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.

RENESAS

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.

SuperH RISC engine C/C++ Compiler Package

APPLICATION NOTE: [IDE User's Guide]

Test Automation and Support Facilities

This document explains the macro-recording support facility and test support facility for High-performance Embedded Workshop.

Table of contents

1. Macro-recording Support Facility

The macro-recording support facility allows a series of operations related to applications[#1], builds[#2], and debugging[#3] on a High-performance Embedded Workshop (herein as *Renesas IDE*) system to be recorded to a file. Macros are used with the Renesas IDE command line.

#1 Such as changing projects, sessions, or configurations.

#2 Such as compiling and building. Support for this facility is dependent on the debugging platform.

#3 Such as downloading modules, changing memory values or registers, setting and removing software breaks, and executing programs.

1.1 Recording, executing, editing, and deleting macros

A file in which macros are saved is called a *macro file*. Macro files are text files with the extension `.hdc`, and are kept in the `Macro` directory within the Renesas IDE installation directory. `Default.hdc` is used by default. For example, if the Renesas IDE installation directory is `C:¥Program Files¥Renesas¥Hew`, `C:¥Program Files¥Renesas¥Hew¥Macro¥Default.hdc` is used by default.

To create a new macro file, from the **Tools** menu, choose **Macros**, and then click the **New** button in the **Macro** dialog box (Figure 1-1) displayed. When the **New** button is clicked, the **Add New Macro File** dialog box is displayed. In the **Add New Macro File** dialog box, enter the name of the new macro file to be created (within 64 alphanumeric characters and underscores), and click the **OK** button to add the name of the new macro file to the **Current macro file** dropdown list.

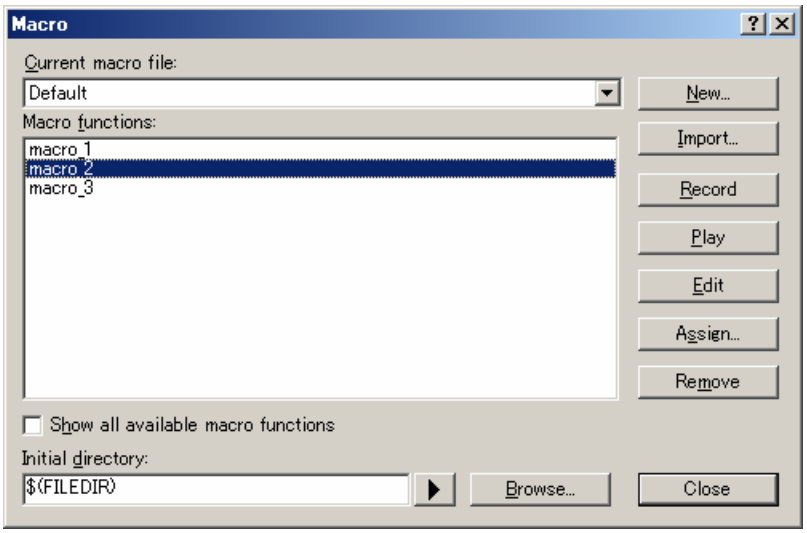

Figure 1-1

To switch the macro file used, select a macro from the **Current macro file** dropdown list in the **Macro** dialog box (Figure 1-1).

Initial directory indicates the base directory when a relative path is specified for the macro. This item needs to be set when a relative path is used to edit a macro directly.

1.1.1 Recording a macro

To record a macro, from the **Tools** menu, choose **Record Macro**. Or, please select **Macro Button** in the toolbar.

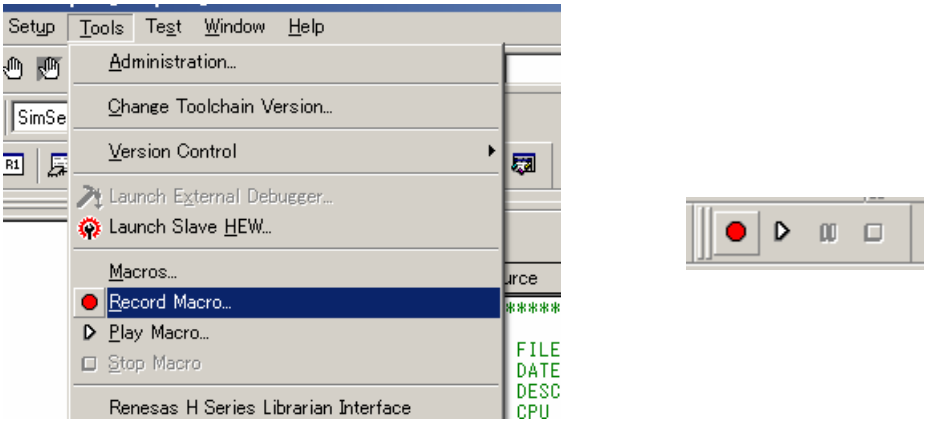

Figure 1-2

While the macro is recording, is displayed for the mouse cursor.

The contents recorded are reflected in real-time to the **Macro** tab in the output window (**Output** in the **View** menu). To stop macro recording, from the **Tools** menu, choose **Stop Macro**. When macro recording is stopped, the **Add New Macro Function** dialog box is displayed (Figure 1-3).

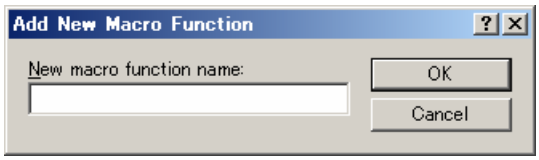

Figure 1-3

Enter the name of the new macro, and click the **OK** button. When recording is stopped, the mouse cursor returns to its previous state. Figure 1-4 shows an example of recording operations (1) to (3) below.

(1) Insert a breakpoint at line 31 in `c:¥workspace¥hew_test¥cmn_src¥hew_test.c` (from the **Edit** menu, choose **Toggle Breakpoints**)

(2) Perform a reset go (from the **Debug** menu, choose **Reset go**)

(3) Perform step over (from the **Debug** menu, choose **Step Over**)

The contents performed are written to the macro file being used, with the specified macro name.

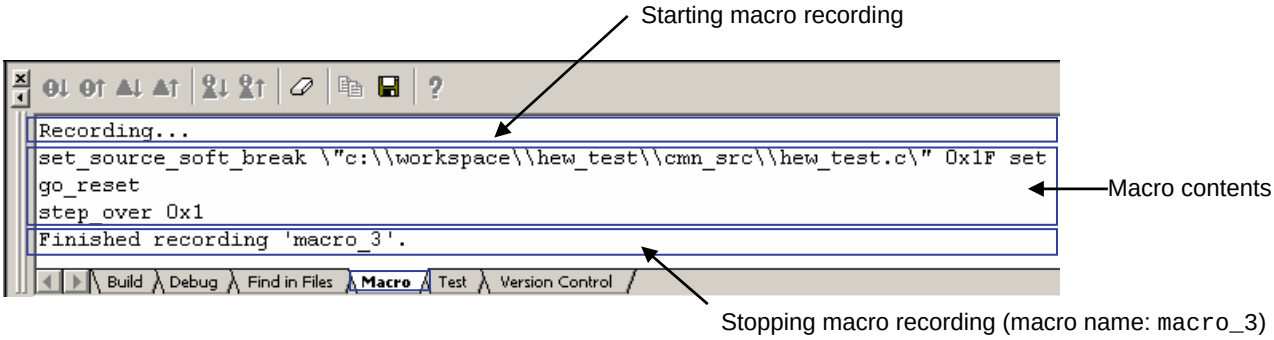

Figure 1-4

1.1.2 Editing a macro

To edit a macro, from the **Tools** menu, choose **Macros** to display the **Macro** dialog box (Figure 1-1).

When the macro to be edited is selected from **Macro functions** in the Figure 1-1 dialog box, the **Edit** button is enabled.

Click the **Edit** button to open the macro file in the editor window. The format for macros is the same as for the Renesas IDE command line. Edit the macro file directly as necessary.

For example, change the name of the source file for which breakpoints are to be set as shown in the following figure from an absolute path to a relative path.

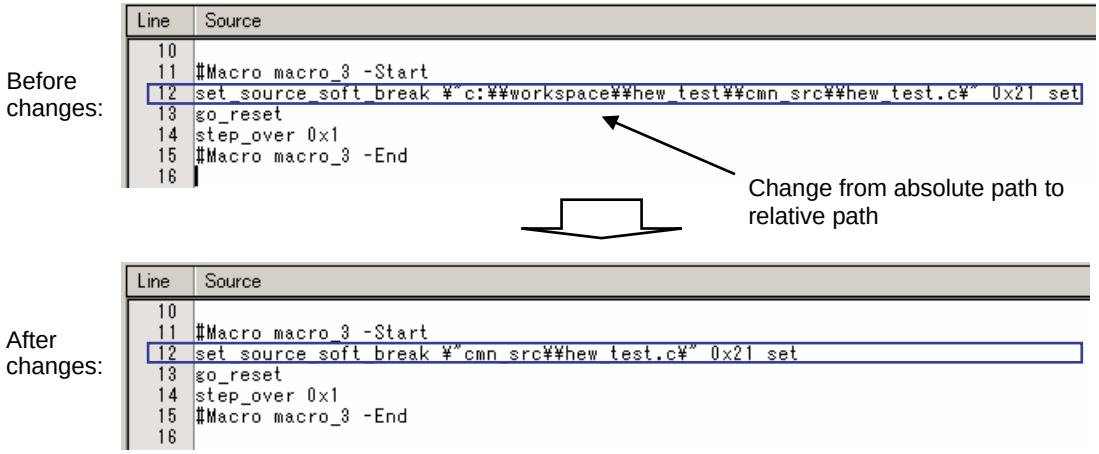

Figure 1-5

1.1.3 Executing a macro

To execute a macro, from the **Tools** menu, choose **Play Macro** to display the **Select Macro Function** dialog box, from which the macro to be executed can be selected. This dialog box only displays macros in **Current macro file** of the **Macro** dialog box (Figure 1-1).

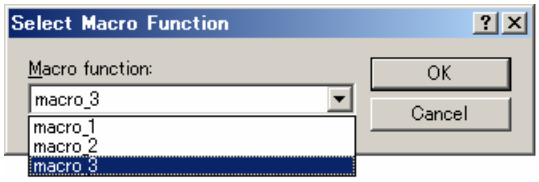

Figure 1-6

1.1.4 Importing an existing macro file

Macro files created on other computers can also be imported. To import a macro file, from the **Macro** dialog box (Figure 1-1), click the **Import** button, and then specify the file to be imported. The imported macro file is copied to the `Macro` directory in the Renesas IDE installation directory.

1.1.5 Deleting macros and macro files

To delete a macro, in the **Macro** dialog box (Figure 1-1), select the macro to be deleted from **Macro functions**, and then click the **Remove** button.

When deleting a macro file, delete the file directly from the `Macro` directory in the Renesas IDE installation directory.

1.2 Scope of the macro-recording support facility

Not all operations in Renesas IDE can be recorded to a macro file. The operations that can be recorded are classified as follows:

- Operations that can be recorded across all Renesas IDE instances
- Operations that can be recorded depending on the debugging platform

1.2.1 Operations that can be recorded across all Renesas IDE instances

The following operations can be recorded on all debugging platforms.

