【注意事項】
RX ファミリ用 C/C++コンパイラパッケージ
(注意事項 No.66)

R20TS0906JJ0100
Rev.1.00
2023.02.16 号

概要

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

1. ループ内でデータ転送または文字列長を計算する場合の注意事項 (No.66)

注：注意事項の後ろの番号は、注意事項の識別番号です。

1. ループ内でデータ転送または文字列長を計算する場合の注意事項 (No.66)

1.1 該当製品

CC-RX V2.06.00 – V2.08.01

CC-RX V3.00.00 – V3.05.00

1.2 内容

多重ループの内側に、メモリ間のデータ転送のみを行うループ、または文字列長の計算のみを行うループを記述すると、コンパイラの生成コードが不正になる場合があります。

1.3 発生条件 1

次の条件をすべて満たす場合に発生する可能性があります。

(1) -optimize=0 オプションまたは-optimize=1 オプションを指定していない。

(2) ループ処理の中にループ処理がある。

(3) 内側のループ処理は、メモリ間のデータ転送のみを行っている。

(4) (3)で述べたループの継続条件が、次に示す (a) (b) 2 つの値の比較であり、かつ (a)が(b)より符号なし比較で小さい(※)場合に真となる。

  (a) 初期値が 0 で、ループの周回ごとに 1 加算され、型が整数型(32 ビット以下)の変数。

  (b) 整定数、またはループ中には値が変わらない変数。

  (※)：(b)が整定数の場合は、等号付きの比較演算子も含みます。

(5) (3)で述べたデータ転送は、周回あたり 1 バイトの代入を繰り返す。

(6) (5)で述べた代入について、その代入元と代入先は以下をすべて満たしている。

  (a) 代入元と代入先は重複していない。

  (b) char 型の配列ではない。

  (c) 最初のメンバーが char 型の配列の構造体ではない。

  (d) 型に修飾子 volatile または修飾子__evenaccess がついていない。

[発生例 1]　ccrx -isa=rxv1 tp1.c　　// (1)

```
/* tp1.c */
int callee(char*);
short gv[0x100];
void caller(unsigned l){
  struct{
    short value;
    char array[0x100];
  }lv;
  char* from = (char *)&lv;
  char* to = (char *)&gv;
  /* 構造体をバイト単位で配列にコピーする */
  while(callee(lv.array)){    // (2)
    unsigned i;
    for (i = 0; i < l; i++){  // (2)(4)
      to[i] = from[i];        // (3)(5)(6)
    }
  }
}
```

1.4　回避策 1

以下のいずれかを行うことで回避できます。

(1)　-optimize=0 オプションまたは-optimize=1 オプションを指定する。

(2)　発生条件 1 の(3)で述べたデータ転送に使用する変数に修飾子 volatile または修飾子__evenaccess を付与する。修飾子の付与は片方でも両方でもよい。

(3)　発生条件 1 の(3)で述べたループが行う処理を標準ライブラリ関数 memcpy()で実現する。

1.5　発生条件 2

次の条件をすべて満たす場合に発生する可能性があります。

(1)　-optimize=0 オプションまたは-optimize=1 オプションを指定していない。

(2)　ループ処理の中にループ処理がある。

(3)　内側のループは以下をすべて満たしている。

(a)　char 型または unsigned char 型の値を指示するポインタの指示先が'¥0'ではないときにループを継続する。

(b)　(a)のポインタの指示先型に修飾子 volatile や修飾子__evenaccess がついていない。

(c)　ループ処理ではポインタを 1 つ先に進めることのみ行われる。

(4) (3)で述べたループの後で、進める前後のポインタの差を計算する。

[発生例 2] ccrx -isa=rxv1 tp2.c // (1)

```
/* tp2.c */
char* callee(int);
void caller(void){
  int length = 0;
  char* string;
  while(string = callee(length)){      // (2)
    /* 文字列の終端までポインタを進めるループ */
    char* pointer = string;  // (b)
    while(*pointer){         // (2)(a)
      ++pointer;             // (c)
    }
    /* 進める前後のポインタの差を計算する処理 */
    length = pointer - string;  // (4)
  }
}
```

1.6 回避策 2

以下のいずれかを行うことで回避できます。

(1) -optimize=0 オプションまたは-optimize=1 オプションを指定する。

(2) 発生条件 2 の(3)で述べたループ処理の継続条件に使用するポインタの指示先の型に修飾子 volatile あるいは修飾子__evenaccess を付与する。修飾子の付与は片方でもよい。

(3) 発生条件 2 の(3)で述べたループが行う処理を標準ライブラリ関数 strlen()で実現する。

1.7 恒久対策

CC-RX V3.06.00 で改修する予定です。リリース時期は未定です。

以上

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	Feb.16.23	-	新規発行

本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。

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。

本社所在地

〒135-0061 東京都江東区豊洲 3-2-24（豊洲フォレシア）

www.renesas.com

お問合せ窓口

弊社の製品や技術、ドキュメントの最新情報、最寄の営業お問合せ窓口に関する情報などは、弊社ウェブサイトをご覧ください。

www.renesas.com/contact/

商標について

ルネサスおよびルネサスロゴはルネサス エレクトロニクス株式会社の商標です。すべての商標および登録商標は、それぞれの所有者に帰属します。

TS Colophon 4.3