

RX610 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

本資料に記載の全ての情報は本資料発行時点のものであり、ルネサス エレクトロニクスは、予告なしに、本資料に記載した製品または仕様を変更することがあります。
ルネサス エレクトロニクスのホームページなどにより公開される最新情報をご確認ください。

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のですが、誤りが無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、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と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っていると、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。

RSKRX610 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	RSKRX610 ユーザーズマニュアル	R20UT0423JG
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	RSKRX610 ソフトウェアマニュアル	R20UT0428JG
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グ方法の説明	RSKRX610 チュートリアル	R20UT0424JG (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	RSKRX610 クイックスタートガイド	R20UT0425JG
回路図	CPU ボードの回路図	RSKRX610 CPU ボード回路図	R20UT0427EG
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	RX610 グループ ハードウェアマニュアル	RJJ09B0488

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 ドライブライブラリ
RSK	Renesas Starter Kit	ルネサススタータキット
USB	Universal Serial Bus	-

目次

1. 概要.....	6
1.1 目的.....	6
1.2 特徴.....	6
2. はじめに.....	7
3. チュートリアルプロジェクトワークスペース	8
4. プロジェクトワークスペース	9
4.1 はじめに.....	9
4.2 HEWの開始とE1 エミュレータの接続.....	9
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.....	10
4.3.1 ビルドコンフィグレーション.....	10
4.3.2 デバッグセッション.....	10
5. チュートリアルプログラムのビルド.....	11
5.1 コードのビルド.....	11
5.2 エミュレータの接続.....	12
6. チュートリアルのダウンロードと実行	15
6.1 プログラムコードのダウンロード.....	15
6.2 コードの実行.....	15
7. チュートリアルレビュー.....	16
7.1 プログラム初期化	16
7.2 メイン関数	19
8. 追加情報.....	24

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グ
- スイッチ、LED、ポテンシオ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رو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ーンまたは 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、デバッグ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

