User's Manual

16

RL78/L1C

Renesas Starter Kit
コード生成支援ツール チュートリアルマニュアル (CubeSuite+)

16 ビット・シングルチップ・マイクロコントローラ
RL78 ファミリ

ルネサス エレクトロニクス
www.renesas.com

Rev.1.02 2014.04

ご注意書き

1. 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・システムの設計において、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て、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。
2. 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。
3. 本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害に関し、当社は、何らの責任を負うものではありません。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。
4. 当社製品を改造、改変、複製等しないでください。かかる改造、改変、複製等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。
5. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、
   各品質水準は、以下に示す用途に製品が使用されることを意図しております。

   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、産業用ロボット等

   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、各種安全装置等

   当社製品は、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（原子力制御システム、軍事機器等）に使用されることを意図しておらず、使用することはできません。たとえ、意図し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損害が生じても、当社は一切その責任を負いません。なお、ご不明点がある場合は、当社営業にお問い合わせください。
6. 当社製品をご使用の際は、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他の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。
7. 当社は、当社製品の品質および信頼性の向上に努めていますが、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、社会的損害等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、お客様の機器・システムとしての出荷保証を行ってください。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、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
8. 当社製品の環境適合性等の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。
9. 本資料に記載されている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・システムに使用することはできません。また、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に使用しないでください。当社製品または技術を輸出する場合は、「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。
10. お客様の転売等により、本ご注意書き記載の諸条件に抵触して当社製品が使用され、その使用から損害が生じた場合、当社は何らの責任も負わず、お客様にてご負担して頂きますのでご了承ください。
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禁じます。

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。

(2012.4)

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、統合開発環境CubeSuite+およびRL78用Application Leading Tool（以下Appliletと称す）を使用してRSKプラットフォーム用プロジェクトを作成するための方法を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、段階的に CubeSuite+中のプロジェクトをロードし、デバッグする指示を含みますが、RSK プラットフォーム上のソフトウェア開発のガイドではありません。

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。

RSKRL78L1C 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	RSKRL78L1C ユーザーズマニュアル	R20UT2203JG
チュートリアルマニュアル	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	RSKRL78L1C チュートリアルマニュアル	R20UT2204JG
コード生成支援ツール チュートリアルマニュアル	コード生成支援ツールの使用方法の説明	RSKRL78L1C コード生成支援ツール チュートリアルマニュアル	R20UT2892JG (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	RSKRL78L1C クイックスタートガイド	R20UT2205JG
回路図	CPU ボードの回路図	RSKRL78L1C CPU ボード回路図	R20UT2202EG
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	RL78/L1C ユーザーズマニュアル ハードウェア編	R01UH0409JJ

2. 略語および略称の説明

略語／略称	英語名	備考
ADC	Analog 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
CPU	Central Processing Unit	中央処理装置
DVD	Digital Versatile Disc	ディジタルヴァーサタイルディスク
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
GUI	Graphical User Interface	グラフィカルユーザインタフェース
LCD	Liquid Crystal Display	液晶ディスプレイ
LED	Light Emitting Diode	発光ダイオード
MCU	Micro-controller Unit	マイクロコントローラユニット
ROM	Read-Only Memory	リードオンリーメモリ
RSK	Renesas Starter Kit	ルネサススタータキット
SAU	Serial Array Unit	シリアルアレイユニット
TAU	Timer Array Unit	タイマアレイユニット
UART	Universal Asynchronous Receiver/Transmitter	調歩同期式シリアルインタフェース
USB	Universal Serial Bus	-
WDT	Watchdog Timer	ウォッチドッグタイマ

目次

1. 概要 7
1.1 目的 7
1.2 特徴 7

2. はじめに 8

3. Applilet を使用したコード生成 9
3.1 はじめに 9
3.2 Applilet ツアー 10
3.3 コード生成 12
3.3.1 共通/クロック発生回路 12
3.3.2 割り込み機能 13
3.3.3 LCD コントローラ/ドライバ 14
3.3.4 タイマ・アレイ・ユニット 15
3.3.5 ウォッチドッグ・タイマ 17
3.3.6 A/D コンバータ 17
3.3.7 シリアル・アレイ・ユニット 18
3.3.8 ポート機能 20

4. CubeSuite+への組み込み 22
4.1 CubeSUite+へのコード組み込み 22
4.2 プロジェクト設定 24
4.3 LCD パネルコードの統合 26
4.4 スイッチコードの統合 27
4.4.1 割り込みコード 27
4.4.2 デバウンス用タイマコード 31
4.4.3 A/D コンバータコードとメインスイッチコード 32
4.5 デバッグコードの統合 36
4.6 UART コードの統合 36
4.6.1 シリアル・アレイ・ユニットコード 36
4.6.2 メイン UART コード 39
4.7 LED コードの統合 41

5. プロジェクトデバッグ 43

6. チュートリアルコードの実行 44
6.1 コードの実行 44

7. 追加情報 45

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、統合開発環境 CubeSuite+ および RL78 用コード生成支援ツール Applilet を使用してプロジェクトを作成する方法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：

- Applilet を使用してのコード生成
- CubeSuite+によるプロジェクト作成およびビルド
- スイッチ、LED、ポテンショメータ等のユーザ回路

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

本マニュアルは統合開発環境 CubeSuite+および RL78 用コード生成支援ツール Applilet を使用してプロジェクトを作成する方法についてチュートリアル形式で説明しています。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。

- Applilet を使用した周辺機能設定とコード生成について
- 生成したコードの CubeSuite+プロジェクトへの組み込み
- カスタムコードの統合
- CubeSuite+プロジェクトのビルド

プロジェクトジェネレータは、選択可能な 3 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。

- ‘DefaultBuild’はデバッガのサポートおよび最適化レベル 2 を含むプロジェクトを構築します。
- ‘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。最適化は行いません。
- ‘Release’は最適化された製品リリース用に適したコードを構築します。

本マニュアルにおいて、スクリーンショット内に RL78XXX と記載されている部分は、RL78/L1C を指します。

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、CubeSuite+、コンパイラまたは 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. Applilet を使用したコード生成

3.1 はじめに

本製品で提供しているサンプルコードの一部は、Applilet を使用してコードを生成しています。Applilet は C ソースコード生成とマイクロコントローラの生成のための GUI ツールです。Applilet は直感的な GUI を使用することで、様々なマイクロコントローラの周辺機能や動作に必要なパラメータを設定することができ、開発工数の大幅な削減が可能です。
Applilet によって生成されるコードは、特定の周辺ごとに 3 つのコードを生成します（「r_cg_xxx.h」、「r_cg_xxx.c」、「r_cg_xxx_user.c」）。例えば A/D コンバータの場合、周辺を表す xxx は'adc'と名付けられます。これらのコードはユーザの要求を満たすために、カスタムコードを自由に加えることができます。カスタムコードを加える場合、以下に示すコメント文の間にカスタムコードを加えてください。

```
/* Start user code for adding. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */
```

Applilet の GUI 上で設定した内容を変更したい場合等、再度コード生成を行う場合に Applilet はこれらのコメント文を見つけて、コメント文の間に加えられたカスタムコードを保護します。

Applilet は Renesas ウェブサイトからダウンロードすることができます。
http://japan.renesas.com/applilet（日本サイト）
http://www.renesas.com/applilet （グローバルサイト）

ユーザは本チュートリアルを進めることで、Applilet を使用したコード生成方法および CubeSuite+プロジェクト CG_Tutorial への組み込み方法を学ぶことができます。

チュートリアルプロジェクト CG_Tutorial は、スイッチによる割り込み、A/D モジュール、タイマ・アレイ・ユニット（TAU）、シリアル・アレイ・ユニット（SAU）を使用し、A/D 変換値をターミナルソフトや LCD パネルに表示します。

次のセクション 3.2 では Applilet のユーザインタフェースについて、セクション 3.3 では各周辺機能ダイアログについて、4 章では生成されたコードの CubeSuite+プロジェクトへの組み込み、カスタムコードの追加方法、チュートリアルコードの構造について説明します。

3.2 Applilet ツアー

このセクションでは、Applilet の簡単な操作方法を示しています。各操作の詳細につきましては、Application Leading Tool 共通操作編ユーザーズマニュアル(R20UT2663)を参照ください。

スタートメニュー → すべてのプログラム → Renesas Electronics Application Leading Tool → RL78 → Vx.xx.xx → RL78 Vx.xx.xx Application Leading Tool を選択すると Applilet が起動します。メニューバーの”ファイル”から”新規プロジェクト”を選択すると、図 3-1 のダイアログが表示されます。

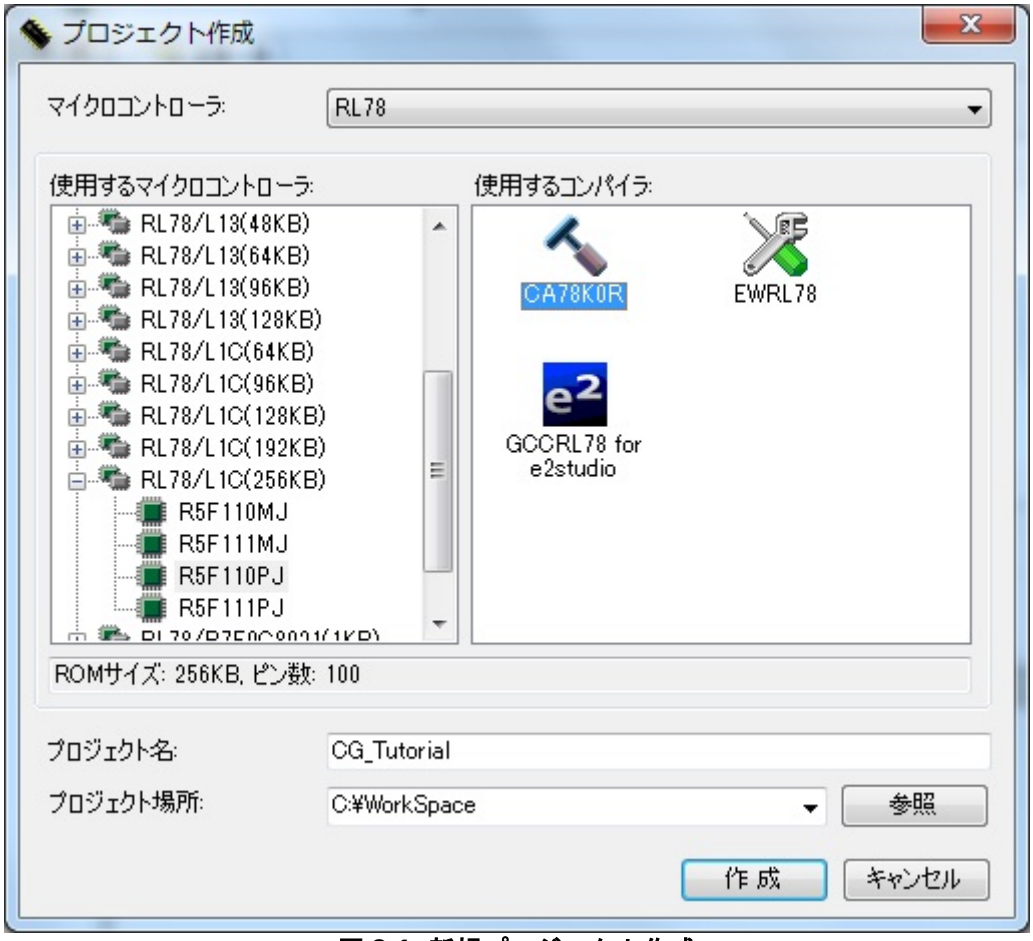


図 3-1: 新規プロジェクト作成

“使用するマイクロコントローラ”から”RL78/L1C(256KB)”を展開して、”R5F110PJ”を選択してください。次に、”使用するコンパイラ”から”CA78K0R”を選択してください。プロジェクト名およびプロジェクト場所を図 3.1 と同じように入力してください。

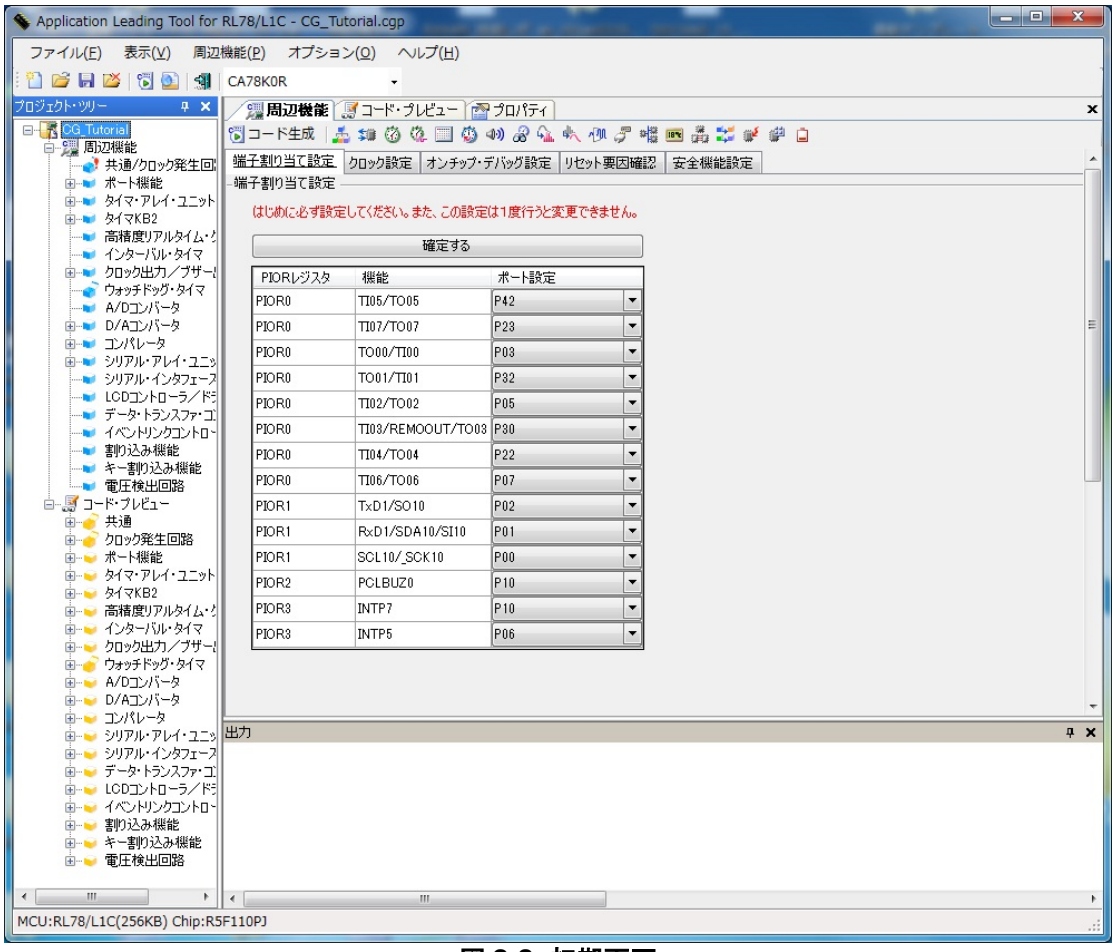


図 3-2: 初期画面

Applilet では、MCU 設定を GUI で操作することができます。ユーザが必要な設定を完了し、”コード生成”ボタンをクリックすると、設定した内容のコードが生成されます。

