User's Manual

E1/E20 Emulator

Additional Document for User's Manual (Notes on Connection)

Supported Devices:
R8C Family / R8C/3x Series
R8C Family / R8C/Lx Series

Notes on R5F213MCQ have been added in the following pages of this document.
Page 42 "4.13.1 [System] tab"
Page 112 "Table 7.10 Program Area for the Emulator (R8C/3MQ)"

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.2.01 Oct 2011

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. 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.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. 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.
5. When exporting the 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. You should not use Renesas Electronics products or the 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. 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.
6. 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.
7. Renesas Electronics products are classified according to the following three quality grades: “Standard”, “High Quality”, and “Specific”. The recommended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below. 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 categorized as “Specific” without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. 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 an application categorized as “Specific” or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is “Standard” unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

   “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.

   “High Quality”: Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.

   “Specific”: Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. 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.
9. 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 system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. 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.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

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

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

Contents

Page
1. Inside the E1/E20 Emulator User's Manual 6
2. E1/E20 Emulator Specifications 7
2.1 Target MCUs 7
2.2 Emulator specifications 7
2.3 Applicable tool chain and third-party products 10
3. Designing the User System 12
3.1 Connector for connecting the E1 or E20 emulator and the user system 12
3.2 Small connector conversion adapter 15
3.3 Connecting system ground 16
3.4 Recommended connection example between the E1/E20 connecting connector and the MCU 17
3.5 Interface circuit in the E1 or E20 emulator 19
4. Preparations for Debugging 20
4.1 Activating High-performance Embedded Workshop 20
4.2 Creating a new workspace (with a toolchain not in use) 21
4.3 Creating a new workspace (with a toolchain in use) 23
4.4 Opening an existing workspace 25
4.5 Connecting the emulator 26
4.5.1 Connecting the emulator 26
4.5.2 Reconnecting the emulator 26
4.6 Disconnecting the emulator 26
4.6.1 Disconnecting the emulator 26
4.7 Quitting the High-performance Embedded Workshop 26
4.8 Making debugging-related settings 27
4.8.1 Specifying a module for downloading 27
4.8.2 Setting up automatic execution of command line batch files 28
4.9 Procedure for launching the E1/E20 emulator debugger 29
4.10 Setting the emulator debugger during startup 31
4.11 [Initial Settings] dialog box 32
4.11.1 [Device] tab 33
4.11.2 [Communication] tab 35
4.12 Showing the connection dialogs 36
4.12.1 When the E1/E20 emulator is used in other emulator debuggers such as RX Family 37
4.12.2 When the E1/E20 emulator firmware version is old 38
4.12.3 When the [Power supply] setting in the [Initial Settings] dialog box is incorrect 39
4.12.4 When an ID code other than "FFFFFFFFFFFFFFh" is written in the ID code area 40
4.12.5 When the emulator cannot communicate with the MCU 41
4.13 [Configuration Properties] dialog box 42
4.13.1 [System] tab 42
4.13.2 [MCU] tab 43
4.13.3 [Internal flash memory overwrite] tab 44
4.14 Starting up the R8C E1/E20 Emulator Debugger 45
5. Debugging Functions 46
5.1 List of debugging functions 46
5.2 Downloading a program 48
5.3 Opening a source file 49
5.3.1 Viewing the source code 49
5.3.2 Switching off a column in all source files 50
5.3.3 Switching off a column in one source file 50
5.3.4 Viewing the assembly-language code 51
5.3.5 Modifying the assembly-language code 51
5.4 Memory access functions 52
5.4.1 Memory read/write function 52
5.4.2 Other memory operation functions 52
5.4.3 Notes on accessing the SFR areas 53
5.5 Outline of break functions 55
5.5.1 Forced break 55
5.5.2 S/W break (software break) 55
5.5.3 On-chip break 55

5.6 Using S/W breakpoints 56
5.6.1 Adding/removing S/W breakpoints 57
5.6.2 Enabling/disabling S/W breakpoints 58
5.7 Outline of on-chip break functions 59
5.7.1 [On-Chip Event] dialog box 59
5.7.2 [Event] tab 60
5.7.3 [Before PC Break] tab 62
5.7.4 [Event] dialog box 63
5.7.5 [Event] dialog box (event type: data access) 64
5.7.6 Notes on the event settings 65
5.8 Adding on-chip events (DA events) 66
5.8.1 Adding or changing from the [On-Chip Event] dialog box 66
5.8.2 Dragging and dropping from other windows (addition only) 67
5.8.3 Deleting, enabling, or disabling from the [On-Chip Event] dialog box 68
5.8.4 Command line 68
5.9 Adding on-chip events (PC event) 69
5.9.1 Adding from the [On-Chip Event] dialog box 69
5.9.2 [On-chip Break points] column of the [Editor] window 70
5.9.3 Dragging and dropping from other windows (addition only) 71
5.9.4 Deleting, enabling, or disabling from the [On-Chip Event] dialog box 72
5.9.5 Command Line 72
5.10 Registering events 73
5.10.1 [Registered Events] dialog box 73
5.10.2 Registering events 74
5.10.3 Creating events for each instance of usage or reusing events 76
5.10.4 Activating events 76
5.11 Saving/loading the set contents of on-chip events 77
5.11.1 Saving [On-Chip Event] Settings 77
5.11.2 Loading the set contents of [On-Chip Event] settings 77
5.12 Saving/loading the set contents of the [Registered Events] dialog box 77
5.12.1 Saving [Registered Events] dialog box settings 77
5.12.2 Loading the set contents of [Registered Events] dialog box settings 77
5.13 Trace function 78
5.13.1 Outline of trace function 78
5.13.2 Trace setting items 79
5.13.3 Trace menu 80
5.13.4 Branch trace 80
5.13.5 Statistics 81
5.13.6 Saving trace information in files 81
5.14 Status bar 82
5.15 Start/Stop function 83
5.15.1 Opening the [Start/Stop function setting] dialog box 83
5.15.2 Specifying the routine to be executed 83
5.15.3 Restrictions on the Start/Stop function 84
5.15.4 Limitations on statements within specified routines 84
5.16 Simple stack overflow detection function 85
5.17 Online help 85
6. Tutorial 86
6.1 Introduction 86
6.2 Starting the High-performance Embedded Workshop 86
6.3 Connecting the emulator 86
6.4 Downloading the tutorial program 87
6.4.1 Downloading the tutorial program 87
6.4.2 Displaying the source program 88
6.5 Setting S/W breakpoints 89
6.6 Executing the program 90
6.6.1 Resetting the CPU 90
6.6.2 Executing the program 90
6.7 Checking breakpoints 91
6.7.1 Checking breakpoints 91
6.8 Altering register contents 92

6.9 Referring to symbols 93
6.10 Checking memory contents 94
6.11 Referring to variables 95
6.12 Showing local variables 97
6.13 Stepping through a program 98
6.13.1 Executing [Step In] 98
6.13.2 Executing [Step Out] 99
6.13.3 Executing [Step Over] 99
6.14 Forcibly breaking program execution 100
6.15 On-chip break facility 101
6.15.1 Stopping a program when it executes the instruction at a specified address (pre-PC break) 101
6.15.2 Stopping a program when It accesses memory 102
6.16 Tracing facility 103
6.16.1 Showing the information acquired in tracing 104
6.17 Stack trace facility 106
6.18 What next? 107
7. Notes on Using the E1 or E20 Emulator 108
7.1 MCU resources used by the emulator 108
7.1.1 Program area for the emulator 108
7.1.2 Pins used by the E1 or E20 emulator 115
7.1.3 Interrupts (unusable) 115
7.1.4 Stack area used by the E1 or E20 emulator 115
7.1.5 SFRs used by the E1/E20 emulator program 115
7.1.6 Option function select area 116
7.1.7 Registers initialized by the E1 or E20 emulator 117
7.1.8 RAM initialization 117
7.1.9 MCU reserved area 117
7.1.10 DTC during a user program halt (not applicable for the R8C/LAxA) 117
7.1.11 Note on internal power low consumption 117
7.1.12 Note on debugging at less than 2.7V (not applicable for the R8C/LAxA) 117
7.1.13 Debugging the functions to reduce power consumption 117
7.2 Reset 118
7.3 Internal ROM area (flash memory) 120
7.3.1 Changing the internal ROM area 120
7.3.2 Notes on debugging in CPU rewrite mode 120
7.3.3 Note on rewriting flash memory by the E1 or E20 emulator 121
7.3.4 Flash memory during the user program execution 121
7.3.5 MCUs used for debugging 121
7.3.6 Flash memory ID code 122
7.4 Power supply 123
7.5 Operation during a user program halt 123
7.6 Memory access during user program execution 123
7.7 Final evaluation of the program 123
7.8 Debug functions 124
7.8.1 Step execution 124
7.8.2 Other debug functions 125
7.9 Notes on using the CAN module (applicable for the R8C/3xW and R8C/3xX only) 126
7.9.1 Notes on CAN module in operation 126
7.9.2 Operation during a user program halt 126
Appendix A Menus 127
Appendix B Notes on the High-performance Embedded Workshop 132

1. Inside the E1/E20 Emulator User's Manual

The E1/E20 manual consists of two documents: the E1/E20 Emulator User's Manual and the E1/E20 Emulator Additional Document for User's Manual (this document). Be sure to read both documents before using the E1/E20 emulator.
In this user's manual, the symbol # is used to show active LOW. (e.g. RESET#)

(1) E1/E20 Emulator User's Manual
The E1/E20 Emulator User's Manual describes the hardware specifications.

- Components of the emulators
- Emulator hardware specifications
- Connecting the emulator to the host computer or user system

(2) E1/E20 Emulator Additional Document for User's Manual
The E1/E20 Emulator Additional Document for User's Manual describes functions of the emulator debugger, how to use the debugger, content dependent on the MCUs and precautionary notes.

- Debugging functions supported by the R8C E1/E20 Emulator Debugger
- Individual functions of the emulator debugger
- Example of the emulator connection or interface circuit necessary for designing the hardware
- MCU resources used by the emulator
- Notes on using the emulator
- Setting the emulator debugger during startup
- Operating the emulator debugger
- Tutorial: From starting up the emulator debugger to debugging

Note:
- For the specifications and supported MCUs of the optional FDT, please check the Flash Development Tool Kit page of our website (http://www.renesas.com/tools).
- FDT stands for the Flash Development Toolkit.

Trademarks
Microsoft, MS-DOS, Visual SourceSafe, 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 trademarks or registered trademarks of their respective owners.

Terminology

Firmware:	This means a control program stored in the E1 and E20 emulators. 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.
Debug monitor:	Written into the target MCU, this analyzes the contents of communications with the emulator debugger and controls the target MCU

2. E1/E20 Emulator Specifications

2.1 Target MCUs

Table 2.1 shows the target MCUs covered in this user's manual.

Table 2.1 Target MCUs

Item	Description
Target MCUs	R8C Family R8C/3x Series R8C/3xC: R8C/32C, R8C/33C, R8C/34C, R8C/35C, R8C/36C, R8C/38C, R8C/3GC and R8C/3JC Groups R8C/3xM: R8C/32M, R8C/33M, R8C/34M, R8C/35M, R8C/36M, R8C/38M, R8C/3GM and R8C/3JM Groups R8C/3xT: R8C/33T, R8C/3JT and R8C/3NT Groups R8C/3xW: R8C/34W, R8C/36W and R8C/38W Groups R8C/3xX: R8C/34X, R8C/36X and R8C/38X Groups R8C/3xY: R8C/34Y, R8C/36Y and R8C/38Y Groups R8C/3xZ: R8C/34Z, R8C/36Z and R8C/38Z Groups R8C/3xGHPR: R8C/32G, R8C/32H, R8C/33G, R8C/33H, R8C/34P and R8C/34R Groups R8C/3xKU: R8C/34K, R8C/34U, R8C/3MK and R8C/3MU Groups R8C/3MQ Group
	R8C Family R8C/Lx Series R8C/LxC: R8C/L35C, R8C/L36C, R8C/L38C and R8C/L3AC Groups R8C/LxM: R8C/L35M, R8C/L36M, R8C/L38M and R8C/L3AM Groups R8C/LA6A and R8C/LA8A Groups R8C/LA3A and R8C/LA5A Groups R8C/LAPS Group
Available operating modes	Single-chip mode

2.2 Emulator specifications

Table 2.2 shows the specifications of the emulator supported by the R8C E1/E20 Emulator Debugger. Table 2.3 and Table 2.4 show the Operating Environment of the R8C E1/E20 Emulator Debugger. Table 2.5 shows the MCU-related specifications of the R8C E1/E20 Emulator Debugger.

Table 2.2 Emulator Specifications

Item	Description		
Emulators	E1 (R0E000010KCE00) E20 (R0E000200KCT00)		
Emulator power supply	Unnecessary (USB bus powered, power supplied from the host machine)		
Applicable emulator debugger	R8C E1/E20 Emulator Debugger		
Operating Environment	Temperatures	Active	: 10°C to 35°C
		Inactive	: –10°C to 50°C
	Humidity	Active	: 35% RH to 80% RH, no condensation
		Inactive	: 35% RH to 80% RH, no condensation
	Vibrations	Active	: maximum 2.45 m/s^2
		Inactive	: maximum 4.9 m/s^2
		Transportation	: maximum 14.7 m/s^2
	Ambient gases	No corrosive gases	

Table 2.3 R8C E1/E20 Emulator Debugger Operating Environment (Windows® XP)

Item	Description
PC	IBM PC/AT compatible.
OS	Windows® XP (32-bit editions)* The 64-bit editions of Windows® XP are not supported.
CPU	Pentium 4 running at 1.6 GHz or more recommended.
Interface with host machine	USB (USB 2.0, full speed/high speed)* * Also connectable to host computers that support USB 1.1 * Operation with all combinations of host machine, USB device and USB hub is not guaranteed for the USB interface.
Memory	1 Gbyte plus 10 times the file size of the load module or larger recommended.
Pointing device such as mouse	Mouse or any other pointing device usable with the above OS that can be connected to the host machine.
CD drive	Needed to install the emulator debugger or refer to the user's manual.
Hard disk	Emulator debugger installation needs 600 MB or more free space. (In view of swap area, keep another free space which is more than twice the memory capacity. (More than four times the memory capacity recommended.))
Display resolution	1024 × 768 or greater recommended.

Table 2.4 R8C E1/E20 Emulator Debugger Operating Environment (Windows Vista® or Windows® 7)

Item	Description
PC	IBM PC/AT compatible.
OS	Windows® 7 (32-bit and 64-bit editions) Windows Vista® (32-bit editions) The 64-bit editions of Windows Vista® are not supported.
CPU	Pentium 4 running at 3 GHz or Core 2 Duo running at 1 GHz or more recommended.
Interface with host machine	USB (USB 2.0, full speed/high speed)* * Also connectable to host computers that support USB 1.1 * Operation with all combinations of host machine, USB device and USB hub is not guaranteed for the USB interface.
Memory	2 Gbytes plus 10 times the file size of the load module or larger recommended (32-bit editions). 3 Gbytes plus 10 times the file size of the load module or larger recommended (64-bit editions).
Pointing device such as mouse	Mouse or any other pointing device usable with the above OS that can be connected to the host machine.
CD drive	Needed to install the emulator debugger or refer to the user's manual.
Hard disk	Emulator debugger installation needs 600 MB or more free space. (In view of swap area, keep another free space which is more than twice the memory capacity. (More than four times the memory capacity recommended.))
Display resolution	1024 × 768 or greater recommended.

Table 2.5 R8C E1/E20 Emulator Debugger Specifications (MCU-related)

Item	Description
Power voltages	1.8 - 5.5 V [*1] For details, refer to the hardware manual of the MCU.
Operating frequency range	Maximum operating frequency : 20MHz Minimum operating frequency : 32.768KHz
Interface with user system	14-pin connector* * When connecting to the E20 emulator, the included “38-pin to 14-pin conversion adapter” is required.
Pins for communication with MCU	MODE pin
Communication method with MCU	1-line clock asynchronous serial interface
MCU resources to be used	- Internal ROM size: 2 KB (for some products, see ”7.1 MCU resources used by the emulator” on page 108.) - Stack 8 bytes - Address match interrupt - Vector area (BRK instruction, address match, single-step, address break)
Power supply function	Can supply 3.3 V or 5.0 V to the user system (maximum 200 mA) [*2]

Notes:

*1 For rewriting the flash memory, set the power voltage within the programming and erasure voltage range specified in the hardware manual of the MCU.
For details, refer to “7.1.12 Note on debugging at less than 2.7V” on page 117.

*2 The power supply function is available with the E1 emulator only.
Do not use the power-supply function of the E1 emulator when it is being used to program flash memory as part of a mass-production process. Separately supply power from the user system in accord with the specifications of the MCU.
Use FDT when you need to program flash memory during mass-production, etc.
Voltage supplied from the E1 emulator depends on the quality of the USB power supply of the host computer, and as such, precision is not guaranteed.

2.3 Applicable tool chain and third-party products

You can debug a module created by the inhouse tool chain and third-party products listed in Table 2.6 below.

Table 2.6 Applicable Tool Chain and Third-party Products

Tool chain	M3T-NC30WA V.5.20 Release 01 or later
Third-party products	TASKING M16C C/C++/EC++ Compiler V.2.3r1 or later [*1] IAR EWM16C V.2.12 or later

Note:

*1 Notes on debugging the load modules created in ELF/DWARF2 format

If the load module was created in ELF/DWARF2 format using TASKING M16C C/C++/EC++ compiler V3.0r1, the precautionary note described below must be observed when displaying member variables of the base class in the [Watch] window.

Precautionary Note:

If any class object with a base class is defined, the following problems may occur:

Case 1: Member variables of the base class cannot be referenced directly from the class object (*1).
=>Use indirect references from the class object to refer to member variables of the base class (*2) (*3).

Case 2: If the PC value resides in any member function of a derived class, member variables of the base class cannot be referenced directly (*4).
=> Use indirect references from “this” pointer to refer to member variables of the base class (*5) (*6).

Figure 2.1 shows a code example, and Figure 2.2 shows a [Watch] window registration example.

```
////////////////////////////////////////////////////
  *.h
        class BaseClass
        {
        public:
              int m_iBase;
        public:
              BaseClass() {
                    m_iBase = 0;
              }
              void BaseFunc(void);
        };

        class DerivedClass : public BaseClass
        {
        public:
              int m_iDerive;
        public:
              DerivedClass() {
                    m_iDerive     = 0;
              }
              void DerivedFunc(void);
        };


  *.cpp
        main()
        {
              class DerivedClass ClassObj;
              ClassObj.DerivedFunc();
              return;
        }

       void BaseClass::BaseFunc(void)
        {
              m_iBase = 0x1234;
        }

        void DerivedClass::DerivedFunc(void)
        {
              BaseFunc();
              m_iDerive     = 0x1234;
        }
////////////////////////////////////////////////////
```

Figure 2.1 Example Code

```
////////////////////////////////////////////////////
      Case 1: If the PC value resides in the main() function
      (1)"ClassObj.m_iBase"                  : Cannot be referenced (*1)
      (2)"ClassObj.__b_BaseClass.m_iBase"    : Can be referenced (*2)
      (3)"ClassObj"
            -"__b_BaseClass"
                  -"m_iBase"                 : Can be referenced (*3)
            -"m_iDerive"
                                    -: Expansion symbol
      Case 2: If the PC value resides in the DerivedClass::DerivedFunc() function
      (1)"m_iBase"                           : Cannot be referenced (*4)
      (2)"this->__b_BaseClass.m_iBase"       : Can be referenced (*5)
      (3)"__b_BaseClass.m_iBase"             : Can be referenced (*5)
      (4)"this"
            -"*"
              -"__b_BaseClass"
                    -"m_iBase"               : Can be referenced (*6)
              -"m_iDerive"
      (5)"__b_BaseClass"
            -"m_iBase"                       : Can be referenced (*6)
////////////////////////////////////////////////////
```

Figure 2.2 [Watch] Window Registration Example

3. Designing the User System

3.1 Connector for connecting the E1 or E20 emulator and the user system

Before connecting the E1 or E20 emulator to the user system, a connector must be installed in the user system so a user system interface cable can be connected. Table 3.1 shows the recommended connector for the E1 and E20 emulators. Figure 3.1 shows the connection of the user system interface cable to the user system when the 14-pin connector is in use with the E1 emulator. Figure 3.2 shows the connection of the user system interface cable to the user system when the 38-pin to 14-pin conversion adapter is in use with the E20. The same connection is also required when using the E1 or E20 emulator as a programmer in the FDT.

When designing the user system, refer to Figure 3.3 “Pin Assignments of the 14-pin Connector for the E1 and E20” on page 14 and “3.4 Recommended connection example between the E1/E20 connecting connector and the MCU” on page 17. Before designing the user system, be sure to read the E1/E20 Emulator User’s Manual and related device hardware manuals.

Table 3.1 Recommended Connector

	Type Number	Manufacturer	Specification
14-pin connector	2514-6002	3M Limited	14-pin straight type (for use outside Japan)
	7614-6002	3M Limited	14-pin straight type (for use in Japan)

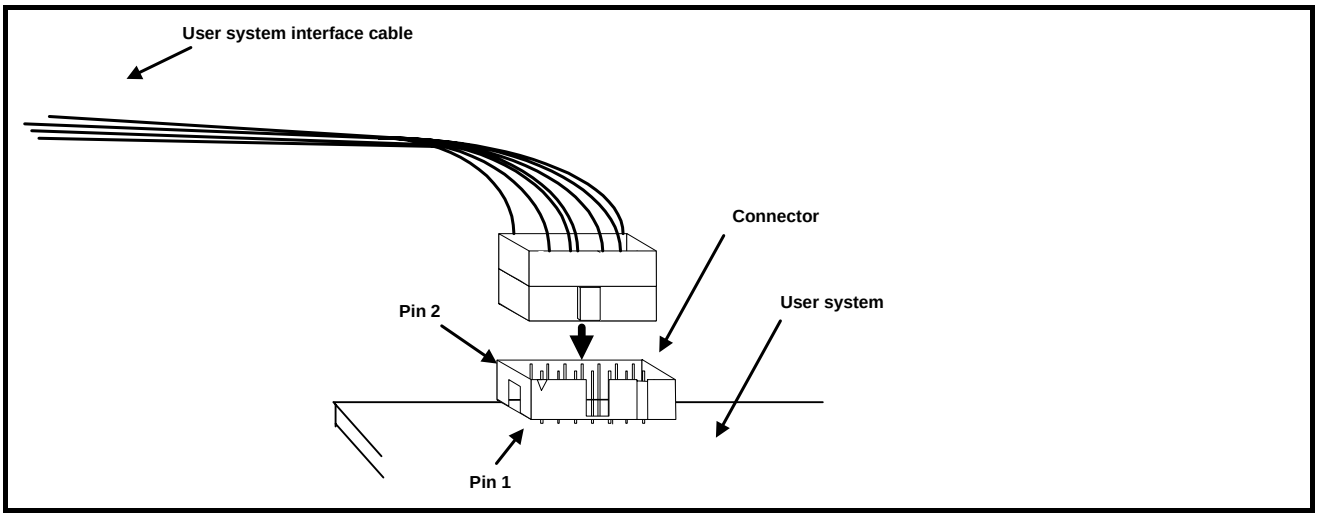


Figure 3.1 Connecting the User-System Interface Cable when the 14-Pin Connector is in Use

Notes:

- Do not mount components with a height exceeding 10 mm within 5 mm around the 14-pin connector on the user system.
- Connect 14-pin connector pins 2, 12 and 14 firmly to the GND on the user system board. These pins are used as an electric GND and monitor the connection of the user system connector.
- When inserting or removing the user system interface cable from the connector section of the user system, be sure to hold the connector cover at the head of the cable. Removal by pulling the cable portion instead of grasping the cover causes breakage of the cable connection.
- The trace function for the MCU used is accomplished by using the MCU’s internal facilities only. Therefore, the external trace output function is not supported, so be sure to use the 38-pin to14-pin conversion adapter supplied with the E20 emulator to connect to the 14-pin connector on the user system.

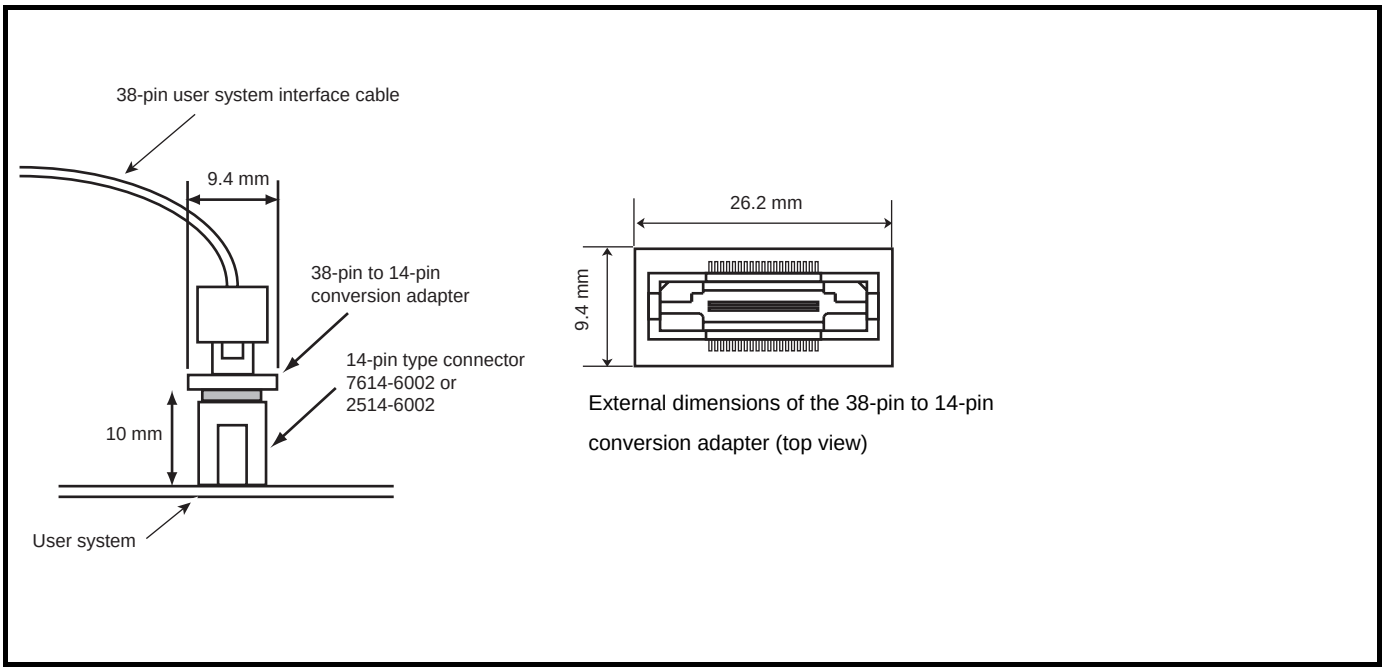
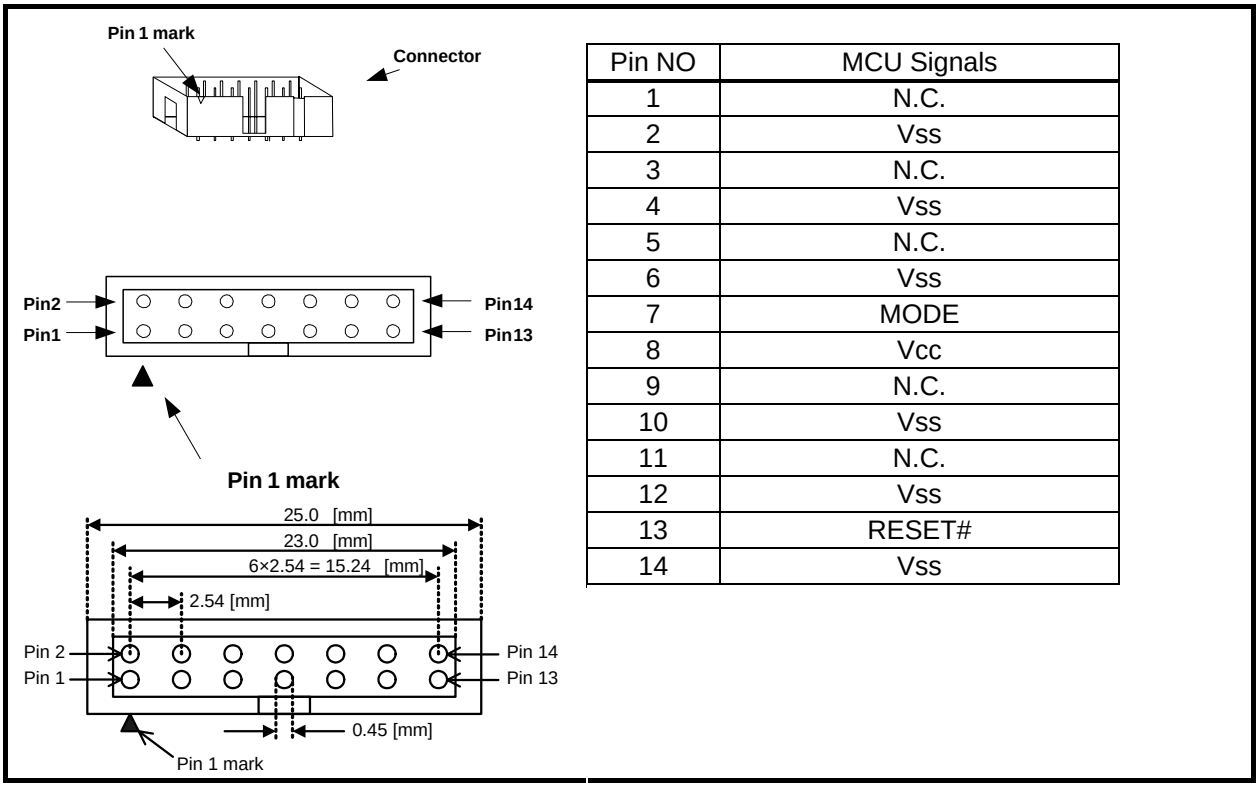


Figure 3.2 Connecting the User System Interface Cable when the 38-pin to 14-pin Conversion Adapter is in Use with the E20

Notes:

- Do not mount components with a height exceeding 10 mm within 5 mm around the 14-pin connector on the user system.
- Connect 14-pin connector pins 2, 12 and 14 firmly to the GND on the user system board. These pins are used as an electric GND and monitor the connection of the user system connector.
- When inserting or removing the user system interface cable from the connector section of the user system, be sure to hold the connector cover at the head of the cable. Removal by pulling the cable portion instead of grasping the cover causes breakage of the cable connection.



Pin NO	MCU Signals
1	N.C.
2	Vss
3	N.C.
4	Vss
5	N.C.
6	Vss
7	MODE
8	Vcc
9	N.C.
10	Vss
11	N.C.
12	Vss
13	RESET#
14	Vss

Figure 3.3 Pin Assignments of the 14-pin Connector for the E1 and E20

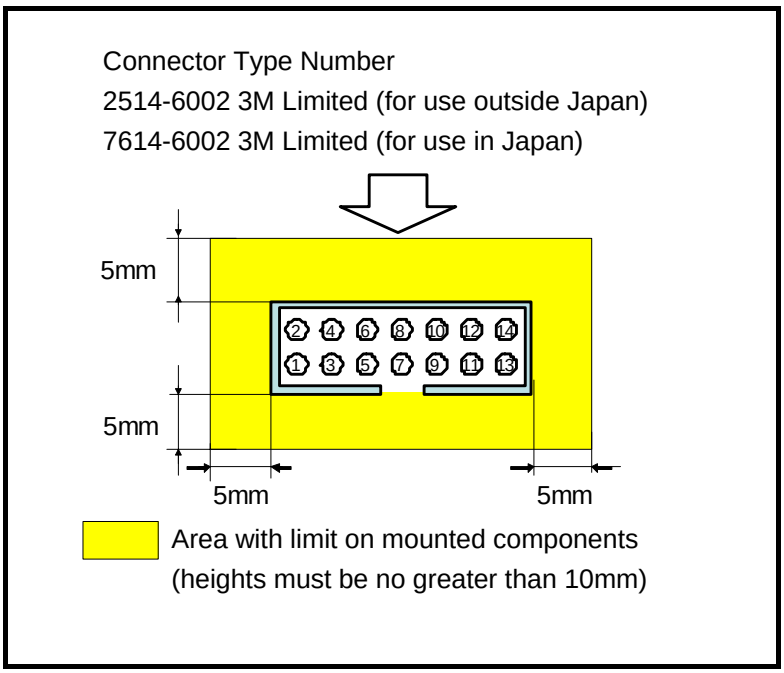


Figure 3.4 Upper Limit Applied to the Heights of Components Mounted around the 14-pin Connector

Notes:

- Pin 14 is used for checking the connection between the E1 or E20 emulator and the user system, and is not directly connected to the Vss inside the emulator. Make sure pins 2, 4, 6, 10, 12, and 14 are all connected to the Vss.
- Note the pin assignments for the user system connector.
- Do not connect anything to the N.C. pin.
- An upper limit applies to the heights of components mounted around the connector on the user system.

3.2 Small connector conversion adapter

When you use the small connector conversion adapter for the E1 emulator (R0E000010CKZ11) that is separately available from Renesas, be aware that the connector pin assignments differ from those of the E1 emulator's standard interface connector.

The 14-pin connector pin assignments when the small connector conversion adapter for the E1 emulator is used are shown in Table 3.2.

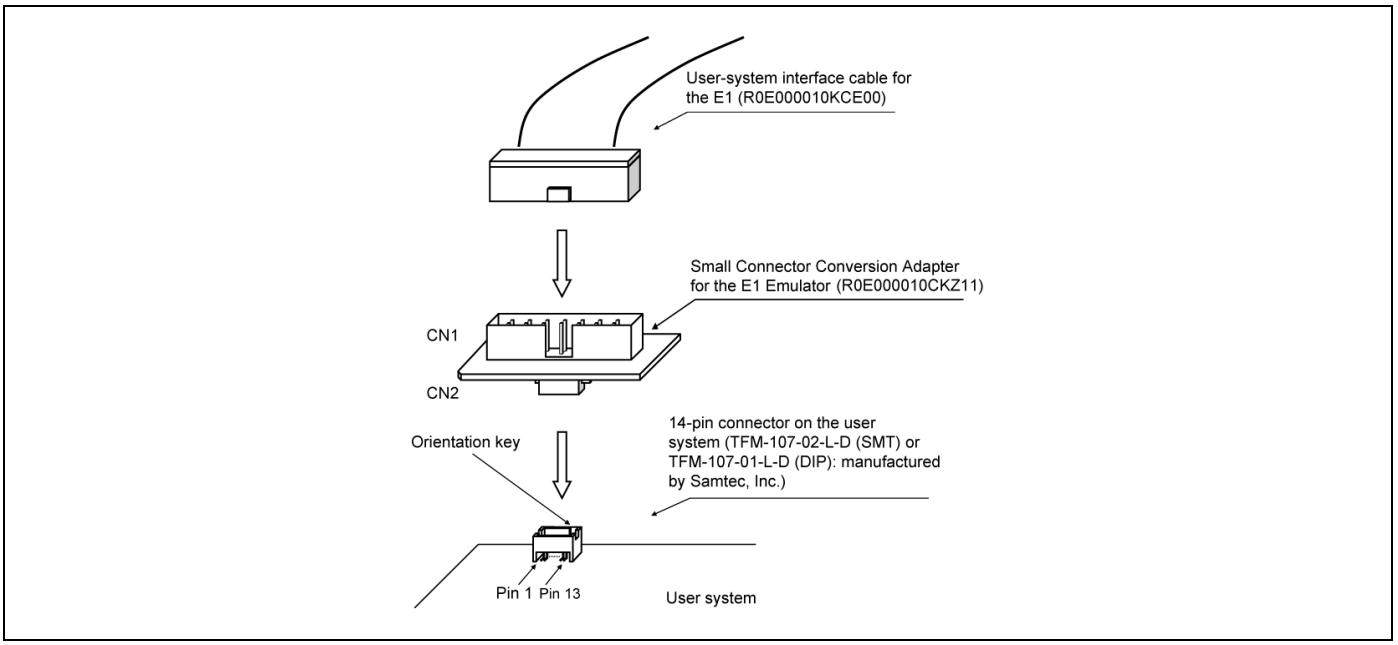


Figure 3.5 Usage of the R0E000010CKZ11

Table 3.2 Connector Pin Assignments when Small Connector Conversion Adapter for the E1 Emulator is Used

Pin NO	MCU Signals
1	Vss
2	MODE
3	Vcc
4	N.C
5	RESET#
6	Vss
7	Vss
8	N.C
9	N.C
10	N.C
11	Vss
12	Vss
13	N.C
14	Vss

Notes:

- Pin 14 is used for checking the connection between the E1 or E20 emulator and the user system, and is not directly connected to the Vss inside the emulator. Make sure pins 1, 6, 7, 11, 12, and 14 are all connected to the Vss.
- Note the pin assignments for the user system connector.
- Do not connect anything to the N.C. pin.
- An upper limit applies to the heights of components mounted around the connector on the user system.
- For the specifications of the small connector conversion adapter for the E1 emulator, refer to the R0E000010CKZ11 user's manual.

3.3 Connecting system ground

The emulator's signal ground is connected to the user system's signal ground. In the emulator, the signal ground and frame ground are connected. In the user system, connect the frame ground only; do not connect the signal ground to the frame ground (See Figure 3.6).
If it is difficult to separate the frame ground from the signal ground in the user system, set the GND for DC power input (AC adapter) of the host computer and the frame ground of the user system as the same potential. If the GND potential is different between the host computer and the user system, an overcurrent will flow in the low-impedance GND line and thin lines might be burned.

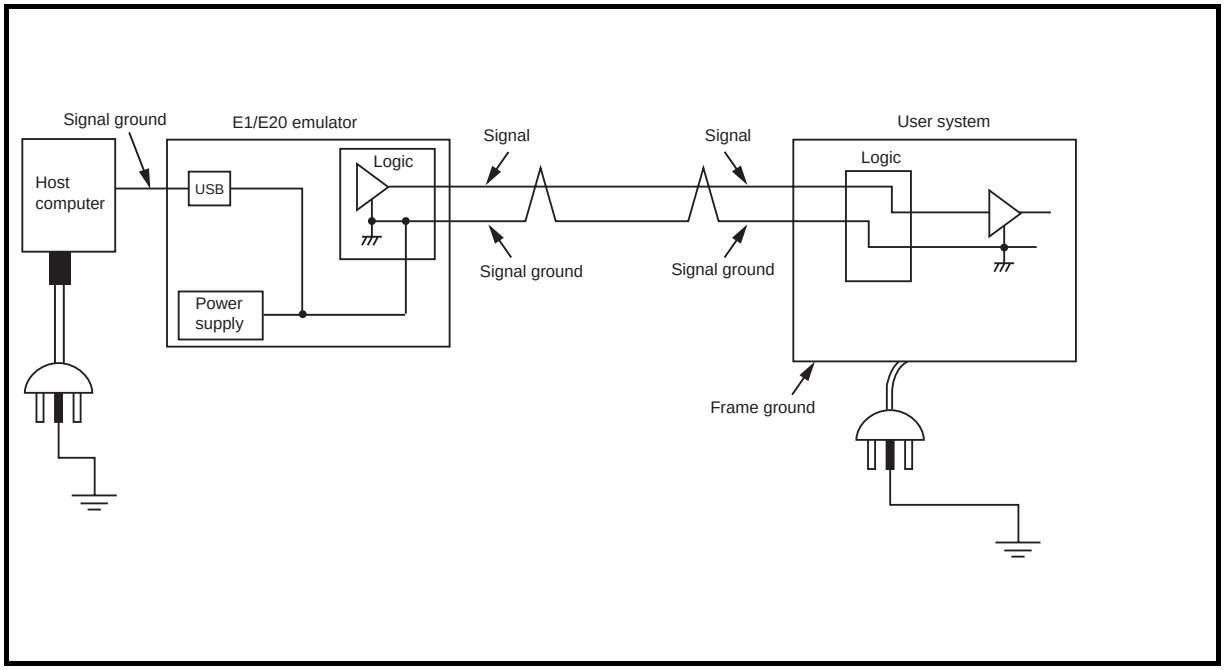


Figure 3.6 Connecting System Ground

⚠ WARNING

Make sure the system ground is separated between the frame ground and signal ground on the user system. Do not connect the emulator to the user system while its frame ground and signal ground are connected. Such an act could cause smoke, fire, or an electric shock due to the difference in ground potential.

3.4 Recommended connection example between the E1/E20 connecting connector and the MCU

Figure 3.7 shows a recommended example of connection between the E1/E20 connecting connector and the MCU. The same connection is also required when using the E1 or E20 emulator as a programmer in the FDT.

