[Note]

C/C++ Compiler Package for RX Family (No.55-58)

R20TS0649EJ0100
Rev.1.00
Jan. 16, 2021

Overview

When using the CC-RX Compiler package, note the following points.

1. Using rmpab, rmpaw, rmpal or memchr intrinsic functions (No.55)
2. Performing the tail call optimization (No.56)
3. Using the -ip_optimize option (No.57)
4. Using multi-dimensional array (No.58)

Note: The number following the note is an identification number for the note.

1. Using rmpab, rmpaw, rmpal or memchr intrinsic functions (No.55)

1.1 Applicable products

CC-RX V2.00.00 to V3.02.00

1.2 Details

The execution result of a program including the intrinsic function rmpab, rmpaw, rmpal, or the standard library function memchr may not be as intended.

1.3 Conditions

This problem may arise if all of the conditions from (1) to (3) are met.

(1) One of the following calls is made:
  (1-1) rmpab or __rmpab is called.
  (1-2) rmpaw or __rmpaw is called.
  (1-3) rmpal or __rmpal is called.
  (1-4) memchr is called.

(2) One of (1-1) to (1-3) is met, and neither -optimize=0 nor -noschedule option is specified.
  (1-4) is met, and both -size and -avoid_cross_boundary_prefetch (Note 1) options are specified.

(3) Memory area that overlaps with the memory area (Note2) read by processing (1) is written in a single function. (This includes a case where called function processing is moved into the caller function by inline expansion.)

Note 1: This is an option added in V2.07.00.
Note 2: The aria pointed to by the third or fourth argument, when one of (1-1) to (1-3) is met, or the first argument, when (1-4) is met.

1.4 Examples

An example of the problem is shown below. The parts corresponding to the error conditions are shown in red.

ccrx tp1.c -isa=rxv1 -optimize=2 // (2)

```
// tp1.c
signed short lhs[128];
signed short rhs[128];
long long test(void){
  long long tmp;
  lhs[0] = 0;     // (3)
  tmp  = __rmpaw(0LL, 128, lhs, rhs);   // (1-2)
  lhs[1] = 0;
  return tmp;
}
```

In this example, writing to lhs[0] moves toward the exit of the function beyond the __rmpaw call.

As a result, the execution result of __rmpaw is not as intended.

1.5 Workaround

You can avoid this problem by one of the following methods.

(a) If any of the conditions from (1-1) to (1-3) apply, do one of the following.
- Specify -noschedule.
- Specify -optimize=0.
- Specify -optimize=1 and do not specify -schedule.

(b) If condition (1-4) applies, specify -speed, or do not specify -avoid_cross_boundary_prefetch.

1.6 Schedule for fixing the problem

This problem will be fixed in CC-RX V3.03.00. This version will be released in January 2021.

2. Performing the tail call optimization (No.56)

2.1 Applicable products

CC-RX V2.00.00 to V3.02.00

2.2 Details

Necessary type conversion may not be performed on the return value of a function.

2.3 Conditions

This problem may arise if all of the conditions from (1) to (4) are met.

(1) Neither -optimize=0 nor -optimize=1 is specified.
(2) There is an integer-type function with a return value of either 1 byte or 2 bytes. (Note 1)
(3) There is an integer-type function whose return value type is the same size as the function (2) but with a different signedness. (Note 1)
(4) In the function of (3), the result of type conversion of the return value of the function of (2) to the return type of the function of (3) is returned.
*:Implicit type conversion is also included.

Note 1: 1- or 2-byte integer type includes the boolean type, enumerated type when -auto_enum is specified and int type when -int_to_short is specified. The boolean type is regarded as a signed 1-byte type.

2.4 Examples

An example of the problem is shown below. The parts corresponding to the error conditions are shown in red.

ccrx tp2.c -isa=rxv1 -optimize=2 // (1)

```
// tp2.c
extern unsigned char callee();    /* (2) */
signed char caller(){             /* (3) */
  signed char returnValue;
  returnValue = callee();
  return returnValue;             /* (4) */
}
```

In this example, the return value of callee() is supposed to be sign-extended in caller() before returning, but this is not done and the upper bits are returned as 0.

2.5 Workaround

You can avoid this problem by one of the following methods. The workarounds are shown in blue.

(a) Specify the -optimize=0 or optimize=1 option.
(b) Assign the return value of the applicable function call to a volatile-qualified automatic variable before it is passed to the return statement.

```
// tp2.c

extern unsigned char callee();      /* (2) */

signed char caller(){               /* (3) */

  volatile signed char returnValue;  /* (b) */

  returnValue = callee();

  return returnValue;                /* (4) */

}
```

(c) Change the type of the return value of the caller function to a 4-byte type.

```
// tp2.c
extern unsigned char callee();    /* (2) */
signed long caller(){             /* (c) */
  signed long returnValue;        /* (c) */
  returnValue = callee();
  return returnValue;             /* (4) */
}
```

2.6 Schedule for fixing the problem

This problem will be fixed in CC-RX V3.03.00. This version will be released in January 2021.

3. Using the -ip_optimize option (No.57)

3.1 Applicable products

CC-RX V2.00.00 to V3.02.00

3.2 Details

When the -ip_optimize option is used, access to static variables may be deleted incorrectly.

