User's Manual

32

RX62T グループ

Renesas Starter Kit チュートリアルマニュアル

ルネサス 32 ビットマイクロコンピュータ
RX ファミリ
RX600 シリーズ

ルネサス エレクトロニクス
www.renesas.com

Rev.2.01 2011.7

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。

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本文を参照してください。なお、本マニュアルの本文と異なる記載がある場合は、本文の記載が優先するものとします。

1. 未使用端子の処理
   【注意】未使用端子は、本文の「未使用端子の処理」に従って処理してください。
   CMOS 製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI 周辺のノイズが印加され、LSI 内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。未使用端子は、本文「未使用端子の処理」で説明する指示に従い処理してください。
2. 電源投入時の処置
   【注意】電源投入時は，製品の状態は不定です。
   電源投入時には、LSI の内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。
   同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。
3. リザーブアドレスのアクセス禁止
   【注意】リザーブアドレスのアクセスを禁止します。
   アドレス領域には、将来の機能拡張用に割り付けられているリザーブアドレスがあります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。
4. クロックについて
   【注意】リセット時は、クロックが安定した後、リセットを解除してください。
   プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。
   リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。
5. 製品間の相違について
   【注意】型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。
   同じグループのマイコンでも型名が違うと、内部 ROM、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ輻射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

このマニュアルの使い方

1. 目的と対象者

このマニュアルは、RSK ハードウェア概要と電気的特性をユーザに理解していただくためのマニュアルです。様々な周辺装置を使用して、RSK プラットフォーム上のサンプルコードを設計するユーザを対象にしています。

このマニュアルは、RSK 製品の機能概観を含みますが、組み込みプログラミングまたはハードウェア設計ガイドのためのマニュアルではありません。また、RSK および開発環境のセットアップに関するその他の詳細は、チュートリアルに記載しています。

このマニュアルを使用する場合、注意事項を十分確認の上、使用してください。注意事項は、各章の本文中、各章の最後、注意事項の章に記載しています。

改訂記録は旧版の記載内容に対して訂正または追加した主な箇所をまとめたものです。改訂内容すべてを記録したものではありません。詳細は、このマニュアルの本文でご確認ください。

RSKRX62T では次のドキュメントを用意しています。ドキュメントは最新版を使用してください。最新版はルネサスエレクトロニクスのホームページに掲載されています。

ドキュメントの種類	記載内容	資料名	資料番号
ユーザーズマニュアル	RSK ハードウェア仕様の説明	RSKRX62T ユーザーズマニュアル	RJJ10J2788
ソフトウェアマニュアル	Renesas Peripheral Driver Library（RPDL）を備えたサンプルコードの機能とその相互作用の説明	RSKRX62T ソフトウェアマニュアル	RJJ10J2791
チュートリアル	RSK および開発環境のセットアップ方法とデバッギング方法の説明	RSKRX62T チュートリアル	RJJ10J1289 (本マニュアル)
クイックスタートガイド	A4 紙一枚の簡単なセットアップガイド	RSKRX62T クイックスタートガイド	RJJ10J2790
回路図	CPU ボードの回路図	RSKRX62T CPU ボード回路図	RJJ99J0072
ユーザーズマニュアル ハードウェア編	ハードウェアの仕様（ピン配置、メモリマップ、周辺機能の仕様、電気的特性、タイミング）と動作説明	RX62T グループ ユーザーズマニュアル ハードウェア編	R01UH0034JJ

2. 略語および略称の説明

略語／略称	英語名	備考
ADC	Analogue to Digital Converter	A/D コンバータ
API	Application Programming Interface	アプリケーションプログラムインタフェース
CD	Compact Disk	コンパクトディスク
CPU	Central Processing Unit	中央処理装置
E1	Renesas On-chip Debugging Emulator	ルネサスオンチップデバッギングエミュレータ
E20	Renesas On-chip Debugging Emulator	ルネサスオンチップデバッギングエミュレータ
HEW	High-performance Embedded Workshop	ルネサス統合開発環境
LCD	Liquid Crystal Display	液晶ディスプレイ
LED	Light Emitting Diode	発光ダイオード
ROM	Read-Only Memory	リードオンリーメモリ
RPDL	Renesas Peripheral Driver Library	周辺 I/O ドライバライブラリ
RSK	Renesas Starter Kit	ルネサススタータキット
USB	Universal Serial Bus	-

目次

1. 概要

1.1 目的

本 RSK はルネサスマイクロコントローラ用の評価ツールです。本マニュアルは、コードのダウンロードや基本的なデバッグ操作について説明しています。

1.2 特徴

本 RSK は以下の特徴を含みます：

- ルネサスマイクロコントローラのプログラミング
- ユーザコードのデバッギング
- スイッチ、LED、ポテンショメータ等のユーザ回路
- サンプルアプリケーション
- 周辺機能初期化コードのサンプル

CPU ボードはマイクロコントローラの動作に必要な回路を全て備えています。

2. はじめに

本マニュアルは Renesas Starter Kit（RSK）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。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。

- RSK で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？
- 組み込み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。

本マニュアルで引用された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を使用してインストールされます。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。

チュートリアルは RSK の使用方法の説明を目的とするものであり、High-performance Embedded Workshop、コンパイラツールチェインまたは E1 エミュレータの入門書ではありません。これらに関する詳細情報は各関連マニュアルを参照してください。

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の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のメニューを使用し、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