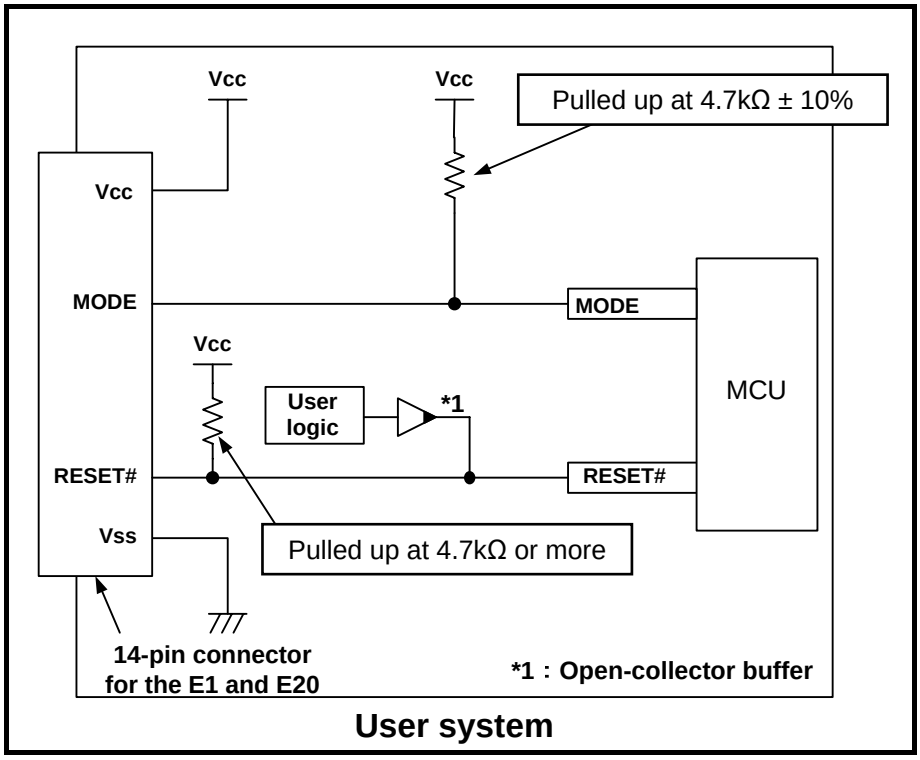


Figure 3.7 Example of an E1/E20 Emulator Connection

Notes:

- When adjacent resistors are used for pull-up, they may be affected by noise from other pins. In particular, separate the resistor for MODE from the other resistors.
- Wiring patterns between the connector and the MCU must be as short as possible (within 50 mm is recommended). Do not connect the signal lines between the connector and MCU to other signal lines.
- For the handling of pins while the E1 or E20 emulator is not in use, refer to the hardware manual for the MCU.

(1) MODE pin

The E1 and E20 emulators use the MODE pin for MCU control and forced break control.
Do not connect a capacitor etc. to this pin.

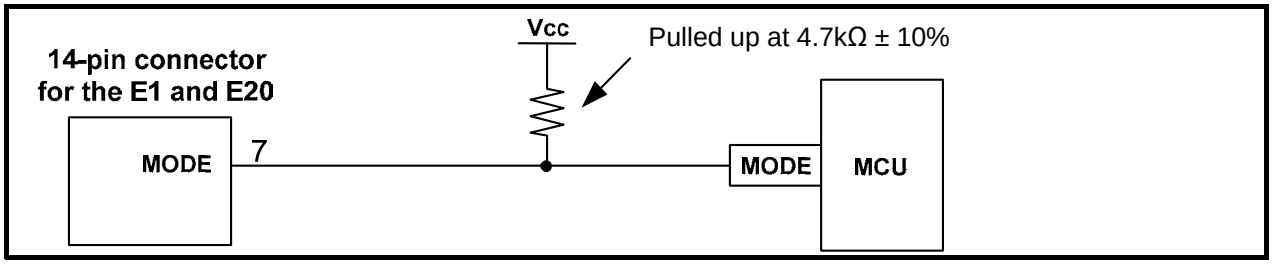


Figure 3.8 Connection of the Emulator and the MODE Pin

(2) RESET# pin

The RESET# pin is used by the emulator for outputting “L” and monitoring pin states. Therefore, use an open-collector output buffer or a CR reset circuit as the reset circuit for the user system. The recommended pull-up value is 4.7 kΩ or more.

The MCU can be reset by outputting “L” from the emulator. However, if the reset IC output is “H”, the user system reset circuit cannot be set to “L”. As such, the emulator will not operate normally.

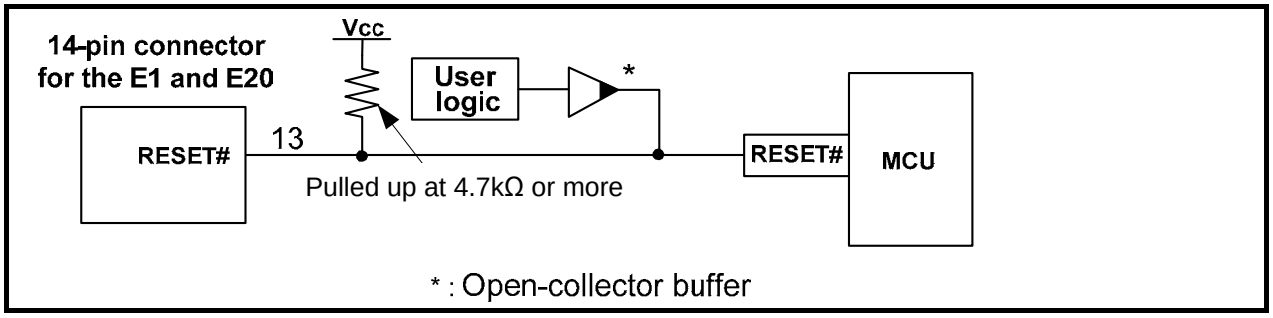


Figure 3.9 Connection of the Emulator and the #RESET Pin

(3) Other pins

- Connect Vss and Vcc to the Vss and Vcc of the MCU, respectively.
- The amount of voltage input to Vcc must be within the specified range of the MCU.
- Pin 14 is used for checking the connection between the E1 or E20 emulator and the user system, and pins 4, 6 and 10 are connected to the internal circuit. These pins are not directly connected to the Vss inside the emulator.
- Make sure pins 2, 4, 6, 10, 12 and 14 are all connected to the Vss.
- Do not connect anything to the N.C. pin.

⚠ WARNING

When supplying power, ensure that there are no short circuits between Vcc and GND. Only connect the E1 or E20 emulator after confirming that there are no mismatches in pin assignments of the E1/E20 connecting connector. Incorrect connection will result in the host computer, the emulator, and the user system emitting smoke or catching fire.

3.5 Interface circuit in the E1 or E20 emulator

Figure 3.10 shows the interface circuit in the E1 emulator, and Figure 3.11 shows the interface circuit in the E20 emulator. Use these figures as a reference when determining the pull-up resistance value.

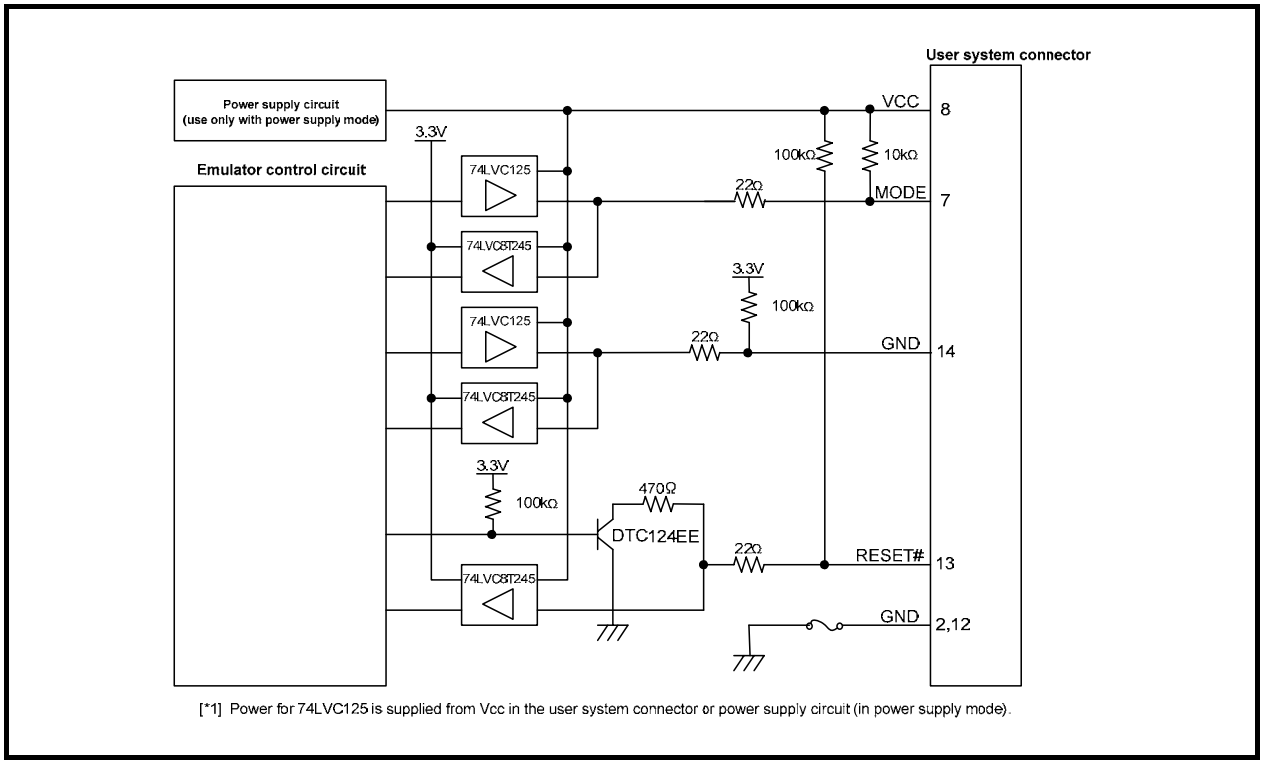


Figure 3.10 Interface Circuit inside the E1 Emulator (For Reference)

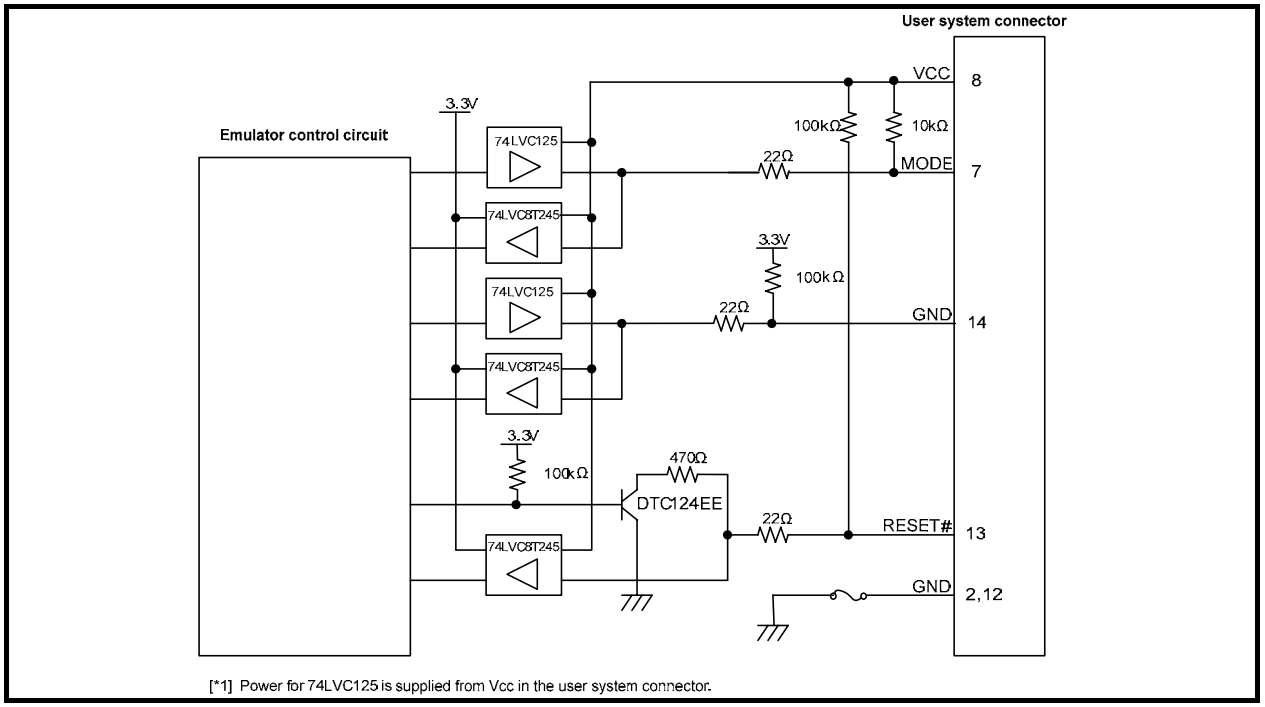


Figure 3.11 Interface Circuit inside the E20 Emulator (For Reference)

4. Preparations for Debugging

4.1 Activating High-performance Embedded Workshop

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

(1) Connect the emulator to the host computer and the user system.

(2) Select [Renesas -> High-performance Embedded Workshop -> High-performance Embedded Workshop] from [Programs] in the [Start] menu.
The [Welcome!] dialog box is displayed.

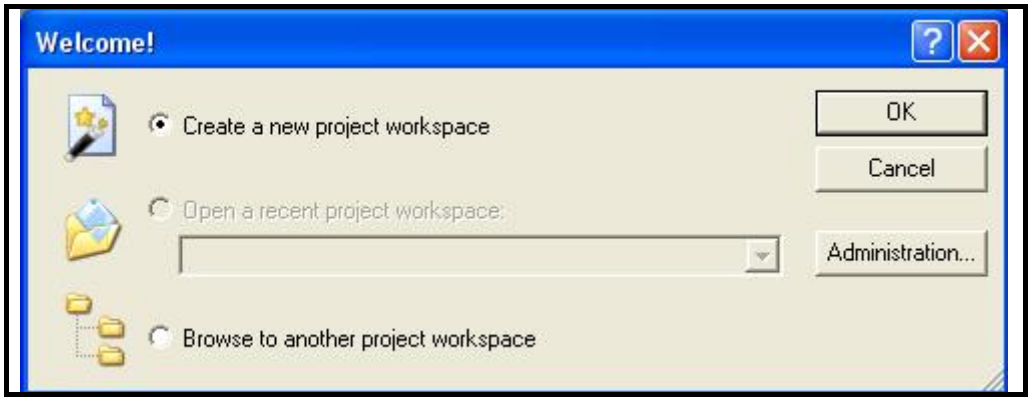


Figure 4.1 [Welcome!] Dialog Box

(3) Select a startup method from the following.

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

4.2 Creating a new workspace (with a toolchain not in use)

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

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

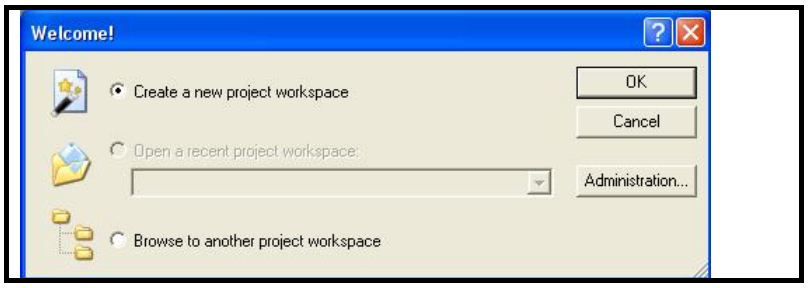


Figure 4.2 Create a new project workspace

(2) The Project Generator is started.

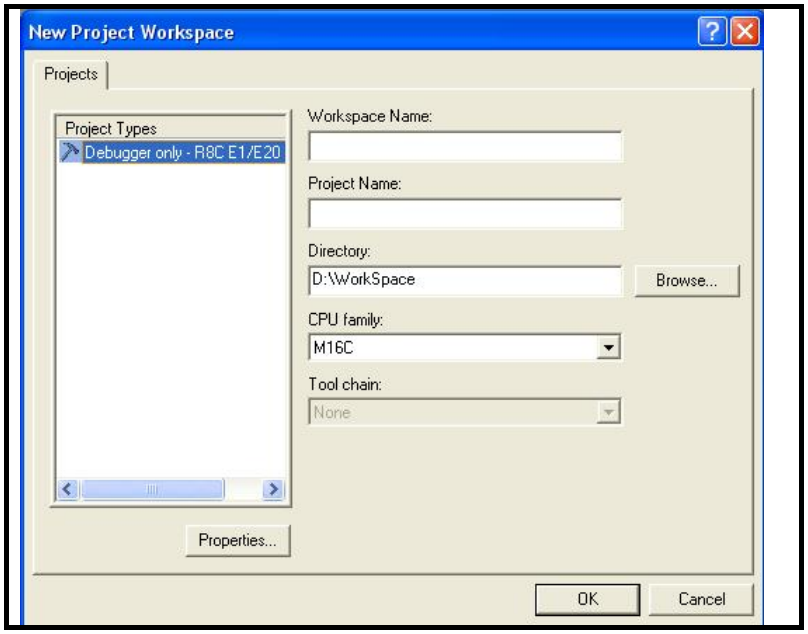


Figure 4.3 [New Project Workspace] Dialog Box

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

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

(3) Select the target for debugging.

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

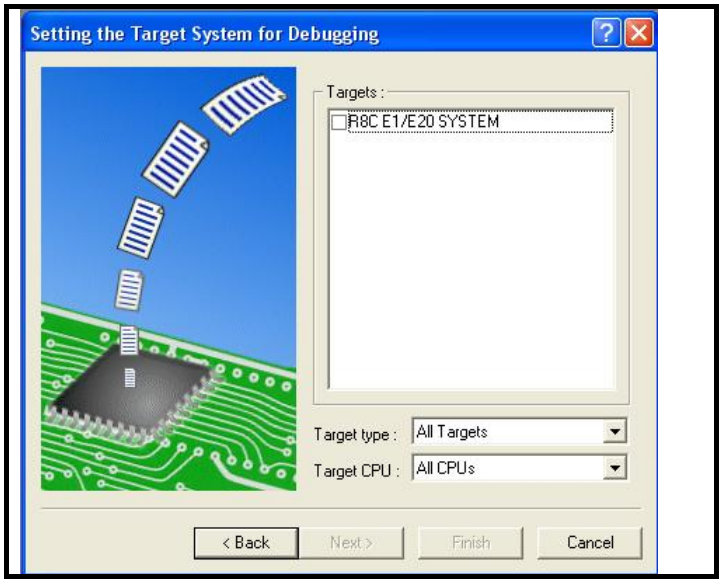


Figure 4.4 [Setting the Target System for Debugging] Dialog Box

(4) Set the configuration file name.

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

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

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

(5) Click the [Finish] button, and the [Summary] dialog box will be displayed.

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

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

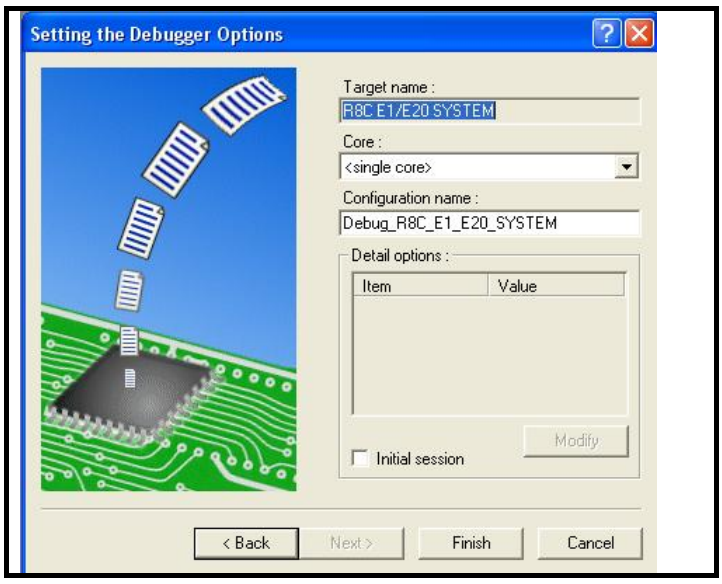


Figure 4.5 [Setting the Debugger Options] Dialog Box

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 [Create a new project workspace] radio button and click the [OK] button.

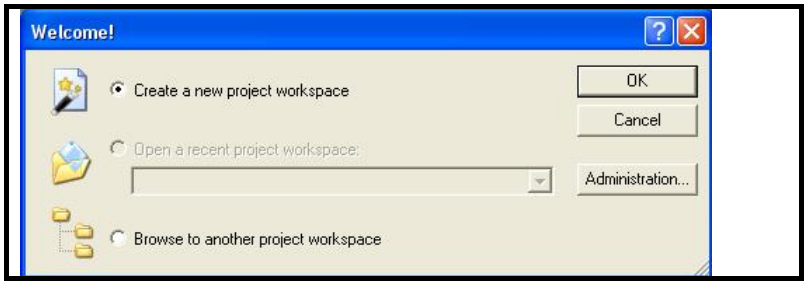


Figure 4.6 [Welcome!] Dialog Box

(2) The Project Generator is started.

[Workspace Name]: Enter a new workspace name.

[Project Name]: Enter a project name. You do not need to enter any name if you wish this to be the same as the workspace name.

[Directory]: Enter the directory in which you want the workspace to be created. Alternatively, click the [Browse] button and select a workspace directory from the dialog box.

[CPU family]: Select the CPU family of the MCU you are using.

[Tool chain]: To use a toolchain, select the appropriate toolchain here. If you do not use any toolchain, select [None]. After making these settings, click the [OK] button.

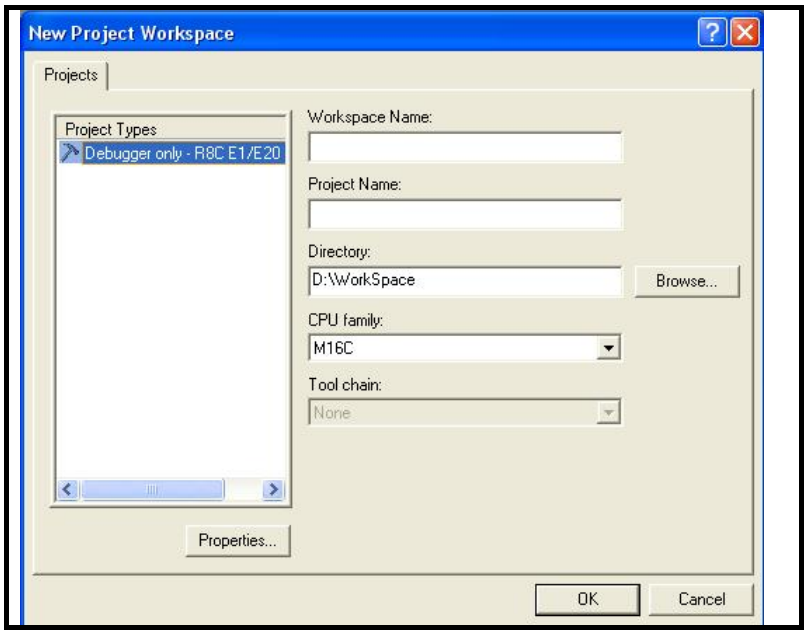


Figure 4.7 [New Project Workspace] Dialog Box

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

(4) Select the target for debugging.

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

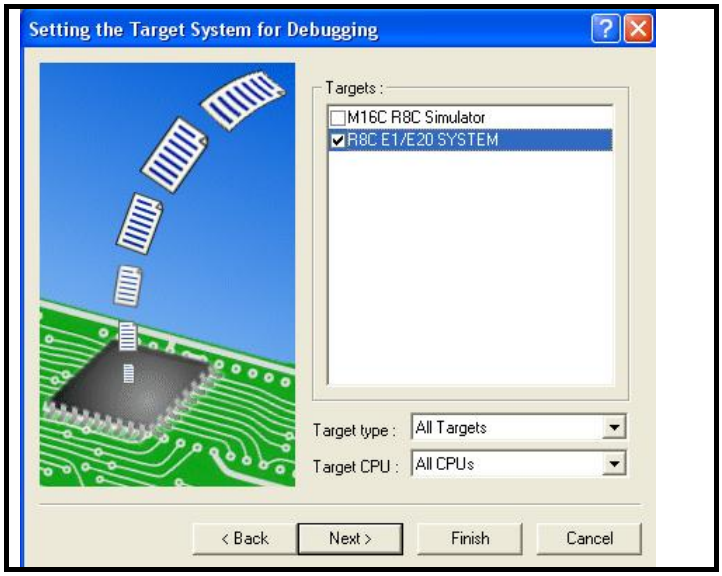


Figure 4.8 [Setting the Target System for Debugging] Dialog Box

(5) Set the configuration file name.

If you have selected two or more target platforms, click the [Next] button and then set a configuration name for each of the selected target platforms. When you have finished setting the configuration names, settings related to the emulator debugger have been completed.

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

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

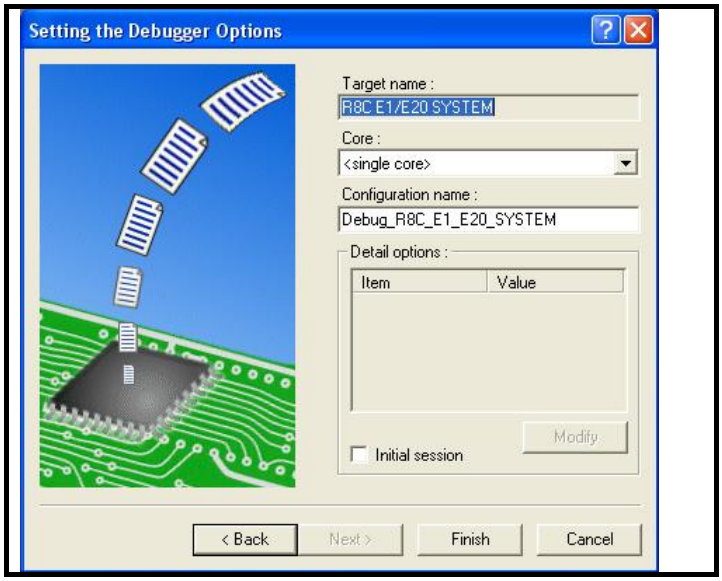


Figure 4.9 [Setting the Debugger Options] Dialog Box

4.4 Opening an existing workspace

Follow the procedure below to open an existing workspace.

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

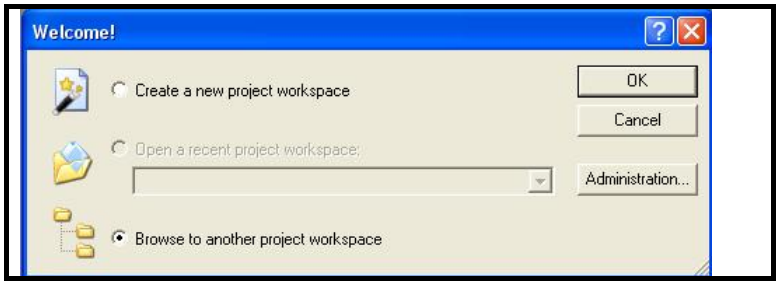


Figure 4.10 [Welcome!] Dialog Box

(2) The [Open Workspace] dialog box is displayed.
Specify the directory in which the workspaces was created, select the workspace file (extension ".hws"), and click the Select button.

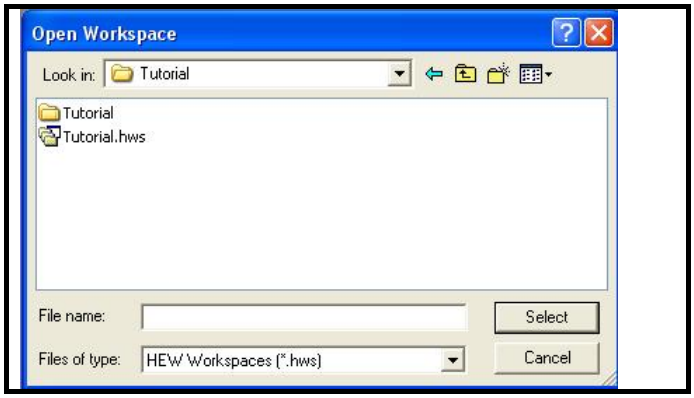


Figure 4.11 [Open Workspace] Dialog Box

(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" on page 26.

4.5 Connecting the emulator

4.5.1 Connecting the emulator

The following methods for connecting the emulator are available.

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

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

4.5.2 Reconnecting the emulator

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

a) Choose [Connect] from the [Debug] menu.
b) Click the [Connect] toolbar button ( ).
c) 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.

a) Choose [Disconnect] from the [Debug] menu.
b) Click the [Disconnect] toolbar button ( ).
c) 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 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.

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

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

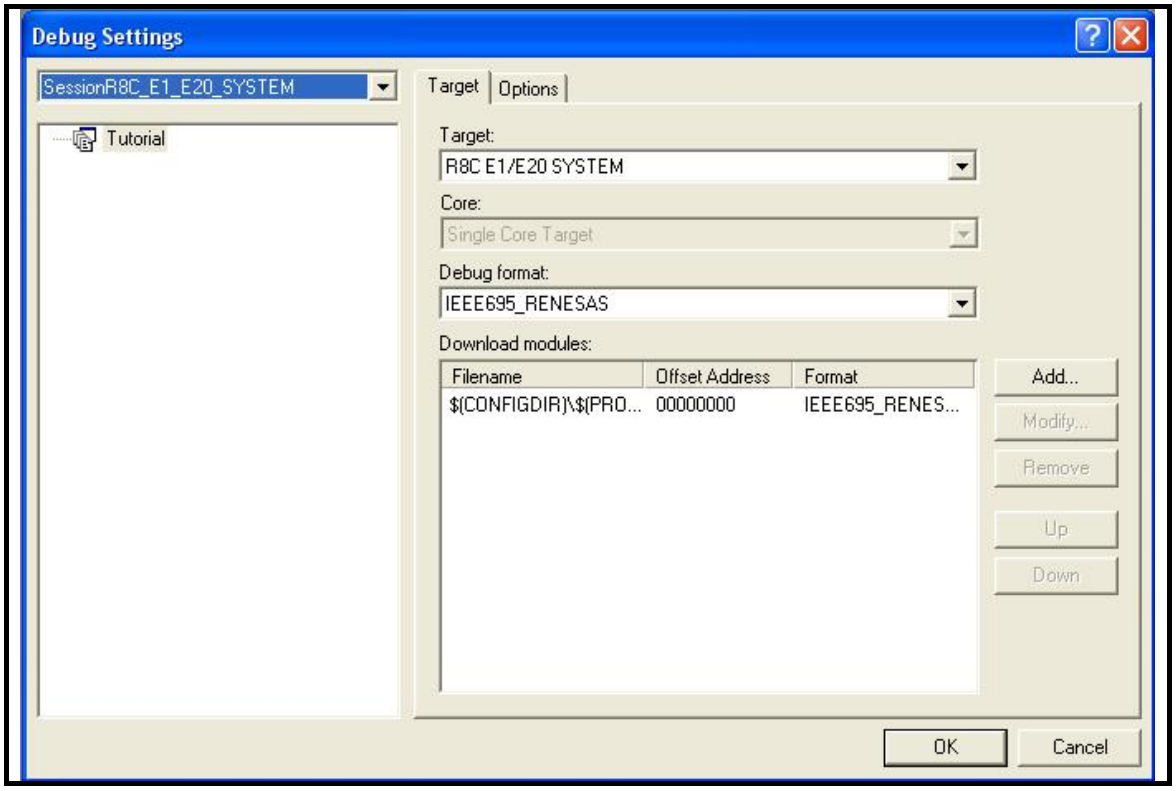


Figure 4.12 [Debug Settings] Dialog Box

Note:

[*1] At this point in time, no programs have been downloaded yet. For details on how to download a program, refer to Section “5.2 Downloading a program” on page 48.

4.8.2 Setting up automatic execution of command line batch files

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

Here, register a command chain to be automatically executed with the specified timing.

Select your desired timing from among the following four choices:

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

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

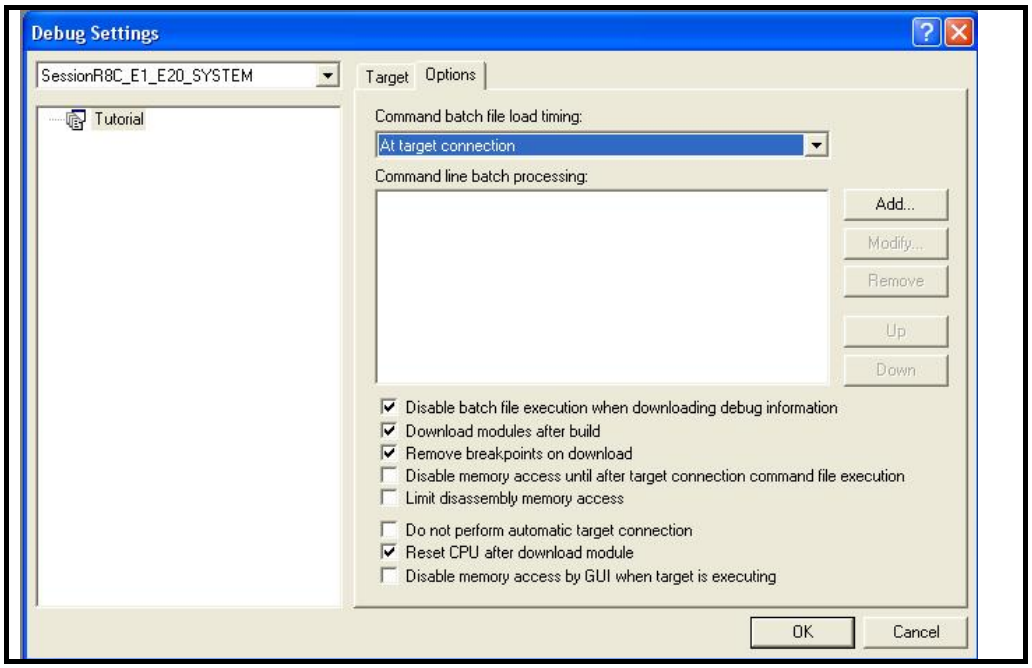


Figure 4.13 [Debug Settings] Dialog Box

4.9 Procedure for launching the E1/E20 emulator debugger

This section covers how to start up the High-performance Embedded Workshop and check the connection of the emulator and the MCU on the user system. Here, use the workspace included as a tutorial for the product.

(1) Take the following steps beforehand.

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

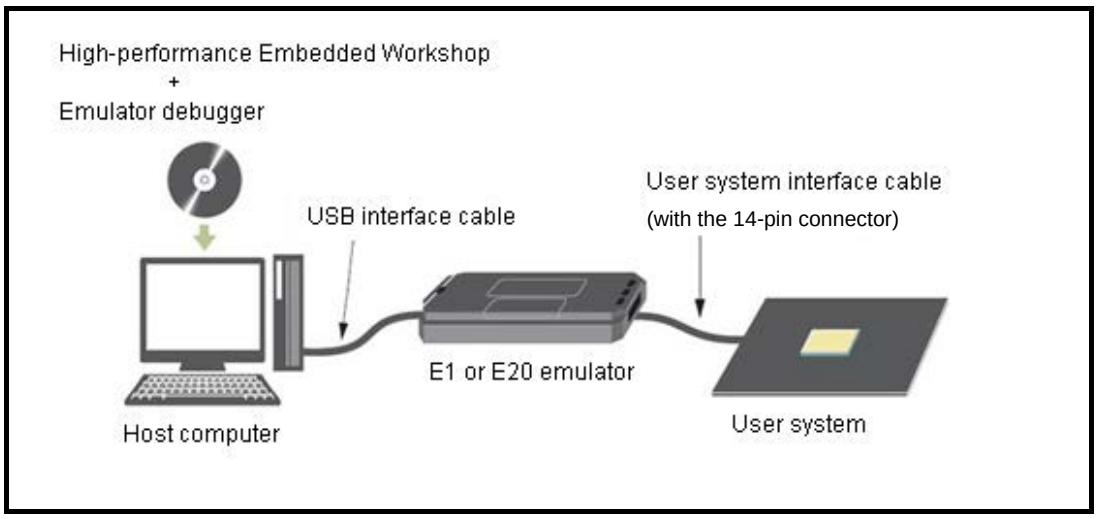


Figure 4.14 Configuration for System Check

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

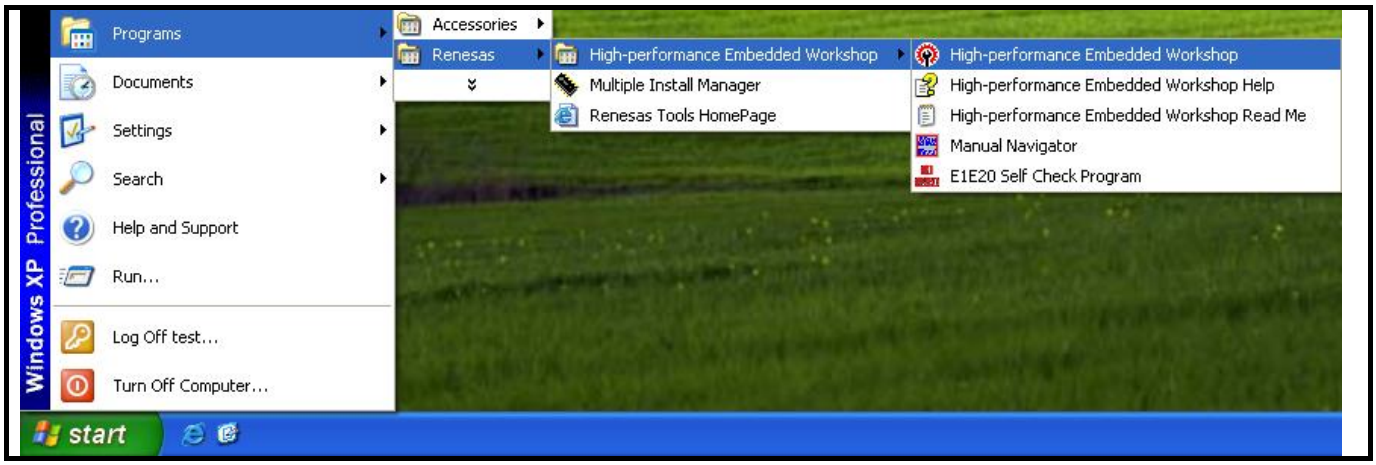


Figure 4.15 [Start] Menu

(3) The [Welcome!] dialog box is displayed.
To use a workspace for the tutorial, select the [Browse to another project workspace] radio button and click the [OK] button.

When the [Open Workspace] dialog box is opened, specify the following directory:
<Drive where the OS has been installed>: \WorkSpace\Tutorial\E1E20\R8C\Tutorial

After the directory has been specified, select the file “Tutorial.hws” and click the [Open] button.

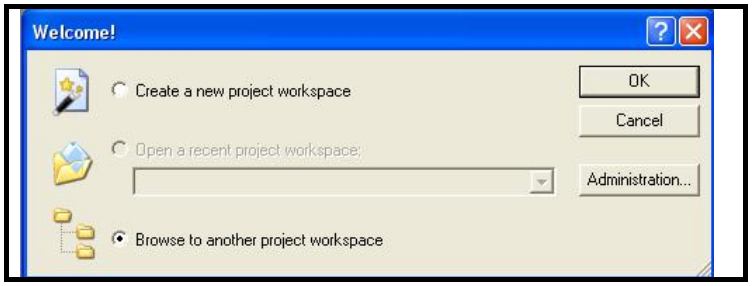


Figure 4.16 [Welcome!] Dialog Box

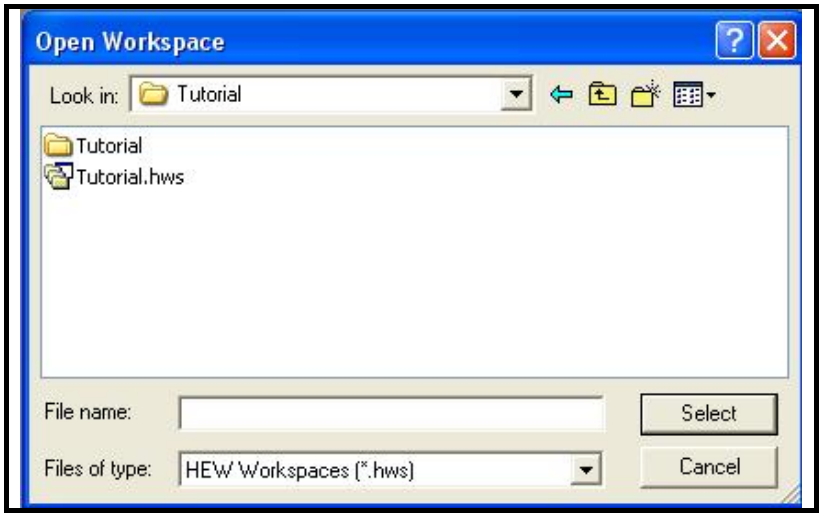


Figure 4.17 [Open Workspace] Dialog Box

4.10 Setting the emulator debugger during startup

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

(1) [Initial Settings] dialog box

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

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

(2) [Configuration Properties] dialog box

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

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

Some options in this dialog box can have their settings changed after startup.

The changeable options are displayed as in active use, while the unchangeable options are inactive (grayed out), with their set contents only displayed.

(3) [Connecting…] dialog box

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

4.11 [Initial Settings] dialog box

The [Initial Settings] dialog box is provided for setting items that need to be set when the debugger is launched. The contents set from this dialog box (excluding [Power supply] group box items) also become valid the next time the debugger is launched.
When launching the debugger for the first time after creating a new project work space, the [Initial Settings] dialog box is displayed with the Wizard.
The settings you have made here cannot be changed after the emulator is booted up. To change the settings, you need to cancel the process of booting-up and then reboot the emulator.

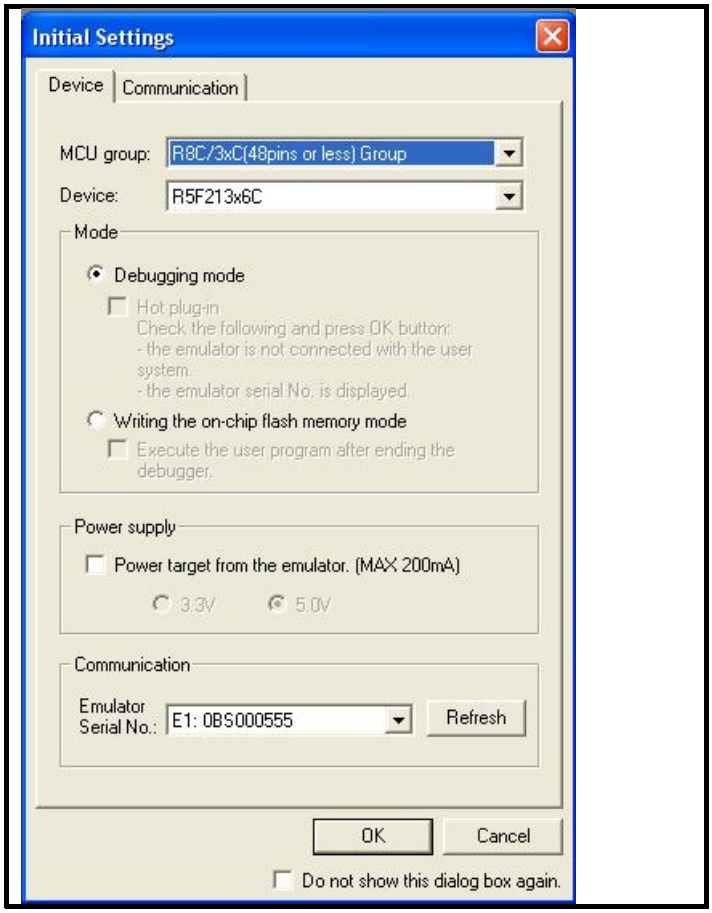


Figure 4.18 [Initial Settings] Dialog Box

If you check [Do not show this dialog box again.] at the bottom of the [Initial Settings] dialog box, the [Initial Settings] dialog box will not be displayed the next time the debugger is launched.

