RENESAS ツールニュース

RENESAS TOOL NEWS 2007年09月01日：070901/tn4

H8SX, H8S およびH8ファミリ用C/C++コンパイラパッケージ V.4 ～ V.6 ご使用上のお願い

H8SX, H8S およびH8ファミリ用C/C++コンパイラパッケージ V.4 ～ V.6 の 使用上の注意事項8件を連絡します。

1. 該当製品

H8SX, H8S およびH8ファミリ用 C/C++コンパイラパッケージ
V.4.0 ～ V.6.01 Release 03

製品型名
V.4
Windows版: PS008CAS4-MWR
Solaris版: PS008CAS4-SLR
HP-UX版: PS008CAS4-H7R
V.5
Windows版: PS008CAS5-MWR
V.6
Windows版: R0C40008XSW06R
Solaris版: R0C40008XSS06R
HP-UX版: R0C40008XSH06R

2. 内容

2.1 volatile修飾付でポインタ型にキャストされたアドレス参照に関する注意事項

(H8C-0069)

該当バージョン:
V.6.00 Release 00 ～ V.6.00 Release 03,
V.6.01 Release 00 ～ V.6.01 Release 03

現象:
ポインタ型にキャストしたアドレス参照がvolatile修飾されている場合、
その変数へのアクセスが削除される場合があります。

発生条件:
以下の条件をすべて満たす場合に発生する場合があります。
(1) CPUオプションに2000N,2000A,2600N,2600A,H8SXN,H8SXM,H8SXA,
H8SXXまたは、AE5(-cpu=2000n,2000a,2600n,2600a,h8sxn,h8sxm,
h8sxa,h8sxx,ae5)を選択している。
ただし、出力オブジェクト互換オプション(-legacy=v4)を選択
しているときは、2000N,2000A,2600N,2600Aは該当しません。
また、V.6.00では、2000N,2000A,2600N,2600Aは、該当しません。
(2) 最適化あり(-optimize=1)を選択している。(デフォルト時有効)
(3) 構造体以外の局所変数があり、その変数のアドレスを参照している。
(4) (3)のアドレス参照を*(volatile <type> *)にキャストした代入式が
複数行ある。<type>は(3)の変数と同じ型である。

発生例:

```
------------------------------------------------------------
void main(void){
    int X;
    *(volatile int *)(&X) = 0;  /* 発生条件(3),(4) */
    *(volatile int *)(&X) = 0;  /* 発生条件(3),(4) */
}
------------------------------------------------------------
 _main:
        RTS      ; 代入式のコードが出力されない。
;
------------------------------------------------------------
```

回避策:
以下のいずれかの方法で回避してください。
(1) 最適化なし(-optimize=0)を選択する。
(2) 当該関数に#pragma option nooptimize を指定する。
(3) 発生条件(3)の局所変数を構造体にし、アドレス参照を構造体の
アドレス参照とする。
例 :

```
------------------------------------------------------------
  struct {
     int X;
  } A;
  *(volatile int *)(&A) = 0;
------------------------------------------------------------
```

2.2 組み込み関数tasに関する注意事項(H8C-0070)

該当バージョン:
V.6.01 Release 00 ～ V.6.01 Release 03

現象:
組み込み関数tasが正しく実行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することがあります。
(1) CPUオプションに2000N,2000A,2600Nまたは、
2600A(-cpu=2000n,2000a,2600n,2600a)を選択している。
(2) 出力オブジェクト互換オプション(-legacy=v4)を選択していない。
(3) 組み込み関数tas()を使用している。

発生例:
```
------------------------------------------------------------
#include <machine.h>
void test(long a, long b, char *p)
{
   ...
   tas(p);     /* 発生条件(3) */
   ...
}
------------------------------------------------------------
 _test:
      ...
      TAS     @ER2   ; TAS命令が使用不可のレジスタを割り付けている。
      ...
      RTS
------------------------------------------------------------
```

回避策:
以下のいずれかの方法で回避してください。
(1) 出力オブジェクト互換オプション(-legacy=v4)を選択する。
(2) __asmキーワードを使用して TAS命令が使用可能なレジスタを指定する。
例:
```
 ------------------------------------------------------------
 void test(long a, long b, char *p)
 {
    __asm {
      MOV.L @(p,sp), ER4
      TAS   @ER4
    }
 }
```

```
------------------------------------------------------------
```
(3) #pragma inline_asmを使用して、その中でTAS命令を使用する。
例:
```
------------------------------------------------------------
 #pragma inline_asm tas2
 static void tas2(char *p)
 {
    TAS @ER0
 }
 void test(long a, long b, char *p)
 {
    tas2(p); //#pragma inline_asmで宣言したアセンブリ記述関数
 }
------------------------------------------------------------
```

