RENESAS ツールニュース

RENESAS TOOL NEWS 2005年07月16日：RSO-M3T-CC32R_2-050716D

Cコンパイラパッケージ M3T-CC32R ご使用上のお願い

M32Rファミリ用CコンパイラパッケージM3T-CC32Rの使用上の注意事項を連絡します。

- volatile修飾された配列の要素、構造体のメンバまたは共用体のメンバを 2回以上続けて読み出す場合の注意事項

---

1. **該当製品**
   M3T-CC32R V.1.00 Release 1 ～ V.3.20 Release 1

2. **内容**
   volatile修飾された配列の要素、構造体のメンバまたは共用体のメンバを、 2回以上続けて読み出すと、volatile修飾が無効になる場合があります。
   また、volatile修飾に加えてconst修飾されている場合は、以下のメッセージが表示され、内部エラーが発生する場合があります。

   cg32r: "xxxx", line XX: internal error: illegal IL, refer of tuse null.
   (xxxx はファイル名。XX は行番号。)

   2.1 発生条件
   以下の条件をすべて満たす場合に問題が発生します。
   (1) 最適化オプション-O7,-O6,-O5 および-O4のいずれかを指定している。
   または、-Ospaceまたは-Otimeを単独で指定している。
   (2) 以下のいずれかのオブジェクトを、2回以上続けて読み出している。
   (a) 配列の要素
   (b) 構造体のメンバ
   (c) 共用体のメンバ
   (3) (2)の読み出しと読み出しの間に、以下のいずれの処理も存在し

ない。
(a) ポインタが指すメモリ空間への書き込み
(b) 関数呼び出し

(4) (2)のオブジェクトは、以下のいずれかの修飾がされている。
(a) const修飾およびvolatile修飾
(b) volatile修飾されているが、const修飾はされていない。

注意：
(4)(a)に該当する場合、内部エラーが発生します。
(4)(b)に該当する場合、volatile修飾が無効になります。

2.2 発生例

以下の例1および例3では、インターナルエラーが発生します。
例2では、volatile修飾が無効になります。

例1 --- インターナルエラーになる例
（ソースファイル sample1.c]

```
---------------------------------------------------------------------
extern const volatile int array1[4]; /* 発生条件(2)(a)および(4)(a) */
int a1, b1;

void func1(int index)
{
   a1 = array1[index];          /* 発生条件(3) */
   b1 = array1[index];          /* 発生条件(3) */
}
---------------------------------------------------------------------
```

例2 --- volatile修飾が無効になる例
（ソースファイル sample2.c]

```
---------------------------------------------------------------------
extern volatile int array2[4];     /* 発生条件(2)(a)および(4)(b) */
int a2, b2;

void func2(int index)
{
   a2 = array2[index];          /* 発生条件(3) */
   b2 = array2[index];          /* 発生条件(3) */
}
---------------------------------------------------------------------
```

例3 --- インターナルエラーになる例
（ソースファイル sample3.c]

```
------------------------------------------------------------------
struct {
   const volatile int var;        /* 発生条件(2)(b)および(4)(a) */
} data3;

int func3(void)
{
   return (data3.var + data3.var);   /* 発生条件(3) */
}
------------------------------------------------------------------
```

3. **回避策**

以下のいずれかの方法で回避できます。

(1) コンパイラをV.4.00 Release 1 以降にバージョンアップする。

(2) レベル4の最適化を抑止する。
-O7,-O6,-O5および-O4のいずれかを指定している場合は-O3,-O2,-O1および -O0のいずれかに変更する。
-Ospaceまたは-Otimeを指定している場合は、同時に-O3,-O2,-O1,および -O0のいずれかを指定する。

(3) 発生条件(2)に該当する読み出しを、ポインタによる間接参照に変更する。

発生例1 の回避例

```
------------------------------------------------------------------
extern const volatile int array1[4];
int a1, b1;

void func1(int index)
{
   const volatile int *ptr; /* array1[index] へのポインタを作成 */

   ptr = &array1[index];  /* array1[index] へのアドレスを設定 */
   a1 = *ptr;           /* ポインタ間接に変更 */
   b1 = *ptr;           /* ポインタ間接に変更 */
}
------------------------------------------------------------------
```

発生例2 の回避例

```
------------------------------------------------------------------
extern volatile int array2[4];
int a2, b2;
```

```
void func2(int index)
{
   volatile int *ptr;     /* array2[index] へのポインタを作成 */

   ptr = &array2[index];  /* array2[index] へのアドレスを設定 */
   a2 = *ptr;             /* ポインタ間接に変更 */
   b2 = *ptr;             /* ポインタ間接に変更 */
}
```

発生例3の回避例

```
struct {
   const volatile int var;
} data3;

int func3(void)
{
   const volatile int *ptr; /* data3.var へのポインタを作成 */

   ptr = &data3.var;        /* data3.var へのアドレスを設定 */
   return (*ptr + *ptr);    /* ポインタ間接に変更 */
}
```

---

4. **恒久対策**

本問題は、M3T-CC32R V.4.00以降で改修されています。
