RENESAS ツールニュース

RENESAS TOOL NEWS 2015年11月01日：151101/tn2

RL78ファミリ用Cコンパイラパッケージのご使用上のお願い

RENESAS TOOL NEWS 資料番号 151001/tn2 で連絡した以下2件の注意事項について、 再度連絡します。

- 項番1(CCRL#002): 1.3項(2)の発生条件の範囲を狭くしました。
- 項番3(CCRL#004): 3.3項(1)の対象関数についての表現を変更しました。

項番2(CCRL#003) および 項番4(CCRL#005)は、変更ありません。

1. スタックに退避した引数の値を上書きするコードを出力する注意事項 (CCRL#002)
2. memcmp関数, _COM_memcmp_ff関数, strcmp関数, _COM_strcmp_ff関数の戻り値が 不正となる注意事項 (CCRL#003)
3. strtoul関数, _COM_strtoul_ff関数の戻り値が不正となる注意事項 (CCRL#004)
4. 予約語__sectop, 予約語__secend, 演算子startof, 演算子sizeofでデフォルト ではないセクション名を使用した場合の注意事項 (CCRL#005)

注: 注意事項の後ろの番号は、注意事項の識別番号です。

---

1. スタックに退避した引数の値を上書きするコードを出力する注意事項 (CCRL#002)

1.1 該当製品およびバージョン

CC-RL V1.01.00

1.2 内容

関数呼び出し時に、スタックに退避した引数の値が上書きされるコードを出力する場合があります。

1.3 発生条件

以下の(1)～(3)の条件を全て満たす場合に発生する場合があります。

(1) -Onothing オプションを指定していない。
(2) 以下(2-1)～(2-3)のいずれかに該当している引数を複数個持つ関数が存在する(同じ条件を複数個の場合も含む)。
    (2-1) 実引数のサイズが1バイト(char型、構造体、共用体など)である。
    (2-2) 実引数のサイズが3バイト(char型、構造体、共用体など)である。
    (2-3) farポインタである。
(3) (2)の引数は非volatile修飾である。

発生条件例: -Onothing オプションを指定していない場合

```
void func0(void);
void func1(unsigned char c);
unsigned char data[2];
void func(unsigned char p1, unsigned char p2) {    // 発生条件(2-1)(3)
    func0();
    func1(data[p2]);
    data[1] = data[p2];
    data[0] = p1;
}
```

上記発生例の場合の出力コード例:

```
_func:
    .STACK _func = 6
    push ax                    ; (a) p1, p2をスタックに退避
    call !!_func0
    mov a, [sp+0x00]
    shrw ax, 8+0x00000
    addw ax, #LOWW(_data)
    movw [sp+0x00], ax         ; (b) [SP+1]を上書き
    movw de, ax
    mov a, [de]
    call !!_func1
    pop de
    push de
    mov a, [de]
    mov !LOWW(_data+0x00001), a
    mov a, [sp+0x01]           ; (c) データを参照
    mov !LOWW(_data), a
    pop hl
    ret
```

(a) p1, p2をスタックに退避
    上位および下位バイトに異なる引数(p1およびp2)が割り当てられたレジスタ

ペアAXを関数の前処理でスタックに退避します。
(b) [SP+1]を上書き
movw命令またはpush命令で[SP+0]の領域に値を書き込む際に、[SP+1]を上書きします。
(c) データを参照
上書きされた[SP+1]を参照します。

1.4 回避策
以下のいずれかの方法で回避できます。
(1) -Onothing オプションを指定する。
(2) 発生条件(2)に該当しない引数の型や個数に変更する。
(3) 引数にvolatile修飾子をつける。

1.5 恒久対策
次期バージョンで改修する予定です。

2. memcmp関数, _COM_memcmp_ff関数, strcmp関数, _COM_strcmp_ff関数の戻り値が不正となる注意事項 (CCRL#003)
2.1 該当製品およびバージョン
CC-RL V1.00.00 ～ V1.01.00

2.2 内容
memcmp関数, _COM_memcmp_ff関数, strcmp関数, _COM_strcmp_ff関数を使用して引数の比較を行う際に、戻り値が不正となる場合があります。

