【注意事項】
RH850 ファミリ用 C コンパイラパッケージ

R20TS0379JJ0100
Rev.1.00
2019.01.16 号

概要

RH850 ファミリ用 C コンパイラパッケージ CC-RH の使用上の注意事項を連絡します。

1. 浮動小数点型から 64bit 整数型への型変換を含んだ定数式に関する注意事項 (No.23)
2. -Xmisra2012 オプション指定時の注意事項 (No.24)

注： 注意事項の後ろの番号は、注意事項の識別番号です

1. 浮動小数点型から 64bit 整数型への型変換を含んだ定数式に関する注意事項 (No.23)

1.1 該当製品

CC-RH V1.02.00～V2.00.00

1.2 内容

浮動小数点型から 64bit 整数型への型変換を含んだ定数式の結果が間違った値になる場合があります。

1.3 発生条件

下記(1)～(5)の条件をすべて満たす場合に必ず発生します。

(1) -Xround=zero オプションを指定している。
(2) 定数式を記述している。
(3) (2)の部分式(※1)の中に浮動小数点型である float/double/long double 型のいずれかの定数式がある。
※1：(2)が単独の float/double/long double 型の定数式である場合も含む。
(4) (3)の部分式から暗黙の変換を含めて 64bit **整数型である** signed long long/unsigned long long 型への型変換がある。
(5) (3)の部分式の値が下記範囲である。
(5-1) float 型定数式、または-Xdbl_size=4 オプション指定時の double 型、long double 型定数式のいずれかを次の型に変換する場合
(5-1-a) signed long long 型に変換する場合
- 2.147483e+09 ～ 3.602880e+16
  または
- -2.147483e+09 ～ -3.602880e+16

(5-1-b) unsigned long long 型に変換する場合
- 4.294967e+09 ～ 3.602880e+16

(5-2) -Xdbl_size=8 オプション指定時の double 型、long double 型定数式のいずれかを次の型に変換する場合
(5-2-a) signed long long 型に変換する場合
- 2.147483e+09 以上
  または
- -2.147483e+09 以下

(5-2-b) unsigned long long 型に変換する場合
- 4.294967e+09 **以上**

1.4 発生例

以下に、発生例を記します。

```
long long ll = (long long)(123456789123.0+123+456);    //発生条件(2)(3)(4)(5)
```

・定数式″(long long)(123456789123.0+123+456)″を記述しているため条件(2)に該当します。
・その定数式の部分式に double 型(123456789123.0)が含まれているため条件(3)に該当します。
・long long 型に変換しているため条件(4)に該当します。
・定数式を計算した結果の 123456789702.0 が条件(5-2-a)の範囲に含まれているため該当します。

【アセンブリ・ソース(本注意事項に該当するコンパイル結果)】

```
1:  _ll:
2:          .ddw 0xFFFFFFFFBE991CC6
```

2 行目：条件(1)及び上記の発生例の条件(2)～(5)を満たすと本注意事項に該当するコンパイル結果の値になります。

【アセンブリ・ソース(正しいコンパイル結果)】

```
1:  _ll:
2:          .ddw 0x0000001CBE991CC6
```

2 行目：正しくは上記のコンパイル結果になります。

1.5 回避策

以下のいずれかにより回避可能です。

(1) 条件(2)の定数式を変換後の整数型定数で記述する。下記の【回避 C ソース例(1)】参照。
(2) 変換を定数式ではなくコードによる実行時処理として下記(2-1)または(2-2)のどちらかの方法で記述する。
  (2-1) double 型変数を経由して 64bit 整数型の変数に代入する。下記の【回避 C ソース例(2-1)】参照。
  (2-2) double 型の値を返すインライン関数呼び出しに置き換える。下記の【回避 C ソース例(2-2)】参照。

【回避 C ソース例(1)】

```
long long ll = 123456789702ll;
```

【回避 C ソース例(2-1)】

```
1:  void func(){
2:    double la = 123456789123.0+123+456; //double 型変数の la を定義
3:    long long ll = la;                  //la の値を 64bit 整数型に代入
4:  }
```

【回避 C ソース例(2-2)】

```
1:  #pragma inline dvalue
2:  static double dvalue(){
3:     return 123456789123.0+123+456;       //double 型の値を返すインライン関数を定義
4:  }
5:
6:  void func(){
7:     long long ll = dvalue();                    //インライン関数を呼び出して変数 ll に代入
8:  }
```

1.6 恒久対策

CC-RH V2.01.00 及び V1.07.01 で改修します。（1 月 21 日公開予定）

2. -Xmisra2012 オプション指定時の注意事項 (No.24)