When [Do not show this dialog box again.] is checked, the E1 emulator does not supply power to the user system.

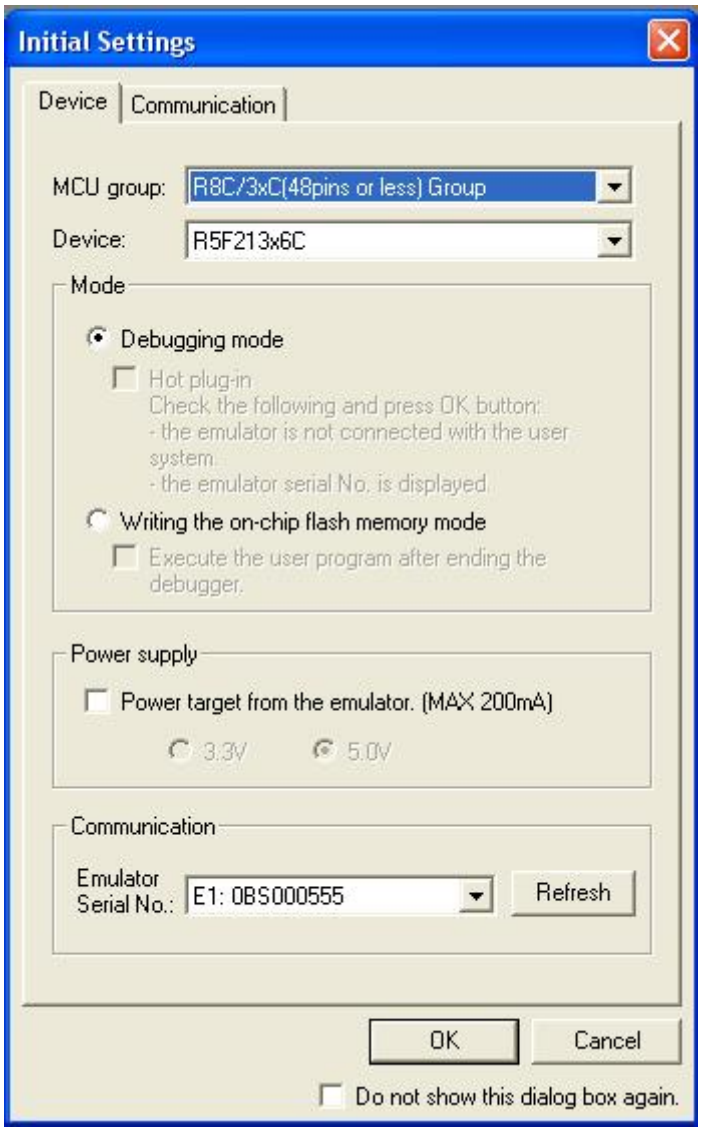
You can open the [Initial Settings] dialog box using one of the following methods:

- After the debugger is launched, select Menu -> [Setup] -> [Emulator] -> [Device Setting…].
- Hold down the Ctrl key while launching the debugger.

Note:
Unsupported options are grayed out depending on the selected types of the MCU.

4.11.1 [Device] tab

Device selection, operation mode specification and power supply setting are made from the [Device] tab of the [Initial Settings] dialog box.



[MCU group]

Select the name of the MCU group to be used from the [MCU group] drop-down list.

[Device]

Select the type of MCU to be used from the [Device] drop-down list.

(See "7.1 MCU resources used by the emulator" on page 108 for the list of applicable MCUs.)

[Mode]

Select the mode to be used.

For details, see "4.11.1 (1) Selecting the Mode " on page 34.

[Power supply]

Select the power supply to the user system.

- When supplying power to the user system from the E1, select the [Power Target from Emulator. (MAX 200mA)] checkbox.
- Select either [3.3V] or [5.0V] by checking the checkbox. [*1]

Communication

You can select another target emulator for connection via USB.

The [Emulator Serial No.] list box shows unique identifying information on the emulator connected via USB. Clicking on the [Refresh] button updates the information. [*2]

Figure 4.19 [Initial Settings] Dialog Box ([Device] Tab)

Notes:

[*1] This option cannot be selected with the E20 emulator because it does not support the power-supply function. Only one power supply option is available depending on the selected types of the MCU. For the information of the operating supply voltage of the MCU, refer to the hardware manual of the MCU used.

[*2] If the emulator serial No. is not displayed, it means that the host machine does not recognize the emulator (in which case the item in the Power supply section cannot be selected). If the serial No. is not displayed even by clicking the [Refresh] button, it is possible that the emulator is not connected to the host machine or the USB driver is not installed. So check again to see that the emulator is connected correctly and that the USB driver is installed.

(1) Selecting the Mode

Table 4.1 Selecting the Mode

Mode	Description
Debugging mode [*1]	This is a mode for debugging. When starting the debugger, the E1 or E20 emulator writes the E1/E20 emulator program and the vector area used by the E1 or E20 emulator. The emulator rewrites the OFS, OFS2 and ID code areas.
Writing the on-chip flash memory mode [*2]	This is a mode to write only the user program (E1/E20 emulator program is not written). Therefore, only the following operations can be executed in this mode. - Downloading the user program - Connecting or disconnecting the MCU Performing operations other than the above will result in errors. When [Execute the user program after ending the debugger.] is selected, with the E1 or E20 emulator connected to the user system, the user program is executed at the same time the debugger is terminated. This checkbox setting is available only when the [writing the on-chip flash memory mode] is selected. If this checkbox is not selected, the emulator debugger ends after resetting the MCU.

Notes:

[*1] In this mode, vector addresses are used by the E1/E20 emulator program. After a program has been downloaded, you cannot disconnect the emulator and operate the user system as a stand-alone unit. (Programs written in this mode cannot be executed from the CPU.)
If you want to execute a program from the CPU, use [writing the on-chip flash memory mode].

Note that, after downloading the user program, the lock bits of the downloaded blocks will be the one set by the user program.
Writing of ID codes (intended to prohibit reading of the internal ROM) to the internal ROM of the MCU is not possible in this mode.

[*2] The user program cannot be debugged in this mode.

Note that, after downloading the user program, the lock bits of the downloaded blocks will be unlocked.
The ID code settings made on the user program will be written to the internal ROM of the MCU.

4.11.2 [Communication] tab

Select communication baud rate between the E1 or E20 emulator and the MCU in the [Communication] tab. 500000 bps (default setting) should be selected during normal use. [*1]

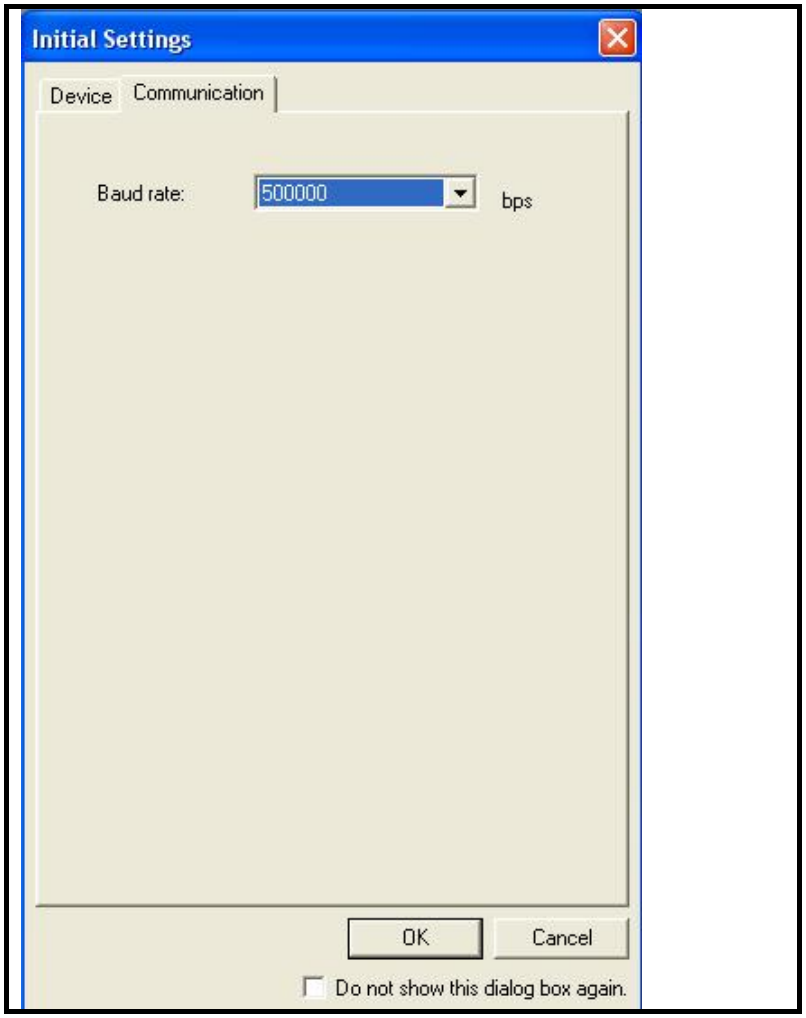


Figure 4.20 [Initial Settings] Dialog Box ([Communication] Tab)

Notes:

[*1] Depending on the wired length of the MODE signal and how it is wired on the user system, communication at the selected baud rate may not be performed.
Reducing this baud rate may help to solve the problem.
Also, the communication information you set here cannot be changed after the emulator debugger has started.
To change the communication baud rate, you need to disconnect the emulator and the MCU temporarily and then reconnect.

[*2] The baud rate of 57600 bps or below is designated for checking purpose in case there is a failure in the connection with the emulator. With such a low baud rate, it takes a long time to write into the flash memory of the target MCU, and the emulator debugger may appear to be giving no response.
Also note if the data of 1024 bytes or larger is handled when displaying the memory contents or in memory fill function, a time-out error may occur because the communication takes up much time.

4.12 Showing the connection dialogs

When you click the [OK] button in the [Initial Settings] dialog box, the [Connecting…] dialog box is displayed to start initial communication with the MCU.
Then the [Configuration Properties] dialog box shown in Section 4.13 appears except in the following five cases.

Table 4.2 Display Contents of the [Connecting…] Dialog Box

No.	Contents	Displayed window
1	When the E1/E20 emulator is used in other emulator debuggers such as RX Family	See 4.12.1
2	When the E1/E20 emulator firmware version is old	See 4.12.2
3	When the [Power supply] setting in the [Initial Settings] dialog box is incorrect [*1]	See 4.12.3
4	When an ID code other than “FFFFFFFFFFFFFFh” is written in the ID code area	See 4.12.4
5	When the emulator cannot communicate with the MCU	See 4.12.5

You can confirm the progress of boot-up processing by checking the [Connecting…] dialog box.
The [Connecting…] dialog box continues displaying progress information from when boot-up processing starts till when it ends.
While the [Device setting…] and the [Configuration Properties] dialog boxes are displayed, you cannot manipulate this dialog box.

The contents shown here are recorded in a trouble report. To check the content of a trouble report, select [Technical Support -> Create Bug Report] from the Help menu.

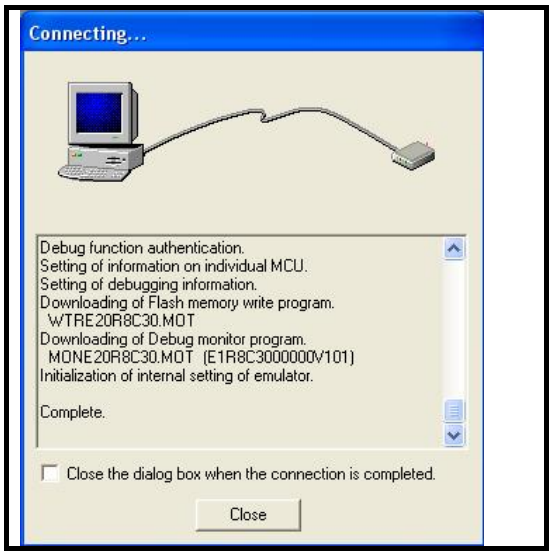


Figure 4.21 [Connecting…] Dialog Box

Note:
[*1] Applies to the E1 emulator only. The E20 emulator does not support the power-supply function.

4.12.1 When the E1/E20 emulator is used in other emulator debuggers such as RX Family

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

(1) Confirming whether or not to update the emulator firmware

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

Figure 4.22 [Update Firmware] Dialog Box

(2) Updating the emulator firmware

The [Downloading] progress bar is displayed during the process of updating. Booting-up of the emulator resumes after the firmware has been updated. [*1]

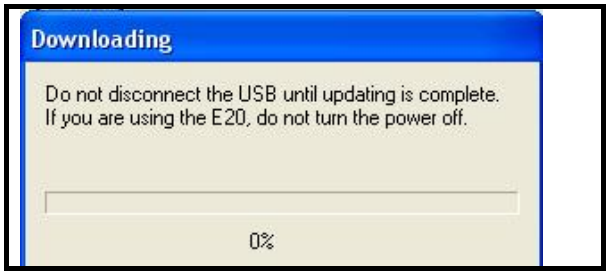


Figure 4.23 [Downloading] Progress Bar

Note:

[*1] The USB interface cable must not be disconnected until writing is complete. Early disconnection may damage the E1 or E20 emulator.

4.12.2 When the E1/E20 emulator firmware version is old

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

(1) Confirming whether or not to update the emulator firmware

Clicking the [OK] button starts updating of the emulator firmware.

Once started, you cannot cancel the updating.

Figure 4.24 [Confirm Firmware] Dialog Box

(2) Updating the emulator firmware

The [Downloading] progress bar is displayed during the process of updating. Booting-up of the emulator resumes after the firmware has been updated. [*1]

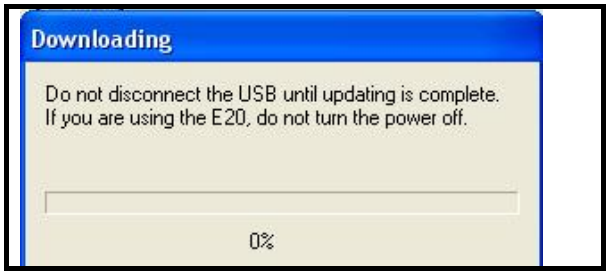


Figure 4.25 [Downloading] Progress Bar

Note:

[*1] The USB interface cable must not be disconnected until writing is complete. Early disconnection may damage the E1 or E20 emulator.

4.12.3 When the [Power supply] setting in the [Initial Settings] dialog box is incorrect

When the power-supply setting in the [Initial Settings] dialog box and the power supply state of the user system do not match, the following windows are displayed.

These windows are displayed only with the E1 emulator. The E20 emulator does not support the power-supply function.

- When [Power target from the emulator.] is checked and the power is already supplied to the user system
- When [Power target from the emulator.] is not checked and the power is already supplied to the user system

(1) When [Power target from the emulator.] is checked and the power is already supplied to the user system

If you have selected the [Power target from the emulator.] checkbox and an external power supply is supplying power to the user system, the dialog box appears as shown in Figure 4.26.

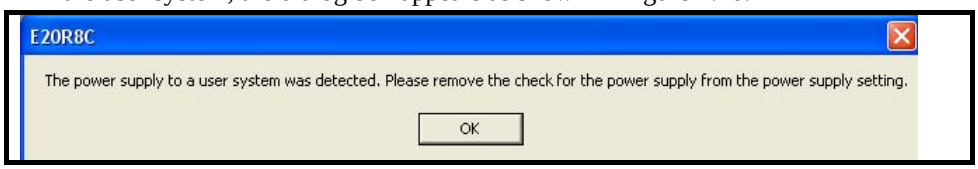


Figure 4.26 External Power for the User System Dialog Box

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

Figure 4.27 [Power Supply] Dialog Box

(2) When [Power target from the emulator] is not checked and the power is already supplied to the user system

If you have not selected the [Power target from the emulator.] checkbox and an external power supply is supplying power to the user system, the dialog box below appears.

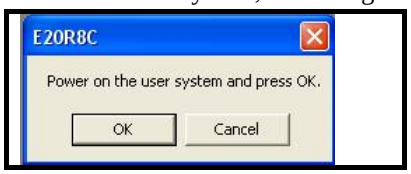


Figure 4.28 Lack of Power Supply Dialog Box

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

Figure 4.29 [Power Supply] Dialog Box

4.12.4 When an ID code other than “FFFFFFFFFFFFFFh” is written in the ID code area

(1) If an ID code other than “FFFFFFFFFFFFFFh” has been set in the target MCU’s ID code area, the following window appears to verify the ID code.

(2) Enter the ID code and click the [OK] button.

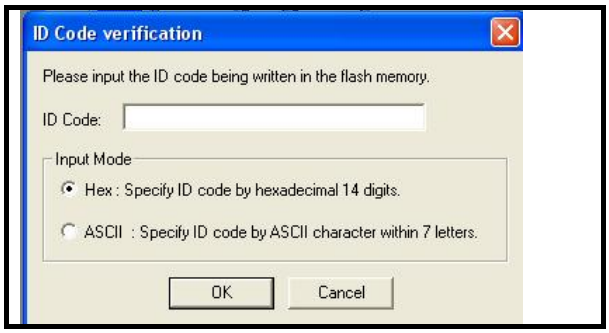


Figure 4.30 [ID Code verification] Dialog Box

Note:
[*1] For details on ID codes, refer to the following:
(1) Hardware manual for the MCU
(2) “7.3.6 Flash memory ID code” on page 122

4.12.5 When the emulator cannot communicate with the MCU

Click the OK button in the [Initial Settings] dialog box to start initial communication with the MCU.

The contents and the causes of errors if encountered during communication with the MCU at this time are listed in Table 4.3.

Table 4.3 Error Messages at the Start of Communication with the MCU and Cause of Errors

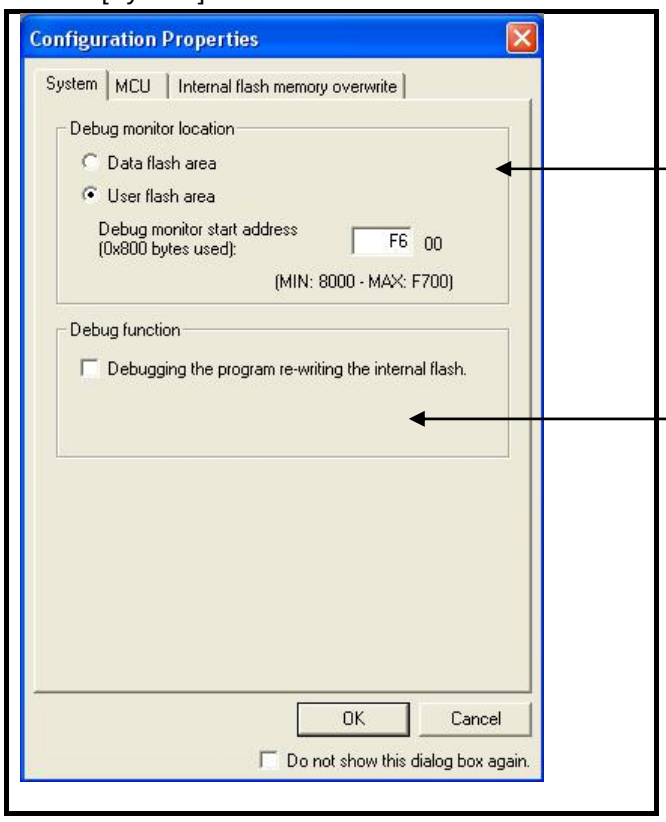
Error message	Cause	Solution
Command processing cannot be executed because the MODE pin on the user system is fixed in the low state. Check the state of the MODE pin.	Since the MODE pin is stuck low, the E1 or E20 emulator cannot communicate with the MCU.	Pull up the MODE pin. (See Figure 3.8.)
Command processing cannot be executed because the reset pin on the user system is in the low state. Check the state of the reset pin.	Since the RESET pin is stuck low, the E1 or E20 emulator cannot communicate with the MCU.	Check the RESET pin circuit. (See Figure 3.9.)
The target MCU and the selected device do not match. Check the device name specified at debugger startup again.	The MCU type name selected in the [Initial Settings] dialog does not agree with the MCU actually connected to the E1 or E20 emulator.	Change the MCU type name selected in the [Initial Settings] dialog box to make it agree with the MCU actually connected to the E1 or E20 emulator.
Could not activate the target MCU. There may be a problem in the pin processing of the MCU etc.	The E1 or E20 emulator and the MCU are connected incorrectly. Or the connection does not conform to the recommended circuit. If the connection differs from the one shown in the recommended circuit, communication may not be performed normally.	Check again the connection between the E1 or E20 emulator and the MCU. (See Figure 3.7.)
	The MCU's operating voltage does not conform to the designated voltage.	Refer to the hardware manual of the MCU.
	The MCU is mounted in place using an IC socket but is imperfectly contacted.	Check again to see if the IC socket on the user system matches the package of the MCU used, and whether it is insufficiently screwed up and imperfectly contacted.
	Not all power supply pins of the E1 or E20 emulator are connected.	See "3.4 (3) Other pins" on page 18.
The power voltage has exceeded 5.9V. Please check the user system setting.	The voltage being supplied to the user system may be greater than the defined limit.	Check the user-system settings for the power supply voltage.

4.13 [Configuration Properties] dialog box

The [Configuration Properties] dialog box is used to make settings related to the emulator and debugging functions. This dialog box appears after the [Initial Settings] dialog box.
The settings remain valid the next time the emulator is booted up. Clicking the [OK] button on the tabs completes emulator debugger setting and the debugger is booted up.
The [Configuration Properties] dialog box can be displayed again after the debugger is launched. Select Menu -> [Setup] -> [Emulator] -> [System...] to open the dialog box.

Note:
Unsupported options are grayed out or not displayed depending on the selected types of the MCU.

4.13.1 [System] tab



[Debug monitor location]

Select the area in which the debug monitor is located. Specify the address that will not be used by the user system.

- [Debug monitor location]
  Specify the area in which the debug monitor is located either from [Data flash area] or [User flash area]. Specify 2k bytes that will not be used by the user system. [*1] [*2] [*5]

[Debug function]

Select the debug function.

- [Debugging the program re-writing the internal flash.]
  Select this checkbox when debugging in CPU rewrite mode (i. e., debugging a program that involves overwriting of the internal flash memory (program ROM area) of the MCU). [*3] [*4]

Figure 4.31 [Configuration Properties] Dialog Box ([System] Tab)

Notes:
[*1] Depending on the type of MCU, the available options may vary as described below.
- The [Data flash area] option cannot be selected.
- The options in the [Debug monitor location] group box may be displayed in gray because this setting is unnecessary.

[*2] When you have selected the [Data flash area] for the debug monitor location, use a 250000 bps or lower communication baud rate.

[*3] Memory accesses attempted while the user program is halted are always made to the internal cache of the E1 or E20 emulator. Actual access to the flash memory is executed before the user program restarts and immediately after it has stopped.
The content of the cache is updated immediately after the user program has stopped when this option is selected, but not updated when this option is unselected. In this case, the displayed information and the content of the MCU's internal ROM do not match. Therefore, be sure to check the checkbox for this option when you want to rewrite the internal flash memory during user program execution by using the "CPU rewrite mode".

[*4] S/W break cannot be used when [Debugging the program re-writing the internal flash.] is selected. Also, do not rewrite the internal ROM area in the [Memory] window, etc.

[*5]

4.13.2 [MCU] tab

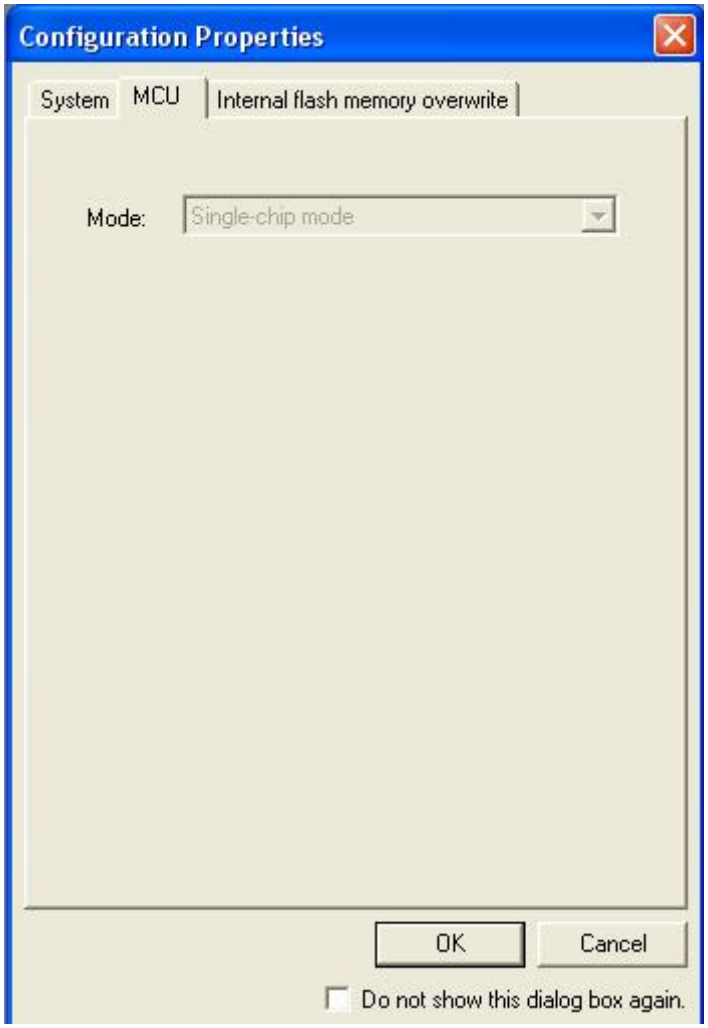


Figure 4.32 [Configuration Properties] Dialog Box（[MCU] Tab）

[Mode]

Select operating mode of the MCU. This setting is not necessary for your MCU.

4.13.3 [Internal flash memory overwrite] tab

The [Internal flash memory overwrite] tab allows you to specify whether or not individual blocks of flash memory should be overwritten. Settings made can be altered in the [Configuration Properties] dialog box even after the emulator debugger has been booted up.

All blocks of flash memory are automatically shown in the list according to selected device.

Settings made in this tab will be applied also in [writing the on-chip flash memory mode].

(1) Blocks whose checkboxes checked

1. Data are loaded for the selected blocks temporarily.
2. All blocks are erased.
3. When the user program is downloaded by a download command, etc., the data in 1. are merged (with the data in the device left intact).

(2) Blocks whose checkboxes unchecked

1. All blocks are erased.
2. The user program is downloaded by a download command, etc.
   Areas that have no data for the program to be downloaded remain set to the initial values of the flash memory.

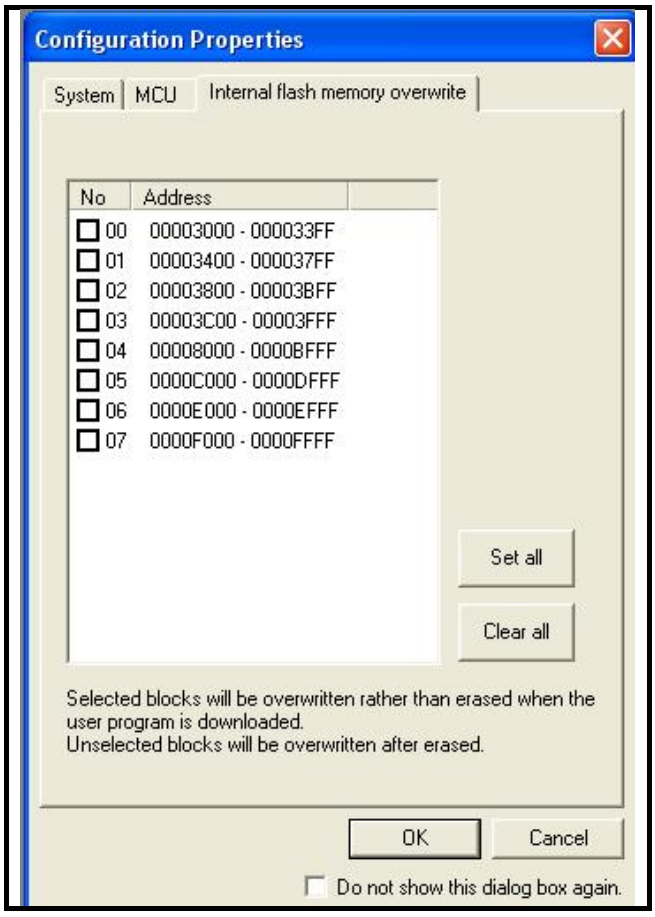


Figure 4.33 [Configuration Properties] Dialog Box ([Internal flash memory overwrite] Tab)

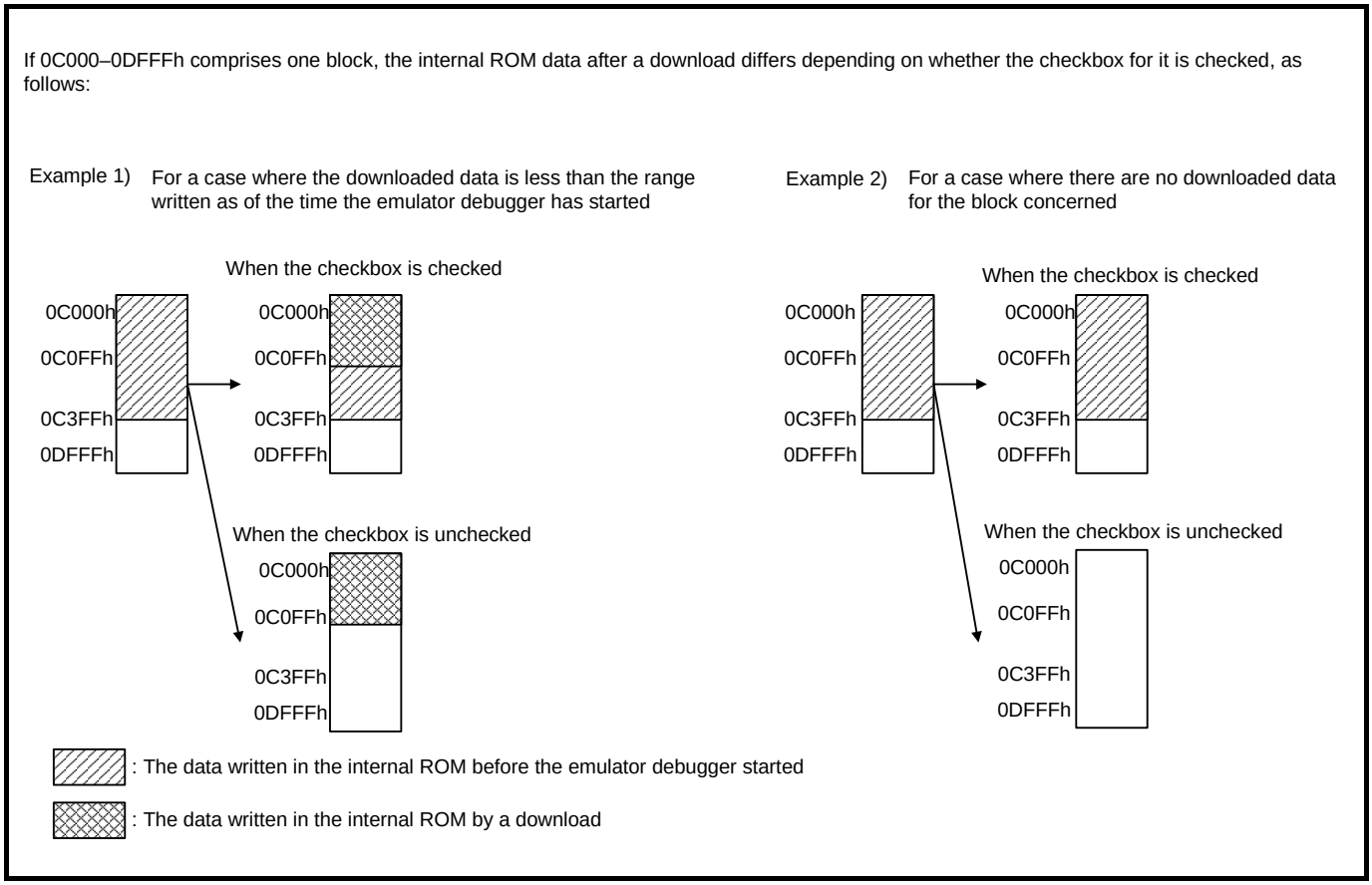


Figure 4.34 Difference in the Internal ROM Data after Download Depending on Whether the Checkbox is Checked

4.14 Starting up the R8C E1/E20 Emulator Debugger

When "Connected" is displayed in the [Output] window of the High-performance Embedded Workshop after clicking the [OK] button at the completion of the [Configuration Properties] dialog box setting, the emulator initiation is completed.

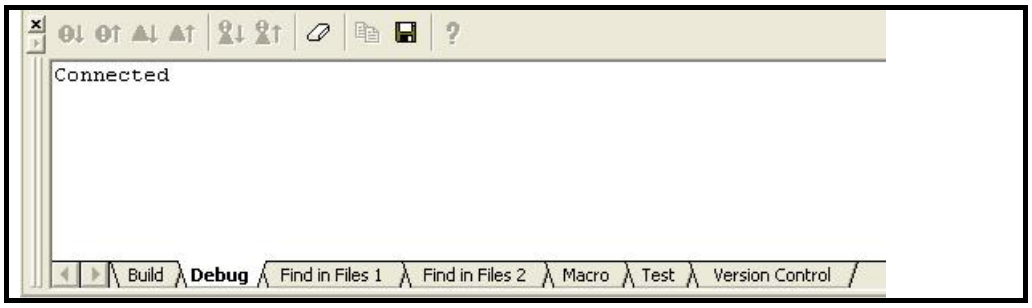


Figure 4.35 [Output] Window

Note:

[*1] When the user program has already been downloaded to the internal flash memory, source-level debugging is not possible because there is no debugging information on the user program after the emulator has been activated. To perform source-level debugging, download a load-module file that includes debugging information after the E1 or E20 emulator has been activated.

5. Debugging Functions

5.1 List of debugging functions

Table 5.1 shows the list of functions supported when the operation mode is [Writing the on-chip flash memory mode].
Table 5.2 shows the list of functions supported when the operation mode is [Debugging mode].

Table 5.1 List of R8C E1/E20 Emulator Debugger Functions ([Writing the on-chip flash memory mode])

Item	Support	Content
Program download	○	Subject: Internal ROM (settable in block units)
Online help	○	Describes the usage of each function or command syntax input from the command line window.
Command line function	○	Supports command input. Batch processing is enabled when a file is created by arranging commands in input order. Only the program downloading commands can be used.
Other function	×	

○: Supported, △: Alternative functions supported, ×: Not supported

Table 5.2 List of R8C E1/E20 Emulator Debugger Functions ([Debugging mode])

Item			Support	Content
Program download			○	Subject: Internal RAM and internal ROM (settable in block units)
Reset function			○	Reset from the emulator debugger when the user program has stopped
Memory access functions				
	Set/Display/Fill/Move/Compare/Search		○	Subject: SFR registers, internal RAM, internal ROM
	Realtime RAM monitor		△	Automatic memory content update function during user program execution available
	Single-line assembly		○	
	Disassembly display		○	
	CPU register access		○	Displaying/Setting of the CPU registers
	SFR register access		○	Displaying/Setting from the IO window
User program execution functions				
	Execution functions		○	Go/Free go/Go to cursor/Reset Go
	Step functions		○	(1) Single stepping (one step: one instruction) (2) Source-level step (one step: one-line source) (3) Step over (no break within subroutine) (4) Source-level step over (no break within subroutine) (5) Step out (when the PC points to a location within a subroutine, execution continues until it returns to the calling function)
	Break functions		○	S/W breakpoint (up to 256 points)
			○	Forced break
		On-chip break function	○	Pre-PC break (address match break), 8 points
			○	Data access break, 2 points Event A: Comparison with the address/data mask, and access condition (R, W, R/W) can be set. Event B: Comparison with the address mask, and access condition (R, W, R/W) can be set.
			○	Trace full break
		Cause of break	○	Cause of break display
	Trace functions	Internal trace	○	Built-in trace function (using the trace buffer in the MCU)
		External trace	×	No external trace function available
		Trace content	○	4 branch instructions (branch source/destination PC) or Up to 8 data cycles can be specified.
		Search/filter	×	
Performance measurement function			×	Uses a counter in the MCU to measure the number of cycles taken to execute a specified range of code.
Debug console function			×	Allows standard input and output for the user program.
Stack trace function			○	Shows which function called the function corresponding to the current PC value.
Start/Stop function			○	Facility for running specific routines in the user program immediately before the start of and immediately after the end of execution of the user program.
Command line function			○	Supports command input. Batch processing is enabled when a file is created by arranging commands in input order.
Execution time measurement function			○	Shows the time for which the user program has been executed.
Hot plug-in			×	
Coverage function			×	
Online help			○	Describes the usage of each function or command syntax input from the command line window.

○: Supported, △: Alternative functions supported, ×: Not supported

5.2 Downloading a program

Download the load module to be debugged.

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

The program is downloaded to the RAM or flash memory.

This function also downloads information required for source-level debugging such as debugging information.

Notes:

[*1] Before a program can be downloaded, you must have it registered as a load module in the High-performance Embedded Workshop.

[*2] If the specified access size is other than “1 byte,” odd portions of data cannot be downloaded correctly.
For the specification of [Access size] in the download function below, make sure the access size at download time is fixed to 1 (byte).
(1) Dialog opened by clicking on the [Download A New Module] menu
(2) Dialog opened from the [Memory] window’s [Load] menu

[*3] The vector area, ID code area and the OFS, OFS2 areas have their values rewritten by the emulator, so be aware that their checksums do not match (when the operation mode is [Debugging mode]).
For the reset vector addresses also, although downloaded values are displayed in the [Memory] window, etc., be aware that they are the emulator’s internal values in the cache, and that the values set in the actual flash memory are those that were specified in the emulator.

5.3 Opening a source file

5.3.1 Viewing the source code

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

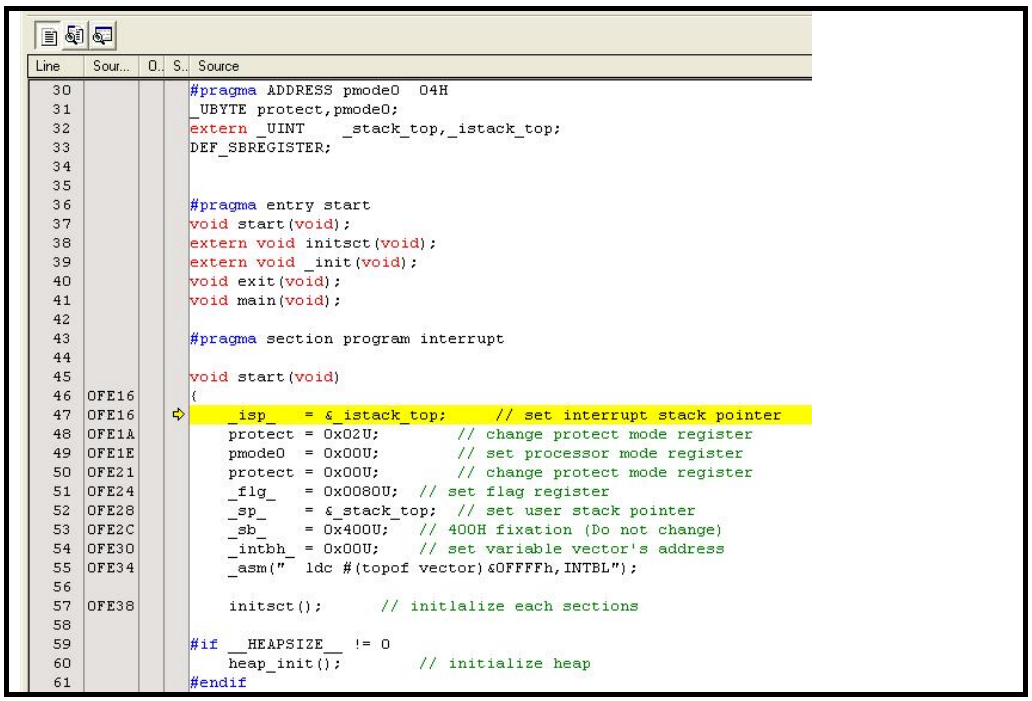


Figure 5.1 [Editor] Window

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

The first column ([Source Address] column): Address information

The second column ([On-Chip Breakpoint] column): On-chip breakpoint information

The third column ([S/W Breakpoint] column): PC, bookmark, and breakpoint information

[Source Address] column

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

[On-Chip Breakpoint] column

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

●: Indicates the address condition for a pre-PC breakpoint.

The number of address conditions that can be set for pre-PC breakpoints is the same as the overall number of events, which varies with the MCU.

Double-clicking on the [On-Chip Breakpoint] column produces a bitmap symbol as shown above in the corresponding line.

[S/W Breakpoint] column

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

: Bookmark

: S/W breakpoint

: Program counter (PC)

5.3.2 Switching off a column in all source files

(1) Right-click on the [Editor] window.

(2) Click the [Define Column Format…] menu item.

(3) The [Global Editor Column States] dialog box is displayed.

A checkbox indicates whether the column is enabled or not. If it is checked, the column is enabled.