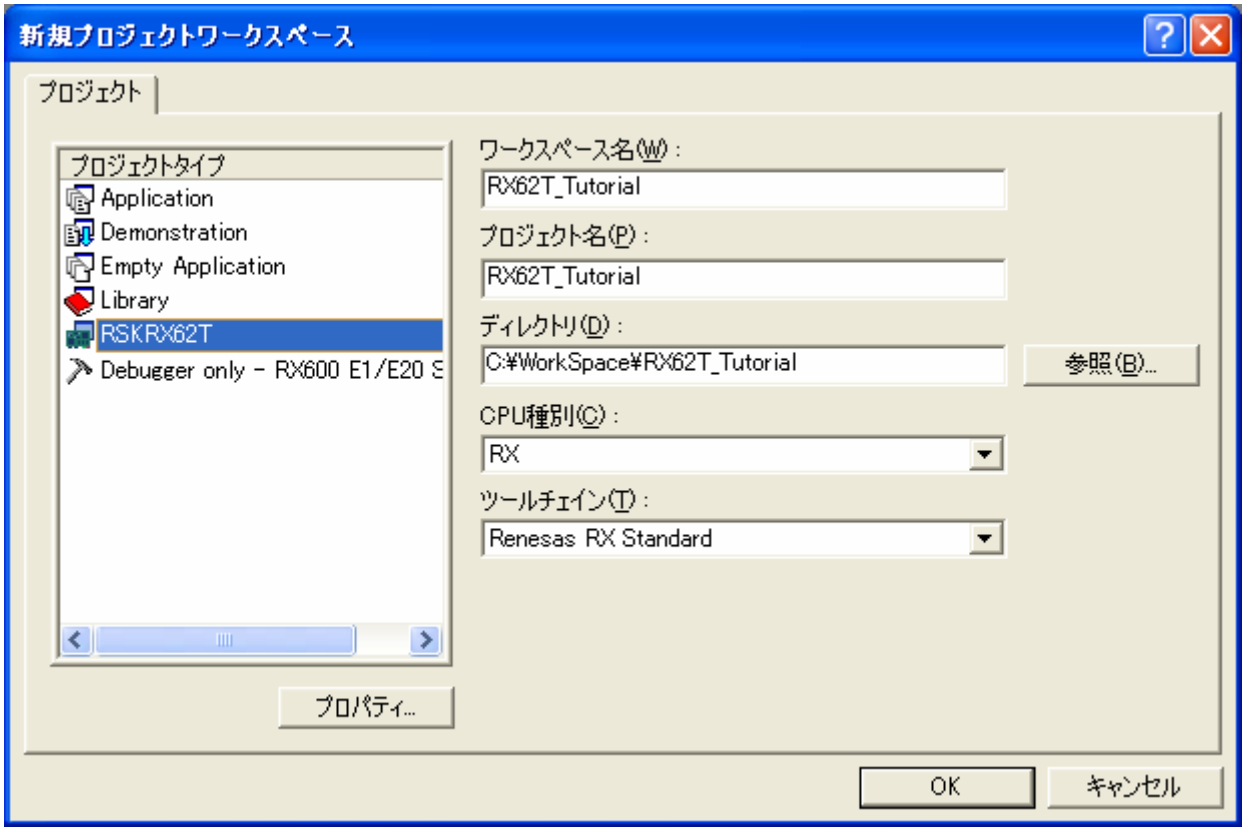
4.1 はじめに

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、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 HEWの開始とE1 エミュレータの接続

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。



上図は RSKRX62T 選択時の新規プロジェクトワークスペースの一例です。

- ‘RX‘CPU 種別および‘Renesas RX Standard’ツールチェインを選択します。
- プロジェクトリストから‘RSKRX62T’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。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。

- プロジェクトとして'Tutorial'を選択し、<Next>をクリックします。
- <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の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。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

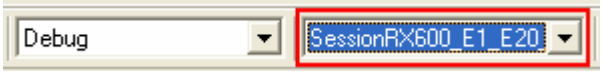
ツールバーの左側のドロップダウンリストからビルドコンフィグレーションを選択します。利用可能なオプションは、Debug と Release です。Debug ビルドは、デバッガとの使用に設定されています。Release ビルドは、最終 ROM コード用の設定です。
これら 2 種のビルドの違いとして、最適化設定が挙げられます。最適化が有効の場合、デバッガがコードを予想外の順序で実行するようなケースがあり、デバッグをスムーズに処理する為には、デバッグされるコードの最適化を無効にすることを推奨します。

- '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。

- 'SessionRX600_E1_E20_SYSTEM'デバッグセッションを選択します。

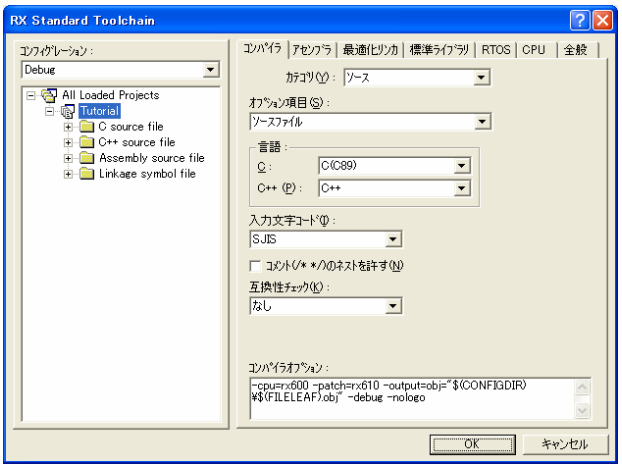


5. チュートリアルプログラムのビルド

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。

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グ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。

