RENESAS ツールニュース

RENESAS TOOL NEWS 2005年07月16日：RSO-H8C-050716D

H8SX, H8S およびH8ファミリ C/C++コンパイラパッケージ V.6 ご使用上のお願い

H8SX, H8S およびH8ファミリC/C++コンパイラパッケージ V.6 の使用上の 注意事項を連絡します。

1. **該当製品**
   V.6.00 Release 00 ～ V.6.00 Release 03 およびV.6.01 Release 00

   製品型名：

Windows版	R0C40008XSW06R
Solaris版	R0C40008XSS06R
HP-UX版	R0C40008XSH06R

2. **内容**

   2.1 非数をオペランドに持つ式の注意事項 (H8C-0017)

   変数と非数との比較式および、非数との加減式で間違った値になる場合が あります。

   発生条件:
   以下の条件をすべて満たす場合に発生します。
   1. cpu=2000n,2000a,2600n,2600a,h8sxn,h8sxm,h8sxaまたは、h8sxx オプションを選択している。
   2. 以下の比較式または、加減式がある。
      - if (変数 == 非数) または if (変数 != 非数) (float型)
      - 変数+非数 または 変数-非数 (float型およびdouble型)

   例:
   ----
   float x,y;

```
const float z_nan = 0.0/0.0;
void test() {
   x = y - z_nan;   /* 発生条件2 */
                /* 非数の場合でも、x=y;になる */
}
```

回避策:
非数の変数定義を別ファイルにしてコンパイルする。

発生例の回避例
```
---------------------------------------------------
 float x,y;
 extern const float z_nan;  /* z_nanを別ファイルに定義する */
 void test() {
   . . . . . . . . . . . . . . . . . . . . . . .
 /* 別ファイル */
 const float z_nan=0.0/0.0;
```

2.2 構造体のメンバへの代入式の注意事項 (H8C-0018)

構造体サイズが4バイト以下である構造体型変数を関数内で宣言したとき、その構造体メンバへの代入式が実行されない場合があります。

発生条件:
以下の条件をすべて満たす場合に発生します。

1. cpu=2000n,2000a,2600n,2600a,h8sxn,h8sxm,h8sxaまたは、h8sxx オプションを選択している
2. optimize=1オプションを選択している
3. 構造体型変数を関数内で宣言している
4. 3.の構造体のサイズは、4バイトまたは3バイトで、以下のいずれかの順番にメンバが宣言されている (以下ではsignedおよびunsignedを省略しています。signedまたはunsignedまたは、それらの組み合わせになります)。
   -char, char, char, char
   -char, char, int/short
   -char, char, char
   -char, short/int, char (pack=1オプションを選択している場合)
5. 3.のchar型の構造体メンバに関数の返却値を代入している。
6. 3.の構造体型変数にvolatile限定子が宣言されていない。
7. 3.の同一構造体型の2つ以上の変数に引数格納用レジスタが割り付いている。
   - regparam=2オプションを選択している場合
     引数格納用レジスタ: ER0, ER1
   - regparam=3オプションを選択している場合

引数格納用レジスタ:ER0, ER1, ER2

例:

```
-------------------------------------------------------
struct _str {                   /* 発生条件4 */
   char  m_c1;
   char  m_c2;
   char  m_c3;
   char  m_c4;
};
extern char sub();
void func() {
    struct _str str1;           /* 発生条件3および 6 */
    struct _str str2;           /* 発生条件3および 6 */
    :
    str1.m_c3 = sub();          /* 発生条件5 */
                /* str1.m_c3に値が代入されない */
    :
    str2.m_c1 = sub();          /* 発生条件5 */
    :
}
-------------------------------------------------------
```

回避策:

以下のいずれかの方法で回避してください。

(a) optimize=0オプションを選択する

(b) 構造体型変数にvolatile限定子を宣言する

発生例の回避例

```
-------------------------------------------------------
volatile struct _str str1;
volatile struct _str str2;
-------------------------------------------------------
```

(c) 構造体のサイズを4バイトより大きくする

発生例の回避例

```
-------------------------------------------------------
struct _str {
   char  m_c1;
   char  m_c2;
   char  m_c3;
   char  m_c4;
   char  dummy;    // ダミーのメンバを追加
};
-------------------------------------------------------
```

2.3 配列要素への代入式の注意事項 (H8C-0019)

if文の前とthen節内で、同じ配列への代入文が存在するとき、その配列の値 が正しく代入されない場合があります。

発生条件:

以下のすべての条件を満たす場合に発生します。

1. cpu=2000n,2000a,2600n,2600a,h8sxn,h8sxm,h8sxaまたは、h8sxx オプションを選択している
2. optimize=1オプションを選択している
3. 繰り返し文の中に選択文(else節のないif文)がある
4. if文の前と(3)のthen節に同じ配列要素への代入式がある
5. if文の前の代入式の右辺を"式1"、if文内then節の代入式の右辺を"式2" として、式1、式2の型が異なっている
6. 5.の式1と式2は、以下のいずれかを満たしている
   - 式1が式2の型で表せる範囲外の値をとることがある
   - 式2が式1の型で表せる範囲外の値をとることがある