- Operations that can be recorded for menus, shortcut keys, or toolbar buttons

Table 1-1 Operations that can be recorded for menus, shortcut keys, or toolbar buttons

Menu	Menu Option		Recordable Renesas IDE Command-Line Commands
file	Open Workspace		OPEN_WORKSPACE
	Save Workspace		SAVE_WORKSPACE
	Close Workspace		CLOSE_WORKSPACE
	New Session		CHANGE_SESSION
	Import Session		CHANGE_SESSION
	Save Session		SAVE_SESSION
	Refresh Session		REFRESH_SESSION
	Download A New Module		See Dialog boxes.
	Recent Workspaces		OPEN_WORKSPACE
	Recent Downloaded Modules		FILE_LOAD
Edit	Toggle Breakpoint		SET_DISASSEMBLY_SOFT_BREAK SET_SOURCE_SOFT_BREAK
	Enable/Disable Breakpoint		STATE_DISASSEMBLY_SOFT_BREAK STATE_SOURCE_SOFT_BREAK
View	Workspace		See Windows.
	Disassembly		
	CPU	Registers	
		Memory	
		IO	
Project	Set Current Project		CHANGE_PROJECT
	Insert Project		CHANGE_PROJECT
	Edit Project Configuration *2		SAVE_SESSION
Build *2	Build File		BUILD_FILE
	Build		BUILD
	Build All		BUILD_ALL
	Build Multiple		See Dialog boxes.
	Clean Current Project		CLEAN
	Clean All Project		CLEAN
	Build Configurations		See Dialog boxes.
Debug	Debug Sessions		
	Reset CPU		RESET
	Go		GO
	Reset Go		GO_RESET
	Go to Cursor		GO_TILL
	Set PC to Cursor		REGISTER_SET *1
	Run		GO_TILL

Menu	Menu Option		Recordable Renesas IDE Command-Line Commands
	Step In		STEP
	Step Over		STEP_OVER
	Step Out		STEP_OUT
	Step		See Dialog boxes.
	Step Mode	Auto	STEP_MODE
		Assembly	STEP_MODE
		Source	STEP_MODE
	Halt Program		HALT
	Initialize		INITIALIZE
	Connect *2		CONNECT
	Disconnect *2		DISCONNECT
	Save Memory		FILE_SAVE
	Verify Memory *2		FILE_VERIFY
	Download Modules	<File name of the download module>	FILE_LOAD
		All Download Modules	FILE_LOAD_ALL
	Unload Modules	<File name of the download module>	FILE_UNLOAD
		All Download Modules	FILE_UNLOAD_ALL
Setup	Radix	Hex	RADIX
		Decimal	RADIX
		Oct	RADIX
		Bin	RADIX

*1. Refer to the help of an emulator or simulator.
*2. Support for this function depends on the debugging platform.

- Operations that can be recorded for windows

Table 1-2 Operations that can be recorded for windows

Window	Target	Function/Handling	Recordable Renesas IDE Command-Line Commands
Projects tab of the Workspace window	Pop-up menu of a workspace	Clean All Projects *2	CLEAN
	Pop-up menu of a project	Build *2	BUILD
		Build All *2	BUILD_ALL
		Clean Current Project *2	CLEAN
		Set as Current Project	CHANGE_PROJECT
	Pop-up menu of a project file	Build <File name> *2	BUILD_FILE
	Pop-up menu of a Download modules folder	Download all module	FILE_LOAD
		Download A New Module	See Dialog boxes.
	Pop-up menu of a download module	Download module	FILE_LOAD
		Download module (debug data only)	FILE_LOAD
		Unload module	FILE_UNLOAD
		Download A New Module	See Dialog boxes.
	<File name of the download module>	Download module by doubleclicking	FILE_LOAD
Editor and Disassembly (source mode)	Pop-up menu	Build <File name> *2	BUILD_FILE
		Toggle Breakpoint	SET_SOURCE_SOFT_BREAK
		Enable/Disable Breakpoint	STATE_SOURCE_SOFT_BREAK
		Go To Cursor	GO_TILL
		Set PC Here	REGISTER_SET *1
	S/W Breakpoints column	Insert/delete a breakpoint by double-clicking	SET_SOURCE_SOFT_BREAK
Editor and Disassembly (mixed/disassembly mode)	Pop-up menu	Go To Cursor	GO_TILL
		Set PC Here	REGISTER_SET *1
		Toggle Breakpoint	SET_DISASSEMBLY_SOFT_BREAK
		Enable/Disable Breakpoint	STATE_DISASSEMBLY_SOFT_BREAK
	S/W Breakpoints - ASM column	Insert/delete a breakpoint by double-clicking	SET_DISASSEMBLY_SOFT_BREAK
Register	Value	In-place edit	REGISTER_SET *1
	Flag register *2	Edit by clicking	REGISTER_SET *1
	Pop-up menu	Edit	REGISTER_SET *1
Memory	Value	In-place edit	MEMORY_EDIT
	Pop-up menu/toolbar button	Set	MEMORY_EDIT
		Fill	MEMORY_FILL
		Move	MEMORY_MOVE
		Compare *2	MEMORY_COMPARE
		Save	FILE_SAVE
		Load	FILE_LOAD
IO	Value	In-place edit	MEMORY_EDIT
		Open the edit dialog box by double-clicking	MEMORY_EDIT

*1. Refer to the help of an emulator or simulator.
*2. Support for this function depends on the debugging platform.

- Operations that can be recorded for dialog boxes

Table 1-3 Operations that can be recorded for dialog boxes

Dialog box	Function/Handling	Recordable Renesas IDE Command- Line Commands
Download Module	Enter Offset, Format, Filename, Access size, and Perform memory verify during download option	FILE_LOAD
Build Multiple *1	Click the Build button or the Build All button	BUILD_MULTIPLE
	Click the Clean button	CLEAN
Build Configurations	Change the Current configuration drop-down list	CHANGE_CONFIGURATION
Debug Sessions	Change the Current session dropdown list	CHANGE_SESSION
Step Program	Select the Step over calls checkbox	STEP_OVER
	Do not select the Step over calls checkbox	STEP

*1. Support for this function depends on the debugging platform.

1.2.2 Commands that can be recorded depending on debugging platform

The following operations may or may not be able to be performed depending on the debugging platform.

- Operations that can only be performed in SuperH RISC engine and H8 family emulators and simulator debuggers

Table 1-4 Operations that can be recorded for menus, shortcut keys, or toolbar buttons

Menu	Menu Option		Recordable Renesas IDE Command-Line Commands
View	Symbol	Label	See Windows.
		Watch	
		Locals	

Table 1-5 Operations that can be recorded for windows

Window	Target	Function/Handling		Recordable Renesas IDE Command-Line Commands
Label	Pop-up menu/toolbar button	Add		SYMBOL_ADD
		Delete		SYMBOL_CLEAR
		Delete All		SYMBOL_CLEAR
		Load		SYMBOL_ADD
	BP column	Insert/delete a breakpoint by double-clicking		SET_DISASSEMBLY_SOFT_BREAK
Watch	Pop-up menu/toolbar button	Auto Update		WATCH_AUTO_UPDATE
		Auto Update All		WATCH_AUTO_UPDATE
		Delete Auto Update		WATCH_AUTO_UPDATE
		Delete Auto Update All		WATCH_AUTO_UPDATE
		Record Update Value	Start Recording	WATCH_RECORD
			Stop Recording	WATCH_RECORD
		Add Watch		WATCH_ADD
		Edit Value		WATCH_EDIT
		Delete		WATCH_DELETE
		Delete All		WATCH_DELETE
		Radix	Hexadecimal	WATCH_RADIX
			Decimal	WATCH_RADIX
			Octal	WATCH_RADIX
			Binary	WATCH_RADIX
	+/-mark	Expand/collapse an array by clicking		WATCH_EXPAND
	Value	In-place edit}		WATCH_EDIT
Locals	Pop-up menu/toolbar button	Edit		WATCH_ADD WATCH_EDIT WATCH_DELETE
	Value	In-place edit		WATCH_ADD WATCH_EDIT WATCH_DELETE

- Operations that can only be performed in SuperH RISC engine and H8 family simulator debuggers

Table 1-6 Operations that can be recorded for menus, shortcut keys, or toolbar buttons

Menu	Menu Option		Recordable Renesas IDE Command-Line Commands
View	CPU	Simulated I/O	See Windows.
	Code	Coverage	
		Trace	
		Eventpoints	

Table 1-7 Operations that can be recorded for windows

Window	Target	Function/Handling		Recordable Renesas IDE Command-Line Commands
Simulated I/O	Pop-up menu/toolbar button	Erase All		SIMULATEDIO_CLEAR
Coverage	Pop-up menu/toolbar button	Enable All		COVERAGE
		Clear All		COVERAGE
		Add Range		COVERAGE_RANGE
		Edit Range		COVERAGE_RANGE
		Enable		COVERAGE
		Clear Data		COVERAGE
		Save Data		COVERAGE_SAVE
		Load Data		COVERAGE_LOAD
Trace	Pop-up menu/toolbar button	Acquisition		TRACE_ACQUISITION
Event - Software Break	Pop-up menu/toolbar button	Add	PC Breakpoint	BREAKPOINT
			Break Access	BREAK_ACCESS
			Break Data	BREAK_DATA
			Break Register	BREAK_REGISTER
			Break Sequence	BREAK_SEQUENCE
			Break Cycle	BREAK_CYCLE
		Edit	PC Breakpoint	BREAK_CLEAR BREAKPOINT
			Break Access	BREAK_CLEAR BREAK_ACCESS
			Break Data	BREAK_CLEAR BREAK_DATA
			Break Register	BREAK_CLEAR BREAK_REGISTER
			Break Sequence	BREAK_CLEAR BREAK_SEQUENCE
			Break Cycle	BREAK_CLEAR BREAK_CYCLE
		Enable		BREAK_ENABLE
		Disable		BREAK_ENABLE
		Delete		BREAK_CLEAR
		Delete All		BREAK_CLEAR

Window	Target	Function/Handling		Recordable Renesas IDE Command-Line Commands
Event - Software Event	Pop-up menu/toolbar button	Add	PC Breakpoint	BREAKPOINT
			Break Access	BREAK_ACCESS
			Break Data	BREAK_DATA
			Break Register	BREAK_REGISTER
			Break Sequence	BREAK_SEQUENCE
			Break Cycle	BREAK_CYCLE
		Edit	PC Breakpoint	BREAK_CLEAR BREAKPOINT
			Break Access	BREAK_CLEAR BREAK_ACCESS
			Break Data	BREAK_CLEAR BREAK_DATA
			Break Register	BREAK_CLEAR BREAK_REGISTER
			Break Sequence	BREAK_CLEAR BREAK_SEQUENCE
			Break Cycle	BREAK_CLEAR BREAK_CYCLE
		Enable		BREAK_ENABLE
		Disable		BREAK_ENABLE
		Delete		BREAK_CLEAR
		Delete All		BREAK_CLEAR

2. Test Support Facility

The test support facility automates user application regression testing.

In regression testing, the correct results for the execution for a series of given procedures (called a *test image*) are determined, and the same procedures are performed again after changing the program or changing build options to make sure that the calculated results are the same as the original correct results.

The test support facility consists of a facility for using macro files to automatically execute programs, and a facility to compare the saved results of program execution with the execution results or program re-execution. The combination of these two facilities is managed by units called *test suites*.

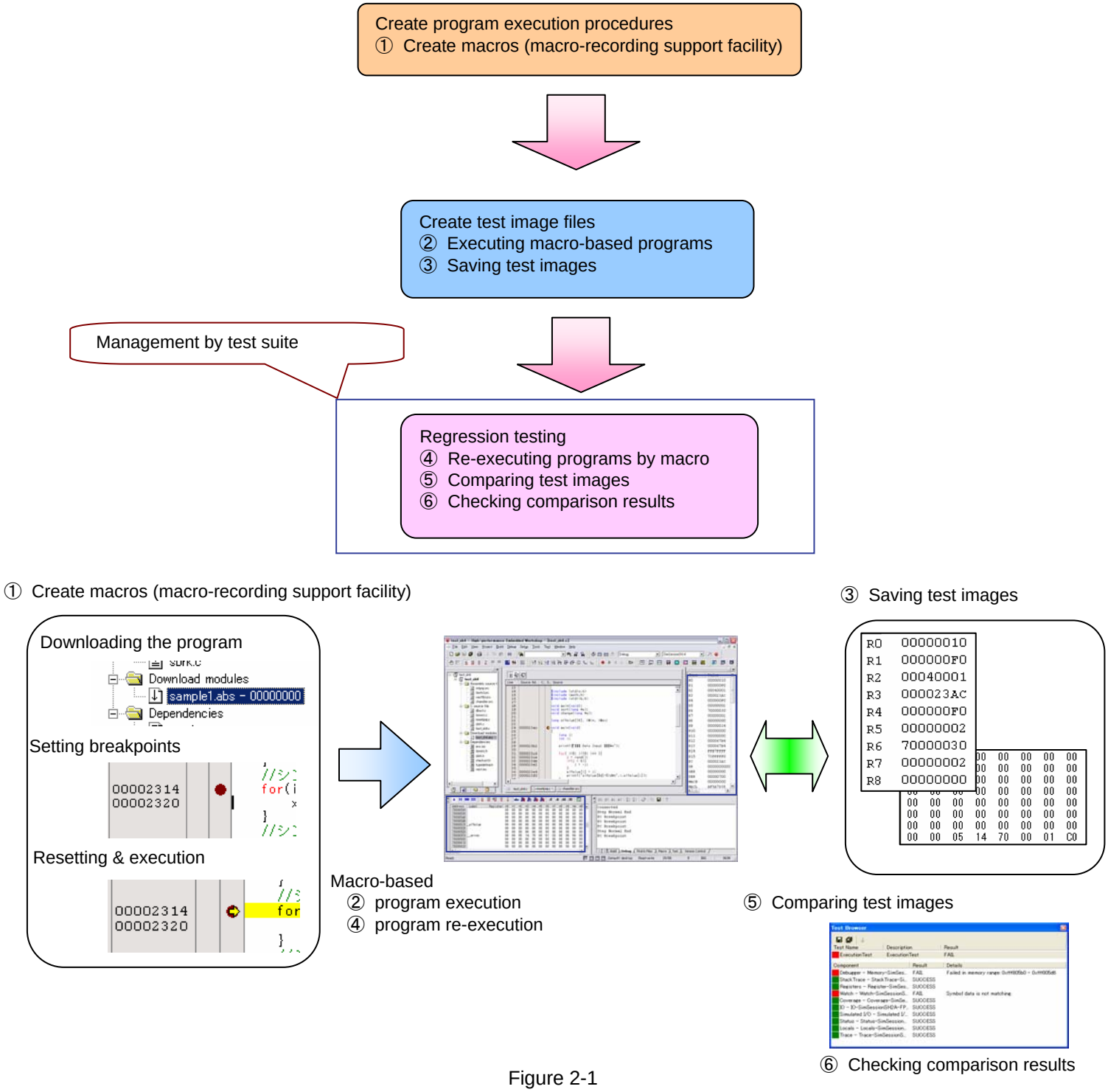

Figure 2-1

2.1 Test suite

The macros and test images needed by the test support facility are managed in *test suites*. A test suite consists of one or more *test cases*, each of which consisting of one or more macros and one test image.

