box

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 an event you have set as a condition for the break, trace or performance-analysis function.

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

These 16 points can be placed as desired.

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

(1) Types of events

Events are of the following types.

Table 5.9 Event types

Instruction fetch	The emulator detects that the CPU has executed the instruction at the specified address. Detection is in the cycle of execution by the CPU rather than the cycle of prefetching by the instruction queue.
Data access	The emulator detects access under a specified condition to a specified address or specified address range.
Interrupt	The emulator detects interrupt generation or return from an interrupt handler.
Trigger input	The emulator detects a signal fed in from the input cable for external trigger signals being in a specified state.

(2) Event combinations

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

Table 5.10 Types of event combination

OR	The condition is met when any one of the specified events occurs.
AND (cumulative)	The condition is met when all of the specified events occur regardless of the timing.
AND (simultaneous)	The condition is met when all of the specified events occur at the same time.
Subroutine	The condition is met when a specified event occurs within a specified address range.
Sequential	The condition is met when the specified events occur in a specified order.
State transitions	The condition is met when the events occur in an order specified in the state transition diagram.

5.7.2 Adding Events

Select one of the following ways 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

[Creating an event in the Hardware Break, Trace conditions, or Performance Analysis Conditions dialog box]

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

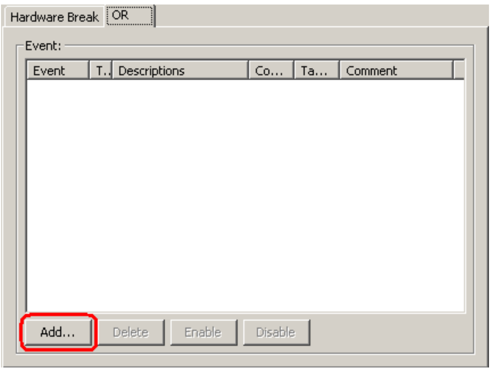


Figure 5.24 Hardware Break dialog box

2. The Event dialog box shown below will be displayed. In this dialog box, set the details of the event condition and then click on the OK button.

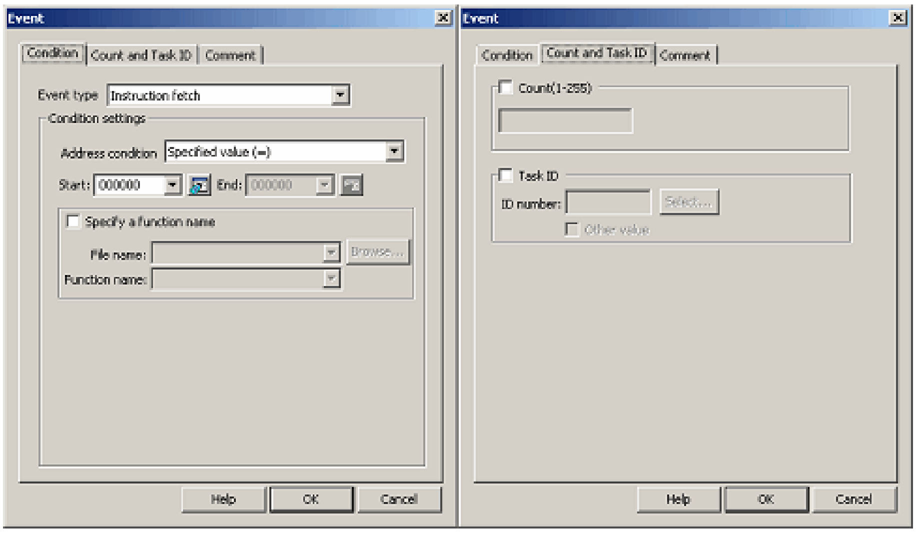


Figure 5.25 Event dialog box

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

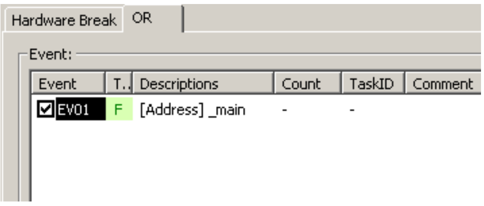


Figure 5.26 Hardware Break dialog box

4. If you create an event that would make the total number of events exceed 16, an error message is displayed. In this case, the event you have added is invalid.

[Adding an event from the Registered Events dialog box]

1. Click on the Add button in the Registered Events dialog box.

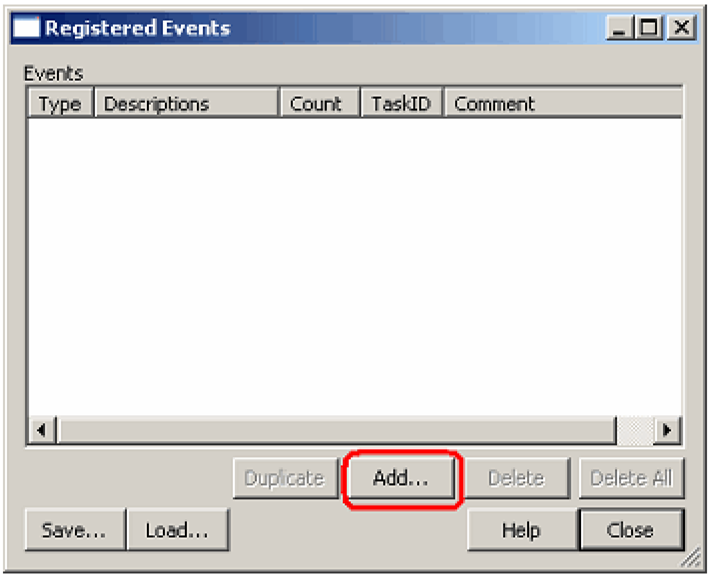


Figure 5.27 Registered Events dialog box

2. The Event dialog box shown below will be displayed. Set details of the event condition in this dialog box. Enter a comment if any is necessary. Then click on the OK button.

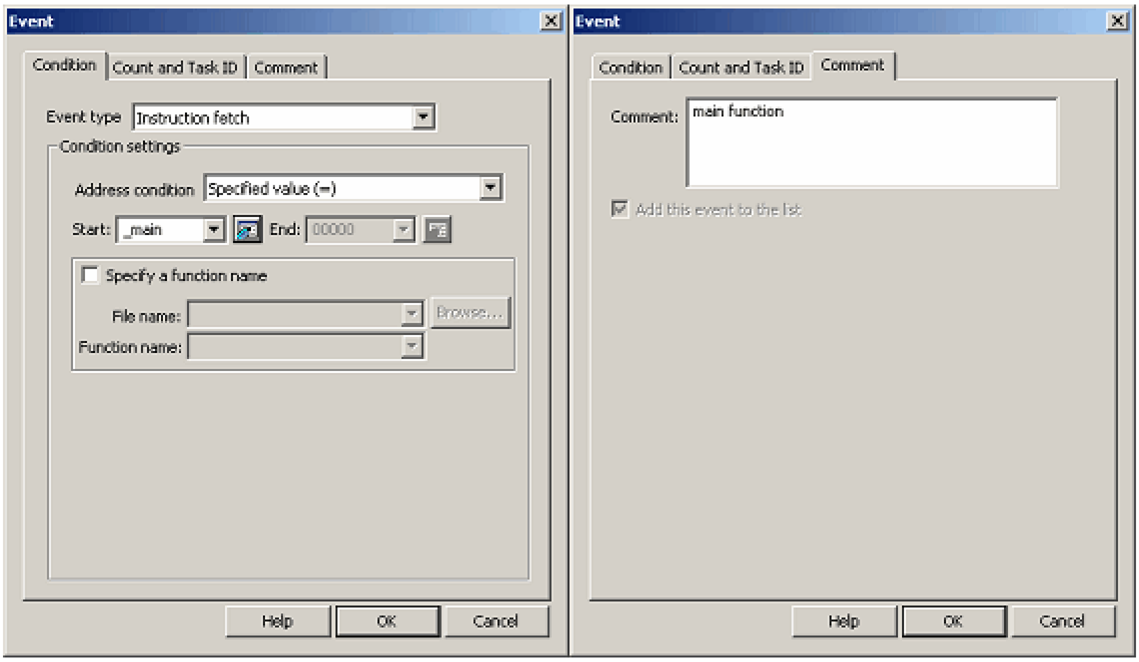


Figure 5.28 Event dialog box

3. The event is added to the list of registered events.

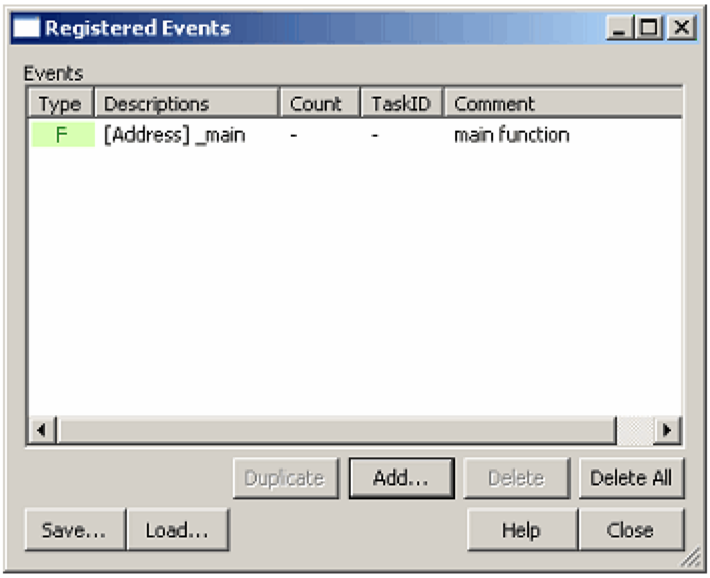


Figure 5.29 Registered Events dialog box

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

[Adding a hardware breakpoint]

1. Select HW Break Point from the popup menu opened by double-clicking or right clicking in the Event column of the Editor window.
   This sets fetching from the corresponding address as the condition for a hardware breakpoint, i.e an instruction fetch condition.

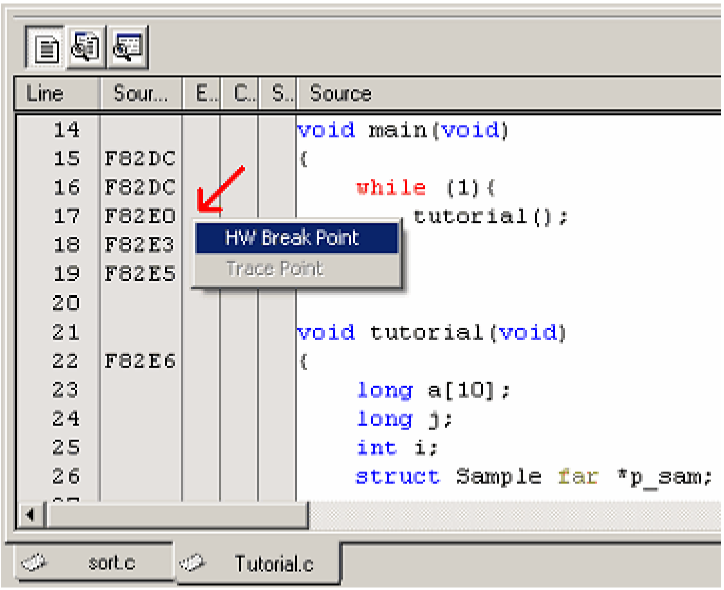


Figure 5.30 Editor window

2. If the number of events currently set allows room for another, the event you have added from the Editor window is added as an OR condition. If there is no room, an error message is displayed.

CAUTION

If you are editing the contents of the Hardware Break dialog box, you cannot set a hardware breakpoint from the Event column of the Editor window.

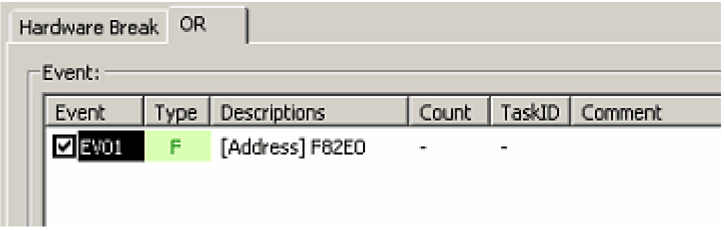


Figure 5.31 Hardware Break dialog box

[Adding a trace point]

1. Double-click or right click in the Event column of the Editor window then select Trace Point from the popup menu. This sets fetching from the corresponding address as the condition for a trace point, i.e an instruction fetch condition. Double-click on the instruction fetch event in the Event column of the Editor window to delete it.

CAUTION

Trace points cannot be set in the Event column of the Editor window in the following cases.

- The contents of the Trace conditions dialog box are being edited.

- The selected trace mode is Fill until stop or Fill until full.

(3) Adding events by dragging and dropping

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

1. By dragging and dropping a variable name into the Event column, you can set access to that variable as an event to be detected, i.e. a data-access condition.
   At this time, the size of the variable is automatically set as a condition of the data access event.
   Only global or static variables taking up 1, 2, or 4 bytes can be registered for event detection. Static variables in functions cannot be registered.
2. By dragging and dropping a function name into the Event column, you can set instruction fetching from the address where that function starts as an event to be detected.

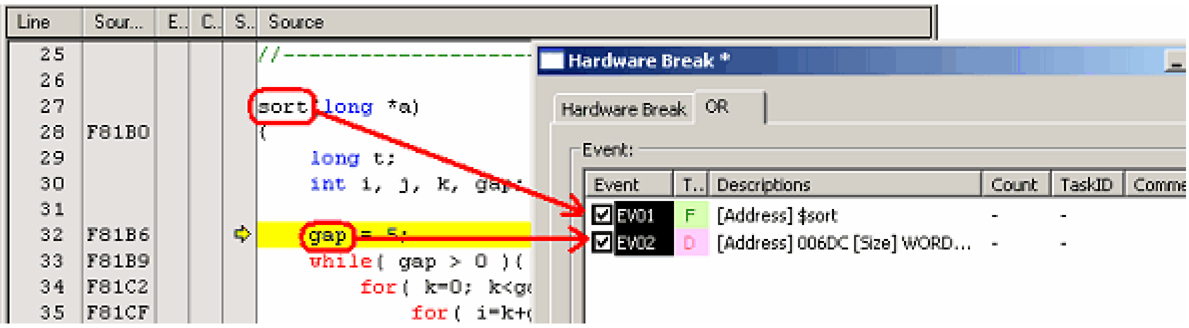


Figure 5.32 Editor window and Hardware Break dialog box

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

Select an address range in the Memory window and drag and drop it into the Event column. In this way, you can set access to an address in the selected address range as a data access event to be detected, i.e. a data access condition.

[Dragging and dropping a label in the Label window]

You can set fetching from the label as an event to be detected, i.e. an instruction fetch condition.

5.7.3 Removing Events

The following ways of removing events are available.

[Deleting an event from the Hardware Break, Trace conditions, or Performance Analysis Conditions dialog box]

1. To remove one point, select the line you want to remove in the Event list and then click on the Delete button (or use the keys Ctrl + Del instead of clicking on the button).
   The selected event will be removed from the Event list.

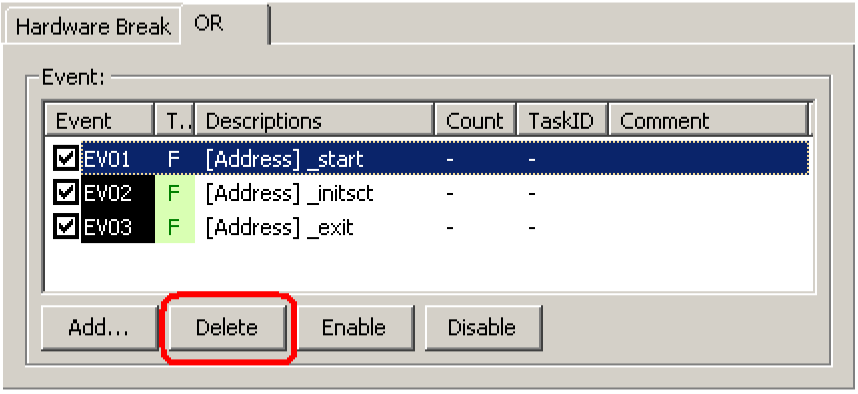


Figure 5.33 Hardware Break dialog box

2. To remove multiple events, hold down the Shift or the Ctrl key while you select lines you want to remove in the Event list and then click on the Delete button (or use the keys Ctrl + Del instead of clicking on the button).
   The selected events will be removed from the Event list.

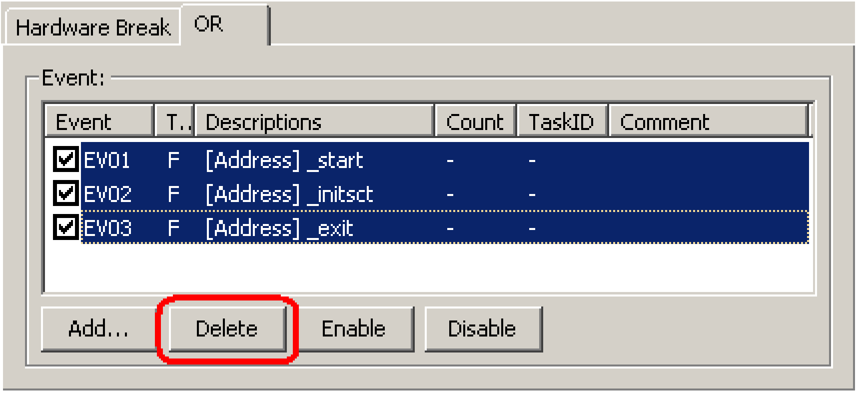


Figure 5.34 Hardware Break dialog box

[Deleting an event from the Registered Events dialog box]

To remove one point, select the line you want to remove in the Registered Events dialog box and then click on the Delete button (or use the keys Ctrl + Del instead of clicking on the button).
The selected event will be removed from the list of registered events.
To delete all events, click on the Delete All button.

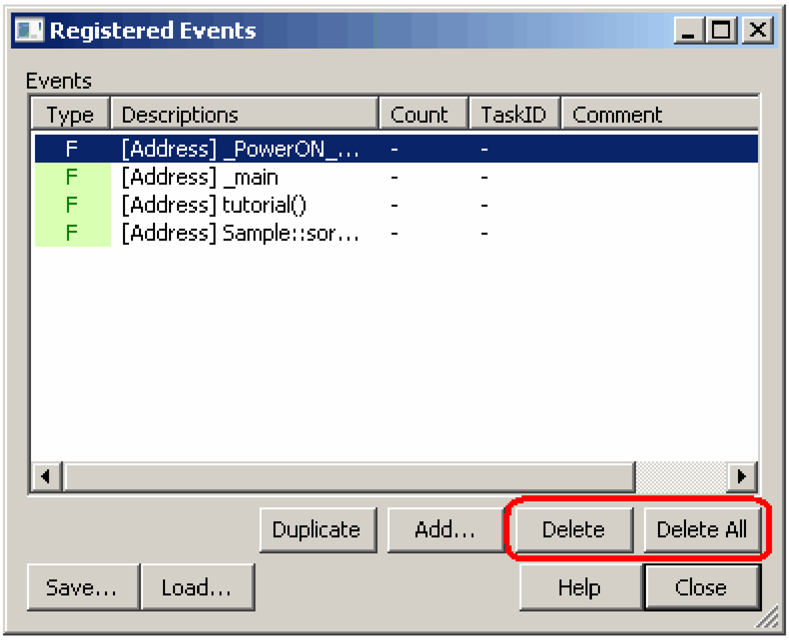


Figure 5.35 Registered Events dialog box

5.7.4 Registering Events

"Registering an event" refers to placing an event in the list of registered events. A registered event can be reused at a later time. Select one of the following ways to register an event. Up to 256 events can be registered.

(1) Registering events

[Creating an event in the Event dialog box]

1. Open the Comment page of the Event dialog box and select the "Add this event to the list" checkbox. Then click on the OK button.

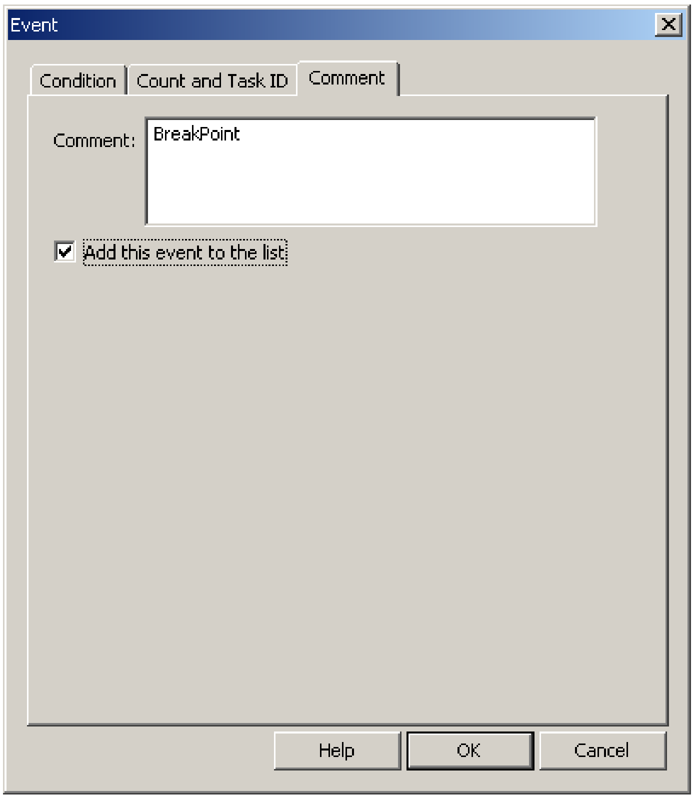


Figure 5.36 Event dialog box

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

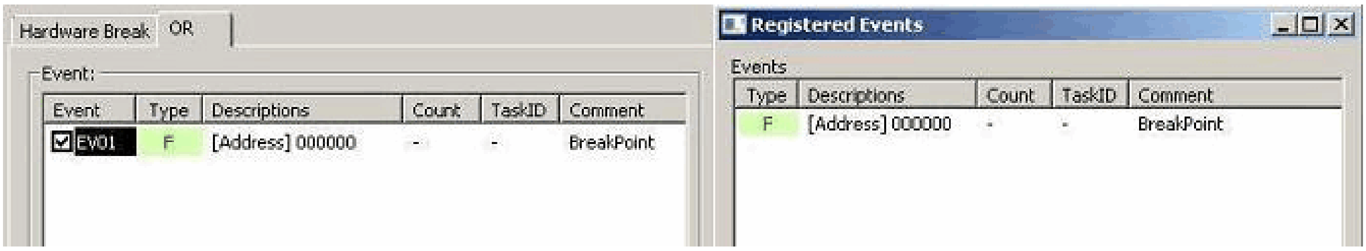


Figure 5.37 Hardware Break dialog box and Registered Events dialog box

[Registering an event by dragging and dropping]

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

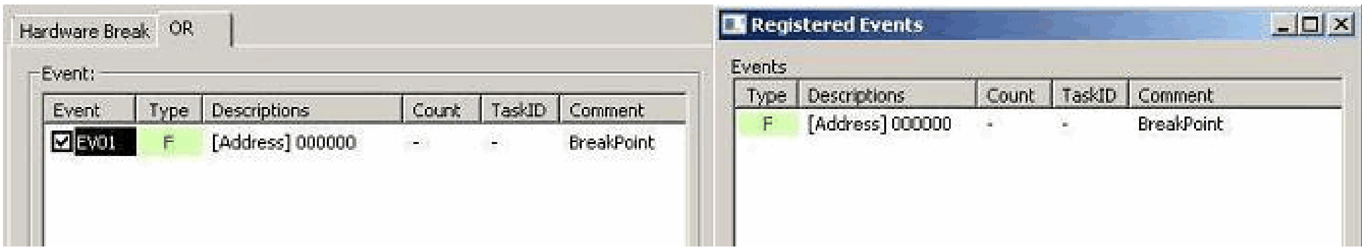


Figure 5.38 Hardware Break dialog box and Registered Events dialog box

[Registering an event in the Registered Events dialog box]

Click on the Add button to create an event. Any event you create here is added to the Registered Events dialog box.

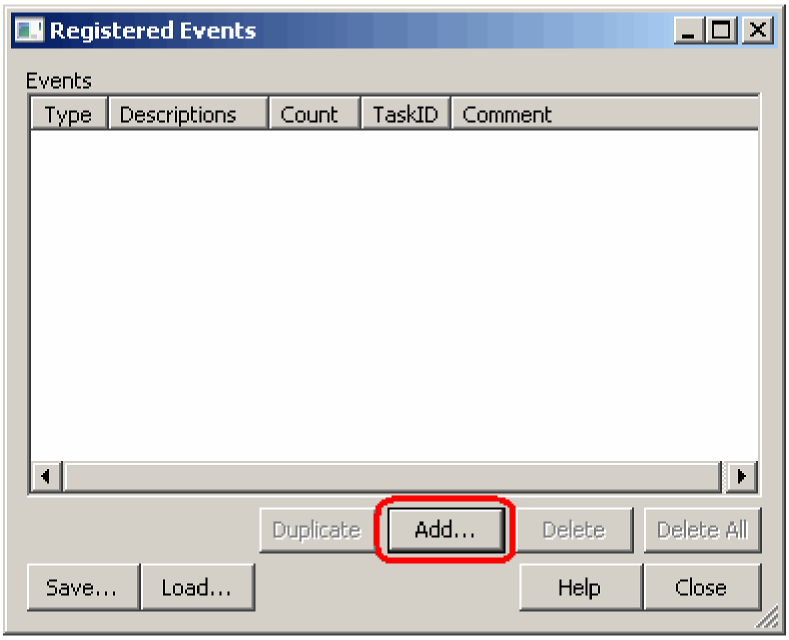


Figure 5.39 Registered Events dialog box

(2) Attaching comments

An explanatory comment for the event can be attached. Check the Registered Events dialog box to see the registered events and comments.

5.7.5 Creating Events for Each Instance of Usage or Reusing Events

The following two approaches are available for setting events in the Hardware Break, Trace conditions, or Performance Analysis Conditions dialog box.
One is to create events in the dialog box each time they are to be used. The other is to choose a condition from the Registered Events dialog box and drag and drop it into the Event list in the Hardware Break, Trace conditions, or Performance Analysis Conditions dialog box.
Here, we refer to the former as creating events per usage and the latter as reusing events.

[Creating events per usage]

Select this method if you intend to use a specific condition only once. The event you have created is used without ever being registered.
Once the event is no longer in use (i.e., it has been changed or deleted), its setting is nonexistent.
Any event created by a simple operation such as double-clicking in the Event column of the Editor window constitutes an event created per usage.

[Reusing events]

Any event registered in the Registered Events dialog box can be reused by dragging and dropping it into the Event list in the Hardware Break, Trace conditions, or Performance Analysis Conditions dialog box.

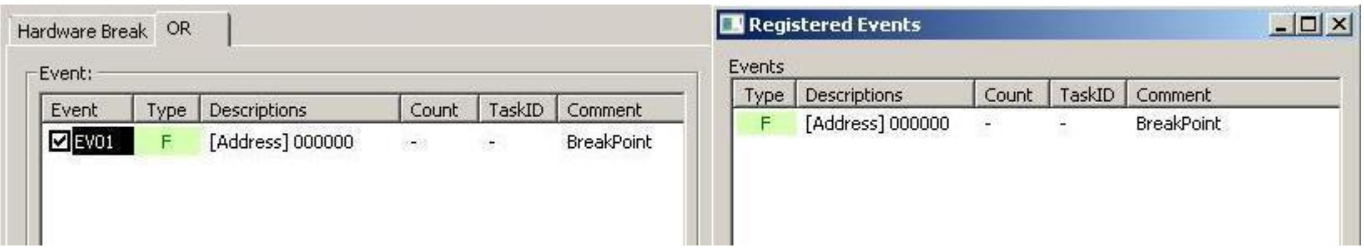


Figure 5.40 Reusing an event

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

An event in the Registered Events dialog box can be dragged and dropped into multiple dialog boxes.
If a condition of an event is altered after the event has been dragged and dropped, the alteration is not reflected in the setting of the original event in the Registered Events dialog box.

(2) Registering duplicates in the Registered Events dialog box

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

5.7.6 Activating Events

To activate the settings for events that you have created, click on the Apply button. Settings you make do not become effective until you click on the Apply button.

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

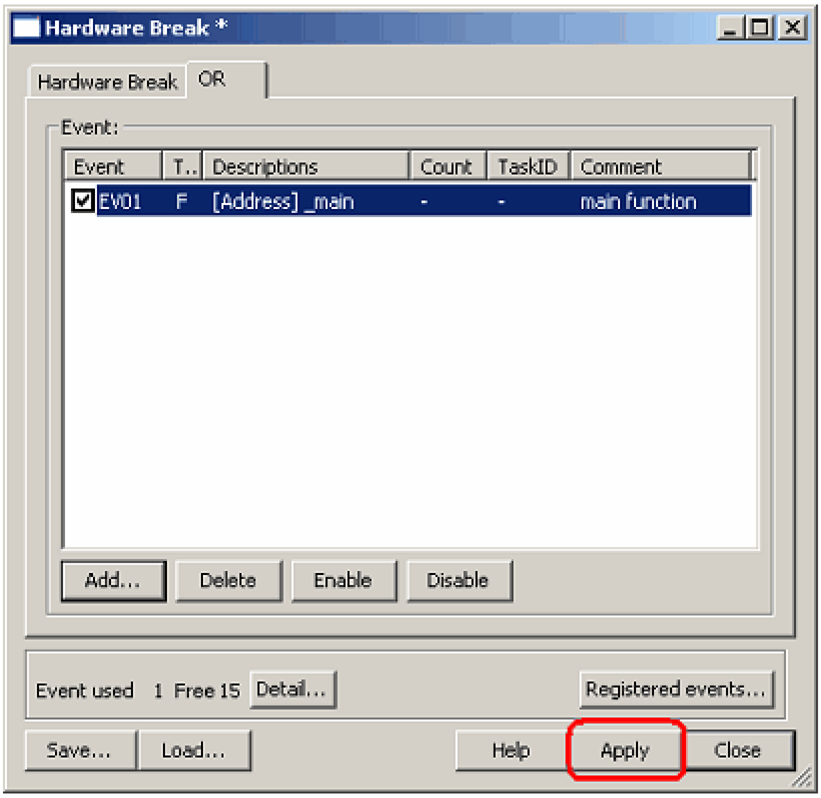


Figure 5.41 Activating the settings

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 specific 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 a hardware breakpoint, you can set an OR condition, other conditions (AND (cumulative), AND (simultaneous), subroutine, sequential or state transitions) and detection of exceptional events.
For each hardware breakpoint, you can specify all or only one from among the OR condition, other conditions, and detection of exceptional events.

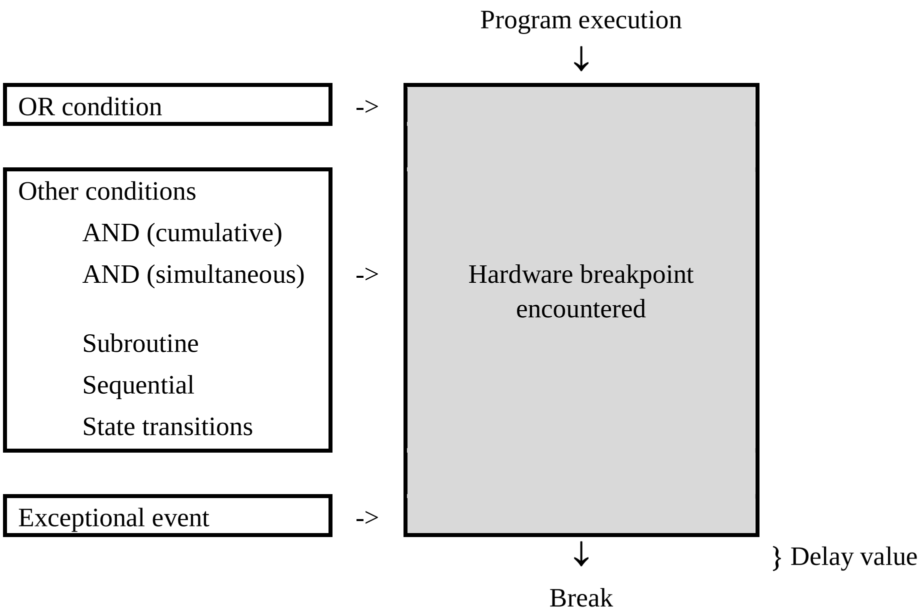


Figure 5.42 A hardware break in outline

(2) Setting an OR condition

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.
When the OR condition is re-enabled, the previous event settings on the OR page (with their checkboxes being selected) are restored.
However, if re-enabling the OR condition would bring the total number of events to more than 16, the events are restored with their checkboxes not selected (disabled) on the OR page.

