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　8月　11日

― ルネサス 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 ルネサス テクノロジ　応用技術統括部
マイコンツール技術部

製品分類	開発環境	発行番号	TN-CSX-052A	Rev.	第1版
題名	SuperH RISC engine C/C++コンパイラ Ver.7.1.03 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		対象ロット等	関連資料		有効期限
	P0700CAS7-MWR P0700CAS7-SLR P0700CAS7-H7R	全ロット	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-702-444A 第2版		永年

SuperH RISC engine C/C++コンパイラパッケージ Ver.7.1.03 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。

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
	7.1.00	7.1.00
	7.1.01 7.1.02	7.1.01
P0700CAS7-SLR	7.0B	7.0B
	7.0.02	7.0.03
	7.0.03	7.0.04
	7.0.04	7.0.06
	7.1.00	7.1.00
	7.1.01 7.1.02	7.1.01
P0700CAS7-H7R	7.0B	7.0B
	7.0.02	7.0.03
	7.0.03	7.0.04
	7.0.04	7.0.06
	7.1.00	7.1.00
	7.1.01 7.1.02	7.1.01

添付：P0700CAS7-030801J
SuperH RISC engine C/C++ Compiler Package
Ver.7.1.03 アップデート内容

SuperH RISC engine C/C++ Compiler Package
Ver.7.1.03 アップデート内容

本パッケージのアップデート内容(不具合修正および機能追加)を以下に示します。

ただし、項番 1 と項番 2 は PC 版のみです。

1. High-performance Embedded Workshop (PC 版)

1.1 ナビゲーション機能強化

C++ソースのナビゲーションをサポートしました。

1.2 スマートエディタのサポート

C++ソースファイルをコーディング中、「.」や「->」、「::」を入力した時にメンバ候補リストを表示し、そのリスト中からメンバを選択することが出来ます。また、関数の場合、開き括弧を入力すると引数並びが表示されます。

1.3 ネットワークデータベースのサポート

ネットワーク経由で HEW を使用する際の、ワークスペースへのアクセス権を設定できるようにしました。

1.4 メイクファイルのインポート機能追加

プロジェクト新規作成時に、メイクファイルのソースファイルパス情報を取り込みます。コンパイラ等のツールオプションは取り込めません。

1.5 複数 CPU の同時デバッグ

複数の CPU を同じ HEW アプリケーション内で接続し､同期させてデバッグできるようにしました。

1.6 複数のツールチェインがインストールされている時の動作改善

同一ワークスペース内に異なるツールチェインのプロジェクトを作成できるようにしました。また、同一ツールチェインのバージョン異なるものが複数インストールされている場合に、どちらのバージョンを使用するかを選択できるようにしました。

1.7 マニュアル一覧表示ツール(Manual Navigator)

マニュアルを一覧表示するツール(Manual Navigator)を添付しました。

1.8 ファイル比較機能サポート

ファイルを比較して、その差分を表示する機能を追加しました。

1.9 TCL/TK ウィンドウのサポート

スクリプト言語 TCL/TK で記述したスクリプトを実行できるようにしました。

1.10タイプライブラリの強化

ワークスペース操作、プロジェクト操作、デバッグ機能の I/F を追加しました。

1.11キャッツ(株)製 ZIPC ツールとの連動の強化

キャッツ(株)製 ZIPC ツールの STM 設計書ブレーク機能および状態遷移トレース機能に対応しました。

1.12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。

SH7294、SH7616

また、以下の CPU の I/O 定義ファイル(iodefine.h)を修正しました。
SH7011、SH7014、SH7016、SH7017、SH7046、SH7047、SH7049、SH7050、SH7050F、SH7051、SH7051F、SH7052F、SH7053F、SH7054F、SH7065、SH7148、SH7615、SH7622

