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

2003年　1月　21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-047A	Rev.	第１版
題名	H8S.H8/300 シリーズ C/C++コンパイラパッケージ リビジョンアップのお知らせ		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	PS008CAS5-MWR, PS008CAS4-MWR, PS008CAS4-SLR, PS008CAS4-H7R	対象ロット等 全ロット	関連資料	H8S,H8/300 シリーズ 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-702-303 第１版	有効期限 永年	

H8S,H8/300 シリーズ C/C++コンパイラパッケージをVer 5.0.03(WindowsR版)、Ver.4.0.07(UNIX 版)にリビジョンアップしました。

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。

WindowsR版　：H8S,H8/300 シリーズC/C++コンパイラパッケージ 型名：PS008CAS5-MWR）のVer5.0, Ver5.0.01, Ver.5.0.02
H8S,H8/300 シリーズC/C++コンパイラパッケージ 型名：PS008CAS4-MWR）の
Ver.4.0, Ver.4.0r1, Ver.4.0A, Ver.4.0Ar1, Ver.4.0Ar2

SPARCR版　：H8S, H8/300 Series C/C++コンパイラパッケージ 型名：PS008CAS4- SLR）の
Ver.4.0, Ver.4.0A, Ver.4.0Ar1, Ver.4.0B, Ver.4.0.05, Ver.4.0.06

HP9000版　：H8S, H8/300 Series C/C++コンパイラパッケージ 型名：PS008CAS4-H7R）の
Ver.4.0, Ver.4.0A, Ver.4.0Ar1, Ver.4.0B, Ver.4.0Br1, Ver.4.0.05, Ver.4.0.06

記

お問い合わせ先：(株)日立製作所 半導体グループ
マイコンテクニカルサポートセンタ
Email:　micontech@sic.hitachi.co.jp

添付：

(1) H8S,H8/300 シリーズ C/C++コンパイラ パッケージ WindowsR版 V5.0.03、UNIX 版 Ver.4.0.07 アップデート内容
(PS008CAS5-030115J)　3枚

H8S,H8/300 シリーズ C/C++コンパイラ パッケージ Windows®版 Ver.5.0.03、UNIX 版 Ver.4.0.07 アップデート内容

1. Hitachi Embedded Workshop 2 (Windows®版 Ver.5.0.03 パッケージのみの変更点)

1.1 メニュー、ダイアログの日本語化対応

Hew のメニューやダイアログが日本語表示になりました。

1.2 ドラッグ&ドロップのサポート

Watch ウィンドウへの変数の登録をエディタウィンドウからマウス操作による
ドラッグ&ドロップで可能になりました。

1.3 Hew サーバの公開

COM 技術による Hew サーバを公開しました。 本サーバは Out-of-process サーバ
として機能します。詳細は添付資料を参照してください。

1.4 キャッツ(株)製 ZIPC ツールとの接続

キャッツ(株)製 ZIPC ツールと連動してプログラムのデバッグが可能に
なりました。

1.5 プロジェクトジェネレータ生成データの追加、修正

新たに以下の CPU のプロジェクト生成を追加しました。

H8S/2168F, H8S/2367F, H8S/2375R, H8S/2377R, H8S/2628F

また、以下の CPU の I/O 定義ファイル(iodefine.h)を修正しました。

H8S/2148, H8S/2612, H8S/2678

2. コンパイラ（Ver. 4.0.03 → Ver. 4.0.04）

2.1 単項マイナス演算子に関する不正コードの修正

【内容】

char/short/int 型の変数が，型変換（暗黙の型変換も含む）により符号拡張された後に単項マイナス演算を行った場合、
演算結果が不正となる不具合を修正しました。

【例１】

[ソースプログラム]

```
char  c;
short s;
int   i;
long  l;
vois sub()
{
    s=-c;         /* c=-128   の時，s=-128    （正しくは s=128）   */
    i=-c;         /* c=-128   の時，i=-128    （正しくは i=128）   */
    l=-(long)c;   /* c=-128   の時，l=-128    （正しくは l=128）   */
    l=-(long)s;   /* s=-32768 の時，l=-32768  （正しくは l=32768） */
    l=-(long)i;   /* i=-32768 の時，l=-32768  （正しくは l=32768） */
}
```