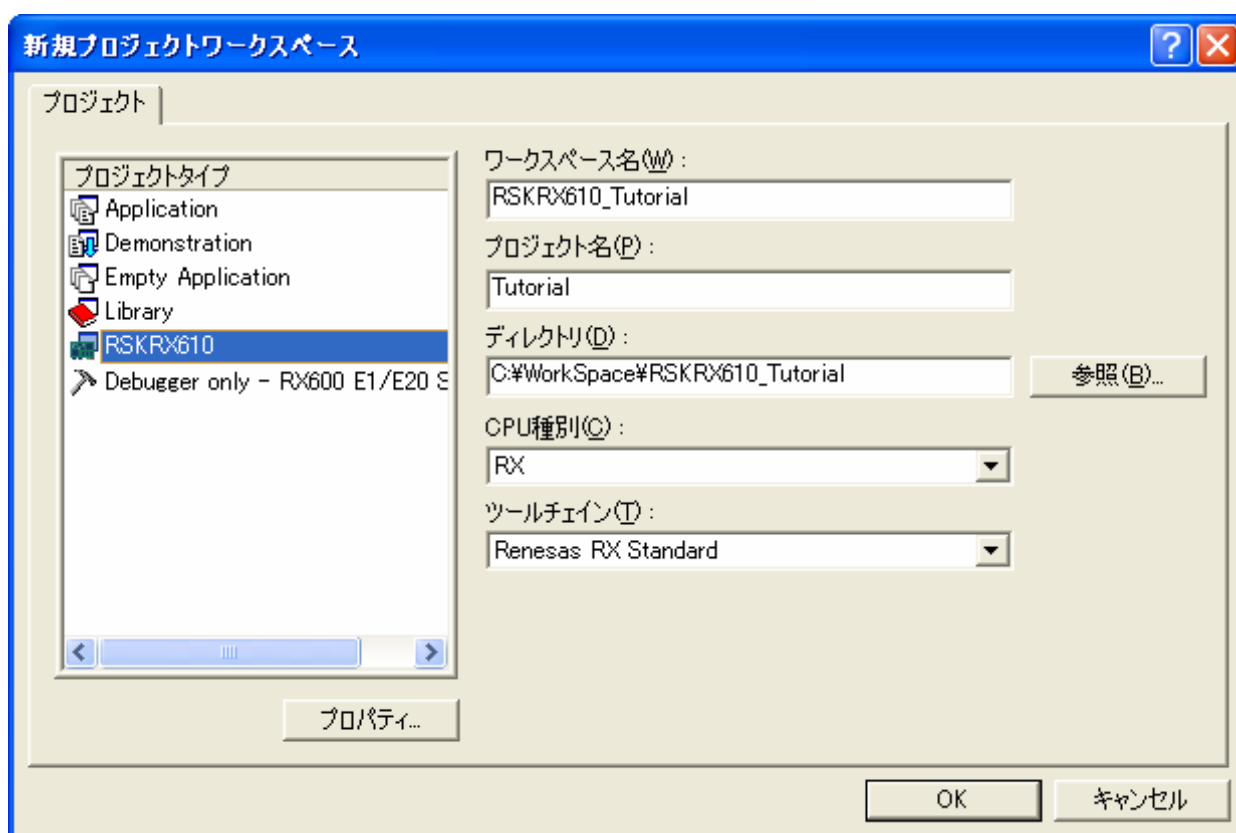
4.1 はじめに

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

まず、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 RSKRX610 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' ツールチェーンを選択します。
- プロジェクトリストから'RSKRX610'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も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、デバグ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、ユーザがデバグ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、デバグ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、デバグ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、デバグをスムーズに処理する為には、デバ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グを可能にする同様のデバ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グ・ツールの使用や、異なったデバグ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。

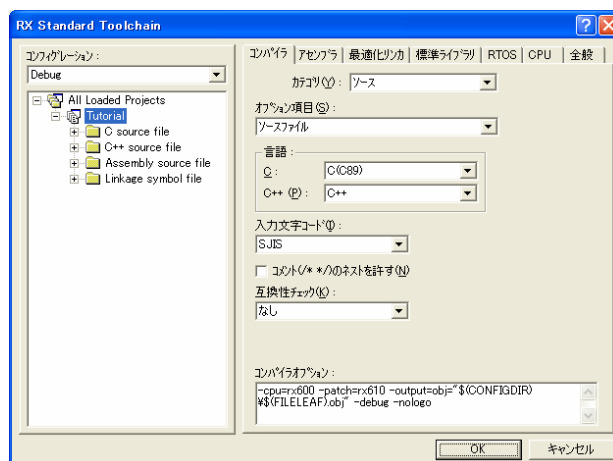


5. チュートリアルプログラムのビルド

チュートリアルプロジェクトのビルド設定は、ツールチェーンオプションで既に設定されています。ツールチェー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ー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ーンにより異なります。




コンフィグレーション画面は、全ツールチェー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グ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

