RL78 ファミリ

FPB ボードでスタンドアロン版 QE を用いたタッチアプリケーション開発

要旨

本アプリケーションノートでは、タッチ電極が搭載された RL78/G22 FPB (Fast Prototyping Board) (型名: RTK7RLG220C00000BJ) を使用し、静電容量タッチセンシングを応用したアプリケーションの作成に必要な手順を説明します。

本アプリケーションノートは、“CS+、スタンドアロン版スマート・コンフィグレータ、スタンドアロン版 QE for Capacitive Touch”を用いた開発ガイドです。

スタンドアロン版 QE は、デバイスや IDE に依存せずに開発を行うことができます。

別の開発環境として、静電容量タッチ評価システム (RTK0EG0042S01001BJJ) を使用し、“CS+、スタンドアロン版スマート・コンフィグレータ、スタンドアロン版 QE for Capacitive Touch”を用いて開発する場合は、以下のアプリケーションノートをご参照ください。

- RL78 ファミリ スタンドアロン版 QE を使用した静電容量タッチアプリケーションの開発 (R01AN6574)

また、スタンドアロン版 QE ではない別の開発環境として、“e² studio、プラグイン版スマート・コンフィグレータ、プラグイン版 QE for Capacitive Touch”を使用する場合は、以下のアプリケーションノートをご参照ください。

- RL78 ファミリ QE と SIS を使用した静電容量タッチアプリケーションの開発 (R01AN5512)

動作確認デバイス

RL78/G22

静電容量センサユニット (CTSU) をサポートする RL78 ファミリ

目次

1. システム概要

QE for Capacitive Touch は、静電容量式タッチセンサを使用した組み込みシステム開発に必要なタッチインタフェースの初期設定や感度調整に対応した開発支援ツールです。

QE for Capacitive Touch の主な機能は次のとおりです。

- タッチインタフェース構成の作成機能
  ボタン等のタッチインタフェースの配置とタッチセンサ (電極) の割当を視覚的に設定できます。

- チューニング機能
  タッチインタフェースのオフセットや感度を自動的にチューニングできます。

- モニタリングとパラメータ調整機能
  タッチインタフェースの動作をモニタリングでき、さらにパラメータの細かな調整を行うことができます。

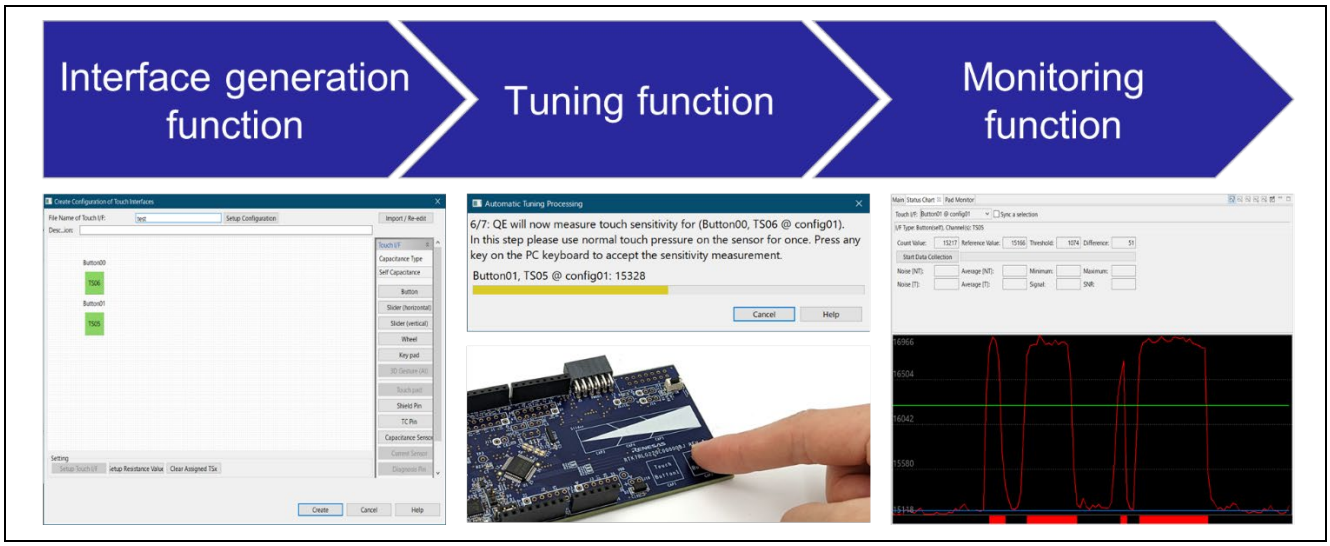


図 1-1 QE for Capacitive Touch の主な機能

2. 動作環境

本アプリケーションノートの動作環境は、表 2-1、表 2-2 のとおりです。

本アプリケーションノートでは、スタンドアロン版 QE にて生成したプログラムを CS+で RL78/G22 に書き込み、書き込んだプログラムを RL78/G22 で動作させます。

本アプリケーションノートは、RX/RA/RL78 ファミリ Renesas Synergy™プラットフォームの静電容量タッチ IP 搭載のデバイスに活用することができます。

表 2-1 動作環境 (ソフトウェア)

項目	内容	バージョン
統合開発環境 (IDE)	CS+ for CC	8.09.00 以降
コンパイラ	CC-RL	1.12.00 以降
QE	スタンドアロン版 QE for Capacitive Touch	3.2.0 以降
スマート・コンフィグレータ	RL78 スマート・コンフィグレータ	1.5.0 以降

注意　タッチセンサのチューニング時に、CC-RL 無償評価版の V1.12.00 以降のバージョンを使用してコンパイルする場合は、コンパイラの最適化レベルを”デバッグ優先 (-onothing)”を設定してビルドしてください。

表 2-2 動作環境 (ハードウェア)

項目	内容
使用マイコン	RL78/G22 (R7F102GGE2DFB)
ターゲットボード	RL78/G22 Fast Prototyping Board (RTK7RLG220C00000BJ)

3. 開発環境構築

ツールのインストールとボードの接続方法を説明します。

本アプリケーション例では、以下のツールを使用します。

- スタンドアロン版 QE for Capacitive Touch
- CS+
- スマート・コンフィグレータ

本アプリケーションノートでは、スタンドアロン版 QE for Capacitive Touch のインストールと、ボード接続について説明し、CS+、RL スマート・コンフィグレータのインストール手順は省略します。CS+、RL スマート・コンフィグレータをインストールしていない場合は、各ツールのインストール手順に従ってインストールしてください。

3.1 スタンドアロン版 QE for Capacitive Touch のインストール

スタンドアロン版 QE for Capacitive Touch をインストールします。

すでにインストール済みの場合は、本節は不要です。

1. ルネサス エレクトロニクスホームページから、”QE for Capacitive Touch”をダウンロードします。

2. ダウンロードした zip ファイルには、プラグイン版 QE とスタンドアロン版 QE が同梱されています。
   ダウンロードした zip ファイルを展開し、スタンドアロン版 QE をインストールします。
   このとき、Windows のパス名の文字数制限 (260 文字) を超えないようにドライブのルートに近い場所に展開してください。
   展開先の例：C:¥Renesas フォルダ以下

3.2 ボードの接続

PC とターゲットボードを接続します。

図 3-1 のように PC とターゲットボードを USB ケーブルで接続します。

本アプリケーション例では、ボードへの電源供給は USB を使用します。ターゲットボードの回路を確認し、必要に応じてスイッチやジャンパを設定してください。

本アプリケーション例では、ターゲットボードのジャンパを以下のように設定します。

- J16 : QE シリアル接続切り替えジャンパ
  QE のシリアル接続機能を実行する場合 オープン
  COM Port デバッグを実行する場合 ショート
- J17 : 電源選択ヘッダ 1-2 ショート (5V 電源選択)

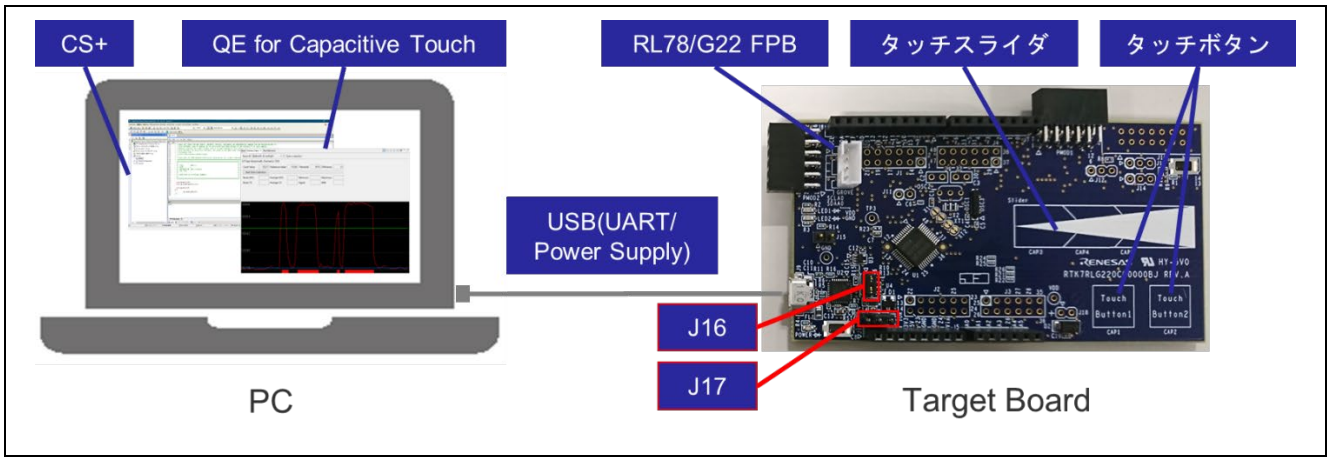


図 3-1 PC とターゲットボードの接続

4. アプリケーション開発手順

アプリケーション開発の手順を説明します。

開発の流れは QE for Capacitive Touch のワークフローに従います。

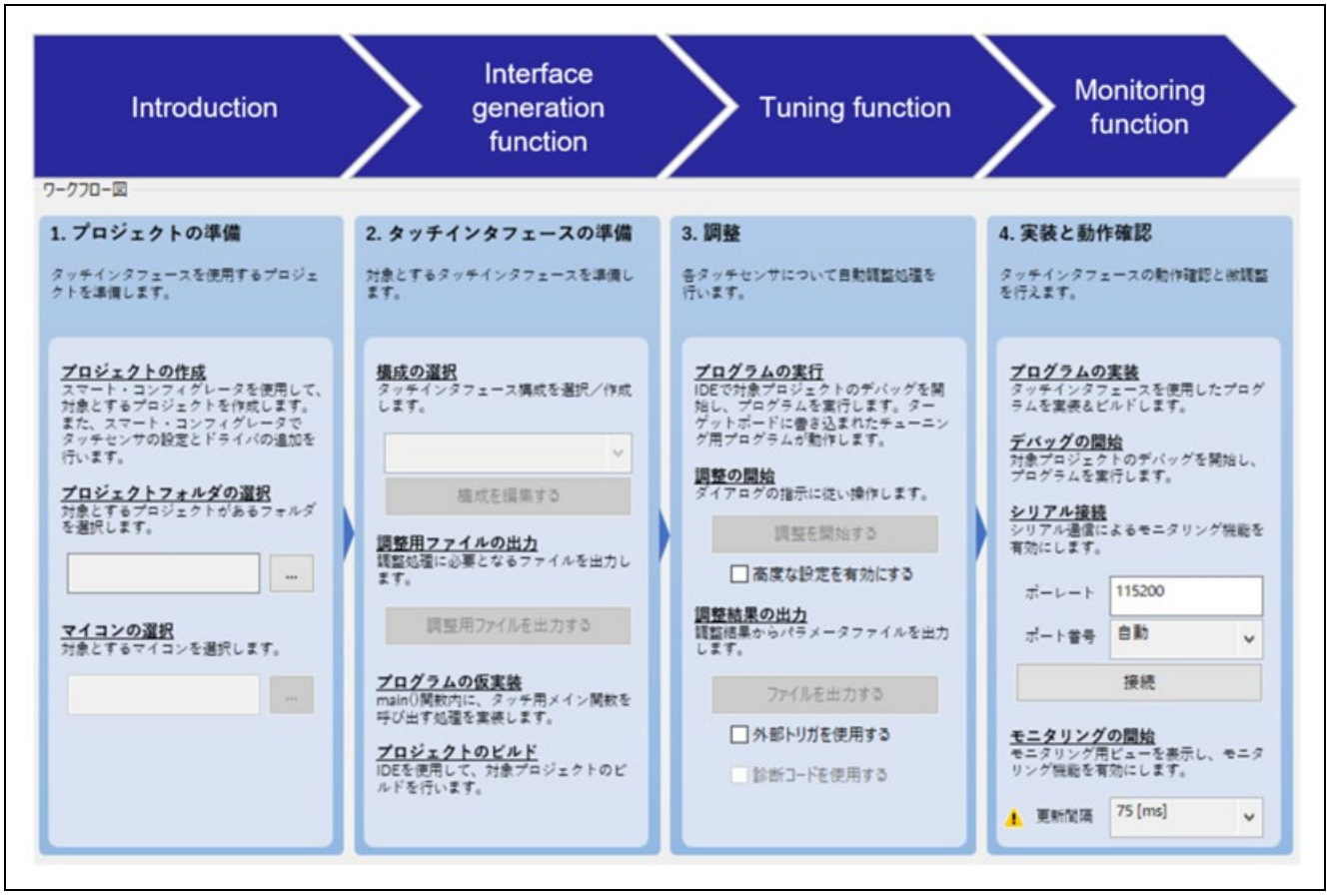


図 4-1 アプリケーション開発手順

ワークフローの各項目を表 4-1 に示します。本表の章番号は、関連章にリンクされています。章番号をクリックして使い方を確認してください。プロジェクトの作成、プログラムの実装、プロジェクトのビルド、デバッグの開始は、IDE やスマート・コンフィグレータを使います。

表 4-1 QE for Capacitive Touch の項目

項目				章番号
プロジェクトの準備	プロジェクトの作成	IDE によるプロジェクトの作成		6
		スマート・コンフィグレータの設定		7
			クロックとシステム	7.2
			CTSU 用ドライバ	7.3.2
			タッチ用ミドルウェア	7.3.3
			シリアルインタフェース (UART 通信)	7.4
			未使用端子の Low レベル出力	7.5
	プロジェクトフォルダの選択			8.2
	マイコンの選択			
タッチインタフェースの準備	構成の選択			8.3
	調整用ファイルの出力			
	プログラムの仮実装			
	プロジェクトのビルド			
調整	プログラムの実行			8.4
	調整の開始			
	調整結果の出力			
実装と動作確認	プログラムの実装			8.5
	デバッグの開始			
	シリアル接続			
	モニタリングの開始			

5. アプリケーション例

5.1 アプリケーション例の概要

本アプリケーションノートでは、2 つのボタンおよび 1 つのスライダを使用するアプリケーションを例に説明します。

6 章以降で、2 つのボタンおよび 1 つのスライダを使用するアプリケーションを作成し、ボタンまたはスライダをタッチした場合の検出状況をモニタリングする方法を示します。

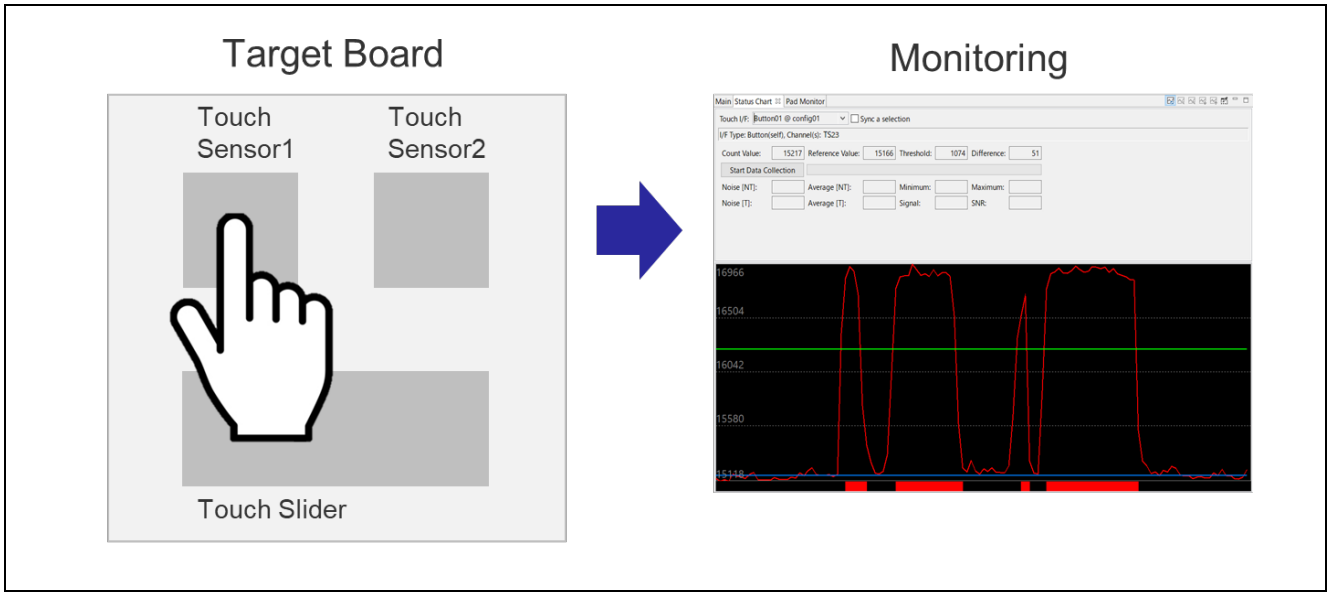


図 5-1 アプリケーション例

5.2 使用端子一覧

本アプリケーション例の使用端子を、表 5-1 に示します。

UART 通信およびタッチセンサは、使用するターゲットボードの仕様にあわせて設定する必要があります。

表 5-1 本アプリケーション例の使用端子一覧

項目	端子	用途
UART 通信	RxD0/P11	チューニング モニタリング
	TxD0/P12	
タッチセンサ 1	TS24/P26	ボタン (TS_B1)
タッチセンサ 2	TS23/P25	ボタン (TS_B2)
タッチスライダ	TS20/P22	スライダ (TS_S)
	TS21/P23	
	TS22/P24	

本アプリケーション例で使用するタッチセンサの配置を図 5-2 に示します。

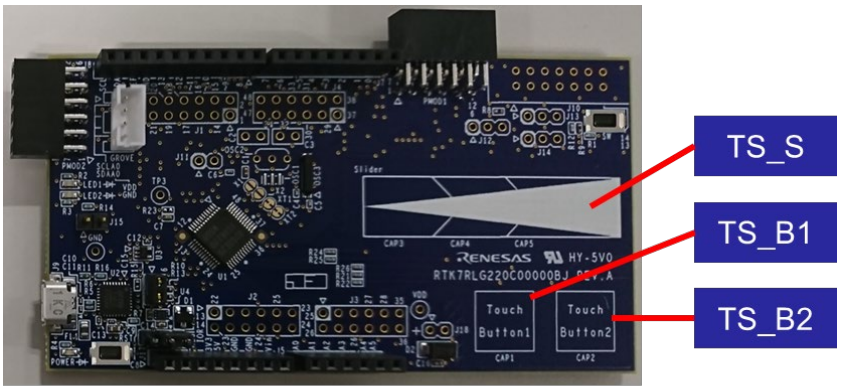


図 5-2 タッチセンサの配置

6. プロジェクトの作成

CS+を起動し、プロジェクトを作成します。

“プロジェクト作成”ダイアログでは、以下を選択します。

- マイクロコントローラ(T) : RL78
- 使用するマイクロコントローラ(M) : R7F102GGExFB (48pin)
- プロジェクトの種類 : アプリケーション (CC-RL)
- プロジェクト名 : (任意のプロジェクト名)
- 作成場所 : (任意の作成場所)

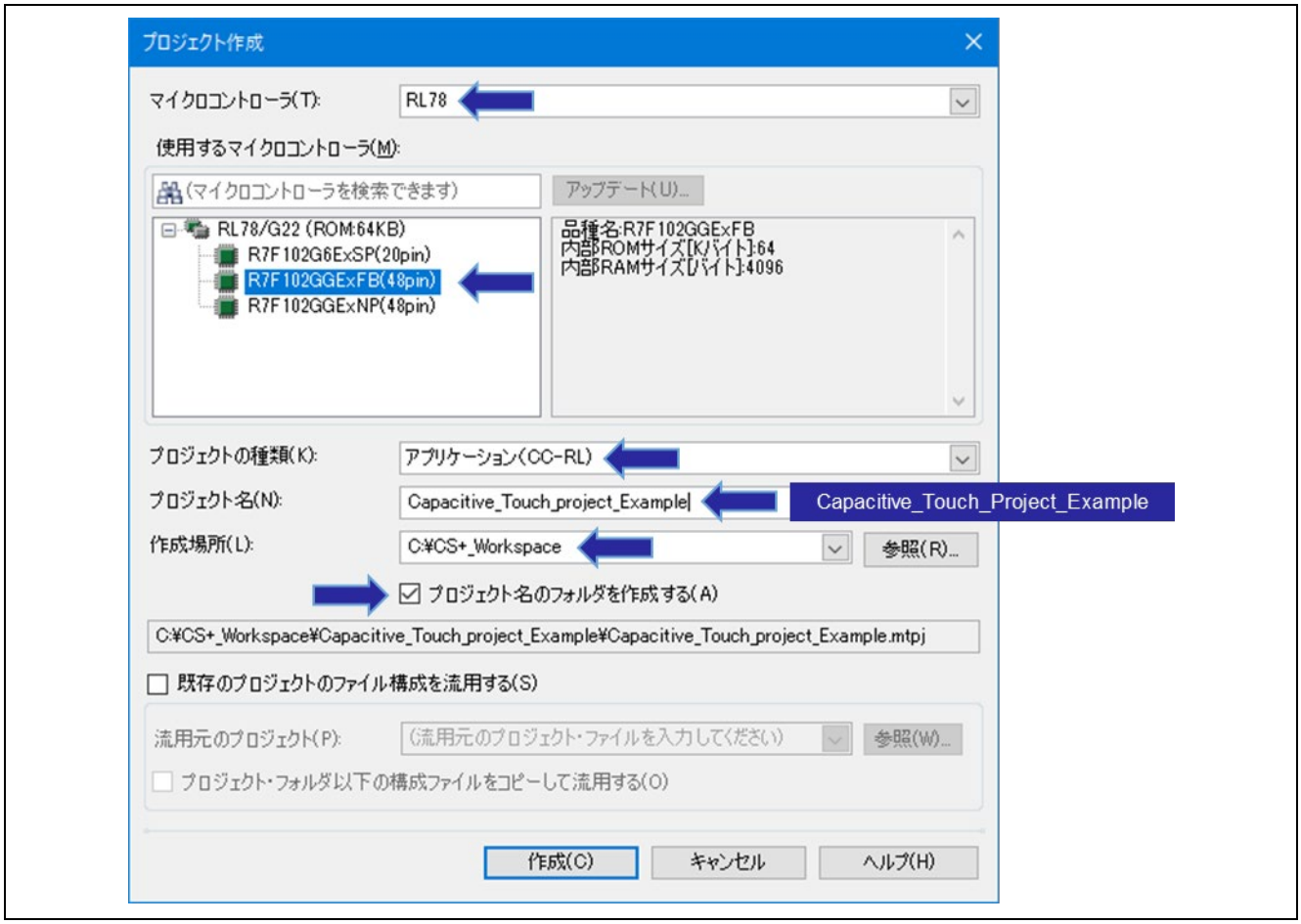


図 6-1 プロジェクトの作成

7. スマート・コンフィグレータの設定

スマート・コンフィグレータの設定手順を説明します。本アプリケーション例で必要な設定は、以下のとおりです。

- クロックとシステム
- CTSU 用ドライバ
- タッチ用ミドルウェア
- シリアルインタフェース (UART 通信)
- 未使用端子の Low レベル出力

7.1 スマート・コンフィグレータの起動

CS+の”プロジェクト・ツリー”で、”スマート・コンフィグレータ”をダブルクリックし、起動します。

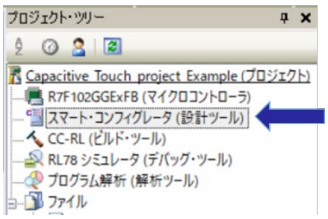


図 7-1 スマート・コンフィグレータを起動

上記方法でスマート・コンフィグレータが起動しない場合は、以下の 2 点を確認してください。

- スマート・コンフィグレータのプロパティで”ファイルパス”が正しく設定されているか
- メニューの [ツール] – [プラグインの管理]から、”RL78 用スマート・コンフィグレータ通信プラグイン”がチェックされているか

