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

要旨

本アプリケーションノートでは、RL78 MCU を使用した静電容量タッチアプリケーションの開発手順を説明します。

本アプリケーションノートは、“e² studio、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 を用いた開発ガイドです。

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。

動作確認デバイス

RL78/G23

対象デバイス

RL78/G22

RL78/G23

RL78/L23

静電容量センサユニット(CTSU)をサポートする RL78 ファミリ

本アプリケーションノートを他のマイコンへ適用する場合、そのマイコンの仕様にあわせて変更し、十分評価してください。

注.　本アプリケーションノートの以下の内容について、RL78/G16 はサポート対象外です。

- **9. 静電容量タッチセンサ・チューニング向けデバッグ構成の設定変更**
- **10. QE for Capacitive Touch を使用した静電容量タッチセンサ・チューニング**
- **13. “式”ウィンドウと QE for Capacitive Touch によるモニタリング　の手順 9～13 (エミュレータを介したモニタリングの説明)**

RL78/G16 は、シリアル通信(UART)を介したチューニングおよびモニタリングのみサポートしています。そのため、RL78/G16 を使用する場合は、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)」もあわせてご参照ください。

目次

1. 概要 4

2. 開発ツールについて 4

3. 動作確認環境 5
3.1 ハードウェア設定 5
3.2 ソフトウェア設定 6
3.3 動作確認条件 7
3.3.1 オプション・バイト設定 7

4. 静電容量タッチアプリケーション開発手順の概要 9

5. アプリケーション例の概要 9

6. 静電容量タッチアプリケーション開発の開始～モジュール追加 10
6.1 新規プロジェクトの作成 10
6.2 スマート・コンフィグレータによるモジュール追加 13
6.2.1 CTSU コンポーネント設定 21
6.2.2 Touch コンポーネント設定 23
6.2.3 PORT コンポーネント設定 24
6.2.4 LVD コンポーネント設定 26
6.2.5 BSP コンポーネント設定 26
6.3 コード生成 27

7. [追加機能] UART を使用したシリアル通信モニタの設定 (1/3) 28
7.1 Touch コンポーネント設定(シリアル通信モニタ用) 28
7.2 UART 通信コンポーネント設定 29

8. 静電容量タッチインタフェース作成 36

9. 静電容量タッチセンサ・チューニング向けデバッグ構成の設定変更 41

10. QE for Capacitive Touch を使用した静電容量タッチセンサ・チューニング 45

11. アプリケーションに rm_touch ミドルウェアの API コールを追加 51

12. [追加機能] UART を使用したシリアル通信モニタの設定 (2/3) 54

13. “式”ウィンドウと QE for Capacitive Touch によるモニタリング 56

14. [追加機能] UART を使用したシリアル通信モニタの設定 (3/3) 67

15. qe_touch_sample.c (ソフトウェアタイマ使用例) 73

16. [応用例] ハードウェアタイマでのタッチ計測 75
16.1 スマート・コンフィグレータの設定 (ハードウェアタイマ) 75

16.2　フローチャート(ハードウェアタイマ使用例)..... 79
16.3　qe_touch_sample.c (ハードウェアタイマ使用例) 81

17.　参考ドキュメント.....83

ホームページとサポート窓口.....83

1. 概要

本アプリケーションノートでは、RL78 MCU を使用した静電容量タッチ機能をシステムに組み込む以下の手順を説明します。

- RL78/G23 搭載静電容量タッチ評価システムを使用したスマート・コンフィグレータによるプロジェクト作成
- QE for Capacitive Touch によるタッチインタフェース作成とチューニング、モニタリング

2. 開発ツールについて

本アプリケーションノートでは、実際に動作するアプリケーションの作成手順を簡単に紹介します。このアプリケーション例で使用されている各ツールに関する質問、より詳細な使用方法に関しては、

e^2 studio/スマート・コンフィグレータ、Software Integration System (SIS) のドライバ/ミドルウェア、Renesas Code Generator や QE for Capacitive Touch のヘルプ (e^2 studio のヘルプに含まれています) などのドキュメントを参照してください。

3.　動作確認環境

サンプルコードの動作確認環境を示します。

3.1　ハードウェア設定

表 3-1 に動作確認環境(ハードウェア)を示します。

表 3-1 動作確認環境(ハードウェア)

項目	内容
使用マイコン	RL78/G23 (R7F100GSN2DFB)
ターゲットボード	RL78/G23 静電容量タッチ評価システム (製品型名 : RTK0EG0030S01001BJ)
エミュレータ	E2 エミュレータ Lite (RTE0T0002LKCE00000R)

表 3-2 に本アプリケーションでのターゲットボードにおけるジャンパの設定を示します。ボードへの電源供給は、エミュレータを使用します。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。

図 3-1 のように PC とターゲットボードを E2 エミュレータ Lite と USB ケーブルで接続します。

表 3-2 ターゲットボードのジャンパ設定

Reference	回路グループ	ジャンパ設定	機能
JP1	電源	1-2 オープン (初期設定から変更)	USB 電源をボードデバイスに供給しない (理由: エミュレータから電源供給するためです)
JP2		クローズ (初期設定)	ボードデバイス電源を MCU に供給
JP3		1-2 クローズ (初期設定)	JP1 の電源をボードデバイスに供給
JP4			

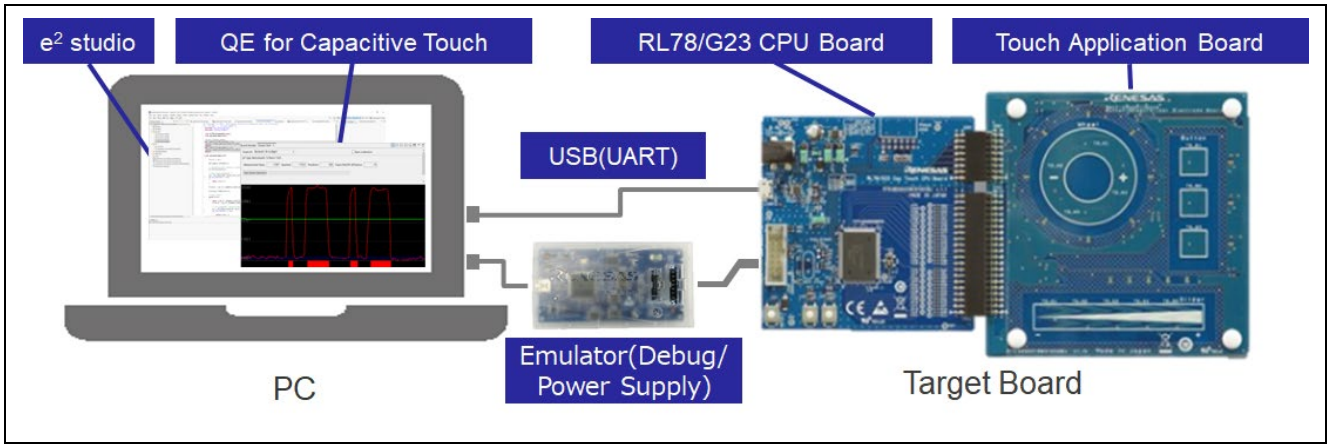

図 3-1 PC とターゲットボードの接続

3.2　ソフトウェア設定

表 3-3 に動作確認環境(ソフトウェア)を示します。

表 3-3 動作確認環境(ソフトウェア)

項目	内容	バージョン
統合開発環境(IDE)	e² studio	Version 2025-07 (25.7.0) (Version 2025-07 (25.7.0)以降)
コンパイラ	CC-RL	V1.15.00 (V1.12.00 以降)
静電容量式タッチセンサ対応開発支援ツール	e² studio プラグイン版 QE for Capacitive Touch	V4.2.0 (V4.2.0 以降)
スマート・コンフィグレータ	e² studio プラグイン版 RL78 スマート・コンフィグレータ (e² studio に同梱)	V1.14.0 (V1.5.00 以降)
SIS (Software Integration System) モジュール	・Capacitive Sensing Unit driver. (r_ctsu) ・Touch middleware. (rm_touch)	V2.20 (V2.20 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)”に設定してビルドしてください。

図 3-2 にスマート・コンフィグレータの使用コンポーネントとバージョンの一覧を示します。

図 3-2 スマート・コンフィグレータの使用コンポーネント一覧

3.3　動作確認条件

表 3-4 に動作確認条件を示します。

表 3-4 動作確認条件

項目	内容
動作電圧	5.0 V (2.7 V ～ 5.5 V で動作可能) LVD0 検出電圧：リセット・モード 立ち上がり時 TYP. 2.67V (2.59 V～2.75 V) 立ち下がり時 TYP. 2.62V (2.54 V～2.70 V)
動作周波数	• メイン・システム・クロック 高速オンチップ・オシレータ・クロック (f_{IH}) : 32 MHz • CPU／周辺ハードウエア・クロック (f_{CLK}) : 32 MHz
	• サブシステム・クロック注 低速オンチップ・オシレータ・クロック (f_{IL}) : 32.768 kHz 低速周辺クロック周波数 (f_{SXP}) : 32.768 kHz

注　サブシステム・クロックは、ハードウェアタイマを用いたタッチ計測の場合に使用します(ソフトウェアタイマを用いたタッチ計測では使用しません)。

3.3.1　オプション・バイト設定

表 3-5 に本アプリケーション例でのオプション・バイトの設定を示します。

表 3-5 オプション・バイト設定 (RL78/G23)

アドレス	設定値	内容
000C0H / 040C0H	1110 1111B (0xEF)	ウォッチドッグ・タイマのカウンタの動作停止 (リセット解除後、カウント停止)
000C1H / 040C1H	1111 1100B (0xFC)	LVD0 検出電圧 ： リセット・モード 立ち上がり時 ：2.67V (TYP) (2.59 V～2.75 V) 立ち下がり時 ：2.62V (TYP) (2.54 V～2.70 V)
000C2H / 040C2H	1110 1000B (0xE8)	HS (高速メイン)モード、 高速オンチップ・オシレータの周波数 ：32MHz
000C3H / 040C3H	1000 0100B (0x84)	オンチップ・デバッグ動作許可

オプション・バイトの設定はコード生成後に、プロジェクトのプロパティから、[**C/C++ビルド**] - [**設定**] - [**ツール設定**] - [**Linker**] - [**デバイス**]を開き、「ユーザ・オプション・バイト値」および「オンチップ・デバッグ制御値」にて確認することができます。

