【注意事項】
RX ファミリ用 C/C++コンパイラパッケージ

R20TS0122JJ0100
Rev.1.00
2016.12.16 号

概要

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

1. 配列要素、構造体メンバまたは共用体メンバへの値の更新が反映されない注意事項(No.41)

注: 注意事項の後ろの番号は、注意事項の識別番号です。

1. 配列要素、構造体メンバまたは共用体メンバへの値の更新が反映されない注意事項(No.41)

1.1 該当製品

CC-RX V2.00.00 ～ V2.05.00

1.2 内容

配列要素、構造体メンバまたは共用体メンバの値の更新が反映されない場合があります。

1.3 発生条件

以下の(1)～(5)の条件をすべて満たす場合に、(3)で参照される配列要素、構造体メンバまたは共用体メンバの値が、(4)で更新される前の値となる場合があります。

(1) -optimize=2 もしくは -optimize=max オプションを指定している または 最適化レベルを指定していない。

(2) 配列要素、構造体メンバまたは共用体メンバを参照している。構造体または共用体メンバの場合、さらに次の(2-a)(2-b)の条件も満たしている。

(2-a) 構造体の場合

最初のメンバが配列である。

(2-b) 共用体の場合

配列のメンバを含んでいる。

(3) (2)の参照後に(2)と同じ配列要素、構造体メンバまたは共用体メンバを参照している。なお、(2)の参照がループ中にある場合、ループによって繰り返し実行される(2)の参照も含む。また、呼び出した関数の中にある参照も含む。

(4) (2)と(3)の参照の間で、(2)の参照と同じ配列要素、構造体メンバまたは共用体メンバの値を更新している(注 1)。

注 1：以下の(4-1)～(4-4)の条件をすべて満たしている更新時のみ該当します。

(4-1) 以下のいずれかを使用して更新している。

(4-1-a) 配列の場合

・配列アドレス

または

・配列に含まれるいずれかの要素のアドレスを指すポインタ

(4-1-b) 構造体の場合

・構造体アドレス

または

・構造体に含まれるいずれかのメンバのアドレスを指すポインタ

(4-1-c) 共用体の場合

・共用体アドレス

または

・共用体に含まれるいずれかのメンバのアドレスを指すポインタ

(4-2) (2)と(3)で参照している配列要素、構造体メンバまたは共用体メンバとは異なる配列要素、構造体メンバまたは共用体メンバから更新を開始している。

(4-3) 更新を開始する配列要素、構造体メンバまたは共用体メンバのサイズ(注2)単位で更新を行っている。
注 2：構造体メンバまたは共用体メンバが配列型である場合、その要素型のサイズも含みます。

(4-4) (2)(3)で参照する配列要素、構造体メンバまたは共用体メンバと、(4)で更新を開始する配列要素、構造体メンバまたは共用体メンバにおいて、以下の(4-4-a)と(4-4-b)のいずれかの関係が成立している。

(4-4-a) “アドレスの大きい方に配置されている要素またはメンバの先端アドレス”－“アドレスの小さい方に配置されている要素またはメンバの終端アドレス”＞ (3)で参照しているサイズ

(4-4-b) “アドレスの大きい方に配置されている要素またはメンバの先端アドレス”－“アドレスの小さい方に配置されている要素またはメンバの終端アドレス”＞ (4)で一度に更新される領域のサイズ

(5) (2)と(3)の参照の間で、(2)の参照と同じ配列要素、構造体メンバまたは共用体メンバの値の更新が(4)の他にない。

1.4 発生例

```
1:  #include <string.h>
2:  typedef struct test {
3:      unsigned char  param01[8];    // 発生条件(4-2)：更新を開始する構造体メンバ
4:      unsigned char  param02;
5:      unsigned short param03;
6:      unsigned int   param04;       // 発生条件(2)(3)：参照する構造体メンバ
7:  } test_t;
8:
9:  int test_func(void) {
10:   test_t        t1;
11:   test_t        t2;
12:   int          ret = 1;
13:   unsigned char  *src;
14:   unsigned char  *dst;
15:   int          size;
16:   memset(&t1, 0, sizeof(test_t));
17:   if (t1.param04 == 0) {               // 発生条件(2)
18:        t2.param04 = 10;
19:        src = (unsigned char *)&t2;
20:        dst = (unsigned char *)&t1;
21:        size = sizeof(test_t);
22:        while (size-- > 0){
23:            *dst++ = *src++;      // 発生条件(4)：構造体変数 t1 の更新
24:        }
25:   }
26:   if (t1.param04 < 5) {          // 発生条件(3)
27:        ret = -1;
28:   }
29:   return(ret);
30: }
```

23 行目は構造体アドレスを使用して更新していますので発生条件(4-1)に該当します。また、発生条件(2)(3)で参照している t1.param04 の先端アドレスと、発生条件(4-2)で更新を開始している t1.param01 の終端アドレスの差は、3byte (構造体のパッキングを行なう場合) または 4byte (構造体のパッキングを行なわない場合)です。従って、発生条件(4-3)で更新されるサイズ(unsigned char 型の 1 バイト)よりも大きいため発生条件(4-4)に該当します。つまり 23 行目は発生条件(4)に該当する更新です。

また、t1.param04 に対する 17 行目の参照から 26 行目の参照までに発生条件(4)の更新しか存在しないため発生条件(5)にも該当します。

この結果、26 行目で参照する t1.param04 の値には、22～24 行目の更新が反映されません。

1.5 回避策

以下のいずれかにより回避可能です。

(1) 最適化レベル-optimize=0 または -optimize=1 を指定する。

(2) 発生条件(4)における更新を以下のいずれかに変更する。

・構造体の場合、構造体代入に変更する。

【変更前】

```
19:         src = (unsigned char *)&t2;
20:         dst = (unsigned char *)&t1;
21:         size = sizeof(test_t);
22:         while (size-- > 0){
23:             *dst++ = *src++;
24:         }
```

【変更後】

```
19:         t1 = t2;
```

・memcpy ライブラリ関数の呼び出しに変更する。

【変更前】

```
22:         while (size-- > 0){
23:             *dst++ = *src++;
24:         }
```

【変更後】

```
22:         memcpy(dst, src, size);
```

1.6 恒久対策

次期バージョンで改修予定です。

以上

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2016.12.16	-	新規発行

ルネサスエレクトロニクス株式会社

〒135-0061　東京都江東区豊洲 3-2-24（豊洲フォレシア）

■総合お問い合わせ先

http://japan.renesas.com/contact/