RJJ10J2680-0100

Renesas Starter Kit for R8C/38C
チュートリアルマニュアル

ルネサスマイクロコンピュータ

R8C ファミリ R8C/3x シリーズ

Rev.1.00
発行：2010 年 4 月 1 日

株式会社ルネサスソリューションズ
japan.renesas.com

目次

1. まえがき

ご注意

本書の内容の一部または全は、予告無しに変更されることがあります。

本書の著作権は(株)ルネサスソリューションズにあります。(株)ルネサスソリューションズの書面での承諾無しに、本書の一部または全てを複製することを禁じます。

商標

本書で使用する商標名又は製品名は、各々の企業、組織の商標または登録商標です。

著作権

ウェブサイト: http://japan.renesas.com/ (日本サイト)

http://www.renesas.com/ (グローバルサイト)

用語解説

CD Compact Disc
(コンパクトディスク)

CPU Central Processing Unit
(中央処理装置)

E8a
(E8a オンチップデバッギングエミュレータ)

LCD Liquid Crystal Display
(液晶ディスプレイ)

LED Light Emitting Diode
(発光ダイオード)

MCU Microcontroller Unit
(マイクロコントローラユニット)

PC Program Counter
(プログラムカウンタ)

RAM Random Access Memory
(ランダムアクセスメモリ)

ROM Read Only Memory
(リードオンリーメモリー)

RSK Renesas Starter Kit
(ルネサススタータキット)

USB Universal Serial Bus
(ユニバーサルシリアルバス)

2. はじめに

本マニュアルは、Renesas Starter Kit をご使用の際、最も多く寄せられる質問に対し、チュートリアル形式でお答えするものです。本チュートリアルでは、以下の項目について説明しています。

- Renesas Starter Kit で簡単なプログラムをコンパイル、リンク、ダウンロードおよび実行する方法は？
- エンベデッド・アプリケーションの構築方法は？
- ルネサス・ツールの使用方法は？

プロジェクト・ジェネレータは、選択可能な 2 種類のビルド・コンフィグレーションを持つチュートリアル・プロジェクトを作成します。

- 'Debug' は、デバッガのサポートを含むプロジェクトを構築します。
- 'Release' は、製品リリース用に適したコードを構築します。

本マニュアルで参照ファイルは、チュートリアルを進めていく過程で、プロジェクト・ジェネレータにてインストールします。本チュートリアルの使用例は、クイックスタートガイドに記載のインストールが完了していることを前提としています。コンフィグレーション設定の詳細については、クイックスタートガイドをご覧下さい。

ご注意： これらのチュートリアルは、Renesas Starter Kit の使用方法の説明を目的とするものであり、High-performance Embedded Workshop デバッガ、コンパイラ・ツールチェインまたは E8a エミュレータの入門書ではありません。これらに関する詳しい情報は、関連するマニュアルをご覧下さい。

3. チュートリアルプロジェクトワークスペース

ワークスペースには、2 種類のビルド・コンフィグレーション用の全ファイルを含みます。チュートリアル・コードは、デバッグおよびリリース・ビルド・コンフィグレーションの両方で共通です。

High-performance Embedded Workshop のビルド・コンフィグレーション・メニューを使用し、各々のビルド・コンフィグレーションから特定のファイルを除外して、プロジェクトを作成することができます。これにより、デバック・ビルドにはモニタを含み、リリース・ビルドには含まないといった設定が可能になります。共通の C ファイルの内容は、ビルド・コンフィグレーション・オプションの defines セットアップおよび同ファイル内の`#ifdef` ステートメントで管理されます。