If the check box is gray, the column is enabled in some files and disabled in others. Deselect the check box of a column you want to switch off.

(4) Click the [OK] button for the new column settings to take effect.

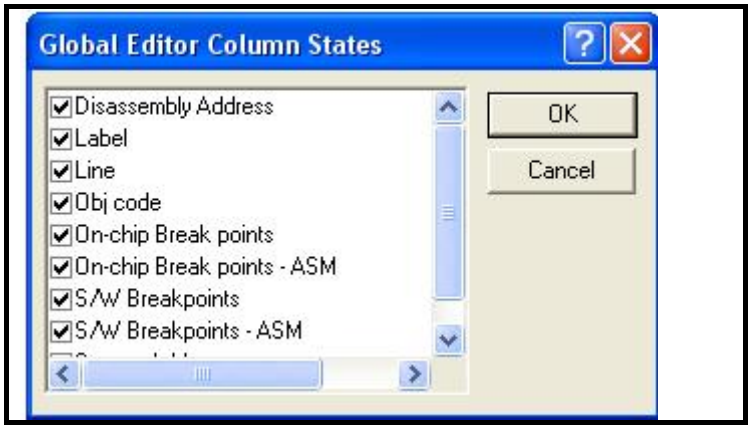


Figure 5.2 [Global Editor Column States] Dialog Box

5.3.3 Switching off a column in one source file

(1) Open the source file which contains the column you want to remove and right-click on the [Editor] window to open a popup menu.

(2) Click on the [Columns] menu item to display a cascaded menu item. The columns are displayed in this popup menu. If a column is enabled, it has a tick mark next to its name. Clicking on the entry will toggle whether the column is displayed or not.

5.3.4 Viewing the assembly-language code

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

If you do not have a source file, but want to view code in the assembly-language level, either choose [View -> Disassembly] or click the [Disassembly] toolbar button ( ).
The [Disassembly] window opens at the current PC location and shows [Address] and [Code] (optional) which show the disassembled mnemonics (with labels when available).

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

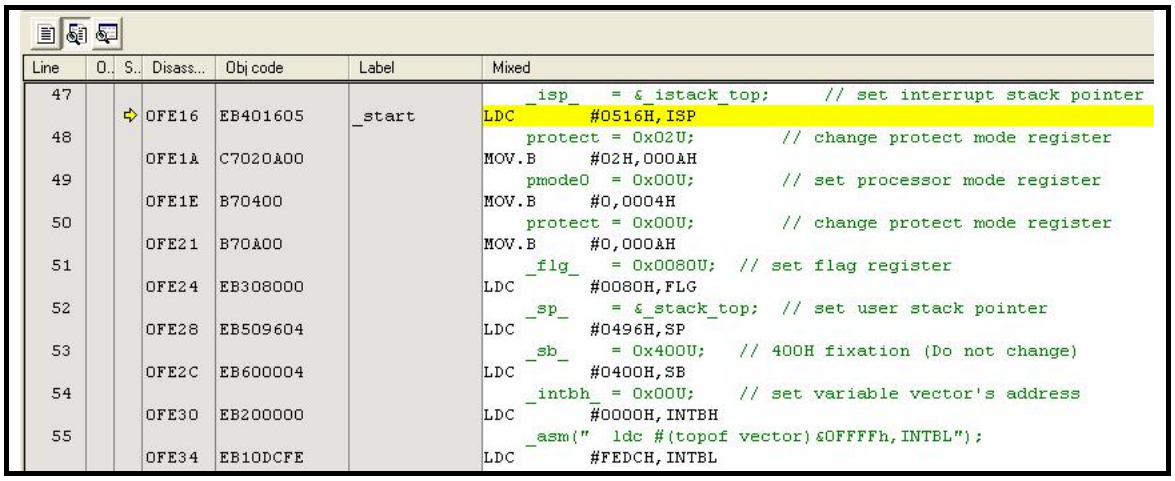


Figure 5.3 [Disassembly] Window

5.3.5 Modifying the assembly-language code

You can modify the assembly-language code by double-clicking on the instruction that you want to change.

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

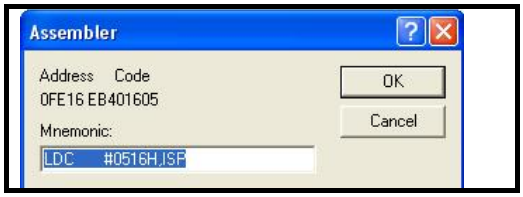


Figure 5.4 [Assembler] Dialog Box

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

5.4 Memory access functions

The E1 and E20 emulators have the following memory access functions.

5.4.1 Memory read/write function [*1]

The memory contents are displayed in the [Memory…] window.

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

Enter the start address to be displayed in the [Display Address] edit box to open the [Memory] window.

(1) User program downloading function

A load module registered in the workspace can be downloaded. For details, see “5.2 Downloading a program” on page 48.

(2) Memory data uploading function

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

(3) Memory data downloading function

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

(4) Displaying variables

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

(5) Automatic updating of data in memory [*2]

The automatic memory update function permits you to inspect in the memory window etc. the memory content of the MCU and how it is accessed.

The display of data in memory in the [Memory] window can be automatically updated while the user program is running. By making use of the MCU’s built-in debug functions, it helps to minimize the idle time during which the user program is halted. Memory locations whose values have changed from the previous session are displayed in red. The interval for updating the display is also specifiable.

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

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

5.4.2 Other memory operation functions [*3] [*4]

Other functions are as follows (for details, refer to the online help.):

- Memory fill
- Memory move
- Memory compare
- Memory search
- Displaying label and variable names and their contents

Notes:

[*1] Only the address range displayed in the [Memory] window is referenced. Since there is no cache in the emulator except the internal ROM area, the target MCU is always accessed.
If the memory is written in the [Memory] window, the entire address range displayed in the [Memory] window will be accessed for updating the window. When the [Memory] window is not to be updated, select [Lock Refresh] from the popup menu opened by right-clicking in the window.

[*2] If a reset such as a S/W reset or hardware reset (RESET#) is issued while a user program is running, the memory access occurred at that point cannot be set or changed correctly.

[*3] Do not specify 4 bytes or 8 bytes as the data size for the move, compare, and search functions.
If these sizes are specified, the emulator will not operate correctly.
Also, these functions have a maximum range of 16 Mbytes.
These functions are enabled only for the internal RAM and internal ROM areas.

[*4] These functions do not work normally for the area the debug monitor is located.

5.4.3 Notes on accessing the SFR areas

You can reference or set the contents of the SFR areas in the [Memory] window or the [IO] window. Note the followings.

(1) When accessing the special registers

You may not be able to access some special registers successfully during the user program halt. [*1] [*2]

These registers include:

- Access prohibited addresses
- Registers for which access order is specified from high-order byte to low-order byte
- Registers that can be accessed only by a specific instruction
- Registers whose bus width specification does not match the bus width set in the [Memory] window
- Registers that can be accessed on conditions (one of which is that fOCO-F must be faster than the CPU clock to access the register)

Notes:

[*1] Follow the instructions in the hardware manual of the target MCU to access to the SFR areas.

[*2] When the user program halts, the following registers may not be overwritten in some cases (applicable for R8C/35C, R8C/36C and R8C/38C only).

Timer RC counter: 0126h, 0127h
Timer RD counter: 0156h, 0157h
Timer RG counter: 0176h, 0177h

(2) SFR registers that can be accessed from the [IO] window [*1] [*2]

By default, an I/O file is displayed in the [IO] window according to the MCU group selected in the [Initial Settings] dialog box. As shown in the table below, an I/O file for the MCU group which has the largest pin count is displayed.

Table 5.3 I/O Files Displayed by Default in the [IO] Window

Types	[Initial Settings] dialog box display	I/O files shown by default
R8C/3xC	R8C/3xC(48pins or less) Group	R8C/35C
	R8C/35C(52pin) Group	R8C/35C(16KB, 24KB, 32KB Internal ROM)
		R8C/38C(48KB, 64KB, 96KB, 128KB Internal ROM)
	R8C/3xC(64pins or more) Group	R8C/38C
R8C/3xM	R8C/3xM(48pins or less) Group	R8C/35M
	R8C/35M(52pin) Group	R8C/35M(16KB, 24KB, 32KB Internal ROM)
		R8C/38M(48KB, 64KB, 96KB, 128KB Internal ROM)
	R8C/3xM(64pins or more) Group	R8C/38M
R8C/3xT	R8C/3xT Group	R8C/3JT
R8C/3NT	R8C/3NT Group	R8C/3NT
R8C/3MQ	R8C/3MQ Group	R8C/3MQ
R8C/3xW	R8C/3xWXYZ Group	R8C/38W
R8C/3xX	R8C/3xWXYZ Group	R8C/38X
R8C/3xY	R8C/3xWXYZ Group	R8C/38Y
R8C/3xZ	R8C/3xWXYZ Group	R8C/38Z
R8C/3xGHPR	R8C/3xGHPR Group	R8C/34P (not attached)
R8C/3xU,3xK	R8C/3xU, 3xK Group	R8C/34K
R8C/L3xC	R8C/L3xC Group	R8C/L3AC
R8C/L3xM	R8C/L3xM Group	R8C/L3AM
R8C/LA3A,LA5A	R8C/LA3A, LA5A Group	R8C/LA5A (not attached)
R8C/LA6A,LA8A	R8C/LA6A,LA8A Group	R8C/LA8A
R8C/LAPS	R8C/LAPS Group	R8C/LAPS (not attached)

Notes:

[*1] You can change the I/O file to that of your target MCU installed with the debugger.
You can edit the I/O file (xxxx.io) to add or delete registers to the set for display in the [IO] window.

For the contents to be included in the I/O file (xxxx.io), refer to reference 6, I/O File Format, in the High-performance Embedded Workshop User's Manual.
The following directory contains the I/O file (xxxx.io).
<High-performance Embedded Workshop folder>:
\Tools\Renesas\DebugComp\Platform\E1E20\E20R8C\IOFiles

[*2] The I/O file attached to the R8C E1/E20 Emulator Debugger was created based on the SFR header file which was already made accessible to the public when the debugger was developed.
As such, it is possible that the target SFR header file, if updated due to incorrect entry, etc., may not match the I/O file. You can edit the I/O file as shown above.

5.5 Outline of break functions

The R8C E1/E20 Emulator Debugger provides three break functions: forced break, S/W break and on-chip break. The break functions can be set singly or multiply.

The list of break functions supported for the MCUs in this user's manual is shown in Table 5.4.

5.5.1 Forced break

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

5.5.2 S/W break (software break)

This function breaks the program by rewriting the instruction of the specified address to an instruction (BRK instruction) dedicated to the debugger. [*1]

Since the op-code at the specified address is written to the instruction dedicated to the debugger, when a S/W breakpoint is set, a write to on-chip memory (flash memory and RAM) will occur. (Similarly, removing a S/W breakpoint involves a write to memory.)

For details on the S/W break function, see "5.6 Using S/W breakpoints" on page 56.

5.5.3 On-chip break

With the MCUs in this user's manual, the following three on-chip break functions are available: pre-PC break, data access break and trace full break.

For details on the on-chip break function, see "5.7 Outline of on-chip break functions" on page 59.

- Pre-PC break (Address match break)

  This function breaks the program immediately before a specified address instruction is executed. It can be realized using the address match interrupt of the MCU. An event in this function is called a "PC event".
- Data access break

  This function breaks the program when a specified event is encountered. You can combine two points of the data access event. An event in this function is called a "DA event".
- Trace full break

  This function breaks the program when the trace buffer is filled.

Table 5.4 Break Functions

Event type			Number of points that can be set (max.)	Number of points that can be registered (max.)	Break condition	Flash memory rewrite	Can a breakpoint be set while the program is running?
Forced break					No	No	Yes
S/W break [*2]			256		Specified address	Yes	No [*3]
On-chip break	Pre-PC break (address match break) [*2]		8	32	Specified address	No	Yes
	Data access break	EventA	1	8	Specified address & specified data	No	No
		EventB	1	8	Specified address	No	No
	Trace full break				Trace buffer full	No	No

Notes:

[*1] When the instruction of the address where a S/W breakpoint was placed is displayed in the [Memory] window, what is displayed is not "BRK" instruction (00h).

[*2] When execution is restarted from the address where it stopped at a breakpoint (S/W breakpoints or pre-PC breakpoints), the actual instruction at the address must be executed as a single step before further execution continues. Operation is thus not in real time.

[*3] You can set or remove S/W breakpoints in the internal RAM.

5.6 Using S/W breakpoints

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

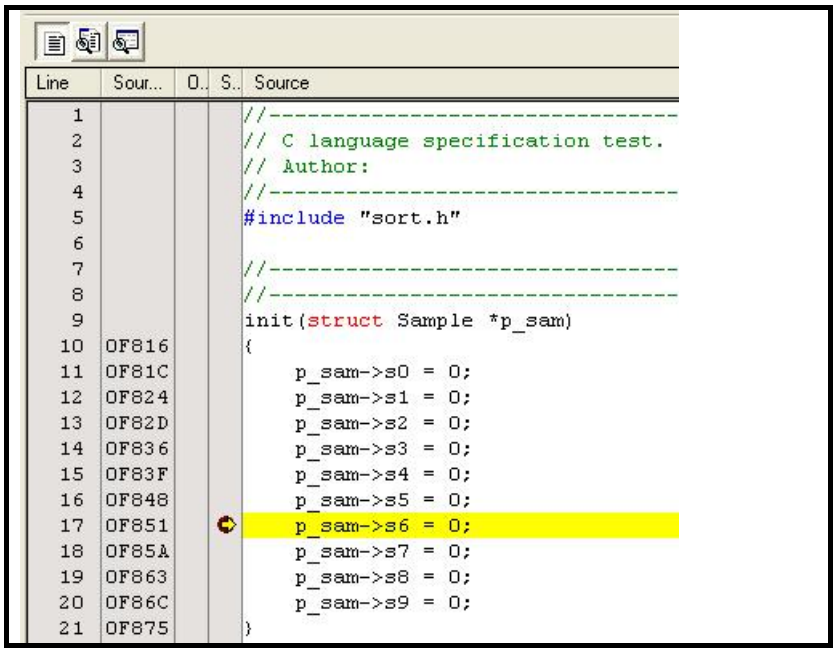


Figure 5.5 Stop at S/W Break

Notes:

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

[*2] If multiple S/W breakpoints are set, the program breaks at any one of those breakpoints reached.

[*3] S/W breaks cannot be specified in other than the MCU's internal RAM and internal ROM areas.

[*4] When you change a program in the [Editor] window and download that program for the MCU, the addresses at which you've set S/W breaks may not be corrected normally, depending on changed contents. For this reason, it is possible that the addresses at which S/W breaks are set will shift.
After the user program is downloaded, check again the addresses at which S/W breaks are set.

[*5] When the user program is executed from the address at which a S/W break is set, the relevant address (i.e., the user program start address) is stepped through before the user program is run. Therefore, the program behaves differently, with regard to the execution time and interrupts, than it would when no S/W breaks are set.

[*6] If you do not remove the S/W breakpoint before disconnecting the emulator from the target MCU and reconnect it, the S/W breakpoint will remain in the following case:

- The flash memory block that contains the S/W breakpoint was checked (so that the data in the flash memory remains) in the [Configuration Properties] dialog box at debugger startup.

[*7] When you have S/W breaks set in the internal RAM of the MCU, do not rewrite the content of their addresses there in the [Memory] Window, etc.

5.6.1 Adding/removing S/W breakpoints

Follow one of the following methods to add or remove S/W breakpoints

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

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

1. Check to see that the [Editor] or the [Disassembly] window that is currently open includes the position at which you want to set a S/W breakpoint.

2. In the S/W Breakpoints column, double-click the line where you want the program to stop.
   Or you use the method described below to set a breakpoint.
   - Select [Toggle Breakpoint] from the popup menu or press the F9 key on the keyboard.

3. When a S/W breakpoint is set, a red circle [ ] is displayed at the corresponding position in the S/W Breakpoints column of the Editor or the [Disassembly] window.
   Double-clicking one more time removes the breakpoint.

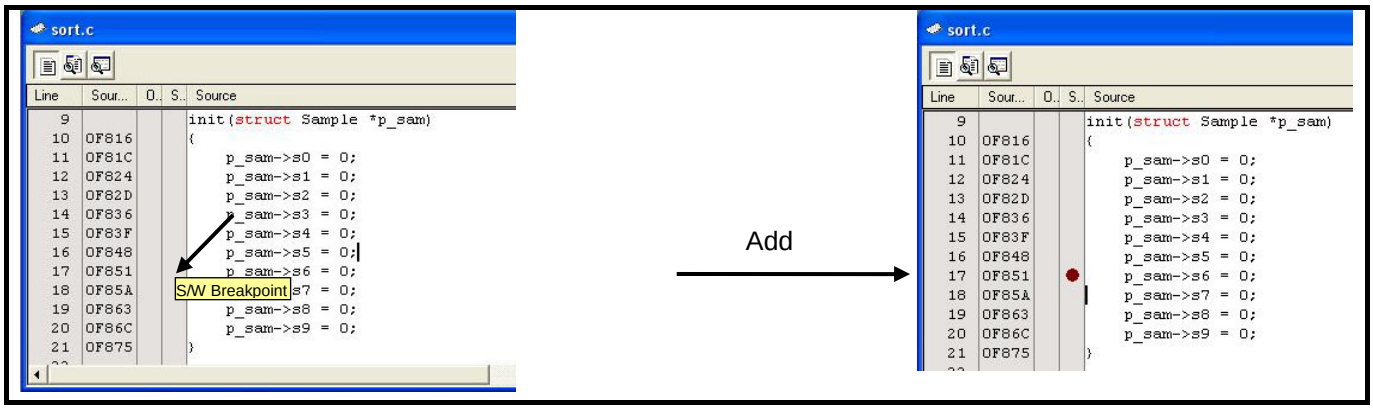


Figure 5.6 Adding or Removing a S/W Breakpoint in the [Editor] Window

(2) From the [Breakpoints] dialog box

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

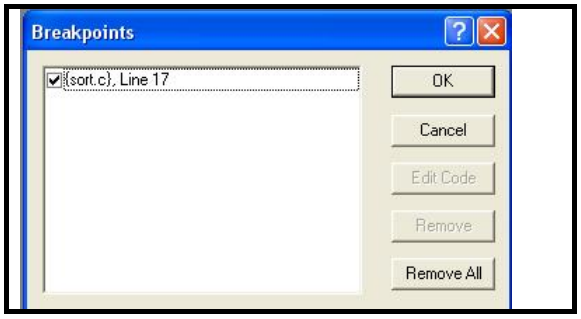


Figure 5.7 Enabling or Disabling Breakpoints in [Breakpoints] Dialog Box

(3) From the command line

Refer to the help file.

5.6.2 Enabling/disabling S/W breakpoints

Follow one of the following methods to enable or disable S/W breakpoints.

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

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

1. Place the cursor at the line where a S/W breakpoint exists and then select [Enable/Disable Breakpoint] from the pop-up menu. Or press the Ctrl and F9 keys together.
2. The S/W breakpoint is enabled or disabled alternately.

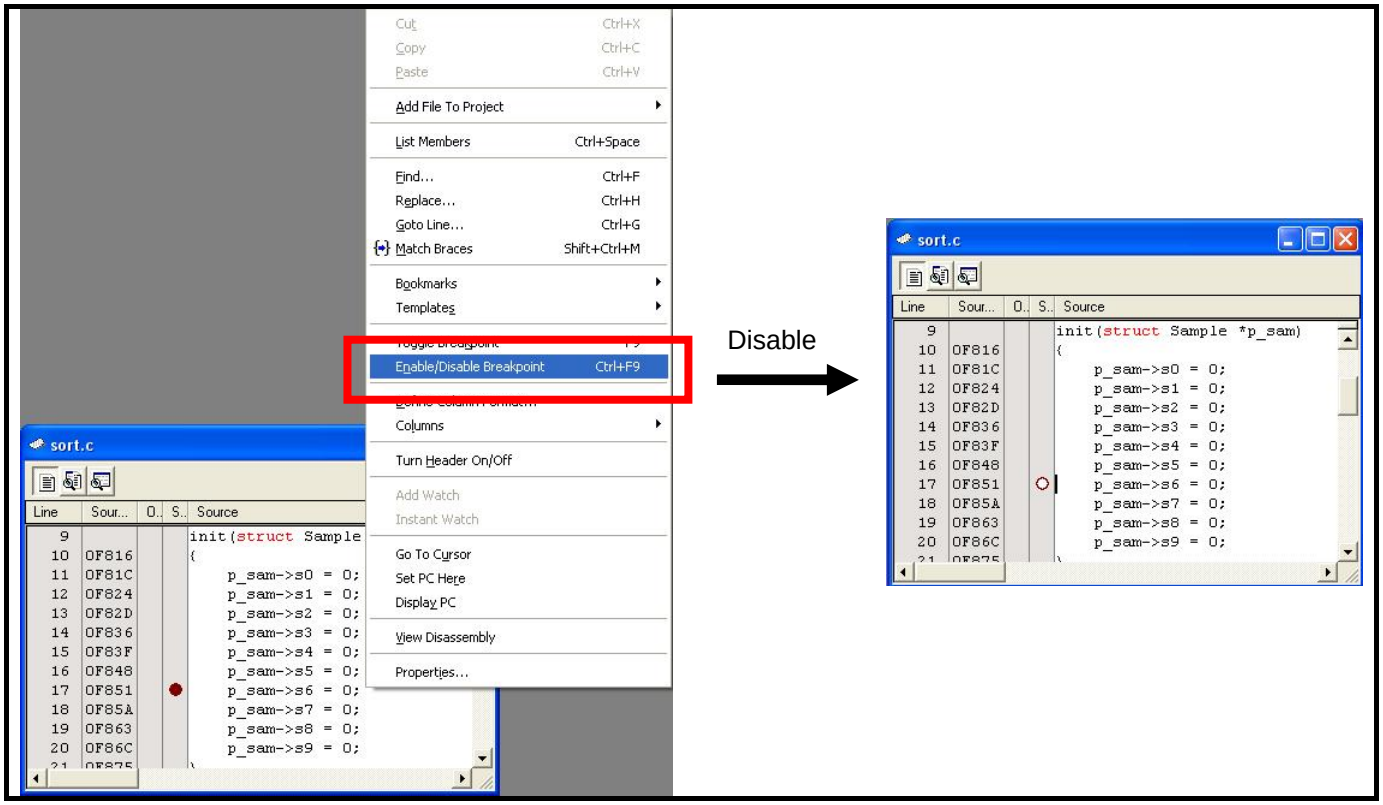


Figure 5.8 [Editor] Window and Pop-up Menu

(2) From the [Breakpoints] dialog box

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

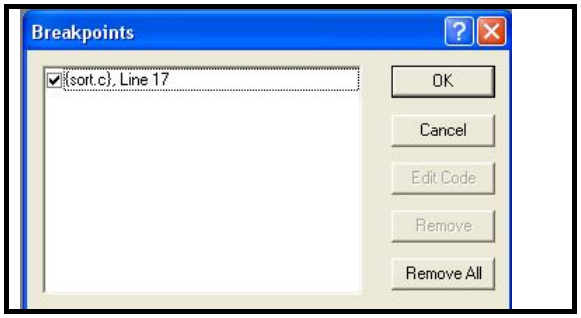


Figure 5.9 Enabling or Disabling Breakpoints in [Breakpoints] Dialog Box

(3) From the command line

Refer to the help file.

5.7 Outline of on-chip break functions

5.7.1 [On-Chip Event] dialog box

This window is provided for setting on-chip event break conditions and trace conditions.

Select [View -> Event -> On-Chip Event] to bring up the [On-Chip Event] dialog box.

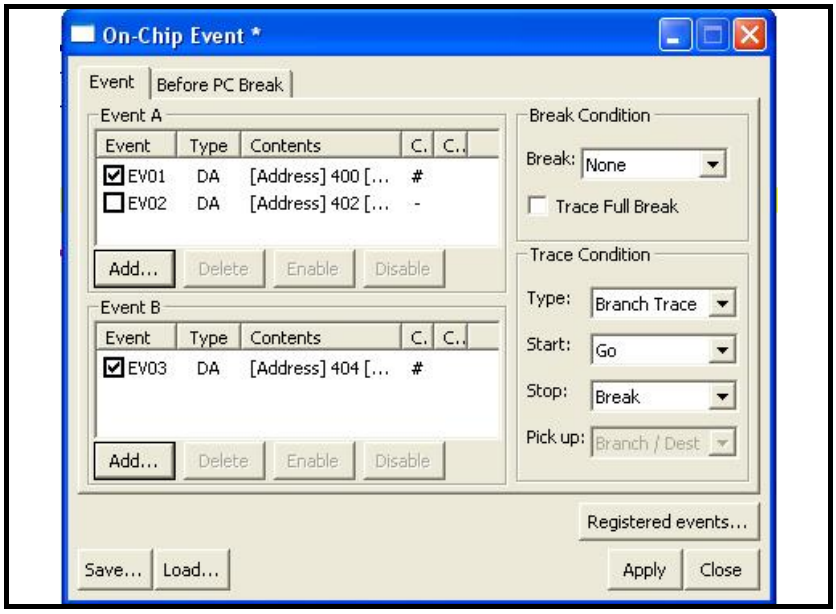


Figure 5.10 [On-Chip Event] Dialog Box

(1) Condition setting tabs

In this dialog box, you can add, delete, enable, or disable the following two kinds of events.

Tab	Content	Remark
Event	Sets Event A, B-related settings and trace conditions	Default display when the window is opened
Before PC Break	Sets pre-PC breaks using PC events	

(2) Registered events…

Displays the currently registered events in list form. See “5.10 Registering events” on page 73 for details.

(3) Save…

Saves the contents of this dialog box that you’ve set in a file.

(4) Load…

Loads the set contents of the [On-Chip Event] dialog box from a file in which they were saved. When normally loaded, the set contents of this dialog box are discarded and replaced with the contents of the file.

(5) Apply

Clicking the [Apply] button causes the contents set in this dialog box to be set in the E1 or E20 emulator. If any item of the set contents is changed in the [On-Chip Event] dialog box or the set contents are loaded from a file using the [Load] button, the window name on the title bar is marked with an asterisk (*) at its side. This asterisk disappears when you click the [Apply] button.

(6) Close

Closes this dialog box. If you did not click the [Apply] button after changing the set contents in this dialog box, a message box is displayed for your confirmation. The following results depending on what you select in this message box.

- When you select [Yes]: The set contents of this dialog box are applied before closing this dialog box.
- When you select [No]: The set contents of this dialog box are discarded before closing this dialog box.

5.7.2 [Event] tab

Use this event tab to add, delete, enable or disable Event A and Event B, or set the break and trace conditions for these events. Figure 5.11 shows the [On-Chip Event] dialog box, Table 5.5 shows the settings of Event A and Event B, and Table 5.6 shows the settings of break conditions, and Table 5.7 shows the settings of trace conditions.

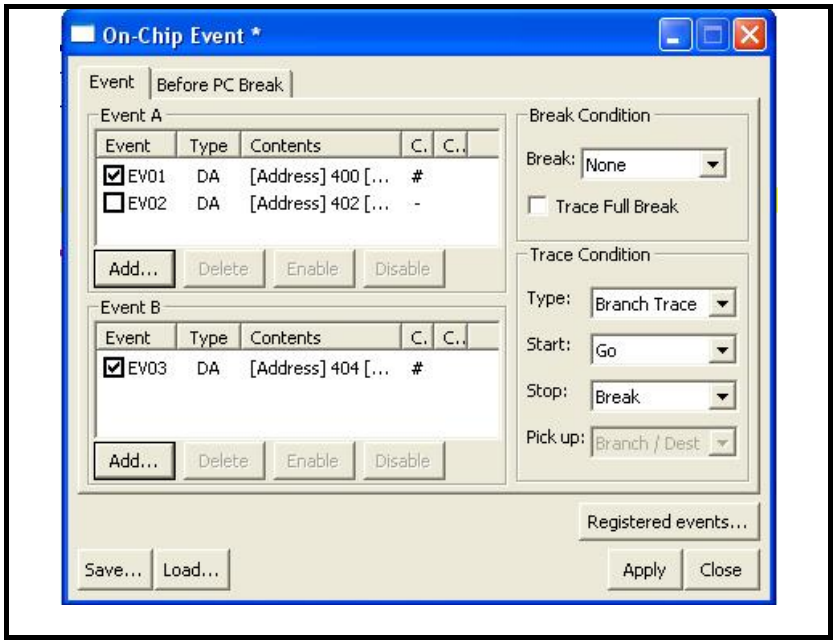


Figure 5.11 [On-Chip Event] Dialog Box ([Event] Tab)

Table 5.5 Settings of Event A and Event B

Item	Content		
Display content	When you double-click a registered event, the [Event] dialog box is displayed, allowing you to inspect or set the event content. By dropping text of a function name, etc. into the list you can register a new one.		
	Item	Content	Remark
	Check column	Checking the checkbox enables the event.	Event A and Event B can each be enabled one event at a time. If the checkbox for a second event is checked, the first event is disabled and the second event is enabled.
	Event	Displays “EVxx”	Event A and Event B can have up to a total of 8 events registered, respectively.
	Type	Displays DA (data access event).	
	Contents	Displays the set contents of events.	
	Channel	Displays “0" for Event A, or “1” for Event B.	
	Comments	Displays comments for each event.	
[Add] button	Brings up the [Event] dialog box, allowing you to add event contents. A newly added event is enabled. If there are 8 registered events, this button is grayed out and you cannot register a new one.		
[Delete] button	Deletes all of the selected events from the list. (Instead of using the [Delete] button, you can use the Ctrl + Del keys to delete.)		
[Enable] button	Enables the event selected in the list. If there are multiple selected events, this button is grayed out.		
[Disable] button	Disables the event selected in the list.		

Table 5.6 Settings of Break Conditions

Option	Description
Break	Select a break condition. - None: No break by event (default) - Event A: Breaks the program when the Event A is encountered. - Event A or B: Breaks the program when either the Event A or Event B is encountered. - Event A and B: Breaks the program when both the Event A and Event B are encountered. - Event B->A: Breaks the program when an event is encountered in the order of the Event B and Event A.
Trace full break	Check it to break the program when the trace buffer is filled. This option is unchecked by default (trace full break in disabled). Trace full break can be set with the break condition by event.

Table 5.7 Settings of Trace Conditions

Option	Description	
Type	Select a trace type from the following:	
	Branch trace (default):	Branch trace Traces the branch source address and branch destination address of branch processing that occurred during the execution of the user program.
	Data access:	Data access Traces the data access information of events that occurred during the execution of the user program.
Start	Select a start condition for the trace measurement from the following options.	
	Go (default):	Starts a measurement when starting executing the user program.
	Event A	Starts a measurement when the Event A is encountered.
	Event A or B:	Starts a measurement when either the Event A or Event B is encountered.
	Event A and B:	Starts a measurement when both the Event A and Event B are encountered.
	Event B->A:	Starts a measurement when an event is encountered in the order of the Event B and Event A.
Stop	Select a stop condition for the trace measurement from the following options.	
	Break (default):	Stops a measurement when stopping executing the user program.
	Trace FULL:	Stops a measurement when the trace data is filled.
	Event A	Stops a measurement when the Event A is encountered.
	Event A or B:	Stops a measurement when either the Event A or Event B is encountered.
	Event A and B:	Stops a measurement when both the Event A and Event B are encountered.
	Event B->A:	Stops a measurement when an event is encountered in the order of the Event B and Event A.
Pick up	Select an event to record when the trace type is [Data access] from the following:	
	Event A (default):	Records only data access which encounters the condition of the Event A.
	Event A or B:	Records only data access which encounters the condition of either the Event A or Event B.

Notes:

[*1] Note on the trace start condition
When setting an event (other than [Go]) for the trace start condition, a data when the event is encountered is not recorded to the trace data. The data of the event which is encountered the next time is recorded.

[*2] Notes on the trace stop condition
When the trace start and trace stop conditions occur simultaneously, the trace stop condition becomes invalid.
When setting other than [Break] for the trace stop condition, the display contents of the [Trace] window will not be updated until the user program stops even after a trace stop condition is encountered.

[*3] If a reset such as a hardware reset (RESET#) or watchdog timer reset is issued while a user program is running, the encountered event records and trace records made until the reset is issued will be cleared.

[*4] Only access via the CPU bus is traceable. No record of access by the DTC is included in trace data.

5.7.3 [Before PC Break] tab

Use this event tab to setup a pre-PC break. Figure 5.12 shows the [Before PC Break] tab of the [On-Chip Event] dialog box, and Table 5.8 shows the tab settings.

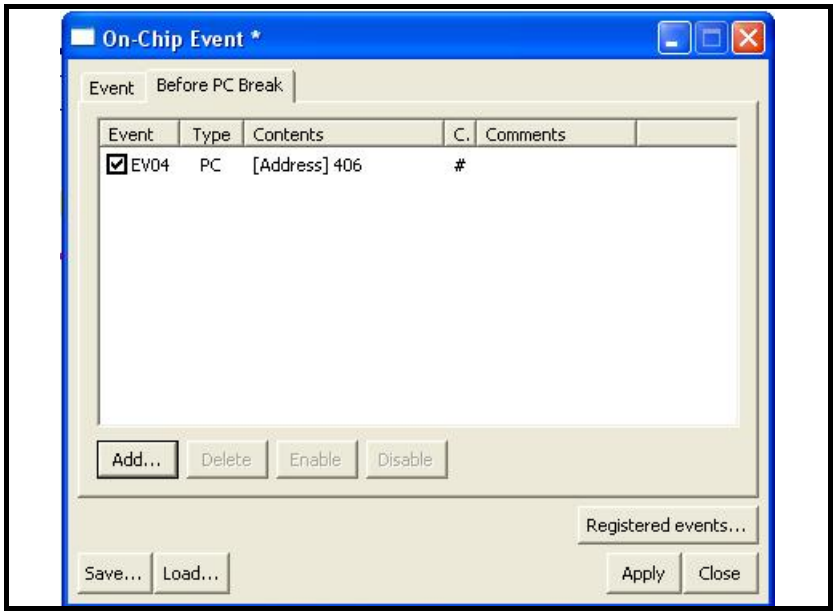


Figure 5.12 [On-Chip Event] Dialog Box ([Before PC Break] Tab)

Table 5.8 [On-Chip Event] Dialog Box ([Before PC Break] Tab)

Item	Content		
Display content	When you double-click a registered event, the [Event] dialog box is displayed, allowing you to inspect or set the event content. By dropping text of a function name, etc. into the list you can register a new one.		
	Item	Content	Remark
	Check column	Checking the checkbox enables the event.	Up to 8 events can be enabled.
	Event	Displays “EVxx”	Up to a total of 32 events can be registered.
	Type	Displays PC (address match event).	
	Contents	Displays the set contents of events.	
	Channel	Displays one of 0 to 7. [*1]	
	Comments	Displays comments for each event.	
[Add] button	Brings up the [Event] dialog box, allowing you to add event contents. [*2] If there are 32 registered events, this button is grayed out and you cannot register a new one.		
[Delete] button	Deletes all of the selected events from the list. (Instead of using the [Delete] button, you can use the Ctrl + Del keys to delete.)		
[Enable] button	Enables all of the events selected in the list.		
[Disable] button	Disables the event selected in the list.		

Notes:
[*1] Displays “#” until the [Apply] button is clicked.
[*2] For up to 7 events, a newly added event is enabled.
If there are already 8 registered events, a message “PC event is full.” will be displayed, and the newly added event is registered in a disabled state.

5.7.4 [Event] dialog box

This window is used to set the contents of PC events (address match breaks) and DA events (Event A, B breaks).
This window is displayed when you set events from the [Event] tab or [Before PC Break] tab of the [On-Chip Event] dialog box or register events in the [Registered Events] dialog box.

a) Event type

The event type [Address Match] (for PC events) or [Data Access] (for DA events) can be selected.
The set items switch over when the selected event type is changed.

The event type [Data Access] is selected when this dialog box is opened from the [Event] tab of the [On-Chip Event] dialog box or [Address Match] is selected when this dialog box is opened from the [Before PC Break] tab.
When using the [Registered Events] dialog box to register an event, the settings can be registered with either event type. [*1]

b) Condition setting tabs

This dialog box allows you to set event conditions on the [Event] tab and set comments on events on the [Comment] tab.

(1) [Event] dialog box (event type: address match)

When [Address Match] is selected for the event type, this dialog is displayed as shown in Figure 5.13.

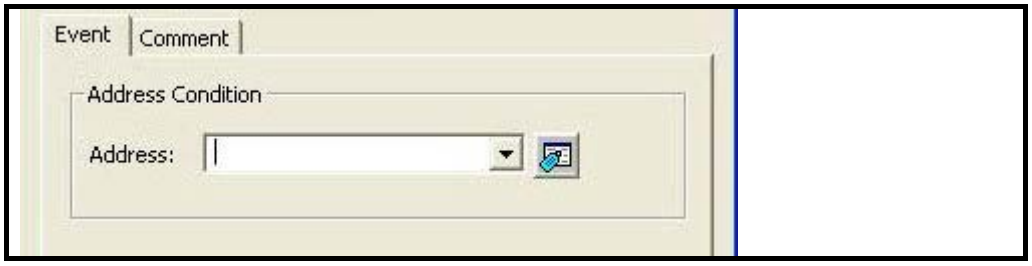


Figure 5.13 [Event] Dialog Box (When [Address Match] is Selected)

Note:
[*1] No events can be registered exceeding the maximum registrable number of events.

5.7.5 [Event] dialog box (event type: data access)

When [Data Access] is selected for the event type, this dialog is displayed as shown in Figure 5.14. The display contents are shown in Table 5.9.

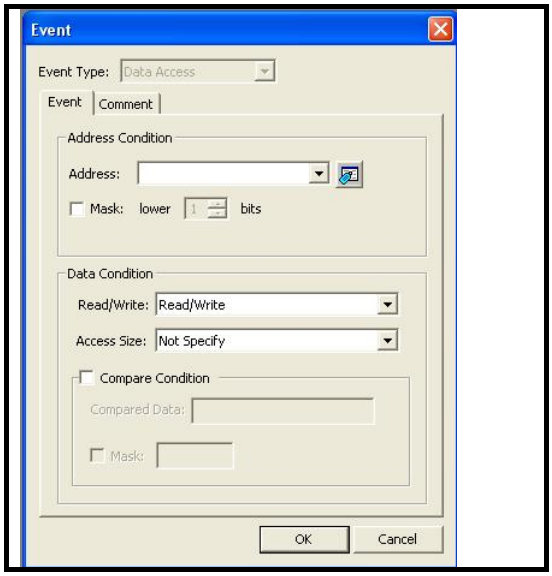


Figure 5.14 [Event] Dialog Box (when [Data Access] is Selected)

Table 5.9 Display Contents of the [Event] Dialog Box

Item		Description
Address condition		
	Address	Specify an address to detect the data access.
	Mask	• Checking this checkbox makes it possible to mask an address. This checkbox is, by default, unchecked (not masked). • Specify the bit number to set the address mask. The specified lower bits of the specified address are masked. Specifiable in the range 1 to 15.
Data Condition		
	Read/Write	Select an access type from the following: - Read/Write: Read or Write - Read - Write
	Access Size	Select one from Byte, Word or Not specify for the Access Size. If a data access which does not match the specified access size occurs, the event is not encountered. When specifying Word for the Access Size, specify the even address for the Address item.
	Compare Condition [*1]	• Checking this checkbox makes it possible to specify data comparison. This checkbox is, by default, unchecked (no data comparison). [*2]
	Compare Data	Specify the data value to compare. Access size (when Byte specified): 00-FFh Access size (when Word specified): 0000-FFFFh
	Mask	• Checking this checkbox makes it possible to mask data. This checkbox is, by default, unchecked (not masked). • Data is not masked if the mask value is blank.

Notes:

[*1] Conditions here can be set for only Event A. (For Event B, although possible to enter conditions here, they cannot be registered.)
No events can be registered exceeding the maximum registrable number of events.
You cannot register an Event B which includes any Compare Condition settings.

[*2] If the [Compare Data] box contains no data in it while this checkbox is checked, this dialog box is not closed even when you click the [OK] button.

5.7.6 Notes on the event settings

When setting the Event A or Event B, set the address, access size and access type referring to Table 5.10 below.

Table 5.10 Availability of the Event Setting

Event setting condition	Availability of event setting	Example of Event Setting dialog box
Byte read to even address	Available	Address: 400h Access size: Byte Access type: Read or Read/Write
Byte write to even address	Available	Address: 400h Access size: Byte Access type: Write or Read/Write
Word read to even address	Available	Address: 400h Access size: WORD Access type: Read or Read/Write
Word write to even address	Available	Address: 400h Access size: WORD Access type: Write or Read/Write
Byte read to odd address	Available	Address: 401h Access size: Byte Access type: Read or Read/Write
Byte write to odd address	Available	Address: 401h Access size: Byte Access type: Write or Read/Write
Word read to odd address	Available	Address: 401h Access size: Byte [*1] Access type: Read or Read/Write
Word write to odd address	Available	Address: 401h Access size: Byte [*1] Access type: Write or Read/Write

Notes:

[*1] To detect wordwise accesses to odd addresses, specify [Byte] for the access size.
Note that the data comparable on this condition is one lower byte.

[*2] Note on setting the Event A
When setting an event for the Event A, do not specify a mask for an address and data simultaneously. If you mask them simultaneously, an event will not be encountered.

[*3] Note on setting an event
Do not specify the following addresses as the address of the event. Otherwise, an unauthorized break may occur.
- Address in the interrupt vector table
- Address set in the interrupt vector table (interrupt routine start address)
- Branch destination address of the branch instruction

Both fixed vector table and variable vector table are included with the interrupt vector table above.

5.8 Adding on-chip events (DA events)

Use one of the following to set up an event.

- [On-Chip Event] dialog box
- Drag-and-drop from another window (for add only)
- Command line

5.8.1 Adding or changing from the [On-Chip Event] dialog box

(1) Select [View -> Event -> On-Chip Event] to bring up the [On-Chip Event] dialog box. Then switch to the [Event] tab.

