RENESAS ツールニュース

RENESAS TOOL NEWS 2005年1月16日：RSO-SHC-050116D

Renesas C/C++ Compiler Package for SuperH RISC engine family V.9.00 Release 01へのリビジョンアップのお知らせ 日本語版・英語版

SuperH RISC engineファミリ用C/C++コンパイラパッケージをV.9.00 Release 00からV.9.00 Release 01にリビジョンアップしました。

1. **該当製品**

   Renesas C/C++ Compiler Package for SuperH RISC engine family
   製品型名：

   Windows版 ： R0C40700XSW09R

   Solaris版 ： R0C40700XSS09R

   HP-UX版 ： R0C40700XSH09R

2. **リビジョンアップ内容**

   2.1 新機能および機能改善

   2.1.1 シミュレータ（Windows版のみ）

   (1) SH-2シミュレータでリトルエンディアン方式のデータ転送機能をサポートしました。

   (2) プログラムをシミュレータにダウンロードする際に、メモリリソースを自動で確保する機能を追加しました。

   (3) メモリアクセスエラー発生時に、メモリアクセスエラーが発生したアドレスを表示する機能を追加しました。

   2.1.2 コンパイラ

   (1) 以下のオプションを追加しました。

   (a) 未初期化変数のメモリ割り当て順の指定

bss_order={declaration|definition}

(b) 変数の配置指定
stuff[={bss|data|const}[,...]], およびnostuff

オプションの詳細については、「コンパイラ、アセンブラ、最適化リンケージエディタ コンパイラパッケージ V.9.00 ユーザーズマニュアルRev.1.01」を参照してください。

(2) SH-2のリトルエンディアン方式によるデータ転送機能をサポートしました。 ただし、リトルエンディアンをサポートしていないマイコンでは使用できません。

2.2 改修内容

2.2.1 High-performance Embedded Workshop（Windows版のみ）

ELF/DWARF2フォーマットで作成されたロードモジュールをデバッグするとHigh-performance Embedded Workshopが異常終了する場合がある不具合を改修しました。
詳細は2004年11月1日発行のRENESAS TOOL NEWS "High-performance Embedded Workshop ご使用上のお願い --ELF/DWARF2ロードモジュールのアンロードについて"を参照してください。

2.2.2 シミュレータ (Windows版のみ)

(1) SH7206マイコンのプロジェクトを作成し、シミュレーションを実行する際、オペランドキャッシュのメモリリソースが確保されず、 シミュレーションの停止やメモリアクセスエラーが発生する問題を改修しました。
詳細は2004年10月16日発行のRENESAS TOOL NEWS "SuperH RISC engineファミリ C/C++コンパイラパッケージ V.9.00 Release 00 ご使用上のお願い"を参照してください。

(2) SH2A-FPU Cycle Base SimulatorでRESBANK命令を実行するとサイクル数が不正になる問題を改修しました。
詳細は2004年11月1日発行のRENESAS TOOL NEWS "SuperH RISC engineファミリ C/C++コンパイラパッケージ V.9.00 Release 00 ご使用上のお願い"を参照してください。

2.2.3 コンパイラ

以降12点の問題を改修しました。

(1) 以下のDSPライブラリ関数をCPUコアSH4AL-DSP用のプログラムで使用した場合に正しく動作しない場合がある。
FftComplex, FftReal, IfftComplex, IfftReal, FftInComplex, FftInReal, IfftInComplex, IfftInReal, Fir, ConvComplete, ConvCyclic, ConvPartial, Correlate, CorrCyclic, MatrixMult, MsPower, およびVariance
本改修により、CPUコアSH4AL-DSP用のライブラリファイル名

が以下の通り変更になります。

指定オプション	ライブラリファイル
-pic=0 -endian=big	sh4aldspnb.lib
-pic=1 -endian=big	sh4aldsppb.lib
-pic=0 -endian=little	sh4aldspnl.lib
-pic=1 -endian=little	sh4aldsppl.lib

(2) 条件式を含むループにおいて、ループが不正に実行される場合がある(SHC-0008)。

例1

```
-----------------------------------------------
int b[100];
unsigned int c=0;
void func1() {
  unsigned int i;
  for(i=0;i<=100;i++) {     // 無限ループになる
     if(i != c) {
     b[i]=0;
     }
  }
}
-----------------------------------------------
```

例2

```
-----------------------------------------------
int b[100];
unsigned int c=0;
void func2() {
  unsigned int i;
  for(i=0;i<c;i++) {        // c=0の時もループを1回以上まわる
    if(i != 5) {
       b[i]=0;
    }
  }
}
-----------------------------------------------
```