プロジェクト・ファイルは 1 つのセットのみを取扱うことで、管理の簡素化が図れます。

4. プロジェクトワークスペース

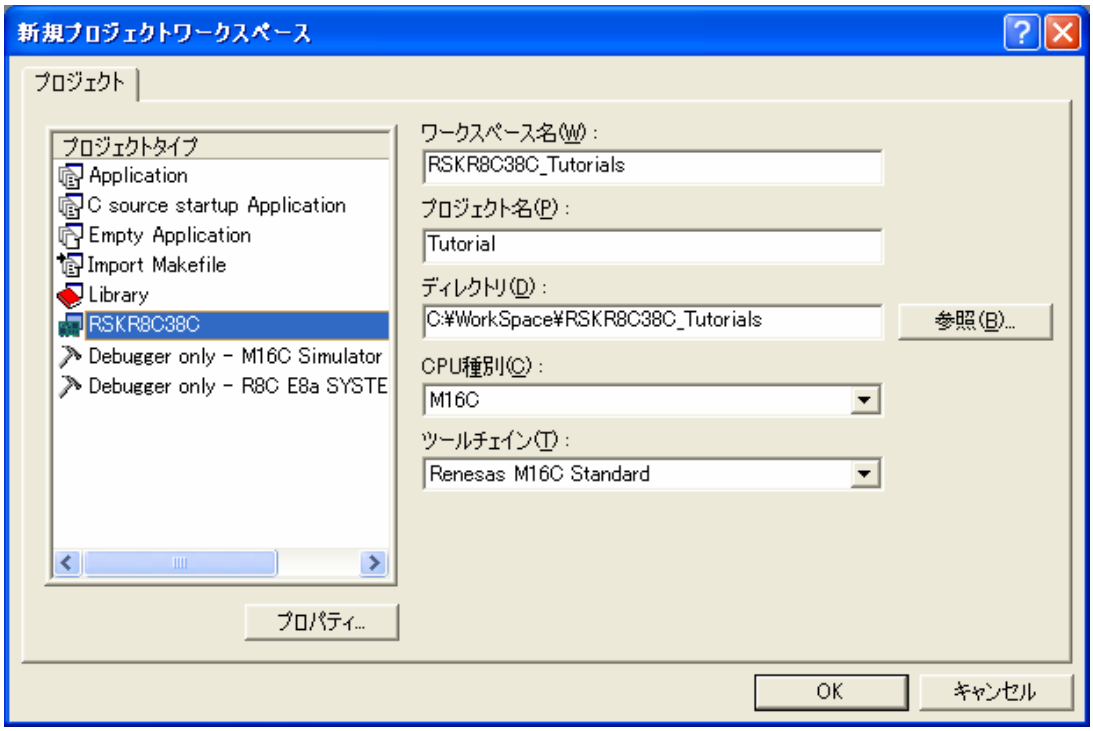
4.1 はじめに

High-performance Embedded Workshop はルネサスの統合開発ツールで、ユーザはこれを使用して、ルネサス・マイクロコントローラのソフトウェア・プロジェクトをコンパイル、プログラム、デバッグすることができます。High-performance Embedded Workshop は Renesas Starter Kit 製品インストール時にインストールされます。本チュートリアルでは、提供のチュートリアル・コードの作成およびデバッグに必要な作業を段階的に説明します。

4.2 新規プロジェクトワークスペースの作成

先ず、Windows のスタートメニューから High-performance Embedded Workshop を起動して、チュートリアル・プログラムを見てみましょう。

[ファイル -> 新規ワークスペース...]メニューから新規ワークスペースを開くか、または 'ようこそ！' ダイアログで '新規プロジェクトワークスペースの作成' を選択して下さい。



上の図は RSKR8C38C 選択時の新規プロジェクト・ワークスペースの例です。

- Renesas Starter Kit 用に 'M16C' CPU 種別およびツールチェインを選択します。
- プロジェクト・リストから Renesas Starter Kit 用プロジェクト・タイプ 'RSKR8C38C' を選択します。
- ワークスペース名を入力します。全てのファイルはこの名称のディレクトリ下に置かれます。
- プロジェクト名欄は、上記ワークスペースと同じ名前で自動的に入力されますが、これは変更可能です。
  ご注意： High-performance Embedded Workshop では複数のプロジェクトを 1 つのワークスペースに追加できます。後にサンプル・コードのプロジェクトを保存する可能性がありますので、ここでは本チュートリアル・プロジェクトに適した名称をつけることを推奨します。
- <OK>をクリックし、Renesas Starter Kit プロジェクト・ジェネレータ・ウィザードを起動します。

次のダイアログに、利用可能なプロジェクト例が表示されます。後に説明する Tutorial コードを選択して下さい。その他のオプションとして、各種周辺機能の使用例を示す Sample コードがあります。これを選択すると、新たなダイアログが開き、デバイス周辺機能用のサンプル・コードがいくつか表示されます。最後のオプションは、アプリケーション・ビルド用で、デバッガは設定されていますが、プログラム・コードはありません。これは、ユーザがデバッガを設定せずにコードを新規作成したい場合に適しています。

- 作成するプロジェクト・タイプとして“Tutorial”を選択し、<Next>を押します。
- <Finish>をクリックし、プロジェクトを作成します。

プロジェクト・ジェネレータ・ウィザードが確認ダイアログを表示します。<OK>をクリックすると、プロジェクトを作成し、必要なファイルをコピーします。

このプロジェクトの全ファイルを示すツリーが High-performance Embedded Workshop に表示されます。

- ワークスペース画面で ‘main.c’ ファイルをダブルクリックします。画面にコードが表示されます。

4.3 ビルドコンフィグレーションとデバッグセッション

作成されたワークスペースには、2 つのビルド・コンフィグレーションと 2 つのデバッグ・セッションが含まれています。このビルド・コンフィグレーションでは、同じプロジェクトを異なるコンパイラオプションでビルドすることが可能です。ユーザが利用できるオプションは、High-performance Embedded Workshop のマニュアルに詳しく記載されています。