- 各タブの'カテゴリ'ドロップダウンリストをチェックして、利用可能なオプションを確認して下さい。ここでは、オプションの変更は不要です。
- 選択終了後に<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。

- ツールバーの'すべてをビルド'ボタンです。プロジェクト中の全ファイルをビルドします。標準ライブラリは一度だけビルドされます。

- ツールバーの'ビルド'ボタンです。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、標準ライブラリはビルドされません。

- キーボードの'F7'ボタンです。上記の'ビルド'ボタン選択の場合と同じです。

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 エミュレータの接続

本チュートリアルでは、外部から CPU ボードに電源を供給する必要はありません。電源は USB ポートから取得されます。その USB ポートに多くのデバイスが接続している場合、Windows がシャットダウンするかもしれないので注意してください。この問題が発生した場合、一部のデバイスを削除して、もう一度やり直してください。外部電源を供給する際、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。

このキットに同梱の電圧堅守回路（LVD）サンプルコードは電圧検出のために可変電源を必要とします。その場合には、外部電源を使用する必要があります。詳細は RSKRX62T ユーザーズマニュアルを参照してください。

E1 のホストコンピュータへの接続方法は、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、E1 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。

- LCD モジュールを CPU ボードの LCD コネクタに取り付け、コネクタの全てのピンが正しく接続されていることを確認して下さい。
- E1 をご使用のコンピュータの USB ポートに接続して下さい。
- E1 を CPU ボードに接続します。’E1’のシルク印字のある E1 コネクタに接続して下さい。
- 外部電源を CPU ボードに供給します。’PWR’のシルク印字のある PWR コネクタに接続して下さい。

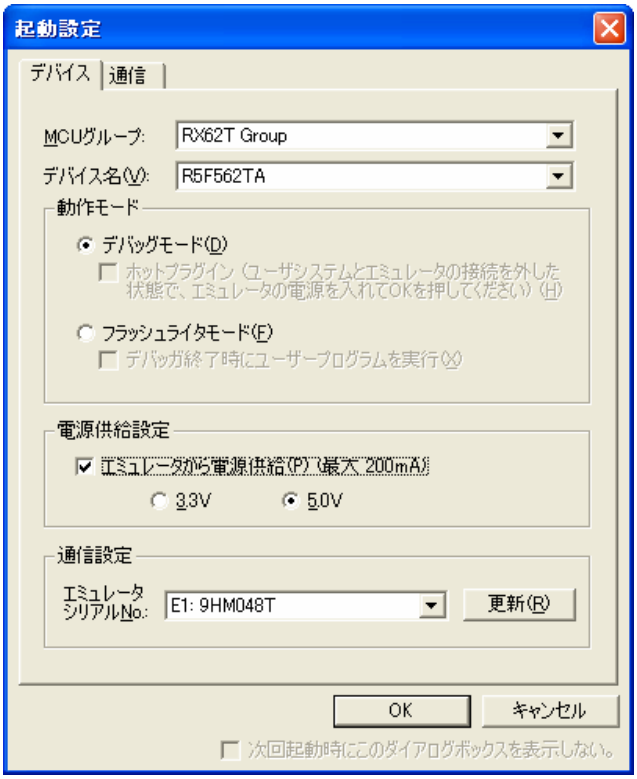
5.3 E1 によるターゲットの接続

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。

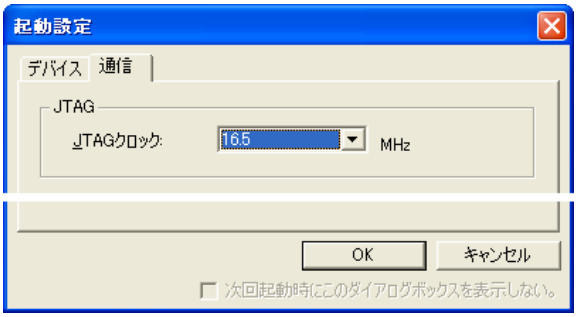
初回接続時と 2 回目以降の接続時とでダイアログ表示が異なります。一部の接続オプションは初回接続時に設定した内容が有効になっていますので、変更の必要がない限り、同じ接続オプションを選択して下さい。

- ‘SessionRX600_E1_E20_SYSTEM’デバッグセッションを選択します。
- デバッグツールバーの<接続>ボタンをクリックします。

- ‘起動設定’ダイアログが表示されます。以下の通り設定します:
  - MCU グループ: RX62T Group
  - デバイス名: R5F562TA
  - 動作モード: デバッグモード
- E1 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 の'を選択します。それ以外の場合、適切な電源を接続してください。(詳細については、RSKRX62T のユーザーズマニュアルを参照してください。)

