H8 C/C++コンパイラ Ver.6 使用上のご注意(3)

H8C/C++コンパイラ Ver.6 には、以下に記載しました不具合があります。H8C/C++コンパイラ Ver.6 をの場合には、ご注意いただきますようお願いいたします。
なお、本ページに記載されている不具合は、コンパイラ V.6.00.02 以降（コンパイラパッケージ V.6.00 Release 02 以降）では、全て修正されています。製品をリビジョンアップしていただきますよう、お願いいたします。

製品型名

製品型名	パッケージバージョン	コンパイラバージョン
PS008CAS6-MWR	6.0.00, 6.0.01	6.0.00, 6.0.01
PS008CAS6-SLR	6.0.00, 6.0.01	6.0.00, 6.0.01
PS008CAS6-H7R	6.0.00, 6.0.01	6.0.00, 6.0.01

1. 構造体の配列型メンバのアドレス取得不正

現象

構造体に配列型メンバを先頭以外に宣言し、配列型データのアドレスを参照する式を記述した場合、正しいアドレスを取得できない場合があります。

[例]
```
   struct S {
      int b;
      int a[8];
   } s;
   int *p;

   void test() {
      p = &s.a[7];
   }
```

〈不正コード〉
```
   _test:
   mov.l   #_s+h'e:32,@_p:32      ; &s.a[6]を指している。
   rts
```

〈正常コード〉
```
   _test:
   mov.l   #_s+h'10:32,@_p: 32      ; &s.a[7]を指す。
   rts
```

発生条件

下記の条件を全て満たす場合、発生することがあります。
(1) CPU 種別として H8SXN, H8SXM, H8SXA, H8SXX のいずれかを指定している。
(2) 構造体の先頭以外のメンバに配列型データを宣言している。
(3) 配列型メンバの先頭でない要素のアドレスを取得する式を記述している。

回避策

該当箇所が存在した場合、以下の方法で回避してください。
・ポインタ型変数に volatile 修飾子を付加して定義する。
　アドレスを取得する配列型メンバの先頭を上記変数に代入し、該当配列の要素にアクセスする。

[例]
```
   struct S {
      int b;
      int a[8];
    } s;
    int *volatile p;

    void test() {
      p = &s.a;     /*  配列型メンバ先頭のアドレスを取得 */
      p += 7;       /*  参照する要素に移動する */
    }
```

対象バージョン

Ver.6.0.00 以降

2. プレ/ポスト インクリメント/デクリメントの不正な最適化

現象

speed オプションを指定し、プレインクリメント/プレデクリメント、ポストインクリメント/ポストデクリメントのいずれかを記述した場合、正しい値が得られないことがあります。

[例]
```
void func(char *par1, char *par2)  {
  *--par1 = *--par2;
  *--par1 = *--par2;
  *--par1 = *--par2;
  *--par1 = *--par2;
}
                        〈不正コード〉
  ADD.L   #10:16,ER0     ADD.L   #10:16, ER0
  MOV.B   @-ER0,R1L      ; (不正に削除される) … (a)
  MOV.L   ER0,ER4        MOV.L     ER0, ER4    … (a)が削除されているため、
                                                 異なる値が代入される。
  MOV.L   SP,ER2         MOV.L  SP, ER2
  ADD.L   #16:16,ER2     ADD.L #16:16, ER2
  MOV.B   R1L,@-ER2      MOV.B  @-ER0,@-ER2   … (a)のコードと不正に統合される
    :                       :
```

発生条件

下記の条件を全て満たす場合、発生することがあります。

(1) CPU 種別として H8SXN, H8SXM, H8SXA, H8SXX のいずれかを指定している。

(2) speed オプションを指定している。

(3) ソース内にて以下のいずれかを記述している。

- プレインクリメント
- プレデクリメント
- ポストインクリメント
- ポストデクリメント

回避策

該当箇所が存在した場合、以下の方法で回避してください。

・当該変数に対して、volatile 修飾子を付加する。

[例]

```
  void func(volatile char *par1, volatile char *par2){
     *--par1 = *--par2;
     *--par1 = *--par2;
     *--par1 = *--par2;
     *--par1 = *--par2;
  }
```

対象バージョン

Ver.6.0.00 以降

3. table 展開方式 switch 文でレジスタ割り付け不正

現象

table 展開方式で switch 文を展開した場合、レジスタの退避/回復が正しく行われない場合があります。

[例]

```
   void main(void) {
      val=c_1;
      ans2=ga1;
      ans2 += val; /* ans2 がスタックに退避される */
      switch(val){
        case  2:  ans=0; break;
            :
        default:  ans=0; break;
      } /* ans2 がスタックから回復されないことがある */
        :
        :
    }
```

発生条件

下記の条件を全て満たす場合、発生することがあります。

(1) CPU 種別として H8SXN, H8SXM, H8SXA, H8SXX のいずれかを指定している。

(2) case=table オプションを指定している。または、case=auto オプションを指定し、switch 文が table 展開方式でされている。

(3) switch 文の直前に、レジスタ退避回復命令が出力されている。

回避策

該当箇所が存在した場合、以下の方法で回避してください。

・case=Ifthen オプションを指定する

対象バージョン

Ver.6.0.00 以降

4. 0 クリアコード不正削除

現象

分岐ごとに 0 を設定するコード(SUB 命令)を記述した場合、0 を設定するコードを不正に削除してオブジェクト不正となることがあります。

[例]

```
      sub.b R0H, R0H ; R0H に 0 を設定
              :
              :
    L55:
              :
      bls  L36
      sub.b R0H, R0H       ; R0H が 0 のままなので削除可能な命令となる。
      add.w R0, R0         ; ここで R0H の値が変わる可能性がある。
              :
    L48:
      sub.b R0H, R0H       ; 不正に削除される
      add.w R0, R0         ; R0H の値が 0 でない値の場合、オブジェクト不正となる。
```

