RENESAS Tool News

RENESAS TOOL NEWS on February 16, 2016: 160216/tn3

A Note on Using the C/C++ Compiler Package for the SuperH RISC engine MCU Family V.9

When using the C/C++ compiler package V.9 for the SuperH RISC engine family of MCUs, take note of the following problems:

- Problem with using the -speed option when code includes nested loops . (SHC-0095)
  Here, SHC-XXXX at the end of each item is a consecutive number for indexing the problem in the compiler concerned.

---

1. Product and Versions Concerned
   C/C++ compiler package for SuperH RISC engine family
   V.9.04 Release 00 to V.9.04 Release 02

2. Description:
   When the -speed option is used, the values of the loop counters of nested loops may be incorrect.

3. Conditions:
   This problem may arise if the following conditions are all met:
   (1) Neither the -optimize=0 nor the -optimize=debug_only option is used.
   (2) The -speed option is used.
   (3) A loop has a loop counter(Note 1) of any of the following types (except if qualified as volatile).
      (a) signed char
      (b) unsigned char
      (c) signed short
      (d) unsigned short
   (4) The loop counter within the loop in (3) is only updated once.
   (5) When the loop counter in (4) is updated, the increase or decrease in its value is a constant.
   (6) The loop in (3) is within another loop (is a nested loop).
   (7) Following exit from the loop in (3), the only reference to the loop counter is within the loop (see note 2) in (3).

Note 1: This refers to the variable which is incremented or decremented
by a fixed integer value on each iteration and is used to
control the continuation of iterations of the loop as long as
the value of the variable is greater than, less than, or equal
to a specified value.
Note 2: "within the loop" means only the statements executed in each
iteration. In the following example, statement A is not "within
the loop".
for (A; B; C){ D }

Example:

```
---------------------------------------------------------------
int a[100];
short u[10];
void func() {
   int i;
   short j;                          // Condition(3)
   for (i = 0, j = 0; i < 10; i++) {  // Condition(6)
      // Condition(7) j is not referenced.
      for (;
          j < u[i]; j++) {          // Conditions(3)(4)(5)
         a[j] = 0;
      }
      // Condition(7) j is not referenced.
   }
   // Condition(7) j is not referenced.
}
---------------------------------------------------------------
```

Executing the inner loop when i> = 1 leads to the loop counter (j) being erroneously set to its initial value "0" in the outer loop.

4. Workaround:
Avoid this problem through any of the following steps.
(1) Qualify the loop counter of the loop in condition (3) as volatile.
(2) Change the type of the loop counter in condition (3) to signed or unsigned long, or to signed or unsigned int.
(3) Make a reference to the loop counter in condition (3) so that condition (7) is not met.
(4) Use the -size or -nospeed option.
(5) Use the -optimize=0 or -optimize=debug_only option.

Example modified where Workaround(3) used:
Add a line outside the loop that includes a reference to the loop counter within the loop corresponded to condition (3).

```
------------------------------------------------------
long i, t1, t2;
int a[100];
short u[10];
volatile short dummy;                  // Workaround (3)
void func() {
    int i;
    short j;                          // condition (3)
    for (i = 0, j = 0; i < 10; i++) { // condition (6)
        for (;
            j < u[i]; j++) {          // conditions (3),(4),(5)
        a[j] = 0;
        }
    }
  dummy = j;                           // Workaround (3)
}
------------------------------------------------------
```

5. Schedule of Fixing the Problems

   We plan to fix these problems in the C/C++ compiler package V.9.04 Release 03 for the SuperH RISC engine family of MCUs.