4.3.1 ビルドコンフィグレーション

ツールバーの左側のドロップダウンリストからビルド・コンフィグレーションを選択します。利用可能なオプションは、Debug と Release です。Debug ビルドは、デバッガとの使用に設定されています。Release ビルドは、最終 ROM コード用の設定です。

これら 2 種のビルドの違いとして、最適化設定が挙げられます。最適化が有効の場合、デバッガがコードを予想外の順序で実行するようなケースがあり、デバッグをスムーズに処理する為には、デバッグされるコードの最適化を無効にすることを推奨します。

- ‘Debug’ コンフィグレーションを選択します。

4.3.2 デバッグセッション

デバッグ・セッションはツールバーの右側のドロップダウンリストから選択します。Renesas Starter Kit の種類によってオプションは異なりますが、どのオプションも必ずデバッグを可能にする同様のデバッグ・インタフェースを含みます。その他の選択として ‘DefaultSession’ があります。デバッグ・セッションの目的は、同一プロジェクトで異なったデバッガ・ツールの使用や、異なったデバッガ設定を可能にすることにあります。

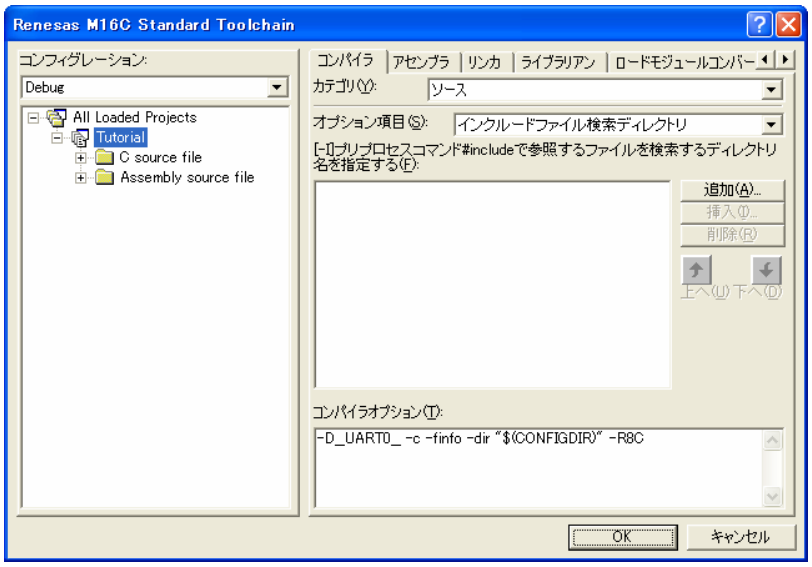
- ‘SessionR8C_E8a_SYSTEM’ デバッグ・セッションを選択します。

5. チュートリアルプロジェクトのビルド

チュートリアル･プロジェクトのビルド設定は、ツールチェインオプションで既に設定されています。ツールチェインオプションを表示する為には、‘ビルド’メニュー項目のツールチェインを選択して下さい。

表示されるダイアログは、選択したツールチェインにより異なります。

画面左側のコンフィグレーション画面は、全ツールチェインオプションに存在します。どのような設定を変更する場合でも、変更する部分の現在のコンフィギュレーションに注意して下さい。全てのまたは複数のビルド･コンフィギュレーションの変更は、‘コンフィグレーション’ドロップダウンリストから‘All’または‘Multiple’を選択することで可能になります。



- 各々のタブおよび‘カテゴリ’ドロップダウンリストをチェックし、利用可能なオプションを確認して下さい。

選択終了後、<OK>をクリックしてダイアログを閉じます。

5.1 コードのビルド

プロジェクトのビルド用に 3 つのショートカットがあります。

1. ツールバーの‘すべてをビルド’ボタンを選択。 
   プロジェクト中の全ファイルをビルドします。これは、標準ライブラリを含みます。
2. ツールバーの‘ビルド’ボタンを選択。 
   前回から変更のあった全ファイルをビルドします。オプションを変更しない限り、ここでは標準ライブラリはビルドされません。
3. ‘F7’を押す。
   これは、上記の‘ビルド’ボタン選択の場合と同等です。