発生条件

下記条件を全て満たす場合、発生することがあります。

(1) CPU 種別として H8/300, H8/300L のいずれかを指定している。

(2) 最適化あり(optimize=1:デフォルト)を指定している。

(3) 分岐毎に 0 クリアを行うコードを記述している。

回避策

該当箇所が存在した場合、以下の方法で回避してください。

・最適化なし(optimize=0)でコンパイルする。

対象バージョン

Ver.4.0.04 以降

5. 4byte 以下の構造体の代入式不正削除

現象

4 バイト以下の構造体変数を使用した場合に、代入式が不正に削除されることがあります。

[例]

```
   typedef struct {
      char c;
```

```
  int  i;
} ST;
   :
void func()
{
  ST lst1,lst2,lst3;
  ST lst4,lst5,lst6;
  ST lst7,lst8,lst9;
   :
  lst9.c = 1;

  switch(x){
    case 1:
      y+=x;
      switch(y){
        case 3:
          :
        lst9.c = 3; /* 代入式が不正に削除されます */
        break;
          :
    }
}
```

発生条件

下記の条件を全て満たす場合、発生することがあります。

(1) CPU 種別として H8SXN, H8SXM, H8SXA, H8SXX のいずれかを指定している。

(2) 最適化あり(optimize=1:デフォルト)を指定している。

(3) 宣言した構造体が次のいずれかの条件を満たしている。

- 構造体の先頭から 2byte の部分のいずれかを 1byte でアクセスするメンバ、もしくは構造体の先頭から 1byte 目を 2byte でアクセスするメンバが存在する。
- サイズが 2,3,4 バイトのいずれかの構造体で、2 つ以上のメンバを持ち、少なくともそのひとつのメンバがビットフィールドメンバである。

(4) (3)のメンバをアクセスしている。

回避策

該当箇所が存在した場合、以下のいずれかの方法で回避してください。

- 最適化なし(optimize=0)を指定する。
- 当該変数に volatile 修飾子を指定する。

対象バージョン

Ver.6.0.00 以降

6. 構造体メンバの参照方法混在時不正

現象

同一メモリ領域をポインタを使用した式と使用しない式で連続してアクセスした場合、 設定コードが不正に削除されることがあります。

```
[例]
  struct tag {
     long e;
   }str = {10};

   int func() {
     struct tag *p;
     long i,k = 0;

     p = &str;
     for(i=0; i<2; i++){
       str.e = i;          /* 本命令が不正に削除される */
       k += p->e;          /* str.e と p->e は同じメモリ領域を指している。 */
     }
     return k;
   }
```

発生条件

下記の条件を全て満たす場合、発生することがあります。

(1) CPU 種別として H8SXN, H8SXM, H8SXA, H8SXX のいずれかを指定している。

(2) 最適化あり(optimize=1:デフォルト)を指定している。

(3) 同じメモリ領域への値の設定/参照の式が連続して存在する。

(4) (3)の式はそれぞれポインタを使用した式とポインタを使用しない式で構成されている。

回避策

該当箇所が存在した場合、以下のいずれかの方法で回避してください。

・最適化なし(optimize=0)でコンパイルする。

・同じアクセス方法を使用して値を設定/参照する。

[例]

```
   str.e = i;
   k += str.e;
```

または、

```
   p->e = i;
   k += p->e;
```

と記述する。

対象バージョン

Ver.6.0.00 以降

7. ループ変数の 2 次式の計算で結果不正

現象

ループ内で、m*(i*i+b*i)という形のループ変数 i の 2 次式がある場合、最適化により結果不正となることあります。

[例]

```
long a[100];
f() {
  long i;
  for (i=0;i<100;i++){
    a[i] = 3 * (i * i + 555 * i); /* 3*i*i+555*i に不正に変換される*/
  }
}
```

発生条件

下記条件を全て満たす場合、発生することがあります。

(1) CPU 種別として H8SXN, H8SXM, H8SXA, H8SXX のいずれかを指定している。

(2) 最適化あり(optimize=1：デフォルト)を指定している。

(3) 下記の条件を全て満たすループが存在する。

- (a) long/unsigned long 型のループ変数がある。
- (b) (a)のループ変数 2 次式がそのループ内に存在する。
- (c) (a)のループ変数を x とすると、(b)の 2 次式が「m * (x * x + b * x)」という形をしている。

回避策

該当箇所が存在した場合、以下のいずれかの方法で回避してください

・ 最適化なし(optimize=0)を指定する。

・ 該当するループ変数に volatile 修飾子を付加して定義する。

・ 該当するループ変数を long/unsigned long 以外の型で宣言する

・ 該当する 2 次式の係数 m をあらかじめ((m*i*i) + m*b*i)に分配する。

対象バージョン

Ver.6.0.00 以降

株式会社 日立超LSIシステムズ