(2) To add an event, click the [Add] button in either the [Event A] or [Event B] section in the [Event] tab and bring up the [Event] dialog box.

To change or inspect the existing event, double-click on the event and bring up the [Event] dialog box.

(3) Set or inspect the event conditions in the [Event] dialog box and click the [OK] button.

(4) Settings made according to the step (3) will be added to the [On-Chip Event] dialog box. Click the [Apply] button, then the [OK] button.

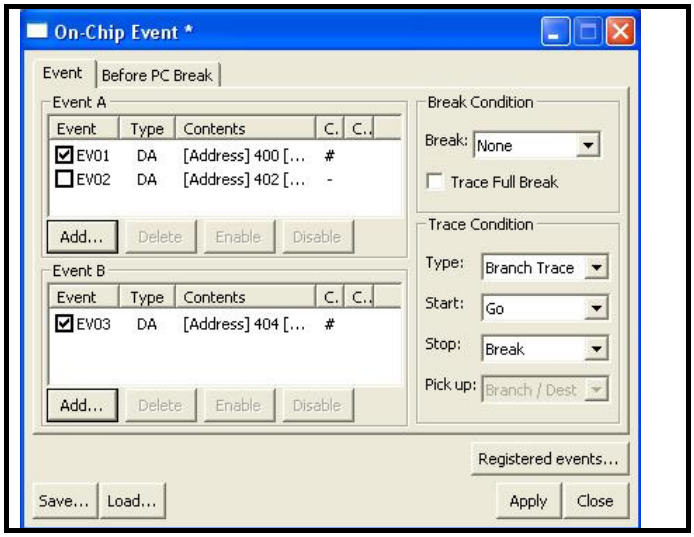


Figure 5.15 Example of On-chip Event (DA Event) Setting

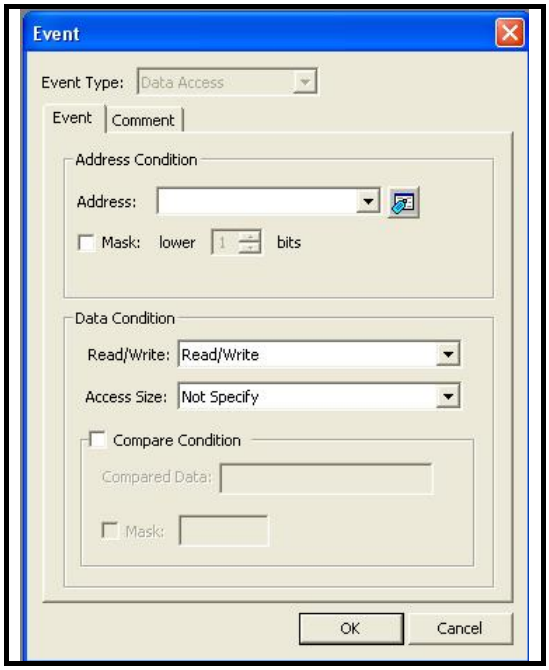


Figure 5.16 Example of [Event] Dialog Box (DA Event) Setting

5.8.2 Dragging and dropping from other windows (addition only)

(1) When dragging and dropping a variable name or a function name in the [Editor] window

By dragging and dropping a variable name into the [Event] column of the [On-Chip Event] dialog box, you can set access to that variable as an event to be detected. 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 or 2 bytes can be registered for event detection. Static variables in functions cannot be registered.

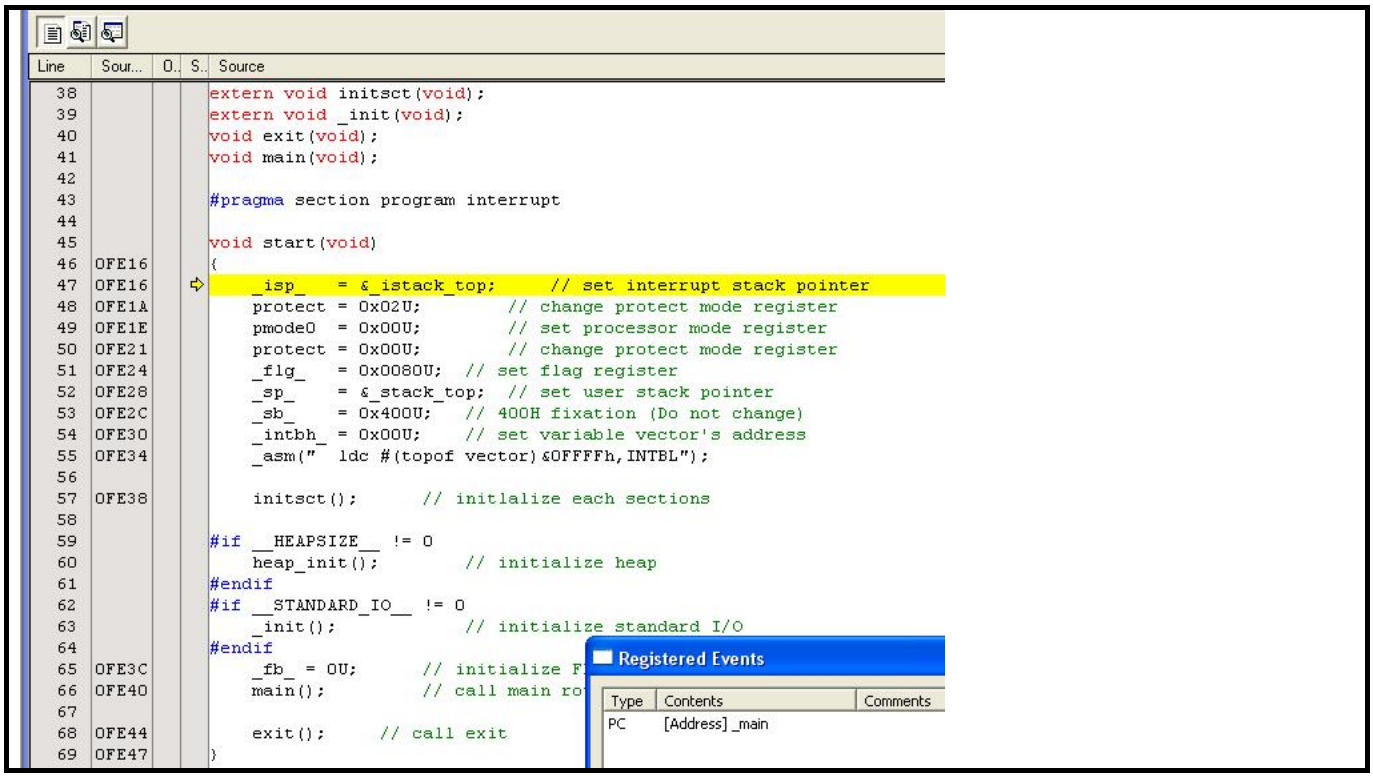


Figure 5.17 Adding an Event by Dragging and Dropping

(2) When 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 of the [On-Chip Event] dialog box. In this way, you can set access to an address in the selected address range as an event to be detected.

5.8.3 Deleting, enabling, or disabling from the [On-Chip Event] dialog box

(1) Select [View -> Event -> On-Chip Event] to bring up the [On-Chip Event] dialog box. Then switch to the [Event] tab.

(2) Follow the instructions below to set the Event A or Event B section in the [Event] tab.

Deleting an event:	Select the event you want to delete and click the [Delete] button. To delete multiple events at a time, select the lines you want to delete while holding down the Shift or Ctrl key and click the [Delete] button. (Instead of the [Delete] button, you can use the Ctrl + Del keys to delete.)
Enabling an event:	Select the event you want to enable and click the [Enable] button.
Disabling an event:	Select the event you want to disable and click the [Disable] button.

(3) Click the [Apply] button, then the [OK] button.

5.8.4 Command line

Refer to the online help.

5.9 Adding on-chip events (PC event)

Use one of the following to set up an event.

- [On-Chip Event] dialog box
- [On-chip Break points] column of the [Editor] window
- Drag-and-drop from another window (for add only)
- Command line

5.9.1 Adding from the [On-Chip Event] dialog box

(1) Select [View -> Event -> On-Chip Event] to bring up the [On-Chip Event] dialog box. Then switch to the [Before PC Break] tab.

(2) To add an event, click the [Add] button to bring up the [Event] dialog box.
To change or inspect the existing event, double-click the event to bring up the [Event] dialog box.

(3) Set or inspect the event conditions in the [Event] dialog box and click the [OK] button.

(4) Settings made according to the step (3) will be added to the [On-Chip Event] dialog box. Click the [Apply] button, then the [OK] button.

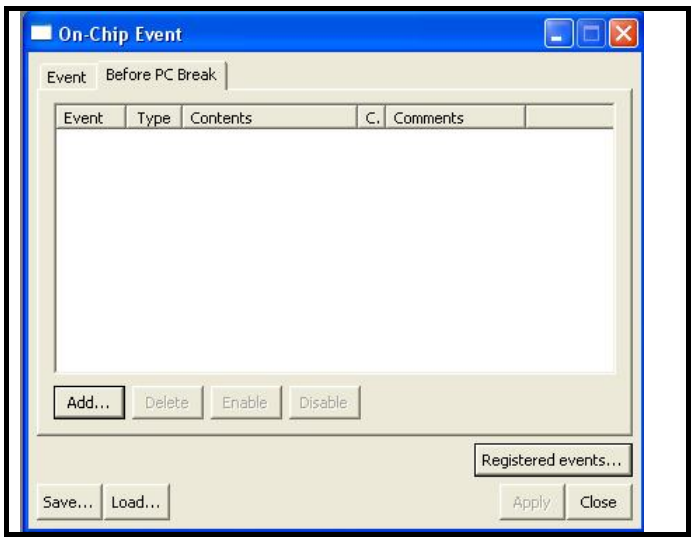


Figure 5.18 Example of On-chip Event (PC Event) Setting

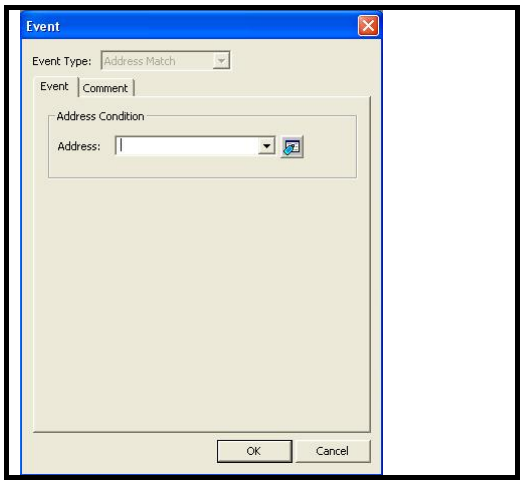


Figure 5.19 Example of [Event] Dialog Box (PC Event) Setting

5.9.2 [On-chip Break points] column of the [Editor] window

(1) Adding/deleting break conditions

In the [Editor] window, double-click the [On-chip Break points] column on a line that has no pre-PC breaks set. That way you can set a pre-PC break in an enabled state (blue ●).

Also, when you double-click the [On-chip Break points] column on a line that has a pre-PC break in an enabled state (blue ●) or disabled state (white ○), the pre-PC break is deleted.

Note that setting pre-PC break conditions from the [Editor] window is possible even when the user program is running.

(2) Enabling/disabling break conditions

Break conditions cannot be enabled or disabled from this window. Use the [On-Chip Event] dialog box or the [Command line] to do it.

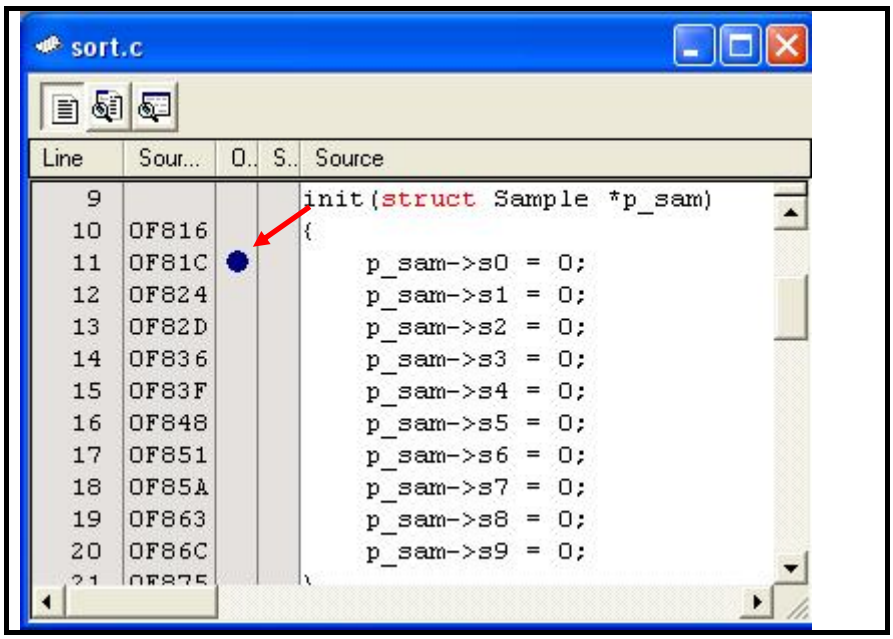


Figure 5.20 [Editor] Window

Notes:

[*1] You cannot set pre-PC breakpoints in the [Editor] window unless you've clicked the [Apply] button after changing settings in the [On-Chip Event] dialog box, i.e., when the set contents are not enabled (the title bar is marked with an asterisk (*) at the end of it).

[*2] The column display in the [Editor] window is updated when you've clicked the [Enable] button after changing settings in the [On-Chip Event] dialog box or set on-chip events from the command line.
Also, the contents of pre-PC break conditions you've set in the [Editor] window are reflected in those of pre-PC break conditions you set in the [On-Chip Event] dialog box or from the command line.

5.9.3 Dragging and dropping from other windows (addition only)

(1) When dragging and dropping a variable name or a function name in the [Editor] window

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

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

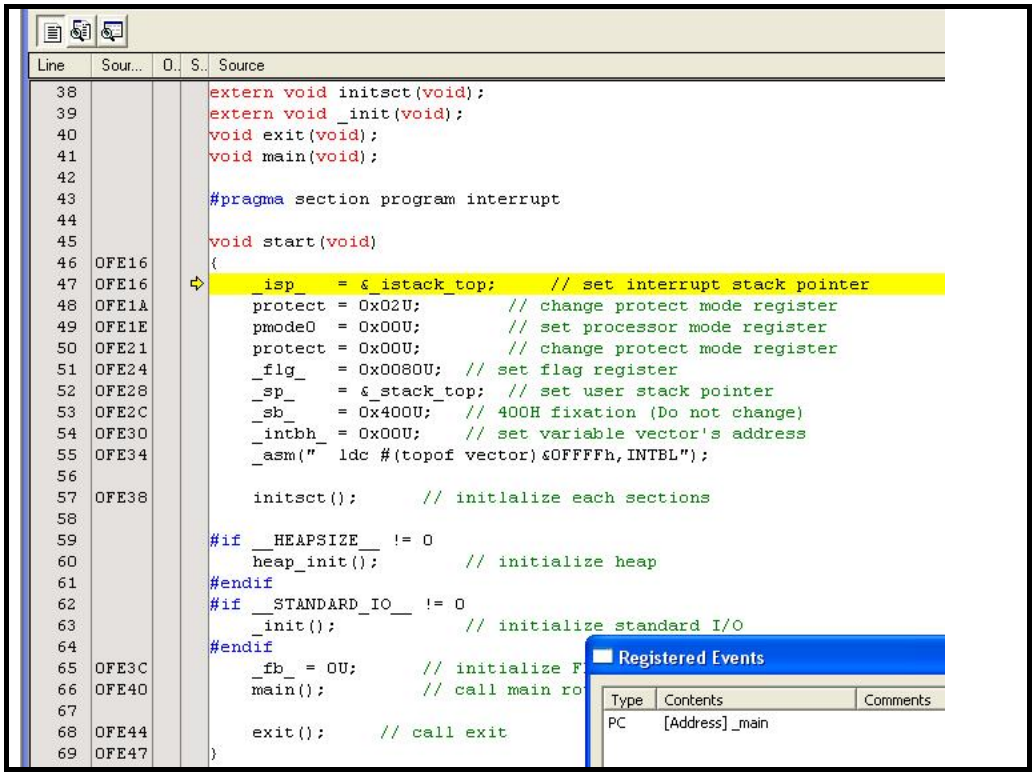


Figure 5.21 Adding an Event by Dragging and Dropping

(2) When dragging and dropping a label in the [Label] window

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

5.9.4 Deleting, enabling, or disabling from the [On-Chip Event] dialog box

(1) Select [View -> Event -> On-Chip Event] to bring up the [On-Chip Event] dialog box. Then switch to the [Before PC Break] tab.

(2) Follow the instructions below.

Deleting an event:	Select the event you want to delete and click the [Delete] button. To delete multiple events at a time, select the lines you want to delete while holding down the Shift or Ctrl key and click the [Delete] button. (Instead of the [Delete] button, you can use the Ctrl + Del keys to delete.)
Enabling an event:	Select the event you want to enable and click the [Enable] button.
Disabling an event:	Select the event you want to disable and click the [Disable] button.

(3) Click the [Apply] button, then the [OK] button.

5.9.5 Command Line

Refer to the online help.

5.10 Registering events

5.10.1 [Registered Events] dialog box

The [Registered Events] dialog box can be opened when you click the [Registered events…] button located in the lower right part of the [On-Chip Event] dialog box (Figure 5.22).

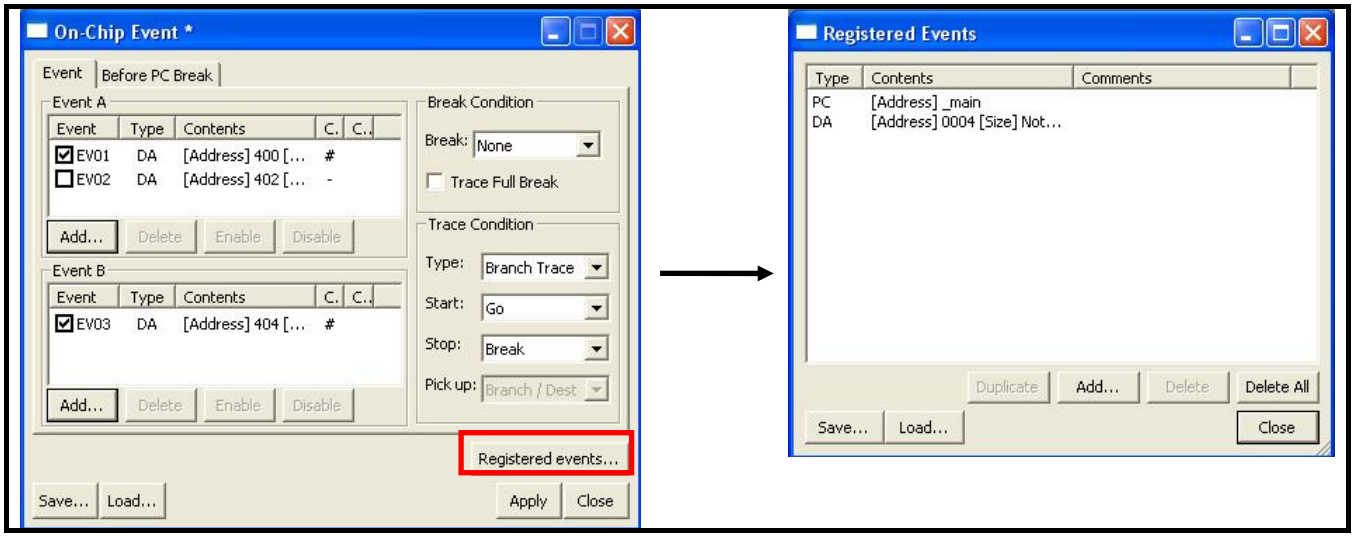


Figure 5.22 How to Display the [Registered Events] Dialog Box

(1) Duplicate

Select an event you want to copy and click the [Duplicate] button.

The selected event is copied.

(2) Add

The [Event] dialog box is displayed, allowing you to add event contents.

(3) Delete

Select the line you want to delete and click the [Delete] button. The selected event is deleted from the list.

(Instead of the [Delete] button, you can use the Ctrl + Del keys to delete.)

(4) Delete All

Deletes all events.

(5) Save…

Saves the contents of this dialog box that you've set in a file.

(6) Load…

Loads the saved event information from a file. When normally loaded, the set contents of this dialog box are discarded and replaced with the contents of the file.

(7) Close

Closes this dialog box.

Table 5.11 Display Contents of the [Registered Event] Dialog Box

Address	Description	Remark
Type	Displays DA (data access event) or PC (address match event)	
Contents	Displays the set contents of events	
Comments	Displays comments on each event	

5.10.2 Registering events

"Registering an event" refers to placing an event in the list of [Registered Events] dialog box. A registered event can be reused at a later time.

Select one of the following ways to register an event.

- Through the [On-Chip Event] dialog box
- By dragging and dropping
- Through the [Registered Events] dialog box

(1) Registering events through the [On-Chip Event] dialog box

Open the [Comment] tab and select the [Add this event to the list] checkbox. After checking the [Add this event to the list] checkbox, click the [OK] button.

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

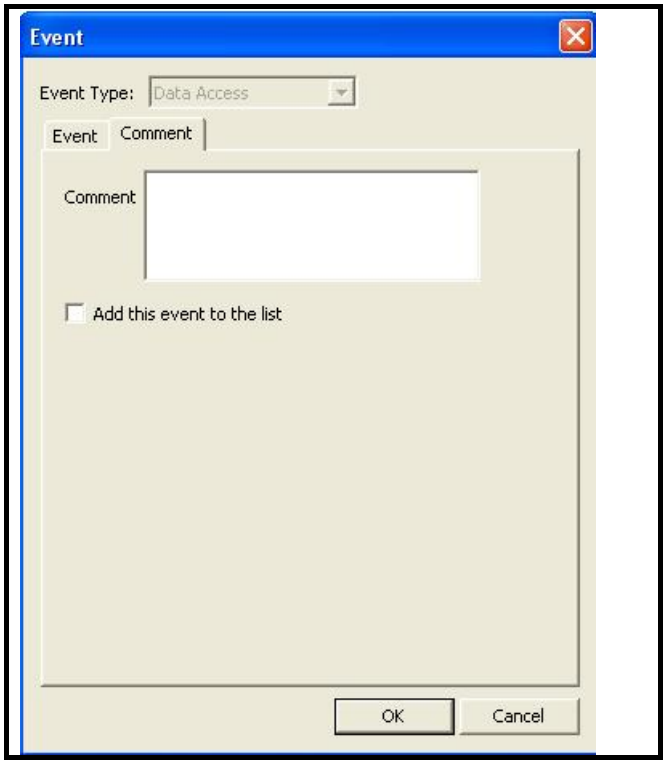


Figure 5.23 Registering an Event through the [Event] Dialog Box

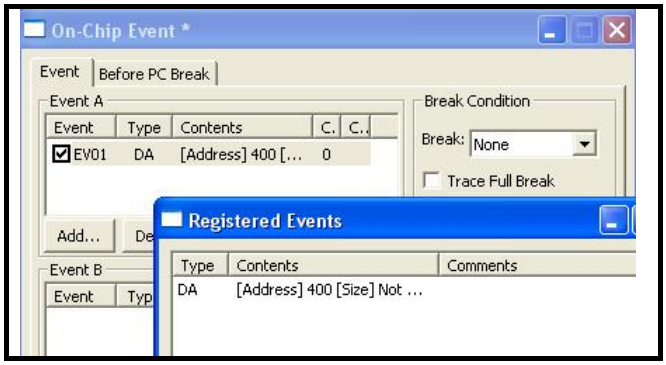


Figure 5.24 [On-Chip Event] Dialog Box and [Registered Events] Dialog Box

(2) Registering events by dragging and dropping

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

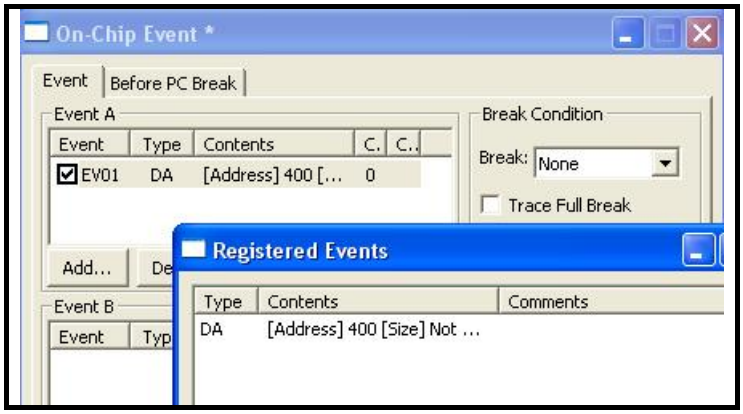


Figure 5.25 [On-Chip Event] Dialog Box and [Registered Events] Dialog Box

(3) Registering an event through the [Registered Events] dialog box

Click on the [Add…] button in the [Registered Events] dialog box so you can create events. Any event you create through the [Registered Events] dialog box is added to the [Registered Events] dialog box.

An explanatory comment for the event can be attached. You can check the [Registered Events] dialog box to see the contents of registered events and comments.

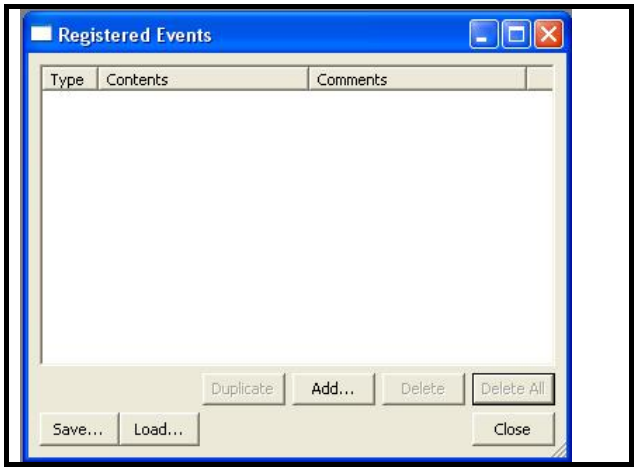


Figure 5.26 [Registered Events] Dialog Box

5.10.3 Creating events for each instance of usage or reusing events

The following two approaches are available for setting data access and address match events.

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 box in the [On-Chip Event] dialog box.

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

Table 5.12 Types of Event Setting Approaches

Event setting approaches	Description
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 [On-chip Break points] 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 box in the [On-Chip Event] dialog box.

(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 event list boxes in the [On-Chip Event] dialog box.

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.

5.10.4 Activating events

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

[*] after the title on the title bar of the [On-Chip Event] 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.

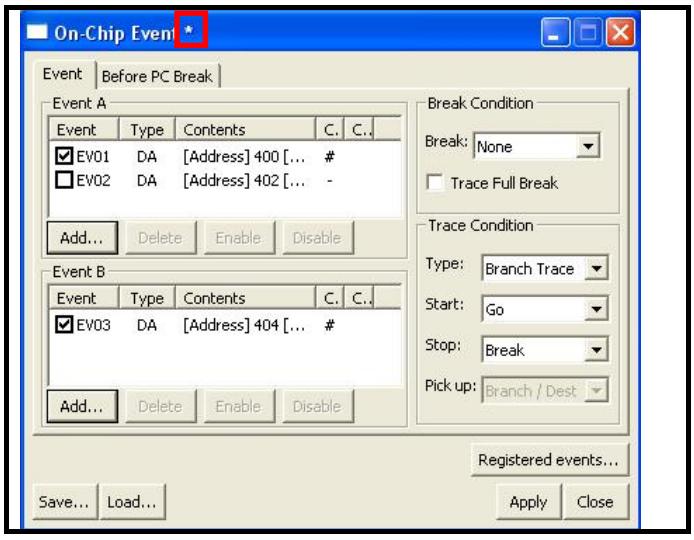


Figure 5.27 Activating the Settings

5.11 Saving/loading the set contents of on-chip events

You can save or load the contents of the [On-Chip Event] dialog box.

5.11.1 Saving [On-Chip Event] Settings

(1) Click the [Save…] button of the [On-Chip Event] dialog box. The [Save] dialog box will be displayed.
(2) Specify a file name to which you want the settings to be saved.
The file name extension is .rev. If omitted, the extension .rev is automatically attached.

5.11.2 Loading the set contents of [On-Chip Event] settings

(1) Click the [Load…] button of the [On-Chip Event] dialog box.
The [Load] dialog box will be displayed.
(2) Specify the file name you want to load.
When you load a file, the [On-Chip Event] settings you had before you have loaded the file are discarded and the contents are reset with the loaded settings.
(3) Click the [Apply] button of the [On-Chip Event] dialog box to confirm the [On-Chip Event] settings you have loaded.

5.12 Saving/loading the set contents of the [Registered Events] dialog box

You can save or load the contents of the [Registered Events] dialog box.

5.12.1 Saving [Registered Events] dialog box settings

(1) Click the [Save…] button of the [Registered Events] dialog box.
The [Save] dialog box will be displayed.
(2) Specify a file name to which you want the settings to be saved.
The file name extension is .lev. If omitted, the extension .lev is automatically attached.

5.12.2 Loading the set contents of [Registered Events] dialog box settings

(1) Click the [Load…] button of the [Registered Events] dialog box.
The [Load] dialog box will be displayed.
(2) Specify the file name you want to load.
When you load a file, the [Registered Events] dialog box settings you had before you have loaded the file are discarded and the contents are reset with the loaded settings.
(3) Click the [Apply] button of the [Registered Events] dialog box to confirm the [Registered Events] dialog box settings you have loaded.

5.13 Trace function

5.13.1 Outline of trace function

A trace refers to the function that acquires information on branches that occurred during user program execution and information on bus activities resulting from data accesses. It tracks the flow of user program execution, allowing you to examine points in the program where problems occurred.

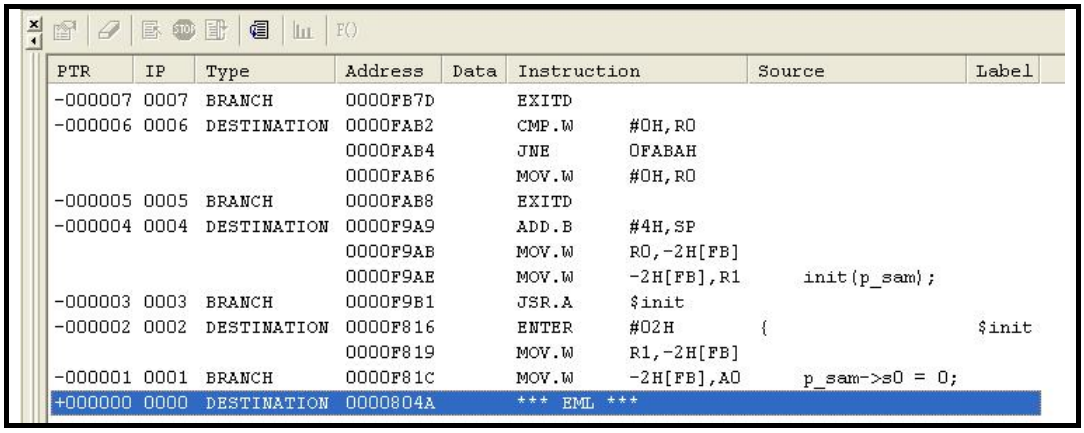
The MCUs in this user's manual provide their built-in trace functions (see Table 5.13).

Table 5.13 Trace Function of the MCU Used

Type of trace		Content
Branch trace	Number of jumps	Up to 4 jumps
	Branch information	The branch source and the branch destination acquired in a set
	Contents	Displays addresses, mnemonics and source lines of the branch source and destination
Data trace	Number of data	Up to 8 accesses
	Contents	Displays data accesses when a data access event is encountered.

To display a trace result, select [View -> Code -> Trace] to bring up the [Trace] window.

Figure 5.28 shows an example of a trace result. Table 5.14 shows the items displayed in the window.



PTR	IP	Type	Address	Data	Instruction	Source	Label
-000007	0007	BRANCH	0000FB7D		EXITD		
-000006	0006	DESTINATION	0000FAB2		CMP.W #0H,R0		
			0000FAB4		JNE 0FABAH		
			0000FAB6		MOV.W #0H,R0		
-000005	0005	BRANCH	0000FAB8		EXITD		
-000004	0004	DESTINATION	0000F9A9		ADD.B #4H,SP		
			0000F9AB		MOV.W R0,-2H[FB]		
			0000F9AE		MOV.W -2H[FB],R1	init(p_sam);	
-000003	0003	BRANCH	0000F9B1		JSR.A $init		
-000002	0002	DESTINATION	0000F816		ENTER #02H	{	$init
			0000F819		MOV.W R1,-2H[FB]		
-000001	0001	BRANCH	0000F81C		MOV.W -2H[FB],A0	p_sam->s0 = 0;	
+000000	0000	DESTINATION	0000804A		*** EML ***		

Figure 5.28 Example of a Trace Result

Table 5.14 Trace Display

Item	Content
PTR	Displays the pointer numbers in the trace buffer. Displays them in ascending order with the trace end position as 0.
IP	Displays the instruction pointer (absolute values of the pointer numbers) . When the data trace is set, nothing is displayed.
Type	Displays the type of trace information. When the branch trace is set, BRANCH/DESTINATION is displayed. When the data trace is set, READ/WRITE is displayed.
Address	When the branch trace is set, an address of the branch source and destination is displayed. When the data trace is set, an address or address range set for the encountered event is displayed.
Data	When the data trace is set, the accessed value is displayed. When the branch trace is set, nothing is displayed.
Instruction	When the branch trace is set, the mnemonic of the address is displayed. When the data trace is set, nothing is displayed. "*** EML ***" may be displayed in the Instruction column. This shows that the target program accessed the area of emulator use to control breaks, etc. It is not an error.
Source	If there is a source line information correspondent to the Instruction, the correspondent source line is displayed. When the data trace is set, nothing is displayed.
Label	If there is a label correspondent to an address in the Instruction, the correspondent label is displayed. When the data trace is set, nothing is displayed.

5.13.2 Trace setting items

To set the trace condition, open the [On-Chip Event] dialog box. Table 5.15 shows trace conditions.

Table 5.15 Trace Condition

Option	Description	
Type	Select a trace type from the following:	
	Branch trace (default):	Branch trace Traces the branch source address and branch destination address of branch processing that occurred during the execution of the user program.
	Data access:	Data access Traces the data access information of events that occurred during the execution of the user program.
Start	Select a start condition for the trace measurement from the following options.	
	Go (default):	Starts a measurement when starting executing the user program.
	Event A	Starts a measurement when the Event A is encountered.
	Event A or B:	Starts a measurement when either the Event A or Event B is encountered.
	Event A and B:	Starts a measurement when both the Event A and Event B are encountered.
	Event B->A:	Starts a measurement when an event is encountered in the order of the Event B and Event A.
Stop	Select a stop condition for the trace measurement from the following options.	
	Break (default):	Stops a measurement when stopping executing the user program.
	Trace FULL:	Stops a measurement when the trace data is filled.
	Event A	Stops a measurement when the Event A is encountered.
	Event A or B:	Stops a measurement when either the Event A or Event B is encountered.
	Event A and B:	Stops a measurement when both the Event A and Event B are encountered.
	Event B->A:	Stops a measurement when an event is encountered in the order of the Event B and Event A.
Pick up	Select an event to record when the trace type is [Data access] from the following:.	
	Event A (default):	Records only data access which encounters the condition of the Event A.
	Event A or B:	Records only data access which encounters the condition of either the Event A or Event B.

Notes:

[*1] Note on the trace start condition
When setting an event (other than [Go]) for the trace start condition, a data when the event is encountered is not recorded to the trace data. The data of the event which is encountered the next time is recorded.

[*2] Notes on the trace stop condition
When the trace start and trace stop conditions occur simultaneously, the trace stop condition becomes invalid.
When setting other than [Break] for the trace stop condition, the display contents of the [Trace] window will not be updated until the user program stops even after a trace stop condition is encountered.

[*3] If a reset such as a hardware reset (RESET#) or watchdog timer reset is issued while a user program is running, the encountered event records and trace records made until the reset is issued will be cleared.

[*4] Only access via the CPU bus is traceable. No record of access by the DTC is included in trace data.

5.13.3 Trace menu

The toolbar menu on the trace window is shown in Table 5.16.

Table 5.16 Toolbar Menu on the [Trace] Window

Menu name	Description
Set	Sets conditions for acquiring trace information. (For the MCU used here, this function has no effect.)
Clear	Clears all trace records.
Show source	Shows the source program corresponding to the relevant address on the selected line.
STOP	Stops acquiring trace information. (For the MCU used here, this function has no effect.)
Restart	Starts acquiring trace information from where it stopped. (For the MCU used here, this function has no effect.)
Branch trace	Selects whether to enable source interpolation during a branch trace. (Grayed out when data trace is selected.)
Statistics	Executes analysis of statistical information.
Function call	Shows spots where function calls were made.

5.13.4 Branch trace

When the [Trace] window is set for branch trace, you can interpolate the execution state from the branched-to address to the branched-from address based on source information, etc. A window shown in Figure 5.29 is displayed when you click the [Branch trace] button on the toolbar menu. Select [Enable] in this window, and source-interpolated information will be displayed in the trace window.

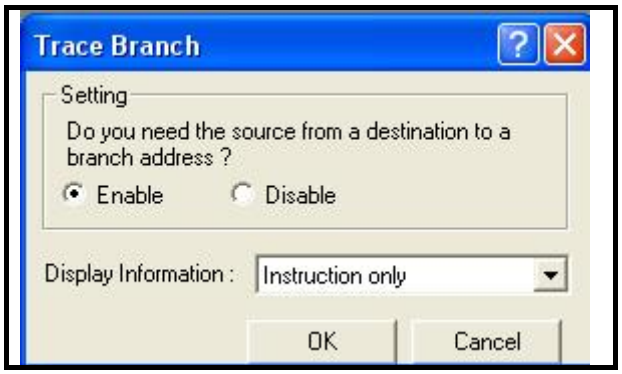


Figure 5.29 Selecting to Enable Interpolation in Brach Trace

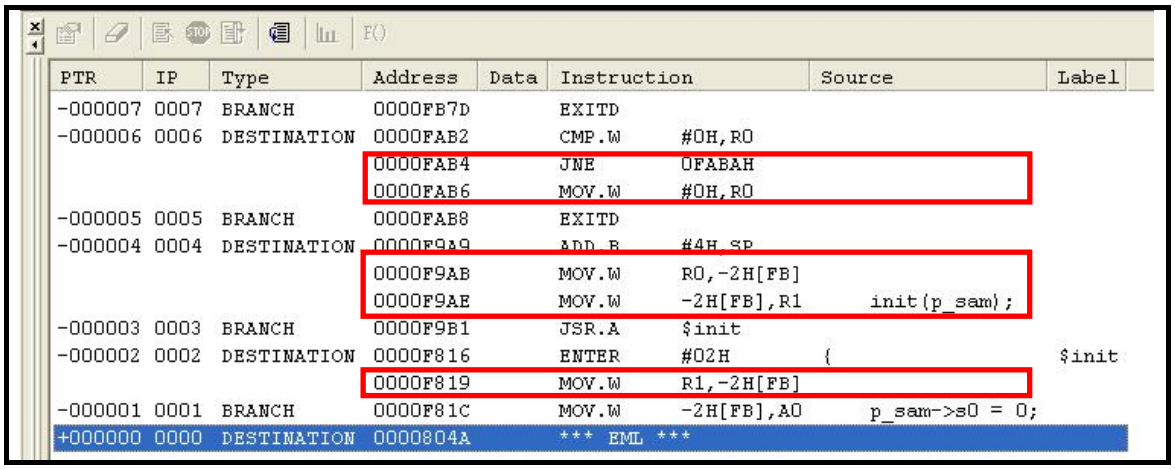


Figure 5.30 Example of a Source-Interpolated Branch Trace

5.13.5 Statistics

From the trace result information, it is possible to show the number of occurrences of trace information that meet a specified condition and the trace buffer pointer Nos. A window shown in Figure 5.31 is displayed when you click the [Statistics] button on the toolbar menu. Select [Item] in this window and click the [New] and then the [Result] button. The number of occurrences of trace information and the trace buffer pointer Nos. will be displayed.

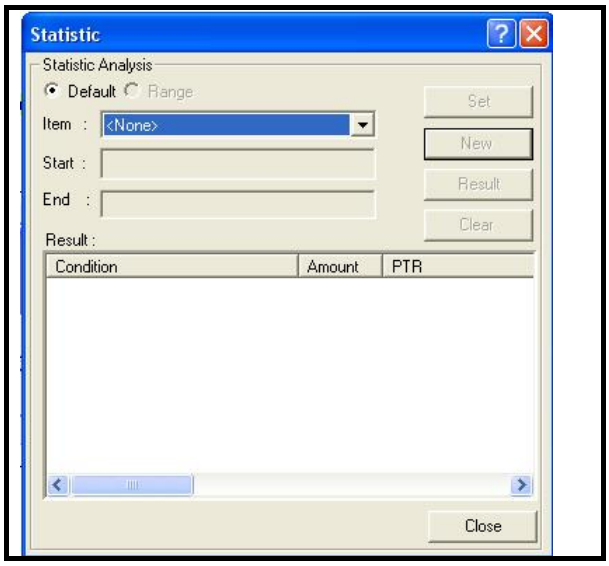


Figure 5.31 Example of Statistics Menu

(1) Function call

From the trace result information, it is possible to show only the lines where function calls were made.

A window is displayed when you click the [Function call] button “F()” on the toolbar menu. Select [Enable] in this window, and information on only function call lines will be displayed in the trace window.