ここで、‘F7’を押すか、または上記アイコンの 1 つを選択し、プロジェクトをビルドします。

ビルド中の各段階で、アウトプット画面にビルド状況が表示されます。

ビルド終了時、ビルド中に発生したエラーおよび警告の表示がされます。

5.2 デバッガの接続

本チュートリアルでは、外部から CPU ボードに電源を提供する必要はありません。電源は E8a を経由して USB ポートから供給します。USB ポートに接続されているデバイスが多すぎると、OS（Windows）がシャットダウンすることがありますのでご注意下さい。この場合、デバイスをいくつか取外し、再度、接続を試して下さい。外部電源を供給することも可能ですが、極性および電源電圧が適切であることを必ず確認して下さい。

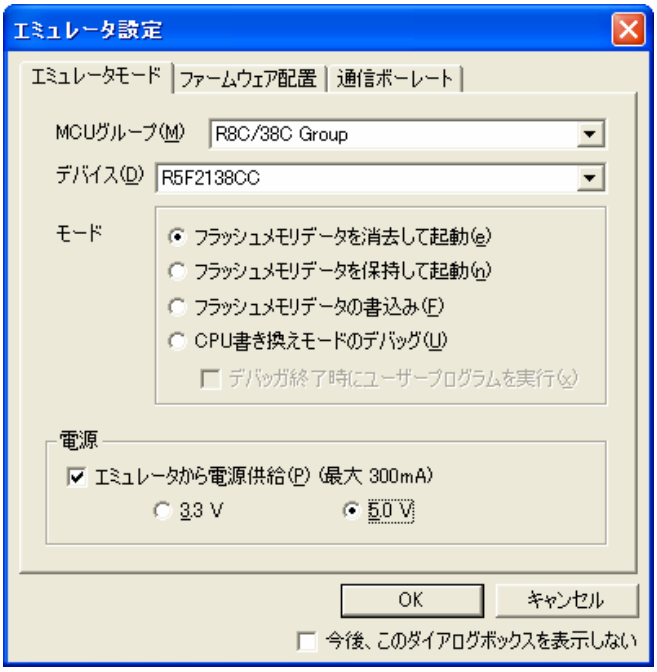
E8a のホスト・コンピュータへの接続方法は、Renesas Starter Kit に同梱のクイックスタートガイドに詳しく記載されています。以下は、クイックスタートガイドの手順が踏まれ、E8a 用のドライバが既にインストールされていることを前提としています。

4. LCD モジュールを CPU ボードの LCD コネクタに取り付け、J3 の上に位置するようにします。コネクタの全てのピンがきちんとソケットに収まっていることを確認して下さい。
5. E8a をご使用のコンピュータの USB ポートに接続します。
6. E8a を CPU ボードに接続します。その際、DC パワージャックの近くにある、E8a とシルク印字されたコネクタに接続されることを確認して下さい。
7. ボードに外部電源を供給の場合、この時点で電源を供給します。

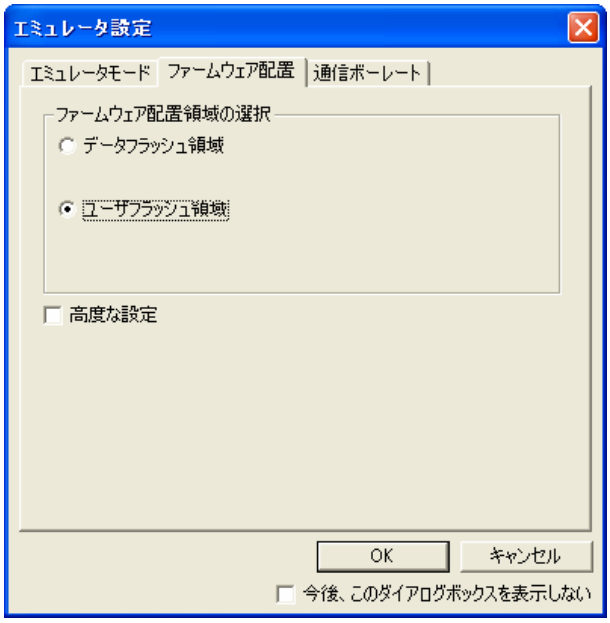
5.3 E8a使用時のターゲットへの接続

ここでは、デバイスへの接続、フラッシュへのプログラミングおよびコード実行について説明します。

8. ‘SessionR8C_E8a_SYSTEM’ デバッグ・セッションを選択します。
9. デバッグツールバーの<接続>ボタンをクリックします。 
10. 適切な MCU グループとデバイスを選択して下さい（RSKR8C38C の場合、R8C/38C Group、R5F2138CC を選択）。
11. ‘フラッシュメモリデータを消去して起動’ を選択します。
12. E8a が CPU ボードに電源を供給する場合は、’エミュレータから電源供給’ を選択し、’5.0V’ を選択します。それ以外の場合は、センタープラスの外部電源(5V)をボードに供給して下さい。

