RL78 Compiler CC-RL V1.03.00 Release Note

R20UT3815EJ0100
Rev.1.00
Jun. 30, 2016

Contents

Chapter 1 User's Manuals

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

Manual Name	Document Number
CC-RL Compiler	R20UT3123EJ0103

Chapter 2 Changes

This chapter describes changes to V1.03.00 of the CC-RL compiler.

2.1 Changes to V1.03.00 of the CC-RL Compiler

This section describes the changes in CC-RL from V1.02.00 to V1.03.00. Note that the features which are only available to users holding a registered license for the Professional edition are indicated as [Professional edition].

2.1.1 Enhanced optimization

For V1.03.00, optimization has been further enhanced on points (1) to (4), listed and described below.

(1) Merging of stack areas allocated for auto arrays in different local scopes (reducing the stack size)

The compiler merges stack areas allocated for auto arrays that belong to different blocks ({}) whose lifetimes do not overlap.

```
<Example of source code>
int *g;
void func01(void)
{
  {
    int array01[10];
    g = array01;
    foo();
  }
  {
    int array02[10];          // Lifetimes of array01[10] and
                              // array02[10] do not overlap.
    g = array02+2;
  }
}
```

```
<Code generated by V1.02.00>
_func01:
        .STACK _func01 = 44          ; Size of allocated stack =44byte
        subw sp, #0x28
        movw ax, sp
        movw !LOWW(_g), ax
        call !!_foo
        movw ax, sp
        addw ax, #0x0018
        movw !LOWW(_g), ax
        addw sp, #0x28
        ret
```

```
<Code generated by V1.03.00>
_func01:
        .STACK _func01 = 24          ; Size of allocated stack =24byte
        subw sp, #0x14
        movw ax, sp
        movw !LOWW(_g), ax
        call !!_foo
        movw ax, sp
        addw ax, #0x0004
        movw !LOWW(_g), ax
        addw sp, #0x14
        ret
```

(2) Optimization of constant propagation

Obviously recognizable calculations of constants within loops are omitted.

```
<Example of source code>
int func02(int xtra, int n4, int e1[]) {
  int i,ix,j,k,l;
  j = 1;
  k = 2;
  l = 3;

  for (ix=0; ix<xtra; ix++) {
    for (i=0; i<n4; i++) {
      j = j*(k-j)*(l-k);          // j is always 1
      k = l*k-(l-j)*k;            // k is always 2
      l = (l-k)*(k+j);            // l is always 3
      e1[l-2] = j+k+l;            // l-2 is always 1, j+k+l is always 6
      e1[k-2] = j*k*l;            // k-2 is always 0, j+k*l is always 6
    }
  }
  return e1[0]+e1[1];
}
```

```
<Code generated by V1.02.00 (1/3)>
.BB@LABEL@1_3:          ; bb49
        movw ax, [sp+0x0E]
        xor a, #0x80
        movw [sp+0x0A], ax
        movw ax, hl
        movw hl, sp
        xor a, #0x80
        cmpw ax, [hl+0x0A]
        bnc $.BB@LABEL@1_5
.BB@LABEL@1_4:          ; bb1
                        ; j*(k-j)*(l-k) and
                        ; l*k-(l-j)*k, (l-k)*(k+j) are calculated each time.
        movw ax, [hl]
        subw ax, bc
        movw bc, ax
        movw ax, [sp+0x04]
        movw hl, ax
        mulh
        movw [sp+0x04], ax
        movw ax, [sp+0x02]
```

```
<Code generated by V1.02.00 (2/3)>
        subw ax, hl
        movw bc, ax
        movw ax, [sp+0x04]
        mulh
        movw hl, ax
        movw [sp+0x04], ax
        pop bc
        push bc
        movw ax, [sp+0x02]
        mulh
        movw [sp+0x0A], ax
        movw ax, [sp+0x02]
        movw bc, ax
        movw ax, [sp+0x00]
        subw ax, hl
        mulh
        movw bc, ax
        movw ax, [sp+0x0A]
        subw ax, bc
        movw [sp+0x02], ax
        addw ax, hl
        movw [sp+0x0A], ax
        movw hl, sp
        movw bc, ax
        movw ax, [hl]
        subw ax, [hl+0x02]
        mulh
        movw bc, ax
        movw [hl], ax
        movw ax, bc
        addw ax, ax
        addw ax, de
        addw ax, #0xFFFC
        movw hl, ax
        movw ax, [sp+0x0A]
        addw ax, bc
        movw [hl], ax
        movw ax, [sp+0x04]
        movw bc, ax
        movw ax, [sp+0x02]
        movw hl, ax
        mulh
        movw [sp+0x0A], ax
        pop bc
        push bc
        movw ax, [sp+0x0A]
        mulh
        movw bc, ax
        movw ax, hl
        addw ax, ax
        addw ax, de
        addw ax, #0xFFFC
        movw hl, ax
        movw ax, bc
        movw [hl], ax
        movw ax, [sp+0x06]
        movw hl, ax
```

```
<Code generated by V1.02.00 (3/3)>
        incw hl
        movw ax, hl
        movw [sp+0x06], ax
        movw ax, [sp+0x02]
        movw bc, ax
        movw [sp+0x02], ax
        br $!.BB@LABEL@1_3
```

```
<Code generated by V1.03.00>
.BB@LABEL@1_3:          ; bb48
        movw ax, [sp+0x04]
        xor a, #0x80
        movw hl, ax
        movw ax, bc
        xor a, #0x80
        cmpw ax, hl
        bnc $.BB@LABEL@1_5
.BB@LABEL@1_4:          ; bb1
                        ; 6 is always assigned to e1[1].
                        ; 6 is always assigned to e1[0].
        movw ax, #0x0006
        movw [de+0x02], ax
        movw [de], ax
        incw bc
        br $.BB@LABEL@1_3
```