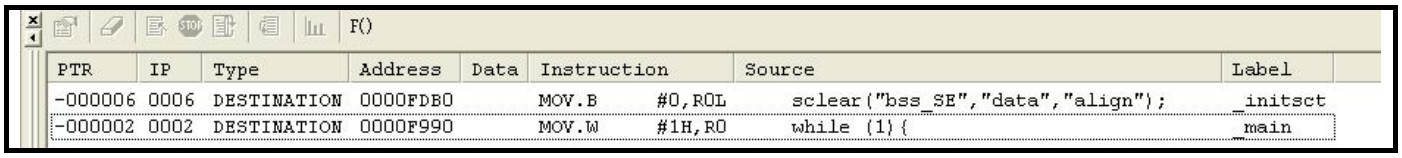


Figure 5.32 Example of the Function Call Menu

5.13.6 Saving trace information in files

To save trace information in a file, select [Save] from the popup menu. The trace information displayed in the [Trace] window is saved in a text format.

5.14 Status bar

To find out the current status of the debugging platform while E1 or E20 emulator is connected, display the [Status] bar. To open the [Status] bar, select [View -> Status Bar].

Figure 5.33 [Status] bar Example

Item	Contents	Display content	Remarks
PC	Status during the execution of the program	Normal	User program is in stopped state.
		Running	User program is in execution state.
		Stop	Target MCU is in stop mode.
		Wait	Target MCU is in wait mode. [*1]
Task ID	A current task ID number is displayed when the module used by the OS is loaded.	Task ID: xx	XX: current Task ID
Break Condition	Factors that causes the user program to stop	Trace memory overflow break	Trace full break
		Stepping completed	Step execution
		Software break	S/W break
		Data access break at DAx,…	Data access break
		Data access break (Sequential)	Data access break (sequential)
		Data access break (AND)	Data access break (events encountered with AND condition)
		Address match break	Address match break
		User break	Forced break
		Go to cursor break	Execute the program to the cursor position
		Unknown break cause	Cause is unknown
Execution Time	Execution time between RUN and STOP	hh:mm:ss.xxx.uuu	hh=hour, mm=minute, ss=second, xxx= millisecond, uuu= microsecond

Note:
[*1] “Stop” is displayed instead of “Wait” when entering wait mode by setting the CM30 bit to “1”.

5.15 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 5.34 [Start/Stop function setting] Dialog Box

5.15.2 Specifying the routine to be executed

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

5.15.3 Restrictions on the Start/Stop function

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

(1) The debugging functions listed below are not to be used while the start/stop function is in use.

- Memory setting and downloading into the program area of a specified routine
- Breakpoint setting in the program area of a specified routine

(2) In the specified routines, the following restrictions apply to registers and flags.

Table 5.17 Restrictions on Registers and Flags

Register and Flag Names	Restrictions
ISP register	When execution of a specified routine is ended, the register must be returned to its value at the time the routine started.
Flag U	When a specified routine ends, the value of this flag must always be set to 0.
Flag I	No interrupts are allowed during execution of a specified routine.

(3) When either of the specified routines is running, the debugging functions listed below have no effect.

- Tracing
- Break-related facilities
- Setting events within the specified routines

(4) While either of the specified routines is running, non-maskable interrupts are always disabled.

(5) 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 5.18 MCU Status at Start of the User Program

MCU Resource	Status
CPU 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	Access to memory after execution of the specified routine is reflected.
Peripheral functions	Operation of the MCU peripheral functions after execution of the specified routine is continued.

5.15.4 Limitations on statements within specified routines

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

- If a specified routine uses a stack, the stack must always be the user stack.
  To terminate the processing of a specified routine, write a return subroutine instruction.
- Ensure that a round of processing by a specified routine is complete within 100 ms. If, for example, the clock is turned off and left inactive within a specified routine, the emulator may become unable to control program execution.
- The values stored in the registers at the time a specified routine starts running are undefined. Ensure that each specified routine initializes the register values.

5.16 Simple stack overflow detection function

Setting the size of the stack too small in software development raises the possibility of a program going out of control or malfunctioning.

If the stack exceeds the RAM area when the register is saved, the R8C E1/E20 Emulator Debugger displays a stack overflow message, “The stack has exceeded the address range of the RAM area” in the [Output] window. [*1]

When this error occurs, perform a reset from the emulator debugger.

Note:

[*1] A stack overflow is detected only when the register is saved (when the ISP or USP address is lower than 400h). A stack overflow cannot be detected if the register is restored exceeding the RAM area.
A stack overflow is detected normally if the lower limit of the stack range is located within 4 bytes from 400h, which is the start address of the internal RAM of the MCU (equivalent to 003FC - 003FFh).

5.17 Online help

The online help describes the usage of each function or command syntax input from the command line window.

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

6.Tutorial

6.1 Introduction

A tutorial program for the E1 and E20 emulators is provided as a means of presenting the emulator's main features to you. The tutorial is described in this section. This section describes the main features of the E1 and E20 emulators by using a tutorial program.

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

The main function generates random data to be sorted.
The sort function sorts the generated random data in ascending order.
The change function then sorts the data in descending order.

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

Notes:
[*1] If the tutorial program is recompiled, the addresses in the recompiled program may not be the same as those described in this section.
[*2] The file tutorial.c contains source code for the tutorial program. The file Tutorial.x30 is a compiled load module in the IEEE695 format.

6.2 Starting the High-performance Embedded Workshop

Open a workspace by following the procedure described in "4.9 Procedure for launching the E1/E20 emulator debugger" on page 29.

Specify the directory given below.
(Drive where the OS is installed)\Workspace\Tutorial\E1E20\R8C\Tutorial

Specify the file shown below.

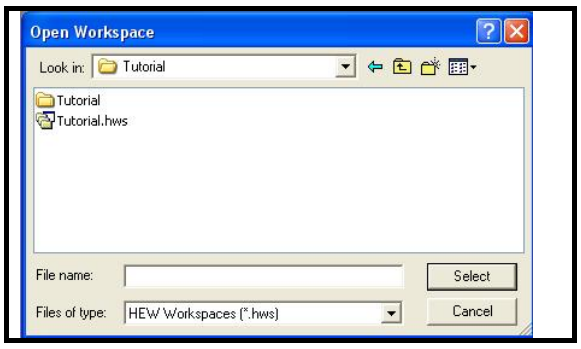


Figure 6.1 [Open Workspace] Dialog Box

6.3 Connecting the emulator

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

6.4 Downloading the tutorial program

6.4.1 Downloading the tutorial program

Download the object program you want to debug. Note, however, that the address where the program will be downloaded depends on the MCU in use. Accordingly, strings shown in the screen shots should be altered to those for the MCU in use. Right-click on the [Tutorial.x30] under [Download modules] and choose [Download] from the popup menu.

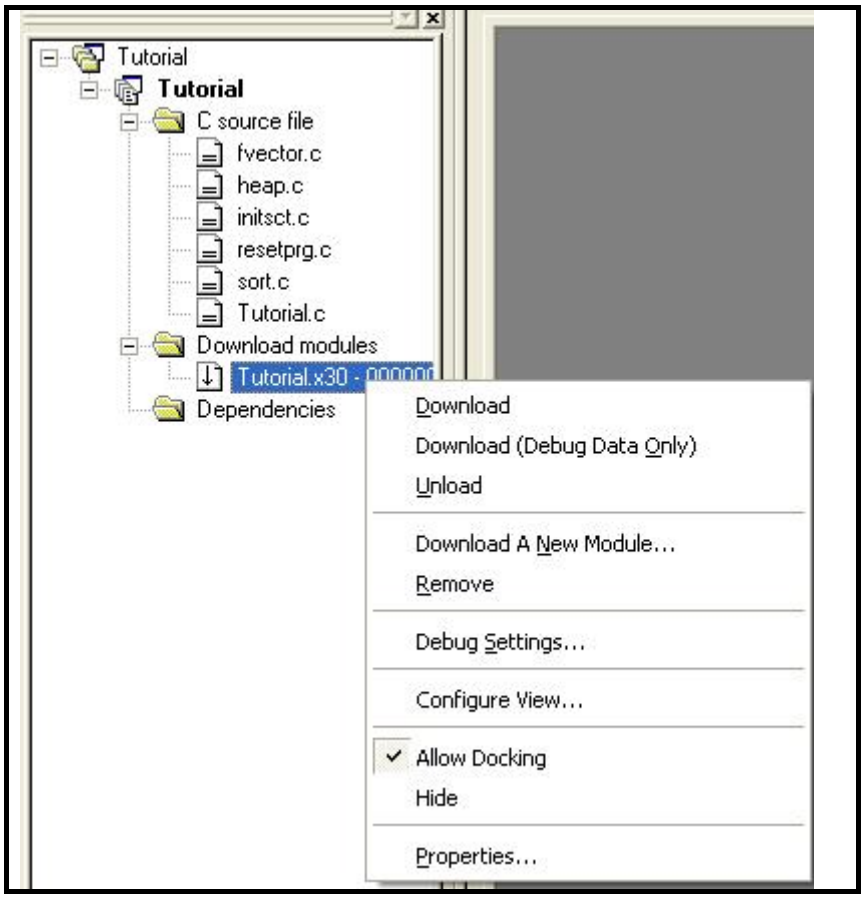


Figure 6.2 Downloading the Tutorial Program

6.4.2 Displaying the source program

In the High-performance Embedded Workshop you can debug programs at the source level.

Double-click on the C source file [Tutorial.c] under [C source file].

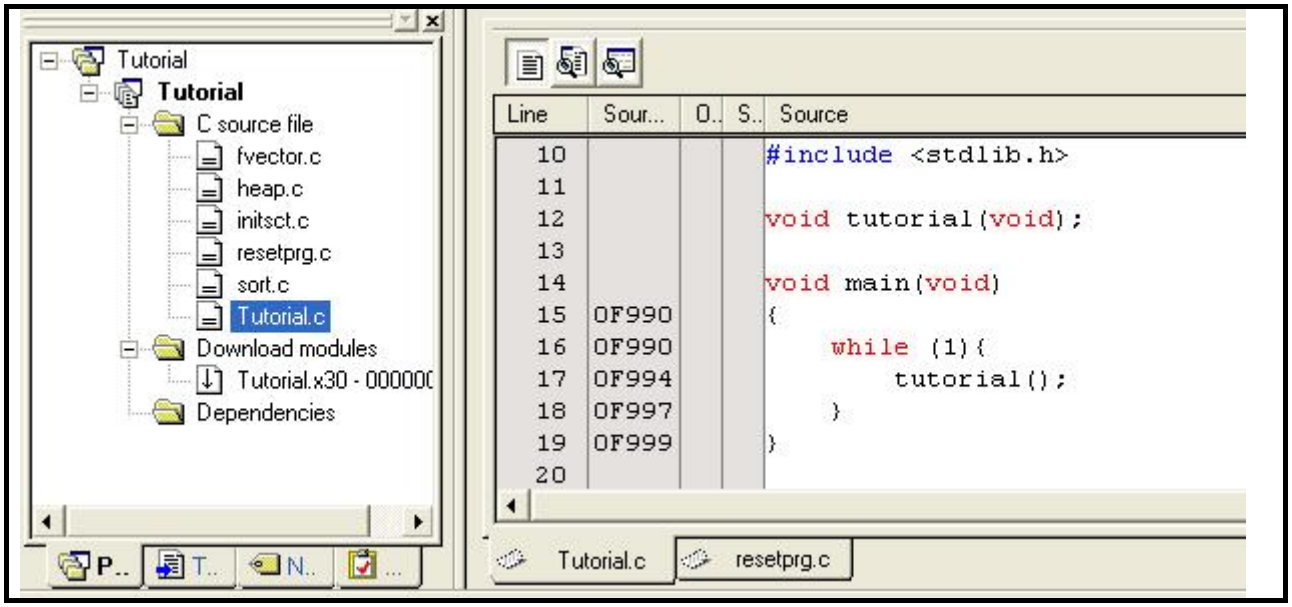


Figure 6.3 [Editor] Window (Displaying the Source Program)

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.

6.5 Setting S/W breakpoints

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

Double-click in the row of the [S/W Breakpoints] column which corresponds to the source line containing the call of the sort function.
The source line that includes the sort function will be marked with a red circle, indicating that a S/W breakpoint has been set there.

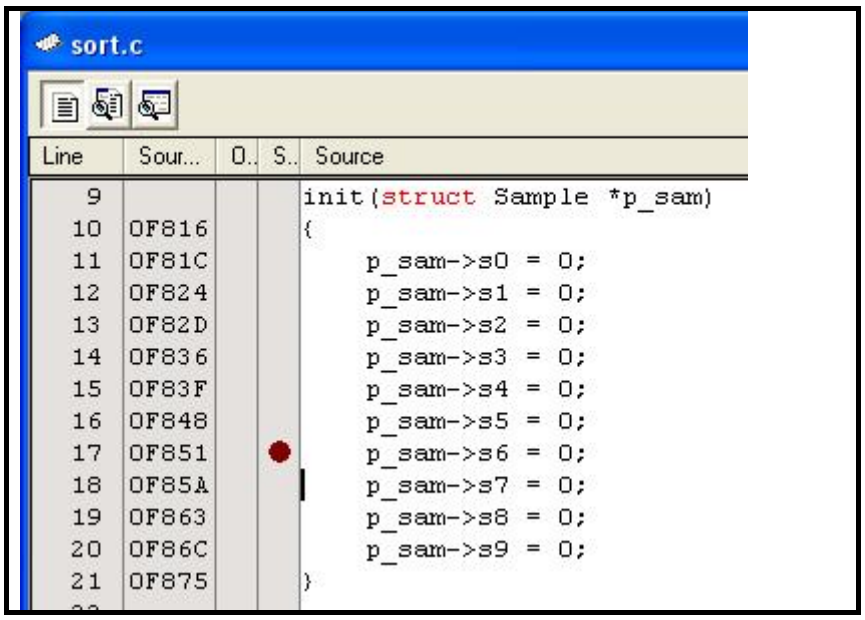


Figure 6.4 [Editor] Window (Setting a S/W Breakpoint)

6.6 Executing the program

The following describes how to run the program.

6.6.1 Resetting the CPU

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

6.6.2 Executing the program

To execute the program, choose [Go] from the [Debug] menu or click on the [Go] toolbar button ( ). When the program execution is started, “Running” is displayed on the status bar.

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.

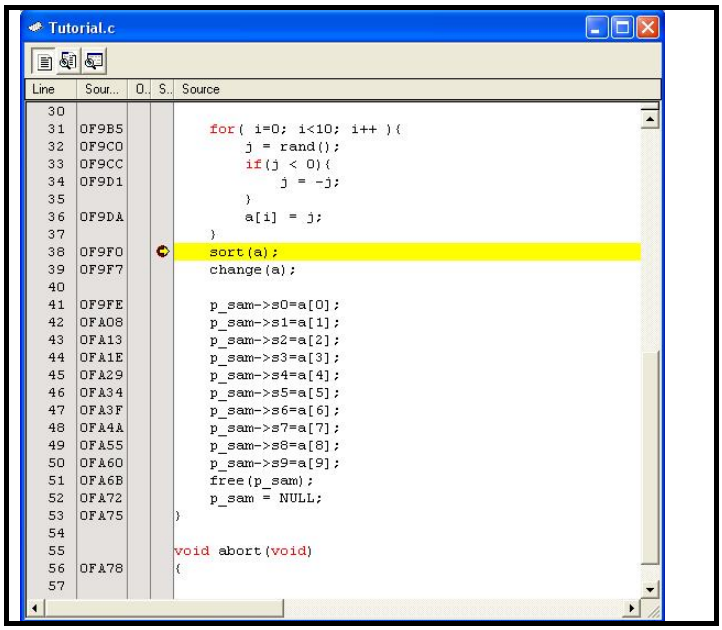


Figure 6.5 [Editor] Window (Break)

Note:

When the source file is displayed after a break, a path of the source file may be inquired.
The location of the source file is as follows:
<Drive where the OS has been installed>:\Workspace\Tutorial\E1E20\R8C\Tutorial\Source

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

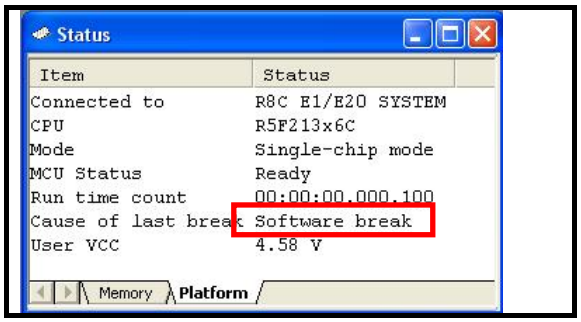


Figure 6.6 [Status] Window

6.7 Checking breakpoints

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

6.7.1 Checking breakpoints

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

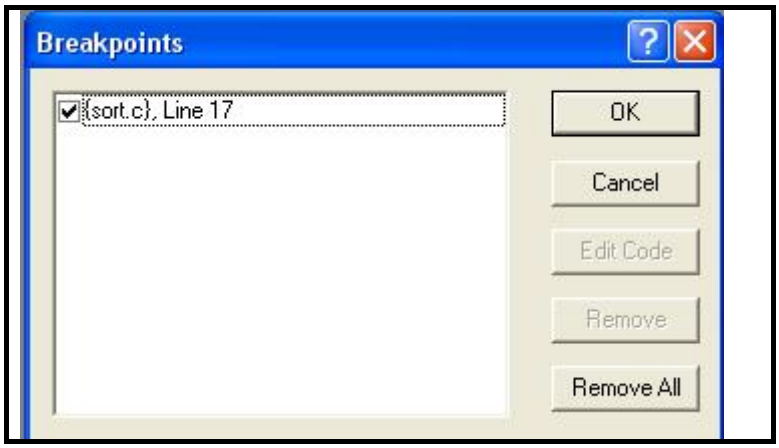


Figure 6.7 [Breakpoints] Dialog Box

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

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

6.8 Altering register contents

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

(1) Switching between register banks

Initially, the register data of Bank1 is displayed. Right-click on the [Register] window to open the popup menu. You can switch the register bank between the Bank 0 and the Bank 1.

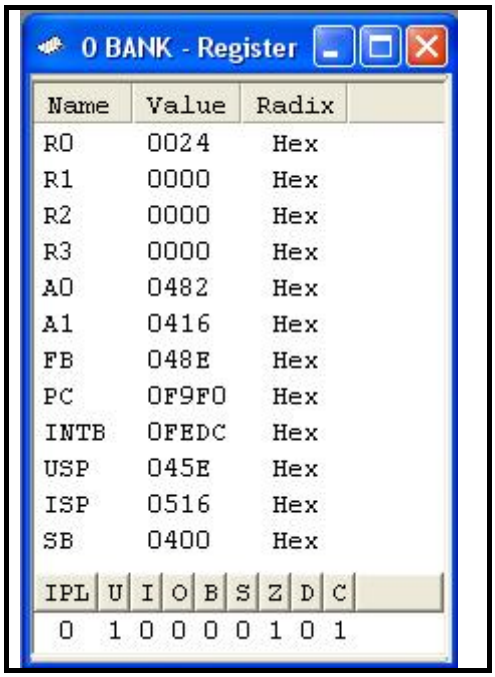


Figure 6.8 [Register] Window

(2) Altering the contents of registers

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.

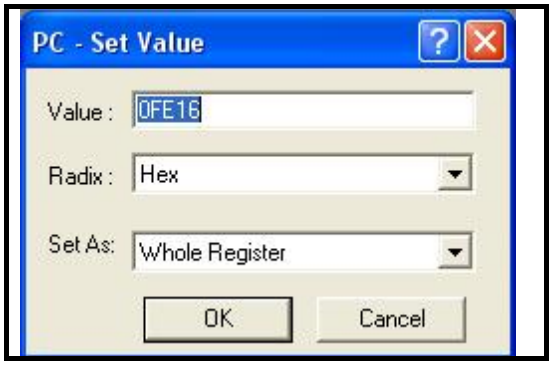


Figure 6.9 [Set Value] Dialog Box (PC)

Note:

[*1] The values of the CPU registers can be set in the [Register] window and by script commands. Note that the values set are reflected immediately before execution of the user program.

6.9 Referring to symbols

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

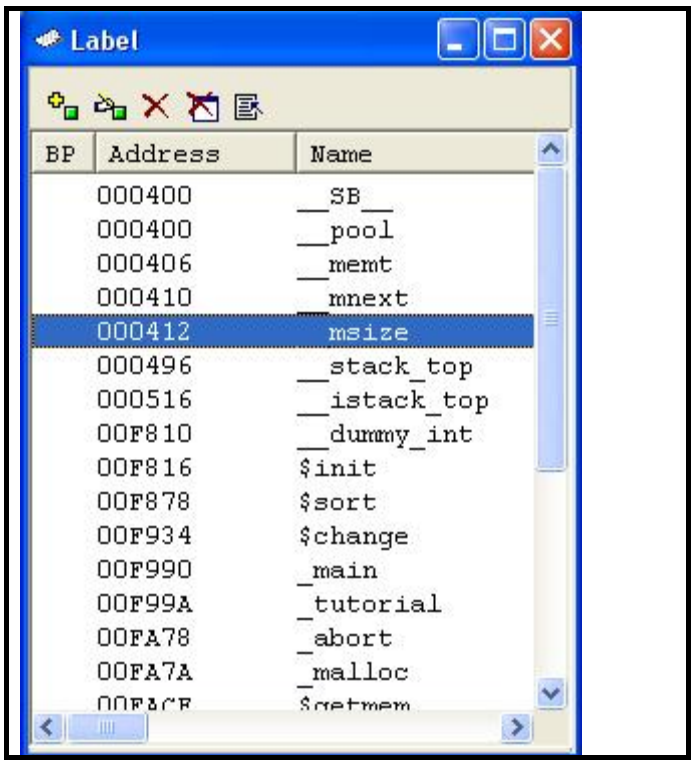


Figure 6.10 [Label] Window

6.10 Checking memory contents

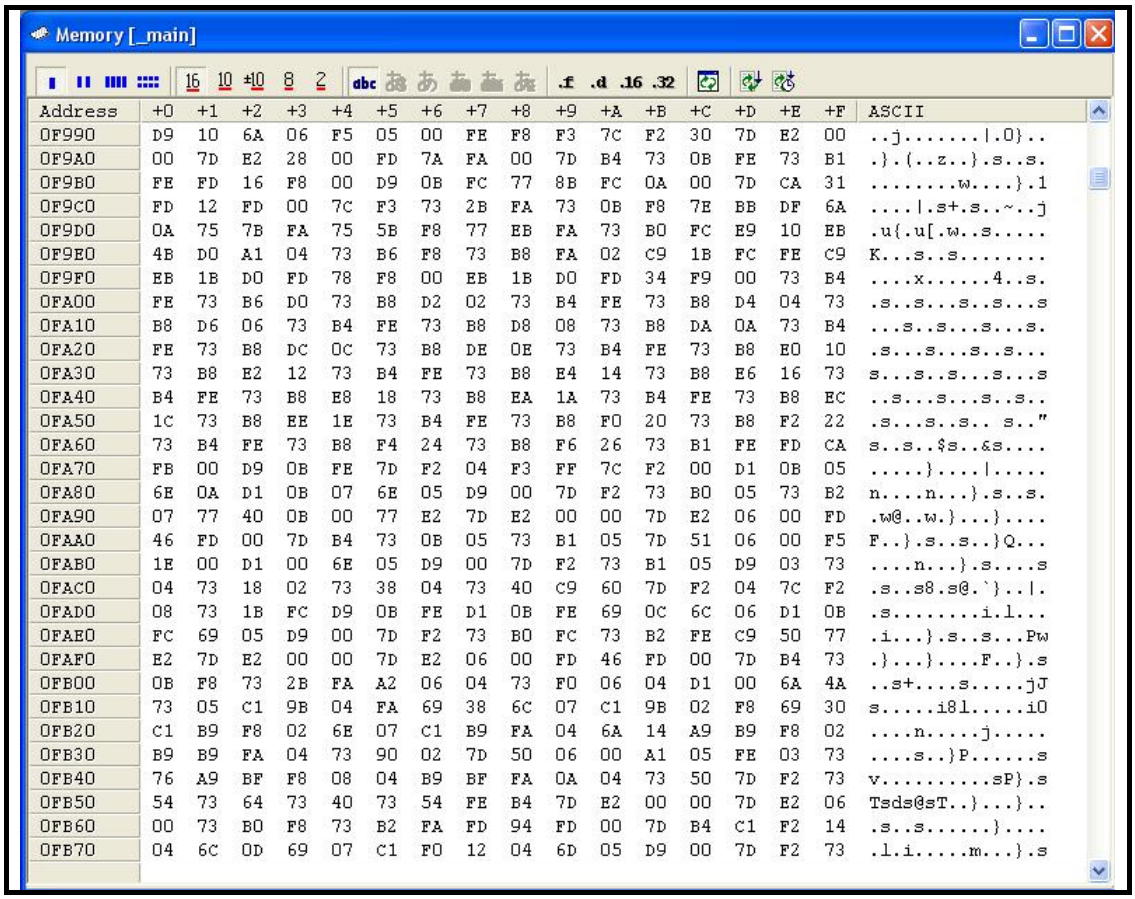
After you have specified a label name, you can use the [Memory] window to check the contents of memory where that label is registered. For example, you can check the contents of memory corresponding to _main in word size, as shown below.

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

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

Figure 6.11 [Display Address] Dialog Box

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



Address	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F	ASCII
0F990	D9	10	6A	06	F5	05	00	FE	F8	F3	7C	F2	30	7D	E2	00	..j.....|.0}..
0F9A0	00	7D	E2	28	00	FD	7A	FA	00	7D	B4	73	0B	FE	73	B1	.}.(..z..}.s..s.
0F9B0	FE	FD	16	F8	00	D9	0B	FC	77	8B	FC	0A	00	7D	CA	31	........w....}.1
0F9C0	FD	12	FD	00	7C	F3	73	2B	FA	73	0B	F8	7E	BB	DF	6A	....|.s+.s..~..j
0F9D0	0A	75	7B	FA	75	5B	F8	77	EB	FA	73	B0	FC	E9	10	EB	.u{.u[.w..s.....
0F9E0	4B	D0	A1	04	73	B6	F8	73	B8	FA	02	C9	1B	FC	FE	C9	K...s..s........
0F9F0	EB	1B	D0	FD	78	F8	00	EB	1B	D0	FD	34	F9	00	73	B4	....x.....4..s.
0FA00	FE	73	B6	D0	73	B8	D2	02	73	B4	FE	73	B8	D4	04	73	.s..s...s..s...s
0FA10	B8	D6	06	73	B4	FE	73	B8	D8	08	73	B8	DA	0A	73	B4	...s..s...s...s.
0FA20	FE	73	B8	DC	0C	73	B8	DE	0E	73	B4	FE	73	B8	E0	10	.s...s...s..s...
0FA30	73	B8	E2	12	73	B4	FE	73	B8	E4	14	73	B8	E6	16	73	s...s..s...s...s
0FA40	B4	FE	73	B8	E8	18	73	B8	EA	1A	73	B4	FE	73	B8	EC	..s...s...s..s..
0FA50	1C	73	B8	EE	1E	73	B4	FE	73	B8	F0	20	73	B8	F2	22	.s...s..s.. s.."
0FA60	73	B4	FE	73	B8	F4	24	73	B8	F6	26	73	B1	FE	FD	CA	s..s..$s..&s....
0FA70	FB	00	D9	0B	FE	7D	F2	04	F3	FF	7C	F2	00	D1	0B	05	.....}....|.....
0FA80	6E	0A	D1	0B	07	6E	05	D9	00	7D	F2	73	B0	05	73	B2	n....n...}.s..s.
0FA90	07	77	40	0B	00	77	E2	7D	E2	00	00	7D	E2	06	00	FD	.w@..w.}...}....
0FAA0	46	FD	00	7D	B4	73	0B	05	73	B1	05	7D	51	06	00	F5	F..}.s..s..}Q...
0FAB0	1E	00	D1	00	6E	05	D9	00	7D	F2	73	B1	05	D9	03	73	....n...}.s....s
0FAC0	04	73	18	02	73	38	04	73	40	C9	60	7D	F2	04	7C	F2	.s..s8.s@.`}..|.
0FAD0	08	73	1B	FC	D9	0B	FE	D1	0B	FE	69	0C	6C	06	D1	0B	.s........i.l...
0FAE0	FC	69	05	D9	00	7D	F2	73	B0	FC	73	B2	FE	C9	50	77	.i...}.s..s...Pw
0FAF0	E2	7D	E2	00	00	7D	E2	06	00	FD	46	FD	00	7D	B4	73	.}...}....F..}.s
0FB00	0B	F8	73	2B	FA	A2	06	04	73	F0	06	04	D1	00	6A	4A	..s+....s.....jJ
0FB10	73	05	C1	9B	04	FA	69	38	6C	07	C1	9B	02	F8	69	30	s.....i8l.....i0
0FB20	C1	B9	F8	02	6E	07	C1	B9	FA	04	6A	14	A9	B9	F8	02	....n.....j.....
0FB30	B9	B9	FA	04	73	90	02	7D	50	06	00	A1	05	FE	03	73	....s..}P......s
0FB40	76	A9	BF	F8	08	04	B9	BF	FA	0A	04	73	50	7D	F2	73	v.........sP}.s
0FB50	54	73	64	73	40	73	54	FE	B4	7D	E2	00	00	7D	E2	06	Tsds@sT..}...}..
0FB60	00	73	B0	F8	73	B2	FA	FD	94	FD	00	7D	B4	C1	F2	14	.s..s......}....
0FB70	04	6C	0D	69	07	C1	F0	12	04	6D	05	D9	00	7D	F2	73	.l.i.....m...}.s

Figure 6.12 [Memory] Window

6.11 Referring to variables

When stepping through a program, you can watch that the values of variables used in the user program are changed. 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.

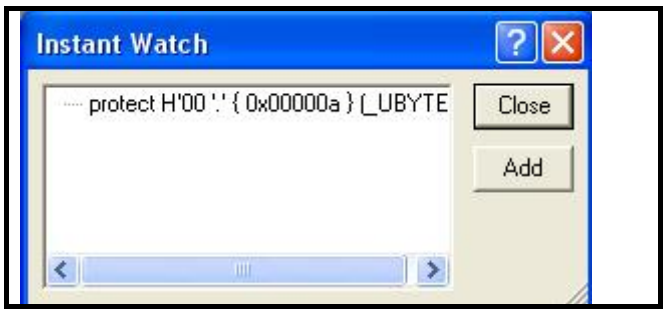


Figure 6.13 [Instant Watch] Dialog Box

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

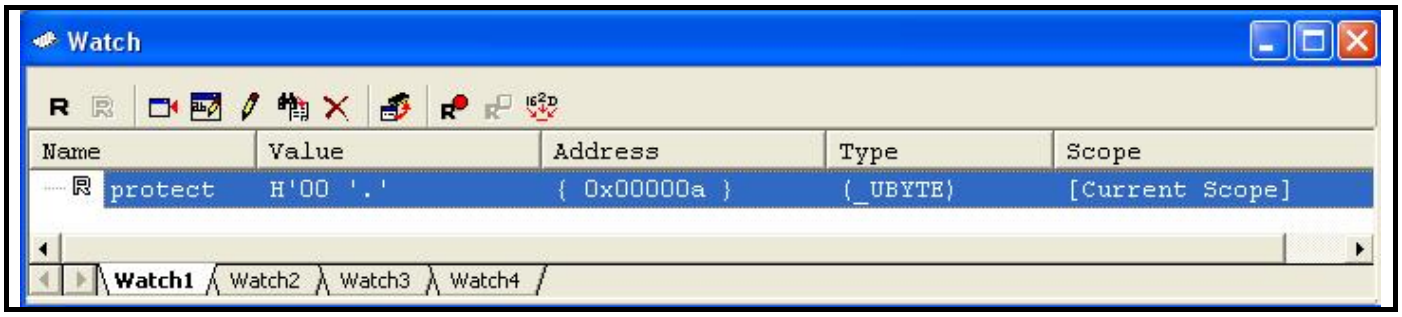


Figure 6.14 [Watch] Window (Array Display)

Alternatively, you can specify a variable name to be added to the [Watch] window. Right-click in the [Watch] window and choose [Add Watch] from the popup menu. The dialog box shown below will be displayed.

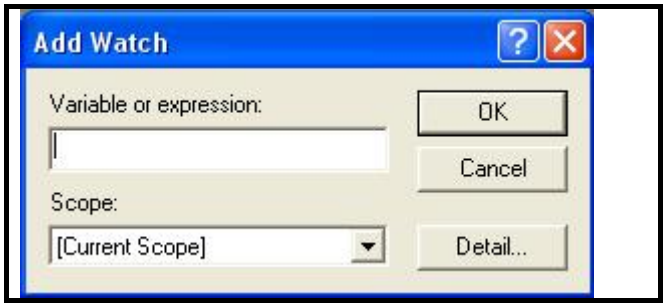


Figure 6.15 [Add Watch] Dialog Box

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.

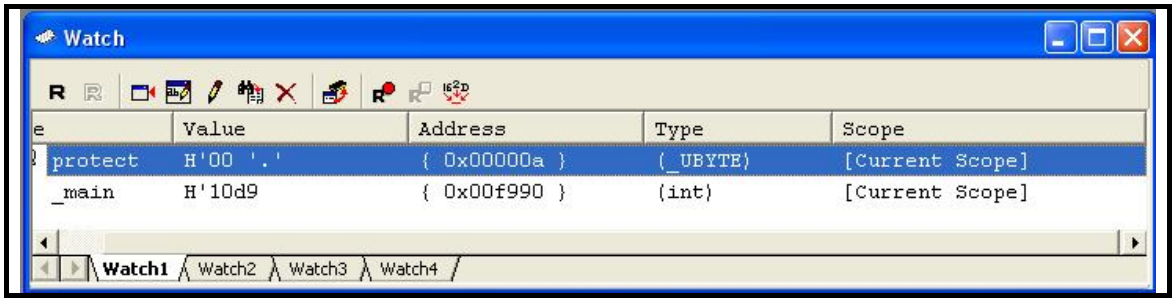


Figure 6.16 [Watch] Window (Showing a Variable)

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.’

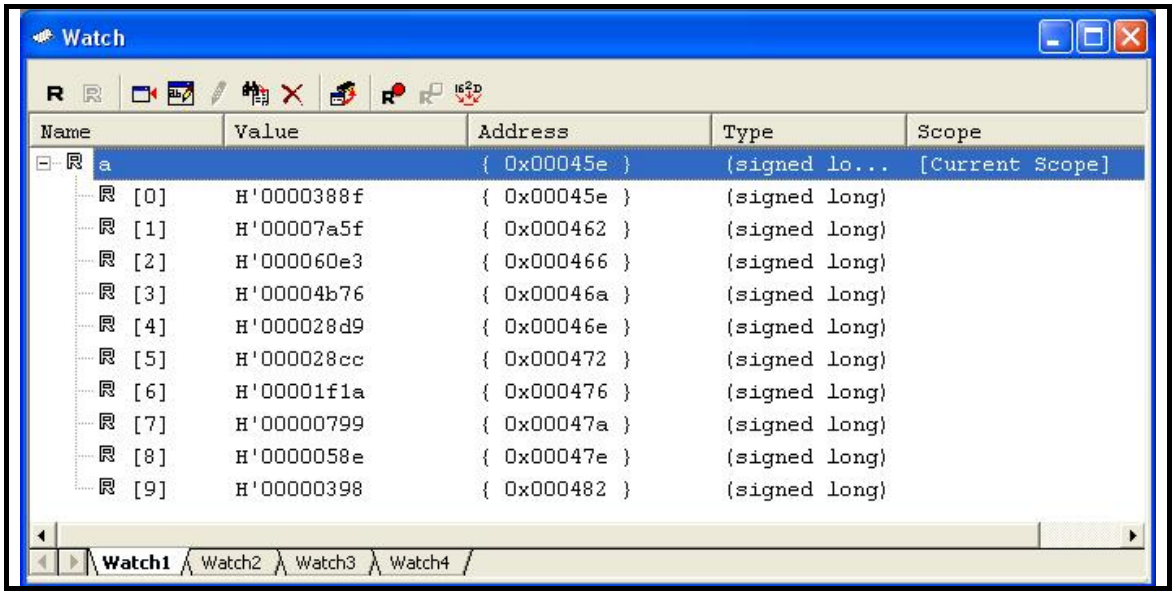


Figure 6.17 [Watch] Window (Showing Array Elements)

6.12 Showing local variables

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

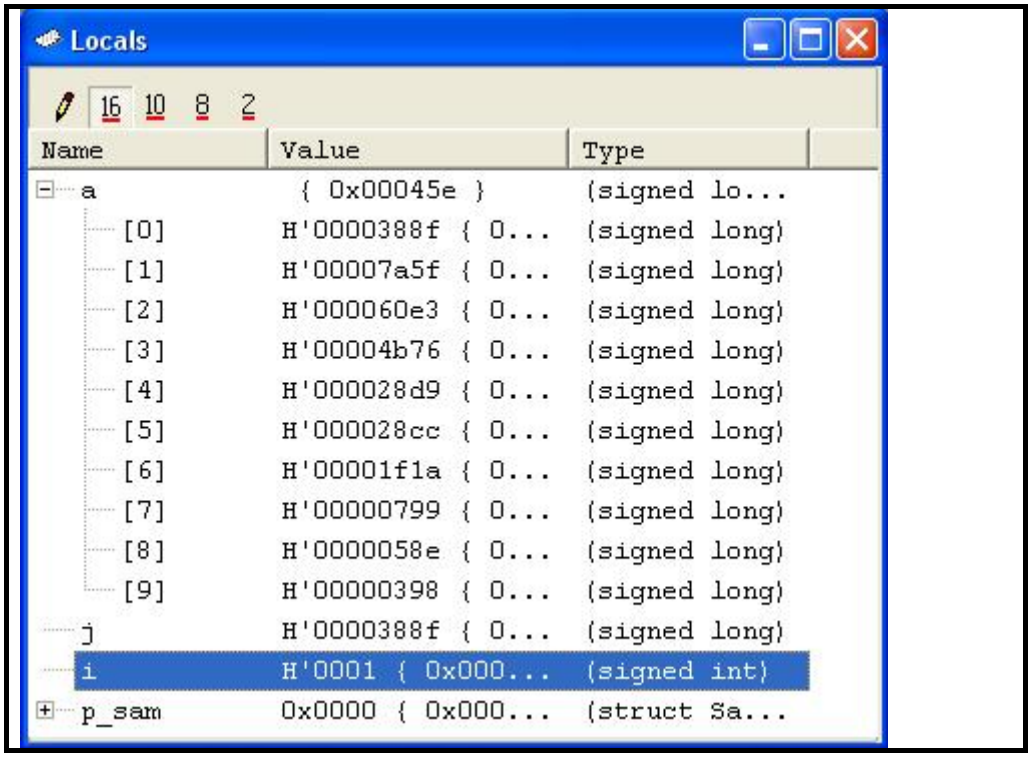


Figure 6.18 [Locals] Window

Click on the "+" mark shown to the left of array a in the [Locals] window to display the elements of class instance 'a'.
Confirm that the random data are being sorted into descending order by inspecting the elements of class instance 'a' before and after execution of the sort function.

6.13 Stepping through a program

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

Table 6.1 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.

6.13.1 Executing [Step In]

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

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

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

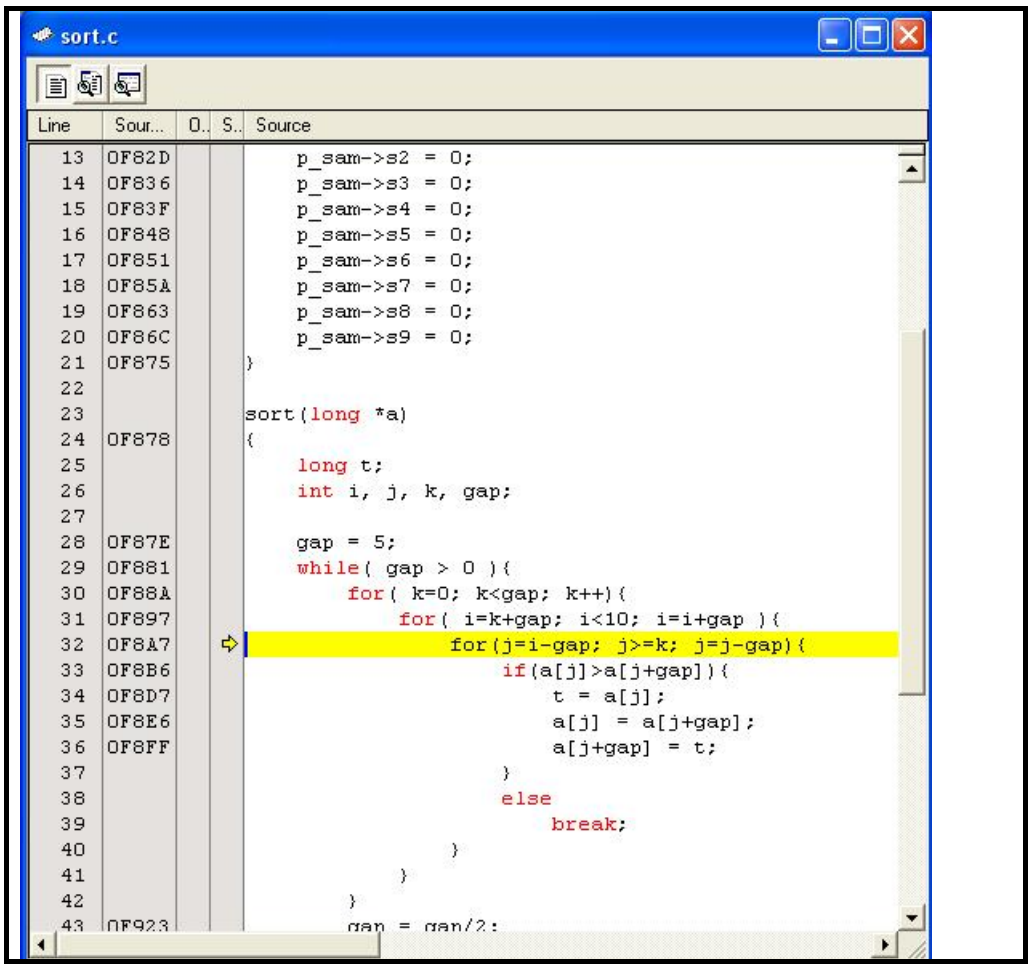


Figure 6.19 [Editor] Window (Step In)

6.13.2 Executing [Step Out]

[Step Out] steps out of the called function and stops at the next statement of the calling statement in the main function.

