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

ご注意書き

1. 本資料に記載されている内容は本資料発行時点のものであり、予告なく変更することがあります。当社製品のご購入およびご使用にあたりましては、事前に当社営業窓口で最新の情報をご確認いただきますとともに、当社ホームページなどを通じて公開される情報に常にご注意ください。
2. 本資料に記載された当社製品および技術情報の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、当社は、一切その責任を負いません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
3. 当社製品を改造、改変、複製等しないでください。
4. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器の設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因しお客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
5. 輸出に際しては、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。本資料に記載されている当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途の目的で使用しないでください。また、当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器に使用することができません。
6. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質水準を「標準水準」、「高品質水準」および「特定水準」に分類しております。また、各品質水準は、以下に示す用途に製品が使われることを意図しておりますので、当社製品の品質水準をご確認ください。お客様は、当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途に当社製品を使用することができません。また、お客様は、当社の文書による事前の承諾を得ることなく、意図されていない用途に当社製品を使用することができません。当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途または意図されていない用途に当社製品を使用したことによりお客様または第三者に生じた損害等に関し、当社は、一切その責任を負いません。なお、当社製品のデータ・シート、データ・ブック等の資料で特に品質水準の表示がない場合は、標準水準製品であることを表します。

   標準水準： コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット

   高品質水準： 輸送機器（自動車、電車、船舶等）、交通用信号機器、防災・防犯装置、各種安全装置、生命維持を目的として設計されていない医療機器（厚生労働省定義の管理医療機器に相当）

   特定水準： 航空機器、航空宇宙機器、海底中継機器、原子力制御システム、生命維持のための医療機器（生命維持装置、人体に埋め込み使用するもの、治療行為（患部切り出し等）を行うもの、その他直接人命に影響を与えるもの）（厚生労働省定義の高度管理医療機器に相当）またはシステム等
8. 本資料に記載された当社製品のご使用につき、特に、最大定格、動作電源電圧範囲、放熱特性、実装条件その他諸条件につきましては、当社保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めておりますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害などを生じさせないようお客様の責任において冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、機器またはシステムとしての出荷保証をお願いいたします。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
10. 当社製品の環境適合性等、詳細につきましては製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを固くお断りいたします。
12. 本資料に関する詳細についてのお問い合わせその他お気付きの点等がございましたら当社営業窓口までご照会ください。

注 1. 本資料において使用されている「当社」とは、ルネサスエレクトロニクス株式会社およびルネサスエレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

User's Manual

Renesas Starter Kit for M16C/65

チュートリアルマニュアル
ルネサス 16 ビットシングルチップ マイクロコンピュータ
M16C ファミリ／M16C/60 シリーズ

ルネサス エレクトロニクス
www.renesas.com

Rev.1.00　2009.10

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。

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

CPU Central Processing Unit
(中央処理装置)

HEW High-performance Embedded Workshop
(統合開発環境)

LED Light Emitting Diode
(発光ダイオード)

PC Program Counter
(プログラムカウンタ)

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

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

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

RSK Renesas Starter Kit
(ルネサススタータキット)

LCD Liquid Crystal Display
(液晶ディスプレイ)

ADC Analog to Digital Converter
(A/D コンバータ)

CD Compact Disc
(コンパクトディスク)

E8a
(E8a オンチップデバッギングエミュレータ)

ROM Read Only Memory
(リードオンリーメモリ)

