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 06 月 01 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-073A/JA	Rev.	第 1 版
題名	SuperH RISC engine C/C++コンパイラパッケージ V.8.00 Release02 リビジョンアップのお知らせ		情報分類	仕様変更	
適用製品	P0700CAS8-MWR、R0C40700XSW08R P0700CAS8-SLR、R0C40700XSS08R P0700CAS8-H7R、R0C40700XSH08R	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 Release02 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、下記の注意事項を参照して下さい。

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	8.0.01
P0700CAS8-SLR	8.0.00 8.0.01	8.0.00 8.0.01
R0C40700XSS08R	8.00 Release01	8.0.01
P0700CAS8-H7R	8.0.00 8.0.01	8.0.00 8.0.01
R0C40700XSH08R	8.00 Release01	8.0.01

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、項番 2 は Windows®版のみです。

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

1.1 ワークスペースウィンドウ表示改善

ワークスペースウィンドウの[Projects]タブのファイルを、変更日時順にソート表示する機能をサポートしました。また、ビルド対象ファイル（前回のビルド以後に更新されたファイル）をワークスペースウィンドウで識別できるようになりました。

1.2 メイクファイル生成機能拡張

HMake および NMake ファイル形式のメイクファイルに加え GNUMake ファイル形式のメイクファイルを生成できるようになりました。これにより GNUMake をサポートする、汎用のメイクツールを使用可能です。また、コンパイルオプションを別ファイル（サブコマンドファイル）に出力できるようになりました。

1.3 リンク順序のカスタマイズ機能

リンクする順番をカスタマイズする機能をサポートしました。

1.4 バーチャルデスクトップ機能

バーチャルデスクトップ機能をサポートしました。ウィンドウを最大 4 つまでウィンドウコンフィグレーションで保持し、ウィンドウコンフィグレーションを切り替えることで画面を有効に活用することができます。

1.5 ウィンドウ内容の保存機能拡張

新たに以下のウィンドウで表示している内容をテキストファイルに保存できるようになりました。

Cache(SH のみ)、I/O、PA、Register、TLB(SH のみ)

1.6 Watch ウィンドウ仕様変更

Watch ウィンドウに登録した変数は、明示的に削除（登録解除）を行わない限り、ウィンドウを閉じても保持されるようになりました。

1.7 ツールチェインダイアログ改善

ツールチェインオプションダイアログのセクション設定ダイアログのサイズを可変にしました。

1.8 PC のあるソースの直接表示機能

現在の PC があるソースファイルを表示する機能（ツールバーボタン）を追加しました。

1.9 ダウンロード機能拡張

ロードモジュールをダウンロードする際に、ソールファイルが変更されているかどうかを確認し、変更されていた場合、自動的にビルドを行ってからダウンロードを行うように設定できるオプションを追加しました。

また、ロードモジュールダウンロード後、自動的にターゲットをリセットするオプションを追加しました。

1.10 アドレスフィールド改善

アドレスフィールドにラベルリストを参照する機能を追加しました。また、アドレスフィールドに入力された過去 20 件のデータをドロップダウンリストで表示できるようになりました。

1.11 自動バックアップ機能

ワークスペース、プロジェクト、およびセッションファイルを定期的にバックアップする機能をサポートしました。

1.12 表示形式のカスタマイズ機能拡張

フォントやサイズをカスタマイズできる機能を拡張しました。

1.13 エディタ編集中のフリーズ

ナビゲーション機能を有効にしている状態でファイルをエディタにて編集していると、HEW が停止状態となる不具合を対策しました。

1.14 カスタムビルドフェーズを追加後の HEW の不正終了

追加したカスタムフェーズの[オプション]設定で出力ファイルを指定し、[OK]ボタンを押下すると HEW が不正終了する不具合を対策しました。

1.15 ワークスペースウィンドウにヘッダファイルの二重表示

同一のヘッダファイルを大文字と小文字でそれぞれ定義されている場合、ワークスペースウィンドウのプロジェクトタブにヘッダファイル名が二重に表示されてしまう不具合を対策しました。

【例】

File1.c:　#include "SAMPLE.H"

File2.c:　#include "sample.h"

1.16 ナビゲーション表示不正

配列型の変数宣言で要素数にスペース(' ')があると、ワークスペースウィンドウのナビゲーションタブの情報が不正に表示される不具合を対策しました。

【例】

```
extern int tbl[ 2 ];
```

1.17 カスタムビルドフェーズの依存関係不正

プロジェクトにカスタムビルドフェーズを追加した後に、プロジェクトにファイルを追加すると、カスタムビルドフェーズに追加した依存ファイル名が変更されてしまう不具合を対策しました。

