RL78 Compiler CC-RL V1.01.00 Release Note

R20UT3393EJ0100
Rev.1.00
Feb 26, 2015

Contents

Chapter 1 User's Manuals

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

Manual Name	Document Number
CC-RL Compiler	R20UT3123EJ0101

Chapter 2 Changes

This chapter describes changes in V1.01.00 of the CC-RL compiler.

2.1 Changes in V1.01.00 of the CC-RL Compiler

This section describes changes in the revision of the CC-RL compiler from V1.00.00 to V1.01.00.

2.1.1 Improved Optimization

The performance of generated code has been improved.

2.1.2 Added Relocation Attributes and Directives [Assembler]

The following relocation attributes have been added.

Assembler Relocation Attribute	Explanation
SBSS_BIT	Sets a relocation attribute for a non-initialized bit section within the saddr area.
BSS_BIT	Sets a relocation attribute for a non-initialized bit section within the RAM area.
BIT_AT	Sets a relocation attribute for a non-initialized bit section that is allocated to a specified address.

The following directives have been added.

Assembler Directive	Explanation
.BSEG	Specifies the start of a bit section for the assembler.
.DBIT	Reserves a 1-bit area.

2.1.3 Added Facilities to Support Porting [Assembler]

The following descriptions have been supported in the porting support functions.

Statement (CA78K0R Assembly Language Specifications)	Corresponding Statement (RL78 Assembly Language Specifications)
BSEG without a relocation attribute	.BSEG SBSS_BIT
BSEG UNIT	.BSEG SBSS_BIT
BSEG AT	.BSEG BIT_AT
DBIT	.DBIT

2.1.4 Improved Directives for Reserving Areas in Support of Porting [Assembler]

For better support of porting, the use of multiple operands with the following directives is now possible.

Assembler Directive
DB
DW
DG

2.1.5 Improved Handling of Include Directives in Support of Porting [Assembler]

To improve porting support, the assembler now supports "$INCLUDE(…)" directives where the filename is preceded by a space (below indicated by ∆).

"$INCLUDE(∆a.inc)" has the same meaning as "$INCLUDE(a.inc)".

2.1.6 Addition of a Macro [Assembler]

The following macro has been added.

Macro Name	Definition
__RENESAS_VERSION__	If the version is V. XX.YY.ZZ, it is entered as 0xXXYYZZ00.

Note that this macro is only effective when ccrl is started; it is not added when asrl is started.

2.1.7 Addition of Sections for the Standard Library and Runtime Library Code [Library]

The following sections have been added.

Section Name	Relocation Attribute	Explanation
.SLIB	TEXTF	A section for code from the standard library
.RLIB	TEXTF	A section for code from the runtime library

2.1.8 Added Functions and Macros [Library]

The following functions and macros have been added.

Function Name/Macro Name	Explanation
printf_tiny	A simplified version of printf
sprintf_tiny	A simplified version of sprintf
__PRINTF_TINY__	Replaces calls of printf/sprintf with calls of printf_tiny/sprintf_tiny.

2.1.9 Changes to Handling of NaNs by ldexp and ldexpf [Library]

Handling of NaNs by library functions ldexp and ldexpf has been changed as follows.

▪ When the input parameter is NaN, NaN is returned and the macro EDOM is set in the global variable errno.

2.1.10 Improved Performance for Mathematical Functions [Library]

The processing time and size of the following mathematical functions have been improved.

Function Name	Explanation
sqrt	Square root (in double precision)
sqrtf	Square root (in single precision)

2.1.11 Added and Modified Macros [Compiler]

The following macros have been added.

Macro Name	Definition
__CNV_CA78K0R__	The value is set to decimal constant 1 (defined when ca78k0r is specified with the -convert_cc option).
__CNV_NC30__	The value is set to decimal constant 1 (defined when nc30 is specified with the -convert_cc option).
__CNV_IAR__	The value is set to decimal constant 1 (defined when iar is specified with the -convert_cc option).
__BASE_FILE__	Sets the name of a file of C source code.

The following macro was updated so that it is only enabled when -ansi is specified.

__STDC_VERSION__	The value is set to decimal constant 1199409L.

2.1.12 Change to the Specifications of Macros, Allowing Variable Arguments [Compiler]

Variable arguments are now allowed for macros.

2.1.13 Addition of a Keyword for the Compiler [Compiler]

The following keyword has been added.

Keyword	Explanation
__callt	Calls a function with the callt instruction.

2.1.14 Addition of #pragma Directives [Compiler]

The following #pragma directives have been added.

#pragma Directive	Explanation
#pragma callt	Causes a call of a function with the callt instruction.
#pragma saddr	Causes the allocation of static variables to the saddr area.

2.1.15 Added Locations where Enabling Description of Binary Constants [Compiler]

The C-language syntax and the following #pragma directives now allow the description of binary constants.

#pragma Directive	Location where the Statement of Binary Constants is Allowed
#pragma interrupt	Parameters of vect
#pragma rtos_interrupt	Parameters of vect
#pragma address	Addresses

2.1.16 Added Embedded Functions [Compiler]

The following Embedded Functions have been added.