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

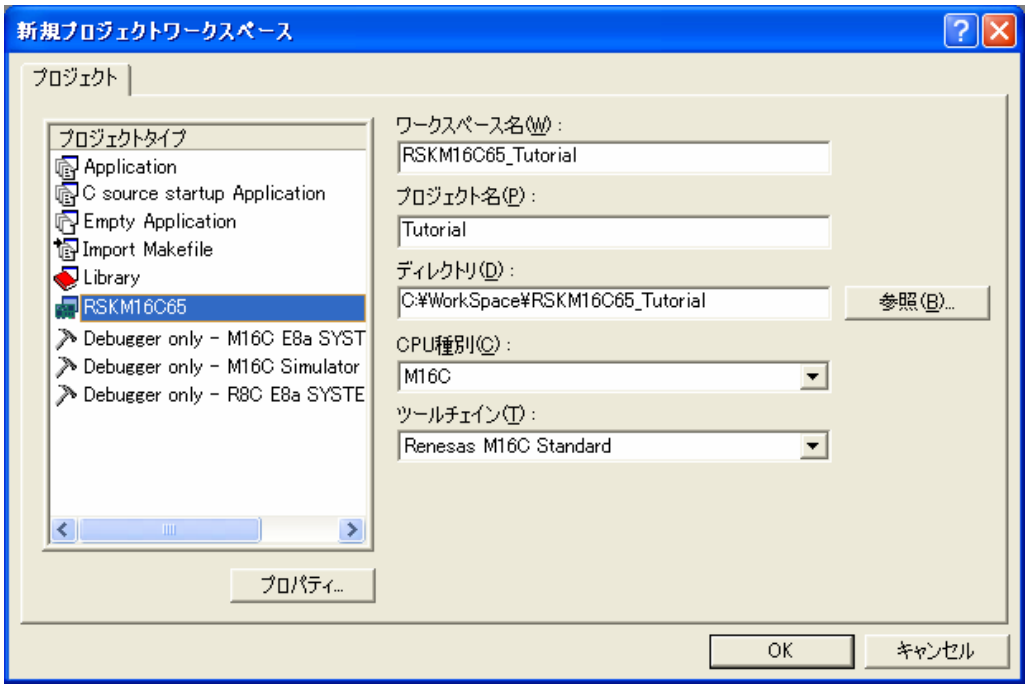
4.1 はじめに

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、全てのルネサス・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 RSKM16C65 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 'RSKM16C65' 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。

- ‘SessionM16C_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

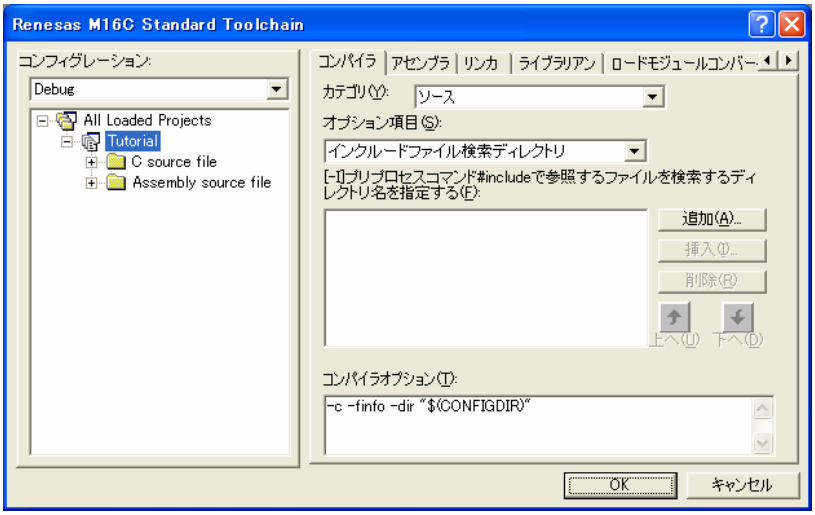
チュートリアル･プロジェクトのビルド設定は、ツールチェインオプションで既に設定されています。ツールチェインオプションを表示する為には、‘ビルド’メニュー項目のツールチェインを選択して下さい。

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 キー”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 キー”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ます。RS232 トランシーバ(U4)の上方にくるよう接続してください。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。

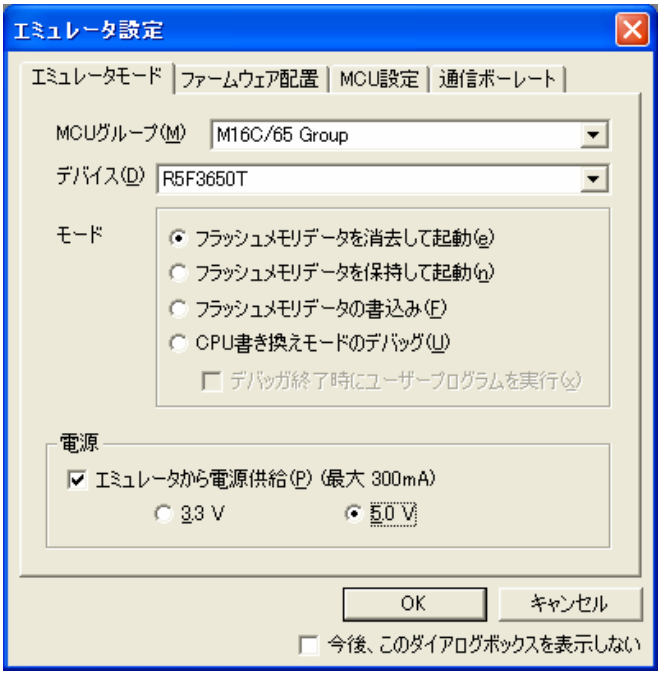
5.3 E8a使用時のターゲットへの接続

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。

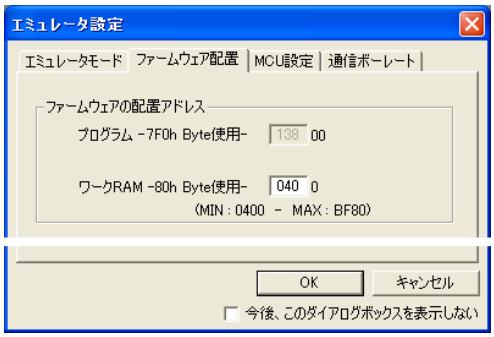
8. ‘SessionM16C_E8a_SYSTEM’ デバッグ・セッションを選択します。
9. デバッグツールバーの<接続>ボタンをクリックします。 

