[Notes]
C Compiler Package for RL78 Family (CCRL#035-#037)

R20TS1145EJ0100
Rev.1.00
Jul. 05, 2025

Outline

When using the CC-RL C Compiler Package for RL78 Family, note the following point.

1. Note on Using the -dbl_size=8 Option (CCRL#035)
2. Note on Using Multiple Section Address Operators (CCRL#036)
3. Note On Using the extern Declaration of a const Variable with a far attribute (CCRL#037)

*The number in the parentheses is the identification number of the note.

1. Note on Using the -dbl_size=8 Option (CCRL#035)

1.1 Applicable Products

CC-RL V1.13.00 to V1.15.00

1.2 Details

If an interrupt occurs during the execution of a runtime library for that involves multiplying the double-precision floating-point type (double type), the result of multiplication may be incorrect.

1.3 Conditions

The problem might occur when both conditions (1) and (2) listed below are met.

(1) The -dbl_size=8 option is specified.
Note: The -dbl_size=8 option is only available when the -cpu=S3 option is specified.

(2) A program which calls the _COM_dmul runtime library for multiplication of the double type has been written.
Note: The _COM_dmul runtime library may be called when a double-type multiplication or division has been written or any of the following standard libraries has been used.

Applicable functions
acos, asin, atan, atan2, cos, sin, tan, cosh, sinh, tanh, exp, frexp, ldexp, log, log10, pow, sqrt, fmod, acosh, asinh, atanh, log1p, scalbn, scalbln, acosl, asinl, atanl, atan2l, cosl, sinl, tanl, acoshl, asinhl, atanhl, coshl, sinhl, tanhl, expl, frexpl, ldexpl, logl, log10l, log1pl, scalbnl, scalblnl, powl, sqrtl, fmodl, printf, scanf, snprintf, sprintf, sscanf, vprintf, vscanf, vsnprintf, vsprintf, vsscanf, atof, _COM_atof_f, strtod, _COM_strtod_ff, strtold, _COM_strtold_ff

You can check whether the problem applies or not by specifying -list or -show=symbol to the linker and see if the generated link map file contains the symbol __COM_dmul as shown below.

Example of the output of a link map file
FILE=_COM_dmul 000020a1 000022dc 23c __COM_dmul

[Example]

An example of commands and code that will cause the problem is given below.

ccrl -cpu=S3 -dbl_size=8 tp.c // (1)

```
/* tp.c */
double d;
void main(void) {
  d = d * 7.0;   // (2)
}
```

1.4 Workaround

Prohibit interrupts by calling an intrinsic function such as __DI() before calling the __COM_dmul runtime library.

Alternatively, specify -dbl_size=4 to handle the double type as the single-precision floating-point type.

1.5 Schedule for Fixing the Problem

The problem will be fixed in upcoming versions. The release dates of the new versions have not been determined.

2. Note on Using Multiple Section Address Operators (CCRL#036)

2.1 Applicable Products

CC-RL V1.15.00

2.2 Details

When multiple section address operators (__sectop or __secend) have been written, the result of access may be incorrect.

2.3 Conditions

Access to the correct address might not be possible when both conditions (1) and (2) listed below are met.

(1) Access to an address acquired with the use of a section address operator (__sectop or __secend) has occurred multiple times within the same function.
(2) Multiple addresses acquired under the above condition (1) include different higher-order 16 bits.
Note: You can check the addresses in the link map file (extension: .map) which is output by specifying the -list option at the time of linking.

The __sectop operator acquires the start address indicated with START. The __secend acquires the end address indicated by END and incremented by 1.

For example, for the link map file shown below, __sectop (“ram_text_f”) is 0x00003000 and __secend (“ram_text_f”) is 0x0000302a (0x00003029+1). The higher-order 16 bits are 0x0000 for both addresses. Similarly, __sectop (“ram_text_fR”) is 0x000ff000 and __secend (“ram_text_fR”) is 0x000ff02a (0x000ff029+1). Again, the higher-order 16 bits of both are 0x000f.

