To our customers,

Old Company Name in Catalogs and Other Documents

On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

Renesas Electronics website: http://www.renesas.com

April 1st, 2010
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (http://www.renesas.com)

Send any inquiries to http://www.renesas.com/inquiry.

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.

To all our customers

Regarding the change of names mentioned in the document, such as Hitachi Electric and Hitachi XX, to Renesas Technology Corp.

The semiconductor operations of Mitsubishi Electric and Hitachi were transferred to Renesas Technology Corporation on April 1st 2003. These operations include microcomputer, logic, analog and discrete devices, and memory chips other than DRAMs (flash memory, SRAMs etc.) Accordingly, although Hitachi, Hitachi, Ltd., Hitachi Semiconductors, and other Hitachi brand names are mentioned in the document, these names have in fact all been changed to Renesas Technology Corp. Thank you for your understanding. Except for our corporate trademark, logo and corporate statement, no changes whatsoever have been made to the contents of the document, and these changes do not constitute any alteration to the contents of the document itself.

Renesas Technology Home Page: http://www.renesas.com

Renesas Technology Corp.
Customer Support Dept.
April 1, 2003

Cautions

Keep safety first in your circuit designs!

1. Renesas Technology Corporation puts the maximum effort into making semiconductor products better and more reliable, but there is always the possibility that trouble may occur with them. Trouble with semiconductors may lead to personal injury, fire or property damage.
   Remember to give due consideration to safety when making your circuit designs, with appropriate measures such as (i) placement of substitutive, auxiliary circuits, (ii) use of nonflammable material or (iii) prevention against any malfunction or mishap.

Notes regarding these materials

1. These materials are intended as a reference to assist our customers in the selection of the Renesas Technology Corporation product best suited to the customer's application; they do not convey any license under any intellectual property rights, or any other rights, belonging to Renesas Technology Corporation or a third party.
2. Renesas Technology Corporation assumes no responsibility for any damage, or infringement of any third-party's rights, originating in the use of any product data, diagrams, charts, programs, algorithms, or circuit application examples contained in these materials.
3. All information contained in these materials, including product data, diagrams, charts, programs and algorithms represents information on products at the time of publication of these materials, and are subject to change by Renesas Technology Corporation without notice due to product improvements or other reasons. It is therefore recommended that customers contact Renesas Technology Corporation or an authorized Renesas Technology Corporation product distributor for the latest product information before purchasing a product listed herein.
   The information described here may contain technical inaccuracies or typographical errors.
   Renesas Technology Corporation assumes no responsibility for any damage, liability, or other loss rising from these inaccuracies or errors.
   Please also pay attention to information published by Renesas Technology Corporation by various means, including the Renesas Technology Corporation Semiconductor home page (http://www.renesas.com).
4. When using any or all of the information contained in these materials, including product data, diagrams, charts, programs, and algorithms, please be sure to evaluate all information as a total system before making a final decision on the applicability of the information and products. Renesas Technology Corporation assumes no responsibility for any damage, liability or other loss resulting from the information contained herein.
5. Renesas Technology Corporation semiconductors are not designed or manufactured for use in a device or system that is used under circumstances in which human life is potentially at stake. Please contact Renesas Technology Corporation or an authorized Renesas Technology Corporation product distributor when considering the use of a product contained herein for any specific purposes, such as apparatus or systems for transportation, vehicular, medical, aerospace, nuclear, or undersea repeater use.
6. The prior written approval of Renesas Technology Corporation is necessary to reprint or reproduce in whole or in part these materials.
7. If these products or technologies are subject to the Japanese export control restrictions, they must be exported under a license from the Japanese government and cannot be imported into a country other than the approved destination.
   Any diversion or reexport contrary to the export control laws and regulations of Japan and/or the country of destination is prohibited.
8. Please contact Renesas Technology Corporation for further details on these materials or the products contained therein.

User's Manual

RENESAS

SuperH™ RISC engine High-performance Embedded Workshop, Debugging Interface

Tutorial

Renesas Microcomputer Development Environment System

Renesas Electronics
www.renesas.com

Rev.1.0 2000.12

Cautions

1. Hitachi neither warrants nor grants licenses of any rights of Hitachi's or any third party's patent, copyright, trademark, or other intellectual property rights for information contained in this document. Hitachi bears no responsibility for problems that may arise with third party's rights, including intellectual property rights, in connection with use of the information contained in this document.
2. Products and product specifications may be subject to change without notice. Confirm that you have received the latest product standards or specifications before final design, purchase or use.
3. Hitachi makes every attempt to ensure that its products are of high quality and reliability. However, contact Hitachi's sales office before using the product in an application that demands especially high quality and reliability or where its failure or malfunction may directly threaten human life or cause risk of bodily injury, such as aerospace, aeronautics, nuclear power, combustion control, transportation, traffic, safety equipment or medical equipment for life support.
4. Design your application so that the product is used within the ranges guaranteed by Hitachi particularly for maximum rating, operating supply voltage range, heat radiation characteristics, installation conditions and other characteristics. Hitachi bears no responsibility for failure or damage when used beyond the guaranteed ranges. Even within the guaranteed ranges, consider normally foreseeable failure rates or failure modes in semiconductor devices and employ systemic measures such as fail-safes, so that the equipment incorporating Hitachi product does not cause bodily injury, fire or other consequential damage due to operation of the Hitachi product.
5. This product is not designed to be radiation resistant.
6. No one is permitted to reproduce or duplicate, in any form, the whole or part of this document without written approval from Hitachi.
7. Contact Hitachi's sales office for any questions regarding this document or Hitachi semiconductor products.

Trademarks:

Read First:

1. Hitachi, Ltd. (including its subsidiaries, hereafter collectively referred to as Hitachi) pursues a policy of continuing improvement in design, performance, and safety of the system. Hitachi reserves the right to change, wholly or partially, the specifications, design, user's manual, and other documentation at any time without notice.
2.
3. Hitachi assumes no responsibility for any intellectual property claims or other problems that may result from applications based on the examples described herein.

Preface

This tutorial will introduce you to the basic usage of the Hitachi Embedded Workshop (hereafter referred to as *HEW*) and the Hitachi Debugging Interface (hereafter referred to as *HDI*). The HEW tutorial describes a series of usage such as how to invoke HEW, how to create and modify a project, how to build a program, and how to invoke HDI from HEW. The HDI tutorial describes how to debug a sample program using the Simulator/Debugger. For the details of each component of HEW and HDI, refer to the following manuals.

- Hitachi Embedded Workshop User's Manual
- SuperH™ RISC engine C/C++ Compiler, Assembler, Optimizing linkage editor User's Manual
- SuperH™ RISC engine Simulator/Debugger User's Manual

For details on the SuperH™ RISC engine CPU, refer to a programming manual or a hardware manual of the corresponding CPU.

This tutorial is written assuming that all the tools included in the SuperH™ RISC engine C/C++ compiler Package are installed.

Document Conventions

This manual uses the following typographic conventions:

Table 1 Typographic Conventions

CONVENTION	MEANING
[Menu->Menu Option]	Text with ‘->’ is used to indicate menu options (for example, [File->Save As...]).

Contents

Section 1 HEW Tutorial

1.1 Invoking the HEW

After the installation of the HEW, the installer creates a folder whose name is “Hitachi Embedded Workshop” in the “Program” folder of the “Start” menu of the Windows®. In the “Hitachi Embedded Workshop” folder, short cuts of the HEW support files will be registered (figure 1.1). The contents of the “Start” menu and its submenus differ depending on your installation.

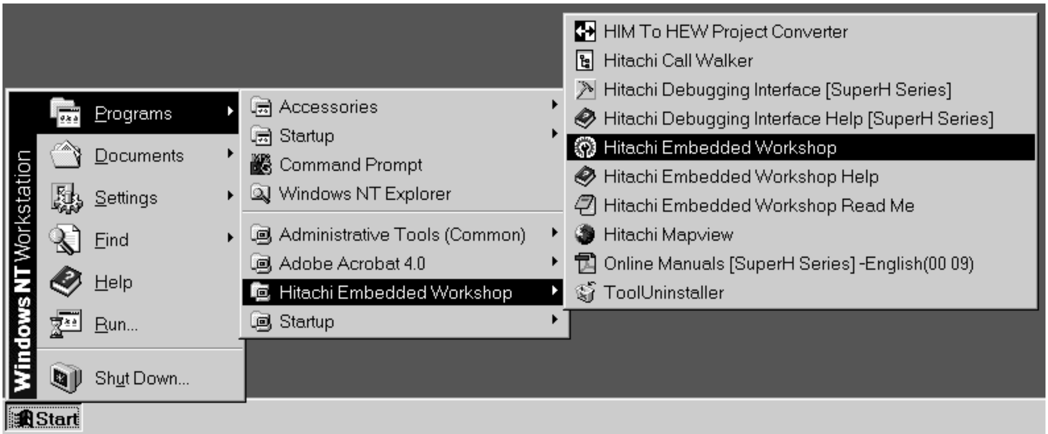


Figure 1.1 Invoking the HEW from the “Start” Menu

If you click “Hitachi Embedded Workshop” from the menu, the HEW will be invoked and the “Welcome!” dialog box (figure 1.2) will be displayed. You can open the project promptly without the “Welcome!” dialog if you change the setting via [Tools->Options]. For details of this setting, refer to chapter 6, “Customizing the Environment”, of the Hitachi Embedded Workshop User’s Manual.

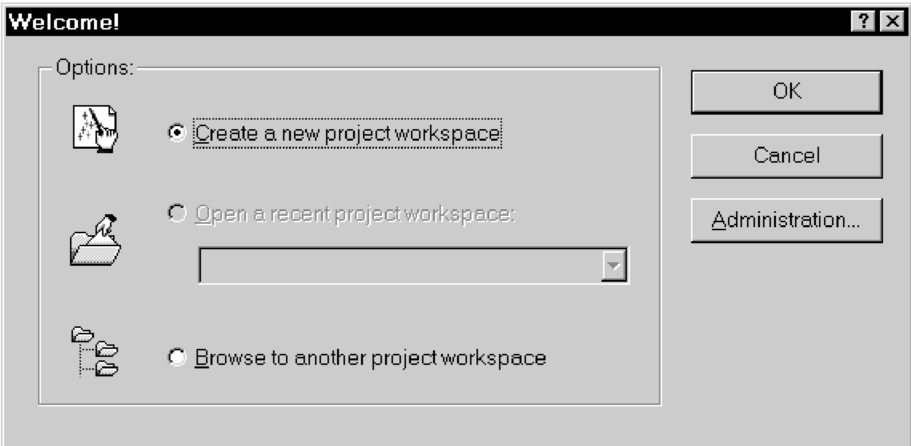


Figure 1.2 Welcome! Dialog Box

If you use the HEW for the first time or if you want to work on a new project, select the [Create a new project workspace] radio button and click [OK]. If you want to work on an existing project, select the [Open a recent project workspace] or [Browse to another project workspace] radio button and click [OK].

In this tutorial, select the [Create a new project workspace] radio button and click [OK].

1.2 Creating a Project

When you have selected the [Create a new project workspace] radio button, then clicking [OK] in the “Welcome!” dialog box will launch the “New Project Workspace” dialog box (figure 1.3), which is used to create a new workspace and project. You can also launch the “New Project Workspace” dialog box by selecting [File -> New Workspace…] after you have launched HEW. In this dialog box, you will specify a workspace name (when newly creating a workspace name, the project name will be the same as the workspace), CPU family, project type, and so on. For details of the project type, refer to table 1.1.
In this tutorial, enter “tutorial” as a workspace name in the [Name] field, and select “Application” in the [Project type] list. If you have entered “tutorial” in the [Name] field, then the [Directory] field will show “c:\tutorial” and a project will be generated under this directory. If you want to change the directory used for the new workspace, click the [Browse…] button and specify a directory, or enter a directory path manually in the [Directory] field.

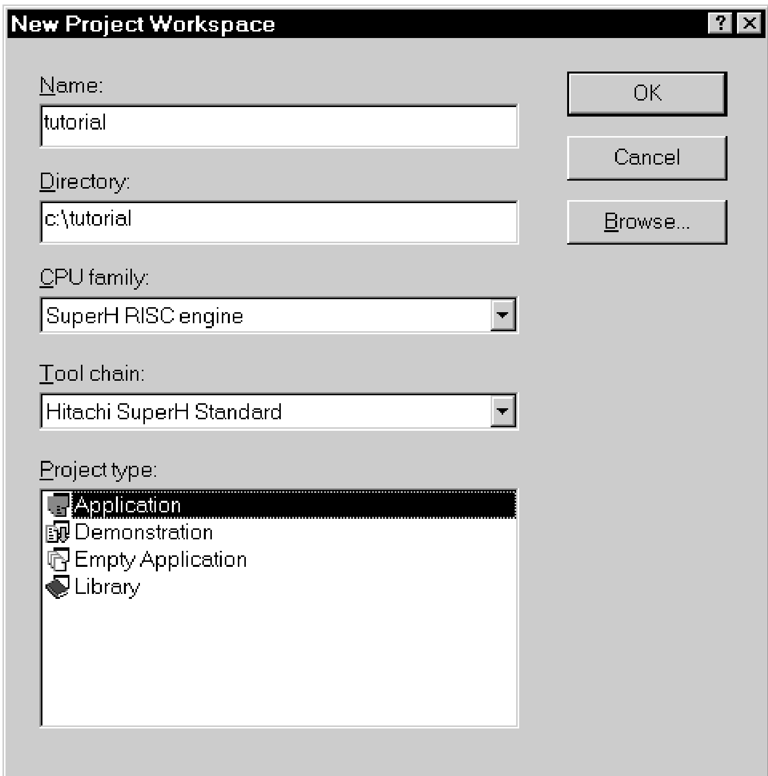


Figure 1.3 New Project Workspace Dialog Box

Table 1.1 Project Type

Project Type	Description
Application	Projects to develop programs including initial routine files written in C/C++ or assembly language.
Demonstration	Projects to develop demonstration programs written in C/C++ or assembly language.
Empty Application	Projects to develop programs (But are not generated files. They are used to set only the options of the tool.)
Library	Projects to develop library files (But are not generated files. They are used to set only the options of the tool.)

1.2.1 Selecting a CPU

When you click [OK] in the “New Project Workspace” dialog box, the project generator will be invoked and will launch the Step 1 dialog box (CPU selection) (figure 1.4). In this dialog box, select the type of target CPU for which you wish to develop programs. In this tutorial, select “SH-1” in the [CPU Series] list and “SH7020” in the [CPU Type] list. Click the [Next >] button after selecting a CPU type from the [CPU Type] list. This will launch the Step 2 dialog box (Setting the options unique to the CPU).

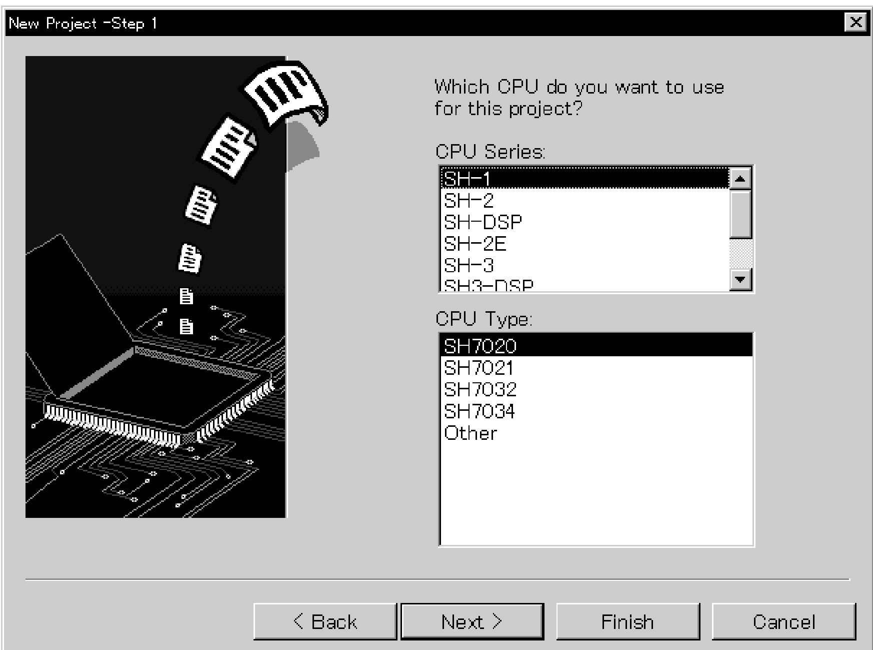


Figure 1.4 Selecting the CPU in the Project Generator (Step 1)

CPU types shown in the [CPU Type:] list are classified into the CPU series shown in the [CPU Series:] list. When you select a CPU type from the [CPU Series:], the project generator will set a CPU type option for the standard library generator, C/C++ compiler, and assembler. Depending on what you select in [CPU Series:] and [CPU Type:], different types of files will be generated by the Project Generator. Select the type of target CPU for which you wish to develop programs. If you cannot find the CPU you want in the list, then select one close to its hardware specifications or select “Other.”

- Click [Next >] to get the project generator to launch the next dialog box.
- If you click [< Back], the project generator will return to the previous dialog box.
- If you click [Finish], the project generator will open the Summary dialog box.
- If you click [Cancel], the project generator will return to the “New Project Workspace” dialog box.
- [< Back], [Next >], [Finish], and [Cancel] all function in the same say in the Project Generator.

1.2.2 Setting Options Unique to the CPU

When you click [Next >] in the Step 1 dialog box, the Project Generator will launch the dialog box shown in figure 1.5.
In this dialog box, you can set options unique to the CPU and that can be used commonly through all the project files.
In the tutorial, do not set anything and click [Next >]. The Project Generator will launch the Step 3 dialog box (Setting the Generation File).

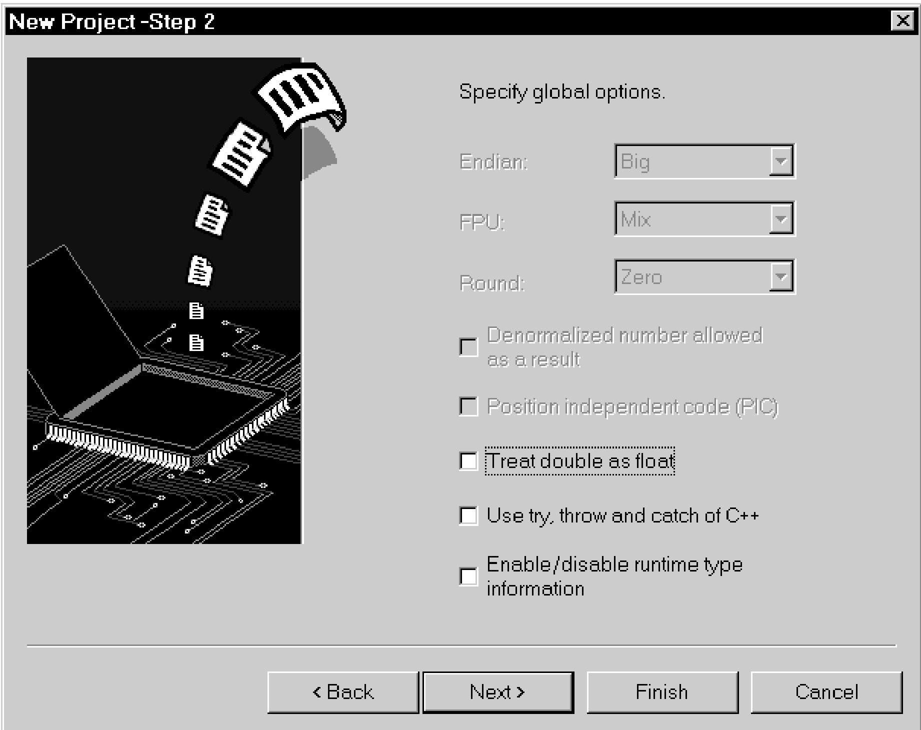


Figure 1.5 Setting Options Unique to the CPU in the Project Generator (Step 2)

You may set the following options by the Step 1 and 2 dialog boxes.
If you want to change the settings of the options after you have generated a project, you must modify the option setting dialog box in the tools below ([Options ->] in HEW).