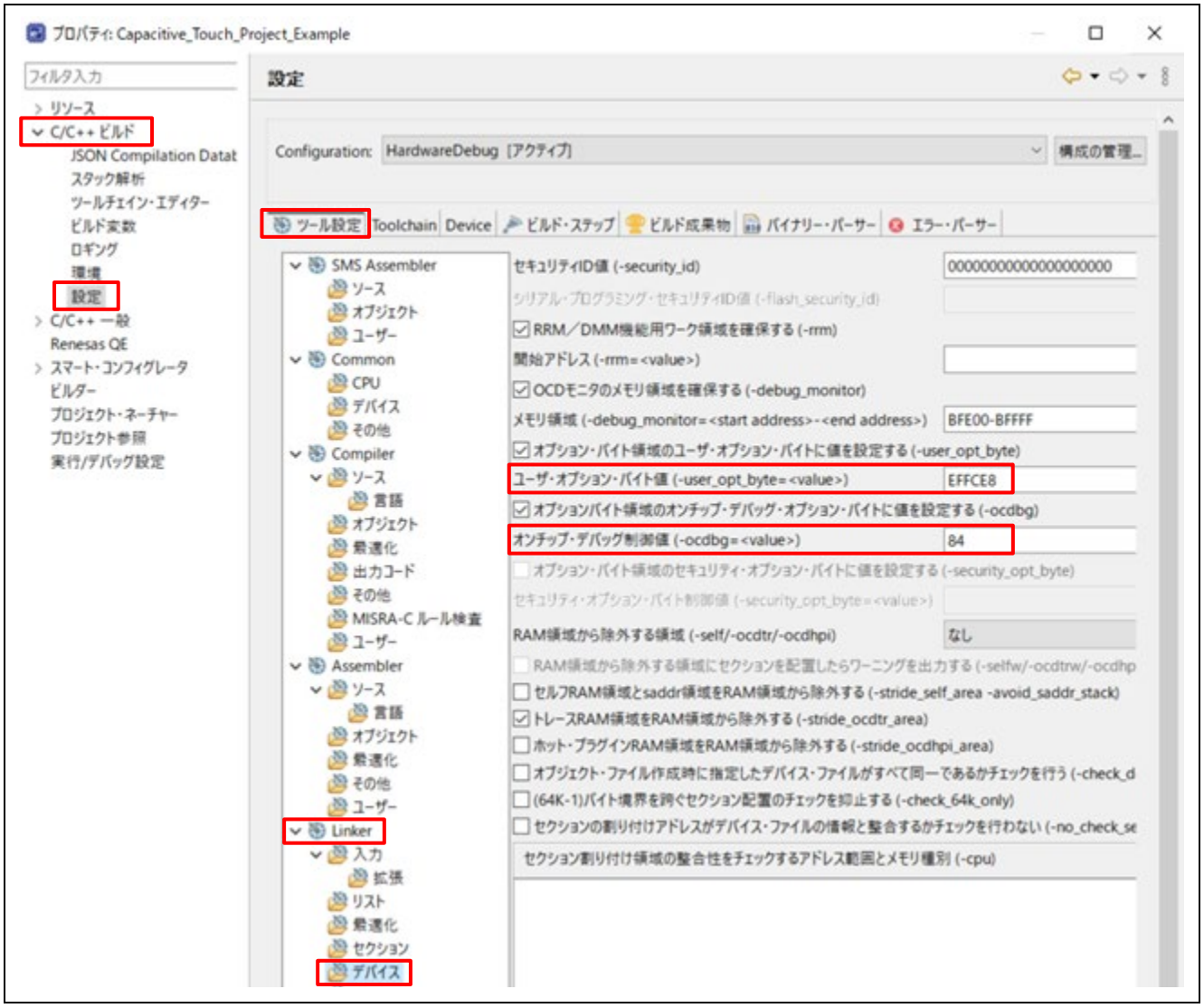

図 3-3 オプション・バイト設定

4.　静電容量タッチアプリケーション開発手順の概要

プロジェクトにタッチセンサ検出を統合するために必要な手順の概要を、以下に示します。これらの手順は一般的なユーザアプリケーション開発に適用可能です。

- e^2 studio のプロジェクト作成ウィザードを使用して新規プロジェクトを作成します。
- スマート・コンフィグレータを使用して、必要なモジュールをプロジェクトに追加します。
- QE for Capacitive Touch を使用して、静電容量タッチインタフェースを作成します。
- QE for Capacitive Touch を使用して、プロジェクトをチューニングします。
- 必要な SIS モジュールの API コールをプロジェクトに追加し、静電容量タッチ制御を有効にします。
- QE for Capacitive Touch を使用してプロジェクトをモニタし、静電容量タッチ検出を確認します。

5.　アプリケーション例の概要

アプリケーション例のメインループの実装は以下のとおりです。

- rm_touch ミドルウェアの処理を実行すべきか判断するグローバルフラグをチェックします。
  - グローバルフラグが TRUE の場合(準備完了)
    - グローバルフラグを FALSE にリセットします。
    - rm_touch ミドルウェアの API をコールし、前回計測時のデータの処理と必要なデータを更新し、次のスキャンを開始します。
    - rm_touch ミドルウェアの API をコールし、ユーザが作成したグローバル変数に、ターゲットボード上のセンサのタッチ有無をバイナリ形式で格納します。

完成したアプリケーション例のコードリストに関しては「**15. qe_touch_sample.c** 」章を参照してください。

6. 静電容量タッチアプリケーション開発の開始～モジュール追加

6.1 新規プロジェクトの作成

e² studio を使用して新規プロジェクトを作成します。

1. Windows のスタートメニューまたはデスクトップのショートカットから e² studio を起動します。
   ダイアログが表示されたら、任意の場所にワークスペースを作成します。

2. e² studio のメニュー[**ファイル**] - [**新規**] - [**C/C++ Project**]を選択し、新規プロジェクトの作成を開始します。

3. ダイアログが表示されたら“**Renesas RL78**” - “**Renesas CC-RL C/C++ Executable Project**”を選択し、[**次へ**]をクリックします。

4. 次のダイアログで“**プロジェクト名(P):**”に任意のプロジェクト名を入力します。
   このアプリケーション例では“**Capacitive_Touch_Project_Example**”を入力します。
   入力完了後、[**次へ**]をクリックします。

5. 次のダイアログでは、以下を選択します。

- 言語: C
- ツールチェーン: Renesas CC-RL
- ツールチェーン・バージョン: v1.15.00
- Target Board: Custom
- ターゲット・デバイス: R7F100GSNxFB (選択方法は、図 6-2 を参照。)
- “**Hardware Debug 構成を生成**”をチェック。E2 Lite (RL78)をプルダウンメニューから選択。

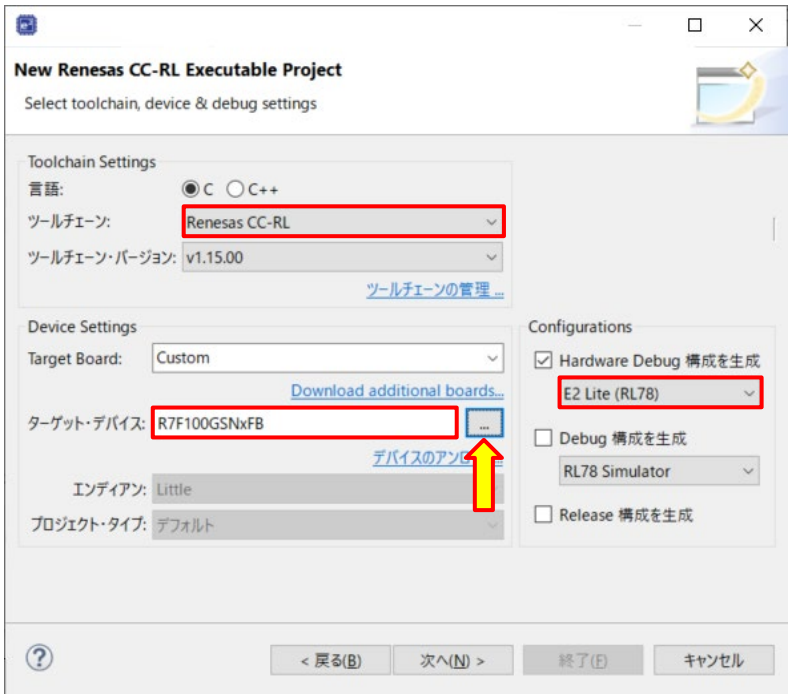

図 6-1 ツールチェーン，デバイス，デバッグ設定の選択

“ターゲット・デバイス”は、[…]ボタンを押下して"Device Selection"ウィンドウに表示されるデバイスから選択します。

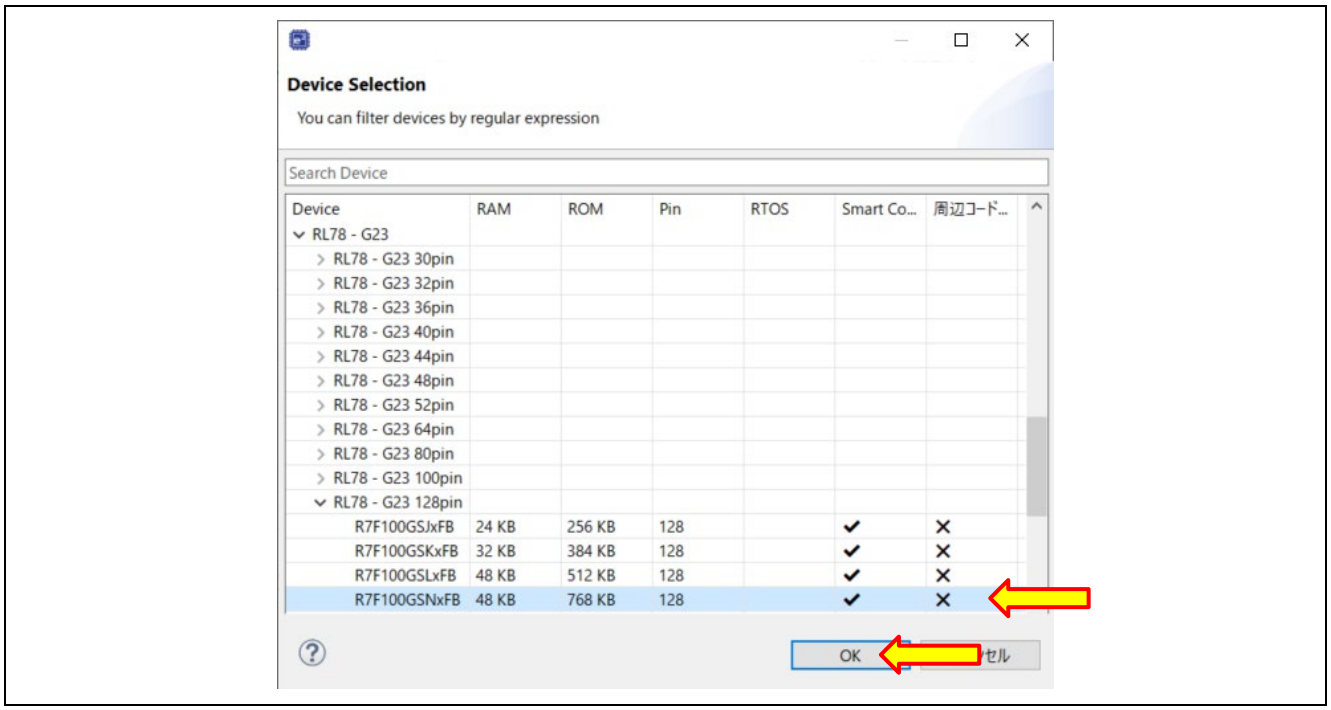

図 6-2 ターゲット・デバイスの選択

他の RL78 製品の RSSK で動作確認を行う場合は、以下のとおりにターゲット・デバイスを選択してください。

- RL78/G22 (型番名: RTK0EG0042S01001BJ): R7F102GGExFB
- RL78/L23 (型番名: RTK0EG0063S01001BJ): R7F100LPLxFB
- RL78/G16 (型番名: RTK0EG0047S01001BJ): R5F121BCxFP

6. 完了したら、[**次へ**]をクリックします。

7. 次のダイアログが表示されたら、“**Use Smart Configurator**”をチェックし、[**終了**]をクリックしてください。

8. 完了すると e^2 studio のデフォルトウィンドウにスマート・コンフィグレータのパースペクティブが表示され、プロジェクトの設定が可能な状態になります。これで新規プロジェクトの作成は完了です。