In an application test, the variables of a program need to be checked when the program is in a given state. As such, a test consists of a macro to create the given state, and its corresponding test image.

The following shows an example test suite configuration.

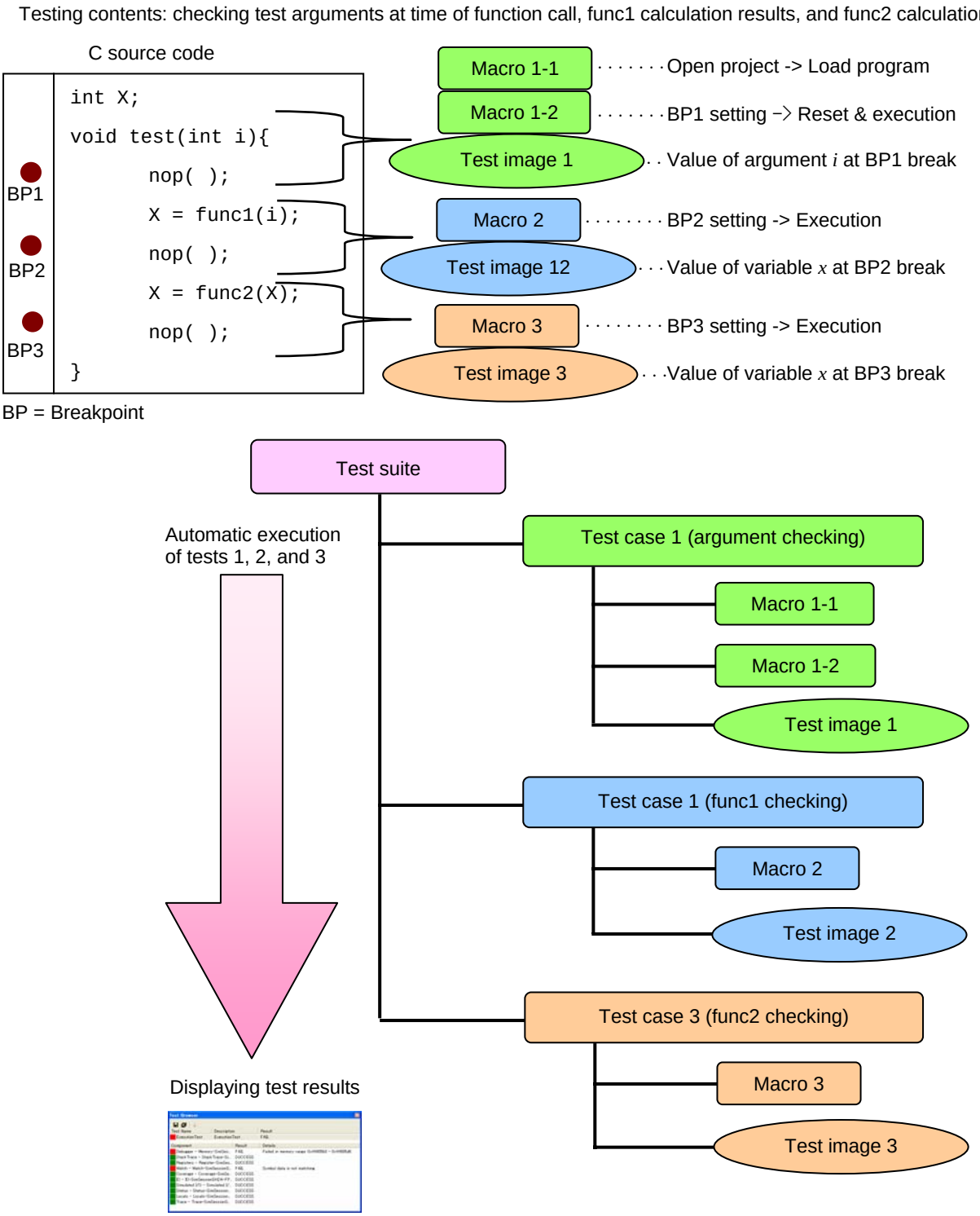

Figure 2-2

2.1.1 Creating a test suite

To create a test suite, from the **Test** menu, choose **Create New Test Suite**. The **Create New Test Suit**e dialog box is displayed.

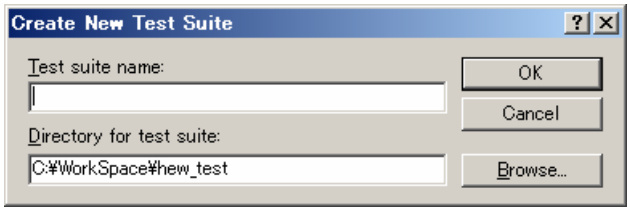

Figure 2-3

In the **Test suite name** field, enter the test suite name. The workspace directory is already displayed in **Directory for test suite** (if no workspace is open, the Renesas IDE installation directory is displayed). Edit this as necessary.

Click the **OK** button to create a test suite. When a test suite is created, it is added to the **Test** tab in the Workspace window. This tab can be used for easy access of test suite operations and test cases.

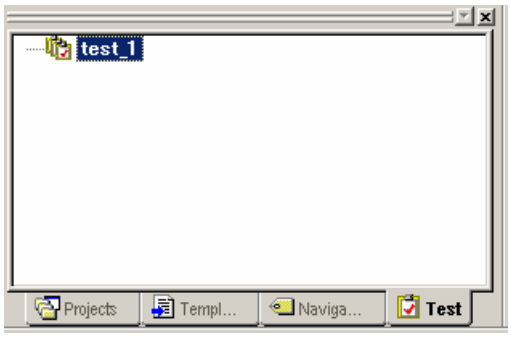

Figure 2-4

Test suite files are saved in the save destination, with the `.hts` extension appended.

2.1.2 Opening and closing a test suite

- Opening a test suite
  From the **Test** menu, choose **Open Test Suite**. In the **Open Test Suite** dialog box, select the test suite file, and then click the **Select** button. When a test suite is opened, the other items in the **Test** menu are enabled for selection, and the contents of the test suite are displayed in the **Test** tab of the Workspace window.
  The four most recently opened test suite files are displayed in the **Recent Test Suites** submenu of the **File** menu, and can be used to reopen a recently used test suite.

- Closing a test suite
  From the **Test** menu, choose **Close Test Suite**. The current test suite is closed, and all items are deleted from the **Test** tab of the Workspace window.
  Test suites can also be closed from the pop-up menu of the **Test** tab in the Workspace window.

2.1.3 Editing a test suite

Multiple *test cases* can be registered in a test suite. The following explains how to add a test case to a test suite, and edit the test cases registered in a test suite.

Creating a test case

A *test case* consists of procedures (macros) for executing a program, and a test image (as explained in *2.2*).To create a test case, from the **Test** menu, choose **Edit Test Suite** to display the **Modify Test Suite** dialog box.

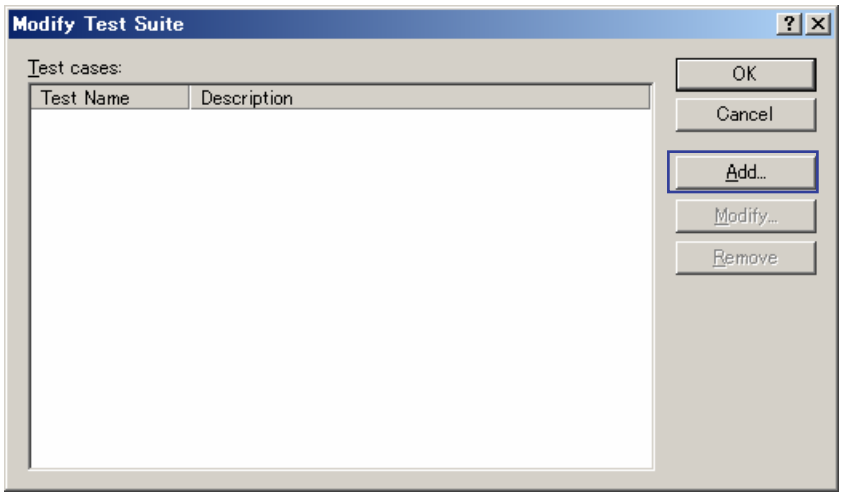

Figure 2-5

In the **Modify Test Suite** dialog box, click the **Add** button to display the **Add New Test** dialog box (Figure 2-6), and then enter the test name and test description. Do not specify any space characters in the test name. For the test description, enter a detailed description that makes the test contents easy to grasp.

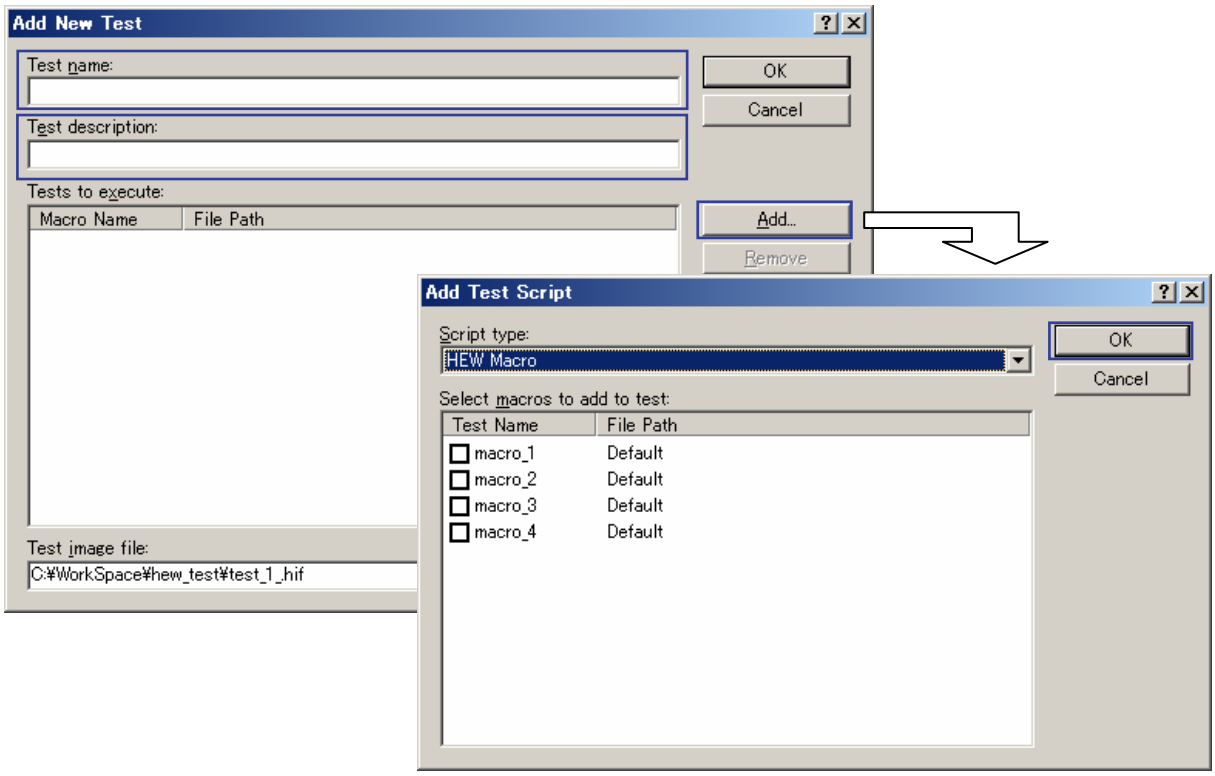

Figure 2-6

The macros used in a test are registered by clicking the **Add** button in the **Add New Test** dialog box. Click the **Add** button to display the **Add Test Script** dialog box (Figure 2-6). From the **Script Type** dropdown list, choose **HEW Macro** to display all of the registered macros in the **Select macros to add to test** list. Select the checkboxes next to the names of the macros to be used. Once a macro is added, the order of the registered macros can be changed in **Tests to execute**, in the **Add New Test** dialog box. When a test is executed, its macros are processed from the top down. Use the **Move to top**, **Move up**, **Move down**, and **Move to bottom** buttons to change the macro order (Figure 2-7).