- C/C++ Compiler (Same for Standard Library Generator)
  CPU
  Method of division
  Byte arrangement in memory (endian type)
  FPU
  Rounding
  How to deal with denormalized numbers
  Program section position independent
  Double to float conversion
  Exception processing function
  Runtime type information
- Assembler
  CPU
  Endian
  Rounding
  How to deal with denormalized numbers
- Optimizing Linkage Editor
  Output format (determined by the project type)
  Specification of library files (only for the DSP library)

You cannot modify options that are not supported by the Project Generator for the CPU series you selected in the Step 1 dialog box. In this tutorial, since you have selected SH-1, options except [Treat double as float], [Use try, throw and catch of C++], and [Enable/disable runtime type information] will be grayed out, meaning that you cannot set them.
If you selected Library for the project type, there are no other settings you have to make. Therefore, just click [Finish].
If you selected Demonstration for the project type, click [Next >] to launch Step 6, which is shown in section 1.2.7, Modifying Generated File Name. If you selected Demonstration, the Project Generator will launch the Step 3 dialog box.

1.2.3 Setting the Generating Files

When you click [Next >] in the Step 2 dialog box, the Project Generator will launch the dialog box shown in figure 1.6.

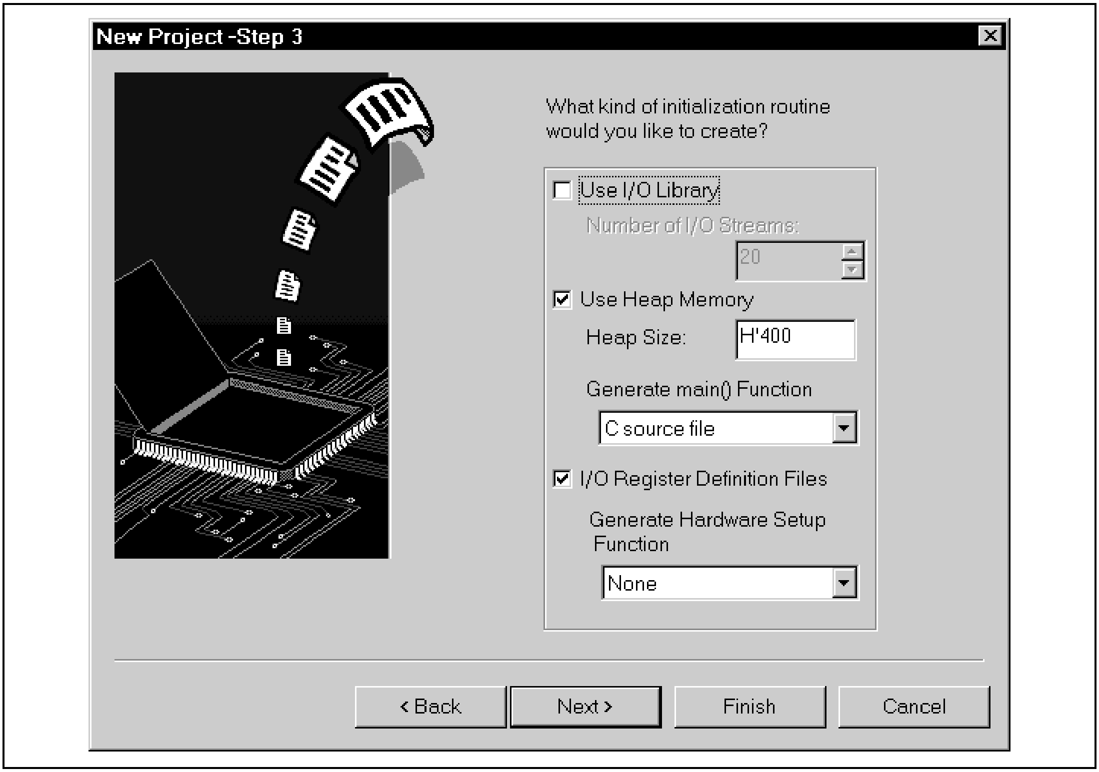


Figure 1.6 File Generation Using the Project Generator (Step 3)

In this dialog box, you must decide the kind of file to create.
In this tutorial, do not choose anything and click [Next >]. The project generator will launch the Step 4 dialog box (Setting the Standard Library).

If you select checkbox [Use I/O Library], you can use the standard I/O library. Here, the [Number of I/O Streams:] is always 20.

If you select checkbox [Use Heap Memory], you can use function sbrk() for controlling the heap area. Then you can use [Heap Size:] to set the size of the heap area to control as follows.

H'ABCDE: Hexadecimal
0xABCDE: Hexadecimal
1234: Decimal

If you want to modify [Heap Size] after you have generated a project, you must modify the number in the following line of “sbrk.h”.

```
#define  HEAPSIZE  0x400
```

You can select the way to generate the main function in [Generate main() Function] (C, C++, or no generation).
If you select checkbox [I/O Register Definition Files], the Project Generator generates an I/O register definition file written in C language. Then you can set [Generate Hardware Setup Function].

Different types of files will be created by the Project Generator depending on what you set in the this dialog box and what you select for the project type. If you select a checkbox in table 1.2, the Project Generator will create the corresponding files.

Table 1.2 The Relationship between the Dialog Box Settings and the Generated Files

Control	Stage	Generated File
[Use I/O Library]	Selected	lowlvl.src lowsrc.c
[Number of I/O Streams:]	Nothing can be entered	—
[Use Heap Memory]	Selected	sbrk.c
[Heap Size:]	Enter a value	sbrk.h
[Generate main() Function]	C source file C++ source file None	tutorial.c tutorial.cpp — When Project Generator creates a file, the project name becomes the file name.
[I/O Register Definition Files]	Selected	iodefine.h
[Generate Hardware Setup Function]	None Assembly source file C/C++ source file	— hwsetup.src hwsetup.c (or cpp) (same file extension as the main function file)

Note: To use the main (_main) function that has already been generated, you must not select [Generate main() Function], and add the corresponding file to the project to generate a project. To use another function that has already been generated, you must modify the function calling section in the resetprg.src.

1.2.4 Setting the Standard Library

When you click [Next >] in the Step 3 dialog box, the Project Generator will display the dialog box shown in figure 1.7.

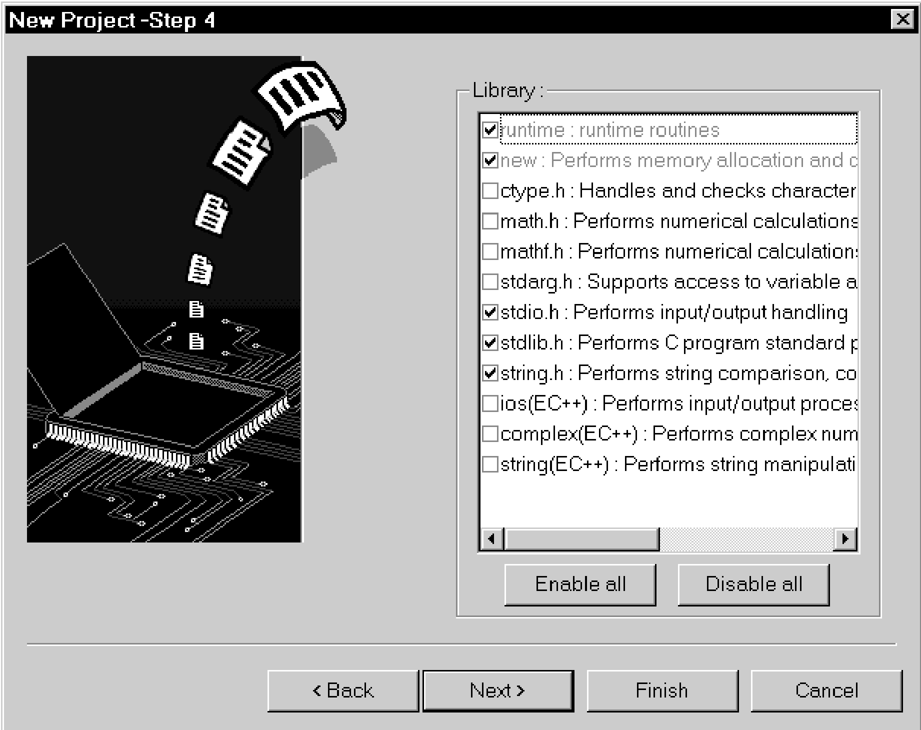


Figure 1.7 Setting Standard Library in Project Generator (Step 4)

In this dialog box, you can set the standard library to use in the Project Generator. When you build a project, the Project Generator will create the standard library according to the items you have selected in [Library].

If you want to modify the settings after you have created the project, modify the library through the option setting dialog box (select [Options -> SH Library Generator…] in HEW) of the standard library generator.

In this tutorial, do not set anything and click [Next >]. The Project Generator will display the Step 5 dialog box (setting stack area).

1.2.5 Setting Stack Area

When you click [Next >] in the Step 4 dialog box, the Project Generator will display the dialog box shown in figure 1.8.

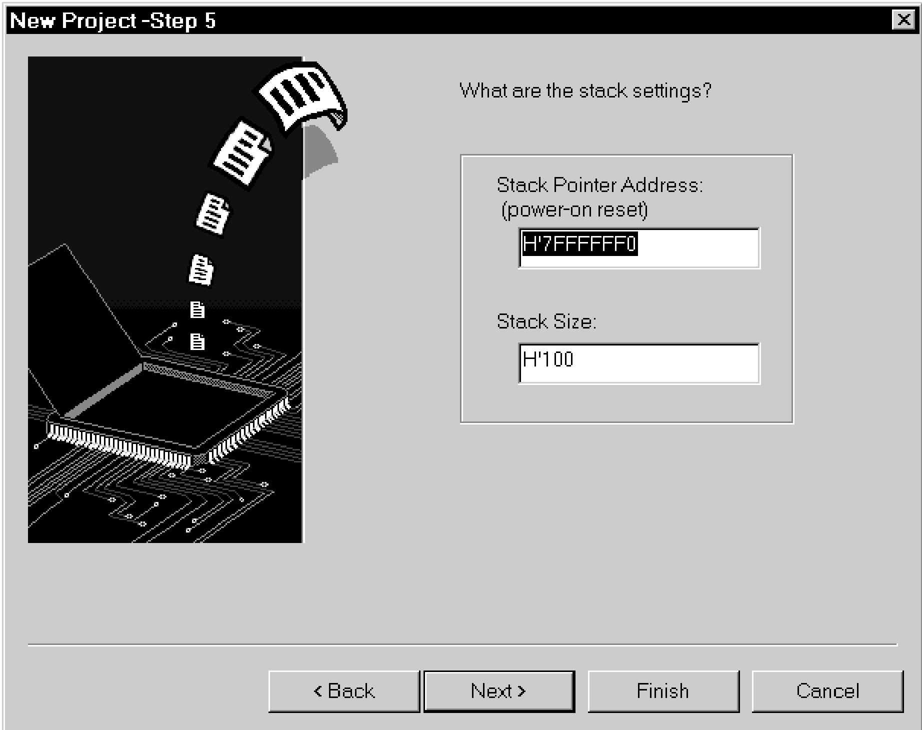


Figure 1.8 Setting Stack Area in Project Generator (Step 5)

In this dialog box, you can set the stack area.
In this tutorial, do not set anything and click [Next >]. The Project Generator will display the Step 6 dialog box (setting vector definition).

You can enter a value in hexadecimal or decimal in [Stack Pointer Address:] and [Stack Size:]. Enter a value as follows:

H'ABCDE: Hexadecimal
0xABCDE: Hexadecimal
1234: Decimal

If you enter a value in [Stack Size:], it will be set in stacksct.src. If you enter a value in [Stack Pointer Address:] and [Stack Size:], the Project Generator will determine the start address of the stack section by using the start option of the optimizing linkage editor. The start address of the stack section will be determined as follows:

Stack section name: Stack
Start section address = value of [Stack Pointer Address:] – value of [Stack Size:]

To modify the stack size after you have created the project, modify the stack area size of stacksct.src and modify the start address of the start section by using the start option in the option dialog box (select [Options -> Optlinker…] in HEW) of the optimizing linkage editor.

```
stacksct.src
    .section  Stack,STACK
    .export   _PowerON_Reset_SP
    .export   _Manual_Reset_SP
StackEND:           .res.b     H'100          -----> Stack area size
_PowerON_Reset_SP:  .equ $
_Manual_Reset_SP:   .equ $
    .end
```

1.2.6 Setting the Vector

When you click [Next >] in the Step 5 dialog box, the Project Generator will display the dialog box shown in figure 1.9.

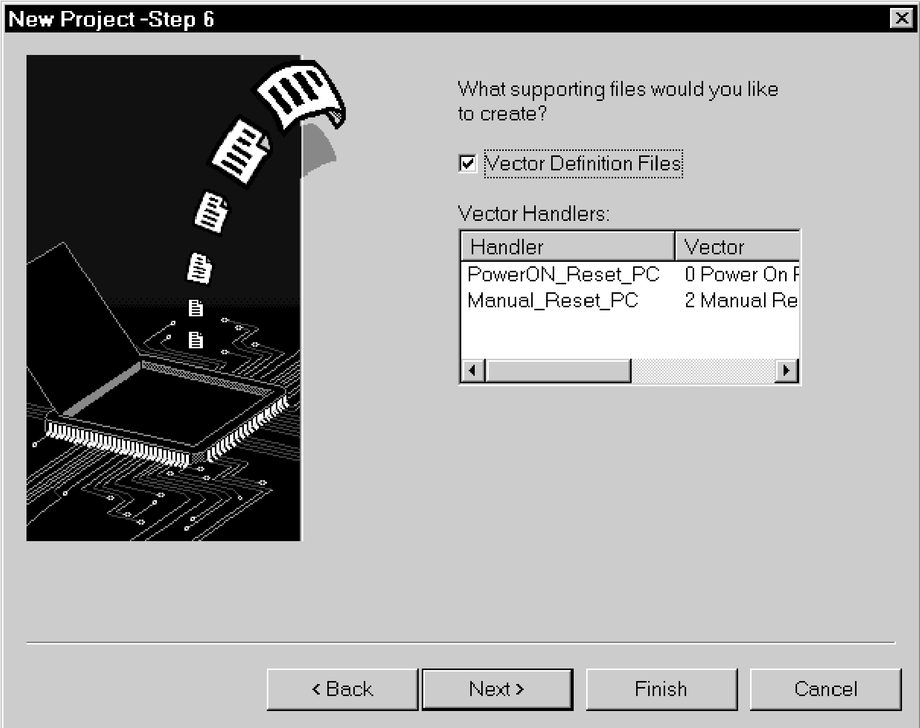


Figure 1.9 Setting Vector in Project Generator (Step 6)

In this dialog box, you can set the vector.
In this tutorial, do not set anything and click [Next >]. The Project Generator will display the Step 7 dialog box (confirming generated files).

When you select [Vector Table Definition Files], the Project Generator will create vector table definition files. At this time, you can modify the reset vector by [Vector Handlers:].
The [Handler] column in [Vector Handlers:] shows the handler program name, and the [Vector] column explains the vector. To modify the handler program name, enter the program name after you have selected the handler program by clicking it in the dialog box.

If you modify the handler name of [Vector Handlers:], the Project Generator will not create reset program file (resetprg.src).

If you select [Vector Definition Files], the Project Generator will create the following files:

vecttbl.c (vecttbl.src will be created in SH-3 and SH-4 series.)
intprg.c (intprg.src will be created in SH-3 and SH-4 series.)
vect.h (vect.inc will be created in SH-3 and SH-4 series.)
resetprg.src
vhander.src (Will be created only in SH-3 and SH-4 series.)

To edit the interrupt program or modify other programs after you have created the project, modify the source file according to the following procedure:

- Editing the interrupt program
  - The reset program is written in assembly language in file resetprg.src. You can add processes to the file as required. The file includes only easy initial settings such as section initialization and stack register settings (only for SH-3 and SH-4).
  - The interrupt program is written in C language in SH-1 and SH-2 series in file intprg.c, and is written in assembly language in SH-3 and SH-4 series in file intprg.src. Both files include only the program names, so you must write the necessary processes.
- Changing the interrupt program into another program
  - To change the SH-1 and SH-2 series interrupt programs into other programs, you must modify the function for interrupt vectors, which are the target functions to modify in the definition file (vect.h). Furthermore, you must delete the same function from the interrupt function file (intprg.c) and add a new function to be used as an interrupt function.

Example: To modify vector 4 program (INT_Illegal_code) to void UserIntProg4(void)

```
vect.h
    .
    .
// 4 Illegal code
#pragma interrupt INT_Illegal_code     --->  Changed to
                                     #pragma interrupt UserIntProg4
extern void INT_Illegal_code(void);    --->  Changed to
    .                                extern void UserIntProg4 (void);
    .

intprg.c
    .
    .
// 4 Illegal code
void INT_Illegal_code(void){/* sleep(); */}  --->  Delete and write
   .                                   void UserIntProg4(void).
   .                                   void UserIntProg4(void) can be
   .                                   written to another file.
```

  - To modify the SH-3 and SH-4 series interrupt programs, you must modify the label (assembly module name) that corresponds to the interrupt cause. The interrupt cause is the target to be modified in the definition file (vect.inc). Furthermore, you must delete the same function from the interrupt function file (intprg.c) and add a new function to be used as an interrupt function.

Example: To modify a program with interrupt cause code H'040 (_INT_TLBMiss_Load) to _UserIntProg040

```
vect.inc
   .
   .
   ;H'040 TLB miss/invalid (load)
   .global_INT_TLBMiss_Load     --->   changed to
   .                                  .global  _UserIntProg040
   .

intprg.src
   .
   .
;H'040 TLB miss/invalid (load)
INT_TLBMiss_Load --->   Delete and add a file that includes
   .                  _UserIntProg040 to the project
   .

vecttbl.src
   .
   .
   ;H'040 TLB miss/invalid (load)
   .data.l     _INT_TLBMiss_Load     --->   change to
   .                                  .data.l  _UserIntProg040
   .
```

1.2.7 Modifying Generated File Name

When you click [Next >] in the Step 6 dialog box, the Project Generator will display the dialog box shown in figure 1.10.

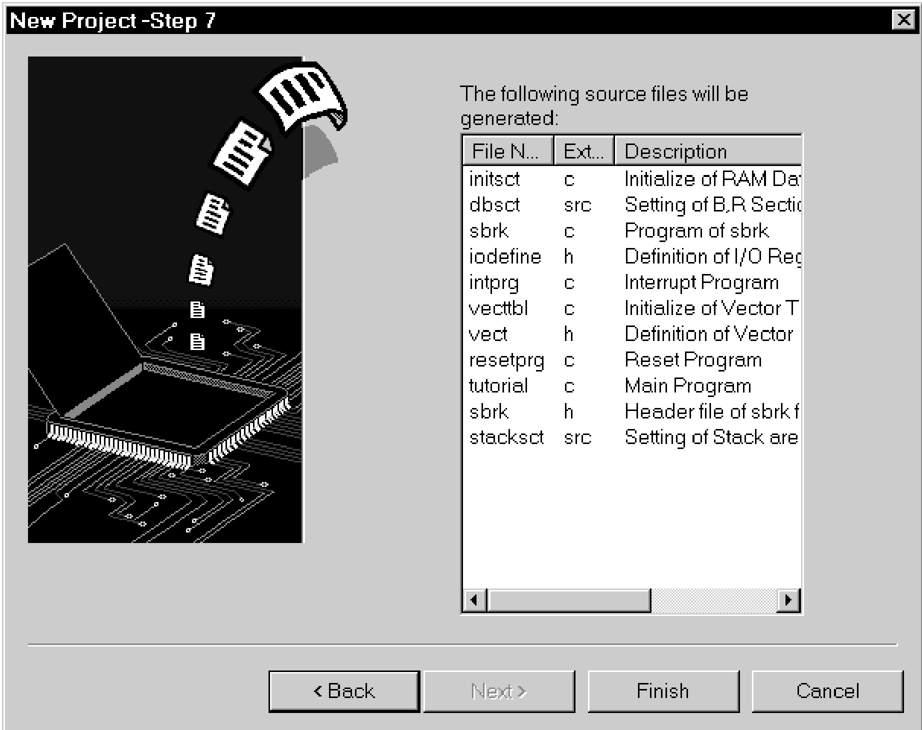


Figure 1.10 Confirming Generated File Name in Project Generator (Step 7)

This dialog box will be the last one in the New Project wizard. It will display all of the files that have been created by the Project Generator. The [File Name] column in the list shows the file names, the [Extension] column shows the extensions, and the [Description] column explains the files. The file names can be modified. To change the file names, enter the new name after selecting the file name.