Example of the output of a link map file

```
*** Mapping List ***

SECTION                 START       END         SIZE    ALIGN
ram_text_f
                        00003000    00003029    2a      1
ram_text_fR
                        000ff000    000ff029    2a      1
```

[Example]

An example of commands and code that will cause the problem is given below.

ccrl -cpu=S3 tp2.c

```
/* tp2.c */
#define SIZE  42

void test(void){
  int i;
  unsigned char __far *dst, *src;
  src = __sectop("ram_text_f");
  dst = __sectop("ram_text_fR");
  for(i = 0; i < SIZE; i++) {
    *dst++ = *src++;  // (1) Access is to addresses based on one acquired by
__sectop.
  }
}
```

The higher-order 16 bits of addresses acquired by __sectop are different in sections ram_text_f and ram_text_fR. (2)

2.4 Workaround

Specify the address acquired by the section address operator as the initial value of a volatile-qualified global variable and access the address via that global variable.

Workaround

```
#define SIZE 42

static unsigned char __far* const volatile secTopRamTextF =
__sectop("ram_text_f");
static unsigned char __far* const volatile secTopRamTextFR =
__sectop("ram_text_fR");
void test(void){
  int i;
  unsigned char __far *dst, *src;
  src = secTopRamTextF;
  dst = secTopRamTextFR;
  for(i = 0; i < SIZE; i++) {
    *dst++ = *src++;
  }
}
```

2.5 Schedule for Fixing the Problem

The problem will be fixed in upcoming versions. The release dates of the new versions have not been determined.

3. Note on Using the extern Declaration of a const Variable with a far attribute (CCRL#037)

3.1 Applicable Products

CC-RL V1.15.00

3.2 Details

When the extern declaration is used with a const variable having the far attribute, incorrect reference to the value of the variable may be occurred.

3.3 Conditions

Correct reference to the value of the variable might not be possible when all conditions (1) to (4) listed below are met.

(1) far-attribute global variables qualified as const are present.
Note: This also applies to cases when the global variables have the far attribute from the -far_rom option as well as from the __far qualifier.

(2) Multiple variables among those global variables are used within the same function.
(3) At least one variable among those in use is declared as extern.
(4) Some variables in use are allocated to different sections and have different higher-order 16 bits of address.
Note: Note that this condition is only satisfied if features from among those listed below are used.

- #pragma section
- One of the following linker options
  - -rom
  - -split_section
  - -stride_dsp_memory
  - -stride_self_area
  - -stride_ocdtr_area
  - -stride_ocdhpi_area

You can check the addresses of the variables in the link map file (extension: .map) which is output by specifying the -list and -show=symbol options at the time of linking.

For example, in the link map file shown below, the address of the gv1 variable is 0x0000001a and its higher-order 16 bits are 0x0000. In a similar way, we can see that the address of the gv2 variable is 0x00010000 and its higher-order 16 bits are 0x0001.

Example of the output of a link map file

```
*** Symbol List ***

SECTION=
FILE=                                   START       END      SIZE
  SYMBOL                                ADDR        SIZE     INFO        COUNTS   OPT

SECTION=.constf
FILE=C:¥test1.obj
                                        0000001a    0000001b           2
  _gv1
                                        0000001a           2    data ,g          *

SECTION=$.constf_part1
FILE=C:¥test2.obj
                                        00010000    00014001        4002
  _gv2
                                        00010000           2    data ,g          *
```

[Example]

An example of commands and code that will cause the problem is given below.

ccrl -cpu=S3 tp3.c

```
/* tp3.c */
const int __far gv1 = 0x100;  // (1)
extern const int __far gv2;    // (1) (3)
int test1(void){
  return gv1 + gv2;   // (2)
}
```

It is assumed that variables gv1 and gv2 have different higher-order 16 bits of address. (4)

In this example, correct reference to the value of the gv1 variable is not possible.

3.4 Workaround

Use one of the following methods so that the variables used within the same function have the same higher-order 16 bits of address.

- Modify the start addresses of the sections where the variables are allocated by using the -start linker option.
- Allocate the variables within the same section.

3.5 Schedule for Fixing the Problem

The problem will be fixed in upcoming versions. The release dates of the new versions have not been determined.

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Jul.05.25	-	First edition issued

Corporate Headquarters

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

Trademarks

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/

TS Colophon 4.3