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

2.1 メモリウィンドウの Look&Feel の改善

ウィンドウデザインを見直し、Look & Feel を改善しました。

2.2 コマンドラインウィンドウ機能強化

コマンド入力時、入力文字に応じてコマンド候補リストを表示します。また、コマンド決定後にコマンドパラメータを表示します。

2.3 カバレジ機能強化

カバレジ結果の統計を表示できます。また、ファイル単位のカバレジ測定ができます。

2.4 画像および波形ウィンドウ機能強化

画像ウィンドウおよび波形ウィンドウのリアルタイム表示をサポートしました。プログラム実行中にリアルタイムにウィンドウを更新します。

2.5 トリガ機能強化

設定できるトリガ数を最大で 256 個にしました。

2.6 レジスタウィンドウ機能強化

レジスタウィンドウに表示するレジスタを選択できます。

2.7 ウォッチ機能強化

ウォッチウィンドウを 4 つのシート形式にしました。

2.8 ダウンロード時のメモリ書き込みサイズ指定のサポート

ロードモジュールをダウンロードする際、メモリへの書き込みサイズを指定できるようにしました。

2.9 タイマーサポート

SH-1,SH-2,SH-3,SH-4,SH2-DSP,SH3-DSP に内蔵するタイマーモジュールを 1ch サポートしました。

2.10 イベントウィンドウサポート

従来のブレークウィンドウの機能をイベントウィンドウに移しました。

2.11 メモリマップ操作性向上

メモリマップとメモリリソースのダイアログを 1 枚にして操作性を向上しました。

3. コンパイラ

3.1 ゼロ拡張命令の不当削除

【現象】

ループ内で符号なし変数を複数回参照した時、必要なゼロ拡張命令が削除される場合がある不具合を解決しました。

＜例＞

```
extern unsigned char X;
int sub(int a, int b, int n) {
        int i, sum=0;
        for (i = 0; i < n; i++) {
                if (X == (unsigned char)0xff) {    // X の参照
```

```
            sum += a;
        }
        if (X == (unsigned char)0xf0) {   // X の参照
            sum += b;
        }
        X = X + 1;                        // X の定義
    }
    return (sum);
}

_sub:
        MOV.L   R12,@-R15
        MOV.L   R13,@-R15
        MOV.L   R14,@-R15
        MOV     R5,R13
        MOV     #0,R5
        MOV.L   L19,R1              ; _X
        MOV     R6,R7
        MOV     R4,R12
        MOV     R5,R6
        MOV     #-1,R4
        MOV.B   @R1,R2
                                    ; <- EXTU.B R2,R2 が削除
        MOV     #-16,R14
        EXTU.B  R4,R4
        BRA     L11
        EXTU.B  R14,R14
L12:
        CMP/EQ  R4,R2               ; 比較結果が正しくない場合あり(X>127 の時)
        BF      L14
        ADD     R12,R5
L14:
        CMP/EQ  R14,R2              ; 比較結果が正しくない場合あり(X>127 の時)
        BF      L16
        ADD     R13,R5
L16:
        ADD     #1,R2
        EXTU.B  R2,R2
        ADD     #1,R6
L11:
        CMP/GE  R7,R6
        BF      L12
        MOV.B   R2,@R1
        MOV     R5,R0
        MOV.L   @R15+,R14
        MOV.L   @R15+,R13
        RTS
        MOV.L   @R15+,R12
```

【発生条件】

以下の(1)**から**(4)**の条件、もしくは**(5)**から**(8)**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 **を指定している。**

(2) unsigned char/unsigned short **型変数を使用している**

(3) (2)の変数がループ内で 2 回以上参照した後、定義されている(例参照)。
(4) (2)の変数が最初の参照前にメモリからロードされる。
(5) optimize=1 を指定している。
(6) unsigned char/unsigned short 型ローカル変数を使用している。
(7) (6)の変数がループ内で右シフトのシフト元変数として使用されている。
(8) (7)のシフト数が 2 以上である(SHLR2/SHLR8/SHLR16 に展開される)。

3.2 浮動小数点定数ロードの不当削除

【現象】

同じ値の浮動小数点定数を複数回使用した場合、不当に浮動小数点定数ロード式を削除する場合がある不具合を解決しました。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。

【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 を指定している。
(2) 同じ値の浮動小数点定数をループ内で 2 回以上使用している。
(3) (2)と同じ値の定数をループ外で使用している。