In this tutorial, do not set anything and click [Next >]. The Project Generator will display the Summary dialog box.

Note: When the extension in [Extension] is .h or .inc, this shows that the file is an include file. To modify the names of these files, you must modify the include statements included in these files.

1.2.8 Confirming Settings

When you click [Finish] in the Step 7 dialog box, the Project Generator will display the dialog box shown in figure 1.11.

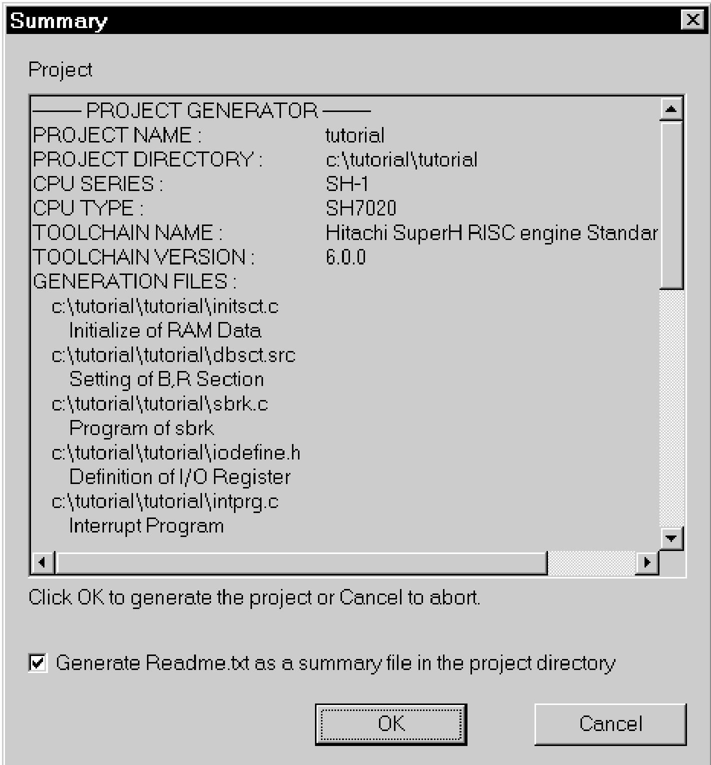


Figure 1.11 Summary Dialog Box in Project Generator

[Project Summary:] displays the following contents. Confirm the contents and click [OK]. To modify the settings, click [Cancel].

The Project Generator will output the displayed contents as text files (Readme.txt). When the Project Generator does not output anything, uncheck the check box [Generate Readme.txt as a summary file in the project directory]. When you click [OK], the Project Generator will create the project.

HEW opens the projects in which files generated by the project generator are installed. The start window of HEW is divided into windows such as the work space window, which shows the contents of the project (left side of the window), output window, which shows the results of build and string literal detection among files (bottom part of the window), and editor window, which is used to edit the text file (right side of the window). For details on modifying the state of HEW window or the functions of windows including the editor window, refer to Hitachi Embedded Workshop User's Manual.

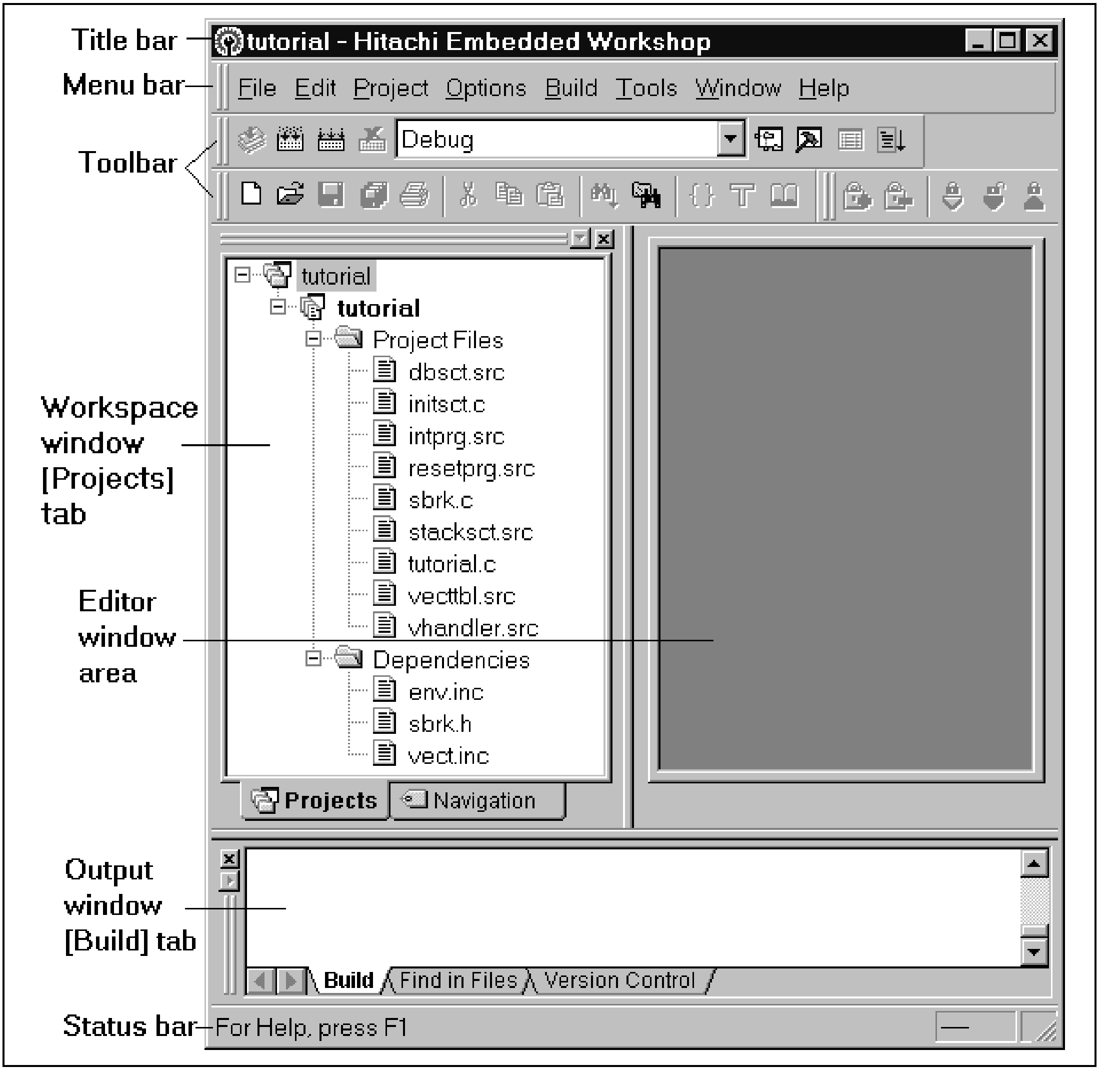
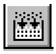


Figure 1.12 HEW Window Configuration

When the Project Generator creates a project, the Project Generator sets basic options for the tools to perform a build such as the standard library generator, C/C++ compiler, assembler, and optimizing linkage editor. Therefore, it is possible to perform build by selecting [Build->Build] immediately after creating a project. See figure 1.13.

You can also perform a build by using the following button on the toolbar.



These basic options are defined as the default for each tool, so you do not have to set them except for the options that must be set for each file after it has been added to the project.

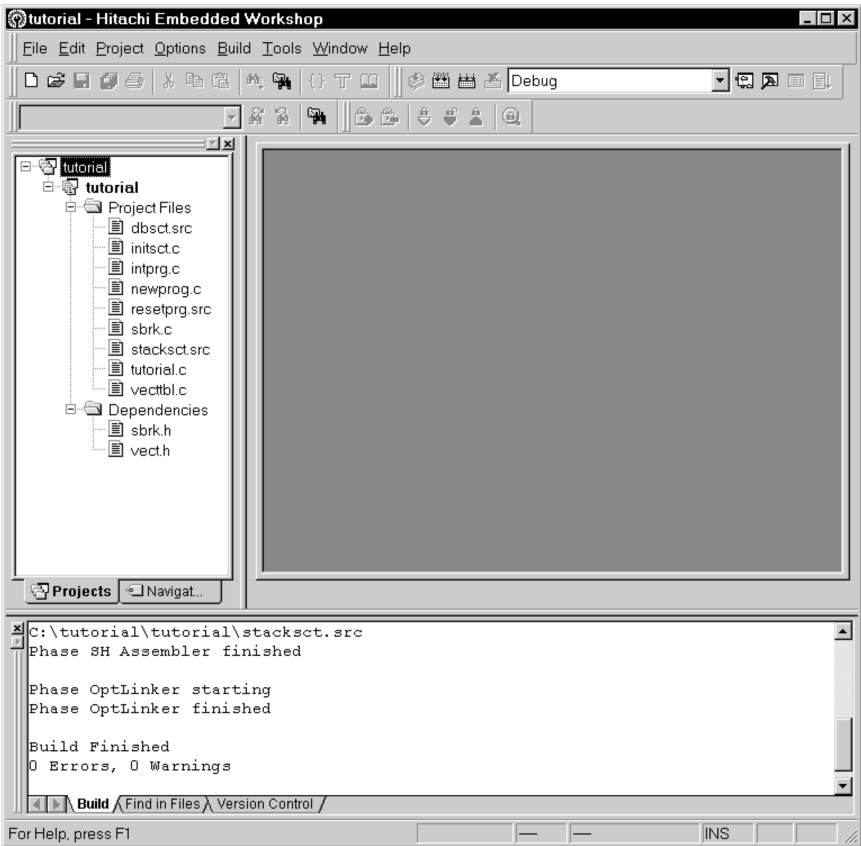


Figure 1.13 Window of Build

1.3 Modifying the Project

When you create a project using the project generator, basic program such as reset routine and initialization routine of RAM area will already be given in the project. This tutorial describes how to modify a file and how to add a file to the project.

1.3.1 Editing and Creating a Source Program File

To open and edit a file in the project, double click the file in the [Projects] tab of the “Workspace” window. In this tutorial, double click “tutorial.c” (figure 1.14).

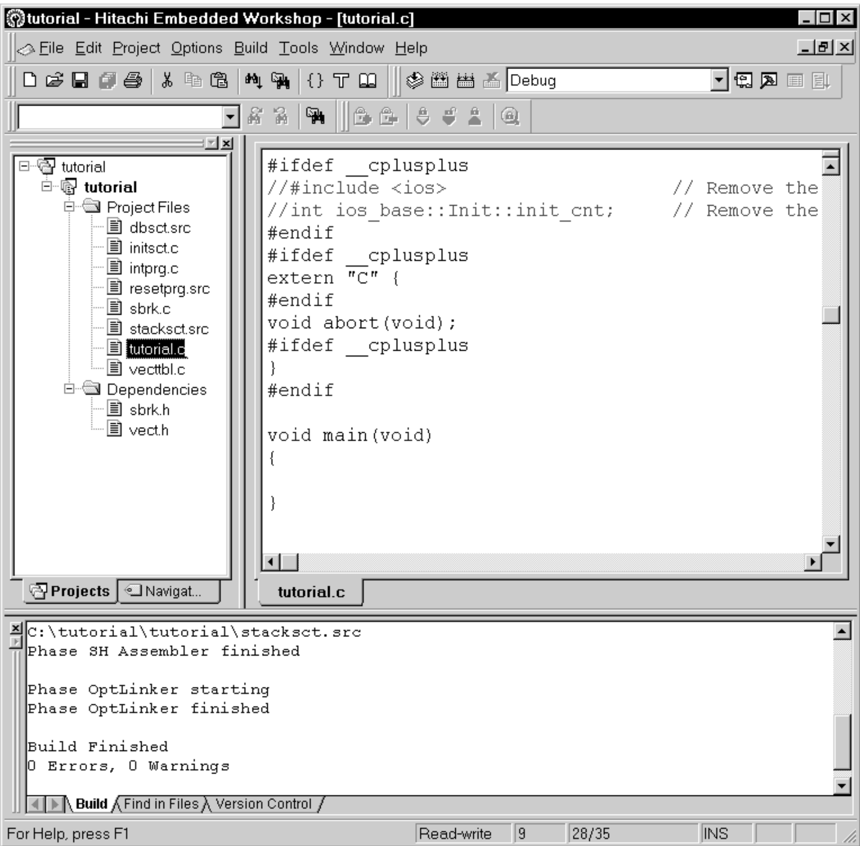


Figure 1.14 Opening a Project File

Then you can edit the file opened. If you edit the file, an asterisk (*) will be added to the file path shown on the title bar of the HEW (figure 1.15). If you save the file, the asterisk will disappear.

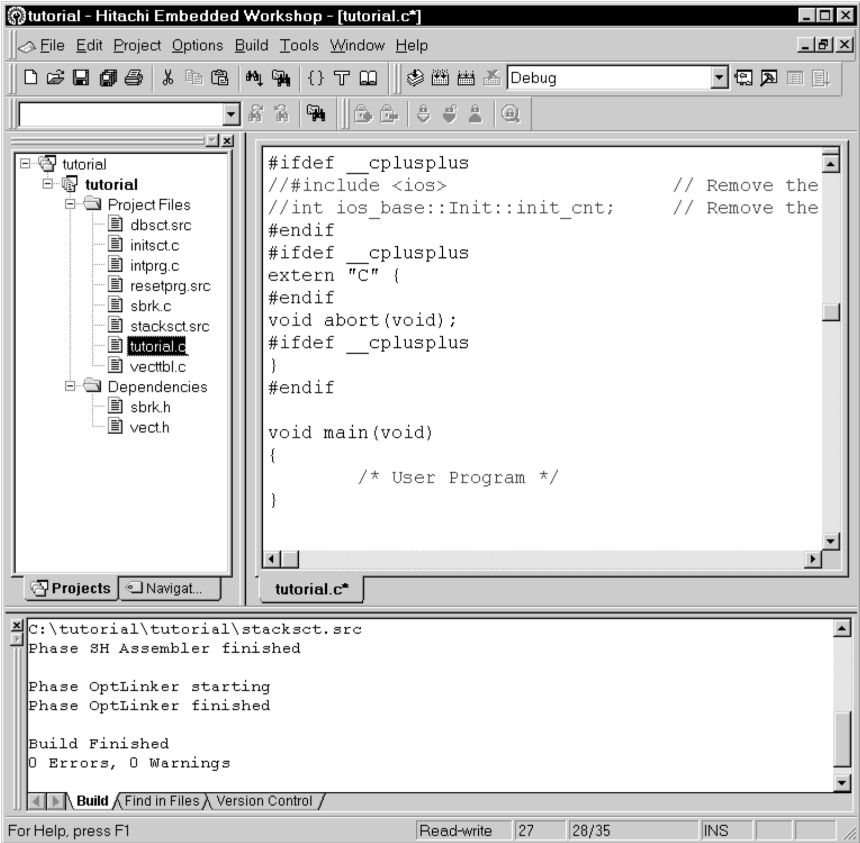


Figure 1.15 Editing a Project File

If you want to open an existing text file with the editor of the HEW, select [File->Open]. If you want to edit a new text file with the editor, select [File->New]. In this tutorial, select [File->New], input the three lines shown below, select [File->Save As…] and save the file as “newprog.c”.

```
void NewProgram(void)
{
}
```

1.3.2 Adding / Removing a File To / From a Project

This section describes how to add a file to a project and how to delete a file from a project. In this tutorial, add the file, "newprog.c" created in the previous section to the project as follows.
Select [Project->Add Files…], then the "Add File(s)" dialog box (figure 1.16) will be launched.
Select a file, "newprog.c" in this example, to be added and click the [Add] button. Thus "newprog.c" will be added to the project.

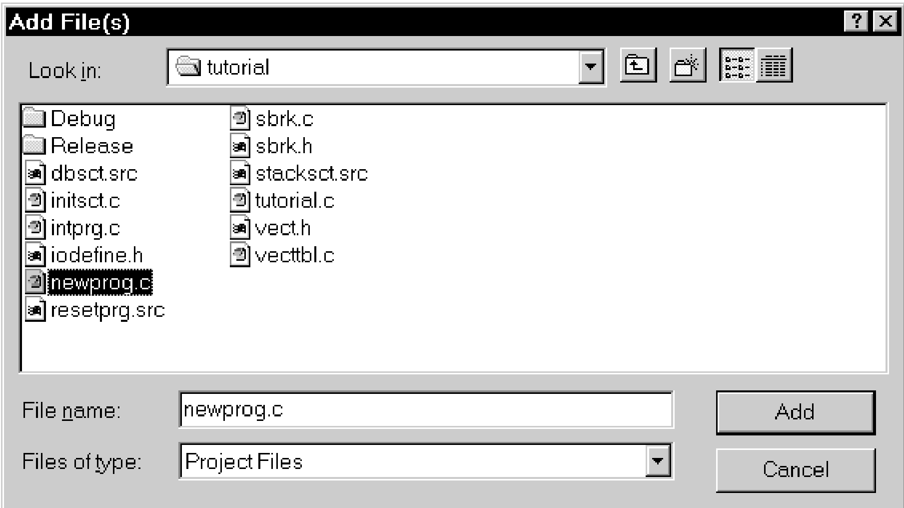


Figure 1.16 Adding a File To a Project

RENESAS

The added file will be displayed on the [Projects] tab of the "Workspace" window as shown in figure 1.17.

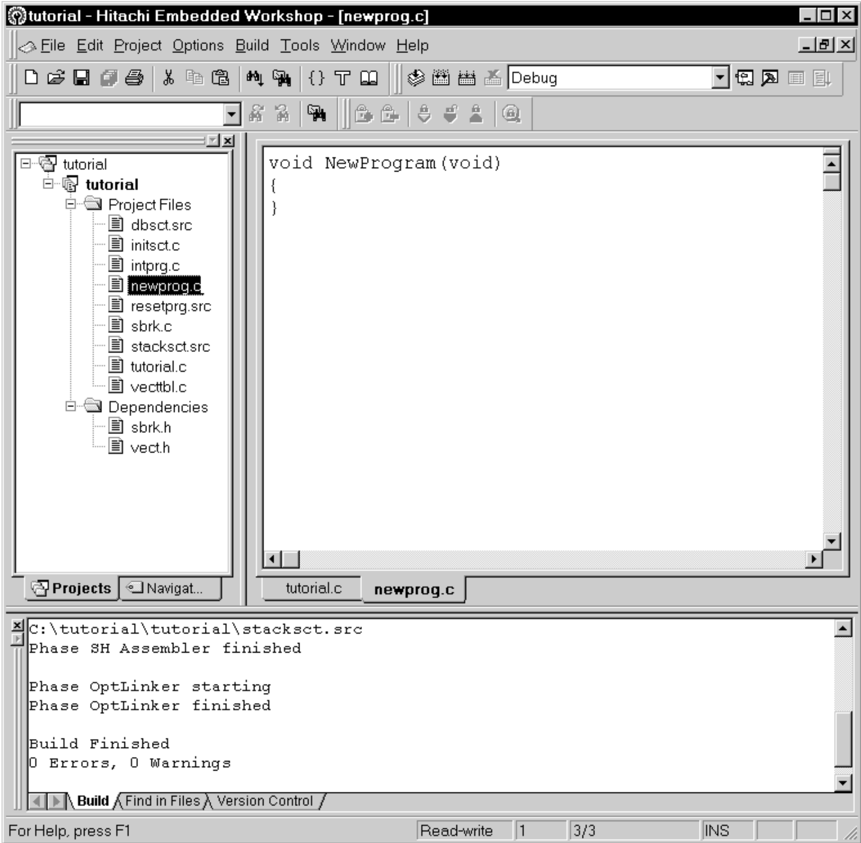


Figure 1.17 Project With a File Added

Next, how to delete a file from a project will be shown. Select [Project->Remove File…], then the "Remove Project Files" dialog box (figure 1.18) will be displayed. Select one or more project files on this dialog box and click the [Remove] button. Then the selcted file(s) will disappear from the list of the dialog box. Clicking [OK] actually removes the file(s) from the project. Clicking [Cancel] instead will nullify the removal.