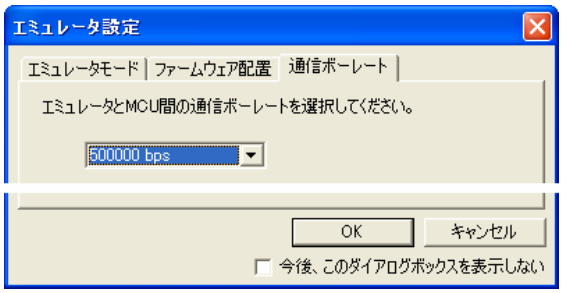


13. ‘ファームウェア配置’タブをクリックします。高度な設定はチェックせずに、‘ユーザフラッシュ領域’を選択します。



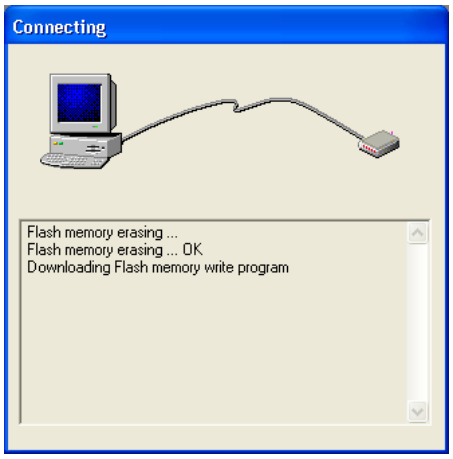
14. ‘通信ボーレート’タブをクリックし、プルダウンメニューから 500000bps を選択します

15. <OK>をクリックします。



16. フラッシュメモリ書き込みプログラムがターゲットに書き込まれます。

17. High-performance Embedded Workshop のアウトプット画面に‘Connected’と表示されます。



ここで、High-performance Embedded Workshop のセッションを保存することを推奨します。

18. ‘ファイル’|‘セッションの保存’を選択します。

ワークスペースの設定を変更した場合、ワークスペースを保存することを推奨します。

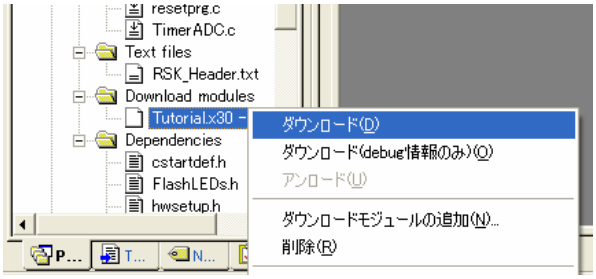
19. ‘ファイル’|‘ワークスペースの保存’を選択します。

6. チュートリアルのダウンロードと実行

High-performance Embedded Workshop でのコード作成が完了したら、それを CPU ボードにダウンロードする必要があります。

この時点でワークスペースビューに ‘Download Modules’ のカテゴリが追加されます。

20. ダウンロードモジュールのリストから関連モジュールを右クリックし、‘ダウンロード’ を選択します。



ダウンロードが完了すると、デバッガとコードの実行が可能になります。

デバッグを始める前に、デバッガおよびターゲットをリセットする必要があります。

21. デバッグツールバーの ‘CPU リセット’ を押します。 

ファイル画面にチュートリアル・コードが開始位置で開きます。矢印はプログラムカウンタの現在位置を示します。

```
/*""FUNC COMMENT""*****************************************************************
* Outline       : start
* Description   : Power on reset function. This function executes following to
*                 power on reset. It first calls hardware initialisation
*                 function & then 'main()' function.
* Argument      : none
* Return value  : none
*""FUNC COMMENT END""*************************************************************/

void start(void)
086B0 {
        /* Set interrupt stack pointer */
086B0   _isp_   = &_istack_top;
        /* Change protect mode register */
086B4   prcr    = 0x02U;
        /* Set processor mode register */
086B8   pm0 = 0x00U;
        /* Change protect mode register */
086BB   prcr    = 0x00U;
        /* Set flag register */
086BE   _flg_   = __F_value__;
#if __STACKSIZE__!=0
        /* Set user stack pointer */
086C2   _sp_    = &_stack_top;
#endif
        /* Setting 400H (Do not change) */
086C6   _sb_    = 0x400U;
        /* Set variable vector's address */
086CA   _intbh_ = 0x00U;
086CE   _asm("  ldc #(topof vector)&0FFFFh,INTBL");

        /* Initialize each sections */
086D2   initsct();

#if __HEAPSIZE__ != 0

        /* Initialize heap */
        heap_init();
#endif
#if __STANDARD_IO__ != 0
        /* Initialize standard I/O */
        _init();
#endif
        /* Initialize FB register for debugger */
086D6   _fb_ = 0U;

        /* Set up the hardware  */
086DA   HardwareSetup();

        /* Call main() routine */
086DE   main();

        /* Call exit */
086E2   exit();
086E5 }
/*****************************************************************************
End of function start
*****************************************************************************/
```

