【注意事項】
RX ファミリ用 C/C++コンパイラパッケージ

R20TS0073JJ0100
Rev.1.00
2016.09.01 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. 2 回以上のループを含むプログラムに関する注意事項 (No.39)
2. 関数の引数をスタックで渡す際の注意事項(No.40)

注: 注意事項の後ろの番号は、注意事項の識別番号です。

1. 2 回以上のループを含むプログラムに関する注意事項 (No.39)

1.1 該当製品

CC-RX V2.00.00 ～ V2.05.00

1.2 内容

最適化オプション指定時に、C ソースファイル上の 2 回以上ループするプログラム記述に対して、1 回しかループしないコードを出力する場合があります。

1.3 発生条件

以下の条件をすべて満たす時に発生する場合があります。

(1) 最適化オプション-optimize=0 および -optimize=1 を指定していない。

(2) 「ループ制御変数を含む式 > 判定値を含む式」の形のループ判定式がある。
「式」は、ループ制御変数のみ、判定値のみの記述も含む。

(3) 条件(2)の「ループ制御変数」は signed 型の変数であり、volatile 変数ではない。

(4) 条件(2)の「判定値」は定数ではなく、volatile 変数でもない。

(5) 条件(4)の値が条件(3)の型の最小値であり、ループ中で不変である。

```
1:  unsigned long count;
2:  signed char var1, var2;                          /* 発生条件(3)(4) */
3:  void func(void)
4:  {
5:     count = 0x00000000;
6:     var2 = -128;                                  /* 発生条件(5) */
7:     for( var1 = 127; var1 > var2; var1--){count++;} /* 発生条件(2) */
8:     if(0x000000FF != count){funcsub();}
9:     …
10: }
```

7 行目は 255 回のループとなり、変数”count”は 255 となるプログラムですが、最適化により 1 回のループとなります。

その結果、8 行目以降で変数”count”は 1 として動作します。

1.4 回避策

以下のいずれかにより回避可能です。

(1) 不具合発生条件を満たさないよう最適化オプションを変更する。

(2) ループ判定式の形を変更して、発生条件を満たさないようにする。

```
for( var1 = 127; var2 < var1; var1--){count++;}
```

```
for( var1 = 127; var1 >= var2+1; var1--){count++;}
```

(3) ループ制御変数を、同じサイズの unsigned 型に変更するか、volatile 変数にする。
ループ制御変数を unsigned 型に変更した場合は、初期値や判定値等の型も合わせて変更する。

```
volatile signed char var1;
```

(4) 判定値を定数に変更するか、volatile 変数にする。

```
volatile signed char var2;
```

1.5 恒久対策

次期バージョンで改修予定です。

2. 関数の引数をスタックで渡す際の注意事項 (No.40)

2.1 該当製品

CC-RX V2.00.00 ～ V2.05.00

2.2 内容

関数の引数をスタックで渡す際のスタック上の引数の配置が、マニュアルに記載の関数呼び出しインタフェースと異なることがあります。

呼び出し側の関数と呼び出される側の関数を異なるバージョンまたはリビジョンのコンパイラでコンパイルした場合、生成したオブジェクトファイル同士を結合するとプログラムが正常に動作しないことがあります。

2.3 発生条件

以下の条件(1)～(4)をすべて満たす時に、関数呼び出しインタフェースと異なるコードを生成します。

(1) 引数の中に、以下のいずれかのデータ型のレジスタ渡し引数（以降これを引数[a]と呼ぶ）がある。
- long long, unsigned long long,
- double, long double（この 2 つは dbl_size=8 を指定した場合のみ該当）

(2) 引数[a]より右側のどこかに、汎用レジスタ R4 でレジスタ渡しとなる引数（以降これを引数[b]と呼ぶ）がある。

(3) 引数[b]の右側に、スタック渡しとなる引数（以降これを引数[c]と呼ぶ）がある。

(4) 引数[b]の左側のどこかに、スタック渡しとなる引数が 1 個以上ある

上記(1)～(4)の条件を満たした場合、下記の現象となります。

【V2.01.00 以前のバージョンの場合】
スタック上の引数の割り付け領域において、条件(4)の引数のうち、構造体型もしくは共用体型の引数すべてが、引数[c]より下位アドレスに配置されるべきであるが、引数[c]より上位アドレスに配置され、関数呼び出しインタフェースと異なる。

【V2.02.00 以降のバージョンの場合】
スタック上の引数の割り付け領域において、条件(4)の引数すべてが、引数[c]より下位アドレスに配置されるべきであるが、引数[c]より上位アドレスに配置され、関数呼び出しインタフェースと異なる。

さらに、上記(1)～(4)に加えて以下の条件を満たす場合は、下記の現象となります。
- 引数[b]が 8 バイトのスカラ型であり、引数[c]の右側にスタック渡しとなる引数（以降これを引数[d]と呼ぶ）がある。

【V2.01.00 以前のバージョンの場合】
スタック上の引数の割り付け領域において、条件(4)の引数のうち、構造体型もしくは共用体型の引数すべてが、引数[c]、引数[d]より下位アドレスに配置されるべきであるが、引数[c]、引数[d]より上位アドレスに配置され、関数呼び出しインタフェースと異なる。

【V2.02.00 以降のバージョンの場合】
スタック上の引数の割り付け領域において、条件(4)の引数すべてが、引数[c]、引数[d]より下位アドレスに配置されるべきであるが、引数[c]、引数[d]より上位アドレスに配置され、関数呼び出しインタフェースと異なる。

2.4 回避策

次の回避策(1) または (2)のいずれかで、プログラムの不正動作を回避することができます。

(1) プログラムを構成するすべてのオブジェクトを、V2.01.00 以前、または V2.02.00 以降のいずれか一方のバージョンのコンパイラで作成する注。
ただし、C/C++言語関数とアセンブリ言語関数を相互に呼び出し、かつ発生条件に該当している場合は、本回避策で回避することができません。
注：V2.01.00 以前で生成したオブジェクト同士、もしくは V2.02.00 以降で生成したオブジェクト同士であれば、関数呼び出しインタフェースに準拠していなくても、引数を正常に受け渡すことが可能です。そのため、プログラムの不正動作を回避することができます。

(2) 発生条件 (1) ～ (4) のいずれかの条件を満たさなくなるように引数の構成を変更する。
なお、関数呼び出しインタフェースに関しては、下記マニュアルをご参照下さい。
- CC-RX コンパイラユーザーズマニュアル
https://www.renesas.com/search/keyword-search.html#genre=document&q=r20ut3248

2.5 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	2016.09.01	-	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/