You have another way to delete a file from a project. Select the file in the [Projects] tab of the "Workspace" window and press the [Delete] key. Then the file will be deleted from the project. In this way, you cannot cancel the deletion although you can cancel it on the [Remove Project Files] dialog box.

If you want to exclude a file from a build temporarily instead of removing the file from the project, refer to section 1.4.4, "Excluding a Project File".

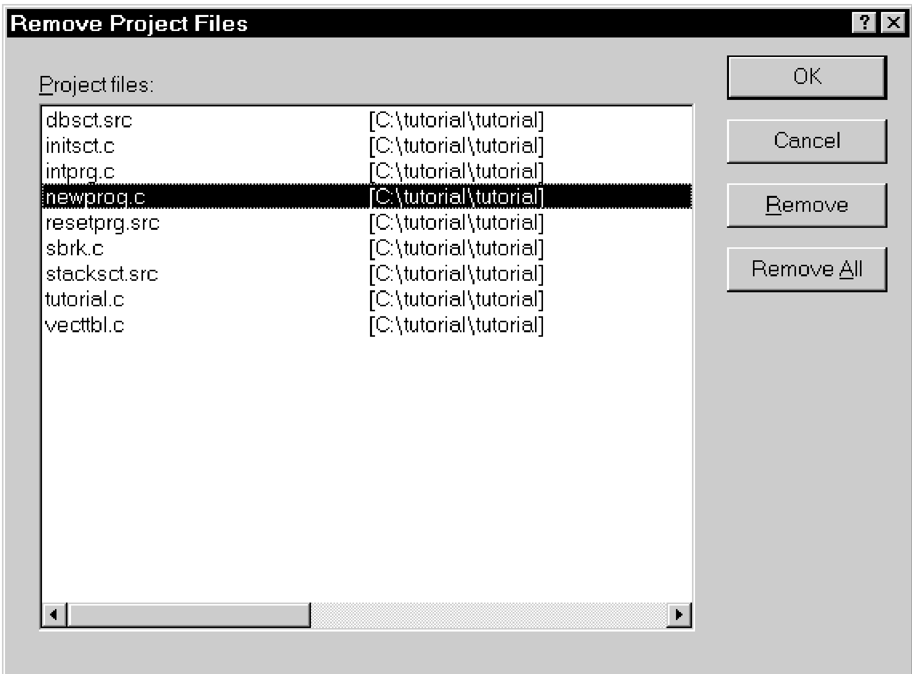


Figure 1.18 Remove Project Files Dialog Box

1.3.3 File Groups Used in a Project

In this section, a file group to which a file added to a project belongs is described. For example, the project generator adds a file described in C programming language or assembly language to the project. A file with a file extension ".c" belongs to the "C source file" group, and a file with a file extension ".src" belongs to the "Assembly source file" group.

The HEW identifies a group to which a file belongs by checking the file extension. Select [Project->File Extensions…], then the "File Extensions" dialog box (figure 1.19) will be displayed. File extensions used in the project are shown on this dialog box. Every file extension belongs to a file group. A file group can be associated to a tool in a build process.

If you want to use your own file extension in the project, define a new file extension by clicking the [Add…] button.

The following icon displayed in the [Extension] column of the "File Extensions" dialog box can show a tool which opens the file. If the icon of a file extension is the same as the icon shown below, a file with that file extension can be opened by the editor of the HEW.

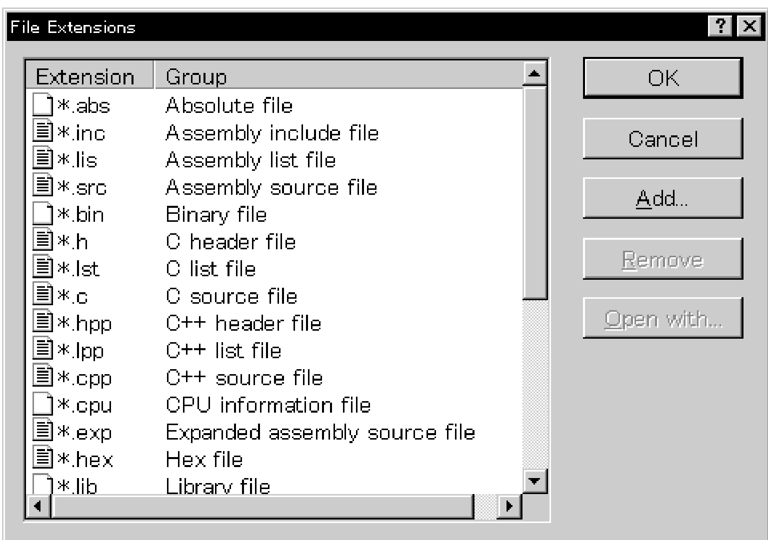


Figure 1.19 File Extensions Dialog Box (Initial Screen)

Click the [Add…] button on the “File Extensions” dialog box, then the “Define File Extension” dialog box (figure 1.20) will be launched. Enter a new file extension in the [File extension] field and specify the [File group] group box. If you want to add an extension to an existing file group, select the [Existing group] radio button and select a file group from the drop-down list. If you want to create a new file group, select the [New group] radio button and enter the name of the file group in the edit field.

In this tutorial, enter “asm” in the [File extension] field, select “Assembly source file” from the drop-down list and click [OK]. Then a file with the extension, “*.asm”, will be treated as a file in the “Assembly source file” group (figure 1.21).

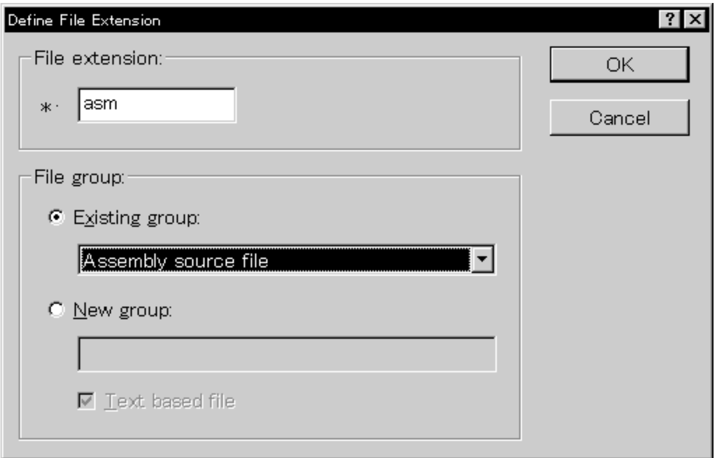


Figure 1.20 Define File Extension Dialog Box

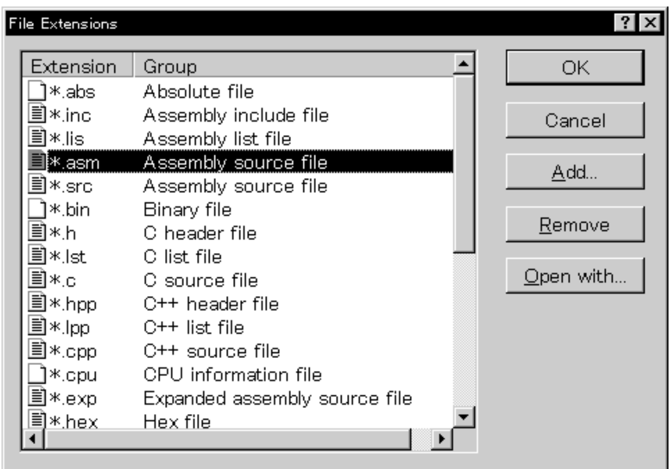


Figure 1.21 File Extension Dialog Box (After Adding an File Extension)

1.3.4 Customizing the Workspace Window

Click the right mouse button on the [Projects] tab of the “Workspace” window, then a pop-up menu will be displayed. Select [Configure View…] on this menu, then “Configure View” dialog box (figure 1.22) will be launched. On this dialog box, you can specify whether the file dependencies are shown for each file or not, whether a file is shown in a file group folder (figure 1.23) or not, and so on, can be selected.

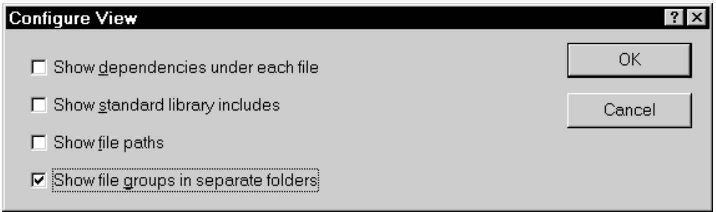


Figure 1.22 Configure View Dialog Box

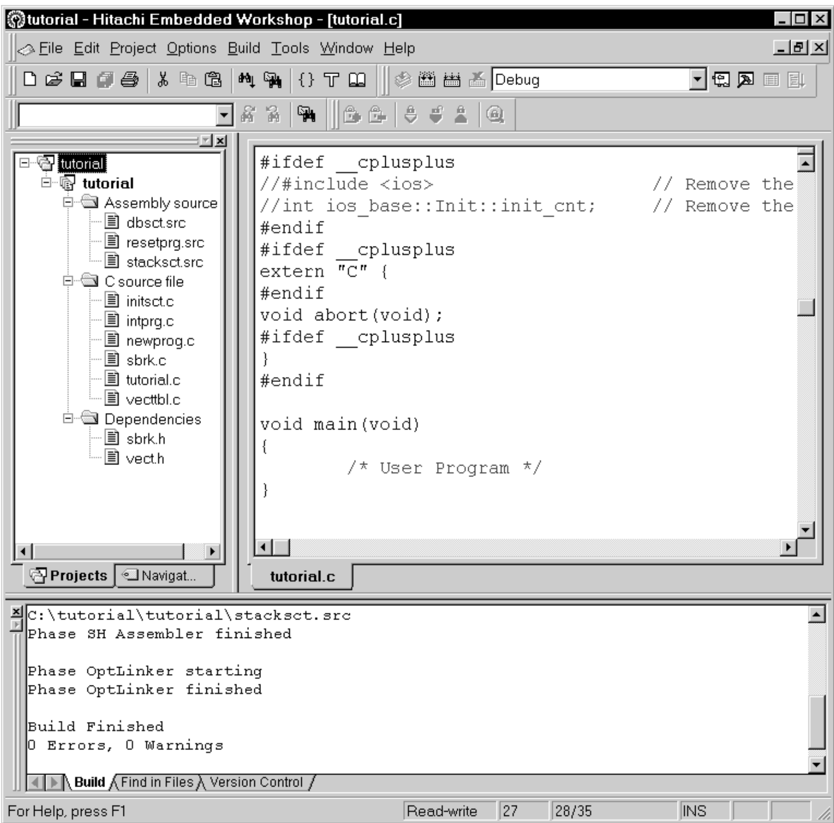


Figure 1.23 Configuration of the Workspace Window

1.4 Building a Project

1.4.1 Build Overview

The HEW provides two ways in building a project. One is “Build All”, which applies a series of tools like a compiler, an assembler and a linker to all the source program files. The other is “Build”, which applies a tool only to a source program file updated after the last build.
(The update means not only the update of a source program file, but also an update of a file included by the source program file and an update of an option applied to the source program file.)
The “Build” corresponds to [Build->Build] and the “Build All” corresponds to [Build->Build All].
If you want to compile or assemble a single source program file, select the file on the [Projects] tab of the “Workspace” window and select [Build->Build File].

CAUTION ON BUILD

1. Depending of the setting of the editor window, a file being edited might be saved on executing a build. So close a file which you do not want to be saved beforehand. For details of customizing the editor, refer to the Hitachi Embedded Workshop User’s Manual.
2. Do not save a file used in a project during the build because doing so might disturb the build process.

1.4.2 Setting Options

In this section, overview of the option setting is described.

When the project generator generates a project, the minimum options required for a build are already specified. If you want to modify the options, however, select [Options-> <Tool name>] and set options on the option dialog box of each tool.

Figure 1.24 shows the option dialog box of a C/C++ compiler. On the left hand side of the dialog box is a file list, and on the right hand side are option controls divided into some tabs. If you select a file in the file list, you can specify options only to the selected file. If you select a file group folder in the file list, you can specify the same option to all the files in the file group. To specify the same options to specific files, select the files while pushing the [Ctrl] key or the [Shift] key.

If you select "Default Options" in a file group folder, you can specify initial options for the file group. The HEW automatically adds the options specified to the "Default Options" to a file of the corresponding file group of the corresponding file extension of the file. For example, if you add a file with an extension "C" ("*.C") to the project, the settings of the "Default Options" of the "C source file" group will be added to the file automatically. For the details of the options refer to the manual of the corresponding tool.

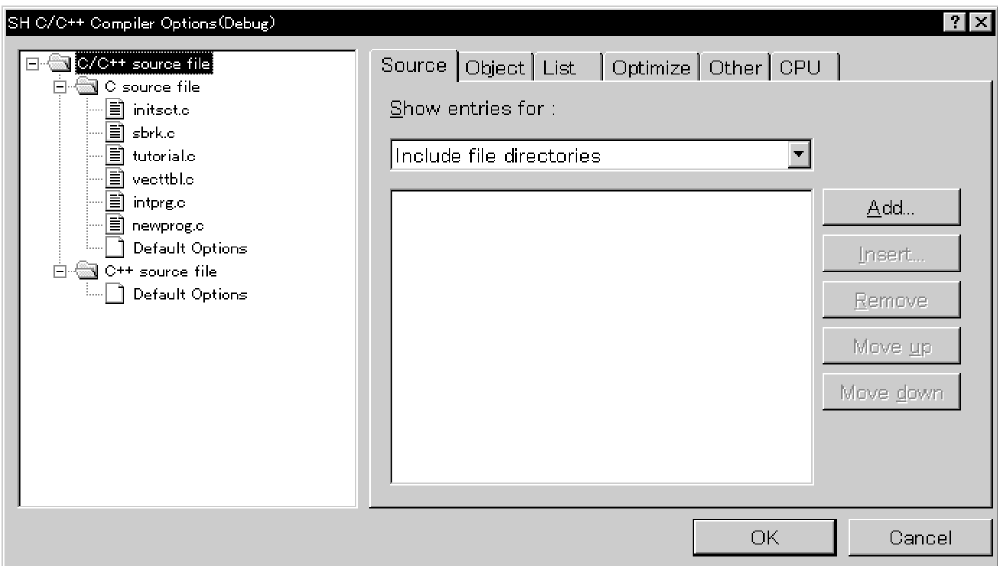


Figure 1.24 Option Dialog Box of C/C++ Compiler

1.4.3 Customizing the Configuration

In the previous section, how to set options to a project file is described. In this section, how to create more than one suite of option settings for a build will be described. A suite of option settings for a build is called a configuration.

The project generator creates two configurations, [Debug] and [Release]. Difference between them is only the setting of the debug option. To switch the current configuration to another, launch the "Build Configurations" dialog box (figure 1.25) by selecting [Options-> Build Configurations…], select a configuration from the "Current configuration" drop-down list, and click [OK]. You can also change the current configuration by selecting one from the drop-down list on the toolbar.

On the "Build Configurations" dialog box, you can create a new configuration or delete a configuration. For details, refer to chapter 2, "Build Basics", of the Hitachi Embedded Workshop User's Manual.

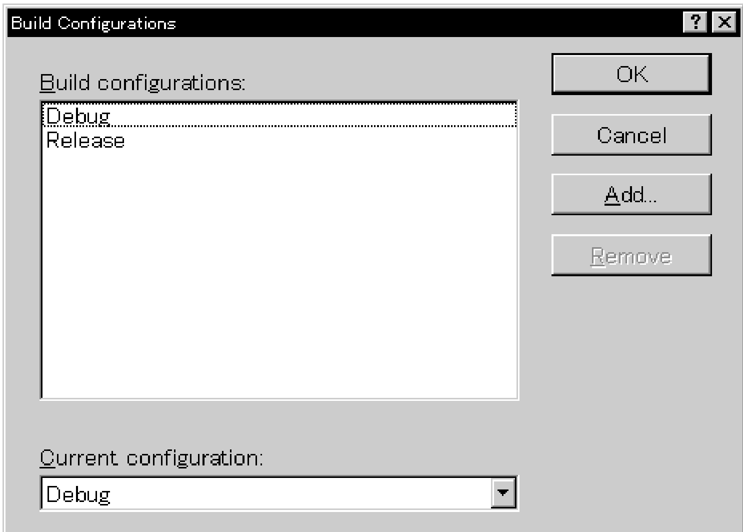


Figure 1.25 Build Configurations Dialog Box

1.4.4 Excluding a Project File from Build

In this section, how to exclude a project file from build will be described. Excluding a project file from build does not delete the file from a project, but exclude a project file from build in configuration by configuration basis.

Select a file on the “Projects” tab of the “Workspace” window, click the right mouse button, and select [Exclude Build <file>] on the pop-up menu. Then a red cross will be added to the icon of this file as shown in figure 1.26. This file will be excluded from a build.

To include an excluded file into build, select the file on the “Projects” tab of the “Workspace” window, click the right mouse button, and select [Include Build <file>] on the pop-up menu.

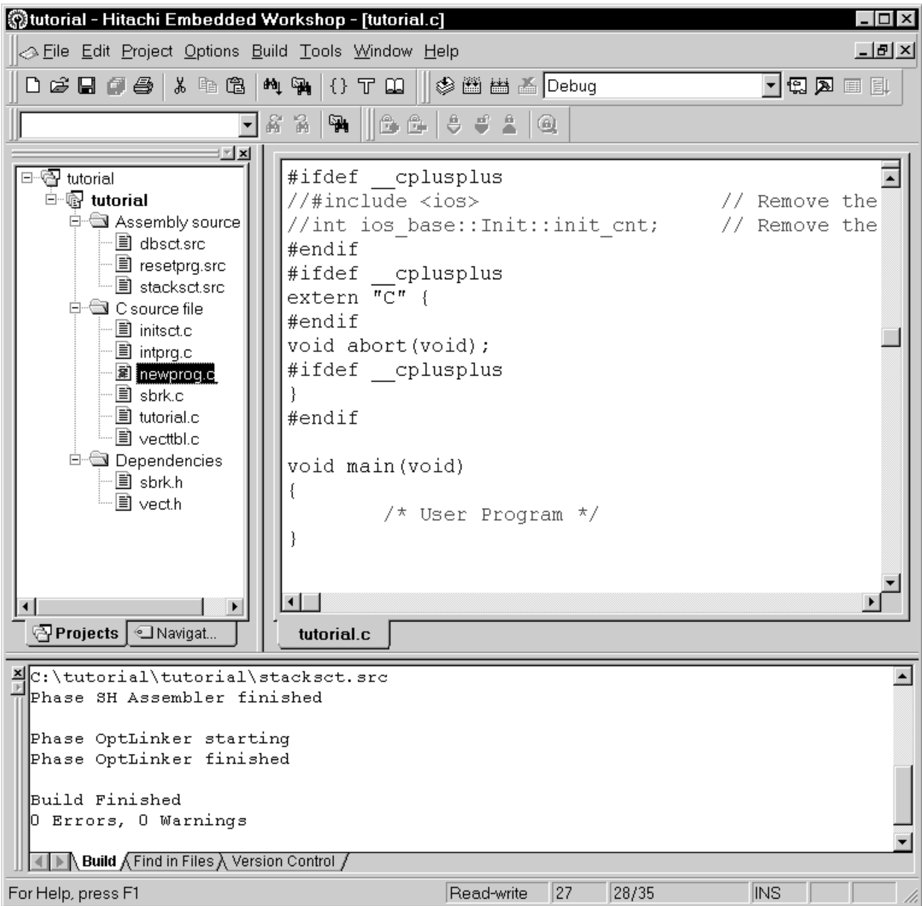


Figure 1.26 Excluding a Project File from Build

1.4.5 Correcting Errors in a Build

In this section, how to jump to a line of a file which caused an error in compiling or assembling will be described.

Figure 1.27 shows an example in which a compiler detected a syntax error in a source program file. When you build a project, messages, including error messages, from a tool in a build will be displayed on the “Output” window. If you double click the line of the error message on the [Build] tab of the “Output” window, the file which caused the error will be opened and the cursor will be placed on the line which caused the error. (The cursor might not be placed on the line which caused the error if you have already edited the file.)