[正しくない出力コード] 不具合例１の s=-c; の場合

```
MOV.B       @_c,R0L
NEG.B       R0L
EXTS.W      R0
MOV.W       R0,@_s
```

[正しい出力コード] 不具合例１の s=-c; の場合

```
MOV.B       @_c,R0L
EXTS.W      R0
NEG.W       R0
MOV.W       R0,@_s
```

【例２】

```
char c;
int  i1,i2;
```

```
        i1 = -c + i2; /* c=-128    の時，-c が-128    （正しくは-c は 128）    */
```
【例３】
```
        char c;
        int  func();
        int func()
        {
            return( -c ); /* c=-128    の時，-c が-128    （正しくは-c は 128）    */
        }
```

2.2 不正なＡＮＤ命令生成の修正

【内容】

シフト演算後にビットごとＡＮＤ演算を行った場合，または連続して複合代入(&=,|=,^=)を行った場合，
コンパイル実行環境(【注】参照)によって不正なＡＮＤ命令が生成されることがある不具合を修正しました。

【例１】

[ソースプログラム]
```
  unsigned int X, Y;
  void sub( void )
  {
        X = (Y >> 14) & 0x2 ;
  }
```
[正しくない出力コード]
```
        MOV.W       @_Y,     R0
        ROTL.W      #H'2,    R0
        AND.L       #H'20002,ER0    <--- 正しくないコード
        MOV.W       R0,      @_X
```
[正しい出力コード]
```
        MOV.W       @_Y,     R0
        ROTL.W      #H'2,    R0
        AND.W       #H'2,    R0     <--- 正しいコード
        MOV.W       R0,      @_X
```
【例２】
```
     int sub(int Y)
     {
         Y &= 0x3;
         Y &= 0x2;
         return Y;
     }
```
[正しくない出力コード]
```
        AND.L       #H'20002,ER0    <--- 正しくないコード
```
[正しい出力コード]
```
        AND.W       #H'2,    R0     <--- 正しいコード
```
【注】コンパイラ解放済みメモリ領域の値に依存するため、同一プログラム同一オプションでもコンパイル環境（ホストコンピュータのオペレーティングシステム）によって再現しないことがあります。

2.3 ループ内の不変式の括り出し最適化が効くように修正

【内容】

コンパイラの最適化の１つである“ループ内の不変式の括り出し最適化”がコンパイル実行環境(【注】参照)によって効かない場合がある現象を修正しました。

【例１】

[ソースプログラム]
```
    for (  …  ;  …  ;  …  ){
            for (   …  ;   …   ;   …  )
                X =  0;
    }
```
のようなループ式を書いた場合，最適化前の生成コードとして，下記のように出力します。

[最適化していない出力コード]
```
       <省略>
        BRA    L1              <----- (A)
  L2:
```

```
        <省略>
         BRA    L3
    L4:
        <省略>
         SUB.W      E0,E0     <----- (※)
         MOV.W      E0,@_X
    L3:
        <省略>
         Bcc    L4
    L1:
        <省略>
         Bcc    L2            <----- (B)
```

このとき,（※）で使用しているレジスタ E0 が（A）～（B）の範囲で
変更されていないのであれば，下記のように最適化（不変式の移動）を実施します。

[最適化した出力コード]

```
        <省略>
         SUB.W      E0,E0   <-------------+
         BRA    L1          <----- (A)    |
    L2:                                   |
        <省略>                            | (移動)
         BRA    L3                        |
    L4:                                   |
        <省略>                            |
                             -------------+
         MOV.W      E0,@_X
    L3:
        <省略>
         Bcc    L4
    L1:
        <省略>
         Bcc    L2            <----- (B)
```

