RENESAS ツールニュース

RENESAS TOOL NEWS 2010年10月01日：101001/tn2

M32Rファミリ用C/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/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。

- const修飾されたauto変数に関する注意事項

---

1. 該当製品

M32Rファミリ用C/C++コンパイラパッケージ
V.4.00 Release 1 ～ V.5.01 Release 01

2. 内容

関数内のconst修飾されたauto変数を読み出すコードを実行すると、マイコンのアドレス例外が発生する場合があります。

2.1 発生条件

以下のすべての条件を満たす場合に発生します。
(1) コンパイル時の最適化オプションが以下のいずれかを満たす。
　(a) -O2, -O3, -O6, -O7および -Oのいずれかを選択している。
　(b) -Ospaceおよび -Otimeのいずれかを選択し、かつ
　　-O0, -O1, -O4および-O5 を選択していない。
(2) const修飾されていて、かつ記憶クラスがautoである変数を
　初期化式付きで定義している関数がある。
(3) (2)の変数は、2バイトあるいは1バイトの整数型である。
(4) (2)の変数はvolatile修飾されていない。
(5) (2)の定義において、初期化式と変数の間で、型が一致しないか、
　または修飾子が一致していない。
　注： 初期化式にキャストを含む場合は、キャストを除外して判断する。
(6) (2)の変数は、関数内の3箇所以上で参照されている。
(7) (2)の変数は、4の倍数でないオフセットのスタックに割り付けられている。
　注： 以下の方法で確認できます。
　　-CSまたは-csオプションを使用して、Cソース行を含むアセンブリ
　　言語ソースファイルを出力する。

このファイルでは、アセンブリコードの各関数の先頭に、
;[ASSIGN] 変数名に続けて、割り付け先の情報を以下のように
表示します。

割り付け先がレジスタの場合: Rn
注： nはレジスタ番号
割り付け先がスタックの場合: @(off,R15)
注： offはスタック上のオフセット
例：

```
-----------------------------------------
;[ASSIGN] var02 R9        ;var02はレジスタR9に割付け
;[ASSIGN] var01 @(31,R15)  ;var01はオフセット31のスタックに割付け
-----------------------------------------
```

2.2 発生例

以下の例の変数var01が本件に該当します。

ソース例： sample.c

```
-----------------------------------------------------------------
unsigned char gv01,gv02;
unsigned char gfunc03(void);
int table[100];
unsigned char buf[100];
void func1(void)
{
   int i, j;
   int d1 = 800, d2 = 500;
   const unsigned char var01 = gv01;   /* 条件(2)(3)(4)および(5) */
   unsigned char *ptr = &buf[d1];
   const unsigned char var02 = gv02;
   int d3 = 0, d4 = 1, d5 = 0;
   unsigned char var03 = gfunc03();
   *ptr++ = var02 & 0xf;
   *ptr++ = (var02>>4) & 0x0f;
   *ptr++ = var01 & 0xf;              /* 条件(6) */
   *ptr++ = (var01>>4) & 0x0f;        /* 条件(6) */
   for (i = 0; i < var01; i++) {      /* 条件(6) */
     d1 = buf[i] * d3;
     *ptr++ = buf[i]/d4;
   }
   for (i = 0; i < var02; i++) {
     d3 = buf[i] * d5;
     *ptr++ = buf[i]/d2;
   }
   *ptr++ = d1 + d2 + d3;
```

```
}
```

コマンドライン例：

```
cc32R -CS -O7 sample.c    (条件(1))
```

-CSオプションを使用して出力したアセンブリ言語ソースファイル例：
(sample.msファイルの抜粋)

```
. . . . . .
;[ASSIGN] var01 @(31,R15)    ; 条件(7) 31が4の倍数でないため該当
. . . . . .
;[ASSIGN] var02 R9
. . . . . .
$func1:
. . . . . .
;[LINE] 16 "sample.c"
;[SOURCE]     *ptr++ = var01 & 0xf;
     LD       R3,@(31,R15)          ; アドレス例外発生箇所
. . . . . .
```

この例では、var01はすべての発生条件を満たすため該当します。
var02はソースコード上はvar01と同じ条件ですが、レジスタR9に割付けられているため、条件(7)を満たさず該当しません。
また、var03はconst修飾がなく、条件(2)を満たさないため該当しません。

3. 回避策

以下のいずれかの方法で回避してください。

(1) 該当する変数の型を4バイト整数 (int または long) に変更する。

発生例の回避例：

```
const unsigned char var01 = gv01;
        ↓
const unsigned int var01 = gv01;
```

---

(2) 該当する変数からconst修飾を外す。

発生例の回避例：

```
const unsigned char var01 = gv01;
        ↓
     unsigned char var01 = gv01;
```

[免責事項]
過去のニュース内容は発行当時の情報をもとにしており、現時点では変更された情報や無効な情報が含まれている場合があります。
ニュース本文中のURLを予告なしに変更または中止することがありますので、あらかじめご承知ください。