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

The data of the variable ‘a’ displayed in the [Watch] window will have been sorted into ascending order.

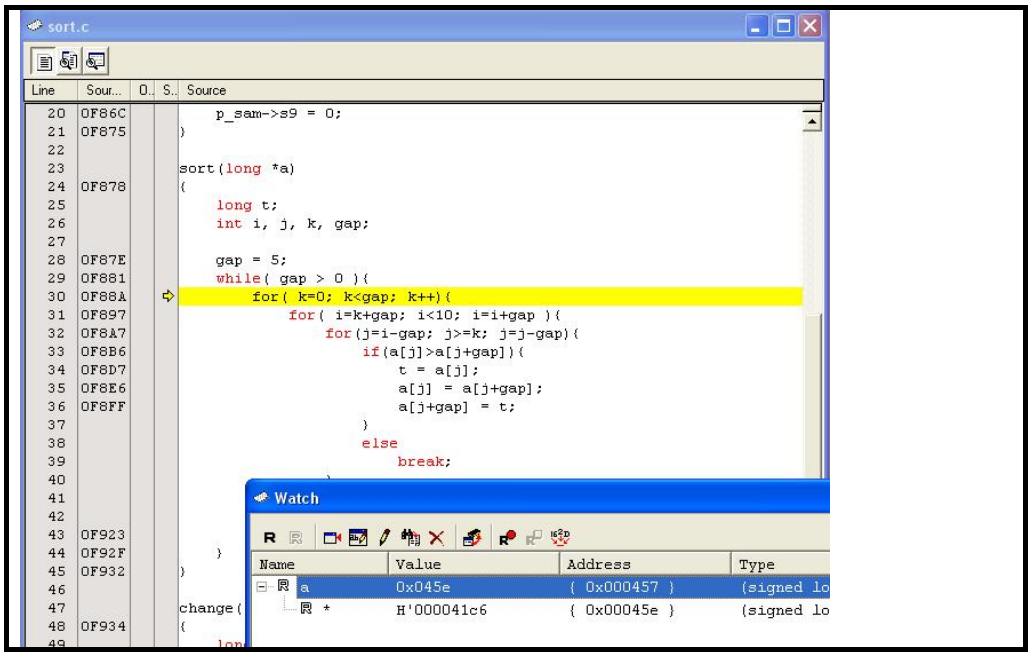


Figure 6.20 [Editor] Window (Step Out)

6.13.3 Executing [Step Over]

[Step Over] executes the whole of a function call as one step and then stops at the next statement of the main program.

To execute all statements in the change function at once, choose [Step Over] from the [Debug] menu or click on the [Step Over] toolbar button ( ).

The data of the variable ‘a’ displayed in the [Watch] window will have been sorted into descending order.

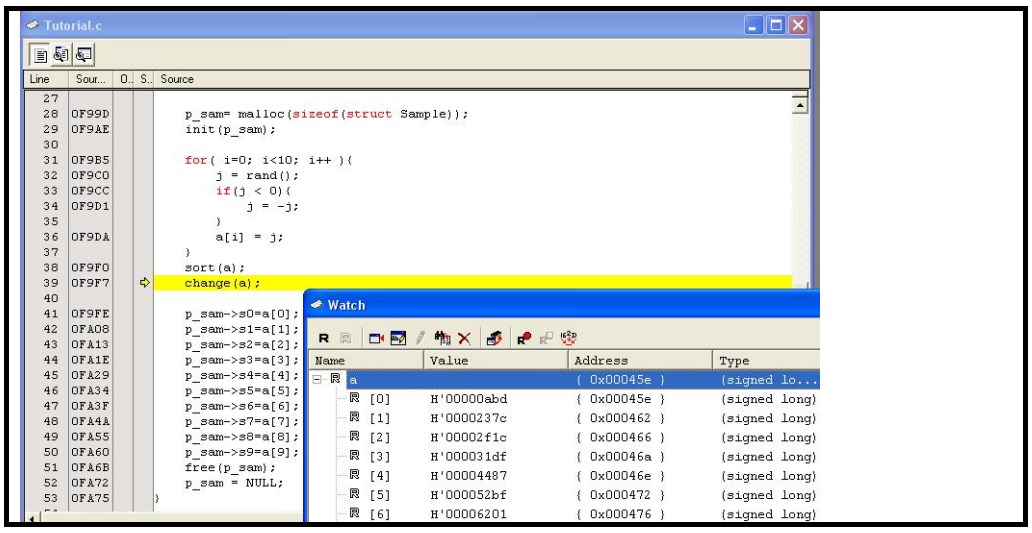


Figure 6.21 [Editor] Window (Step Over)

6.14 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 ( ).

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

6.15 On-chip break facility

The on-chip break facility is available if it is supported by the MCU. An on-chip break causes the program to stop when it executes the instruction at a specified address (instruction fetch) or reads from or writes to a specified memory location (data access).

6.15.1 Stopping a program when it executes the instruction at a specified address (pre-PC break)

It's easy to set an instruction-fetch event as an on-chip breakpoint 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 [On-Chip Breakpoint] column which corresponds to the source line containing the call of the sort function.
The source line that includes the sort function will be marked with ●, indicating that an on-chip breakpoint that will cause a program to stop when it fetches an instruction has been set there.

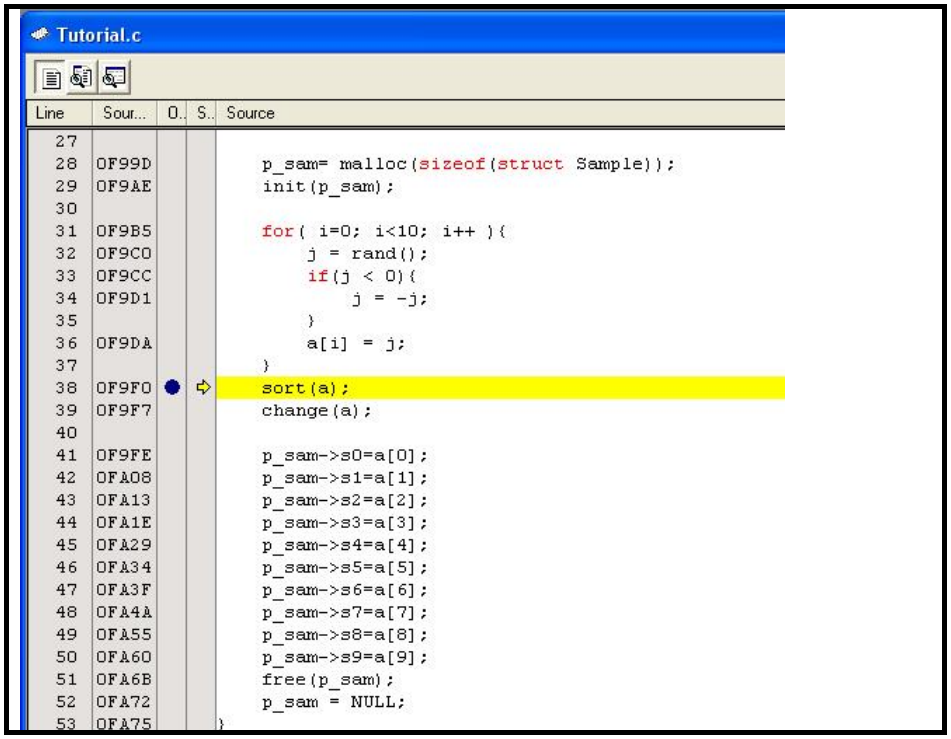


Figure 6.22 [Editor] Window (Setting an On-Chip Breakpoint)

6.15.2 Stopping a program when It accesses memory

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

Choose [View -> Event -> On-chip Event] to open the [On-Chip Event] dialog box.

Select the [Event] tab of the [On-Chip Event] dialog box.

Select a global variable in the [Editor] window, and drag-and-drop the selected variable into the [Event A] list box in the [Event] tab so that the program will stop when it reads or writes the value of that variable.

Select [Event A] from the [Break] drop-down list box in the [Break Condition] group box, and click on the [Apply] button.

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

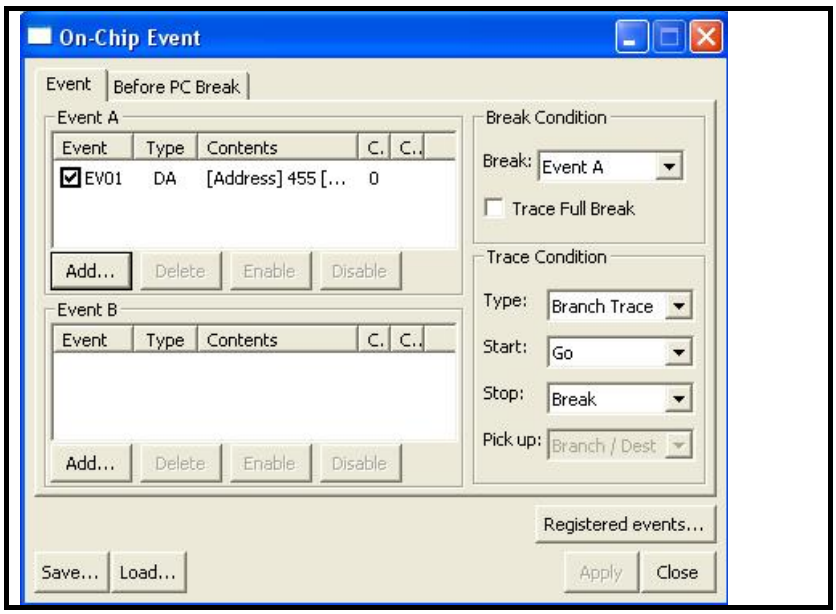


Figure 6.23 [On-Chip Event] Dialog Box

Note:

[*1] Only global variables of 1 or 2 bytes in size can be registered.
Local variables cannot be set as on-chip break conditions.

6.16 Tracing facility

The E1 and E20 emulators allow you to use the tracing facility which uses the trace buffer in the MCU.
The R8C E1/E20 Emulator Debugger can display the branch or data access information.
The contents of the displayed information and the number of cycles that can be acquired vary with the MCU.

The following is an example of settings.
Choose [View -> Code -> Trace] or click on the [Trace] toolbar button ( ).
The [Trace] window will be displayed.

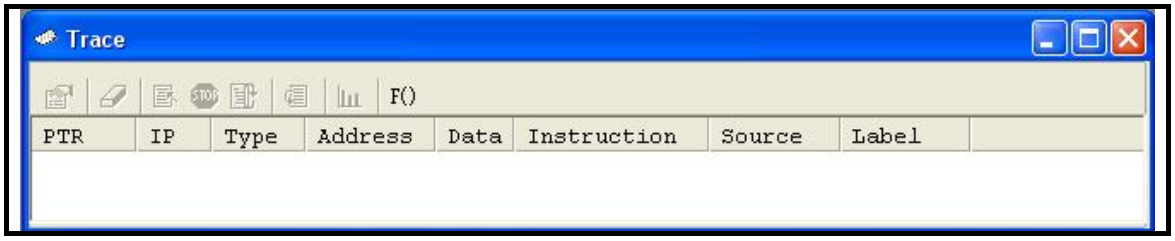


Figure 6.24 [Trace] Window

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

6.16.1 Showing the information acquired in tracing

Trace information from when the trace start condition is encountered until when the trace stop condition is encountered is acquired. Here, let's take the following conditions as an example.

Trace start condition: when user program execution starts

Trace stop condition: when user program execution stops

(1) Choose [View -> Event -> On-Chip Event] or click on the [On-Chip Event] button ( ).

(2) Check the following options are selected for the trace condition, and click the [Close] button.

Type: Branch Trace
Start: Go
Stop: Break

If you have changed any settings in the [On-Chip Event] dialog box, click the [Apply] button first, then the [Close] button.

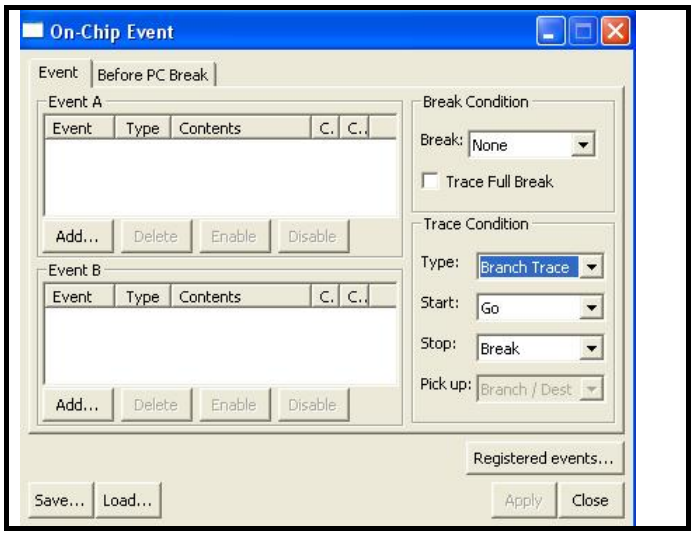


Figure 6.25 [On-Chip Event] Dialog Box

(3) Set a S/W breakpoint on the line of the tutorial function: p_sam->s0=a[0];".

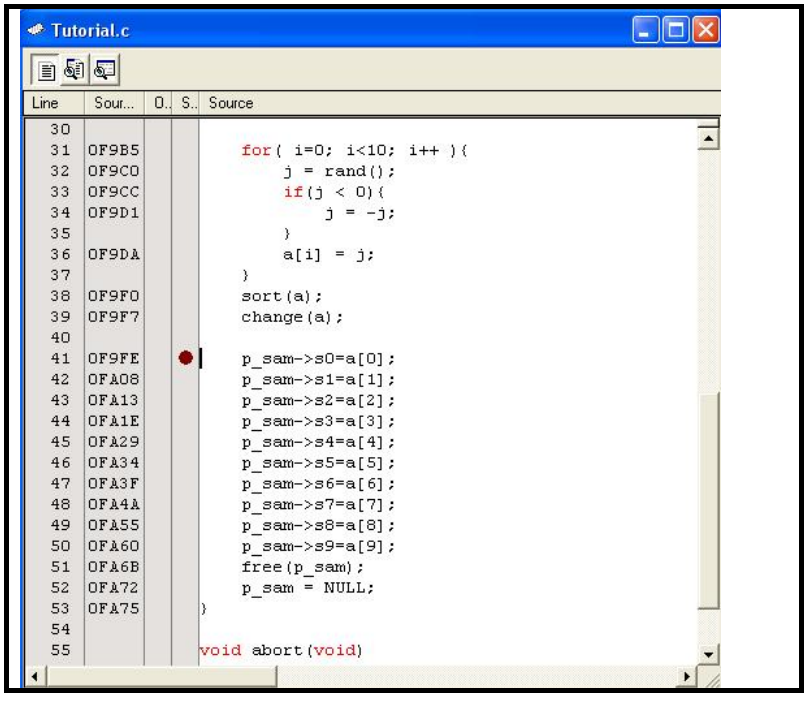


Figure 6.26 [Editor] Window (Setting a S/W Breakpoint in the Tutorial Function)

(4) 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.

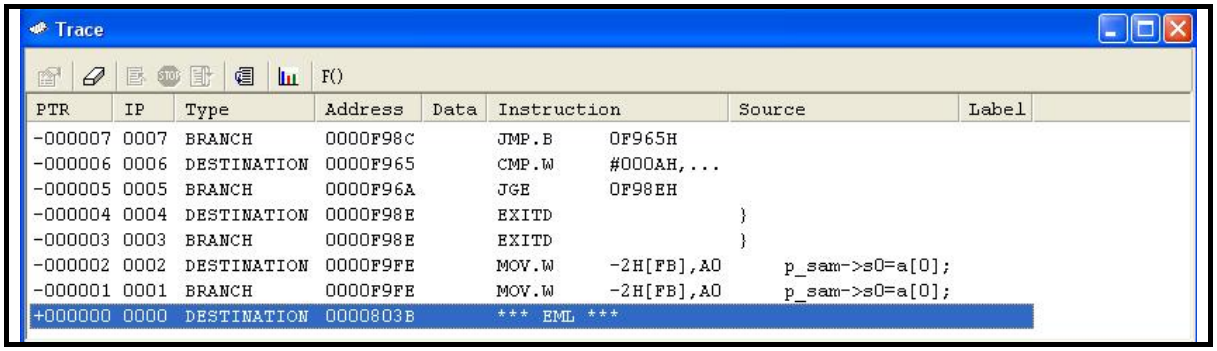


Figure 6.27 [Trace] Window

6.17 Stack trace facility

Stack information can be used to find out which function called the function corresponding to the current PC value. [*1]
Set a S/W breakpoint in any line of the sort function by double-clicking on the corresponding row in the [S/W Breakpoints] column. Choose [Reset Go] from the [Debug] menu.

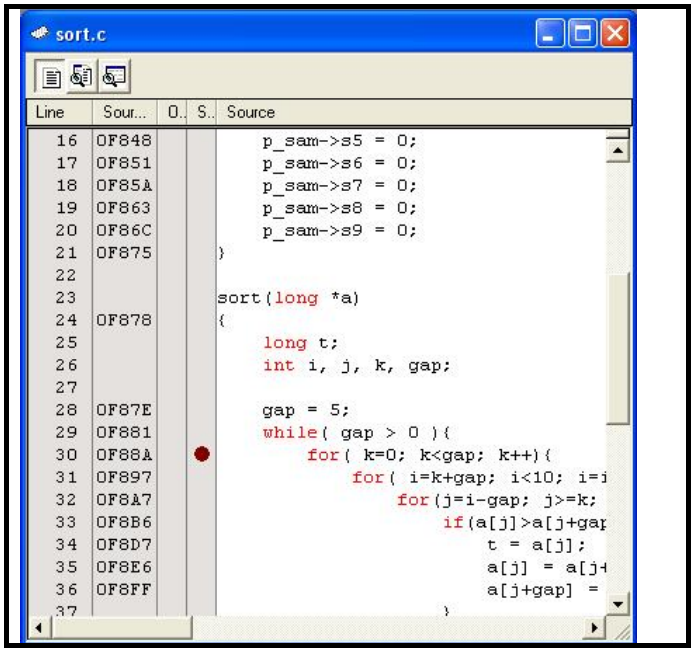


Figure 6.28 [Editor] Window (Setting a S/W Breakpoint)

Note:
[*1] This function can be used only when the load module that has the IEEE695-type debugging information is loaded.

After the break, choose [View -> Code -> Stack Trace] to open the [Stack Trace] window.

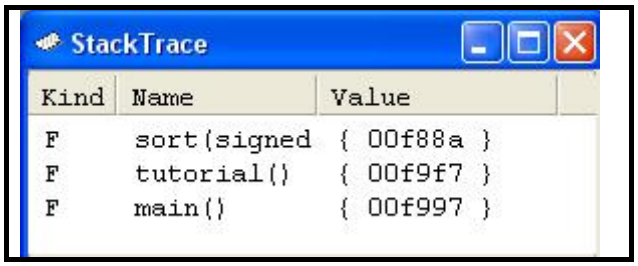


Figure 6.29 [Stack Trace] Window

You will see that the current PC value is within the sort function, and that the sort function was called from the tutorial function.
Clear the S/W breakpoint that you set on a line of the sort function by again double-clicking on the corresponding row in the [S/W Breakpoints] column.

6.18 What next?

In this tutorial, we have introduced to you several features of the E1 and E20 emulators and usage of the High-performance Embedded Workshop.
The emulation facilities of the E1 and E20 emulators 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.

7. Notes on Using the E1 or E20 Emulator

7.1 MCU resources used by the emulator

7.1.1 Program area for the emulator

Table 7.1 to Table 7.16 list the program area for the emulator.

Do not change this area, otherwise the emulator will not control the MCU. In this case, disconnect the debugger and then reconnect it.

Table 7.1 Program Area for the Emulator (R8C/3xC)

Group	Part No.	[Initial Settings] dialog box		Internal ROM Size		Program Area for the Emulator	
		MCU group	Device	Program ROM	Data Flash	Vector Area	ROM Area
R8C/32C	R5F21321C	R8C/3xC (48 pins or less) Group	R5F213x1C	4 KB	1 KB (4 blocks)	FFE4h - FFE7h, FFE8h - FFEBh, FFECh - FFEFh, FFF4h - FFF7h, FFFCh - FFFEh	-
	R5F21322C		R5F213x2C	8 KB			-
	R5F21324C		R5F213x4C	16 KB			-
R8C/33C	R5F21331C		R5F213x1C	4 KB			-
	R5F21332C		R5F213x2C	8 KB			-
	R5F21334C		R5F213x4C	16 KB			-
	R5F21335C		R5F213x5C	24 KB			-
	R5F21336C		R5F213x6C	32 KB			2 KB of the ROM area [*1]
R8C/34C	R5F21344C		R5F213x4C	16 KB			-
	R5F21345C		R5F213x5C	24 KB			-
	R5F21346C		R5F213x6C	32 KB			2 KB of the ROM area [*1]
R8C/35C	R5F21354C	R8C/3xC (52 pin) Group	R5F213x4C	16 KB			-
	R5F21355C		R5F213x5C	24 KB			-
	R5F21356C		R5F213x6C	32 KB			2 KB of the ROM area [*1]
	R5F21357C		R5F213x7C	48 KB			-
	R5F21358C		R5F213x8C	64 KB			-
	R5F2135AC		R5F213xAC	96 KB			-
	R5F2135CC		R5F213xCC	128 KB			2 KB of the ROM area [*1]
R8C/36C	R5F21364C	R8C/3xC (64 pins or more) Group	R5F213x4C	16 KB			-
	R5F21365C		R5F213x5C	24 KB			-
	R5F21366C		R5F213x6C	32 KB			-
	R5F21367C		R5F213x7C	48 KB			-
	R5F21368C		R5F213x8C	64 KB			-
	R5F2136AC		R5F213xAC	96 KB			-
	R5F2136CC		R5F213xCC	128 KB			2 KB of the ROM area [*1]
R8C/38C	R5F21386C		R5F213x6C	32 KB			-
	R5F21387C		R5F213x7C	48 KB			-
	R5F21388C		R5F213x8C	64 KB			-
	R5F2138AC		R5F213xAC	96 KB			-
	R5F2138CC		R5F213xCC	128 KB			2 KB of the ROM area [*1]
R8C/3GC	R5F213G1C	R8C/3xC (48 pins or less) Group	R5F213x1C	4 KB			-
	R5F213G2C		R5F213x2C	8 KB			-
	R5F213G4C		R5F213x4C	16 KB			-
	R5F213G5C		R5F213x5C	24 KB			-
	R5F213G6C		R5F213x6C	32 KB			2 KB of the ROM area [*1]
R8C/3JC	R5F213J2C		R5F213x2C	8 KB			-
	R5F213J4C		R5F213x4C	16 KB			-
	R5F213J5C		R5F213x5C	24 KB			-
	R5F213J6C		R5F213x6C	32 KB			2 KB of the ROM area [*1]

Note:

[*1] When starting the debugger, the [Configuration Properties] dialog box is displayed. Specify the area which will not be used by the user system. For details, see “4.13.1 [System] tab” on page 42.

Table 7.2 Program Area for the Emulator (R8C/3xM)

Group	Part No.	[Initial Settings] dialog box		Internal ROM Size		Program Area for the Emulator	
		MCU group	Device	Program ROM	Data Flash	Vector Area	ROM Area
R8C/32M	R5F21321M	R8C/3xM (48pins or less) Group	R5F213x1M	4 KB	1 KB (4 blocks)	FFE4h - FFE7h, FFE8h - FFEBh, FFECh - FFEFh, FFF4h - FFF7h, FFFCh - FFFEh	-
	R5F21322M		R5F213x2M	8 KB			-
	R5F21324M		R5F213x4M	16 KB			-
R8C/33M	R5F21331M		R5F213x1M	4 KB			-
	R5F21332M		R5F213x2M	8 KB			-
	R5F21334M		R5F213x4M	16 KB			-
	R5F21335M		R5F213x5M	24 KB			-
	R5F21336M		R5F213x6M	32 KB			2 KB of the ROM area [*1]
R8C/34M	R5F21344M		R5F213x4M	16 KB			-
	R5F21345M		R5F213x5M	24 KB			-
	R5F21346M		R5F213x6M	32 KB			2 KB of the ROM area [*1]
R8C/35M	R5F21354M	R8C/3xM(52pin) Group	R5F213x4M	16 KB			-
	R5F21355M		R5F213x5M	24 KB			-
	R5F21356M		R5F213x6M	32 KB			2 KB of the ROM area [*1]
	R5F21357M		R5F213x7M	48 KB			-
	R5F21358M		R5F213x8M	64 KB			-
	R5F2135AM		R5F213xAM	96 KB			-
	R5F2135CM		R5F213xCM	128 KB			2 KB of the ROM area [*1]
R8C/36M	R5F21364M	R8C/3xM (64pins or more) Group	R5F213x4M	16 KB			-
	R5F21365M		R5F213x5M	24 KB			-
	R5F21366M		R5F213x6M	32 KB			-
	R5F21367M		R5F213x7M	48 KB			-
	R5F21368M		R5F213x8M	64 KB			-
	R5F2136AM		R5F213xAM	96 KB			-
	R5F2136CM		R5F213xCM	128 KB			2 KB of the ROM area [*1]
R8C/38M	R5F21386M		R5F213x6M	32 KB			-
	R5F21387M		R5F213x7M	48 KB			-
	R5F21388M		R5F213x8C	64 KB			-
	R5F2138AM		R5F213xAM	96 KB			-
	R5F2138CM		R5F213xCM	128 KB			2 KB of the ROM area [*1]
R8C/3GM	R5F213G2M	R8C/3xM (48pins or less) Group	R5F213x2M	8 KB			-
	R5F213G4M		R5F213x4M	16 KB			-
	R5F213G5M		R5F213x5M	24 KB			-
	R5F213G6M		R5F213x6M	32 KB			2 KB of the ROM area [*1]
R8C/3JM	R5F213J2M	R8C/3xM (48pins or less) Group	R5F213x2M	8 KB			-
	R5F213J4M		R5F213x4M	16 KB			-
	R5F213J5M		R5F213x5M	24 KB			-
	R5F213J6M		R5F213x6M	32 KB			2 KB of the ROM area [*1]

Note:
When starting the debugger, the [Configuration Properties] dialog box is displayed. Specify the area which will not be used by the user system. For details, see “4.13.1 [System] tab” on page 42.

Table 7.3 Program Area for the Emulator (R8C/3xW)

Group	Part No.	[Initial Settings] dialog box		Internal ROM Size		Program Area for the Emulator	
		MCU group	Device	Program ROM	Data Flash	Vector Area	ROM Area
R8C/34W	R5F21346W	R8C/3xW, R8C/3xX, R8C/3xY, R8C/3xZ Groups	R5F213x6W	32 KB	1 KB (4 blocks)	FFE4h - FFE7h FFE8h - FFEBh, FFECh - FFEFh, FFF4h - FFF7h, FFFCh - FFFEh,	-
	R5F21347W		R5F213x7W	48 KB			
	R5F21348W		R5F213x8W	64 KB			
	R5F2134AW		R5F213xAW	96 KB			
	R5F2134CW		R5F213xCW	128 KB			2 KB of the ROM area [*1]
R8C/36W	R5F21368W		R5F213x8W	64 KB			-
	R5F2136AW		R5F213xAW	96 KB			
	R5F2136CW		R5F213xCW	128 KB			2 KB of the ROM area [*1]
R8C/38W	R5F21388W		R5F213x8W	64 KB			-
	R5F2138AW		R5F213xAW	96 KB			
	R5F2138CW		R5F213xCW	128 KB			2 KB of the ROM area [*1]

Table 7.4 Program Area for the Emulator (R8C/3xX)

Group	Part No.	[Initial Settings] dialog box		Internal ROM Size		Program Area for the Emulator	
		MCU group	Device	Program ROM	Data Flash	Vector Area	ROM Area
R8C/34X	R5F21346X	R8C/3xW, R8C/3xX, R8C/3xY, R8C/3xZ Groups	R5F213x6X	32 KB	-	FFE4h - FFE7h, FFE8h - FFEBh, FFECh - FFEFh, FFF4h - FFF7h, FFFCh - FFFEh	-
	R5F21347X		R5F213x7X	48 KB			
	R5F21348X		R5F213x8X	64 KB			
	R5F2134AX		R5F213xAX	96 KB			
	R5F2134CX		R5F213xCX	128 KB			2 KB of the ROM area [*1]
R8C/36X	R5F21368X		R5F213x8X	64 KB			-
	R5F2136AX		R5F213xAX	96 KB			
	R5F2136CX		R5F213xCX	128 KB			2 KB of the ROM area [*1]
R8C/38X	R5F21388X		R5F213x8X	64 KB			-
	R5F2138AX		R5F213xAX	96 KB			
	R5F2138CX		R5F213xCX	128 KB			2 KB of the ROM area [*1]

Note:
When starting the debugger, the [Configuration Properties] dialog box is displayed. Specify the area which will not be used by the user system. For details, see “4.13.1 [System] tab” on page 42.

Table 7.5 Program Area for the Emulator (R8C/3xY)

Group	Part No.	[Initial Settings] dialog box		Internal ROM Size		Program Area for the Emulator	
		MCU group	Device	Program ROM	Data Flash	Vector Area	ROM Area
R8C/34Y	R5F21346Y	R8C/3xW, R8C/3xX, R8C/3xY, R8C/3xZ Groups	R5F213x6Y	32 KB	1 KB (4 blocks)	FFE4h - FFE7h, FFE8h - FFEBh, FFECh - FFEFh, FFF4h - FFF7h, FFFCh - FFFEh	-
	R5F21347Y		R5F213x7Y	48 KB			
	R5F21348Y		R5F213x8Y	64 KB			
	R5F2134AY		R5F213xAY	96 KB			
	R5F2134CY		R5F213xCY	128 KB			2 KB of the ROM area [*1]
R8C/36Y	R5F21368Y		R5F213x8Y	64 KB			-
	R5F2136AY		R5F213xAY	96 KB			
	R5F2136CY		R5F213xCY	128 KB			2 KB of the ROM area [*1]
R8C/38Y	R5F21388Y		R5F213x8Y	64 KB			-
	R5F2138AY		R5F213xAY	96 KB			
	R5F2138CY		R5F213xCY	128 KB			2 KB of the ROM area [*1]

Table 7.6 Program Area for the Emulator (R8C/3xZ)

Group	Part No.	[Initial Settings] dialog box		Internal ROM Size		Program Area for the Emulator	
		MCU group	Device	Program ROM	Data Flash	Vector Area	Device
R8C/34Z	R5F21346Z	R8C/3xW, R8C/3xX, R8C/3xY, R8C/3xZ Groups	R5F213x6Z	32 KB	-	FFE4h - FFE7h, FFE8h - FFEBh, FFECh - FFEFh, FFF4h - FFF7h, FFFCh - FFFEh	-
	R5F21347Z		R5F213x7Z	48 KB			
	R5F21348Z		R5F213x8Z	64 KB			
	R5F2134AZ		R5F213xAZ	96 KB			
	R5F2134CZ		R5F213xCZ	128 KB			2 KB of the ROM area [*1]
R8C/36Z	R5F21368Z		R5F213x8Z	64 KB			-
	R5F2136AZ		R5F213xAZ	96 KB			
	R5F2136CZ		R5F213xCZ	128 KB			2 KB of the ROM area [*1]
R8C/38Z	R5F21388Z		R5F213x8Z	64 KB			-
	R5F2138AZ		R5F213xAZ	96 KB			
	R5F2138CZ		R5F213xCZ	128 KB			2 KB of the ROM area [*1]

Table 7.7 Program Area for the Emulator (R8C/3xGHPR)

Group	Part No.	[Initial Settings] dialog box		Internal ROM Size		Program Area for the Emulator	
		MCU group	Device	Program ROM	Data Flash	Vector Area	ROM Area
R8C/32G	R5F21324G	R8C/3xG, R8C/3xH, R8C/3xP, R8C/3xR Groups	R5F213x4G	16 KB	1 KB (4 blocks)	FFE4h - FFE7h, FFE8h - FFEBh, FFECh - FFEFh, FFF4h - FFF7h, FFFCh - FFFEh	–
	R5F21326G		R5F213x6G	32 KB			2 KB of the ROM area [*1]
R8C/33G	R5F21334G		R5F213x4G	16 KB			–
	R5F21336G		R5F213x6G	32 KB			2 KB of the ROM area [*1]
R8C/34P	R5F21344P		R5F213x4P	16 KB			–
	R5F21346P		R5F213x6P	32 KB			2 KB of the ROM area [*1]
R8C/32H	R5F21324H		R5F213x4H	16 KB	-		–
	R5F21326H		R5F213x6H	32 KB			2 KB of the ROM area [*1]
R8C/33H	R5F21334H		R5F213x4H	16 KB			–
	R5F21336H		R5F213x6H	32 KB			2 KB of the ROM area [*1]
R8C/34R	R5F21344R		R5F213x4R	16 KB			–
	R5F21346R		R5F213x6R	32 KB			2 KB of the ROM area [*1]

Note:

[*1] When starting the debugger, the [Configuration Properties] dialog box is displayed. Specify the area which will not be used by the user system. For details, see “4.13.1 [System] tab” on page 42.

Table 7.8 Program Area for the Emulator (R8C/3xT)

Group	Part No.	[Initial Settings] dialog box		Internal ROM Size		Program Area for the Emulator	
		MCU group	Device	Program ROM	Data Flash	Vector Area	ROM Area
R8C/33T	R5F21334T	R8C/3xT Group	R5F213x4T	16 KB	1 KB (4 blocks)	FFE4h - FFE7h, FFE8h - FFEBh, FFECh - FFEFh, FFF4h - FFF7h, FFFCh - FFFEh	-
	R5F21335T		R5F213x5T	24 KB			-
	R5F21336T		R5F213x6T	32 KB			2 KB of the ROM area [*1]
R8C/3JT	R5F213J4T		R5F213x4T	16 KB			-
	R5F213J5T		R5F213x5T	24 KB			-
	R5F213J6T		R5F213x6T	32 KB			2 KB of the ROM area [*1]

Table 7.9 Program Area for the Emulator (R8C/3NT)

Group	Part No.	[Initial Settings] dialog box		Internal ROM Size		Program Area for the Emulator	
		MCU group	Device	Program ROM	Data Flash	Vector Area	ROM Area
R8C/3NT	R5F213N7T	R8C/3NT Group	R5F213N7T	48 KB	1 KB (4 blocks)	FFE4h - FFE7h, FFE8h - FFEBh, FFECh - FFEFh, FFF4h - FFF7h, FFFCh - FFFEh	-
	R5F213N8T		R5F213N8T	64 KB			-
	R5F213NAT		R5F213NAT	96 KB			-
	R5F213NCT		R5F213NCT	128 KB			2 KB of the ROM area [*1]

Table 7.10 Program Area for the Emulator (R8C/3MQ)

Group	Part No.	[Initial Settings] dialog box		Internal ROM Size		Program Area for the Emulator	
		MCU group	Device	Program ROM	Data Flash	Vector Area	ROM Area
R8C/3MQ	R5F213M6Q	R8C/3MQ Group	R5F213M6Q	32 KB	1 KB (4 blocks)	FFE4h - FFE7h, FFE8h - FFEBh, FFECh - FFEFh, FFF4h - FFF7h, FFFCh - FFFEh	-
	R5F213M7Q		R5F213M7Q	48 KB			-
	R5F213M8Q		R5F213M8Q	64 KB			-
	R5F213MAQ		R5F213MAQ	96 KB			-
	R5F213MCQ		R5F213MCQ	~~128 KB~~ 112 KB [*2]			2 KB of the ROM area [*1]

Table 7.11 Program Area for the Emulator (R8C/3xUK)

Group	Part No.	[Initial Settings] dialog box		Internal ROM Size		Program Area for the Emulator	
		MCU group	Device	Program ROM	Data Flash	Vector Area	ROM Area
R8C/34U	R5F21346U	R8C/3xUK Group	R5F213x6U	32 KB	1 KB (4 blocks)	FFE4h - FFE7h, FFE8h - FFEBh, FFECh - FFEFh, FFF4h - FFF7h, FFFCh - FFFEh	-
	R5F21348U		R5F213x8U	64 KB			-
R8C/34K	R5F21348K		R5F213x8K	64 KB			-
	R5F2134CK		R5F213xCK	128 KB			2 KB of the ROM area [*1]
R8C/3MU	R5F213M6U		R5F213x6U	32 KB			-
	R5F213M8U		R5F213x8U	64 KB			-
R8C/3MK	R5F213M8K		R5F213x8K	64 KB			-
	R5F213MCK		R5F213xCK	128 KB			2 KB of the ROM area [*1]

Note:

[*1] When starting the debugger, the [Configuration Properties] dialog box is displayed. Specify the area which will not be used by the user system. For details, see “4.13.1 [System] tab” on page 42.

[*2] The program ROM for the R5F213MCQ has been changed from 128 KB to 112 KB. For details, refer to our technical notification, which can be accessed from the following URL.
http://documentation.renesas.com/doc/products/mpumcu/doc/r8c/tnr8ca028ae.pdf
The debugger does not have the choice for 112 KB program ROM. When using the R5F213MCQ, follow the instructions in "3 Usage note for the development tool" in the technical notification.

Table 7.12 Program Area for the Emulator (R8C/L3xC)

Group	Part No.	[Initial Settings] dialog box		Internal ROM Size		Program Area for the Emulator	
		MCU group	Device	Program ROM	Data Flash	Vector Area	ROM Area
R8C/L35C	R5F2L357C	R8C/L3xC Group	R5F2L3x7C	48 KB	1 KB (4 blocks)	FFE4h - FFE7h, FFE8h - FFEBh, FFECh - FFEFh, FFF4h - FFF7h, FFFCh - FFFEh	-
	R5F2L358C		R5F2L3x8C	64 KB			
	R5F2L35AC		R5F2L3xAC	96 KB			
	R5F2L35CC		R5F2L3xCC	128 KB			2 KB of the ROM area [*1]
R8C/L36C	R5F2L367C		R5F2L3x7C	48 KB			-
	R5F2L368C		R5F2L3x8C	64 KB			
	R5F2L36AC		R5F2L3xAC	96 KB			
	R5F2L36CC		R5F2L3xCC	128 KB			2 KB of the ROM area [*1]
R8C/L38C	R5F2L387C		R5F2L3x7C	48 KB			-
	R5F2L388C		R5F2L3x8C	64 KB			
	R5F2L38AC		R5F2L3xAC	96 KB			
	R5F2L38CC		R5F2L3xCC	128 KB			2 KB of the ROM area [*1]
R8C/L3AC	R5F2L3A7C		R5F2L3x7C	48 KB			-
	R5F2L3A8C		R5F2L3x8C	64 KB			
	R5F2L3AAC		R5F2L3xAC	96 KB			
	R5F2L3ACC		R5F2L3xCC	128 KB			2 KB of the ROM area [*1]

Table 7.13 Program Area for the Emulator (R8C/L3xM)

Group	Part No.	[Initial Settings] dialog box		Internal ROM Size		Program Area for the Emulator	
		MCU group	Device	Program ROM	Data Flash	Vector Area	ROM Area
R8C/L35M	R5F2L357M	R8C/L3xM Group	R5F2L3x7M	48 KB	1 KB (4 blocks)	FFE4h - FFE7h, FFE8h - FFEBh, FFECh - FFEFh, FFF4h - FFF7h, FFFCh - FFFEh	-
	R5F2L358M		R5F2L3x8M	64 KB			
	R5F2L35AM		R5F2L3xAM	96 KB			
	R5F2L35CM		R5F2L3xCM	128 KB			2 KB of the ROM area [*1]
R8C/L36M	R5F2L367M		R5F2L3x7M	48 KB			-
	R5F2L368M		R5F2L3x8M	64 KB			
	R5F2L36AM		R5F2L3xAM	96 KB			
	R5F2L36CM		R5F2L3xCM	128 KB			2 KB of the ROM area [*1]
R8C/L38M	R5F2L387M		R5F2L3x7M	48 KB			-
	R5F2L388M		R5F2L3x8M	64 KB			
	R5F2L38AM		R5F2L3xAM	96 KB			
	R5F2L38CM		R5F2L3xCM	128 KB			2 KB of the ROM area [*1]
R8C/L3AM	R5F2L3A7M		R5F2L3x7M	48 KB			-
	R5F2L3A8M		R5F2L3x8M	64 KB			
	R5F2L3AAM		R5F2L3xAM	96 KB			
	R5F2L3ACM		R5F2L3xCM	128 KB			2 KB of the ROM area [*1]

Note:

[*1] When starting the debugger, the [Configuration Properties] dialog box is displayed. Specify the area which will not be used by the user system. For details, see “4.13.1 [System] tab” on page 42.

