# RENESAS Tool News

**MAEC TOOL NEWS: MAECT-M3T-NC308WA_2-030316D**

## Notes on Using C Compilers M3T-NC308WA and M3T-NC30WA

Please take note of the following problems in using C compilers (with an assembler and integrated development environment) M3T-NC308WA and M3T-NC30WA:

- On using the #pragma ADDRESS declaration
- On using the #pragma ADDRESS declaration or the volatile qualifier for a variable to be assigned to another

---

### 1. Problem on Using the #pragma ADDRESS Declaration

Products and Versions Concerned

For the M32C/80 and M16C/80 series MCUs

M3T-NC308WA V.1.00 Release 1 through V.5.00 Release 1

For the M16C/60, M16C/30, M16C/20, and M16C/10 series MCUs

M3T-NC30WA V.1.00 Release 1 through V.5.00 Release 2

1.2 Description

The volatile type qualifier of a variable declared using the #pragma ADDRESS directive may be invalidated.

1.3 Conditions

This problem occurs if the following two conditions are satisfied:
(1) A variable is defined.
(2) After defining the variable in (1), it is declared using the #pragma ADDRESS directive.

1.4 Example

```
-----------------------------------------------
  int       gi;             /* Condition (1) */
```

```
  #pragma    ADDRESS gi 500H  /* Condition (2) */
------------------------------------------
```

1.5 Workaround

Declare the variable using the #pragma ADDRESS directive before defining it.

```
------------------------------------------
  #pragma    ADDRESS gi      500H
  int gi;
------------------------------------------
```

1.6 Schedule of Fixing the Problem

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

---



## 2. Problem on Using the #pragma ADDRESS Declaration or the Volatile Qualifier for a Variable to Be Assigned to Another

2.1 Product and Versions Concerned

For the M32C/80 and M16C/70 series MCUs

M3T-NC308WA V.1.00 Release 1 through V.5.00 Release 1

2.2 Description

When an assignment-destination variable (a variable that another is assigned to) is 8 bits wide and an assignment-source variable (a variable that is assigned to another) (*) is 16 bits wide, the latter may be referenced as of 8 bits wide. Therefore, be aware that I/O registers and others that must be read out in 16 bits wide will be done in 8 bits wide.

Note: * The variable involved must be qualified as volatile or declared using the #pragma ADDRESS directive.

2.3 Conditions

This problem occurs if the following four conditions are satisfied:

(1) An assignment statement exists.

(2) The assignment-destination variable in (1) is 8 bits in width.

(3) The assignment-source variable in (1) is 16 bits in width.

(4) The variable in (3) is qualified as volatile or declared using the #pragma ADDRESS directive.

2.4 Example

```
------------------------------------------------
  #pragma    ADDRESS X0R 2C0H
  char       gc;          /* Condition (2) */
  int  X0R;               /* Condition (3) */
  void func(void)
  {
    gc = X0R;             /* Condition (1) */
  }
------------------------------------------------
```

2.5 Workaround

Assign the assignment-source variable to a temporary 16-bit variable that is not qualified as volatile; then assign the temporary variable to the assignment-destination variable.

```
------------------------------------------------------------
  volatile int gi;
  char       gc;
  void func(void)
  {
    int tmp;
    tmp = gi;   /* The source variable assigned to
                                a tmp variable */
    gc = tmp;   /* A tmp variable assigned to
                           the destination variable */
  }
------------------------------------------------------------
```

2.6 Schedule of Fixing the Problem

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

---