RENESAS Tool News

RENESAS TOOL NEWS on December 16, 2003: RSO-M3T-CC32R-031216D

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 the same two or more indirection references that use pointer

---

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

2. **Description**
   Consider the case where, in a function that takes a pointer variable as a parameter, the same two or more indirection references that use the above pointer variable are made before and after a function call. Compiling this program with a level-4 or equivalent optimizing option being selected will neglect the indirection reference made after the function call and create code that uses the value to which an indirection reference was made before the function call.

   2.1 Conditions
   This problem occurs if the following conditions, (1) through (8), are satisfied:
   (1) An optimizing option covering the -O4 option's function is selected at compilation (any of the -O4, -O5, -O6, and -O7 options or -Ospace only or -Otime only is selected).
   (2) In the program exists function A that takes a pointer variable as a parameter.
   (3) In function A exists indirection reference B that uses the same pointer variable as the one described in (2) above and references the following (a), (b), and (c):
   (a) An array
   (b) Any member in a structure except the first member of the structure pointed to by the pointer

(c) An address represented by the pointer plus an integer

(4) None of the following (a) through (e) is performed from the beginning of the processing block (*1) containing indirection reference B to the end of this indirection reference:

(a) Assignment to the pointer in (2)

(b) Writes to memory by the indirection reference using a pointer

(c) Assignment to a global variable

(d) Function calls

(e) Execution of an asm function

NOTE:

*1. This processing block means a set of statements that are executed in the order of their descriptions in any of the following three statements enclosed with the braces {...}:

(i) Statements in a function

(ii) Statements selected in an if or else statement

(iii) Statements executed repeatedly in a while, do, or for statement

(5) One or more function calls are made after indirection reference B.

(6) After the function calls in (5), another indirection reference C that is the same as indirection reference B is made in any of the following statements and expressions:

(a) Statements selected in an if-else statement

(b) Statements executed repeatedly in a do-while statement

(c) A controlling expression in a while statement

(d) A controlling expression in a for statement

(e) Repeatedly executed statements in a while or for statement the value of whose controlling expression is a constant other than zero

(7) Indirection reference B, a function call, and indirection reference C are made in this order.

(8) None of the following (a) through (d) exist from the end of indirection reference B to the beginning of indirection reference C:

(a) Assignment to the pointer in (2)

(b) Writes to memory by the indirection reference using a pointer

(c) Assignment to a global variable

(d) A while or for statement

2.2 Examples

1. Source file sample1.c

```
-----------------------------------------------------------------
extern void extfunc1(void);

int func1(char *ptr, int cnt)  /* Condition (2) */
{
   int c1 = 0, c2 = 0;
   while (--cnt) {
                        /* Condition (4) */
      c1 = ptr[0];          /* Condition (3)(a) */
      extfunc1();           /* Condition (5) */
                        /* Condition (8) */

      if (cnt) {           /* Conditions (6)(a) and (7) */

                        /* Condition (8) */
         c2 = ptr[0];       /* Conditions (6)(a) and (7) */
      }
   }
   return c1 + c2;
}
-----------------------------------------------------------------
```

2. Source file sample2.c

```
-----------------------------------------------------------------
extern void extfunc2(void);
struct st2 { int a; int b; };

int func2(struct st2 *ptr, int cnt) /* Condition (2) */
{
   int c1 = 0, c2 = 0;

                        /* Condition (4) */
   c1 = ptr->b;             /* Condition (3)(b) */
   extfunc2();              /* Condition (5) */
                        /* Condition (8) */
   do {                   /* Conditions (6)(b) and (7) */
                        /* Condition (8) */
```

```
      c2 = ptr->b;          /* Conditions (6)(b) and (7) */
    } while (--cnt);
    return c1 + c2;
}
```

---

3. Source file sample3.c

```
extern void extfunc3(void);

int func3(char *ptr, int flg)  /* Condition (2) */
{
    int c1 = 0, c2 = 0;
    if (flg) {
                          /* Condition (4) */
        c1 = *(ptr+3);        /* Condition (3)(c) */
        extfunc3();           /* Condition (5) */
                          /* Condition (8) */
        while (*(ptr+3)) {     /* Conditions (6)(c) and (7) */
            ++c2;
        }
    }
    return c1 + c2;
}
```