2.3 ファイル間インライン展開オプション使用時の変数の初期値に関する注意事項(H8C-0071)

該当バージョン:
V.6.01 Release 00 ～ V.6.01 Release 03

現象:
ファイル間インライン展開オプションで指定したファイル内に変数の
定義があり、その変数を別変数の初期値として使用した場合に、
初期値が設定されないことがあります。

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オプションに2000N,2000A,2600N,2600A,H8SXN,H8SXM,H8SXA,
   H8SXXまたは、AE5(-cpu=2000n,2000a,2600n,2600a,h8sxn,h8sxm,
   h8sxa,h8sxx,ae5)を選択している。
   ただし出力オブジェクト互換オプション(-legacy=v4)を選択
   しているときは、2000N,2000A,2600N,2600Aは該当しません。
(2) ファイル間インライン展開オプション(-file_inline)を選択している。
(3) 大域変数を別の大域変数を使用して初期化している。
(4) (3)の初期値に使用されている大域変数をextern宣言している。
(5) (4)の大域変数は、ファイル間インライン展開オプションで指定した
   ファイルで定義され、(3)の初期値としてのみ使用している。

発生例:
```
------------------------------------------------------------
 <test1.c>
 extern int aa;        /* 発生条件(4)     */
```

```
  const int *a = &aa;   /* 発生条件(3),(5) */
  <test2.c>
  int aa;             /* 発生条件(5)     */

  <コマンドライン>
  ch38 -cpu=2600n -file_inline=test2.c  test1.c
 ------------------------------------------------------------
    .EXPORT     _a
    .SECTION    D,DATA,ALIGN=2
   _a:          ; static: a
    .DATA.L     _aa  ; シンボルaaのimport制御命令が出力されていない
    .END
 ------------------------------------------------------------
```

回避策:
以下のいずれかの方法で回避してください。
(1) データの初期値としての変数を初期値として使用している
ファイル内に定義する。

上記例の場合、

```
  <test1.c>
  int aa;
  const int *a = &aa;
```

(2)ファイル間インライン展開オプション(-file_inline)を使用しない。

2.4 __asm{ }内外で構造体メンバを参照する場合に関する注意事項(H8C-0072)

該当バージョン:
V.6.00 Release 00 ～ V.6.00 Release 03,
V.6.01 Release 00 ～ V.6.01 Release 03

現象:
構造体型局所変数又は構造体型引数を宣言し、これらの構造体メンバを__asm{ }内外で参照した場合、__asm{ }外で参照したメンバに対して正しくないアドレスでアクセスする場合があります。

発生条件:
以下のすべての条件を満たす場合に発生することがあります。
(1) CPUオプション
に、2000N,2000A,2600N、2600A、H8SXN、H8SXM、H8SXA、H8SXX
またはAE5(-cpu=2000n,2000a,2600n,2600a,h8sxn,h8sxm,h8sxa,h8sxx,ae5)を選択している。
(2) 最適化あり(-optimize=1)を選択している。(デフォルト時有効)

(3) 出力オブジェクト互換オプション(-legacy=v4)を選択していない。
(4) __asm{ }を含む関数の定義がある。
(5) (4)の関数内で構造体型局所変数を定義している又は、構造体型引数を
宣言している。
(6) (5)の構造体型局所変数は(4)の関数が使用しているスタックの先頭にない。
(7) (5)のメンバを(4)の__asm{ }内で参照している。
(8) (7)で参照したメンバを含む局所変数の任意のメンバを__asm{}の外側
でも参照している。

発生例：

```
------------------------------------------------------------
struct st{
   long a;

   long b;
};
long func(){
    long l;                        /* スタックの先頭  */
    struct st str2 = {1L,2L};

   __asm{                          /* 条件(4)        */
      mov.l  @(str2.a :32, sp ) , ER0/* 条件(7)        */
      mov.l  ER0, @(l:32,sp)
   }
   return l+str2.b;                 /* 条件(8)        */

}
------------------------------------------------------------
_func:
      PUSH.L     ER2
      SUB.W      #H'000C:16,R7
      MOV.L      SP,ER1
      ADDS.L     #4,ER1
      MOV.L      #L28,ER0
      SUB.L      ER2,ER2
      MOV.B      #8:8,R2L
      JSR        @$MVN$3:24
      MOV.L      @(4:32,SP),ER0
      MOV.L      ER0,@(0:32,SP)
      MOV.L      @SP,ER0
      MOV.L      @(4:16,SP),ER1;間違った領域を参照
                        ;正しくは、MOV.L @(8:16,SP),ER1
      ADD.L      ER1,ER0
```

```
        ADD.W       #H'000C:16,R7
        POP.L       ER2
        RTS
```