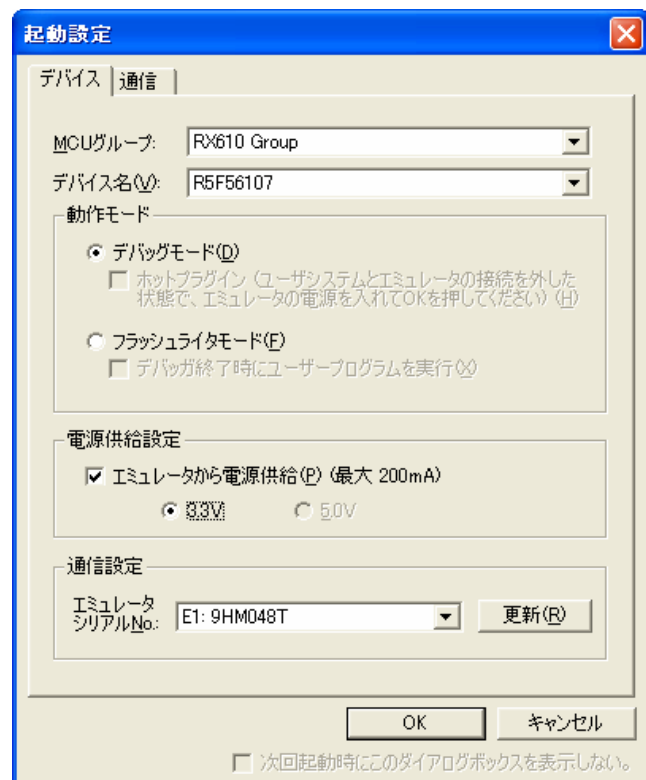
本チュートリアルでは、外部から CPU ボードに電源を供給する必要はありません。電源は USB ポートから取得されます。その USB ポートに多くのデバイスが接続している場合、Windows がシャットダウンするかもしれないので注意してください。この問題が発生した場合、一部のデバイスを削除して、もう一度やり直してください。外部電源を供給する際、極性および電源電圧が適切であることを必ず確認して下さい。

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。

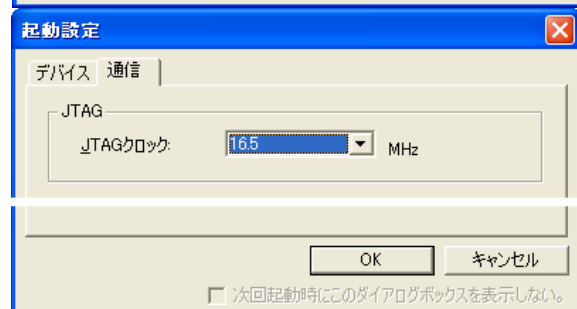
- LCD モジュールを CPU ボードの LCD コネクタに取り付け、コネクタの全てのピンが正しく接続されていることを確認して下さい。
- E1 をご使用のコンピュータの USB ポートに接続して下さい。
- E1 を CPU ボードに接続します。'E1'のシルク印字のある E1 コネクタに接続して下さい。
- 外部電源を CPU ボードに供給します。'PWR'のシルク印字のある PWR コネクタに接続して下さい。

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: RX610 Group
 - デバイス名: R5F56107
 - 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、'3.3V'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、RSKRX610 のユーザーズマニュアルを参照してください。)



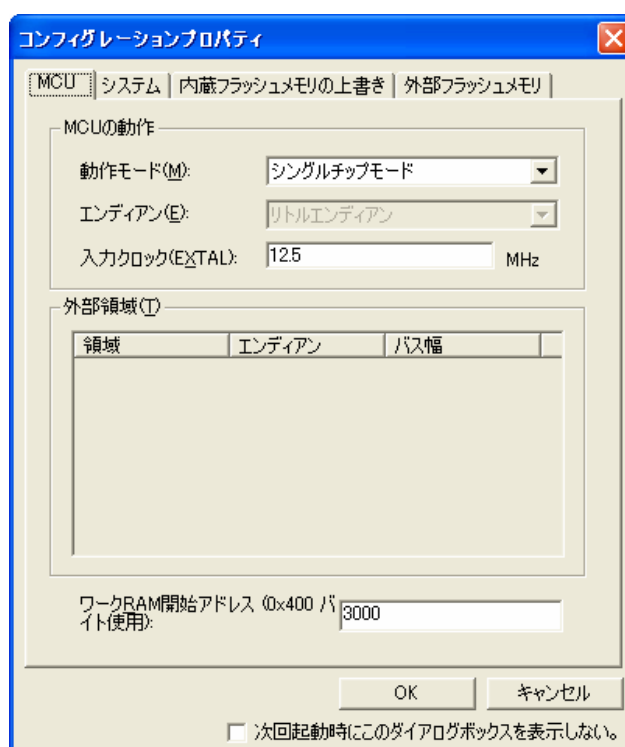
- 通信タブをクリックします。JTAG クロックが 16.5MHz に設定されていることを確認し、<OK>をクリックして下さい。
- CPU ボードとの接続を開始します。