6.2　スマート・コンフィグレータによるモジュール追加

スマート・コンフィグレータを使用して、必要なモジュールのソースファイルをプロジェクトに追加します。

1. e² studio の中央下部のタブから[**クロック**]タブを選択し、RL78 MCU のクロック設定を表示します。

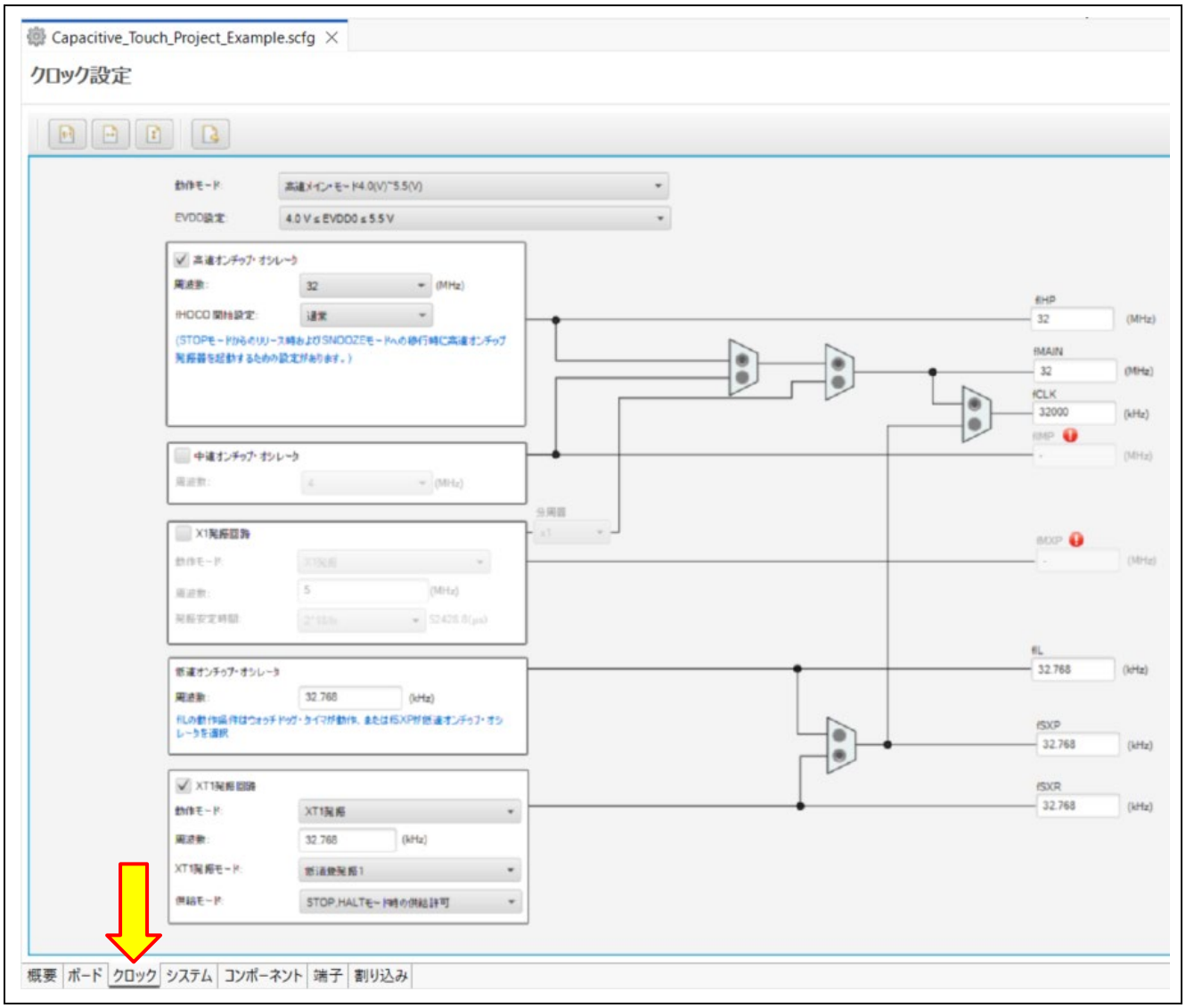

図 6-3 [クロック]タブの選択

2. 本アプリケーション例では、MCU を高速メイン・モード (HS モード) かつ動作電圧範囲 2.7V～5.5V で使用するため、下図のように"動作モード"を選択します。

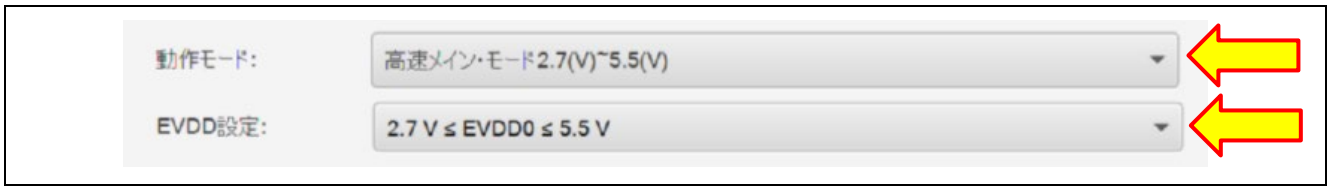

図 6-4 動作モードおよび EVDD 設定

【参考情報】
他の MCU で動作確認する場合について
EVDD の設定項目が存在するかどうかは、使用する MCU により異なります。

3. 本アプリケーション例で使用するクロック設定(デフォルト設定)を以下に示します。

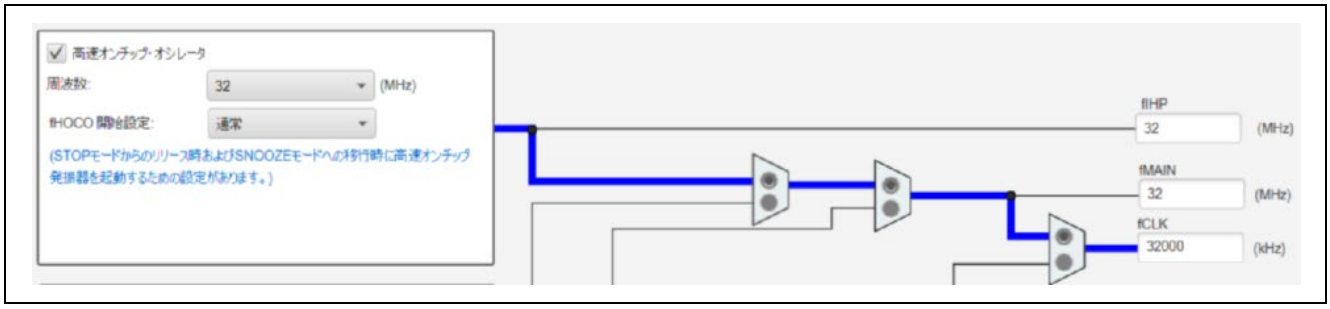

図 6-5 クロック設定 (1)

4. 下図のように、XT1 発振回路のチェックを外します。

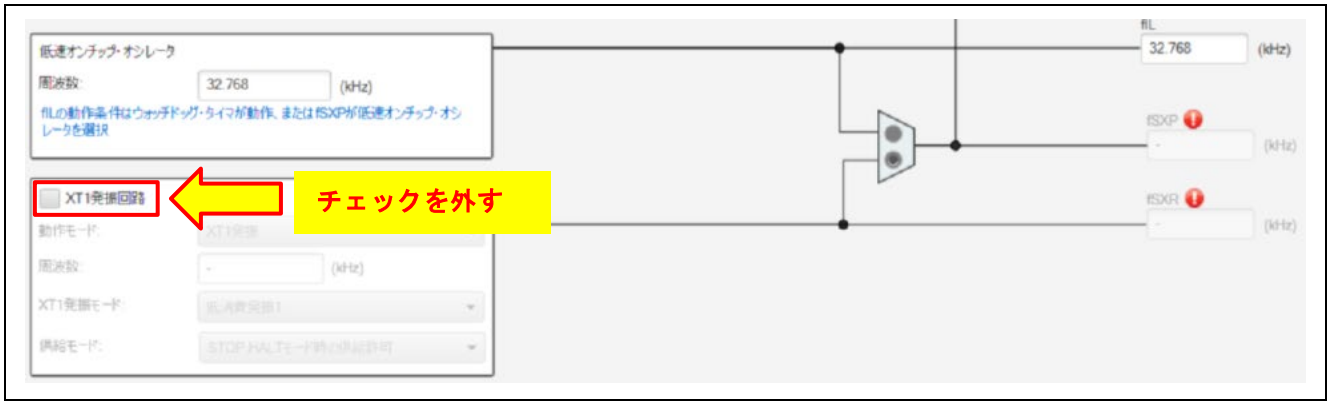

図 6-6 クロック設定 (2)

5. [**システム**]タブに移動します。

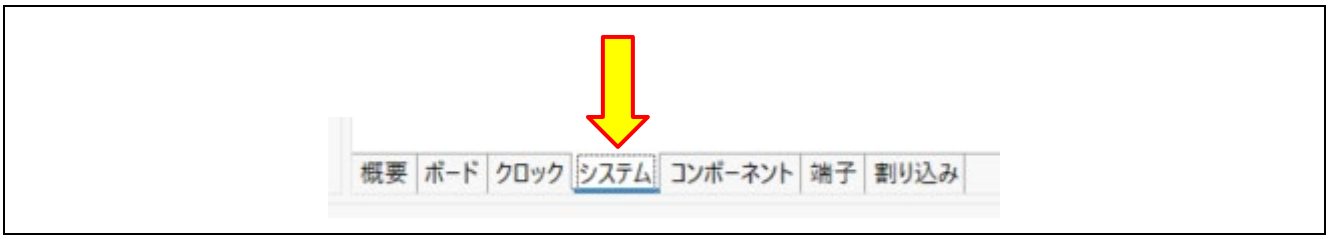

図 6-7 [システム]タブの選択

6. 以下のように“**エミュレータを使う**”および“**E2 Lite**”を選択し、“**セキュリティ ID を設定する**”のチェックを外します。

図 6-8 システム設定

7. [**コンポーネント**]タブに移動します。

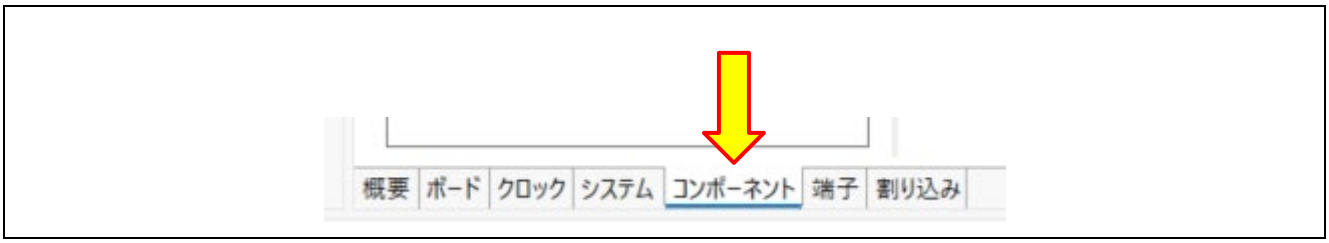

図 6-9 [コンポーネント]タブの選択

8. プロジェクトに SIS モジュールと各種コンポーネントを追加します。

   コンポーネントの追加ボタン (図 6-10 の赤枠の箇所) をクリックし、一覧から追加するコンポーネントを選択します。

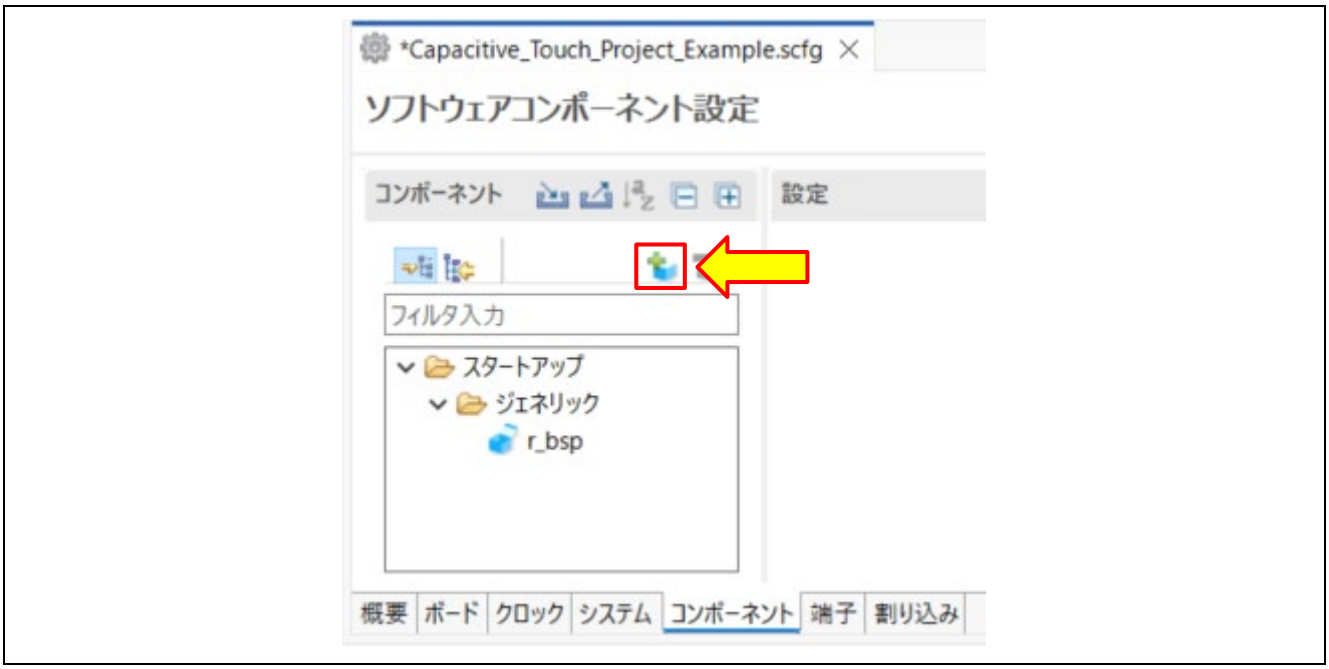

図 6-10 コンポーネントの追加 (1)