ここでは、初期化コードを飛ばして、メイン・チュートリアルに移ります。

22. ‘resetprg.c’ファイルを開きます。

23. main()にブレークポイントを設定します。

ブレークポイントは、次の方法で設定可能です。ブレークポイントを挿入したい行（矢印と同じ欄）をダブルクリックする、または行を選択しF9を押す、または行を右クリックし、‘ブレークポイントの挿入/削除’を選択します。その他に、ブレークポイント欄の左側の欄をクリックして、イベントポイントを設定することも可能です。イベントポイントは8つまで設定できます。これは、フラッシュメモリの書き換えが発生しませんので、ブレークポイントよりも応答が速いです。

24. デバッグツールバーの‘リセット後実行’を押します。 

コードは設定したブレークポイントまで実行されます。この時点で、CPUおよび周辺機能の初期化は終了しています。

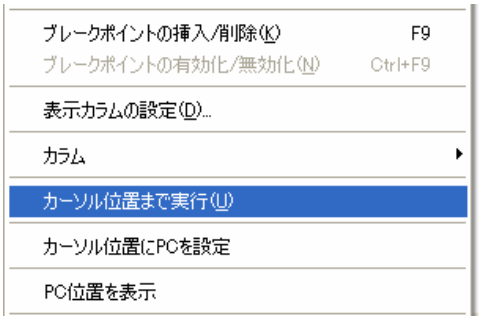
25. デバッグツールバーの‘ステップイン’を押します。 

‘main.c’が開き、プログラムカウンタは新しい位置を示します。

```
                /*""FUNC COMMENT""*****************************************************************
                * Outline        : main
                * Description    : Main program. This function calls timer, ADC & LCD
                                   initialisation functions. The user LEDs flashes until
                                   the user presses a switch on the RSK.
                * Argument       : none
                * Return value   : none
                *""FUNC COMMENT END""*************************************************************/

                void main(void)
085B0           {
                    /*  Reset the LCD module. */
085B0               InitialiseDisplay();

                    /*  Display Renesas Splash Screen. */
085B4               DisplayString(LCD_LINE1, "Renesas");
085C4               DisplayString(LCD_LINE2, NICKNAME);

                    /*  Flash the user LEDs for some time or until a push button is pressed. */
085D5               FlashLEDs();

                    /*  Flash the user LEDs at a rate set by the user potentiometer (ADC) using
                        interrupts. */
085D9               TimerADC();

                    /*  Demonstration of initialised variables. Use this function with the
                        debugger.    */
085DD               Statics_Test();

                    /* This function must not exit */
085E0               while(1);
085E4           }
                /******************************************************************************
                End of function main
                ******************************************************************************/
```

26. TimerADC()関数にブレークポイントを設定します。

27. FlashLEDs()上で右クリックし、‘カーソル位置まで実行’を選択します。



コードは選択した行まで実行され、停止します。このとき、自動的にブレークポイントが挿入され、ブレークが発生した後にブレークポイントが削除されます。

28. デバッグツールバーの‘ステップオーバ’を押します。 

コードが実行され、LED が 200 回点滅します。200 回の点滅が終了するまで、または CPU ボード上のスイッチが押されるまで、デバッガは終了しません。

29. LED の点滅を停止したい場合、CPU ボードの SW ボタンを押して FlashLEDs()から抜けて下さい。

設定済みの TimerADC()のブレークポイントまでコードが実行されます。

TimerADC 関数は、内部タイマの割り込みを初期化します。タイマ・モジュールのコンペアマッチで割込みが発生します。TimerADC コードでは、割り込みによって外部ポテンショメータの AD 変換値を読み、その結果を次のコンペアマッチの値の設定に使用します。その後、AD 変換が再スタートします。

割込みの初期化はハードウェアセットアップの一部として実行されます。これは‘interrupts.c’ファイルに含まれています。

30. ワークスペースビュー上で‘interrupts.c’をダブルクリックし、ファイルを開きます。
31. このファイル中で、LED 表示を変更する割り込み機能_timer_rc(void)を確認します。
32. LED 表示が変更される行にブレークポイントを設定します。
33. <実行>または<F5>を押して、現在の PC 位置からコードを実行します。 