周辺機能の設定は、次の 3 つの方法で行うことが可能です。

- プロジェクト・ツリー内の周辺機能をダブルクリック
- メニューバーから”周辺機能(P)”を選択
- グラフィカルツールバーから周辺機能アイコンをクリック

プロジェクト・ツリー内のプロジェクトからコード・プレビューにある周辺機能をダブルクリックすることで生成されるコードのプレビューを表示します。

メニューバーの”表示(V)”から、周辺機能、コード・プレビュー、プロパティタブ画面を表示することができます。

Applilet で新規プロジェクトを作成した場合、”端子割り当て設定”タブが現れます。この設定は、RL78/L1C の周辺 I/O リダイレクト機能に該当し、兼用機能を割り当てるポートを切り替える機能です。他の周辺機能を設定する前に確定してください。ただし、一度確定すると”端子割り当て設定”は変更することができなくなりますのでご注意ください。変更したい場合は、再度新規プロジェクトを作成する必要があります。

ここでは”端子割り当て設定”を変更せずに、”確定する”ボタンを押して次に進んでください。

3.3 コード生成

このセクションでは、MCU 設定、スイッチによる割り込み設定、タイマ設定、A/D コンバータ設定、UART の設定を行っていきます。

3.3.1 共通/クロック発生回路

共通/クロック発生回路を図 3-3 に示します。”クロック設定”タブをクリックしてください。図のように設定値を入力してください。チュートリアルでは fMX に 12MHz、fSUB に 32.768kHz を使用します。fMX は、CPU および周辺に使用します。fSUB は、RTC、インターバル・タイマおよび LCD に使用します。

“オンチップ・デバッグ設定”、”リセット要因確認”、”安全機能設定”につきましては、初期設定の状態で問題ありません。

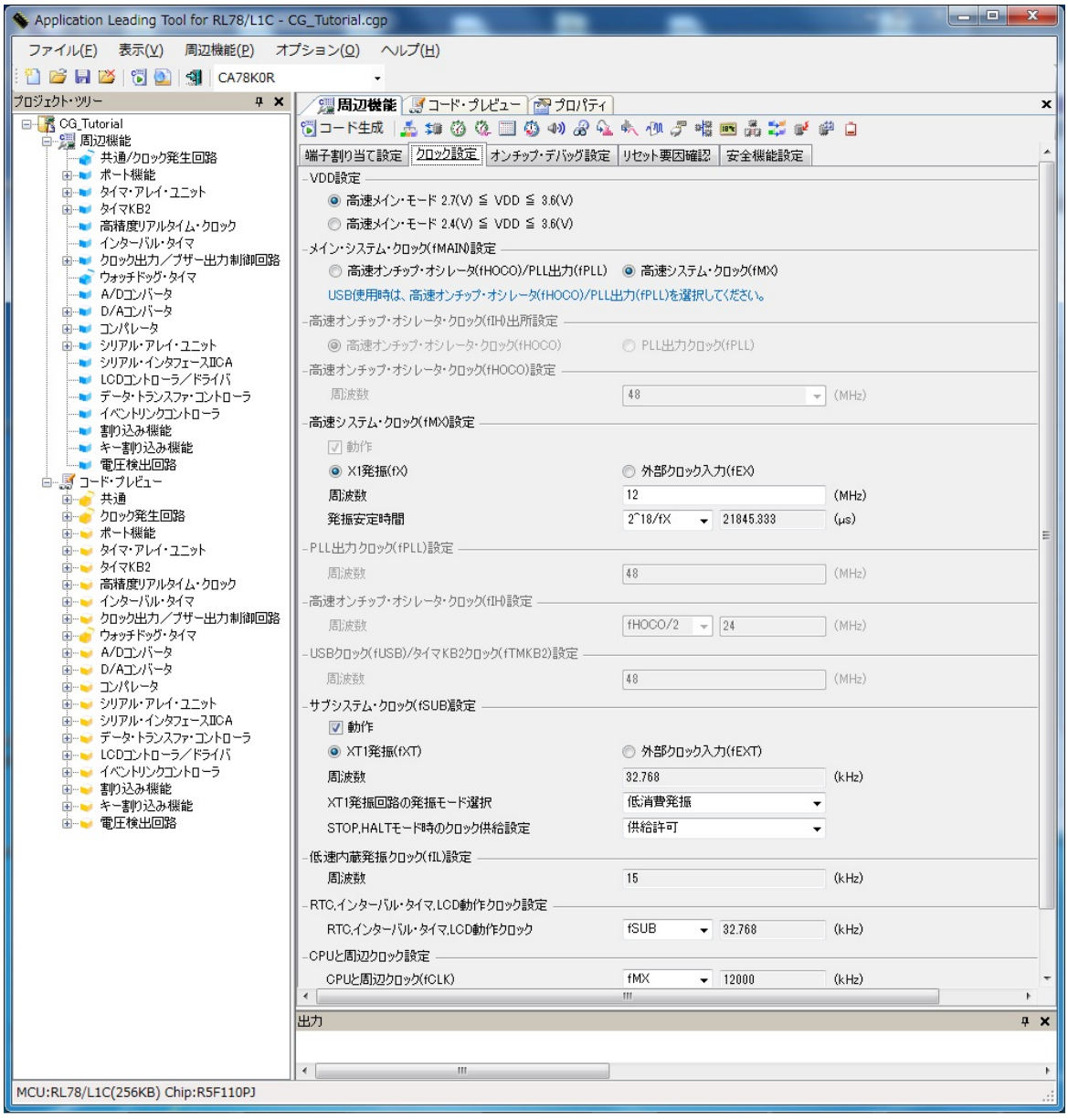


図 3-3: クロック設定タブ

次に割り込み機能を設定します。ポート機能については、周辺機能で使用するポートに依存する箇所があるため、周辺機能の設定が完了してから行います。

3.3.2 割り込み機能

RSKRL78L1C の CPU ボードでは、SW1 は INTP0、SW2 は INTP1、SW3 は INTP2 が接続されています。
割り込み機能の設定を図 3-4 に示します。

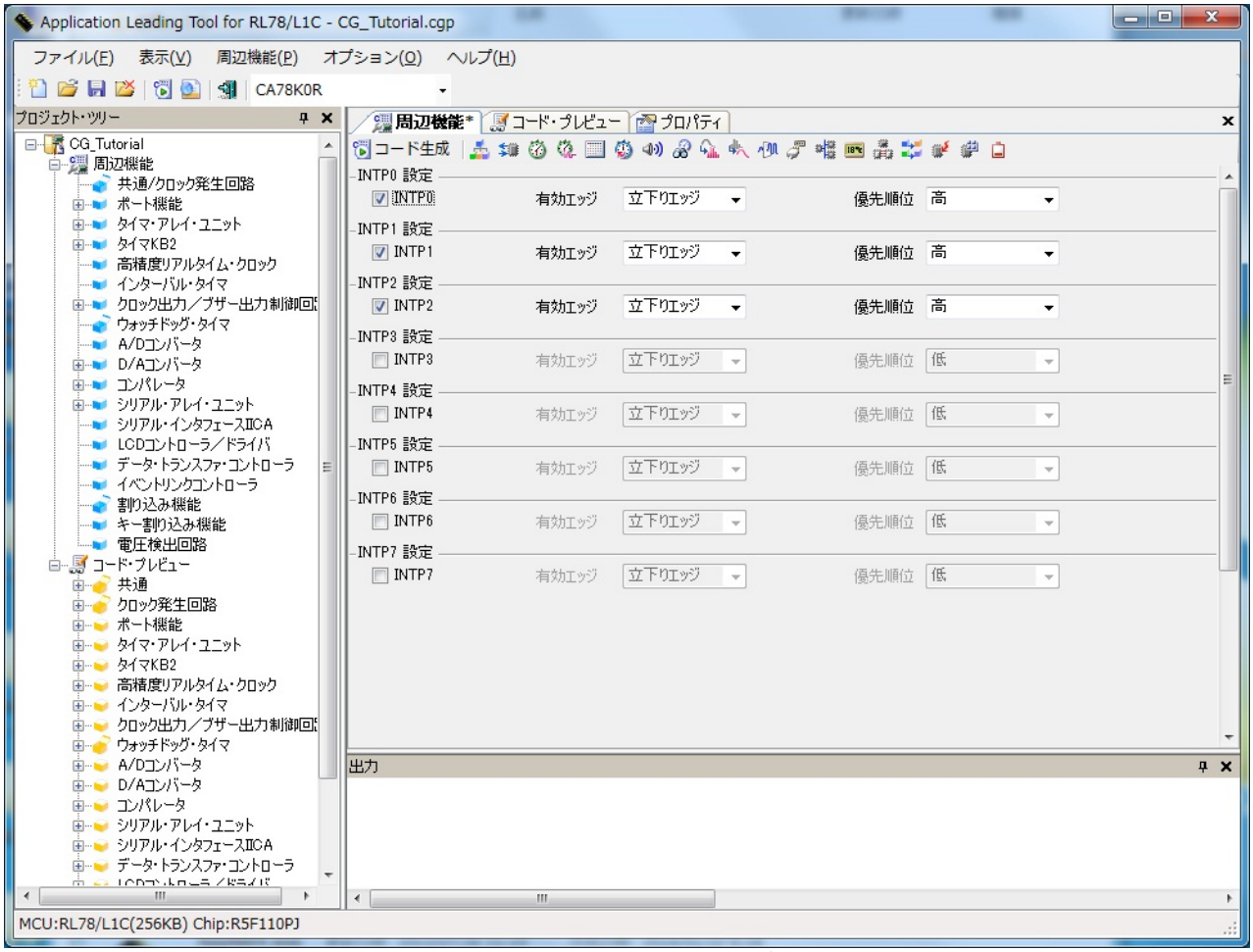


図 3-4: 割り込み機能タブ

3.3.3 LCD コントローラ/ドライバ

LCD コントローラ/ドライバの設定を図 3-5 に示します。
注意：SEG51 と SEG52 の横にある”！”マークは、すでに INTP1 と INTP2 に使用されていることを示します。

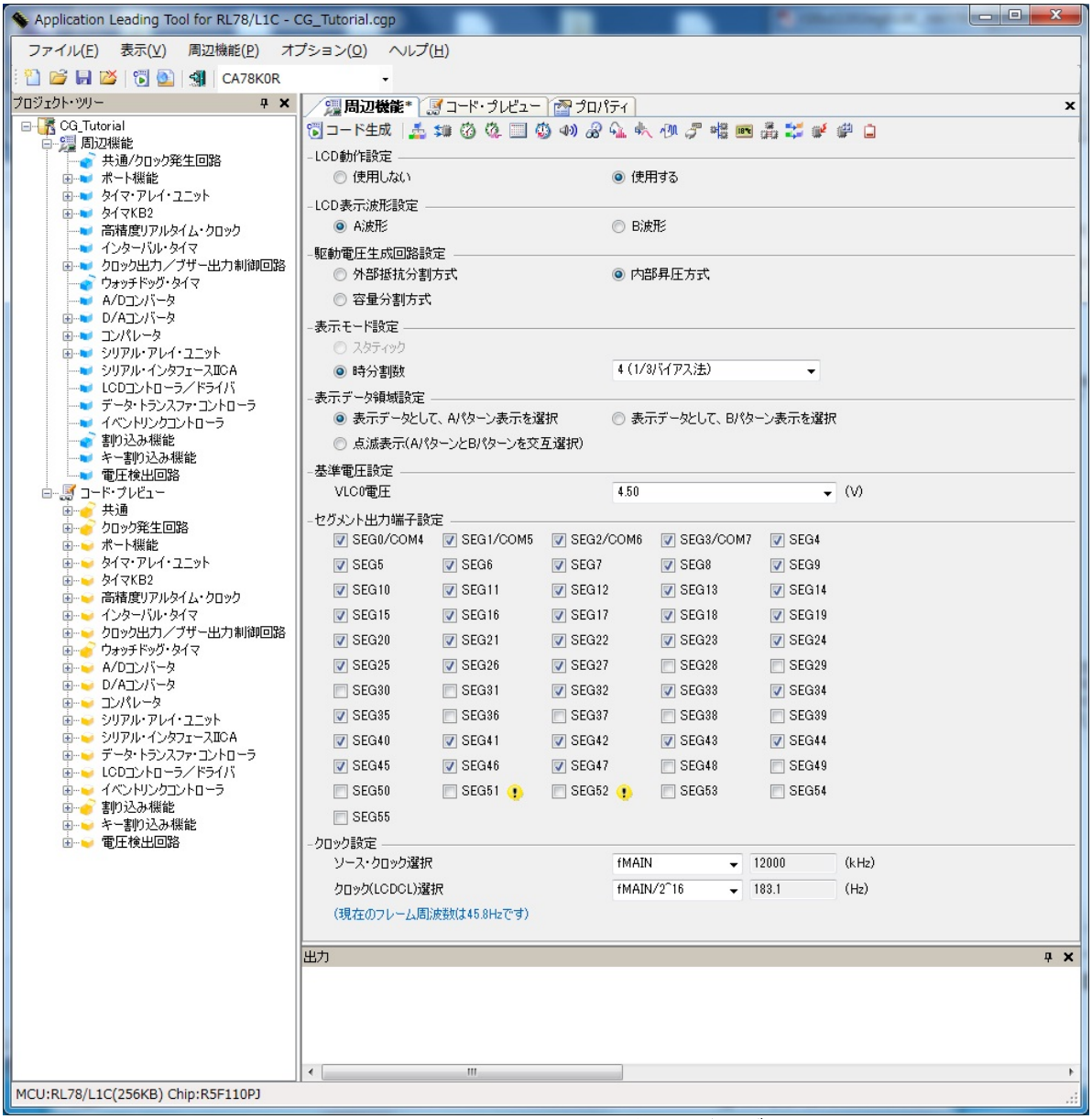


図 3-5: LCD コントローラ/ドライバタブ

3.3.4 タイマ・アレイ・ユニット

図 3-6 にタイマ・アレイ・ユニットの設定を示します。“一般設定”タブのチャネル 0～3 のプルダウンメニューからインターバル・タイマを選択します。チャネル 0 は、正確な遅れ時間を生成するためのタイマとして使用します。チャネル 1 および 2 は、スイッチ割り込みのデバウンス用タイマとして使用します。