2.1 該当製品

CC-RH V1.04.00～V2.00.00【professional 版】(ルール 16.1, 16.4)

CC-RH V1.07.00～V2.00.00【professional 版】(ルール 15.6, 15.7, 16.1, 16.2)

2.2 内容

-Xmisra2012 オプションを指定して MISRA C:2012 ルールによるソース・チェックを行う際に、ルールに違反しない記述に対してメッセージを出力する場合や、ルールに違反する記述に対してメッセージを出力しない場合があります。

MISRA-C とは、C 言語で記述する組み込みシステムで安全性・可搬性・信頼性を確保することを目的としたソフトウェア設計標準規格です。

2.3 発生条件

下記に示す 5 つのルールをチェック対象とする際に該当します。

- ルール 15.6
  -lang=c99 オプションを指定すると、ルールに違反する記述に対してメッセージを出力しません。
- ルール 15.7
  -lang=c99 オプションを指定すると、ルールに違反する記述に対してメッセージを出力しません。
- ルール 16.1
  下記条件をすべて満たす場合、ルールに違反する記述に対してメッセージを出力しません。
  (1) switch(制御式)の直後に″{″を記述する。
  (2) (1)の switch 文に case 節と default 節の両方を記述する。
  (3) (2)の case 節と default 節はすべて break 文で終了するか、break 文を最後に記述した複合文(※1)(ブロック)で終了する。
  (4) (3)の case 節や default 節の 1 つ以上が、以下の条件をすべて満たす。
  (4-1) 最後の文として選択文(if, switch)や繰り返し文(while, do-while, for)でない複合文(ブロック)を記述する。
  (4-2)(4-1)の複合文(ブロック)の前に文を記述する。
  ※1：複合文とは″{ }″で囲んだ文のことです。if 文などで″{ }″を記述した場合も複合文に該当します。
- ルール 16.2
  下記条件をすべて満たす場合に、ルールに違反する記述に対してメッセージを出力しません。
  (1) -lang=c99 オプションを指定する。
  (2) switch(制御式)の直後に″{″を記述せずに、case ラベルまたは default ラベルを記述する。
- ルール 16.4
  下記条件のいずれかを満たす場合に、ルールに違反しない記述に対してメッセージを出力する場合があります。
  (1) -lang=c を指定し、関数定義に複合文(ブロック)を記述する。
  (2) -lang=c99 を指定し、関数定義内に複合文(ブロック)、選択文(if, switch)、繰り返し文(while, do-while, for)のいずれかを記述する。
  ″{ }″がなくても選択文や繰り返し文を記述した場合は該当します。

2.4 発生例

以下に、発生例を記します。赤文字が発生条件の該当箇所です。
【C ソース】（ルール 16.1 の場合）

```
1:  int x;
2:  void func(void) {
3:      switch(x) {                          //発生条件(1)
4:      case 1:                              //発生条件(2)
5:          ++x;                             //発生条件(4-2)
6:          {                                //発生条件(4-1)
7:              --x;
8:              break;                       //発生条件(3)
9:          }                                //発生条件(4-1)
10:     default:                             //発生条件(2)
11:         break;                           //発生条件(3)
12:     }                                    //発生条件(1)
13: }
```

上記の C ソースは MISRA C:2012 のルール 16.1 に違反しますが、メッセージを出力しません。

3,12 行目：switch（制御式）の直後に″{″があるため、条件(1)に該当します。

4,10 行目：case 節と default 節の両方があるため、条件(2)に該当します。

8,11 行目：case 節と default 節は break 文で終了しているため、条件(3)に該当します。

6,9 行目：case 節の最後が複合文(ブロック)であるため、条件(4-1)に該当します。

5 行目：複合文(ブロック)の前に文があるため、条件(4-2)に該当します。

【C ソース】（ルール 16.2 の場合）

```
1:  int x;
2:    void func(void) {
3:        switch(x)                          //発生条件(2)
4:        case 1:                            //発生条件(2)
5:          break;
6:  }
```

上記の C ソースは MISRA C:2012 のルール 16.2 に違反し、-lang=c 指定時はメッセージを出力しますが、-lang=c99 指定時はメッセージを出力しません。
3,4 行目：switch(制御式)の直後に″{″を記述せずに case ラベルを記述しているため、条件(2)に該当します。

2.5 回避策

ありません。

2.6 恒久対策

CC-RH V2.01.00 及び V1.07.01 で改修します。（1 月 21 日公開予定）

以上

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2019.01.16	-	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

https://www.renesas.com/contact/