- 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。

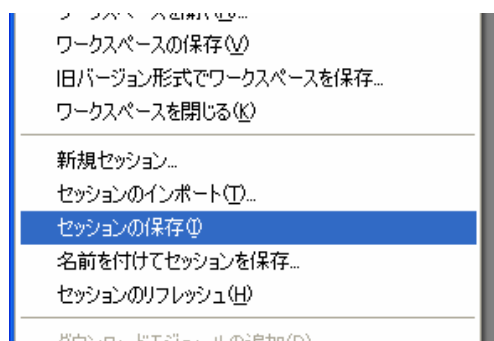


- 接続処理が終了したら'コンフィグレーションプロパティ'ダイアログが表示されます。
- 以下の通り設定します:
 - 動作モード: シングルチップモード
 - エンディアン: リトルエンディアン
 - 入力クロック: 12.5MHz
 - ワーク RAM 開始アドレス: 3000
- <OK>をクリックして下さい。High-performance Embedded Workshop のアウトプット画面に'Connected'と表示されます。
- ツールバー上のデバッグボタンがアクティブになります。これらのボタンの機能は本マニュアル中で説明されます。



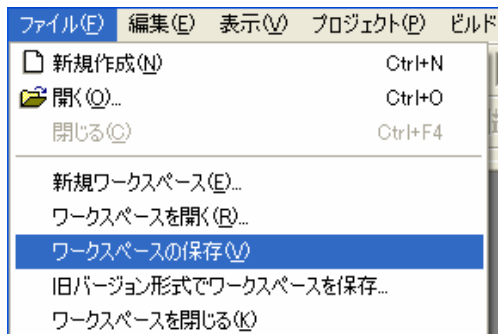
ここで、High-performance Embedded Workshop のセッションを保存することを推奨します。