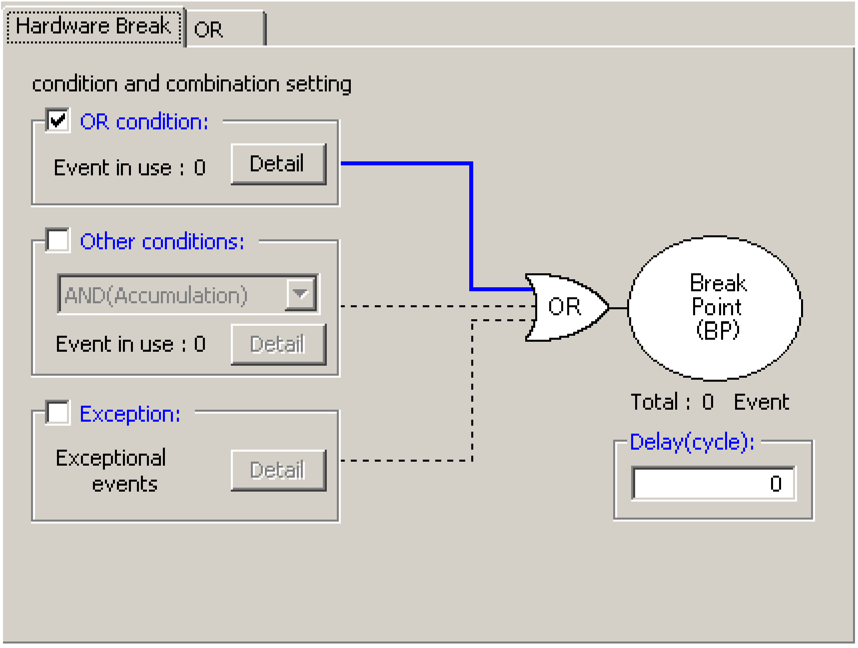


Figure 5.43 Hardware Break dialog box

Table 5.11 OR condition

Type	Description
OR condition	A breakpoint is encountered when any of the specified events occurs.

(3) Setting other conditions

You can select one from among five available choices: AND (cumulative), AND (simultaneous), Subroutine, Sequential and State transitions. To set any condition, select the checkbox to the left of “Other conditions.” Other conditions are disabled by default (the checkbox to the left of “Other conditions” is not selected). Cumulative AND is listed as “AND(Accumulation)” in the dialog box.

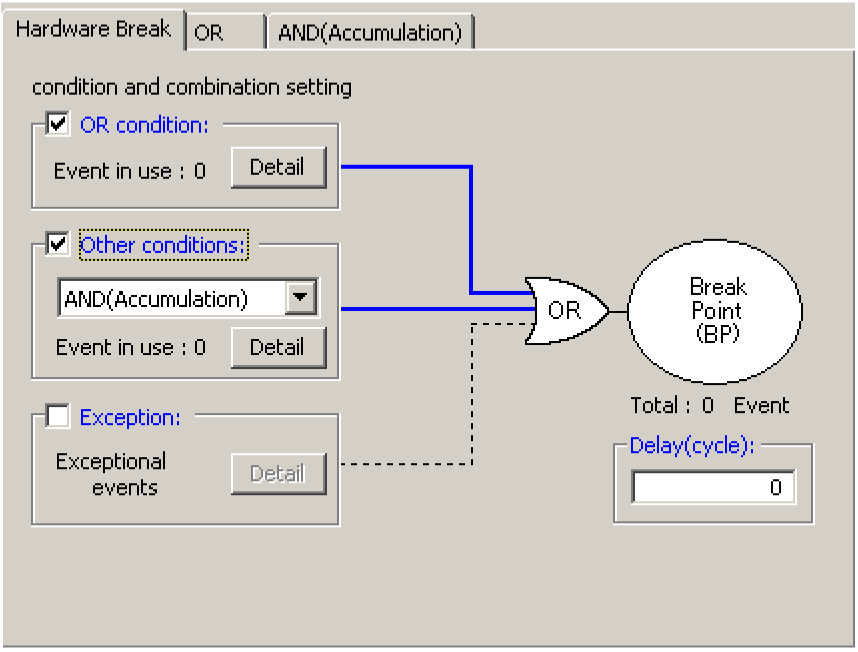


Figure 5.44 Hardware Break dialog box

Table 5.12 Other conditions

Type	Description
AND (cumulative)	A breakpoint is encountered when all of the specified events have occurred regardless of their timing and order.
AND (simultaneous)	A breakpoint is encountered when all of the specified events occur at the same time.
Subroutine	A breakpoint is encountered when a specified event occurs within a specified address range (subroutine or function).
Sequential	6 steps (forward direction) + reset point A breakpoint is encountered when the specified events occur in a specified order.
State transitions	3 steps, 9 paths + reset point A breakpoint is encountered when the specified events occur in a specified order.

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

CAUTION

When a time-out condition is set in State transitions (Hardware break point) dialog box, the time to make transition from a set state to another then back to the original set state must be 10 µs or more. Transition time of less than 10 µs will result in an incorrect timeout detection.

(4) Detection of exceptional events

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

- Violation of access protection
- Reading from a non-initialized memory area
- Stack access violation
- Performance-measurement overflow
- Realtime profile overflow
- Trace memory overflow
- Task stack access violation
- OS dispatch

(5) Specifying a delay value

If this checkbox is selected, program execution breaks the specified number of bus cycles after the breakpoint is encountered. The delay value is specifiable in the range from 0 to 65,535 (default = 0).

5.8.3 Saving/Loading Hardware Break Settings

(1) Saving hardware break settings

Click on the Save button of the Hardware Break dialog box. The Save dialog box will be displayed.
Specify the name of the file where you want the break settings to be saved. The file-name extension is ".hev". If this is omitted, the extension ".hev" is automatically appended.

(2) Loading hardware break settings

Click on the Load button of the Hardware Break dialog box. The Load dialog box will be displayed. Specify the name of the file you want to load.
When you load a file, the previous hardware break settings are discarded and the new settings appear in the dialog box.
Click on the Apply button of the Hardware Break dialog box to activate the new hardware break settings you have loaded.

5.9 Viewing Trace Information

5.9.1 Viewing Trace Information

Tracing means the acquisition of bus information per cycle and storage of this information in trace memory during user program execution. You can use tracing to track the flow of application execution or to search for and examine the points where problems arise.

The E100 emulator allows acquisition of up to 4-M bus cycles.

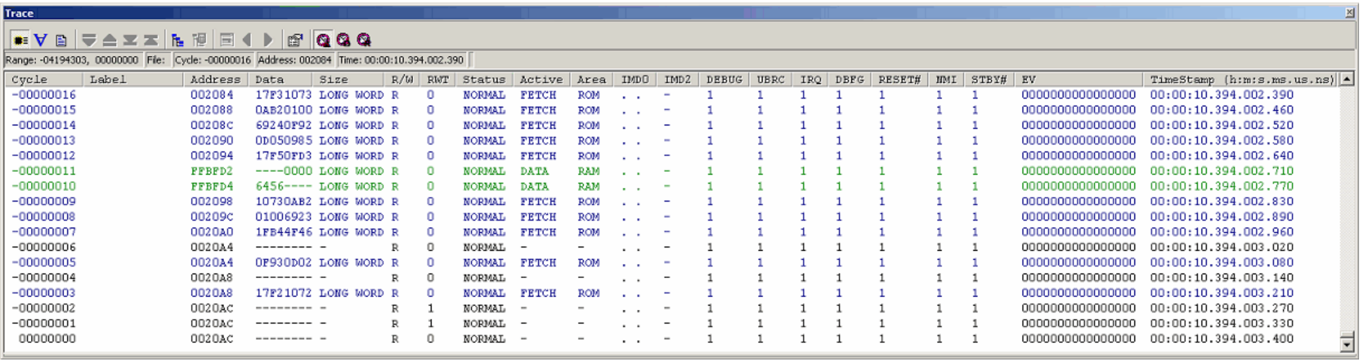
When program execution stops (due to an exception break, forced stop or breakpoint), the contents of trace memory at the time the program has stopped are displayed as the result of tracing, even if no trace points have been encountered yet.

5.9.2 Acquiring Trace Information

In cases where no trace acquisition conditions are set, the default behavior of the E100 emulator is to acquire information on all bus cycles unconditionally (trace mode = Fill until stop).

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

The acquired trace information is displayed in the Trace window.



Cycle	Label	Address	Data	Size	R/W	RWT	Status	Active	Area	IMD0	IMD2	DEBUG	UBRC	IRQ	DBFG	RESET#	NMI	STBY#	EV	TimeStamp (h:m:s.ms.us.ns)
-00000016		002084	17F31073	LONG WORD	R	0	NORMAL	FETCH	ROM	. .	-	1	1	1	1	1	1	1	0000000000000000	00:00:10.394.002.390
-00000015		002088	0AB20100	LONG WORD	R	0	NORMAL	FETCH	ROM	. .	-	1	1	1	1	1	1	1	0000000000000000	00:00:10.394.002.460
-00000014		00208C	69240F92	LONG WORD	R	0	NORMAL	FETCH	ROM	. .	-	1	1	1	1	1	1	1	0000000000000000	00:00:10.394.002.520
-00000013		002090	0D050985	LONG WORD	R	0	NORMAL	FETCH	ROM	. .	-	1	1	1	1	1	1	1	0000000000000000	00:00:10.394.002.580
-00000012		002094	17F50FD3	LONG WORD	R	0	NORMAL	FETCH	ROM	. .	-	1	1	1	1	1	1	1	0000000000000000	00:00:10.394.002.640
-00000011		FFBFD2	----0000	LONG WORD	R	0	NORMAL	DATA	RAM	. .	-	1	1	1	1	1	1	1	0000000000000000	00:00:10.394.002.710
-00000010		FFBFD4	6456----	LONG WORD	R	0	NORMAL	DATA	RAM	. .	-	1	1	1	1	1	1	1	0000000000000000	00:00:10.394.002.770
-00000009		002098	10730AB2	LONG WORD	R	0	NORMAL	FETCH	ROM	. .	-	1	1	1	1	1	1	1	0000000000000000	00:00:10.394.002.830
-00000008		00209C	01006923	LONG WORD	R	0	NORMAL	FETCH	ROM	. .	-	1	1	1	1	1	1	1	0000000000000000	00:00:10.394.002.890
-00000007		0020A0	1FB44F46	LONG WORD	R	0	NORMAL	FETCH	ROM	. .	-	1	1	1	1	1	1	1	0000000000000000	00:00:10.394.002.960
-00000006		0020A4	--------	-	R	0	NORMAL	-	-	. .	-	1	1	1	1	1	1	1	0000000000000000	00:00:10.394.003.020
-00000005		0020A4	0F930D02	LONG WORD	R	0	NORMAL	FETCH	ROM	. .	-	1	1	1	1	1	1	1	0000000000000000	00:00:10.394.003.080
-00000004		0020A8	--------	-	R	0	NORMAL	-	-	. .	-	1	1	1	1	1	1	1	0000000000000000	00:00:10.394.003.140
-00000003		0020A8	17F21072	LONG WORD	R	0	NORMAL	FETCH	ROM	. .	-	1	1	1	1	1	1	1	0000000000000000	00:00:10.394.003.210
-00000002		0020AC	--------	-	R	1	NORMAL	-	-	. .	-	1	1	1	1	1	1	1	0000000000000000	00:00:10.394.003.270
-00000001		0020AC	--------	-	R	1	NORMAL	-	-	. .	-	1	1	1	1	1	1	1	0000000000000000	00:00:10.394.003.330
00000000		0020AC	--------	-	R	0	NORMAL	-	-	. .	-	1	1	1	1	1	1	1	0000000000000000	00:00:10.394.003.400

Figure 5.45 Trace window

The following items are shown in the Trace window (in bus display mode).

Table 5.13 Items shown in the Trace window

Column	Description
Cycle	Number of the cycle within trace memory. By default, the number of the last cycle to have been acquired is 0, and earlier cycles are assigned progressively lower numbers in sequence, i.e. –1, –2, etc. If a delay count is set, the cycle on which the trace stop condition is met is numbered 0 and the cycles that were executed until the program actually stopped (cycles during a delay period) are assigned progressively larger numbers +1, +2, etc. in sequence up to the last cycle to be acquired.
Label	Label corresponding to the address (displayed only when a label has been set)
Address	Address on the address bus
Data	Data on the data bus (in hexadecimal)
Size	Unit of access (byte, word, or longword)
R/W	Data bus state, indicated as "R" for reading, "W" for writing, or "–" for no access
RWT	Whether the bus cycle is valid or not. The value "0" indicates a valid bus cycle. The Address and Data information is valid when RWT is "0".
Status	Current mode of the target MCU. NORMAL Normal operation S-ACT Subactive mode SLEEP Sleep mode S_SLEEP Subsleep mode AMCS All-module clock-stop mode S-STBY Software standby mode H-STBY Hardware standby mode D-STBY Deep standby mode - Other * Impossible combination with ACTIVE
Active	Action taken by the target MCU. DMAC Access by DMAC operation DTC Access by DTC operation HUDI Access by HUDI operation DATA Data access by CPU operation FETCH Instruction fetch by CPU operation STACK Stack access by CPU operation - Other * Impossible combination with STATUS

Column	Description
Area	Area being accessed. EXT32 32-bit external access space EXT16 16-bit external access space EXT8 8-bit external access space EXTMEM32 32-bit external emulation memory EXTMEM16 16-bit external emulation memory EXTMEM8 8-bit external emulation memory ROM Internal ROM I/O32 Internal I/O space (32-bit I/O) I/O16 Internal I/O space (16-bit I/O) I/O8 Internal I/O space (8-bit I/O) RAM Internal RAM - Other Notes: Access to an area for writing to the internal ROM (H'D00000 to H'DFFFFF) is indicated as "I/O16". Access to the internal RAM area for use by the FCU (H'EF8000 to H'EF97FF) is indicated as "I/O16". Access to the firmware area for use by the FCU (H'C00000 to H'C01FFF) is indicated as "-". Access to an internal ROM area to which emulation memory has been assigned is indicated as "-".
IMD0	States of interrupt mask bits of the condition code register in interrupt control mode 0 Value / Description: Bit CCR I, Bit CCR Ui . . 0 0 . Ui 0 1 I . 1 0 I Ui 1 1 - The entry under IMD0 is "-" if IMD2 values are being displayed.
IMD2	Interrupt mask levels of the extended control register in interrupt control mode 2 Value / Description: Bit EXT I2, Bit EXT I1, Bit EXT I0 0 0 0 0 1 0 0 1 2 0 1 0 3 0 1 1 4 1 0 0 5 1 0 1 6 1 1 0 7 1 1 1 - The entry under IMD0 is "-" if IMD2 values are being displayed.
DEBUG	0 indicates that the emulator has taken over the MCU bus while the user program was running. The emulator takes over the MCU bus when access to memory is attempted by a debugger operation. Note: Execution of the user program is temporarily stopped during such access to memory.
UBRC	Whether the cycle was a user-bus release cycle (i.e. the external bus of the target MCU had been released and a low-level signal was being output). 0: The external bus had been released. 1: The external bus had not been released.
IRQ	Monitoring of user IRQ. 0: An interrupt has been detected. 1: No interrupt has been detected.
DBFG	Whether the write-data buffer function was used and a low-level signal was being output. 0: The write-data buffer function was used. 1: The write-data buffer function was not used.

Column	Description
RESET#	User reset input. 0: A user reset signal has been input. 1: No user reset signal has been input.
NMI	NMI input. 0: An NMI signal has been input. 1: No NMI signal has been input.
STBY#	STANDBY input. 0: An STANDBY signal has been input. 1: No STANDBY signal has been input.
EV	If an event occurred, the number of the event. To show the EV column, you need to select Event number on the Option page of the Trace conditions dialog box opened by choosing Acquisition from the popup menu of the Trace window.
TID	Task ID (when the RTOS is in use) Task IDs are shown in the form "task ID (task entry label)", such as 1 (_Task1). To show the Task ID column, you need to select Task ID on the Option page of the Trace conditions dialog box opened by choosing Acquisition from the popup menu of the Trace window.
EXT	Signal fed in from the external trigger cable; "1" and "0" indicate the signal being at the high and low levels, respectively. To show the EXT column, you need to select External trigger on the Option page of the Trace conditions dialog box opened by choosing Acquisition from the popup menu of the Trace window.
Timestamp	Time elapsed since the target program has started. Each time the user program starts running, time stamping starts from 0. Note: After the counter has overflowed, the times displayed will not be correct. The maximum timestamp value is 3 hours 03 minutes 15 seconds.

Columns of the Trace window can be hidden if you do not require them. To hide a column, right-click in the header column and select the column you want to hide from the popup menu.

5.9.3 Setting Conditions for Trace Information Acquisition

Since the size of the trace buffer is limited, the oldest trace data is overwritten with new data after the buffer has become full. You can set trace conditions to restrict the acquired trace information to that which is useful, thus more effectively using the trace buffer.

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

(1) Selecting the trace mode

Start by selecting the trace mode.

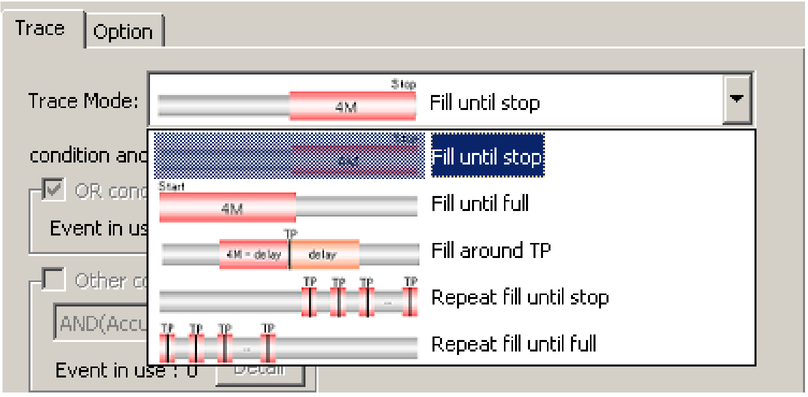


Figure 5.46 Trace conditions dialog box

(2) Setting trace points

If you have selected Fill around TP, Repeat fill until stop or Repeat fill until full, you need to set a trace point.
For trace points, you can specify conditions using events and/or the detection of specific exceptional events.
For Fill around TP, you can also specify a delay value.

(3) Selecting Capture or Do not capture

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

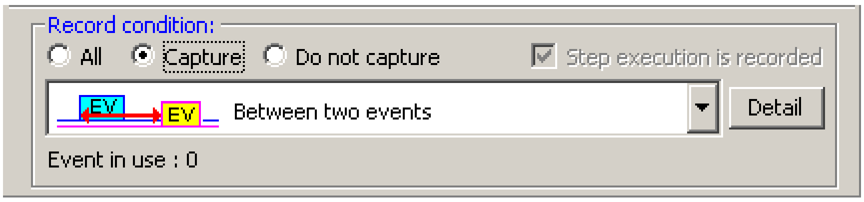


Figure 5.47 Record condition group box

You can specify events so as to extract only the required portions or to eliminate non-required portions of the trace information.

(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.

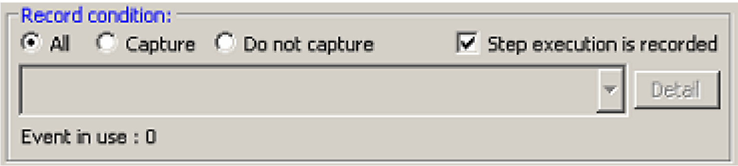


Figure 5.48 Recording step execution

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

(5) Selecting the type of trace information to be acquired

Use the Option page of the Trace conditions dialog box to select the type of trace information to be stored in the trace memory. By default, 'Event number' is selected as the type of trace information.

5.9.4 Selecting the Trace Mode

(1) Selecting the trace mode

The following five trace modes are available.

Table 5.14 Trace modes

Trace mode	Description
Fill until stop	Trace acquisition continues until the program stops running.
Fill until full	Trace acquisition stops when the trace memory becomes full.
Fill around TP	Trace acquisition stops a specified number of cycles after a trace point is encountered. A delay value can be specified in the range up to the maximum value of trace capacity.
Repeat fill until stop	For each trace point encountered in program execution, information for a total of 512 cycles* before and after the point is acquired, and acquisition continues in the same way until the program stops running.
Repeat fill until full	For each trace point encountered in program execution, information for a total of 512 cycles* before and after the point is acquired, and acquisition continues in the same way until the trace memory is full.

CAUTION

*Recording is for 512-cycles units, consisting of the lines for the cycle at the trace point, for the 255 cycles before that point, and for the 256 cycles after that point.

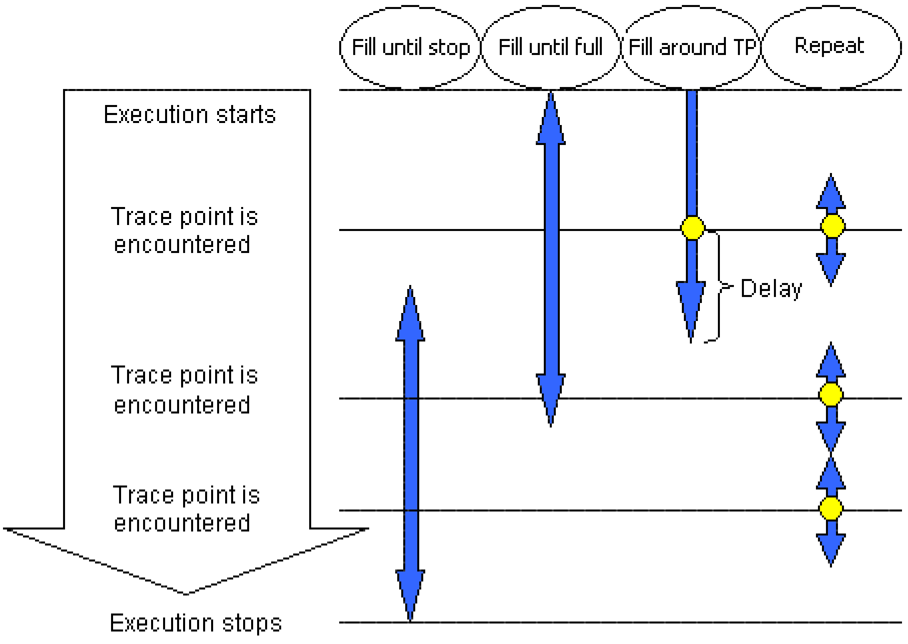


Figure 5.49 Differences between the trace modes

Specifiable conditions vary with the trace mode, as summarized in the tables below.

1. Fill until stop

The trace memory can hold up to 4-M bus cycles. When the buffer becomes full, the oldest data among the acquired trace information are overwritten with new data. The emulator continues acquiring trace information in this way until the program is stopped.

Table 5.15 Specifiable conditions: Fill until stop

Trace point	Delay	Specifying capture/do not capture	Recording of step execution
-	-	Possible	Possible

2. Fill until full

Once the trace memory of the emulator overflows during trace acquisition, the emulator stops acquiring trace information.

Table 5.16 Specifiable conditions: Fill until full

Trace point	Delay	Specifying capture/do not capture	Recording of step execution
-	-	Possible	-

3. Fill around TP

Trace acquisition is halted a specified number of cycles 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. The delay value can be chosen as 0M, 1M, 2M, 3M or 4M cycles.

Table 5.17 Specifiable conditions: Fill around TP

Trace point	Delay	Specifying capture/do not capture	Recording of step execution
Possible	Possible	Possible	-

4. Repeat fill until stop

For each time trace point encountered, information for a total of 512 cycles before and after that point is acquired, and acquisition continues in the same way until the program stops running. Acquisition continues until it is halted by a break or forced stop. The positions where trace points have been encountered can be checked in the Trace window.

Table 5.18 Specifiable conditions: Repeat fill until stop

Trace point	Delay	Specifying capture/do not capture	Recording of step execution
Possible	-	-	-

5. Repeat fill until full

For each time trace point encountered, information for a total of 512 cycles before and after that point is acquired. Acquisition continues in the same way until the trace memory overflows, at which time acquisition is halted. The positions where trace points have been encountered can be checked in the Trace window.

Table 5.19 Specifiable conditions: Repeat fill until full

Trace point	Delay	Specifying capture/do not capture	Recording of step execution
Possible	-	-	-

CAUTION

If trace points are encountered in consecutive cycles in the repeat fill until stop or repeat fill until full mode, the yellow highlight that indicates a trace point only appears for the trace point in the first of the cycles.

5.9.5 Setting Trace Points

(1) Setting trace points

For trace points, you can set an OR condition, other conditions (AND (cumulative), AND (simultaneous), subroutine, sequential or state transitions) and detection of exceptional events.

You can specify all or only one of the OR condition, other conditions and detection of exceptional events at a time.

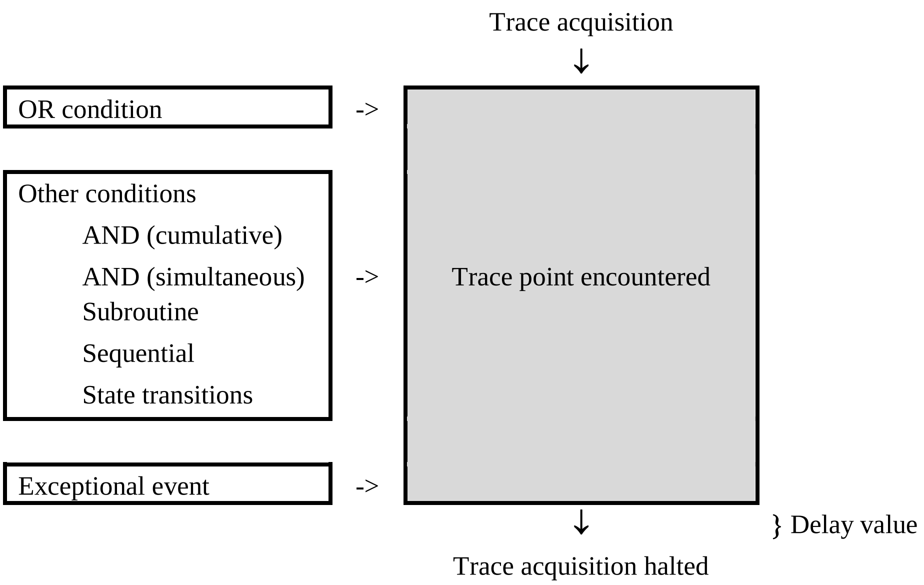


Figure 5.50 A trace point in outline

(2) OR condition

You can choose to enable or disable the OR condition. By default, the OR condition is enabled.
When the OR condition is re-enabled, the previous event settings on the OR page (with their checkboxes being selected) are restored.
However, if re-enabling the OR condition would bring the total number of events to more than 16, the events are restored with their checkboxes not selected (disabled) on the OR page.

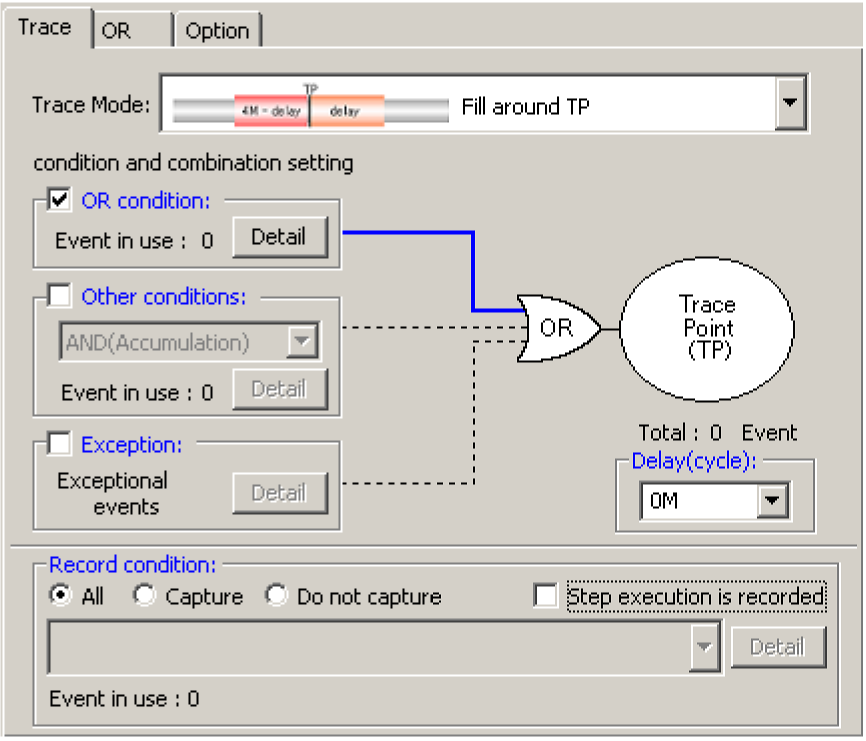


Figure 5.51 Trace conditions dialog box

Table 5.20 OR condition

Type	Description
OR condition	A trace point is encountered when any of the specified events occurs.

(3) Other conditions

You can select one from among five available choices: AND (cumulative), AND (simultaneous), Subroutine, Sequential and State transitions. To set any condition, select the checkbox to the left of "Other conditions."
Other conditions are disabled by default (the checkbox to the left of "Other conditions" is not selected). Cumulative AND is listed as "AND(Accumulation)" in the dialog box.

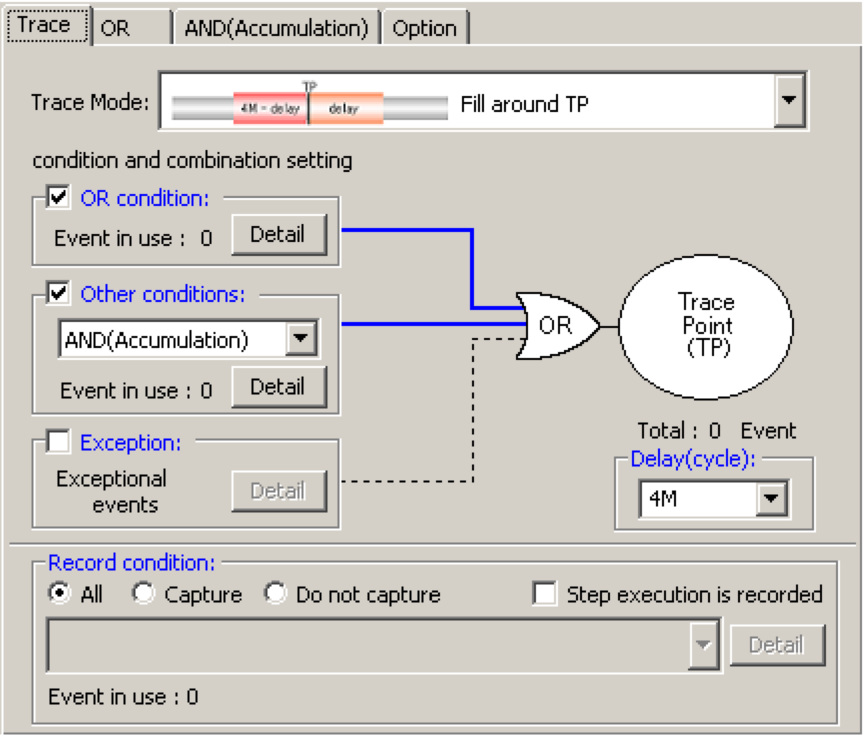


Figure 5.52 Trace conditions dialog box

Table 5.21 Other conditions

Type	Description
AND (cumulative)	A trace point is encountered when all of the specified events have occurred, regardless of the timing.
AND (simultaneous)	A trace point is encountered when all of the specified events occur at the same time.
Subroutine	A trace point is encountered when a specified event occurs within a specified address range (subroutine or function).
Sequential	6 steps (forward direction) + reset point A trace point is encountered when the specified events occur in a specified order.
State transitions	3 steps, 9 paths + reset point A trace point is encountered when the specified events occur in a specified order.

CAUTION
When a time-out condition is set in State transitions (Trace) dialog box, the time to make transition from a set state to another then back to the original set state must be 10 μs or more. Transition time of less than 10 μs will result in an incorrect timeout detection.

(4) Detection of exceptional events

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

- Violation of access protection
- Reading from a non-initialized memory area
- Stack access violation
- Performance-measurement overflow
- Realtime profile overflow
- Task stack access violation
- OS dispatch

(5) Specifying a delay value

If this checkbox is selected, tracing stops the specified number of bus cycles after the trace point is encountered.
The delay value is selectable as 0M, 1M, 2M, 3M or 4M bus cycles (default: 0M).
Select the desired value from the Delay drop-down list box.

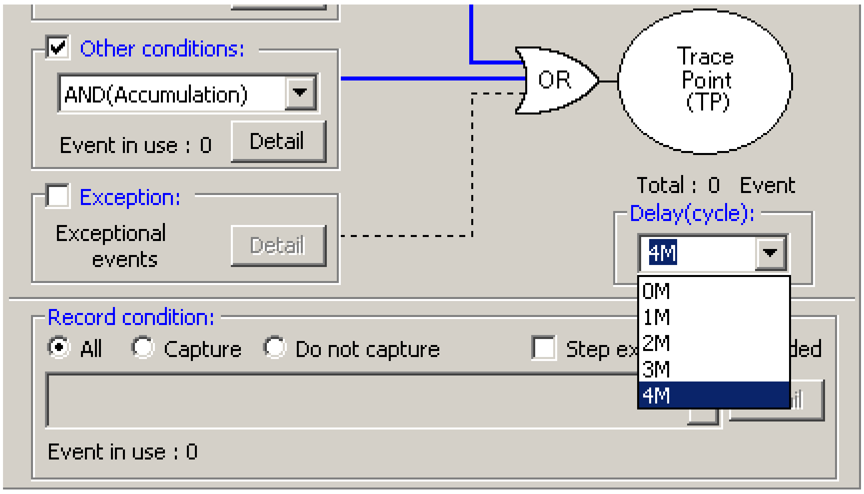


Figure 5.53 Selecting a delay value

5.9.6 Setting Extraction or Elimination Conditions

If the selected trace mode is Fill until stop, Fill until full or Fill around TP, you can specify a condition for capturing or not capturing information.