Supplementation: The script file of the user definition can be used by selecting "Tcl command line batch file" by the **Script Type** drop down list.

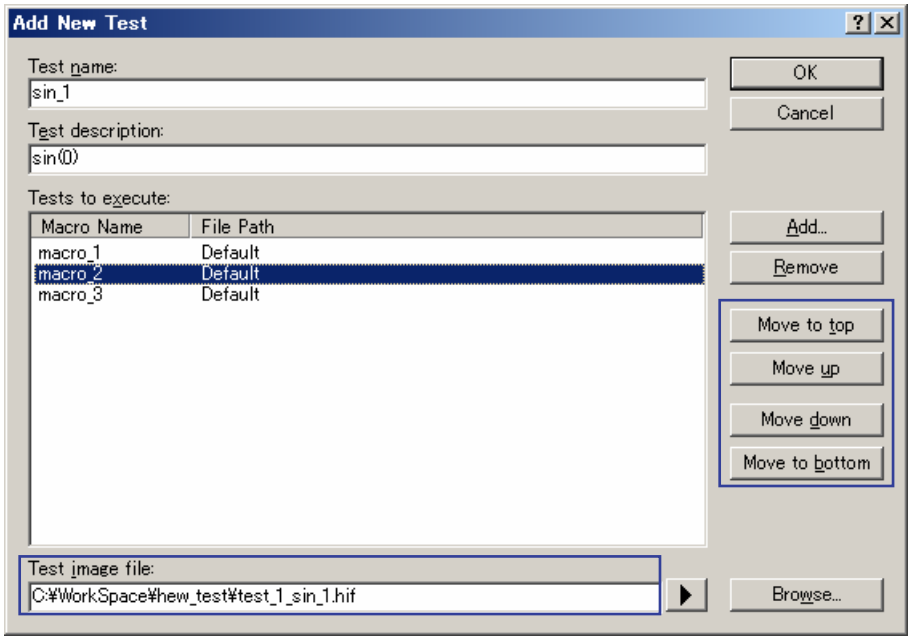

Figure 2-7

The test image file used for a test can be specified in the **Test image file** field in the **Add New Test** dialog box. The default file name is *test-suite-name+test-name*`.hif`, in the same directory as the test suite. Click the **OK** button in the **Modify Test Suite** dialog box (Figure 2-5) to create the empty test image file. For details about test image files, see *2.2 Test image file*.

Editing or deleting a test case

To edit or delete an existing test case, in the **Modify Test Suite** dialog box (Figure 2-5), select the test case to be changed, and click the **Modify** or **Remove** button.

When an existing test is edited, the **Modify Test** dialog box is displayed, with the same contents as the **Add New Test** dialog box (Figure 2-7). Change the test case with the same information used for creating one.

2.2 Test image file

Regression testing requires the maintenance of correct execution results of a program. These results are called a *test image*, and the file in which a test image is saved is called a *test image file*. One test image file is created per test case. The extension for test image files is `.hif`.

2.2.1 Editing a test image file

The target items for a test image are set in the test image file. In the **Test** tab of the Workspace window, right-click the test case for which a test image file is to be created. In the displayed pop-up menu (Figure 2-8), choose **Edit Test Image File** to display the dialog box for editing the test image file(Figure 2 9). Once the test image file has been edited, the value of each item set by the test image file edit is saved in the test image file.

The test images that can be obtained depend on the platform. For details, see *High-performance Embedded Workshop V.4.02 User's Manual - 16.6 Facilities that can be saved as test image data in a test image file*.

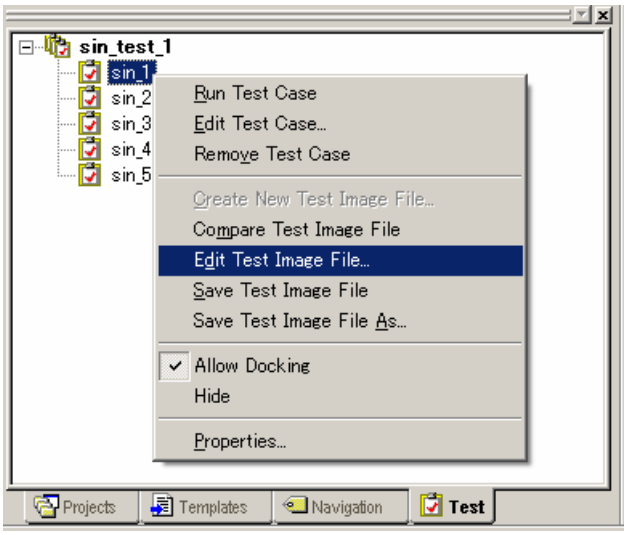

Figure 2-8

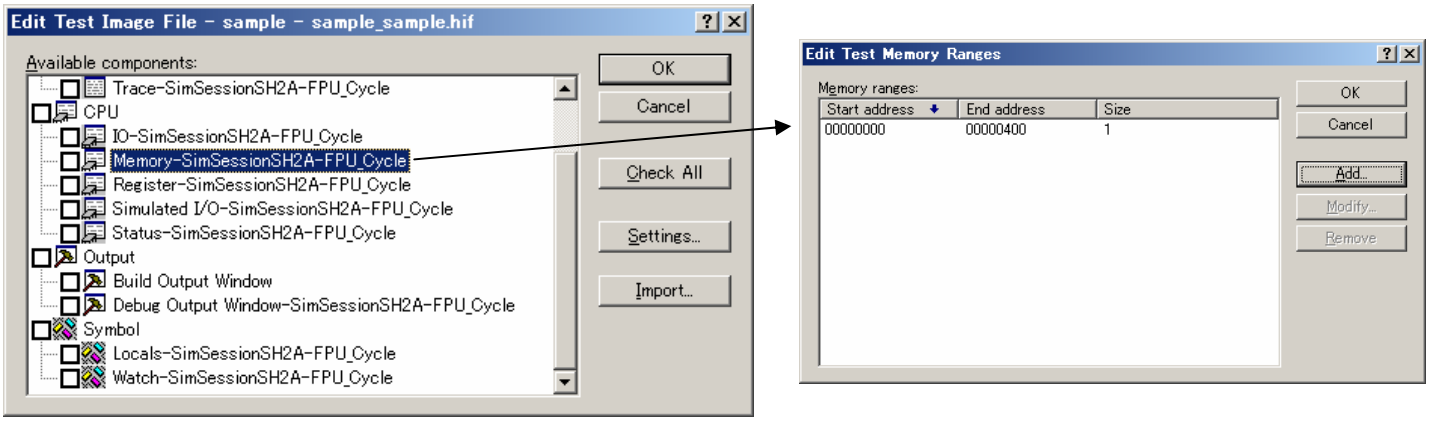

Figure 2-9

2.2.2 Saving a test image file

The item selected in **Edit Test Image File** can be saved in the test image file from **Save Test Image File**.

In the **Test** tab of the Workspace window, right-click the test case of the test image to be saved, and in the displayed pop-up menu, choose **Save Test Image File**. The value of each item when **Save Test Image File** is chosen is saved in the test image file.

2.3 Executing a test

To execute a test, from the **Test** menu, choose **Run Tests** to display the **Run Tests** dialog box.

The test cases registered in the test suite are displayed in the **Test cases** list. Select the checkbox for the test cases to be executed. When a test case is selected, the **Move up** and **Move down** buttons can be clicked to change the execution order of the test case.

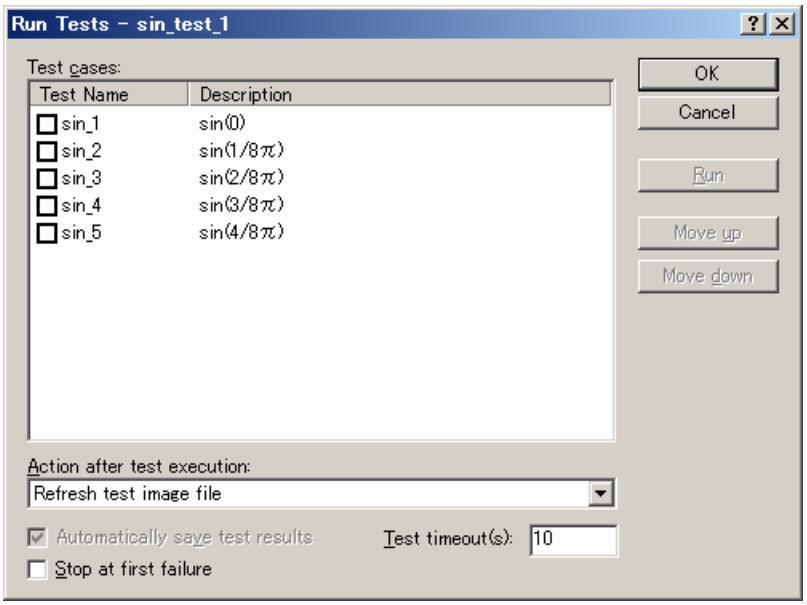

Figure 2-10

- The **Action after test execution** dropdown list has the following two options:
  - The **Compare system against saved test image file** option compares the test image file (`#.HIF`) attached to the related test case to the current system, for normal operation. These results are added to the test browser, and provided as information about test success/failure and the cause of failure. Specify this to compare the test execution results with the previous results.
  - The **Refresh test image file** option updates the test image file (`#.HIF`) attached to the related test case after it is executed. Specify this for the first test (when there is no previous data).
- The **Automatically save test results** checkbox can be selected to automatically save the execution results from each test in a text file, the location of which is the same as the test suite. The file name is the name of the current test suite with the time of test execution appended.
- The **Stop at first failure** checkbox can be selected to stop test execution when the first error occurs. This prevents other tests from being executed over and over if subsequent tests fail because of the first test.
- If a test takes longer than the number of seconds specified for **Test timeout**, the test is cancelled and judged to have failed.
  When performing tests with a long processing time, set a slightly longer timeout time.

2.4 Checking test results

Once testing is completed, the test results are automatically displayed in the test results browser.

2.4.1 Contents displayed in the test results browser

The test results browser can be used to check test results.

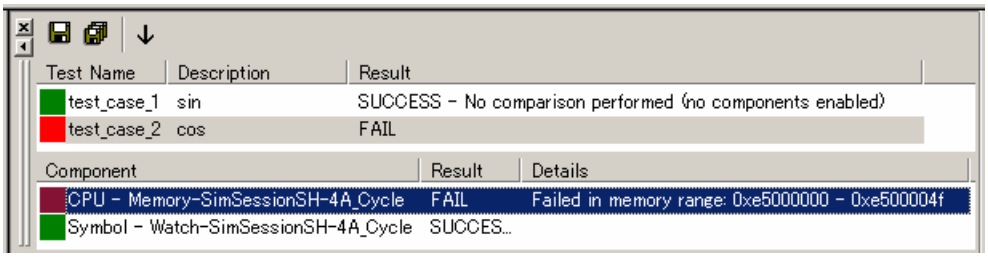

Figure 2-11

The execution results from each currently executed test case are displayed in the top pane of the test results browser. When all testing is executed normally, and there is no difference between the current test results and the test image, a green icon is displayed indicating that the test was successful. If there is a difference between the results, a red icon is displayed indicating that the test failed.