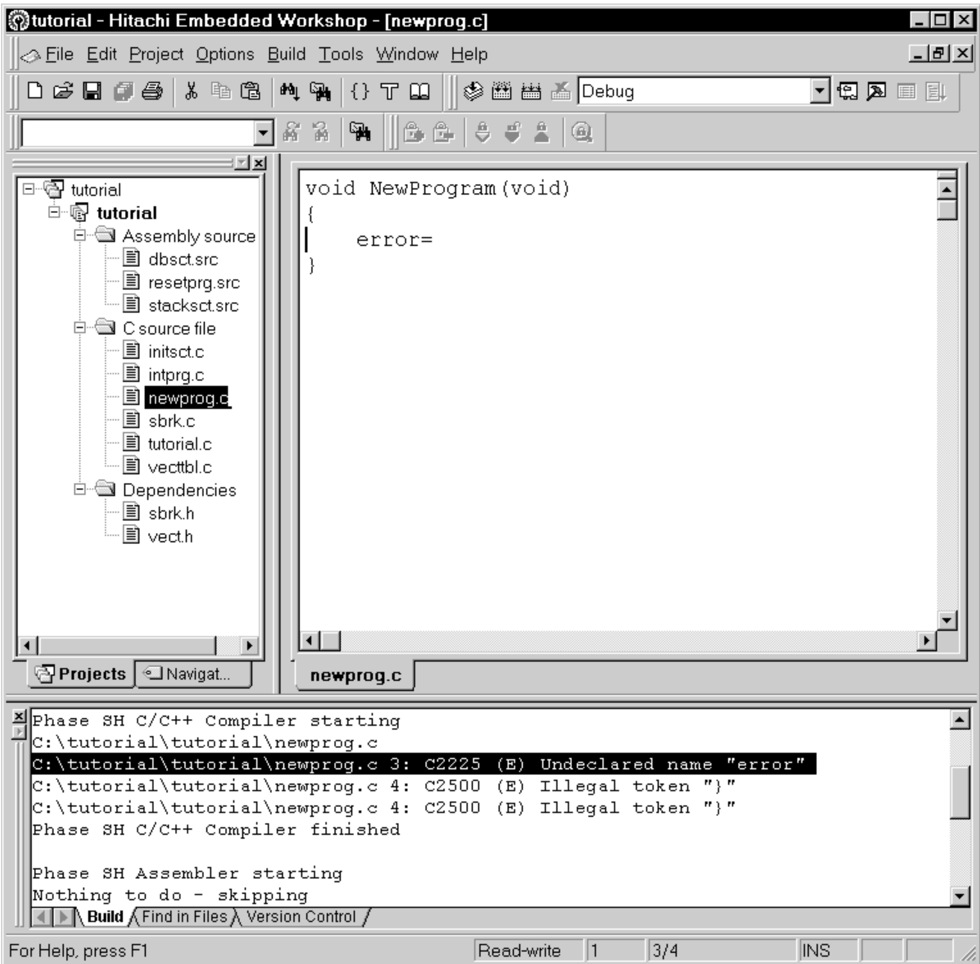


Figure 1.27 Correcting the Line which Caused an Error

1.5 HDI Interface

1.5.1 Launching HDI

The HEW can launch the HDI (Version 4.0 or greater). To register the HDI in the HEW, open the [Debugger] tab (figure 1.28) of the “Tools Customize” dialog box which is launched by selecting [Tools->Customize…]. Specify the path of the HDI.EXE in the [HDI location (V4.0 or greater)] field. In the [Session file] field, specify the path of the session file to be loaded. How to generate a session file is shown in the SuperH RISC engine Simulator/Debugger User’s Manual. If you set the two fields, the toolbar button shown below will be activated. Thus clicking the button will launch the HDI and it will load the session file.

If you specify the path of the load module file on the [Download module] field, the HDI will be focused immediately after the load module file is updated by the HEW.

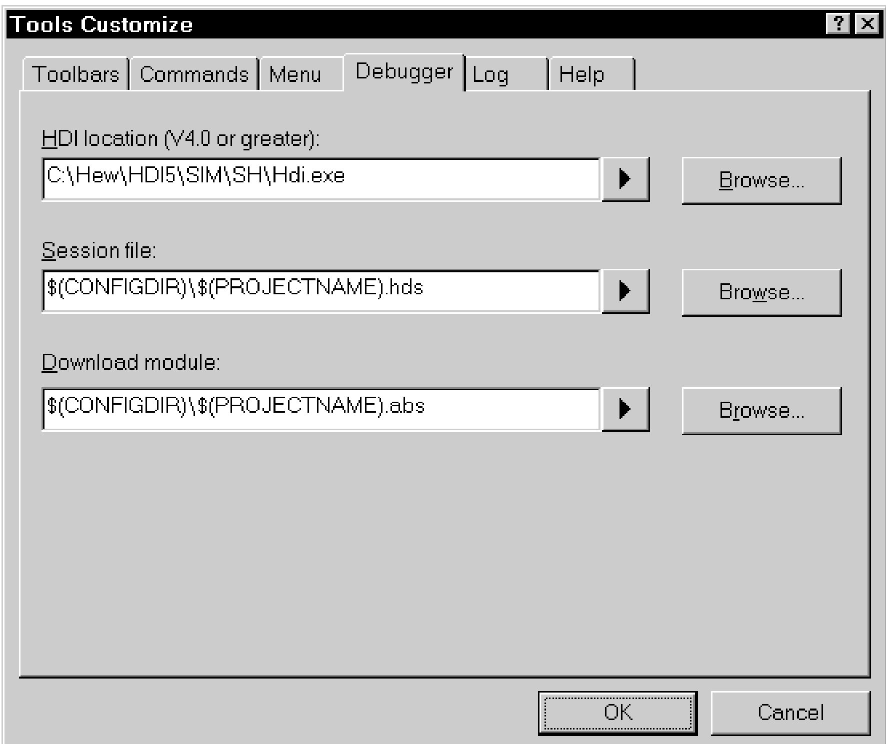


Figure 1.28 Tools Customize Dialog Box, Debugger Tab

If you double click a line on the source window (figure 1.29) of the HDI, the corresponding file will be opened by the HEW and the cursor will be placed on the same line as you have double clicked.

Figure 1.30 shows the HEW after the line of “void main(void)” of “tutorial.c” is double clicked on the source window of the HDI.

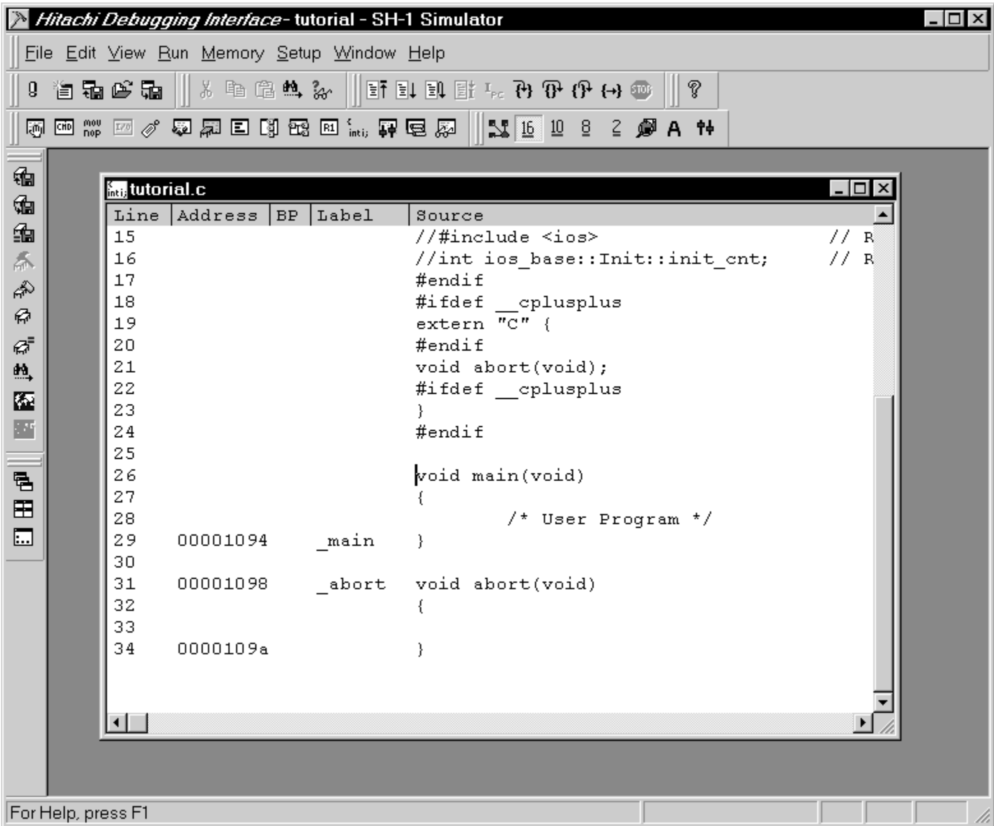


Figure 1.29 Source Window of the HDI

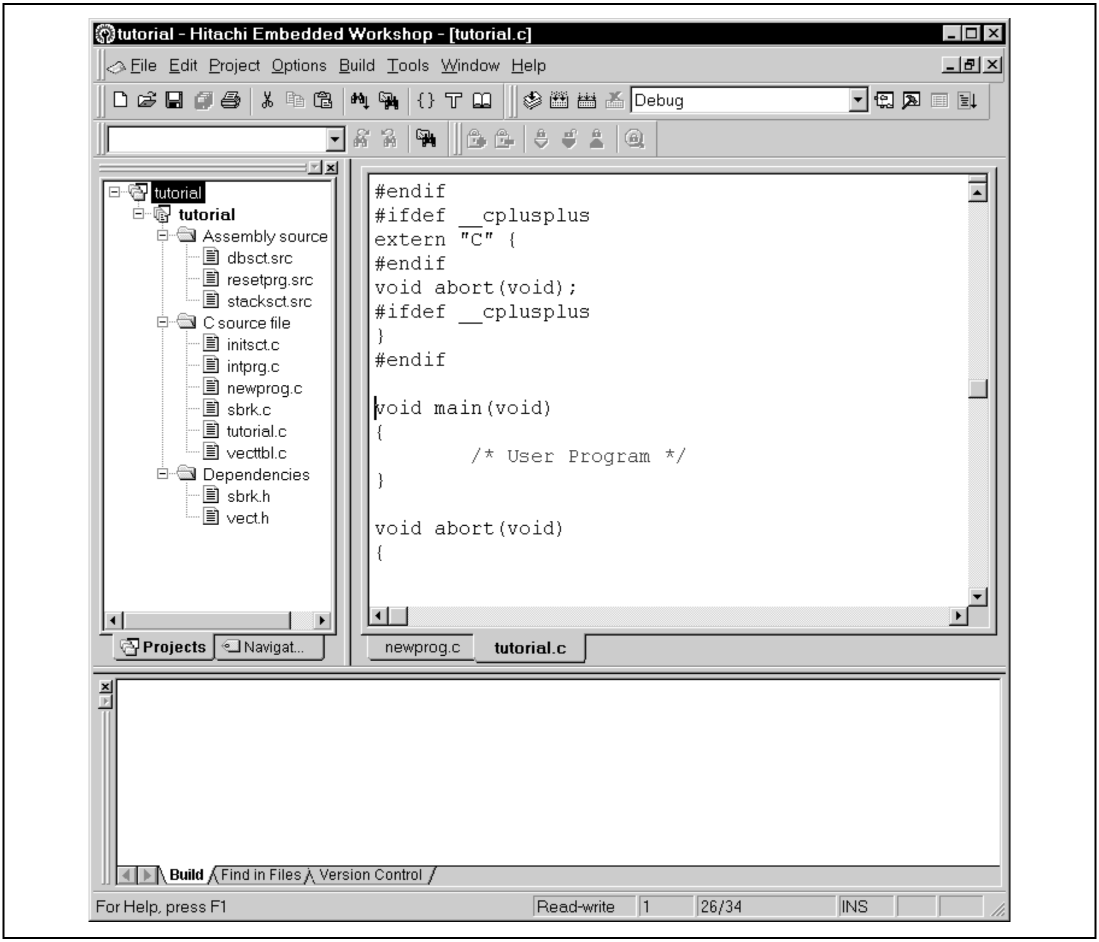


Figure 1.30 The Editor Windows of the HEW Opened by the HDI

RENESAS

1.5.2 Demonstration Project

If you choose Demonstration for the project type in the New Project Workspace (figure 1.3) when you create a project, you can create a sample project that has a Simulated I/O function of the HDI Simulator/Debugger. This project uses a standard output function printf to display the execution status of the program and data on function main.

Before you use the Simulated I/O function, you must set the System Configuration Dialog Box (open this dialog box by selecting [Setup -> Configure Platform…]) after you have set the debugging environment via the HDI.

Select the [Enable] checkbox for the System Call Address in this dialog box and enter the value of SIM_IO, defined in lowlvl.src by HEW, as the address. The address may differ depending on the type of the CPU.

SIM_IO: .EQU H'0000 (In this case, enter 0 for the address.)

If you open the Simulated I/O Window in the HDI (open this dialog box by selecting [View -> Simulated I/O]) and run the program, you can debug the program using the Simulated I/O function.

For details, refer to the SuperH RISC engine Simulator/Debugger User's Manual.

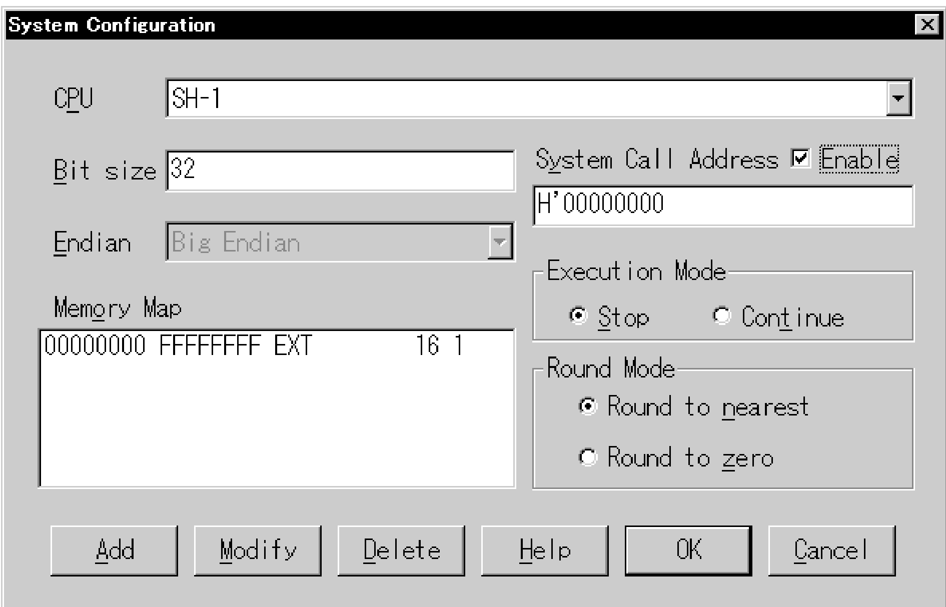


Figure 1.31 HDI System Configuration Dialog Box

1.6 Exiting From the HEW

Selecting [File->Exit] will close the HEW. Depending on the setting, the message box (figure 1.32) which asks you whether to save the workspace or not will be launched. Not only the setting like this regarding exiting from the HEW but also setting regarding initiation of the HEW can be specified via [Tools->Options…]. For details, refer to the Hitachi Embedded Workshop User's Manual.

The HDI and the other tools launched from the HEW will not be closed when the HEW is closed. Close the tools by yourself.

Figure 1.32 Confirmation Dialog Box for Saving Workspace on Exiting from the HEW

Section 2 HDI Tutorial

2.1 Using HDI with the Simulator/Debugger

The following describes a sample program debugging session, designed to introduce the main features of the simulator/debugger used in conjunction with the Hitachi Debugging Interface (HDI) software.

2.1.1 Introduction

This sample program is based on a C program that sorts ten random data items first in ascending order, then in descending order.

The sample program performs the following actions:

- With the `main` function, random data to be sorted is generated.
- With the `sort` function, an array is input, within which the random data generated by the `main` function is stored, and the data is sorted in ascending order.
- With the `change` function, the array generated by the `sort` function is input, and the data is sorted in descending order.

The sample program is provided on the installation disk as file `sort.c.` A compiled version of the tutorial is provided in file `sort.abs`.

2.1.2 Running HDI

Run the HDI interface software independently and not from the Hitachi Embedded Workstation (HEW) since the sample program is used to explain the functions of the HDI.

To run the HDI interface software, double-click the **[HDI for SH series simulator]** icon.

Figure 2.1 Icon of HDI for SH series simulator

2.1.3 Selecting the Target

The HDI can be extended to support multiple target platforms. If your system is set up for more than one platform, you will be prompted to choose a platform for the current session.

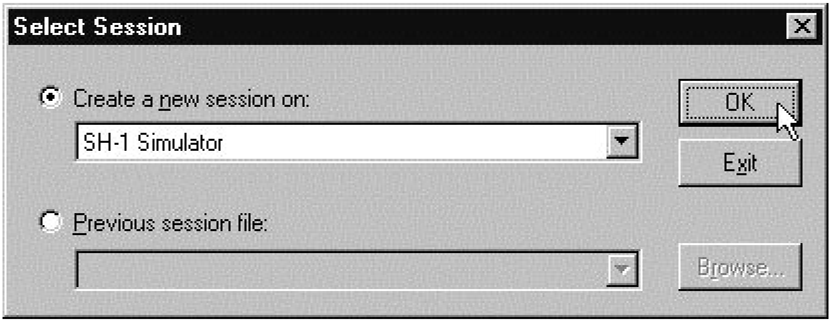


Figure 2.2 Select Session Dialog Box

For this tutorial, select `SH1 Simulator` and click the **[OK]** button to continue.

Note that you can change the target platform at any time by choosing **[New Session…]** from the **[File]** menu. If you have only one platform installed, this menu option will not be available.

When the simulator/debugger has been successfully set up, the Hitachi Debugging Interface window will be displayed, with the message "Link up" in the status bar. Figure 2.3 shows the key functions of the window:

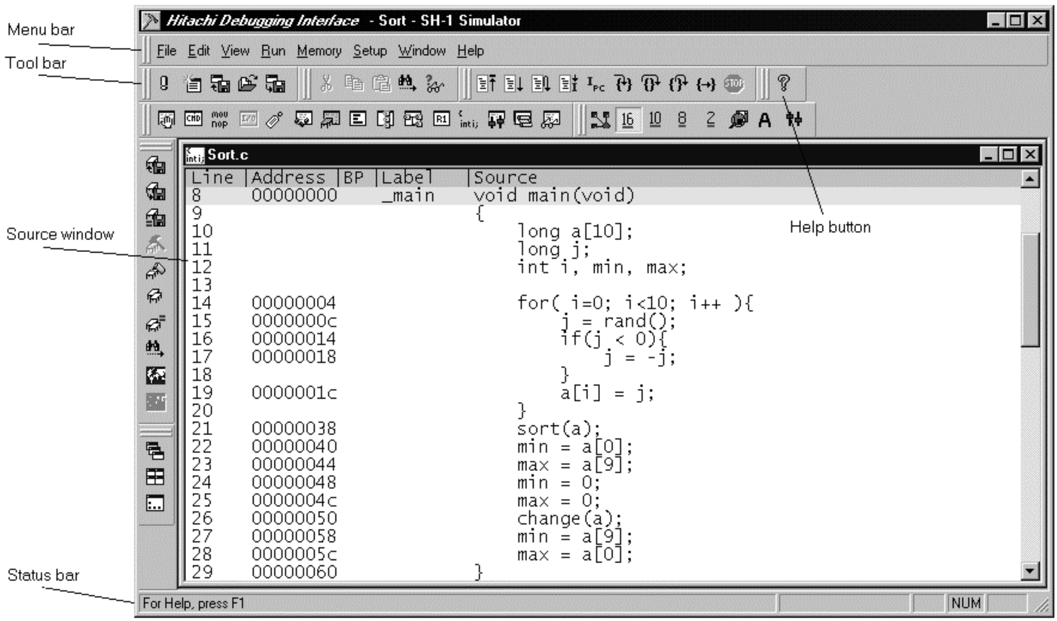


Figure 2.3 Hitachi Debugging Interface Window

The key functions of Hitachi Debugging Interface are described in the following sections.