- 'ファイル' | 'セッションの保存'を選択します。



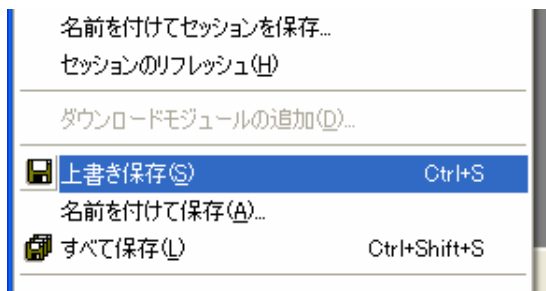
ワークスペースの設定を変更した場合、ワークスペースを保存することを推奨します。

- '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。

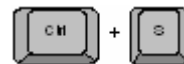
- 'ファイル' | '上書き保存'を選択します。



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、キーボードからも保存することができます。



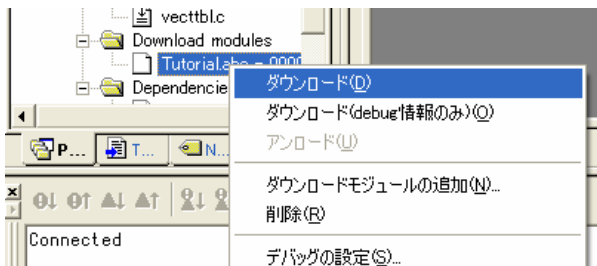
6. チュートリアルダウンロードと実行

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認します。

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 Ad...	O.	S.	Source
95				* Outline : PowerOn_Reset_PC
96				* Description : Power on reset program. Entry point af
97				* Argument : none
98				* Return value : none
99				***FUNC COMMENT END***
100				#pragma entry PowerON_Reset_PC
101	FFFE93E8			void PowerON_Reset_PC(void)
102				{
103	FFFE93F6			set_intb((unsigned long)__sectop("CSVECT"));
104	FFFE93FF			set_fpsw(FPSW_init);
105				
106	FFFE9406			_INITISCT();
107				
108				// _INIT_IOLIB(); // Use SIM I/O
109				
110				// errno=0; // Remove the comment wh
111				// srand((__UINT)1); // Remove the comment wh
112				// _slptr=NULL; // Remove the comment wh
113				
114	FFFE940A			HardwareSetup(); // Use Hardware Setup
115	FFFE940E			nop();
116				
117				// _CALL_INIT(); // Remove the comment when you u
118				
119	FFFE940F			set_psw(PSW_init); // Set Ubit & Ibit for P
120				
121	FFFE9417			Change_PSW_FM_to_UserMode();// Change PSW Pmbit (Sup
122				
123	FFFE942C			main();
124				
125				// _CLOSEALL(); // Use SIM I/O
126				
127				// _CALL_END(); // Remove the comment when you u
128				
129	FFFE9430			brk();
130				}
131				*****

- ‘HardwareSetup()’を左クリックで選択して下さい。

```

99
100
101 FFFE93E8 #pragma entry PowerON_Reset_PC
102
103 FFFE93F6 void PowerON_Reset_PC(void)
104 FFFE93FF {
105
106 FFFE9406     set_intb((unsigned long)__sectop("CSVECT"));
107
108             set_fpsw(FPSW_init);
109
110             _INITISCT();
111
112             // _INIT_IOLIB();           // Use SIM I/O
113
114             // errno=0;                 // Remove the comment when you
115             // srand(_UINT1);           // Remove the comment when you
116             // _slptr=NULL;           // Remove the comment when you
117
118             HardwareSetup();           // Use Hardware Setup
119             nop();
120
121             // _CALL_INIT();           // Remove the comment when you use glc
122
123             set_psw(PSW_init);         // Set Ubit & Ibit for PSW
124
125             Change_PSW_PM_to_UserMode(); // Change PSW PMbit (SuperVis
126 FFFE9417
    
```

- 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。

```

51
52
53 #include "hwsetup.h"
54
55 /*"FUNC COMMENT"*****
56 * Outline       : HardwareSetup
57 * Description   : Contains all the setup functions (
58 * Argument      : none
59 * Return value  : none
60 *"FUNC COMMENT END"*****
61 void HardwareSetup(void)
62 {
63     ConfigureOperatingFrequency();
64     ConfigureOutputPorts();
65     ConfigureInterrupts();
66     EnablePeripheralModules();
67 }
68
69 End of HardwareSetup
70
    
```

- ‘hwsetup.c’ファイルを下へスクロールすると、セットアップのためのいくつかの関数を見ることができます。
- CofigureOperatingFrequency 関数は R_CGC_set API によってクロック設定を行います。

```

/*"FUNC COMMENT"*****
* Outline       : ConfigureOperatingFrequency
* Description   : ConfigureOperatingFrequency
* Argument      : none
* Return value  : none
*"FUNC COMMENT END"*****
void ConfigureOperatingFrequency(void)
{
    /* Modify the MCU clocks */
    R_CGC_Set(12.5E6, 100E6, 50E6, 25E6, PDL_CGC_BCLK_DISABLE);

    /*
    Clock Description           Frequency
    -----
    Main Clock Frequency.....12.5MHz
    Internal Clock Frequency.....100MHz
    Peripheral Clock Frequency.....50MHz
    External Bus Clock Frequency.....25MHz */
}
End of ConfigureOperatingFrequency
    
```