2. SuperH RISC engine シミュレータ・デバッガ(Windows®版)

2.1 リードサイクル数、ライトサイクル数の設定

リードサイクル数とライトサイクル数の設定が可能になりました。

3. コンパイラ

3.1 コピー伝播不正

複数の分岐元を持つブロックにコピー命令が存在した場合、不正にコピー命令を削除する場合がある不具合を対策しました。

【例】

```
int func(int *x) {
    int ret=0;
    while(*x++){
        if(*x==1){
            ret+=2;
        }
    }
    return (ret+2);
}

_func:
    MOV     #0,R5    ; 不正にコピーを削除した結果 R7 -> R5 へ変換
```

```
L11:
    MOV.L   @R4,R2
    ADD     #4,R4
                ; *1 MOV R7,R5 を不正に削除
    TST     R2,R2
    ADD     #2,R5
    BT      L13
    MOV.L   @R4,R0
    CMP/EQ  #1,R0
    BT      L11     ; *2 *3 により BF L11 を変換
    BRA     L11
    NOP             ; *3 MOV R5,R7 を不正に削除
L13:
    RTS
    MOV     R5,R0
```

【発生条件】

以下のすべての条件を満たした場合に発生することがあります。

(1) optimize=1 **を指定している。**

(2) **条件文を記述している。**

(3) **複数の分岐元を持つブロックにコピー命令が存在する(例では***1)。

(4) (3)**の分岐元のブロックにコピー命令のコピー元レジスタ(例では** R7)**の定義がないパスがある(例では***2 **から** L11 **へ分岐するパス)。**

3.2 不要式削除不正

条件文の then **節、あるいは** else **節に代入式があり、その直後に同じ変数同士の代入式を記述した場合、不正に条件文を削除する場合がある不具合を対策しました。**

【例】

```
int x;
void f(int y){
    if (y>=256){    /*  不正に削除 */
        x=0;    /* *1 */
      }
     x=x;           /*  *2  同じ変数同士の代入文削除 */
     x++;
 }
        ↓
void f(int y){
     x=0;
     x++;           /*  x=0 を伝播 */
 }
        ↓
```

```
void f(int y){
    x=1;
 }
```

【発生条件】

以下のすべての条件を満たした場合に発生することがあります。

(1) optimize=1 を指定している。

(2) 条件文を記述している。

(3) (2)の条件文の then 節もしくは else 節に代入式がある(例では*1)。

(4) (2)の条件文の後に(3)の代入先変数同士の代入式がある(例では*2)。

3.3 スタック渡し引数アクセス不正

スタック渡し引数を持つ関数の出口直前に関数呼び出しがある場合、speed オプションを指定するとスタック渡し引数を参照するアドレスが不正になる場合がある不具合を対策しました。

【例】

```
typedef struct {
    int x;
} ST;
extern void g(ST *x);
void f(int a, ST b) { /* b はスタック渡し引数 */
    if (a) {
        g(&b);
       /* (A) */
     }
   /* (B) */
 }
```

```
                    ;  関数入口での引数 b の格納アドレス = R15
_f:
    TST     R4,R4
    BT      L12
    MOV     R15,R4
   MOV.L   L14,R2  ; _g
    JMP     @R2     ; (A)
    ADD     #4,R4   ; R4 <- R15+4 : b のアドレスではない
L12:
    RTS             ; (B)
    NOP
```

【発生条件】

以下のすべての条件を満たした場合に発生することがあります。

(1) optimize=1 を指定している。

(2) speed オプションを指定している。

（3） 当該関数がスタック渡し引数を持つ(例では b)。

（4） 当該関数の関数出口が複数ある(例では(A),(B)の 2 ヵ所)。

(5) (4)の中で直前が関数呼び出しである出口がある(例では g(&b);)。

（6） 当該関数内での関数呼び出しは(5)のみである。

3.4 GBR 相対論理演算不正

#pragma gbr_base/gbr_base1 を指定した 1byte の配列、もしくはビットフィールドメンバに対し論理演算を行った場合、論理演算の結果を不正な領域に書き込む場合がある不具合を対策しました。

【例】

```
#pragma gbr_base a,b
char a[2],b[2];
void f() {
    a[0] = b[0] & 1;
 }


    MOV     #_b-(STARTOF $G0),R0
    RTS
    AND.B   #1,@(R0,GBR)          ; 演算結果を b[0]に書き込み
```

【発生条件】

以下のすべての条件を満たした場合に発生することがあります。

(1) gbr=user を指定している。

(2) #pragma gbr_base/gbr_base1 で以下の変数を指定している。

- (unsigned)char 型の配列
- (unsigned)char 型のメンバを持つ構造体配列
- (unsigned)char 型の配列メンバを持つ構造体
- 8bit 以下のビットフィールドメンバを持つ構造体