9. “コンポーネントの追加”ダイアログが表示され、プロジェクトに追加可能なモジュールが表示されます。

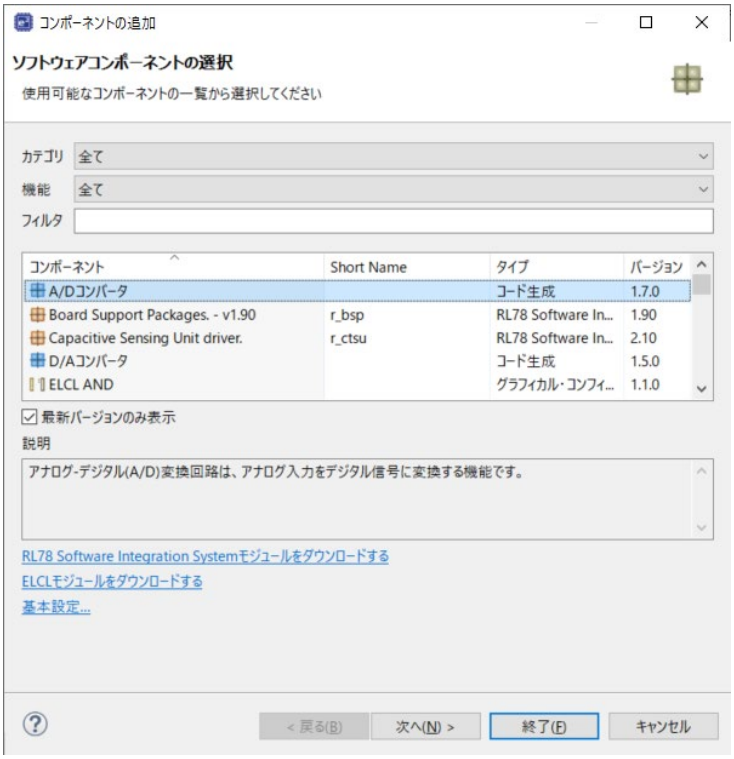

図 6-11 “コンポーネントの追加”ダイアログ

10. タッチ機能の SIS (Software Integration System) モジュールを初めて使用する場合は、図 6-12～図 6-14 の①～④の手順でダウンロードしてください。すでにダウンロード済みの場合、この操作は不要です。

図 6-12 SIS モジュールのダウンロードリンク

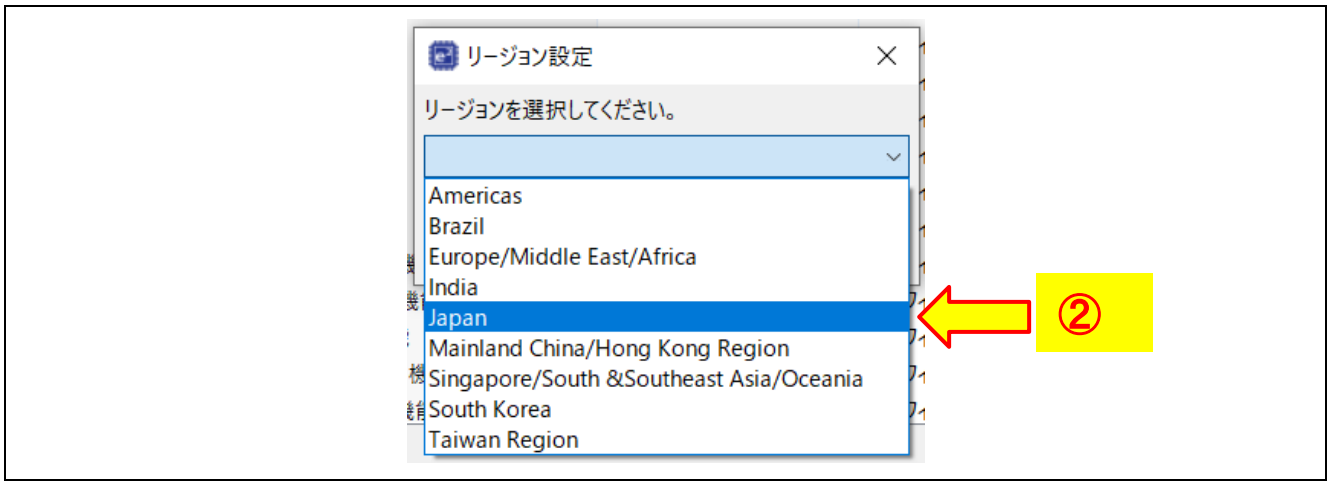

図 6-13　リージョン選択

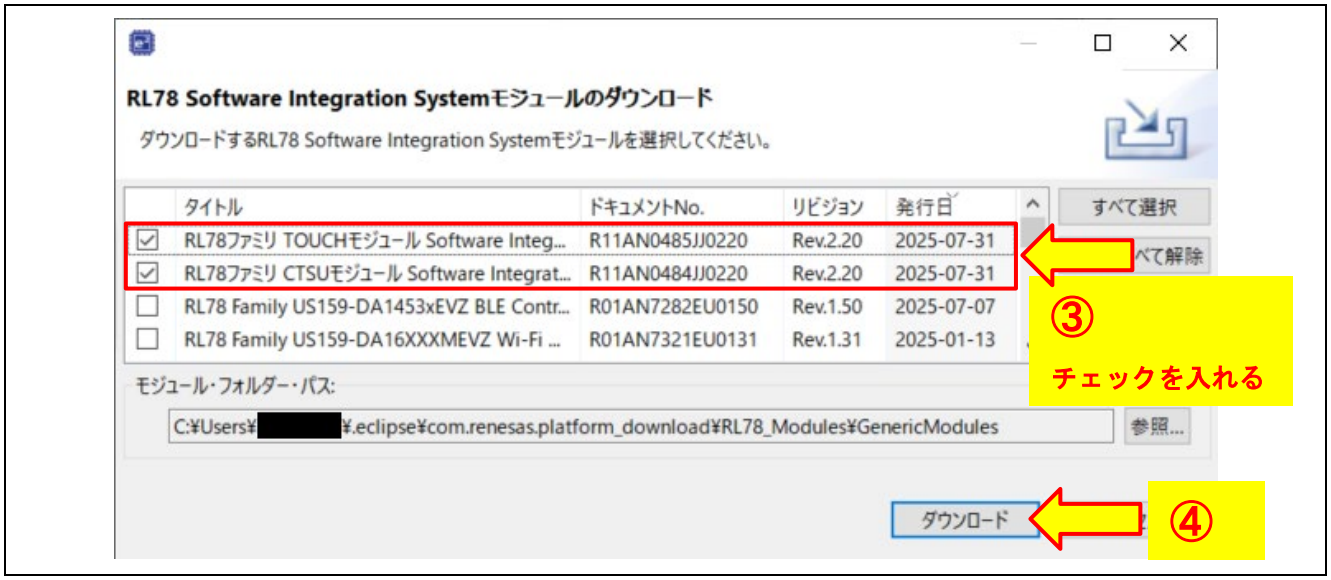

図 6-14 SIS モジュールのダウンロード

【注意】
上記ダウンロード画面に、TOUCH モジュールおよび CTSU モジュールが表示されない場合は、別の手順でダウンロードする必要があります。
Renesas WEB から各モジュールをダウンロードし、手動で SIS モジュールのダウンロードフォルダにファイルを追加します。
各モジュールの WEB ページおよび適用方法については以下を参照してください。

- CTSU モジュールおよび TOUCH モジュールの WEB ページ

RL78 ファミリ　CTSU モジュール　Software Integration System
RL78 Family CTSU Module Software Integration System Rev.x.xx - Sample Code

RL78　ファミリ TOUCH モジュール　Software Integration System
RL78 Family TOUCH Module Software Integration System Rev.x.xx - Sample Code

- SIS モジュールの適用方法

Knowledge Base: Renesas Web 等からダウンロードした Software Integration System モジュールの適用方法

11. スマート・コンフィグレータで以下のコンポーネントを選択してください。

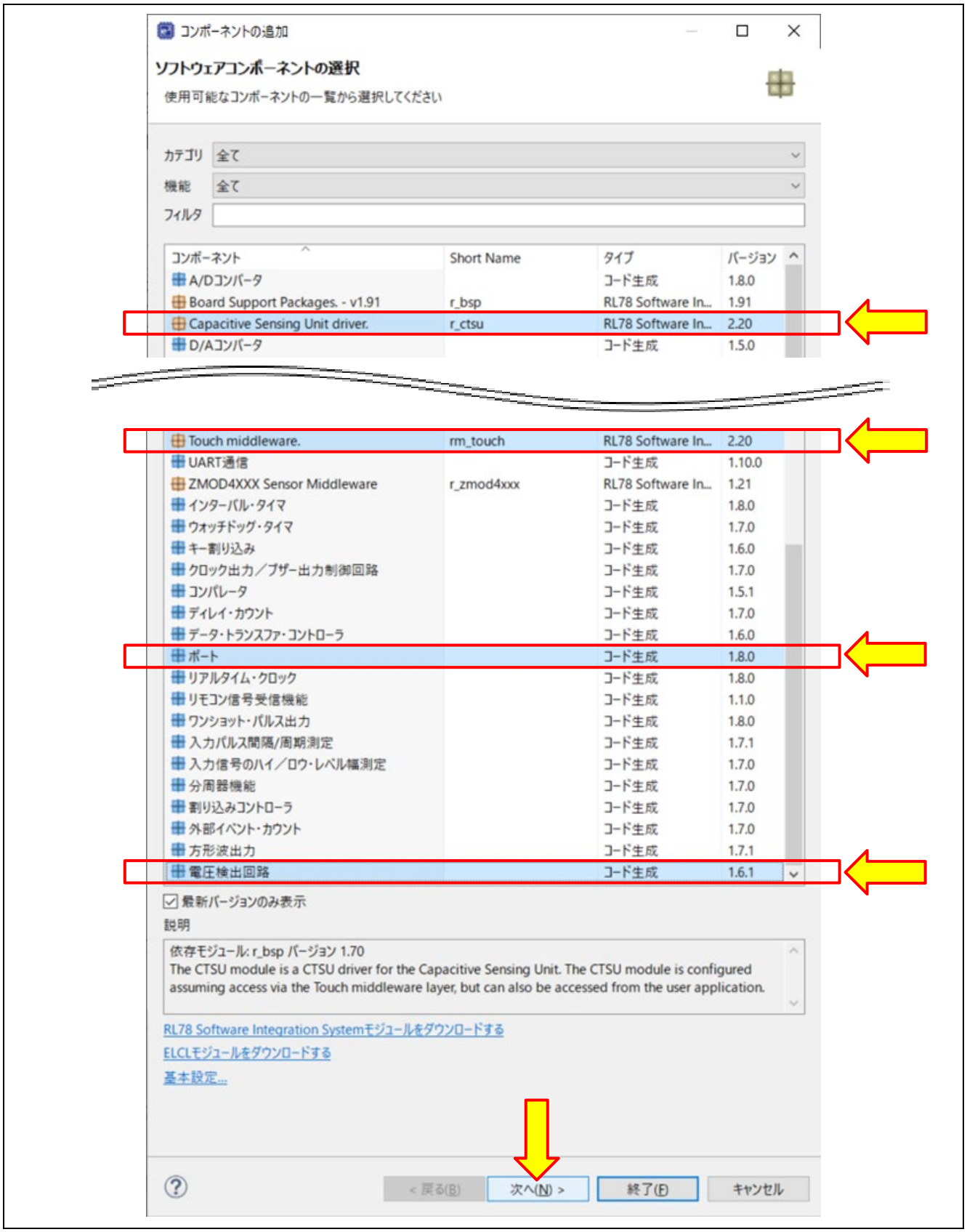

図 6-15 ソフトウェアコンポーネントの選択 (1)