You can specify events so as to extract only the required portions or to eliminate non-required portions of the trace information.

(1) Extraction and elimination conditions

The following types of condition are available.

Table 5.22 Extraction and elimination conditions

Type			Description
Extraction		Between two events	Trace information is extracted from the cycle in which the event set as [Start event] occurs to the cycle preceding the event set as [End event] (information is not acquired for the cycle where [End event] occurs).
		Duration of an event	Trace information is extracted over the cycles corresponding to occurrence of the specified event.
		Duration of an event occurring in a subroutine	Trace information is extracted over the cycles corresponding to occurrence of the specified event within the specified address range (subroutine or function).
		Instruction accessing specific data	Information is extracted for instructions that access specified data.
Elimination		Between two events	Trace information is eliminated from the cycle in which the event set as [Start event] occurs to the cycle preceding the event set as [End event] (information is not acquired for the cycle where [End event] occurs).
		Duration of an event	Trace information is eliminated over the cycles corresponding to occurrence of the specified event.
		Duration of an event occurring in a subroutine	Trace information is eliminated over the cycles corresponding to occurrence of the specified event within the specified address range (subroutine or function).

Select the desired condition 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.

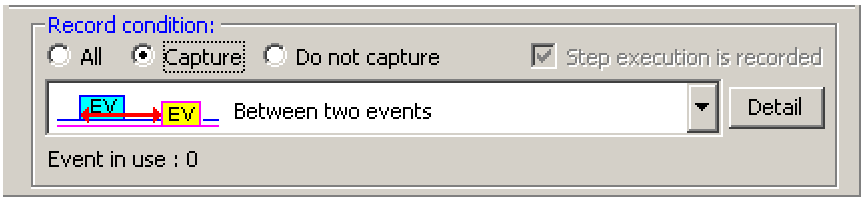


Figure 5.54 Record condition group box

Then click on the Detail button. The Event dialog box will appear.

CAUTION

When you specify conditions for extraction or elimination, you cannot select DIS (disassembly mode) or SRC (source mode) from Display Mode in the popup menu of the Trace window.

When you specify a data-access event as a condition for extraction or elimination, be sure to specify MCU bus as the access type.

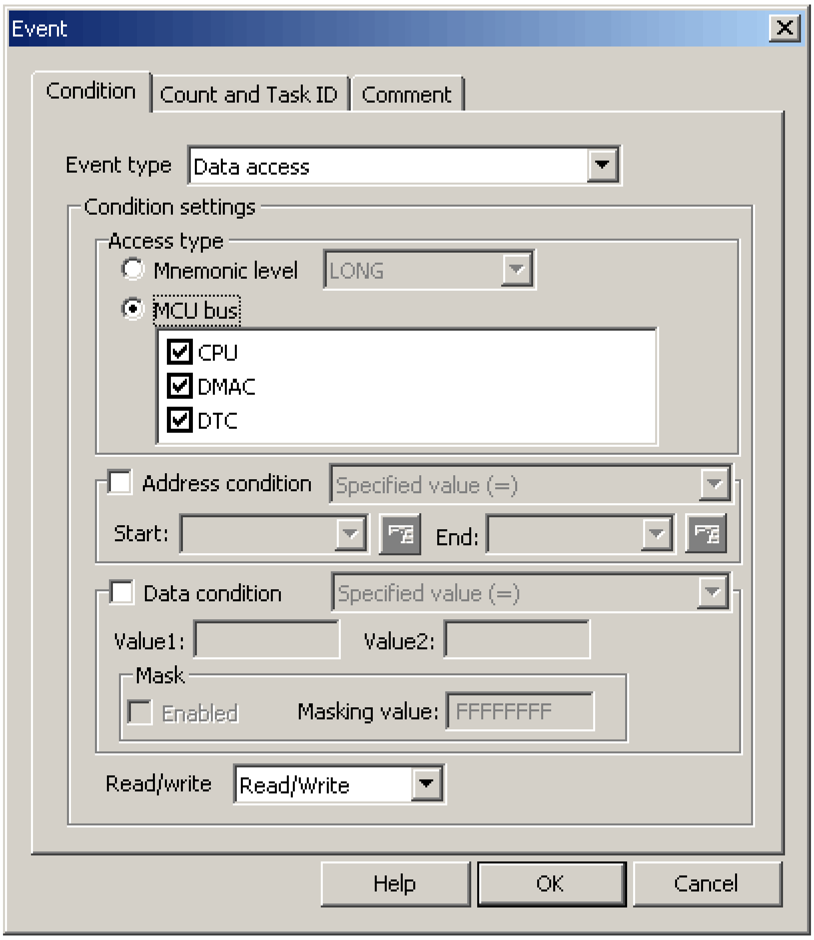


Figure 5.55 Event dialog box

5.9.7 Selecting the Type of Trace Information to be Acquired

Select the type of trace information to be stored in the trace memory. Make this selection on the Option page of the Trace conditions dialog box.

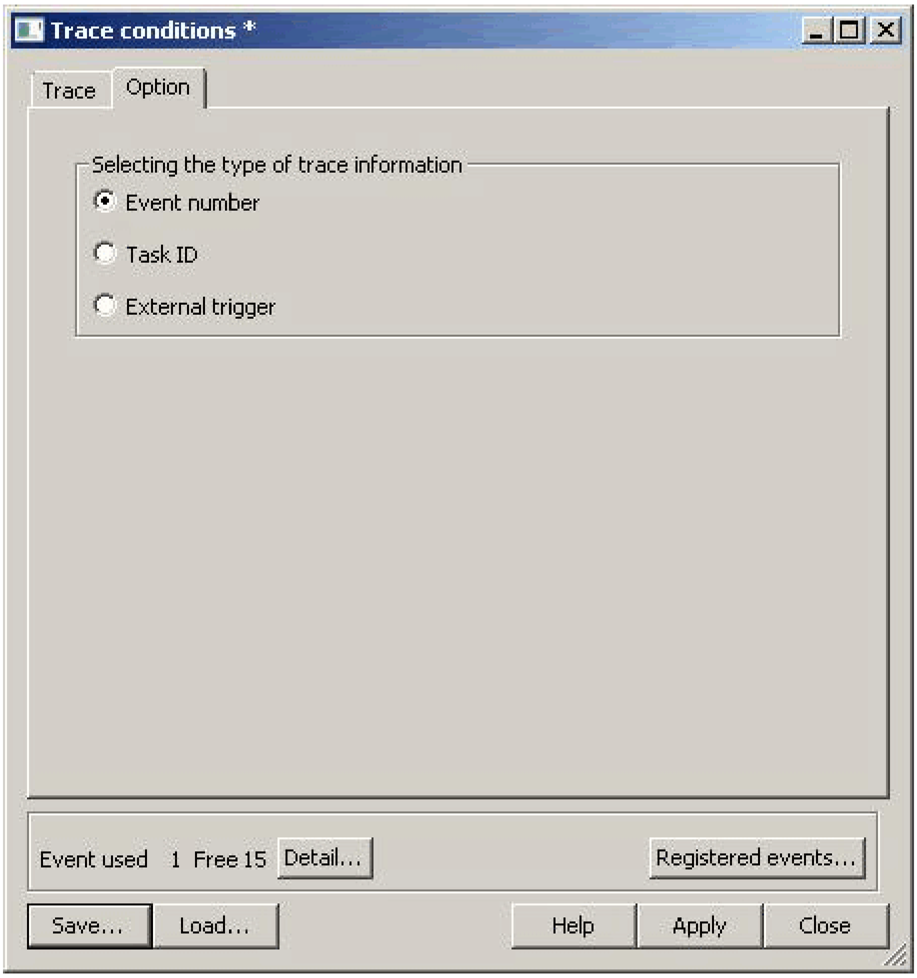


Figure 5.56 Trace conditions dialog box

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

CAUTION

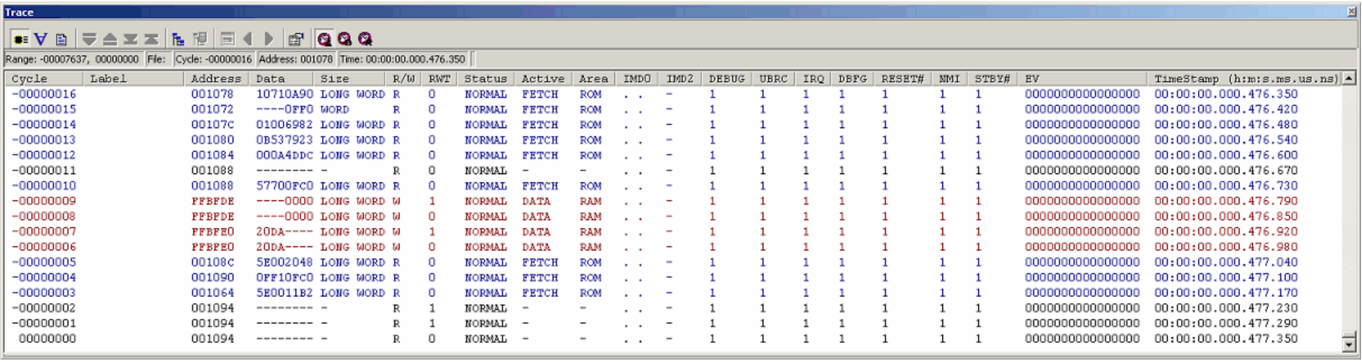
If you want to view the history of tracing information on a realtime OS program, select Task ID.

5.9.8 Viewing Trace Results

To check trace results, open the Trace window. Trace results can be shown in one of the following display modes: bus, disassembly, source, or mixed. The display can be switched by changing the selection of Display Mode in the popup menu of the Trace window.

(1) Bus Display Mode

In the popup menu, select Display Mode -> BUS. Bus information is displayed for all traced cycles (this is the default display mode).



Cycle	Label	Address	Data	Size	R/W	RWT	Status	Active	Area	IMD0	IMD2	DEBUG	UBRC	IRQ	DBFG	RESET#	NMI	STBY#	EV	TimeStamp (h:m:s.ms.us.ns)
-00000016		001078	10710A90	LONG WORD	R	0	NORMAL	FETCH	ROM	. .	-	1	1	1	1	1	1	1	0000000000000000	00:00:00.000.476.350
-00000015		001072	----0FF0	WORD	R	0	NORMAL	FETCH	ROM	. .	-	1	1	1	1	1	1	1	0000000000000000	00:00:00.000.476.420
-00000014		00107C	01006982	LONG WORD	R	0	NORMAL	FETCH	ROM	. .	-	1	1	1	1	1	1	1	0000000000000000	00:00:00.000.476.480
-00000013		001080	0B537923	LONG WORD	R	0	NORMAL	FETCH	ROM	. .	-	1	1	1	1	1	1	1	0000000000000000	00:00:00.000.476.540
-00000012		001084	000A4DDC	LONG WORD	R	0	NORMAL	FETCH	ROM	. .	-	1	1	1	1	1	1	1	0000000000000000	00:00:00.000.476.600
-00000011		001088	--------	-	R	0	NORMAL	-	-	. .	-	1	1	1	1	1	1	1	0000000000000000	00:00:00.000.476.670
-00000010		001088	57700FC0	LONG WORD	R	0	NORMAL	FETCH	ROM	. .	-	1	1	1	1	1	1	1	0000000000000000	00:00:00.000.476.730
-00000009		FFBFDE	----0000	LONG WORD	W	1	NORMAL	DATA	RAM	. .	-	1	1	1	1	1	1	1	0000000000000000	00:00:00.000.476.790
-00000008		FFBFDE	----0000	LONG WORD	W	0	NORMAL	DATA	RAM	. .	-	1	1	1	1	1	1	1	0000000000000000	00:00:00.000.476.850
-00000007		FFBFE0	20DA----	LONG WORD	W	1	NORMAL	DATA	RAM	. .	-	1	1	1	1	1	1	1	0000000000000000	00:00:00.000.476.920
-00000006		FFBFE0	20DA----	LONG WORD	W	0	NORMAL	DATA	RAM	. .	-	1	1	1	1	1	1	1	0000000000000000	00:00:00.000.476.980
-00000005		00108C	5E002048	LONG WORD	R	0	NORMAL	FETCH	ROM	. .	-	1	1	1	1	1	1	1	0000000000000000	00:00:00.000.477.040
-00000004		001090	0FF10FC0	LONG WORD	R	0	NORMAL	FETCH	ROM	. .	-	1	1	1	1	1	1	1	0000000000000000	00:00:00.000.477.100
-00000003		001064	5E0011B2	LONG WORD	R	0	NORMAL	FETCH	ROM	. .	-	1	1	1	1	1	1	1	0000000000000000	00:00:00.000.477.170
-00000002		001094	--------	-	R	1	NORMAL	-	-	. .	-	1	1	1	1	1	1	1	0000000000000000	00:00:00.000.477.230
-00000001		001094	--------	-	R	1	NORMAL	-	-	. .	-	1	1	1	1	1	1	1	0000000000000000	00:00:00.000.477.290
00000000		001094	--------	-	R	0	NORMAL	-	-	. .	-	1	1	1	1	1	1	1	0000000000000000	00:00:00.000.477.350

Figure 5.57 Trace window

(2) Disassembly Display Mode

From the popup menu, choose Display Mode -> DIS. This mode shows a disassembly of the machine-language instructions that have been executed.

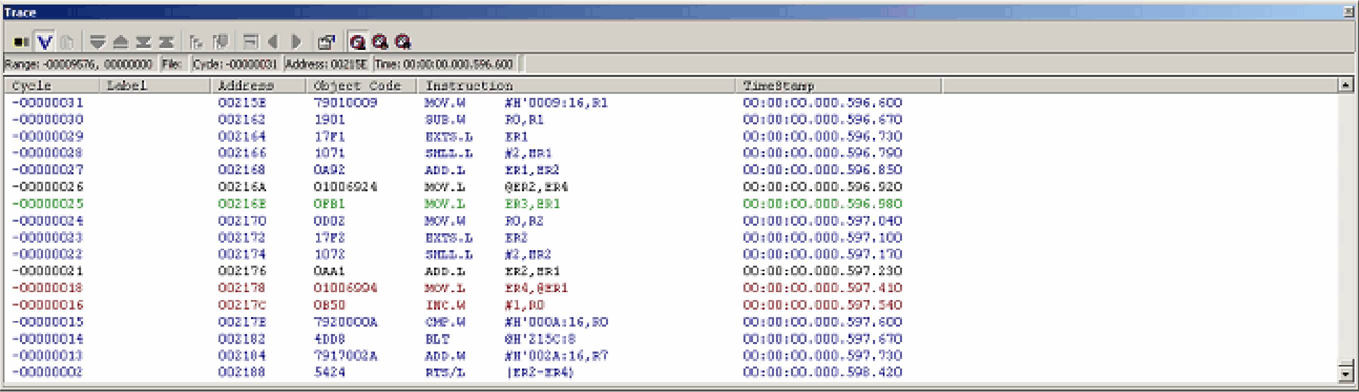


Figure 5.58 Trace window

(3) Source Display Mode

From the popup menu, choose Display Mode -> SRC. This mode shows the flow of execution of the source program.

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

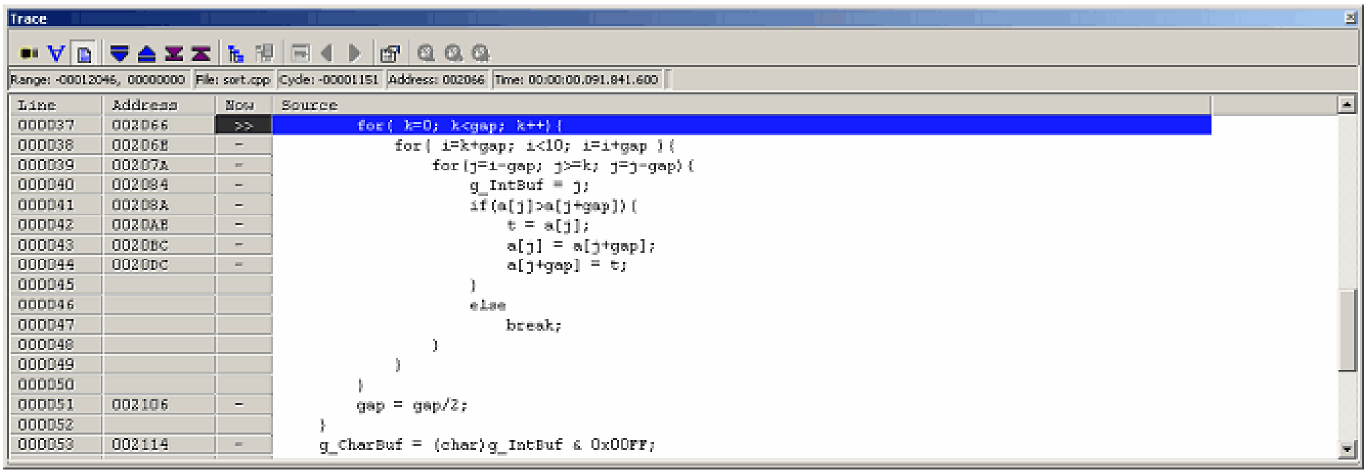


Figure 5.59 Source mode screen

(4) Mixed Display Modes

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

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

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

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

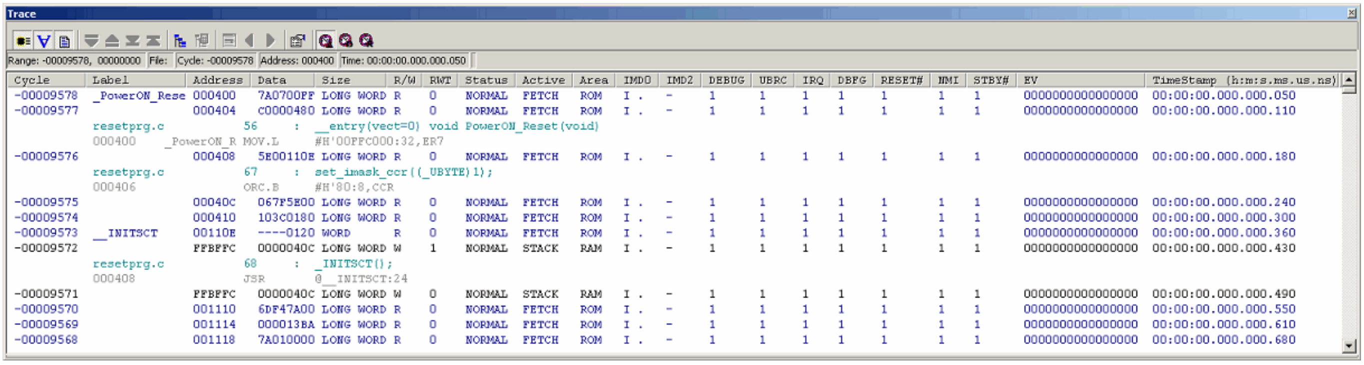


Figure 5.60 Trace window

5.9.9 Filtering Trace Information

Use the filtering facility to extract only the records you need from the acquired trace information. This facility is achieved by software filtering of the trace information that has been acquired by hardware.

Unlike "Capture/Do not Capture", where the conditions must be set before getting the trace information, the filter settings can be changed any number of times. This makes it easy to extract required information, significantly facilitating data analysis.

Filtering does not affect the trace memory, so that its contents remain intact.

Filtering is available when the selected trace mode is Fill until stop, Fill until full or Fill around TP and the selected display mode is bus or disassembly.

(1) Auto-filtering

To use the filtering facility, choose Auto Filter from the popup menu of the Trace window. When Auto Filter is turned on, each of the columns in the Trace window is marked with an auto-filter arrow [ ].

By simply clicking on the arrows [ ] and selecting desired conditions from the drop-down lists, you can filter the records to get those that meet the conditions. Selecting Option in the drop-down list brings up the Option dialog box. In this dialog box, you can set detailed conditions.

Items such as Address and Data do not have a manageably small fixed set of items, so the only entry in the drop-down list for these columns is Option… Selecting All returns the window to the non-filtered state.

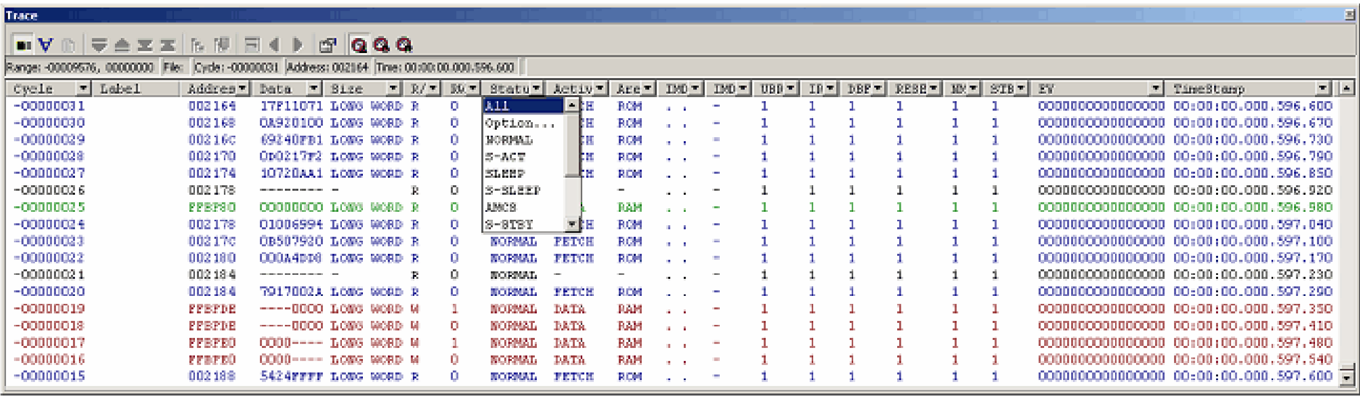


Figure 5.61 Trace window

If you switch the display mode to disassembly or source after filtering records in bus mode, Auto Filter is deselected. Similarly, if you switch the display mode to bus or source after filtering records in disassembly mode, Auto Filter is deselected.

If you have specified multiple items in an Option dialog box, these items constitute an OR condition for use in filtering.

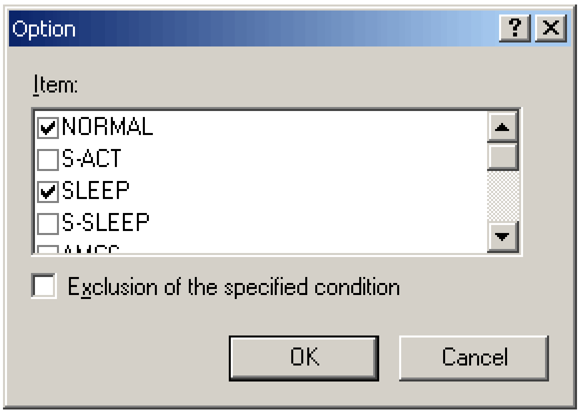
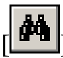


Figure 5.62 Option dialog box

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. Open this by choosing Find -> Find from the popup menu of the Trace window or clicking on the Find toolbar button [].

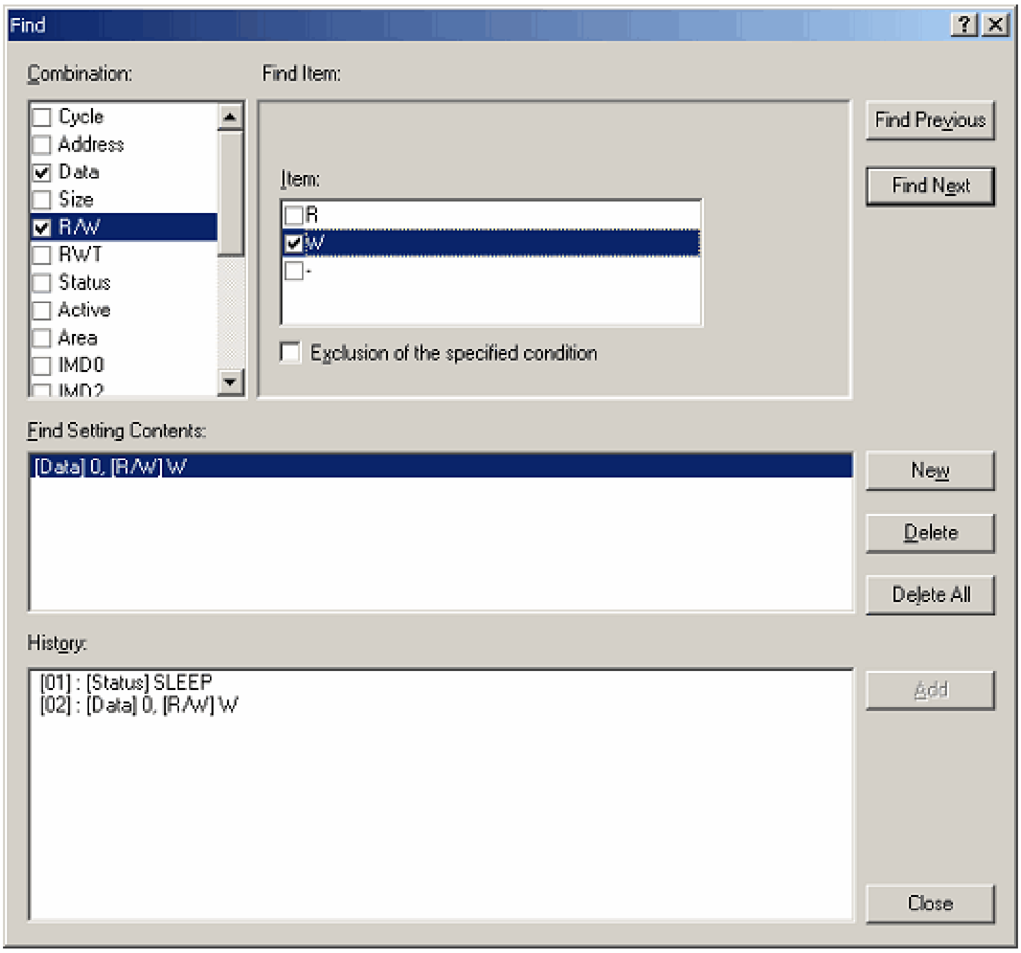


Figure 5.63 Find dialog box

In the Combination column, select the checkboxes for the items of trace information for which you want to set up criteria.

The criteria that correspond to the currently selected items appears in the Find Item column. Select the required criteria.

If you have checked more than one item in the Combination column, set criteria for each of them. The items you have set are used as an AND condition.

The criteria you have set are shown in Find Setting Contents.

After setting the criteria, click the Find Previous or Find Next button to start a search. Searching then proceeds forwards or backwards through the trace records from the line you have clicked in the Trace window (the line highlighted in blue).

When a matching trace record is found, the corresponding line is highlighted in the Trace window. 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 popup menu. This initiates a search for the next instance of the trace record.

(1) Search history

The search conditions that have been used are recorded in the History column and are retained throughout a session of the High-performance Embedded Workshop.
If you want to perform a search again, choose the corresponding line from the history and click on the Add button to initiate a new search for trace information with the same condition.
Up to the last 10 searches are retained in the search history.

(2) OR search

You can perform a search with two or more search conditions combined in an OR condition.
To set an OR condition, begin by setting the first condition (shown on the first line in the Find Setting Contents column) and then click on the New button.
Then enter the second condition. At this time, the second condition is added as a second line in the Find Setting Contents column.
In this case, the search is for lines satisfying the logical OR of the conditions on the first and second lines in the Find Setting Contents column.
Up to 16 conditions (16 lines) can be set.

CAUTION
Conditions set on the same line of the Find Setting Contents column are treated as an AND condition.

5.9.11 Saving Trace Information in Files

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

(1) Saving in the binary format

To save trace information in the binary format, choose “Trace Data File: Memory Image (*.rtt)” in the Save As Type list box of the dialog box that is displayed when you choose File -> Save from the popup menu.
When information is saved in the binary format, information for all cycles is saved. This type of file can be loaded back into the Trace window.

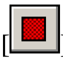
(2) Saving in the text format

To save trace information in the text format, choose “Text Files: Save Only (*.txt)” in the Save As Type list box of the dialog box that is displayed when you choose File -> Save from the popup menu.
When information is saved in the text format, saving of information for a range of cycles can be specified. This type of file can only be saved and cannot be loaded back into the Trace window.

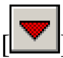
5.9.12 Loading Trace Information from Files

To load trace information from a file, choose File -> Load from the popup menu or click on the Load toolbar button []. Specify a trace information file that was saved in the binary format. The current results of tracing are overwritten.
Before loading a file saved in the binary format, switch to the trace mode in which the saved trace information was acquired. This switching should be performed in the Trace conditions dialog box that is displayed when you choose Acquisition from the popup menu of the Trace window.
If the current trace mode differs from that in which the saved information was acquired, an error occurs. Trace information files saved in the text format cannot be loaded back into the Trace window.

5.9.13 Temporarily Stopping Trace Acquisition

To temporarily stop the acquisition of trace information during user program execution, choose Trace -> Stop from the popup menu of the Trace window or click on the Stop toolbar button [].
Trace acquisition will be stopped, with the trace display updated. Use this function when you only want to stop acquisition and check the trace information but not to stop program execution.

5.9.14 Restarting Trace Acquisition

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

5.9.15 Switching the Timestamp Display

The display of timestamps in the Trace window can be switched to absolute time, differential time or relative time. In the initial state, the timestamps are displayed in absolute time.

(1) Absolute time

Choose Time -> Absolute Time from the popup menu or click on the Absolute Time toolbar button []. The displayed timestamps will be displayed in absolute time since the program started running.

(2) Differential time

Choose Time -> Differences from the popup menu or click on the Differences toolbar button []. Each displayed timestamp is the difference in time from the preceding cycle.

(3) Relative time

Choose Time -> Relative Time from the popup menu or click on the Relative Time toolbar button []. The displayed timestamps are times relative to the time of a specified cycle.

5.9.16 Viewing the History of Function Execution

To view the history of function execution extracted from the acquired trace information, choose Function Execution History -> Function Execution History from the popup menu or click on the Function Execution History toolbar button [ ].

An upper pane will be opened in the Trace window (the pane is blank by default).

When you choose Analyze Execution History from the popup menu or click on the Analyze Execution History toolbar button [ ], the emulator starts analyzing the history of execution history from the end of the results of tracing. The results of analysis are displayed in a tree structure.

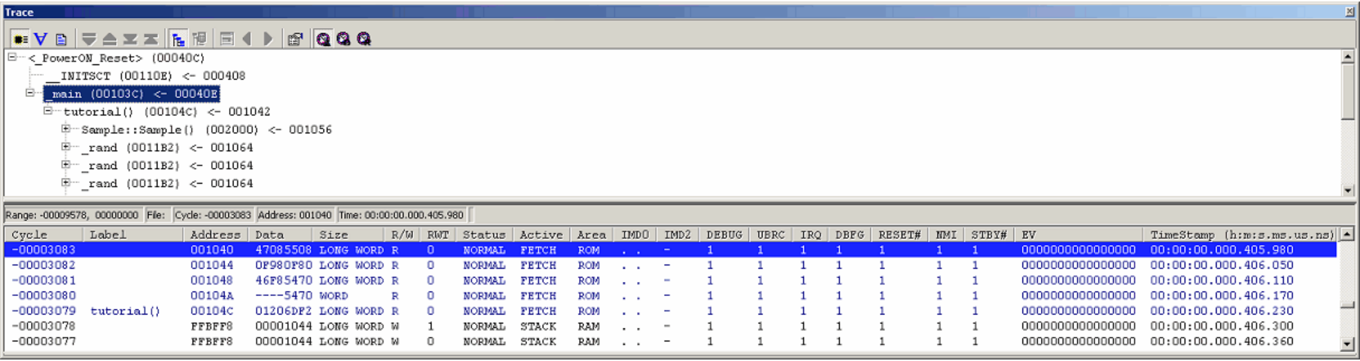


Figure 5.64 Trace window

The lower pane of the window shows results of tracing from the cycle in which the function selected in the upper pane was called.

Results in the lower pane can be displayed in disassembly, source, or a mixed mode.

CAUTION

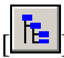
If extraction or elimination conditions are specified, the history of function execution cannot be displayed.

If the 'repeat fill until stop' or 'repeat fill until full' mode is selected, the history of function execution cannot be displayed.

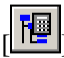
5.9.17 Viewing the History of Task Execution

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

Furthermore, to view 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 popup menu of the Trace window.

To show the history of function execution extracted from the acquired trace information, choose Show Function Execution History from the popup menu or click on the Show Function Execution History toolbar button [].

The upper pane of the window will be opened (the pane is blank by default).

