【注意事項】 RH850 ファミリ用 C コンパイラパッケージ (注意事項 No.34-38)

R20TS0792JJ0100
Rev.1.00
2022.01.16 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. 構造体型または共用体型の引数の使用に関する注意事項 (No.34)
2. ポインタ型のキャストの使用に関する注意事項 (No.35)
3. 無名共用体または無名構造体の使用に関する注意事項 (No.36)
4. アドレスを記憶領域に書き込んだ後で、読み出して参照する場合の注意事項 (No.37)
5. 静的変数の参照方法に関する注意事項 (No.38)

注：注意事項の後ろの番号は、注意事項の識別番号です。

1. 構造体型または共用体型の引数の使用に関する注意事項 (No.34)

1.1 該当製品

CC-RH V1.00.00-V2.03.00

1.2 内容

ポインタ型のメンバを持つ構造体型または共用体型の引数を使用する場合に、不正なコードを生成する場合があります。

本注意事項には 3 種類の発生条件があり、それぞれに発生例、回避策を示します。

1.3 発生条件 1

次の(1)から(7)のすべてを満たす場合に、発生する可能性があります。

(1) オプション-Onothing を指定していない。
(2) ポインタ型のメンバを持つ構造体型または共用体型の変数が存在する。
(3) (2)の構造体型または共用体型は、サイズが 8 バイト以下である。
(4) 次の(4-1)から(4-3)のすべての処理がある関数 A がある。
  (4-1) (2)のポインタ型メンバに、アドレスを代入している。
  (4-2) (4-1)の代入先の構造体型または共用体型の変数を引数に渡して関数 B を呼び出す。
  (4-3) (4-1)で代入したアドレスの指示先を参照している。
(5) (4-2)で呼び出す関数 B は、(4-2)の構造体型または共用体型引数をレジスタ渡しで受け取る。
(6) (4-2)で呼び出す関数 B の中で、(4-2)の構造体型または共用体型引数の、ポインタ型メンバの指示先を参照している。
(7) (4-2)で呼び出す関数 B をインライン展開する。

[発生例]
ccrh -Ospeed tp.c　(1), (7)

```
/* tp.c */
#include <stdio.h>
typedef struct {     // (2), (3)
  int* _pointer;
  int _value;
}myStruct;

int flg = 0;
void func( myStruct arg ) {      // (5)
  if (*(arg._pointer) != 10) {  // (6)
    flg = 1;
  }
}
```

```
void main(void) {
  int val = 10;
  volatile myStruct st;
  st._pointer = &val;      // (4-1)
  func(st);                // (4-2)
  val = 20;                // (4-3)
  if (flg == 1) {
    printf("ng¥n");
  } else {
    printf("ok¥n");
  }
}
```

この例の場合は、変数 flg が 1 になることはなく、printf により"ok"が出力されることが正しいですが、不具合により変数 val に 10 を代入するコードが削除され、その結果、”ng”が出力されます。

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。

(a) オプション-Onothing を指定する。
(b) 構造体型または共用体型のサイズを、条件(3)に該当しないように変更する。
(c) 構造体型または共用体型の引数を、レジスタ渡しにならないように調整する。
(d) インライン展開を抑止する。

1.5 発生条件 2

次の(1)から(8)のすべてを満たす場合に、発生する可能性があります。

(1) オプション-Onothing を指定していない。
(2) オプション-Xintermodule を指定している。
(3) ポインタ型のメンバを持つ構造体型または共用体型の変数が存在する。
(4) (3)の構造体型または共用体型は、サイズが 16 バイト以下である。
(5) 次の(5-1)から(5-3)のすべての処理がある関数 A がある。
  (5-1) (3)のポインタ型メンバに、アドレスを代入している。
  (5-2) (5-1)の代入先の構造体型または共用体型の変数を引数に渡して関数 B を呼び出す。
  (5-3) (5-1)で代入したアドレスの指示先を参照している。
(6) (5-2)で呼び出す関数 B は、(5-2)の構造体型または共用体型引数をレジスタ渡しで受け取る。
(7) (5-2)で呼び出す関数 B の中で、(5-2)の構造体型または共用体型引数の、ポインタ型メンバの指示先を参照している。
(8) (5-2)で呼び出す関数 B をインライン展開する。