例:

```
-------------------------------------------------------
#define MAX 10
int A[MAX];
 int x, y, z;
 char sc0, sc1, sc2;
 test(){
    int i;
    for(i = 0; i < MAX; i++){
       A[i] = x + y;          /* 発生条件4, 5および 6 */
       /* if(z)がFALSEになるときに配列A[i]にx+yの値が
          正しく代入されない */
       if(z){                 /* 発生条件3 */
          A[i] = sc0;         /* 発生条件4, 5および 6 */
       }
       x++;
       y++;
       sc0++;
    }
 }
-------------------------------------------------------
```

回避策:

以下のいずれかの方法で回避してください。

(a) optimize=0オプションを選択する。

(b) then節にnop()組み込み関数を入れる。

発生例の回避例

```
-----------------------------------------------------
#include <machine.h>
. . . . . . . . . . . . . . . . . . . . . . . .
 if(z){
 nop(); //組み込み関数の追加
   A[i] = sc0;
 }
-----------------------------------------------------
```

(c) else節を用意し、nop()組み込み関数を入れる。

発生例の回避例

```
-----------------------------------------------------
#include <machine.h>
. . . . . . . . . . . . . . . . . . . . . . . .
 if(z){
 A[i] = sc0;
 } else{
 nop();          /* 組み込み関数の追加 */
 }
-----------------------------------------------------
```

(d) 式1とif文の間にnop()組み込み関数を入れる

```
-----------------------------------------------------
#include <machine.h>
. . . . . . . . . . . . . . . . . . . . . . . .
   A[i] = x + y;
   nop();          /* 組み込み関数の追加 */
   if(z){
      A[i] = sc0;
   }
-----------------------------------------------------
```

2.4 __asm内のユーザラベル記述の注意事項 (H8C-0020)

__asm{}使用時、シンボル(関数、変数およびラベル)数が256個以上ある ときに、値を参照する領域が間違っている場合があります。

発生条件:

以下の条件をすべて満たす場合に発生することがあります。

1. cpu=2000n,2000a,2600n,2600a,h8sxn,h8sxm,h8sxaまたは、h8sxx オプションを選択している。
2. プログラム内に、埋め込みアセンブル機能(__asm{})を記述している
3. __asm{}にラベル定義がある。
4. シンボル(関数、変数およびラベル)数が256個以上ある。
5. 定義された変数のうち.EXPORTされていないシンボルがある(-code=asm オプションを

選択してコンパイルし、生成されたアセンブリプログラムを確認してください)。

6. 5.のシンボルを使用している式がある。

例:

```
-----------------------------------------------------
int x000,x001,x002,x003,x004,x005,x006,x007,x008,x009;
 . . . . . .
int x250,x251,x252,x253,x254,x255;        /* 発生条件4 */
void f1(){
   x013=10;                              /* 発生条件6 */
   /* x013が.EXPORTされていない場合に、
      間違った領域に値を書き込みます */
   __asm{                                /* 発生条件2 */
      label1;                           /* 発生条件3 */
      label2;
   }
}
-----------------------------------------------------
```

回避策:

シンボル数が255以下になるようファイルを分割してコンパイルする。

2.5 アセンブラソースファイル出力時のデータ幅出力の注意事項 (H8C-0021)

-code=asmcodeオプションを選択、アセンブラ埋め込みがあり、 かつBRA/BC，BRA/BS命令のアドレッシングモードが絶対アドレスの場合に ビット幅(@aa:nのn)が出力されない場合があります。

発生条件:

以下の条件をすべて満たす場合に発生します。

1. cpu=h8sxn,h8sxm,h8sxaまたは、h8sxxオプションを選択している。
2. code=asmcodeオプションを選択している
3. #pragma asm～#pragma endasmまたは，#pragma inline_asmの アセンブラ記述がある。
4. BRA/BC、BRA/BSのコードに展開される記述がある。
5. BRA/BC、BRA/BSのアドレッシングモードが絶対アドレスで出力される。

例:

```
-----------------------------------------------------
struct ST {
   unsigned char HH:1;
   unsigned char HL:1;
};
#define STR (*(volatile struct ST *)0xFFFFA1)
volatile int dd;
```

```
void func(){
   if(STR.HL){
      return;
   }
 /*****間違った生成コード*****/
   BRA/BS      #6,@H'A1,L22
 /*****正しい生成コード*****/
   BRA/BS      #6,@H'A1:8,L22
    dd = 0;
    return;
    #pragma asm
    ;SLEEP
    #pragma endasm
}
```