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

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

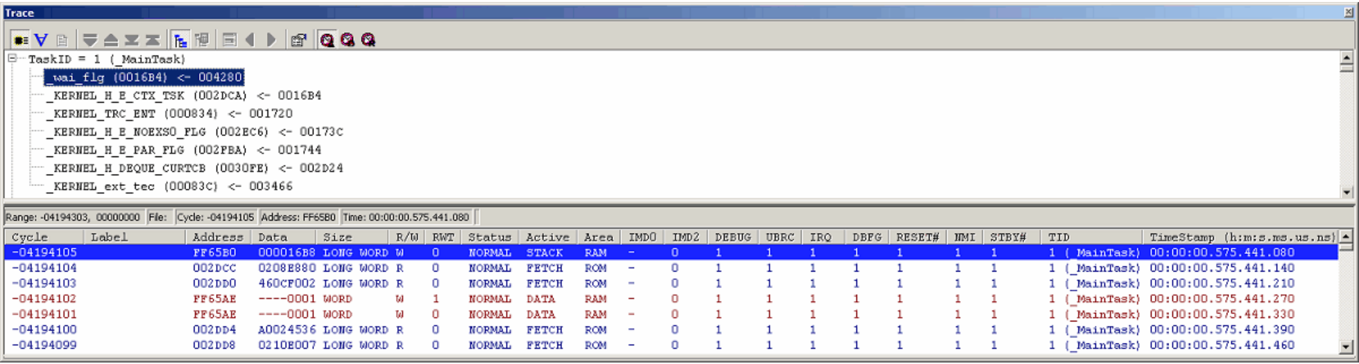


Figure 5.65 Trace window

The lower pane of the window shows results of tracing from the cycle in which the task selected in the upper pane was called. The lower pane of the window can show trace results in disassembly, source, or a mixed mode.

CAUTION

If extraction or elimination conditions are specified, the history of task execution cannot be displayed.

If the 'repeat fill until stop' or 'repeat fill until full' mode is selected, the history of task execution cannot be displayed.

5.10 Measuring Performance

5.10.1 Measuring Performance

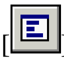
The performance measurement facility of the emulator is capable of measuring the maximum, minimum, average and total execution times and the number of passes for each of up to eight specified sections of the user program, and shows ratios of time relative to the overall execution time (Go–Break) as percentages and graphically.

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

Performance measurement conditions cannot be manipulated during program execution.

5.10.2 Viewing the Results of Performance Measurement

Results of measurement are displayed in the Performance Analysis window.

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

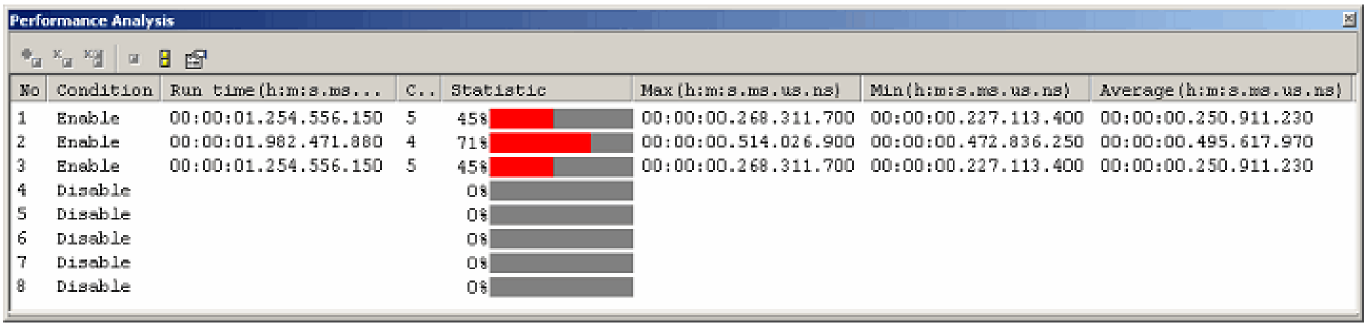


Figure 5.66 Performance Analysis window

The Performance Analysis window shows the ratios of execution time per condition you have set for the most recent execution of the program as percentages and graphically.

Any 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 popup menu.

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

The contents displayed in this window are listed below.

Table 5.23 Columns and contents

Column	Description
No	Numbers from 1–8 that are assigned to the measurement sections set up in the Performance Analysis Conditions dialog box. Click Settings on the popup menu to open the Performance Analysis Conditions dialog box.
Condition	The entry is "Enable" when a measurement condition is set in the Performance Analysis Conditions dialog box. Otherwise, the entry is "Disable".
Run time (h:m:s.ms.us.ns)	Cumulative execution time. This is the cumulative total of measured execution times.
Count	Shows the number of times measurement for the section has proceeded.
Statistic	Shows the ratio of the cumulative execution time relative to the 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 Analysis window, select the line of a section number to use for the condition and choose Set from the popup menu. The Performance Analysis Conditions dialog box will be displayed.

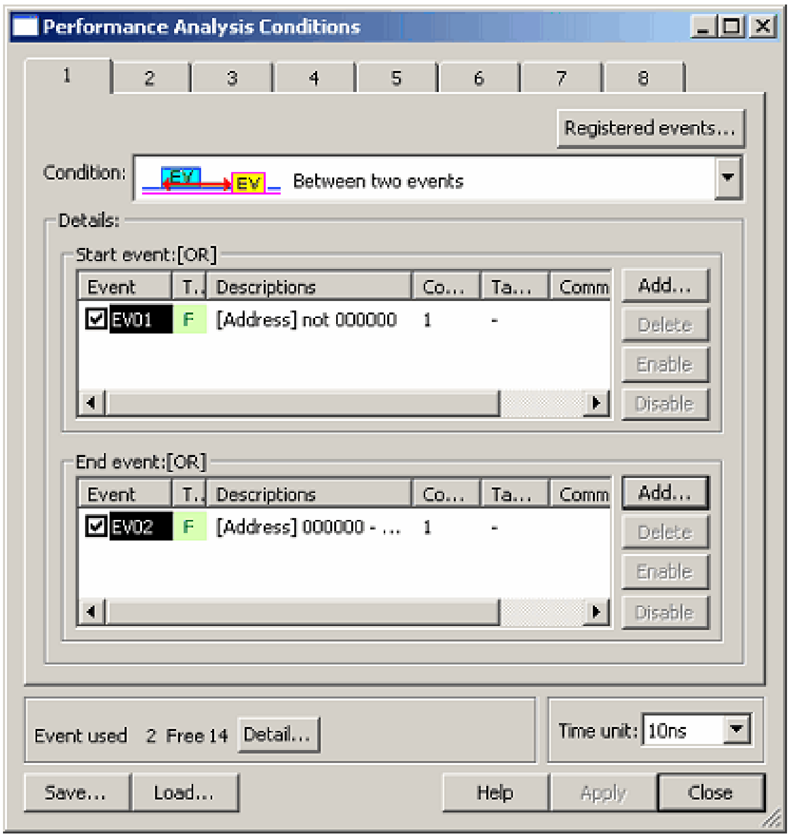


Figure 5.67 Performance Analysis Conditions dialog box

(1) Setting measurement conditions

The measurement mode can be selected from among the four choices listed in Table 5.24. Select one measurement mode for one section. Use events to specify the beginning and end of a section. The value of Count is fixed to 1. The event count is always 1, even if you have attempted to specify some other value.

Table 5.24 Measurement modes

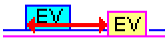
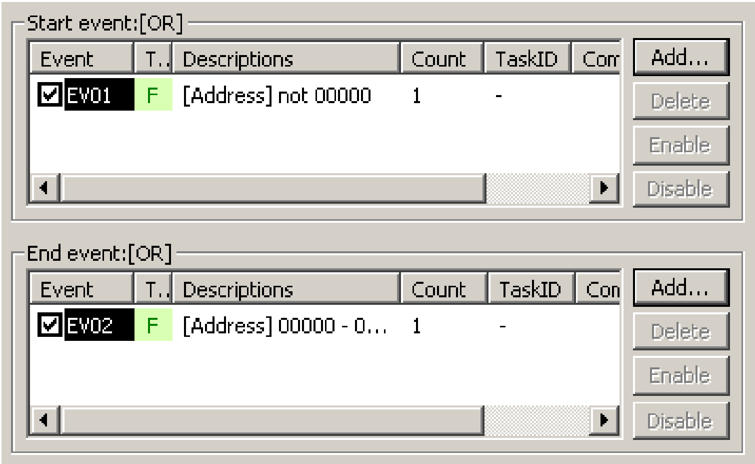
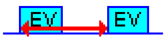
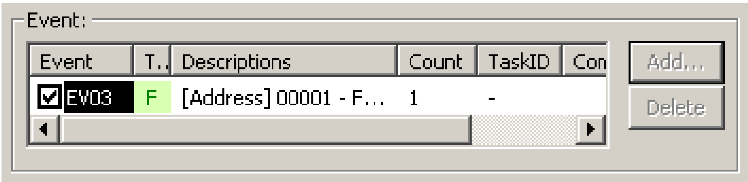
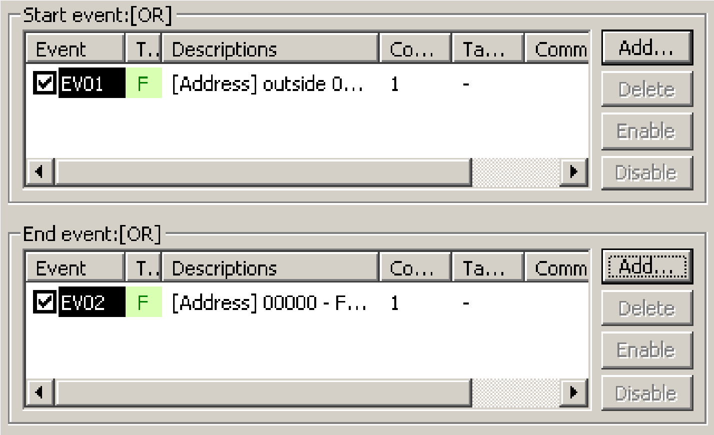
	[Disabled] Measurement is disabled.
	[Between two events]  Figure 5.68 Between two events Measurement is performed between the start event and the end event. Specifically, the time execution takes and number of passes through the range between the start event and the end event are measured. The measurement of time starts when the start event occurs and is suspended when the end event occurs. The number of passes through the section is incremented by one each time the pair of the start event and end event for the specified range occur. Start event: One or multiple events can be set. End event: One or multiple events can be set.
	[Period of an event]  Figure 5.69 Period of an event Measurement is performed during the event. Namely, the period between occurrences of the event the number of times it occurs are measured. The time from one occurrence of the event to the next is measured as one instance. The number of times is incremented by one each time the event occurs. Event: Only one event point can be set.

Table 5.25 Measurement modes (continued)

	[Interrupt-disabled range between two events]  Figure 5.70 Interrupt-disabled range between two events Measurement is of ranges over which interrupts are disabled from the start event to the end event. Specifically, the intervals over which interrupts are disabled and number of times interrupts are disabled within the range specified by Start event and End event are measured. The measurement of time starts when interrupts are disabled and is suspended when interrupts are re-enabled. The number of times is incremented by one each time interrupts are disabled. Start event: One or multiple events can be set. End event: One or multiple events can be set.

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

(2) Selecting the unit of measurement

This setting applies in common to all 8 sections. The following units of measurement are available:
10 ns (default), 20 ns, 40 ns, 80 ns, 160 ns, 1.6 µs
The maximum measurement time varies with the unit of measurement you set.

5.10.4 Starting Performance Measurement

When the user program is run, performance measurement is automatically started according to the conditions set on performance measurement.
When the user program is halted, the results of measurement are displayed in the Performance Analysis window.
When execution of the user program is halted and then restarted without changing the conditions of measurement, the newly measured times are added to the previous values.
To perform the measurements afresh, clear the results of measurement 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 popup menu to display the Performance Analysis Conditions dialog box. In the Performance Analysis Conditions dialog box, disable the condition you want to clear.

Figure 5.71 Performance Analysis Conditions dialog box

5.10.6 Clearing Results of Performance Measurement

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

5.10.7 Maximum Time of Performance Measurement

(1) Maximum measurement time

The timer used for performance measurement is comprised of a 40-bit counter.
The maximum measurement time varies with selected unit of measurement.
To select the unit of measurement, use the Time unit list box of the Performance Analysis Conditions dialog box.
The maximum measurable times for the respective units are listed in the table below.

Table 5.26 Maximum measurable times

Resolution	Maximum measurable time
10 ns	Approx. 3 hours, 03 minutes, 15 seconds
20 ns	Approx. 6 hours, 06 minutes, 30 seconds
40 ns	Approx. 12 hours, 13 minutes, 00 seconds
80 ns	Approx. 24 hours, 26 minutes, 00 seconds
160 ns	Approx. 48 hours, 52 minutes, 01 seconds
1.6 µs	Approx. 488 hours, 40 minutes, 18 seconds

CAUTION

Note that results of performance measurement carry an error equal to ±1 times the resolution (e.g. ±20 ns when the resolution is 20 ns).

(2) Maximum measured number of passes

Numbers of passes through sections are measured by a 32-bit counter. Measuring up to 4,294,967,295 passes is thus possible.

5.11 Measuring Code Coverage

5.11.1 Measuring Code Coverage

Code coverage refers to measures of the condition of a program in terms of 'digestion' by tests, i.e., the degree of thoroughness of tests of the software code (and the paths within it).
Information on instruction execution is displayed for the C/C++ and assembly-language levels.
This function collects information on instruction execution without causing execution of the program to break. Therefore, measuring code coverage does not affect the realtime characteristic of user-program execution.
The results of coverage are updated when a break is encountered.

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

Table 5.27 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 2 Mbytes of code-coverage memory for C0 level coverage and up to 1 Mbyte of code-coverage memory for C0 + C1 level coverage.
With the initial settings, code-coverage memory is automatically allocated to addresses in the ROM and RAM areas, in that order.

5.11.2 Opening the Code Coverage Window

Choose Code -> Code Coverage from the View menu or click on the Code Coverage toolbar button [].
The Code Coverage window is initially empty.

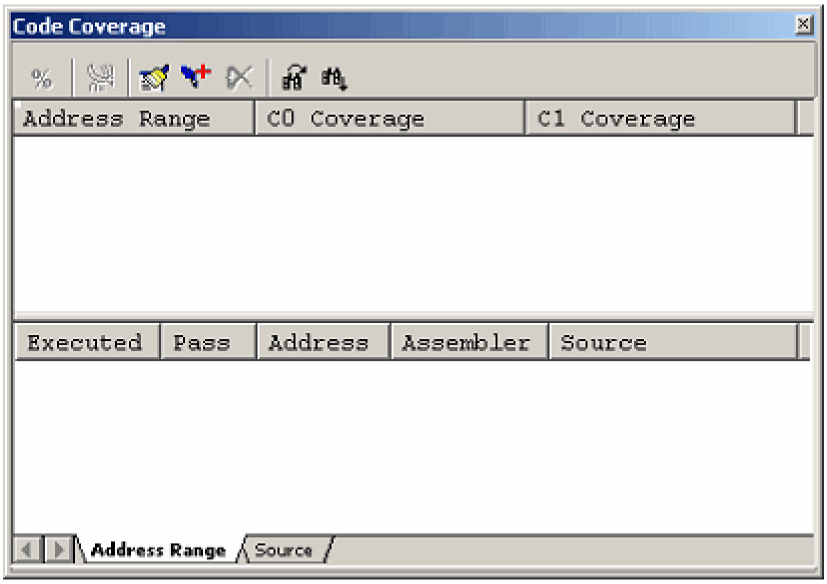


Figure 5.72 Code Coverage window

(1) Measurement method

The Code Coverage window has two sheets.

Table 5.28 Sheets of the Code Coverage window

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

The respective sheets permit registration of multiple ranges.
Up to two instances of the Code Coverage window can be open 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 assigned to the target address range. Coverage data can only be obtained from an address range to which memory has been allocated.
To allocate code coverage memory, use the Allocation of Code Coverage Memory dialog box.
To open this dialog box, select [Hardware Settings…] from the popup menu of the Code Coverage window.

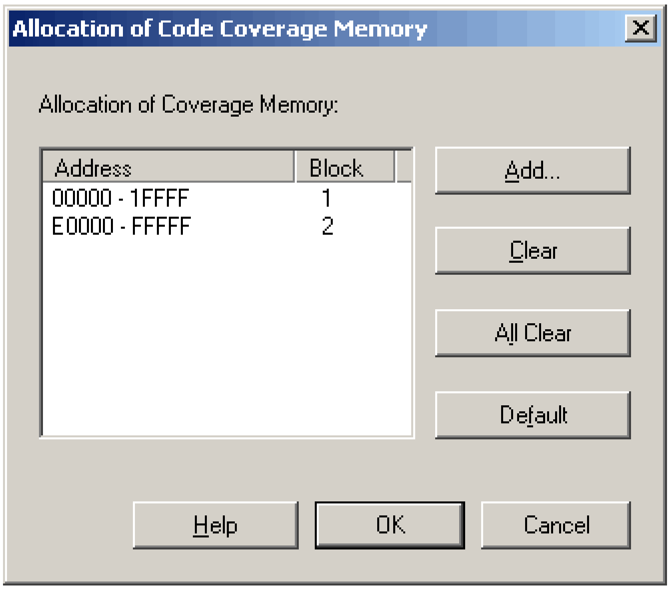


Figure 5.73 Allocation of Code Coverage Memory dialog box

When using C0 level coverage and C1 level coverage, you can specify a number of blocks from 1 to 8 (for a total of up to 2 Mbytes), each beginning on a 256-Kbyte boundary, and a number of blocks from 1 to 8 blocks (for a total of up to 1 Mbyte), each beginning on a 128-Kbyte boundary, as areas for the respective forms of code coverage measurement.
The blocks may be contiguous or non-contiguous.
With the initial settings, the coverage memory is automatically allocated to addresses in the ROM and RAM areas.

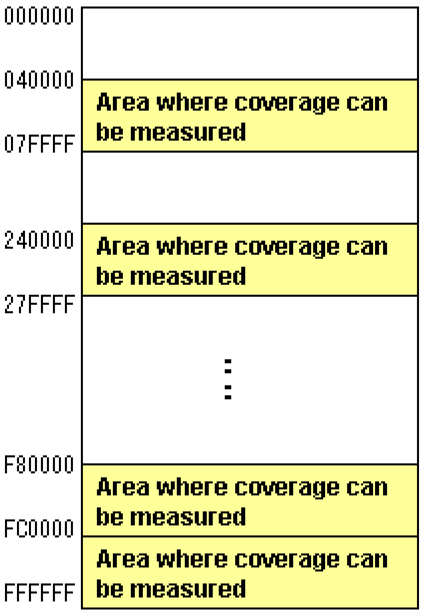


Figure 5.74 Schematic view of coverage memory allocation

(2) Changing memory allocation

When the allocation of coverage memory is changed, the coverage data acquired from the target address ranges prior to the change is retrieved from coverage memory into a dedicated coverage buffer.

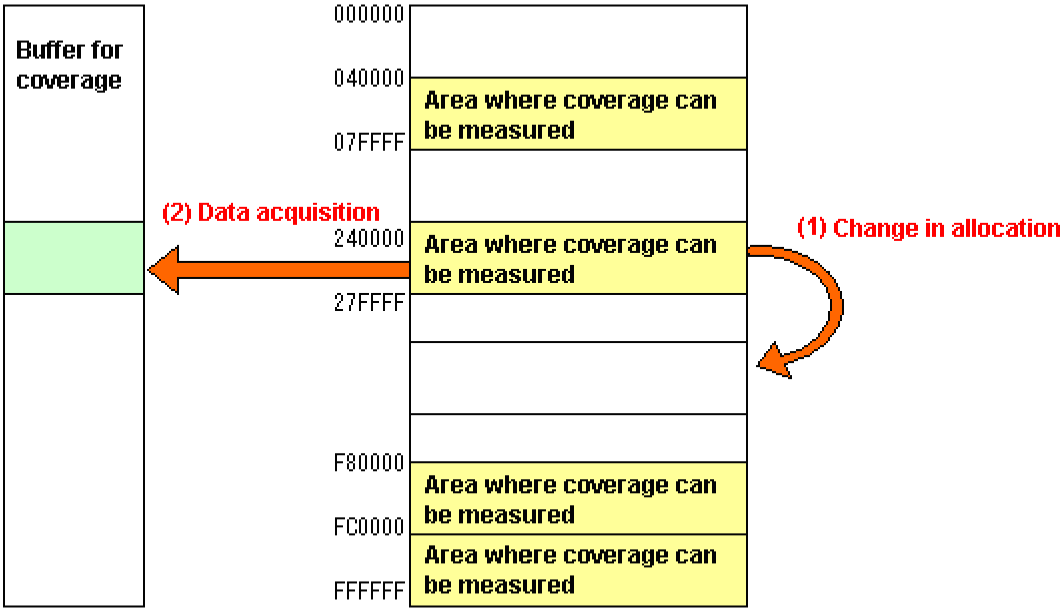


Figure 5.75 Schematic view of a change in coverage memory allocation

Acquired coverage information is accumulated in the coverage buffer until it is cleared by the user. However, coverage information is not updated for areas to which coverage memory is not allocated.

The coverage information shown in the Code Coverage window includes the information from the contents of the coverage 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 larger than 2 Mbytes or even an area to which no coverage memory has been allocated can be specified. However, when coverage memory has not been allocated to an area, coverage information on that area is not updated.

Areas for which coverage information is not updated are grayed-out.

An example display is shown below.

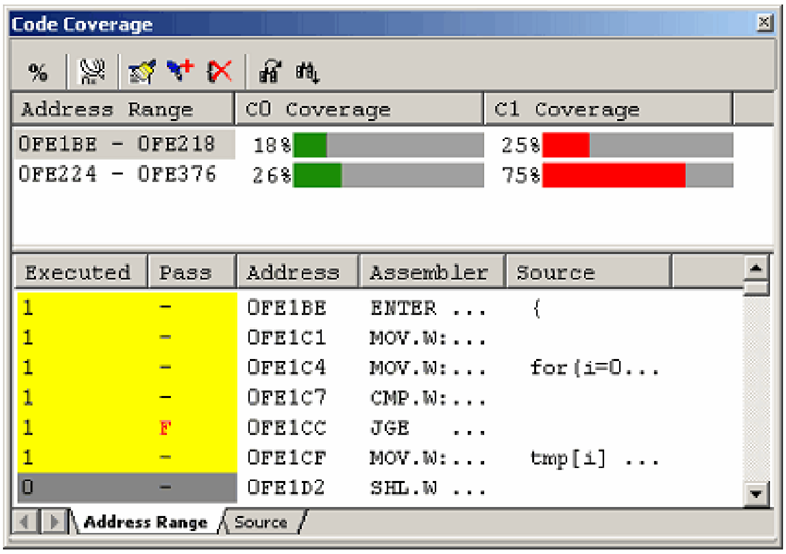


Figure 5.76 Code Coverage window (address specification)

The Code Coverage window is vertically divided in two by the splitter.

The upper pane shows the address ranges to be measured, and the degrees of C0 coverage and C1 coverage.

Table 5.29 Contents of the upper pane of the Code Coverage window

[Address Range]	Address range for which coverage is measured
[C0 Coverage]	C0 coverage as a percentage and graph
[C1 Coverage]	C1 coverage as a percentage and graph

The lower pane shows a detailed (assembly-language level) view of the address range selected in the upper pane.

Table 5.30 Contents of 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.
[Address]	Address of the instruction
[Assembler]	Disassembled program
[Source]	C/C++ or assembly source program

Acquired coverage information is accumulated in memory until it is cleared by the user.
When you double click on an assembler instruction in the Address Range sheet, the corresponding source code is shown in the Editor window.
Be aware that the source code will not be displayed in the cases listed below.

- A source file that corresponds to the assembler line does not exist.
- No source line corresponds to the assembler line.
- Where no debugging information was included, such as when the assembler line is for a library.

5.11.5 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.
A source file larger than 2 Mbytes or even an area to which no coverage memory has been allocated can be specified.
However, when coverage memory has not been allocated for a portion of the code, coverage information on that area is not updated.
Address lines where coverage information is not updated are grayed-out.
An example display is shown below.

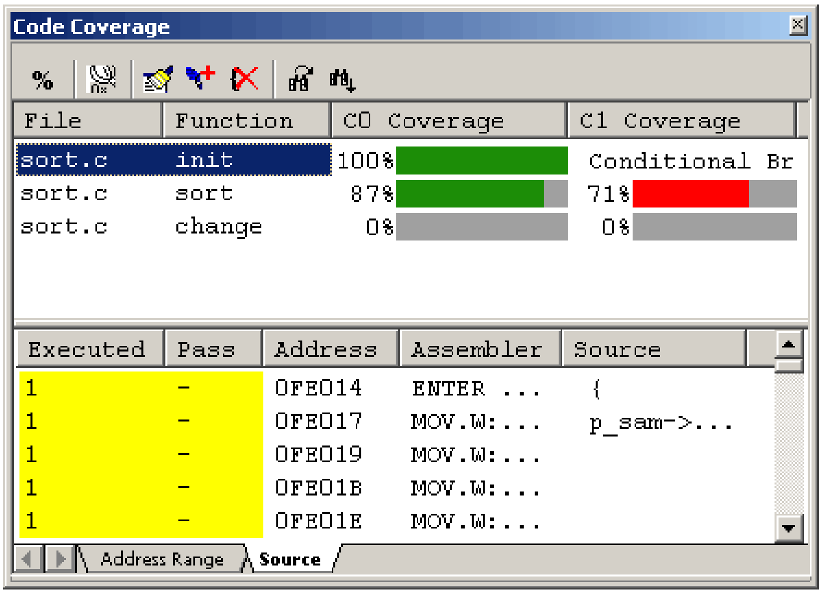


Figure 5.77 Code Coverage window (source file specification):

The Code Coverage window is vertically divided in two by the splitter.
The upper pane shows the address ranges to be measured (file and function names), C0 coverage and C1 coverage.

Table 5.31 Contents of the upper pane of the Code Coverage window

[File]	File name
[Function]	Function name
[C0 Coverage]	C0 coverage as a percentage and graph
[C1 Coverage]	C1 coverage as a percentage and graph

The lower pane shows a detailed (assembly-language level) view of the address range selected in the upper pane.

Table 5.32 Contents of 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.
[Address]	Address of the instruction
[Assembler]	Disassembled program
[Source]	C/C++ or assembly source program

The acquired coverage information is accumulated in memory until it is cleared by the user.

5.11.6 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 popup 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.

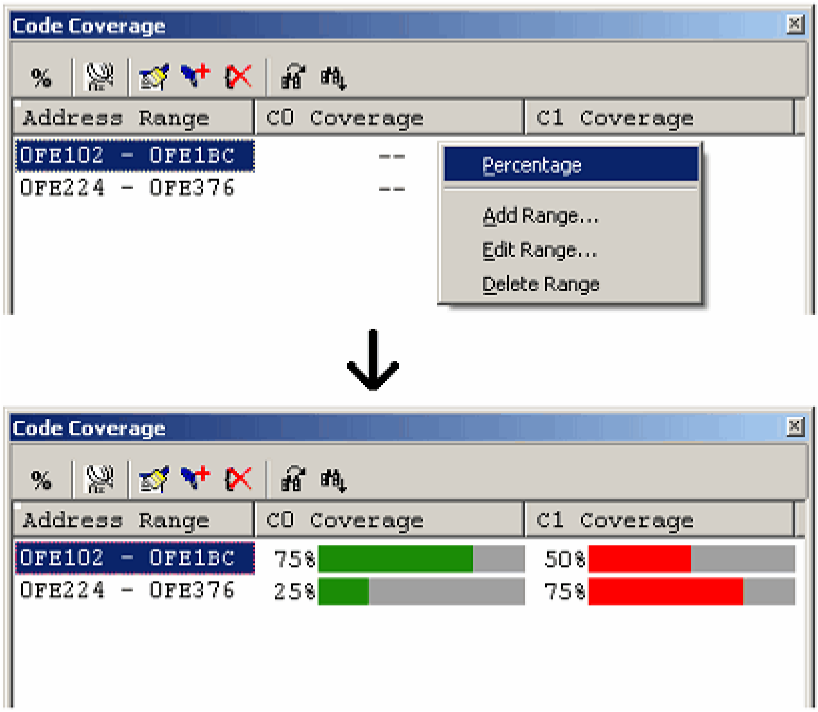


Figure 5.78 Code Coverage window

5.11.7 Sorting Coverage Data

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 file name. Lines for 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.
Clicking on the column once sorts the values into descending order. Clicking on the column a second time sorts the values into 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 that order