12. 選択したコンポーネントに対してリソースを設定します。本アプリケーション例では以下の設定で使用します。

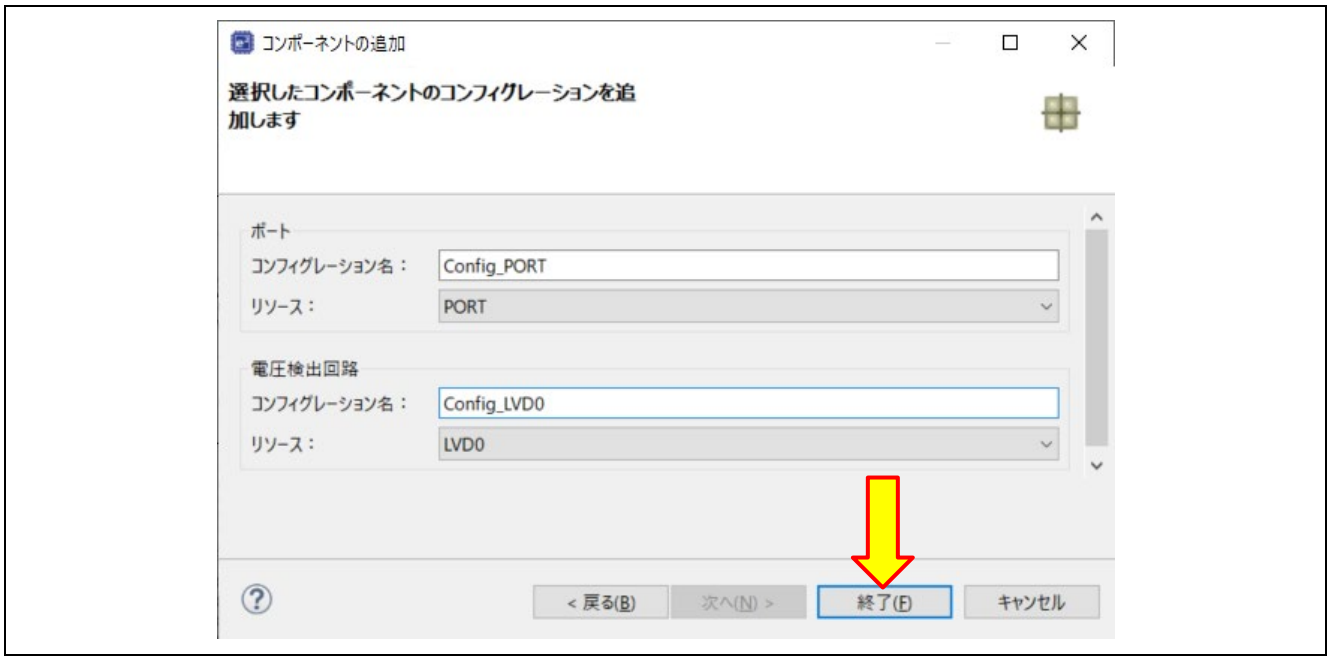

図 6-16 コンポーネントのリソース設定 (1)

13. 以下に示すようにコンポーネントが追加されます。

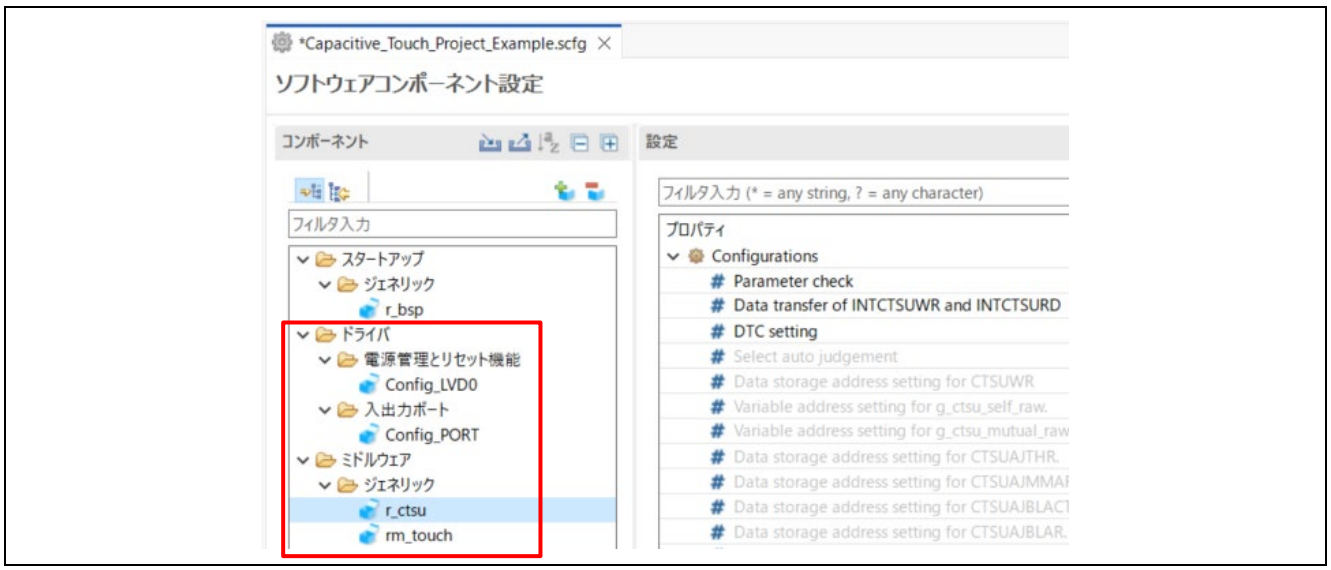

図 6-17 ソフトウェアコンポーネント設定 (コンポーネント追加後)

6.2.1　CTSU コンポーネント設定

“**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 TSxx 端子を有効にします。本アプリケーション例では、2 つのタッチボタン(**TS05** と **TS06**)を使用します。

以下のセンサポートを設定パネルでクリックし、プロジェクトで使用できるようにしてください。

- TSCAP 端子
- TS05 端子
- TS06 端子

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。

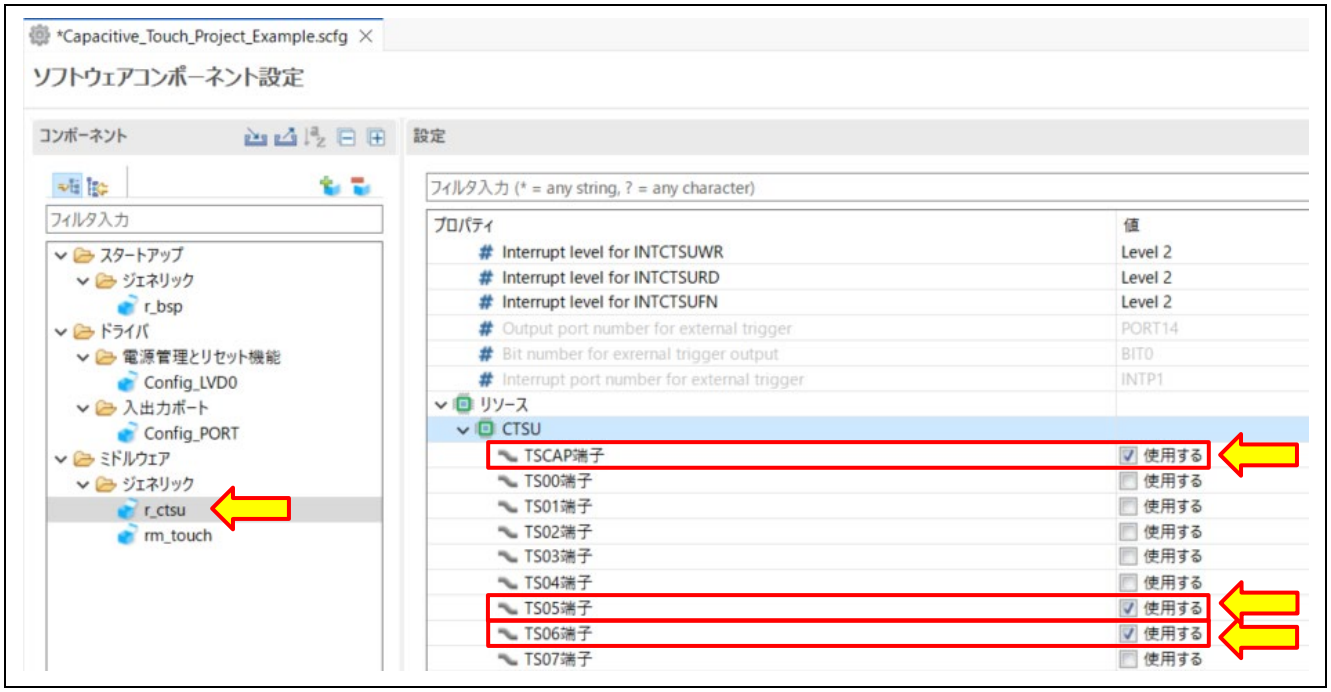

図 6-18 アプリケーションで使用する TS 端子の有効化

注.　タッチアプリケーションで使用しない TSxx 端子は、ロウ・レベル出力駆動に設定することを推奨します。CTSU2 ではタッチアプリケーションで使用しない TSxx 端子を“☑ 使用する”で有効にした場合、非計測端子としてロウ・レベル出力の設定になります。実際に回路を作成される場合は、端子処理などを適切に行い、電気的特性を満たすように設計してください。

