[Notes]
C Compiler Package for RH850 Family

R20TS0528EJ0100
Rev.1.00
Jan. 16, 2020

Outline

When using the C compiler package for RH850 family CC-RH, note the following point.

1. Using the -Xalias=ansi option (No.28)

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

1. Using the -Xalias=ansi option (No.28)

1.1 Applicable Products

CC-RH V1.06.00, V1.07.00, V1.07.01, V2.00.00, V2.01.00

1.2 Details

When the -Xalias=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) -Xalias=ansi is specified.
(2) -O, -Osize, or -Ospeed is specified.
(3) Either of the following variables, (3-1) or (3-2), is used.
(3-1) Structure-type variable that satisfies all of the following conditions:

(3-1-a) The structure-type variable has an array-type member.
(3-1-b) One of the elements of (3-1-a) is referenced three or more times in the function.
(3-1-c) Both reference methods (reference by the [] operator and reference by the * operator) are used in (3-1-b).
(3-1-d) The reference in (3-1-b) involves both a value read and assignment.

(3-2) Union-type variable that satisfies all of the following conditions:

(3-2-a) The union-type variable has array-type members of different element types.
(3-2-b) An area-overlapping element of (3-2-a) is referenced three or more times in the function.
(3-2-c) There are two or more references by the [] operator in (3-2-b).
(3-2-d) The reference in (3-2-b) involves both a value read and assignment.
(3-2-e) References in (3-2-b) contains a reference to a different member.

(4) A structure- or union-type variable that is not qualified with volatile is used.
(5) 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.

ccrh -O -Ounroll=10 tp.c -Xalias=ansi // Conditions (1)(2)

```
#include<stdio.h>
struct {       //Structure-type global variable
               //not qualified with volatile Condition(4)(5)
 int ary[10];  //Has an array-type member (3-1-a)
} data = {0};
void main (void) {
  int cnt = 0;
  for (cnt = 0; cnt < 10; cnt++) {
    data.ary[cnt] = cnt + 1;    //First reference (3-1-b)
                                //Use of the []operator (3-1-c)
                                //and assignment (3-1-d)
  }
  for (cnt = 0; cnt < 10; cnt++) {
   *(data.ary+cnt) = cnt + 2;   //Second reference (3-1-b)
                                //Use of the * operator (3-1-c)
                                //and assignment (3-1-d)
  }
 printf("%d¥n", data.ary[0]);  //Third reference (3-1-b)
                              //Use of the [] operator (3-1-c)
                              //and value read (3-1-d)
}
```

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

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

ccrh -O tp.c -Xalias=ansi // Condition (1)(2)

```
#include<stdio.h>
union{          //Union-type global variable not qualified with volatile
                //Condition(4)(5)
  int i[2];     //int-type array member (3-2-a)
  short s[4];   //short-type array member (3-2-a)
} un;
int g;
void main (void) {
 un.s[0] = 1;     //First reference (3-2-b)
                  //Use of the [] operator (3-2-c)  and assignment (3-2-d)
 g = un.i[0];     //Second reference (3-2-b)
                  //Use of the [] operator (3-2-c), value read (3-2-d)
                  //and reference to a different member (3-2-e)
 un.s[0] = 2;     //Third reference (3-2-b)
                  //Use of the [] operator (3-2-c)
                  //and assignment (3-2-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 -Xalias=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-RH V2.02.00. (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/