When a test case is selected in the top pane, the execution results for each test item set in the test image of the test case are displayed in the bottom pane. As with the top pane, if there is no difference between the current test results and the test image, a green icon is displayed. If there is a difference between the results, a red icon is displayed. Contents for test items for which results were different are displayed in the **Details** column.

Test items displayed with a red icon can be double-clicked in the bottom pane to check the data that does not match (Figure 2-12).

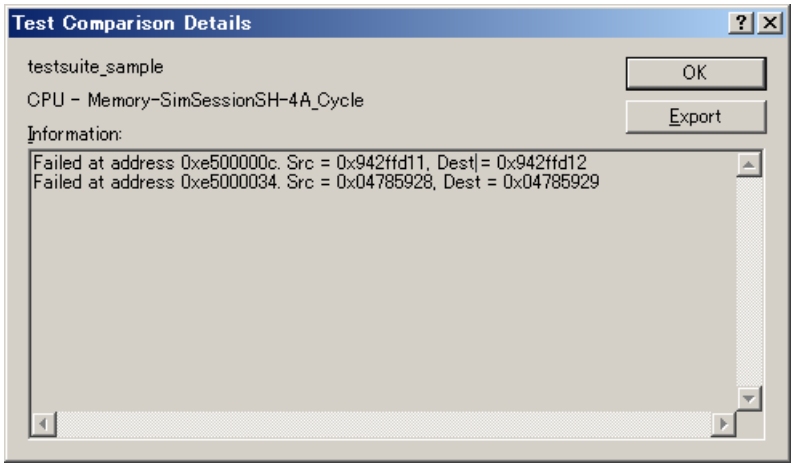

Figure 2-12

2.5 Comparing test results

The test results browser is automatically displayed after test execution, but can also be used at other times to check test results. From the **Test** menu, choose **Compare Test Image File** to display the **Compare** dialog box.

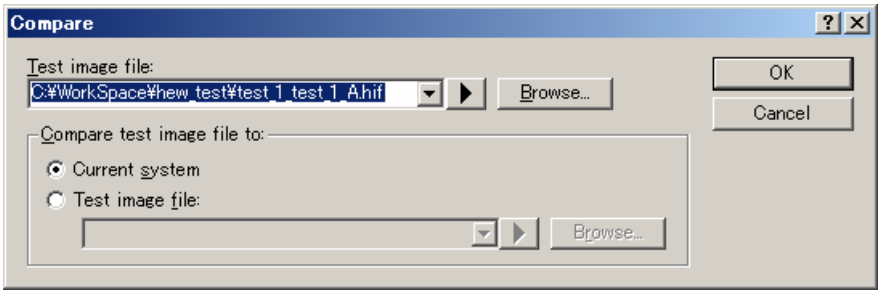

Figure 2-13

For the **Test image file** field, enter the test image file from which comparison will be performed.

Then, in **Compare test image file to**, specify the target to be compared with that specified for **Test image file**. The current system or another previously saved test image file can be specified for **Compare test image file to**. The **Current system** option is useful for manual test execution and comparisons checking the current test image against a previously saved test image file.

3. Tutorial

The test support facility is useful for checking for degradations. This chapter uses a concrete example to explain how to use the test support facility. The sample project created in this chapter is provided from the download site with this document. Place the sample project in `C:¥WorkSpace¥sample`.

The test suite is created as follows:

(1) Creating a new macro file
(2) Recording a new testing macro
(3) Editing the macro
(4) Creating a test suite
(5) Registering a test using the generated macro with the test suite
(6) Creating a test image for each created test

The test results are checked as follows:

(a) The sample program is checked for proper execution.
(b) The sample program is changed, and checked for degradation.
(c) Any malfunctions detected in (b) are corrected, and checked for proper correction.

3.1 About the sample project

The sample project used for this explanation is created on the following environment. Keep in mind that the sample project cannot be opened on an environment less recent than this one.

- C/C++ compiler package for the Renesas SuperH family ····· V.9.01Release00
- High-performance Embedded Workshop ····· V. 4.02.00
- Toolchain ····· V. 9.1.0.0

The sample project adds functions to the files generated by the Renesas IDE project creation facility.

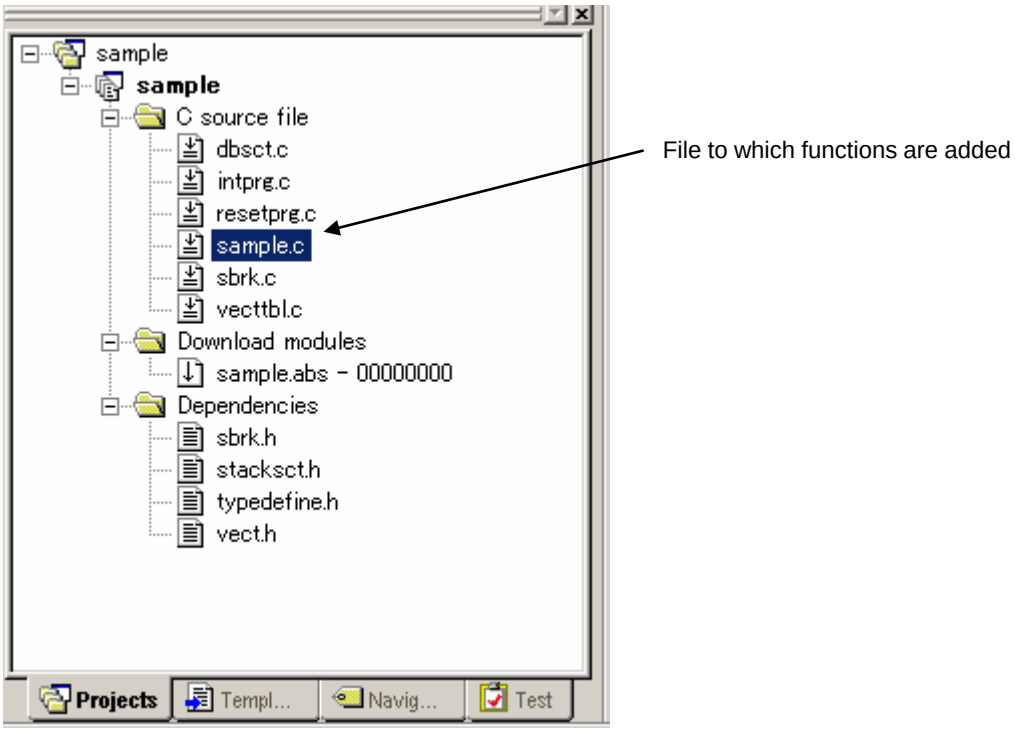

Figure 3-1

The sample program (Figure 3-2) contains a definition of the `func()` function to be tested. The `func()` function substitutes `0` for the *x* and *y* variables when the argument is not `0`, and leaves the values for the *x* and *y* variables the same when the argument is `0`. The two test items for this program are (i) and (ii) below. For cases (a) to (c), each of the (i) and (ii) items is tested.

(i) Check that the values of the *x* and *y* variables do not change when the argument of the `func()` function is `0`.

(ii) Check that the values of the *x* and *y* variables change to `0` when the argument of the `func()` function is not `0`.

Line	Source
28	`#include <machine.h>`
29	
30	`int x,y;`
31	
32	`void func(int a);`
33	
34	`void main(void)`
35	`{`
36	`  x = y = 1;`
37	`  nop();`
38	
39	`  func(0);`
40	`  nop();`
41	
42	`  func(1);`
43	`  nop();`
44	`}`
45	
46	`void func(int a)`
47	`{`
48	`  if (a)`
49	`  x = 0;`
50	`  if (a)`
51	`  y = 0;`
52	`}`

Figure 3-2

3.2 Preparing for testing

3.2.1 Generating a macro file

First, create a new macro file. From the **Tools** menu, choose **Macros**, and in the displayed **Macro** dialog box, click the **New** button. In the **Add New Macro File** dialog box, enter the name of the new macro file (such as `sample_macro`), and click the **OK** button.

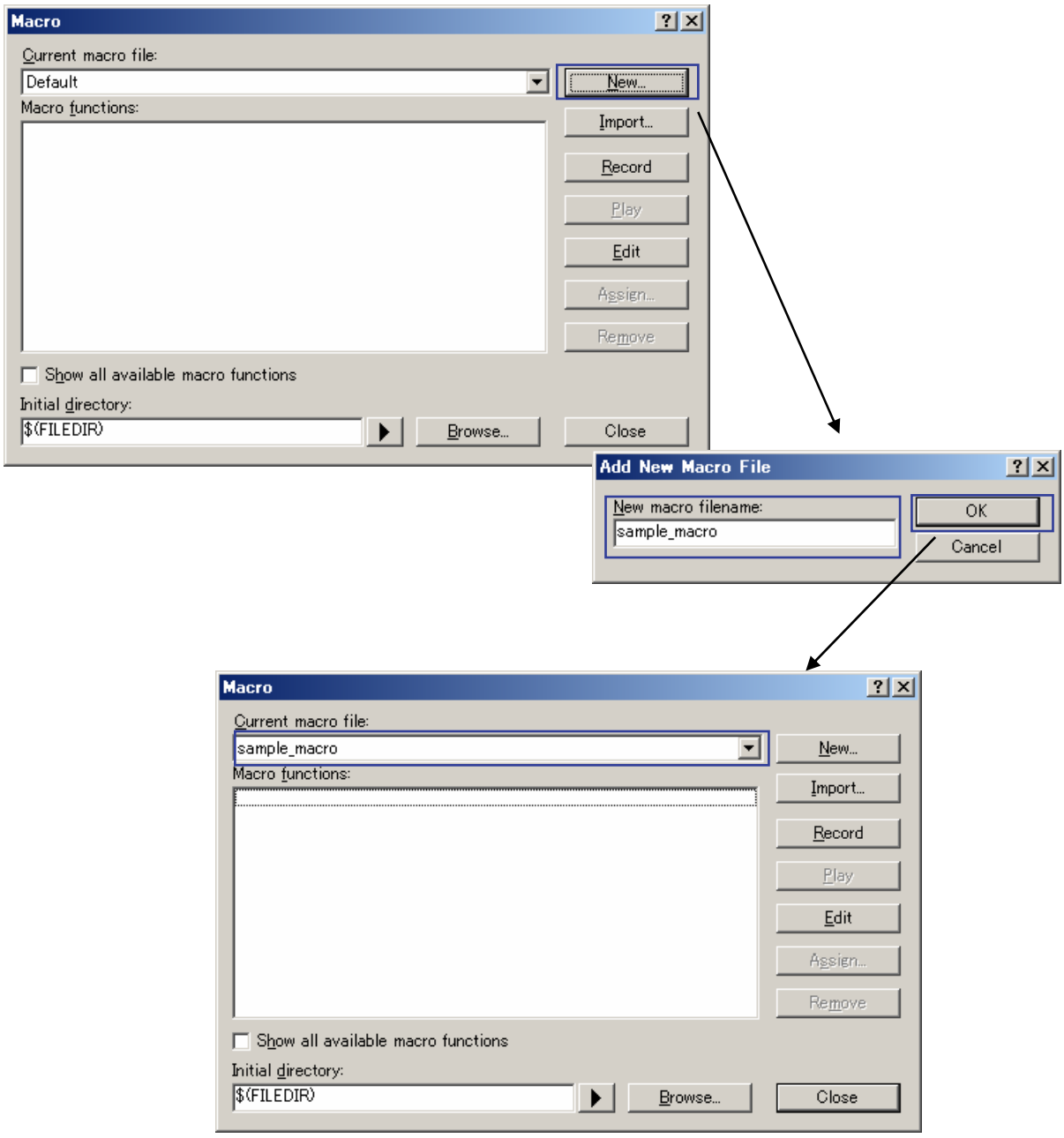

Figure 3-3

3.2.2 Recording a macro

Record a macro to perform the operations for automatically executing the program for testing. For details about recording macros, see *1.1.1 Recording a macro*. The following will create two macros:

(i) Test case 1: `func(0)`

This checks that the values of the *x* and *y* variables do not change when the argument of the `func()` function is `0`.