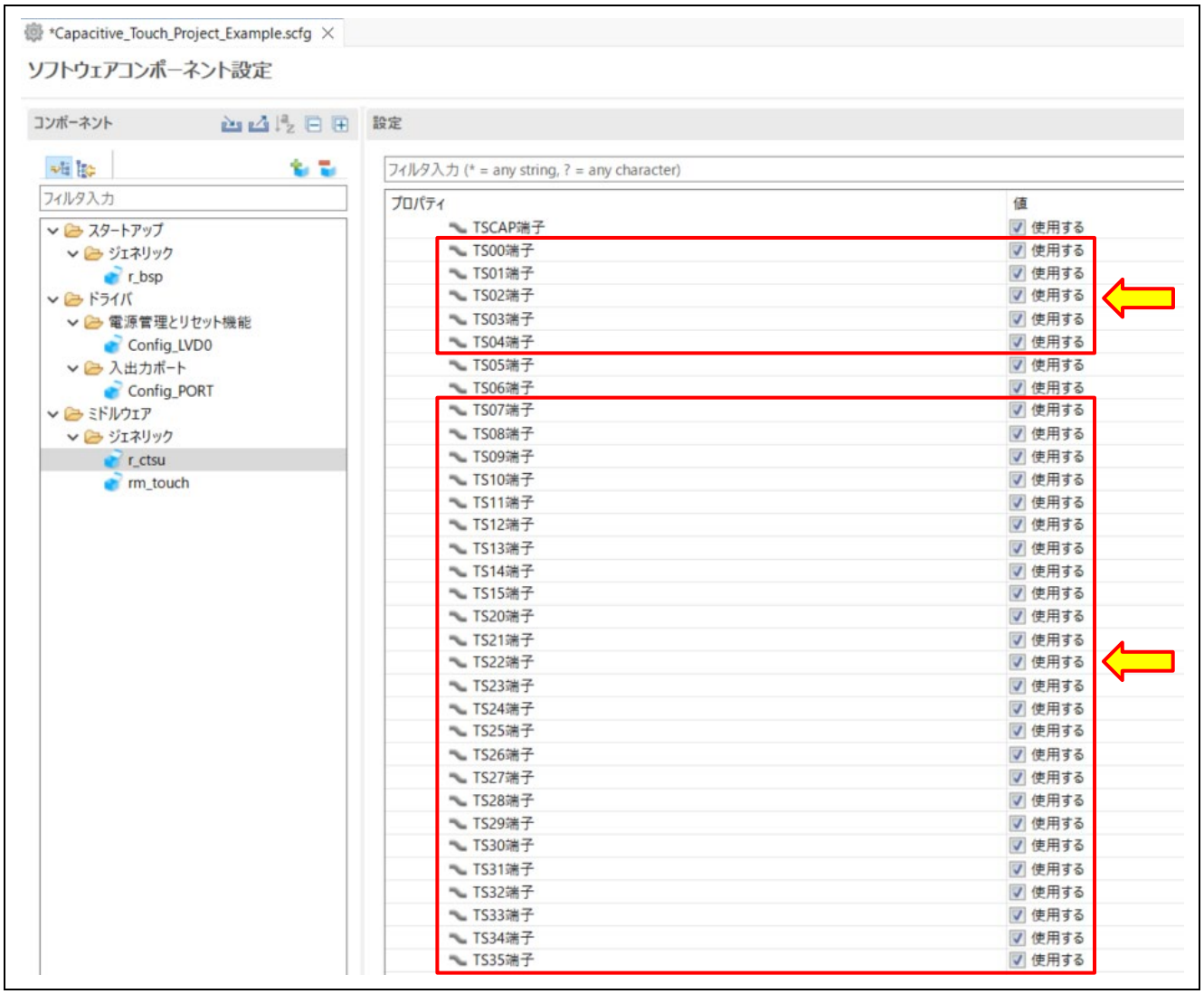

図 6-19 アプリケーションで使用しない TS 端子の有効化

6.2.2　Touch コンポーネント設定

デフォルト設定のまま使用します。

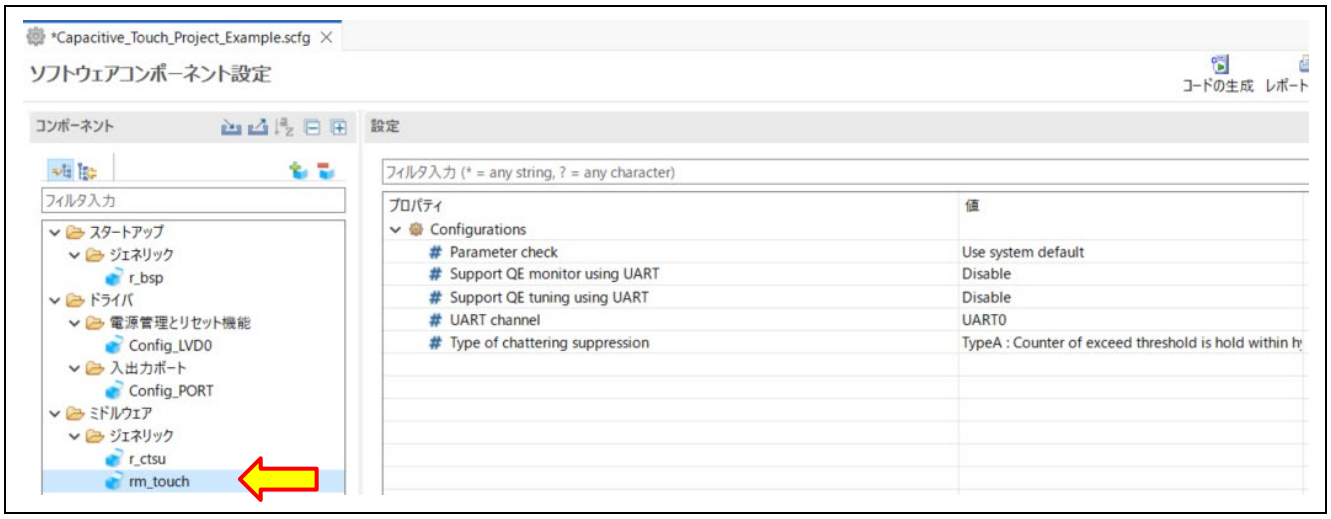

図 6-20 Touch コンポーネント設定

6.2.3　PORT コンポーネント設定

ポートの設定を行います。ここでは、未使用端子の設定を行います。

1. タッチアプリケーションで使用しないポートをスタートアップでロウ・レベル出力駆動に設定することを推奨します。“**Config_PORT**”モジュールをクリックし、**PORT0** のチェックボックスをクリックします。

注.　ここでは使用例として一部のポートを設定しています。実際に回路を作成される場合は、端子処理などを適切に行い、電気的特性を満たすように設計してください。

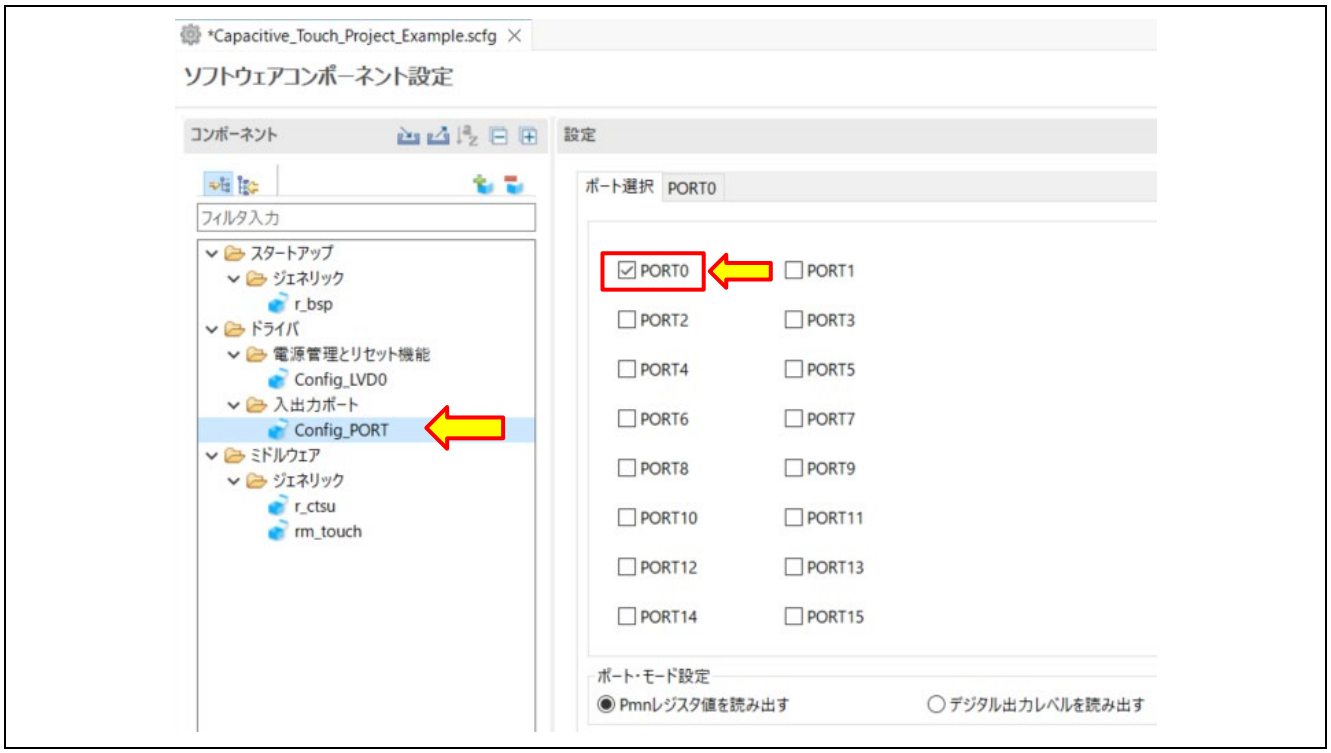

図 6-21 PORT コンポーネント設定 (ポート選択タブ)

2. [**PORT0**]タブを選択し、P07 を“**出力**”に設定します。設定パネルは以下の図のようになります。

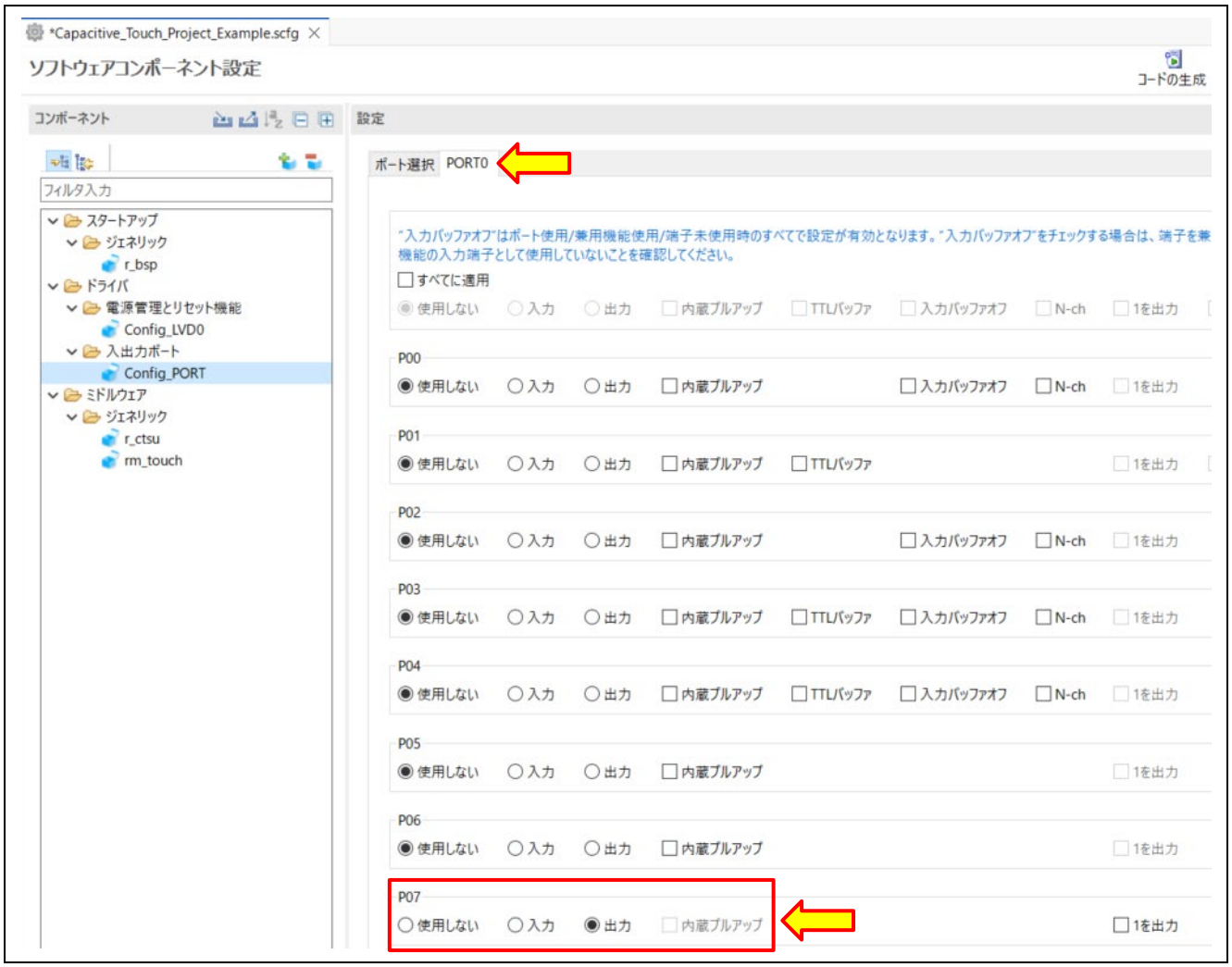

図 6-22 PORT コンポーネント設定 (PORT0 タブ)