3.3 Conditions

If all of the conditions from (1) to (8) are met, access to a variable in condition (7) may be deleted incorrectly.

(1) -ip_optimize or -whole_program is specified. (Note 1)

(2) Neither -optimize=0 nor -optimize=1 is specified.

(3) There is a structure-type or union-type having a pointer-type member.

(4) The pointer-type member in (3) is not const-qualified.

(5) There is a const-qualified static variable(Note 2) of the structure-type or union-type in (3).

(6) The initial value of the pointer-type member (3) of the static variable in (5) is the address of a variable.

(7) The variable with the address in (6) is a static variable(Note 2) that is not const-qualified.

(8) There is a const-qualified pointer-type static variable(Note 2) whose initial value is the address of the static variable in (5).

Note 1: When -whole_program is specified, -ip_optimize is also implicitly specified.

Note 2: A static variable corresponds to a global variable or a ‘`static`’ variable.

3.4 Examples

An example of the problem is shown below. The parts corresponding to the error conditions are shown in red.

ccrx -isa=rxv2 -ip_optimize tp.c (1) (2)

```
/* tp.c */
int GGG;                        /* (7) */
typedef struct {                /* (3) */
  int* mmm;                     /* (4) */
}Str;
const Str SSS = {               /* (5) */
  &GGG                          /* (6) */
};
const Str* PPP = &SSS;          /* (8) */


int func(void) {
  GGG = 1;
  *(PPP->mmm) = 2;
  return GGG;
}
```

In this example, although function func() is supposed to return 2 because PPP->mmm points to the address of the variable GGG, it returns 1.

3.5 Workaround

You can avoid this problem by one of the following methods:

(a) Do not specify either -ip_optimize or whole_program.

(b) Specify -optimize=0 or -optimize=1.

(c) Remove the const qualifier from the structure-type or union-type static variable in condition (5).

(d) Remove the const qualifier from the pointer-type static variable in condition (8).

3.6 Workaround

This problem will be fixed in CC-RX V3.03.00. This version will be released in January 2021.

4. Using multi-dimensional array (No.58)

4.1 Applicable products

CC-RX V2.00.00 to V3.02.00

4.2 Details

The execution result of a program including a multi-dimensional array with three or more dimensions may not be as intended.

4.3 Conditions

This problem may arise if all of the conditions from (1) to (8) are met.

(1) Neither -optimize=0 nor -optimize=1 is specified.
(2) A multi-dimensional array with three or more dimensions exists.
(3) The multi-dimensional array in (2) is a 1- or 2-byte integer-type array.
(4) The multi-dimensional array in (2) is neither volatile-qualified nor __evenaccess-qualified.
(5) The multi-dimensional array in (2) includes two or more elements to which integer constants are set by either of the following methods:
(5-a) integer constants are assigned by assignment statements.
(5-b) The multi-dimensional array is an automatic variable and the initial values at the time of declaration are integer constants.
(6) When integer constants are set by the method (5-b), the number of initial values for the initialization is insufficient for the number of elements in the multi-dimensional array. (Note 1)
(7) The elements (5) to which integer constants are set includes two adjacent elements whose indices other than the least significant index are different. (Note 2)
(8) The processes for setting integer constants for the adjacent two elements in (7) are in the same function.

Note 1: For example, the following matches the condition because there are only seven initial values (which is insufficient) for the number of elements 2×2×2 = 8.
signed char array[2][2][2] = {{{1, 2}, {3, 4}}, {{5, 6}, {7}}};
Note 2: For example, in a three-dimensional array data[][][] with 5×5×5 elements, all the following element combinations match the condition "two adjacent elements whose indices other than the least significant index are different".
- Combination of data[0][0][4] and data[0][1][0]
- Combination of data[0][4][4] and data[1][0][0]
- Combination of data[3][4][4] and data[4][0][0]

4.4 Examples

An example of the problem is shown below. The parts corresponding to the error conditions are shown in red.

ccrx -isa=rxv2 tp.c (1)

```
/* tp.c */
unsigned char aaa[2][1][3];  /* (2)(3)(4) */
unsigned char XXX, YYY, ZZZ;

void func(void) {

  if (aaa[1][0][0] == 0) {
    XXX++;
  }

  aaa[0][0][2] = 0;          /* (5-a)(7)(8) */
  aaa[1][0][0] = 100;        /* (5-a)(7)(8) */

  YYY = aaa[0][0][2];
  ZZZ = aaa[1][0][0];
}
```

In this example, the value of an array element aaa[1][0][0] (before 100 is assigned) is assigned to the variable ZZZ, although 100 (after it is assigned to aaa[1][0][0]) is supposed to be assigned.

4.5 Workaround

You can avoid this problem by one of the following methods:

(a) Specify -optimize=0 or -optimize=1.

(b) Add either volatile qualifier or __evenaccess qualifier to the applicable multi-dimensional array.

4.6 Schedule for fixing the problem

This problem will be fixed in CC-RX V3.03.00. This version will be released in January 2021.

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Jan.16.21	-	First edition issued

Corporate Headquarters

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

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/

Trademarks

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

TS Colophon 4.2