RL78 Compiler CC-RL V1.06.00 Release Note

R20UT4228EJ0101
Rev.1.01
Jan 16.2021

Contents

Chapter 1 User's Manuals

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

Manual Name	Document Number
CC-RL Compiler	R20UT3123EJ0106

Chapter 2 Changes

This section describes changes to CC-RL from V1.05.00 to V1.06.00.

The features of the latter can only be used if the compiler is registered under the professional license.

They are indicated by [Professional edition] from here on.

2.1 C99 standard

To select conformance of the language specifications with the C99 standard, the **-lang** and **-strict_std** compiler options have been added.

Note that this version of the compiler does not support variable-length arrays and complex types, and some standard library functions.

-lang=*{c|c99}*

When the -lang=c option is specified, the language specifications conform with the C90 standard.

When the -lang=c99 option is specified, the language specifications conform with the C99 standard.

-strict_std

This option selects processing of the C source program in strict accordance with the language standard (C90 or C99) specified with the -lang option. Error and warning messages are output in response to code that violates the given standard.

V1.05.00 and earlier versions have the -ansi option to select the processing of C source programs in strict accordance with the C90 standard; however, in V1.06.00 and later versions, the -strict_std option is used. If the -ansi option is specified in V1.06.00 and later versions, it is automatically converted to the -strict_std option for input to the compiler.

2.2 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 for use with the C99 standard, which selects checking by the compiler of source code against the specified MISRA-C:2012 rules.

[Mandatory rules] **17.6**

[Required rules] **8.14 , 9.4 , 9.5, 13.1, 18.7, 21.11**

[Advisory rules] **21.12**

The following are the numbers of MISRA-C:2012 rules against which each revision of compilers can check source code for compliance.

Rule classification (number of rules in the standard)	*V1.02.00*	*V1.03.00*	*V1.04.00*	*V1.05.00*	***V1.06.00***
Mandatory rules (16)	3	3	4	6	**7**
Required rules (108)	31	58	76	80	**86**
Advisory rules (32)	7	21	23	25	**26**
Total number of rules (156)	41	82	103	111	**119**

2.3 Feature for detecting illegal indirect function calls [Professional edition]

A feature for detecting indirect function calls to illegal addresses has been added.

The compiler checks the branch destination addresses of indirect calls of functions through the following steps and calls an error function if it detects a problem.

1. The compiler automatically extracts functions which may be indirectly called in the C-source program and the linker consolidates the information to generate a list of functions in the executable files.

2. The compiler inserts processing for calling the “__control_flow_integrity” checking function immediately before calls of indirect functions it finds in analyzing the C-source program. The branch destination address of the call of the indirect function is passed as an argument to this checking function.

3. Within the checking function at the time of execution, the branch destination address given as the argument is checked to see if it is included in the list of functions. If the address is not included, it is regarded as an illegal indirect function call, so the “__control_flow_chk_fail” error function will be called.

The following C-source program shows an example of an illegal indirect function call.

```
extern void func1(void);
extern void func2(void);

void (*fp)(void) = &func1;

void main(void) {
   (*fp)();              // Function func1 is indirectly called.
   func2();              // Function func2 is directly called.
}
```

Since the address of func1 is acquired in the fourth line, the call of func1 is regarded as indirect and added to the list of functions.
Since a function pointer fp is used to indirectly call func1 in the seventh line, the compiler acquires the value of fp immediately before this call and generates code to call the "__control_flow_integrity" checking function by specifying the acquired value as an argument. Within the checking function, a check of whether the value specified by the argument (the address of func1 in the case of normal operation) is included in the list of functions is conducted and subsequent operation is as follows.

- If the list includes the value, the compiler continues to process the C-source program.
- If the list does not include the value, the "__control_flow_chk_fail" error function is called.

Illegal indirect function calls can thus be detected in the way described above.
Since the call of func2 is direct, the eighth line is not detected.
Specify the following options to enable this feature.

[compile option]

-control_flow_integrity

This option selects the generation of code for detecting illegal indirect function calls.

[linker option]

-cfi

This option selects the generation of a list of functions for use in detecting illegal indirect function calls.

The following linker options have also been added in association with this feature.

- **-cfi_add_func**
  This option adds the symbols or addresses of functions which are specified as arguments to the list of functions.
- **-cfi_ignore_module**
  This option selects the non-addition of the addresses of functions included in a file which is specified as an argument to the list of functions.
- **-show=cfi**
  This option selects the output of the contents of the list of functions to the list file which is output in response to specifying the -list option.