この最適化によって，ループ速度が向上します。
しかし，今回，この最適化が最適化条件を満たしているにもかかわらず効かない場合があります。

【注】コンパイラ解放済みメモリ領域の値に依存するため、同一プログラム同一オプションでもコンパイル環境
（ホストコンピュータのオペレーティングシステム）によって再現しないことがあります。

2.4 不正な分岐の修正

【内容】

比較演算で，被比較式の演算結果がオーバーフローする場合、正常に比較分岐できない不具合を修正しました。（ア）
なお、型変換の記述により、コンパイラの最適化によって、演算の最適化（int 演算をすべきところ、char 演算を実施）
が実施され、その結果オーバーフローになる不具合も含めて修正しました。（イ）

【例１】（ア）の場合

```
int i1=1;
int i2=-32767;

if ( (i1 - i2) < 0)
    printf("OK\n");
else
    printf("NG\n");
```

※上記被比較式 (i1 - i2) の結果はコンパイラの実装による定義では -32768 であり、“OK”にもかかわらず,
“NG”となってしまいます。

[正しくない出力コード]

```
MOV.W       @_i1,R0
MOV.W       @_i2,R1
SUB.W       R0,R1
BGE         Ln
```

[正しい出力コード]

```
MOV.W       @_i1,R0
MOV.W       @_i2,R1
```

```
SUB.W       R0,R1
MOV.W       R0,R0           <--- 上の正しくないコードでは本命令が誤って削除されている
BGE         Ln
```

BGE 命令は CCR の V（オーバフロー）フラグの影響を受けます。この BGE 命令は上の MOV.W R0,R0 が CCR の V フラグを常にクリアすることを仮定して使われています。しかし、上の SUB R0,R1 は V フラグを変更することがあります。MOV.W R0,R0 を削除すると BGE 命令を実行する時に V フラグが常にクリアされているとは限りません。

【例２】（イ）の場合

```
char c1=1;
char c2=-127;

if ( (char)(c1 - c2) < 0)
    printf("OK\n");
else
    printf("NG\n");
```

※上記被比較式　(char)(c1 - c2)　の結果は -128　であり，“OK”にもかかわらず，“NG”となってしまいます。
この場合，(c1-c2)は ANSI 仕様によると int 型演算を実施するので，オーバーフローはしませんが、
コンパイラの最適化で小さいサイズ(char 型)で演算されることにより、オーバーフローが発生します。

3. 最適化リンケージエディタ (Ver.7.1.06 -> Ver.7.1.07)

3.1 内部エラーの解決

以下の内部エラーが生じる不具合を解決しました。

・アセンブラ出力 Object について分岐命令最適化指定時の内部エラー (8705)

3.2 form={binary | stype | hexadecimal}指定時の不正動作

output オプションで存在しないディレクトリを指定した際、binary/stype/hexadecimal 形式で出力指定をした場合に、出力ファイルが作成されないにも関わらずエラーメッセージが出力されない問題を解決しました。

3.3 未参照シンボル削除最適化指定時のオブジェクト不正

下記条件を満たす場合、最適化によってアクセスする配列の要素が不正となる場合がある不具合を解決しました。

(1) アクセスされる配列 A_arr[]が存在する。
(2) 最適化にて削除対象となる配列 B_arr[](もしくは変数)が存在する。
(3) A_arr[],B_arr[]はそれぞれ別セクションに存在している。
　　ここではそれぞれのセクションを A,B とする。
(4) start オプション指定により、A セクションと B セクションのアドレスが重複するようにセクションを配置する。
(5) 未参照シンボル削除最適化を有効にする。

3.4 C++オブジェクトをリンクする際の不正なエラー出力

C++で、テンプレートを含むオブジェクトをリンクした際に、不正に P3300(F)エラーが生じる場合がある不具合を解決しました。

3.5 SYSROF 形式へ変換した際のデータ欠如

下記条件を満たす場合、データが不正に欠落してしまう場合がある不具合を解決しました。

(1) C++によりソースが記述されている。
(2) リンク時に、未参照シンボル削除最適化を有効にしている。
(3) コンバータによりオブジェクトフォーマットを ELF->sysrof に変換している。

以上