コードは割り込みルーチンで停止します。ここで、割り込み関数をステップして確認できます。

34. この関数から抜け出る前に割り込みのブレークポイントをダブルクリックし、ブレークポイントを解除します。
35. <実行>を押して、現在の PC 位置からコードを実行します。 

コードは、main 関数内の無限ループまで実行されます。ここで、LED が点滅しているはずです。ボードのポテンショメータを調整することで点滅のレートを変更できます。

36. デバッグツールバーの<停止>を押します。 

37. ‘CTRL-B’を押してブレークポイント画面を開きます。

イベントポイント(本書 11 ページ上段参照)を使用している場合、ブレークポイント画面は開きません。‘CTRL-E’でイベントポイント画面が表示されます。イベントポイント画面中の Breakcondition タブでイベントポイントの設定を見ることができます。

38. ‘すべて削除’を選択します。

39. <OK>を押します。

40. ‘main.c’ファイルを開きます。

41. `Statics_Test()`にブレークポイントを設定します。

StaticsTest 関数を用いて、フラッシュに格納されている初期化済みの変数が、RAM にコピーできたことを示します。

42. デバッグツールバーの<リセット後実行>を押します。 

コードは設定したブレークポイントで停止します。(ボードの SW1 ボタンを押して LED 点滅テストを回避します)

43. デバッグツールバーの<ステップイン>を押します。 

コードのデバッグ中、変数を‘watch’(監視)することができます。監視したい変数上にマウスポインタを置き、その変数が有効であれば、ポインタ位置に出現する小画面に、現在の変数値が表示されます。

44. ‘ucStr’変数の上にマウスポインタを置いて、現在の数値を見てみましょう。変数を右クリックし、‘インスタントウォッチ’を選択します。

ダイアログが開き、その変数と関連オプションが表示されます。

45. <登録>を押します。

ダイアログを閉じると、ワークスペースにその変数を含む新規画面が開きます。

文字列が‘STATIC’に正常に初期化されたことを確認できます。

46. for ループ内の `DisplayString()`にブレークポイントを設定します。

47. <実行>を押して、現在の PC の位置からコードを実行します。 

プログラムが停止した時点で、LCD の 2 行目に変更された文字列が表示されます。
変数列の一番目の文字が変換定数列の一番目の文字に置き換えられていることが、watch 画面で確認できます。

48. ブレークポイントを解除します。

49. ループ後の `DisplayString()`を右クリックし、‘カーソル位置まで実行’を選択します。

これは、変数がプログラムのスタートアップで初期化されており、‘TESTTEST’で上書き可能であることを示します。

チュートリアル・コードを実行し、デバッガのいくつかの機能を使用してみました。関数の多くは、コード実行、コンパイラ規則などの重要な情報を備えていますので、残りのチュートリアル・コードもご覧になることをお勧め致します。プロジェクト・ファイルの詳細は 7 章のプロジェクトファイルを参照して下さい。

7. プロジェクトファイル

7.1 標準プロジェクトファイル

Renesas Starter Kit チュートリアルは、複数の Renesas Starter Kit 製品で同じチュートリアル・コードを使用できるよう構成されています。これにより、異なったプロセッサ・コアを同等のコードで評価することが可能です。以下に示すファイルは、全てのマイコンおよびツールチェインで共通です。
チュートリアル・ファイルは各々のコードの機能を詳しく説明するコメントを含んでいます。コンパイラ特定の用途、動作の詳細は、ソース・コードを参照して下さい。

7.1.1 初期化コード(resetprg.c / resetprg.h)

下記はメイン・チュートリアル・コードの開始位置です。

```
/*""FUNC COMMENT""*****************************************************************
* Outline       : start
* Description   : Power on reset function. This function executes following to
*                 power on reset. It first calls hardware initialisation
*                 function & then 'main()' function.
* Argument      : none
* Return value  : none
*""FUNC COMMENT END""*************************************************************/

void start(void)
{
    /* Set interrupt stack pointer */
    _isp_   = &_istack_top;
    /* Change protect mode register */
    prcr    = 0xA502U;
    /* Set processor mode register */
    pm0 = 0x00U;
    /* Change protect mode register */
    prcr    = 0xA500U;
    /* Set flag register */
    _flg_   = __F_value__;
#if __STACKSIZE__!=0
    /* Set user stack pointer */
    _sp_    = &_stack_top;
#endif
    /* Setting 400H (Do not change) */
    _sb_    = 0x400U;
    /* Set variable vector's address */
    _intbh_ = 0x00U;
    _asm("  ldc #(topof vector)&0FFFFh,INTBL");

    /* Initlalize each sections */
    initsct();

#if __HEAPSIZE__ != 0

    /* Initialize heap */
    heap_init();
#endif
#if __STANDARD_IO__ != 0
    /* Initialize standard I/O */
    _init();
#endif
    /* Initialize FB register for debugger */
    _fb_ = 0U;

    /* Set up the hardware  */
    HardwareSetup();

    /* Call main() routine */
    main();

    /* Call exit */
    exit();
}
/*******************************************************************************
End of function start
*******************************************************************************/
```