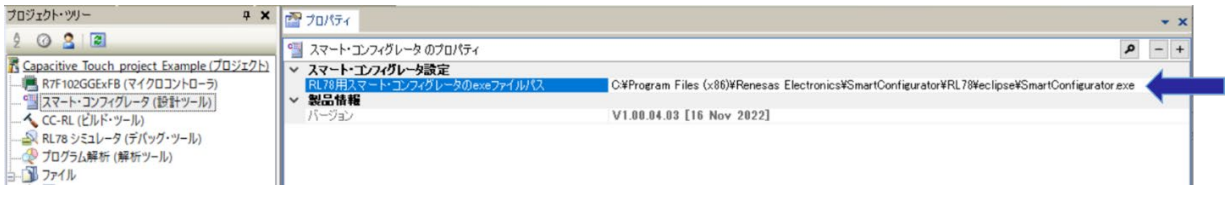


図 7-2 スマート・コンフィグレータのファイルパス

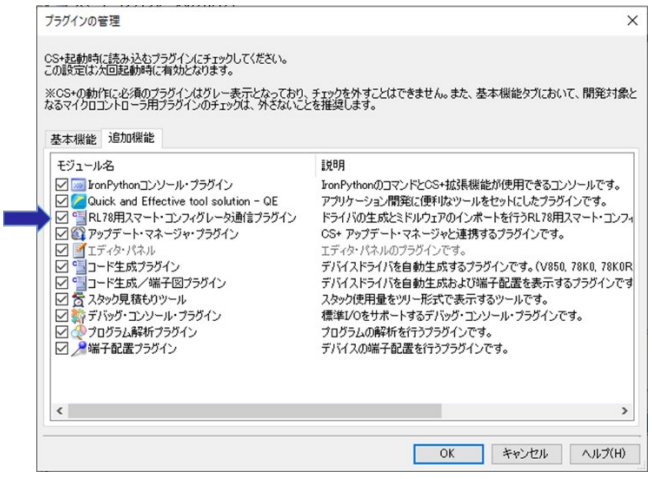


図 7-3 プラグインの管理

7.2　クロックとシステムの設定

クロックとシステムの設定手順を説明します。

1. スマート・コンフィグレータの中央下部のメニューから[クロック]タブを選択し、クロックを設定します。

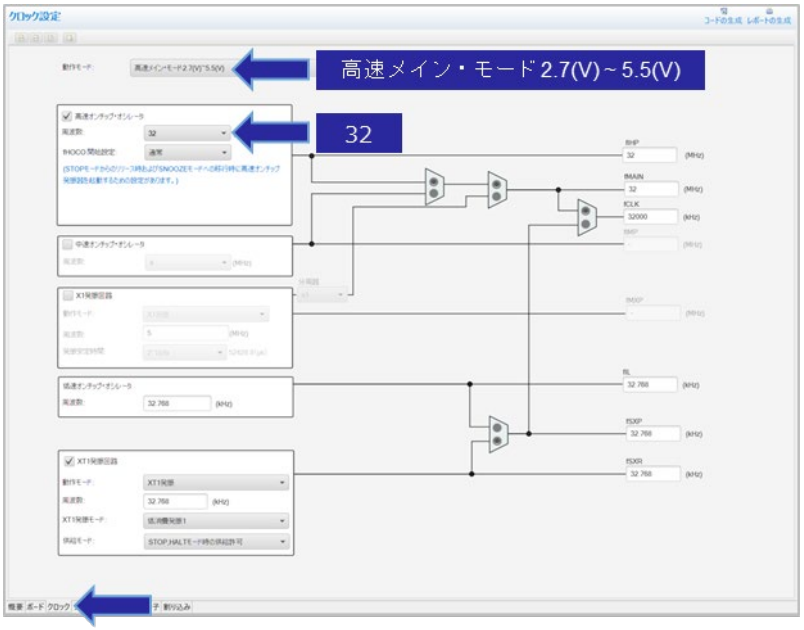


図 7-4 クロックの設定

2. [システム]タブを選択し、デバッグ環境を設定します。

図 7-5 デバッグの設定

7.3 SIS (Software Integration System) モジュールの設定

QE for Capacitive Touch を使用するために必要な”CTSU 用ドライバ”と”タッチ用ミドルウェア”の 2 つの SIS モジュールの追加方法と、設定手順を説明します。

7.3.1 SIS モジュールのダウンロード

スマート・コンフィグレータで、”CTSU 用ドライバ”と”タッチ用ミドルウェア”をダウンロードします。すでにダウンロード済みの場合は、本項は不要です。

1. [コンポーネント]タブを選択し、 アイコンをクリックします。

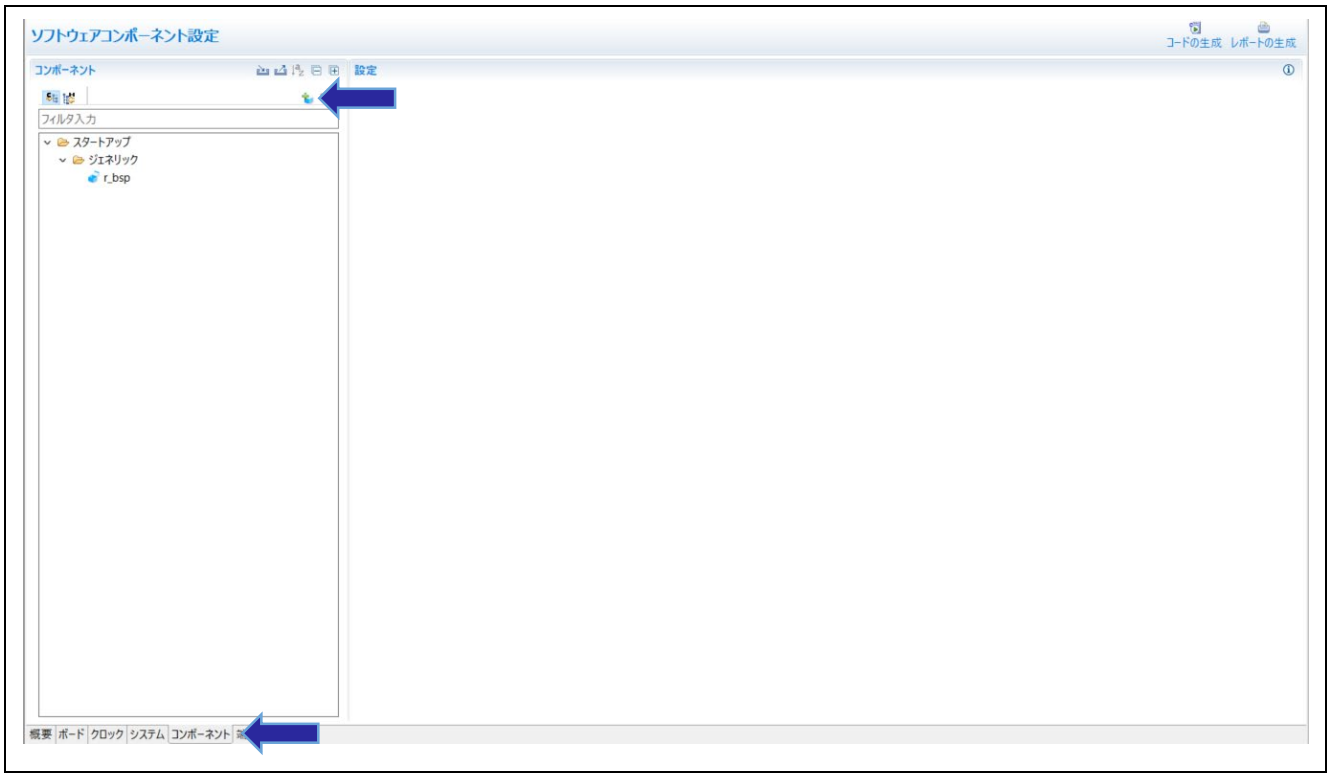


図 7-6 ソフトウェアコンポーネント設定画面

2. “コンポーネントの追加”ダイアログの下側にある”RL78 Software Integration System モジュールをダウンロードする”をクリックします。

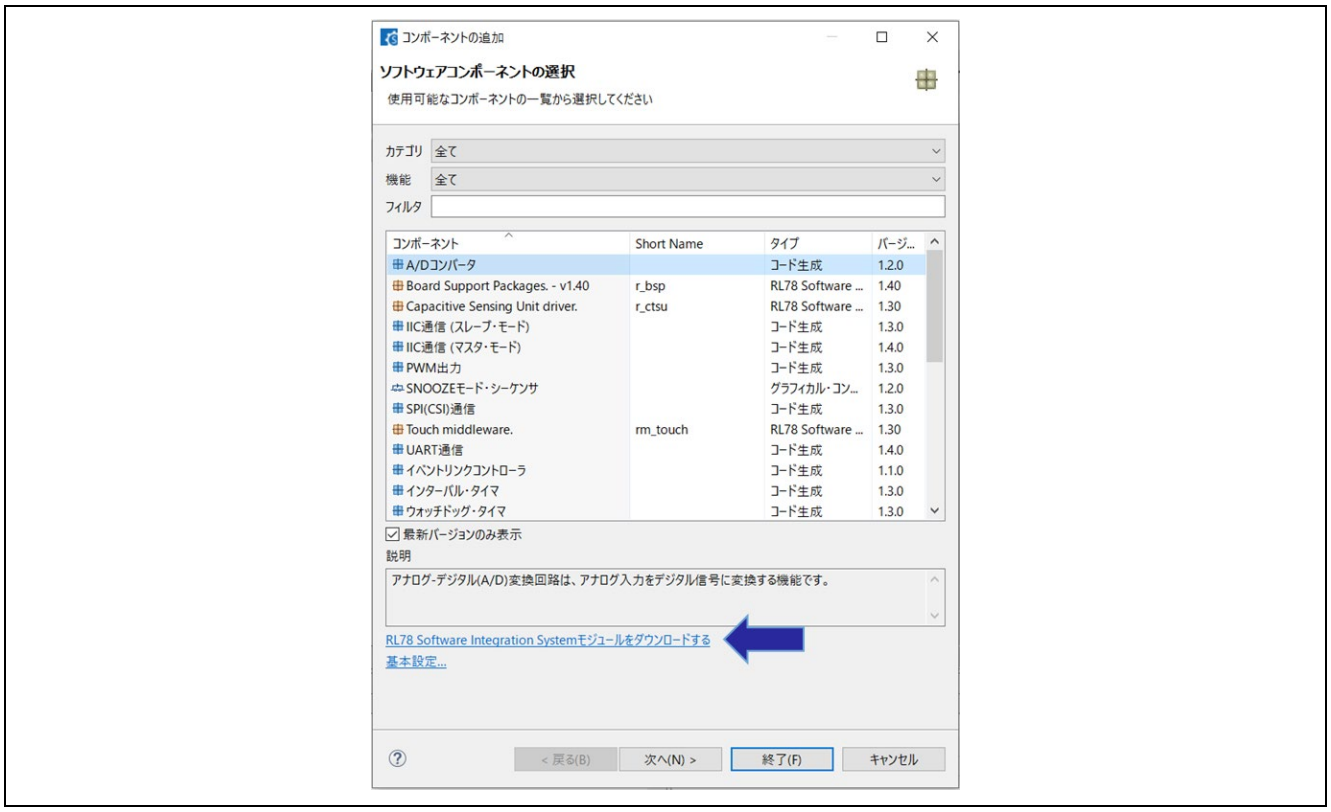


図 7-7 コンポーネントの追加ダイアログ

3. 表示されたダイアログで以下を選択し、ダウンロードをクリックします。
   — RL78 ファミリ CTSU モジュール Software Integration System
   — RL78 ファミリ TOUCH モジュール Software Integration System

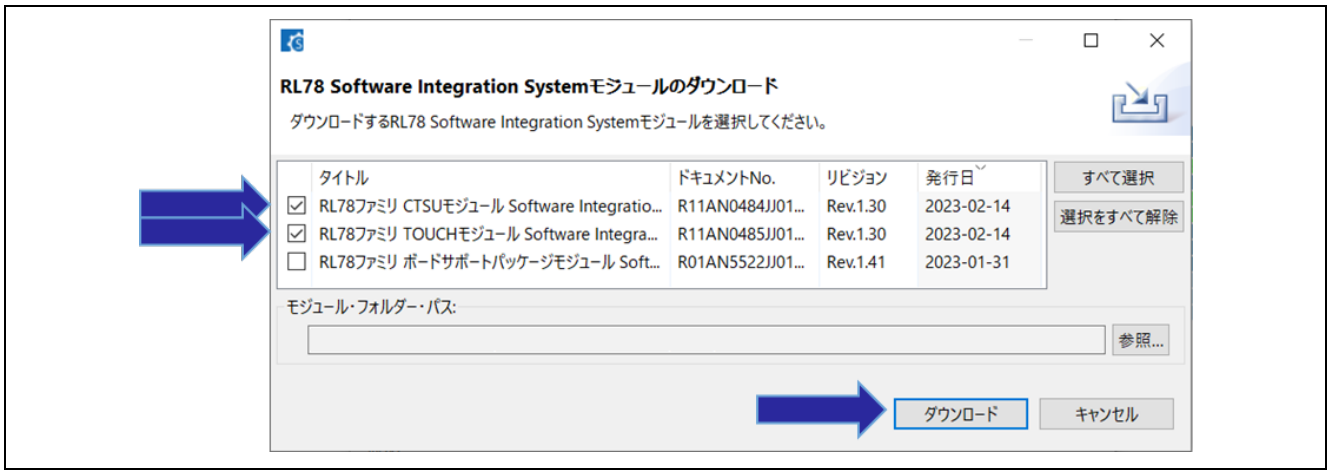


図 7-8 SIS モジュールのダウンロード

7.3.2 CTSU (静電容量タッチセンサ) 用ドライバの設定

CTSU 用のドライバの設定手順を説明します。

1. [コンポーネント]タブを選択し、アイコンをクリックします。表示されたダイアログで、”r_ctsu”モジュールを選択し、[終了]をクリックします。

図 7-9 r_ctsu モジュール

2. 追加した”r_ctsu”モジュールをクリックし、アプリケーションで使用する TS 端子を有効にします。
本アプリケーション例では、5 つの TS 端子を使用します。
TS 端子とタッチセンサの割り当ては、使用するターゲットボードのユーザーズマニュアルをご確認ください。

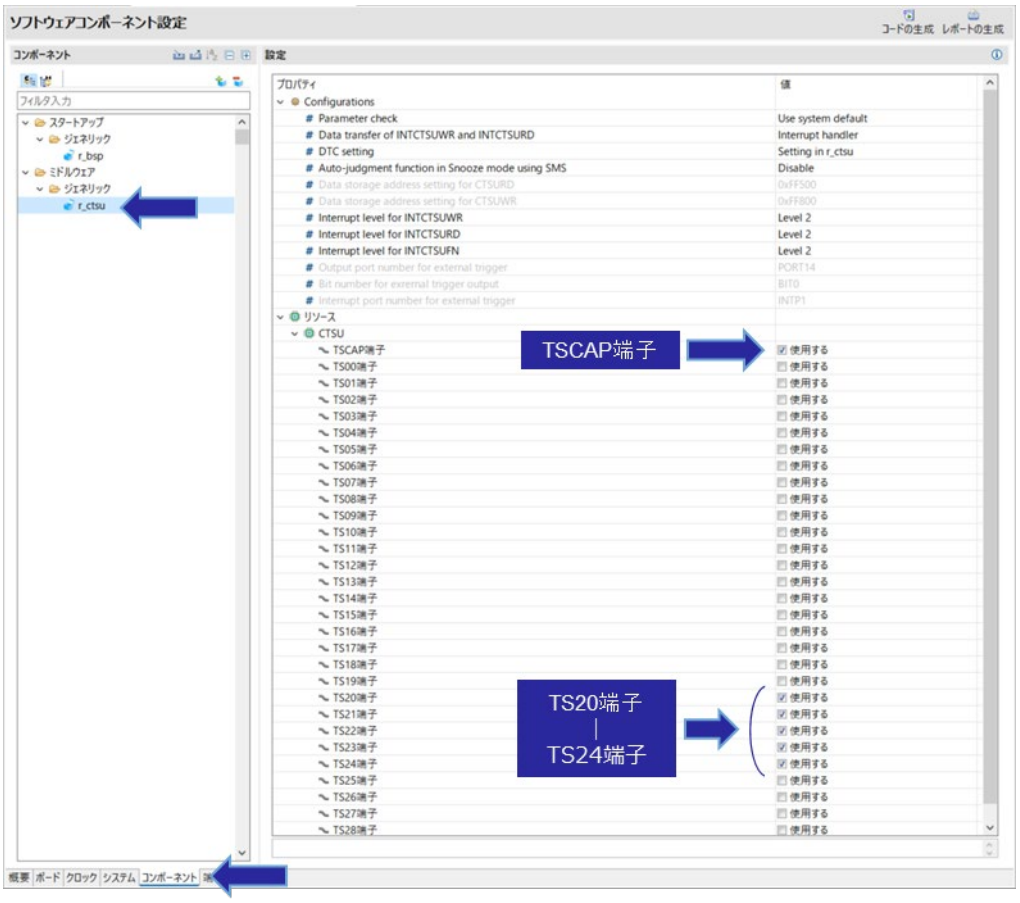


図 7-10 使用する TS 端子の有効化

3. アプリケーションで使用しない TS 端子は、Low レベル出力に設定することを推奨します。CTSU2 では、アプリケーションで使用しない TS 端子を有効にした場合、非計測端子として、Low レベル出力の設定になります。

   本アプリケーション例では、使用しない TS 端子も含めて全て有効にします。ただし、TS12/TS13 端子の兼用機能 (UART0) を本アプリケーションで使用しているため、TS12/TS13 端子は除外します。

   また、回路を作成する際は、端子処理などを適切に行い、電気的特性を満たすようにしてください。

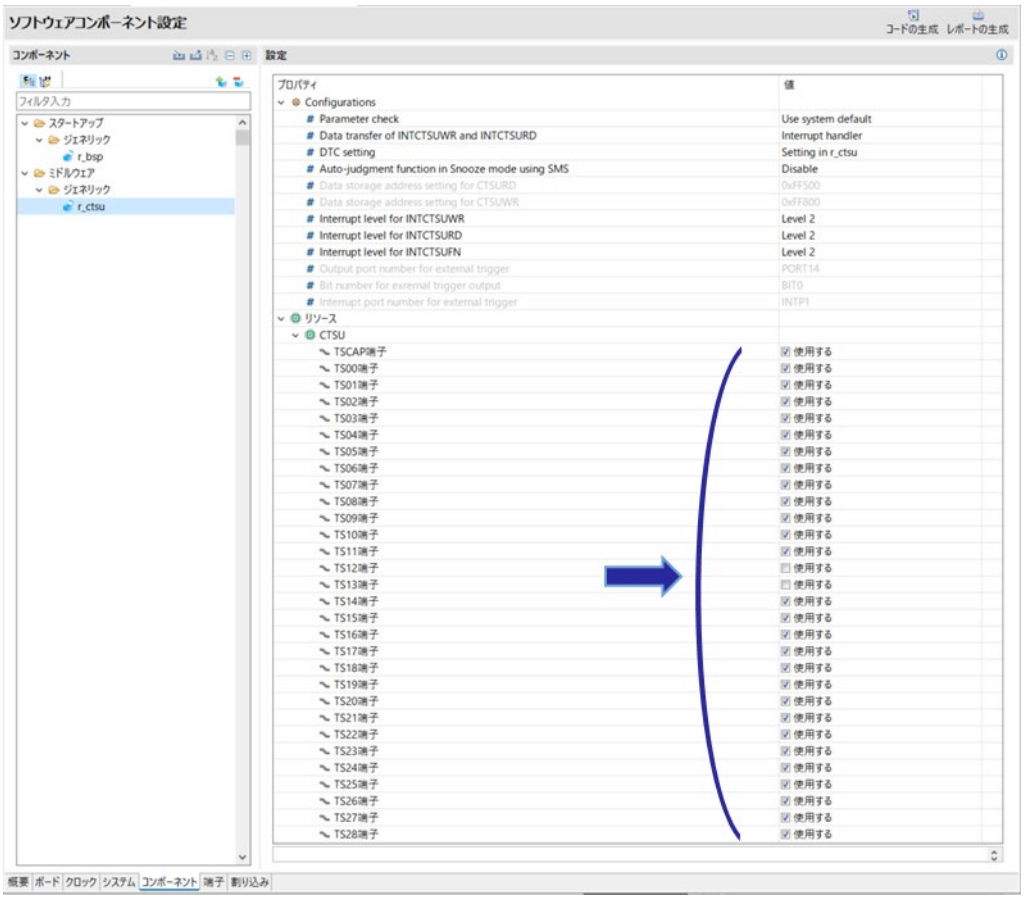


図 7-11 アプリケーションで使用しない TS 端子の有効化

7.3.3　タッチ用ミドルドライバの設定

タッチ用ミドルドライバの設定手順を説明します。

タッチアプリケーションのタッチ性能のモニタリングは、OCD (On-Chip Debugging) エミュレータを介した通信によって確認できます。ただし、RL78 ファミリの場合、モニタリングパフォーマンスは、RL78 ファミリの OCD 機能によって制限されます。

シリアル通信を介してタッチ性能のモニタリングを行うことで、スムーズなモニタリングが可能になります。タッチセンサのチューニングもシリアル通信を介すことができます。

1. アイコンをクリックし、表示されたダイアログで、”rm_touch”モジュールを選択し、[終了]をクリックします。

図 7-12 rm_touch モジュール

2. 追加した”rm_touch”モジュールをクリックし、以下を設定します。
   — UART を用いたモニタリングの有効化
   — UART を用いたチューニングの有効化
   — UART チャネル　UART0

   設定する UART チャネルは、使用するターゲットボードによって異なります。

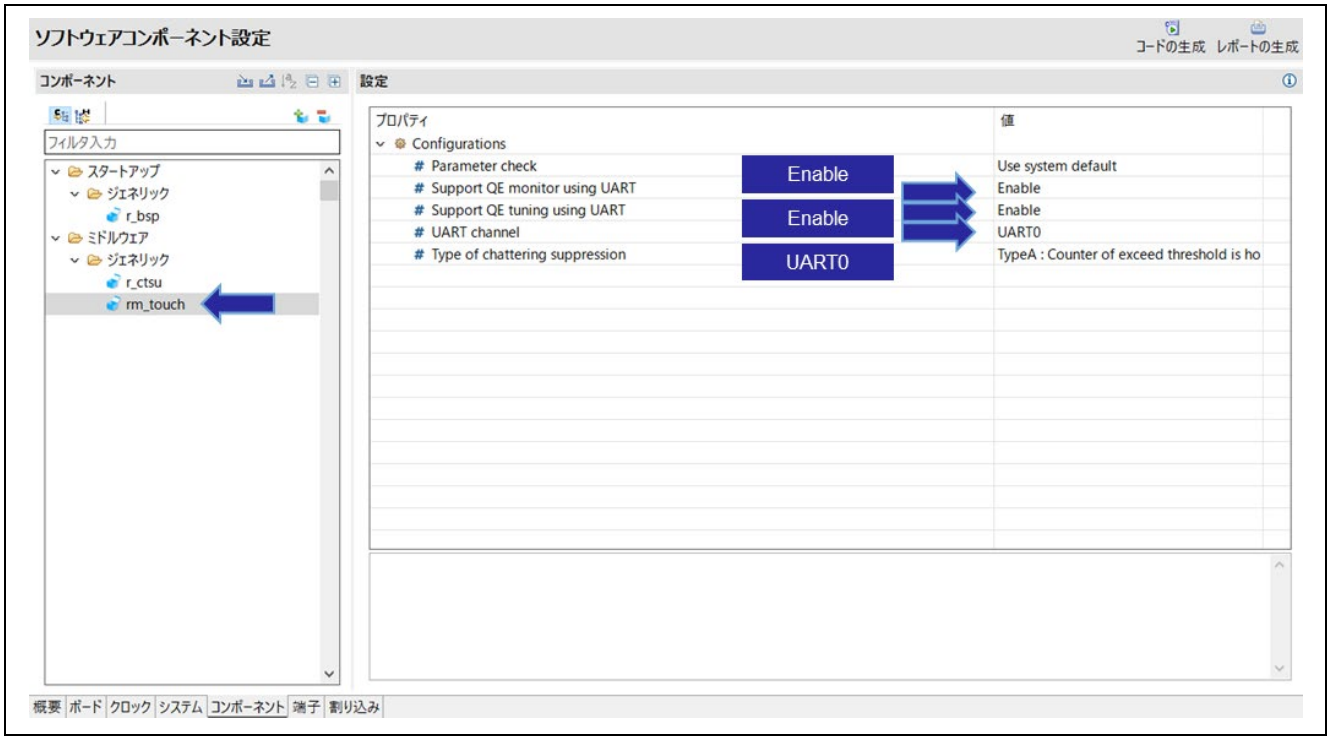


図 7-13 rm_touch の設定

7.4　シリアルインタフェース（UART 通信）の設定

タッチセンサのチューニング、モニタリングで使用する UART の設定手順を説明します。

設定する UART チャネルおよびポートは、使用するターゲットボードによって異なります。

1.  アイコンをクリックします。表示されたダイアログで、”UART 通信”モジュールを選択し、[次へ]をクリックします。表示された画面にて、以下の設定をし、[終了]をクリックします。
   — 動作　　：送信/受信
   — リソース　　：UART0