Record the following steps (1) to (3) as the macro name `sample_break_1`:

(1) Perform build (choose **Build** from the **Build** menu)
(2) In the editor window, insert a breakpoint in the 40th line of the program source (choose **Toggle Breakpoints** from the **Edit** menu)
(3) Execute the program (choose **Reset go** from the **Debug** menu)

Line	Source A...	S...	Source
28			#include <machine.h>
29			
30			int x,y;
31			
32			void func(int a);
33			
34	00001000		void main(void)
35			{
36	00001002		x = y = 1;
37	0000100C		nop();
38			
39	0000100E		func(0);
40	00001012		nop();
41			
42	00001014		func(1);
43	00001018		nop();
44	0000101C		}
45			
46	0000101E		void func(int a)
47			{
48	0000101E		if (a)
49	00001022		x = 0;
50			if (a)
51	00001026		y = 0;
52	0000102C		}
53			

Figure 3-4

(ii) Test case 2: `func(1)`

This checks that the values of the *x* and *y* variables change to `0` when the argument of the `func()` function is not `0`.

Record the following steps (1) and (2) as the macro name `sample_break_2`:

(1) In the editor window, insert a breakpoint in the 43rd line of the program source (choose **Toggle Breakpoints** from the **Edit** menu)

(2) Execute the program (choose **Go** from the **Debug** menu)

```
Line  Source A...  S...  Source
 28                      #include <machine.h>
 29
 30                      int x,y;
 31
 32                      void func(int a);
 33
 34   00001000           void main(void)
 35                      {
 36   00001002               x = y = 1;
 37   0000100C               nop();
 38
 39   0000100E               func(0);
 40   00001012   ●           nop();
 41
 42   00001014               func(1);
 43   00001018   ➡           nop();
 44   0000101C           }
 45
 46   0000101E           void func(int a)
 47                      {
 48   0000101E               if (a)
 49   00001022                   x = 0;
 50                          if (a)
 51   00001026                   y = 0;
 52   0000102C           }
 53
```

Figure 3-5

The above two macros are recorded to the macro file `sample_macro.hdc` in the `Macro` directory in the Renesas IDE installation directory. The contents recorded are as follows.

```
Line  Source
  1
  2  #Macro sample_break_1 -Start
  3  build wait
  4  set_source_soft_break ¥"c:¥¥workspace¥¥sample¥¥sample¥¥sample.c¥" 0x28 set
  5  go_reset
  6  #Macro sample_break_1 -End
  7
  8  #Macro sample_break_2 -Start
  9  set_source_soft_break ¥"c:¥¥workspace¥¥sample¥¥sample¥¥sample.c¥" 0x2B set
 10  go
 11  #Macro sample_break_2 -End
 12
```

Figure 3-6

3.2.3 Creating a test suite

Create a test suite, and then add test cases to it.

To create a test suite, from the **Test** menu, choose **Create New Test Suite** to display the **Create New Test Suite** dialog box. In the **Create New Test Suite** dialog box, enter the name of the created test suite (such as `testsuite_sample`) in the **Test suite name** field, and click the **OK** button.

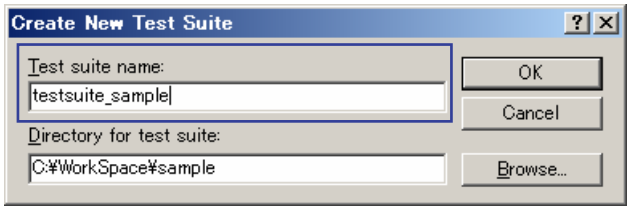

Figure 3-7

The `C:¥WorkSpace¥sample¥testsuite_sample.hts` file is created, and the **testsuite_sample** test suite icon is displayed in the **Test** tab of the Workspace window (**Workspace** in the **View** menu).

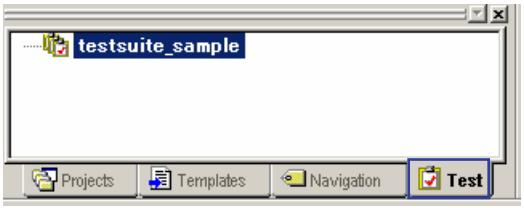

Figure 3-8

From the **Test** menu, choose **Edit Test Suite** to display the **Modify Test Suite** dialog box.

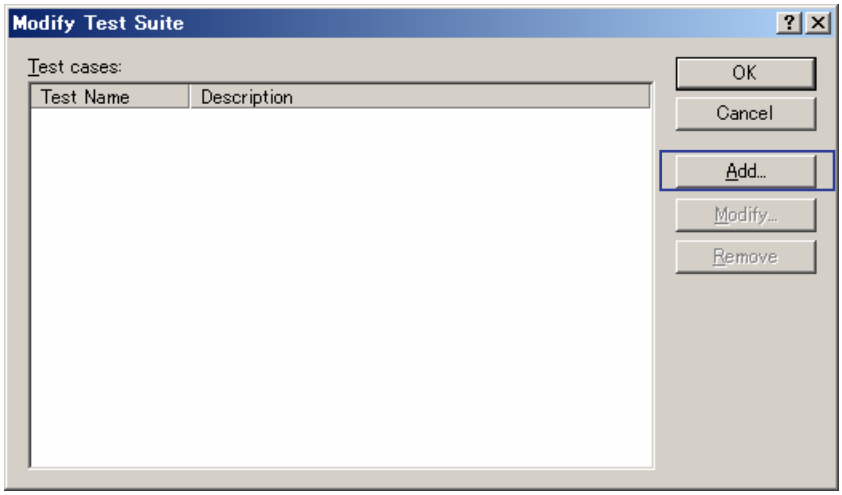

Figure 3-9

In the **Modify Test Suite** dialog box, click the **Add** button to display the **Add New Test** dialog box.

First, add `test_case_1` for the test case in *(i) Test case 1: func(0)*. Enter the **Test name** field and **Test description** field. Then, click the **Add** button to display the **Add Test Script** dialog box.

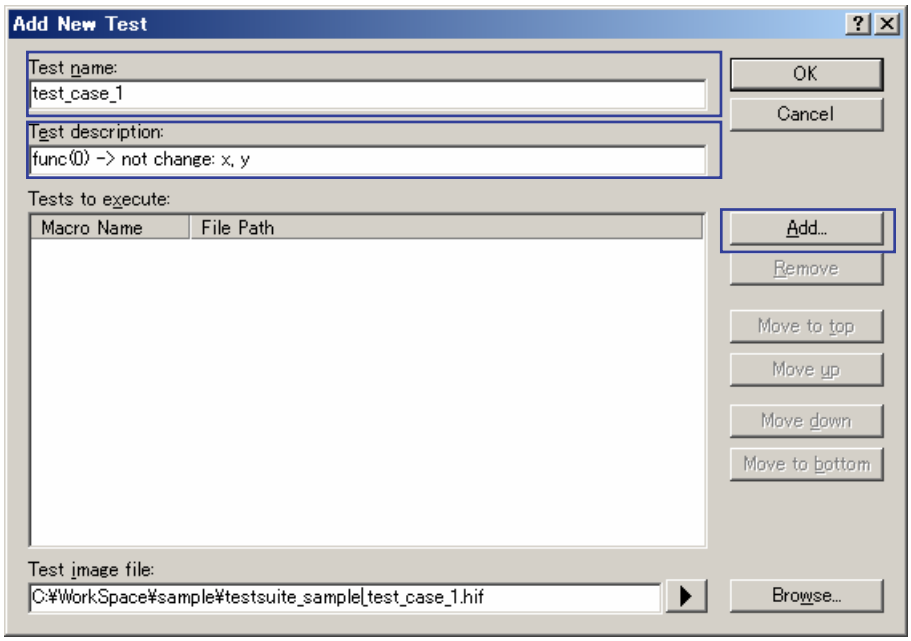

Figure 3-10

Then, register the previously created macro (`sample_break_1`). In the **Select macros to add to test** list, select the **sample_break_1** checkbox, and click the **OK** button.

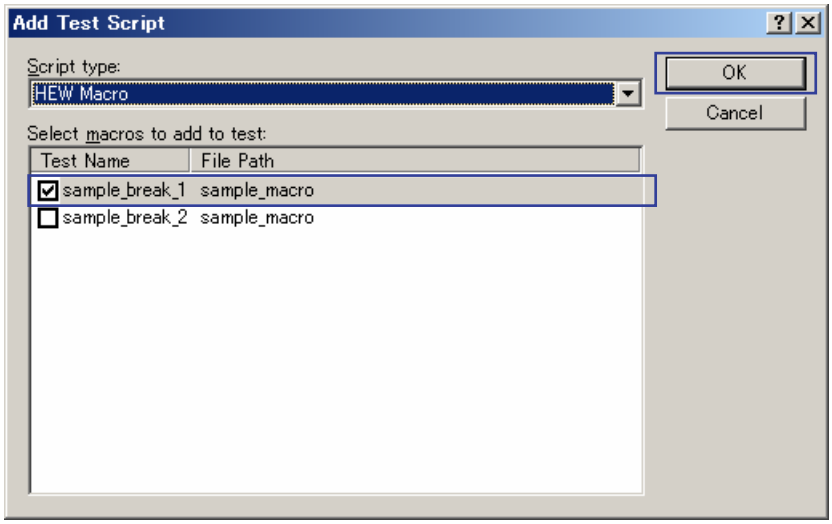

Figure 3-11

The **Add New Test** dialog box is displayed again. Click the **OK** button.

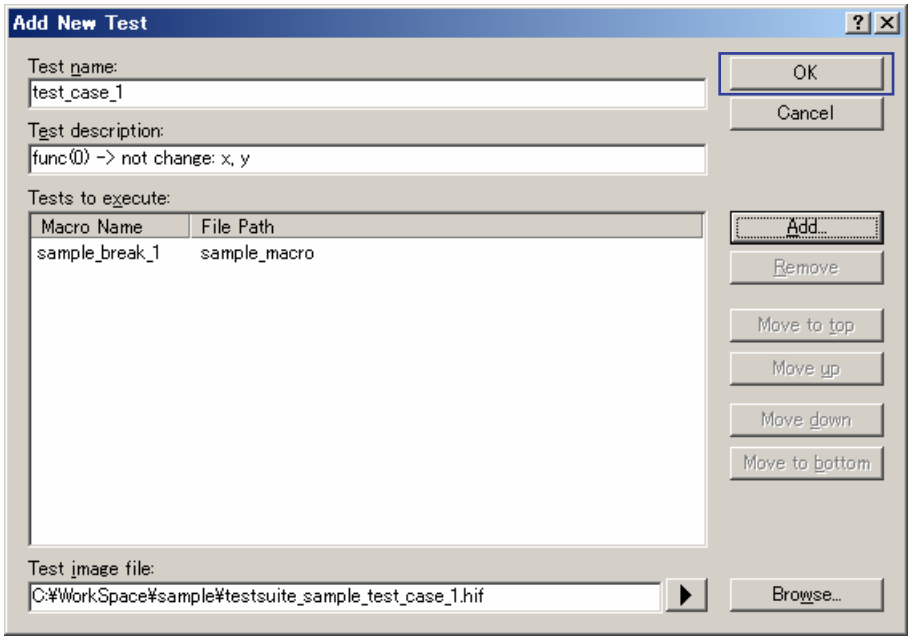

Figure 3-12

The **Modify Test Suite** dialog box is displayed again. Create `test_case_2` for the test case in *(ii) Test case 2: func(1)* as follows.