発生条件：以下の条件をすべて満たした場合に発生することがあります。

1. optimize=1オプションを使用している。
2. ループが存在する。

3. 2のループ制御変数の増分値が1または-1である。
4. 2のループ内にif文が存在する。
5. 以下5a、または5bのいずれかの条件を満たす。

   5a. 4のif文の条件式が2.のループ制御変数と、2.のループ内で値が不変の式(例1の変数c)の比較である。

   5b. 2のループ制御変数の初期値または上限値がループ内で値が不変の式(例2の変数c)である。

6. 5のループ内不変式が整数型である。

(3) do-whileループで、正しくは1回で終了するものが2回以上ループする場合がある(SHC-0010)。

例：

```
----------------------------------------
int func() {
int count=0;
int limit=0x60000000;
do {
   count++;
   limit += 0x10000000;
} while(limit < -0x60000000);  // 正しい動作では1回目のループ後の
                               // 判定で偽となり、ループを抜ける。
   return (count);             // count=1が正しいが、違う値になる。
}
----------------------------------------
```

発生条件：以下の条件をすべて満たした場合に発生することがあります。

1. optimize=1オプションを使用している。
2. do-whileループが存在する。
3. 2のループ制御変数の型がint, signed int, long, または signed longである。
4. 2のループ判定式が、定数値との大小比較(<,<=,>,>=)である。
5. 2のループの比較演算子、制御変数の初期値、更新値、および 判定式の比較値が以下5a～5dのいずれかの条件を満たす。

   5a. 判定式が、ループ制御変数<定数 の場合に以下全てに該当する。

   ・更新値が正

   ・比較値<=初期値

・0x00000000<=(比較値-初期値-1)<=0x7FFFFFFF

5b. 判定式が、ループ制御変数<=定数 の場合に以下全てに該当する。

・更新値が正
・比較値<初期値
・0x00000000<=(比較値-初期値-1)<=0x7FFFFFFF

5c. 判定式が、ループ制御変数>定数 の場合に以下全てに該当する。

・更新値が負
・初期値<=比較値
・0x00000000<=(初期値-比較値-1)<=0x7FFFFFFF

5d. 判定式が、ループ制御変数>=定数 の場合に以下全てに該当する。

・更新値が負
・初期値<比較値
・0x00000000<=(初期値-比較値-1)<=0x7FFFFFFF

(4) 1ビットの符号付きビットフィールドと1との比較を行ったとき、またはある比較結果に対して演算を行いその結果を1と比較したときに、比較結果が間違っている場合がある(SHC-0011)。

例1：

```
-----------------------------------------
struct {
  char b0:1;
} ST;
void func() {
  if (ST.b0 != 1) {
  . . . . . .
  }
}
-----------------------------------------
```

例2：

```
-----------------------------------------
int a;
void func2() {
  int t;
  t = ((a & 0x40) == 0);
  t = t - 1;
```

```
  t = -t;
  if(~t==1) {
     a = 1;
  } else {
     a = 2;
  }
}
```

発生条件：以下1、2のいずれかの条件を満たした場合に発生することがあります。

1. 以下のすべての条件を満たした場合。
   1a. optimize=1を指定している。
   1b. 1ビットの符号付きビットフィールドを使用している。
   1c. 1b.と1の比較(==,!=)を行っている。
2. 以下のすべての条件を満たした場合。
   2a. optimize=1を指定している。
   2b. 比較結果に対して以下のいずれかの演算を行っている。
   比較結果と1の減算、比較結果と1の排他的論理和、
   比較結果の符号反転、または比較結果のビット反転
   2c. 2b.の演算結果と1の比較(==,!=)を行っている。

(5) 0との加算および減算、または1との乗算をした結果を他の演算で使用したときに、変数の値を誤って更新する場合がある(SHC-0012)。

例：

```
int a[4], b;
void func() {
      a[3&(b-0)]=0;
}
```

発生条件：以下の条件をすべて満たした場合に発生することがあります。

1. 変数と0の加算および減算、または変数と1の乗算をしている。
2. 1の結果を以下の演算で使用している。
   加算、減算、論理演算(&,|,^)、除算、剰余算、またはシフ

ト

(6) pack=1を指定した構造体または共用体のdouble型メンバを参照したときに、レジスタR2の値を誤って変更する場合がある(SHC-0013)。

例：

```
----------------------------------------
#pragma pack 1
struct {
        double d;
} ST;
int t;
double d[2];
void func() {
        d[t]=ST.d;
}
----------------------------------------
```

発生条件：以下の条件をすべて満たした場合に発生することがあります。

1. pack=1を指定した構造体または共用体がある。
2. 1.の構造体または共用体にdouble型のメンバが存在する。
3. cpu=sh4,sh4a,sh2afpuを指定している。
4. sizeまたはunaligned=runtimeオプションを使用している。
5. 実行時ルーチン_pack1_ld64の呼び出しがある。