図 7-14 UART 通信モジュール

図 7-15 UART チャネルの選択

2. 追加した”UART 通信”モジュールをクリックし、”送信”および”受信”タブにて動作クロックと転送レート (ボーレート) を設定します。

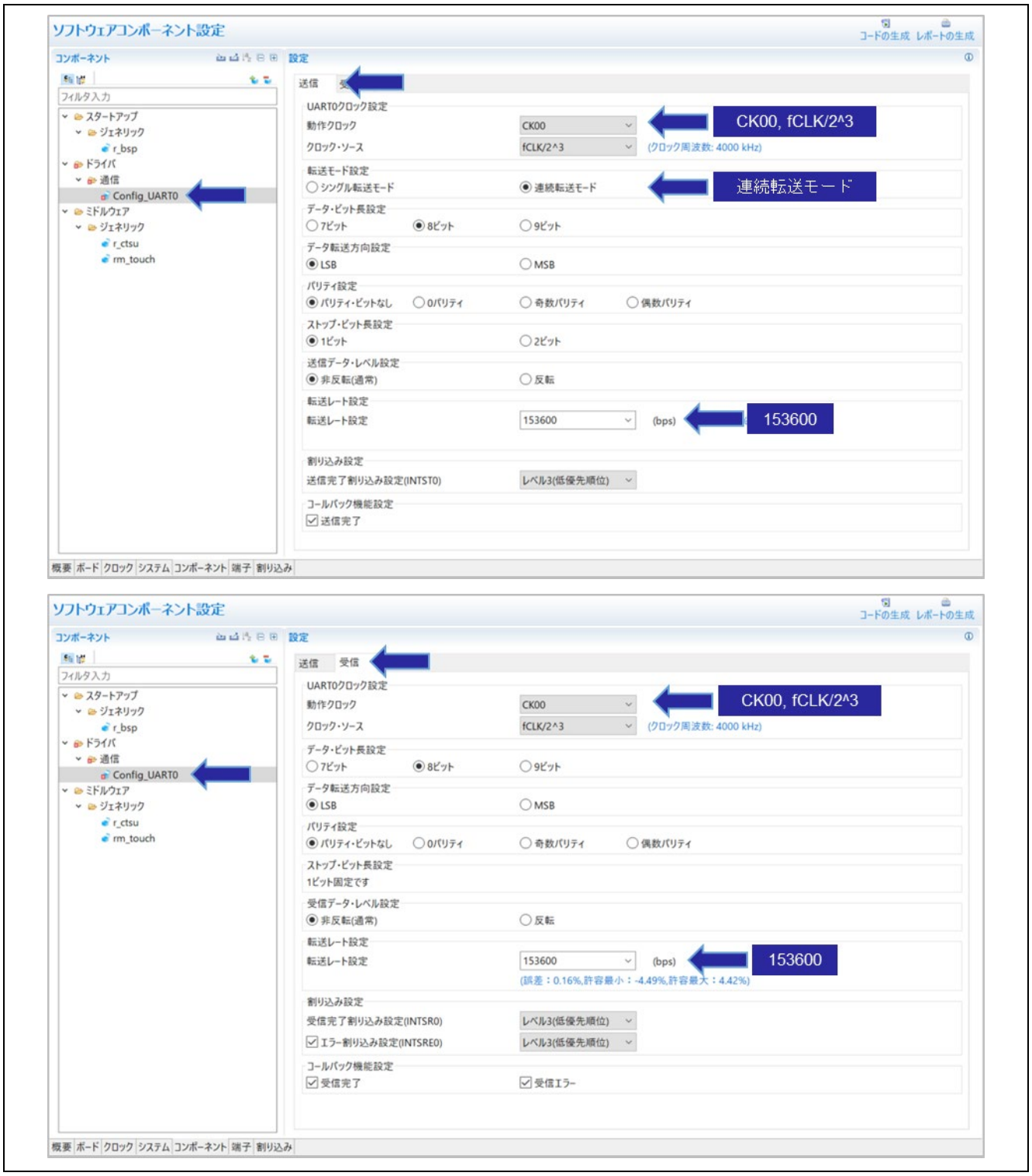


図 7-16 UART 通信モジュールの設定 (UART0)

3. [端子]タブを選択し、UART0 (SAU00) に以下の端子を割り当てます。
   — RxD0 : 21
   — TxD0 : 20

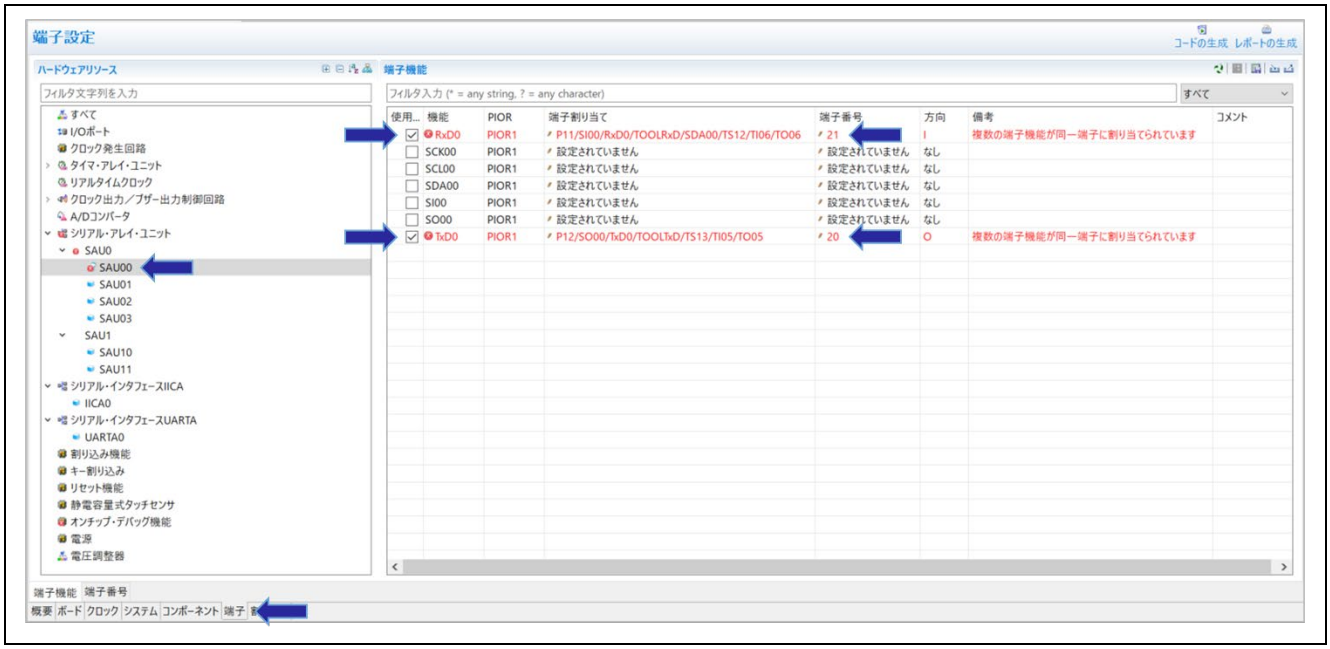


図 7-17 UART チャネル (UART0) の端子割り当て

使用するツールバージョンによっては UART0 の端子割り当てエラーが発生しますが、本エラーは無視してください。
本アプリケーションノートでは、COM ポートデバッグ機能により生成したプログラムを CS+で RL78/G22 に書き込みます。書き込みに使用する端子 (TOOLRxD/TOOLTxD) は UART0 の RxD0/TxD0 と兼用端子のため、スマート・コンフィグレータの設定上で端子の競合が発生することがありますが、CS+とスタンドアロン版 QE を同時に使用しないので実使用上の競合は発生しません。
CS+使用時 (プログラム書き込み) : TOOLRxD/TOOLTxD 端子として動作
スタンドアロン版 QE 使用時 : RxD0/TxD0 端子として動作

図 7-18 UART0 の端子割り当てエラー

7.5 未使用端子の Low レベル出力の設定

TS 端子と同様にアプリケーションで使用しないポートを Low レベル出力に設定することを推奨します。また、回路を作成する際は、端子処理などを適切に行い、電気的特性を満たすようにしてください。

Low レベル出力に設定するポートは、ターゲットボードのユーザーズマニュアルをご確認ください。

本節では、例として”PORT63”を Low レベル出力に設定する手順を説明します。

1. [コンポーネント]タブを選択し、  アイコンをクリックします。表示されたダイアログで、”ポート”モジュールを選択し、[終了]をクリックします。

図 7-19 ポートモジュール

2. 追加した”ポート”モジュールを選択し、”PORT6”にチェックを付けます。

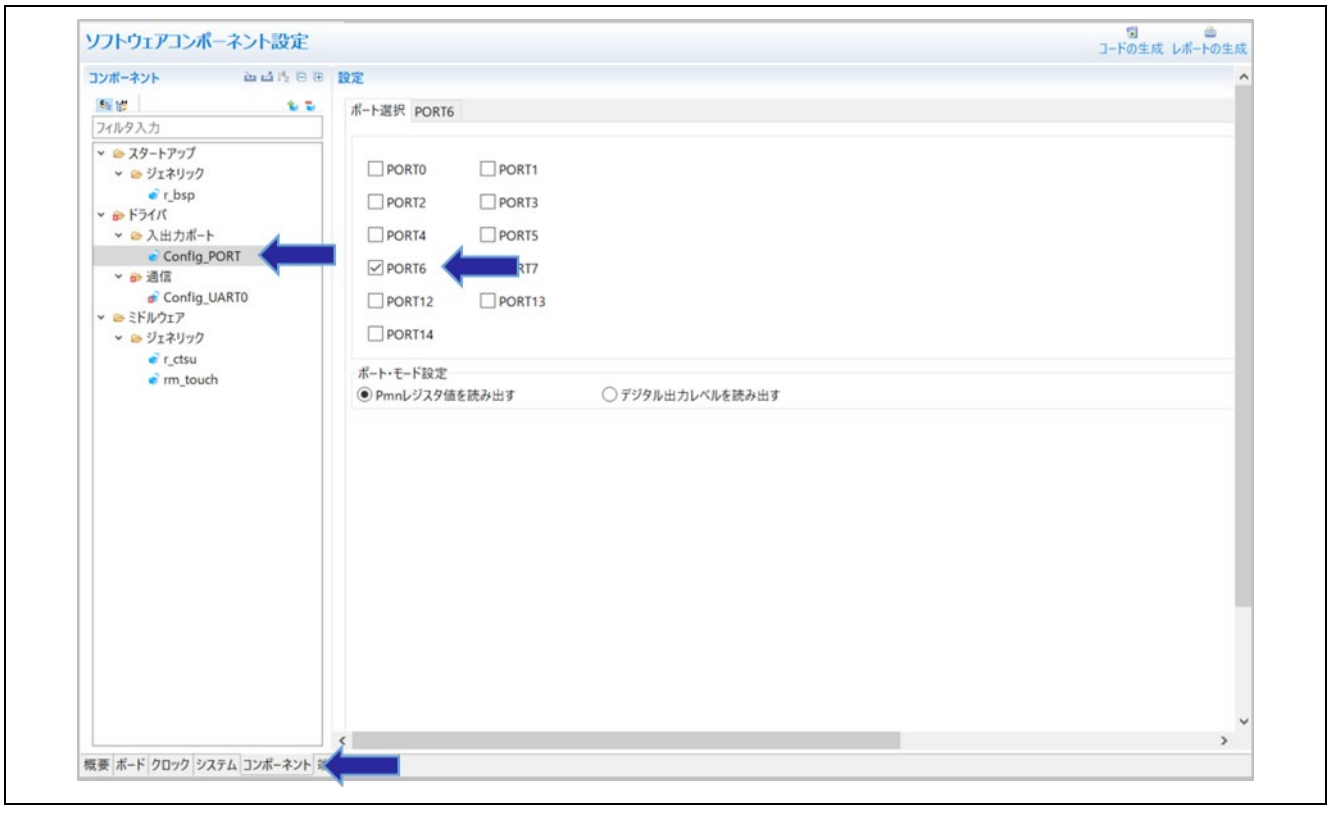


図 7-20 ポートモジュールの設定

3. [PORT6]タブをクリックし、”P63”を出力に設定します。

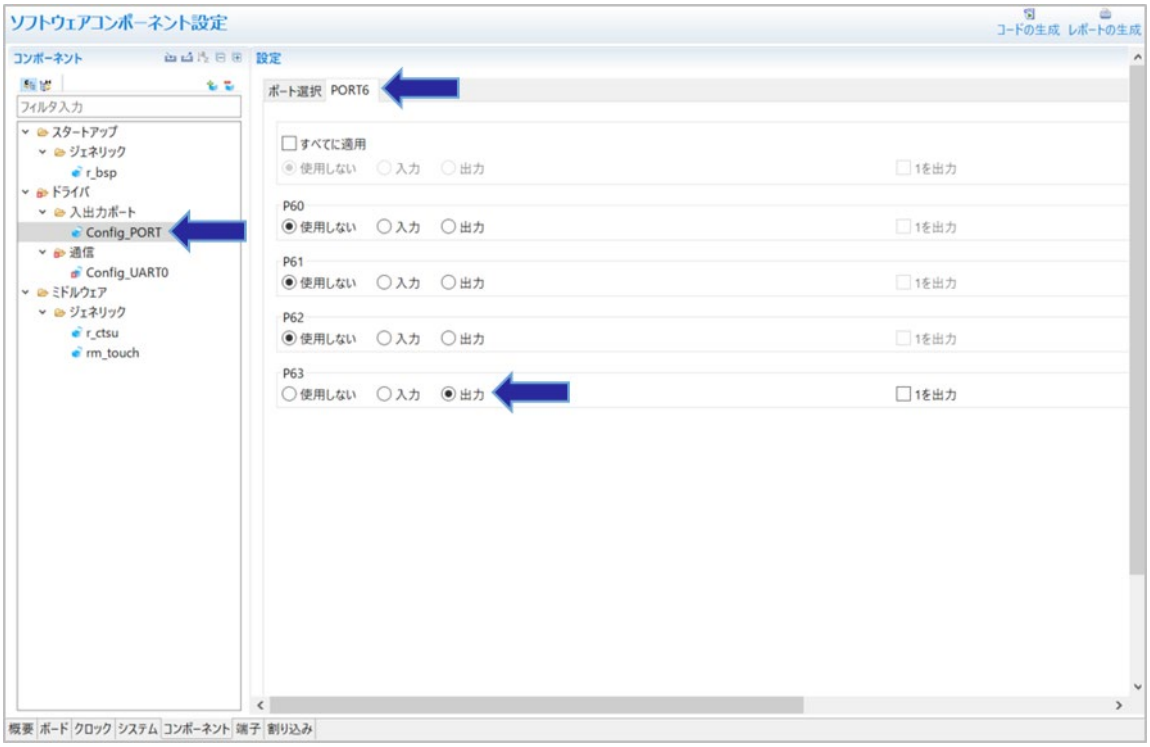


図 7-21 P63 を出力に設定

7.6　コード生成

コードの生成を行います。

1. “r_bsp”モジュールを選択し、”Initialization of peripheral functions by Code Generator/Smart Configurator”が”Enable”に設定されていることを確認してください。

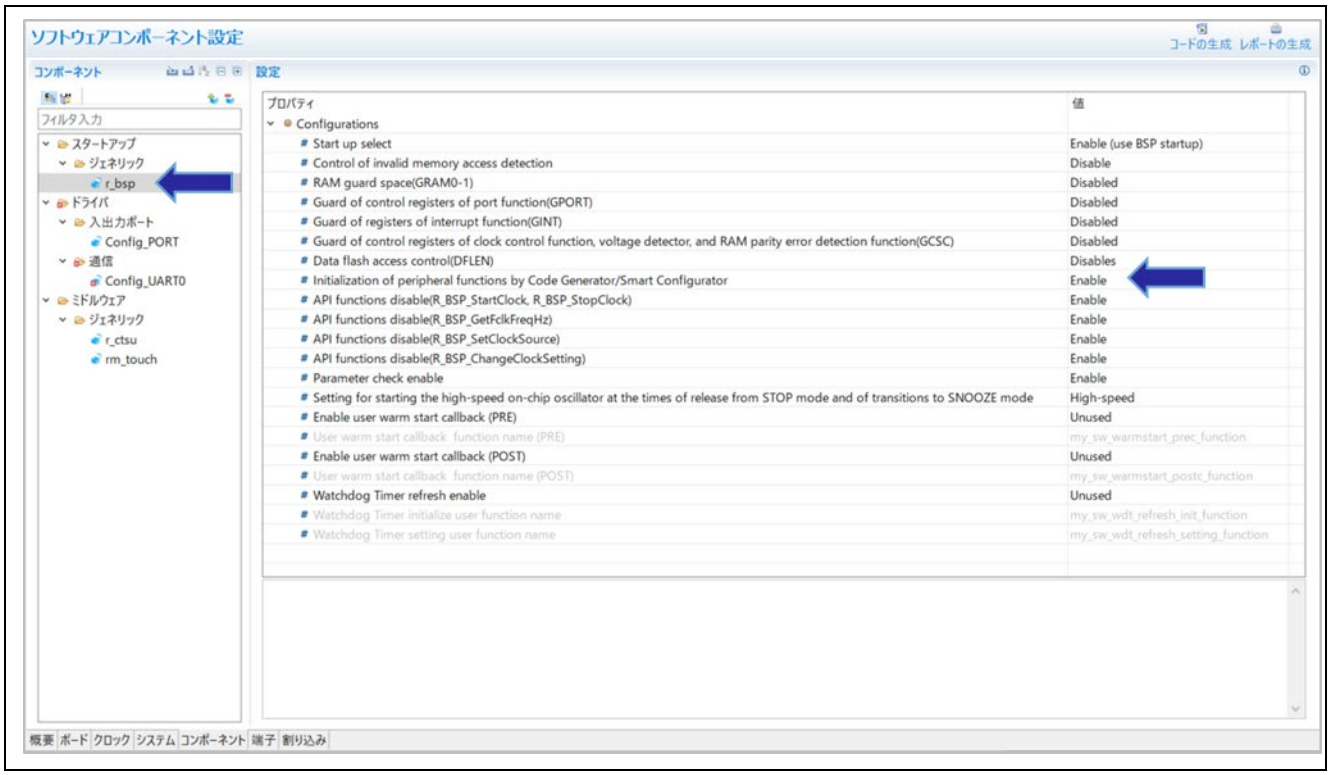


図 7-22 r_bsp の設定

2. スマート・コンフィグレータの アイコンをクリックし、コードの生成を行います。

スマート・コンフィグレータでオンチップ・デバッグ設定またはオプション・バイト設定を変更した場合、以下のメッセージが表示されることがあります。変更内容を確認した後に、[OK]をクリックします。

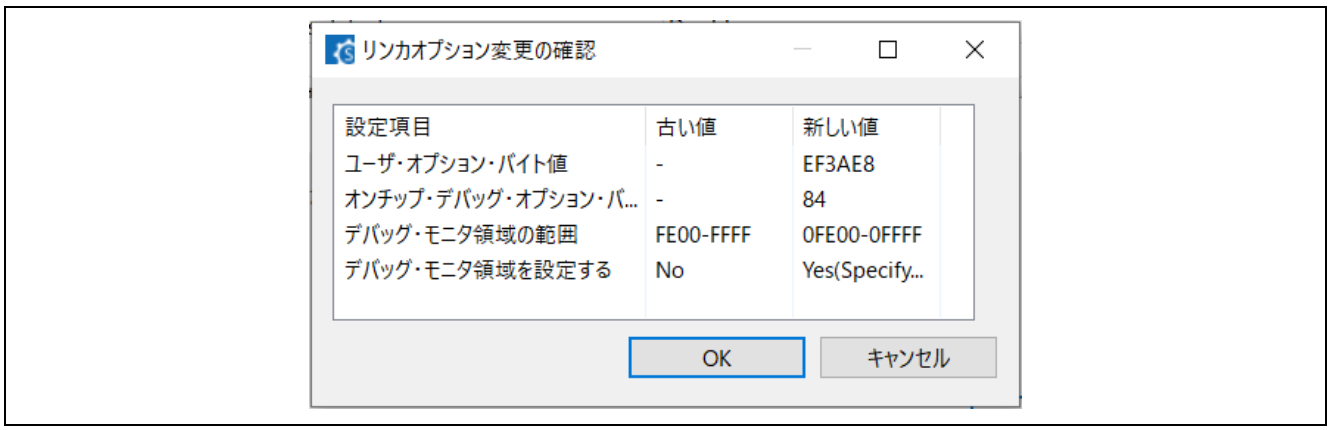


図 7-23 リンカオプション変更の確認

8. QE for Capacitive Touch の設定

8.1 QE for Capacitive Touch の起動

スタンドアロン版 QE for Capacitive Touch (以下、QE) を起動します。

1. “QE-CapTouch (QE のインストールフォルダ) / eclipse / qe-captouch.exe”より QE を起動します。

2. 起動後の画面を以下に示します。

図 8-1 QE 起動後画面

全画面表示にした際に表示が崩れる場合は、Windows 設定よりディスプレイの”拡大と縮小レイアウト”を”100%”に設定してください。

8.2 プロジェクトの準備

QE 起動後画面中央にあるワークフロー図の“プロジェクトの準備”に従い、設定します。

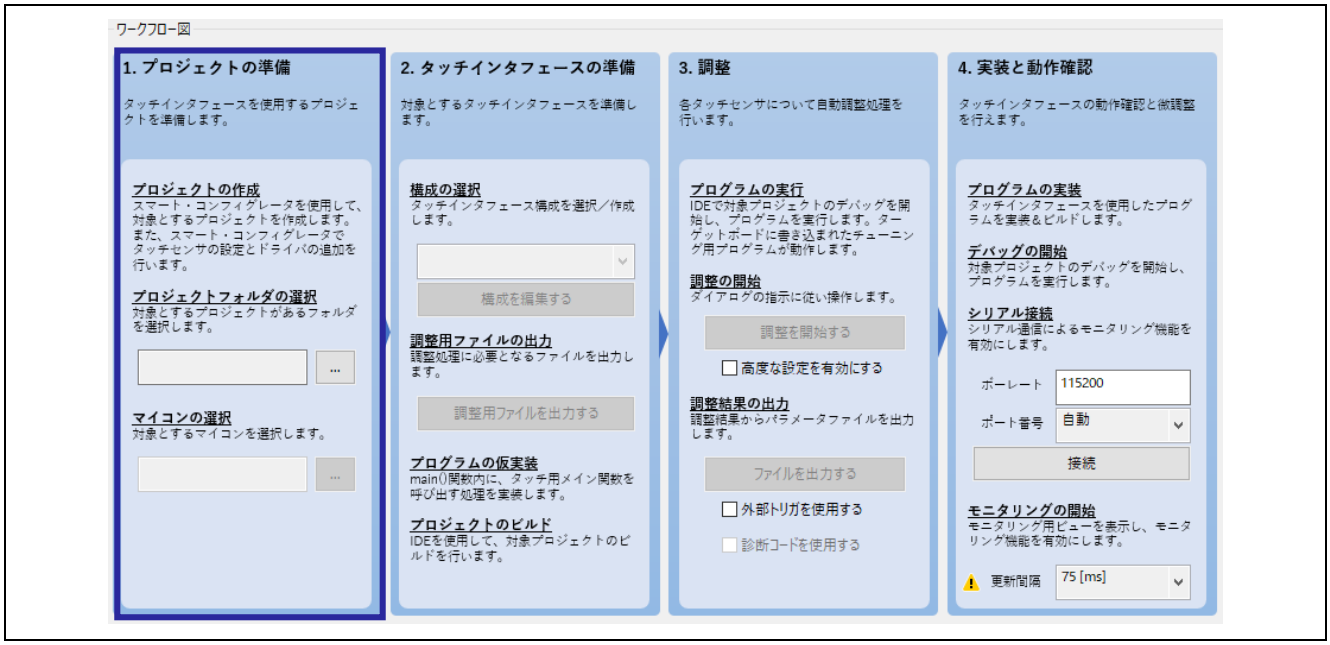


図 8-2 ワークフロー図 (プロジェクトの準備)

1. “プロジェクトフォルダの選択”下の [···] をクリックし、CS+で作成したプロジェクトフォルダを選択します。

2. “マイコンの選択”下の [···] をクリックし、使用するマイクロコントローラを選択します。

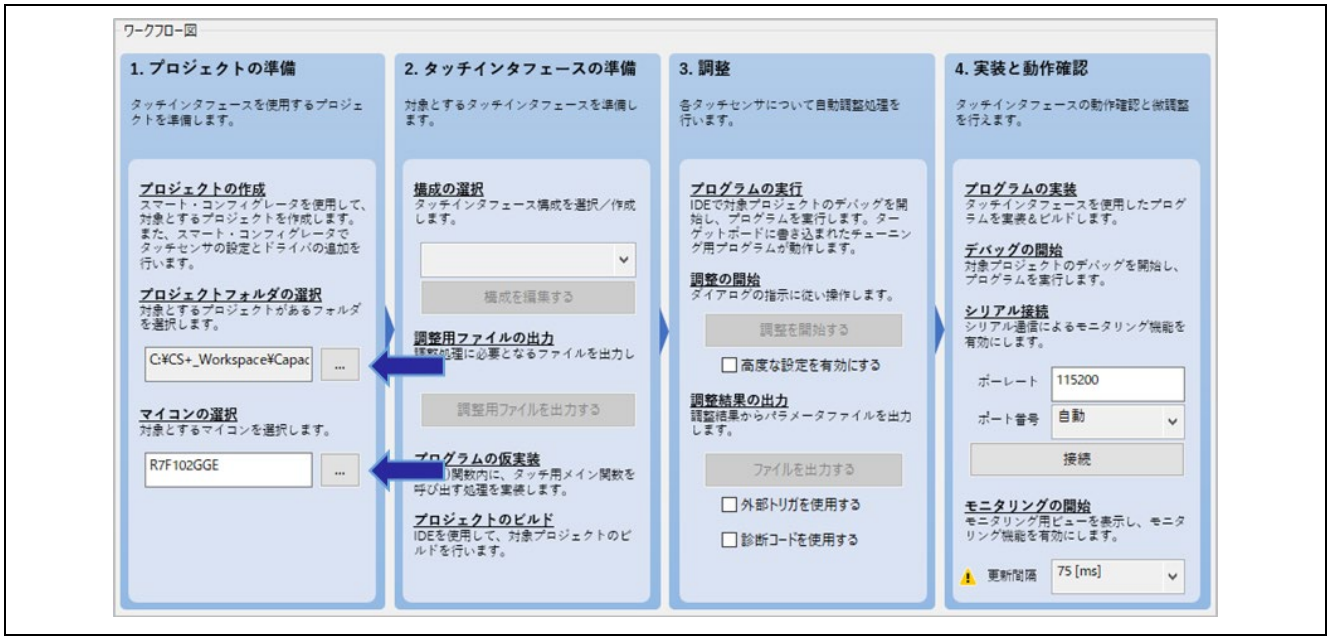


図 8-3 プロジェクトの準備

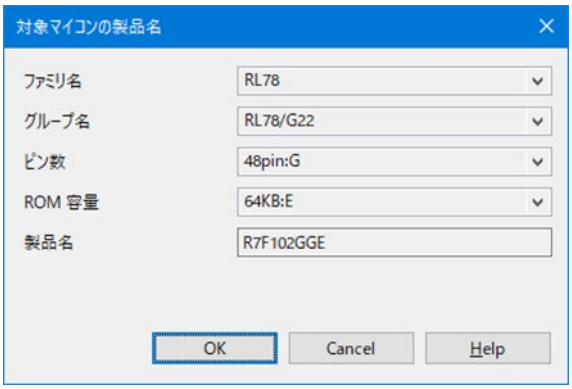


図 8-4 マイコンの選択

“マイコンの選択”で、以下のようなエラーが出た場合は、QE のインストールフォルダの格納場所に問題がある可能性があります。一度 QE を終了し、フォルダを C:¥Renesas フォルダ以下などに移動させたのち、QE を起動させてください。

