RENESAS Tool News

RENESAS TOOL NEWS on August 1, 2005: RSO-SHC-050801D

The C/C++ Compiler Package for the SuperH RISC Engine Family Revised to V.9.00 Release 03

We have revised the C/C++ compiler package for the SuperH RISC engine family of MCUs from V.9.00 Release 02 to V.9.00 Release 03.

1. **Product Concerned**
   The SuperH RISC engine family C/C++ compiler packages V.9

2. **Descriptions of Revision**
   2.1 Functions Introduced and Improved
   2.1.1 In the High-performance Embedded Workshop (an Integrated Development Environment)
   The High-performance Embedded Workshop bundled with the package has been revised to V.4.00.02.
   For details, see RENESAS TOOL NEWS No. RSO-HEW-050701D, "The High-performance Embedded Workshop, an Integrated Development Environment, Revised to V.4.00.02" issued on July 1, 2005.

   2.1.2 In the Simulator
   (1) The timer simulation function has been separated from the simulator body to be executed as an I/O DLL.
   (2) The ROM cache simulation function has been introduced to the SH2A-FPU Functional Simulator and the SH2A-

FPU Cycle Base Simulator.

(3) The Register Bank window for displaying and changing register banks has been introduced to the SH2A-FPU Functional Simulator and the SH2A-FPU Cycle Base Simulator.

2.1.3 In the Assembler

In arithmetic operations of floating-point numbers in decimal notation, omission of the second "F'" and later ones is allowed. Note, however, that when they are omitted, decimal fraction must not be omitted.

2.2 Problems Fixed

The problems described in Sections 2.2.1 through 2.2.4 below have been fixed.

2.2.1 In the Installer

On executing the batch file for setting environmental variables

For details, see RENESAS TOOL NEWS No. RSO-SHC_1-050316D, "A Note on Using the C/C++ Compiler Packages for the SuperH RISC Engine; and the H8, H8S and H8SX Families (Windows Versions Only)" issued on March 16, 2005.

2.2.2 In the Simulator

(1) On using the simulator debugger for the SuperH RISC engine family and the one for the H8, H8S and H8SX families in the same host system

For details, see RENESAS TOOL NEWS No. RSO-SHC_2-050316D, "A Note on Using the Simulator Debuggers for the SuperH RISC Engine; and the H8, H8S, and H8SX Families" issued on March 16, 2005.

(2) On setting access to memory

resources
For details, see RENESAS TOOL NEWS No. RSO-SH-SIM-050416D, "A Note on Using the Simulator Debugger for the SuperH RISC Engine; and for the H8S Families and the H8/300 Series of MCUs --On Setting Access to Memory Resources--" issued on April 16, 2005.

(3) On the number of execution cycles
For details, see RENESAS TOOL NEWS No. RSO-SH-SIM-050516D, "A Note on Using the SuperH RISC Engine Simulator Debugger" issued on May 16, 2005.

2.2.3 In the Compiler
Eight problems encountered in using the C/C++ compiler package V.9 for the SuperH RISC engine family
For details, see RENESAS TOOL NEWS No. RSO-SHC- 050616D, "Notes on Using the C/C++ Compiler Package V.9 for the SuperH RISC Engine Family of MCUs" issued on June 16, 2005.

2.2.4 In the Linker (Optimizing Linkage Editor optlnk)

(1) On linkage list files generated by the optimizing linkage editor
For details, see RENESAS TOOL NEWS No. RSO-SHC-050416D, "A Note on Using the Optimizing Linkage Editor for the SuperH RISC Engine, H8, H8S, and H8SX Families of MCUs" issued on April 16, 2005.

(2) On displaying the error message shown below accidentally ** L0103 (I) Multiple stack sizes specified to the symbol "Function Name"

Conditions:
This problem may occur if the following conditions are all satisfied:

1. The version of the linker is V.9.00.00 or later.*
2. An extern-qualified variable that has the same name as a function defined in a C/C++ source file exists in another C/C++ source file.
   Or, an import label of "_<Function Name>" exists in an assembly source file.
3. The message option is used at linking.
4. Also the stack option is used at linking.

(3) On generating an incorrect stack information file (.sni) Conditions:
This problem may occur if the following conditions are all satisfied:

1. The version of the linker is V.9.00.00 or later.*
2. An extern-qualified variable that has the same name as a function defined in a C/C++ source file exists in another C/C++ source file.
   Or, an import label of "_<Function Name>" exists in an assembly source file.
3. The stack option is used at linking.