(3) Optimization of induction variables

The compiler does not generate code for redundantly updating loop induction variables.

```
<Example of source code>
void callee(unsigned i);
void caller(void){
   unsigned i;
   for(i=128; i != 0; --i){
      callee(i);
   }
}
```

```
<Code generated by V1.02.00 (1/2)>
_caller:
        .STACK _caller = 8
        subw sp, #0x04
        movw ax, #0x0080
        movw [sp+0x02], ax
        movw [sp+0x00], ax ; Loop induction variable
                           ; is redundantly initialized.
.BB@LABEL@1_1:          ; bb
        call !!_callee
        movw ax, [sp+0x02]
        addw ax, #0xFFFF
        movw [sp+0x02], ax
        movw ax, [sp+0x00] ; Loop induction variable
                           ; is redundantly initialized.
```

```
<Code generated by V1.02.00 (2/2)>
        decw ax
        movw [sp+0x00], ax
        bnz $.BB@LABEL@1_1
.BB@LABEL@1_2:          ; return
        addw sp, #0x04
        ret
```

```
<Code generated by V1.03.00>
_caller:
        .STACK _caller = 6
        push hl
        movw ax, #0x0080
        movw [sp+0x00], ax
.BB@LABEL@1_1:          ; bb
        call !!_callee
        movw ax, [sp+0x00]
        addw ax, #0xFFFF
        movw [sp+0x00], ax
        bnz $.BB@LABEL@1_1
.BB@LABEL@1_2:          ; return
        pop hl
        ret
```

(4) Deleting unused code

The ability to delete unused code has been further enhanced.

```
<Example of source code>
unsigned long test(unsigned long long variable, int var) {
  if (var) {
    variable &= 0x012345678abcdefULL;
  }
  return (variable >> 32);
}
```

```
<Code generated by V1.02.00 (1/2)>
_test:
        .STACK _test = 4
        or a, x
        movw ax, [sp+0x0A]
        movw bc, ax
        movw ax, [sp+0x08]
        movw de, ax
        movw ax, [sp+0x06]
        movw hl, ax
        movw ax, [sp+0x04]
        bz $.BB@LABEL@1_2
.BB@LABEL@1_1:          ; if_then_bb
        and a, #0xCD        ;ax([sp+4]) is not referenced
                            ; in the subsequent lines.
        xch a, x            ;
        and a, #0xEF        ;
        xch a, x            ;
        movw ax, hl         ;hl([sp+6]) is not referenced
                            ; in the subsequent lines.
```

```
<Code generated by V1.02.00 (2/2)>
        and a, #0x78        ;
        xch a, x            ;
        and a, #0xAB        ;
        xch a, x            ;
        movw ax, de
        and a, #0x34
        xch a, x
        and a, #0x56
        xch a, x
        movw de, ax
        movw ax, bc
        clrb a
        xch a, x
        and a, #0x12
        xch a, x
        movw bc, ax
.BB@LABEL@1_2:          ; if_break_bb
        movw ax, de
        ret
```

```
<Code generated by V1.03.00>
_test:
        .STACK _test = 4
        or a, x
        movw ax, [sp+0x0A]
        movw bc, ax
        movw ax, [sp+0x08]
        movw de, ax
        bz $.BB@LABEL@1_2
.BB@LABEL@1_1:          ; if_then_bb
        and a, #0x34
        xch a, x
        and a, #0x56
        xch a, x
        movw de, ax
        movw ax, bc
        clrb a
        xch a, x
        and a, #0x12
        xch a, x
        movw bc, ax
.BB@LABEL@1_2:          ; if_break_bb
        movw ax, de
        ret
```