Menu bar: Give you access to the HDI commands for using the HDI debugger.

Toolbar: Provides convenient buttons as shortcuts for the most frequently used HDI commands. Tool buttons from the same menu are displayed in the same block. The tool buttons can be moved within the toolbar by clicking dragging them. Whether to display the tool buttons can be selected by using the **[Toolbar]** option in the **[Customize]** submenu in the **[Setup]** menu.

Source window: Displays the source of the program being debugged.

Status bar: Displays the status of the simulator/debugger, and progress information about downloading.

Help button: Activates context sensitive help about any feature of the HDI user interface.

2.1.4 Mapping the Memory Resource

The next step is to map the memory resource used to operate an application being developed.

- Choose **[Configure Map...]** from the **[Memory]** menu to display the current memory mapping.

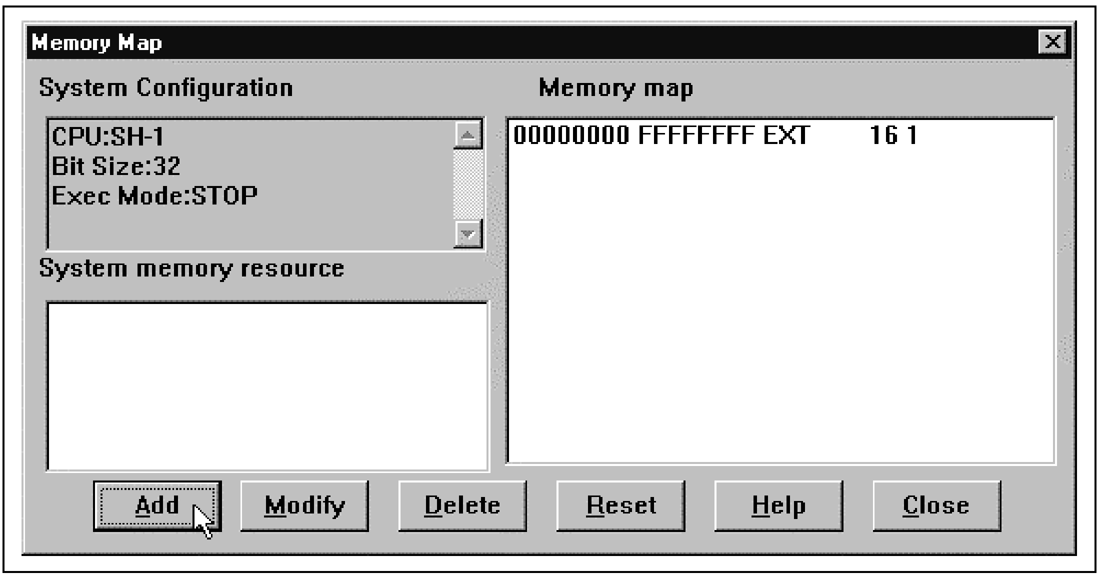


Figure 2.4 Memory Map Dialog Box

- Clicking the **[Add]** button displays the **System Memory Resource Modify** dialog box.

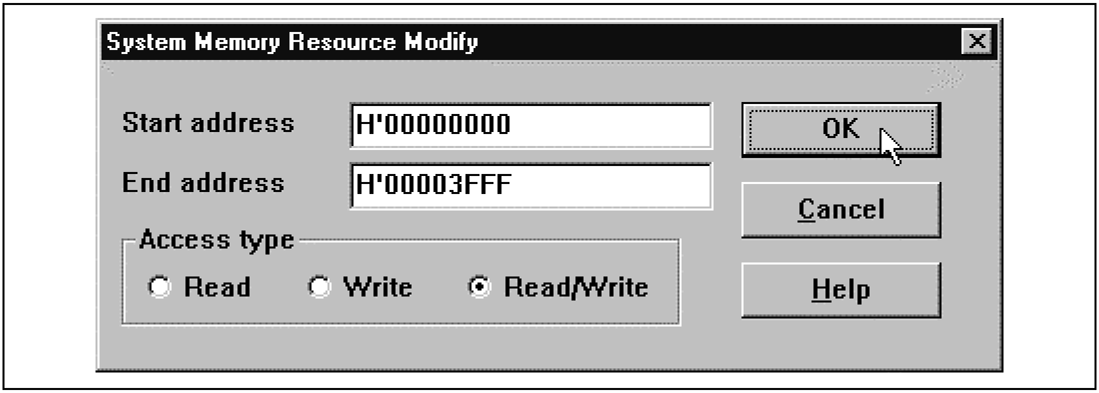


Figure 2.5 System Memory Resource Modify Dialog Box

In the **[Access type]** box, you can specify one of the following three access types:

— Read: Only read enabled

— Write: Only write enabled

— Read/Write: Read and write enabled

For this tutorial, map the memory area of addresses ranging from H'00000000 to H'00003FFF as a read/write enabled area.

- Edit the **[Start address]** and **[End address]** fields to H'00000000 and H'00003FFF, respectively, set the **[Access type]** as **[Read/Write]**, and click the **[OK]** button.

  The **Memory Map** dialog box will now show the modified ranges.

- Click the **[Close]** button to close the dialog box.

2.1.5 Downloading the Tutorial Program

- To open the **Load Program** dialog box, choose **[Load Program...]** from the **[File]** menu.

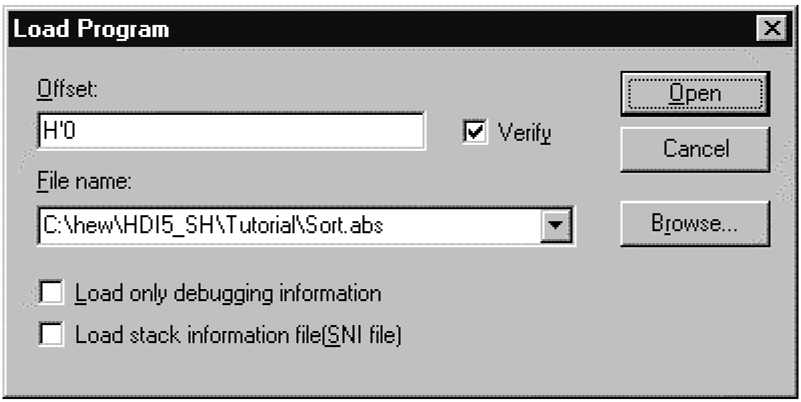


Figure 2.6 Load Program Dialog Box

- In **[File name]**, enter the path for the directory where file `sort.abs` is stored, or click the **[Browse...]** button to open the **Open** dialog box and select the file `sort.abs` and click the **[Open]** button.

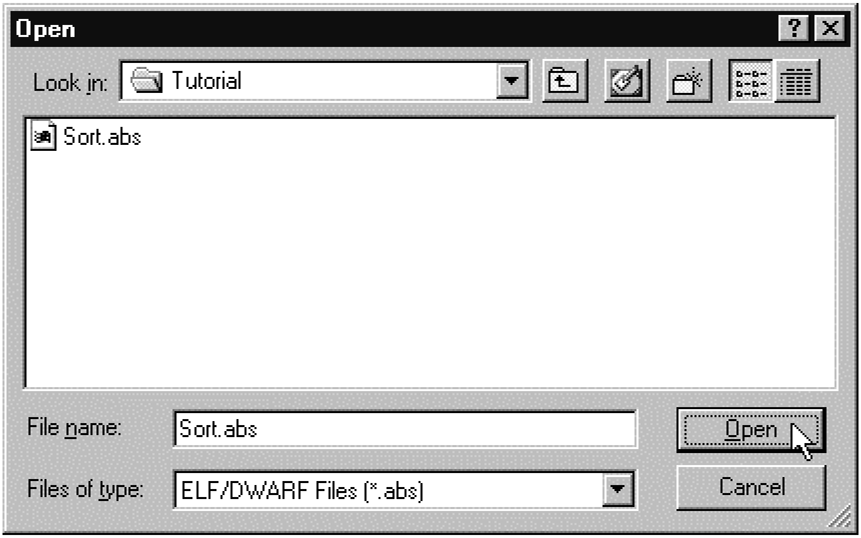


Figure 2.7 Open Dialog Box (Load Program...)

- Click the **[Open]** button in the **Load Program** dialog box.

  When the file has been loaded, the following dialog box displays information about the memory areas that have been filled with the program code.

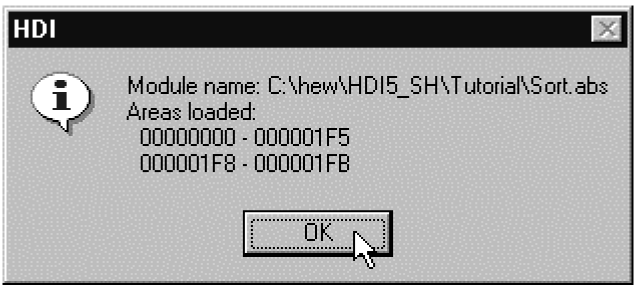


Figure 2.8 HDI Dialog Box

- Click the **[OK]** button to continue.

2.1.6 Displaying the Source Program

The HDI allows you to debug a program at the source level, so that you can see a listing of the C program alongside the machine code as you debug. To do this, read the C source file that corresponds to the object file.

- Choose **[Source...]** from the **[View]** menu.
- You will be prompted for the C source file that corresponds to the object file you have loaded.

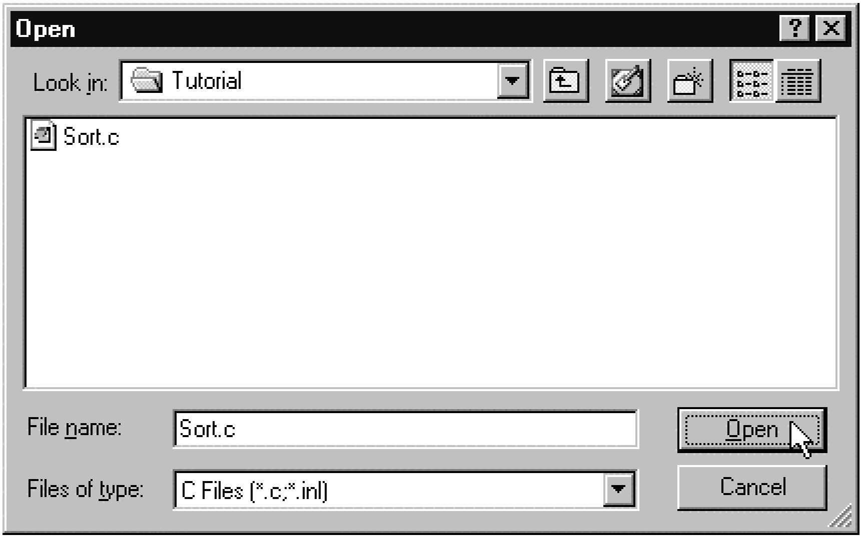


Figure 2.9 Open Dialog Box (Source...)

- Select `sort.c` and click the **[Open]** button to display the **Source** window.

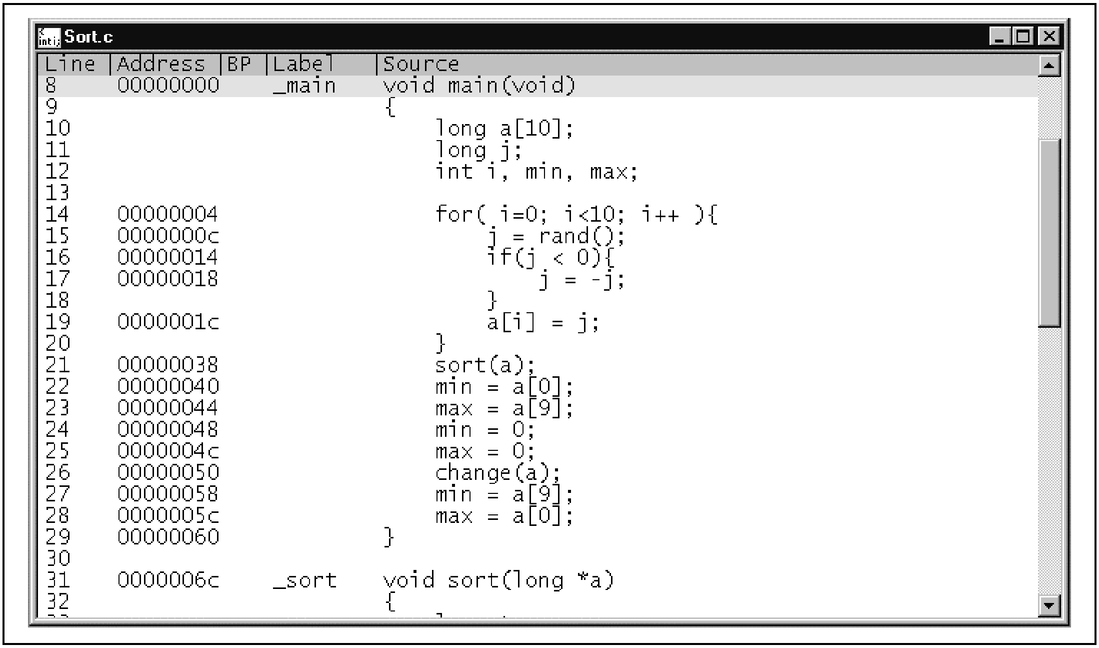


Figure 2.10 Source Window (Displaying the Source Program)

- If necessary, choose the **[Font]** option from the **[Customise]** submenu on the **[Setup]** menu to choose a font and size suitable for your computer.

Initially the **Source** window shows the start of the main program, but you can use the scroll bar to scroll through the program to see the other statements.

2.1.7 Setting a PC Breakpoint

The **Source** window provides a very simple way of setting a breakpoint at any point in a program. For example, to set a breakpoint at the `sort` function call:

- Double-click the **[BP]** column on the line containing the `sort` function call.

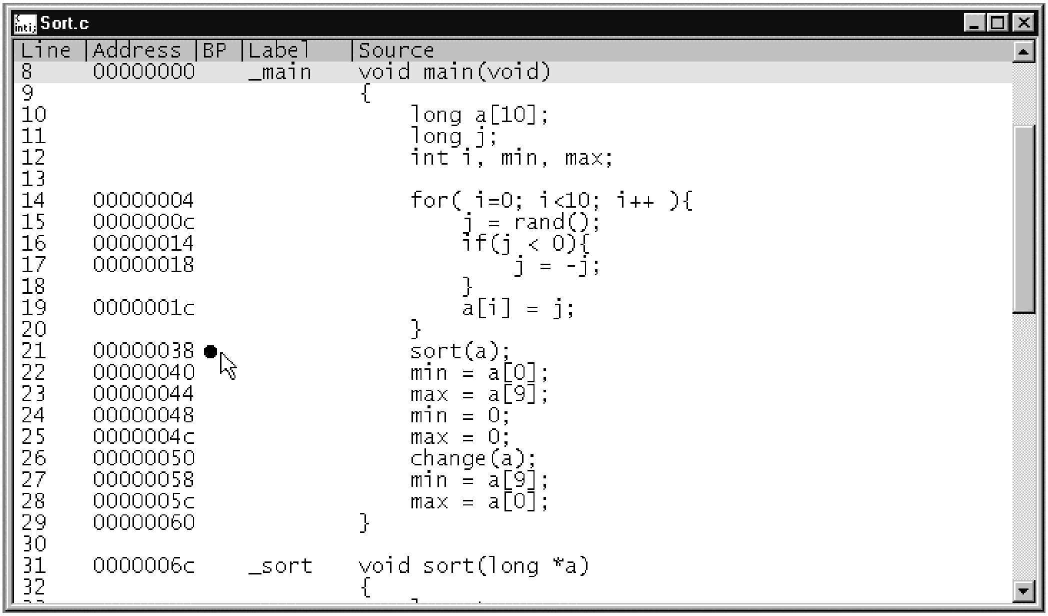


Figure 2.11 Source Window (Setting the Breakpoint)

The mark ● will be displayed on the line containing the `sort` function to show that a PC breakpoint is set at that address.

2.1.8 Setting Trace Information Acquisition Conditions

- Choose **[Trace]** from the **[View]** menu to open the **Trace** window. Click the right mouse button on the **Trace** window and choose **[Acquisition]** from the pop-up menu.

  The following **Trace Acquisition** dialog box is displayed.

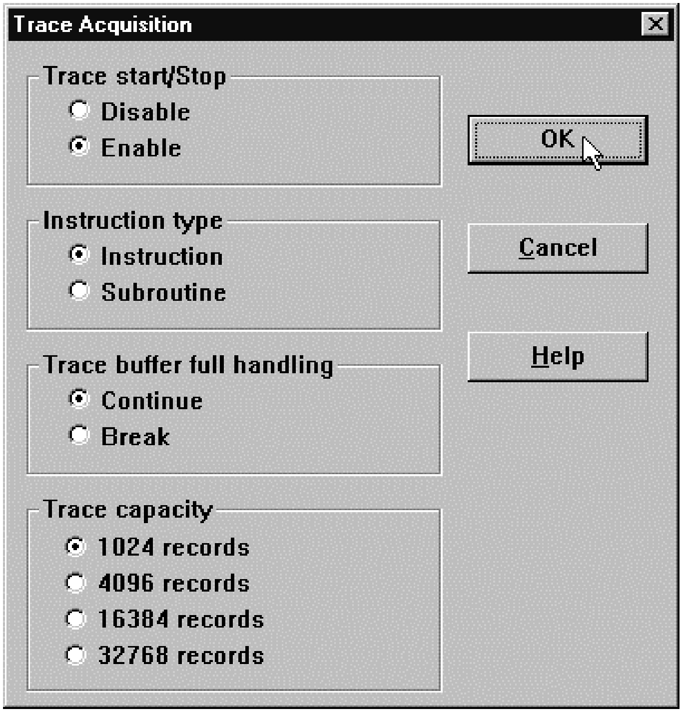


Figure 2.12 Trace Acquisition Dialog Box

- Set **[Trace start/Stop]** to **[Enable]** in the **Trace Acquisition** dialog box, and click the **[OK]** button to make the trace information acquisition effective.

2.1.9 Setting Performance Analysis

- Choose **[Performance Analysis]** from the **[View]** menu to open the **Performance Analysis** window. Click the right mouse button on the **Performance Analysis** window and choose **[Add Range...]** from the pop-up menu.
- The following **Performance Option** dialog box will be displayed.

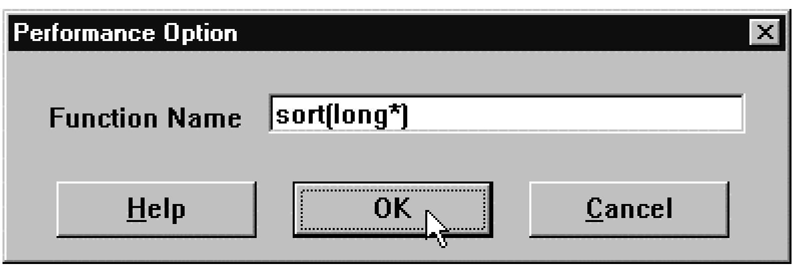


Figure 2.13 Performance Option Dialog Box

- Set **[Function Name]** as `sort` in the **Performance Option** dialog box, and click the **[OK]** button.

  In the **Performance Analysis** window, `sort(long*)` is then set in **[Function]**.

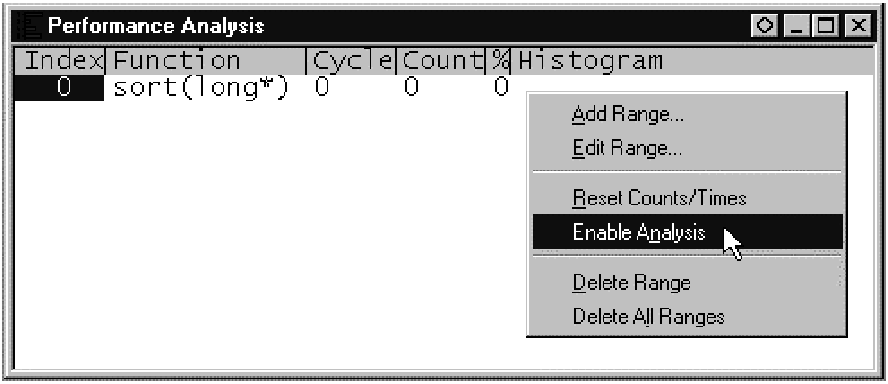


Figure 2.14 Performance Analysis Window (Setting)

- Click the right mouse button on the **Performance Analysis** window and choose **[Analysis Enabled]** from the pop-up menu to activate the performance analysis information acquisition.
- Click **[X]** in the title bar to close the window.