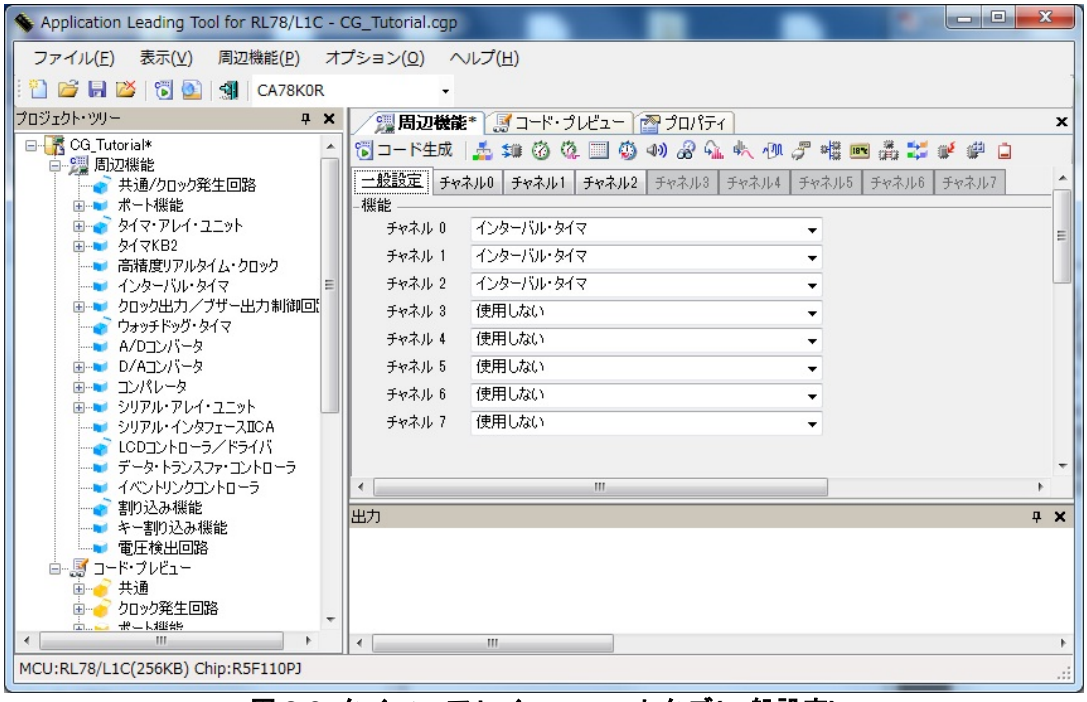


図 3-6: タイマ・アレイ・ユニットタブ(一般設定)

図 3-7 にチャネル 0 の設定を示します。チャネル 0 のインターバル・タイマ設定を 1ms 毎に割り込み(レベル高)が発生するようにします。このタイマは、正確な遅れ時間の生成に使用します。

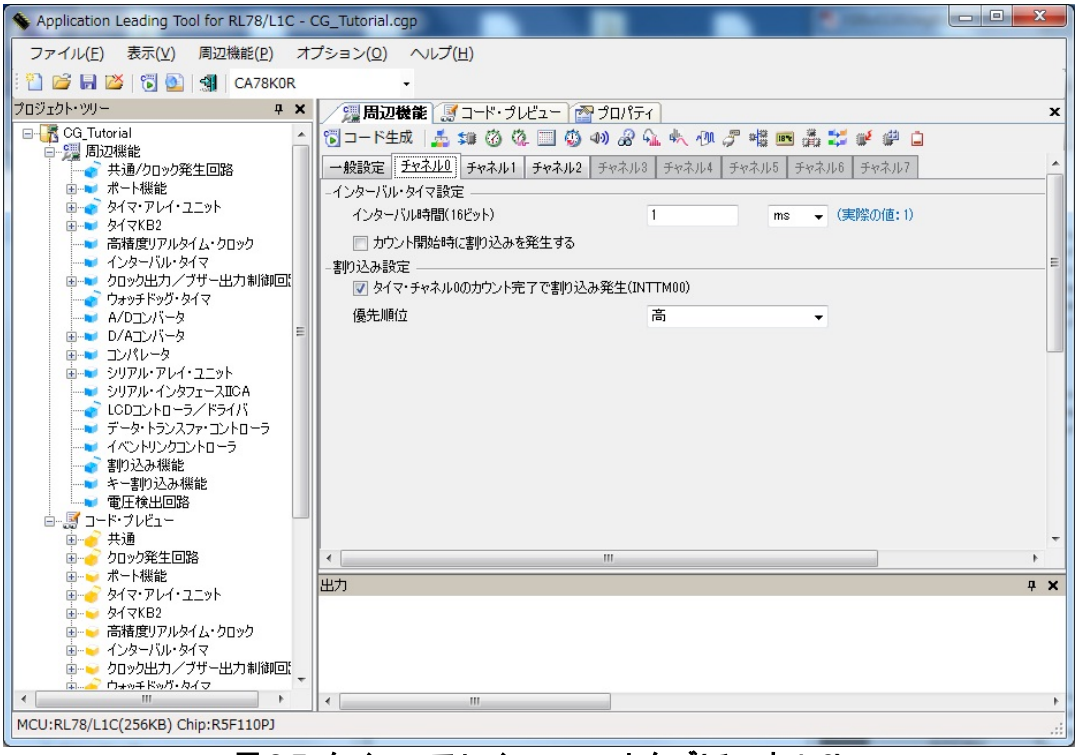


図 3-7: タイマ・アレイ・ユニットタブ(チャネル 0)

図 3-8 にチャネル 1 の設定を示します。チャネル 1 のインターバル・タイマ設定を 20ms 毎に割り込み(レベル高)が発生するようにします。このチャネルのタイマは、スイッチ短絡のデバウンスとして使用します。

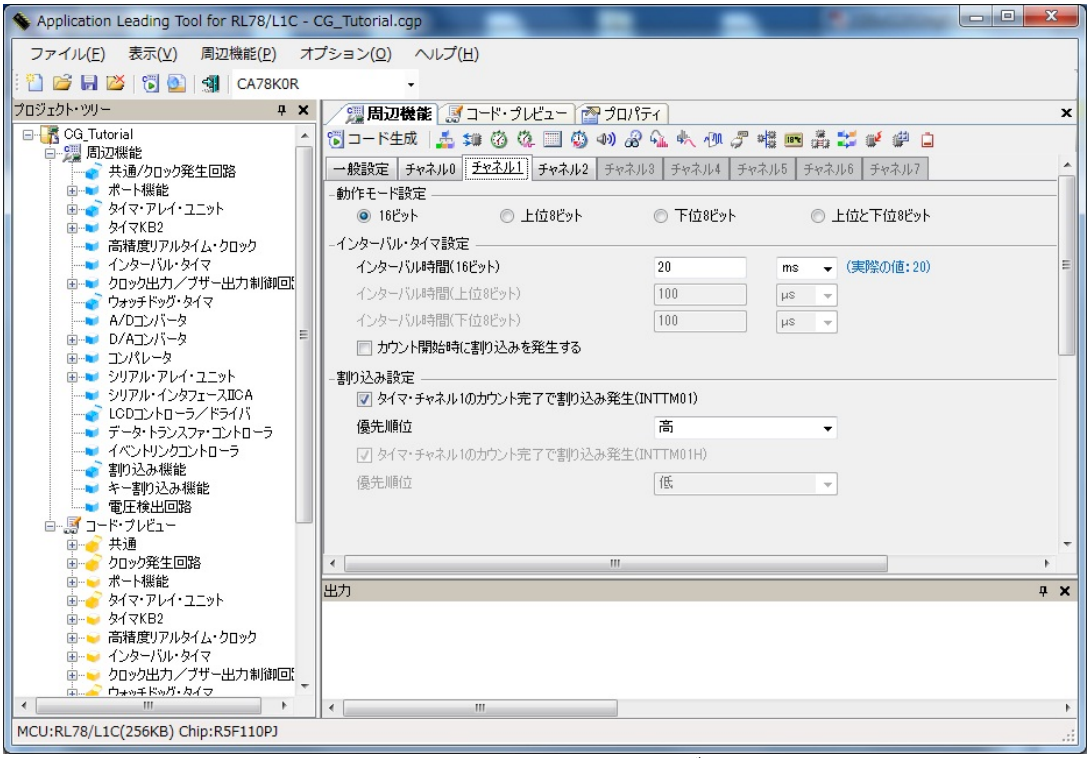


図 3-8: タイマ・アレイ・ユニットタブ(チャネル 1)

図 3-9 にチャネル 2 の設定を示します。チャネル 2 のインターバル・タイマ設定を 200ms 毎に割り込み(レベル高)が発生するようにします。このチャネルのタイマは、スイッチ長押しのデバウンスとして使用します。

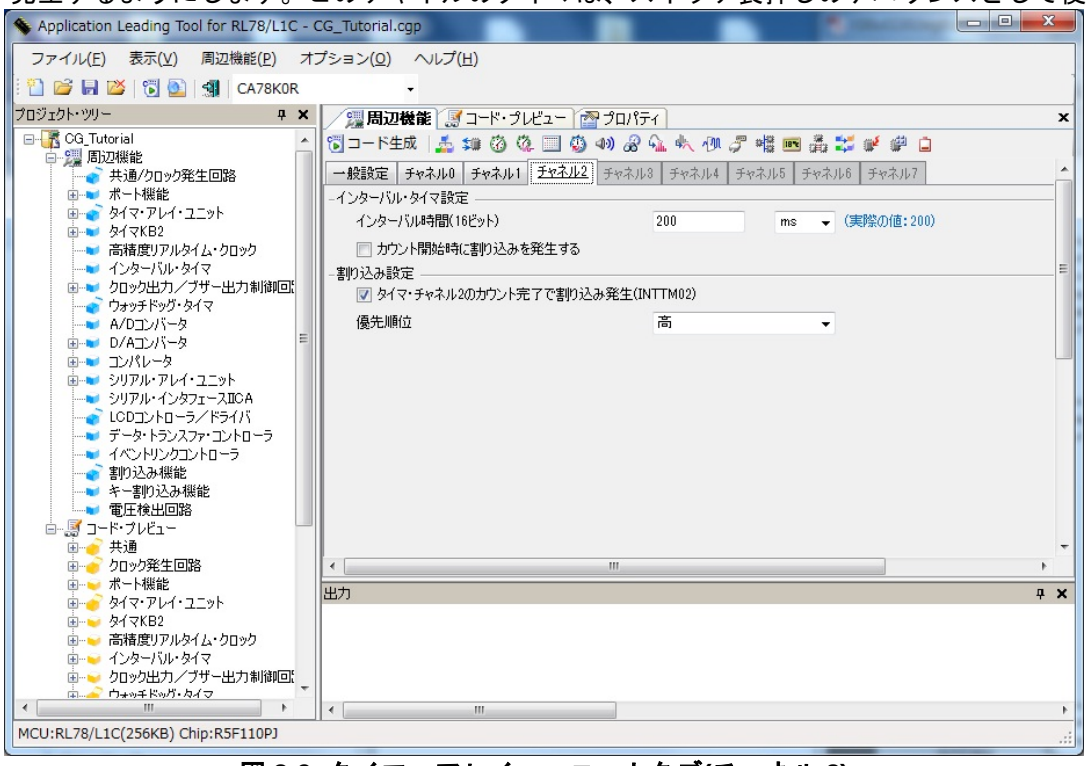


図 3-9: タイマ・アレイ・ユニットタブ(チャネル 2)

3.3.5 ウォッチドッグ・タイマ

ウォッチドッグ・タイマは、初期設定で有効になっています。”使用しない”に設定してください。

3.3.6 A/D コンバータ

図 3-10 に A/D コンバータの設定を示します。A/D コンバータは、CPU ボード上のポテンショメータから ANI0 端子に入力される電圧を分解能 12bit のワンショット・セレクト・モードで A/D 変換を行います。