Embedded Function Name	Explanation
__get_psw	Returns the value of the PSW.
__set_psw	Sets a value for the PSW.

2.1.17 Change to the Specification of the Format of #pragma rtos_interrupt [Compiler]

The specification of vect can now be omitted and the behavior of the directive when vect is not specified has been defined.

- #pragma∆rtos_interrupt∆[(]<function name>[(vect=address)][)]

2.1.18 Changes to the Specification of the near and far Attributes of Interrupt Handlers for which #pragma interrupt/interrupt_brk/rtos_interrupt is Specified [Compiler]

Specifications of #pragma interrupt and #pragma rtos_interrupt have been changed.

- When vect is specified, the interrupt handler is forcibly assigned the __near attribute.
- When vect is not specified, handling of the __near or __far attributes of interrupt handlers is not changed.

The specification of #pragma interrupt_brk has been changed.

- An interrupt handler is forcibly assigned the __near attribute.

2.1.19 Change to the Specifications of New Section Names in #pragma section [Compiler]

The specification has been changed so that "." (dot) can only be used at the beginning of a new section name when the section type is specified.

2.1.20 Change to Character Set gb2312 [Compiler]

The syntax of the -character_set option for the C language compiler has been changed so that the gbk multibyte character set is specifiable instead of gb2312.

2.1.21 Added Options [Compiler]

The following options have been added.

Option Name	Explanation
-convert_cc	Supports the porting of programs for other compilers.
-pack	Sets the alignment of structure members to 1 byte.

2.1.22 Specification of Options Available for the Professional Version [Compiler]

The following options are now available in the professional version.

Option Name
-misra2004
-ignore_files_misra
-check_language_extension

2.1.23 Avoiding Indirect References to Variables of Types Having Two or More Bytes at Odd-Numbered Addresses [Compiler]

A warning is output regarding the possible range in case of indirect references to variables of types having two or more bytes at odd-numbered addresses.

2.1.24 Additional Checked Items [Linker]

The following items have been added to the set of items checked.

- Allocation of sections to span 64-Kbyte or (64 KB – 1)-byte boundaries.
- Allocation to memory and consistency of allocation to memory of sections.

2.1.25 Added Options [Linker]

The following options have been added.

Option Name	Explanation
-VFINFO	Selects the output of a file of information on variables and functions.
-AUTO_SECTION_LAYOUT	Selects the automatic allocation of sections.
-DEBUG_MONITOR	Specifies the memory area of the OCD monitor.
-RRM	Specifies a working area for the RRM/DMM function.
-SELF	Specifies the allocation of no sections to the self-RAM area.
-SELFW	Specifies the output of a warning if a section is allocated to the self-RAM area.

-OCDTR	Specifies the allocation of no sections to the trace-RAM and self-RAM areas.
-OCDTRW	Specifies the output of a warning if a section is allocated to the trace-RAM and self-RAM areas.
-OCDHPI	Specifies the allocation of no sections to the hot plug-in RAM, trace-RAM, and self-RAM areas.
-OCDHPIW	Specifies the output of a warning if a section is allocated to the hot plug-in RAM, trace-RAM, and self-RAM areas.
-CHECK_DEVICE	Specifies checking for consistency between the device files used in object files and specified by the linker option.
-CHECK_64K_ONLY	Specifies no checking of whether sections are allocated to span (64 K – 1)-byte boundaries.
-NO_CHECK_SECTION_LAYOUT	Specifies no checking for consistency between the address to which a section is allocated and the memory addresses the device actually has.

2.1.26 Addition to Information Output when the –show Option is Specified [Linker]

When -show=struct is specified, information on the members of structures and unions is output to the link map file.

2.1.27 Addition to Symbols Generated by Specifying Options [Linker]

The following symbols have been added to those generated by specifying options.

Symbol Name	Explanation
__STACK_ADDR_START	Indicates the highest address + 1 of the stack area.
__STACK_ADDR_END	Indicates the lowest address of the stack area.
__RAM_ADDR_START	Indicates the lowest address of the RAM area.
__RAM_ADDR_END	Indicates the highest address + 1 of the RAM area.
.monitor1	Indicates the area from 0xCE to 0xD7.
.monitor2	Indicates the area from the address where the OCD monitor starts to the address where it ends.

2.1.28 Change at Startup

The specification of the stack area setting at startup has been changed.

2.1.29 Elimination of Points for Caution

The following five points for caution have been eliminated.

1. Point to note regarding the use of both judgment of a match and greater or less than for variables

   Within a given function, a given if statement or loop might be resolved wrongly when an if statement or loop-control expression includes a combination of expressions to compare for a match ("!=" or "==") between, and for other types of comparison of a constant and variable.

   [Conditions]

   This problem arose if the following conditions were all met.