3.3 配列アクセス不正

【現象】

配列アクセス a[c-exp]において、c が定数、exp の型が unsigned char/unsigned short のとき配列を正しくアクセスできない場合がある不具合を解決しました。

＜例＞

```
unsigned char dd[2];
void func(unsigned char a, unsigned char b) {
      dd[15-a]=b;
 }

_func:
        MOV.L    L11,R2          ; _dd
        NEG              R4,R6   ; R6 <- -a
        EXTU.B   R6,R0           ; -a をゼロ拡張
        ADD              #15,R2
        RTS
        MOV.B    R5,@(R0,R2)     ; 異なるアドレスにアクセス
```

【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 を指定している。
(2) 配列の要素を"定数-式(unsigned char/unsigned short)"でアクセスしている
(例では dd[15-a])。

3.4 memmove ライブラリ不正

【現象】

memmove ライブラリ関数でメモリ領域を上位アドレスのメモリ領域に移動すると、指定した文字数以上の移動を行ってしまう不具合を解決しました。

【発生条件】

以下の条件をすべて満たした場合に発生します。
(1) 移動サイズが 31byte 以上である。
(2) 移動先アドレスが移動元アドレスより大きい(メモリ上位へ移動)。

(3) (移動元アドレス+移動サイズ)が移動先領域内にある(移動領域に重なりがある)。
(4) 移動先アドレス、移動元アドレスのどちらかが 4 の倍数でない。

3.5 無条件分岐の不当削除

【現象】

関数の最後の処理が条件文で、条件がネストしており、かつ最後の条件節が関数呼び出し+return 文、その前の条件節が関数呼び出しで終わっている場合、関数出口への無条件分岐が不当に削除される場合がある不具合を解決しました。

＜例＞

```
void sub(int parm) {
    if (parm == 0) {
        ;
    } else if (parm == 1) {
        ;
    } else if (parm == 2) {
        ;
    } else if (parm == 3) {
        ;
    } else if (parm == 4) {
        ;
    } else if (parm == 5) {
        func1();    /* <A> */
    } else {
        func2();    /* <B> */
        return;     /* <B> */
    }
    return;
}

_sub:
        STS.L       PR,@-R15
        TST         R4,R4
        BT          L11
        MOV         R4,R0
        CMP/EQ      #1,R0
        BT          L11
        CMP/EQ      #2,R0
        BT          L11
        CMP/EQ      #3,R0
        BT          L11
        CMP/EQ      #4,R0
        BT          L11
        CMP/EQ      #5,R0
        BF          L18
        MOV.L       L20+2,R2        ; _func1
        JSR         @R2
        NOP
L11:
                                    ; L19 への分岐命令を削除
L18:
        MOV.L       L20+6,R2        ; _func2
        JMP         @R2             ; 常に関数呼び出しされる
        LDS.L       @R15+,PR
```

```
L19:
        LDS.L       @R15+,PR
        RTS
        NOP
```

【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 **を指定している。**
(2) **関数の最後の処理が条件文でかつ条件がネストしている。**
(3) **最後の条件節が関数呼び出し**+return **文で終わっている(例の**<B>)。
(4) (3)**の直前の条件節が関数呼び出しで終わっている(例の**<A>)。

3.6 unsigned **型->**float **型キャスト不正**

【現象】
unsigned **型の変数を** float **型に明示的にキャストした時、キャストが不当に削除される場合がある不具合を解決しました。**

＜例＞

```
unsigned short us1;
volatile unsigned short us0;
volatile float f0;
float *p;
void func() {
      f0 = *p = ((float)us0, (float)us1);
}

_func:
        MOV.L       L29+50,R2       ; _us0
        MOV.L       L29+54,R5       ; _p
        MOV.W        @R2,R6
        MOV.L       L29+58,R6       ; _us1
        MOV.W        @R6,R2
        EXTU.W       R2,R6
        MOV.L       @R5,R2
        MOV.L       R6,@R2 ; float に変換しないで*p にストア
        MOV.L       @R5,R2
        MOV.L       @R2,R6
        MOV.L       L29+10,R2       ; _f0
        RTS
        MOV.L       R6,@R2 ; float に変換しないで f0 にストア
```

