RENESAS Tool News

RENESAS TOOL NEWS on January 16, 2010: 100116/tn1

A Note on Using the C Compiler Package for the R32C/100 MCU Series

Please take note of the following problem in using the C compiler package for the R32C/100 MCU series:

- With accessing arrays successively

---

1. Product and Versions Concerned

The C compiler package for the R32C/100 series
V.1.01 Release 00 and V.1.02 Release 00

2. Description

If three or more expressions that make assignments to arrays exist successively in the program, incorrect code may be generated.

2.1 Conditions

This problem may arise if the following conditions are all satisfied:
(1) Any of the following compile options is selected:
-O1 through -O5, -OS, -OR, -OS_MAX (-OSM), and -OR_MAX (-ORM)
(2) In the program exist three or more successive expressions and they assign variables or constants to arrays of integers or pointers.
Here, the bit widths of integers are not 64 bits.
(3) In all the assignment expressions in (2), the subscript of each array is any of the following:
(3-1) An 8-bit or 16-bit external variable of type integer
(3-2) An 8-bit or 16-bit external or local variable of type unsigned integer if the type of an array is of 16 bits wide
(3-3) A 32-bit external or local variable of type integer
(4) In two or more successive assignment expressions of those in (2), the arrays are the same in the bit widths of their data types and in their subscripts.
(5) In the assignment expression that immediately follows those in

(4), the array is different from the arrays in (4) either in the bit widths of data types or in the variables of subscripts.

2.2 Example

Command line: nc100 -O sample.c

Source code:

```
------------------------------------------------------------
char  c, cary1[4], cary2[4];
short s, sary1[4], sary2[4], sary3[4];
short index1;
unsigned short index2;

void func()
{
   unsigned short index3;

   sary1[index1] = s;    //  Conditions (2), (3-1), and (4)
   sary2[index1] = s;    //  Conditions (2), (3-1), and (4)
   cary1[index2] = c;    //  Conditions (2), (3-1), (4), and (5)
   cary2[index2] = c;    //  Conditions (2), (3-1), and (4)
   sary3[index3] = s;    //  Conditions (2), (3-2), and (5)
}
------------------------------------------------------------
```

3. Workaround

To avoid this problem, insert an asm() between the last of a group of assignment expressions that satisfy Condition (4) and an expression that satisfies Condition (5) in every case where the latter immediately follows the former.

Example:

```
------------------------------------------------------------
char  c, cary1[4], cary2[4];
short s, sary1[4], sary2[4], sary3[4];
short index1;
unsigned short index2;

void func()
{
   unsigned short index3;

   sary1[index1] = s;
   sary2[index1] = s;
   asm();             //  Here asm() inserted.
```

```
    cary1[index2] = c;
    cary2[index2] = c;
    asm();              //  Here asm() inserted.
    sary3[index3] = s;
}
-------------------------------------------------------------
```

4. Schedule of Fixing the Problem

We plan to fix this problem in the C compiler package for the R32C/100 series V.1.02 Release 01.
