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

発行日：2004 年 10 月 27 日

RENESAS TECHNICAL UPDATE

〒100-0004　東京都千代田区大手町 2-6-2 日本ビル
株式会社 ルネサス テクノロジ
問合せ窓口　E-mail: csc@renesas.com

製品分類	開発環境	発行番号	TN-CSX-A086A/J	Rev.	第 1 版
題名	SuperH RISC engine C/C++コンパイラパッケージ V.8.00 Release04 リビジョンアップのお知らせ		情報分類	技術情報	
適用製品	R0C40700XSW08R (P0700CAS8-MWR) R0C40700XSS08R (P0700CAS8-SLR) R0C40700XSH08R(P0700CAS8-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 RJJ10B0052-0100H Rev.1.00	

SuperH RISC engine C/C++コンパイラパッケージ V.8.00 Release04 にリビジョンアップしました。
次の表にあるパッケージバージョンをお持ちのお客様は、以下を参照して下さい。

型名	パッケージバージョン	コンパイラバージョン
P0700CAS8-MWR	8.0.00 8.0.01	8.0.00 8.0.01
R0C40700XSW08R	8.00 Release01 8.00 Release02 8.00 Release03	8.0.01 8.00.02 8.00.03
P0700CAS8-SLR	8.0.00 8.0.01	8.0.00 8.0.01
R0C40700XSS08R	8.00 Release01 8.00 Release03	8.0.01 8.00.03
P0700CAS8-H7R	8.0.00 8.0.01	8.0.00 8.0.01
R0C40700XSH08R	8.00 Release01 8.00 Release03	8.0.01 8.00.03

Windows®版をお持ちのお客様は、アップデータを以下の URL より入手できます。

http://www.renesas.com/jpn/products/mpumcu/tool/index.html

UNIX 版をお持ちのお客様は、リビジョンアップ依頼を販売元までご連絡下さい。

本パッケージのアップデート内容を以下に示します。
ただし、項番 1 は Windows®版のみです。

1. High-performance Embedded Workshop (Windows®版)

1.1 Make ファイル生成機能の改善

ソースファイルからのインクルードファイルのネストレベルが 10 レベル以下の場合、インクルードファイルのディレクトリ情報が、相対パスで Make ファイルに出力されていましたが、 相対パスで出力されるネストレベルを 10 から 50 レベル以下へ拡張しました。

51 レベル以上の場合は絶対パスで出力されます。

1.2 不具合対策

次の問題を対策しました。

(1) ソースファイルに日本語のコメントが含まれる場合、Navigation ウィンドウからエディタウィンドウに表示されているソースファイルの宣言行または定義行へジャンプすると、実際の宣言行または定義行より後方の行にジャンプすることがある。

(2) ナビゲーション機能が有効になっている状態で、アセンブラで書かれたソースファイルを含むプロジェクトをオープンすると、ナビゲーション機能の自動解析作業が終了しない。

2. コンパイラ

2.1 減算結果と 0 の大小比較 (SHC-0004)

1 バイトもしくは 2 バイトの同じ型の変数同士を減算し、減算結果を同じサイズの signed の型に変換して 0 と大小比較した時に、型変換でオーバフローが生じた場合に型変換後のビットサイズを常に使用するように変更しました。

【例】

```
extern void g(void);
unsigned short a,b;
 f()
 {
   short t = a - b;  // a = 0xffff, b = 0x0000 の場合 t = 0xffff
   if (t > 0) {      //  比較結果は偽となる
        g();
     }
 }
```

3. 最適化リンケージエディタ

3.1 オプションの追加

以下のオプションを追加しました。

(1) DAta_stuff

セクションの境界調整によりコンパイル単位ごとに生じる同一セクション内の空き領域を詰めて、データ領域サイズを削減する。

3.2 最適化指定時のデバッグ情報不正

リンク時の最適化により、デバッグ情報(行情報)が不正に編集されデバッガ上での表示が不正になる場合がある不具合を対策しました。

【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(アセンブル)時に debug,goptimize オプションを指定している。

(2) リンク時の最適化が有効である。

(3) 最適化によって、デバッグ情報の行情報が出力される行の間隔が 255 行になる。

3.3 共通コード統合最適化によるオブジェクト不正(optlnk V.8.00.03 以降)

共通コード統合最適化(optimize=same_code)が有効な場合にオブジェクト不正となる場合がある不具合を対策しました。

不具合は、optlnk V.8.00.03 以降で発生します。

【発生条件】

以下の条件(1)～(3)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) 以下の(a)から(d)のいずれかの条件を満たしている。

(a) エントリ関数(#pragma entry もしくは entry オプション指定された関数)の前後に同一ソースファイル内で記述された関数が存在し、その関数が(2)の最適化の対象となる。

【ソース例】

```
#pragma entry f_entry


void f_a(){
      :
  }


void f_entry(){     // エントリ関数
      :
  }


void f_b(){         // 以降の関数が最適化対象となる
      :
  }
```

(b) リロケータブルファイルを入力時に、関数末尾のコードの直後にリテラルが存在しない関数が(2)の最適化の対象となる。

(c) リロケータブルファイルを入力時に、(2)の最適化によって関数のコード量が 4n+2(2,6,10...)byte 変化する。

(d) 連続して定義された関数(A と B)の両方が(2)の最適化の対象となる、かつ、関数 A は最適化によってコード量が 4n+2(2,6,10...)byte 変化する。

以上