RENESAS Tool News

RENESAS TOOL NEWS on February 01, 2015: 150201/tn4

Note on Using the C Compiler Package for RH850 Family

When using the CC-RH C Compiler package for the RH850 Family, take note of the problem described in this note regarding the following point.

- Point to note regarding the use of both judgment of a match and greater or less than for variables (No.1)

Note: The number which follows the description of the precautionary note is an identifying number for the precaution.

---

1. Products Concerned
   CC-RH V1.00.00 to V1.01.00

2. Description
   Within a given function, a given if statement or loop might be resolved wrongly when the if statement or other loop-control expression includes a combination of expressions to compare for a match ("!=" or "==") and for the comparison of size.

3. Conditions
   This problem arises if the following conditions are all met:
   (1) Any from among the -Odefault option, -Osize option, and -Ospeed options is in use.
   (2) A comparison expression "!=" or "==" comparing a constant (Note1) and a variable (Note2) is present.
   Note1: Includes expressions in which the constant is statically known to be a constant.
   Note2: Includes array variables, structure members, and union members.
   (3) An expression that applies "<", ">", "<=", or ">=" to compare the

constant and variable covered by (2), within a function which contains a comparison expression of the type described in (2).

(4) The variable in (2) is not modified by volatile.

(5) The comparison expressions covered by (2) and (3) meet any of the following conditions.

(5-1) The comparison expressions covered by (2) and (3) are connected by "||" or "&&".

(5-2) The conditions of the "if" statement or "?:" operator contain both of the comparison expressions covered by (2) and (3), and a statement and conditional expression ("?"), or statements or conditional expressions, are executed in succession.

(6) There is no another expression between the comparison expressions covered by (2) and (3).

Example of condition:
in a case where the -Odefault option was specified Condition (1)

```
-----------------------------------------------------------------
int g(void);
int f(void)
 {
   int x = g();     // Condition (4)
   if (x == -2 ||   // Condition (2)(5-1)
      x > 1200) {  // Condition (3)(6)
     return 1;
   }
   return -1;
 }
-----------------------------------------------------------------
```

4. Workaround

To avoid this problem, take any of the following steps.

(1) Designate the -Onothing option.

(2) Modify the variable in Condition (2) by declaring it as volatile.

(3) In the comparisons covered by (2) and (3), refer to the dummy volatile variable just before the next expression to be executed.

Application example of the workaround: case of the workaround (3)

```
-----------------------------------------------------------
int g(void);
volatile int dummy;       // Declaration of dummy as a volatile variable
int f(void)
{
  int x = g();
  if (x == -2 ||
     (dummy, x > 1200)) { // Workaround (3)
```

```
    return 1;
  }
  return -1;
}
```

---

5. Schedule for Fixing the Problem
   This problem will be fixed in the next version.
