お客様各位

カタログ等資料中の旧社名の扱いについて

2010年4月1日を以ってNECエレクトロニクス株式会社及び株式会社ルネサステクノロジが合併し、両社の全ての事業が当社に承継されております。従いまして、本資料中には旧社名での表記が残っておりますが、当社の資料として有効ですので、ご理解の程宜しくお願い申し上げます。

ルネサスエレクトロニクス ホームページ（http://www.renesas.com）

2010年4月1日
ルネサスエレクトロニクス株式会社

【発行】ルネサスエレクトロニクス株式会社（http://www.renesas.com）

【問い合わせ先】http://japan.renesas.com/inquiry

（注）本文中のURLは、以下の様に変更になりました。
http://tool-support.renesas.com/jpn/toolnews/shc/shcv7/dr_shcv7_2.html

2002年　9月　9日

― 日立半導体技術情報 ―

〒１００－０００４
東京都千代田区大手町2丁目6番2号
(日本ビル)
TEL　(03)5201-5022（ダイヤルイン）
株式会社 日立製作所 半導体グループ

製品分類	開発環境		発行番号	TN-CSX-040A	Rev.	第1版
題名	SuperH RISC engine C/C++コンパイラ Ver.7.0 不具合のご連絡(4)		情報分類	１．仕様変更 ２．ドキュメント訂正追加等 ③　使用上の注意事項 ４．マスク変更 ５．ライン変更		
適用製品	P0700CAS7-MWR P0700CAS7-SLR P0700CAS7-H7R	対象ロット等 全ロット	関連資料	SuperH RISC engine C/C++コンパイラ、アセンブラ、最適化リンケージエディタユーザーズマニュアル ADJ-702-304A　第1版	有効期限 永年	

SuperH RISC engine C/C++コンパイラ Ver.7.0 台に別紙に示す不具合があります。
次に示す製品を御使用のお客様につきましては周知願います。

型名	パッケージバージョン	コンパイラバージョン
P0700CAS7-MWR	7.0B	7.0B
	7.0.01	7.0.03
	7.0.02	7.0.04
	7.0.03	7.0.06
P0700CAS7-SLR	7.0B	7.0B
	7.0.02	7.0.03
	7.0.03	7.0.04
	7.0.04	7.0.06
P0700CAS7-H7R	7.0B	7.0B
	7.0.02	7.0.03
	7.0.03	7.0.04
	7.0.04	7.0.06

なお、プログラムが本不具合に該当しているかを検出するチェックツールを以下よりダウンロードできます。

http://www.hitachisemiconductor.com/sic/jsp/japan/jpn/PRODUCTS/MPUMCU/TOOL/download/crosstool/release/rest_shcv7002.html

添付 ： P0700CAS7-020715J
SuperH RISC engine C/C++コンパイラ Ver.7 不具合内容(4)

SuperH RISC engine C/C++コンパイラ Ver.7
不具合内容(4)

SuperH RISC engine C/C++コンパイラ Ver.7.0 台における不具合内容を以下に示します。

1. #pragma global_register 指定レジスタの代入削除

【現象】

#pragma global_register 指定したレジスタの代入を不正に削除する場合がある。

【発生条件】

以下の条件をすべて満たす場合、発生することがあります。

(1) #pragma global_register を指定している。
(2) optimize=1 を指定している。
(3) (1)で指定した変数が複合代入式などで 1 つの式で定義・使用される。
＜例＞

```
#pragma global_register (a=R14)
            :
a += b;      // もしくは a=a+b;
```

【回避方法】

以下のいずれかの方法で回避していただきますようお願いします。

(1) #pragma global_register 指定をやめる。
(2) optimize=0 を指定する。

2. 32bit ビットフィールド比較不正

【現象】

32bit ビットフィールドを 0 と比較した場合、比較前に不正な 0 との AND が生成され、判定結果が常に真(または偽)となる場合がある。

【例】

```
struct ST {
   unsigned int b: 32;
};

void f(struct ST *x) {
   if (x->b) {
      :
   }
}
```

```
        :
MOV.L     @R4,R0
AND       #0,R0    ; <- 0 と AND をとる
TST       R0,R0    ; 必ず真となる
BF        L12      ; L12 への分岐は起きない
        :
```