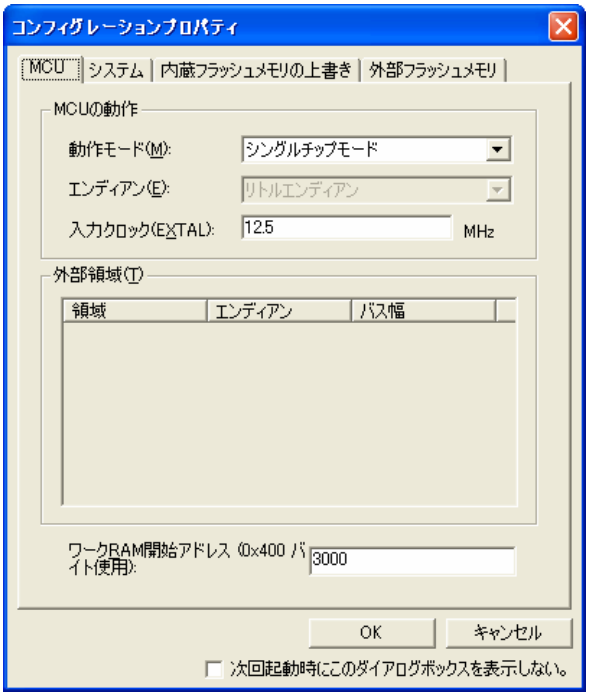


- 通信タブをクリックします。JTAG クロックが 16.5MHz に設定されていることを確認し、<OK>をクリックして下さい。
- CPU ボードとの接続を開始します。



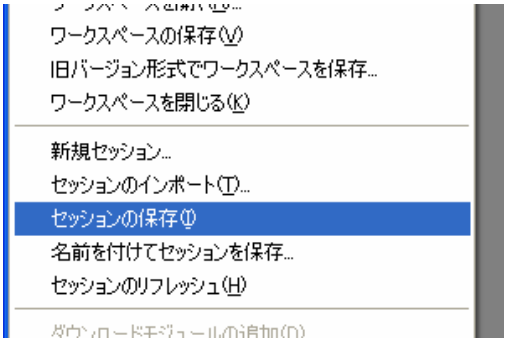
- 接続中にプロセスのステータスを示すダイアログが表示されます。初期設定では、接続処理が終了したらダイアログが閉じるように設定されています。

- 接続処理が終了したら’コンフィグレーションプロパティ’ダイアログが表示されます。
- 以下の通り設定します:
  - 動作モード: シングルチップモード
  - エンディアン: リトルエンディアン
  - 入力クロック: 12.5MHz
  - ワーク RAM 開始アドレス: 3000
- <OK>をクリックして下さい。High-performance Embedded Workshop のアウトプット画面に’Connected’と表示されます。
- ツールバー上のデバッグボタンがアクティブになります。これらのボタンの機能は本マニュアル中で説明されます。



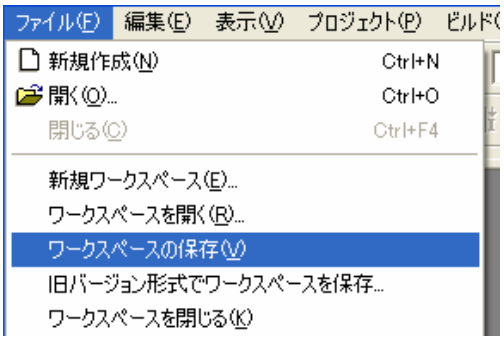
ここで、High-performance Embedded Workshop のセッションを保存することを推奨します。

- 'ファイル'|'セッションの保存'を選択します。



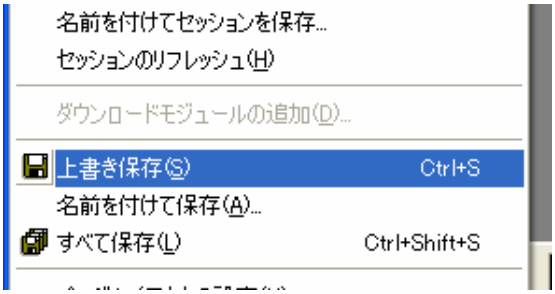
ワークスペースの設定を変更した場合、ワークスペースを保存することを推奨します。

- 'ファイル' | 'ワークスペースの保存'を選択します。



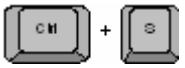
ファイルを変更した場合、次の操作でファイルを保存することができます。

- 'ファイル'|'上書き保存'を選択します。



ツールバーの'上書き保存'ボタンまたは'すべて保存'ボタンでも保存することができます。

また、キーボートからも保存することができます。

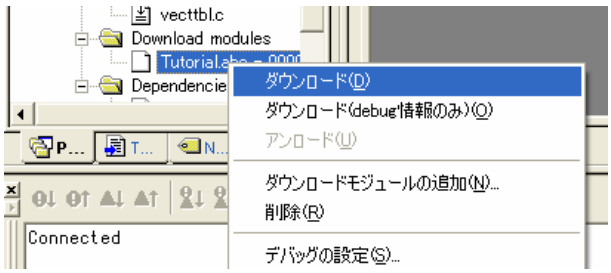


6. チュートリアルのダウンロードと実行

6.1 プログラムコードのダウンロード

High-performance Embedded Workshop でのコード作成が完了したら、それを CPU ボード上のマイクロコントローラ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。

- ダウンロードモジュールリストから関連する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、コードの実行およびデバッグ準備が整います。



6.2 コードの実行

プログラムが CPU ボード上のマイクロコントローラにダウンロードされると、プログラムを実行することができます。次章に移る前に、'リセット後実行'をクリックし、プログラムを実行させてみてください。