【発生条件】
以下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) unsigned **型の変数を** float **型にキャストしている。**
(2) unsigned **型の変数を** double **型にキャストし、かつ** double=float **または** fpu=single **オプションを指定、または** long double **型にキャストし、かつ** fpu=single **を指定している。**

3.7 ld_ext()、st_ext()**使用時のスタックポインタ不正移動**

【現象】
SH-4 **で** ld_exp()**または** st_ext()**組み込み関数使用時にパラメタにローカル配列を指定した場合、不正にスタックポインタを移動するコードを出力する場合がある不具合を解決しました。**

＜例＞

```
#include <machine.h>

void main() {
        float table[4][4], data1[4][4], data2[4][4];
                :
        ld_ext(table);
        mtrx4mul(data1,data2);
                :
}
```

```
                :
        FRCHG
        FMOV.S      @R15+,FR0       ; R15 を更新、この間で例外が発生した場合
                                    ;上位のスタック領域が破壊
        FMOV.S      @R15+,FR1       ;                       |
        FMOV.S      @R15+,FR2       ;                       |
        FMOV.S      @R15+,FR3       ;                       |
        FMOV.S      @R15+,FR4       ;                       |
        FMOV.S      @R15+,FR5       ;                       |
        FMOV.S      @R15+,FR6       ;                       |
        FMOV.S      @R15+,FR7       ;                       |
        FMOV.S      @R15+,FR8       ;                       |
        FMOV.S      @R15+,FR9       ;                       |
        FMOV.S      @R15+,FR10      ;                       |
        FMOV.S      @R15+,FR11      ;                       |
        FMOV.S      @R15+,FR12      ;                       |
        FMOV.S      @R15+,FR13      ;                       |
        FMOV.S      @R15+,FR14      ;                       |
        FMOV.S      @R15+,FR15      ;                       |
        FRCHG                       ;                       |
        ADD         #-64,R15        ;                       V
                :
```

【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=sh4 **を指定し、かつ** ld_ext()、st_ext()**を使用している。**

(2) ld_ext(),st_ext()**のパラメタにローカル配列を指定している。**

3.8 データ出力時不正

【現象】

"シンボルアドレス+オフセット値"の初期値をもつ変数が多く存在する場合、内部エラーもしくはオブジェクト不正になる場合がある不具合を解決しました。

【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) code=machinecode(**デフォルト**)**を指定している。**

(2) listfile **オプションを指定していない、もしくは** listfile **オプションと** show=noobject **オプションを同時に指定している。**

(3) **初期値つき変数の定義が存在する。**

(4) (3)**の初期値が"シンボルアドレス＋オフセット値"または構造体先頭でないメンバのアドレスである。**

(5) (3)**の変数、および初期値が以下の条件を満たしている。**

((3)**の数**+(4)**のオフセット値の** 10 **進桁数の合計**)≧33000

＜例＞

```
extern char g;
```

```
#define DATA1A      (&g+2147483647)
#define DATA10A     DATA1A,  DATA1A,  DATA1A,  DATA1A,  DATA1A,¥
                    DATA1A,  DATA1A,  DATA1A,  DATA1A,  DATA1A
#define DATA100A DATA10A, DATA10A, DATA10A, DATA10A, DATA10A,¥
                    DATA10A, DATA10A, DATA10A, DATA10A, DATA10A

/*  以下は(変数の数+offset の合計桁数)=(3001+10*3001)=33011>33000 となる  */
char *a1[1000] = {
      DATA100A, DATA100A, DATA100A, DATA100A, DATA100A,
      DATA100A, DATA100A, DATA100A, DATA100A, DATA100A
};
char *a2[1000] = {
      DATA100A, DATA100A, DATA100A, DATA100A, DATA100A,
      DATA100A, DATA100A, DATA100A, DATA100A, DATA100A
};
char *a3[1000] = {
      DATA100A, DATA100A, DATA100A, DATA100A, DATA100A,
      DATA100A, DATA100A, DATA100A, DATA100A, DATA100A
};
char *a = DATA1A;
```

以上