2.1.2 Improvements to the feature for checking source code against MISRA-C:2012 rules [Professional edition]

The following rule numbers have been added to those which can be designated as arguments of the -misra2012 option, which selects checking by the compiler of source code against the specified MISRA-C:2012 rules.

2.6 2.7

9.2 9.3

12.1 12.3 12.4

14.4

15.1 15.2 15.3 15.4 15.5 15.6 15.7

16.1 16.2 16.3 16.4 16.5 16.6 16.7

17.1 17.7

18.4 18.5

19.2

20.1 20.2 20.3 20.4 20.5 20.6 20.7 20.8 20.9 20.10 20.11 20.12 20.13 20.14

The following are the numbers of MISRA-C:2012 rules against which the V1.02.00 and V1.03.00 compilers can check source code for compliance.

Rule classification (number of rules in the standard)	*V1.02.00*	*V1.03.00*
Mandatory rules (10)	3	3
Required rules (101)	31	**58**
Advisory rules (32)	7	**21**
Total number of rules (143)	41	**82**

2.1.3 Enhancing the security of dynamic memory management [Professional edition]

A feature has been added for the detection of illicit operations in the release of heap space. This feature can be used by linking a dedicated standard library of secure functions related to dynamic memory allocation.

An error with code E0562310 will occur if the compiler is not registered for a license to the Professional edition.

Run the dedicated standard library as follows.

(1) As well as the actual areas allocated for users in the heap space by the calloc, malloc, and realloc functions, two extra bytes for use in the detection of illicit operations are added before and after each area, for a total of four added bytes.

(2) When called, the free and realloc functions determine if any of (a) to (c) applies.

(a) The argument is not a pointer to an actual area allocated by calloc, malloc, or realloc.

(b) The pointer is to an area that has already been released.

(c) Neither of the two two-byte margins for detecting illicit operations has been overwritten.

(3) In the event of any of the above, an illicit operation is assumed to have proceeded, and _heap_chk_fail will be called.

The user must write the __heap_chk_fail function. Write the processing which should be executed when any illicit operation has been detected in the heap space. For example, if either of the two two-byte margins for detecting illicit operations has been overwritten in the 6th line in the following program, _heap_chk_fail will be executed when the related heap space is released by the 8th line.

```
1: #include <string.h>
2: #include <stdlib.h>
3: void func(char *str) {
4:    char *buf;
5:    buf = malloc(4);
6:    strcpy(buf, str);     // The heap space will be corrupted depending
on the length of str
7:
8:    free(buf);
9: }
```

By using this feature, you can easily counter security problems through measures against the dual release of memory and against buffers overflowing.

2.1.4 Addition of the –Opipeline option

The –Opipeline option for pipeline optimization has been added.

2.1.5 Addition of intrinsic functions for bit manipulation

Intrinsic functions __set1, __clr1, and __not1 for bit manipulation have been added.

2.1.6 Rectified points for caution

Points for caution on the following two items no longer apply.

- External labels defined after conditional assembly control instructions (CCRL#006)
- Designating a member of a packed structure or union in an initializing declaration (CCRL#007)
- Writing an instruction operand in ways which are not included in the list of instruction operations (CCRL#008)
- Outputting code which overwrites a register for interrupt handlers (CCRL#009)

- Scope of optimization (CCRL#010)

2.1.7 Other changes and improvements

Other major changes and improvements are described below.

(a) Improved debugging information

A problem with C and assembly source code not being displayed properly during debugging has been corrected.

(b) Improved prevention of internal errors

A problem with an internal error during building has been corrected.

Chapter 3 Points for Caution

Please refer to the user's manual for caution regarding V1.03.00 of the CC-RL compiler.

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

Colophon 5.0