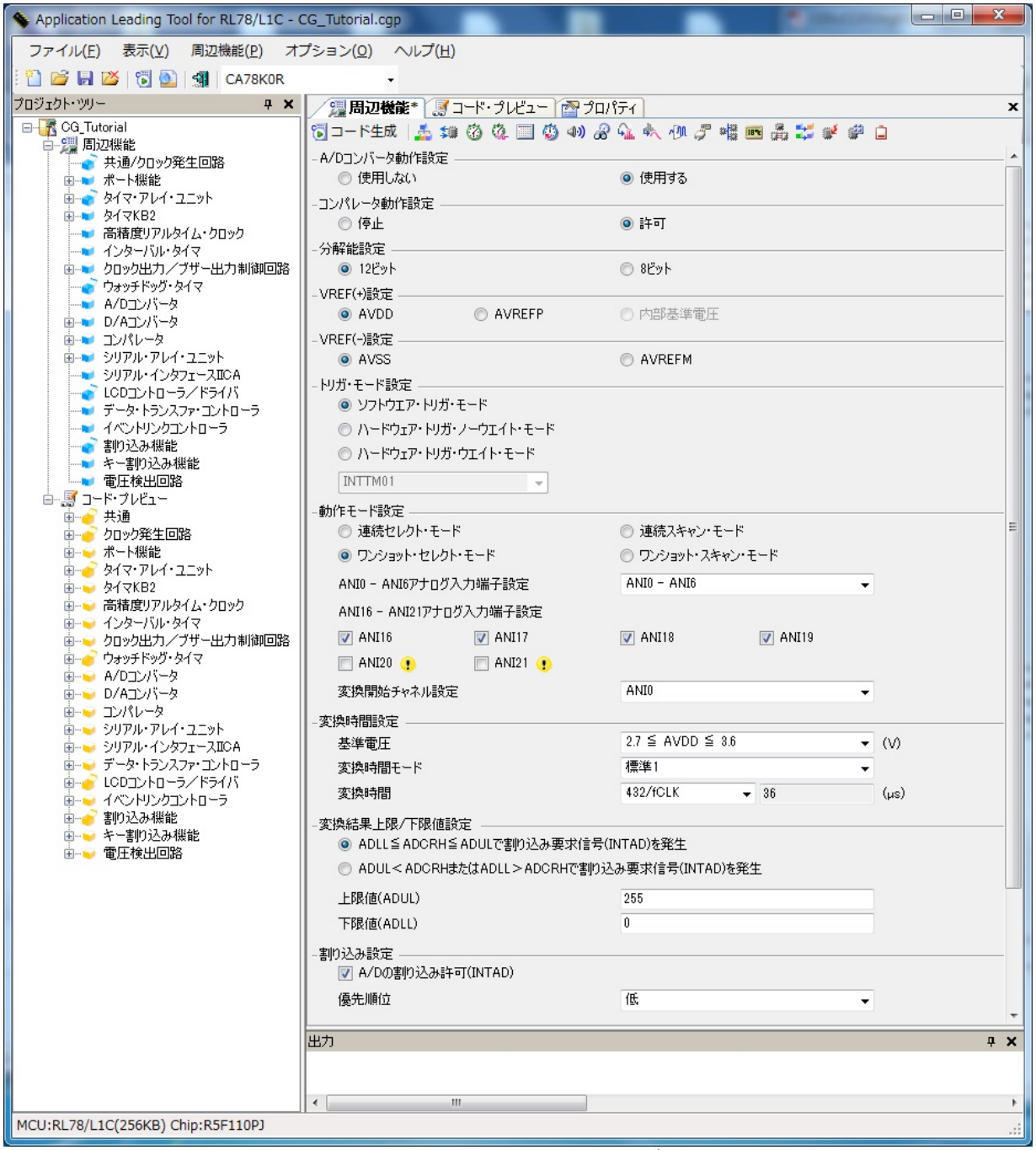


図 3-10: A/D コンバータタブ

3.3.7 シリアル・アレイ・ユニット

図 3-11 にシリアル・アレイ・ユニットの設定を示します。RSKRL78L1C の UART1（TXD1、RXD1）は、RS232 トランシーバに接続されています。RL78/L1C の UART1 は、シリアル・アレイ・ユニット 0 のチャネル 2 にあります。チャネル 2 の”UART1”および”送信/受信機能”を選択してください。

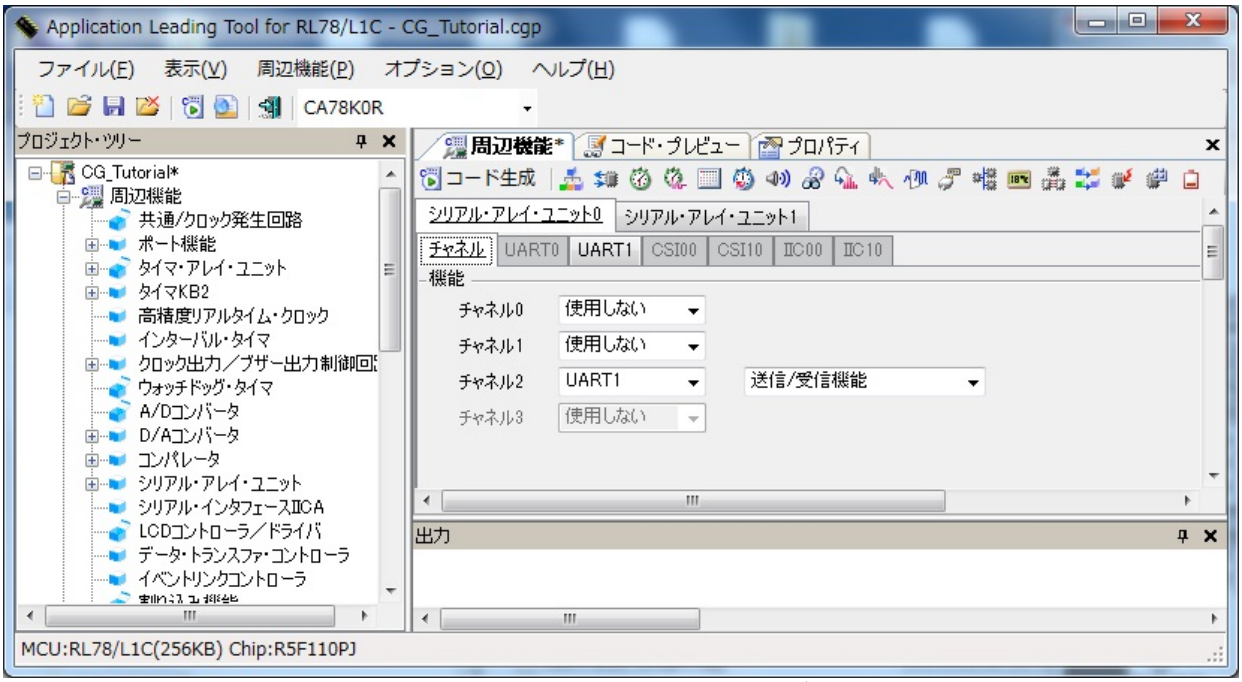


図 3-11: シリアル・アレイ・ユニットタブ(チャネル)

UART1 の設定を図 3-12 に示します。UART1 受信は、データビット長を 8 ビット、転送レートを 19200、パリティなしに設定してください。また、コールバック機能設定で受信完了とエラーに両方チェックが入っていることを確認してください。UART1 送信も同様の設定を行ってください。

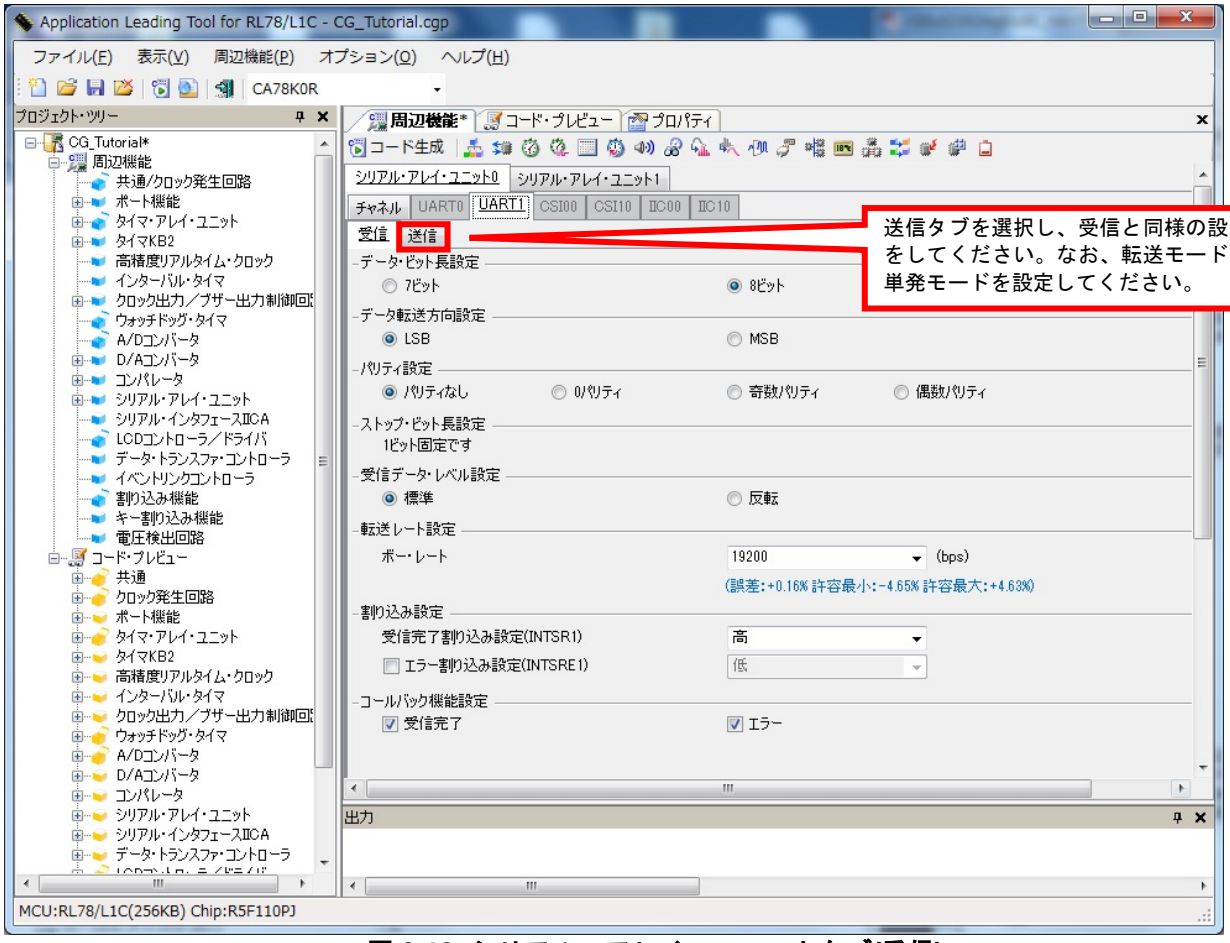


図 3-12: シリアル・アレイ・ユニットタブ(受信)

注意：送信タブも同様に設定してください。

3.3.8 ポート機能

図 3-13 にポートの設定を示します。CPU ボードは、LED0 が P05、LED1 が P07、LED2 が P41、LED3 が P42 に接続されています。Port0 の設定を図 3-13 に、Port4 の設定を図 3-14 に示します。また、P05、P07、P41、P42 は、”1 を出力”のチェックボックスにチェックが入っていることを確認してください。

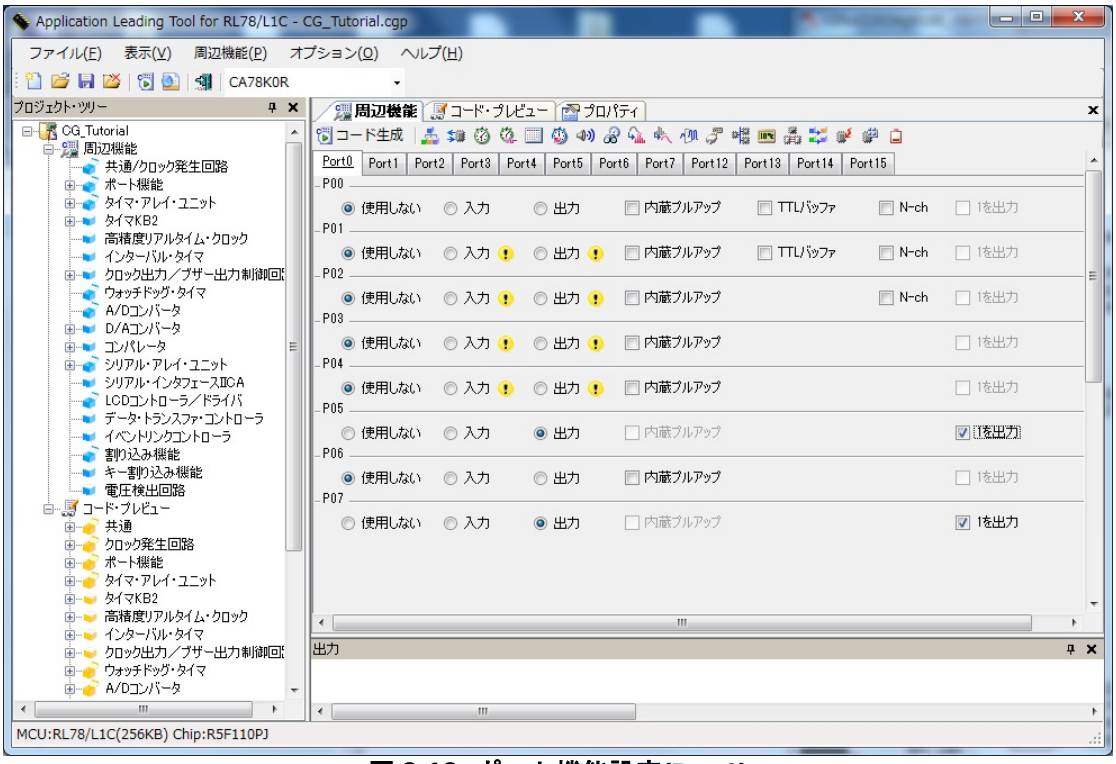


図 3-13: ポート機能設定(Port0)