- `ConfigureOutputPorts` 関数は `R_IO_PORT_Set` API によってポート制御レジスタ、方向レジスタを設定します。LED、LCD 用のピンはすべて出力に設定しています。
- 次のコマンド群ではスイッチ、ポテンショメータ用のピンを入力に設定しています。マイクロコントローラのほとんどのピンは複数の機能を持っているので、プログラムごとに異なるポート設定が必要になります。

```
void ConfigureOutputPorts(void)
{
    /* Configure the LED control pins */
    R_IO_PORT_Set(PDL_IO_PORT_8_3, PDL_IO_PORT_OUTPUT, 0, 0, 0);
    R_IO_PORT_Set(PDL_IO_PORT_8_4, PDL_IO_PORT_OUTPUT, 0, 0, 0);
    R_IO_PORT_Set(PDL_IO_PORT_3_3, PDL_IO_PORT_OUTPUT, 0, 0, 0);
    R_IO_PORT_Set(PDL_IO_PORT_3_4, PDL_IO_PORT_OUTPUT, 0, 0, 0);
    R_IO_PORT_Write(PDL_IO_PORT_8_3, 1);
    R_IO_PORT_Write(PDL_IO_PORT_8_4, 1);
    R_IO_PORT_Write(PDL_IO_PORT_3_3, 1);
    R_IO_PORT_Write(PDL_IO_PORT_3_4, 1);

    /* Configure the LCD control pins */
    R_IO_PORT_Write(RS_PIN, 0);
    R_IO_PORT_Write(E_PIN, 0);
    R_IO_PORT_Write(PDL_IO_PORT_9, 0x00);
    R_IO_PORT_Set(PDL_IO_PORT_9, 0xF0, 0, 0, 0);
    R_IO_PORT_Set(PDL_IO_PORT_8_5, PDL_IO_PORT_OUTPUT, 0, 0, 0);
    R_IO_PORT_Set(PDL_IO_PORT_8_6, PDL_IO_PORT_OUTPUT, 0, 0, 0);

    /* Enable the push switch inputs */
    R_IO_PORT_Set(PDL_IO_PORT_0_0, PDL_IO_PORT_INPUT,
                  PDL_IO_PORT_INPUT_BUFFER_ON, 0, 0);
    R_IO_PORT_Set(PDL_IO_PORT_0_1, PDL_IO_PORT_INPUT,
                  PDL_IO_PORT_INPUT_BUFFER_ON, 0, 0);
    R_IO_PORT_Set(PDL_IO_PORT_1_3, PDL_IO_PORT_INPUT,
                  PDL_IO_PORT_INPUT_BUFFER_ON, 0, 0);

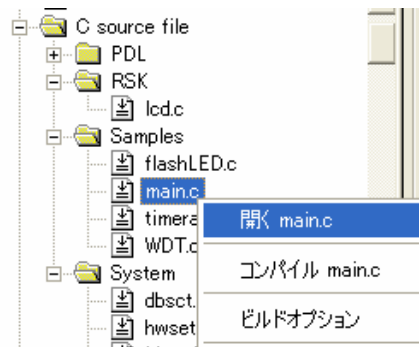
    /* Configure the ADC Pot input */
    R_IO_PORT_Set(PDL_IO_PORT_4_0, PDL_IO_PORT_INPUT, 0, 0, 0);
}
```

ハードウェア設定に関する詳細情報は、RSKRX610 ユーザーズマニュアルおよび RX610 グループハードウェアマニュアルを参照して下さい。

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。

<pre> 76 77 78 79 FFFF3212 80 81 82 FFFF3212 83 84 85 FFFF3216 86 FFFF3224 87 88 89 FFFF3233 90 91 92 FFFF3237 93 94 95 FFFF323B 96 97 98 FFFF323E 99 100 101 102 103 </pre>	<pre> * Argument : none * Return value : none **"FUNC COMMENT END"***** void main(void) { /* Initialise the LCD Display */ InitialiseLCD(); /* Displays the Renesas splash screen */ DisplayLCD(LCD_LINE1, "Renesas"); DisplayLCD(LCD_LINE2, NICKNAME); /* Begins the initial LED flash sequence */ FlashLED(); /* Begins the ADC-varying flash Sequence */ TimerADC(); /* Begins the static variable test */ Statics_Test(); /* Defines an infinite loop to keep the MCU running */ while(1); } ***** End main function ***** </pre>
--	--