7. チュートリアルレビュー

本章では、チュートリアルコードがどのように動作し、より複雑なコードへ実装されるためにどのようにそれを変更することができるかを確かめます。

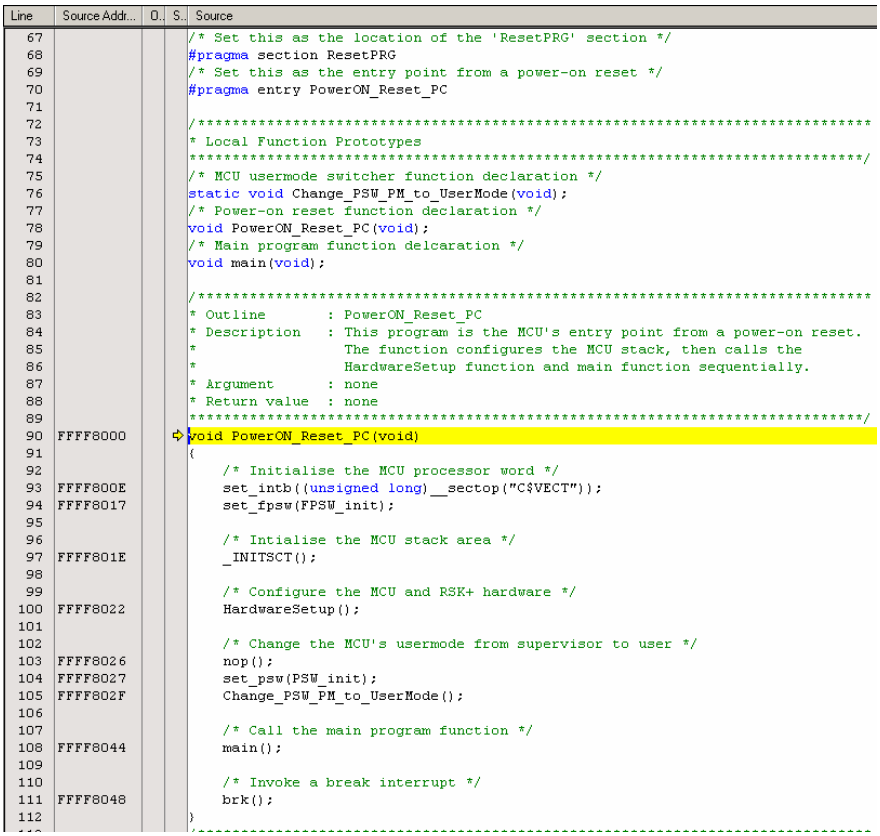
7.1 プログラム初期化

メインプログラムが実行される前に、マイクロコントローラは初期化されます。チュートリアルコードの以下の部分は、主要機能が正確に実行できるように、CPU ボード上のマイクロコントローラを初期化するために使用されます。マイクロコントローラはリセットスイッチまたはパワーオンリセットによってリセットされるごとに、初期化コードが実行されます。

チュートリアルコードがマイクロコントローラにダウンロードされていることを確認し、デバッグツールバーの'CPU リセット'をクリックして下さい。

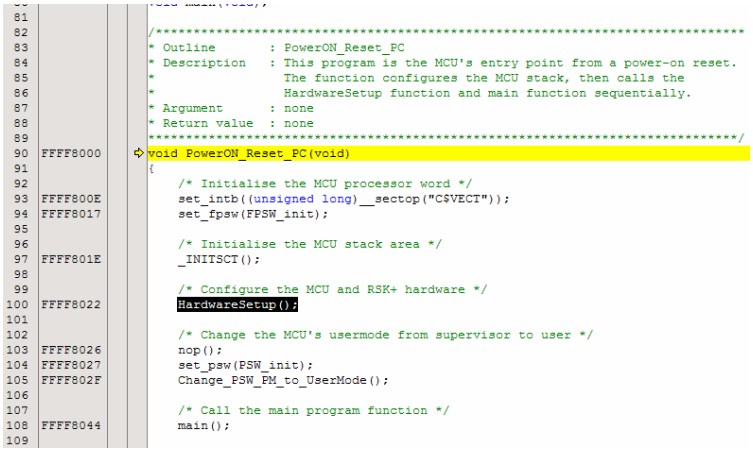
- チュートリアルコードの開始位置でファイルが開きます。矢印と黄色のハイライトは現在のプログラムカウンタ位置を示します。
- コード表示を下のボタンで'ソースモード'、'逆アセンブリモード'、'混合モード'に切り替えることができます。

コード表示を'ソースモード'に設定して下さい。



```
Line  Source Addr...  O..  S..  Source
  67                            /* Set this as the location of the 'ResetPRG' section */
  68                            #pragma section ResetPRG
  69                            /* Set this as the entry point from a power-on reset */
  70                            #pragma entry PowerON_Reset_PC
  71
  72                            /*****************************************************************************
  73                            * Local Function Prototypes
  74                            *****************************************************************************/
  75                            /* MCU usermode switcher function declaration */
  76                            static void Change_PSW_PM_to_UserMode(void);
  77                            /* Power-on reset function declaration */
  78                            void PowerON_Reset_PC(void);
  79                            /* Main program function delcaration */
  80                            void main(void);
  81
  82                            /*****************************************************************************
  83                            * Outline       : PowerON_Reset_PC
  84                            * Description   : This program is the MCU's entry point from a power-on reset.
  85                            *                 The function configures the MCU stack, then calls the
  86                            *                 HardwareSetup function and main function sequentially.
  87                            * Argument      : none
  88                            * Return value  : none
  89                            *****************************************************************************/
  90  FFFF8000                  void PowerON_Reset_PC(void)
  91                            {
  92                                /* Initialise the MCU processor word */
  93  FFFF800E                      set_intb((unsigned long)__sectop("C$VECT"));
  94  FFFF8017                      set_fpsw(FPSW_init);
  95
  96                                /* Intialise the MCU stack area */
  97  FFFF801E                      _INITSCT();
  98
  99                                /* Configure the MCU and RSK+ hardware */
 100  FFFF8022                      HardwareSetup();
 101
 102                                /* Change the MCU's usermode from supervisor to user */
 103  FFFF8026                      nop();
 104  FFFF8027                      set_psw(PSW_init);
 105  FFFF802F                      Change_PSW_PM_to_UserMode();
 106
 107                                /* Call the main program function */
 108  FFFF8044                      main();
 109
 110                                /* Invoke a break interrupt */
 111  FFFF8048                      brk();
 112                            }
```