2.1.10 Setting the Stack Pointer

Set the stack pointer to run the program.

- Choose **[Registers]** from the **[View]** menu to open the **Registers** window.
- Double-click the **[Value]** column of **[R15]** in the **Registers** window to modify the value of the stack pointer R15.

The following dialog box enables the value to be modified.

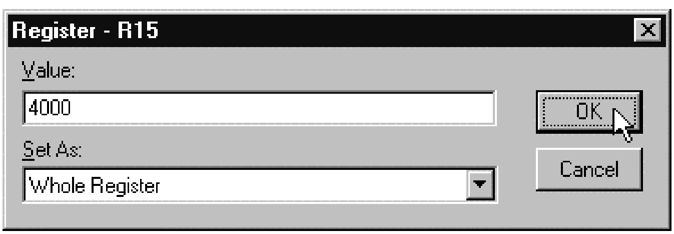


Figure 2.15 Register Dialog Box

- Set `H'4000` for the value of the stack pointer in this sample program, and click the **[OK]** button.

2.1.11 Executing the Program

- To run the program, choose **[Go]** from the **[Run]** menu, or click the **[Go]** button on the toolbar.

Figure 2.16 Go Button

The program will be executed up to the breakpoint you inserted, and a statement will be highlighted in the **Source** window to show that the program has halted, with the message Break=PC Breakpoint in the status bar.

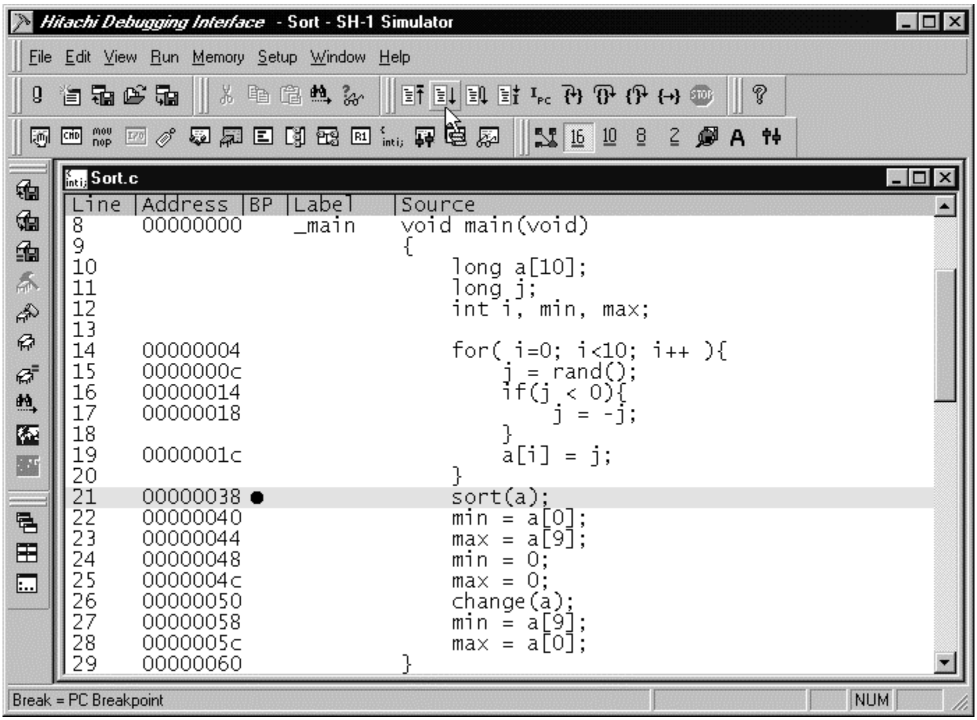


Figure 2.17 Source Window (Break Status)

When you start the HDI by clicking the **Launch Debugger** button in the HEW (see section 1.5, HDI Interface), you can display and edit the source file displayed on the source window through the HEW text editor by double-clicking the **[Source]** column in the **Source** window. In the tutorial sample program, this can be done by compiling and linking sample program `sort.c`.

You can see the cause of termination in the **System Status** window.

- Choose the **[Status]** button from the **[View]** menu to open the **System Status** window. Then choose the **[Platform]** sheet in the **System Status** window.

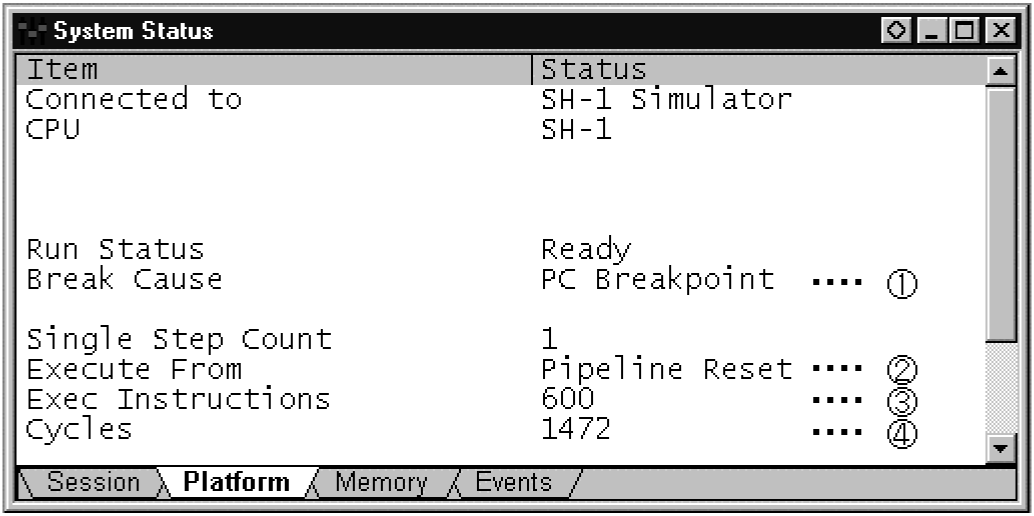


Figure 2.18 System Status Window

This shows the following execution results:

(1) The cause of the break was a PC Breakpoint.

(2) Execution was from the pipeline reset.

(3) The number of instructions executed by the GO command was 600.

(4) The number of cycles executed from the pipeline reset was 1472.

You can also see the value of the registers in the **Registers** window.

- Choose **[Registers]** from the **[View]** menu.

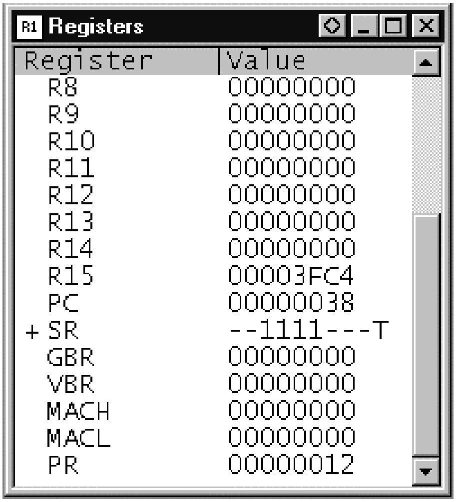


Figure 2.19 Registers Window

You can see the value of each register at the program stop.

2.1.12 Using the Trace Buffer

The trace buffer allows us to look back over previous execution cycles to see what accesses took place.

- Open the **Trace** window by choosing **[Trace]** from the **[View]** menu. Scroll up the window so that you can see the first few cycles.
- If necessary, choose the **[Font]** option from the **[Customise]** submenu on the **[Setup]** menu to choose a font and size suitable for your computer.

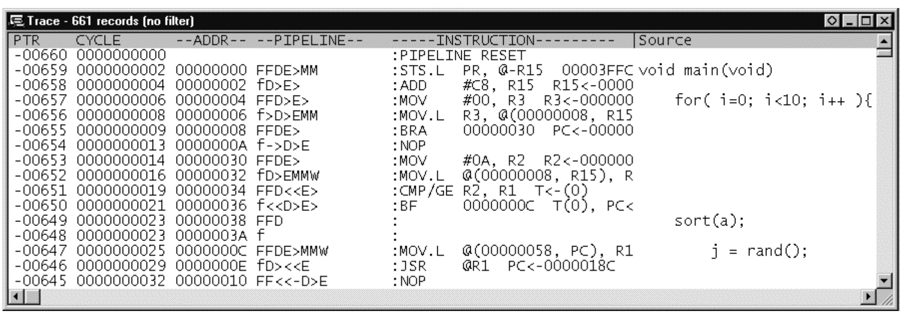


Figure 2.20 Trace Window (Displaying Trace Information)

RENESAS

You can see that the `main` function is executed from cycle 0000000000 (pipeline reset) in the trace display shown in Figure 2.20.

2.1.13 Trace Search

Click the right mouse button on the **Trace** window and choose **[Find...]** from the pop-up menu to open the **Trace Search** dialog box.

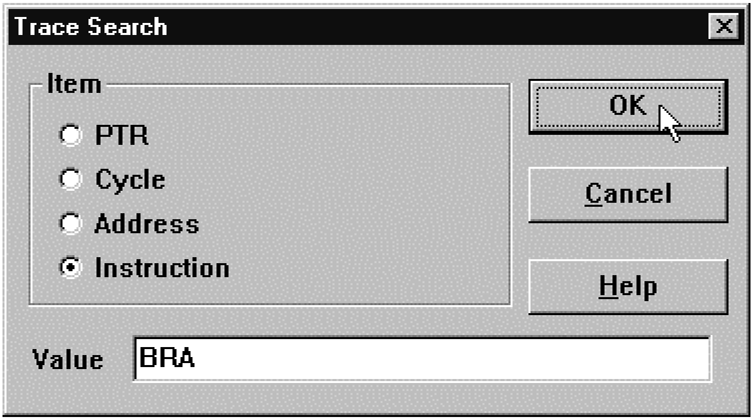


Figure 2.21 Trace Search Dialog Box

A trace search is executed by specifying the search item **[Item]** and search contents **[Value]** then clicking the **[OK]** button. When the corresponding trace information is found, the first line of the information is highlighted. When continuing trace search with the same contents, click the right mouse button on the **Trace** window and choose **[Find Next]** from the pop-up menu. The next corresponding line is highlighted.

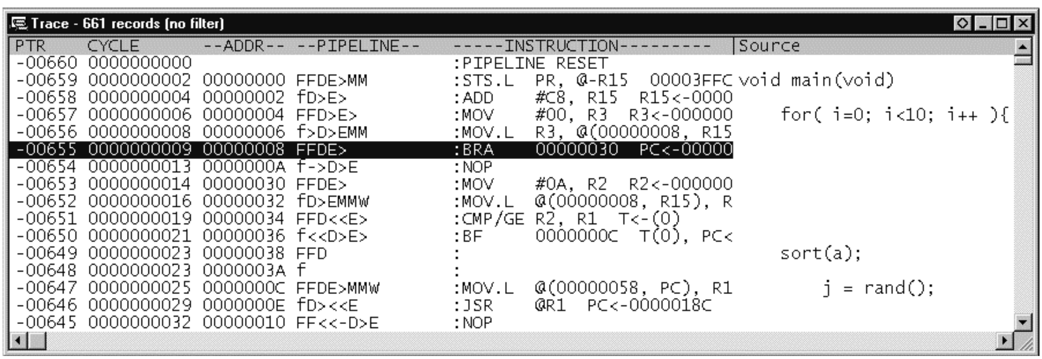


Figure 2.22 Trace Window (Search Results)

2.1.14 Reviewing Breakpoints

You can see a list of all the breakpoints set in the program in the **Breakpoints** window.

- Choose **[Breakpoints]** from the **[View]** menu.

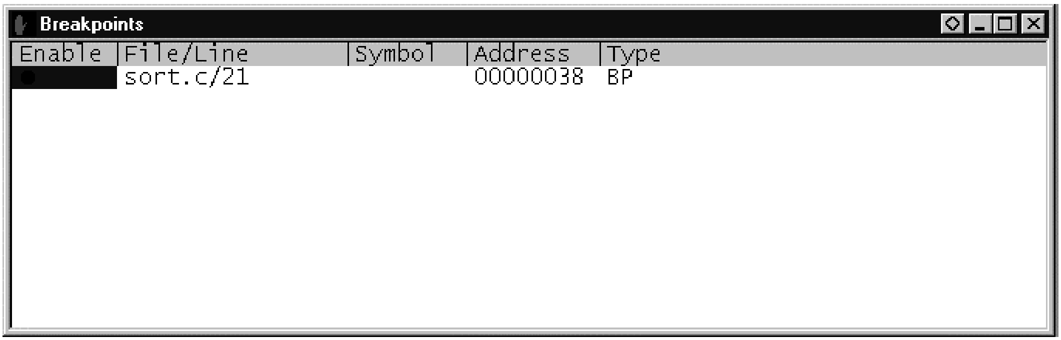


Figure 2.23 Breakpoints Window

The **Breakpoints** window also allows you to enable and disable breakpoints, define new breakpoints, and delete breakpoints.

- Close the **Breakpoints** window.

2.1.15 Viewing Memory

You can view the contents of a memory block in the **Memory** window. For example, to view the memory corresponding to array main in word size:

- Choose **[Memory…]** from the **[View]** menu, enter `main` in the **[Address]** field, and set **[Format]** as `Word`.

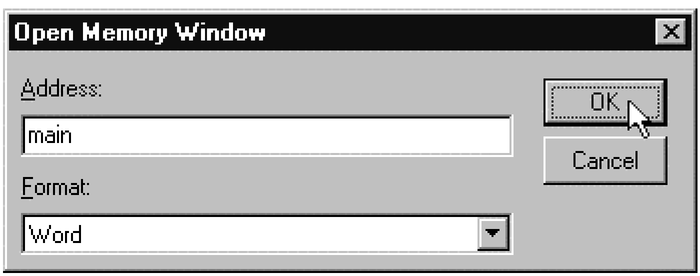


Figure 2.24 Open Memory Window Dialog Box

- Click the **[OK]** button to open the **Word Memory** window showing the specified area of memory.

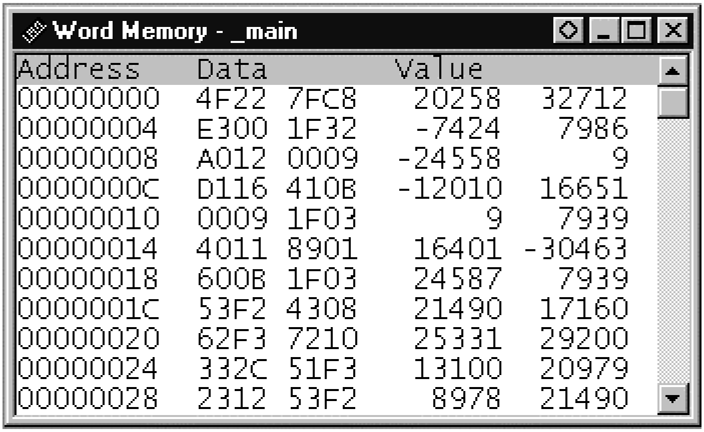


Figure 2.25 Word Memory Window

2.1.16 Watching Variables

You can view the contents of variables in the **Watch** window.

For example, set a watch on the long-type array "a" declared at the beginning of the program, by using the following procedure:

- Position the cursor to the left of "a" in the **Source** window.
- Click the right mouse button on the **Source** window and choose **[Instant Watch...]** from the pop-up menu.

  The following dialog box will be displayed.

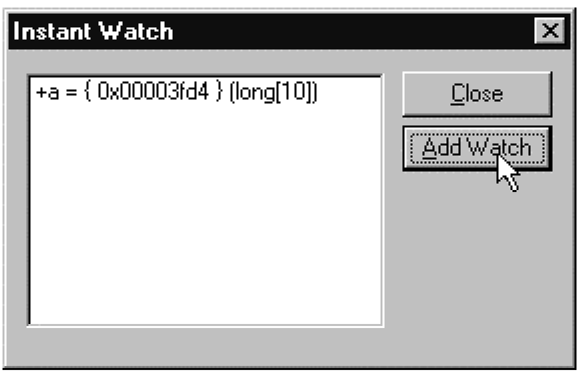


Figure 2.26 Instant Watch Dialog Box

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

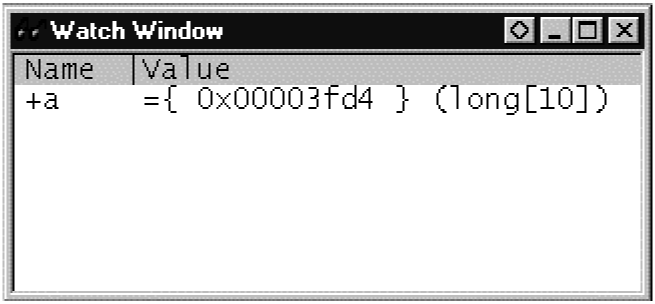


Figure 2.27 Watch Window (Displaying the Array)

You can also add a variable to the **Watch** window by specifying its name.

- Click the right mouse button on the **Watch** window and choose **[Add Watch...]** from the pop-up menu.

  The following dialog box will appear.

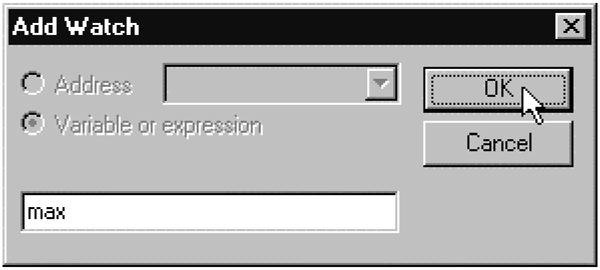


Figure 2.28 Add Watch Dialog Box

- Type variable `max` and click the **[OK]** button.

  The **Watch** window will now also show the int-type variable `max`.

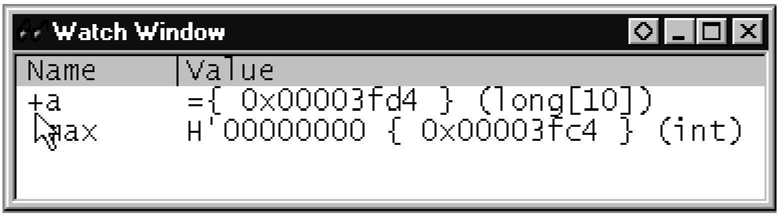


Figure 2.29 Watch Window (Displaying the Variable)

You can double-click the + symbol to the left of array "`a`" in the **Watch** window to expand the variable and show the individual elements in the array.

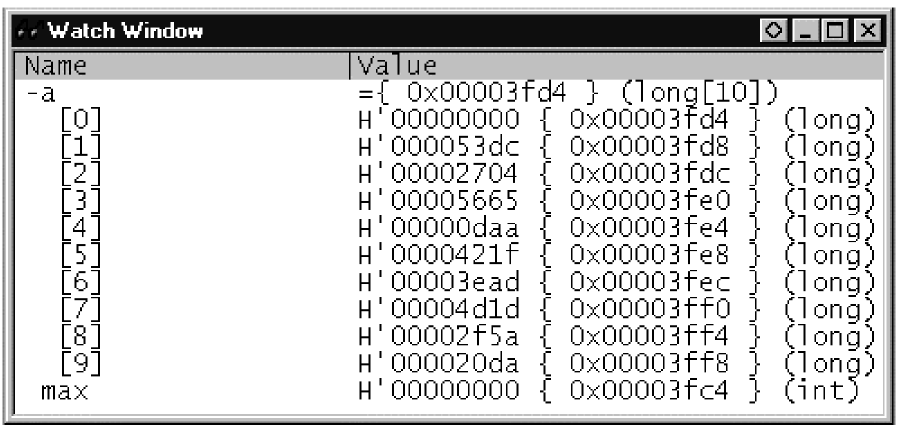


Figure 2.30 Watch Window (Displaying Array Elements)

2.1.17 Stepping Through a Program

The simulator/debugger provides a range of step menu commands that allow efficient program debugging.

