RENESAS Tool News

RENESAS TOOL NEWS on July 16, 2005: RSO-M3T-NC308WA_2-050716D

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

--On Using Optimizing Option "-OR"--

Please take note of the following problem in using the C compiler packages M3T-NC308WA and M3T-NC30WA (they are used for the M16C family of MCUs):

- On using optimizing option "-OR"

1. **Products and Versions Concerned**
   - M3T-NC308WA V.5.00 Release 1 through V.5.20 Release 02
     (for the M32C/90, M32C/80 and M16C/80 series of MCUs)
   - M3T-NC30WA V.5.00 Release 1 through V.5.30 Release 02
     (for the M16C/60, M16C/30, M16C/20, M16C/10, M16C/Tiny and R8C/Tiny series of MCUs)

2. **Description**
   If source files containing function calls are compiled with the -OR optimizing option selected, an application error may arises, resulting in the compiler being forcefully shut down.
   Note that if the conditions described below are satisfied, compilation will be completed occasionally. In such cases, the generated code has no problems.

   2.1 Conditions
   This problem may occur if the following conditions are all satisfied:

   (1) The optimizing option -OR is selected.

   (2) Two or more function calls are made as follows:
   a. The same function calls are made at more than one place in a function.
   b. Function calls are indirectly made using the same pointer at more than one place in a function.
   c. Function calls are indirectly made using pointers belonging to a union at more than one place in a

function.

NOTE:
Items b and c include every case where the same function calls or different function calls are made at more than one place.

(3) After the function calls in (2) are made, two or more processing flows merge into one at a point.

(4) Among the functions called before the merging point in (3), some are such that the number of arguments passed to them via registers is different from each other.

NOTICE: Conditions (2)-a and -b are both programming errors.

2.2 Examples

1. Case where Condition (2)-c satisfied (function calls indirectly made using pointers belonging to a union)

```
----------------------------------------------------------
----------
union {
   void (*pf0)(void);
   void (*pf1)(int);
} u;

int example2(int i)
{
   switch (i) {
   case 0:
   case 2:
      (*u.pf1)(0);    /* Conditions (2)-c and (4) */
      break;
   default:
      (*u.pf0)();     /* Conditions (2)-c and (4) */
      break;
   }
   return 0;          /* Condition (3) */
}
----------------------------------------------------------
----------
```

2. Case where Condition (2)-a satisfied (same function calls made at more than one place)

NOTE:
This example is a result of a programming error. The number of arguments taken by the function called at the "else" side does not match up with the one in the function's prototype declaration. When a source program like this is compiled, the warning message saying "too few parameters" will be delivered even if compilation terminates normally.

```
-----------------------------------------------------------
----------
void func(int);

int example1(int i)
{
   if (i == 0) {
      func(0);       /* Conditions (2)-a, (3), and (4) */
   } else {
      func();        /* Conditions (2)-a, (3), and (4) */
   }
   return 0;         /* Condition (3) */
}
-----------------------------------------------------------
----------
```

3. Workaround

This problem can be circumvented in either of the following ways:

(1) If the same function calls are made or function calls are indirectly made using the same pointer, match the number of arguments taken by a function to the one in the function's prototype declaration.

(2) If function calls are indirectly made using pointers belonging to a union, place a dummy asm function immediately after any of the function calls that are made before their processing flows merge into one.

Modification of Example 1

```
---------------------------------------------------------------------------
union {
   void (*pf0)(void);
```

```
    void (*pf1)(int);
} u;

int example2(int i)
{
    switch (i) {
    case 0:
    case 2:
        (*u.pf1)(0);
        break;
    default:
        (*u.pf0)();
        asm();          /* Place asm(); here */
        break;
    }
    return 0;
}
-------------------------------------------------------------------------
```

4. Schedule of Fixing the Problem

We plan to fix this problem in the next release of the products.

[Disclaimer]

The past news contents have been based on information at the time of publication. Now changed or invalid information may be included. The URLs in the Tool News also may be subject to change or become invalid without prior notice.