RENESAS Tool News

RENESAS TOOL NEWS on August 1, 2011: 110801/tn1

A Note on Using the C Compiler Packages for the R8C and M16C Families of MCUs

When using the C compiler packages for the R8C and M16C families of MCUs, take note of the following problem:

- With changing the value of a variable by using a cast expression that converts the address of the variable to far pointer type

---

1. Products and Versions Concerned

- C compiler package for the M32C series (M3T-NC308WA)
  V.5.00 Release 1 through V.5.42 Release 00A
- C compiler package for the M16C series, R8C family (M3T-NC30WA)
  V.5.00 Release 1 through V.5.45 Release 01

2. Description

If a constant is assigned to a variable; then the value is changed by using a cast expression that converts the address of the variable to far pointer type, the value will stay unchanged.

2.1 Conditions

The conditions of this problem vary depending on C compiler packages.

2.1.1 In M3T-NC30WA

This problem arises if the following conditions are all satisfied:
(1) Any of the following compile options is used:
-O, -O1 through -O5, -OR, -OS, -ORM(-OR_MAX), and -OSM(-OS_MAX)
(2) A constant is assigned to a variable. In this case is included any constant that substitutes for an expression as a result of optimization.
Note, however, that the variables qualified to be volatile are

excluded, and only in V.5.45 Release 00 and later, external variables are also excluded.

(3) After the assignment expression in (2), the variable in (2) is referenced.

(4) The address of the variable in (2) is converted to pointer type by using a cast expression, and then this expression is used to change the value of the above variable.

(5) The expression in (4) is placed between the expressions in (2) and (3).

(6) Between the expressions in (2) and (3) exist no function calls or no inline-assemble functions.

(7) The attribute of the pointer type in (4) is far.

2.1.2 In M3T-NC308WA

This problem arises if the following conditions are all satisfied:

(1) Any of the following compile options is used:
-O, -O1 through -O5, -OR, -OS, -ORM(-OR_MAX), and -OSM(-OS_MAX)

(2) A constant is assigned to a variable whose attribute is near.
In this case is included any constant that substitutes for an expression as a result of optimization.
Note, however, that the variables qualified to be volatile are excluded, and only in V.5.42 Release 00 and later, external variables are also excluded.

(3) After the assignment expression in (2), the variable in (2) is referenced.

(4) The address of the variable in (2) is converted to pointer type by using a cast expression, and then this expression is used to change the value of the above variable.

(5) The expression in (4) is placed between the expressions in (2) and (3).

(6) Between the expressions in (2) and (3) exist no function calls or no inline-assemble functions.

(7) The attribute of the pointer type in (4) is far.

2.2 Example

```
------------------------------------------------------------
unsigned int       gui;

void func(void)
{
    unsigned int aui;

    aui = 0;              /* Condition (2) */
    *(int far *)&aui = 1;    /* Conditions (4), (5), (6), (7) */
    gui = aui;            /* Condition (3) */
```

```
}
------------------------------------------------------------
```

3. Workaround

To avoid this problem, place a dummy asm function between the expressions in Conditions (2) and (3).

```
------------------------------------------------------------
unsigned int       gui;

 void func(void)
{
    unsigned int aui;

    aui = 0;
    *(int far *)&aui = 1;
    asm();                /* Dummy asm function placed */
    gui = aui;
}
------------------------------------------------------------
```

4. Schedule of Fixing the Problem

For the C compiler package for the M32C series (M3T-NC308WA), sorry we have no plan to fix this problem.
For the C compiler package for the M16C series, R8C family (M3T-NC30WA), we plan to fix this problem in the next version (V.5.XX) of the product. (The release date has not yet been determined.)
