RENESAS Tool News

RENESAS TOOL NEWS on October 1, 2010: 101001/tn2

A Note on Using the C/C++ Compiler Package for the M32R MCU Family (M3T-CC32R)

When you use the C/C++ compiler package for the M32R MCU family (M3T-CC32R), take note of the following problem:

- With reading auto variables qualified to be const

1. Product and Versions Concerned

The C/C++ compiler package for the M32R family
V.4.00 Release 1 through V.5.01 Release 01

2. Description

If a code for reading out an auto variable qualified to be const in a function is executed, an address exception may occurs.

2.1 Conditions

This problem arises if the following conditions are all satisfied:

(1) The combination of optimizing options in compilation is either of the following:
  (a) Any one of those, -O2, -O3, -O6, -O7, and -O, is selected.
  (b) Either -Ospace or -Otime is selected, and -O0, -O1, -O4 and -O5 are not selected.

(2) In a function is defined a variable that is qualified to be const and whose storage class is auto, together with its initializer.

(3) The variable in (2) is 1 or 2 bytes long of an integral type.

(4) The variable in (2) is not qualified to be volatile.

(5) In the definition in (2), the variable and its initializer are different in type or qualifier. If the initializer contains a cast operator, neglect the type conversion to evaluate the difference.

(6) The variable in (2) is referenced at three or more locations in the function.

(7) The variable in (2) is assigned to a stack whose offset is not

a multiple of 4.
To check to see the assignment, generate the assembly-language source file including C source lines by using the -CS or -cs option. In the generated file, information about assignments is displayed immediately before each function in assembly code as follows.

Examples:

```
------------------------------------------
;[ASSIGN] var02 R9         ; var02 assigned to register R9.
;[ASSIGN] var01 @(31,R15)  ; var01 assigned to stack with offset 31.
------------------------------------------
```

That is, a register and a stack name follow ;[ASSIGN] and a variable name each. Here,

Rn:	The name of the register to which the variable is assigned. (n indicates the register number.)
@(off,R15):	The offset and name of the stack to which the variable is assigned. (off indicates the offset on the stack.)

2.2 Examples

In the examples below, the variable var01 satisfies all the conditions in 2.1 and is involved in this problem.

Example 1. Source file: sample.c

```
----------------------------------------------------------------
 unsigned char gv01,gv02;
 unsigned char gfunc03(void);
 int table[100];
 unsigned char buf[100];
 void func1(void)
 {
    int i, j;
    int d1 = 800, d2 = 500;
    const unsigned char var01 = gv01;   /* Conditions (2),(3),(4),
                                        and (5) */
    unsigned char *ptr = &buf[d1];
    const unsigned char var02 = gv02;
    int d3 = 0, d4 = 1, d5 = 0;
    unsigned char var03 = gfunc03();
    *ptr++ = var02 & 0xf;
    *ptr++ = (var02>>4) & 0x0f;
    *ptr++ = var01 & 0xf;              /* Condition (6) */
    *ptr++ = (var01>>4) & 0x0f;        /* Condition (6) */
    for (i = 0; i < var01; i++) {      /* Condition (6) */
       d1 = buf[i] * d3;
```

```
        *ptr++ = buf[i]/d4;
      }
      for (i = 0; i < var02; i++) {
        d3 = buf[i] * d5;
        *ptr++ = buf[i]/d2;
      }
      *ptr++ = d1 + d2 + d3;
   }
------------------------------------------------------------
```

Example 2. Command line:

```
------------------------------------------------------------
 cc32R -CS -O7 sample.c    (Condition (1))
------------------------------------------------------------
```

Example 3. Assembly-language source file generated by using
    -CS option (excerpt from sample.ms)

```
------------------------------------------------------------
. . . . . . . . . . . . . . . . . . . . . . . . .
;[ASSIGN] var01 @(31,R15)    ; Condition (7) (Satisfied because 31
                              is not a multiple of 4.)
. . . . . . . . . . . . . . . . . . . . . . . . .
;[ASSIGN] var02 R9
. . . . . . . . . . . . . . . . . . . . . . . . .
$func1:
. . . . . . . . . . . . . . . . . . . . . . . . .
;[LINE] 16 "sample.c"
;[SOURCE]     *ptr++ = var01 & 0xf;
     LD       R3,@(31,R15)          ; Here occurs address exception
. . . . . . . . . . . . . . . . . . . . . . . . .
------------------------------------------------------------
```

In this example, var01 satisfies all the conditions in 2.1 and is involved in the problem.
The variable var02 satisfies the same conditions as var01 in the source code. However, because it is assigned to register R9, it does not satisfy Condition (7). So it is not involved in the problem.
The variable var03 is not qualified to be const and does not satisfy Condition (2), so it is also not involved in the problem.

3. Workaround

To avoid this problem, use either of the following ways:

(1) Change the type of the variable involved from 1 or 2 bytes long of an integral type to 4 bytes long of type int or long.
   Example:

```
Original:   const unsigned char var01 = gv01;
Change to:  const unsigned int var01 = gv01;
---------------------------------------------------
```

(2) Do not qualify the variable involved to be const.
Example:

```
---------------------------------------------------
Original:   const unsigned char var01 = gv01;
Change to:  unsigned char var01 = gv01;
---------------------------------------------------
```