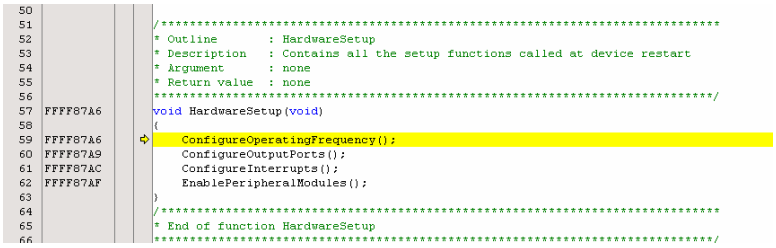
- ‘HardwareSetup()’を左クリックで選択して下さい。



- デバッグツールバーの‘カーソル位置まで実行’をクリックして、選択した行までプログラムを実行させます。

- 実行後、‘ステップイン’をクリックして、HardwareSetup 関数にエントリします。

- プログラムカウンタは HardwareSetup 関数に移ります。この関数はマイクロコントローラが正確にセットアップされるための関数を持ち、メインプログラムが実行される前に実行されます。



- ‘ステップイン’をクリックして ConfigureOperatingFrequency 関数にエントリします。

- ConfigureOperatingFrequency 関数はシステムクロックの設定に使用されます。
- 次に、HardwareSetup 関数をスキップしてメインプログラムコードを見てみましょう。

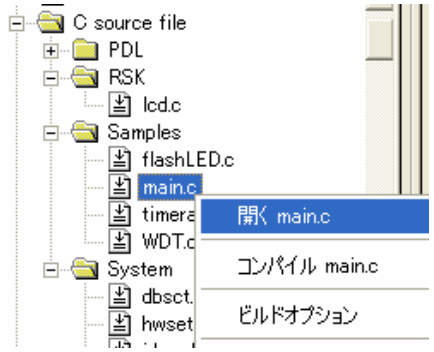
```
/***********************************************************************
* Outline       : ConfigureOperatingFrequency
* Description   : Configures the clock settings for each of the device clocks
* Argument      : none
* Return value  : none
***********************************************************************/
void ConfigureOperatingFrequency(void)
{
    /* Declare error flag */
    bool err = true;

    /* Modify the MCU clocks */
    err &=  R_CGC_Set
            (
                12E6,
                96E6,
                48E6,
                24E6,
                PDL_NO_DATA
            );
    /*
    Clock Description              Frequency
    ----------------------------------------
    Input Clock Frequency.............12MHz
    Internal Clock Frequency..........96MHz
    Peripheral Clock Frequency.........48MHz
    External Bus Clock Frequency.......24MHz */

    /* Halt in while loop when RPDL errors detected */
    while(!err);
}
/***********************************************************************
* End of function ConfigureOperatingFrequency
***********************************************************************/
```

ハードウェア設定に関する詳細情報は、RSKRX62T ユーザーズマニュアルおよび RX62T グループユーザーズマニュアルハードウェア編を参照して下さい。

7.2 メイン関数

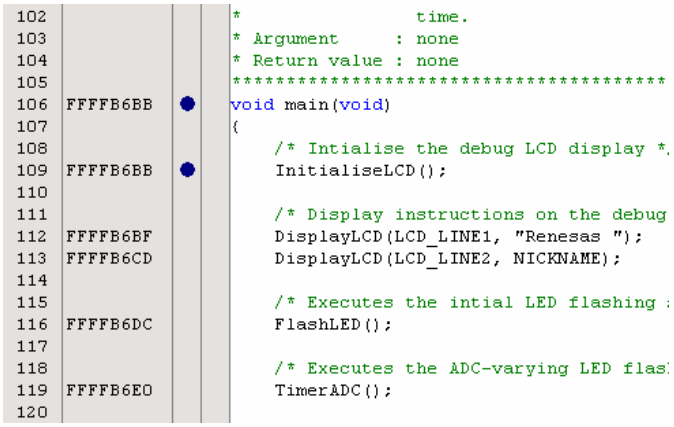
このセクションでは、メイン関数がコールされたプログラムコードがどのように動作するかを見ます。

- 画面左にリストアップされている’main.c’を右クリックして、’開く main.c’を選択して下さい。



- ‘main()’に該当するオンチップブレークポイント行をダブルクリックし、ブレークポイントを設定します。

2 つのブレークポイントが現れますが、これらが同じソースコードアドレスであることを意味します。



E1 エミュレータは本マニュアルでは説明していない高度なイベント機能やブレーク機能を持っています。E1 エミュレータの詳細情報は、RX ファミリ用 E1/E20 エミュレータユーザーズマニュアルを参照して下さい。