6.2.4　LVD コンポーネント設定

ユーザ・オプション・バイト の電圧検出回路 0（LVD0）を設定します。

“**Config_LVD0**”モジュールをクリックし、動作モードと検出電圧の設定を行います。

リセット発生電圧 (VLVD0) を 2.62 (V)に設定します。

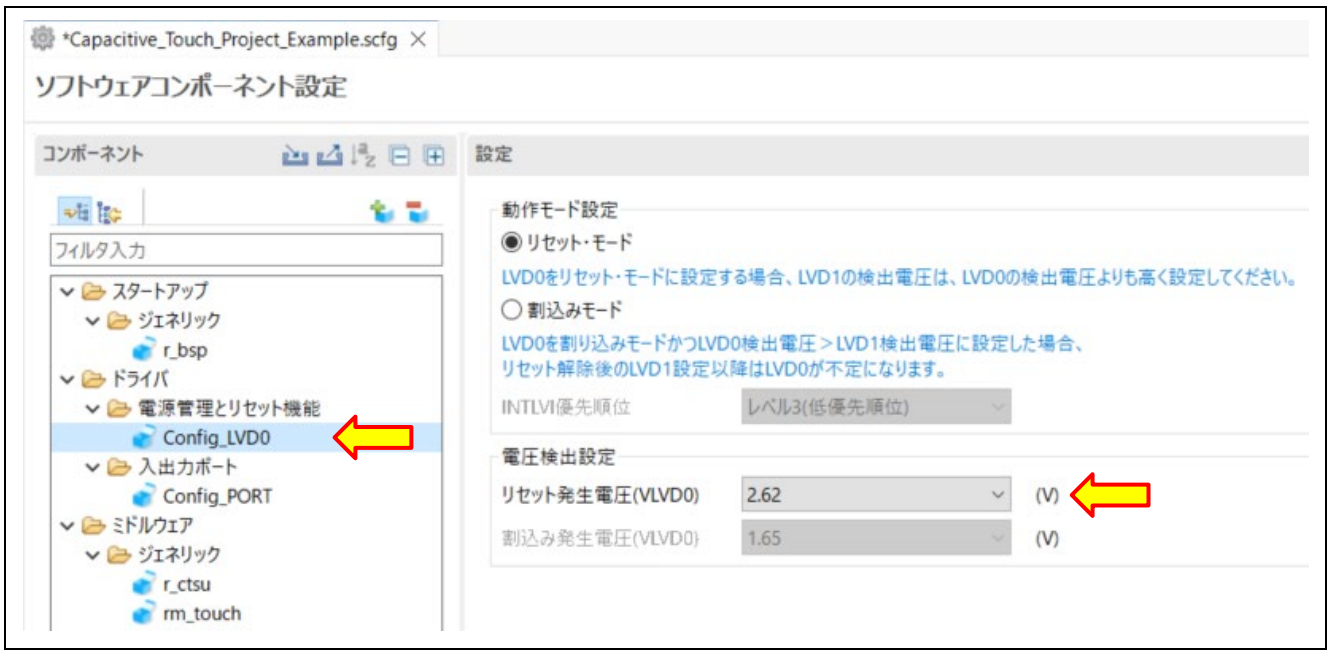

図 6-23 LVD コンポーネント設定

6.2.5　BSP コンポーネント設定

“r_bsp”モジュールをクリックし、“**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ます。

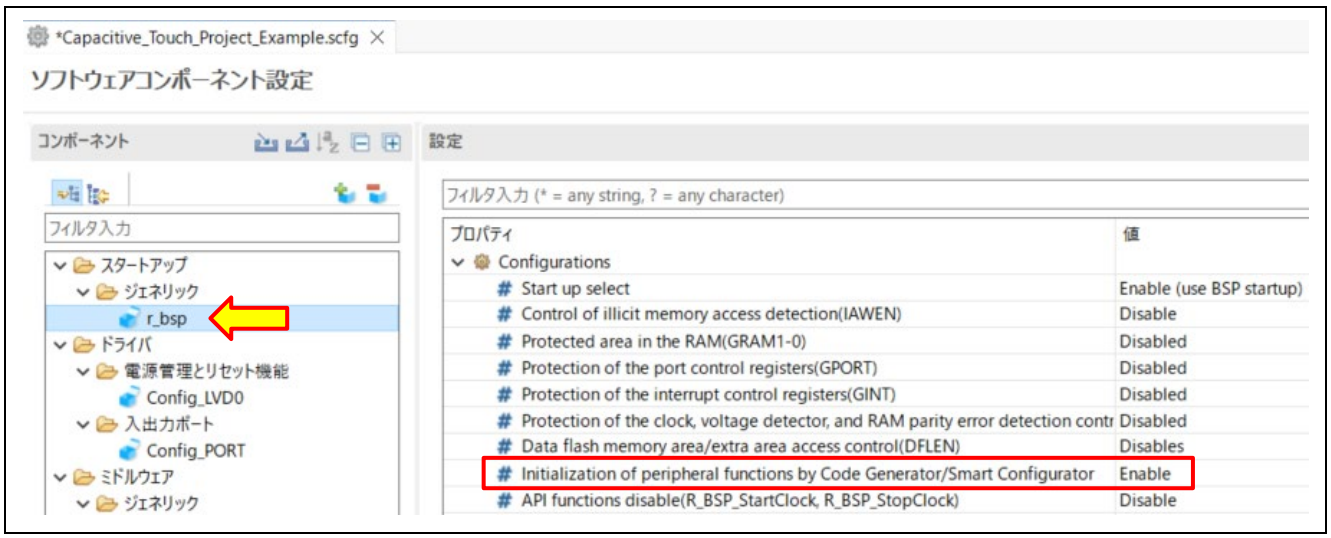

図 6-24 BSP コンポーネント設定

6.3　コード生成

スマート・コンフィグレータの右上の“コードの生成”アイコン をクリックして、プロジェクトに必要なコンポーネントのコードを追加します。以上でコンポーネントの追加は完了です。

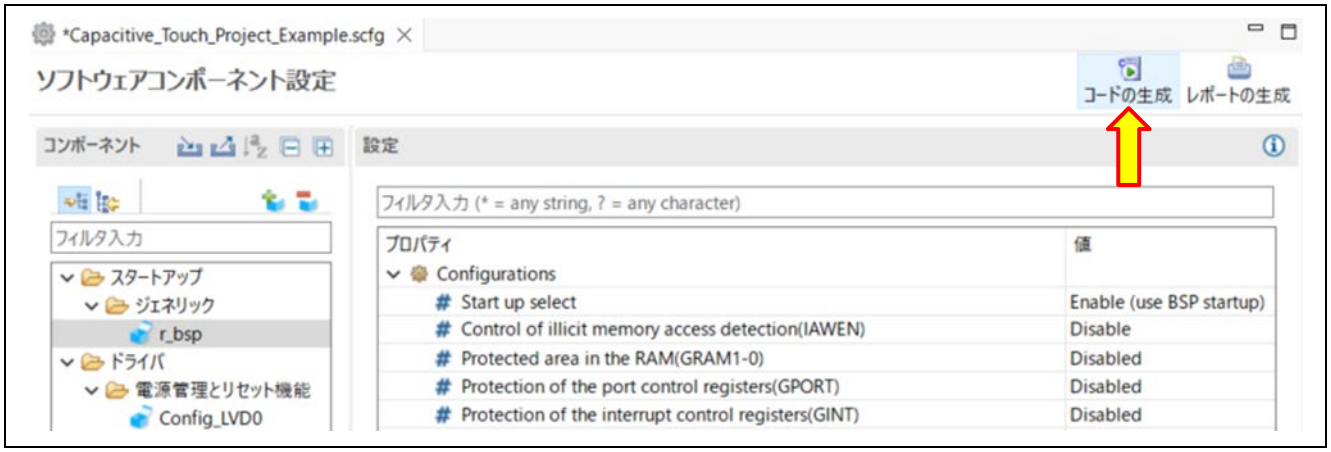

図 6-25　“コードの生成”アイコンの選択 (1)

注.　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ます。 変更内容を確認した後に、[OK]をクリックします。

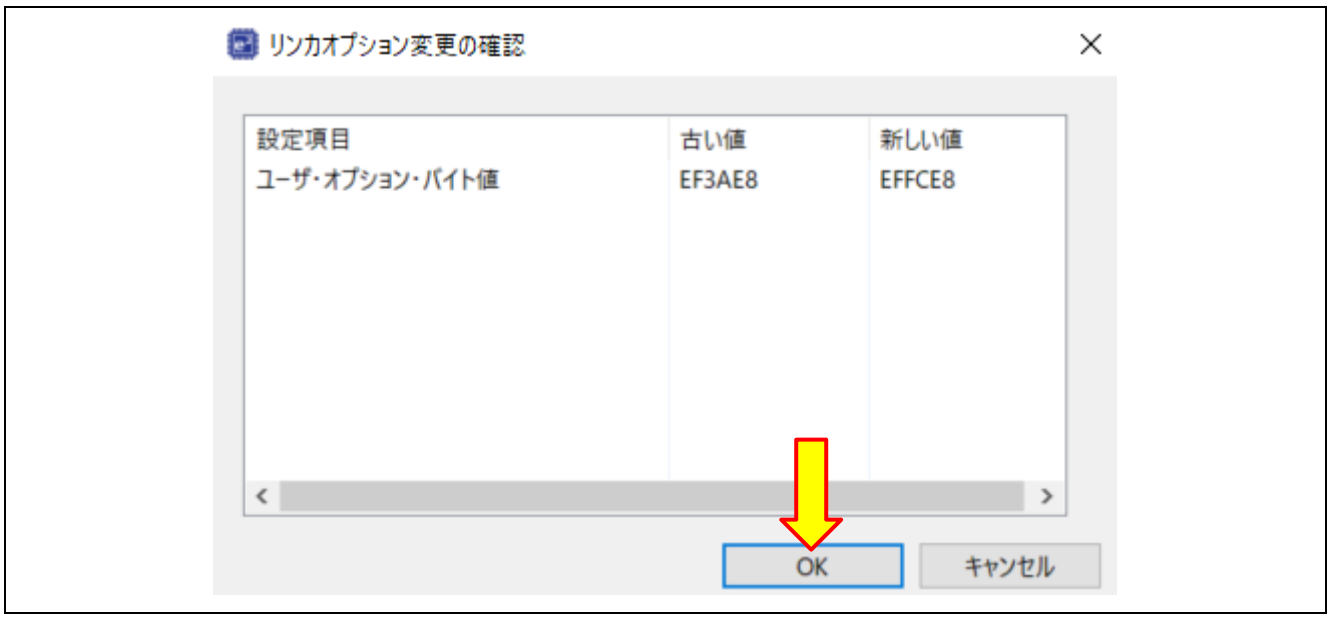

図 6-26 リンカオプション変更の確認

7.　[追加機能] UART を使用したシリアル通信モニタの設定 (1/3)

注.　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。
また一方で、タッチ性能のモニタリングは、シリアル通信を介して行うこともできます。 したがって、スムーズにモニタリングを行いたい場合は、シリアル通信を介したモニタリング機能を追加してください(推奨設定)。

以下に示す 7 章、12 章および 14 章(本章を含む)では、UART を使用したシリアル通信モニタの設定について説明します。

- 7. [追加機能] UART を使用したシリアル通信モニタの設定 (1/3)
- 12. [追加機能] UART を使用したシリアル通信モニタの設定 (2/3)
- 14. [追加機能] UART を使用したシリアル通信モニタの設定 (3/3)

7.1　Touch コンポーネント設定(シリアル通信モニタ用)

以下のように、[コンポーネント]タブで、“**rm_touch**”モジュールを選択し、“Support QE monitor using UART”を“**Enable**”に、“UART channel”を“**UARTA1**”に、設定します。

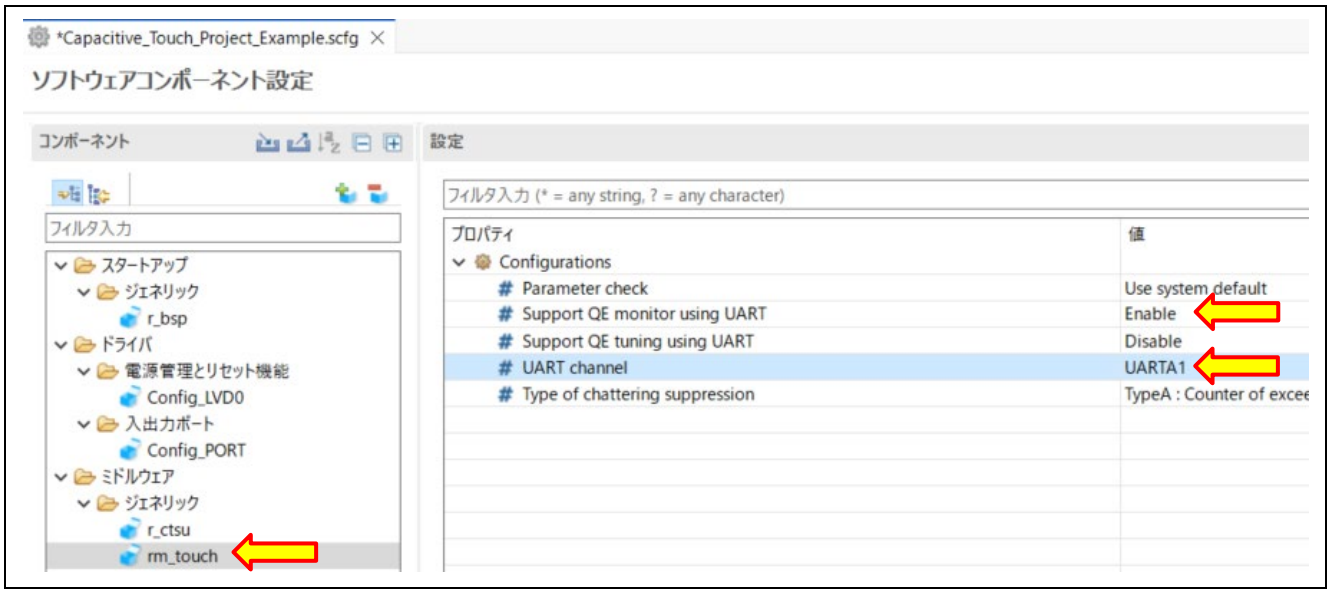

図 7-1 Touch コンポーネント設定(シリアル通信モニタ用)