(1) Any of the –Odefault, -Osize, or –Ospeed options is enabled.
(2) A comparison expression "!=" or "==" comparing a constant (Note 1) and a variable (Note 2) is present.
Note 1: Includes expressions in which the constant is statically known to be a constant.
Note 2: Includes array variables, structure members, and union members.
(3) An expression that applies "<", ">", "<=", or ">=" to compare the constant and variable covered by (2), within a function which contains a comparison expression of the type described in (2).
(4) The variable in (2) is not modified by volatile.
(5) The comparison expressions covered by (2) and (3) meet any of the following conditions.
(5-1) The comparison expressions covered by (2) and (3) are connected by "||" or "&&".
(5-2) The comparison expressions covered by (2) and (3) are included in "if" statements or "?:" expressions and the "if" statements or "?:" expressions are executed in succession.
(6) There is no other expression or statement between the comparison expressions covered by (2) and (3).
Example where the problem arises if condition (1) is also satisfied.
[C source code]

```
int g(void);
int f(void)
 {
    int x = g();      // Condition (4)
    if (x == -2 ||    // Conditions (2) and (5-1)
        x > 1200) {   // Conditions (3) and (6)
      return 1;
    }
    return -1;
 }
```

[Workaround]
Any of the following steps were required to avoid this problem.
(1) Designate the -Onothing option.
(2) Modify the variable in condition (2) by declaring it as volatile.
(3) In the comparisons covered by (2) and (3), refer to a dummy volatile variable just before the next expression to be executed.
(Example) The following shows a case of workaround (3).

```
int g(void);
volatile int dummy;          // Declaration of dummy as a volatile variable
int f(void)
{
  int x = g();
  if (x == -2 ||
      (dummy, x > 1200)) { // Workaround (3)
    return 1;
  }
  return -1;
}
```

2. Point for caution regarding access not being correct when a pointer is used as an index

This point for caution applied when a pointer was in use as an index and correct access by the code for reference to the address to which the pointer should point became impossible if a symbol with static storage duration was cast to the integer type and the result of casting was added to another pointer.

[Conditions]
This problem arises if the following conditions are all met.
It may also arise if the following conditions are met as a result of optimization such as inline expansion or folding.
(1) The address of a symbol with static storage duration is cast to the integer type.
(2)The result of casting in (1) is added to another pointer (see the note below).
Note: Addition also includes cases where the result of casting is used as an index of an array.
For example, the following code may operate incorrectly.
array[i + (int)&globalVariable] = 0;

(3) The code includes a statement for reference to, loading, or storing the location pointed to by the pointer produced in (2).
(Example)
[C source code]

```
extern char __near addressIsIndex;
extern char __near array[10];
char noteForAddressWithWordBase(char* base){
  return base[(signed)&addressIsIndex]; // Conditions (1), (2) and (3)
}
```

[Workaround]
Before using the value of a symbolic address cast to the integer type, that value had to be stored as an automatic variable of the volatile-qualified integer type, and reference made by using that automatic variable.

(Example) The example shown in (Note) in the condition had to be changed as follows.

```
volatile int int_globalVariableAddress = (int)&globalVariable;
array[i + int_globalVariableAddress] = 0;
```

3. Point for caution regarding specification of -D __RL78_MEDIUM__ while the -cpu=S1 option is specified and the -memory_model option is not

When the -cpu=S1 option is specified and the -memory_model option is not specified, -D __RL78_MEDIUM__ was incorrectly specified instead of -D __RL78_SMALL__.
[Conditions]
This problem arose if the following conditions were all met:
(1) The -cpu=S1 option is specified.
(2) The -memory_model option is not specified.
[Workaround]
Either of the following steps were required to avoid this problem.
(1) Specify -memory_model=small.
(2) Specify -D __RL78_SMALL -U __RL78_MEDIUM__.

4. Point for caution regarding CRC calculations leading to unexpected results when the range for CRC calculation includes an area that includes data expanded by padding.

When the range for CRC calculation included an area including data expanded by padding, the result of CRC calculation may not have been as expected.
[Conditions]
This problem arose if the following conditions were all met:
(1) The –padding option is specified for padding.
(2) The –crc option is specified.
(3) The range for CRC calculation includes an area having data expanded by padding.
[Workaround]
Either of the following steps was required to avoid this problem.
(1) Do not specify the –padding option.
(2) Do not include an area where data are expanded by padding in the range for CRC calculation.

Chapter 3 Points for Caution

Please refer to the user's manual for caution regarding V1.01.00 of the CC-RL compiler.

All trademarks and registered trademarks are the property of their respective owners.

Notice

1. 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.
2. 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.
3. 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.
4. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.
   "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 etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; and safety equipment etc.
   Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (nuclear reactor control systems, military equipment etc.). 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 for which it is not intended. 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 which the product is not intended by Renesas Electronics.
6. 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.
7. 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 systems manufactured by you.
8. 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.
9. 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. You should not use Renesas Electronics products or 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. When exporting the Renesas Electronics 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.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document, Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.
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.

RENESAS

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

© 2015 Renesas Electronics Corporation. All rights reserved.

Colophon 5.0