2.3 発生条件
以下(1)と、(2)または(3)のいずれかを満たす場合に発生します。
(1) 以下のいずれかの関数を使用して引数の比較を行っている。
- memcmp(s1, s2, n)
- COM_memcmp_ff(s1, s2, n)
- strcmp(s1, s2)
- _COM_strcmp_ff(s1, s2)
(2) s1の文字コードが0x80以上で、s2の文字コードとの差が0x80以上である。
(3) s2の文字コードが0x80以上で、s1の文字コードとの差が0x80より大きい。

発生条件例:
```
#include
int x1, x2, x3;
void func(void)
{
   x1 = strcmp("¥xc0", "¥x3e");  // 発生条件(1)(2)
                                 // x1の値が正ではなく負になります
   x2 = strcmp("¥xc0", "¥x40");  // 発生条件(1)(2)
```

```
                                    // x1の値が正ではなく負になります
    x3 = strcmp("¥x40", "¥xc2");  // 発生条件(1)(3)
                                    // x3の値が負ではなく正になります
}
```

2.4 回避策

回避策はありません。

2.5 恒久対策

次期バージョンで改修する予定です。

3. strtoul関数, _COM_strtoul_ff関数の戻り値が不正となる注意事項 (CCRL#004)

3.1 該当製品およびバージョン

CC-RL V1.00.00 ～ V1.01.00

3.2 内容

strtoul関数, _COM_strtoul_ff関数を使用して文字列を整数に変換する際に、
戻り値が不正となる場合があります。

3.3 発生条件

以下(1)～(2)の全てを満たす場合に発生します。

(1) 以下のいずれかの関数を使用して文字列を変換している。

- strtoul(nptr, endptr, base)
- COM_strtoul_ff(nptr, endptr, base)

(2) 変換対象文字列nptrがマイナス符号付きの文字列であり、変換後の値が
表現可能な範囲外である。

発生条件例:

```
#include
char *endptr;
unsigned long ans;
void func(void)
{
    ans = strtoul("-4294967300", &endptr, 10);  // 発生条件(1)(2)
}
```
---

ansの値がULONG_MAXではなく1(-ULONG_MAX)になります。

3.4 回避策

回避策はありません。

3.5 恒久対策

次期バージョンで改修する予定です。

4. 予約語__sectop, 予約語__secend, 演算子startof, 演算子sizeofでデフォルトではないセクション名を使用した場合の注意事項 (CCRL#005)

4.1 該当製品およびバージョン

CC-RL V1.00.00 ～ V1.01.00

4.2 内容

予約語__sectop, 予約語__secend, 演算子startof, 演算子sizeofでデフォルトではないセクション名を使用した場合、セクション中のラベルのアドレスが不正となり、それ以降に配置されているラベル および シンボルのアドレスも不正となります。

4.3 発生条件

以下の(1)～(3)の条件を全て満たす場合に発生します。

(1) 以下の予約語または演算子において、デフォルトではないセクション名を記述している。

- C言語記述の場合
  予約語__sectop または 予約語__secend
- アセンブリ言語記述の場合
  演算子startof または 演算子sizeof

(2) (1)のセクションの中でラベルを定義している。

(3) アセンブリ言語記述の場合、(1)の記述よりも下に(1)の名前のセクションを定義している。

発生条件例:

```
#pragma section bss BSEC1
unsigned char *ucp1;  // 発生条件(2)
int sym1;             // 発生条件(2)
void func(void)
{
   ucp1 = (unsigned char *)__sectop("BSEC1_n");  // 発生条件(1)
}
```

上記発生例の場合の出力リンク・マップ・ファイル例:

```
*** Mapping List ***

SECTION                    START      END        SIZE   ALIGN

BSEC1_n
                           000f9f0e  000f9f11        4   2
```

```
*** Symbol List ***

SECTION=BSEC1_n
FILE=DefaultBuild¥r_main.obj
                    000f9f0e  000f9f11        4
 _ucp1
                    000f9f12        2   data ,g        1
 _sym1
                    000f9f14        2   data ,g        0
```

---

ucp1以降のラベル、シンボルのアドレスが不正となります。

4.4 回避策

予約語__sectop, 予約語__secend, 演算子startof および 演算子sizeofを使用するプログラムとセクション定義を異なるファイルに記述してください。

4.5 恒久対策

次期バージョンで改修する予定です。
