CS+ RH850 Compiler CC-RH V1.03.00 Release Note

R20UT3518EJ0100
Rev.1.00
September. 18, 2015

Thank you for using the CS+ integrated development environment.

This document describes the restrictions and points for caution. Read this document before using the product.

Contents

Chapter 1. Target Devices 2
Chapter 2. User's Manuals 3
Chapter 3. Keywords When Uninstalling the Product 4
Chapter 4. Changes 5
4.1 Changes to the CC-RH compiler 5
4.1.1 Standard and Professional editions 5
4.1.2 Support for the G3KH core 5
4.1.3 Enhanced -Xalign4 option 5
4.1.4 CRC calculation 6
4.1.5 Extension to the -Xpatch option 6
4.1.6 Removal of notes 7
4.1.7 Other changes and improvements 7
4.1.8 Checking of source code against MISRA-C:2004 rules [Professional] 8
4.1.9 Checking of source code against MISRA-C:2012 rules [Professional] 8
4.1.10 Detection of stack smashing [Professional] 8
Chapter 5. Points for Caution 9
5.1 FE level exceptions 9

Chapter 1. Target Devices

The target devices supported by the CC-RH compiler are listed on the Website.

Please see the URL below.

CS+ Product Page:

http://www.renesas.com/cs+

Chapter 2. User's Manuals

Please read the following user's manuals along with this document.

Manual Name	Document Number
CC-RH Compiler User's Manual	R20UT3516EJ0100
CS+ Integrated Development Environment User's Manual: CC-RH Build Tool Operation	R20UT3283EJ0101

Chapter 3. Keywords When Uninstalling the Product

There are two ways to uninstall this product.

- Use the integrated uninstaller (uninstalls all CS+ components)
- Use the Windows uninstaller (only uninstalls this product)

To use the Windows uninstaller, select “CS+ CC-RH V1.03.00” from “Programs and Features” of the control panel.

Chapter 4. Changes

This chapter describes changes to the CC-RH compiler.

4.1 Changes to the CC-RH compiler

This section describes changes to the CC-RH compiler from V1.02.00 to V1.03.00.

4.1.1 Standard and Professional editions

The CC-RH complier has the following two editions.

- Standard edition
- Professional edition

The features of the latter can only be used if the compiler is registered under the professional license. They are indicated as [Professional] from here on.

4.1.2 Support for the G3KH core

g3kh is specifiable as the argument of the -Xcpu option.

When the -Xcpu=*g3kh* option is specified, the generated code is for the instruction set of the G3KH. FPU instructions are generated from floating-point operations for single-precision arithmetic, and instructions to call runtime functions are generated from those for double-precision arithmetic.

4.1.3 Enhanced -Xalign4 option

The -Xalign4 option has been enhanced. When the -Xalign4 option was specified in V1.02.00, only the addresses where functions started were aligned with 4-byte boundaries. On the other hand, the format of the specification has become -Xalign4[=*mode*] in V1.03.00. Any of the modes listed below are specifiable as *mode.* If the specification is omitted, only the address where the function starts will be aligned with a 4-byte boundary. Aligning more items with 4-byte boundaries may increase the speed of execution, but will also increase the size in ROM.

mode	Description
function	Addresses where functions start are aligned with 4-byte boundaries.
loop	Addresses where functions and loops start are aligned with 4-byte boundaries.
innermostloop	Addresses where functions and innermost loops start are aligned with 4-byte boundaries.
all	Addresses where functions start and branch destination addresses are aligned with 4-byte boundaries.

4.1.4 CRC calculation

The -CRC linker option, which performs a CRC (Cyclic Redundancy Check) calculation and places the result at a specified address, has been added. Any of the methods listed below is selectable as the method of calculation. If no method is specified, the operation will be executed as 32-ETHERNET.

Method	Description
CCITT	The result of calculation is obtained by applying CRC-16-CCITT to the input MSB first, with the initial value of the result being 0xFFFF, and XOR inversion.
16-CCITT-MSB	The result of calculation is obtained by applying CRC-16-CCITT to the input MSB first.
16-CCITT-MSB-LITTLE-4	The input is a 4-byte unit with little endian. The result of calculation is obtained by applying CRC-16-CCITT to the input MSB first.
16-CCITT-MSB-LITTLE-2	The input is a 2-byte unit with little endian. The result of calculation is obtained by applying CRC-16-CCITT to the input MSB first.
16-CCITT-LSB	The result of calculation is obtained by applying CRC-16-CCITT to the input LSB first.
16	The result of calculation is obtained by applying CRC-16 to the input LSB first.
SENT-MSB	The input is 1 byte, with the higher- and lower-order 4 bit units in little endian. The result of calculation on the input MSB first is obtained with SENT compliance.
32-ETHERNET	The result of calculation is obtained by applying CRC-32-ETHERNET to the input. The initial value of the result is 0xFFFFFFFF, and is XOR inverted and the bit order is reversed.

