# RENESAS ツールニュース

**RENESAS TOOL NEWS 2011年06月01日：110601/tn7**

# M16Cシリーズ、R8Cファミリ用C/C++コンパイラパッケージ (M3T-NC30WA) V.6.00 Release 00ご使用上のお願い

M16Cシリーズ、R8Cファミリ用C/C++コンパイラパッケージ (M3T-NC30WA) V.6.00 Release 00の使用上の注意事項を連絡します。

- 引数に浮動小数点定数を使用する場合の注意事項
- static指定子を使用した関数を、割り込み関数として定義する場合の注意事項

これらの注意事項は、製品に添付しているリリースノートの3章「注意事項」にも記載しています。

## 1. 引数に浮動小数点定数を使用する場合の注意事項

### 1.1 内容

関数の引数に、負の浮動小数点定数を符号無し整数型にキャストする式を記述すると、以下の警告が出力されキャスト後の値がゼロになります。

```
sample.c(6) : C1841 (W) underflow in floating value
  converting to integer
    ===>    int j = func((unsigned int)-1.0);
```

### 1.2 発生条件

以下の条件をすべて満たす場合に発生します。

(1) 関数の引数が、負の浮動小数点型定数である。

注： この定数には最適化によって定数に置き換わる変数または式も含む

(2) (1)の定数を以下のいずれかの型にキャストしている。

- char
- unsigned char
- unsigned short
- unsigned int
- unsigned long
- unsigned long long

発生例:

```
------------------------------------------------------------------
#include <stdio.h>
int func(int x) { return x; }
void main(void)
 {
   int i = (unsigned int)-1.0;
   int j = func((unsigned int)-1.0);  /* 発生条件(1)および(2) */
   if (i != j) {
      printf("NG (i, j) = (%d, %d)¥n", i, j); /* -1, 0になる */
   } else {
      printf("OK¥n", i, j);
   }
}
------------------------------------------------------------------
```

**1.3 回避策**

発生条件(1)の関数を呼び出す前に発生条件(2)のキャストした定数を
一時変数に代入し、その一時変数を関数の引数に使用してください。
例:

```
------------------------------------------------------------------
#include <stdio.h>
int func(int x) { return x; }
void main(void)
{
   int i = (unsigned int)-1.0;
   unsigned int tmp = (unsigned int)-1.0;    /* 一時変数の定義 */
   int j = func(tmp);
   if (i != j) {
      printf("NG (i, j) =1(%d, %d)¥n", i, j);
   } else {
      printf("OK¥n", i, j);
   }
}
------------------------------------------------------------------
```

**1.4 恒久対策**

次期バージョンで改修する予定です。

**2. static指定子を使用した関数を、割り込み関数として定義する場合の注意事項**

**2.1 内容**

割り込みベクタ番号を記述した#pragma interruptを使用して割り込み
関数を定義した場合、その関数に対して、コードを出力せず可変割り込み

ベクタテーブルに関数アドレスを設定しない場合があります。
その結果、割り込み発生時にこの関数を呼び出せない場合があります。

### 2.2 発生条件

以下の条件をすべて満たす場合に発生します。
(1) コンパイルオプション -OS_MAX (-OSM)を使用している。
　もしくは、-Oforward_function_to_inline (-OFFTI) と
　以下のいずれかを使用している。
　　-O, -O1 ～ -O5, -OR_MAX (-ORM), -OR, -OS
(2) static記憶クラス指定子を使用して関数を定義している。
(3) (2)の関数に割り込みベクタ番号を記述した#pragma interruptを
　使用している。
(4) (2)の関数の呼び出しやアドレス参照がない。

発生例:

```
----------------------------------------------------------------
#pragma interrupt func(vect=31)   /* 発生条件(3) */
static void func(void)            /* 発生条件(2) */
{
}
----------------------------------------------------------------
```

参照のないstatic関数を削除してしまうため、関数に対するコードを生成せず、
可変割り込みベクタに関数アドレスも設定しません。このため割り込み
発生時に、当該関数を呼び出せません。

### 2.3 回避策

-OS_MAX (-OSM) または -Oforward_function_to_inline (-OFFTI) を使用しな
いでください。

### 2.4 恒久対策

次期バージョンで改修する予定です。