[Notes]
C/C++ Compiler Package for RX Family

R20TS0530EJ0100
Rev.1.00
Jan. 16, 2020

Outline

When using the C/C++ compiler package for RX family CC-RX, note the following point.

1. Using the -alias=ansi option (No.54)

* The number after the note is the note's identification number.

1. Using the -alias=ansi Option (No.54)

1.1 Applicable Products

CC-RX V2.07.00, V2.08.00, V3.00.00, V3.01.00

1.2 Details

When the -alias=ansi optional function is used, access to a structure- or union-type variable may be deleted improperly.

1.3 Conditions

This problem arises if the following conditions are all met:

(1) Options to enable -alias=ansi are specified. (One of the following.)

(1-1) -optimize=max is specified and -alias=noansi is not.
(1-2) -optimize=2 and -alias=ansi are specified.
(1-3) -alias=ansi is specified without the -optimize option being specified.

(2) Either of the following variables, (2-1) or (2-2), is used.

(2-1) Structure-type variable that satisfies all of the following conditions:

(2-1-a) The structure-type variable has an array-type member.
(2-1-b) One of the elements of (2-1-a) is referenced three or more times in the function.
(2-1-c) Both reference methods (reference by the [] operator and reference by the * operator) are used in (2-1-b).
(2-1-d) The reference in (2-1-b) involves both a value read and assignment.

(2-2) Union-type variable that satisfies all of the following conditions:

(2-2-a) The union-type variable has array-type members of different element types.
(2-2-b) An area-overlapping element of (2-2-a) is referenced three or more times in the function.
(2-2-c) There are two or more references by the [] operator in (2-2-b).
(2-2-d) The reference in (2-2-b) involves both a value read and assignment.
(2-2-e) References in (2-2-b) contains a reference to a different member.

(3) A structure- or union-type variable that is no qualified with volatile is used.
(4) A structure- or union-type variable is a static variable.

1.4 Examples

Below is an example of the problem. The parts corresponding to the conditions are shown in red.

■ Example 1: When a structure-type is used.

ccrx tp1.c -isa=rxv2 -optimize=2 -alias=ansi // Condition (1-2)

```
// tp1.c
#include<stdio.h>
struct {       //Structure-type global variable
               //not qualified with volatile Condition(3)(4)
 int ary[10];  //Has an array-type member (2-1-a)
}data = {0};
void main (void) {
 data.ary[0] = 1;       //First reference (2-1-b)
                        //Use of the [] operator (2-1-c)
                        //and assignment (2-1-d)
  data.ary[1] = 2;
 *(data.ary + 0) = 2;   //Second reference (2-1-b)
                        //Use of the * operator (2-1-c)
                        //and assignment (2-1-d)
  *(data.ary + 1) = 3;
 printf("%d¥n",data.ary[0]);   //Third reference (2-1-b)
                               //Use of the [] operator (2-1-c)
                               //and value read (2-1-d)
}
```

The printf execution resulted in "1" although it should be "2".

■ Example 2: When a union-type is used.

ccrx tp2.c -isa=rxv2 -optimize=2 -alias=ansi // Condition (1-2)

```
// tp2.c
#include<stdio.h>
union{          //Union-type global variable
                //not qualified with volatile Condition(3)(4)
  int i[2];     //int-type array member (2-2-a)
  short s[4];   //short-type array member (2-2-a)
} un;
int g;
void main (void) {
 un.s[0] = 1;    //First reference (2-2-b)
                 //Use of the [] operator (2-2-c) and assignment (2-2-d)
 g = un.i[0];    //Second reference (2-2-b)
                 //Use of the [] operator (2-2-c) and value read (2-2-d)
                 //and reference to a different member (2-2-e)
 un.s[0] = 2;    //Third reference (2-2-b)
                 //Use of the [] operator (2-2-c) and assignment (2-2-d)
  printf("%d¥n",g);
}
```

The printf execution resulted in an undefined value although it should be "1".

1.5 Workaround

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

(1) Specify -alias=noansi.

(2) In the case of a structure-type variable (select one of the following):

- Add the volatile qualifier to the structure-type variable.
- Add the volatile qualifier to the array members.
- Only use references by the [] operator.

(3) In the case of a union-type variable (select one of the following):

- Add the volatile qualifier to the union-type variable.
- Add the volatile qualifier to all array members that refer to an overlapping area.
- Use references by the * operator.
- Limit the number of uses of the [] operator to one.

1.6 Schedule for Fixing the Problem

This problem will be fixed in CC-RX V2.08.01 and V3.02.00. (Scheduled to be released on January 20.)

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Jan.16.20	-	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/