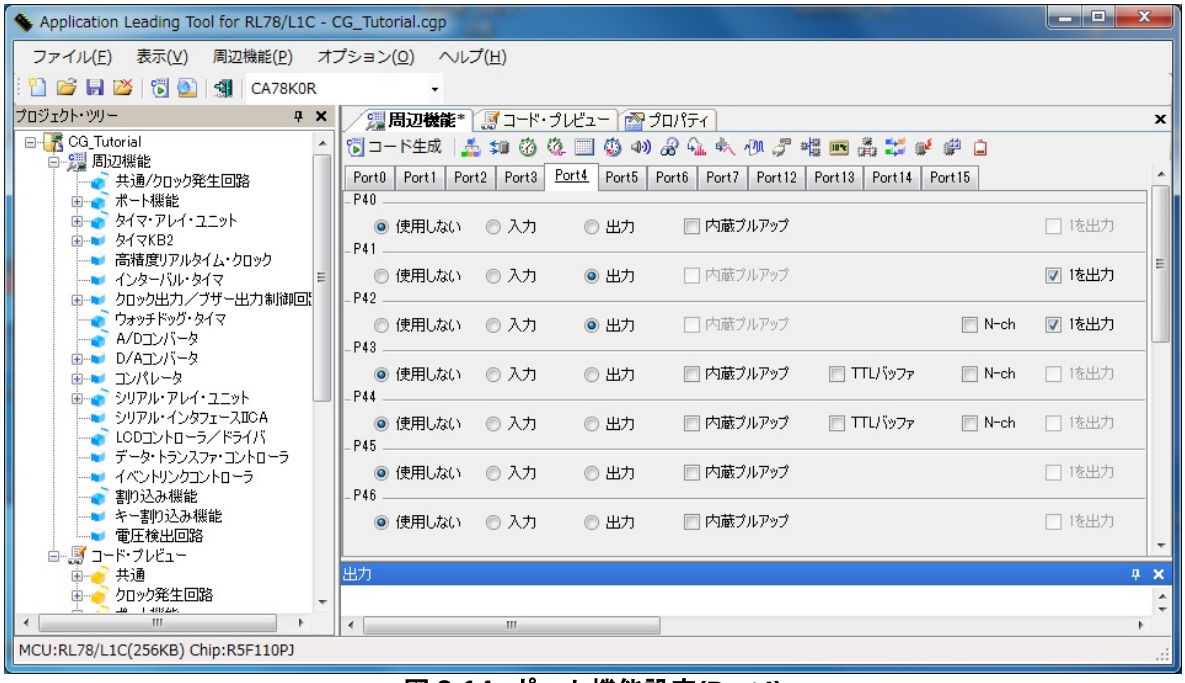


図 3-14: ポート機能設定(Port4)

これで周辺機能の設定は、全て完了しました。メニューバーの”ファイル(F)”からプロジェクトを保存してください。

次に、”コード生成”ボタン  をクリックして、コードを生成してください。図 3-15 の示すようにコードが生成されます。

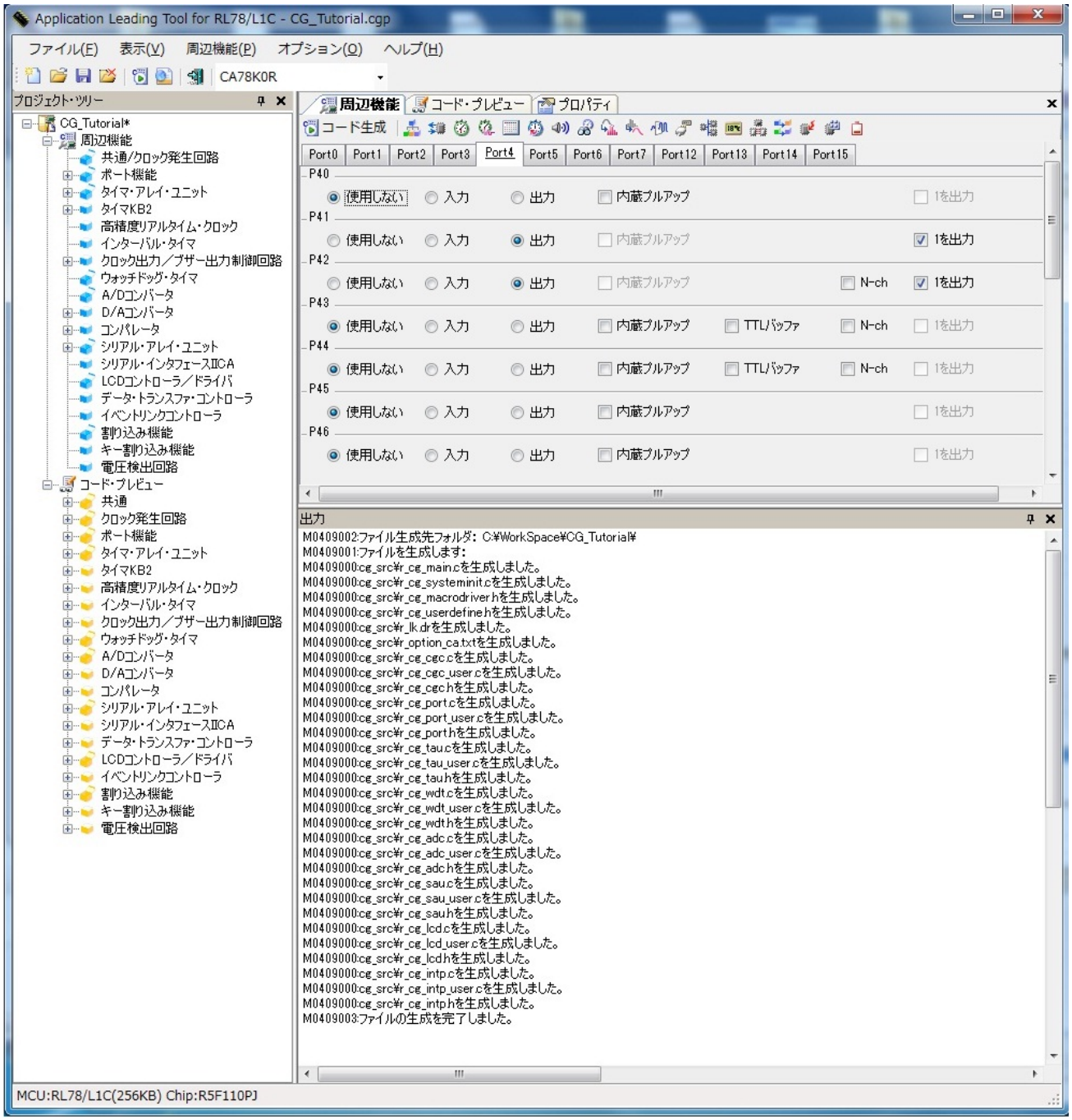
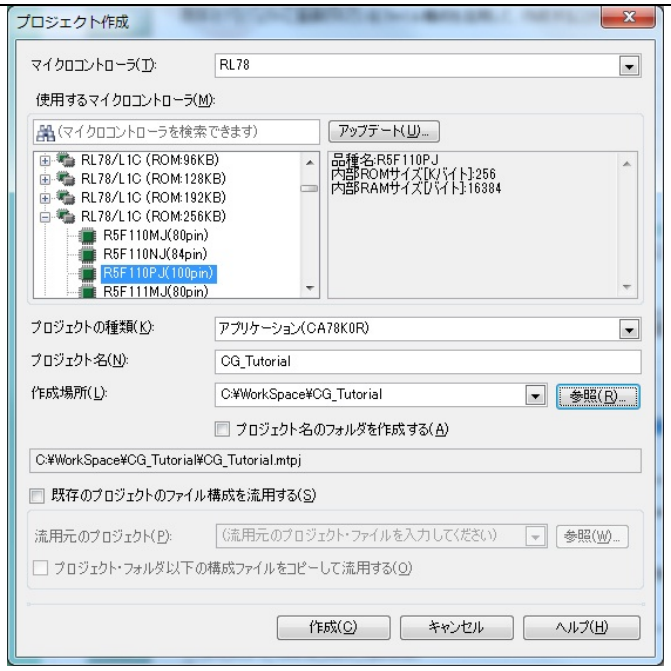
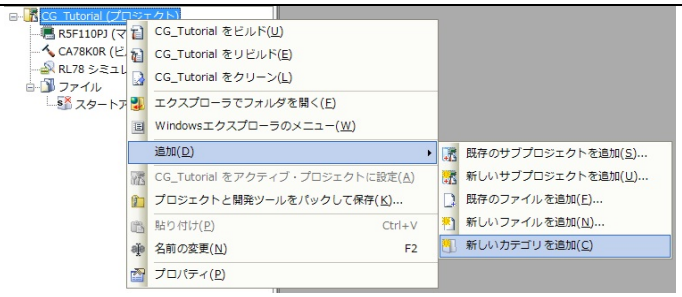
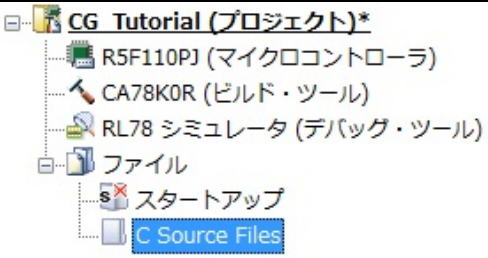


図 3-15: コード生成

4. CubeSuite+への組み込み

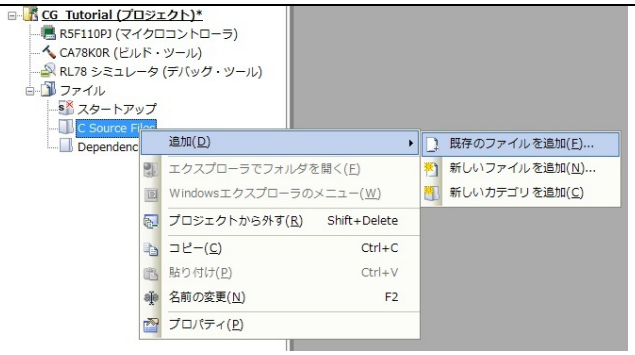
4.1 CubeSUite+へのコード組み込み

• CubeSuite+を起動します。CubeSuite+が起動しましたら、”スタートメニュー”ボタンを押してください。スタートタブが表示されましたら、”新しいプロジェクトを作成する”の”GO”ボタンを押して開始してください。	
• プロジェクト作成ダイアログが表示されましたら、”マイクロコントローラ(T)”のプルダウンメニューから RL78 を選択してください。 • 次に、”使用するマイクロコントローラ”から、RL78/L1C(ROM:256KB)を展開し、”R5F110PJ(100pin)”を選択してください。 • “プロジェクト名(N)”と”作成場所(L)”は、3 章で作成したプロジェクト名およびフォルダを入力してください。	
• カテゴリを作成します。 (1) プロジェクト・ツリーの CG_Tutorial プロジェクトを右クリックして、”追加”を選択、さらに”新しいカテゴリを追加”を選択してください。	
(2) 新しいカテゴリフォルダの名前を”C Source Files”に変更してください。 • 上記(1)、(2)と同様の方法で、”Dependencies”カテゴリフォルダも作成してください。	

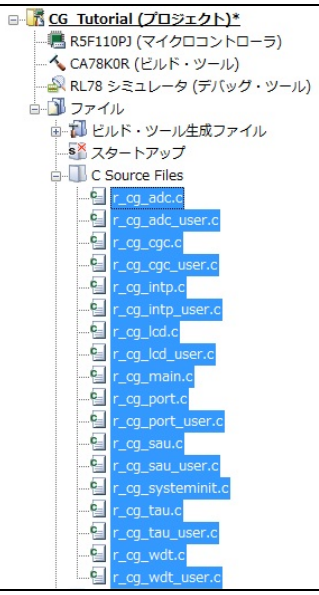
- “C Source Files”カテゴリに 3 章で作成した生成コードの C ファイルを追加します。