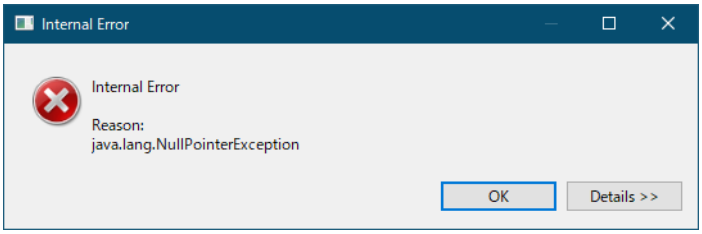


図 8-5 マイコンの選択のエラー

8.3 タッチインタフェースの準備

ワークフロー図の“タッチインタフェースの準備”に従い、設定します。

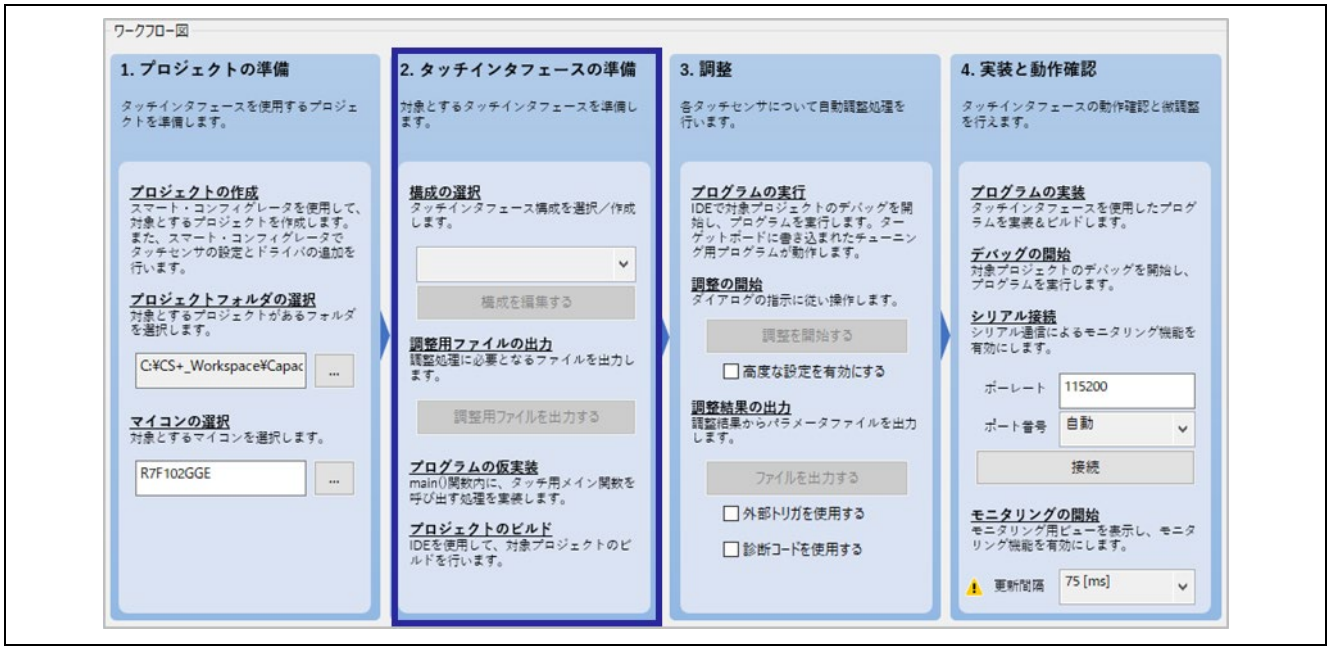


図 8-6 ワークフロー図 (タッチインタフェースの準備)

1. “構成の選択”下の をクリックし、“タッチインタフェース構成の新規作成”を選択します。

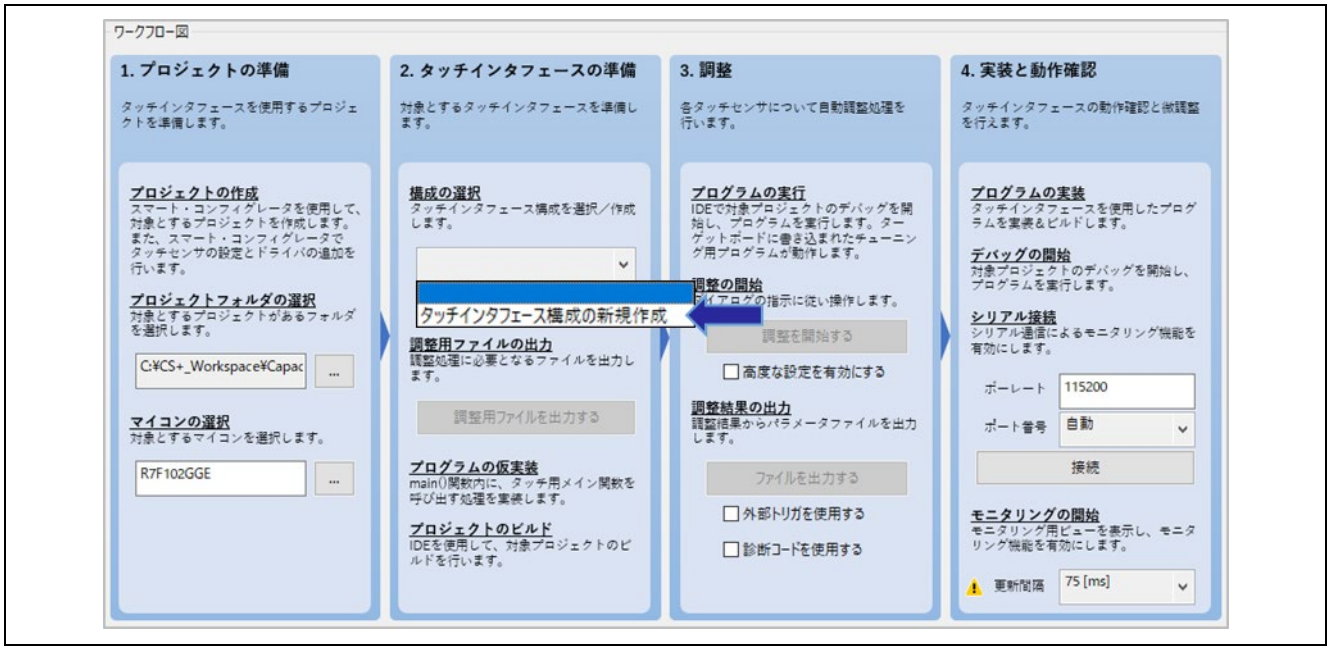


図 8-7 タッチインタフェース構成の新規作成

2. “タッチインタフェース構成の作成”ウィンドウが開き、タッチインタフェースを配置する領域が表示されます。
   右側の“タッチ I/F”パネルから[ボタン]をクリックすると、カーソルがボタンを配置できる状態になり、配置領域でクリックするとボタンを配置できます。
   以下のように 2 つのボタンを配置し、[ESC]キーを押してボタンの配置を終了します。
   同様に、“タッチ I/F”パネルから[スライダ (横方向)]をクリックすると、カーソルがスライダを配置できる状態になり、配置領域でクリックするとスライダを配置できます。
   以下のようにスライダを配置し、[ESC]キーを押してタッチインタフェースの追加を終了します。

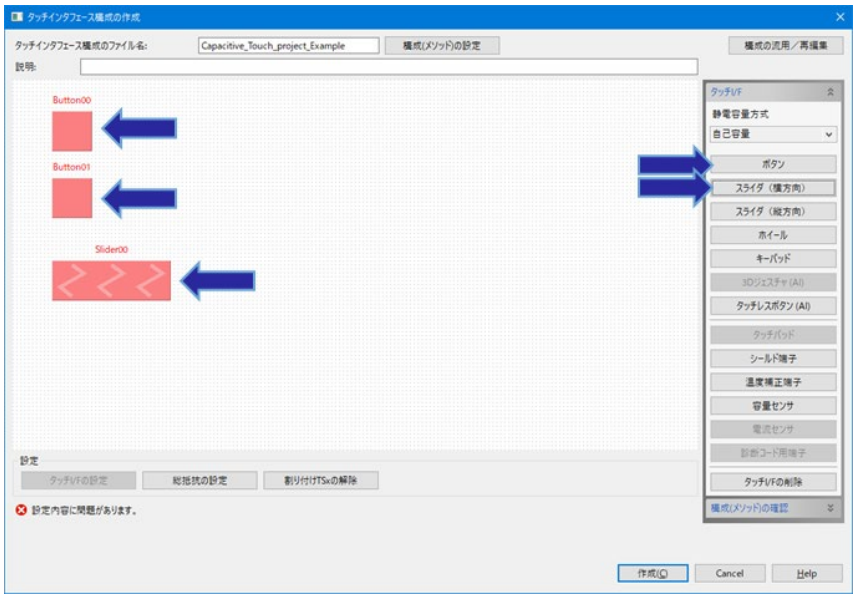


図 8-8 ボタンとスライダの配置

3. 配置した[Button00]をダブルクリックし、表示された”タッチインタフェースの設定”ダイアログで、以下のように設定します。
   — タッチセンサ : TS24
   — 抵抗値[Ω] : 560

   抵抗値は、使用するターゲットボードのユーザーズマニュアル、または回路図をご確認ください。

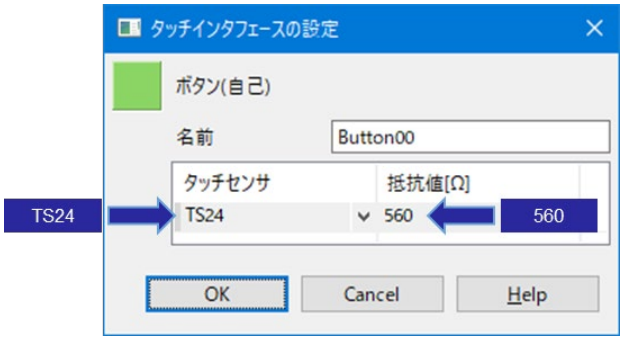


図 8-9 タッチインタフェースの設定 (ボタン)

4. 同様にして[Button01]を以下のように設定します。
   — タッチセンサ　　: TS23
   — 抵抗値[Ω]　　: 560

5. 同様にして[Slider00]を以下のように設定します。
   — タッチセンサ　　: TS20
   　　: TS21
   　　: TS22
   — 抵抗値[Ω]　　: 560

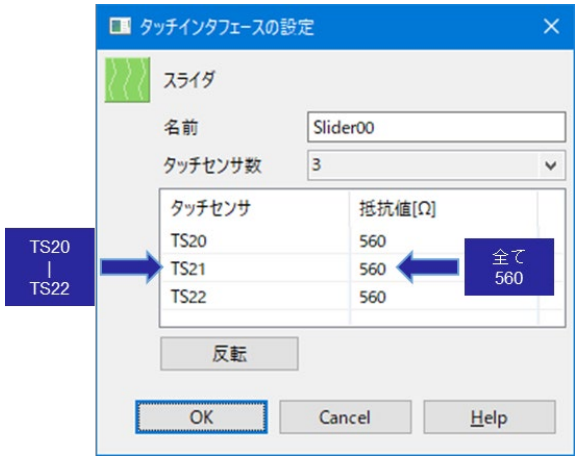


図 8-10 タッチインタフェースの設定 (スライダ)

6. タッチインタフェースの設定後は、以下のようになります。[作成]をクリックします。

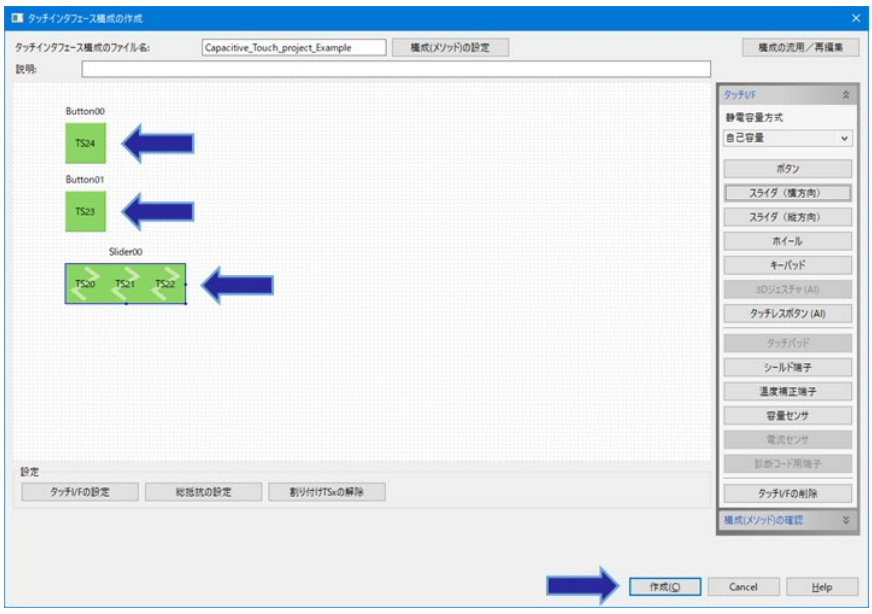


図 8-11 タッチインタフェース設定後の構成

7. “チューニング”パネルに”タッチインタフェースの構成”が表示されます。

図 8-12 タッチインタフェースの構成

8. [調整用ファイルを出力する]をクリックします。出力先のフォルダを選択します。
“Capaticive_Touch_Project_Example/src”の下に、新規に“qe_gen” フォルダを作成し、出力します。

以下が出力されるファイルを含めたフォルダ構成です。

```
Capaticive_Touch_Project_Example        ← CS+プロジェクトフォルダ
  |- src
    |- src_gen
    |- qe_gen                           ← 新規作成フォルダ
      |- qe_touch_config.c              ← 出力ファイル
      |- qe_touch_config.h              ← 出力ファイル
      |- qe_touch_define.h              ← 出力ファイル
      |- qe_touch_sample.c              ← 出力ファイル
```

9. 出力先のフォルダを選択すると、ダイアログが表示されますので、周波数を設定し[OK]をクリックします。

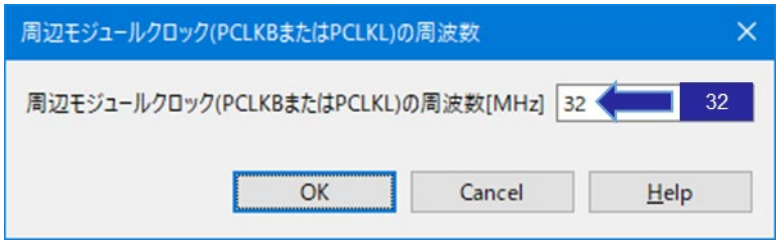


図 8-13 周辺モジュールクロックの周波数の設定

10. 続いて表示された“マイコンへの供給電圧”ダイアログで、電圧値を設定して[OK]をクリックします。
使用するマイクロコントローラの電気的特性をご確認ください。
なお RL78/G22 の場合は、EV_{DD} はありませんので、V_{DD} への供給電圧に応じて設定を行ってください。

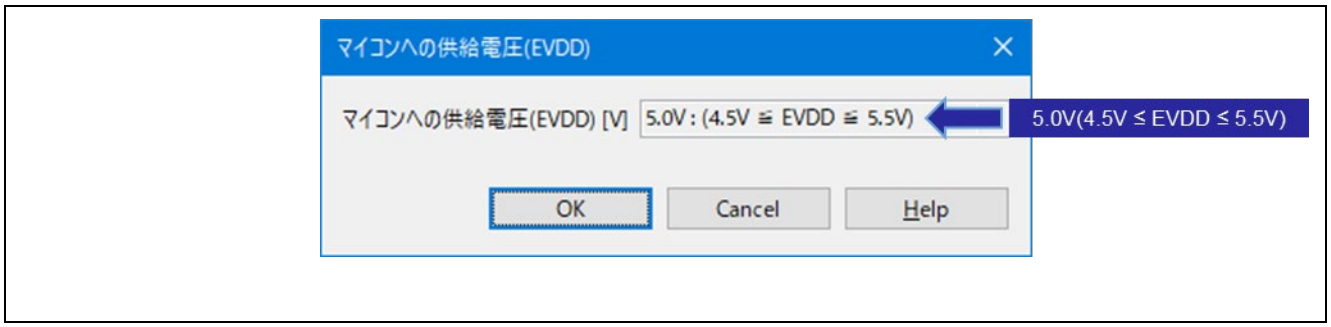


図 8-14 マイコンへの供給電圧の設定

11. 続いて“QE for Capacitive Touch”ダイアログが表示されます。以降の A, B ではダイアログ内の指示に従い設定を行います。
また、ダイアログの内容は、QE 画面下部の“コンソール”パネルにも表示されます。

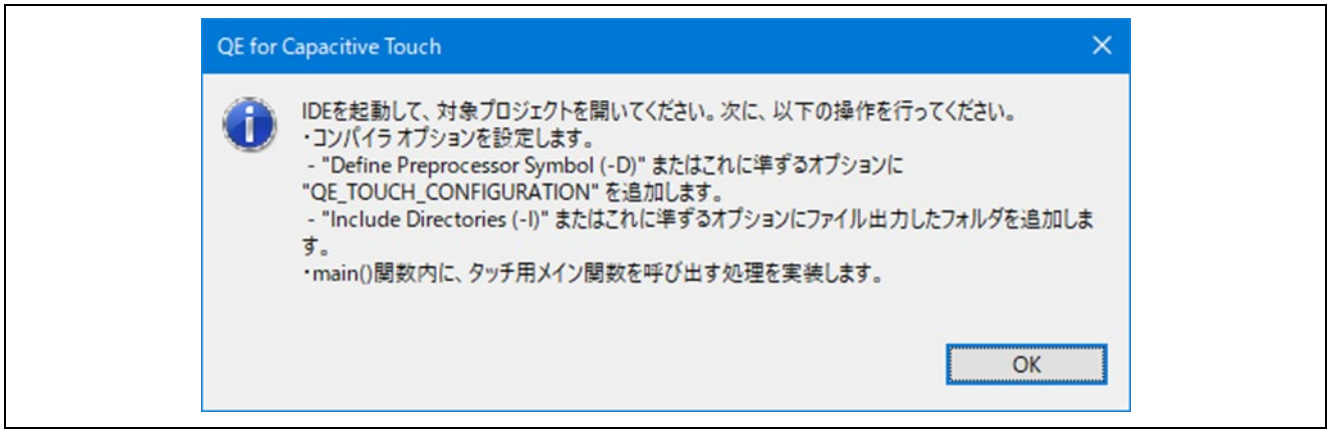


図 8-15 QE for Capacitive Touch ダイアログ

図 8-16 コンソールパネル

A. コンパイル・オプションを設定します。
CS+を開き、“プロジェクト・ツリー” から “CC-RL (ビルドツール)” を選択します。
プロパティ” の “共通オプション” タブのうち、“よく使うオプション (コンパイル) ” の下の “定義マクロ” を選択し、右側に表示された [···] をクリックします。

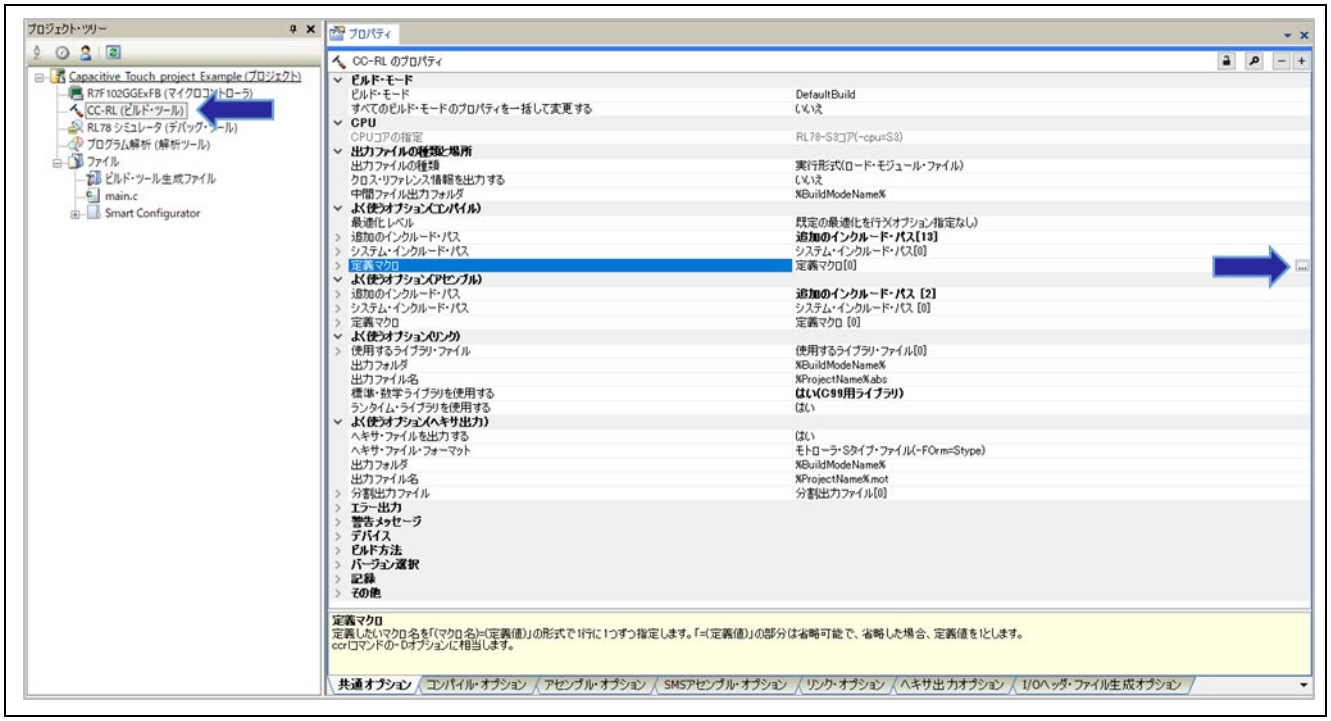


図 8-17 定義マクロの選択

表示された “テキスト編集”ダイアログの“テキスト”フィールド内に“QE_TOUCH_CONFIGURATION” を追加し、[OK]をクリックします。

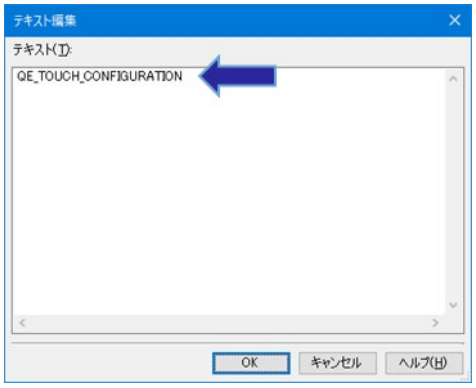


図 8-18 定義マクロの編集

続いて、”よく使うオプション (コンパイル) ” の下の ”追加のインクルードパス” を選択し、右側に表示された[···]をクリックします。
表示された “パス編集” ダイアログの “パス” フィールドに ”src¥qe_gen” を追加し、[OK]をクリックします。

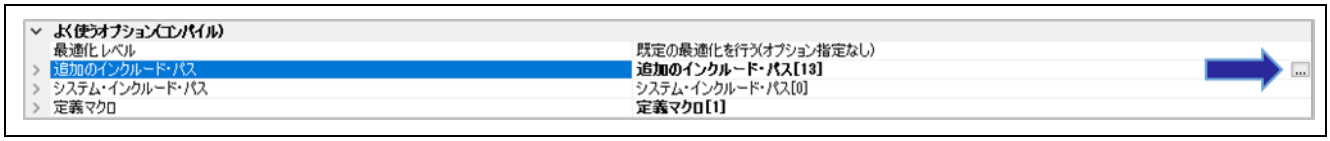


図 8-19 コンパイラのインクルードパスの追加

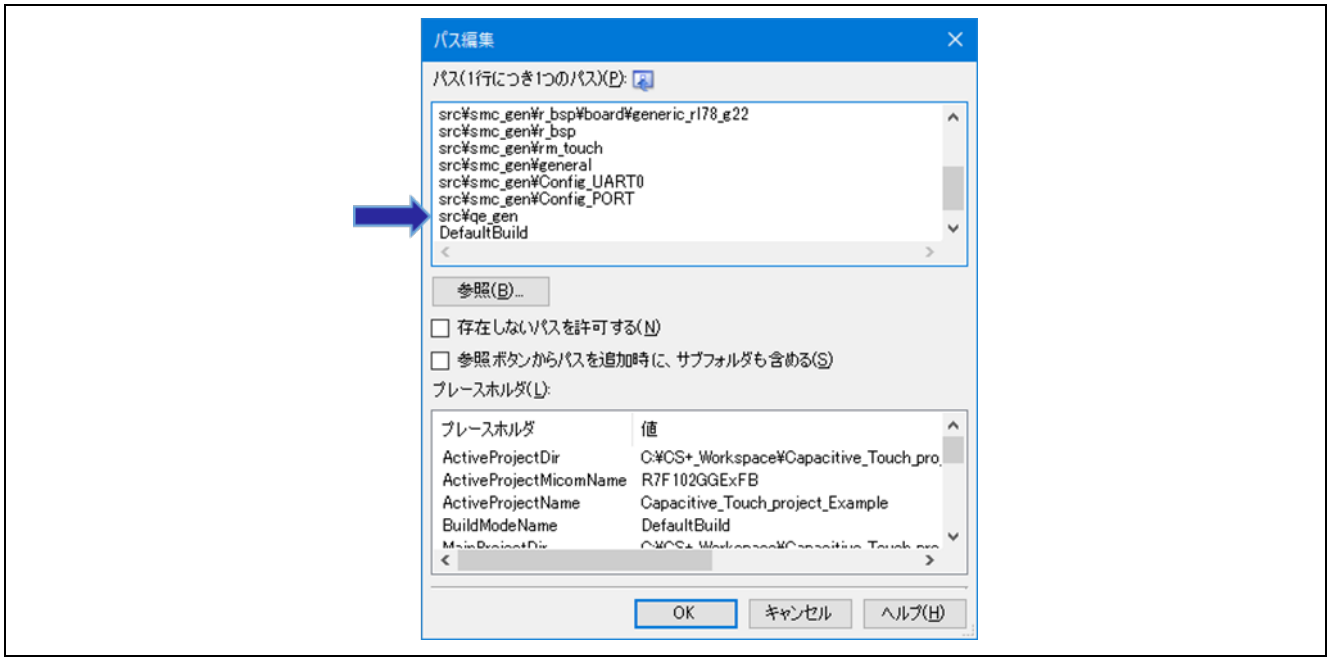


図 8-20 インクルードパスの編集

次に、”よく使うオプション (リンク)” の下の ”標準・数学ライブラリを使用する” をクリックします。右側に表示された をクリックし、“はい (C99 用ライブラリ)” を選択します。

