RENESAS Tool News

RENESAS TOOL NEWS [August 16, 2003: RSO-M3T-CC32R-030816D]

A Note on Using Cross-Tool Kit M3T-CC32R

Please take note of the following problem in using the M3T-CC32R cross-tool kit for the M32R family MCUs:

- On compiling a program in which a structure whose first member is 4 bites in length is copied

---

1. **Versions Concerned**
   M3T-CC32R V.1.00 Release 1 through V.4.10 Release 1

2. **Description**
   When a program contains a description for copying a structure whose first member is 4 bytes in length (as of type long, pointer, etc.) by using the assignment operator (=), compiling the program with a -O1 equivalent option being selected generates the code that writes incorrect data to the destination structure.

   2.1 Conditions

   This problem occurs if the following seven conditions are satisfied:

   (1) At compilation the -Ospace optimizing option is not used, and an option that covers the -O1 option's function is selected; that is, any of the -O1, -O3, -O5, and -O7 options or the -Otime option only is selected.

   (2) A structure consisting of two or more members is declared.

   (3) The first member of the structure in (2) is 4 bytes in length and of type int or pointer not qualified as volatile.

   (4) Another (copy source) structure that satisfies conditions (2) and (3) above and is not qualified as volatile is defined.

   (5) A value is assigned to the first member of the source structure in (4).

   (6) After the assignment in (5), the source structure in (4) is copied to the other

(destination) structure of the same type using the assignment operator (=).

(7) Between the assignment in (5) and the copy in (6) exists none of the following call and writes:

(a) a function call

(b) a write to any member of the source structure in (4) except its first member

(c) the write of a value to a memory-allocated area

2.2 Example

Source file: sample.c

```
-----------------------------------------------------------------
   struct S1 {             /* Condition (2) */
      char *ptr;           /* Condition (3) */
      char chr;
   };

   struct S1 src1;          /* Condition (4) */
   struct S1 dst1;

   void func1(char *str)
   {
      src1.chr = 'a';
      src1.ptr = str;       /* Condition (5) */
                        /* Condition (7) */
      dst1 = src1;          /* Condition (6) */
   }
-----------------------------------------------------------------
```

Source file sample2.c

```
-----------------------------------------------------------------
   typedef struct {         /* Condition (2) */
      unsigned long a;      /* Condition (3) */
      unsigned short b;
      unsigned char c;
   } S2;

   S2 dst2;

   void func2(S2 *psrc2)     /* Condition (4) */
   {
```

```
    psrc2->a = 0;        /* Condition (5) */
                         /* Condition (7) */
    dst2 = *psrc2;       /* Condition (6) */
  }
```

Selections of options

```
  % cc32R -c -Otime -O7 sample1.c    /* Condition (1) */
  % cc32R -c -Otime -O7 sample2.c    /* Condition (1) */
```

Here a % denotes a prompt.

3. Workaround

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

(1) Qualify the source structure as volatile.

Modification of source file sample1.c

```
  struct S1 {
    char *ptr;
    char chr;
  };

  volatile struct S1 src1; /*Source qualified as volatile*/
  struct S1 dst1;

  void func1(char *str)
  {
    src1.chr = 'a';
    src1.ptr = str;

    dst1 = src1;
  }
```

Modification of source file sample2.c

```
  typedef struct {
    unsigned long a;
    unsigned short b;
    unsigned char c;
```

```
} S2;

S2 dst2;

void func2(S2 *psrc2)
{
   psrc2->a = 0;

  {
   volatile S2 *p2 = (volatile S2 *) psrc2;
   dst2 =  *p2;    /* Structure pointer qualified
                          as volatile used */
  }
}
```

---

(2) Place the assignment statement of a value to another variable to which memory is allocated between the assignment statement to the first member of the source structure and the copy statement of the source to the destination.

Modification of source file sample1.c

```
struct S1 {
   char *ptr;
   char chr;
};

struct S1 src1;
struct S1 dst1;

void func1(char *str)
{
   src1.ptr = str;
   src1.chr = 'a';  /* Assigning a value to member chr is
                       performed after that to ptr;
                       Condition (7)-(b) is sidestepped
                       because chr does not meet
                       Condition (3) */
   dst1 = src1;
}
```

Modification of source file sample2.c

```
------------------------------------------------------------
typedef struct {
   char a;
   unsigned long b;
   unsigned short c;
} S2;

S2 dst2;
int dummy;          /* A memory-allocated dummy
                       variable defined */

void func1(S2 *psrc2)
{
   psrc2->b = 0;
   dummy = 0;       /* A write to dummy sidesteps
                       Condition (7)-(c) */
   dst2 = *psrc2;
}
------------------------------------------------------------
```

(3) Use the size-oriented optimizing option -Ospace. Note that the -Otime option cannot be used simultaneously.

(4) Don't use any of the -O1, -O3, -O5 and -O7 optimizing options. If using -Otime, select any of the -O0, -O2, -O4, and -O6 options at the same time.

4. Schedule of Fixing the Problem

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