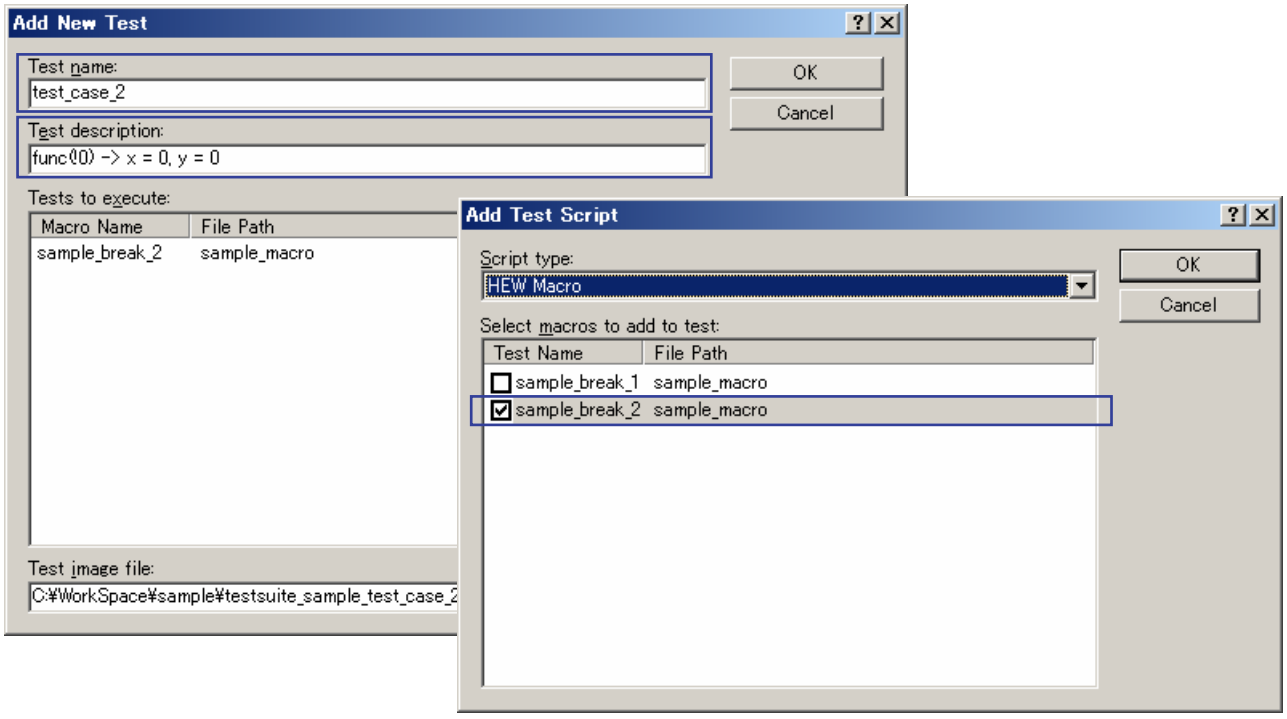

Figure 3-13

After editing the two test cases, click the **OK** button.

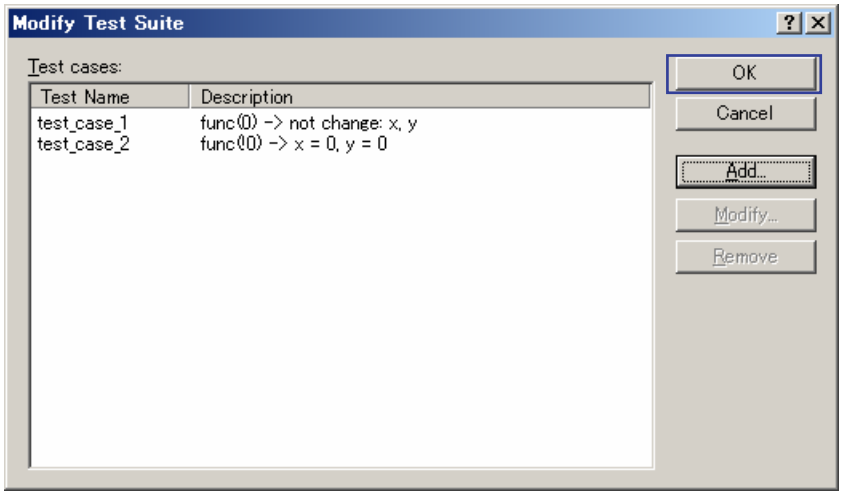

Figure 3-14

The test image files `C:¥WorkSpace¥sample¥testsuite_sample_test_case_1.hif` and `C:¥WorkSpace¥sample¥testsuite_sample_test_case_2.hif` are created. When the **Test** tab in the Workspace window (**Workspace** in the **View** menu) is displayed, the test cases `test_case_1` and `test_case_2` are added to the test suite.

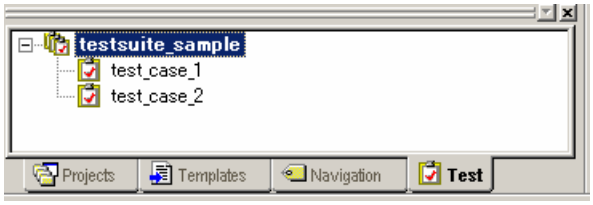

Figure 3-15

3.2.4 Creating a test image file

Execute the test case, and check that the program obtains the correct execution results. If the correct execution results are obtained, the results are output to the test image file. The test targets to be checked (such as variable values and register values) can be specified by editing the test image file. The sample project checks the values of the global *x* and *y* variables. Since the values of the variables are output to the test image file, symbols need to be registered for the *x* and *y* variables in the Watch window (from the **View** menu, choose **Symbol** and then **Watch**). To perform symbol registration, right-click in the Watch window to display a pop-up menu, and then choose **Add Watch** to display the **Add Watch** dialog box. Then, register the *x* and *y* variables in the **Add Watch** dialog box.

Figure 3-16

Execute the `test_case_1` test case first. From the **Test** menu, choose **Run Tests** to display the **Run Tests** dialog box, and then select the **test_case_1** checkbox. Since a macro that performs a build is set in the `test_case_1` test case, the value of the **Test timeout** field needs to be set to a number of seconds for which no timeout will occur during build execution.

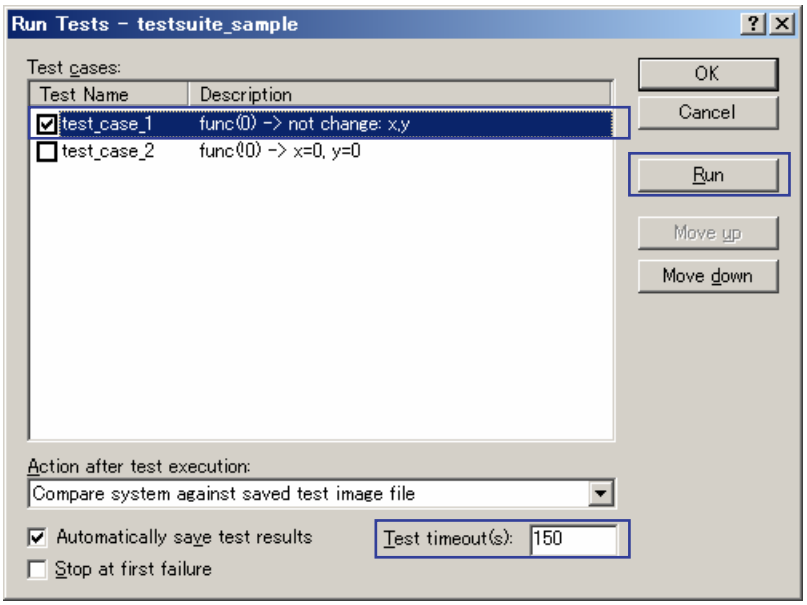

Figure 3-17

When the test is executed, a break occurs in the check location for `test_case_1`. The Watch window (Figure 3-18) can be used to check that the values of *x* and *y* are correct. In this case, the values `x=1` and `y=1` are correct.

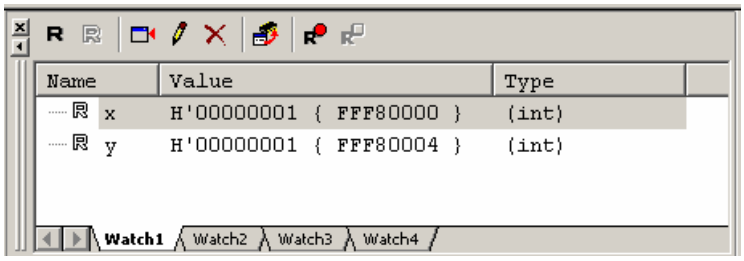

Figure 3-18

Once the values are checked to be correct, edit the test image file by specifying the values of the *x* and *y* variables displayed in the Watch window. Right-click **test_case_1** in **Test** tab of the Workspace window, and choose **Edit Test Image File** from the displayed pop-up menu.

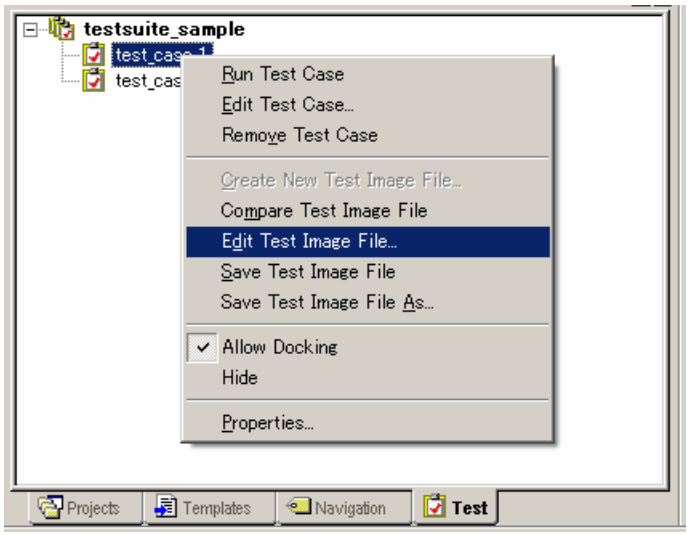

Figure 3-19

The items to be checked for **test_case_1** are the *x* and *y* variables registered in the Watch window.

In the **Edit Test Image File** dialog box, select the **Watch-SimSessionSH2A-FPU_Cycle** checkbox.

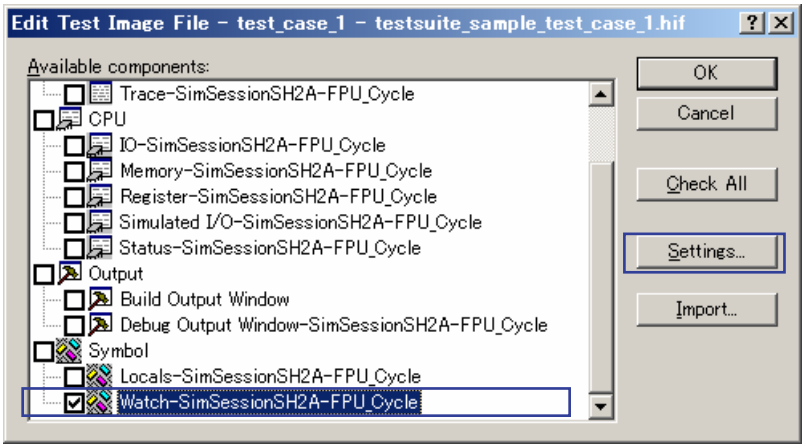

Figure 3-20

Then, click the **Settings** button to display the **Edit Test Watch** dialog box. In the symbol list of the **Edit Test Watch** dialog box, select the checkboxes for the *x* and *y* variable symbols, and click the **OK** button.

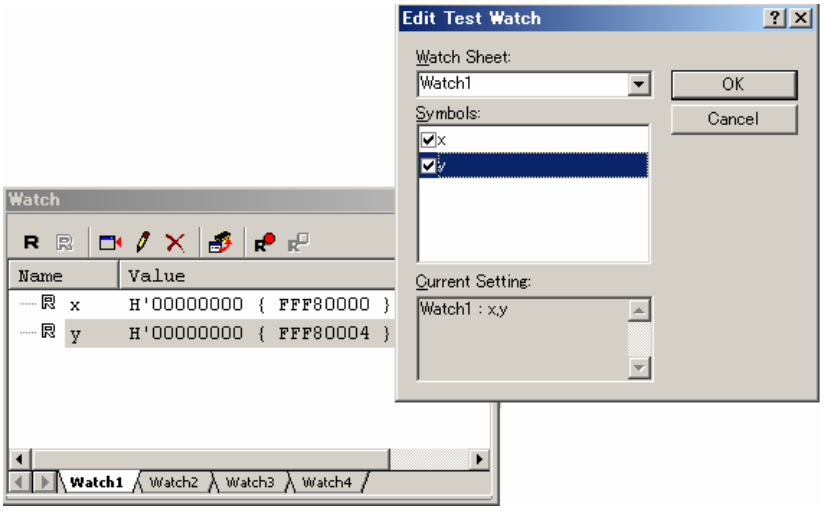

Figure 3-21

The **Edit Test Image File** dialog box is displayed again. Click the **OK** button to save the values of the *x* and *y* variables as displayed in the Watch window, to the test image file `C:¥WorkSpace¥sample¥testsuite_sample_test_case_1.hif`.

Execute the **test_case_2** test the same as **test_case_1** (Figure 3-22).

