RENESAS ツールニュース

RENESAS TOOL NEWS [2003年08月16日：RSO-M3T-CC32R-030816D]

M3T-CC32R
ご使用上のお願い

M32Rファミリ用クロスツールキットM3T-CC32Rの使用上の注意事項を連絡します。

- 先頭に4バイトのメンバを持つ構造体をコピーするプログラムに関する注意事項

---

1. **該当製品**
   M3T-CC32R V.1.00 Release 1 ～ V.4.10 Release 1

2. **内容**

   4バイト(long型やポインタ型など)のメンバを先頭に持つ構造体を代入演算子(=)でコピーする記述を含むプログラムを、レベル1の最適化が有効となるオプションを指定してコンパイルすると、コピー先の構造体に間違ったデータを書き込むコードを生成します。

   2.1 発生条件

   以下7点の条件を全て満たす場合に発生します。

   (1) コンパイル時に -Ospaceの指定がなく、かつ -O1の最適化と同じ内容を含む最適化オプションを指定している。(-O1, -O3, -O5, -O7いずれかの指定が有効になっているか、-Otimeを単独で指定している。)

   (2) メンバを2つ以上持つ構造体型を宣言している。

   (3) (2)の構造体型の先頭に、volatile修飾がなくかつ、4バイトの整数型またはポインタ型のメンバがある。

   (4) volatile修飾がなく、かつ型が(2)である構造体を定義している。

   (5) (4)の構造体の(3)に該当するメンバに値を代入している。

   (6) (5)の代入の後に、(4)の構造体メンバの値を代入演算子(=)を用いて同じ型の別の構造体にコピーしている。

   (7) (5)の代入と(6)の構造体コピーの間に、次のいずれの処理も存在しない。

(a) 関数呼び出し
(b) (4)の構造体の中の、(3)に該当しないメンバへの書き込み
(c) メモリに割りつけられた領域への値の書き込み

2.2 発生例

[ソースファイル例1]
[sample1.c]

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

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

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

[ソースファイル例2]
[sample2.c]

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

  S2 dst2;

  void func2(S2 *psrc2)     /* 条件(4) */
  {
     psrc2->a = 0;         /* 条件(5) */
                        /* 条件(7) */
     dst2 = *psrc2;        /* 条件(6) */
  }
```

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

[オプション指定例]
-----------------------------------------------------------------
  % cc32R -c -Otime -O7 sample1.c     /* 条件(1) */
  % cc32R -c -Otime -O7 sample2.c     /* 条件(1) */
-----------------------------------------------------------------
(% は プロンプトを表します)
```

3. **回避策**

次の4点のいずれかの方法で回避してください。

(1) コピー元の構造体をvolatile修飾する。

```
[ソースファイル例1の変更例]
 [sample1.c]
-----------------------------------------------------------------
  struct S1 {
     char *ptr;
     char chr;
  };

  volatile struct S1 src1;   /* コピー元をvolatile修飾 */
  struct S1 dst1;

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

     dst1 = src1;
  }
-----------------------------------------------------------------

[ソースファイル例2の変更例]
 [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;     /* volatile修飾した構造体ポインタを用いる */
    }
  }
```

---

(2) 4バイトの先頭メンバへの代入と構造体のコピーの間に、他のメモリ領域に割りつけられる変数への代入を記述する。

[ソースファイル例1の変更例]
[sample1.c]

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

  struct S1 src1;
  struct S1 dst1;

  void func1(char *str)
  {
     src1.ptr = str;
     src1.chr = 'a';    /* メンバchrへの代入をptrよりも後に行う。
                           chrは条件(3)に該当しないため、
                           条件(7)(b)が回避できる */
     dst1 = src1;
  }
```

[ソースファイル例2の変更例]
[sample2.c]

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

```
    unsigned short c;
  } S2;

  S2 dst2;
  int dummy;          /* メモリに割りつけられるダミー変数を定義 */

  void func1(S2 *psrc2)
  {
    psrc2->b = 0;
    dummy = 0;       /* ダミー変数への書き込み、条件(7)(c)を回避 */
    dst2 = *psrc2;
  }
-----------------------------------------------------------------
```

(3) -Ospace(サイズ重視最適化)オプションを指定する。
(この場合、-Otimeオプションは同時に指定できません。)

(4) -O1, -O3, -O5 および -O7の最適化オプションを指定しない。
-Otimeを使用する場合は、同時に -O0,-O2,-O4,-O6のいずれかの指定が必要。

4. 恒久対策

本問題は、次期バージョンで改修する予定です。