(1) “C Source Files”カテゴリを右クリックし、”追加”を選択し、さらに”既存のファイルを追加(F)”を選択してください。

(2) 選択するファイルは、”cg_src”フォルダ内の C ファイルを全て選択してください。

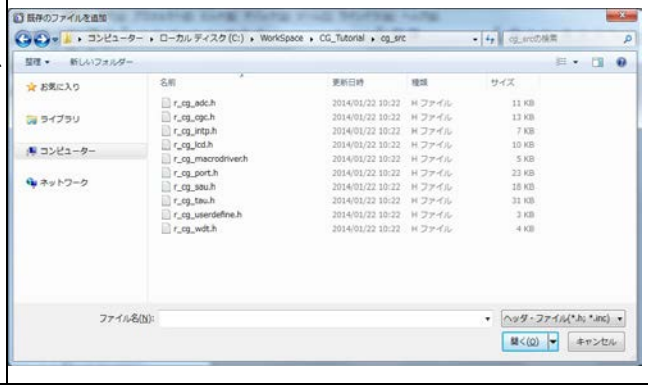


(3) “C Source Files”カテゴリに、C ソースファイルが追加されたことを確認してください。



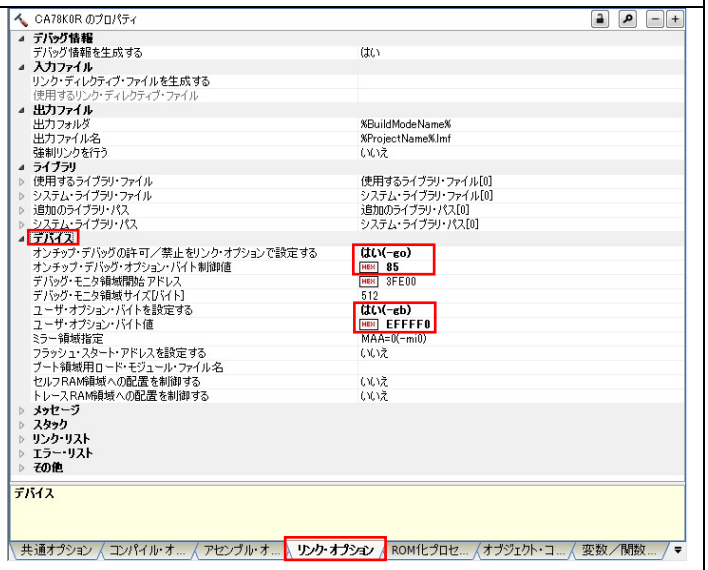
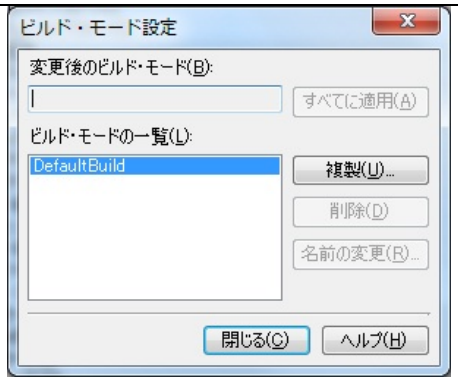
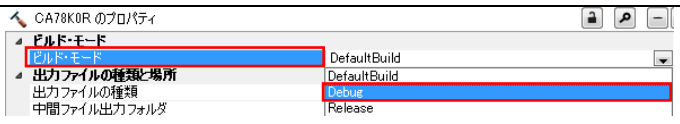
- 上記(1)～(3)と同様の方法で、”Dependencies”カテゴリに H ファイル、INC ファイルを追加してください。

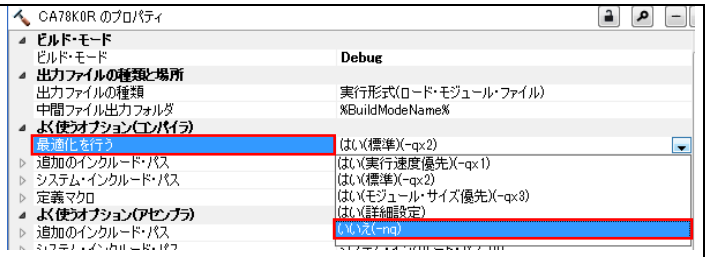
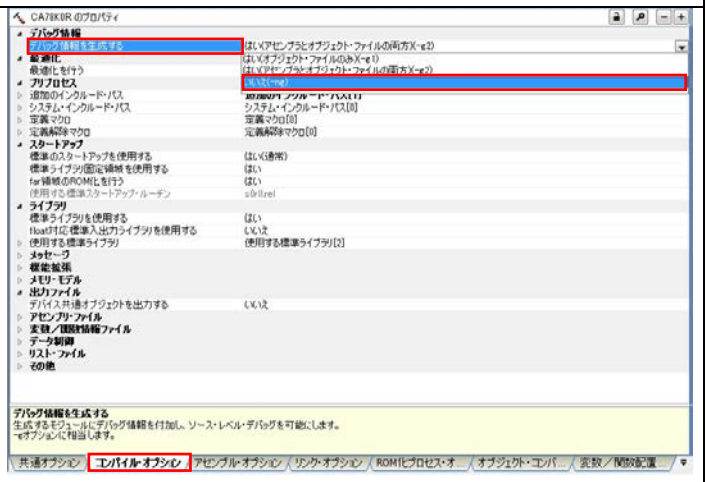
※ファイルフィルタでヘッダファイル (*.h; *.inc)をプルダウンから選択すると、容易にファイルを追加することができます。



- ツールバーの”プロジェクトをビルドします。”ボタンまたはキーボードの F7 ボタンを押してプロジェクトをビルドしてください。

4.2 プロジェクト設定

• プロジェクト・ツリーの”CA780KR(ビルド・ツール)”をダブルクリックして、プロパティを表示してください。 • 新規作成されたプロジェクトは、ビルド・モードに”DefaultBuild”が設定されています。また、標準で最適化が行われます。	
• 下のタブ”リンク・オプション”をクリックしてください。リンク・オプションタブ内の、”デバイス”項目を設定してください。 • オプション・デバッグの許可/禁止をリンク・オプションで設定する：はい(-go) • オンチップ・デバッグ・オプション・バイト制御値：85 • ユーザ・オプション・バイトを設定する：はい(-gb) • ユーザ・オプション・バイト値：EFFFF0	
• メニューバーから”ビルド(B)”を選択し、さらに”ビルド・モード設定(M)”を選択してください。 • ビルド・モード設定画面で、”複製(U)”をクリックしてください。 • “文字列(S)”に”Debug”と入力して”OK”をクリックしてください。 • ビルド・モードの一覧(L)に”Debug”が追加されていることを確認してください。	
• CA78K0R プロパティ画面の”共通オプション”タブのビルド・モードから”Debug”を選択してください。	

• 続けて、”最適化を行う”のプルダウンメニューから”いいえ(-nq)”を選択してください。 • これにより、ビルド・モード”Debug”では、コードの最適化が行われないようになります。	
• RSK の全てのサンプルコードプロジェクトは、3 つのビルド・モード(Default Build、Debug、Release)を選択できるようになっています。 • ビルド・モードに“Debug”を追加したように“Release”を追加してください。 • “Release”では、”最適化を行う”は初期設定の状態にしてください。次に、画面下のコンパイル・オプションタブの”デバッグ情報を生成する”のプルダウンメニューから”いいえ(-ng)”を選択してください。 • メニューバーの”ファイル(F)”から”すべてを保存(L)”を選択して、プロジェクトを保存してください。	

4.3 LCD パネルコードの統合

LCD パネルの API 機能は、RSK とともに提供されます。RSK の付属 DVD 内にある RSKRL78L1C_Tutorial プロジェクトの lcd_panel.h と lcd_panel.c です。
この 2 つのファイルを、C:¥Workspace¥CG_Tutorial¥cg_src のディレクトリにコピーしてください。
次に、コピーしたファイルをセクション 4.1 のように CubeSuite+で”C Source Files”カテゴリに lcd_panel.c を追加、”Dependencies”カテゴリに lcd_panel.h を追加してください。

カスタムコードを加える場合、以下に示すコメント文の間にカスタムコードを加えてください。

```
/* Start user code for _xxxxx_. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */
```

_XXXXX_は、特定のエリアで異なります。たとえば、インクルードファイルを定義する箇所では”include”、ユーザコード記載箇所では”function”、グローバル変数を定義する箇所では”global”と記述されています。
コメント分の間にカスタムコードを加えることにより、Applilet による上書きから保護することができます。

CubeSuite+プロジェクトのプロジェクト・ツリーの”C Source Files”カテゴリを展開して、‘r_cg_main.c’をダブルクリックして開いてください。次に、#include "lcd_panel.h"をコメント文の間に以下のように追加してください。

```
/* Start user code for include. Do not edit comment generated here */
#include "lcd_panel.h"
/* End user code. Do not edit comment generated here */
```

main 関数まで下にスクロールしてください。main 関数のユーザコードエリアコメント文の間に以下のようにコードを追加してください。

```
void main(void)
{
    R_MAIN_UserInit();
    /* Start user code. Do not edit comment generated here */

    /* Enable and configure LCD display. */
    Init_Display_Panel();

    /* Display the device family name on LCD.*/
    Display_Panel_String(PANEL_LCD_LINE1, " RL78");

    while (1U)
    {
        ;
    }
    /* End user code. Do not edit comment generated here */
}
```

メニューバーの”ビルド(B)”から”ビルド・プロジェクト(B)”または、”F7”キーを押してエラーがないことを確認してください。

4.4 スイッチコードの統合

スイッチの API 機能は、RSK とともに提供されます。RSK の付属 DVD 内にある RSKRL78L1C_Tutorial プロジェクトの rskrl78l1cdef.h、switch.h と switch.c です。
この 3 つのファイルを、C:¥Workspace¥CG_Tutorial¥cg_src のディレクトリにコピーしてください。
次に、コピーしたファイルをセクション 4.1 のように CubeSuite+で"C Source Files"カテゴリに switch.c を追加、"Dependencies"カテゴリに rskrl78l1cdef.h と switch.h を追加してください。

スイッチコードでは、スイッチの ON/OFF を検知する割り込み機能とデバウンス用のタイマ機能を使用します。そのため、セクション 3.3.2、3.3.4 で設定されたファイル r_cg_intp.h、r_cg_intp.c、r_cg_intp_user.c、r_cg_tau.h、r_cg_tau.c、r_cg_tau_user.c が必要となります。

4.4.1 割り込みコード

CubeSuite+プロジェクトのプロジェクト・ツリーの”Dependencies”カテゴリを展開して、‘r_cg_intp.h’をダブルクリックして開いてください。次に、ファイル最後尾のユーザコードエリアコメント文の間に以下のように追加してください。

```
/* Start user code for function. Do not edit comment generated here */

/* Function prototypes for detecting and setting the edge trigger of INTP0 */
uint8_t R_INTC0_IsFallingEdge(void);
void R_INTC0_SetFallingEdge(const uint8_t set_f_edge);
void R_INTC0_SetRisingEdge(const uint8_t set_r_edge);

/* Function prototypes for detecting and setting the edge trigger of INTP1 */
uint8_t R_INTC1_IsFallingEdge(void);
void R_INTC1_SetFallingEdge(const uint8_t set_f_edge);
void R_INTC1_SetRisingEdge(const uint8_t set_r_edge);

/* Function prototypes for detecting and setting the edge trigger of INTP2 */
uint8_t R_INTC2_IsFallingEdge(void);
void R_INTC2_SetFallingEdge(const uint8_t set_f_edge);
void R_INTC2_SetRisingEdge(const uint8_t set_r_edge);

/* End user code. Do not edit comment generated here */
```

次に、r_cg_intp.c を開いてファイル最後尾のユーザコメントエリアコメント文の間に以下のように追加してください。

```
/* Start user code for adding. Do not edit comment generated here */

/***********************************************************************
* Function Name: R_INTC0_IsFallingEdge
* Description  : This function returns 1 if the INTP0 is set to falling edge
*                triggered, otherwise 0.
* Arguments    : None
* Return Value : None
***********************************************************************/
uint8_t R_INTC0_IsFallingEdge (void)
{
    uint8_t falling_edge_trig = 0x0;

    if (EGN0 & _01_INTP0_EDGE_FALLING_SEL)
    {
        falling_edge_trig = 1;
    }

    return falling_edge_trig;

}
/***********************************************************************
* End of function R_INTC0_IsFallingEdge
***********************************************************************/

/***********************************************************************
* Function Name: R_INTC0_SetFallingEdge
* Description  : This function sets/clears the falling edge trigger for INTP0.
```

```
* Arguments    : uint8_t set_f_edge, 1 if setting falling edge triggered, 0 if
*                clearing
* Return Value : None
***********************************************************************************/
void R_INTC0_SetFallingEdge (const uint8_t set_f_edge)
{
    if (1 == set_f_edge)
    {
        EGN0 |= _01_INTP0_EDGE_FALLING_SEL;
    }
    else
    {
        EGN0 &= (uint8_t) ~_01_INTP0_EDGE_FALLING_SEL;
    }
}
/***********************************************************************************
* End of function R_INTC0_SetFallingEdge
***********************************************************************************/

/***********************************************************************************
* Function Name: R_INTC0_SetRisingEdge
* Description  : This function sets/clear the rising edge trigger for INTP0.
* Arguments    : uint8_t set_r_edge, 1 if setting rising edge triggered, 0 if
*                clearing
* Return Value : None
***********************************************************************************/
void R_INTC0_SetRisingEdge (const uint8_t set_r_edge)
{
    if (1 == set_r_edge)
    {
        EGP0 |= _01_INTP0_EDGE_RISING_SEL;
    }
    else
    {
        EGP0 &= (uint8_t) ~_01_INTP0_EDGE_RISING_SEL;
    }
}
/***********************************************************************************
* End of function R_INTC0_SetRisingEdge
***********************************************************************************/

/***********************************************************************************
* Function Name: R_INTC1_IsFallingEdge
* Description  : This function returns 1 if the INTP1 is set to falling edge
*                triggered, otherwise 0.
* Arguments    : None
* Return Value : None
***********************************************************************************/
uint8_t R_INTC1_IsFallingEdge (void)
{
    uint8_t falling_edge_trig = 0x0;

    if (EGN0 & _02_INTP1_EDGE_FALLING_SEL)
    {
        falling_edge_trig = 1;
    }

    return falling_edge_trig;

}
/***********************************************************************************
* End of function R_INTC1_IsFallingEdge
***********************************************************************************/

/***********************************************************************************
* Function Name: R_INTC1_SetFallingEdge
* Description  : This function sets/clears the falling edge trigger for INTP1.
* Arguments    : uint8_t set_f_edge, 1 if setting falling edge triggered, 0 if
*                clearing
* Return Value : None
***********************************************************************************/
void R_INTC1_SetFallingEdge (const uint8_t set_f_edge)
{
    if (1 == set_f_edge)
    {
        EGN0 |= _02_INTP1_EDGE_FALLING_SEL;
    }
```

```
    else
    {
        EGN0 &= (uint8_t) ~_02_INTP1_EDGE_FALLING_SEL;
    }
}
/***********************************************************************
* End of function R_INTC1_SetFallingEdge
***********************************************************************/

/***********************************************************************
* Function Name: R_INTC1_SetRisingEdge
* Description  : This function sets/clear the rising edge trigger for INTP1.
* Arguments    : uint8_t set_r_edge, 1 if setting rising edge triggered, 0 if
*                clearing
* Return Value : None
***********************************************************************/
void R_INTC1_SetRisingEdge (const uint8_t set_r_edge)
{
    if (1 == set_r_edge)
    {
        EGP0 |= _02_INTP1_EDGE_RISING_SEL;
    }
    else
    {
        EGP0 &= (uint8_t) ~_02_INTP1_EDGE_RISING_SEL;
    }
}
/***********************************************************************
* End of function R_INTC1_SetRisingEdge
***********************************************************************/

/***********************************************************************
* Function Name: R_INTC2_IsFallingEdge
* Description  : This function returns 1 if the INTP2 is set to falling edge
*                triggered, otherwise 0.
* Arguments    : None
* Return Value : None
***********************************************************************/
uint8_t R_INTC2_IsFallingEdge (void)
{
    uint8_t falling_edge_trig = 0x0;

    if (EGN0 & _04_INTP2_EDGE_FALLING_SEL)
    {
        falling_edge_trig = 1;
    }

    return falling_edge_trig;

}
/***********************************************************************
* End of function R_INTC2_IsFallingEdge
***********************************************************************/

/***********************************************************************
* Function Name: R_INTC2_SetFallingEdge
* Description  : This function sets/clears the falling edge trigger for INTP2.
* Arguments    : uint8_t set_f_edge, 1 if setting falling edge triggered, 0 if
*                clearing
* Return Value : None
***********************************************************************/
void R_INTC2_SetFallingEdge (const uint8_t set_f_edge)
{
    if (1 == set_f_edge)
    {
        EGN0 |= _04_INTP2_EDGE_FALLING_SEL;
    }
    else
    {
        EGN0 &= (uint8_t) ~_04_INTP2_EDGE_FALLING_SEL;
    }
}
/***********************************************************************
* End of function R_INTC2_SetFallingEdge
***********************************************************************/

/***********************************************************************
```

```
* Function Name: R_INTC2_SetRisingEdge
* Description  : This function sets/clear the rising edge trigger for INTP2.
* Arguments    : uint8_t set_r_edge, 1 if setting rising edge triggered, 0 if
*                clearing
* Return Value : None
***********************************************************************************/
void R_INTC2_SetRisingEdge (const uint8_t set_r_edge)
{
    if (1 == set_r_edge)
    {
        EGP0 |= _04_INTP2_EDGE_RISING_SEL;
    }
    else
    {
        EGP0 &= (uint8_t) ~_04_INTP2_EDGE_RISING_SEL;
    }
}
/***********************************************************************************
* End of function R_INTC2_SetRisingEdge
***********************************************************************************/
```

```
/* End user code. Do not edit comment generated here */
```

