RENESAS Tool News

RENESAS TOOL NEWS on July 16, 2004: RSO-M3T-NC308WA-040716D

A Note on Using C Compiler Packages M3T-NC308WA and M3T-NC30WA

Please take note of the following problem in using the M3T-NC308WA and M3T-NC30WA C-compiler packages (with an assembler and integrated development environment):

- On calling the same function in both cases where the controlling expression of an if statement is TRUE and FALSE

---

1. **Products and Versions Concerned**

   M3T-NC308WA V.5.00 Release 1 through V.5.20 Release 1
   for the M32C/80 and M16C/80 series MCUs
   M3T-NC30WA V.5.00 Release 1 through V.5.30 Release 1
   for the M16C/60, M16C/30, M16C/20, M16C/10, and R8C/Tiny series MCUs

2. **Description**

   When the same function is called using the same argument in both the program statements executed if the controlling expression of an if statement is satisfied and if not satisfied, the return value from the function may become incorrect.

   2.1 Conditions

   This problem occurs if the following seven conditions are satisfied:

   (1) Optimizing option "-OR" is used at compilation.

   (2) An if statement exist whose controlling expression makes the comparison of a parameter in a calling function or an auto variable with a 0 (zero).

   (3) The type of the parameter or variable in (2) is any of those; char, signed char, unsigned char, int, unsigned int, short, and unsigned short
   NOTES:

   1. The problem arises regardless of whether the

parameter or variable is qualified as volatile or not.

2. The problem arises if an address operator is used as the parameter or variable.

(4) In both the program statements, A and B, that are executed if the controlling expression of the if statement in (2) is satisfied and if not satisfied respectively, the same function, C, is called.

(5) The return value from the function C is of type int, float, or pointer.

(6) The function C does not have any argument passed to the stack.

(7) In both the program statements A and B, the processing from the conditional jump to the function call is the same.

2.2 Example

```
-------------------------------------------------
char sub(void);              // Conditions (5) and (6)

char func(int dummy, char c)
{
   if (c == 0) {           // Conditions (2) and (3)
         return sub();    // Conditions (4) and (7)
   }

   sub();                 // Conditions (4) and (7)
   return 0;
}
-------------------------------------------------
```

3. Workaround

In either the line immediately before the processing executed if the controlling expression is satisfied or the one executed if the controlling expression is not satisfied, put a dummy asm function.

```
-------------------------------------------------
char sub(void);
```

```
char func(int dummy, char c)
{
   if (c == 0) {
         asm();          // Dummy asm()
         return sub();
   }

   sub();
   return 0;
}
```

---

4. **Schedule of Fixing the Problem**
   We plan to fix this problem in our next release of the product.