---

4. Source file sample4.c

```
extern void extfunc4(void);

int func4(char *ptr, int cnt)  /* Condition (2) */
{
    int c1 = 0, c2 = 0;
                          /* Condition (4) */
    c1 = ptr[1];              /* Condition (3)(a) */
    extfunc4();               /* Condition (5) */
                          /* Condition (8) */
    for ( ; ptr[1]; --c2) {    /* Conditions (6)(d) and (7) */
    }
    return c1 + c2;
}
```

---

5. Source file sample5.c

```
-----------------------------------------------------------------
extern void extfunc5(void);

int func5(char *ptr, int cnt)  /* Condition (2) */
{
   int c1 = 0, c2 = 0;
                        /* Condition (4) */
   c1 = *(ptr+0);           /* Condition (3)(c) */
   extfunc5();             /* Condition (5) */
                        /* Condition (8) */
   while (1) {             /* Conditions (6)(e) and (7) */
                        /* Condition (8) */
      c2 = *(ptr+0);        /* Conditions (6)(e) and (7) */
      if (--cnt) break;
   }
   return c1 + c2;
}
-----------------------------------------------------------------
```

Selections of options

```
 ---------------------------------------------------------------
 % cc32R -c -O4 sample1.c        /* Condition (1) */
 % cc32R -c -O4 sample2.c        /* Condition (1) */
 % cc32R -c -O4 sample3.c        /* Condition (1) */
 % cc32R -c -O4 sample4.c        /* Condition (1) */
 % cc32R -c -O4 sample5.c        /* Condition (1) */
 ---------------------------------------------------------------
```

Here a % denotes a prompt.

3. Workaround

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

(1) Create a pointer variable at the beginning of the processing block and make indirection references using this pointer variable.

Modification of source file sample1.c

```
-----------------------------------------------------------------
extern void extfunc1(void);

int func1(char *dmy, int cnt)  /* Replace ptr with dmy */
{
   int c1 = 0, c2 = 0;
```

```
   while (--cnt) {
      char *ptr = dmy;       /* Create ptr and initialize it
                                        by dmy */

      c1 = ptr[0];          /* Use ptr for the subsequent
                               indirection references */
      extfunc1();

      if (cnt) {

         c2 = ptr[0];       /* Use ptr for the subsequent
                               indirection references */
      }
   }
   return c1 + c2;
}
```

Modification of source file sample2.c

```
extern void extfunc2(void);
struct st2 { int a; int b; };

int func2(struct st2 *dmy, int cnt) /* Replace ptr with dmy */
{
   int c1 = 0, c2 = 0;
   struct st2 *ptr = dmy;     /* Create ptr and initialize it
                                        by dmy */

   c1 = ptr->b;             /* Use ptr for the subsequent
                               indirection references */
   extfunc2();

   do {

      c2 = ptr->b;          /* Use ptr for the subsequent
                               indirection references */
   } while (--cnt);
   return c1 + c2;
}
```

---

(2) To sidestep Condition (4), place a dummy asm function immediately before the

indirection reference in Condition (3).

Modification of source file sample3.c

```
------------------------------------------------------------------
#pragma keyword asm on          /* Validate an asm function */
extern void extfunc3(void);

int func3(char *ptr, int flg)
{
  int c1 = 0, c2 = 0;
  if (flg) {

    asm("");              /* Execute an asm function */
    c1 = *(ptr+3);
    extfunc3();

    while (*(ptr+3)) {
      ++c2;
    }
  }
  return c1 + c2;
}
------------------------------------------------------------------
```

(3) Suppress the optimization covering -O4.
The optimization covering -O4 is performed when -O4, -O5, -O6, or -O7; or -Otime only, or -Ospace only is selected.
If you want to use -Otime or -Ospace, use any of the 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.