2.4 Specifying more than one vector table address for a hardware interrupt handler

#pragma interrupt can be used to specify more than one vector table address for a single function.

```
<Example>
#pragma   interrupt   func(vect=2, vect=8)
 or
#pragma   interrupt   func(vect=2)
#pragma   interrupt   func(vect=8)
```

2.5 Accessing indirect references by pointers in 1-byte units

To support the porting of code written for the CA78K0R compiler, the CC-RL compiler newly supports the -unaligned_pointer_for_ca78k0r option that leads to the generation of code for indirect reference with 1-byte access to types without the volatile qualifier.

2.6 Enhanced optimization

For V1.06.00, optimization has been further enhanced on points (1) to (2), listed and described below.

(1) Handling of bitwise operations

Handling of bitwise operations for data having a narrow bit width has been enhanced.

```
<Example of source code>
void func(unsigned char *t, unsigned char i, unsigned char j, unsigned char v) {
        unsigned char *p = &t[i & 0xff];
        unsigned char m = j >> (i & 0xf);
        *p = v ? m : ~m;
}
```

```
<Code generated by V1.05.00 (1/2) >
_func:
        .STACK _func = 8
        push bc
        push hl
        movw hl, ax
        mov a, b
        mov [sp+0x00], a
        mov a, c
        and a, #0x0F
        mov b, a
        mov a, [sp+0x00]
        shrw ax, 8+0x00000
        cmp0 b
        bz $.BB@LABEL@1_2
.BB@LABEL@1_1: ; entry
        shrw ax, 0x01
        dec b
        bnz $.BB@LABEL@1_1
.BB@LABEL@1_2: ; entry
        movw [sp+0x00], ax
        clrb a
```

```
<Code generated by V1.06.00 (1/2) >
_func:
        .STACK _func = 8
        push bc
        push hl
        movw hl, ax
        mov a, b
        mov [sp+0x00], a
        mov a, c
        and a, #0x0F
        mov b, a
        mov a, [sp+0x00]
        shrw ax, 8+0x00000
        cmp0 b
        bz $.BB@LABEL@1_2
.BB@LABEL@1_1: ; entry
        shrw ax, 0x01
        dec b
        bnz $.BB@LABEL@1_1
.BB@LABEL@1_2: ; entry
        movw bc, ax
        mov a, e
```

```
<Code generated by V1.05.00 (2/2) >
        movw bc, ax
        mov a, e
        cmp0 a
        bnz $.BB@LABEL@1_4
.BB@LABEL@1_3: ; bb22
        movw ax, [sp+0x00]
        mov a, #0xFF
        xch a, x
        xor a, #0xFF
        xch a, x
        movw bc, ax
.BB@LABEL@1_4: ; bb25
        mov a, [sp+0x02]
        shrw ax, 8+0x00000
        addw ax, hl
        movw de, ax
        mov a, c
        mov [de], a
        addw sp, #0x04
        ret
```

```
<Code generated by V1.06.00 (2/2) >
        cmp0 a
        mov a, c
        mov b, a
        bnz $.BB@LABEL@1_4
.BB@LABEL@1_3: ; bb22
        xor a, #0xFF
        mov b, a
.BB@LABEL@1_4: ; bb25
        mov a, [sp+0x02]
        shrw ax, 8+0x00000
        addw ax, hl
        movw de, ax
        mov a, b
        mov [de], a
        addw sp, #0x04
        ret
```

(2) **Alias analysis**

Optimization by alias analysis has been enhanced. Alias analysis was implemented in V1.05.00 and is enabled by specifying the -Oalias=ansi option.

In V1.05.00, alias analysis is disabled when the -Omerge_files option is specified. However, in V1.06.00, it is enabled even when the -Omerge_files option is specified.

When optimization by alias analysis is enabled, the effect is the same as in V1.05.00.

```
<Example of source code>
struct tag1 {
   char member1;
   int    member2;
   long long member3;
} StructArray[2];

struct tag2 {
   short index0;
   short index1;
   short index2;
};

void func(struct tag2 *p) {
   StructArray[p->index1].member1 = 1;
   StructArray[p->index1].member2 = 2;
   StructArray[p->index1].member3 = 3;
}
```

Although the address of StructArray[p->index1] would be calculated three times in V1.04.00, it is only calculated once in V1.06.00.