注：初回接続時、”エミュレータモード”ウイザードが表示されます。2 回目以降の接続では、”エミュレータ設定”ダイアログが表示されますので、同じ接続オプションを選択してください。

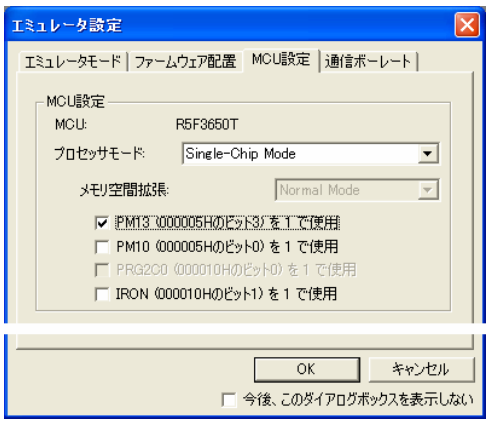
10. “エミュレータ設定”ダイアログが表示されます。適切な MCU グループとデバイスを選択して下さい（RSKM16C65 の場合、M16C/65 Group、R5F3650T を選択）。
11. ‘フラッシュメモリデータを消去して起動’ を選択します。
12. E8a が CPU ボードに電源を供給する場合は、’エミュレータから電源供給’ を選択し、’5.0V’ を選択します。それ以外の場合は、センタープラスの外部電源(5V)をボードに供給して下さい。



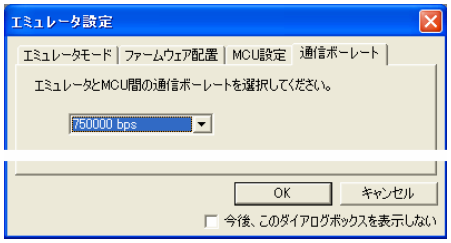
13. ファームウェア配置タブをクリックし、ファームウェアの配置アドレスを設定します。ワーク RAM は 400h を設定してください。



14. MCU 設定タブをクリックし、MCU 設定を行います。プルダウンメニューから 'Single-Chip Mode' を選択して、'PM13(000005H のビット 3)を 1 で使用' をチェックしてください。

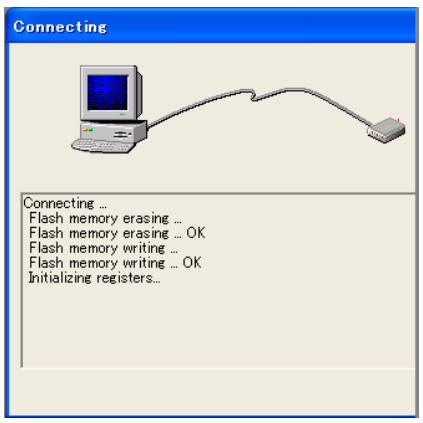


15. 通信ボーレートタブをクリックし、エミュレータと MCU 間の通信ボーレートを設定します。プルダウンメニューから 750000bps(初期設定値)を選択して、<OK>をクリックします。



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。

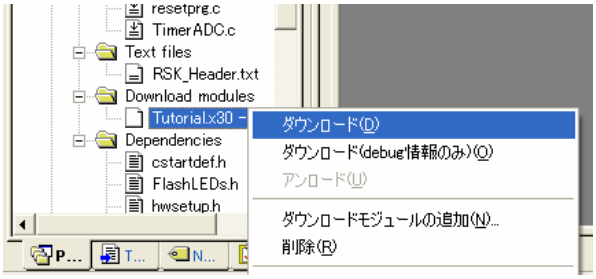
19. 'ファイル' | 'ワークスペースの保存' を選択します。

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。

```
                /***********************************************************************/
                /* Note: The #pragma SECTION directive used below instructs the linker to
                assign a new section name to a section.
                The format of this directive is as follows -
                #pragma SECTION. section name. new section name  */

                #pragma section program interrupt

                /*""FUNC COMMENT""*****************************************************
                * Outline       : start
                * Description   : Reset program
                * Argument      : none
                * Return value  : none
                *""FUNC COMMENT END""*************************************************/

                /* Note: The #pragma entry directive used below specifies the power on reset
                function. Linker uses this information while relocating the code in ROM. */

                #pragma entry start
                void start(void)
80604           {
                    /* Set the interrupt stack pointer */
80604               _isp_    = &_istack_top;

                    /* Enable writing to register prm0 */
80608               prcr     = 0xA502;

                    /* Configure the processor mode register
                    b1:b0    - PM01:PM00 - 0 Single chip mode selected
```