[発生例]
ccrh -Ospeed -Xintermodule tp.c　(1), (2), (8)

```
/* tp.c */
#include <stdio.h>
typedef struct {     // (3), (4)
  int* _pointer;
  int _value1;
  int _value2;
}myStruct;

int flg = 0;
void func( myStruct arg ) {      // (6)
  if (*(arg._pointer) != 10) {  // (7)
    flg = 1;
  }
}
void main(void) {
```

```
  int val = 10;
  volatile myStruct st;
  st._pointer = &val;      // (5-1)
  func(st);                // (5-2)
  val = 20;                // (5-3)
  if (flg == 1) {
    printf("ng¥n");
  } else {
    printf("ok¥n");
  }
}
```

この例の場合は、変数 flg が 1 になることはなく、printf により”ok”が出力されることが正しいですが、不具合により変数 val に 10 を代入するコードが削除され、その結果、”ng”が出力されます。

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。

(a) オプション-Onothing を指定する。
(b) オプション-Xintermodule を指定しない。
(c) 構造体型または共用体型のサイズを、条件(4)に該当しないように変更する。
(d) 構造体型または共用体型の引数を、レジスタ渡しにならないように調整する。
(e) インライン展開を抑止する。

1.7 発生条件 3

次の(1)から(5)のすべてを満たす場合に、発生する可能性があります。

(1) オプション-Onothing を指定していない。
(2) オプション-Xintermodule を指定している。
(3) ポインタ型のメンバを持つ構造体型または共用体型が存在する。
(4) (3)の構造体型または共用体型の引数を持つ関数がある。
(5) (4)の関数の中で、(4)の構造体型または共用体型の引数のポインタ型のメンバを、次に示すいずれか、または両方により参照している。
  (5-1) 読み出しと書き込みの両方がある。
  (5-2) 書き込みが複数回ある。

[発生例]
ccrh -Ospeed -Xintermodule tp.c　　(1), (2)

```
/* tp.c */
typedef struct s_tag {  // (3)
    int *ptr ;
    int dmy1 ;
} STRCT ;
STRCT gv;
void func( STRCT arg ) { // (4)
    int i ;
    for( i = 0 ; i < 1 ; i++ ){
        (*(arg.ptr)) += 1 ;  // (5)
        (*(arg.ptr)) += 2 ;  // (5)
    }
    gv = arg;
}
```

この例の場合では、arg.ptr の指示先に+3 されることが正しいですが、不具合により arg.ptr の指示先に+2 するコードが生成されます。

1.8 回避策 3

以下のいずれかを行うことで回避できます。
(a) オプション-Onothing を指定する。
(b) オプション-Xintermodule を指定しない。

1.9 恒久対策

CC-RH V2.04.00 で改修する予定です。リリース時期は 2022 年 1 月の予定です。

2. ポインタ型のキャストの使用に関する注意事項 (No.35)

2.1 該当製品

CC-RH V1.00.00～V2.03.00

2.2 内容

ポインタ型をキャストして使用する際に、不正なコードを生成する場合があります。

2.3 発生条件

次の(1)から(6)のすべての条件を満たす場合に、発生する可能性があります。
(1) オプション-Onothing を指定していない。
(2) オプション-Xintermodule を指定している。
(3) ポインタ型の値を、ポインタ型以外の型を持つ変数に、キャストしたうえで代入している。
(4) ポインタのポインタ型の変数に、(3)代入先の変数のアドレスを代入している。（注 1）
(5) (4)のポインタのポインタ型の変数の指示先を参照している。
(6) (3)で代入したポインタ型の変数の指示先を、(5)と同じ関数内で参照している。

注 1：ポインタのポインタ型としていますが、単項*演算子が 2 つ以上重なる場合にも該当します。

[発生例]
以下に発生例を記します。
ccrh -Ospeed -Xintermodule tp.c　　(1), (2)

```
/* tp.c */
#include <stdio.h>
void test(int key){
    int gv = 0;
    volatile int variable = (int)&gv;    // (3)
    int** pointer = (int**)&variable;    // (4)
    **pointer = 1; // (5)

    if (gv == 1){    // (6)
        printf("ok");
    }
    else{
        printf("ng");
    }
}
```

この例の場合は、変数 gv に 1 が代入された結果、printf により”ok”が出力されることが正しいですが、不具合により、変数 gv が 0 であるとして gv と 0 を比較する if-else 文が削除され、”ng”が出力されます。