- デバッグツールバーの‘リセット後実行’をクリックして下さい。

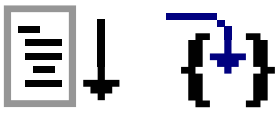
- コードは設定したブレークポイントまで実行されます。この時点で、マイクロコントローラの初期化は完了しています。’main.c’が開き、プログラムカウンタは新しい位置を示します。
- チュートリアルコードは LCD 表示をサポートしています。CPU ボードに LCD モジュールを取り付けておけば、LCD に文字を表示させることができます。なお、LCD インタフェースは常に書き込みモードになるようにボード上で設定されています。

```
74                  /*****************************************************************************
75                  * Outline       : main
76                  * Description   : The main program function. Displays the Renesas splash screen
77                  *                 onto the LCD display, then calls the flashLED and TimerADC
78                  *                 functions. The function then calls the statics test routine,
79                  *                 before waiting in an infinite while loop.
80                  * Argument      : none
81                  * Return value  : none
82                  *****************************************************************************/
83  FFFF1C4 ●       void main(void)
84                  {
85                      /* Initialise the LCD Display */
86  FFFF1C8 ●       ➪   InitialiseLCD();
87
88                      /* Displays the Renesas splash screen */
89  FFFF1CC             DisplayLCD(LCD_LINE1, "Renesas");
90  FFFF1D0             DisplayLCD(LCD_LINE2, NICKNAME);
91
92                      /* Begins the initial LED flash sequence */
93  FFFF1D4             FlashLED();
94
95                      /* Begins the ADC-varying flash Sequence */
96  FFFF1D8             TimerADC();
97
98                      /* Begins the static variable test */
99  FFFF1DC             Statics_Test();
100
101                     /* Defines an infinite loop to keep the MCU running */
102 FFFF1E0             while(1);
103                 }
104                 /*****************************************************************************
105                 End of main
106                 *****************************************************************************/
```

- FlashLED 関数、TimerADC 関数および Statics_Test 関数にブレークポイントを設定して下さい。

```
117
118                     /* Executes the initial LED flashing sequence */
119 FFFFC204 ●        FlashLED();
120
121                     /* Executes the ADC-varying LED flash sequence */
122 FFFFC208 ●        TimerADC();
123
124                     /* Executes the static variable test */
125 FFFFC20C ●        Statics_Test();
126
```

- デバッグツールバーの‘Go’をクリックし、プログラム停止後に‘ステップイン’をクリックして下さい。プログラムカウンタは FlashLED 関数の先頭に移ります。



- FlashLED 関数は周期的な CMT コールバック（一定周期で LED をトグル出力する）を生成する RPDL 関数を使用します。
- if 文の gFlashCount 変数は LED 点滅をカウントダウンする値です。この値が 0 になると CMT タイマを終わらせ関数を抜けます。

```
68                  /*****************************************************************************
69                  * Outline       : FlashLED
70                  * Description   : The flash function used at the beginning of the program
71                  * Argument      : none
72                  * Return value  : none
73                  *****************************************************************************/
74  FFFF1       ➪   void FlashLED (void)
75                  {
76                      /* Declare error flag */
77  FFFF1           bool err = true;
78
79                      /* Configure compare match timer */
80  FFFF1           err &= R_CMT_Create
81                              (
82                                  0,
83                                  PDL_CMT_PCLK_DIV_512,
84                                  0xFE0,
85                                  CMT_Callback,
86                                  3
87                              );
88
89                      /* While loop keeps the function waiting */
90  FFFF1           while(1)
91                  {
92                      /* Checks if the flash count has been reached,
93                         or if a button has been pressed */
94  FFFF1               if((gSwitchFlag)||(gFlashCount > 0xC8))
95                      {
96                          /* Reset the gSwitchFlag flag variable */
97  FFFF1                   gSwitchFlag = 0;
98
99                          /* Exit from the while loop */
100                         break;
101                     }
102                 }
103
104                     /* Destroy Timer */
105 FFFF1           err &= R_CMT_Destroy
106                             (
107                                 0
108                             );
109
110
111                     /* Halt in while loop when RPDL errors detected */
112 FFFF1           while(!err);
```

- ‘Go’をクリックしてプログラムを再開し、ボード上の任意のスイッチを押して下さい。プログラムは TimerADC 関数上のブレークポイントで停止します。
- ‘ステップイン’をクリックして TimerADC 関数にエントリします