回避策

以下のいずれかの方法で回避してください。

(1) 最適化なし(-optimize=0)を選択する。

(2) 当該関数に#pragma option nooptimize を指定する。

(3) __asm{}内で参照するメンバの値を__asm{}の外側で変数へ代入し、その変数を__asm{}内で参照する。

例：

```
---------------------------------------------------------
}
struct st{
  long a;
  long b;
};
long func(){
  long l;
  struct st str2 = {1L,2L};
  long ll = 0;              /* 代入先の変数                  */
  ll = str2.a;              /* __asm{ }外で値を代入          */
  __asm{
    mov.l  @(ll, sp ) , ER0 /* __asm{ }外で代入した変数値を参照*/
    mov.l  ER0, @(l:32,sp)
  }
  str.a = ll;
  return l+str2.b;
}
---------------------------------------------------------
```

2.5 C++のクラスにおいて、要素数0x8000以上の配列を初期化する場合の注意事項(H8C-0073)

該当バージョン:

V.4.0 ～ V.4.0.09

V.5.0 ～ V.5.0.06

V.6.00 Release 00 ～ V.6.00 Release 03

V.6.01 Release 00 ～ V.6.01 Release 03

現象:
C++のコンストラクタを持つクラスにおいて、要素数0x8000以上の
配列を記述すると、コンストラクタで初期化されない配列の要素があります。

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オプションに300HA,2000A,2600A,H8SXA,H8SXXまたはAE5
(-cpu=300ha,2000a,2600a,h8sxa,h8sxx,ae5)を選択している。
(2) コンストラクタを持つクラスを定義している。
(3) 要素数0x8000以上のクラス型配列を定義している。
(4) コンパイル時、以下のいずれかのウォーニングメッセージが表示される。
(a) C5068 (W) Integer conversion resulted in a change of sign
(b) C5069 (W) Integer conversion resulted in truncation

発生例：
```
------------------------------------------------------------
class C {
 public:
  C();          /* 発生条件(2) */
};
C c[0x8000];     /* 発生条件(3) */
------------------------------------------------------------
```

回避策：
以下の方法で回避してください。
・templateクラスを用意し、要素数を0x7FFF以下ずつ初期化する。
例:
```
------------------------------------------------------------
  class C {
  public:
     C();
  };
  template<class T>
  class Array32K {
     T a1[0x4000];
     T a2[0x4000];
  public:
     T& operator [] (size_t i)
     {
        if (i < 0x4000)
           return a1[i];
        else
           return a2[i - 0x4000];
```

```
    }
  };
  Array32K<C> c;      //C c[0x8000]; と同等。
 ------------------------------------------------------------
```

2.6 定数値を使用している式がひとつの関数内に複数個ある場合の注意事項 (H8C-0074)

該当バージョン:
V.6.01 Release 03

現象:
定数値を使用している式が複数個ある場合、プログラムが正しく動作し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、発生することがあります。
(1) 最適化あり(-optimize=1)を選択している。(デフォルト時有効)
(2) 下記(a)または(b)いずれかに該当する式がひとつの関数内に複数個ある。
  (a) 複数個の式がそれぞれ4バイト定数を含み、4バイト定数の上位
    または下位2バイト以上が他の定数のそれと同じ値である。
  (b) 複数個の式がそれぞれ2バイト定数を含み、2バイト定数の上位
    または下位1バイト、もしくは2バイト全てが他の定数のそれと
    同じ値である。

発生例:

```
------------------------------------------------------------
//-cpu=2600a -legacy=v4 -sp
long g;
f1()
{
   g = (long)0x07FFFE01;          /* 発生条件(2) */
   if(g != (long)0x07FFFE01){     /* 発生条件(2) */
     printf("g=%08lX : ", (unsigned long)g); return(FALSE);
   }
   return(TRUE);
}
------------------------------------------------------------
MOV.L      #134217217,ER0
MOV.L      ER0,@_g:32
SUB.B      R0H,R0H       ; 誤ってSUB.B R0H,R0Hを出力
RTS
MOV.L      #134217217,ER0
```

```
MOV.L       ER0,@_g:32
SUB.B       R0H,R0H
BNE         L23:8
.....
-----------------------------------------------------------
```

回避策
以下のいずれかの方法で回避してください。
(1) 最適化なし(-optimize=0)を選択する。
(2) 当該関数に#pragma option nooptimize を指定する。

2.7 構造体、共用体、およびクラス型の戻り値を返す関数に関する注意事項 (H8C-0075)

該当バージョン:
V.6.01 Release 02 ～ V.6.01 Release 03

現象:
構造体、共用体、およびクラス型の戻り値を返す関数が正しく動作しないことがあります。

発生条件:
以下のすべての条件を満たす場合、発生することがあります。
(1) CPUオプションとして 2000N,2000A,2600Nまたは2600A
(-cpu=2000n,2000a,2600nまたは2600a)を選択している。
(2) 出力オブジェクト互換オプション(-legacy=v4)を選択している。
(3) 最適化なし(-optimize=0)を選択している。
(4) 関数の戻り値の型が構造体、共用体、またはクラスである。