2.4 回避策

以下のいずれかを行うことで回避できます。

(a) オプション-Onothing を指定する。
(b) オプション-Xintermodule を指定しない。
(c) ポインタ型を使用する際に、ポインタ型以外の型にキャストして代入したうえで、ポインタのポインタ型の変数を経由して参照しない。

2.5 恒久対策

CC-RH V2.04.00 (2022 年 1 月リリース予定) で改修する予定です。

3. 無名共用体または無名構造体の使用に関する注意事項 (No.36)

3.1 該当製品

CC-RH V1.00.00～V2.03.00

3.2 内容

無名共用体型メンバを持つ構造体及び共用体が、正しく初期化されなかったり、内部エラーを生じたりすることがあります。

3.3 発生条件

次の (1) から (4) のすべてを満たす場合に発生する可能性があります。
(1) 変数または複合リテラルが存在し、次のいずれかの型を持つ。
  (1-1) 構造体型
  (1-2) 共用体型
(2) (1) の型は次のいずれかのメンバを持つ。
  (2-1) 無名構造体メンバ (注 1)
  (2-2) 無名共用体メンバ (注 2)
    注 1: ここでは構造体型メンバの宣言のうち、メンバ名を省略したもののことを指します。
    注 2: ここでは共用体型メンバの宣言のうち、メンバ名を省略したもののことを指します。
(3) (1) の変数は宣言と同時に初期化されている。
(4) 次のいずれかの条件を満たしている。
  (4-1) (1) 及び (2) の共用体型に、先頭メンバより大きなサイズのメンバが存在する。
  (4-2) (3) の初期化リストが、(1) 及び (2) の共用体型のサイズより小さなメンバを初期化している。
  (4-3) (3) の初期化リストが、(1) 及び (2) の構造体型、共用体型、及びクラス型の一部のメンバだけを初期化している。
  (4-4) -Oinline_init=on を指定している。(注 3)

注 3：-Ospeed 指定時は、-Oinline_init=on が暗黙に指定されます。

[発生例]
以下に発生例を記します。
ccrh tp.c

```
/* tp.c */
long func() {
  struct { // (1-1)
    union { // (2-2)
      long a;
    };
    union {
      long b;
    } c;
  } v = {10}; // (3), (4-3)
  return v.c.b;
}
```

この例の場合、v.a (無名共用体型メンバ) が 10 で、v.c.b が 0 でそれぞれ初期化された結果、func は 0 を返すべきです。しかし、誤って v.c.b が 10 で初期化されてしまい、func は 10 を返します。

3.4 回避策

以下のいずれかを行うことで回避できます。

(a) 発生条件 (1) に該当する変数の宣言と初期化を分けて行う。
(b) 発生条件 (4-3) に該当する場合、発生条件 (1)に該当する変数のメンバを初期化時に全て初期化する。
(c) 発生条件 (4-4) に該当する場合、オプション -Oinline_init=on を使用しない、またはオプション-Oinline_init=off を指定する。
(d) 無名共用体型または無名構造体型を使用しない。名前を付けて、共用体型または構造体型として使用する。

3.5 恒久対策

CC-RH V2.04.00 (2022 年 1 月リリース予定) で改修する予定です。

4. アドレスを記憶領域に書き込んだ後で、読み出して参照する場合の注意事項 (No.37)

4.1 該当製品

CC-RH V1.00.00～V2.03.00

4.2 内容

ポインタを、ある記憶領域に書き込んだ後で読み出して参照するとき、書き込み時の型と読み出し時の型が、ポインタ型と非ポインタ型で異なる場合に、不正なコードを生成する場合があります。

4.3 発生条件

次の (1) から (3) のすべてを満たす場合に発生する可能性があります。

(1) オプション-Onothing を指定していない。
(2) ある記憶領域に対して、(2-1)または(2-2)のどちらかを満たす。
  (2-1) アドレスを、ポインタ型の値として記憶領域に書き込み、その後で、書き込んだ値を、非ポインタ型の値として読み出し、読み出した値の型をポインタ型にした上で、その指示先を参照している。
  (2-2) アドレスを、非ポインタ型の値として記憶領域に書き込み、その後で、書き込んだ値を、ポインタ型の値として読み出し、その指示先を参照している。
(3) (2)のアドレスの指示先を、(2)に該当する書込みおよび読出しを経由せずに参照している。
(4) (2)と(3)の参照がそれぞれ 1 つの関数内にあり(インライン展開によって 1 つの関数に入る場合を含める)、そのいずれか 1 つが書き込みである。