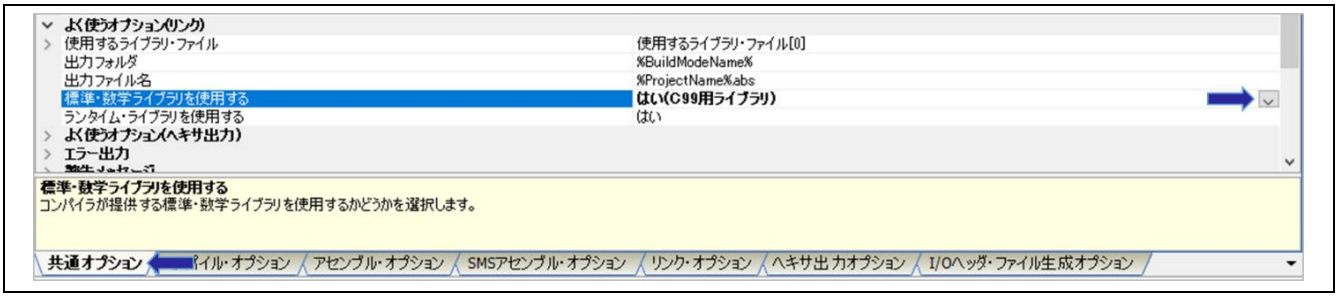


図 8-21 標準・数学ライブラリの選択

次に、”コンパイル・オプション” のタブにある ”ソース” をダブルクリックし、”C ソース・ファイルの言語”をクリックします。右側に表示された をクリックし、“C99 (-lang=c99) ”を選択します。

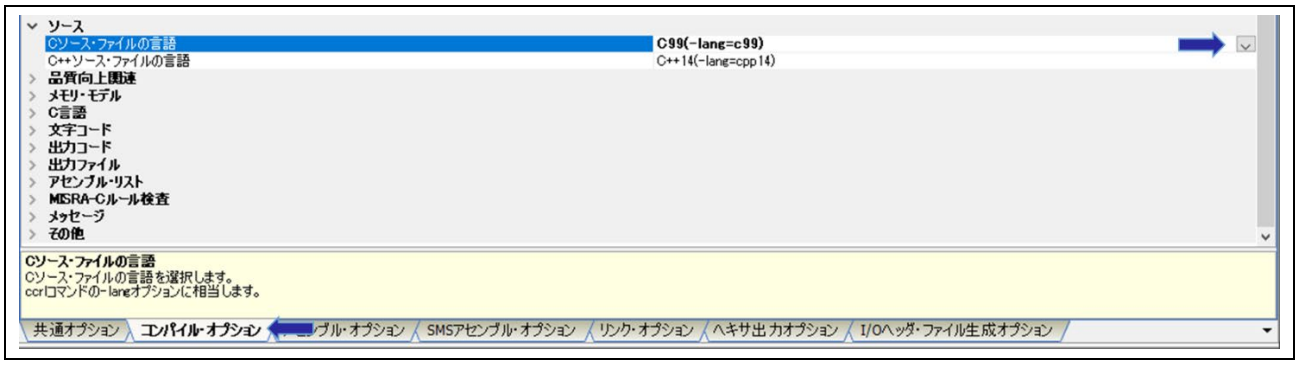


図 8-22 C 言語の規格の選択

次に、”リンク・オプション”のタブにある”デバイス”をダブルクリックし、”オンチップ・デバッグ・オプション・バイト制御値”に”84”を入力します。
同様に、”デバッグ・モニタ領域を設定する”には、”はい (範囲指定) (-DEBUG_MONITOR=<アドレス範囲>) ”を選択します。
また、”ユーザ・オプション・バイト値”には、”EF3AE8”を入力します。
設定するオプション・バイト値は、使用するマイクロコントローラのユーザーズマニュアルをご確認ください。

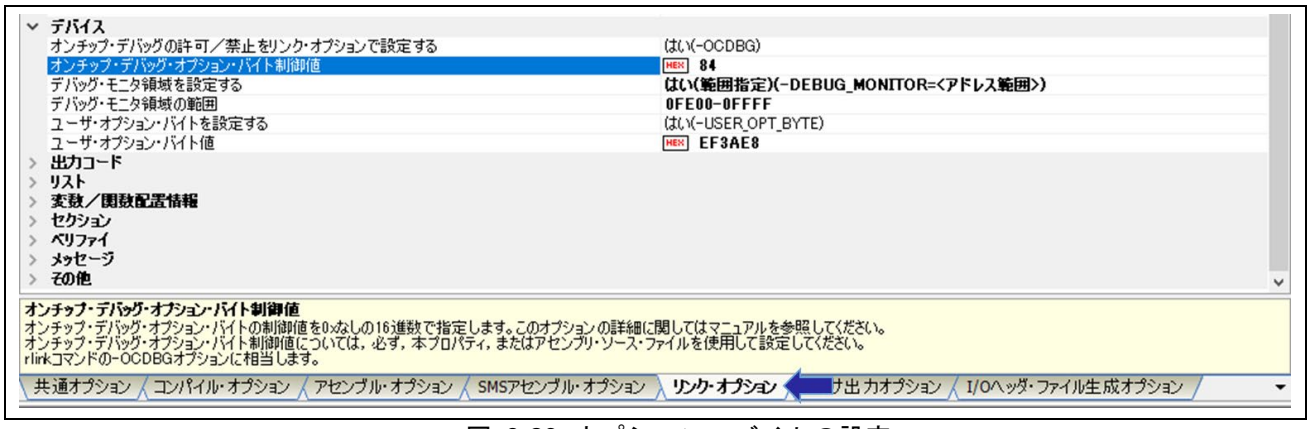


図 8-23 オプション・バイトの設定

CC-RL 無償評価版の V1.12.00 以降のバージョンを使用してコンパイルする場合は、コンパイラの最適化レベルを” デバッグ優先 (-onothing)”を設定してビルドしてください。
”コンパイル・オプション”のタブにある”最適化”をダブルクリックし、”最適化レベル”に”デバッグ優先 (-onothing)”を選択します。

備考　本操作はチューニング時のみ必要となります。チューニング後は、任意の最適化設定で使用可能です。

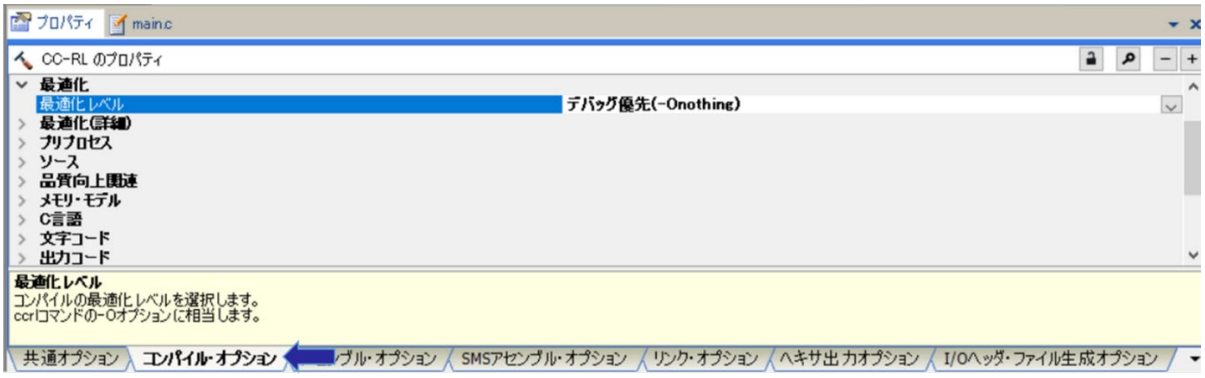


図 8-24 コンパイラの最適化レベルの設定

B. main()関数内に、タッチ用メイン関数を呼び出す処理を実装します。
CS+の“プロジェクト・ツリー”に”qe_gen”フォルダがない場合は、”qe_gen”フォルダをエクスプローラから“プロジェクト・ツリー”にドラッグアンドドロップして追加します。

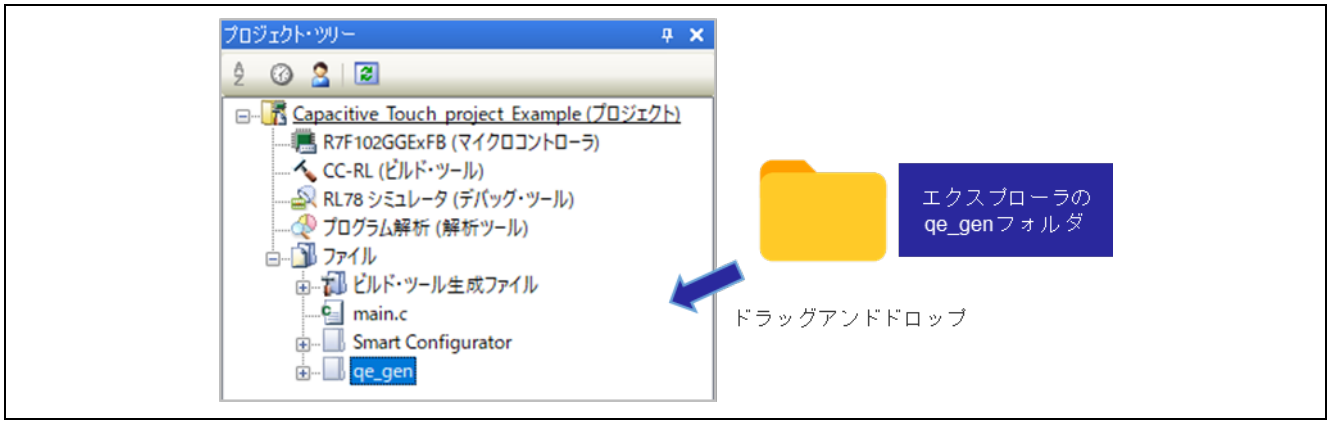


図 8-25 qe_gen フォルダの追加

main()関数から qe_touch_main()関数をコールします。”main.c”ファイルへ以下のコードを追加します。

- extern void qe_touch_main(void);
- qe_touch_main();

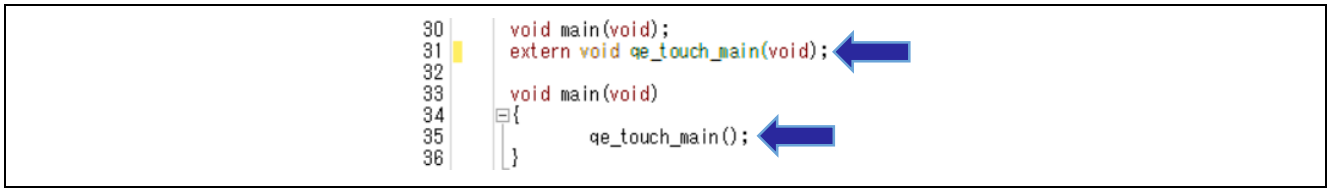


図 8-26 main.c

12.“Config_UART0_user.c”にシリアル通信用の関数を追加します。
以下のコードをそれぞれ追加します。
— extern void touch_uart_callback(uint16_t event);
— touch_uart_callback(0);
— touch_uart_callback(1);

```
52      /* Start user code for global. Do not edit comment generated here */
53      extern void touch_uart_callback(uint16_t event);
54      /* End user code. Do not edit comment generated here */


74      static void r_Config_UART0_callback_sendend(void)
75      {
76          /* Start user code for r_Config_UART0_callback_sendend. Do not edit comment generated here */
77          touch_uart_callback(0);
78          /* End user code. Do not edit comment generated here */
79      }


87      static void r_Config_UART0_callback_receiveend(void)
88      {
89          /* Start user code for r_Config_UART0_callback_receiveend. Do not edit comment generated here */
90          touch_uart_callback(1);
91          /* End user code. Do not edit comment generated here */
92      }
```

図 8-27 Config_UART0_user.c

13.CS+でプロジェクトをビルドします。CS+のメニュー下にある  アイコンをクリックし、ビルドします。エラーまたはワーニングなしで終了することを確認します。

ビルド時に下図のワーニング(W0511187)が出た場合は、本書 36 ページの図 8-24 を参考にコンパイラの最適化設定を「デバッグ優先 (-onothing) 」に変更して、再ビルドしてください。

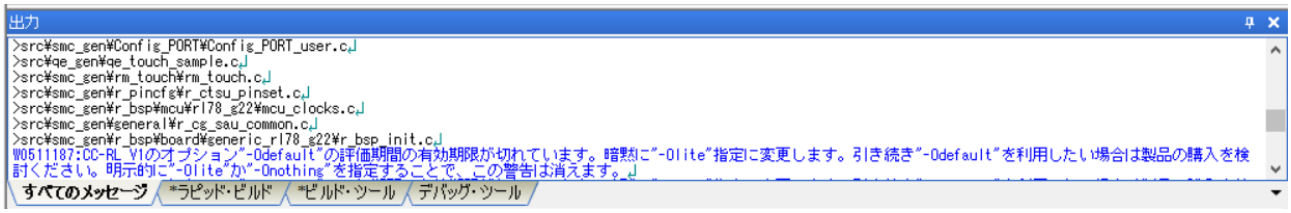


図 8-28 ビルド時のワーニング (W0511187)

8.4 調整

ワークフロー図の“調整”に従い、設定します。

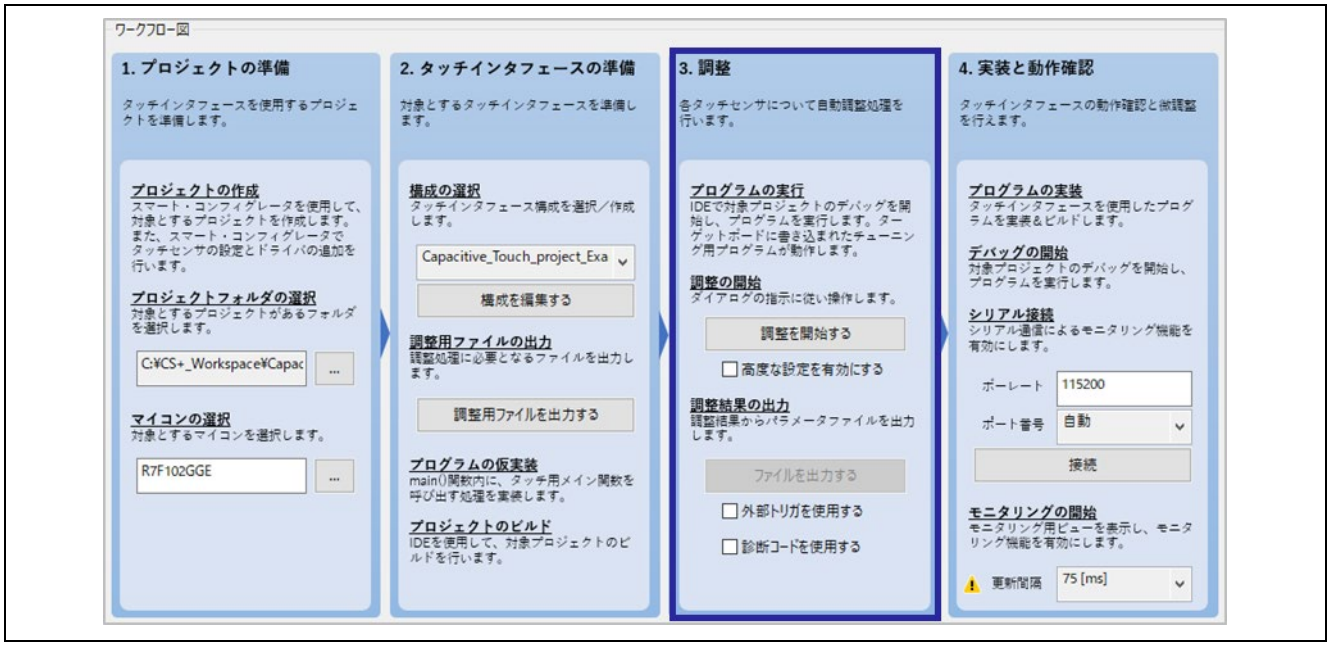


図 8-29 ワークフロー図 (調整)

1. CS+の“プロジェクト・ツリー”で“デバッグ・ツール”を右クリックし、”使用するデバッグ・ツール”の”RL78 COM Port(C)”を選択します。

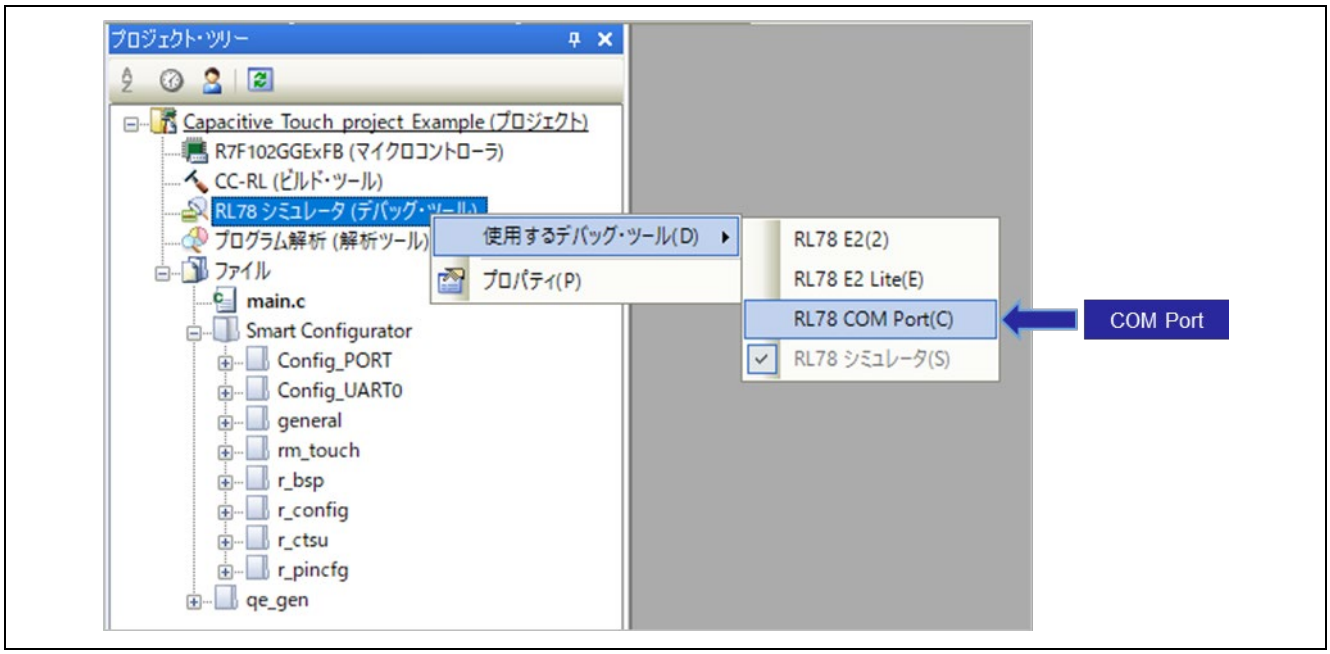


図 8-30 デバッグ・ツールの選択

2. “デバッグ・ツール”のプロパティで”通信ポート”を設定します。
   本アプリケーション例では、COM3 を使用します。

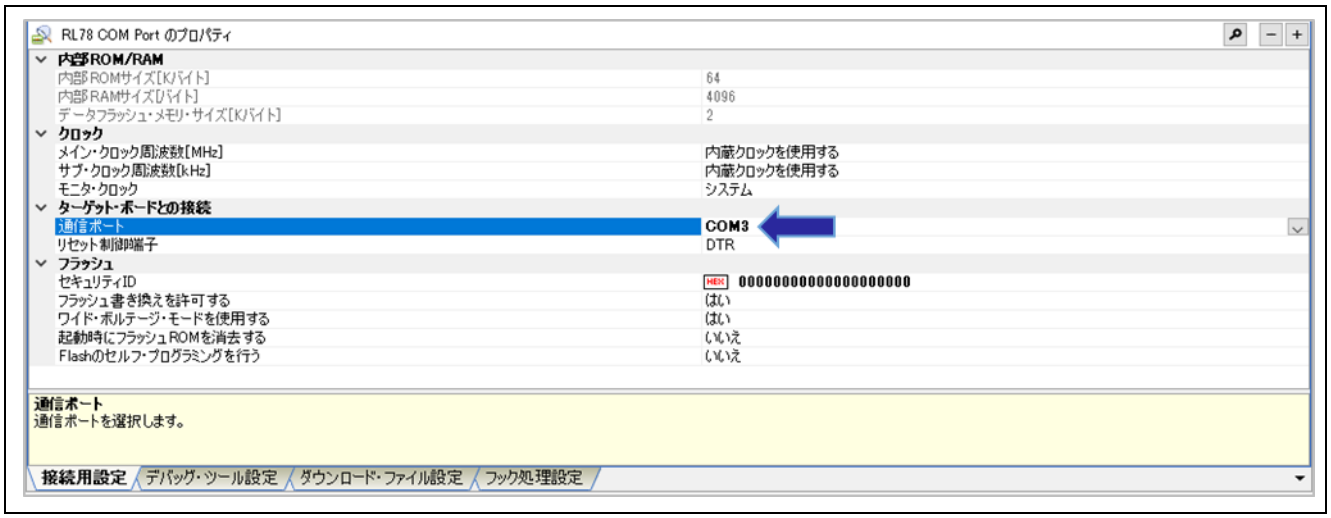


図 8-31 デバッグ・ツールのプロパティ

設定する通信ポートは、デバイス マネージャーで確認できます。

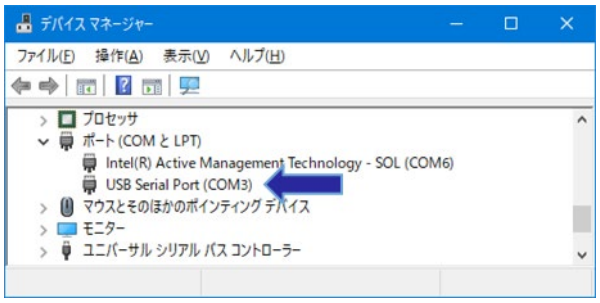


図 8-32 デバイス マネージャー

3. ターゲットボードの QE シリアル接続切り替えジャンパ (J16) がショートされていること、PC とターゲットボードが USB ケーブルで接続されていることを確認し、CS+の  アイコンをクリックし、ビルドとプログラムの書き込みを行います。プログラム書き込み後のダウンロードが完了したら、  アイコンをクリックしてプログラムを停止させ、続いて  アイコンをクリックして切断します。

   切断後、PC とターゲットボードを接続している USB ケーブルを外し、QE シリアル接続切り替えジャンパ (J16) をオープンにします。
   この後に QE と接続するため、PC とターゲットボードを USB ケーブルで再度接続します。この時にターゲットボードは、書き込んだプログラムが動作し、QE との接続待機状態となります。

   QE シリアル接続切り替えジャンパ (J16) は、ターゲットボードのユーザーズマニュアルをご確認ください。また、USB ケーブルはデータ転送対応のケーブルをご使用ください。