ここでは、初期化コードの説明を省略して、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、次の方法で設定します。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 キー”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’を選択する。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できます。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くなります。

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。 

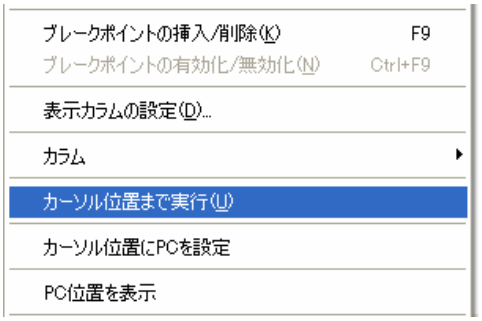
‘main.c’が開き、プログラムカウンタは新しい位置を示します。

```
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();
```

チュートリアルコードは LCD 表示のためのコードを含んでいます。ここでは LCD インタフェースの処理について説明しません。詳細については lcd.c/lcd.h を参照してください。

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 SW1 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 A/D 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、A/D 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_a0(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 キー”を押して、現在の PC 位置からコードを実行します。 

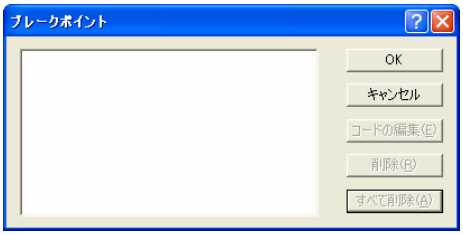
コードは割り込みルーチンで停止します。ここで、割り込み関数をステップして確認できます。

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 キー＋B”を押してブレークポイント画面を開きます。
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(次項)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    : Reset program
* Argument       : none
* Return value   : none
*""FUNC COMMENT END""*************************************************************/

/* Note: The #pragma entry directive used below specifies the power on reset
function. Linker uses this information while relocating the code in ROM. */

#pragma entry start
void start(void)
{
    /* Set the interrupt stack pointer */
    _isp_   = &_istack_top;

    /* Enable writing to register pm0 */
    prcr    = 0xA502U;

    /* Configure the processor mode register
    b1:b0   - PM01:PM00 - 0 Single chip mode selected
    b2      - PM02      - 0 Unused. Set to 0
    b3      - PM03      - 0 CPU reset bit. Set to 0.
    b5:b4   - PM05:PM04 - 0 Unused. Set to 00
    b6      - PM06      - 0 Address output. Set to 0.
    b7      - PM07      - 0 (BCLK clock output Enable)  */

    pm0 = 0x00U;

    /* Disable writing to register pm0 */
    prcr    = 0xA500U;

    /* Set flag register */
    _flg_   = __F_value__;

#if __STACKSIZE__!=0
     /* Set user stack pointer */
    _sp_    = &_stack_top;
#endif

    /* Initialize static base register with the start address of RAM.
       Note: Please do not change this value */
    _sb_ = 0x400U;

    /* Set variable vector's address */
    _asm("  ldc #((topof vector)>>16)&0FFFFh,INTBH");
    _asm("  ldc #(topof vector)&0FFFFh,INTBL");

    /* Initlalize the data sections */
    initsct();

#if __HEAPSIZE__ != 0
    /* Initializes heap  */
    heap_init();
#endif

#if __STANDARD_IO__ != 0
    /* Initialize standard I/O */
    _init();
#endif

    /* Initialize FB registe for debugger */
    _fb_ = 0U;

    /* Call the hardware setup function.    */
    HardwareSetup();

    /* Call main application */
    main();

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

スタックポインタの初期化、ヒープ・データセクションの初期化は `start()` 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 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 the required 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
* 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 key is pressed. */
    FlashLEDs();

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

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

    /* End of the user program. 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 M16C/65
チュートリアルマニュアル
発行日　2009 年 10 月 1 日　Rev.1.00
発行：　Renesas Technology Europe Ltd.
Duke’s Meadow, Millboard Road, Bourne End
Buckinghamshire SL8 5FH, United Kingdom

Renesas Starter Kit for M16C/65
チュートリアルマニュアル

ルネサス エレクトロニクス株式会社
神奈川県川崎市中原区下沼部1753 〒211-8668

RJG10J0071-0100