[発生例]
以下に発生例を記します。
ccrh tp.c　　// (1)

```
/* tp.c */
#include <stdio.h>
int flg = 0;
void main(void){
    volatile union{
        int* _pointer;
        int _value;
    }myUnion;
    int val = 10;
```

```
    myUnion._pointer = &val;    // (2-1)
    if (*((int*)myUnion._value) != 10){ // (2-1)(4)
      flg = 1;
    }
    val = 20;           //(3)(4)
    if (flg == 1){
      printf("ng¥n");
    } else {
      printf("ok¥n");
    }
  }
```

この例の場合、変数 flg が 1 になることはなく、”ok”が出力されることが正しいですが、不具合により val へ 10 を代入するコードが削除され、その結果、”ng”が出力されます。

ccrh tp2.c -Ospeed -Xintermodule　// (1)

```
  /* tp2.c */
  #include<stdio.h>
  typedef struct{
    int *_pointer;
  } MyStruct2;
  void test(void){
    int autoVar = 0;
    volatile struct{
      int _value;
    } myStruct;
    volatile MyStruct2* castedStruct = (volatile MyStruct2*)(&myStruct);
    myStruct._value = (int)&autoVar;    // (2-2)
    *(castedStruct->_pointer) = 1; // (2-2)(4)
    if (autoVar == 1){ //(3)(4)
      printf("ok");
    }
    else{
      printf("ng");
    }
  }
```

このプログラムでは、autoVar が 1 になり“ok”が出力されることが正しいですが、不具合により autoVar は 0 と判断されて if-else 文が削除され、”ng”が出力されます。

4.4　回避策

以下のいずれかを行うことで回避できます。

(a) オプション-Onothing を指定する。
(b) ポインタ型の値を格納する場合には、ポインタ型として書き込み、ポインタ型として読み出して使用する。

4.5　恒久対策

CC-RH V2.04.00 (2022 年 1 月リリース予定) で改修する予定です。

5.　静的変数の参照方法に関する注意事項 (No.38)

5.1　該当製品

CC-RH V1.00.00～V2.03.00

5.2　内容

静的変数(注 1)を参照する際に、参照の方法によっては、不正なコードを生成する場合があります。

注 1：静的変数は、大域変数、static 変数が該当します。

5.3 発生条件

次の (1) から (4) のすべてを満たす場合に発生する可能性があります。
(1) オプション-Onothing を指定していない。
(2) 静的変数を(2-1)から(2-3)のいずれかの方法で参照している。
  (2-1) 静的変数そのもの。（たとえば gv）
  (2-2) 静的変数が構造体もしくは共用体である場合に、そのメンバ。(たとえば gv.member)
  (2-3) 静的変数が配列である場合に、その要素。(たとえば gv[i])
(3) (2)の静的変数を、(2-1)から(2-3)のいずれでもない方法で参照している。
(4) (2)と(3)の参照に、静的変数への書き込みを含む。

[発生例]
以下に発生例を記します。
ccrh -O tp.c　// (1)

```
/* tp.c */
#include <stdio.h>
int gv;
void test(void){
   int myVal = (int)&gv >> 1;
   gv = 0;   // (2)、(4)
   *(int*)((myVal) << 1) = 1;     //(3)、(4)
   if (gv == 1){
      printf("ok");
   }
   else{
      printf("ng");
   }
}
```

この例の場合、myVal 経由で gv に 1 が代入されて、”ok”が出力されることが正しいですが、gv は 0 であるとして if-else 文が削除され、”ng”が出力されます。

5.4 回避策

以下のいずれかを行うことで回避できます。
(a) オプション -Onothing を指定する。
(b) 発生条件(3)の参照を、(2-1)から(2-3)のいずれかの方法に変更する。

5.5 恒久対策

CC-RH V2.04.00 (2022 年 1 月リリース予定) で改修する予定です。

以上

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	Jan.16.22	-	新規発行

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/

TS Colophon 4.3