4. QE で“シリアル接続”の”ボーレート”を 7.4 節で設定した値に設定します。

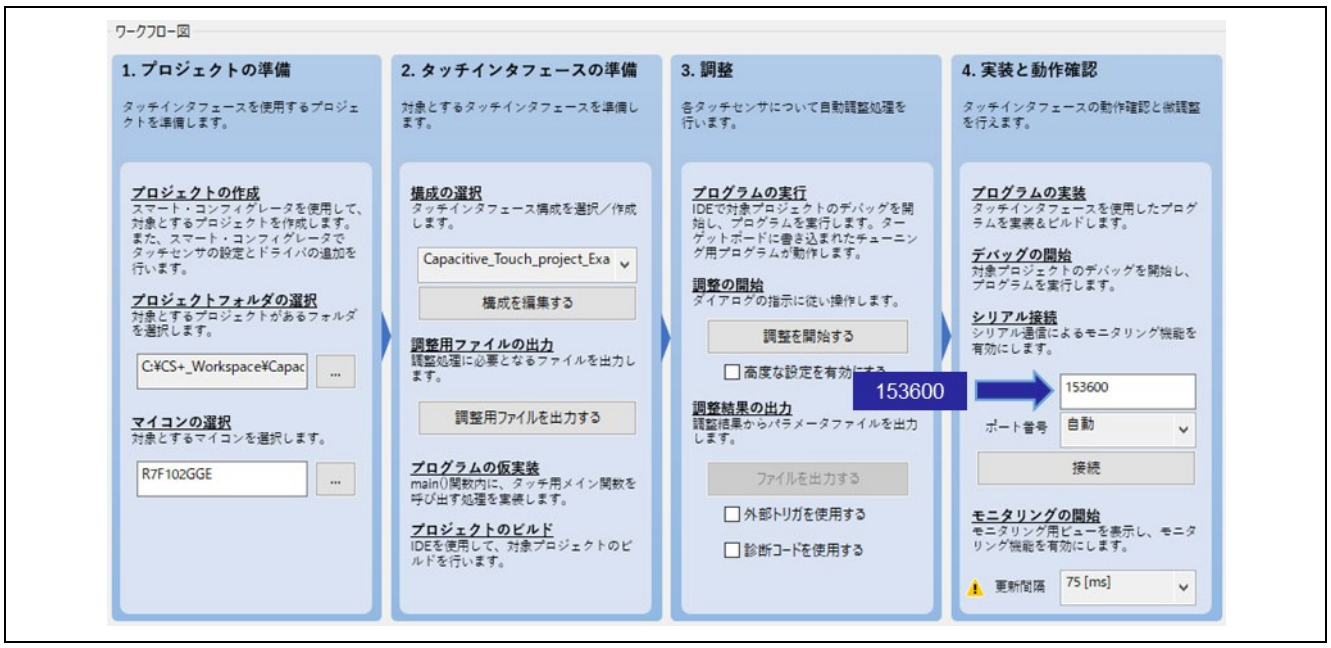


図 8-33 ボーレートの設定

5. [調整を開始する]をクリックし、自動チューニングを開始します。

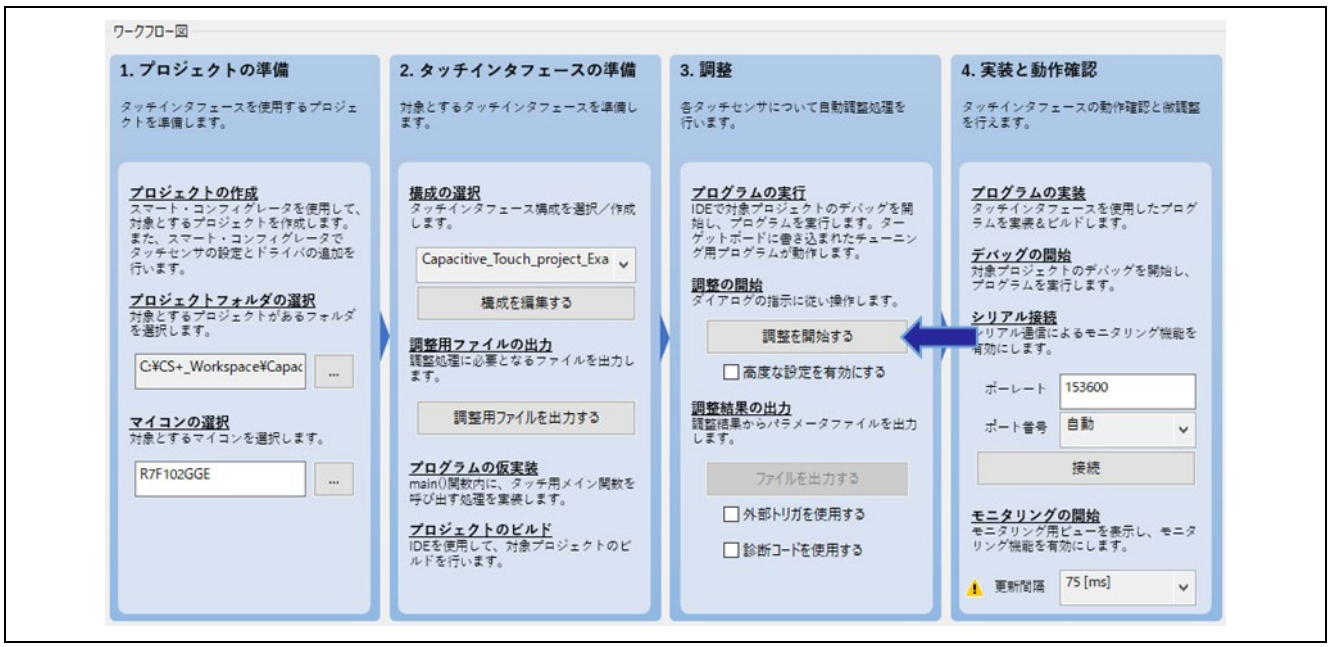


図 8-34 自動チューニング

6. 表示されたダイアログで”ボーレート”を設定し、[接続]をクリックします。

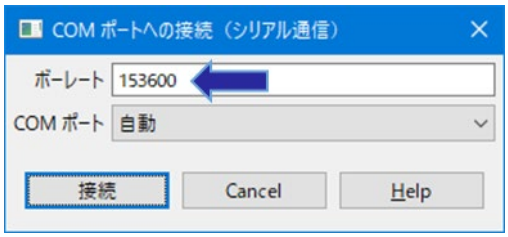


図 8-35 ボーレートの設定

7. 続いて表示されたダイアログで周波数を設定し、[OK]をクリックします。

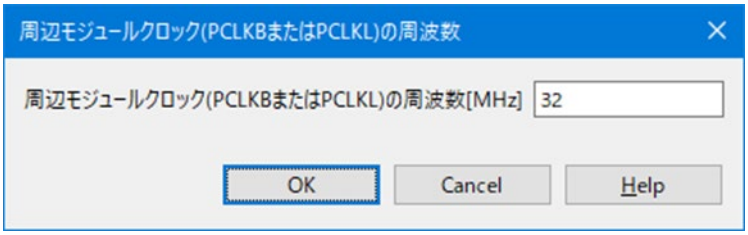


図 8-36 周辺モジュールクロックの周波数の設定

8. 続いて表示されたダイアログで、電圧を設定し、[OK]をクリックします。
使用するマイクロコントローラの電気的特性をご確認ください。
なお RL78/G22 の場合は、EV_{DD} はありませんので、V_{DD} への供給電圧に応じて設定を行ってください。

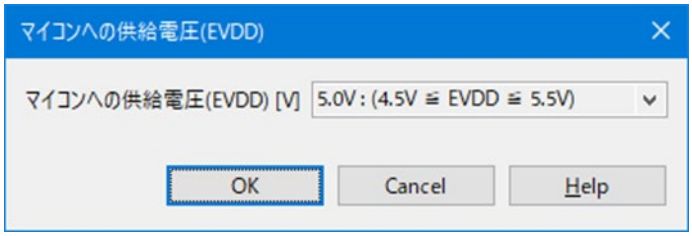


図 8-37 マイコンへの供給電圧の設定

9. 自動チューニングが開始されます。チューニングプロセスをガイドする[自動調整処理中]ダイアログを適宜確認し、ダイアログ中の指示に従い操作を進めていきます。

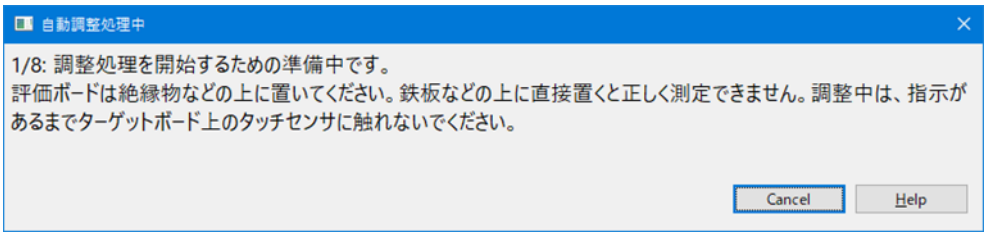


図 8-38 自動調整処理中ダイアログ

いくつかのプロセスを経て、以下のようなダイアログが表示されます。
ここでは、タッチ感度を計測します。ダイアログで表示されているタッチセンサを通常の圧力でタッチします。タッチセンサに触れているとき、バーグラフは右に増加し、数値で示すタッチカウント値が大きくなります。
タッチセンサに触れたまま、PC キーボードのいずれかのキーを押して計測を確定します。

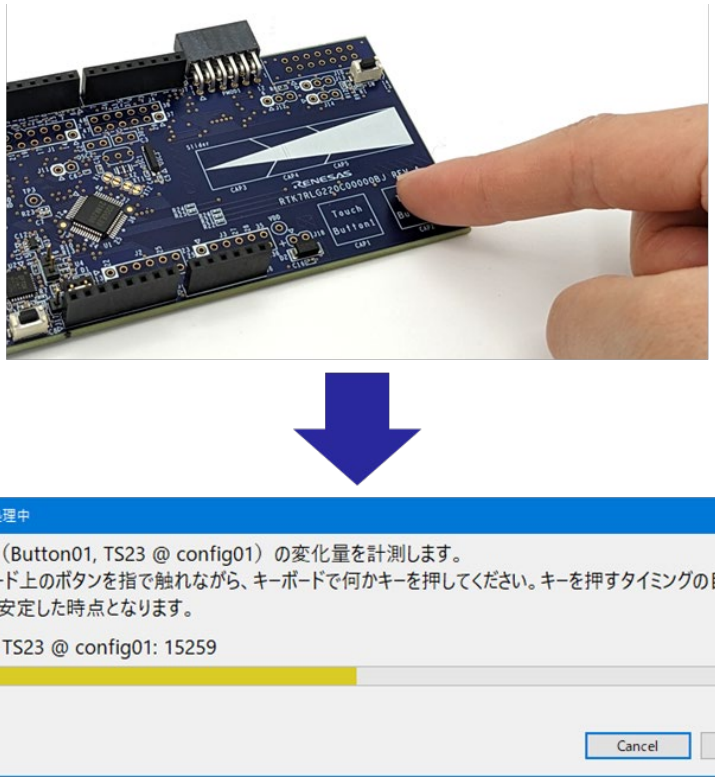


図 8-39 タッチ感度の計測（ボタン）

10. もう一方のタッチセンサについても同様に計測します。

11. スライダのタッチセンサについてもタッチ感度を計測します。ターゲットボード上にあるスライダを通常の圧力で上下または左右に 3～4 回なぞったあと、PC キーボードのいずれかのキーを押して計測を確定します。

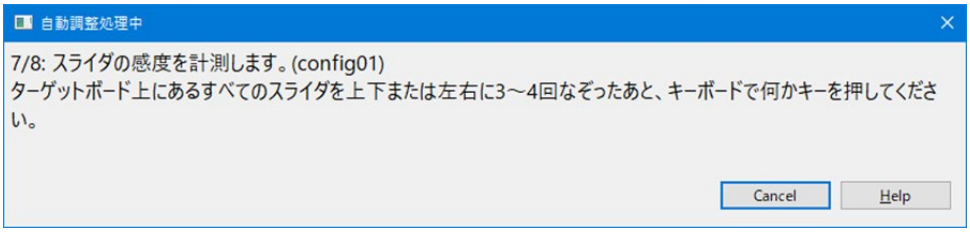


図 8-40 タッチ感度の計測 (スライダ)

12. チューニングが完了すると、以下のようなダイアログが表示され閾値を確認できます。この閾値はミドルウェアでタッチのイベント判定で使用されます。
[調整処理の継続]をクリックします。これでチューニングは終了です。

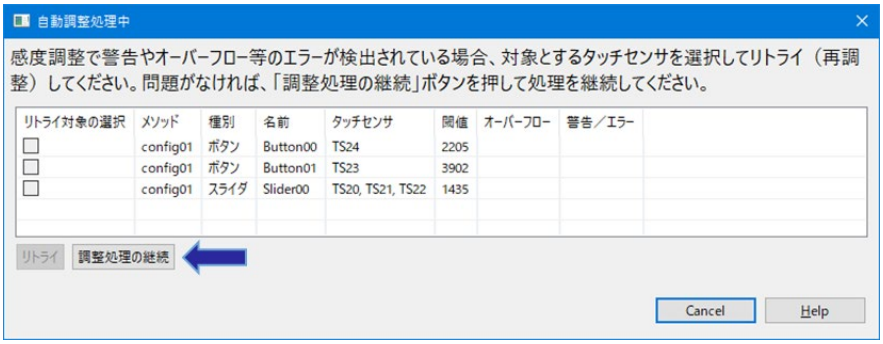


図 8-41 タッチセンサの閾値

13.[ファイルを出力する]をクリックし、調整結果が反映されたパラメータファイルを出力します。ファイルの出力先フォルダは 8.3 節で新規作成した”qe_gen”を選択し、ファイルを上書きします。
出力されるファイル群は、8.3 節の[調整用ファイルを出力する]で出力された以下のファイル群と同じファイル名です。

|- qe_touch_config.c ← 出力ファイル

|- qe_touch_config.h ← 出力ファイル

|- qe_touch_define.h ← 出力ファイル

|- qe_touch_sample.c ← 出力ファイル

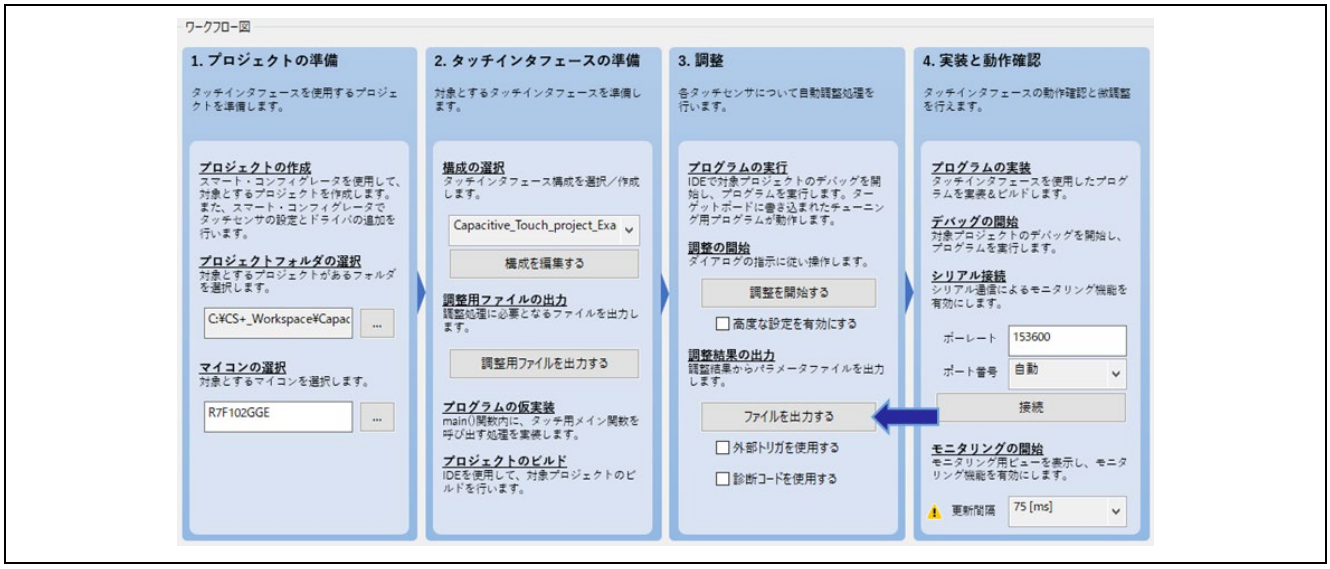


図 8-42 ファイルを出力

8.5　実装と動作確認

8.5.1　モニタリング

ワークフロー図の“実装と動作確認”に従い、設定します。

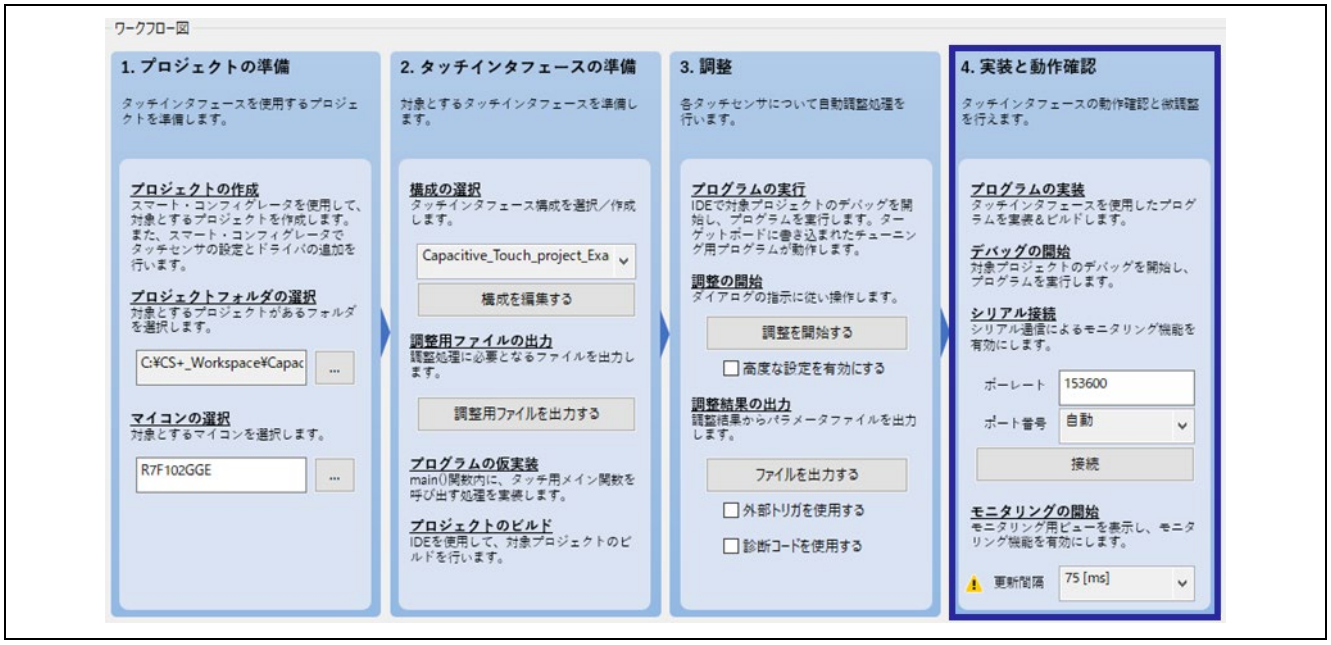


図 8-43 ワークフロー図 (実装と動作確認)

1. PC とターゲットボードを接続している USB ケーブルを外し、QE シリアル接続切り替えジャンパ (J16) をショートします。次に CS+と接続するため、PC とターゲットボードを USB ケーブルで再度接続します。

   CS+の  アイコンをクリックし、ビルドとプログラムの書き込みを行います。プログラム書き込み後のダウンロードが完了したら、  アイコンをクリックしてプログラムを停止させ、続いて  アイコンをクリックして切断します。

   切断後、PC とターゲットボードを接続している USB ケーブルを外し、QE シリアル接続切り替えジャンパ (J16) をオープンにします。
   この後に QE と接続するため、PC とターゲットボードを USB ケーブルで再度接続します。この時にターゲットボードは、書き込んだプログラムが動作し、QE との接続待機状態となります。

2. [接続]をクリックします。[接続]が[切断]に切り替わります。

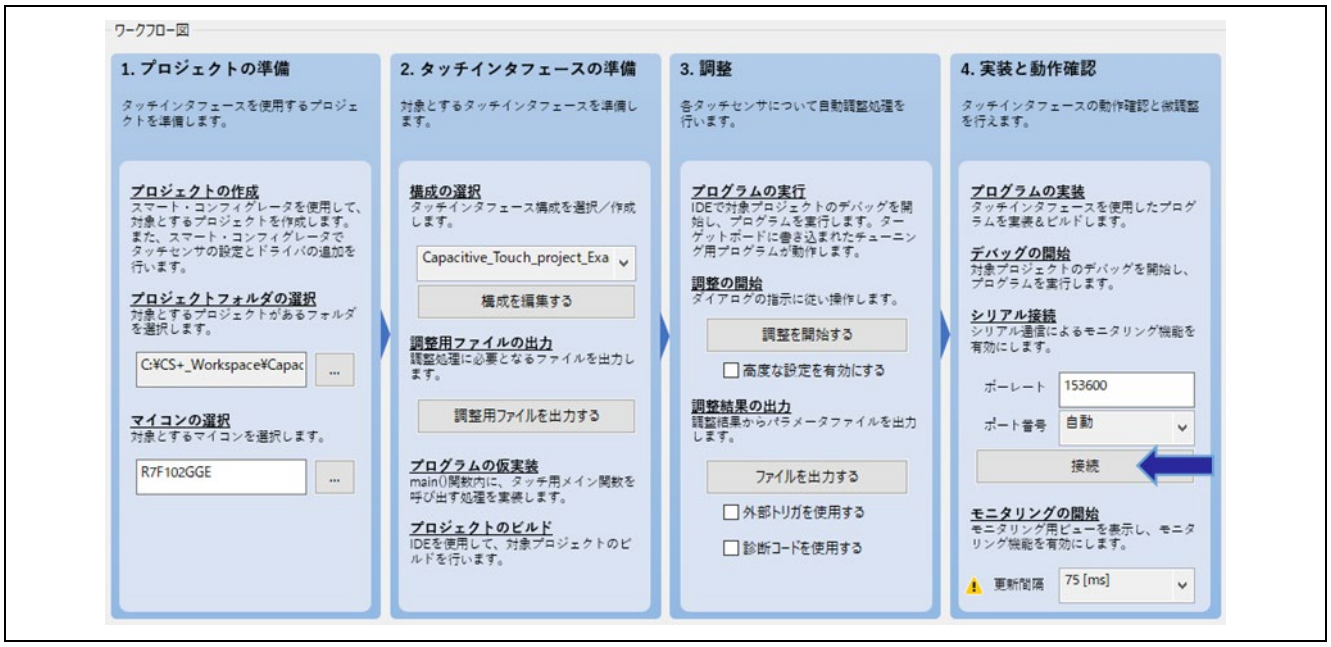


図 8-44 シリアル接続

3. QE 画面の左上にある“ボード・モニタ”パネルの[モニタリングを有効にする]をクリックします。”モニタリング機能: 無効”が“モニタリング機能: 有効”に切り替わります。

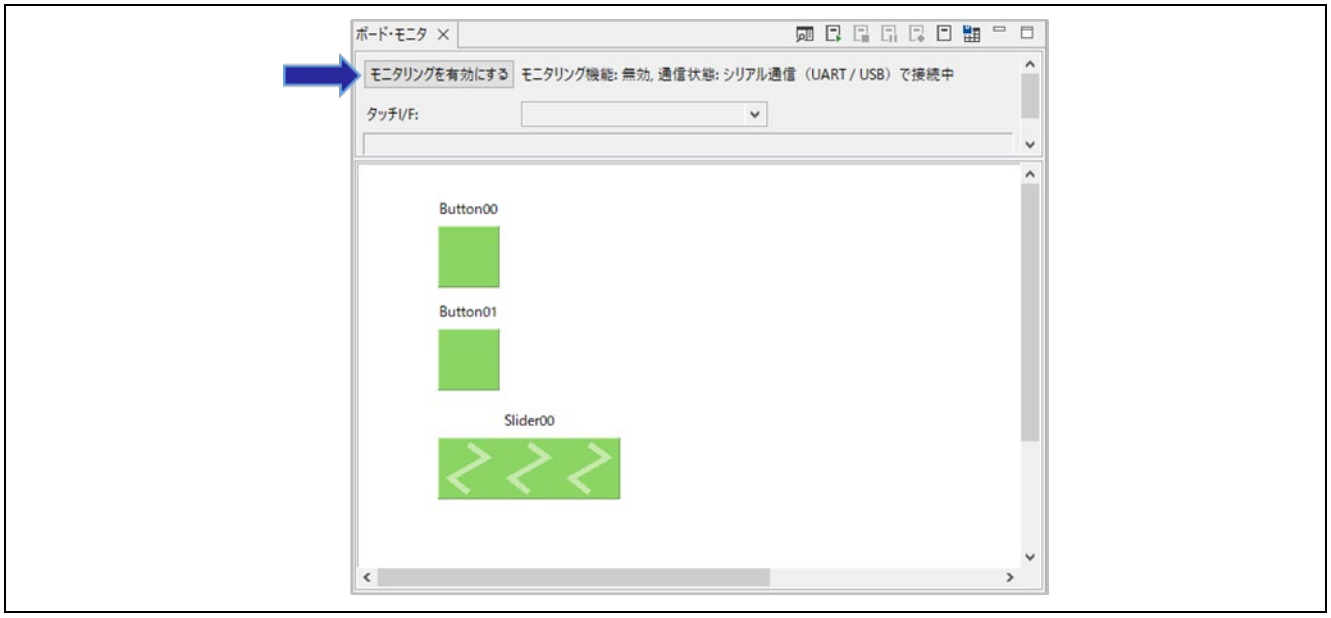


図 8-45 モニタリングの有効化

4. タッチセンサに触れると、その状態が指のアイコンで表されます。

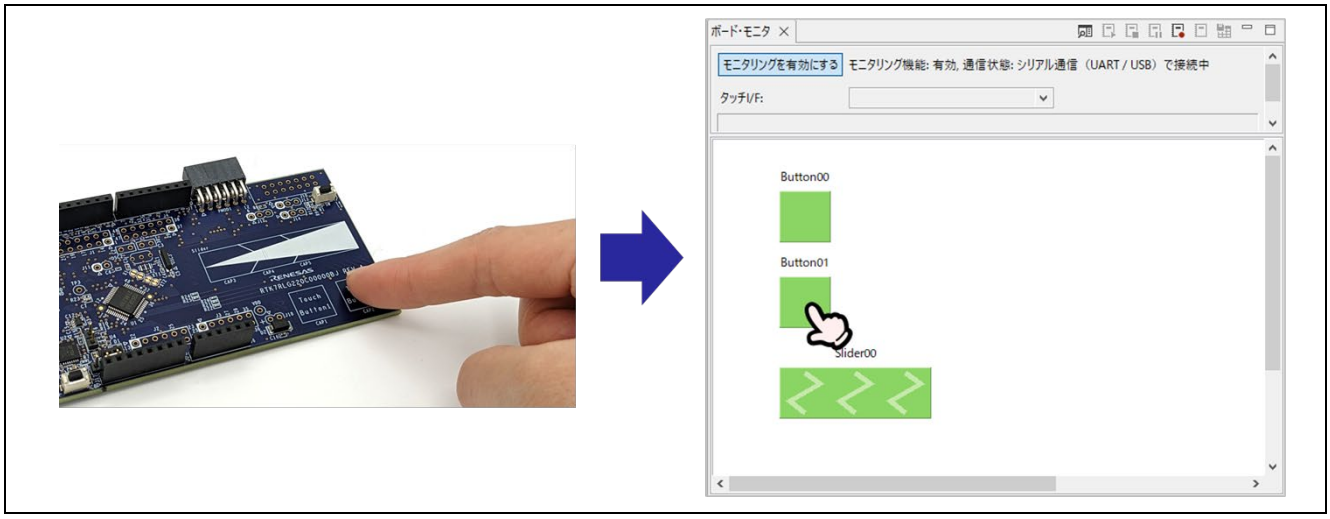


図 8-46 タッチセンサに触れた状態の表示

5. タッチカウント値をグラフィカルに表示します。
   A. “ワークフロー図” が表示されているパネルで[ステータス・チャート]タブをクリックします。

   B. 表示されたステータス・チャート画面の “タッチ I/F”の をクリックし、タッチインタフェースを選択します。

   C. グラフには実行中の値が表示されます。選択したタッチセンサに触れると、タッチカウント値がグラフ上で変化することを確認できます。
   緑のラインは閾値を表し、rm_touch ミドルウェアはボタンが操作/タッチされているかどうかを判断するために使用されます。
   グラフ下部の赤い矩形は、タッチカウント値が閾値を超えてタッチが検出されたことを表示しています。