<Code generated by V1.04.00>

```
movw de, ax
movw bc, #0x000C
movw ax, [de+0x02]
mulh
movw bc, ax
mov LOWW(_StructArray)[bc], #0x01
movw ax, [de+0x02]
movw bc, #0x000C
mulh
addw ax, #LOWW(_StructArray+0x00002)
movw hl, ax
onew ax
incw ax
movw [hl], ax
movw ax, [de+0x02]
movw bc, #0x000C
mulh
addw ax, #LOWW(_StructArray+0x00004)
movw de, ax
clrw ax
movw [de+0x06], ax
movw [de+0x04], ax
movw [de+0x02], ax
movw ax, #0x0003
movw [de], ax
ret
```

<Code generated by V1.06.00>

```
push hl
movw de, ax
movw bc, #0x000C
movw ax, [de+0x02]
mulh
addw ax, #LOWW(_StructArray)
movw [sp+0x00], ax
movw de, ax
movw ax, de
mov [de+0x00], #0x01
incw ax
incw ax
movw de, ax
onew ax
incw ax
movw [de], ax
movw ax, [sp+0x00]
addw ax, #0x0004
movw de, ax
clrw ax
movw [de+0x06], ax
movw [de+0x04], ax
movw [de+0x02], ax
movw ax, #0x0003
movw [de], ax
pop hl
ret
```

2.7 Upper limits on usable amounts of memory

The amounts of memory on the host computer that are usable by the CC-RL compiler have been expanded.

- 2 Gbytes with the 32-bit and 64-bit OSs [V1.05.00 and earlier versions]
- **3 Gbytes** with the 32-bit OS and **4 Gbytes** with the 64-bit OS [V1.06.00 and later versions]

2.8 Control of messages

The **-change_message** compiler option, which is used to change warning messages to error messages, has been added to avoid oversights in the form of warning messages not being noticed.

In addition, the -no_warning_num compiler option that controls the output of warning messages can now specify messages with numbers from 0510000.

- W0520000 to W0559999 can be controlled. [V1.05.00 and earlier versions]
- **W0510000 to W0559999** can be controlled. [V1.06.00 and later versions]

2.9 Fixing of the record length of the Intel HEX file

The **-fix_record_length_and_align** option, which causes the output addresses of Intel HEX files (.hex)

and Motorola S-record files (.mot) to have a specified alignment and be output with a fixed record length, has been added. Since with this option a HEX file is always output with a fixed record length, it can improve the efficiency of work such as comparing HEX files.

The -byte_count option has also been extended to allow its specification along with the -form=stype option.

2.10 Addition of a message at linkage

In V1.05.00 and earlier versions, the warning code W0561322 was output if sections with different alignment conditions but the same names were linked. In V1.06.00, warning code **W0561331** is output when sections with the same names but different alignment conditions, with the condition for one not being a multiple of that of the other, are linked.

W0561322: Section alignment mismatch : " *section*"

W0561331 : Section alignment is not adjusted : " *section* "

In both cases, specification of the greater value of the alignment condition is enabled and the sections are linked.

W0561322 can be ignored since it does not indicate a problem with operation. However, since W0561331 indicates a possible problem with operation, the alignment conditions must be reviewed.

2.11 Rectified points for caution

The points for caution on the following four items no longer apply. For details, refer to Tool News.

- Relational Operators in the Control Expressions of switch Statements (CCRL#015)
- Using a goto statement to move to a label in a switch statement (CCRL#016)
- When a function has multiple arguments and also has assignment or comparison between formal arguments (CCRL#017)
- Loop statements with loop-control variables in which constants are used as the condition for ending the loop (CCRL#018)

2.12 Other changes and improvements

The generation of an internal error in response to building has been corrected.

Chapter 3 Points for Caution

Please refer to the user's manual for caution regarding V1.06.00 of the CC-RL compiler.

Revision History

Rev.	Date	Description	
		Page	Summary
Rev.1.00	Dec 20,2017		First Edition issued
Rev.1.01	Jan 16,2021	9	The error in rectified points for caution is corrected.

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 or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving 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, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. 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 shall be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
6. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended 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; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or 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 damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, 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 and impractical, you are responsible for evaluating the safety of the final products or systems 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. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. Renesas Electronics products and technologies shall 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 shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.
(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1 October 2020)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/.