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 ファミリー用 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せん。'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。

```

76      * Argument      : none
77      * Return value : none
78      *"FUNC COMMENT END"*****
79 FFFF3212 void main(void)
80      {
81      /* Initialise the LCD Display */
82 FFFF3212 InitialiseLCD();
83
84      /* Displays the Renesas splash screen */
85 FFFF3216 DisplayLCD(LCD_LINE1, "Renesas");
86 FFFF3224 DisplayLCD(LCD_LINE2, NICKNAME);
87
88      /* Begins the initial LED flash sequence */
89 FFFF3233 FlashLED();
90
91      /* Begins the ADC-varying flash Sequence */
92 FFFF3237 TimerADC();
93
94      /* Begins the static variable test */
95 FFFF323B Statics_Test();
96
97      /* Defines an infinite loop to keep the MCU running */
98 FFFF323E while(1);
99      }
100
101 End main function
102
103

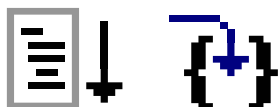
```

- FlashLED 関数にブレークポイントを設定して下さい。
- デバッグツールバーの '実行' 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。

```

77
78      /* Begins the
79 FFFE2CEF FlashLED();
80

```



- FlashLED 関数を見て下さい。この関数は R_TMR_CreatePeriodic API 関数によりタイマ設定を行います。

```

87      * Return value : none
88      *"FUNC COMMENT END"*****
89 FFFF04DF void FlashLED (void)
90      {
91 FFFF04DF /* PDL function creates a timer to flash the LEDs at 1Hz */
92      R_TMR_CreatePeriodic(PDL_TMR_UNIT1, PDL_TMR_FREQUENCY, 4, 50, PDL_NO_FUNC,
93                          TMR_Callback, 5);
94
95      /* While loop keeps the function waiting */
96      while(1)
97      {
98 FFFF055A /* Checks if the flash count has been reached, or if a button has been pressed */
99      if((KeyPress)|| (flash_count > 0x190))
100      {
101 FFFF0578 /* Reset the KeyPress flag variable */
102      KeyPress = 0;
103
104      /* Exit from the while loop */
105      break;
106      }
107      }
108
109 FFFF058B /* Destroy Timer */
110      R_TMR_Destroy(1);
111      }
112

```

- ‘FlashLED()’にブレークポイントを設定し、デバッグツールバーの‘実行’を押して下さい。タイマが期間の終了に達する時、割り込みは TMR_Callback 関数をコールします。また、プログラムカウンタはブレークポイントで停止します。

```

123      * Return value : none
124      *""FUNC COMMENT END""*****
125      FFFF059E      void TMR_Callback(void)
126      {
127          /* Blink LEDs */
128      FFFF059E      ToggleLEDs();
129
130          /* Decrement the flash count */
131      FFFF05A1      flash_count--;
132      }
133      /******
134      End of function TMR_Callback
135      *""FUNC COMMENT END""*****
136  ...
    
```

- TMR_Callback 関数は ToggleLEDs 関数 (LED に割り当てられたピンの出力を反転させます) を実行します。また、flash_count 変数 (200 回の LED 点滅回数カウントに使用します) をデクリメントします。
- 青色のドットシンボルをダブルクリックし、ブレークポイントを削除して下さい。

- ‘main.c’ ファイルを開き、‘TimerADC()’に該当する S/W 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。

```

88      /* Begins the
89      FFFF3237      TimerADC();
90
    
```