Table 1.1 Step Menu Commands

Menu Command	Description
Step In	Executes every statement, including statements within functions.
Step Over	Executes a function call in a single step.
Step Out	Steps out of a function, and stops at the statement following that called the function in the program.
Step…	Steps repeatedly at a specified rate.

To demonstrate program stepping, confirm that the `sort` function statement at address H'00000038 has been executed.

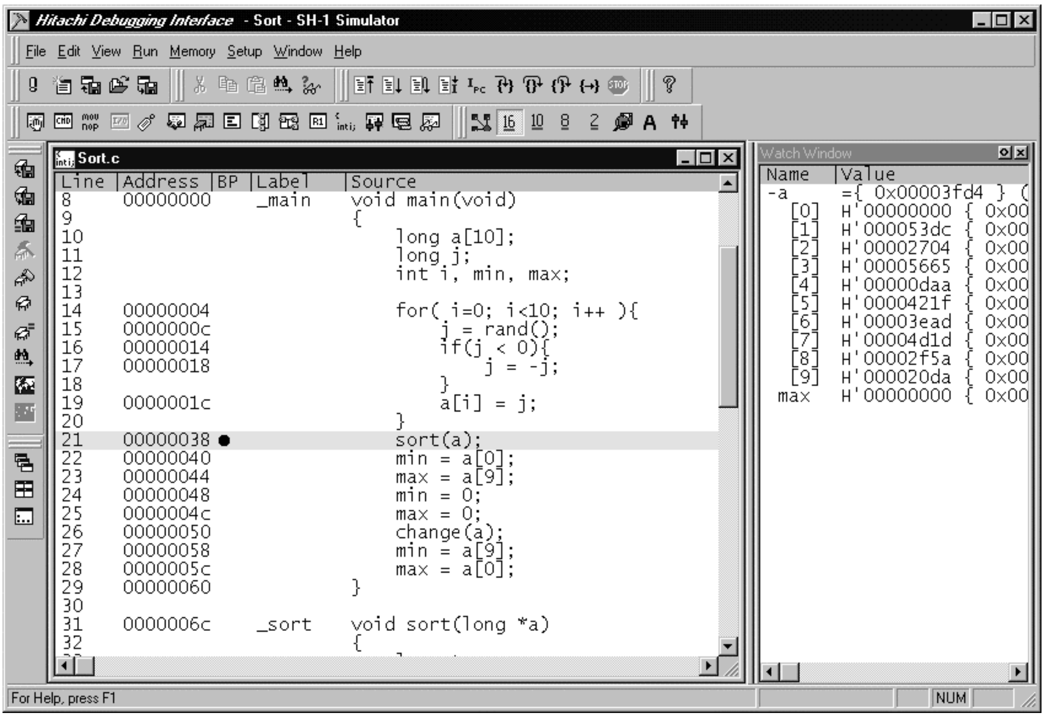


Figure 2.31 Source Window (Stepping)

Clicking the [◊] button in the title bar of the **Watch** window automatically places the **Watch** window next to the **Source** window as shown in Figure 2.31.

Executing [Step In]: The **[Step In]** command steps through the called function and stops at the first statement of the called function.

- To step through the `sort` function, choose **[Step In]** command from the **[Run]** menu, or click the **[Step In]** button in the toolbar.

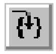


Figure 2.32 Step In Button

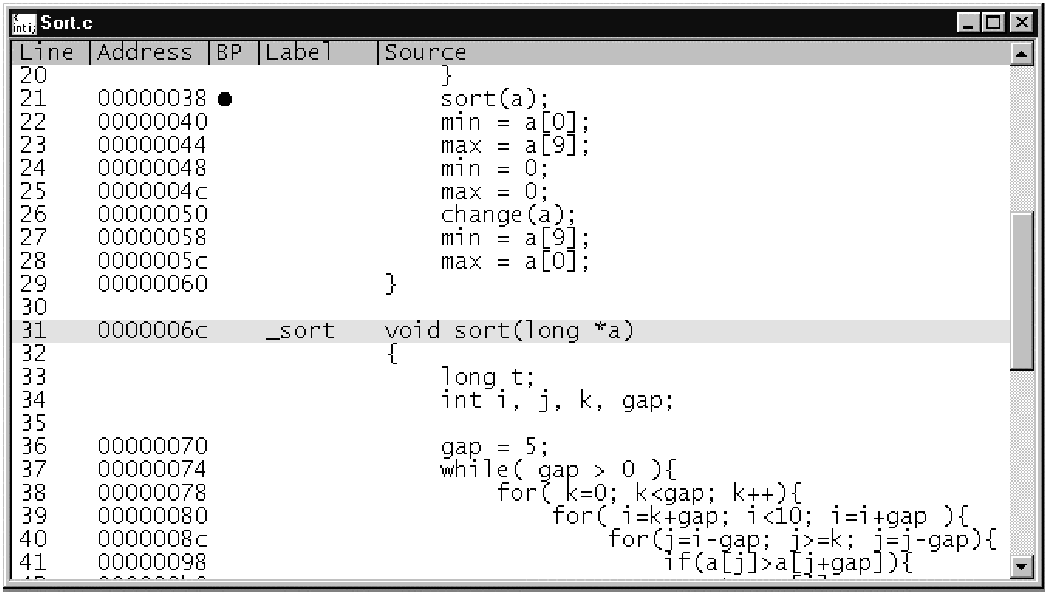


Figure 2.33 Source Window (Step In)

- The highlighted line moves to the first statement of the `sort` function in the **Source** window.

Executing [Step Out]: The **[Step Out]** command steps out of the called function and stops at the next statement in the calling program.

- To step out of the `sort` function, choose the **[Step Out]** command from the **[Run]** menu, or click the **[Step Out]** button in the toolbar.

Figure 2.34 Step Out Button

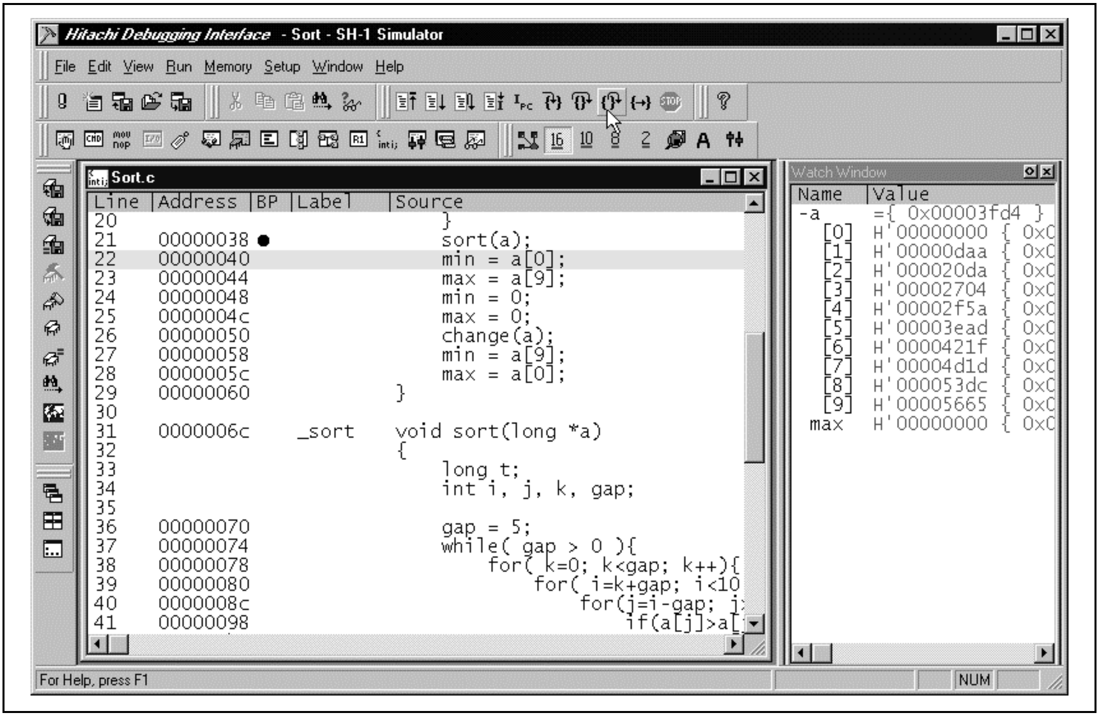


Figure 2.35 Source Window (Step Out)

- The data of variable "a" displayed in the **Watch** window is sorted in ascending order. In the **Watch** window, the values of the modified variables are shown in red.

- To execute two steps, use **[Step In]**

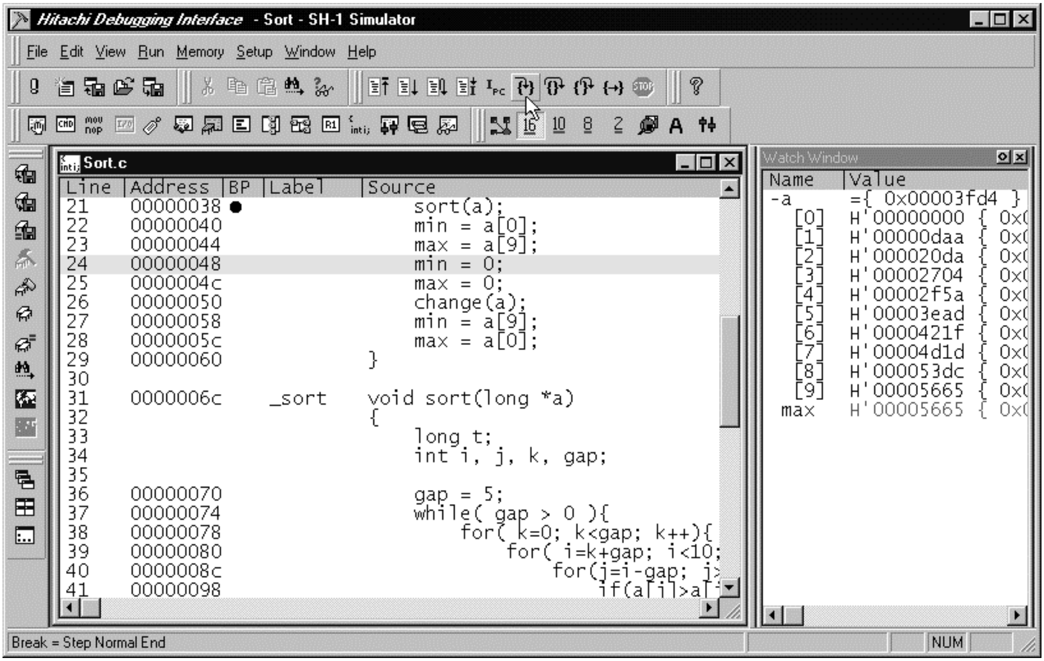


Figure 2.36 Source Window (Step Out –> Step In)

- The value of `max` displayed in the **Watch** window is modified to the maximum data value.

Executing [Step Over]: The **[Step Over]** command executes a function call as a single step and stops at the next statement in the main program.

- To demonstrate **[Step Over]** command, execute two steps to reach the `change` function statement.

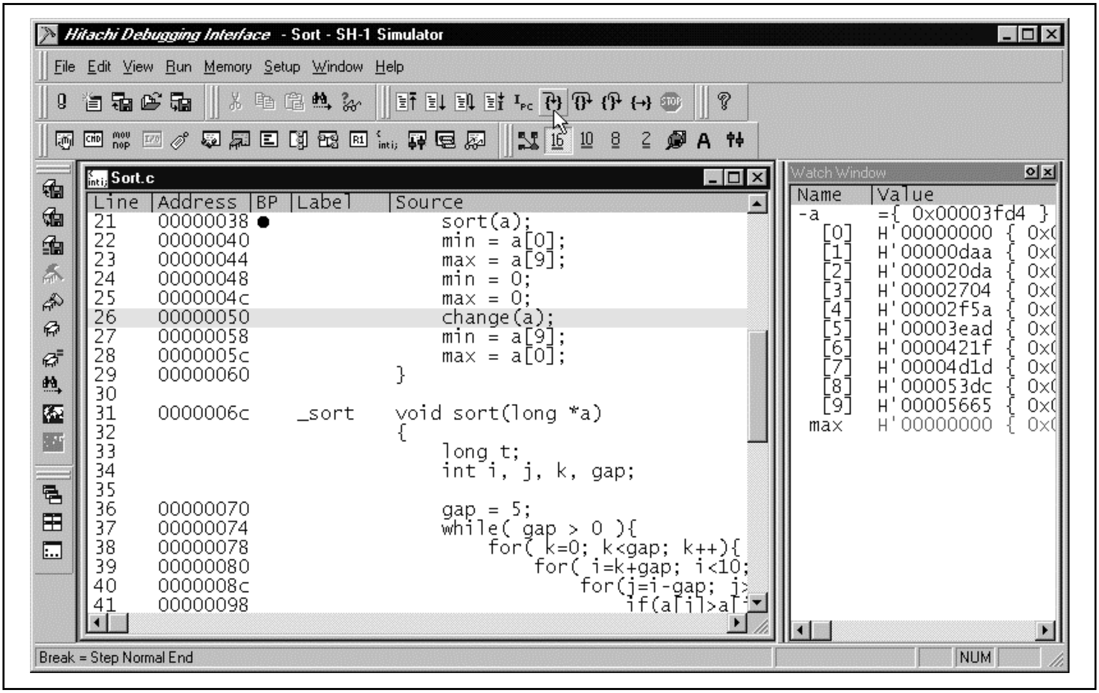


Figure 2.37 Source Window (Before Step Over Execution)

- To step through all statements in the `change` function at a time, choose **[Step Over]** command from the **[Run]** menu, or click the **[Step Over]** button in the toolbar.

Figure 2.38 Step Over Button

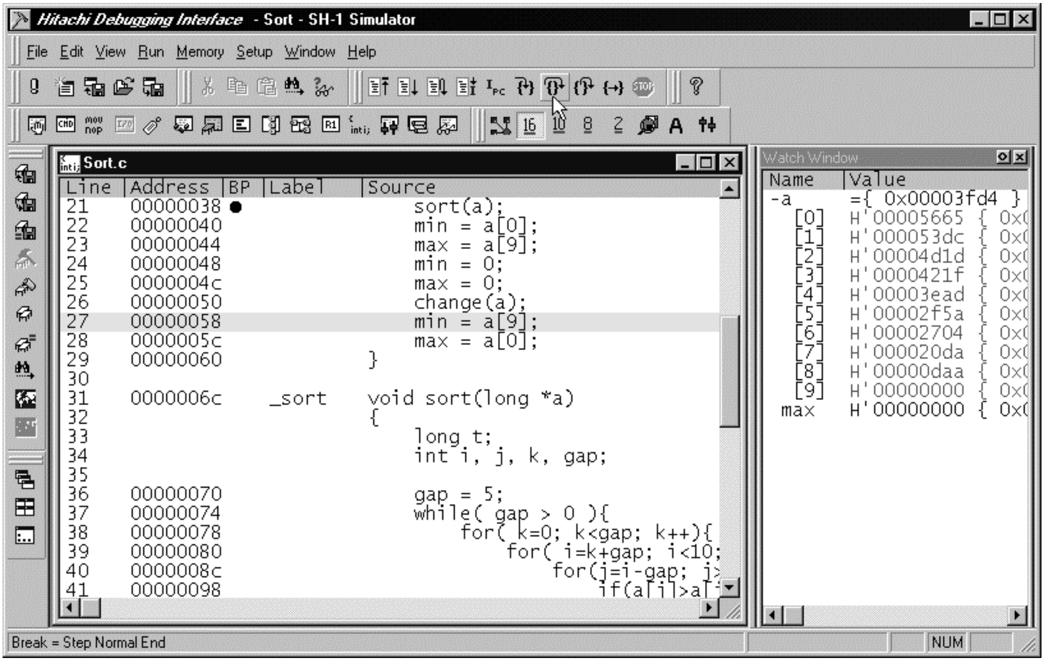


Figure 2.39 Source Window (Step Over)

When the last statement of the `change` function is executed, the data of variable "`a`", which is displayed in the **Watch** window, is sorted in descending order.

2.1.18 Displaying Local Variables

You can display local variables in a function using the **Locals** window. For example, we will examine the local variables in the `main` function, which declares five local variables: `a`, `j`, `i`, `min`, and `max`.

- Open the **Locals** window by choosing **[Locals]** from the **[View]** menu.
  Initially, the **Locals** window is empty because local variables have not yet been declared.
- Choose **[Step In]** from the **[Run]** menu to execute a single step.
  The **Locals** window will now show the local variables and their values.

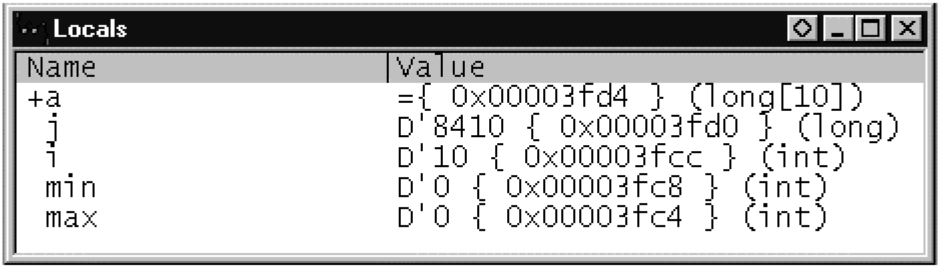


Figure 2.40 Locals Window

- Double-click the + symbol in front of array "a" in the **Locals** window to show the individual elements of array "a".
- Refer to the elements of array "a" before and after the execution of the sort function, and confirm that random data is sorted in ascending or descending order.

2.1.19 Reviewing the Performance Analysis

You can see the performance analysis results of functions in the **Performance Analysis** window.

- Choose **[Performance Analysis]** from the **[View]** menu.

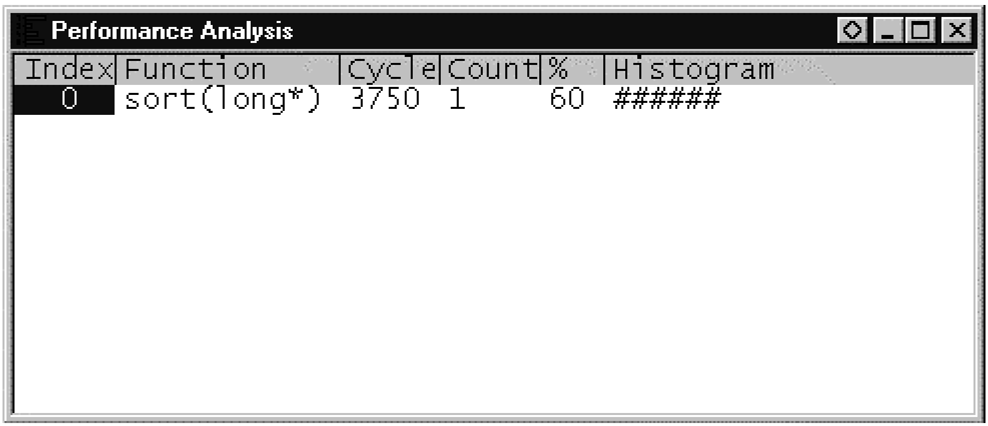


Figure 2.41 Performance Analysis Window (Viewing)

In this window, [**Cycle**] shows the total number of execution cycles required for the sort function, and [**%**] shows the ratio of execution cycle count required for the function to the execution cycle count required for the whole program.

The **Performance Analysis** window allows you to enable and disable the performance analysis, define new functions for performance analysis, and delete functions.

- Close the **Performance Analysis** window.

2.1.20 Saving the Session

Before exiting, it is a good practice to save your session, so that you can resume with the same conditions in your next debugging session.

- Choose **[Save Session]** from the **[File]** menu.
- Choose **[Exit]** from the **[File]** menu to exit from the HDI.

SuperH™ RISC engine
High-performance Embedded Workshop,
Hitachi Debugging Interface Tutorial

Publication Date: 1st Edition, December 2000
Published by: Electronic Devices Sales & Marketing Group
Semiconductor & Integrated Circuits
Hitachi, Ltd.
Edited by: Technical Documentation Group
Hitachi Kodaira Semiconductor Co., Ltd.