NOTICE:
If an incorrect .sni file is read into the stack analyzing tool, Call Walker, incorrect stack usage is provided.

(4) On generating incorrect object code by using the "optimize=symbol _delete" option to optimize the deletion of unreferenced symbols
Conditions:

This problem may occur if the following conditions are all satisfied:

1. The version of the linker is V.9.00.00 or later.*
2. The goptimize option is used at compilation.
3. Also the pack=1 option is used at compilation, or "#pragma pack 1" is put in a C/C++ source file.
4. Optimization of the deletion of unreferenced symbols is valid in the linker.
   Optimization becomes valid in any of the following cases:
   - The optimize=symbol_delete option is used.
   - The optimize=speed option is used.
   - The optimize option is used.
   - The nooptimize option is not used.
5. Constants (const- qualified variables) and variables having their initial values are deleted as unreferenced variables by the optimization in Condition 4.

(5) On generating incorrect debug information by using the compress option to compress debug information
Conditions:
This problem may occur if the following conditions are all satisfied:

1. The version of the linker is V.7.0 or later.*
2. The debug option is used at compilation.
3. Also the compress option is used for generating relocatable files (.rel) at linking.
4. The relocatable files in Condition 3 are read into the linker to generate a load module using the

"compress" option.
NOTICE:
Loading the above load module in the debugger results in errors.

(6) On displaying the error message shown below accidentally depending on the combination of the optimizing options used. ** L3320 (F) Memory overflow
Conditions:
This problem may occur if the following conditions are all satisfied:

1. The version of the linker is V.9.00.00 or later.*
2. The goptimize option is used at compilation.
3. Any of the following optimizing options is used at linking:
   - The three options optimize=symbol_delete, register, and string_unify are used at the same time.
   - The optimize=speed option is used.
   - Also the optimize option is used.
   - The nooptimize option is not used.
4. Several functions are deleted by the optimization in Condition 3.

(7) On generating internal errors
Conditions:
This problem may occur if any of the following conditions is satisfied:

- The linker reads .rel files containing .EQU symbols written in the assembly language. (Internal error L4000-8010 arises.)
- The linker reads object files containing function calls calling .EQU symbols (Internal error

L4000-8874 arises.)
- The linker reads .rel files. (Internal error L4000-8027 or L4001 arises.)

* How to check for the version number of your linker

(1) In the High-performance Embedded Workshop, open the Tool menu and select the Administration command. The Tool Administration dialog box appears.

(2) Out of the Toolchains tree in the Registered Components list, select the name of the compiler package you are using; then click the properties button.

(3) Click the Information tab in the Properties dialog box, and you see the version number of your linker.
Example of Display .. Optimizing Linkage Editor (V.9.00.02)

2.3 A Problem Unfixed

2.3.1 Description

If both the definition of a pointer-type variable whose initial value is a character string and that of a static variable exist in a file having no function definition, the address pointed to by the pointer-type variable may be incorrect.

2.3.2 Conditions

This problem occurs if the following conditions are all satisfied:

(1) The code=machinecode option is used or the code option is not used.

(2) In a file exists no function definition.

(3) In a file exists a static-qualified variable.

(4) A pointer-type variable is defined whose initial value is a character string.

(5) The variables in (3) and (4) satisfy any of the following conditions:

(a) The variable in (3) does not have its initial value.

(b) Only the variable in (4) is qualified to be const.

(c) The variables in (3) and (4) are placed in the same section, and the variable in (4) is defined earlier.

2.3.3 Workaround

This problem can be circumvented in either of the following ways:

(1) Use the code=asmcode option.

(2) Define a function.

(3) Don't qualify the variable in Condition (3) to be static.

3. How to Update Your Package and Purchase the Revised One

3.1 Free-of-Charge Update

Free-of-charge update is available if you are using the product concerned.

(1) For Windows version

To update yours online, download the revised product from **HERE**.

(2) For Solaris or HP-UX version

Supply the following items of information to your local Renesas Technology sales office or distributor. We will send you the latest version of the product package by return:

Host OS	Solaris or HP-UX
Version No.	V.9.00
Release No.	Release 03

3.2 Ordering Information

If you place an order for the product, please supply the following items of

information to your local Renesas Technology sales office or distributor:

Host OS	Windows, Solaris, or HP-UX
Version No.	V.9.00
Release No.	Release 03
