RENESAS Tool News

RENESAS TOOL NEWS on September 1, 2006: 060901/tn2

A Note on Using the C Compiler Package --M3T-NC30WA-- for the M16C Series of MCUs --On Evaluating Two or Three Bits in a Controlling Expression--

Please take note of the following problem in using the C compiler package-- M3T-NC30WA--for the M16C series of MCUs:

- On evaluating two or three bits in a controlling expression

---

1. **Versions Concerned**
   The M3T-NC30WA V.1.00 Release 1 through V.5.40 Release 00(A)
   (the C compiler package for the M16C/60, M16C/30, M16C/20, M16C/10, M16C/Tiny, and R8C/Tiny series)

2. **Description**
   Evaluating two or three bits using the logical AND operator (&&) in a controlling expression may generates incorrect code.

3. **Conditions**
   This problem occurs if the following conditions are all satisfied:

   (1) In the program exists a controlling expression where two or three bits are evaluated using one or two && operators.

   (2) Every bit in (1) above is one bit in width.

   (3) Every bit in (1) above is a member of a bit field, and the bit field is of type char.

(4) All the bits in (1) are positioned in the lower or upper four bits of the bit field

(5) The positions of the bits in (1) are as follows:

(a) If two bits are evaluated, they are separated by 1 or 2 bits from each other.

(b) If three bits are evaluated, any one of them is separated by 1 bit from another.

Examples of the positions of the evaluated bits in a bit field
(when they are in the lower four bits)

Bit Position	7	6	5	4	3	2	1	0
Pattern 1					*		*	
Pattern 2					*			*
Pattern 3						*		*
Pattern 4					*		*	*
Pattern 5					*	*		*

(6) Optimizing option -O4 is used in compilation.

Example:

```
-----------------------------------------------
struct{
  unsigned char b0:1;      // Conditions (2), (4), and
(5)
  unsigned char b1:1;
  unsigned char b2:1;
  unsigned char b3:1;      // Conditions (2), (4), and
(5)
  unsigned char b4:1;
  unsigned char b5:1;
  unsigned char b6:1;
  unsigned char b7:1;
  }bit;                  // Condition (3)

int main()
{
  if(bit.b0 && bit.b3){   // Condition (1)
```

```
   return  1;
   }
}
-----------------------------------------------
```

4. Workarounds

This problem can be circumvented either of the following ways:

(1) Use a nested if statement instead of a && operator.

Example:

```
-----------------------------------------------
. . . . . . . . . . .

int  main()
{
   if(bit.b0){
         if(bit.b3){
         return  1;
         }
       }
 }
 -----------------------------------------------
```

(2) Use optimizing option -O3 instead of -O4.

5. Schedule of Fixing the Problem

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