(7) 定数乗算および定数除算を共に含む式において、演算結果が間違っている場合がある(SHC-0015)。

例：

```
----------------------------------------
unsigned int a=65536;
unsigned int b;
void func() {
    b=(a*65536)/8;   // b=0 ((65536*65536)/8→0/8=0) が正しい
}                    // 結果だが、b=65535<<13と置き換えられる。
----------------------------------------
```

発生条件：以下の条件をすべて満たした場合に発生することがあります。

1. unsigned型の式と定数の乗算式が存在する。
2. 1を含む式を1の定数の正の約数で除算している。

3. 1の乗算結果がその式の型の最大値を超える。

(8) 被シフト数のビットサイズ未満の定数シフトを複数回行い、 そのシフト数の合計が被シフト数のビットサイズ以上になった場合、演算結果が間違っている場合がある(SHC-0016)。

例：

```
----------------------------------------
int x,y;
void func() {
  x=y<<31<<1;  // シフト数の合計は32で、int型のビットサイズ
               // 以上になる
}
----------------------------------------
```

発生条件：以下の条件をすべて満たした場合に発生することがあります。

1. cpuオプションのパラメータにsh1, sh2, sh2e, またはsh2dsp以外を指定している。
2. 以下2a、2bのいずれかの条件を満たす。
   2a. 左シフトもしくは2のべき乗の乗算が2回以上連続し、 かつ個々のシフト数、乗数は被シフト数のビットサイズ未満である。
   2b. 右シフトもしくは2のべき乗の除算が2回以上連続し、 かつ個々のシフト数、乗数は被シフト数のビットサイズ未満である。
3. 2のシフト数と乗数の値の合計が被シフト数のビットサイズ以上となる。

(9) 関数の最後が関数呼び出しの場合、呼び出し先関数のアドレスをレジスタR0に誤って書き込む場合がある。

例：

```
----------------------------------------
char a[3];
void func2() {}
#pragma section A
void func() {
 ++a[2];
 func2();
}
----------------------------------------
```

発生条件：以下の条件をすべて満たした場合に発生することがあ

ります。

1. 呼び出し元関数の最後で関数を呼び出している。
2. 呼び出し先関数の定義が1と同一ファイル内にある。
3. 2の関数定義と1の関数呼び出しは別セクションにある、もしくは4096バイト以上離れた位置にある。
4. 呼び出し元関数内に1以外の関数呼び出しがない。

(10) ポインタを介してビットフィールドメンバに対して演算を行った場合、その演算結果が間違っている場合がある(SHC-0023)。

例：

```
----------------------------------------
typedef union {
  unsigned int a;
  unsigned int b:32;
} UN;
UN un;
void func() {
  int *p=(int *)&un.a;
  un.b=1;
  *p+=1;
}
----------------------------------------
```

発生条件：以下の条件をすべて満たした場合に発生することがあります。

1. 構造体または共用体型を使用している。
2. 1.のメンバを直接参照(例ではun.b)している。
3. 2.と同じ領域のメンバ(例ではun.a)を指すポインタ変数(例ではp)が存在する。
4. 3.のポインタはローカル変数である。
5. 3.のポインタの間接参照(例では*p)が存在する。
6. 当該領域の値を更新する。
7. 構造体または共用体自身の参照を関数内で行っていない。

(11) 構造体または共用体コピーを行った場合、スタック領域が必要以上に確保される場合がある(SHC-0021)。

例：

```
----------------------------------------
typedef struct {
   unsigned char c;
} ST0;
typedef struct {
   ST0 s;
```

```
} ST;
extern ST A[1000];
extern unsigned short i;
void func(ST *d) {
   A[i-1].s=d->s;  // スタックがA[1000]分余分に確保される
}
```

発生条件：以下の条件をすべて満たした場合に発生することがあります。

1. 構造体または共用体型のメンバを持つ構造体または共用体配列が存在する。
2. 1の配列の要素数は2以上である。
3. 構造体または共用体代入文が存在する。
4. 3の左辺は1の構造体または共用体配列の構造体または共用体型メンバである。

(12) C4098、およびC4099インターナルエラーが発生する場合がある(SHC-0007, SHC-0017, SHC-0022, SHC-0025)。

2.2.4 アセンブラ

CPUシリーズSH-2A、またはSH2A-FPU用のプログラムを作成する場合に発生する以降3点の問題を改修しました。

(1) 拡張命令や分岐命令(PC相対)のディスプレースメント値が正しくない場合がある。

例：

```
    .ALIGN  4
    NOP
    MOV.L   @(AA,R1),R0
    BRA     LABEL
    .ALIGN  4
    BT      LABEL
. . . . . .
LABEL: NOP
AA:   .EQU    4
```