【発生条件】

以下の条件をすべて満たす場合、発生することがあります。

(1) 構造体に 32bit ビットフィールドメンバが存在する。

(2) 当該メンバと 0 の比較(==もしくは!=)を行っている。

【回避方法】

以下の方法で回避していただきますようお願いします。

(1) 32bit ビットフィールドを整数型にする。

＜例＞

```
struct ST {
      unsigned int b;
}
```

3. trapa_svc 使用時のスタック移動不正

【現象】

pic=1 を指定してコンパイルした場合、組み込み関数 trapa_svc を使用している関数のアドレスロード時に、不当にスタック移動命令を出力する場合がある。

【例】

```
#include <machine.h>
extern char *b(void (*yyy)(char));

void y(char c) {
    trapa_svc(160, 10, c);
}

char *a(void) {
   return b(y);
}
```

```
_a:
        MOV.L       L14,R4      ; _y-L12
        MOVA        L12,R0
        ADD         R0,R4
L12:
        MOV.L       @R15+,R0    ; <-不要なスタック移動命令
        MOV.L       L14+4,R2    ; _b-L13
        MOVA        L13,R0
        ADD         R0,R2
L13:
        JMP         @R2
        MOV.L       @R15+,R0    ; <-不要なスタック移動命令
```

【発生条件】

以下の条件をすべて満たす場合、発生することがあります。

(1) cpu=sh1 以外を指定している。

(2) pic=1 を指定している。

(3) 組み込み関数 trapa_svc を使用している。

(4) trapa_svc を使用している関数のアドレスロード位置が、trapa_svc の呼び出し位置より後に出現する。

【回避方法】

以下の方法で回避していただきますようお願いします。

(1) 関数内で trapa_svc を使用している関数のアドレスロード位置を、trapa_svc の呼び出し位置より前になるように定義順を変更する。

＜例＞

```
#include <machine.h>
extern char *b(void (*yyy)(char));
void y(char c);    // 原型宣言

char *a(void) {
    return b(y);   // 関数 y のアドレスロードを trapa_svc 呼び出し前にする
}
void y(char c) {
    trapa_svc(160, 10, c);
}
```

4. R0-R7 へのコピー命令の不当移動

【現象】

最適化により、R0-R7 へのコピー命令もしくは拡張命令が関数呼び出しを超えて不正に移動し、CMP/EQ(TST)の結果が不正となる場合がある。

【例】

```
        :
MOV.L       @R4,R2
MOV         R5,R14
MOV         #1,R5       ; H'00000001
ADD         #24,R2
MOV.L       @R2,R6
EXTU.W      R14,R1      ; R1 の定義が JSR を越えて移動
MOV.L       @(8,R2),R7
JSR         @R7         ; 呼び出し先で R1 を破壊する場合あり
ADD         R6,R4
MOV         #4,R2       ; H'00000004
CMP/EQ      R2,R1       ; 破壊された R1 の値を使って比較
EXTU.W      R0,R0
BF          L16
        :
```

【発生条件】

以下の条件をすべて満たす場合、発生することがあります。

(1) optimize=1 を指定している。

(2) 関数内に条件分岐と、関数呼び出しがある。

(3) 以下最適化を実施。

＜最適化前＞

```
MOV   R0, Rn      ; または EXTU   R0,Rn
        :
MOV   Rx, R0      ; または EXTU   Rx,R0
TST   #imm, R0    ; または CMP/EQ   #imm,R0
MOV   Rn, R0      ; または  EXTU   Rn,R0
```

＜最適化後＞

```
MOV   Rx, Rm      ; または EXTU   Rx,Rm
MOV   #imm, Ry
TST   Ry, Rm      ; または CMP/EQ   Ry,Rm
```

(4) (3)の最適化で、Rm レジスタが R0-R7 で当該関数で他に使用されていない。

(5) (3)の最適化で生成した MOV Rx,Rm(または EXTU)命令が他の最適化により、関数呼び出しを超えて移動される。

【回避方法】

該当個所が存在するかを確認するチェックツールを配布いたします。該当個所が存在した場合、以下のいずれかの方法で回避していただきますようお願いします。

(1) optimize=0 を指定する。

チェッカは以下 URL より入手できます。

http://www.hitachisemiconductor.com/sic/jsp/japan/jpn/PRODUCTS/MPUMCU/TOOL/download/crosstool/release/rest_shcv7002.html

なお、本チェックツールは発生条件(3)の最適化が実施されたかどうかの検出を行います。したがって不具合に該当しない関数が検出される場合もあります。

以上