次に、r_cg_intp_user file.c を開いて、”include”ユーザコードエリアコメント文の間に以下のように追加してください。

```
/* Start user code for include. Do not edit comment generated here */
/* Defines switch callback functions required by interrupt handlers */
#include "switch.h"
/* End user code. Do not edit comment generated here */
```

同ファイルの r_intc0_interrupt 関数内のユーザコードエリアコメント文の間に以下のように追加してください。

```
    /* Start user code. Do not edit comment generated here */
    /* Switch 1 callback handler */
    Switch1IsrCallback();

    /* clear INTP0 interrupt flag */
    PIF0 = 0U;
    /* End user code. Do not edit comment generated here */
```

同ファイルの r_intc1_interrupt 関数内のユーザコードエリアコメント文の間に以下のように追加してください。

```
    /* Start user code. Do not edit comment generated here */
    /* Switch 2 callback handler */
    Switch2IsrCallback();

    /* clear INTP1 interrupt flag */
    PIF1 = 0U;
    /* End user code. Do not edit comment generated here */
```

同ファイルの r_intc2_interrupt 関数内のユーザコードエリアコメント文の間に以下のように追加してください。

```
/* Start user code. Do not edit comment generated here */
/* Switch 3 callback handler */
Switch3IsrCallback();

/* clear INTP2 interrupt flag */
PIF2 = 0U;
/* End user code. Do not edit comment generated here */
```

4.4.2 デバウンス用タイマコード

r_cg_tau_user.c を開いて、”include”ユーザコードエリアコメント文の間に以下のように追加してください。

```
/* Start user code for include. Do not edit comment generated here */
/* Defines switch callback functions required by interrupt handlers */
#include "switch.h"
/* End user code. Do not edit comment generated here */
```

同ファイルの r_tau0_channel1_interrupt 関数内の”function”ユーザコードエリアコメント文の間に以下のように追加してください。

```
/* Start user code. Do not edit comment generated here */
/* Stop this timer - we start it again in the de-bounce routines */
R_TAU0_Channel1_Stop();

/* Call the de-bounce call back routine */
SwitchDebounceIsrCallback();
/* End user code. Do not edit comment generated here */
```

同ファイルの r_tau0_channel2_interrupt 関数内の”function”ユーザコードエリアコメント文の間に以下のように追加してください。

```
/* Start user code. Do not edit comment generated here */
/* Stop this timer - we start it again in the de-bounce routines */
R_TAU0_Channel2_Stop();

/* Call the de-bounce call back routine */
SwitchDebounceIsrCallback();
/* End user code. Do not edit comment generated here */
```

4.4.3 A/D コンバータコードとメインスイッチコード

CubeSuite+プロジェクトのプロジェクト・ツリーの”Dependencies”カテゴリを展開して、‘r_cg_userdefine.h’をダブルクリックして開いてください。次に、ユーザコードエリアコメント文の間に以下のように追加してください。

```
/* Start user code for function. Do not edit comment generated here */
#define TRUE            (1)
#define FALSE           (0)

extern volatile uint8_t g_adc_trigger;
/* End user code. Do not edit comment generated here */
```

r_cg_main.c を開いて、”include”ユーザコードエリアコメント文の間に以下のように#include "switch.h"と"r_cg_adc.h"を追加してください。

```
/* Start user code for include. Do not edit comment generated here */
#include "lcd_panel.h"
#include "switch.h"
#include "r_cg_adc.h"
/* End user code. Do not edit comment generated here */
```

次に、ハイライト表示されているスイッチモジュール初期化関数を、main 関数内のユーザコードエリアコメント文の間に以下のように追加してください。

```
void main(void)
{
    R_MAIN_UserInit();
    /* Start user code. Do not edit comment generated here */

    /* Initialise the switch module */
    Switch_Init();

    /* Enable and configure LCD display. */
    Init_Display_Panel();

    /* Display the device family name on LCD.*/
    Display_Panel_String(PANEL_LCD_LINE1, " RL78");

    while (1U)
    {
        ;
    }
    /* End user code. Do not edit comment generated here */
}
```

同ファイルの”global”ユーザコードエリアコメント文の間に以下のように追加してください。

```
/* Start user code for global. Do not edit comment generated here */

/* Prototype declaration for cb_switch_press */
static void cb_switch_press (void);

/* Prototype declaration for get_adc */
static uint16_t get_adc(void);

/* Prototype declaration for lcd_display_adc */
static void lcd_display_adc (const uint16_t adc_result);

/* Variable for flagging user requested ADC conversion */
volatile uint8_t g_adc_trigger = FALSE;

/* End user code. Do not edit comment generated here */
```

main 関数内のユーザコードエリアコメント文の間に、ハイライト表示されているスイッチモジュールコールバック機能関数と while 文の中にコードを以下のように追加してください。

```
void main(void)
{
    R_MAIN_UserInit();
    /* Start user code. Do not edit comment generated here */

    /* Initialise the switch module */
    Switch_Init();

    /* Set the call back function when SW3 is pressed */
    SetSwitchPressCallback(cb_switch_press);

    /* Enable and configure LCD display. */
    Init_Display_Panel();

    /* Display the device family name on LCD.*/
    Display_Panel_String(PANEL_LCD_LINE1, " RL78");

    while (1U)
    {
        /* Wait for user requested A/D conversion flag to be set */
        if (TRUE == g_adc_trigger)
        {
            uint16_t adc_result;

            /* Call the function to perform an A/D conversion  */
            adc_result = get_adc();

            /* Display the result on the LCD */
            lcd_display_adc(adc_result);

           /* Reset the flag */
           g_adc_trigger = FALSE;
        }
    }
    /* End user code. Do not edit comment generated here */
}
```

続けて、cb_switch_press 関数、get_adc 関数、lcd_display_adc 関数をファイル最後尾のユーザコードエリアコメント文の間に、以下を追加してください。

```
/***********************************************************************
* Function Name : cb_switch_press
* Description   : Switch press callback function. Sets g_adc_trigger flag.
* Argument      : none
* Return value  : none
***********************************************************************/
static void cb_switch_press (void)
{
    /* Check if switch 3 was pressed */
    if (g_switch_flag & SWITCHPRESS_3)
    {
        /* set the flag indicating a user requested A/D conversion is required */
        g_adc_trigger = TRUE;

        /* Clear flag */
        g_switch_flag = 0x0;
    }
}
/***********************************************************************
* End of function cb_switch_press
***********************************************************************/

/***********************************************************************
* Function Name : get_adc
* Description   : Reads the ADC result, converts it to a string and displays
*                 it on the LCD panel.
* Argument      : none
* Return value  : uint16_t adc value
***********************************************************************/
```

```
static uint16_t get_adc (void)
{
    /* A variable to retrieve the adc result */
    uint16_t adc_result;

    /* Start a conversion */
    R_ADC_Start();

    /* Wait for the A/D conversion to complete */
    while (FALSE == g_adc_complete)
    {
        /* Wait */
    }

    /* Stop conversion */
    R_ADC_Stop();

    /* Clear ADC flag */
    g_adc_complete = FALSE;

    R_ADC_Get_Result(&adc_result);

    return adc_result;
}
/***********************************************************************
* End of function get_adc
***********************************************************************/

/***********************************************************************
* Function Name : lcd_display_adc
* Description   : Converts adc result to a string and displays
*                 it on the LCD panel.
* Argument      : uint16_t adc result
* Return value  : none
***********************************************************************/
static void lcd_display_adc (const uint16_t adc_result)
{
    /* Declare a temporary variable */
    uint8_t a;

    /* Declare temporary character string */
    char    lcd_buffer[4] = "XYZ";

    /* Convert ADC result into a character string, and store in the local.
       Casting to ensure use of correct data type. */
    a = (uint8_t)((adc_result & 0x0F00) >> 8);
    lcd_buffer[0] = (char)((a < 0x0A) ? (a + 0x30) : (a + 0x37));
    a = (uint8_t)((adc_result & 0x00F0) >> 4);
    lcd_buffer[1] = (char)((a < 0x0A) ? (a + 0x30) : (a + 0x37));
    a = (uint8_t)(adc_result & 0x000F);
    lcd_buffer[2] = (char)((a < 0x0A) ? (a + 0x30) : (a + 0x37));

    /* Display the contents of the local string lcd_buffer */
    Display_Panel_String(PANEL_LCD_LINE3, lcd_buffer);

}
/***********************************************************************
* End of function lcd_display_adc
***********************************************************************/

/* End user code. Do not edit comment generated here */
```

r_cg_adc.h を開いて、”function”ユーザコードエリアコメント文の間に以下のように追加してください。

```
/* Start user code for function. Do not edit comment generated here */
/* Flag indicates when serial transmission is in progress */
extern volatile uint8_t g_adc_complete;
/* End user code. Do not edit comment generated here */
```

r_cg_adc_user.c を開いて、”global”ユーザコードエリアコメント文の間に以下のように追加してください。

```
/* Start user code for global. Do not edit comment generated here */
volatile uint8_t g_adc_complete;
/* End user code. Do not edit comment generated here */
```

r_adc_interrupt 関数内のユーザコードエリアコメント文の間に以下のように追加してください。

```
void r_adc_interrupt(void)
{
    /* Start user code. Do not edit comment generated here */
    g_adc_complete = TRUE;
    /* End user code. Do not edit comment generated here */
}
```

メニューバーの”ビルド(B)”から”ビルド・プロジェクト(B)”または、”F7”キーを押してエラーがないことを確認してください。

5 章のデバッガ設定を行うことで次の動作を確認できるようになります。SW3 を押すことで、ポテンショメータから入力される電圧の A/D 変換値を LCD パネルに表示します。
UART ユーザコードを追加する場合は、再度ここから読み進めてください。

4.5 デバッグコードの統合

シリアルポートを介したデバッグトレースの API 機能は、RSK とともに提供されます。RSK の付属 DVD 内にある RSKRL78L1C_Tutorial プロジェクトの debug.h と debug.c です。
この 2 つのファイルを、C:¥Workspace¥CG_Tutorial¥cg_src のディレクトリにコピーしてください。
次に、コピーしたファイルをセクション 4.1 のように CubeSuite+で”C Source Files”カテゴリに debug.c を追加、”Dependencies”カテゴリに debug.h を追加してください。

debug.h を開いて、以下の記述があるか確認してください。

```
/* Macro for definition of serial debug transmit function - user edits this */
#define SerialDbgWrite R_UART1_Transmit
```