The data for each file is sorted by 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%
```

5.11.8 Searching for Nonexecuted Lines

Search for nonexecuted lines in a selected address range or function. When you click on the Find toolbar button [ ], the Find dialog box shown below appears.

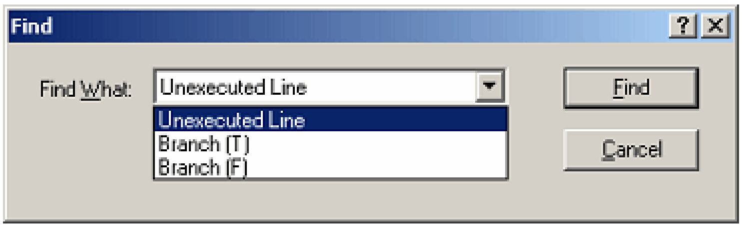


Figure 5.79 Find dialog box

The following three search options are available.

Table 5.33 Search options

Unexecuted Line	Instructions not executed yet
Branch (T)	Branch instructions with condition that is always TRUE when tested
Branch (F)	Branch instructions with condition that is always FALSE when tested

Clicking on the Find Next button [ ] starts a search.
When a matching instruction is found, the corresponding line is highlighted.
When no matching instructions are found, a message is displayed.

5.11.9 Clearing Code Coverage Information

(1) Clearing the code coverage information for a specified range

Selecting Clear Coverage Range from the popup menu opens the Clear Address Range dialog box.

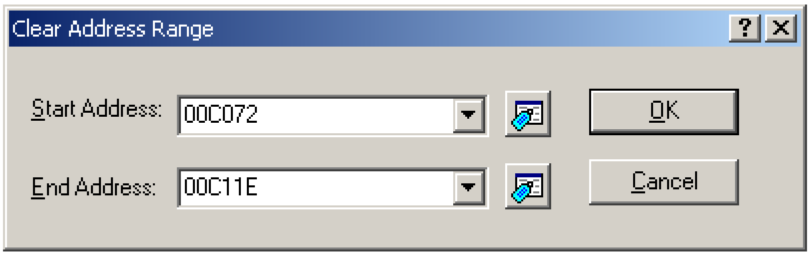


Figure 5.80 Clear Address Range dialog box

Enter the addresses where the range to be cleared starts and ends. Clicking on the OK button then clears the coverage information for the selected range.

(2) Clearing all of the code coverage information

Selecting Clear the Entire Coverage from the popup menu clears all of the code coverage information.

5.11.10 Updating Coverage Information

Selecting Refresh from the popup menu updates the contents of the Code Coverage window.
If Lock Refresh has been selected, the information is not automatically updated when program execution breaks. To view the latest information, therefore, you must manually select updating.

5.11.11 Preventing Updates to Coverage Information

Selecting Lock Refresh from the popup menu prevents updates to the Code Coverage window while the execution of the user program is stopped.

5.11.12 Saving the Code Coverage Information in a File

You can save the code coverage information for the currently selected sheet in a file.
Selecting Save Data from the popup menu opens the Save Coverage Data dialog box.

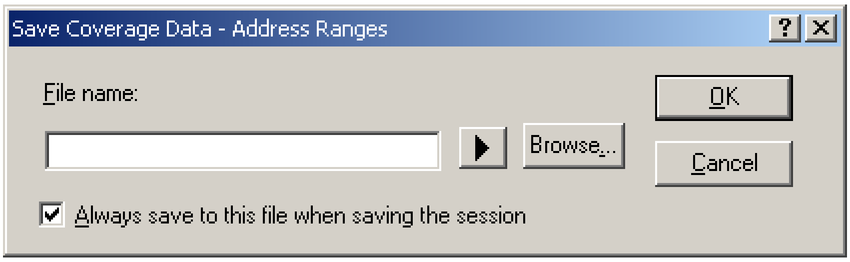


Figure 5.81 Save Coverage Data dialog box

Enter the name of the file where you want the information to be saved.
If the file-name extension is omitted, ".cov" will automatically be appended as the extension.
If you specify an existing file name, that file will be overwritten.

5.11.13 Loading Code Coverage Information from a File

You can load code-coverage information files.
Selecting Load Data from the popup menu opens the Load Coverage Data dialog box.

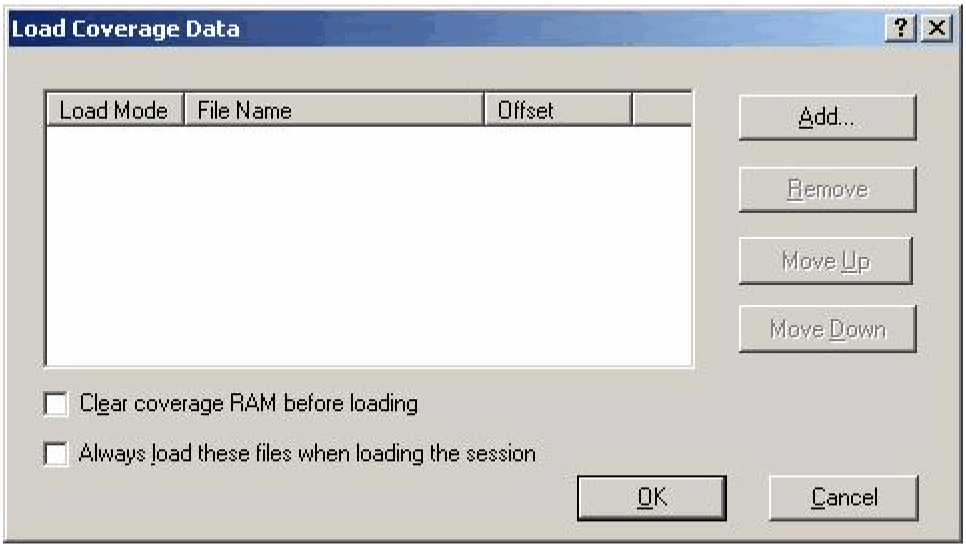


Figure 5.82 Load Coverage Data dialog box

Clicking on the Add button opens the Add Coverage Files dialog box shown below.

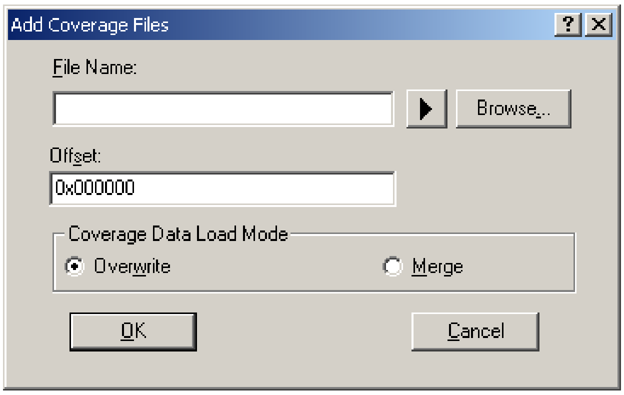


Figure 5.83 Add Coverage Files dialog box

Use this dialog box to specify the coverage information file you want to load. You can also specify a mode of loading and offset for each file you load.

The only file-name extension allowed is ".cov". An error message will appear if any other 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're loading is of the source-file type, you cannot specify an offset.

5.11.14 Modes of Loading for Coverage Information Files

Two modes of loading are available for coverage information files. They are schematically depicted below.

(1) When "Overwrite" has been selected

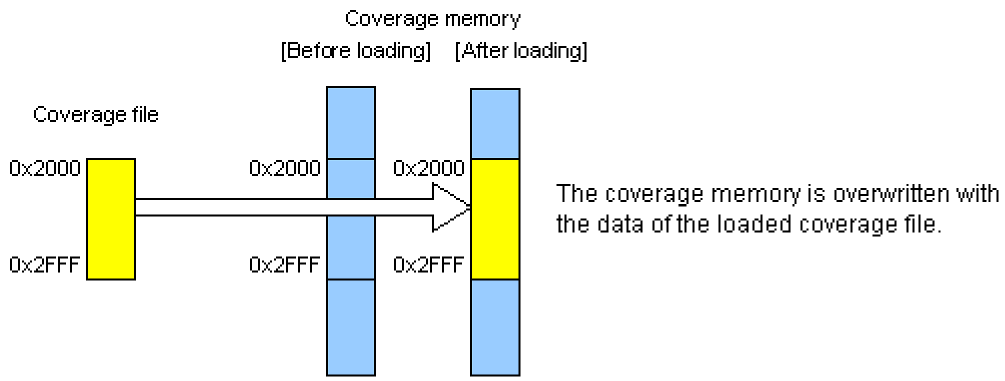


Figure 5.84 Schematic view of the overwrite mode

(2) When "Merge" has been selected

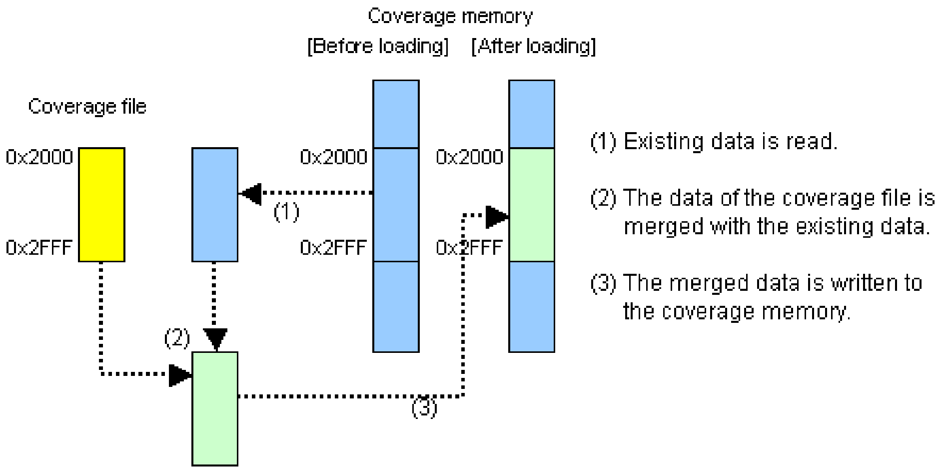


Figure 5.85 Schematic view of the merge mode

(3) Example of application of the merge mode

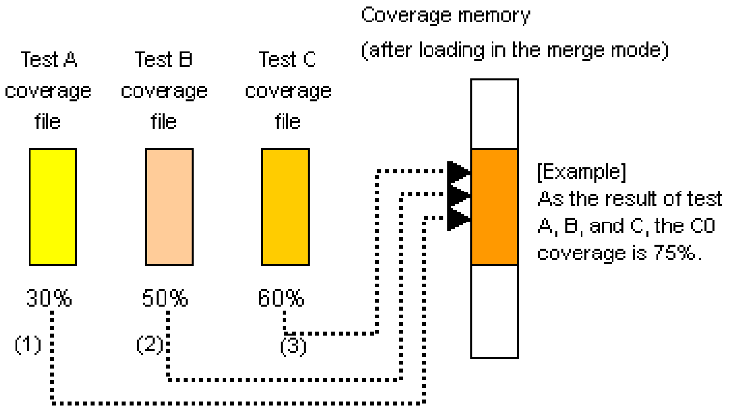


Figure 5.86 Schematic view of a merge-mode application

[Procedure]

(1) Open the Load Coverage Data dialog box.
To begin with, select the "Clear coverage RAM before loading" checkbox.
(2) In the merge mode, add the coverage file for test A.
(3) In the merge mode, add the coverage file for test B.
(4) In the merge mode, add the coverage file for test C.
(5) Click on the OK button.

You have now finished merging three files.
By re-calculating the percentages in the Code Coverage window, you can view the coverage (as percentages) of the tests as a whole.
Furthermore, you can save the merged data in a single file and manage the data accordingly.

5.11.15 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 the Code Coverage column.

Rows of the Code Coverage column that correspond to source lines where the instructions have been executed are highlighted.

If the user changes any setting related to coverage information in the Code Coverage window, the contents of the corresponding Code Coverage column will also be updated.

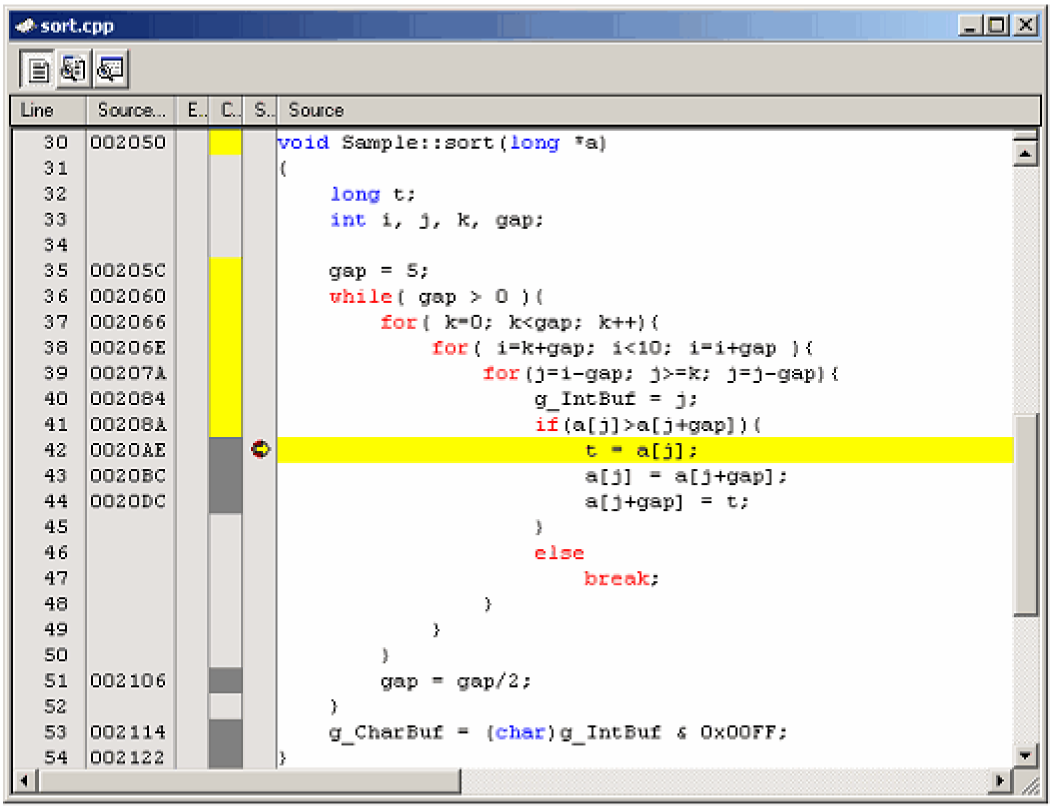


Figure 5.87 Example of code coverage results

5.12 Measuring Data Coverage

5.12.1 Measuring Data Coverage

The code coverage, data coverage and realtime profiling functions of the E100 emulator are mutually exclusive.
To use the data coverage function, choose Data coverage in the Switching function section on the System page of the Configuration properties dialog box.

Data coverage indicates the kinds of access to data areas. The emulator is capable of acquiring information on access per byte without causing program execution to break. Therefore, the realtime characteristic of user-program execution will not be affected.
The coverage results are updated upon a break.

The E100 emulator comes with 512 Kbytes of data coverage memory.
With the initial settings, the data coverage memory is automatically allocated to addresses in the ROM and RAM areas, in that order.

5.12.2 Opening the Data Coverage Window

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

The Data Coverage window is initially empty.

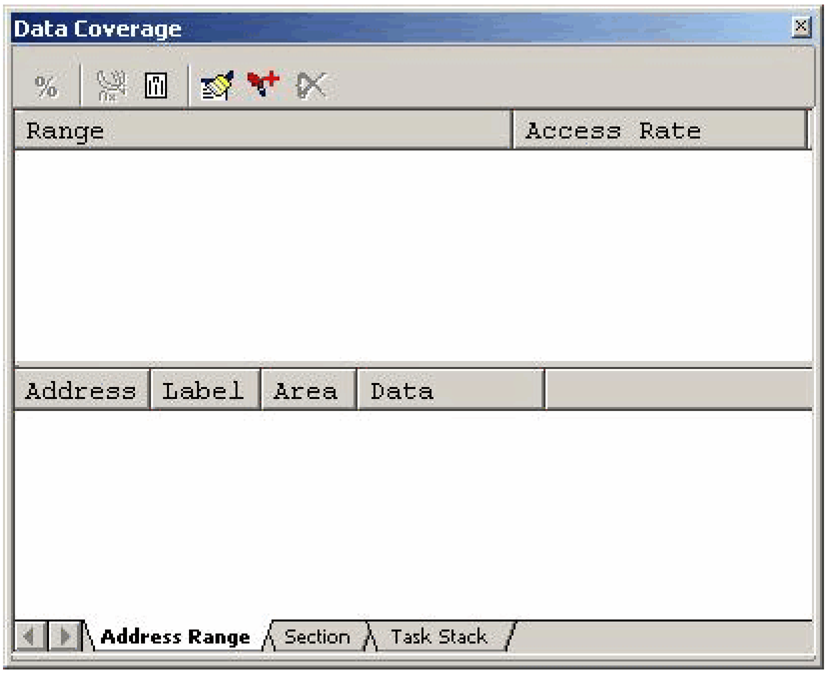


Figure 5.88 Data Coverage window

(1) Measurement method

The Data Coverage window has three sheets.

Table 5.34 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 performed for all task stack areas.

The respective sheets permit multiple ranges to be registered.
The Task Stack sheet only supports 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 only be obtained from an address range to which memory has been 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 popup menu of the Data Coverage window.

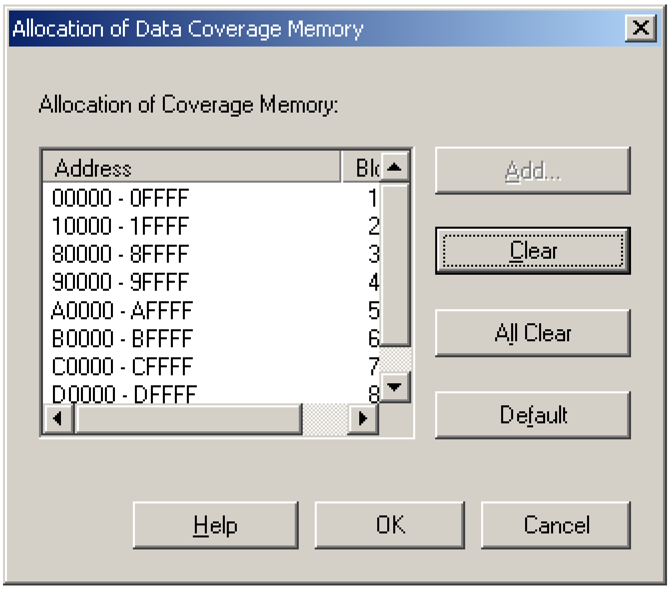


Figure 5.89 Allocation of Data Coverage Memory dialog box

You can specify any number of blocks from 1 to 8 (for a total of up to 512 Kbytes), each beginning on a 64-Kbyte boundary, as areas for data-coverage measurement.
The blocks may be contiguous or non-contiguous.
With the initial settings, the coverage memory is automatically allocated to addresses in the ROM and RAM areas.

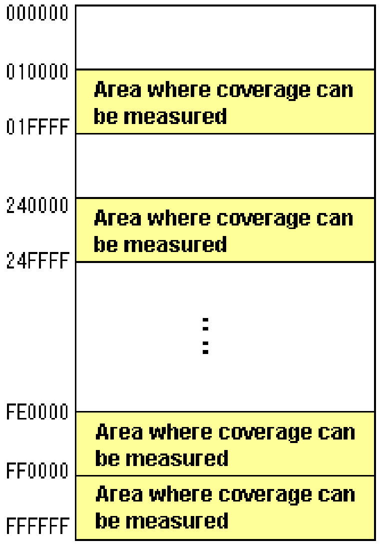


Figure 5.90 Schematic view of data coverage memory allocation

(2) Changing memory allocation

When the allocation of coverage memory is changed, the coverage data acquired from the target address ranges prior to the change is retrieved from coverage memory into a dedicated coverage buffer.

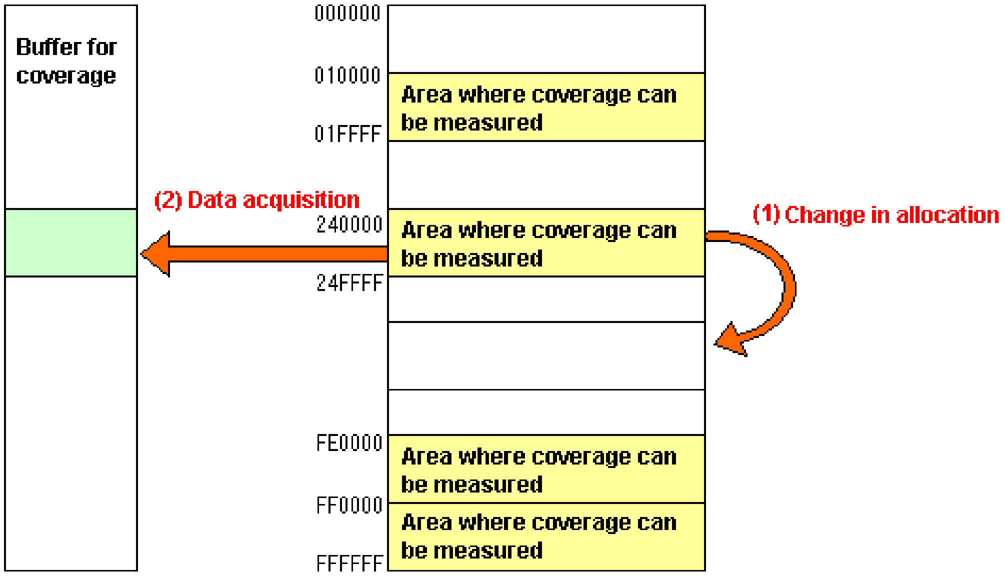


Figure 5.91 Schematic view of a change in data coverage memory allocation

Acquired coverage information is accumulated in the coverage buffer until it is cleared by the user. However, coverage information is not updated for areas to which coverage memory is not allocated.
The coverage information shown in the Data Coverage window includes the information from the contents of the coverage buffer.

5.12.4 Data Coverage in an Address Range

The E100 emulator is capable of collecting the access information for a user-specified address range and of displaying the information.

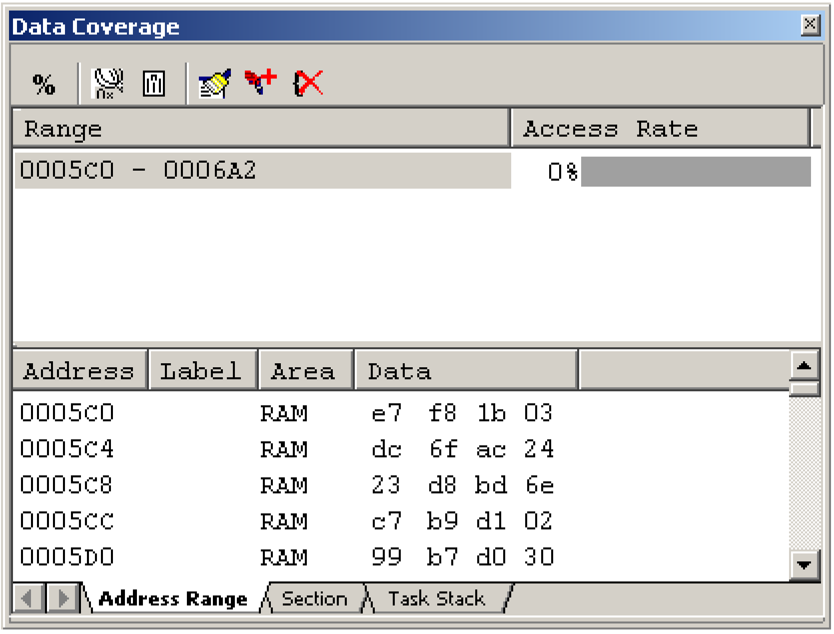


Figure 5.92 Data Coverage window (address specification)

The Data Coverage window is vertically divided in two by the splitter.
The upper pane shows the address ranges to be measured and access rates.

Table 5.35 Contents of the upper pane of the Data Coverage window

[Range]	Address range for which coverage is measured
[Access Rate]	Access rate as a percentage and graph

The lower pane shows a detailed view of the address range selected in the upper pane.

Table 5.36 Contents of the lower pane of the Data Coverage window

[Address]	Address value
[Label]	Label name
[Area]	Memory area (flash ROM, RAM, or SFR). This column is blank when the area is unused.
[Data]	Memory data. Data that have been accessed are displayed against a purple background.

Lines for addresses beyond the area to which coverage memory has been allocated are grayed-out. Although any existing coverage information for such addresses is retained, the coverage information will not be updated by program execution.
Acquired coverage information is accumulated in memory until it is cleared by the user.

5.12.5 Data Coverage in Sections

The E100 emulator is capable of collecting the access information for a user-specified section and of displaying the information.

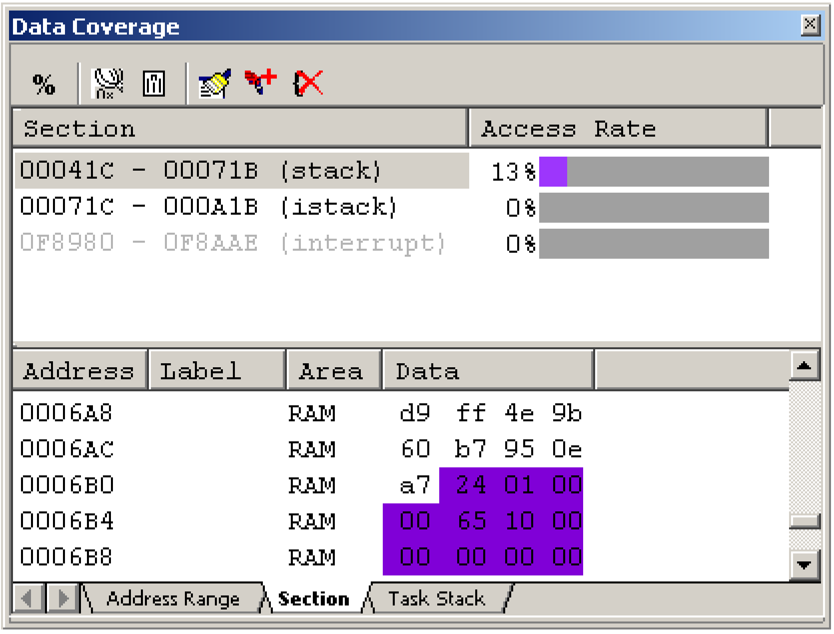


Figure 5.93 Data Coverage window (section name specification)

The Data Coverage window is vertically divided in two by the splitter.
The upper pane shows the address ranges (section names) to be measured and access rates.

Table 5.37 Contents of the upper pane of the Data Coverage window

[Section]	Address range (section) for which coverage is measured
[Access Rate]	Access rate as a percentage and graph

The lower pane shows a detailed view of the address range selected in the upper pane.

Table 5.38 Contents of the lower pane of the Data Coverage window

[Address]	Address value
[Label]	Label name
[Area]	Memory area (flash ROM, RAM, or SFR). This column is blank when the area is unused.
[Data]	Memory data. Data that have been accessed are displayed against a purple background.

Lines for addresses beyond the area to which coverage memory has been allocated are grayed-out. Although any existing coverage information for such addresses is retained, the coverage information will not be updated by program execution.
Acquired coverage information is accumulated in memory until it is cleared by the user.

5.12.6 Data Coverage in the Task Stack

The E100 emulator is capable of collecting the access information for the task stacks and of displaying the information.

The task stack is automatically registered when a load module that includes an OS has been downloaded.

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.

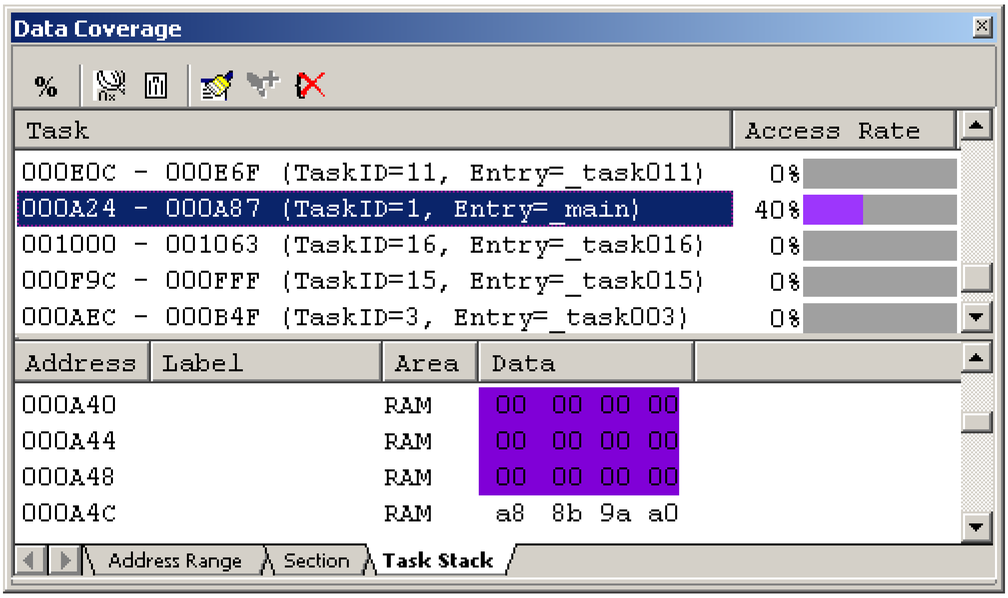


Figure 5.94 Data Coverage window (task stack specification)

The Data Coverage window is vertically divided in two by the splitter.

The upper pane shows the automatically registered task stacks and access rates.

Table 5.39 Contents of the upper pane of the Data Coverage window

[Task]	Task stack (task ID and task entry label)
[Access Rate]	Access rate as a percentage and graph

The lower pane shows a detailed view of the task stack selected in the upper pane.

Table 5.40 Contents of the lower pane of the Data Coverage window

[Address]	Address value
[Label]	Label name
[Area]	Memory area (flash ROM, RAM, or SFR). This column is blank when the area is unused.
[Data]	Memory data. Data that have been accessed are displayed against a purple background.

Lines for addresses beyond the area to which coverage memory has been allocated are grayed-out. Although any existing coverage information for such addresses is retained, the coverage information will not be updated by program execution.
Acquired coverage information is accumulated in memory until it is cleared by the user.

5.12.7 Clearing Data Coverage Information

(1) Clearing the data coverage information for a specified range

Selecting Clear Coverage Range from the popup menu on the Address Range or Section sheet opens the Clear Coverage Range dialog box.

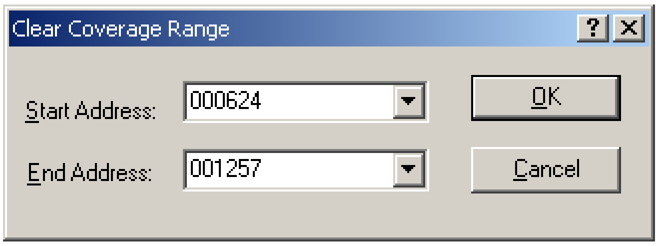


Figure 5.95 Clear Coverage Range dialog box

Enter the addresses where the range to be cleared starts and ends. Clicking on the OK button then clears the coverage information for the selected range.

(2) Clearing all of the data coverage information

Selecting Clear the Entire Coverage from the popup menu clears all of the data coverage information.

5.12.8 Updating Coverage Information

Selecting Refresh from the popup menu updates the content of the Data Coverage window.
If Lock Refresh has been selected, the information is not automatically updated when program execution breaks. To view the latest information, therefore, you must manually select updating.

5.12.9 Preventing Updates to Coverage Information

Selecting Lock Refresh from the popup menu prevents updates to the Data Coverage window while the execution of the user program is stopped.

5.12.10 Saving the Data Coverage Information in a File

You can save the data coverage information for the currently selected sheet in a file.
Selecting Save Data from the popup menu opens the Save Data dialog box.

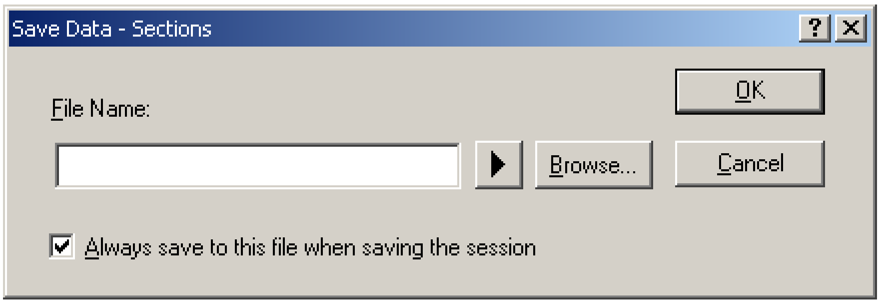


Figure 5.96 Save Data dialog box

Enter the name of the file where you want the information to be saved. If the file-name extension is omitted, ".cdv" will automatically be appended as the extension. If you specify an existing file name, that file is overwritten.

5.12.11 Loading Data Coverage Information from a File

You can load coverage information files.
Selecting Load Data from the popup menu opens the Load Coverage Data dialog box.

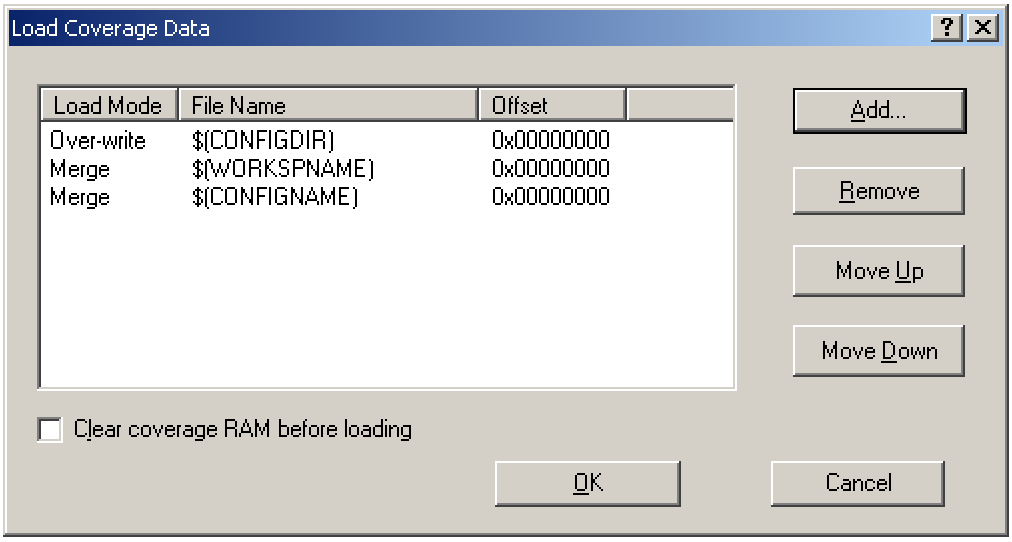


Figure 5.97 Load Coverage Data dialog box

Clicking on the Add button opens the Add coverage data file dialog box shown below.

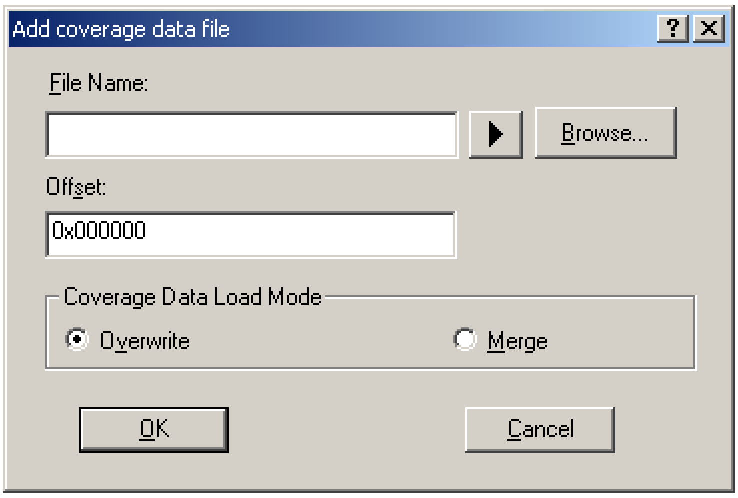


Figure 5.98 Add coverage data file dialog box

Use this dialog box to specify the coverage information file you want to load. You can also specify a mode of loading and offset for each file you load.

The only file-name extension allowed is ".cdv". An error message will appear if any other 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 code coverage, data coverage and realtime profiling functions of the E100 emulator are mutually exclusive.
To use the realtime profiling function, choose Real-time profile in the Switching function section on the System page of the Configuration properties dialog box.
Realtime profiling refers to the measurement of performance per function or task within an area allocated as a range for profiling. Realtime profiling will help you find where and how deterioration in the performance of application programs arises.
The process of measurement does not interfere with execution of the user program.
The results of measurement are updated when execution of the program breaks.

(1) Function profiles

Performance of individual functions can be measured.
For a function, the Realtime Profile window shows its name, the address where it starts, its size, the number of calls, cumulative execution time, the ratio of this to the overall execution time, and the average execution time.
In function profiling by the E100 emulator, execution times for subroutines are not included in the indicated cumulative execution time.

CAUTION
A function profile is subject to the following limitations:

(a) Areas to be measured

The E100 emulator can acquire profile information on all functions in up to 8 blocks, with each block a 128-Kbyte unit.
The blocks can be contiguous or non-contiguous.
Functions located beyond the boundaries of the blocks are not specifiable. In such cases, the entries for the functions (or tasks) are grayed-out.