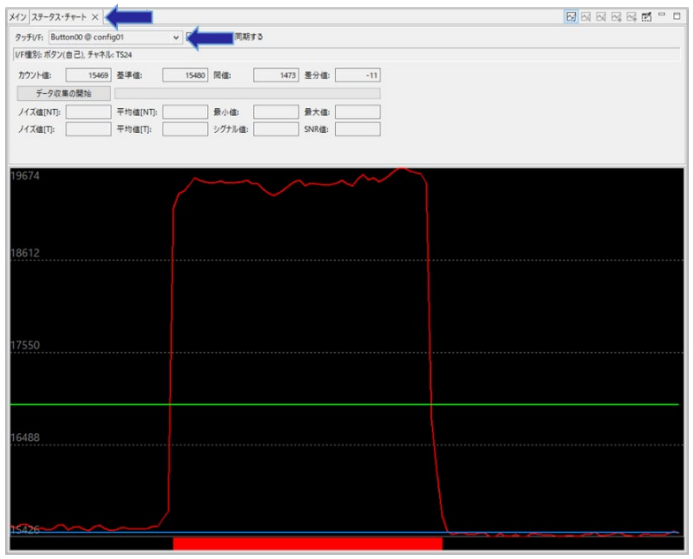


図 8-47 タッチカウント値のグラフ表示 (ボタン)

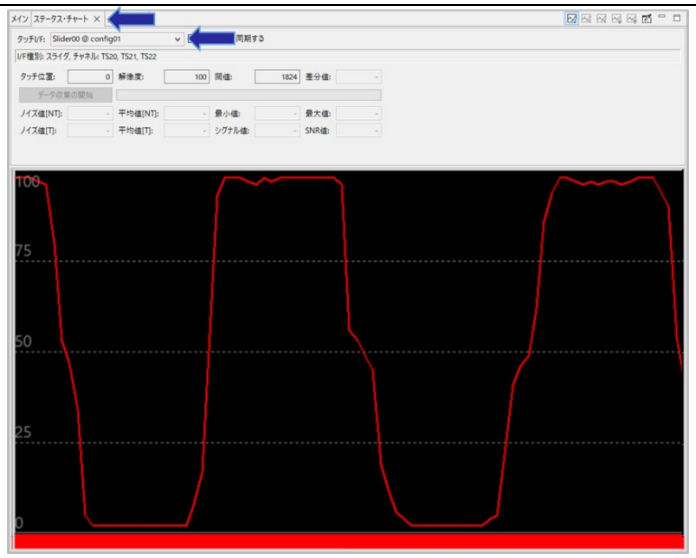


図 8-48 タッチカウント値のグラフ表示 (スライダ)

6. 必要に応じて、標準偏差を測定します。
   A. タッチセンサに触れていない状態で [データ収集の開始] をクリックします。タッチオフ状態を収集している間は電極に触れないでください。
   緑色のバーはデータの収集率を表しています。緑色のバーが右端まで達するとタッチオフ状態のデータ収集は完了します。

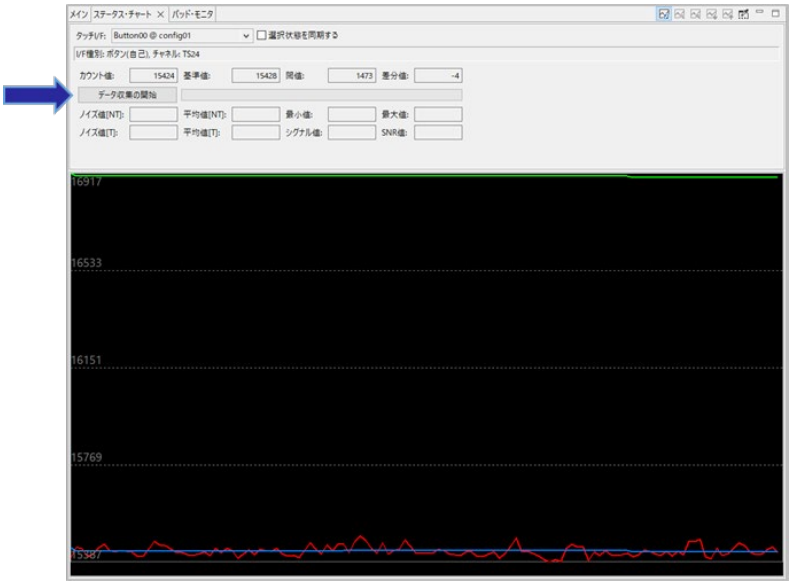


図 8-49 タッチオフ状態のデータ収集

   B. 緑色のバーが右端まで達したときに [データ収集の終了] をクリックしてください。

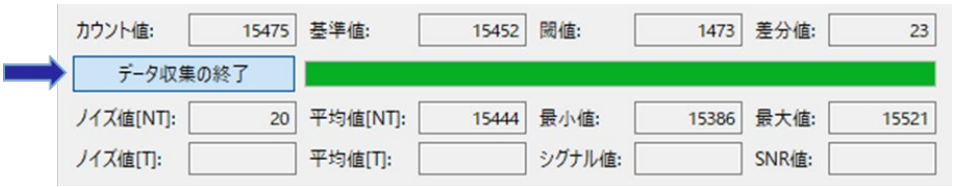


図 8-50 データの収集の終了

   C. 続いて同様の手順で、タッチセンサに触れている状態のデータの収集を行います。

   D. データの収集が完了すると SNR 値が表示されます。

図 8-51 SNR 値

7. 複数のタッチセンサのタッチカウント値をグラフィカルに表示します。
   QE 画面の左下にある“マルチ・ステータス・チャート”パネルで、表示するタッチセンサを選択します。

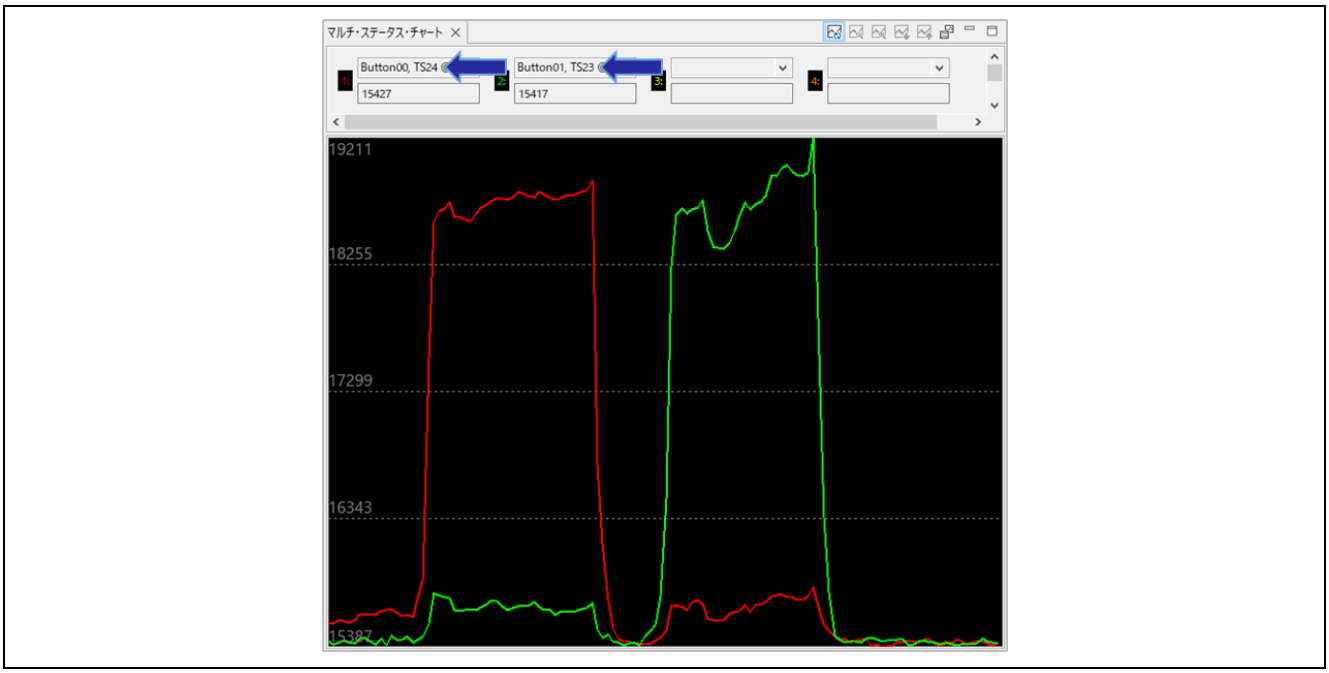


図 8-52 マルチ・ステータス・チャート

8. 必要に応じて、パラメータを手動で調整します。
   QE 画面の右側にある“パラメーター覧”パネルでパラメータを調整します。

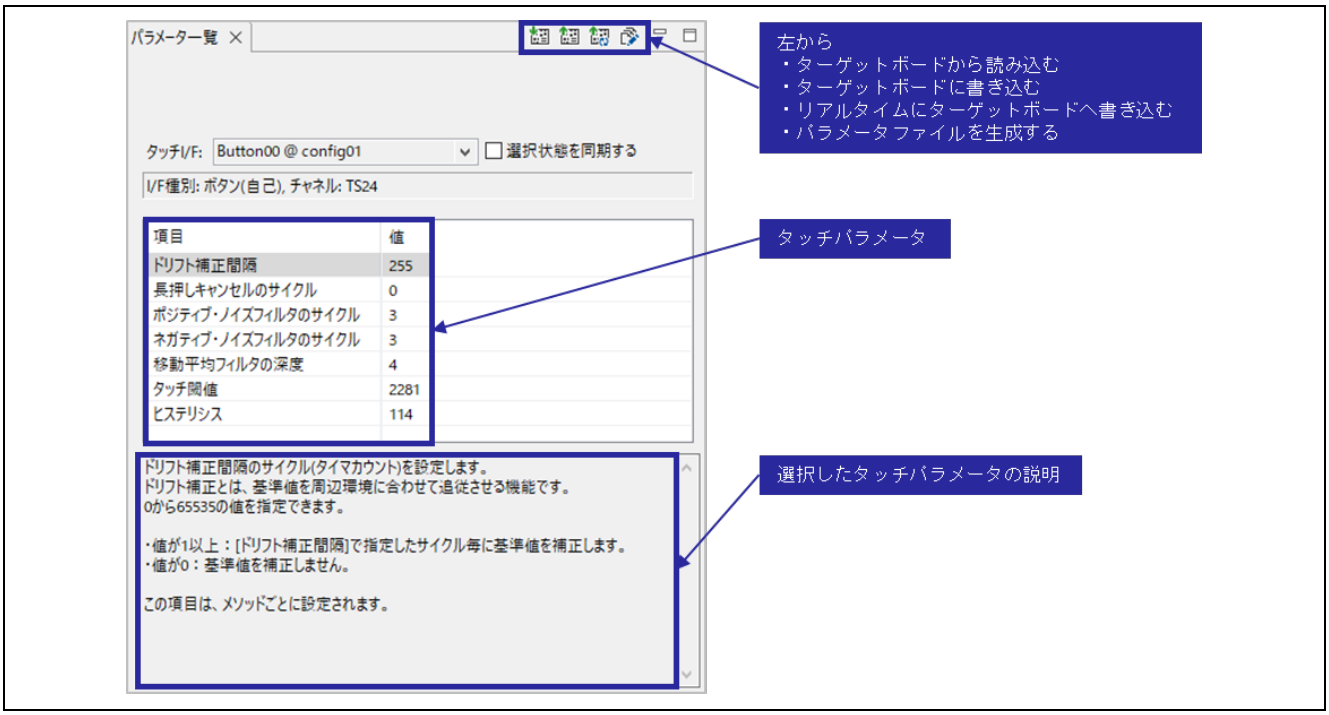


図 8-53 パラメータの調整

9. “モニタリング機能: 有効”の状態で [モニタリングを有効にする] をクリックして、モニタリングを終了します。

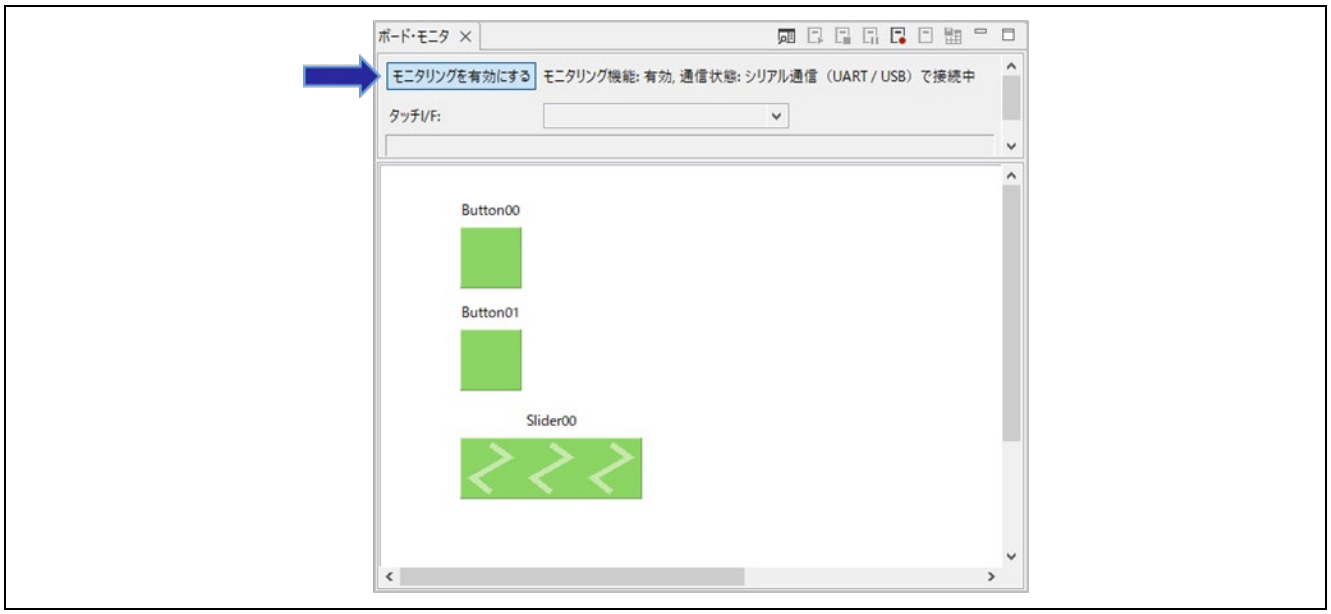


図 8-54 モニタリングを終了する

10.[切断] をクリックし、シリアル接続を切断します。

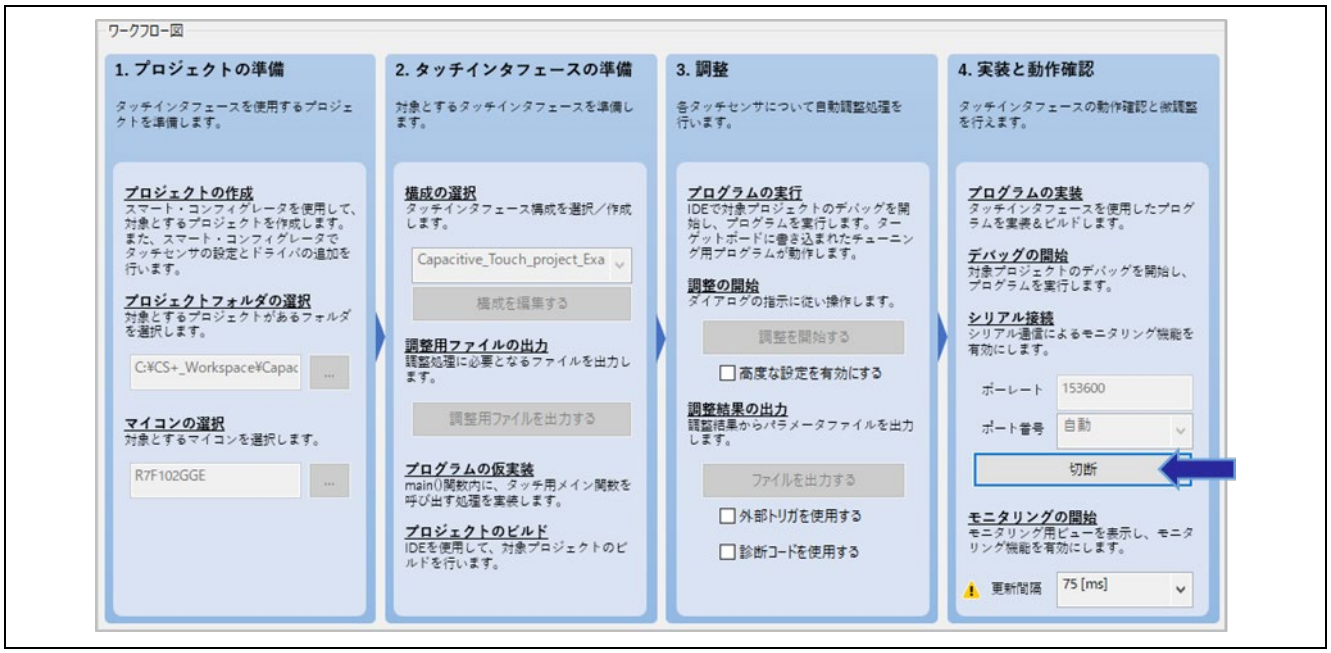


図 8-55 シリアル接続を切断

8.6 サンプルコード

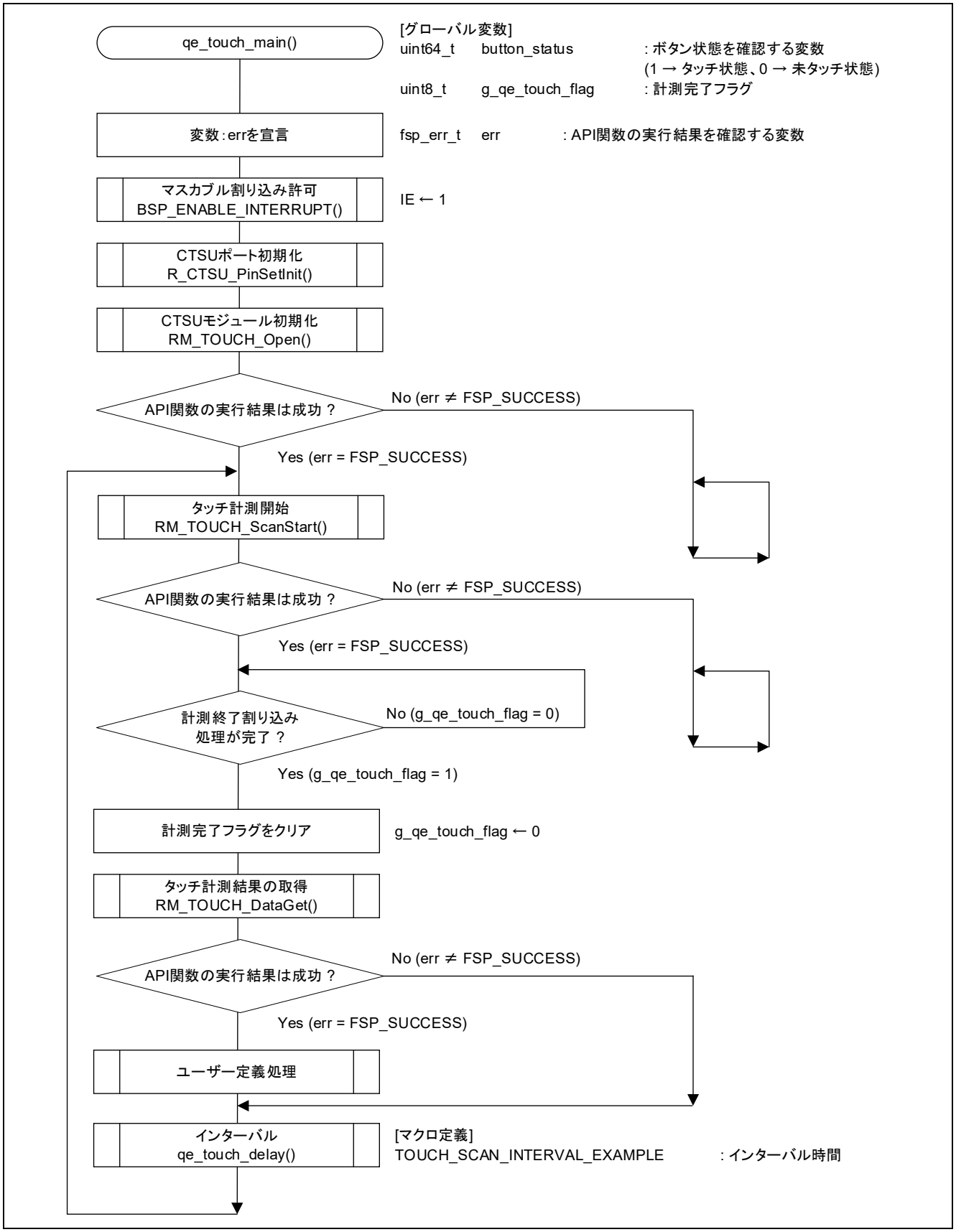
QE for Capacitive Touch から出力されるサンプルコード (qe_touch_sample.c) を以下に示します。

本サンプルコードでは、タッチ計測周期をソフトウェアタイマで生成します。

```
/***********************************************************************
*
* FILE : qe_sample_main.c
* DATE : 2022-02-14
* DESCRIPTION : Main Program for RL78
*
* NOTE:THIS IS A TYPICAL EXAMPLE.
*
***********************************************************************/
#include "qe_touch_config.h"
#define TOUCH_SCAN_INTERVAL_EXAMPLE (20 * 1000)       /* microseconds */

void R_CTSU_PinSetInit(void);
void qe_touch_main(void);
void qe_touch_delay(uint16_t delay_us);

uint64_t button_status;
#if (TOUCH_CFG_NUM_SLIDERS != 0)
uint16_t slider_position[TOUCH_CFG_NUM_SLIDERS];
#endif
#if (TOUCH_CFG_NUM_WHEELS != 0)
uint16_t wheel_position[TOUCH_CFG_NUM_WHEELS];
#endif

void qe_touch_main(void)
{
    fsp_err_t err;
    BSP_ENABLE_INTERRUPT();
    /* Initialize pins (function created by Smart Configurator) */
    R_CTSU_PinSetInit();
    /* Open Touch middleware */
    err = RM_TOUCH_Open(g_qe_touch_instance_config01.p_ctrl, g_qe_touch_instance_config01.p_cfg);
    if (FSP_SUCCESS != err)
    {
        while (true) {}
    }
```

```
    /* Main loop */
    while (true)
    {
        /* for [CONFIG01] configuration */
        err = RM_TOUCH_ScanStart(g_qe_touch_instance_config01.p_ctrl);
        if (FSP_SUCCESS != err)
        {
            while (true) {}
        }
        while (0 == g_qe_touch_flag) {}
        g_qe_touch_flag = 0;

        err = RM_TOUCH_DataGet(g_qe_touch_instance_config01.p_ctrl, &button_status, slider_position, NULL);
        if (FSP_SUCCESS == err)
        {
            /* TODO: Add your own code here. */
        }

        /* FIXME: Since this is a temporary process, so re-create a waiting process yourself. */
        qe_touch_delay(TOUCH_SCAN_INTERVAL_EXAMPLE);
    }
}

void qe_touch_delay(uint16_t delay_us)
{
    uint32_t i;
    uint32_t loops_required;
    uint16_t clock_mhz;
    clock_mhz = (uint16_t)(R_BSP_GetFclkFreqHz() / 1000000);
    if (0 == clock_mhz)
    {
        clock_mhz = 1;
    }
    loops_required = ((uint32_t)delay_us * (uint32_t)clock_mhz);
    loops_required /= 20;
    for (i = 0; i < loops_required; i++)
    {
        BSP_NOP();
    }
}
```

8.7　フローチャート


qe_touch_main()
[グローバル変数]
uint64_t button_status : ボタン状態を確認する変数
(1 → タッチ状態、0 → 未タッチ状態)
uint8_t g_qe_touch_flag : 計測完了フラグ
変数:errを宣言
fsp_err_t err : API関数の実行結果を確認する変数
マスカブル割り込み許可
BSP_ENABLE_INTERRUPT()
IE ← 1
CTSUポート初期化
R_CTSU_PinSetInit()
CTSUモジュール初期化
RM_TOUCH_Open()
API関数の実行結果は成功？
No (err ≠ FSP_SUCCESS)
Yes (err = FSP_SUCCESS)
タッチ計測開始
RM_TOUCH_ScanStart()
API関数の実行結果は成功？
No (err ≠ FSP_SUCCESS)
Yes (err = FSP_SUCCESS)
計測終了割り込み
処理が完了？
No (g_qe_touch_flag = 0)
Yes (g_qe_touch_flag = 1)
計測完了フラグをクリア
g_qe_touch_flag ← 0
タッチ計測結果の取得
RM_TOUCH_DataGet()
API関数の実行結果は成功？
No (err ≠ FSP_SUCCESS)
Yes (err = FSP_SUCCESS)
ユーザー定義処理
インターバル
qe_touch_delay()
[マクロ定義]
TOUCH_SCAN_INTERVAL_EXAMPLE : インターバル時間

