[Notes]
C Compiler Package for RH850 Family

R20TS0452EJ0100
Rev.1.00
Jul. 01, 2019

Outline

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

1. Comparison expressions in a loop (No.25)

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

1 Comparison Expressions in a Loop (No.25)

1.1 Applicable Products

CC-RH V1.00.00 to V2.01.00

1.2 Details and Conditions

Comparison expression in (4) may use an invalid value if all the following conditions are met.
Also, these conditions might be met internally after the compiler performs optimization.

- Conditions

(1) The -Ospeed or -Osize option is specified.
(2) There is a loop in the program.
(3) The loop in (2) contains a loop induction variable (Note 1) that satisfies all the following conditions:
  (3-1) The type is signed char, unsigned char, signed short, or unsigned short.
  (3-2) Not declared as volatile.
(4) A comparison expression of any of the following patterns exists in the loop in (2) (Note 2).
  Where, i is a loop induction variable that satisfies (3).

X and Y are constants (including expressions that are identified as constants at the time of compilation).

(4-1) Any of the following comparison expressions is used:

- (i > X) && (i < Y)
- (i > X) && (i <= Y)
- (i <= X) || (i > Y)
- (i <= X) || (i >= Y)

In the comparison expressions above, X is a constant value which is 0 or more, and less than value i at the start of the loop.

Y is a constant value greater than X.

(4-2) Any of the following comparison expressions is used:

- (i >= X) && (i < Y)
- (i >= X) && (i <= Y)
- (i < X) || (i > Y)
- (i < X) || (i >= Y)

In the comparison expressions above, X is a constant value which is 1 or more, and value i or less at the start of the loop.

Y is a constant value greater than X.

(4-3) Any of the following comparison expressions is used:

- (i == X) || (i == Y)
- (i != X) && (i != Y)

In the comparison expressions above, X is a constant which is 1 or more, and value i or less at the start of the loop.

Y is a constant value that satisfies Y == X+1.

(5) The condition for ending the loop in (2) is comparison of size (<, <=, >, or >=) between the loop induction variable that satisfies (3) and the expression that does not change its value in the loop.

Note 1: A loop induction variable is a variable which increments or decrements by a fixed value every loop.

Note 2: "&&" and "||" in (4-1) to (4-3) include bitwise operators ("&", "|") and AND or OR by nesting of or describing both an if statement and ternary operator. The case where an induction variable is described on the right side of the comparison is also included. For example, "(i < X) && (i < Y)" and "(X > i) && (Y > i)" are regarded as the same comparison expression.

1.3 Example 1

```
1:  long a;
2:  void func1(void){
3:    signed char i; // Condition(3-1) (3-2)
4:    for (i = 1; i <= 100; i++) { // Condition(2)(5)
5:      if ((i >= 1) && (i <= 10)) { // Condition(4-2)
6:        a++;
7:      }
8:    }
9:  }
```

All:

Condition (2) is met because a "for" loop exists in the func1 function.

Note that variable i is a loop induction variable of the loop.

Line 3:

Condition (3-1) is met because the type of variable i is signed char.

Also, Condition (3-2) is met because it is not declared as volatile.

Line 4:

Condition (5) is met because the condition for ending the "for" loop is comparison of size (<=) between variable i and integer constant 100.

Line 5:

The conditional statement of the if statement meets Condition (4-2) because it becomes (1 <= i) && (i <= 10) when using the same inequality sign.

In this example, the comparison expression on line 5 is invalid.

1.4 Example 2

```
1:  long a;
2:  void func2(void){
3:    signed char i; // Condition(3-1) (3-2)
4:    for (i = 1; i <= 100; i++) { // Condition(2) (5)
5:      if ((i != 1) && (i != 2)) { // Condition(4-3)
6:        a++;
7:      }
8:    }
9:  }
```

All:

Condition (2) is met because a "for" loop exists in the func2 function.

Note that variable i is a loop induction variable of the loop.

Line 3:

Condition (3-1) is met because the type of variable i is signed char.

Also, Condition (3-2) is met because it is not declared as volatile.

Line 4:

Condition (5) is met because the condition for ending the "for" loop is comparison of size (<=) between variable i and integer constant 100.

Line 5:

The conditional statement of the if statement meets Condition (4-3).

In this example, the comparison expression on line 5 is invalid.

1.5 Workaround

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

(1) Specify the -Onothing or -Odefault option.
(2) Change the type of the loop induction variable in Condition (3) to any type other than those listed in Condition (3-1) (ex. long type).
(3) Declare the loop induction variable in Condition (3) as volatile.
(4) Describe the condition for ending the loop in Condition (5) with an equality operator ("==", "!=").

1.6 Schedule for Fixing the Problem

This problem will be fixed in the next version. The release date for the next version has not yet been determined.

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Jul.01.19	-	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

TS Colophon 4.0