発生例：

```
-----------------------------------------------------------
struct Test {
  char  cc[2];
} test;

struct Test func(void) {
  struct Test a;
  a.cc[0] = 1;
  return(a);          /* 発生条件(4) */
}
-----------------------------------------------------------
_func:
    PUSH.L  ER6
```

```
    MOV.L   SP,ER6
    PUSH.L  ER2
    SUBS    #2,SP
    MOV.B   #1:8,R0L
    MOV.B   R0L,@(-6:16,ER6)
    MOV.L   ER6,ER0
    ADD.W   #H'FFFA:16,R0
    MOV.L   @(4:16,ER6),ER1  ;間違った領域を参照
                        ;正しくは、MOV.L @(8:16,ER6),ER1
    MOV.L   #2:32,ER2
    JSR     @$MVN$3:24
    BRA     P_0000002A:8
P_0000002a:
    ADDS    #2,SP
    POP.L   ER2
    POP.L   ER6
    RTS
-------------------------------------------------------------
```

回避策
以下のいずれかの方法で回避してください。
(1) 最適化あり(-optimize=1)を選択する。（デフォルト時有効)
(2) 当該関数に#pragma option optimize を指定する
(3) 戻り値のサイズが4バイト以下の場合、
構造体パラメタ、リターン値のレジスタ割付オプション(-structreg)
を選択する。
(4) 戻り値をポインタで受け渡す。
例

```
-------------------------------------------------------
struct Test {
  char  cc[2];
} test;

struct Test* func(void) {
 struct Test a;
 a.cc[0] = 1;
 return(&a);
}
-------------------------------------------------------
```

2.8 サイズが3バイトの構造体および共用体型に関する注意事項(H8C-0076)

該当バージョン:

V.4.0 ～ V.4.0.09
V.5.0 ～ V.5.0.06
V.6.00 Release 00 ～ V.6.00 Release 03
V.6.01 Release 00 ～ V.6.01 Release 03

現象:
関数呼び出しでサイズが3バイトの構造体または共用体型変数を引数にした場合、
間違った実行結果になる場合があります。

発生条件:
以下のすべての条件を満たす場合に発生する場合があります。
(1) CPUオプションとして 300HN,300HA,2000N,2000A,2600Nまたは2600A
(-cpu=300hn,300ha,2000n,2000a,2600nまたは2600a)を選択している。
2000N,2000A,2600N,2600Aを選択しているときは出力オブジェクト
互換オプション(-legacy=v4)を選択している。(V.6.01の場合)
(2) 構造体パラメタ、リターン値のレジスタ割付オプション(-structreg)
を選択している。
(3) 境界調整数が1で、かつサイズが3バイトの構造体または共用体が宣言
されている。
(4) (3)の型を引数とする関数がある。
(5) (4)の引数は、(E)R0以外のレジスタに割り付いている。
(6) (4)の引数を(4)の関数内でアドレス参照している。

発生例：

```
------------------------------------------------------------
struct tmp1 {                    /* 発生条件(3)     */
   char ta;
   char ts;
   char tt;
};
char c;
void func(char a1, struct tmp1 pa) {  /* 発生条件(4),(5)  */
   a1++;
   pa.ts++;
   do {
      c=f2(&a1);
      c=f2(&pa.ta);              /* 発生条件(6)      */
   } while(c);
}
------------------------------------------------------------
 _func:
    PUSH.L  ER6
    SUBS    #4,SP
```

```
    MOV.B   R0L,@(3:16,SP)
    MOV.L   ER1,@SP      ; この4バイト転送命令により、
                         ; その直前に行なった転送命令の
                         ; 結果を壊している
    MOV.L   #_C:32,ER6
    MOV.B   @(3:16,SP),R0L
    INC.B   R0L
    ............
 ------------------------------------------------------------
```

回避策：
以下のいずれかの方法で回避してください。
(1) 構造体パラメタ、リターン値のレジスタ割付オプション(-structreg)
を選択しない。
(2) 構造体または共用体のサイズを4バイトにする。
(3) 構造体または共用体型の引数が(E)R0のレジスタに割り付くように
引数の順番を並び替える。
例

```
 ------------------------------------------------------
  void func(struct tmp1 pa, char a1){
    a1++;
    pa.ts++;
    do {
       c=f2(&a1);
       c=f2(&pa.ta);
    } while(c);
  }
 ------------------------------------------------------
```

3. 恒久対策

本内容は、9月5日にリリースするV.6.02 Release 00で改修します。