Table 7.14 Program Area for the Emulator (R8C/LA3A, LA5A)

Group	Part No.	[Initial Settings] dialog box		Internal ROM Size		Program Area for the Emulator	
		MCU group	Device	Program ROM	Data Flash	Vector Area	ROM Area
R8C/LA3A	R5F2LA32A	R8C/LA3A, R8C/LA5A Groups	R5F2LAx2A	8 KB	1 KB (2 blocks)	FFE4h - FFE7h, FFE8h - FFEBh, FFECh - FFEFh, FFF4h - FFF7h, FFFCh - FFFEh	-
	R5F2LA34A		R5F2LAx4A	16 KB			
	R5F2LA36A		R5F2LAx6A	32 KB			
	R5F2LA38A		R5F2LAx8A	64 KB			2 KB of the ROM area [*1]
R8C/LA5A	R5F2LA52A		R5F2LAx2A	8 KB			-
	R5F2LA54A		R5F2LAx4A	16 KB			
	R5F2LA56A		R5F2LAx6A	32 KB			
	R5F2LA58A		R5F2LAx8A	64 KB			2 KB of the ROM area [*1]

Table 7.15 Program Area for the Emulator (R8C/LA6A, LA8A)

Group	Part No.	[Initial Settings] dialog box		Internal ROM Size		Program Area for the Emulator	
		MCU group	Device	Program ROM	Data Flash	Vector Area	ROM Area
R8C/LA6A	R5F2LA64A	R8C/LA6A, R8C/LA8A Groups	R5F2LAx4A	16 KB	1 KB (2 blocks)	FFE4h - FFE7h, FFE8h - FFEBh, FFECh - FFEFh, FFF4h - FFF7h, FFFCh - FFFEh	-
	R5F2LA66A		R5F2LAx6A	32 KB			
	R5F2LA67A		R5F2LAx7A	48 KB			
	R5F2LA68A		R5F2LAx8A	64 KB			2 KB of the ROM area [*1]
	R5F2LA6AA		R5F2LAxAA	96 KB	2 KB (2 blocks)		-
	R5F2LA6CA		R5F2LAxCA	128 KB			2 KB of the ROM area [*1]
R8C/LA8A	R5F2LA84A		R5F2LAx4A	16 KB	1 KB (2 blocks)		-
	R5F2LA86A		R5F2LAx6A	32 KB			
	R5F2LA87A		R5F2LAx7A	48 KB			
	R5F2LA88A		R5F2LAx8A	64 KB			2 KB of the ROM area [*1]
	R5F2LA8AA		R5F2LAxAA	96 KB	2 KB (2 blocks)		-
	R5F2LA8CA		R5F2LAxCA	128 KB			2 KB of the ROM area [*1]

Table 7.16 Program Area for the Emulator (R8C/LAPS)

Group	Part No.	[Initial Settings] dialog box		Internal ROM Size		Program Area for the Emulator	
		MCU group	Device	Program ROM	Data Flash	Vector Area	ROM Area
R8C/LAPS	R5F2LAP6S	R8C/LAPS Group	R5F2LAx6S	32 KB	2 KB (2 blocks)	FFE4h - FFE7h, FFE8h - FFEBh, FFECh - FFEFh, FFF4h - FFF7h, FFFCh - FFFEh	-
	R5F2LAP7S		R5F2LAx7S	48 KB			
	R5F2LAP8S		R5F2LAx8S	64 KB			
	R5F2LAPAS		R5F2LAxAS	96 KB			
	R5F2LAPCS		R5F2LAxCS	128 KB			2 KB of the ROM area [*1]

Note:

[*1] When starting the debugger, the [Configuration Properties] dialog box is displayed. Specify the area which will not be used by the user system. For details, see “4.13.1 [System] tab” on page 42.

7.1.2 Pins used by the E1 or E20 emulator

The emulator controls the MCUs by using the following pins depending on the usage.

- RESET# pin and MODE pin

7.1.3 Interrupts (unusable)

The BRK instruction interrupt, address match interrupt, single-step interrupt and address break interrupt are used by the E1/E20 emulator program. Therefore, make sure the user program does not use any of these interrupts. The emulator changes these interrupt vector values to the values to be used by the emulator. No problems occur if the interrupt vector values are written in the user program. These vector addresses cannot be rewritten in the [Memory] window.

7.1.4 Stack area used by the E1 or E20 emulator

The emulator uses up to 8 bytes of the stack pointer (ISP) during a user program break. Therefore, set aside 8 bytes for the stack area.

7.1.5 SFRs used by the E1/E20 emulator program

The SFRs listed in Table 7.17 are used by the E1/E20 emulator program as well as the user program.

- Do not change the value in the [Memory] window, etc., by other than the user program. If register contents are referred to, a value that has been set in the E1/E20 emulator program will be read out.

Table 7.17 SFRs Used by the E1/E20 Emulator Program (1)

Address	Register	Symbol	Bit
000Ah	Protect register	PRCR	Bit 0
0023h	High-speed on-chip oscillator control register 0	FRA0	Bit 0 [*1]
0080h	DTC activation control register	DTCTL	Bit 0 [*2]

The SFRs listed in Table 7. 18 are used by the E1/E20 emulator program, not the user program.
Do not change the values, otherwise the E1 or E20 emulator cannot control the MCU.

Table 7. 18 SFRs Used by the E1/E20 Emulator Program (2)

Address	Register	Symbol	Bit
0004h	Processor mode register 0	PM0	Bit 0 [*3]
01C0h – 01C2h	Address match interrupt register 0	RMAD0	All bits
01C3h	Address match interrupt enable register 0	AIER0	All bits
01C4h – 01C6h	Address match interrupt register 1	RMAD1	All bits
01C7h	Address match interrupt enable register 1	AIER1	All bits

Notes:

[*1] When debugging with the E1 or E20 emulator, the high-speed on-chip oscillator does not stop although the options for high-speed on-chip oscillator enable bit are available and FRA00 can be set to “high-speed on-chip oscillator off”. To check the functions of low power consumption etc. with high-speed on-chip oscillator off, make the evaluation with the final products or system manufactured by you in which only the user program is written to the MCU and the emulator is disconnected.
The functions can also be checked by writing only the user program to the MCU, ending the debugger, then executing the user program. To write only the user program to the MCU, select [writing the on-chip flash memory mode] in the [Initial Settings] dialog box displayed when starting the debugger.
If the high-speed on-chip oscillator frequency or the frequency control register needs to be changed, be sure to do that change in the user program. If such changes are made in the [Memory] window, etc., the E1 or E20 emulator will become uncontrollable.

[*2] Not applicable for the R8C/LA8A, R8C/LA6A, R8C/LA5A, R8C/LA3A and R8C/LAPS.

[*3] Only applicable for the R8C/L3xC and R8C/L3xM. The bit value is always “1” when the register is used by the E1 or E20 emulator.

7.1.6 Option function select area

The E1 or E20 emulator sets each of bit 0 and bit 7 of the option function select register (OFS: 0FFFFh) to 1b. Although these addresses can be rewritten and the changed values can be referred to in the [Memory] window, etc., the changed value for bit 0 is invalid.

- b0: Watchdog timer start select bit — 1: Watchdog timer is stopped after reset.
- b7: Count source protection mode after reset select bit — 1: Count source protect mode disabled after reset

Also, the emulator sets the lower 4 bits of the option function select register 2 (OFS2: 0FFDBh) to 1111b.

- b1, b0: Watchdog timer underflow period set bit — 11: 3FFFh
- b3, b2: Watchdog timer refresh acknowledgement period set bit — 11: 100%

Notes:

[*1] During the user program halt, the E1/E20 emulator program refreshes the watchdog timer. Note that if the user program uses the watchdog timer, the watchdog timer will be refreshed by the E1/E20 emulator program during the user program halt, making the refresh timing differ from the actual operational timing.
Also, note that the watchdog timer is not refreshed during the execution of the user program.

[*2] Count source protection mode cannot be debugged with the E1 or E20 emulator.

7.1.7 Registers initialized by the E1 or E20 emulator

When the system is launched, the emulator initializes the CPU registers as shown in Table 7. 19

Table 7. 19 E1/E20 Emulator Register Initial Values

Status	Register	Initial Value
E1/E20 Emulator Activation	PC	Reset vector value in the vector address table
	R0 to R3 (bank 0, 1)	0000h
	A0, A1 (bank 0, 1)	0000h
	FB (bank 0, 1)	0000h
	INTB	00000h
	USP	0000h
	ISP	05FFh (differs from the specification of the MCU)
	SB	0000h
	FLG	0000h

7.1.8 RAM initialization

Note that the E1 or E20 emulator initializes part of the MCU's internal RAM area (00400h-004FFh) with "00h" at startup.

7.1.9 MCU reserved area

The addresses not defined in the MCU hardware manual are a reserved area. Do not change the content of the reserved area. If changed, the E1 or E20 emulator will not control the MCU.

- The value of this area is undefined when referenced in the [Memory] window.
- In this area, the [Memory] window's search, compare and move functions do not work normally.

7.1.10 DTC during a user program halt (not applicable for the R8C/LAxA)

When the user program halts, data transfer using DTC is prohibited.

However, the DTC interrupt request itself does not stop. Therefore, the DTC interrupt request occurred during the user program halt will be executed when the user program is restarted.

7.1.11 Note on internal power low consumption

Make sure that bit 0 of voltage detect register 2 (VCA2) for the E1 or E20 emulator is set to "0: Low consumption disabled". If "1" is selected, the emulator will not control the MCU.

7.1.12 Note on debugging at less than 2.7V (not applicable for the R8C/LAxA)

The minimum voltage for writing into or erasing the flash memory is 2.7V according to the specification of the MCU to be used. As flash rewrite occurs when the operations below are executed, if the supply voltage of the MCU is less than 2.7V, the emulator cannot be used:

- Downloading the user program
- Setting and canceling S/W breaks (Setting/canceling address match and on-chip breaks are available)
- Rewriting the value of the flash memory in the [Memory] window
- Connecting the emulator to the MCU at emulator debugger startup

7.1.13 Debugging the functions to reduce power consumption

When debugging the low-current-consumption read mode, do not operate windows until the program stops at the breakpoint by setting the breakpoint at the line of the program which will be executed after the mode is cancelled.

When debugging the function to stop the flash memory, do not operate windows until the program stops at the breakpoint by setting the breakpoint at the line of the program which will be executed after the function is cancelled.

7.2 Reset

(1) Reset function

The power-on reset and the voltage monitor 0 reset cannot be used. If either reset is executed, the E1 or E20 emulator becomes uncontrollable.

Do not stop the user program while the reset pin remains in the “L” state. A timeout error will occur.

(2) Reset vector address

During a debug with the E1 or E20 emulator, the reset vector addresses are used by a program for the emulator. While the user program is halted, the values set in a downloaded program or the [Memory] window can be referenced. But, when the user program is being run, the values set by a program for the emulator are displayed. If a reset vector address value is set while the user program is halted, only the emulator’s internal cache is updated.

(3) Behavior after a reset

If the MCU is reset while the user program is being run, the E1/E20 emulator program is entered into temporarily before the user program restarts (see Table 7. 20).

Therefore, there is a time lag until the user program starts after a reset. Also, note that the behavior is different between a hardware reset (RESET#) and other resets, as shown in the table below. [*1]

Table 7. 20 Behavior when the MCU Is Reset during the Execution of the User Program and It Is Restarted

Reset	Behavior when the MCU is reset
Hardware reset (RESET#)	(1) A hardware reset (RESET#) is detected by the E1 or E20 emulator.
	(2) The emulator is reconnected to the MCU.
	(3) The MCU is reset and the E1/E20 emulator program is entered into.
	(4) The address jumps to the reset vector address which is retained in the emulator program.
Watchdog timer reset or S/W reset [*2]	(1) The MCU is reset and the E1/E20 emulator program is entered into.
	(2) The address jumps to the reset vector address which is retained in the emulator program.

Notes:

[*1] Address match interrupts can be set or deleted during the execution of the user program, but address match breaks that are set during the execution of the user program are initialized by a reset.

[*2] If a watchdog timer reset or an S/W reset is made during the execution of the user program, address match breaks (Channel 0 to Channel 3) are disabled (no break occurs) because the SFRs are initialized.
The channel numbers of address match breaks can be checked on the [Before PC Break] tab of the [On-Chip Event] dialog box.

(4) Watchdog timer reset

The watchdog timer continues counting even while the user program is halted, with its counts refreshed in a program for the emulator. Therefore, if an underflow or other abnormal condition occurs immediately after the user program breaks, causing the watchdog timer to be reset immediately before it is refreshed, the emulator may become uncontrollable.

(5) Reset-related limitations

Do not perform a hardware reset (RESET#) while the user program is halted. The E1 or E20 emulator will not control the MCU. Note that if a reset is asserted from the MCU pin while the user program is halted, the contents of the CPU registers are not initialized and their values as of the time the user program has stopped or those set in the CPU register window are prioritized.

(6) Contentions

If contention occurs between a reset (hardware reset (RESET#), watchdog timer reset, etc.) and operations by the E1 or E20 emulator (memory access in the [Memory] window, etc.), the emulator may run out of control. Error messages such as a timeout error are displayed.

Also, a message box asking whether to issue a system reset or not is displayed. Clicking the [Yes] button initializes the emulator and stops the user program. Clicking the [No] button neither initializes the emulator nor stops the user program. After clicking either button, you can continue the debugging.

Note:

[*1] If the automatic memory update is enabled in the [Memory] or [Watch] window, do not perform a hardware reset to the MCU. Otherwise communications with the MCU will be lost and the E1 or E20 emulator will run out of control.

7.3 Internal ROM area (flash memory)

7.3.1 Changing the internal ROM area

When changing the contents of the MCU's internal ROM without downloading from the emulator debugger, only the E1 or E20 emulator's internal cache is updated and the changed contents are reflected in the MCU immediately before the user program starts running.

7.3.2 Notes on debugging in CPU rewrite mode [*1]

(1) Unrewritable area in CPU rewrite mode

When debugging in CPU rewrite mode, do not rewrite CPU for the internal ROM area containing the following areas. If these areas are rewritten, the E1 or E20 emulator will not control the MCU.

- Fixed interrupt vector area
- Area containing the emulator program (debug monitor) for the E1 and E20
- Block 3 in program ROM (except for the R8C/LAxA and R8C/LAPS. See the table below.) [*2]

MCUs	Applicable areas	Remarks
4 KB ROM version	-	
8 KB ROM version	-	
16 KB ROM version	-	
24 KB ROM version	Block 3: 0A000h-0BFFFh	
32 KB ROM version	Block 3: 08000h-0BFFFh	Only applicable for the 32 KB ROM versions of the R8C/36C, R8C/38C, R8C/36M, R8C/38M, R8C/34W, R8C/34X, R8C/34Y, R8C/34Z, R8C/3MQ, R8C/34U and R8C/3MU.
48 KB ROM version	Block 3: 08000h-0BFFFh	
64 KB ROM version	Block 3: 08000h-0BFFFh	
96 KB ROM version	-	
128 KB ROM version	-	

(2) Operation in CPU rewrite mode

- When debugging in the CPU rewrite mode, do not halt the user program while the CPU rewrite mode is enabled or while in the erase suspend state. And do not perform a step execution of the instruction which enables a CPU rewrite mode or enters an erase-suspend state.
  If halted, the E1 or E20 emulator may not control the MCU.
  In addition, disable the automatic update in the [Watch] window or fix the display in the [Memory] window before running the program so memory accesses do not occur during an execution.
- To check the internal ROM area after rewriting the internal ROM in the CPU rewrite mode, halt the user program after disabling the CPU rewrite mode and refer to the [Memory] window, etc.
  If CPU rewrite can be executed for the data flash area, and erase process can be suspended, do not use S/W breaks.
- When rewriting the flash memory in the program area, select the [Debugging the program re-writing the internal flash.] checkbox in the [Configuration Properties] dialog box which appears at debugger start up. [*3]

Notes:

[*1] When debugging a program in CPU rewrite mode, memory reference or modification functions can be used. However, do not use these functions in the following condition.
· While write instruction is being executed to the register which requires continuous writing (ex. FMR13 bit)
The MCU does not recognize the writing is continuously executed if the write instruction is interrupted by the memory reference or modification process.

[*2] This is not applicable when you assign programs for the E1/E20 emulator to the data flash area.

[*3] When [Debugging the program re-writing the internal flash.] is selected, do not rewrite the internal ROM area in the [Memory] window, etc.
Also note that a S/W break cannot be used when [Debugging the program re-writing the internal flash.] is selected.

7.3.3 Note on rewriting flash memory by the E1 or E20 emulator

(1) Do not reset nor execute debugging operations to the MCU while the internal ROM (flash memory) is being written by the E1 or E20 emulator.

Flash memory rewrite is executed from when the [Flash memory writing ...] is displayed until when [Flash memory writing ... OK] is displayed in the [Output] window of the High-performance Embedded Workshop.

If the MCU is reset or debugged when rewriting the flash memory, the user program or the E1/E20 emulator program may be disrupted.

Flash memory rewrite occurs:

- When downloading the user program
- After setting S/W breaks in the internal ROM area and performing an operation for executing the user program
- After canceling S/W breaks in the internal ROM area and performing an operation for executing the user program
- After rewriting the value of the internal ROM area in the [Memory] window or from the command line and performing an operation for executing the user program

7.3.4 Flash memory during the user program execution

While the user program is being run, the internal ROM area cannot be changed in other than the user program (in the memory window, etc.).

7.3.5 MCUs used for debugging

When debugging, the Flash memory is frequently rewritten by the E1 or E20 emulator. Therefore, do not use an MCU that has been used for debugging in products. Also, as the E1/E20 emulator program is written to the MCU while debugging, do not save the contents of the MCU Flash memory which were used for debugging nor use them as the ROM data for products.

7.3.6 Flash memory ID code

This MCU function prevents the Flash memory from being read out by anyone other than the user.

The ID code written to the ID code area of the MCU (see Table 7. 21) must match the one entered for the [ID Code] box in the [ID Code verification] dialog (see Figure 7.1) displayed at emulator debugger startup, otherwise the debugger cannot be launched. Note that when the ID code is FFh, FFh, FFh, FFh, FFh, FFh, FFh, the ID code is regarded as undefined. In this case, the ID code is automatically authenticated and the [ID Code verification] dialog box is not displayed.

The values written into the ID code area differs depending on the mode.

- [Writing the on-chip flash memory mode] [*1]: Contents of the user program
- [Debugging mode] [*2]: FFh, FFh, FFh, FFh, FFh, FFh, FFh (regardless of the contents of the downloaded user program)

Table 7. 21 ID Code Storage Area

Address	Description
FFDFh	First byte of ID code
FFE3h	Second byte of ID code
FFEBh	Third byte of ID code
FFEFh	Fourth byte of ID code
FFF3h	Fifth byte of ID code
FFF7h	Sixth byte of ID code
FFFBh	Seventh byte of ID code

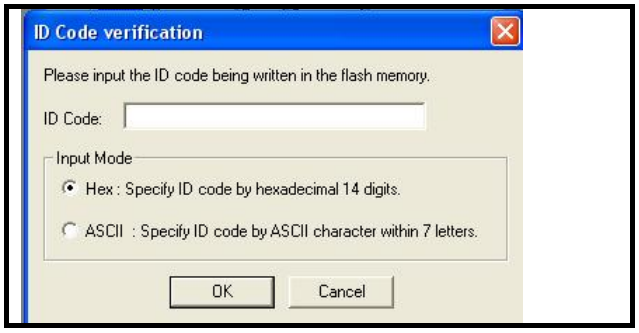


Figure 7.1 [ID Code verification] Dialog Box

Notes:

[*1] Notes on [Writing the on-chip flash memory mode]:

- When the ID code is specified by the -ID option of the lmc30, download the MOT file or HEX file. When the X30 file is downloaded, the ID code is not valid.
- When downloading the X30 file, specify the ID code using an assembler directive command such as ".BYTE".
- The file to which the ID code specified by the assembler directive command ".ID" is output varies depending on the version of the assembler. For details, refer to the Assembler User's Manual.

[*2] If the ID code written to the ID code area of the MCU matches the one entered for the [ID Code] box in the [ID Code verification] dialog displayed at emulator debugger startup, the E1 or E20 emulator writes FFh, FFh, FFh, FFh, FFh, FFh, FFh to the ID code area. Therefore, the [ID Code verification] dialog will not be displayed the next time the debugger starts up.

7.4 Power supply

(1) Consumption current

When the E1 emulator does not supply power to the user system, it consumes the power voltage of the user system from several mA to more than 10 mA. This is because the user system power supply drives 74LVC125 and 74LVC8T245 to make the communication signal level match the user system power supply voltage. [*1]

(2) Note on E1 emulator power supply [*1]

When writing a program with the E1 emulator for mass production processes, the program requires reliability, so do not use the E1 emulator power supply function. Supply power separately to the user system according to the allowable voltage for MCU writing.

Voltage supplied from the E1 emulator depends on the quality of the USB power supply of the host machine, and as such, precision is not guaranteed.

Note:

[*1] The E20 emulator does not support the power supply function.

7.5 Operation during a user program halt

(1) Operation clock during a user program halt

When the user program halts, the emulator changes the CPU clock to the internal high-speed on-chip oscillator clock to operate. However, the peripheral features operate with the clock specified by the user program.

The clock frequency of the internal high-speed on-chip oscillator is determined by the operating supply voltage that was detected at emulator debugger startup. [*2]

(2) Peripheral I/Os during a user program halt

During a user program halt, the maskable interrupt request cannot be accepted, because the emulator disables interrupts. However, since peripheral I/Os continue to run, the interrupt request is accepted immediately after the user program execution is started.

For example, a timer interrupt is not accepted although the timer continues to count when a user program is stopped by a break after the timer started.

Note:

[*2] The frequency is switched with a threshold of 2.7V. Therefore, if you are using the E1 or E20 emulator around a supply voltage of 2.7V, the clock frequency may change depending on the voltage level detected by the emulator. The voltage value detected by the emulator is displayed in the [Connecting…] dialog box which appears at emulator debugger startup or the [Status] window.

7.6 Memory access during user program execution

The E1/E20 emulator and the MCU use the frequency of a high-speed on-chip oscillator as a communication clock. Therefore, accessing memory in the [Memory] window, etc. immediately after the change of high-speed on-chip oscillator frequency may not be performed normally.

7.7 Final evaluation of the program

Before entering the mass-production phase, be sure to perform a final evaluation of the program singly, without the E1 or E20 emulator connected.

7.8 Debug functions

7.8.1 Step execution

(1) Limitations during the step execution

During stepped execution, S/W breaks and on-chip breaks are invalid.

(2) Step out execution

Stepping out of a function that uses a jump instruction (i.e. not a subroutine-return instruction) to return to the origin of the call is not possible.

(3) Note on using automatic memory update

When the automatic memory update is enabled in the [Memory] or [Watch] window, do not execute step out or multiple-step. Otherwise, it will take longer to update memory data and the operation will be delayed.

(4) Exceptional step execution

a) Software interrupt instruction

Step execution cannot be performed in the internal processing of instructions (undefined, overflow, BRK and INT) which generate a software interrupt continuously in the program (see Figure 7.2).

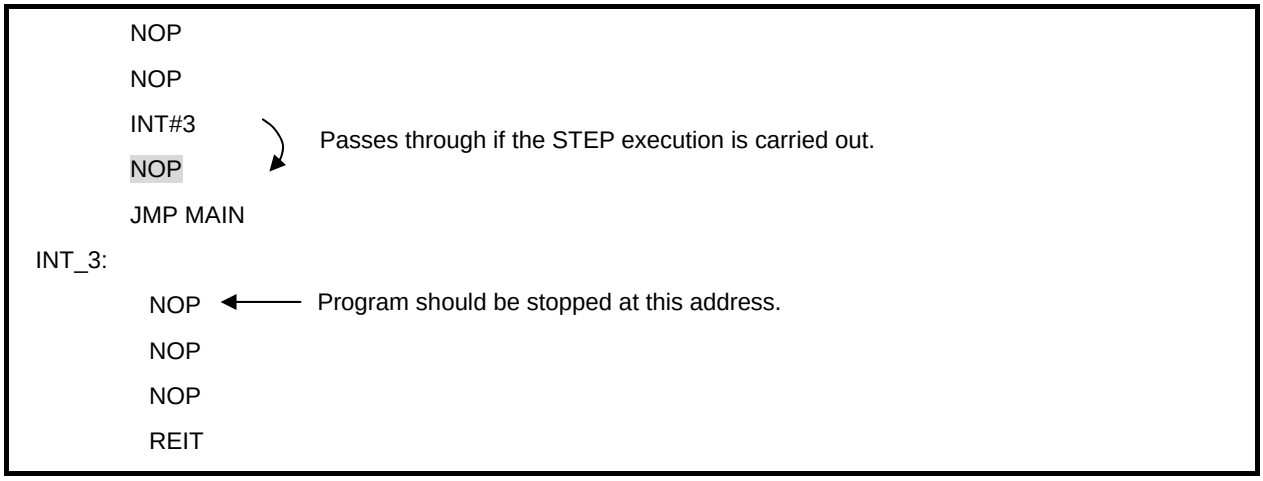


Figure 7.2 Example of Software Interrupt Instruction

b) INT instruction

To debug the user program with the INT instruction, set a S/W break for the internal processing of the INT instruction and execute the program with the GO command (see Figure 7.3).

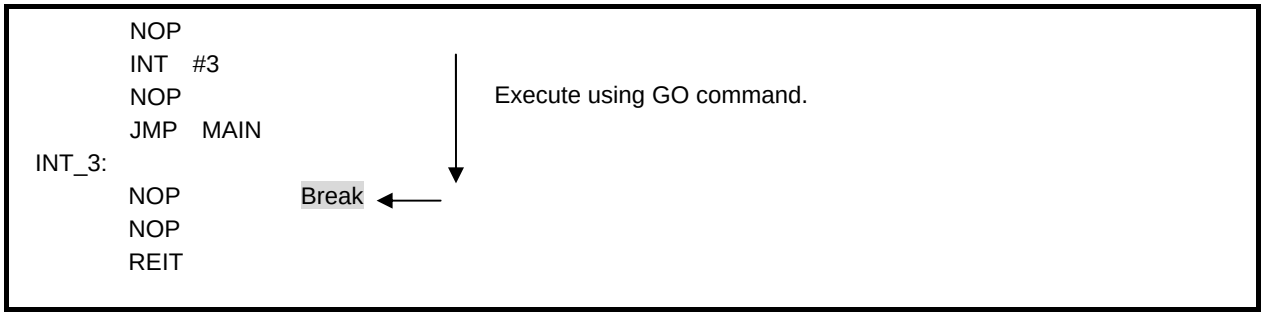


Figure 7.3 Example of INT Instruction

c) Other: Flag manipulating instructions

The following instructions, when single-stepped, only manipulate a flag in the E1 or E20 emulator, with no MCU operations involved. Therefore, when these instructions are executed, be aware that the Start/Stop function does not work.

LDC src, FLG
STC FLG, dest
LDINTB src

7.8.2 Other debug functions

(1) “Go to cursor” function

The “Go to cursor” function is actualized using an address match break.
When you execute the [Go To Cursor] command, all the S/W breaks and on-chip breaks you have set become invalid.

(2) Debugging in stop mode or wait mode

When debugging in stop mode or wait mode, do not operate windows until the program stops at the breakpoint by setting the breakpoint at the line of the program which will be executed after the stop mode or wait mode is cancelled.
And do not perform a step execution of the instruction shifting to stop mode or wait mode.

Also, if memory contents are referenced or changed during the stop mode or wait mode, the program—after exiting the stop or wait mode and accessing the MCU memory—restarts from the instruction next to the one by which it was placed into the stop or wait mode. If the program enters the stop or wait mode in the middle of a memory access, a reference or a change of memory contents may not be performed normally.
Therefore, disable the automatic update in the [Memory] window or [Watch] window or fix the display in the [Memory] window before running the program, and do not make refresh operations during an execution so memory accesses do not occur during user program execution.

When the program is forcibly stopped in stop mode or wait mode, these modes will be cancelled.

(3) If the reset circuit of the user system has a watchdog timer, disable it when using the emulator.

7.9 Notes on using the CAN module (applicable for the R8C/3xW and R8C/3xX only)

7.9.1 Notes on CAN module in operation

(1) When using the CAN module [*1] and if BCLK (CPU clock) is used as the CAN clock source (fCAN), the CPU clock should be used at 4MHz or more. If the CPU clock is used at less than 4MHz in this case, a communication error may occur.

(2) Do not activate the CAN module from the memory window, etc., except from the user program.

(3) When using the CAN module [*1], do not shift into stop mode. Otherwise, a communication error may occur.

(4) Note on S/W breakpoint

If the MCU's operation clock speed is low, setting or removing S/W breaks may take time.

In this case, use on-chip breaks prior to the S/W break.

7.9.2 Operation during a user program halt

(1) Operation clock during a user program halt

When the user program halts, the operation clock differs depending on the status of the CAN module (status of the CAN sleep status flag).

a) In CAN sleep mode (SLPST = 1)

The emulator changes the CPU clock to the internal high-speed on-chip oscillator clock.

However, the peripheral features operate with the clock specified by the user program.

b) Not in CAN sleep mode (SLPST = 0)

The clock specified by the user program is used as the operation clock. [*1]

Note:

[*1] The CAN module is recognized as being used in the following status (other than in CAN sleep mode)
Bit 2 (SLPST: CAN sleep status flag) of the CAN0 status register C0STR (2F42h - 2F43h)

0: Not in CAN sleep mode

Appendix A Menus

Table A.1 View Menu

Menu	Menu option		Support	Remark
View	Differences		○	Opens the [Difference] window.
	Map…		×	Opens the [Map Section Information] window.
	Command Line		○	Opens the [Command Line] window.
	TCL Toolkit		○	Opens the [Console] window.
	Workspace		○	Opens the [Workspace] window.
	Output		○	Opens the [Output] window.
	Status Bar		○	Shows or hides the status bar.
	Disassembly		○	Opens the [Disassembly] window.
	CPU	Registers	○	Opens the [Register] window.
		Memory…	○	Opens the [Memory] window.
		IO	○	Opens the [IO] window.
		Status	○	Opens the [Status] window.
	Symbol	Labels	○	Opens the [Labels] window.
		Watch	○	Opens the [Watch] window.
		Locals	○	Opens the [Locals] window.
	Event	On-chip Event	○	Opens the [On-Chip Event] dialog box.
	Graphic	Image…	○	Opens the [Image Properties] dialog box.
		Waveform…	○	Opens the [Waveform Properties] dialog box.
	Code	Trace	○	Opens the [Trace] window.
		Stack Trace	○	Opens the [Stack Trace] window.
	RTOS	OS Object	○	Opens the [OS Object] window

○: Supported, ×: Not supported

Table A.2 Debug Menu (1/2)

Menu	Menu option		Support	Remark
Debug	Synchronized Debugging…		×	Opens the [Synchronized Debug] dialog box, in which settings for synchronized debugging can be made.
	Debug Sessions…		○	Opens the [Debug Sessions] dialog box for listing, adding, or removing debug sessions.
	Debug Settings…		○	Opens the [Debug Settings] dialog box to set the debugging conditions or load modules.
	Reset CPU		○	Resets the target MCU and sets the PC to the reset vector address.
	Go		○	Starts running the user program from the location currently indicated by the PC.
	Reset Go		○	Resets the target MCU and executes the user program from the reset vector address.
	Free Go		○	Starts running the user program with all breakpoints ignored.
	Go To Cursor		○	Starts running the user program from the address currently indicated by the PC until the PC reaches the address indicated by the current text cursor position.
	Set PC To Cursor		○	Sets the PC to the address at the row of the text cursor.
	Run…		○	Launches the [Run Program] dialog box allowing the user to enter the PC or PC breakpoint during executing the user program.
	Display PC		○	Displays the current PC value in the [Editor] window.
	Step In		○	Executes a block of user program before breaking.
	Step Over		○	Executes a block of user program before breaking. If a subroutine call is reached, then the subroutine will not be entered.
	Step Out		○	Executes the user program to reach the end of the current function.
	Step…		○	Launches the [Step Program] dialog box allowing the user to modify the settings for stepping.
	Step Mode	Auto	○	Steps only one source line when the [Editor] window is active. When the [Disassembly] window is active, stepping is executed in a unit of assembly instructions.
		Assembly	○	Executes stepping in a unit of assembly instructions.
		Source	○	Steps only one source line.
	RTOS Debug	Go To Cursor	○	Enabled only when loading an RTOS program
		Step In	○	
		Step Over	○	
		Step Out	○	

○: Supported, ×: Not supported

Table A.3 Debug Menu (2/2)

Menu	Menu option	Support	Remark
	Halt Program	○	Stops the execution of the user program.
	Initialize	○	Disconnects and then restarts the debugging platform.
	Connect	○	Connects the debugging platform.
	Disconnect	○	Disconnects the debugging platform.
	Save Memory…	○	Saves the specified range of data from memory data in a file (.bin, .hex, or .mot).
	Verify Memory…	○	Verifies file contents against data in memory.
	Configure Overlay	×	
	Download Modules	○	Downloads the object program.
	Unload Modules	○	Unloads the object program.

○: Supported, ×: Not supported

Table A.4 Setup Menu

Menu	Menu option		Support	Remark
Setup	Customize…		○	
	Options…		○	
	Format Views…		○	
	Radix	Hex	○	Uses a hexadecimal for displaying a radix in which the numerical values will be displayed and entered by default.
		Decimal	○	Uses a decimal for displaying a radix in which the numerical values will be displayed and entered by default.
		Oct	○	Uses an octal for displaying a radix in which the numerical values will be displayed and entered by default.
		Bin	○	Uses a binary for displaying a radix in which the numerical values will be displayed and entered by default.
	Emulator	Device setting…	○	Opens the [Initial Settings] dialog box, in which settings for the target MCU can be made.
		System…	○	Opens the [Configuration Properties] dialog box allowing the user to modify the debugging platform settings.
		Start/stop Function Setting…	○	
	RTOS	Select OS Definition File	○	Enabled when loading an RTOS program

○: Supported, ×: Not supported

Table A.5 [Memory] Window Option Menu

Pop-up Menu Option			Support	Remark
Set…			○	
Fill…			○	
Move			○	
Compare…			○	
Test…			×	
Save Memory contents…			○	
Search…			○	
Search Next			○	
Address…			○	
Scroll Area…			○	
Register			×	
Followed Stack Pointer…			×	
Set Start Up Symbol			○	
Refresh			○	
Lock Refresh			○	
Auto Refresh			○	
Refresh Interval…			○	
Data Length	1byte		○	
	2bytes		○	
	4bytes		○	
	8bytes		○	
Radix	Hex		○	
	Dec		○	
	Signed Dec		○	
	Oct		○	
	Bin		○	
Code	ASCII		○	
	SJIS		○	
	JIS		○	
	UNICODE	UTF-8	○	
		UTF-16	○	
	EUC		○	
	Float	Float	○	
		Double	○	
	Complex	Float Complex	○	
		Double Complex	○	
		Float Imaginary	○	
		Double Imaginary	○	
	Fixed	16bit Fixed	○	
		32bit Fixed	○	
		24 bit Accum	○	
		40 bit Accum	○	
Layout	Label		○	
	Register		○	
	Code		○	
Column…			○	
Coverage	Enable		×	
Save…			○	
Load…			○	
Split			○	

○: Supported, ×: Not supported

Table A.6 [Trace] Window Option Menu

Pop-up Menu Option	Support	Remark
Clear	○	Clears the trace result.
Save…	○	Saves the trace result in a file.
View Source	○	When you click this button after selecting any line, the source file of the selected line is displayed in the [Editor] window.
Trim Source	○	Displays a clipped version of the source being displayed in the Source column, with blanks to the left of it removed.
Trace Branch…	○	Interpolates the source from the branched-to address to the next branched-from address. Valid when trace data is for “Branch (branch source/branch destination).”
Statistic…	○	Analyzes data counts that matched a specified condition.
Function Call…	○	Displays a history of function execution. Valid when trace data is for “Branch (branch source/branch destination).”

○: Supported, ×: Not supported

Appendix B Notes on the High-performance Embedded Workshop

(1) Note on moving source file position after creating load module

When the source file is moved after creating the load module, the [Open] dialog box may be displayed to specify the source file during the debugging of the created load module. Select the corresponding source file and click on the [Open] button.

(2) Source-level execution (source file)

Do not display source files that do not correspond to the load module in the program window. For a file having the same name as the source file that corresponds to the load module, only its addresses are displayed in the program window. The file cannot be operated in the program window.

(3) Source-level execution (step)

Even standard C libraries are executed. To return to a higher-level function, use [Step Out]. You can also select not to step into addresses where no debugging information exists. Open the [Options] dialog box via [Setup -> Options] and select the [Only step in when debug information is available] checkbox on the [Debug] page.

In a for statement or a while statement, executing a single step does not move execution to the next line. To move to the next line, execute two steps.

(4) Operation during accessing files

Do not perform other operations during downloading the load module, operating [Verify Memory] or [Save Memory] in the [Memory] window, or saving in the [Trace] window because this will not allow correct file accessing to be performed.

(5) Watch (local variables at optimization)

Depending on the generated object code, local variables in a source file that is compiled with the optimization option enabled will not be displayed correctly. Check the generated object code by displaying the [Disassembly] window.

If the allocation area of the specified local variable does not exist, displays as follows.

Example: The variable name is asc.

asc Not available now.

(6) Watch (variable name specification)

When a name other than a variable name, such as a symbol name or function name, is specified, no data is displayed.

Example: The function name is main.

main Not available now.

(7) Command line interface (Batch file)

To display the message “Not currently available” while executing a batch file, enter the sleep command. Adjust the sleep time length which differs depending on the operating environment.

Example: To display “Not currently available” during memory_fill execution:

sleep d’3000

memory_fill 0 ffff 0

(8) Command line interface (file specification by commands)

The current directory may be altered by file specifications in commands. It is recommended to use absolute paths to specify the files in a command file so that the current directory alteration is not affected.

Example: FILE_LOAD

C:\WorkSpace\Tutorial\E1E20\R8C\Tutorial\Tutorial\Debug\Tutorial.x30

(9) Loading of Motorola S-type files

This High-performance Embedded Workshop does not support Motorola S-type files with only the CR code (0Dh) at the end of each record. Load Motorola S-type files with the CR and LF codes (0D0Ah) at the end of each record.

(10) Note on [Register] window operation during program execution

The register value cannot be changed in the [Register] window during program execution. Even if the changed value is displayed, the register contents are not changed actually.

(11) Note on RUN-TIME display

Although the execution time for the user program is displayed on the status bar and in the [Status] window, values are rounded down to the nearest 100 μs (i.e. values less than 100 μs are discarded) since an internal 32-bit counter of the emulator is used for this measurement. Values are also not accurate during single stepping, [Step Over], or [Step Out].

(12) Memory test function

This product does not support the memory test function, which is activated by selecting [Test…] from the [Memory] menu.

(13) "Writing the on-chip flash memory" mode

When MCUs are to be continuously programmed, be sure to turn the target system on or off.

Debugging functions other than downloading of a program are not usable in the "writing the on-chip flash memory" mode.

(14) Disassembled display

The [Editor] window in the disassembly mode displays undefined codes as '???'.

(15) Watch function

If a symbol is registered when the user program is under execution, the [Value] column of the [Watch] window shows a warning "Not available now." In this case, i.e., while there is a symbol just registered, do not enable the automatic updating of all symbols when the user program is being run.

REVISION HISTORY

Rev.	Date	Description	
		Page	Summary
1.00	May16, 2011	—	First Edition issued
2.00	Aug 30, 2011	7	Following target MCUs added: R8C/3xM: R8C/32M, R8C/33M, R8C/34M, R8C/35M, R8C/36M, R8C/38M, R8C/3GM, R8C/3JM Groups R8C/3xGHPR: R8C/32G, R8C/32H, R8C/33G, R8C/33H, R8C/34P, R8C/34RGroups R8C/3xKU: R8C/34K, R8C/34U, R8C/3MK, R8C/3MU Groups R8C/LxM: R8C/L35M, R8C/L36M, R8C/L38M, R8C/L3AM Groups R8C/3NT Group R8C/3MQ Group R8C/LA3A, R8C/LA5A Group R8C/LAPS Group
		9	Caution for use in mass production added
		12, 17	Connection method for use in FDT added
		14	Pins to be connected to Vss added (pins 4, 6 and 10) in Notes
		15	Pins to be connected to Vss added (pins 7, 11 and 12) in Notes
		33	Information on Power supply checkbox ("Power supply checkbox cannot be selected" when the emulator serial No. is not displayed) added to Note 2
		36	No.1 added to Table 4.2
		37	Item 4.12.1 "When the E1/E20 emulator is used in other emulator debuggers such as RX Family" added
		54	Target MCUs added Information on IO files added to Note 2

Rev.	Date	Description	
		Page	Summary
2.00	Aug 30, 2011	109	Table 7.2 added (Target MCU:R8C/3xM)
		111	Table 7.7 added (Target MCU:R8C/3xGHPR)
		112	Table 7.9 added (Target MCU:R8C/3NT)
		112	Table 7. 11 added (Target MCU:R8C/3MQ)
		112	Table 7.11 added (Target MCU:R8C/3xUK)
		113	Table 7.13 added (Target MCU:R8C/L3xM)
		114	Table 7.14 added (Target MCU:R8C/LA3A, R8C/LA5A)
		114	Table 7.15 updated (96KB,128KB added : R8C/LA6A, R8C/LA8A)
		114	Table 7.16 added (Target MCU:R8C/LAPS)
		115	Target MCUs of Notes [*2] and [*3] added
		120	Target MCU added in 7.3.2(1) Note [*2] added
		123	“Memory access during user program execution” added
		-	Caution regarding trace-complementing function deleted (Restrictions on High-performance Embedded Workshop V.4.08)
2.01	Oct 21, 2011	117	Initial Value correction of INTB

E1/E20 Emulator
Additional Document for User's Manual (Notes on Connection)

Publication Date: Oct 21, 2011 Rev.2.01

Published by: Renesas Electronics Corporation

Edited by: Microcomputer Tool Development Department 2
Renesas Solutions Corp.

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

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

Renesas Electronics America Inc.
2880 Scott Boulevard Santa Clara, CA 95050-2554, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

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

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

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-65030, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
7th Floor, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100083, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 204, 205, AZIA Center, No.1233 Lujiazui Ring Rd., Pudong District, Shanghai 200120, China
Tel: +86-21-5877-1818, Fax: +86-21-6887-7858 / -7898

Renesas Electronics Hong Kong Limited
Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2886-9318, Fax: +852 2886-9022/9044

Renesas Electronics Taiwan Co., Ltd.
7F, No. 363 Fu Shing North Road Taipei, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
1 harbourFront Avenue, #06-10, keppel Bay Tower, Singapore 098632
Tel: +65-6213-0200, Fax: +65-6278-8001

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

Renesas Electronics Korea Co., Ltd.
11F., Samik Lavied' or Bldg., 720-2 Yeoksam-Dong, Kangnam-Ku, Seoul 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

© 2011 Renesas Electronics Corporation and Renesas Solutions Corp. All rights reserved.

Colophon 1.0

E1/E20 Emulator
Additional Document for User's Manual
(Notes on Connection)

Renesas Electronics Corporation

R20UT0081EJ0201