(3) (2)の変数(例では b[0])に対して定数との論理演算(&, |, ^)を行っている。

(4) (3)の演算の代入先変数(例では a[0])が(2)の条件を満たす。

(5) (3)(4)の変数は別変数、同じ配列で別要素、または同じ構造体で別メンバである。

3.5 拡張削除不正

変数、定数アドレスや配列のインデックスを 1,2byte にキャストした後に、その値を用いてメモリアクセスを行った場合、または unsigned short 型の変数に char 型にキャストした式を代入してから比較式で使用した場合、キャストが削除され不正なメモリ領域をアクセスする場合がある不具合を対策しました。

【例 1】

```
unsigned short x;
char a[1000];


void f() {
    a[(char)x] = 0;
 }
```

```
    MOV.L   L11+2,R2        ; _x
    MOV.L   L11+6,R6        ;_a
    MOV.W   @R2,R5
    EXTU.B  R5,R0
                        ; EXTS.B R0,R0 を削除
    MOV     #0,R5           ; H'00000000
    RTS
    MOV.B   R5,@(R0,R6)     ; x が 0～127 の範囲外の時、不正アドレスを
                            ;  参照する場合あり
```

【例 2】

```
unsigned short us0;
unsigned int b;

func1() {
   unsigned short us1;
   us1 = (char)b;
   return(us0 !=us1);
 }

    MOV.L   L11,R2          ; _b
    MOV.L   L11+4,R5        ; _us0
    MOV.L   @R2,R6
      EXTS.B  R6,R2
    MOV.W   @R5,R6
    EXTU.W  R6,R5
    CMP/EQ  R2,R5           ; (char)b を unsigned short にキャストしないまま比較
    MOVT    R0
    RTS
    XOR     #1,R0
```

【発生条件】

以下のすべての条件を満たした場合に発生することがあります。

(1) optimize=1 を指定している。

(2) 以下の(a)(b)のいずれかの条件を満たす。

(a-1) 変数アドレス、定数アドレス、配列のインデックスを明示的に 1,2byte にキャストしている、もしくは当該関数が char/short の仮引数を持ち、その引数を配列のインデックスのみで使用している。

(a-2) (a-1)の値を用いてメモリアクセスを行う。

(b-1) unsigned short 型の変数に、char 型にキャストした式を代入している。

(b-2) (b-1)の変数を比較式で使用している。

3.6 制限事項解除

外部変数定義の初期値にセクションアドレス演算子と数値の加減算を記述した場合、オブジェクト出力において、数値の加減

算の部分が出力されない不具合を対策しました。

【例】

```
char *addr1 = (char *)__sectop("P");                  // OK
char *addr2 = (char *)__sectop("P") + __secsize("P"); // OK
char *addr3 = (char *)__sectop("P") + 10;             // NG
```

3.7 インターナルエラー対策

以下の場合、インターナルエラーが発生する不具合を対策しました。

（1）関数型の typedef を使用した static 関数メンバをもつクラスの宣言のみ存在する C++プログラムを debug オプションを指定してコンパイルした場合。

（2）{}で囲まれてなく、かつ default ラベルのない switch 文を持つ C/C++プログラムを optimize=0 かつ debug オプションを指定してコンパイルした場合。

4. 最適化リンケージエディタ

4.1 mot/hex ファイル出力時の空き領域出力不正

mot ファイル、もしくは hex ファイルを出力した場合に、命令やデータが存在しない空き領域に対して、"0"(アスキーコード:0x30)を出力すべきところに NULL(アスキーコード:0x00)が不当に出力されてしまう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に、発生する場合があります。

（1） コンパイル(アセンブル)時に、endian=little オプションを指定している。

（2） リンク時に、form=stype(もしくは hex)オプションを指定している。

（3） 複数の入力オブジェクトファイルに、同一セクションに割りつくコード(もしくはデータ)が分かれて存在している。

（4） リンク時の境界調整により"0"が(3)のセクションに埋め込まれる。

4.2 共通コード統合最適化指定時のオブジェクト不正

共通コード統合最適化を指定した場合に、共通化したサブルーチンへの分岐命令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、発生することがあります。

（1） 共通コード統合最適化(optimize=same_code)が有効である。

（2） コンパイル時に goptimize オプションを指定している。

（3） 最適化によって、2 つ以上の共通サブルーチンが生成される。

（4） 生成された共通サブルーチン同士の距離が 4096byte 以内である。

4.3 change_message オプションの複数エラーレベルの指定

change_message オプションで、複数のエラーレベルを指定できない不具合を対策しました。

＜例＞ 下記のように change_message オプションを指定した場合、

バージョンが 8.0.03 以前のリンカでは不当にエラーとなってしまいます。

```
optlnk -change_message=e=1000,w=2000 *.obj
```

以上