- デバッグツールバーの‘実行’ボタンまたは F5 キーを押してプログラムを再開し、ボード上の SW1 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。

```

69      * Argument : none
70      * Return value : none
71      *""FUNC COMMENT END""*****
72      FFFF3664      void TimerADC(void)
73      {
74      FFFF3664      StartTimer();
75
76          /* Call the ADC start function */
77      FFFF3667      StartADC();
78
79          /* Disable switch interrupts */
80      FFFF366A      R_INTC_ControlExtInterrupt(PDL_INTC_IRQ3, PDL_INTC_RISING|PDL_INTC
81      FFFF3683      R_INTC_ControlExtInterrupt(PDL_INTC_IRQ8, PDL_INTC_RISING|PDL_INTC
82      FFFF369C      R_INTC_ControlExtInterrupt(PDL_INTC_IRQ9, PDL_INTC_RISING|PDL_INTC
83      }
84      /******
85      End TimerADC function
86      *""FUNC COMMENT END""*****
    
```

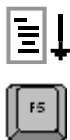
- デバッグツールバーの‘ステップイン’をクリックして TimerADC 関数にエントリします



- TimerADC 関数はタイマおよび AD 変換の両方を設定します。タイマは AD 変換の起動トリガに使用され、AD 変換が完了すると ADC_callback 関数をコールします。

- 画面下にスクロールし、'ADC_callback(void)'にブレークポイントを設定して下さい。

- デバッグツールバーの'実行'または F5 キーを押して、プログラムを再開します。タイマ期間が終了すると関数がコールされ、プログラムカウンタはブレークポイントで停止します。



```

130 | | | * Return value : none
131 | | | *""FUNC COMMENT END""*****
132 | FFFF3752 | void ADC_callback(void)
133 | | | {
134 | | |     /* Fetch ADC value */
135 | FFFF3752 | R ADC_10_Read(0, &ad_value);
136 | | |
137 | | |     /* Toggle LEDs */
138 | FFFF376B | ToggleLEDs();
139 | | |
140 | | |     /* Start a new timer delay */
141 | FFFF376F | R TMR_ControlUnit(1, PDL_TMR_TIME_CONSTANT_A, 0, (ad_value + 0xF), 0);
142 | | | }
143 | | |
144 | | | End ADC_callback function
145 | | |

```

- ADC_callback 関数は最初に AD 変換結果をフェッチし、次に LED を反転出力させるために ToggleLEDs 関数をコールします。その後、フェッチした AD 変換結果によってタイマ期間を再設定し、LED の点滅周期を変更します。

- F5 キーを押して、再度プログラムがループを完了できるようにします。そして、ad_value 変数の値を確認してください。



- ADC_callback 関数に設定したブレークポイントを削除してください。

- 'main.c'ファイルを開きます。
- TimerADC 関数は割り込みによって LED の点滅周期を変えている一方、Static_Test 関数は割り込みを使用せずに動作します。
- Statics_Test 関数は静的変数の文字列を初期化します。初期化後、別の文字列に置き換えます。
- '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「RX610」に戻します。

```

91 | | |
92 | FFFF323B |
93 | | |

```

```

/* Begins the sta
Statics_Test();

```

- その後、プログラムは無限ループに入ります。(TimerADC は割り込みによって動作を続けます)
- 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 Peripheral 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。

RX610 マイクロコントローラに関する詳細情報は、RX610 グループハードウェアマニュアル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 ファミリーコンパイラパッケージユーザズマニュアル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/renesas_starter_kits (日本サイト)
http://www.renesas.com/renesas_starter_kits (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

本書の内容の一部または全てを予告無しに変更することがあります。
本書の著作権はルネサス エレクトロニクス株式会社にあります。ルネサス エレクトロニクス株式会社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。

© 2011 Renesas Electronics Corporation. All rights reserved.
© 2011 Renesas Electronics Europe Ltd. All rights reserved.
© 2011 Renesas Solutions Corporation. All rights reserved.

改訂記録	RSKRX610 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	2010.03.19	－	初版発行
2.00	2011.02.07	－	ドキュメント番号、ドキュメントスタイルを更新

RSKRX610 チュートリアルマニュアル

発行年月日 2011年2月7日 Rev.2.00

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>

RX610 グループ