- TimerADC 関数はボード上のポテンショメータ RV1 による LED 点滅間隔の調整を可能にするために、タイマと A/D コンバータの両方を設定します。

```
66                  /*****************************************************************************
67                  * Outline       : TimerADC
68                  * Description   : Function intialises the Timer and ADC peripherals needed to
69                  *                 vary the LED flash rate based on the AD pot.
70                  * Argument      : none
71                  * Return value  : none
72                  *****************************************************************************/
73  FFFF21(         void TimerADC(void)
74                  {
75                      /* Declare error flag */
76  FFFF21(             bool err = true;
77
78                      /* Call the Timer start function */
79  FFFF21(             StartTimer();
80
81                      /* Call the ADC start function */
82  FFFF21'             StartADC();
83
84                      /* Disable switch SW1 interrupts */
85  FFFF21'             err &= R_INTC_ControlExtInterrupt(
86  FFFF21)                                               PDL_INTC_IRQ0,
87  FFFF21]                                               PDL_INTC_DISABLE
88                                                      );
89
90  FFFF21]             /* Disable switch SW2 interrupts */
91                      err &= R_INTC_ControlExtInterrupt(
92                                                        PDL_INTC_IRQ1,
93                                                        PDL_INTC_DISABLE
94                                                      );
95
96                      /* Disable switch SW3 interrupts */
97                      err &= R_INTC_ControlExtInterrupt(
98                                                        PDL_INTC_IRQ3,
99                                                        PDL_INTC_DISABLE
100                                                     );
101
102 FFFF21]             /* Halt in while loop when RPDL errors detected */
103                     while(!err);
104                 }
105 FFFF21]         /*****************************************************************************
106                 End of TimerADC
107                 *****************************************************************************/
```

- ‘F5’を押してプログラムを再開させます。プログラムは Statics_Test 関数上のブレークポイントで停止します。

- ‘F11’を押して関数の先頭へ移ります。

- Statics_Test 関数は静的変数の文字ストリングを初期化します。初期化後、別の文字ストリングに置き換えます。
- ‘実行’をクリックするか’F5’を押してプログラムを再開させます。LCD の 2 行目の文字が「STATIC」から一文字ごとに「TESTTEST」に置き換わることが確認できます。その後、プログラムは 2 行目の文字を「RX62T」に戻します。

```
114 FFFF1C(         void Statics_Test(void)
115                 {
116                      /* At this point please right click on the 'ucSTR' variable and select
117                         'Instant Watch'. A dialog will be displayed showing the current value
118                         of the variable. Select 'Add' in the dialog and a new 'Watch Window'
119                         will open. Step through the following code to see that the initialised
120                         data is being overwritten with the different data. */
121
122                     /* Initialise delay and count variables */
123                     uint16_t uiCount, delayA, delayB;
124
125                     /* Write ucStr varaible, "STATIC" to LCD */
126 FFFF1C(             DisplayLCD(LCD_LINE2,ucStr);
127
128                     /* Begin for loop which writes one letter of ucReplace to the LCD at a time
129                        The nested while loops generate the delay bewteen each letter change */
130 FFFF1C9             for (uiCount=0; uiCount<8; uiCount++)
131                     {
132
133                         /* Replace letter number uiCount of ucStr from ucReplace */
134 FFFF1C.             ucStr[uiCount] = ucReplace[uiCount];
135 FFFF1C]             DisplayLCD(LCD_LINE2,ucStr);
136
137                         /* Set delay variables */
138 FFFF1C(             delayA = 0xAA;
139 FFFF1C]             delayB = 0xFFFF;
140
141                         /* Nested while delay loop */
142 FFFF1C]             while(delayA)
143                         {
144 FFFF1C]                 while(delayB)
145                             {
146 FFFF1C]                     delayB--;
147                             }
148 FFFF1D(                 delayB = 0xFFFF;
149 FFFF1D(                 delayA--;
150                         }
151                     }
152
153                     /* Clear LCD Display */
154 FFFF1D:             ucStr[uiCount] = '\0';
155
156                     /* Write MCU nickname to LCD again */
157 FFFF1D-             DisplayLCD(LCD_LINE2,NICKNAME);
158                 }
```

- ここまででチュートリアルコードの全体の動作が終わりました。チュートリアルコード中で使用される RPLD 関数についての詳細は、Renesas Pheripheral Driver Library User’s Manual を参照して下さい。

8. 追加情報

サポート

High-performance Embedded Workshop の詳細情報は、CD またはウェブサイトに掲載のマニュアルを参照してください。

RX62T マイクロコントローラに関する詳細情報は、RX62T グループユーザーズマニュアルハードウェア編を参照してください。

アセンブリ言語に関する詳細情報は、RX ファミリユーザーズマニュアルソフトウェア編を参照してください。

オンラインの技術サポート、情報等は以下のウェブサイトより入手可能です：

http://japan.renesas.com/rskrx62t （日本サイト）
http://www.renesas.com/rskrx62t （グローバルサイト）

オンライン技術サポート

技術関連の問合せは、以下を通じてお願いいたします。

アメリカ： techsupport.america@renesas.com
ヨーロッパ： tools.support.eu@renesas.com
日本： csc@renesas.com

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：

http://japan.renesas.com/ （日本サイト）
http://www.renesas.com/ （グローバルサイト）

商標

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

著作権

改訂記録	RSKRX62T チュートリアルマニュアル

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2010.12.03	ー	初版発行
2.00	2011.05.26	13	「5.3 E1 によるターゲットの接続」のコンフィグレーションプロパティ画面を修正。（エンディアン：ビッグからリトルに修正）
2.01	2011.07.01	ー	社名修正

RSKRX62T チュートリアルマニュアル

発行年月日　2011 年 7 月 1 日　Rev.2.01

発行　　株式会社ルネサスソリューションズ
　　〒532-0003　大阪府大阪市淀川区宮原 4-1-6

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口

http://www.renesas.com

※営業お問合せ窓口の住所・電話番号は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス販売株式会社　〒100-0004　千代田区大手町2-6-2（日本ビル）　(03)5201-5307

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：http://japan.renesas.com/inquiry

RX62T グループ

RJJ10J2789-0201

ルネサス エレクトロニクス株式会社