A Note on Using the M3T-NC30WA C-Compiler Package
--On Using the #pragma ADDRESS Directive--

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

- On using the #pragma ADDRESS directive

1. **Versions Concerned**
   M3T-NC30WA V.1.00 Release 1 through V.5.30 Release 02
   (This C compiler package is used for the M16C/60, M16C/30, M16C/20, M16C/10,
   M16C/Tiny, and R8C/Tiny series of MCUs.)

2. **Description**
   Bit-accessing a variable defined by using the #pragma ADDRESS directive may result in an assemble error.

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

   (1) There exists a variable defined by using the #pragma ADDRESS directive, and the variable is a union of the array type.

   (2) The value of the address specified by the #pragma ADDRESS directive in (1) is less than 01FFFH.

   (3) The value of the address of the bit-accessed variable is greater than 01FFFH.

2.2 Example

```c
typedef struct{
```
unsigned char b0:1;
unsigned char b1:1;
unsigned char b2:6;
}BIT;

#pragma ADDRESS bit 01FFEH
BIT bit[10];
void func(void)
{
    bit[2].b0 = 1;
}

3. **Workaround**
   Bit-access the variable using an asm function.

```c
typedef struct{
    unsigned char b0:1;
    unsigned char b1:1;
    unsigned char b2:6;
}BIT;

#pragma ADDRESS bit 01FFEH
BIT bit[10];
void func(void)
{
    asm("or.b #01H,$$,bit[2].b0);
}
```

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

[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.

© 2010-2016 Renesas Electronics Corporation. All rights reserved.