回避策:
アセンブラ埋め込み部分を別ファイルにしてコンパイルする。

2.6 レジスタ値の上書きの注意事項 (H8C-0022)

注意： 本問題が発生するのは V.6.01 Release 00 のみです。
全レジスタ(ER0～ER7)を使用している関数で、レジスタに値を割り付けるとき、そのレジスタにある値を上書きする場合があります。

発生条件:
以下の条件をすべて満たす場合に発生します。

1. cpu=2000n,2000a,2600n,2600a,h8sxn,h8sxm,h8sxaまたは、h8sxxオプションを選択している。
2. 生成したコードにER0からER7までの全レジスタが使用されている。
3. レジスタに変数を割り付けるための、値の割り付いていない空き レジスタがない。

例:

```
 . . . . . .
void sub1(STRUCT *p1,STRUCT *p2,STRUCT *p3,STRUCT *p4,
STRUCT *p5,STRUCT *p6,STRUCT *p7){
   p7->data=pfunc1(p1->data,p2->data,p3->data,
   p4->data,p5->data,p6->data) +
 /*****生成コード*****/
 . . . . . .
   MOV.L       @_pfunc1:32,@(H'0010:16,SP)
   MOV.B       @(H'000F:16,SP),R0L
   MOV.B       R5H,R0H
   MOV.B       R5L,@(9:16,SP)
```

```
  MOV.B       R4H,@(8:16,SP)
  MOV.B       R4L,R2L
  MOV.B       R3H,R2H
 /*****間違った生成コード*****/
  MOV.L       @(H'0010:16,SP),ER1;関数引数として使用する
  JSR         @ER1                ;レジスタに値を上書き
 /*****正しい生成コード*****/
  MOV.B       R5L,R1L
  MOV.B       R4H,R1H
  MOV.L       @(H'0010:16,SP),ER6
  JSR         @ER6
```

回避策:
ありません。

2.7 アセンブラファイル出力時のラベル出力位置の注意事項 (H8C-0023)

遅延分岐命令を含むプログラムをアセンブラファイル出力するとき、 ラベルを間違った位置に出力する場合があります。

発生条件:
以下の条件をすべて満たす場合に発生します。

1. cpu=h8sxn,h8sxm,h8sxaまたは、h8sxxオプションを選択している。
2. optimize=1オプションを選択している。
3. code=asmcodeオプションを選択している。
4. 遅延分岐命令の次命令(*)の後にラベルが存在する。
   *: RTE, RTE/LおよびSLEEP命令以外

例:

```
typedef unsigned char TYPE;
extern char val;
void sub03(void ){
   char ans=0;

   switch(val){
      case  0: ans=0; break;
      case  1: ans=1; break;
      case  3: ans=3; break;
   }
   if(ans==3) sub();       /* 常にans=1になる */
}
```

回避策:
以下のいずれかの方法で回避してください。
(a) optimize=0オプションを選択する。
(b) code=machinecodeオプションを選択する。

2.8 変数の奇数番地割り付きの注意事項 (H8C-0024)

構造体型変数で、境界調整数が2バイトのメンバを奇数番地に配置する 場合があります。

発生条件:
以下の条件をすべて満たす場合に発生します。

1. cpu=2000n,2000a,2600n,2600a,h8sxn,h8sxm,h8sxaまたは、h8sxx オプションを選択している。
2. 以下のサイズの構造体がある。
   Windows版 0x100000バイト以上、0x1FFFFEバイト以下
   UNIX版 0x10バイト以上、0x1Eバイト以下
3. 2.の構造体のメンバには、signedまたはunsigned char以外の型の メンバがある。
4. noalignオプションを選択または、リンク時に optimize=variable_access(*)オプションを選択している。
5. 2.の構造体型変数が、奇数サイズの変数の定義間で定義されている。

*: 一つ以上のファイルに対して、goptimizeオプションの指定 がある場合、短絶対アドレッシングモード活用最適化により、 本現象が発生することがあります。

例:

```
------------------------------------------------------
typedef unsigned char TYPE;
 struct ST {                /* 発生条件2 */
   char   c1;
   char   c2[12];
   int    d1;               /* 発生条件3 */
 } ;
 char dummy1;                    /* 発生条件5 */
 struct ST east1[2];      /* メンバd1が奇数番地に配置される */
 char dummy2;                    /* 発生条件5 */
 struct ST est1;
 char dummy3;
 struct ST est2;

 struct ST  sub6()
 {
    struct ST  *pst1;
    pst1 = &east1[1];
    *--pst1 = est1;
    return(*pst1);
```

```
}
-----------------------------------------------------
```

回避策:
コンパイル時にnoalignオプションおよび、リンク時に
optimize=variable_accessを選択しない。

3. **恒久対策**
本内容は、V.6.01 Release 01 で改修する予定です。