4.6 UART コードの統合

4.6.1 シリアル・アレイ・ユニットコード

CubeSuite+プロジェクトのプロジェクト・ツリーの”Dependencies”カテゴリを展開して、‘r_cg_sau.h’をダブルクリックして開いてください。次に、ファイル最後尾の”function”ユーザコードエリアコメント文の間に以下のように追加してください。

```
/* Start user code for function. Do not edit comment generated here */

/* Function prototype for R_UART1_Transmit */
MD_STATUS R_UART1_Transmit(uint8_t * const tx_buf, const uint16_t tx_num);

/* Flag indicates when serial transmission is in progress */
extern volatile uint8_t g_uart1_tx_busy;

/* Character is used to receive key presses from PC terminal */
extern uint8_t g_rx_char;

/* End user code. Do not edit comment generated here */
```

r_cg_sau.c を開いて、ファイル最後尾のユーザコードエリアコメント文の間に以下のように追加してください。

```
/* Start user code for adding. Do not edit comment generated here */
/***********************************************************************************
* Function Name: R_UART1_Transmit
* Description  : This function transmits data through UART1, but first waiting
*                for the tx_busy to clear.
* Arguments    : tx_buf -
*                    transfer buffer pointer
*                tx_num -
*                    buffer size
* Return Value : status -
*                    MD_OK or MD_ARGERROR
***********************************************************************************/
MD_STATUS R_UART1_Transmit (uint8_t * const tx_buf, const uint16_t tx_num)
{
    MD_STATUS status = MD_OK;

    if (tx_num < 1U)
    {
        status = MD_ARGERROR;
    }
    else
    {
        /* Wait for the g_uart1_tx_busy flag to clear, to avoid overwriting of
           the transmit buffer */
        while (g_uart1_tx_busy)
        {
            /* Wait */
        }

        /* Set the tx busy flag, this is cleared in the transmit end callback
           function */
        g_uart1_tx_busy = 1;

        /* Send the data using the R_UART1_Send function */
        R_UART1_Send(tx_buf, tx_num);
    }

    return (status);
}
/***********************************************************************************
* End of function R_UART1_Transmit
***********************************************************************************/
/* End user code. Do not edit comment generated here */
```

r_cg_sau_user.c を開いて、”global”ユーザコードエリアコメント文の間に以下のように追加してください。

```
/* Start user code for global. Do not edit comment generated here */

/* Flag indicates when serial transmission is in progress */
volatile uint8_t g_uart1_tx_busy = 0;

/* Global used to receive a character from the PC terminal */
uint8_t g_rx_char;

/* End user code. Do not edit comment generated here */
```

r_uart1_callback_receiveend 関数内のユーザコードエリアコメント文の間に以下のように追加してください。

```
static void r_uart1_callback_receiveend(void)
{
    /* Start user code. Do not edit comment generated here */

    /* Check the contents of g_rx_char */
    if (('c' == g_rx_char) || ('C' == g_rx_char))
    {
        g_adc_trigger = TRUE;
    }
    /* Set up UART1 receive buffer and callback function again */
    R_UART1_Receive((uint8_t *)&g_rx_char, 1);

     /* End user code. Do not edit comment generated here */
}
```

r_uart1_callback_sendend 関数内のユーザコードエリアコメント文の間に以下のように追加してください。

```
static void r_uart1_callback_sendend(void)
{
    /* Start user code. Do not edit comment generated here */

    /* Clear the g_uart1_tx_busy flag, this facilitates correct serialisation of
       transmit strings using the R_UART1_Transmit function */
    g_uart1_tx_busy = 0;

    /* End user code. Do not edit comment generated here */
}
```

4.6.2 メイン UART コード

r_cg_main.c を開いて、”include”ユーザコードエリアコメント文の間に以下を追加してください。

```
#include "debug.h"
```

“global”ユーザコードエリアコメント文の間に以下を追加してください。

```
/* Prototype declaration for uart_display_adc */
static void uart_display_adc(const uint8_t adc_count, const uint16_t adc_result);

/* Variable to store the A/D conversion count for user display */
uint8_t adc_count = 0;
```

main 関数内のユーザコードエリアコメント文の間に以下を追加してください。

```
void main(void)
{
    R_MAIN_UserInit();
    /* Start user code. Do not edit comment generated here */

    /* Initialise the switch module */
    Switch_Init();

    /* Set the call back function when SW3 is pressed */
    SetSwitchPressCallback(cb_switch_press);

    /* Enable and configure LCD display. */
    Init_Display_Panel();

    /* Display the device family name on LCD.*/
    Display_Panel_String(PANEL_LCD_LINE1, " RL78");

    /* Set up UART1 receive buffer and callback function */
    R_UART1_Receive((uint8_t *)&g_rx_char, 1);

    /* Enable UART1 operations */
    R_UART1_Start();

    while (1U)
    {
        /* Wait for user requested A/D conversion flag to be set */
        if (TRUE == g_adc_trigger)
        {
            uint16_t adc_result;

            /* Call the function to perform an A/D conversion  */
            adc_result = get_adc();

            /* Display the result on the LCD */
            lcd_display_adc(adc_result);

            /* Increment the adc_count */
            if (16 == ++adc_count)
            {
                adc_count = 0;
            }

            /* Send the result to the UART */
            uart_display_adc(adc_count, adc_result);

            /* Reset the flag */
            g_adc_trigger = FALSE;
        }
    }
    /* End user code. Do not edit comment generated here */
}
```

ファイル最後尾のユーザコードエリアコメント文の間に以下を追加してください。

```
/***********************************************************************
* Function Name : uart_display_adc
* Description   : Converts adc result to a string and sends it to the UART1.
* Argument      : uint8_t : adc_count
*                 uint16_t: adc result
* Return value  : none
***********************************************************************/
static void uart_display_adc (const uint8_t adc_count, const uint16_t adc_result)
{
    /* Declare a temporary variable */
        char a;

    /* Declare temporary character string */
    static char uart_buffer[] = "ADC xH Value: xxxH\r\n";

    /* Convert ADC result into a character string, and store in the local.
       Casting to ensure use of correct data type. */
    a = (char)(adc_count & 0x000F);
    uart_buffer[4] = (char)((a < 0x0A) ? (a + 0x30) : (a + 0x37));
    a = (char)((adc_result & 0x0F00) >> 8);
    uart_buffer[14] = (char)((a < 0x0A) ? (a + 0x30) : (a + 0x37));
    a = (char)((adc_result & 0x00F0) >> 4);
    uart_buffer[15] = (char)((a < 0x0A) ? (a + 0x30) : (a + 0x37));
    a = (char)(adc_result & 0x000F);
    uart_buffer[16] = (char)((a < 0x0A) ? (a + 0x30) : (a + 0x37));

    /* Send the string to the UART */
    DebugPrint(uart_buffer);

}
/***********************************************************************
* End of function uart_display_adc
***********************************************************************/
```

メニューバーの”ビルド(B)”から”ビルド・プロジェクト(B)”または、”F7”キーを押してエラーがないことを確認してください。

5 章のデバッガ設定を行っていれば、次の動作を確認できるようになります。まず、CPU ボードの RS232 シリアルコネクタと PC を RS232 ケーブルで接続してください。ターミナルソフトを起動して、セクション 3.3.7 の通信設定をターミナルソフトに設定してください。
プログラム動作開始後、SW3 を押すかターミナルソフト画面でキーボードの”c”キーを押すことで、ポテンショメータから入力される電圧の A/D 変換値を LCD パネルとターミナルソフト画面に表示します。
LED ユーザコードを追加する場合は、再度ここから読み進めてください。

4.7 LED コードの統合

r_cg_main.c を開いて、”include”ユーザコードエリアコメント文の間に以下を追加してください。

```
#include "rskrl78l1cdef.h"
```

“global”ユーザコードエリアコメント文の間に以下を追加してください。

```
/* Prototype declaration for led_display_count */
static void led_display_count(const uint8_t count);
```

main 関数内のユーザコードエリアコメント文の間に以下を追加してください。

```
void main(void)
{
    R_MAIN_UserInit();
    /* Start user code. Do not edit comment generated here */

    /* Initialise the switch module */
    Switch_Init();

    /* Set the call back function when SW3 is pressed */
    SetSwitchPressCallback(cb_switch_press);

    /* Enable and configure LCD display. */
    Init_Display_Panel();

    /* Display the device family name on LCD.*/
    Display_Panel_String(PANEL_LCD_LINE1, " RL78");

    /* Set up UART1 receive buffer and callback function */
    R_UART1_Receive((uint8_t *)&g_rx_char, 1);

    /* Enable UART1 operations */
    R_UART1_Start();

    while (1U)
    {
        /* Wait for user requested A/D conversion flag to be set */
        if (TRUE == g_adc_trigger)
        {
            uint16_t adc_result;

            /* Call the function to perform an A/D conversion  */
            adc_result = get_adc();


            /* Display the result on the LCD */
            lcd_display_adc(adc_result);

            /* Increment the adc_count and display using the LEDs */
            if (16 == ++adc_count)
            {
                adc_count = 0;
            }
            led_display_count(adc_count);

            /* Send the result to the UART */
            uart_display_adc(adc_count, adc_result);

            /* Reset the flag */
            g_adc_trigger = FALSE;
        }
    }
    /* End user code. Do not edit comment generated here */
}
```

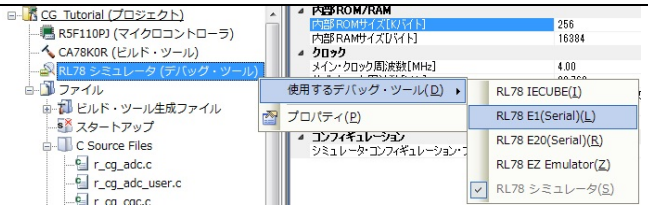
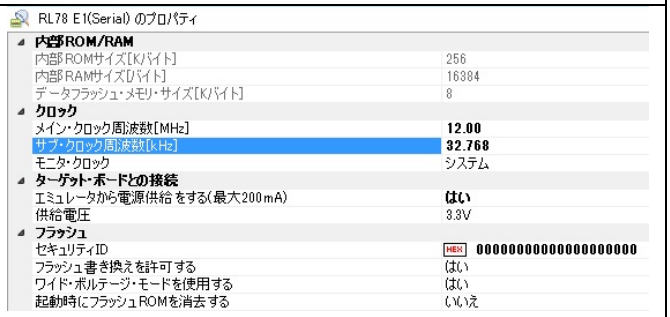
ファイル最後尾のユーザコードエリアコメント文の間に以下を追加してください。

```
/***********************************************************************
* Function Name : led_display_count
* Description   : Converts count to binary and displays on 4 LEDS0-3
* Argument      : uint8_t count
* Return value  : none
***********************************************************************/
static void led_display_count (const uint8_t count)
{
    /* Set LEDs according to lower nibble of count parameter */
    LED0 = (count & 0x01) ? LED_ON : LED_OFF;
    LED1 = (count & 0x02) ? LED_ON : LED_OFF;
    LED2 = (count & 0x04) ? LED_ON : LED_OFF;
    LED3 = (count & 0x08) ? LED_ON : LED_OFF;

}
/***********************************************************************
* End of function led_display_count
***********************************************************************/
```

メニューバーの”ビルド(B)”から”ビルド・プロジェクト(B)”または、”F7”キーを押してエラーがないことを確認してください。
5 章のデバッガ設定を行うことで次の動作を確認できるようになります。基本動作は同じですが、adc_count（A/D 変換カウント）をユーザ LED でバイナリ形式表示します。

5. プロジェクトデバッグ

• RL78 シミュレータ（デバッグ・ツール）を右クリックし、RL78 E1（Serial）を選択してください。	
• RL78 E1（Serial）を右クリックし、プロパティを選択してください。 • 接続用設定タブをクリックしてください。 • メイン・クロック周波数[MHz]：12 • サブ・クロック周波数[kHz]：32.768 • エミュレータから電源供給をする：はい • 供給電圧：3.3V	
• E1 をご使用のコンピュータの USB ポートに接続してください。 • ツールバーの’ダウンロード’ボタンをクリックしてください。	

6. チュートリアルコードの実行

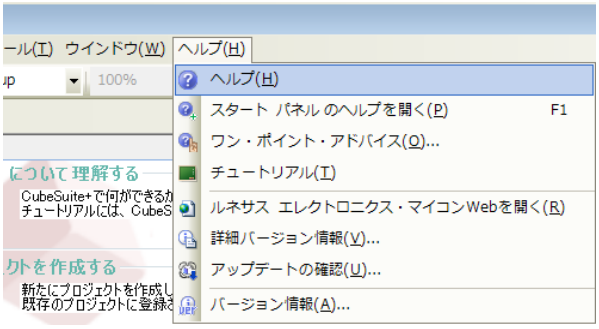
6.1 コードの実行

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。現在のプログラムカウンタ位置からプログラムを始めるため'実行'ボタンをクリックしてください。	

7. 追加情報

サポート

CubeSuite+の使用方法等の詳細情報は、CubeSuite+のヘルプメニューを参照してください。



RL78/L1C マイクロコントローラに関する詳細情報は、RL78/L1C 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、RL78 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/rskrl78l1c （日本サイト）
http://www.renesas.com/rskrl78l1c （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。

日本： csc@renesas.com
グローバル： 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	RSKRL78L1C コード生成支援ツールチュートリアルマニュアル (CubeSuite+)

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2014.01.15	ー	初版発行
1.01	2014.03.19	ー	「2. 略語および略称の説明」を更新
		23	「4.1 CubeSUite+へのコード組み込み」中の最後の説明文にビルド操作に関する説明を追加
		24	オンチップ・デバッグ・オプション・バイト制御値を追加
		32	"r_cg_adc.h.h"誤字修正
		33	cb_switch_press 関数を追加
		36	カテゴリ" C Source Files"修正
1.02	2014.04.04	ー	「2. 略語および略称の説明」を更新
		ー	目次を更新
		19	図 3-12 中に補足文を追加
		22	カテゴリ作成の説明文を更新
		22 – 25, 43, 44	一部の説明文、図に枠線を追加
		23	ファイル追加およびビルド操作の説明文を更新
		24, 25	一部の図に強調枠を追加
		26 - 42	追加/確認テキスト箇所に強調枠を追加
		27	「4.4 スイッチコードの結合」のカテゴリフォルダへのファイル追加方法に関する説明文を更新
		36	「4.5 デバッグコードの結合」のカテゴリフォルダへのファイル追加方法に関する説明文を更新
		41	「4.7 LED コードインテグレーション」を「4.7 LED コードの統合」に変更

RSKRL78L1C コード生成支援ツールチュートリアルマニュアル
(CubeSuite+)

発行年月日　2014 年 4 月 4 日 Rev.1.02

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 エレクトロニクス株式会社 〒100-0004 千代田区大手町2-6-2（日本ビル）

■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/contact/

RL78/L1C

R20UT2892JG0102

ルネサス エレクトロニクス株式会社