Tool News

RENESAS TOOL NEWS on July 16, 2005: RSO-M3T-CC32R_2-050716D

A Note on Using the C Compiler Package M3T-CC32R

Please take note of the following problem in using the C compiler package M3T-CC32R (this is used for the M32R family of MCUs):

- On reading out an element of an array or a member of a structure or union that is qualified to be volatile twice or more times successively

---

1. **Versions Concerned**
   M3T-CC32R V.1.00 Release 1 through V.3.20 Release 1

2. **Description**
   Reading out an element of an array or a member of a structure or union that is qualified to be volatile twice or more times successively may invalidate the volatile qualification.
   And, if the above element or member is qualified to be const as well as volatile, an internal error may arise with the following error message displayed:
   cg32r: "xxxx", line XX: internal error: illegal IL, refer of tuse null. (Here, xxxx denotes a file name and XX a line number.)

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

   (1) Any of the optimizing options -O7, -O6, -O5 and -O4 is used; or -Ospace or -Otime alone is selected.

   (2) Any of the following objects is read out twice or more times successively:
   (a) An element of an array
   (b) A member of a structure
   (c) A member of a union

   (3) Between two reads in (2) above exists none of the

following operations:
(a) A write to the memory space pointed to by a pointer
(b) A function call

(4) The object in (2) is qualified to be:
(a) const and volatile, or
(b) volatile only
Note:
An internal error arises in (a), and the volatile qualification becomes invalid in (b).

2.2 Examples

Internal errors arise in Examples 1 and 3 below, and the volatile qualification becomes invalid in Example 2.

Example 1. (An internal error arises.)
Source file: sample1.c

```
-----------------------------------------------------------------
 extern const volatile int array1[4]; /* Conditions (2)(a) and
(4)(a) */
 int a1, b1;

 void func1(int index)
 {
    a1 = array1[index];              /* Condition (3) */
    b1 = array1[index];              /* Condition (3) */
 }
-----------------------------------------------------------------
```

Example 2. (The volatile qualifier becomes invalid.)
Source file: sample2.c

```
-----------------------------------------------------------------
 extern volatile int array2[4];      /* Conditions (2)(a) and
(4)(b) */
 int a2, b2;

 void func2(int index)
 {
    a2 = array2[index];              /* Condition (3) */
    b2 = array2[index];             /* Condition (3) */
 }
 -------------------------------------------------------------
---------
```

Example 3. (An internal error arises.)
 Source file: sample3.c

```
 -------------------------------------------------------------
---------
 struct {
    const volatile int var;        /* Conditions (2)(b) and
(4)(a) */
 } data3;

 int func3(void)
 {
    return (data3.var + data3.var);   /* Condition (3) */
 }
 -------------------------------------------------------------
---------
```

3. **Workaround**

This problem can be circumvented in any of the following ways:

(1) Upgrade your compiler to V.4.00 Release 1 or later.

(2) Suppress the optimization on level 4 as follows:
   - Use any of the options -O3, -O2, -O1, and -O0; not any of those
     -O7, -O6, -O5, and -O4.
   - Use any of the options -O3, -O2, -O1, and -O0 in addition to
     -Ospace or -Otime alone.

(3) Instead of the direct reads in Condition (2), use indirect references by pointers.

Modification of Example 1

```
-------------------------------------------------------------
---------
extern const volatile int array1[4];
int a1, b1;

void func1(int index)
```

```
{
   const volatile int *ptr; /* Declare pointer to array1[index]
*/
   ptr = &array1[index];  /* Apply address operator to
array1[index] */
   a1 = *ptr;            /* Use indirect reference by pointer */
   b1 = *ptr;            /* Use indirect reference by pointer */
}
------------------------------------------------------
---------
```

Modification of Example 2

```
------------------------------------------------------
---------
extern volatile int array2[4];
int a2, b2;

void func2(int index)
{
   volatile int *ptr;     /* Declare pointer to array2[index] */
   ptr = &array2[index];  /* Apply address operator to
array2[index] */
   a2 = *ptr;            /* Use indirect reference by pointer */
   b2 = *ptr;            /* Use indirect reference by pointer */
}
------------------------------------------------------
---------
```

Modification of Example 3

```
------------------------------------------------------
---------
struct {
   const volatile int var;
} data3;

int func3(void)
{
   const volatile int *ptr; /* Declare pointer to data3.var */
   ptr = &data3.var;       /* Apply address operator to
data3.var */
   return (*ptr + *ptr);    /* Use indirect reference by pointer
*/
}
```

```
--------------------------------------------------------------
---------
```

4. Schedule of Fixing the Problem

This problem has already been fixed in the M3T-CC32R V.4.00 and later.