発生条件：以下の条件をすべて満たした場合に発生することがあります。

1. cpu=sh2aまたはcpu=sh2afpuオプションを指定している。
2. ある遅延分岐命令の前に"MOV @(disp,Rn)"(disp:12が

指定可であるが指定されていない命令)が奇数個存在する。
3. 2の遅延分岐命令の次が.ALIGN制御命令または.ORG制御命令である。
4. 3により2バイトのアドレス補正が行われる。
5. 3の次が拡張命令、または分岐命令(PC相対)である。

(2) アドレス分岐の分岐先が間違っている場合がある。

例：

```
----------------------------------------
    .SECTION  SEC1,CODE
    MOV.L     @(AA,R1),R0  ; 2バイト命令
    NOP
    BRA       L3
    .ALIGN    4            ; ロケーションカウンタ 補正あり
    BF        L1           ; ディレイスロット命令でない
L3:   NOP
    .SECTION  SEC2,CODE
L1:   NOP
AA:   .EQU      4
----------------------------------------
```

発生条件：以下の条件をすべて満たした場合に発生することがあります。

1. cpu=sh2aまたはcpu=sh2afpuオプションを指定している。
2. ある遅延分岐命令の前に”MOV @(disp,Rn)”(disp:12が指定可であるが指定されていない命令)が奇数個存在する。
3. 2の遅延分岐命令の次が.ALIGN制御命令または.ORG制御命令である。
4. 3によりMOV @(disp,Rn)命令数×2バイトの補正がある。
5. 3の次がアドレス分岐命令である。

(3) 偶奇演算子$even2, および$odd2を使用した演算の結果が正しくない場合がある。

2.2.5 最適化リンケージエディタ

以降5件の問題点を改修しました。

(1) 共通コード統合最適化(optimize=same_code)が有効な場合、生成されたデバッグ情報を使ってデバッガでステップ実行を行うと間違った関数へ分岐する場合がある。

(2) .stack制御命令を、定義シンボルには記述せずに参照シンボルのみに指定した場合に、その情報がsniファイルに出力されない。

(3) 共通コード統合最適化が有効な場合、リロケータブルファイルを入力すると間違ったコードが生成される場合がある。
発生条件：以下の条件をすべて満たす場合、発生する可能性があります。
1. コンパイル時にgoptimizeオプションを指定している。
2. 共通コード統合最適化(optimize=same_code)が有効である。
3. 最適化リンケージエディタ V.8.00.03以降のバージョンを使用している。
4. リンク時に、リロケータブルファイルを入力している。
5. 4のリロケータブルファイルのコードが、2の最適化により4n+2バイト(4の倍数でないサイズ)で共通化される。

(4) 定数および文字列統合最適化が有効な場合、不適切なエラー(L2330)が出力される場合がある。
発生条件：以下の条件をすべて満たす場合、発生する可能性があります。
1. コンパイル時にgoptimizeオプションを指定している。
2. 定数および文字列統合最適化(optimize=string_unify)が有効である。
3. コンパイル時にabs16オプションを指定している、または#pragma abs16宣言されているconst指定変数がある。

(5) 以下に該当する場合に内部エラーが発生する。
1. 未参照シンボル削除最適化(optimize=symbol_delete)が有効な場合に出力ファイルをoutputオプションを使用して複数ファイルに分割するように指定している(内部エラー(7041)発生)。
2. profileオプション使用時に、binaryオプションを指定している(内部エラー(L4001)発生)。

3. **リビジョンアップと購入方法**

3.1 リビジョンアップ（無償）

該当製品をお持ちの場合、無償でリビジョンアップできます。

(1) Windows版: R0C40700XSW09Rの場合
オンラインでリビジョンアップできます。**こちら**からダウンロードしてください。

(2) R0C40700XSS09R(Solaris版) および R0C40700XSH09R(HP-UX版)の場合

以下の情報を最寄りのルネサス販売または特約店までご連絡ください。

製品型名：

Solaris版 ： R0C40700XSS09R

HP-UX版 ： R0C40700XSH09R

バージョン番号 ： V.9.00

リリース番号 ： Release 01

3.2 新規購入

ご注文の際には、以下の情報を最寄りのルネサス販売または特約店までご連絡ください。

製品型名：

Windows版 ： R0C40700XSW09R

Solaris版 ： R0C40700XSS09R

HP-UX版 R0C40700XSH09R

バージョン番号 ： V.9.00

リリース番号 ： Release 01