(b) Limit on the number of functions

Measurement of up to 8K – 1 (= 8,191) functions is possible.
A limit of 8K – 1 (= 8,191) applies to the number of functions within the above scope of measurement. Measurement will not be performed for the functions beyond this limit. In such cases, the names, addresses, and sizes of the excess functions are grayed-out.

(c) In-line expansion

The functions that have been written for in-line expansion (optimization by the compiler) are not displayed in the Realtime Profile window.

(d) Recursive functions

Although the execution times of recursive functions can be measured correctly, they are only executed once.

(e) Relationship between the address where Go was executed and the address of a break within a measurement range, and the measurable range

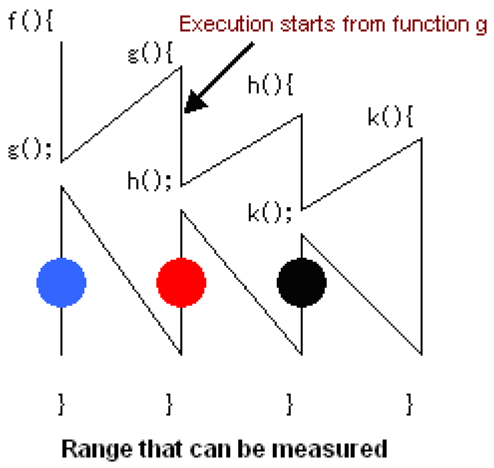


Figure 5.99 Measurable range

The measurable range will be as follows.
When execution of the program breaks at the location of a black dot [●]: Execution time and number of passes for functions h and k
When execution of the program breaks at the location of a red dot [●]: Execution time and number of passes for functions h and k
When execution of the program breaks at the location of a blue dot [●]: Execution time and number of passes for functions h and k
For the function g, the number of passes and time for the executed portion can be measured.

Even after execution has returned to a function higher in the hierarchy of calls, the number of calls cannot be measured for a function from which execution of the program started.

(f) Function measurement
Accurate measurement requires that execution of the function remained in progress for at least 100 ns. If this is not the case, the execution time and number of passes may be incorrect.

(g) Debugging information option
To get the execution time and number of passes for a function, you need to specify the option to output debugging information for the source file or library that includes the function at the time of compilation. If this option has not been specified, measurement of the execution time and number of passes for a function will not be possible.

(h) Maximum and minimum execution time
You cannot use the realtime profiling function to measure the maximum and minimum execution times for a function. To measure the maximum and minimum execution times for a function, use the Performance Analysis window.

(2) Task profile

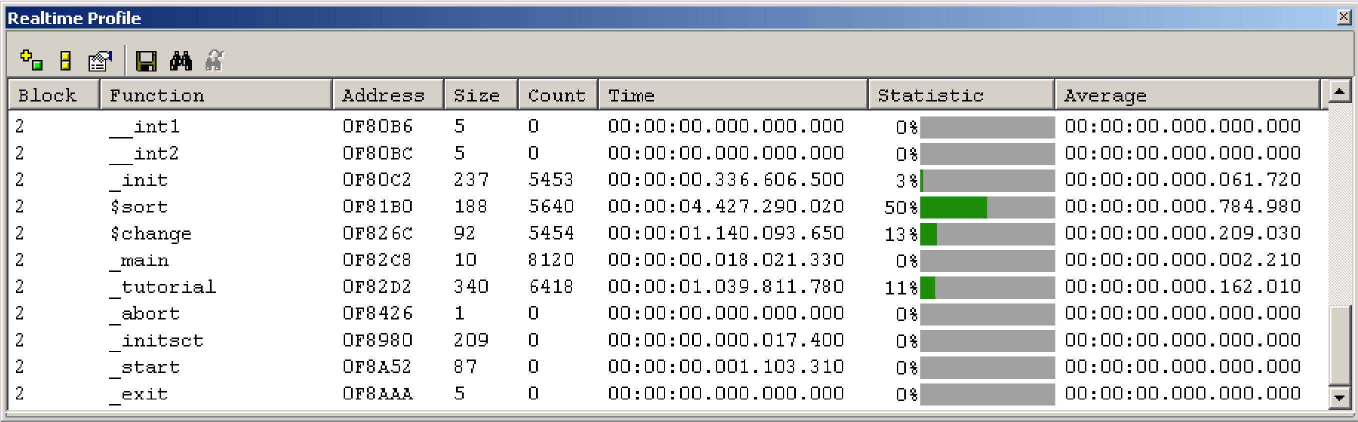
Performance of individual tasks can be measured.
For a task, the Realtime Profile window shows its ID, the number of passes, cumulative execution time, the ratio of this to the overall execution time, and the average execution time.

5.13.2 Selecting a Realtime Profile Measurement Mode

Choose Set Range from the popup menu that is displayed when you right-click in the window.
The Realtime Profile Setting dialog box will be displayed. In the Realtime Profile Mode list box of this dialog box, you can select "Function Profile" or "Task Profile."
When the profile mode is changed, all results of measurement are cleared.

5.13.3 Measuring Function Profiles

The Function Profile mode allows measurement of performance per function.



Block	Function	Address	Size	Count	Time	Statistic	Average
2	__int1	0F80B6	5	0	00:00:00.000.000.000	0%	00:00:00.000.000.000
2	__int2	0F80BC	5	0	00:00:00.000.000.000	0%	00:00:00.000.000.000
2	_init	0F80C2	237	5453	00:00:00.336.606.500	3%	00:00:00.000.061.720
2	$sort	0F81B0	188	5640	00:00:04.427.290.020	50%	00:00:00.000.784.980
2	$change	0F826C	92	5454	00:00:01.140.093.650	13%	00:00:00.000.209.030
2	_main	0F82C8	10	8120	00:00:00.018.021.330	0%	00:00:00.000.002.210
2	_tutorial	0F82D2	340	6418	00:00:01.039.811.780	11%	00:00:00.000.162.010
2	_abort	0F8426	1	0	00:00:00.000.000.000	0%	00:00:00.000.000.000
2	_initsct	0F8980	209	0	00:00:00.000.017.400	0%	00:00:00.000.000.000
2	_start	0F8A52	87	0	00:00:00.001.103.310	0%	00:00:00.000.000.000
2	_exit	0F8AAA	5	0	00:00:00.000.000.000	0%	00:00:00.000.000.000

Figure 5.100 Realtime Profile window (function profile)

The information in each of the columns is described in the table below.

Table 5.41 Details on each column

Block	Block number
Function	Function name
Address	Address where the function starts
Size	Function size
Count	Number of times the function has been called
Time	Cumulative time of function execution The timestamp is in the form shown below. Hours:minutes:seconds.milliseconds.microseconds.nanoseconds
Statistic	Ratio of the time for the given function to Go-Break time
Average	Average of the execution times for individual passes

If a function is outside the areas to which profile memory is allocated, the address line is grayed-out.

Acquired results of profile measurement are accumulated in memory until the user clears them.

5.13.4 Setting Ranges for Function Profile Measurement

Choose Set Range from the popup menu that is displayed when you right-click in the window.
The Realtime Profile Setting dialog box will be displayed. Set a profile measurement range in this dialog box.

[Function profile mode]

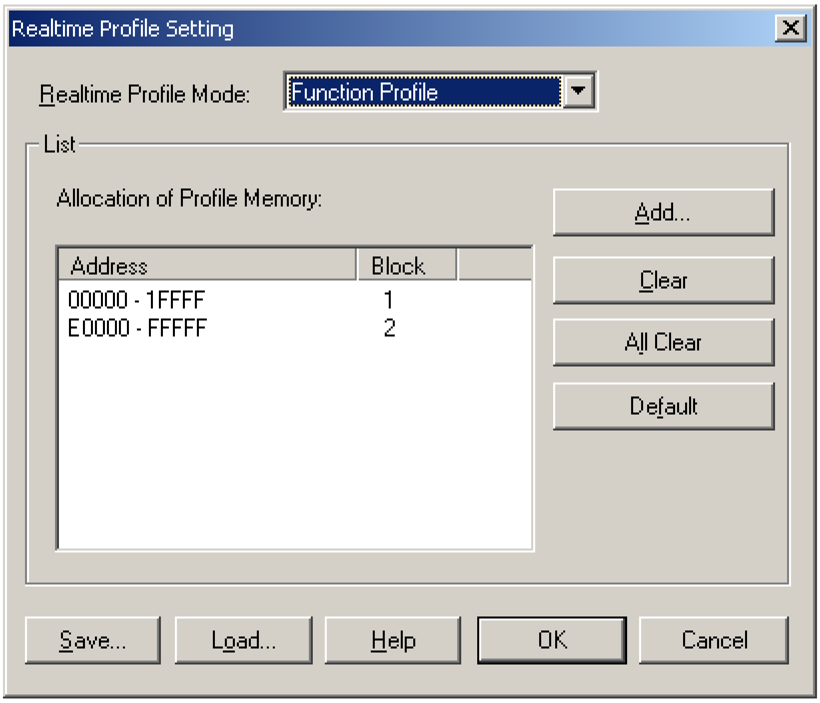


Figure 5.101 Realtime Profile Setting dialog box

(1) Memory allocation

Before function profiles can be measured, profile memory must be allocated to the addresses at which measurement will be performed. Profile data can only be obtained from address ranges to which memory has been allocated.
You can specify any number of blocks 1 to 8 (for a total of up to 1 Mbyte), each beginning on a 128-Kbyte boundary, as areas for profile measurement.
The blocks may be contiguous or non-contiguous.
With the initial settings, the profile memory is automatically allocated to addresses in the ROM and RAM areas.

(2) Automatic detection of functions

When profile memory is assigned to an address range, the E100 emulator automatically detects functions within that range and adds them to the window.

5.13.5 Saving Function Profile Measurement Settings

You can save the current profile mode and measurement ranges (memory allocation) for function profiles.
Click on the Save button of the Realtime Profile Setting dialog box, and the Save As dialog box will be displayed.
Enter the name of the file where you want the function profile measurement settings to be saved.
If the file-name extension is omitted, ".rpf" will automatically be appended as the 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 Settings

You can load function profile measurement settings.
Click on the Load button of the Realtime Profile Setting dialog box, and the Open dialog box will be displayed.

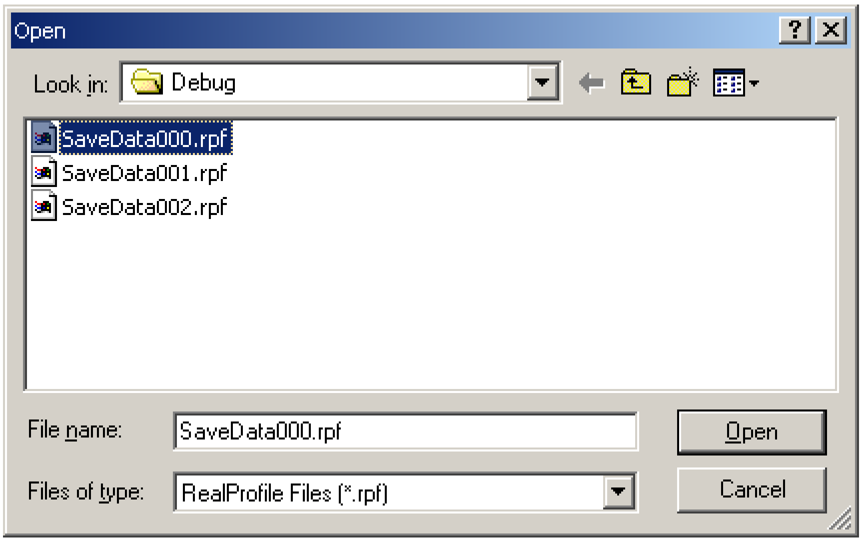


Figure 5.102 Open dialog box

Enter the name of the file you want to load.
Only files bearing the extension ".rpf" can be loaded. If you enter any other file-name extension, an error message will be output.
When loading of the file is complete, the list in the Realtime Profile Setting dialog box is updated.
If the information in the loaded file is for a task profile, the profile mode in the Realtime Profile Setting dialog box is switched to task mode.

5.13.7 Measuring Task Profiles

The Task Profile mode allows measurement of performance per task.

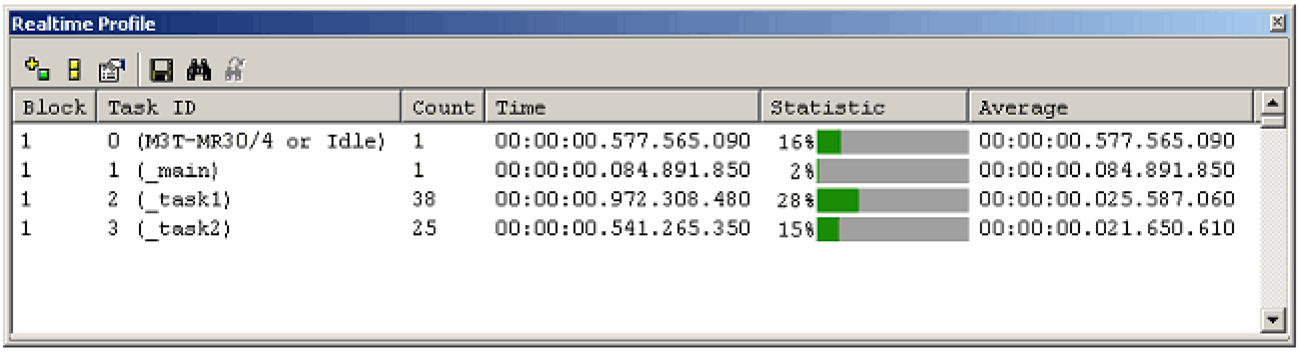


Figure 5.103 Realtime Profile window (task profile)

The information in each of the columns is described in the table below.

Table 5.42 Details on each column

Block	Block number
Task ID	Task ID, entry address
Count	Number of times the task has been called
Time	Cumulative time of task execution The timestamp is in the form shown below. Hours:minutes:seconds.milliseconds.microseconds.nanoseconds
Statistic	Ratio of the time for the given function to Go-Break time
Average	Average of the execution times for individual passes

Disabled tasks are grayed-out.

Acquired results of profile measurement are accumulated in memory until the user clears them.

5.13.8 Setting Ranges for Task Profile Measurement

Choose Set Range from the popup menu that is displayed when you right-click in the window.
The Realtime Profile Setting dialog box will be displayed. Set a profile measurement range in this dialog box.

[Task profile mode]

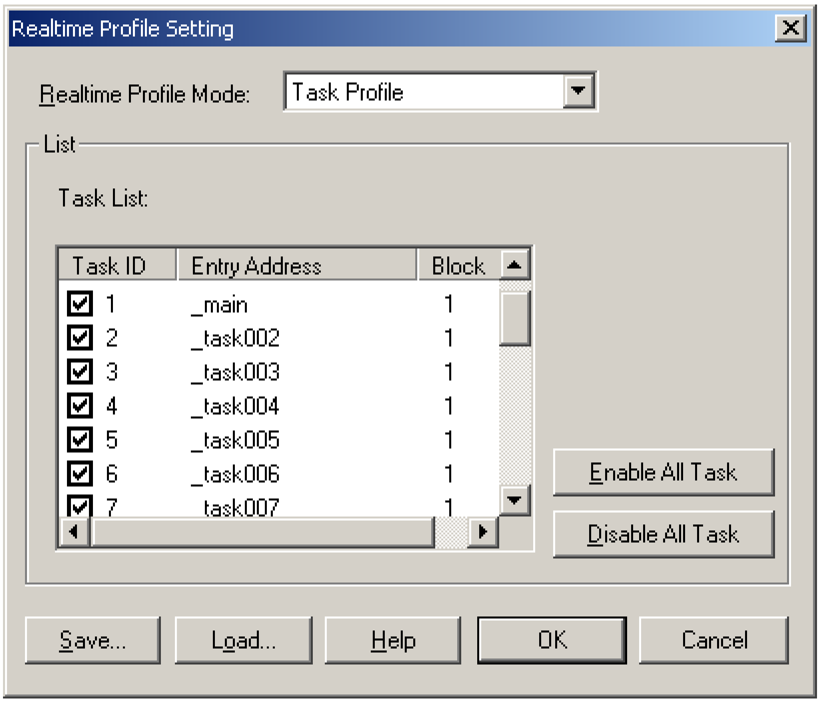


Figure 5.104 Realtime Profile Setting dialog box

(1) Automatic detection of tasks

If you have downloaded a load module that includes an OS, the E100 emulator automatically detects the tasks.

(2) Selecting tasks

Select the checkboxes next to the IDs of tasks you want to measure (by default, all checkboxes are selected).
The selected tasks will automatically be assigned block numbers (1–8).

CAUTION

When the eight blocks have been used up, the block number column for further tasks will be blank, indicating that measurement for tasks with these IDs is not possible. In such cases, deselect checkboxes against the IDs of tasks for which performance measurement is not necessary.

5.13.9 Saving Task Profile Measurement Settings

You can save the current settings regarding tasks for measurement (task IDs and enabled/disabled states) in task mode.
Click on the Save button of the Realtime Profile Setting dialog box, and the Save As dialog box will be displayed.
Enter the name of the file where you want the task profile measurement settings to be saved.
If the file-name extension is omitted, ".rpf" will automatically be appended as the 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 Settings

You can load task profile measurement settings.
Click on the Load button of the Realtime Profile Setting dialog box, and the Open dialog box will be displayed.

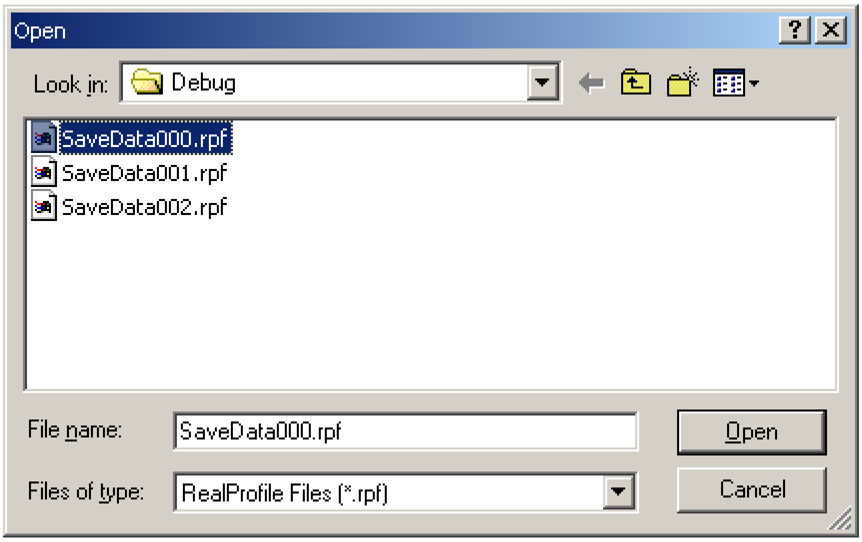


Figure 5.105 Open dialog box

Enter the name of the file you want to load.
Only files bearing the extension ".rpf" can be loaded. If you enter any other file-name extension, an error message will be output.
When loading of the file is complete, the list (of tasks) in the Realtime Profile Setting dialog box is updated.

Even if a loaded task ID does not currently exist, it will be temporarily displayed in the list of tasks in the Realtime Profile Setting dialog box. However, only tasks with the existing IDs will actually be registered when you click on the OK button.
You can re-open the Realtime Profile Setting dialog box to check the currently registered tasks.

If the information in the loaded file is for a function profile, the profile mode in the Realtime Profile Setting dialog box is switched to function mode.

5.13.11 Clearing Results of Realtime Profile Measurement

Choose Clear from the popup menu of the Realtime Profile window, and all results of measurement are cleared.
Unless this is done, measurement results are accumulated in memory.

5.13.12 Saving Results of Realtime Profile Measurement

You can save the current results of realtime profile measurement as text.
Choose Save To File from the popup menu of the Realtime Profile window, and the Save As dialog box will be displayed.

Enter the name of the file where you want the results of measurement to be saved.
If the file-name extension is omitted, ".txt" will automatically be appended as the 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 Unit of Measurement

Choose Properties from the popup menu that is displayed when you right-click in the window.
The Properties dialog box will be displayed.

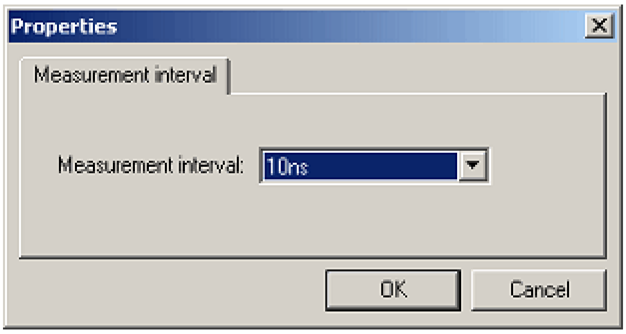


Figure 5.106 Properties dialog box

The unit of measurement can be selected from the following options:
10 ns, 20 ns, 40 ns, 80 ns, 160 ns, 1.6 μs

CAUTION
When the current selection is changed, the measurement results hitherto accumulated are cleared.

5.13.14 Maximum Measurement Time for Realtime Profiles

(1) Maximum measurement time

The timer used for realtime profile measurement is configured with a 40-bit counter. The maximum measurement time varies with the selected unit of measurement.
Select the unit of measurement from the Measurement interval drop-down list of the Properties dialog box.
The maximum measurable times for the respective units are listed below.

Table 5.43 Maximum measurable times

Resolution	Maximum measurable time
10 ns	Approx. 3 hours, 03 minutes, 15 seconds
20 ns	Approx. 6 hours, 06 minutes, 30 seconds
40 ns	Approx. 12 hours, 13 minutes, 00 seconds
80 ns	Approx. 24 hours, 26 minutes, 00 seconds
160 ns	Approx. 48 hours, 52 minutes, 01 seconds
1.6 µs	Approx. 488 hours, 40 minutes, 18 seconds

CAUTION
Note that results of performance measurement carry an error equal to ±(twice the resolution + 100ns), e.g. ±140 ns when the resolution is 20 ns, each time a function is entered. If the resolution is 20 ns and a function is entered 10 times, the error is ±1400 ns.

(2) Maximum measured number of calls
For a realtime profile, a 16-bit counter measures the number of times a task or function is executed. Measurement of up to 65,535 calls is thus possible.

5.14 Detecting Exceptional Events

5.14.1 Detecting Exceptional Events

The E100 emulator permits you to detect the occurrence of various exceptional events during user program execution. Exceptional events include abnormal behavior of the user program, as well as an overflow of the measurement counter for break, trace, or performance analysis. Detection of a specific 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 access in violation of a specified access attribute is attempted.
- Reading from non-initialized memory: An error is detected when a non-initialized area (not written) is read.
- Stack access violation: An error is detected when the value of the stack register is beyond a boundary of the stack area.
- Performance-measurement 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 Violations of Access Protection

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.

(1) Access attributes

The following attributes are specifiable in word units for any area.

Read/Write: Accessible for both reading and writing
Read Only: Only accessible for reading
Write Only: Only accessible for writing
Disable: Access prohibited
Disable (OS): Access other than from the OS is prohibited (this attribute is automatically assigned when a program that includes an OS is downloaded).

(2) Protected areas

Any area in the entire memory space can be protected.
At the time the emulator is booted up, all areas are assigned the Read/Write attribute by default.

(3) Methods of setting protection

There are the following two methods:

- Automatic setting by section information in a downloaded module
- Individually specifying an access attribute for an area

(4) Method of detection

Violation of access protection is detected by internal resources (blocks 1–16) of the emulator.
The blocks are automatically allocated by an original algorithm of the emulator.

CAUTION
Since the emulator’s internal resources are limited, not all blocks can be protected. If an error occurs, reduce the number of assigned blocks by using the ‘Delete’ button before setting protection again.

		Access attribute		
		Read/Write		
Write	->	Read Only	->	Violation detected
Read	->	Write Only	->	Violation detected
Read Write	->	Disable	->	Violation detected

Figure 5.107 Patterns for detecting violation

(5) Action taken when violation of access protection is detected

The following actions are selectable.

- Display a warning.

After the Violation of access protection checkbox has been selected on the Exception Warning page of the Configuration properties dialog box, you will see a warning in the Status window and in a status bar balloon when errors of this type occur.

- Make the detection of violation of access protection a condition of a hardware breakpoint.
- Make the detection of violation of access protection a condition of a trace point.