C コンパイラで使用の変数の初期化およびスタックポインタの初期化は、`initsct()`で行われます。

`HardwareSetup()`の呼び出しでデバイスのハードウェアおよび周辺機能を初期化し、チュートリアル・ソフトウェアへの準備が整います。

`main()`でメイン・デモンストレーション・コードが始まります。

7.1.2 ボード初期化コード(hwsetup.c / hwsetup.h)

マイクロコントローラ・デバイスの設定は、一般的に 4 段階から成り、これを示すコードは 4 つの関数に分かれています。各々の関数は、対応のデバイス用に書かれます。関数コールは以下の通りです。

```
/*""FUNC COMMENT""*****************************************************************
* Outline        : HardwareSetup
* Description    : Sets up the hardware.
*                  This function calls the hardware initialization functions to
*                  configure the CPU operating frequency, port pins & required
*                  on-chip modules in order to setup the RSK for the main
*                  application.
* Argument       : none
* Return value   : none
*""FUNC COMMENT END""*************************************************************/

void HardwareSetup(void)
{
    /* Configures CPU clock */
    ConfigureOperatingFrequency();

    /* Configures port pins */
    ConfigurePortPins();

    /* Enables required on-chip peripherals */
    EnablePeripheralModules();

    /* Configures interrupts    */
    ConfigureInterrupts();
}
/*****************************************************************************
End of function HardwareSetup
*****************************************************************************/
```

7.1.3 メイン・チュートリアル・コード(main.c / main.h)

メイン・チュートリアル・コードは全てのチュートリアル・プロジェクトで共通です。表示の初期化および文字列表示関数はLCD モジュール上で文字列を表示します。ルネサス提供以外の LCD モジュールを接続する場合、ks0066u コントローラとの互換性およびピン接続を接続図にて確認して下さい。

```
/*""FUNC COMMENT""*****************************************************************
* Outline       : main
* Description   : Main program. This function calls timer, ADC & LCD
                  initialisation functions. The user LEDs flashes until
                  the user presses a switch on the RSK.
* Argument      : none
* Return value  : none
*""FUNC COMMENT END""*************************************************************/

void main(void)
{
    /*  Reset the LCD module. */
    InitialiseDisplay();

    /*  Display Renesas Splash Screen. */
    DisplayString(LCD_LINE1, "Renesas");
    DisplayString(LCD_LINE2, NICKNAME);

    /*  Flash the user LEDs for some time or until a push button is pressed. */
    FlashLEDs();

    /*  Flash the user LEDs at a rate set by the user potentiometer (ADC) using
        interrupts. */
    TimerADC();

    /*  Demonstration of initialised variables. Use this function with the
        debugger.   */
    Statics_Test();

    /* This function must not exit */
    while(1);
}
/*****************************************************************************
End of function main
*****************************************************************************/
```

8. 追加情報

High-performance Embedded Workshop の使用法の詳細は、CD またはウェブサイトに掲載の High-performance Embedded Workshop マニュアルをご覧下さい。

本製品に関する情報は以下のルネサス・ウェブサイトをご覧下さい:

日本サイト: http://japan.renesas.com/renesas_starter_kits

グローバルサイト: http://www.renesas.com/renesas_starter_kits

ルネサスのマイクロコントローラに関する総合情報は、以下のウェブサイトより入手可能です:

日本サイト: http://japan.renesas.com

グローバルサイト: http://www.renesas.com

Renesas Starter Kit for R8C/38C
チュートリアルマニュアル
発行日　　2010 年 4 月 1 日　Rev.1.00
発行：　　株式会社ルネサスソリューションズ
　　〒532-0003 大阪市淀川区宮原 4-1-6 アクロス新大阪ビル

Renesas Starter Kit for R8C/38C
チュートリアルマニュアル

株式会社ルネサスソリューションズ
〒532-0003 大阪市淀川区宮原 4-1-6 アクロス新大阪ビル