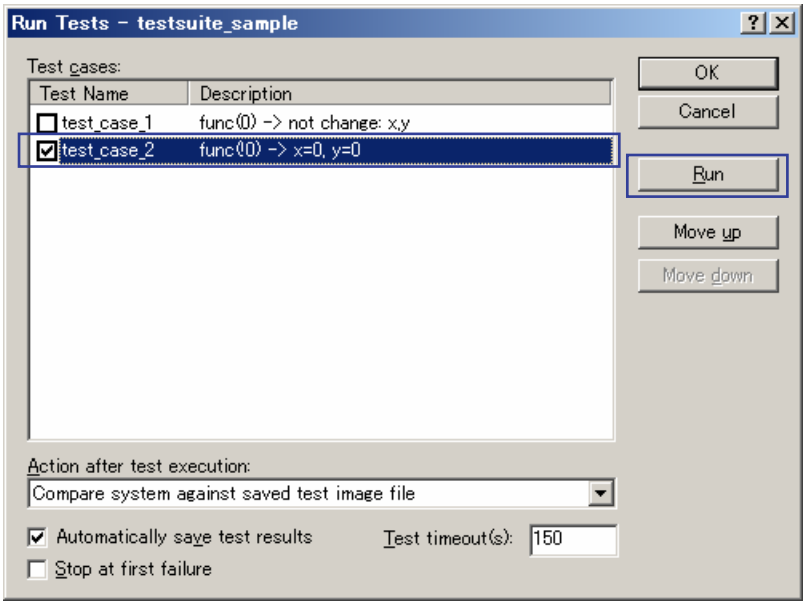

Figure 3-22

When the test is executed, a break occurs in the check location for `test_case_2`. The Watch window (Figure 3-18) can be used to check that the values of *x* and *y* are correct. In this case, the values `x=0` and `y=0` are correct. If the results are correct, edit the test image file for `test_case_2`. As with `test_case_1`, select the **Watch-SimSessionSH2A-FPU_Cycle** checkbox in the **Edit Test Image File** dialog box. When editing is completed for the test image file, the values of the *x* and *y* variables as displayed in the Watch window are to the test image file `C:¥WorkSpace¥sample¥testsuite_sample_test_case_2.hif`.

3.3 Regression testing

The test support facility can be used to compare previously saved test results with test results after a program has been changed, to check for degradation. The following uses two cases to explain this for the sample project.

(1) Checking operation after a program change
The sample program is changed and then checked for degradation.

(2) Checking for normal operation
The malfunction detected in (1) is corrected, and then checked for correctness.

3.3.1 Checking operation after a program change

With the `func()` function, even when the substitution condition for the *x* variable and the substitution condition for the *y* variable are the same, redundant processing is performed in which an `if` statement is specified for each substitution. The program is changed as follows to unify these into one `if` statement.

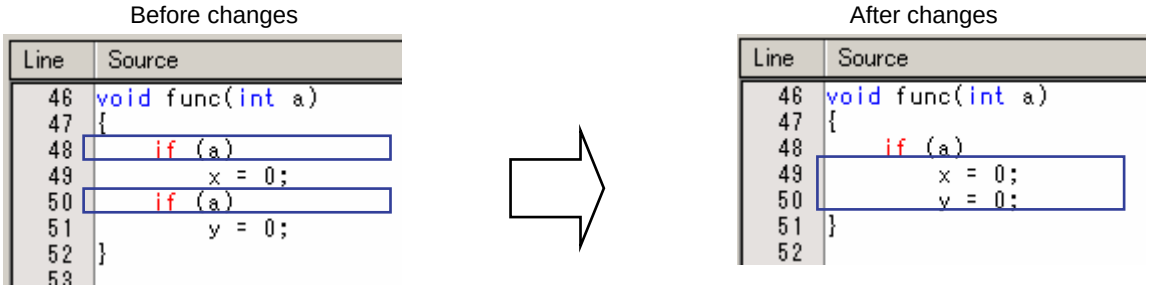

Figure 3-23

Once the program is changed, tests are executed as follows, and the test results after the program change are checked to see if they are the same as the test image.

From the **Test** menu, choose **Run Tests** to display the **Run Tests** dialog box. In the list of test cases in the **Run Tests** dialog box (Figure 3-24), select the checkboxes for test cases **test_case_1** and **test_case_2**.

From the **Action after test execution** drop-down list, choose **Compare system against saved test image file**. Click the **Run** button to execute the tests.

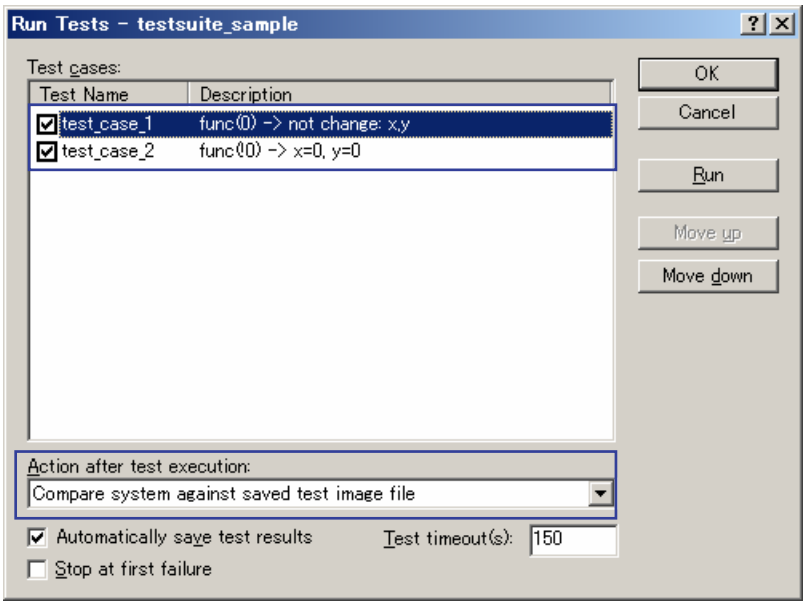

Figure 3-24

After execution of the tests, the test results browser (Figure 3-25) is displayed.

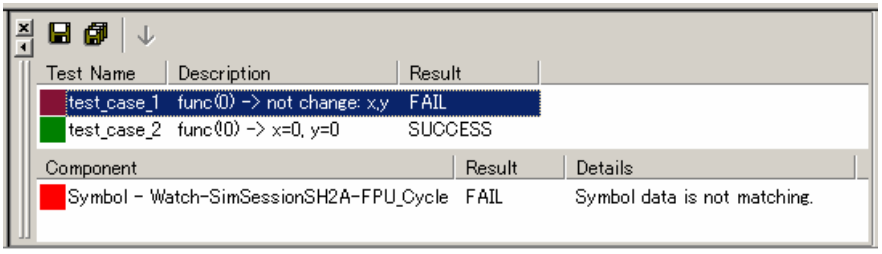

Figure 3-25

Items displayed in red in the test results browser indicate execution results that differ from the test image.

Double-click the red icons displayed in the bottom pane to check details about non-matching data.

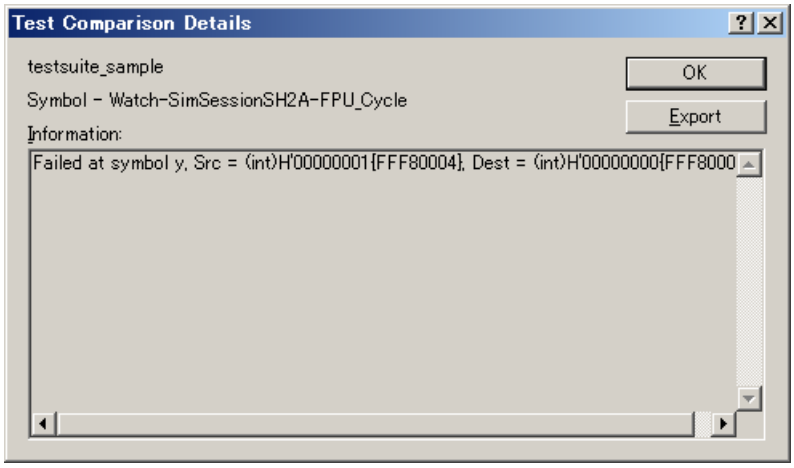

Figure 3-26

3.3.2 Checking for normal operation

In the previous program change, the `then` clause of the `if` statement contained only a substitution expression for *x*, and processing was performed in which *y* was substituted with `0` unconditionally. As such, an error was detected for test case 1 testing. The following changes are performed to make the program run correctly.

Before changes

```
Line  Source
46    void func(int a)
47    {
48        if (a)
49            x = 0;
50            y = 0;
51    }
52
```

After changes

```
Line  Source
46    void func(int a)
47    {
48        if (a) {
49            x = 0;
50            y = 0;
51        }
52    }
```

Figure 3-27

Once the program is changed, testing is executed again to make sure that the malfunction has been corrected.

If all tests are successful, green icons are displayed for all items as follows.

Test Name	Description	Result
test_case_1	func(0) -> not change: x,y	SUCCESS
test_case_2	func(0) -> x=0, y=0	SUCCESS

Component	Result	Details
Symbol - Watch-SimSessionSH2A-FPU_Cycle	SUCCESS	

Figure 3-28

Website and Support <website and support,ws>

Renesas Technology Website

http://japan.renesas.com/

Inquiries

http://japan.renesas.com/inquiry

csc@renesas.com

Revision Record <revision history,rh>

		Description	
Rev.	**Date**	**Page**	**Summary**
1.00	Jan.10.08	--	First edition issued

Notes regarding these materials

1. This document is provided for reference purposes only so that Renesas customers may select the appropriate Renesas products for their use. Renesas neither makes warranties or representations with respect to the accuracy or completeness of the information contained in this document nor grants any license to any intellectual property rights or any other rights of Renesas or any third party with respect to the information in this document.
2. Renesas shall have no liability for damages or infringement of any intellectual property or other rights arising out of the use of any information in this document, including, but not limited to, product data, diagrams, charts, programs, algorithms, and application circuit examples.
3. You should not use the products or the technology described in this document for the purpose of military applications such as the development of weapons of mass destruction or for the purpose of any other military use. When exporting the products or technology described herein, you should follow the applicable export control laws and regulations, and procedures required by such laws and regulations.
4. All information included in this document such as product data, diagrams, charts, programs, algorithms, and application circuit examples, 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 products listed in this document, please confirm the latest product information with a Renesas sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas such as that disclosed through our website. (http://www.renesas.com)
5. Renesas has used reasonable care in compiling the information included in this document, but Renesas assumes no liability whatsoever for any damages incurred as a result of errors or omissions in the information included in this document.
6. When using or otherwise relying on the information in this document, you should evaluate the information in light of the total system before deciding about the applicability of such information to the intended application. Renesas makes no representations, warranties or guaranties regarding the suitability of its products for any particular application and specifically disclaims any liability arising out of the application and use of the information in this document or Renesas products.
7. With the exception of products specified by Renesas as suitable for automobile applications, Renesas products are not designed, manufactured or tested for applications or otherwise in systems the failure or malfunction of which may cause a direct threat to human life or create a risk of human injury or which require especially high quality and reliability such as safety systems, or equipment or systems for transportation and traffic, healthcare, combustion control, aerospace and aeronautics, nuclear power, or undersea communication transmission. If you are considering the use of our products for such purposes, please contact a Renesas sales office beforehand. Renesas shall have no liability for damages arising out of the uses set forth above.
8. Notwithstanding the preceding paragraph, you should not use Renesas products for the purposes listed below:
   (1) artificial life support devices or systems
   (2) surgical implantations
   (3) healthcare intervention (e.g., excision, administration of medication, etc.)
   (4) any other purposes that pose a direct threat to human life

   Renesas shall have no liability for damages arising out of the uses set forth in the above and purchasers who elect to use Renesas products in any of the foregoing applications shall indemnify and hold harmless Renesas Technology Corp., its affiliated companies and their officers, directors, and employees against any and all damages arising out of such applications.
9. You should use the products described herein within the range specified by Renesas, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas shall have no liability for malfunctions or damages arising out of the use of Renesas products beyond such specified ranges.
10. Although Renesas endeavors to improve the quality and reliability of its products, IC products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Please be sure to implement safety measures to guard against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas 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 applicable measures. Among others, since the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
11. In case Renesas products listed in this document are detached from the products to which the Renesas products are attached or affixed, the risk of accident such as swallowing by infants and small children is very high. You should implement safety measures so that Renesas products may not be easily detached from your products. Renesas shall have no liability for damages arising out of such detachment.
12. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written approval from Renesas.
13. Please contact a Renesas sales office if you have any questions regarding the information contained in this document, Renesas semiconductor products, or if you have any other inquiries.