9. 応用例

9.1 ハードウェアタイマでのタッチ計測

ハードウェアタイマ (32 ビット・インターバル・タイマの 8 ビット・カウンタ・モードによるインターバル・タイマ機能) を使用したタッチ計測周期の実装例を説明します。

また、動作確認のため、センサ (ボタン) へのタッチに応じてターゲットボード上の LED を点灯/消灯させます。

9.1.1 スマート・コンフィグレータの設定

1. スマート・コンフィグレータの [クロック] タブを選択し、インターバル・タイマに使用するクロック (fSXP) を設定します。また、XT1 発振回路のチェックを外します。

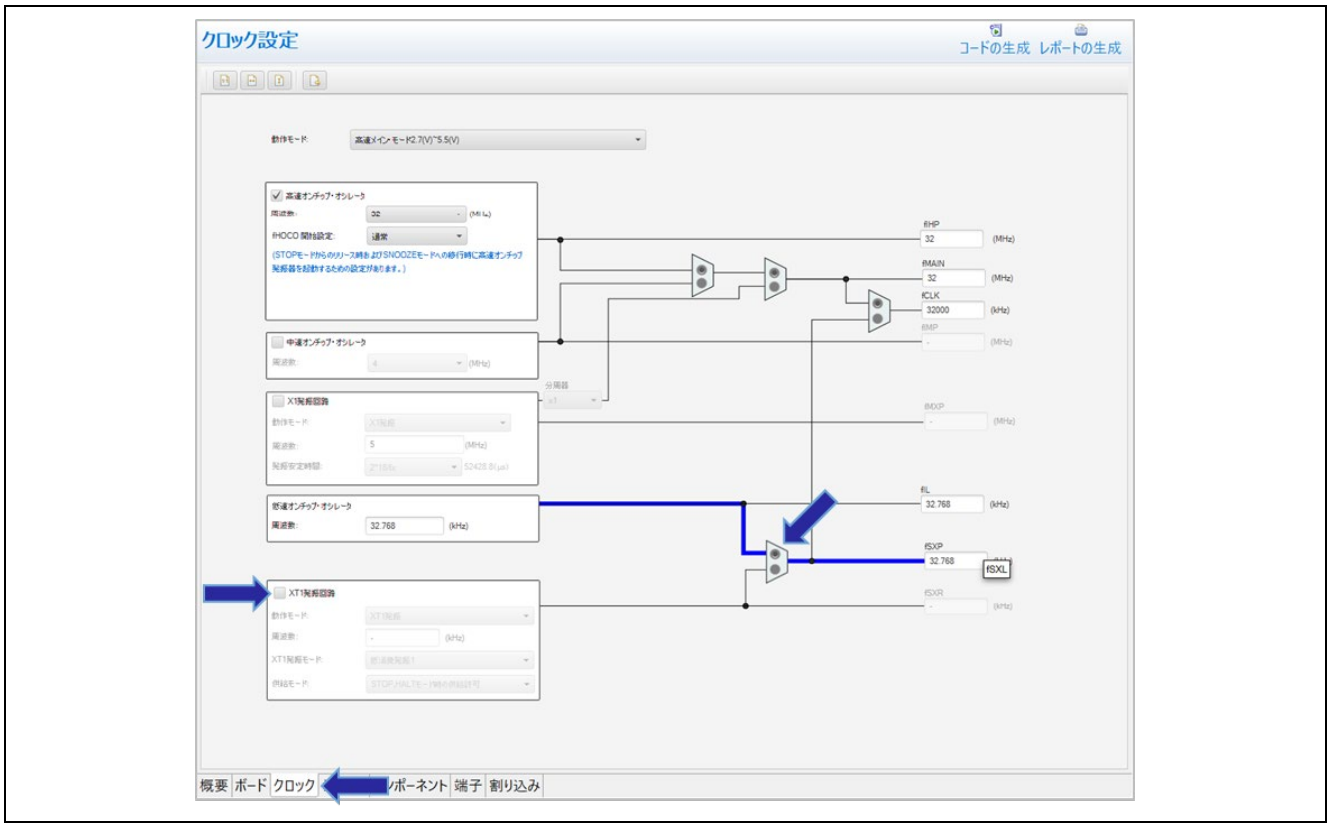


図 9-1 クロックの設定

2. [コンポーネント] タブを選択し、  クリックして”コンポーネントの追加”ダイアログを開きます。“インターバル・タイマ”モジュールを選択し、[次へ] をクリックします。
インターバル・タイマの構成を以下のように設定し、[終了] をクリックします。

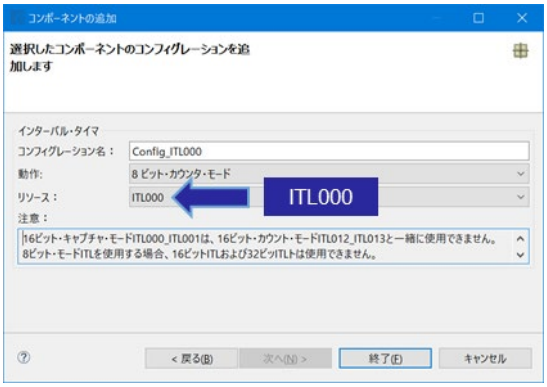


図 9-2 インターバル・タイマの構成

3. 追加した”インターバル・タイマ”モジュールを選択し、クロックなどを設定します。

図 9-3 Config_ITL000 の設定

4. LED に使用する端子の設定を行います。”ポート”モジュールで”P62”を High レベル出力に設定します。

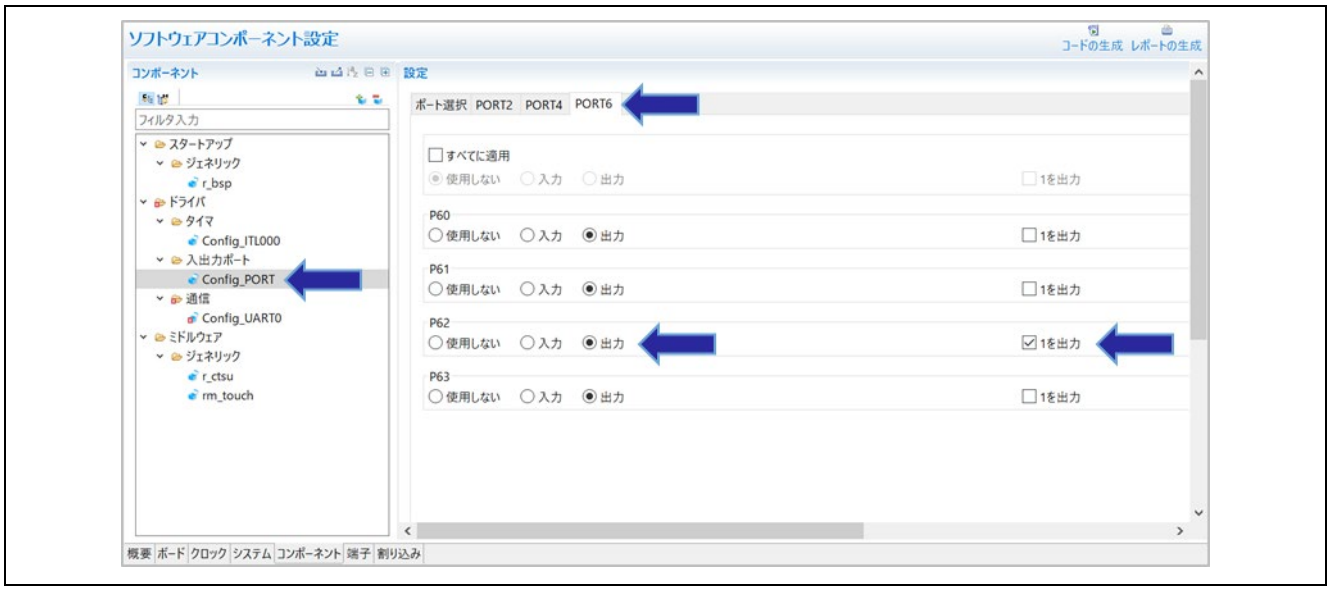


図 9-4 P62 の設定

5. スマート・コンフィグレータの右上の アイコンをクリックして、コードの生成を行います。

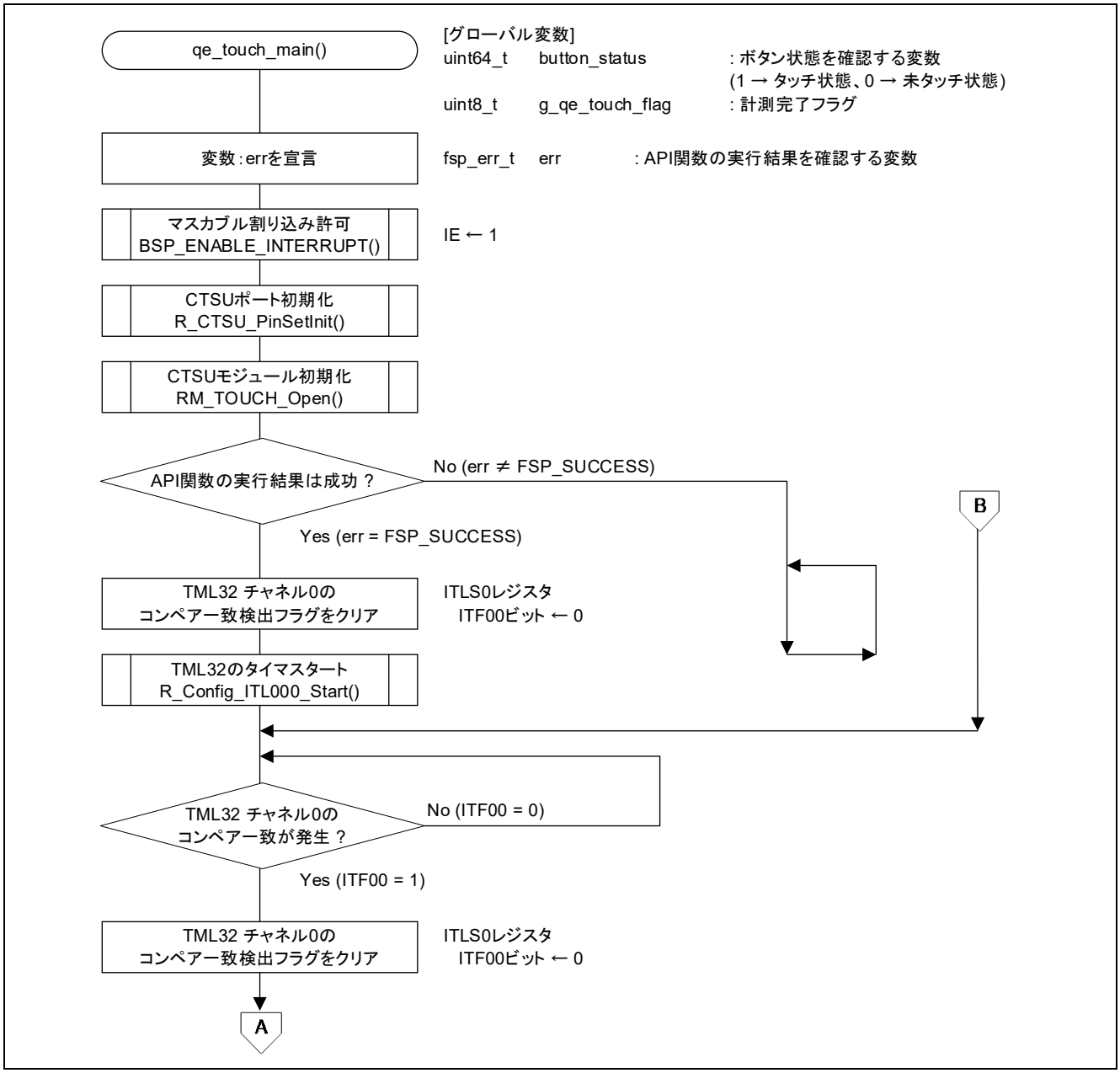
9.1.2　サンプルコード

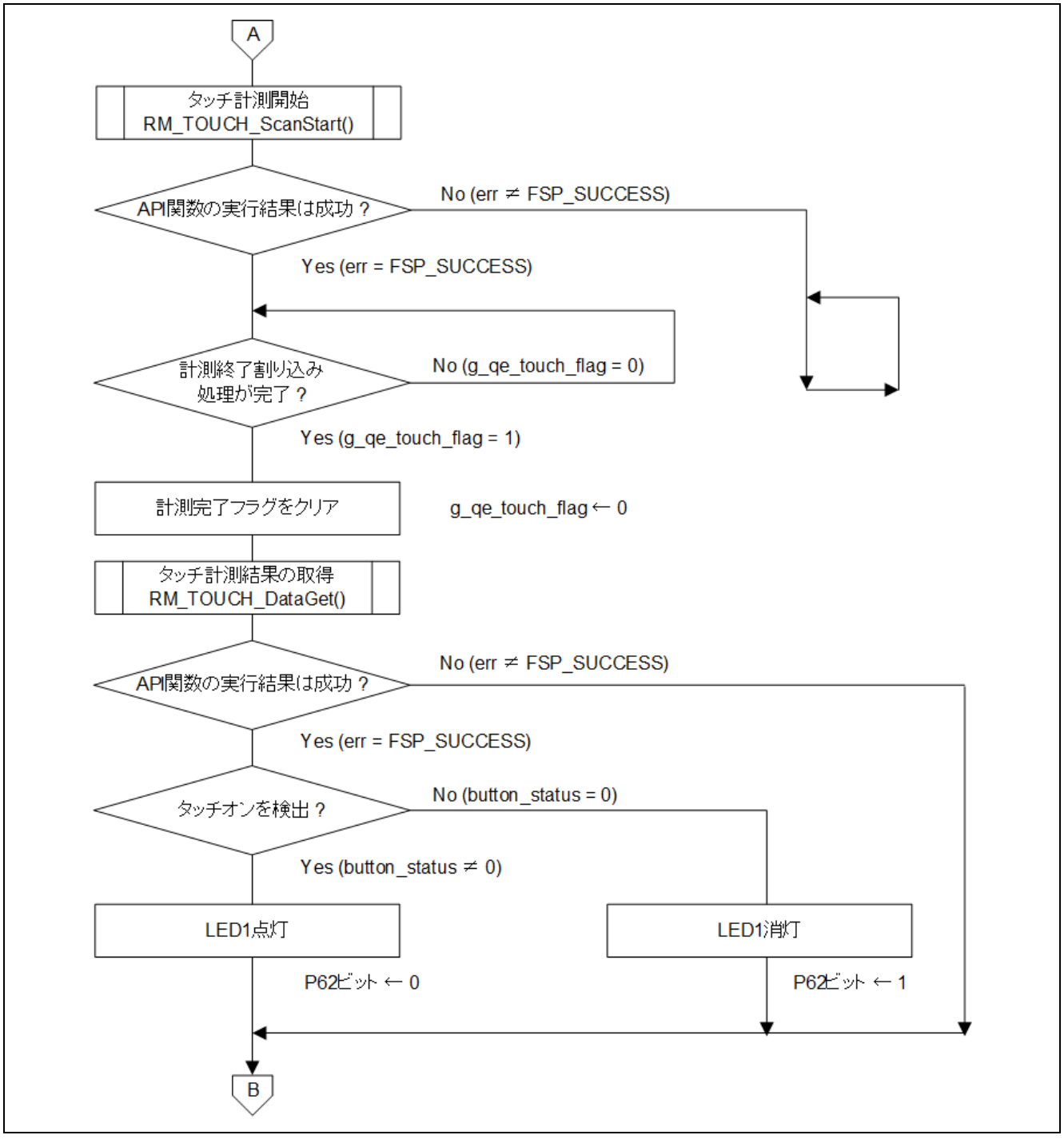
ハードウェアタイマでのタッチ計測のプログラム実装例（qe_touch_sample.c）を以下に示します。

```
/***********************************************************************
*
* FILE : qe_sample_main.c
* DATE : 2022-12-15
* DESCRIPTION : CTSU2L Program for RL78
*
* NOTE:THIS IS A TYPICAL EXAMPLE.
*
***********************************************************************/
#include "qe_touch_config.h"
#include "Config_ITL000.h"

void R_CTSU_PinSetInit(void);
void qe_touch_main(void);

uint64_t button_status;
#if (TOUCH_CFG_NUM_SLIDERS != 0)
uint16_t slider_position[TOUCH_CFG_NUM_SLIDERS];
#endif
#if (TOUCH_CFG_NUM_WHEELS != 0)
uint16_t wheel_position[TOUCH_CFG_NUM_WHEELS];
#endif

void qe_touch_main(void)
{
    fsp_err_t err;
    BSP_ENABLE_INTERRUPT();
    /* Initialize pins (function created by Smart Configurator) */
    R_CTSU_PinSetInit();
    /* Open Touch middleware */
    err = RM_TOUCH_Open(g_qe_touch_instance_config01.p_ctrl, g_qe_touch_instance_config01.p_cfg);
    if (FSP_SUCCESS != err)
    {
        while (true) {}
    }
```

```
    ITLS0 &= ~_01_ITL_CHANNEL0_COUNT_MATCH_DETECTE;

    R_Config_ITL000_Start();

    /* Main loop */
    while (true)
    {
        while (_00_ITL_CHANNEL0_COUNT_MATCH_NOT_DETECTE == (ITLS0 &
_01_ITL_CHANNEL0_COUNT_MATCH_DETECTE)) {}
        ITLS0 &= ~_01_ITL_CHANNEL0_COUNT_MATCH_DETECTE;

        /* for [CONFIG01] configuration */
        err = RM_TOUCH_ScanStart(g_qe_touch_instance_config01.p_ctrl);
        if (FSP_SUCCESS != err)
        {
            while (true) {}
        }
        while (0 == g_qe_touch_flag) {}
        g_qe_touch_flag = 0;

        err = RM_TOUCH_DataGet(g_qe_touch_instance_config01.p_ctrl, &button_status, slider_position, NULL);
        if (FSP_SUCCESS == err)
        {
            /* TODO: Add your own code here. */
            if (0 != button_status)
            {
                P6_bit.no2 = 0;
            }
            else
            {
                P6_bit.no2 = 1;
            }
        }
    }
}
```

9.1.3 フローチャート


qe_touch_main()
[グローバル変数]
uint64_t button_status : ボタン状態を確認する変数
(1 → タッチ状態、0 → 未タッチ状態)
uint8_t g_qe_touch_flag : 計測完了フラグ
変数:errを宣言
fsp_err_t err : API関数の実行結果を確認する変数
マスカブル割り込み許可
BSP_ENABLE_INTERRUPT()
IE ← 1
CTSUポート初期化
R_CTSU_PinSetInit()
CTSUモジュール初期化
RM_TOUCH_Open()
API関数の実行結果は成功 ?
No (err ≠ FSP_SUCCESS)
Yes (err = FSP_SUCCESS)
B
TML32 チャネル0の
コンペアー致検出フラグをクリア
ITLS0レジスタ
ITF00ビット ← 0
TML32のタイマスタート
R_Config_ITL000_Start()
TML32 チャネル0の
コンペアー致が発生 ?
No (ITF00 = 0)
Yes (ITF00 = 1)
TML32 チャネル0の
コンペアー致検出フラグをクリア
ITLS0レジスタ
ITF00ビット ← 0
A


A
タッチ計測開始
RM_TOUCH_ScanStart()
API関数の実行結果は成功？
No (err ≠ FSP_SUCCESS)
Yes (err = FSP_SUCCESS)
計測終了割り込み
処理が完了？
No (g_qe_touch_flag = 0)
Yes (g_qe_touch_flag = 1)
計測完了フラグをクリア
g_qe_touch_flag ← 0
タッチ計測結果の取得
RM_TOUCH_DataGet()
API関数の実行結果は成功？
No (err ≠ FSP_SUCCESS)
Yes (err = FSP_SUCCESS)
タッチオンを検出？
No (button_status = 0)
Yes (button_status ≠ 0)
LED1点灯
LED1消灯
P62ビット ← 0
P62ビット ← 1
B

10. 参考ドキュメント

- RL78/G22 ユーザーズマニュアル ハードウェア編 (R01UH0978)
- RL78 ファミリ ユーザーズマニュアル ソフトウェア編 (R01US0015)
- RL78/G22 Fast Prototyping Board ユーザーズマニュアル (R20UT5121)
- RL78/G23 静電容量タッチ評価システム ユーザーズマニュアル (R12UZ0095)
  (最新版をルネサス エレクトロニクスホームページから入手してください)

- アプリケーションノート RL78 ファミリ
  スタンドアロン版 QE を使用した静電容量タッチアプリケーションの開発 (R01AN6574)
- アプリケーションノート シリアルポートを使用した RL78 デバッグ機能 (R20AN0632)
- アプリケーションノート RL78 ファミリ
  QE と SIS を使用した静電容量タッチアプリケーションの開発 (R01AN5512)
- アプリケーションノート RL78 ファミリ 静電容量センサユニット (CTSU2L) 動作説明 (R01AN5744)
- アプリケーションノート RL78 ファミリ CTSU モジュール Software Integration System (R11AN0484)
- アプリケーションノート RL78 ファミリ TOUCH モジュール Software Integration System (R11AN0485)
- アプリケーションノート 静電容量センサマイコン 静電容量タッチ電極デザインガイド (R30AN0389)
- アプリケーションノート RL78 ファミリ
  RL78/G23 静電容量タッチ低消費電力ガイド (SNOOZE 機能) (R01AN5886)
  RL78/G23 グループ 静電容量タッチ低消費電力ガイド (SMS 機能) (R01AN6670)
  (最新版をルネサス エレクトロニクスホームページから入手してください)

- テクニカルアップデート／テクニカルニュース
  (最新の情報をルネサス エレクトロニクスホームページから入手してください)

ホームページ

- ルネサス エレクトロニクスホームページ
  http://www.renesas.com/

- QE for Capacitive Touch 関連ページ
  https://www.renesas.com/qe-capacitive-touch

- 静電容量センサユニット関連ページ
  https://www.renesas.com/solutions/touch-key

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	Mar.20.23	-	初版

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、本ドキュメントおよびテクニカルアップデートを参照してください。

1. 静電気対策
   CMOS 製品の取り扱いの際は静電気防止を心がけてください。CMOS 製品は強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には、当社が出荷梱包に使用している導電性のトレーやマガジンケース、導電性の緩衝材、金属ケースなどを利用し、組み立て工程にはアースを施してください。プラスチック板上に放置したり、端子を触ったりしないでください。また、CMOS 製品を実装したボードについても同様の扱いをしてください。
2. 電源投入時の処置
   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. 電源オフ時における入力信号
   当該製品の電源がオフ状態のときに、入力信号や入出力プルアップ電源を入れないでください。入力信号や入出力プルアップ電源からの電流注入により、誤動作を引き起こしたり、異常電流が流れ内部素子を劣化させたりする場合があります。資料中に「電源オフ時における入力信号」についての記載のある製品は、その内容を守ってください。
4. 未使用端子の処理
   未使用端子は、「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。
5. クロックについて
   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。
6. 入力端子の印加波形
   入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。CMOS 製品の入力がノイズなどに起因して、V_{IL}（Max.）から V_{IH}（Min.）までの領域にとどまるような場合は、誤動作を引き起こす恐れがあります。入力レベルが固定の場合はもちろん、V_{IL}（Max.）から V_{IH}（Min.）までの領域を通過する遷移期間中にチャタリングノイズなどが入らないように使用してください。
7. リザーブアドレス（予約領域）のアクセス禁止
   リザーブアドレス（予約領域）のアクセスを禁止します。アドレス領域には、将来の拡張機能用に割り付けられている リザーブアドレス（予約領域）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。
8. 製品間の相違について
   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、フラッシュメモリ、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、ノイズ幅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。

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。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、お客様の責任において、お客様の機器・システムを設計ください。これらの使用に起因して生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
2. 当社製品または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またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 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. 当社製品を組み込んだ製品の輸出入、製造、販売、利用、配布その他の行為を行うにあたり、第三者保有の技術の利用に関するライセンスが必要となる場合、当該ライセンス取得の判断および取得はお客様の責任において行ってください。
5. 当社製品を、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング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。
6. 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）、交通制御（信号）、大規模通信機器、金融端末基幹システム、各種安全制御装置等
   当社製品は、データシート等により高信頼性、Harsh environment 向け製品と定義しているものを除き、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（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
7. あらゆる半導体製品は、外部攻撃からの安全性を 100%保証されているわけではありません。当社ハードウェア／ソフトウェア製品にはセキュリティ対策が組み込まれているものもありますが、これによって、当社は、セキュリティ脆弱性または侵害（当社製品または当社製品が使用されているシステムに対する不正アクセス・不正使用を含みますが、これに限りません。）から生じる責任を負うものではありません。当社は、当社製品または当社製品が使用されたあらゆるシステムが、不正な改変、攻撃、ウイルス、干渉、ハッキング、データの破壊または窃盗その他の不正な侵入行為（「脆弱性問題」といいます。）によって影響を受けないことを保証しません。当社は、脆弱性問題に起因しまたはこれに関連して生じた損害について、一切責任を負いません。また、法令において認められる限りにおいて、本資料および当社ハードウェア／ソフトウェア製品について、商品性および特定目的との合致に関する保証ならびに第三者の権利を侵害しないことの保証を含め、明示または黙示のいかなる保証も行いません。
8. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合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、データシート等において高信頼性、Harsh environment 向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他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。
10. 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。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. 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。当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
12. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。
13. 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。
14. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

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直接的、間接的に支配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。

(Rev.5.0-1　2020.10)

本社所在地

〒135-0061 東京都江東区豊洲 3-2-24（豊洲フォレシア）

www.renesas.com

お問合せ窓口

弊社の製品や技術、ドキュメントの最新情報、最寄の営業お問合せ窓口に関する情報などは、弊社ウェブサイトをご覧ください。

www.renesas.com/contact/

商標について

ルネサスおよびルネサスロゴはルネサス エレクトロニクス株式会社の商標です。すべての商標および登録商標は、それぞれの所有者に帰属します。