Tool News

RENESAS TOOL NEWS on May 1, 2004: RSO-M3T-CC32R_2-040501D

A Note on Using C-Compiler Package M3T-CC32R

--On Making a Change to a Variable by Using a Bit Field--

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

- On making a change to a variable by using a bit field

---

1. **Versions Concerned**
   M3T-CC32R V.4.00 Release 1 through V.4.20 Release 1A

2. **Description**
   When an integer-type or floating-type variable is referenced, correct reference may not be made by improper optimization.

   2.1 Conditions
   This problem occurs if the following five conditions are satisfied:
   (1) Any of the optimizing options -O4, -O5, -O6, and -O7 is used; or -Ospace only or -Otime only is selected.
   (2) An assignment is made to an integer-type or floating-type variable.
   (3) An assignment is made to a bit field by using a pointer to a structure containing a bit field and a return value of an address (&) variable.
   (4) The pointer in (3) points to the same address where the variable in (2) is stored.
   (5) The variable in (2) is referenced after the assignment in (3).

   2.2 Examples

   Source file 1: sample1.c

```
unsigned long src1, dst1;
struct btf1 {
  unsigned long x:1;
  unsigned long y:3;
};

void func01(void)
{
    src1 = 0x80000000;            /* Condition (2) */
    ((struct btf1 *)&src1)->y = 2; /* Conditions (3) and (4) */
    dst1 = src1;                  /* Condition (5) */
}
```

Source file 2: sample2.c

```
unsigned long src2, dst2;
struct btf2 {
  unsigned long x:1;
  unsigned long y:3;
} *ptr2;

void initialize(void)
{
    ptr2 = (struct btf2 *) &src2;  /* Condition (4) */
}

void func02(void)
{
  /* An initialize function executed before func02 done */
    src2 = 0;                     /* Condition (2) */
    ptr2->y = 2;                  /* Condition (3) */
    dst2 = src2;                  /* Condition (5) */
}
```

Source file 3: sample3.c

```
long src3, dst3;
struct btf3 {
  unsigned long x:1;
  unsigned long y:3;
} *ptr3;
```

```
long func03(void)
{
   src3 = 255;                /* Condition (2) */
   ptr3 = (struct btf3 *) &src3;  /* Condition (4) */
   ptr3->y = 2;               /* Condition (3) */
   return src3;               /* Condition (5) */
}
----------------------------------------------------------------------
```

Source file 4: sample4.c

```
----------------------------------------------------------------------
long src4, dst4;
struct btf4 {
  unsigned long x:1;
  unsigned long y:3;
} *ptr4;

long func04(long arg4)
{
   src4 = arg4;               /* Condition (2) */
   ptr4 = (struct btf4 *) &src4;  /* Condition (4) */
   ptr4->y = 2;               /* Condition (3) */
   return src4;               /* Condition (5) */
}
----------------------------------------------------------------------
```

3. Workaround

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

(1) If the pointer in Condition (3) is the address of a variable, create a union having two members of the original type of the variable and the type of the structure containing a bit field; then replace the assignment in Condition (3) with the one to a member of the union.

Modification of source file 1:

```
----------------------------------------------------------------------
unsigned long dst1;
struct btf1 {
  unsigned long x:1;
  unsigned long y:3;
};
union {                     /* Replace src1 with a union */
```

```
  unsigned long word;
  struct btf1 bit;
} src1;

void func01(void)
{
    src1.word = 0x80000000;   /* Replace src1 with src1.word */
    src1.bit.y = 2;           /* Modify src1 by bitwise operation */
    dst1 = src1.word;         /* Replace src1 with src1.word */
}
--------------------------------------------------------------------
```

(2) When the variable referenced in Condition (5) is the right term of an assignment expression, replace each of the left and right terms with an indirection reference using a pointer to a volatile object.

Modification of source file 2:

```
--------------------------------------------------------------------
unsigned long src2, dst2;
struct btf2 {
  unsigned long x:1;
  unsigned long y:3;
} *ptr2;

void initialize(void)
{
    ptr2 = (struct btf2 *) &src2;  /* Condition (4) */
}

void func02(void)
{
  /* An initialize function executed before func02 done */
    src2 = 0;
    ptr2->y = 2;
         /* Both terms changed to volatile objects */
    *((volatile unsigned long*)&dst2) = *((volatile unsigned long*)&src2);

}
--------------------------------------------------------------------
```

(3) When -Ospace only or -Otime only is selected, use any of the optimizing options -O0, -O1, -O2, and -O3 at the same time.

4. Schedule of Fixing the Problem

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