5.14.3 Setting Protection for an Area

Follow the procedure below to set protection for an area.

(1) From the Hardware Break dialog box

1. Select the Exception checkbox on the Hardware Break sheet and then click on the Detail button.

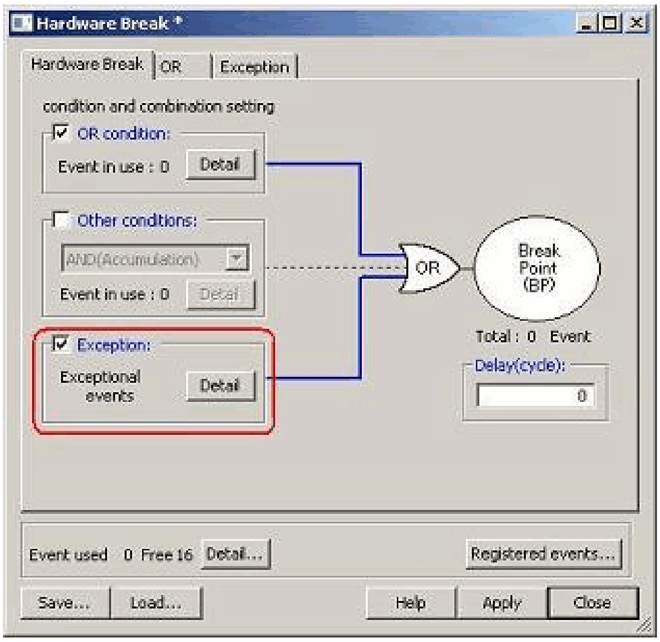


Figure 5.108 Hardware Break dialog box

2. The Exception page shown below will appear. Click the Detail button to the right of the Violation of access protection checkbox.

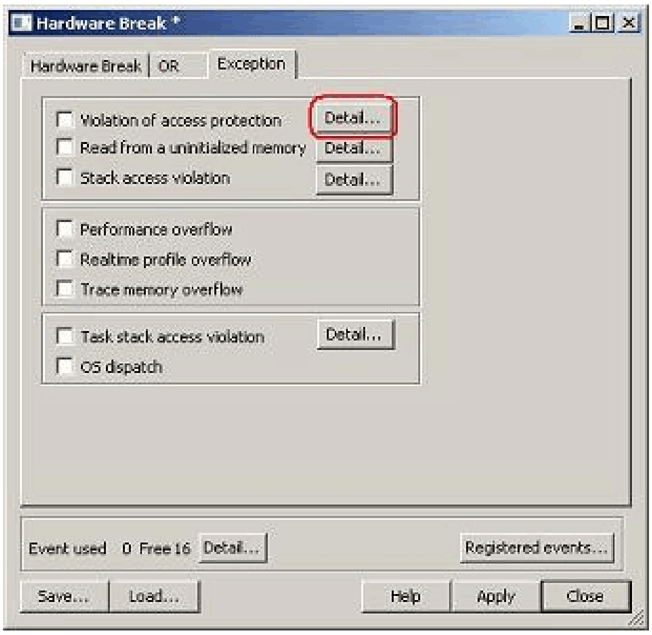


Figure 5.109 Hardware Break dialog box

3. 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 the downloaded module when a program is downloaded, select the checkbox labeled “Automatically set address areas at downloading.”

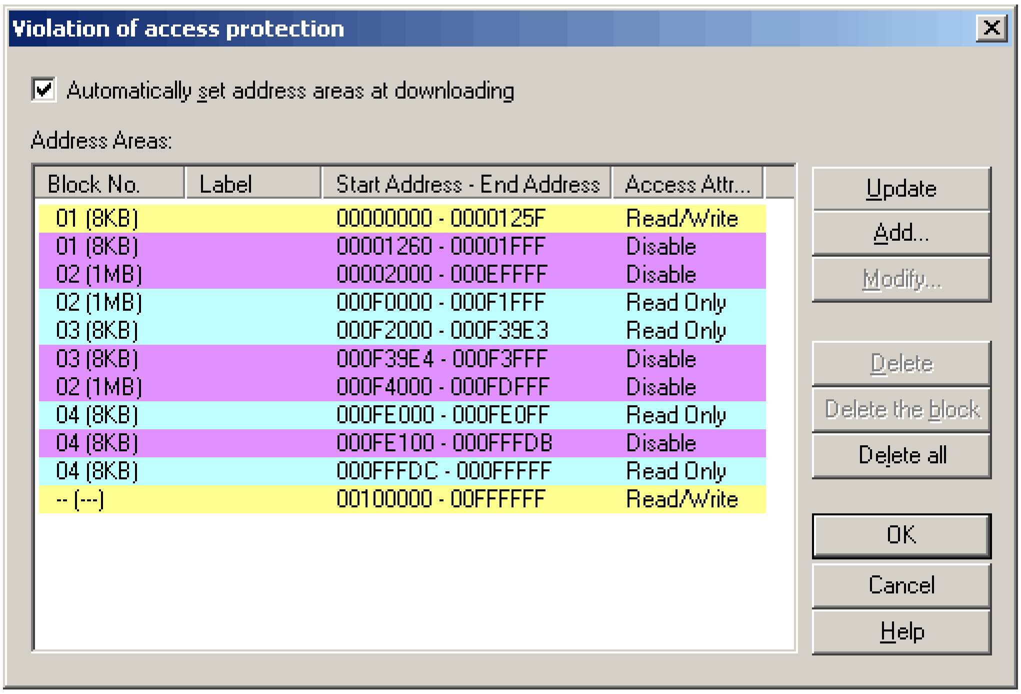


Figure 5.110 Violation of access protection dialog box

4. Click on the Update button, and the access attributes will be updated according to the section information in the downloaded module.

5. To add an access attribute manually, click the Add button. The Access protection condition dialog box shown below will appear. Specify any address range and access attribute.

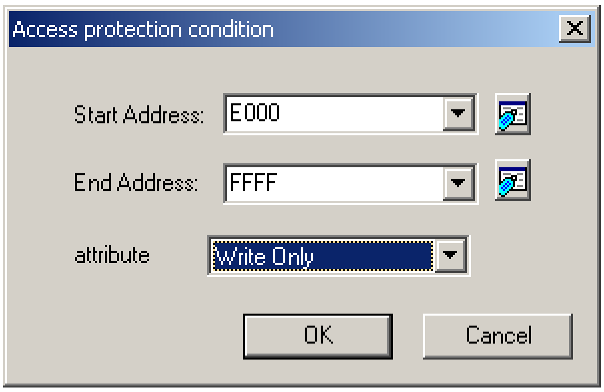


Figure 5.111 Access protection condition dialog box

6. The protected area you have added will be displayed in the Address Areas list of the Violation of access protection dialog box.

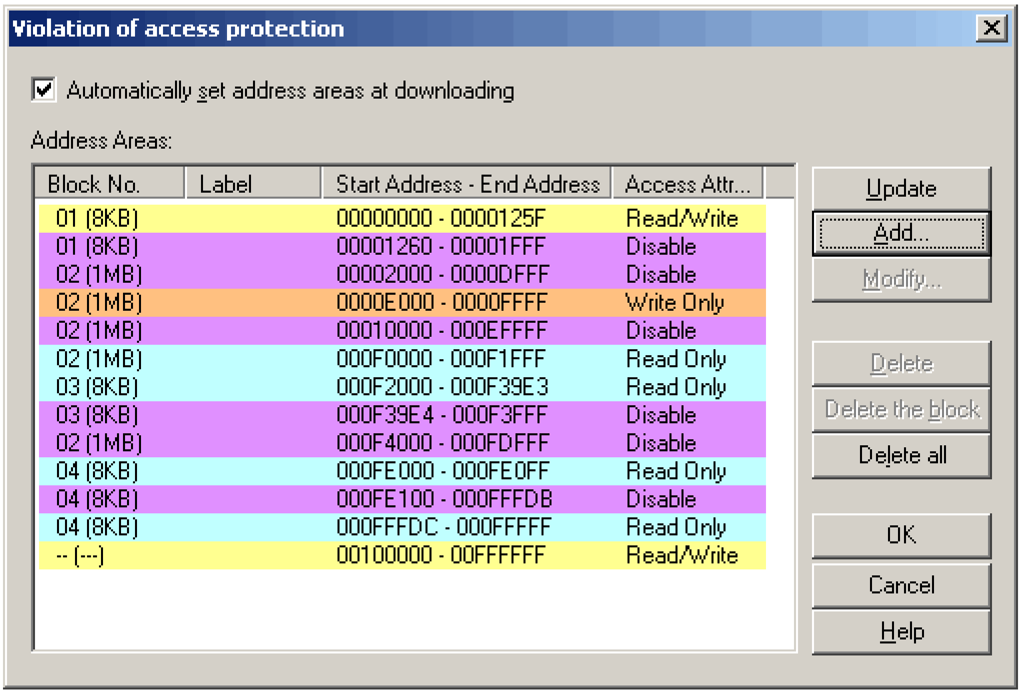


Figure 5.112 Violation of access protection dialog box

(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 on the Detail button.

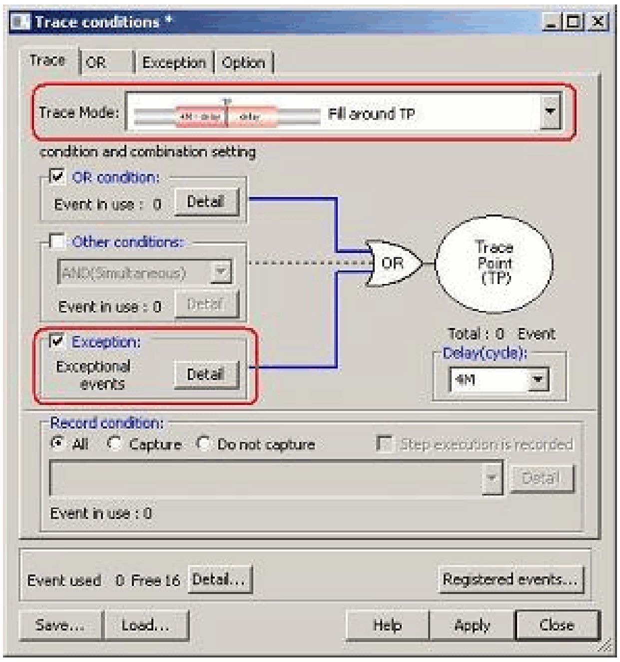


Figure 5.113 Trace conditions dialog box

2. The Exception page shown below will appear. Click on the Detail button to the right of the Violation of access protection checkbox.

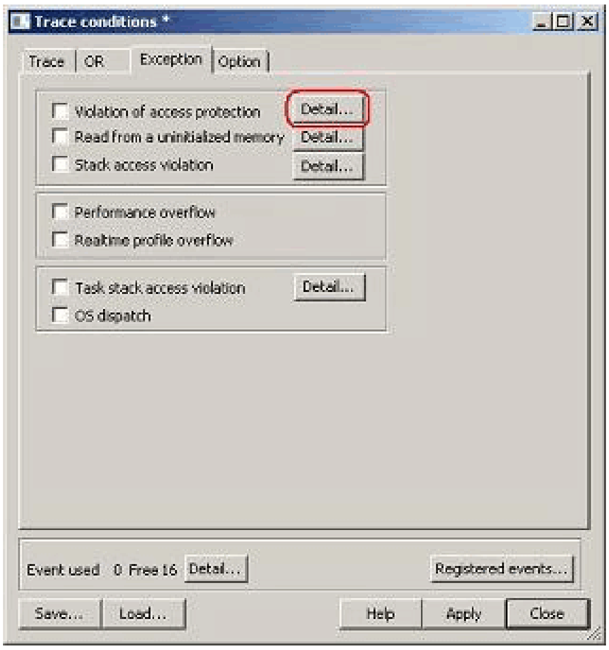


Figure 5.114 Trace conditions dialog box

The Violation of access protection dialog box will be displayed.
The rest of the procedure is the same as if you opened the Violation of access protection dialog box from the Hardware Break dialog box.

5.14.4 Detecting Reading from a Non-initialized Area

Reading from a non-initialized area, i.e. cases of reading from a memory location to which nothing has been written, can be detected as an error.

(1) Method of detection

Reading from a non-initialized area is detected by the RAM monitoring facility.
Allocate a RAM monitoring area to a given address range and enable error detection in that area.

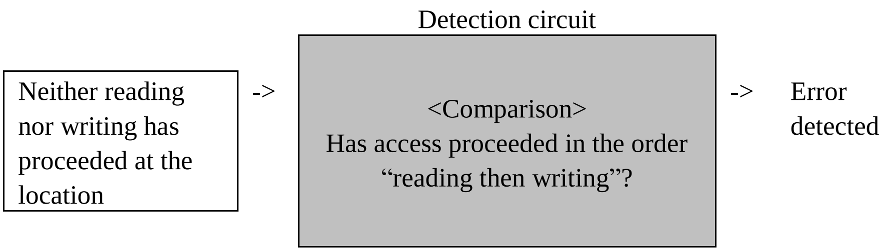


Figure 5.115 Outline of detection of reading from a non-initialized area

(2) Action taken when reading from a non-initialized area is detected

The following actions are selectable.

- Display a warning.

When the Read from uninitialized memory checkbox has been selected on the Exception Warning page of the Configuration properties dialog box, you will see a warning in the Status window and in a status bar balloon when errors of this type occur.

Data is colored in the RAM Monitor window.

- Make the detection of reading from a non-initialized area a condition of a hardware breakpoint.
- Make the detection of reading from a non-initialized area a condition of a trace point.

5.14.5 Detecting Stack Access Violations

Setting the size of the stack too small in software development raises the possibility of a program going out of control or malfunctioning. The E100 emulator actively detects abnormal access by the stack pointer.

(1) Setting stack areas

Selecting a stack section automatically assigns the addresses of the section as a stack area. Alternatively, you can enter any desired address range. Up to 4 stack areas can be specified.

(2) Initial settings when the emulator is booted up

At the time the emulator is booted up, the default section (‘s’) is automatically selected. However, automatic selection does not proceed until a program is downloaded, because there is no address information before this.

(3) Detection method

The emulator monitors the value of ER7 and detects if the value points to a location outside the stack areas.

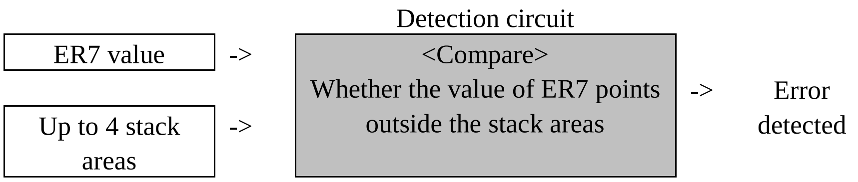


Figure 5.116 Outline of detection of a stack access violation

The emulator will detect the error if the value of the stack pointer is beyond the stack areas on

1. generation of an interrupt or return from an interrupt handler;
2. calling of a function or return from a function; or
3. the stack pointer pointing to a location outside reserved stack areas.

CAUTION

Detection does not cover cases of corruption of data within a stack area.

(4) Actions taken when a stack access violation is detected

The following actions are selectable.

- Display a warning.

When the Stack Access Violation checkbox has been selected on the Exception Warning page of the Configuration properties dialog box, you will see a warning in the Status window and in a status bar balloon when errors of this type occur.

- Make the detection of a stack access violation a condition of a hardware breakpoint.
- Make the detection of a stack access violation a condition of a trace point.

5.14.6 Detecting a Performance-Measurement 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-measurement overflow is detected

The following actions are selectable:

- Display a warning.

A warning is displayed in the Performance Analysis window.
The section of the program where the performance overflow occurred is marked "overflow."
When the Performance Overflow checkbox has been selected on the Exception Warning page of the Configuration properties dialog box, you will see a warning in the Status window and in a status bar balloon when errors of this type occur.

- Make the detection of a performance-measurement overflow a condition of a hardware breakpoint.
- Make the detection of a performance-measurement overflow 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.
Overflows of the counters for time and number of passes for realtime profiling are collectively referred to as realtime profile overflows.

(1) Action taken when a realtime profile overflow is detected

The following actions are selectable.

- Display a warning.

A warning is displayed in the Realtime Profile window.
The line of the function or task in which a timeout or count-out occurred is marked "overflow".
When the Realtime Profile Overflow checkbox has been selected on the Exception Warning page of the Configuration properties dialog box, you will see a warning in the Status window and in a status bar balloon when errors of this type occur.

- Make the detection of a realtime profile overflow a condition of a hardware breakpoint.
- Make the detection of a realtime profile overflow 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) Action taken when a trace memory overflow is detected

The following actions are selectable.

- Display a warning.

When the Trace memory overflow checkbox has been selected on the Exception Warning page of the Configuration properties dialog box, you will see a warning in the Status window and in a status bar balloon when errors of this type occur.

- Make the detection of a trace memory overflow a condition of a hardware breakpoint.

5.14.9 Detecting Task Stack Access Violations

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.

(1) Initial settings when the emulator is booted up

At the time the emulator is booted up, the checkbox labeled "Automatically set address areas at downloading" is selected (flagged by a check mark). However, automatic selection does not proceed until a program is downloaded, because there is no address information before this.

(2) Action taken when a task stack access violation is detected

The following actions are selectable.

- Display a warning.

When the Task stack access violation checkbox has been selected on the Exception Warning page of the Configuration properties dialog box, you will see a warning in the Status window and in a status bar balloon when errors of this type occur.

- Make the detection of a task stack access violation a condition of a hardware breakpoint.
- Make the detection of a task stack access violation a condition of a trace point.

5.14.10 Setting a Task Stack Area

Follow the procedure 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 on the Detail button.

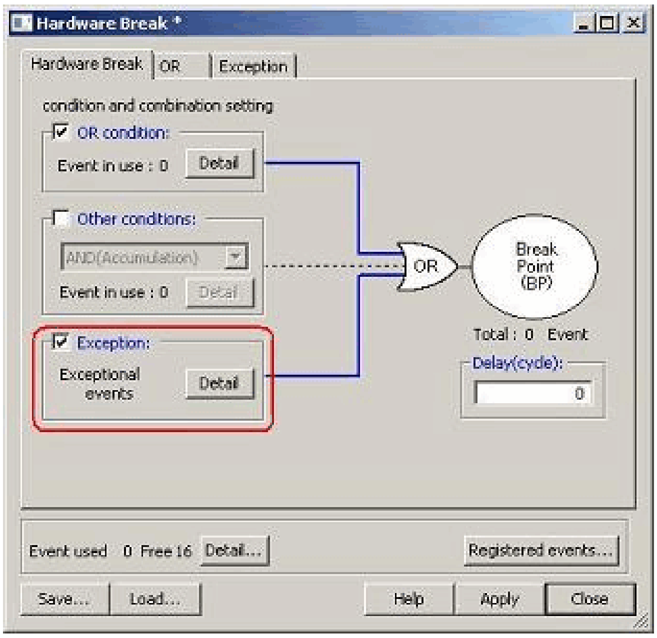


Figure 5.117 Hardware Break dialog box

2. The Exception page shown below will appear. Click on the Detail button to the right of the Task stack access violation checkbox.

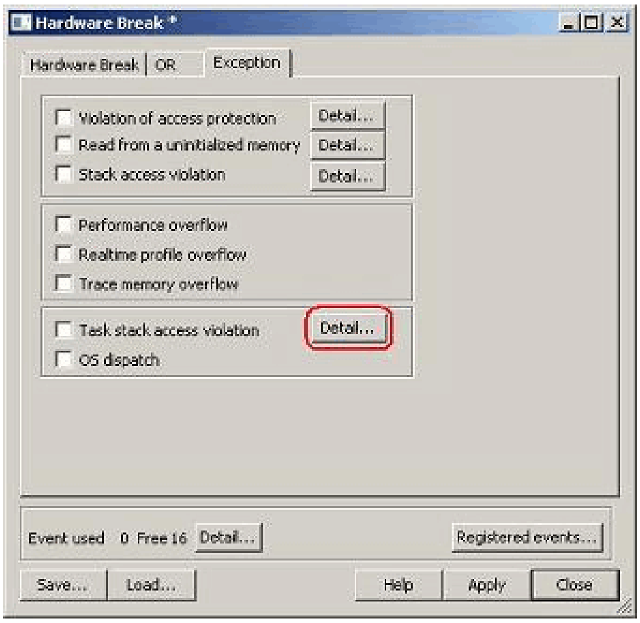


Figure 5.118 Hardware Break dialog box

3. The Violation of task stack access dialog box shown below will be displayed. To have the task stack areas automatically set when a program is downloaded, select the "Automatically set address areas at downloading" checkbox.

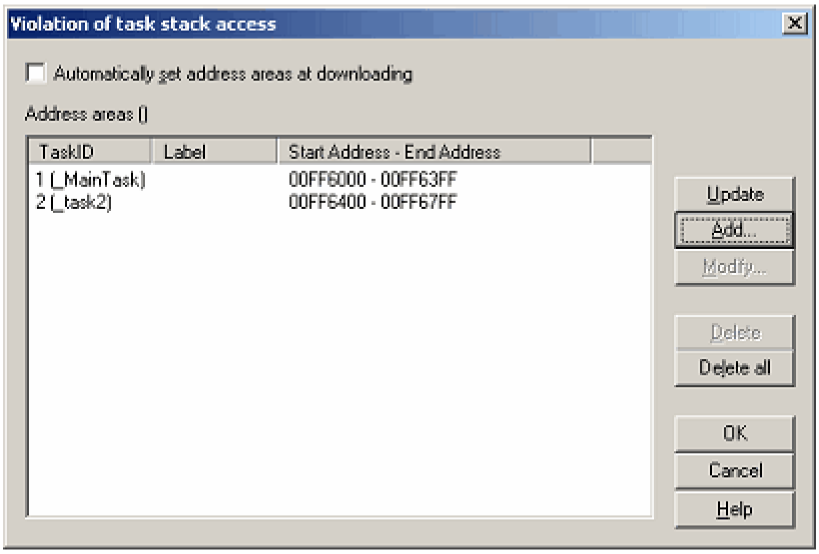


Figure 5.119 Violation of task stack access dialog box

4. Click on the Update button, and the task stack areas will be automatically set.

5. To manually add a task stack area, click on the Add button. The Task stack access condition dialog box shown below will appear. Specify any task ID and the address range of the corresponding task stack.

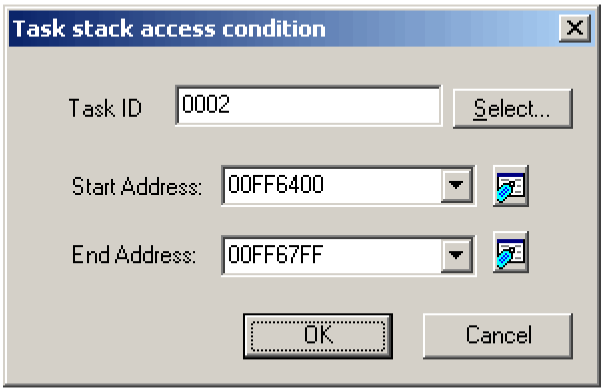


Figure 5.120 Task stack access condition dialog box

6. The task stack area (or areas) 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 on the Detail button.

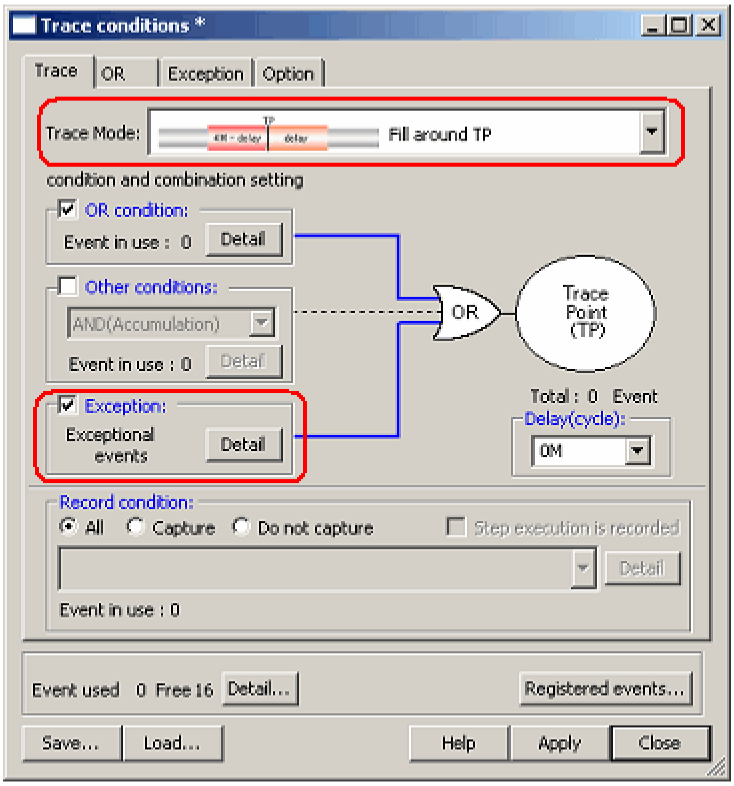


Figure 5.121 Trace conditions dialog box

2. The Exception page shown below will appear. Click on the Detail button to the right of the Task stack access violation checkbox.

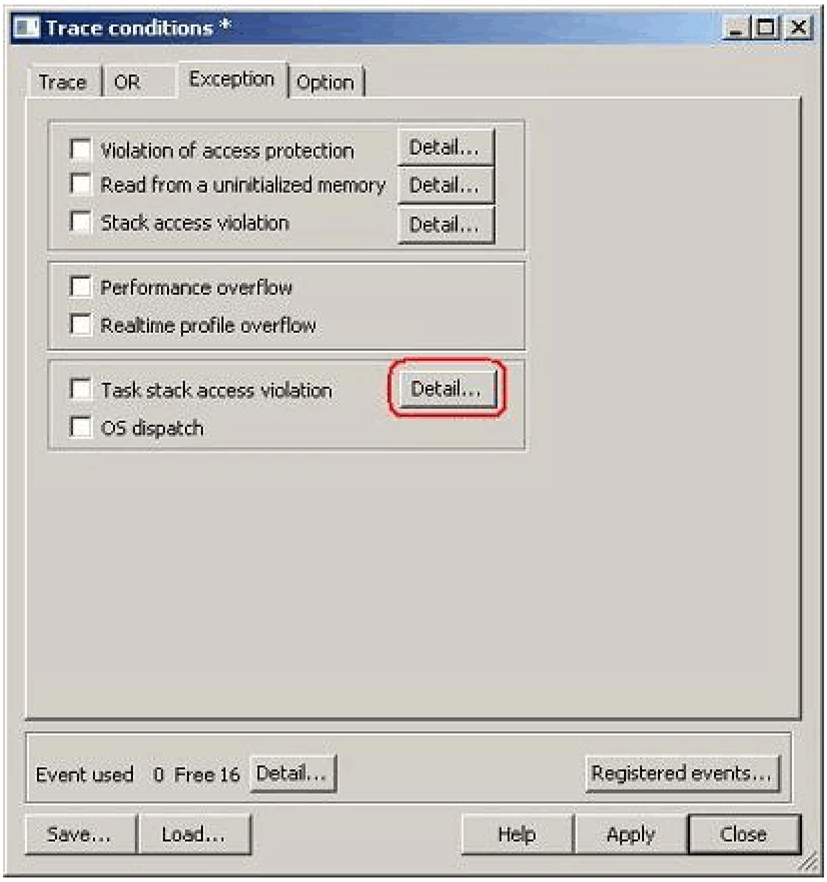


Figure 5.122 Trace conditions dialog box

3. The Violation of task stack access dialog box will be displayed. The rest of the procedure is the same as if you opened the Violation of task stack access dialog box 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.

(1) Action taken when an OS dispatch is detected

The following actions are selectable:

- Display a warning.

When the OS dispatch checkbox has been selected on the Exception Warning page of the Configuration properties dialog box, you will see a warning in the Status window and in a status bar balloon when errors of this type occur.

- Make the detection of an OS dispatch a condition of a hardware breakpoint.
- Make the detection of an OS dispatch a condition of a trace point.

5.15 Using the Start/Stop Function

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

5.15.1 Opening the Start/Stop Function Setting Dialog Box

The routines to be executed immediately before starting and after halting execution of the user program are specified in the [Start/Stop function setting] dialog box.
To open the Start/Stop function setting dialog box, choose Setup -> Emulator -> Start/Stop function setting… from the menu.

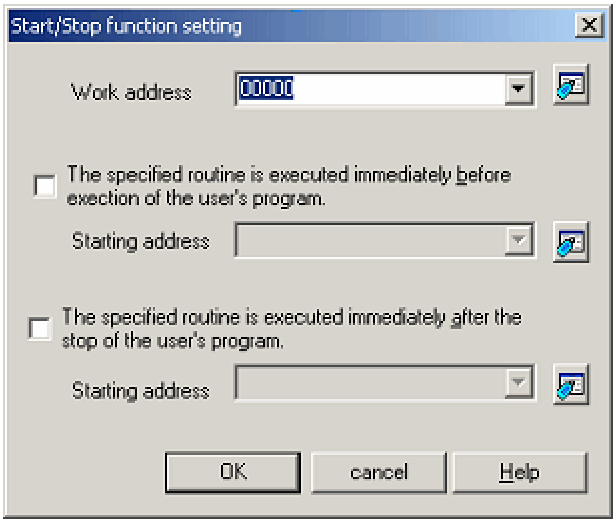


Figure 5.123 Start/Stop function setting dialog box

5.15.2 Specifying the Work address

Use this command to specify the address of a work 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

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

5.15.4 Limitations of the Start/Stop Function

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

- The debugging functions listed below are not to be used while the start/stop function is in use.
  (a) Memory setting and downloading into the program area of a routine specified as a start/stop function.
  (b) Breakpoint setting in the program area of a specified routine
- While either of the specified routines is running, the 4 bytes of memory pointed to by the stack are in use by the emulator.
- The general-purpose registers and flags used in a specified routine are subject to the following limitations.

Table 5.44 Limitations to the registers and flags

Register/flag Name	Limitations
ER7 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.
CCR register, I flag	Interrupts are disabled while a specified routine is executed.

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

Table 5.45 MCU status at start of the user program

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

5.15.5 Limitations on Statements within Specified Routines

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

- If a specified routine uses a stack, the stack must always be the user stack.
- The processing of a specified routine must end with a return-from-subroutine instruction.
- Ensure that a round of processing by a specified routine is complete within 10 ms. If, for example, the clock is turned off and left inactive within a specified routine, the emulator may become unable to control program execution.
- The values stored in the registers at the time a specified routine starts running are undefined. Ensure that each specified routine initializes the register values.

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 5.45 lists the trigger pin numbers and how they operate.

Table 5.46 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'.

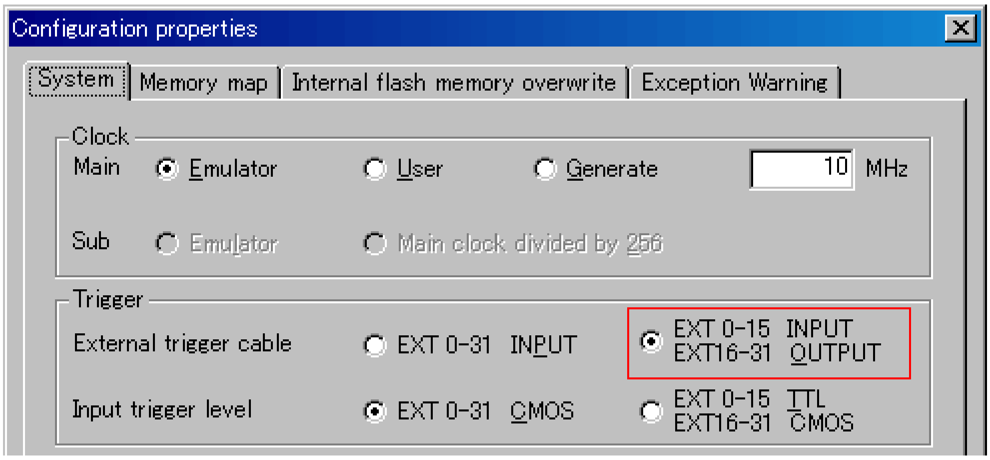
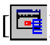


Figure 5.124 Configuration properties dialog box (System page)

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 [].

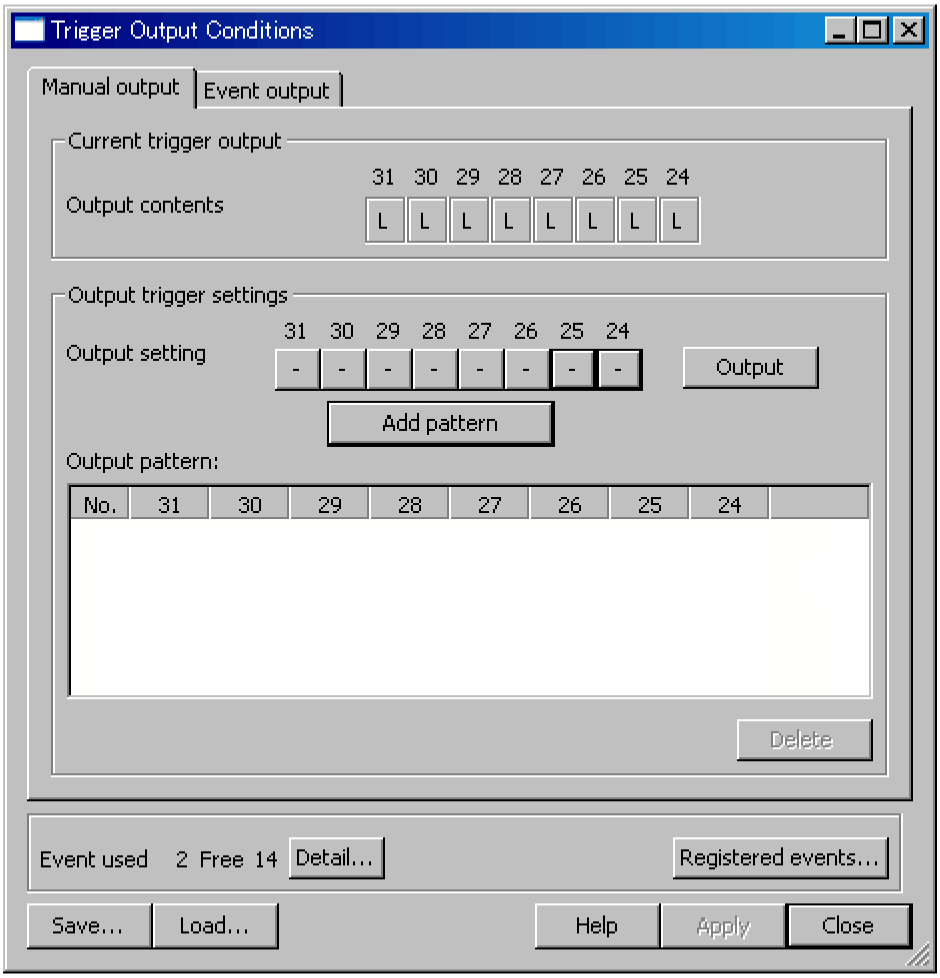


Figure 5.125 Trigger Output Conditions dialog box

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.

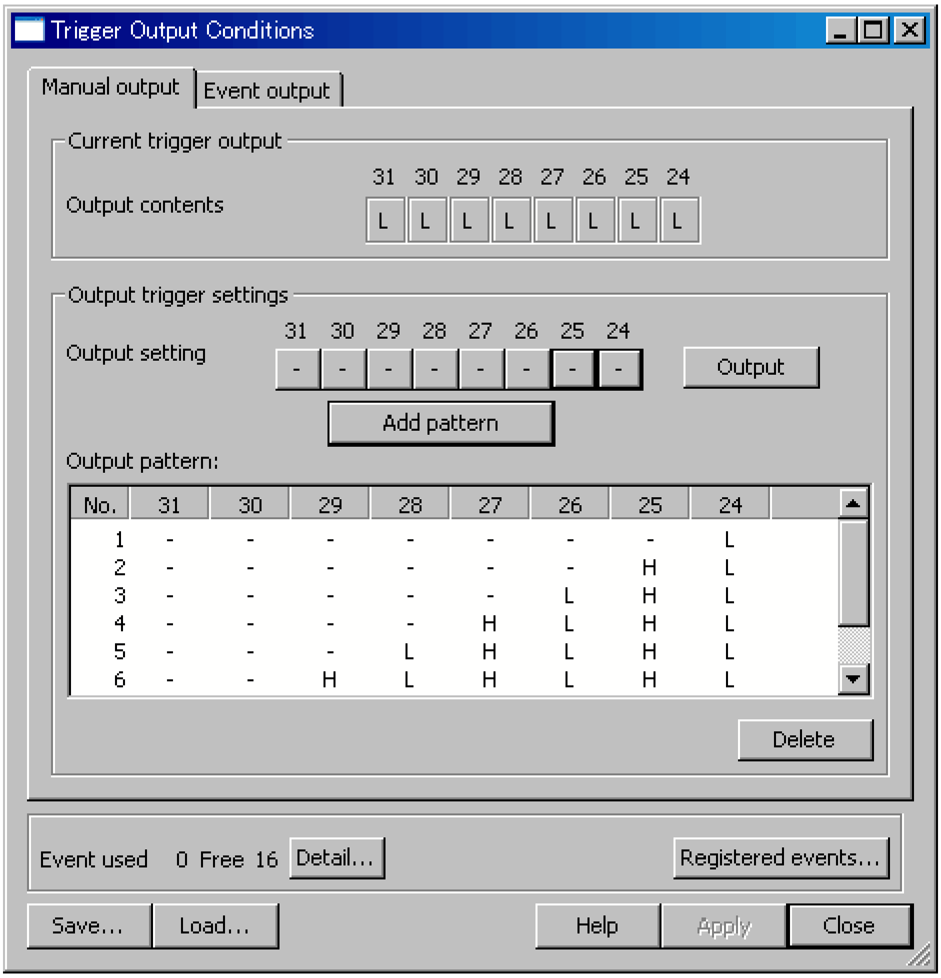


Figure 5.126 Trigger Output Conditions dialog box (Manual output page)

(1) Display of output states: 'Output contents'

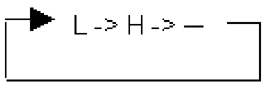
'Output contents' indicates the current signal levels on trigger pins 31 to 24.

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 the 'Output setting'. This simplifies operations.
After making settings for an '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 the '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.

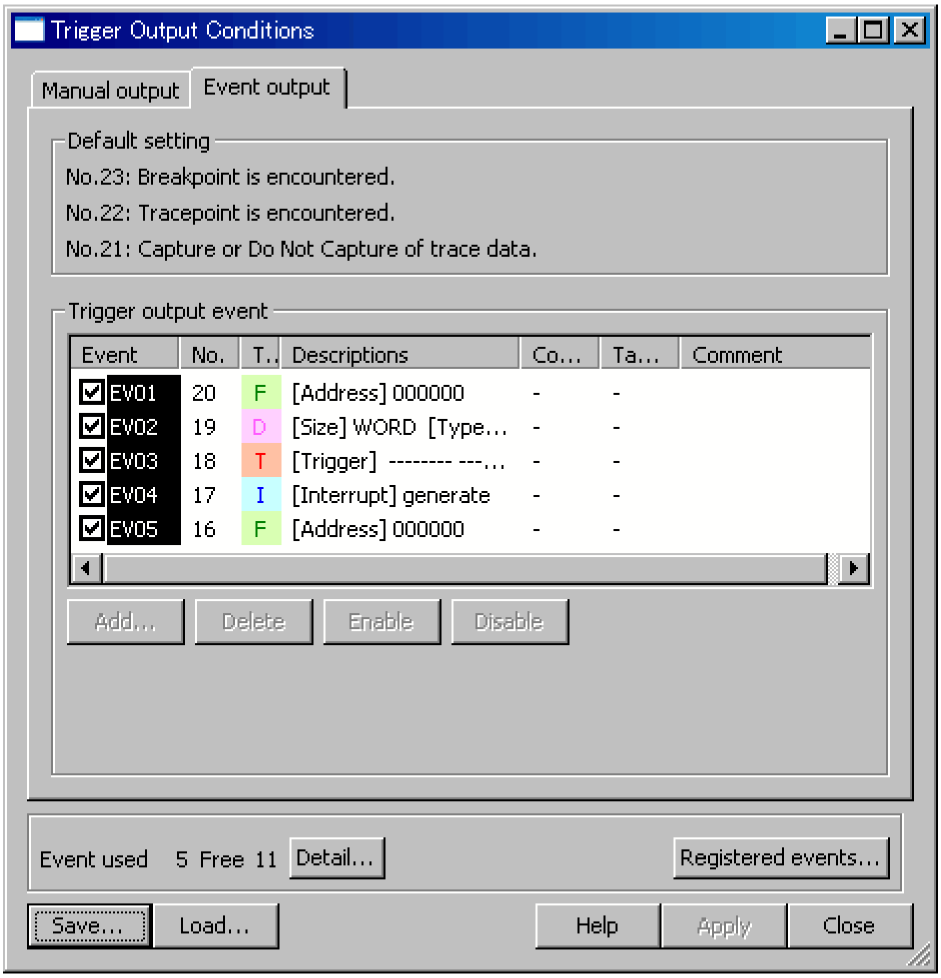


Figure 5.127 Trigger Output Conditions dialog box (Event output page)

(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 5.47 gives details on how the conditions control output.

Table 5.47 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 R0E417250MCU00 is 10 cycles.

5.16.5 Events

For details on the setting of events, see section 5.7, "Using Events" (page 107).

5.17 Measuring the Execution Times in a Specific Section

Measurement of the execution times in a specific section of the program is possible. This facility takes the trace data for instruction fetching at event points (start and end) used to specify the extraction of trace data and then outputs the timestamps and their differences to a file in a format that is editable in Microsoft Excel.
Timestamps for up to 2-M cycles of trace data will be output to the file.
Each section is defined by two events (start and end events) and up to eight sections are specifiable.
Follow the procedure below to measure the execution times in a specific section of the program.

CAUTION
This facility is only supported by command-line operation. To measure the execution times for a specific section of the program, be sure to specify the events on the same line as the command. Such measurement is not possible for events that have been specified in the [Trace conditions] dialog box. Before using this facility, disable all events registered as hardware-break, tracing, and performance-measurement conditions.

5.17.1 Setting Trace Conditions

Trace conditions can be specified on the command line.

(1) Selecting the mode of tracing

Select 'fill until stop', 'fill until full', or 'fill around TP' as the trace mode. Examples of the commands are given below.