4.1.5 Extension to the -Xpatch option

The -Xpatch option has been extended to enable the specification of *syncp* and *switch* as an argument for the microcomputer which loads the G3M core.
When both -Xcpu=g3m option and -Xpatch=*syncp* option are specified, a syncp instruction is inserted at the beginning of interrupt functions of the types described below.

- The interrupt specification for a #pragma interrupt directive includes any of priority=SYSERR/FPI/FENMI/FEINT/EIINT_PRIORITY*X* (*X* = 0 to 15) or a channel is specified for the interrupt function.

When both -Xcpu=g3m option and -Xpatch=*switch* option are specified, it may lead to the generation of code using switch instructions. Switch instructions will not be used unless this option is specified.

4.1.6 Removal of notes

Five notes on using CC-RH below have been removed.

- Designating immediate values as the operands of assembly instructions (No. 2)
- Definition of and reference to a label being in the same assembly unit (No. 3)
- Omitting base registers in assembly instructions (No. 4)
- Assembly statements not in accord with the specification (No. 5)
- Optimization of access to external variables placing them over a 4-Mbyte or greater range of addresses (No. 6)

4.1.7 Other changes and improvements

Other major changes and improvements are described below.

(a) Improved debugging information

Unnecessary debugging information was deleted to reduce the sizes of object files (*.obj) and load module files (*.abs).

(b) Changes to the specification of section operators

The ability to specify non-existent sections for the section operators “STARTOF” and “SIZEOF” has been added to the assembler.

Note: The name of the operators is "section aggregation operators" in V1.01.00 and earlier versions. In V1.02.00, the name has been changed to "section operators"

(c) Change to a specification of inline assembler

The availability to select whether subsequent instructions are expanded or not by using the assembler control instructions $MACRO/$NOMACRO in inline assembler declared under #pragma inline_asm has been added. Subsequent instructions are not expanded in inline assembler by default.

(d) Improved prevention of internal errors

The prevention of compiler-internal errors during building has been improved.

4.1.8 Checking of source code against MISRA-C:2004 rules [Professional]

The following options can only be used if the compiler is registered under the professional license.

- -Xmisra2004
- -Xignore_files_misra
- -Xcheck_language_extension

4.1.9 Checking of source code against MISRA-C:2012 rules [Professional]

A -Xmisra2012 option, which selects the checking of source code against the MISRA-C:2012 rules, has been added. An -Xignore_files_misra option for the selection of files that are not to be checked, and a -Xcheck_language_extension option which enables the source-code checking, which are partially suppressed by language extensions, are specifiable at the same time as each other and -Xmisra2012.

Specifying these options to statically check source code makes improving the quality of user programs in early stages possible.

4.1.10 Detection of stack smashing [Professional]

A feature for the detection of stack smashing has been added. This feature can be realized by the -Xstack_protector/-Xstack_protector_all compiler options, or the extended language specification #pragma stack_protector/#pragma no_stack_protector.

This feature generates the codes which detect stack smashing at the entry and exits of functions. Specifically, instructions to execute the following three processes are produced.

1. 4-byte immediately before (upper direction) the local variable area of the stack frame are allocated at the entrance to the function, and the values specified with the option are stored there.
2. A check is run on whether the value which was stored in step 1 has not been changed at the end of the function.
3. The __stack_chk_fail function is called when the value has been changed, which would lead to smashing the stack.

The __stack_chk_fail function is defined by user and describes the processing to be executed when stack smashing is detected. For example, the __stack_chk_fail function will be executed when the following code, which smashes the stack area, is executed.

```
1: void f1() {
2:    volatile char str[10];
3:    int i;
4:    for (i = 0; i <= 10; i++){
5:       str[i] = i;           // The stack is smashed when i=10
6: }
```

This feature is an active check that can provide a security measure (against stack overflow and thus preventing attacks).

Chapter 5. Points for Caution

This section describes points for caution regarding CC-RH.

5.1 FE level exceptions

When a #pragma interrupt directive is defined for the interrupt function for an FE level exception from which return or recovery is not possible, i.e. when FENMI or SYSERR is designated as the priority, the code for the exit from the interrupt function is not output. This should be handled correctly in the program.

SALES OFFICES

Renesas Electronics Corporation

http://www.renesas.com

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

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

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

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

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

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

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

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

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

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

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

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

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

Colophon 5.0