```
event_trace_mode fr      (fill until stop)
```

```
event_trace_mode fu      (fill until full)
```

```
event_trace_mode po      (fill around TP)
```

(2) Specifying the start and end events

Specify the start and end events that define the points where the desired section starts and ends, along with the following conditions.

Event type: Instruction fetch
Address condition: Specified value (=)
Do not select any other event type or address condition.

To specify sections from H'1000 to H'10FF and from H'2000 to H'20FF, for example, enter commands as follows.

```
event_set ev1 f address eq 0x001000 cnt 0x1
event_set ev2 f address eq 0x0010FF cnt 0x1
event_set ev3 f address eq 0x002000 cnt 0x1
event_set ev4 f address eq 0x0020FF cnt 0x1
```

(3) Selecting an option for 'Record condition'

You should specify extraction of trace data during the event. No other conditions should be selected.

An example of a command that specifies 'Extraction' and 'Duration of an event' for the events set in step (2) is given below.

```
event_trace_acquisition apo ev1 ev2 ev3 ev4
```

(4) Selecting an option for tracing

Select 'Event number' as an option for tracing. Do not select any other options. An example is given below.

```
event_trace_option ev
```

5.17.2 Acquiring Trace Data

Run the user program and acquire the trace data.

5.17.3 Specifying a Section

You can use the TRACE_EXECUTE_SECTION_SET command to specify a section. An example of commands to specify section 1 that starts with event 1 and ends with event 2 and section 2 that starts with event 3 and ends with event 4 is given below.

```
trace_execute_section_set 1 start ev1 end ev2
trace_execute_section_set 2 start ev3 end ev4
```

5.17.4 Saving the Execution Times to a File

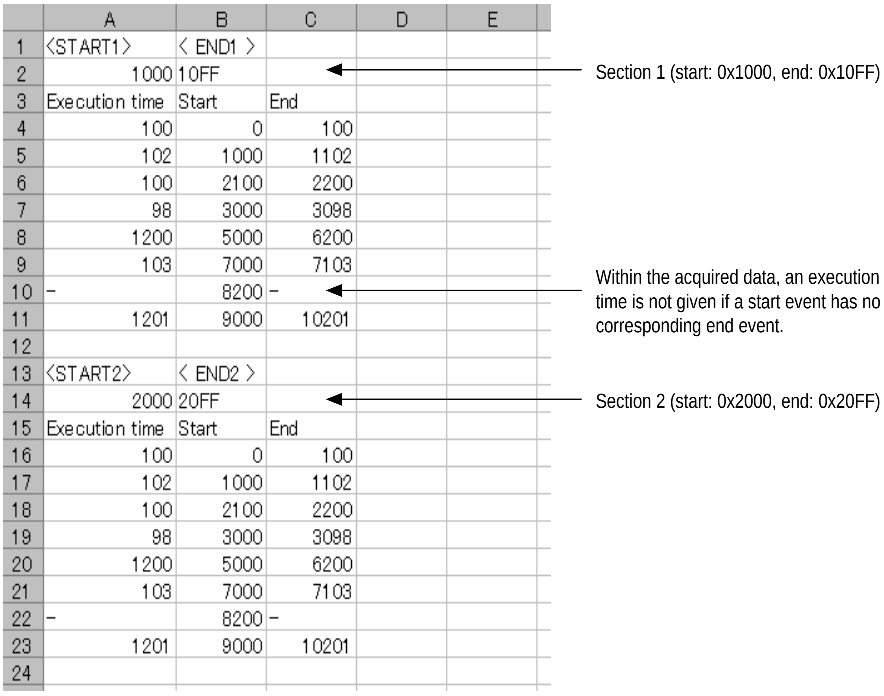
After the trace data has been acquired, you can use the TRACE_EXECUTE_SAVE command to save the execution times in the specified section to a file with extension .csv.

The command description given below is an example where the execution times for sections 1 and 2 are saved in result.csv under the configuration directory.

```
trace_execute_save    $(CONFIGDIR)\\result.csv    1    2
```

The contents of the .csv file opened in Microsoft Excel will be as follows.

Example:



	A	B	C	D	E
1	<START1>	< END1 >			
2	1000	10FF			
3	Execution time	Start	End		
4	100	0	100		
5	102	1000	1102		
6	100	2100	2200		
7	98	3000	3098		
8	1200	5000	6200		
9	103	7000	7103		
10	-	8200	-		
11	1201	9000	10201		
12					
13	<START2>	< END2 >			
14	2000	20FF			
15	Execution time	Start	End		
16	100	0	100		
17	102	1000	1102		
18	100	2100	2200		
19	98	3000	3098		
20	1200	5000	6200		
21	103	7000	7103		
22	-	8200	-		
23	1201	9000	10201		
24					

Figure 5.128 .csv file opened in Microsoft Excel

The execution time, in nanoseconds, is saved in the file.

Example:

1h 23 m 45 s 678 ms 901 μs 234 ns = 01:23:45.678.901.234 -> 5025678901234

[CAUTION]

Measurement of execution times in specific sections is not possible for events set in the Trace conditions dialog box, so be sure to specify the events on the command line.

Execute the TRACE_WAIT command before using the TRACE_EXECUTE_SAVE command in a command file. The TRACE_WAIT command makes the emulator wait for successful acquisition of the trace data.

For details on commands, refer to the online help information.

5.18 Generating Pseudo ECC Errors

A command is available for the generation of pseudo ECC errors in ROM or RAM. For the command-line format, see the online help system.

CAUTION
Generation of pseudo ECC errors for ROM is not possible if emulation memory has been assigned to the corresponding area of internal ROM.

5.19 Generating a State Where External Oscillation Has Stopped

A command is available for generating a state where it appears as if external oscillation has stopped. For the command-line format, see the online help system.

5.20 Blank Checking for the Internal EEPROM

A command is available for checking if the internal EEPROM is blank in 8-byte units. For the command-line format, see the online help system.

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.

http://www.renesas.com/tools

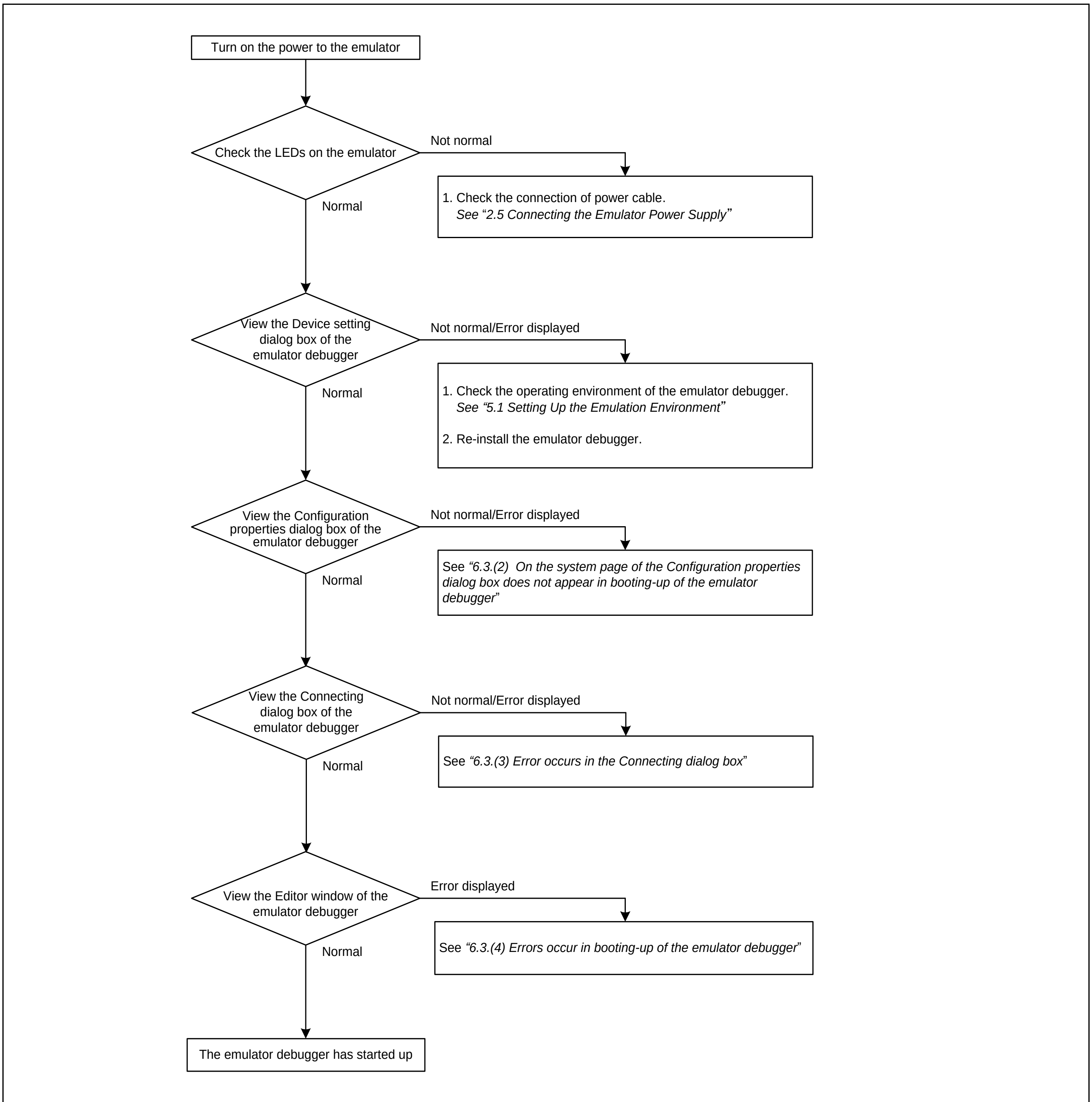


Figure 6.1 Flowchart for remediation of trouble

6.2 Error in Self-checking

When an error occurs in the 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).

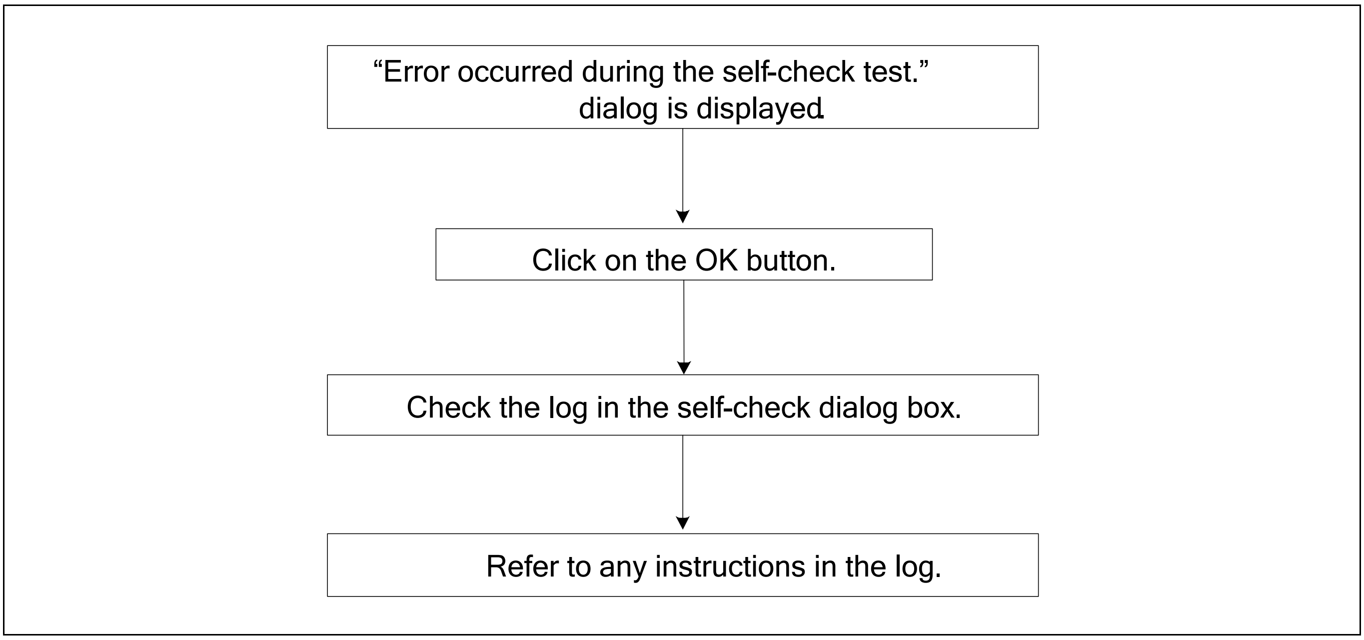


Figure 6.2 Flowchart for checking of an error in self-checking

IMPORTANT
Notes on 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 power cable is connected. *See "2.4 Connecting the Host Machine" (page 27).*
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 26).*
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 a 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. *See "4. Preparation for Debug" (page 67).*
A communication error.	

(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 board 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 26).*
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 the NQPACK etc. mounted on the user system is soldered properly. (2) Check that the connector is installed properly to the user system.

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.

http://tool-support.renesas.com/eng/toolnews/registration/support.txt

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 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 the R0E417250MCU00

Item	Description
Applicable MCU series	H8SX family H8SX/1700 series
Evaluation MCU	R5E61700
Applicable MCU mode	Single-chip mode, On-chip ROM enabled extended mode
Supported MCU Series	H8SX/1700 series (with 1-Mbytes ROM, 64-Kbyte RAM, and 32-Kbyte data-flash ROM)
Power supply voltage	Vcc: 3.0 to 3.6V, 4.5 to 5.5V
Maximum operating frequency	80 MHz

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) Initial values of registers

State	Register name	Emulator	MCU
Power-on/initialized	PC	Reset vector value	Reset vector value
	ER0 to ER6	Undefined	Undefined
	ER7 (SP)	H'10	Undefined
	CCR	The I mask bit is 1 and the other bits are undefined.	The I mask bit is 1 and the other bits are undefined.
Reset	PC	Reset vector value	Reset vector value
	ER0 to ER6	Undefined	Undefined
	ER7 (SP)	H'10	Undefined
	CCR	The I mask bit is 1 and the other bits are undefined.	The I mask bit is 1 and the other bits are undefined.

(2) 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.

(3) A/D converter

The characteristics of the A/D converter differ from those of the 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).

Notes on Maskable Interrupts:

- Even if a user program is not being executed (including when run-time debugging is being performed), 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 (including when run-time debugging is being performed), the maskable interrupt request cannot be accepted, because the emulator disables interrupts. The interrupt request is accepted immediately after the user program execution is started.
- Take note that when the user program is not being executed (including when run-time debugging is being performed), a peripheral I/O interrupt request is not accepted.

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 the R0E417250MCU00

Figure 7.1 shows a partial circuit diagram of the connections of the R0E417250MCU00. 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. See this diagram for reference when you use the MCU unit.

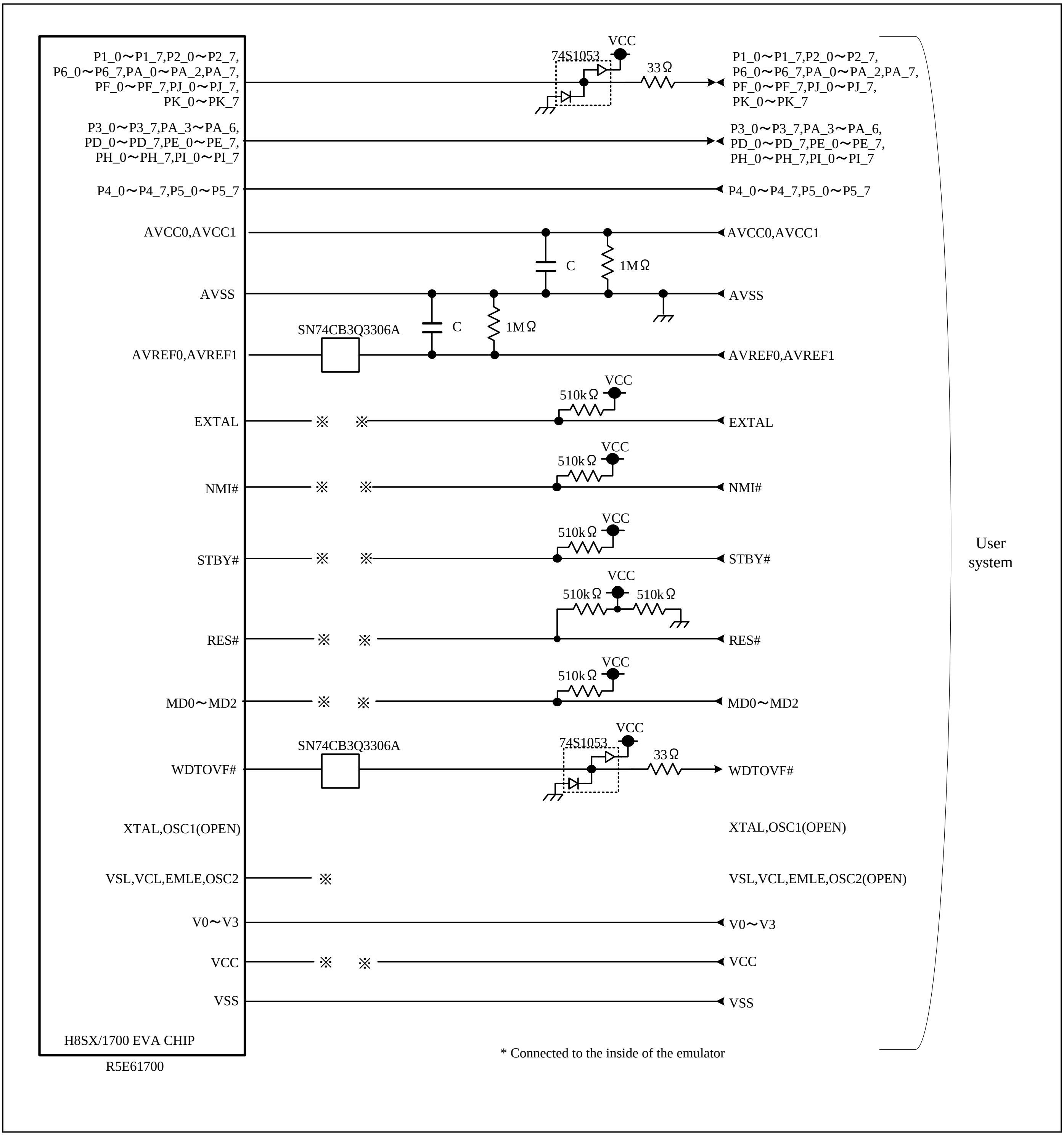


Figure 7.1 Connection diagram for R0E417250MCU00

7.4 External Dimensions

7.4.1 External Dimensions of the E100 Emulator

Figure 7.2 shows external dimensions of the E100 emulator.

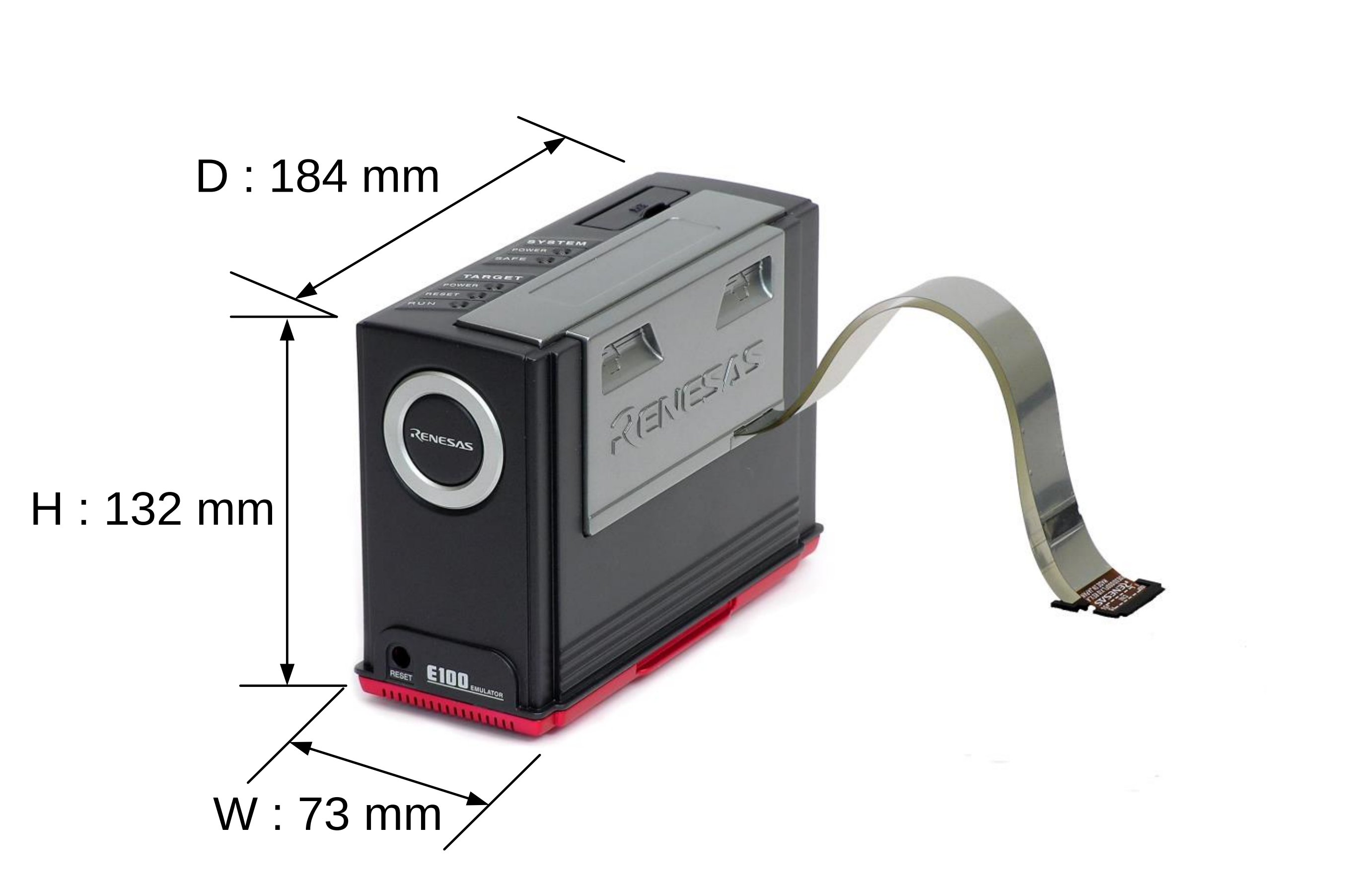


Figure 7.2 External dimensions of the E100 emulator

7.4.2 External Dimensions of the Converter Board R0E0100TNPFK00

Figure 7.3 shows external dimensions and a sample pad pattern of the converter board R0E0100TNPFK00 for a 100-pin 0.5-mm pitch LQFP.

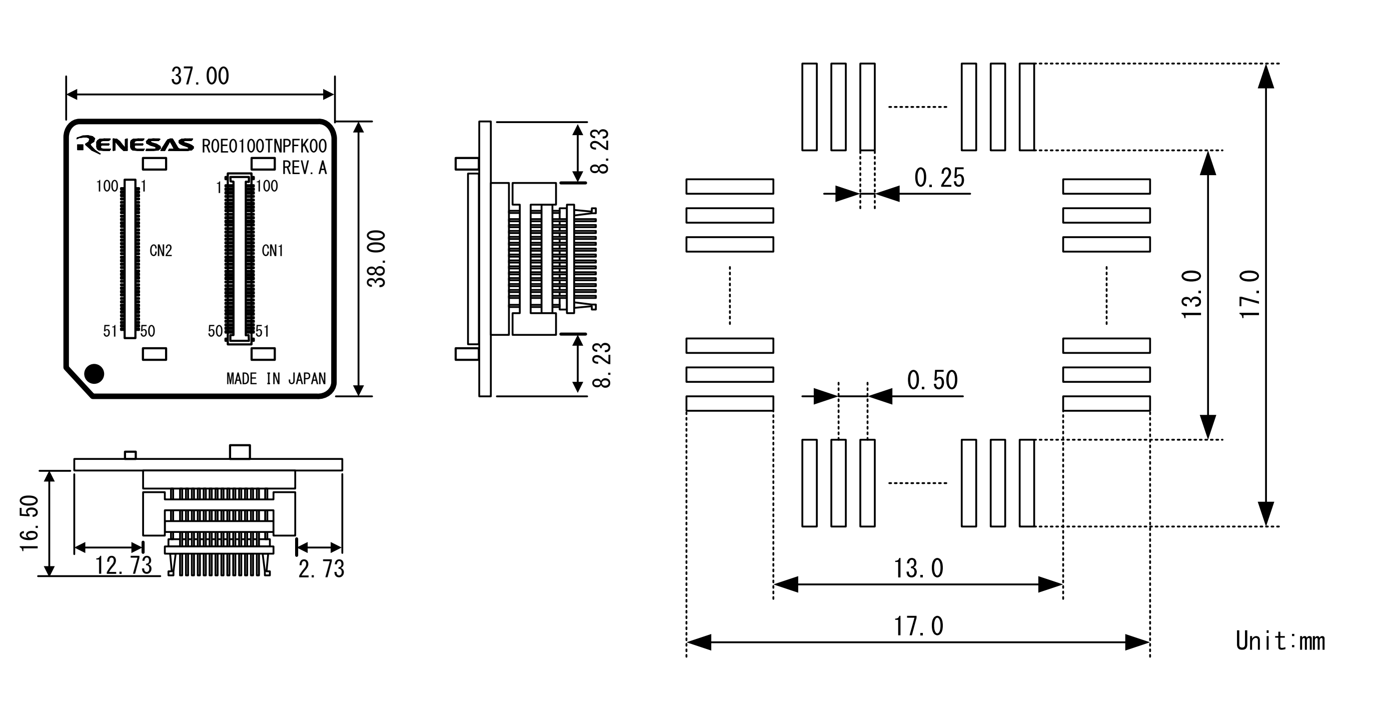


Figure 7.3 External dimensions and a sample pad pattern of the R0E0100TNPFK00

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 the MCU unit with the following emulator debugger. - H8SX E100 Emulator Software V.1.00 release 01 or later
Notes on Downloading Firmware: ● Before using the MCU unit 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.

IMPORTANT

Note on Clock Supply to the MCU:

- A clock source supplied to the evaluation MCU is selected on the System page of the Configuration properties dialog box of the emulator debugger.
  (1) When "Emulator" is selected:
  A clock source 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 source generated by the oscillator in the user system is supplied. It depends on the status of the oscillation (on/off) on the user system.
  (3) When "Generate" is selected:
  A clock source 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 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. Write signals to the areas will be ignored, and values read will be undefined.

Note on Breaks:

- The following break functions are available in the emulator debugger.
  (1) Software break
  This is a debugging function which generates a BRK interrupt by changing an instruction at a specified address to a BRK instruction (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) 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.
  (3) Exceptional event
  This is a debugging function which stops a program when an abnormal operation of the user program or overflow of each function's measurement counter, etc. is detected.

Note on Software Breaks:

- After you have selected the "Debug the program with overwriting of flash ROM" checkbox on the [System] page of the [Configuration properties] dialog box, the setting of software breakpoints in internal ROM becomes impossible. If you remove the tick from the "Debug the program with overwriting of flash ROM" checkbox, setting such software breakpoints becomes possible.

IMPORTANT

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 V ≤ Vcc ≤3.6 V,4.5 V ≤ Vcc ≤ 5.5 V

Note on Flash ROM:

- Since the number of times the flash ROM can be programmed or erased is limited, it must be replaced at the end of its service life.
  If any of the following error messages frequently appears while you are downloading a program, replace the MCU board.
  - An error has occurred in erasing of the flash ROM in the target MCU. The flash ROM may have been degraded.
  - An error has occurred in verification of the flash ROM in the target MCU. The flash ROM may have been degraded.

Note on Debugging with Overwriting of Flash Memory:

- When the emulator reads or writes to the data flash ROM or flash ROM, software in the emulator may alter the values of registers in the FCU. To check if the emulator has actually manipulated the registers in the FCU, select the [Display a message when the emulator was to manipulated registers in the FCU] checkbox on the [System] page of the [Configuration properties] dialog box. If the message “The registers of FCU was manipulated by emulator.” is displayed, debugging with flash-ROM rewriting cannot proceed: initialize the user program before running the program again.

Notes on Manipulation of Memory and Reading of Data from the Flash ROM and EEPROM:

- When downloading of data or programming of the flash ROM or EEPROM is attempted, the emulator writes data to the memory in block units. Even if only a single byte of data in the EEPROM has been modified, all memory resources in the selected block will be rewritten so the block will no longer be blank.
- When reading of data from the flash ROM or EEPROM is attempted, the emulator enters the break state, and reading of data from the actual flash memory only proceeds once. For this reason, you will keep seeing the same value in the emulator even after undefined data have been read out several times. When the actual MCU is in use, on the other hand, the values change per reading operation.
- In cases where a clock source on the user system is selected, no external clock is available, and an internal oscillator is in use, manipulating data in the flash ROM or EEPROM through windows (e.g. [Memory] window) leads to the message “The command is not executable because the FCU operating clock is out of operating range.” Before manipulating memory, the clock frequency must be changed to one that allows programming of the flash ROM or EEPROM.

Notes on the Emulator Limits Access:

- The emulator limits access to the following areas.
- Areas
  - Addresses H'0 to H'FFFFF excluding the flash ROM area.
  - Addresses H'FEC000 to H'FEBFFF excluding the RAM area.
  - Addresses H'E00000 to H'E1FFFF excluding the EEPROM area.
  - Access-prohibited areas.
- Actions
  - When data is viewed in a window (e.g. [Memory]), fixed data 00 is read out.
  - When data is changed in a window (e.g. [Memory]), the data is not written. No errors occur.
  - When data other than 00 is written with verification, a verification error occurs.

Notes on Memory Access during Execution of the User Program:

- While the user program is running, using the emulator to download programs to the flash ROM or EEPROM or to change memory settings is not possible.
- If you have selected the [Debug the program with overwriting of flash ROM] checkbox on the [System] page of the [Configuration properties] dialog box and attempt to read data from the flash ROM or EEPROM while the user program is running, the data read out during the most recent break is displayed rather than the current data in the flash memory.
- If you attempt to read data from the flash ROM or EEPROM while the user program is running under any of the following conditions, the fixed value "FF" is read out.
  - The operating frequency (CPU clock frequency) is below 1 MHz (e.g. while the subclock is in use).
  - The CPU clock has stopped (e.g. the emulator is in the standby or subsleep mode).
  - The emulator is in the reset state.
  - The emulator is waiting for release of the external bus.
- When memory is accessed during execution of the user program, the MCU temporarily occupies the MCU bus. If you wish to maintain realtime operation, select the [Disable memory access by GUI when target is executing] checkbox on the [Options] page of the [Debug Settings] dialog box.

Note on Key-code Error That May Appear during the Boot-up Process:

- An error message "Failed to boot up the EFW program because the keycode value is invalid." may appear during the boot-up process and prevent connection of the emulator. In such a case, perform a diagnostic test.

Note on Key Code

- The range of addresses H'4 to H'7 is a key-code area. Note that there are the following limitations.
  (1) When emulation memory is not assigned to the flash ROM
  Even if you attempt to change values of addresses H'4 to H'7 in a window (e.g. [Memory]), the values will not be actually written.
  (2) When emulation memory is assigned to the flash ROM
  If you attempt to change values of addresses H'4 to H'7 in a window (e.g. [Memory]), the values will be written.

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

(7) (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.

(8) (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.

http://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

Rev.	Date	Description	
		Page	Summary
3.01	Dec 1, 2015	3	Regulatory Compliance Notices changed

E100 Emulator MCU Unit for H8SX/1700 Series
User's Manual
R0E417250MCU00

Publication Date: Dec 01, 2015 Rev.3.01

Published by: Renesas Electronics Corporation

SALES OFFICES

Renesas Electronics Corporation

http://www.renesas.com

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

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

Renesas Electronics Canada Limited
9251 Yonge Street, Suite 8309 Richmond Hill, Ontario Canada L4C 9T3
Tel: +1-905-237-2004

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

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

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

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

Renesas Electronics Hong Kong Limited
Unit 1601-1611, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022

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

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

Renesas Electronics Malaysia Sdn.Bhd.
Unit 1207, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics India Pvt. Ltd.
No.777C, 100 Feet Road, HALII Stage, Indiranagar, Bangalore, India
Tel: +91-80-67208700, Fax: +91-80-67208777

Renesas Electronics Korea Co., Ltd.
12F., 234 Teheran-ro, Gangnam-Gu, Seoul, 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 4.0

R0E417250MCU00
User's Manual

Renesas Electronics Corporation

R20UT3587EJ0301