注.　ツールで設定する 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。

7.2　UART 通信コンポーネント設定

1.　コンポーネントの追加ボタン (図 7-2 の赤枠の箇所) をクリックします。

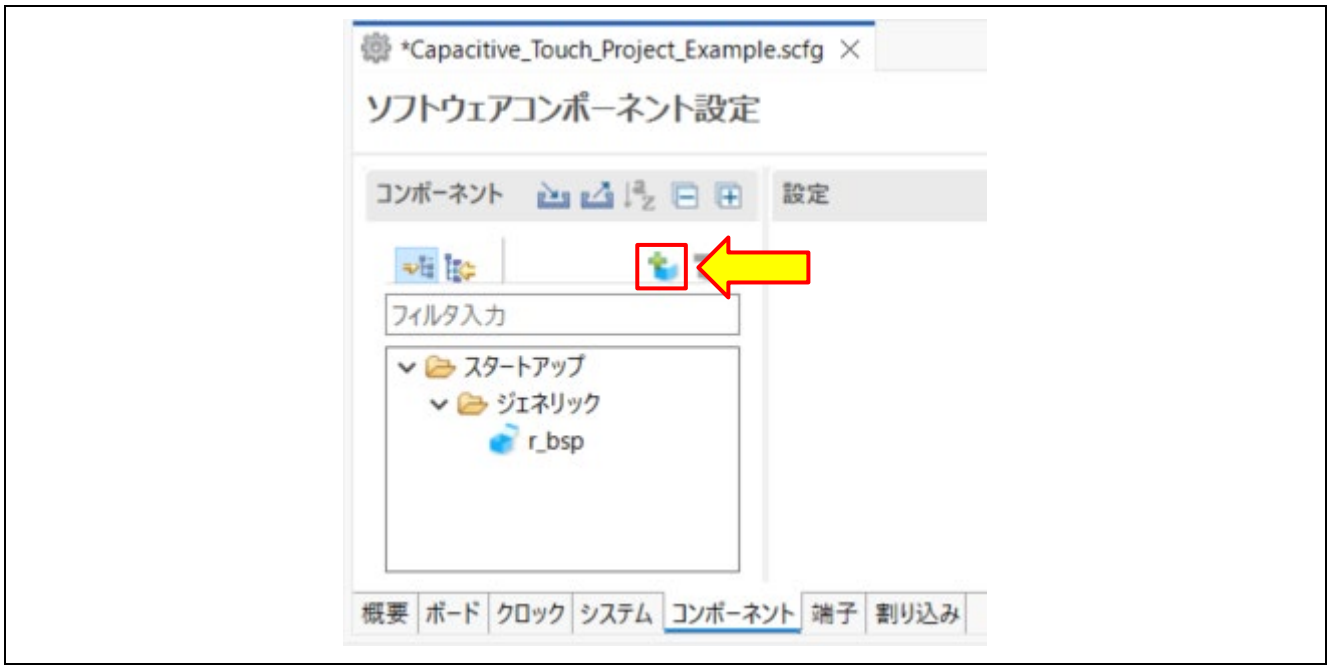

図 7-2 コンポーネントの追加 (2)

2.　“**UART 通信**”モジュールを選択し、ダイアログ下部の[**次へ**]をクリックします。

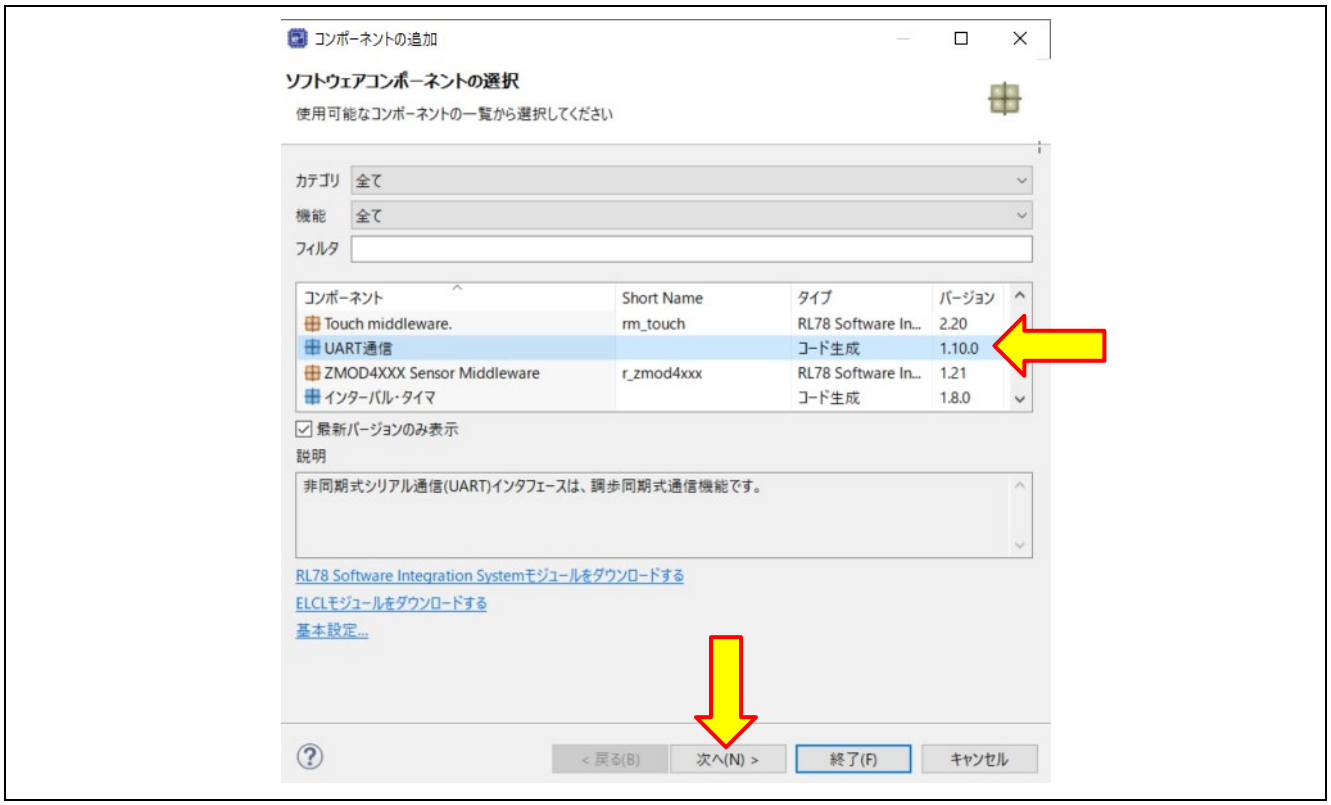

図 7-3 ソフトウェアコンポーネントの選択 (2)

3. “動作モード”を“**送信/受信**”に、“リソース”を“**UARTA1**”に設定し、ダイアログ下部の[**終了**]をクリックします。

図 7-4 コンポーネントのリソース設定 (2)

注.　ツールで設定する 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。

4. [**コンポーネント**]タブで、“**Config_UARTA1**”モジュールを選択します。

   “動作クロック”を“**fSEL/8**”と“**fSEL クロック選択 fIHP**”に、“**割り込みによる連続送信**”を選択し、“転送レート設定”を“**153600**”(bps)に設定します。

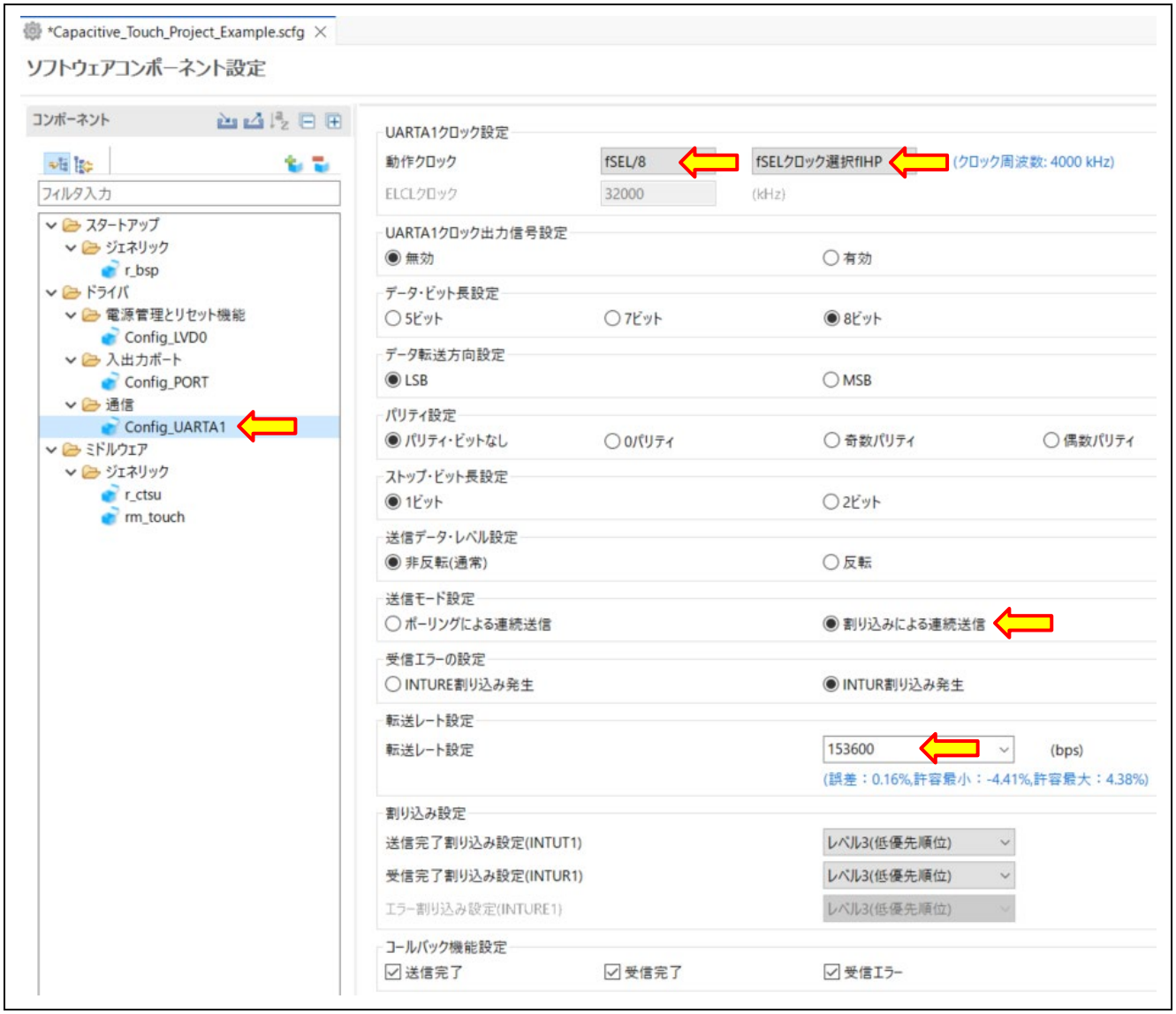

図 7-5 UART 通信コンポーネントの設定 (UARTA1)

注. ツールで設定する 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。UARTAx ではなく UARTx を使用する場合は、次のページ以降で説明している注 2 を参照してください。

注 1.　スマート・コンフィグレータのバージョンによっては、“転送レート設定”でエラーが発生することがあります。このエラーが発生した場合は、コード生成後に下記赤枠部分のプログラムを変更して転送レートを設定してください。

Config_UARTA1.c

```
H3 void R_Config_UARTA1_Create(void)
59 {
60     UARTAEN1 = 0U;
61     UTMK1 = 1U;     /* disable INTUT1 interrupt */
62     UTIF1 = 0U;     /* clear INTUT1 interrupt flag */
63     URMK1 = 1U;     /* disable INTUR1 interrupt */
64     URIF1 = 0U;     /* clear INTUR1 interrupt flag */
65     UREMK1 = 1U;    /* disable INTURE1 interrupt */
66     UREIF1 = 0U;    /* clear INTURE1 interrupt flag */
67     /* Set INTUT1 low priority */
68     UTPR11 = 1U;
69     UTPR01 = 1U;
70     /* Set INTUR1 low priority */
71     URPR11 = 1U;
72     URPR01 = 1U;
73     /* Set INTURE1 low priority */
74     UREPR11 = 1U;
75     UREPR01 = 1U;
76     BRGCA1 = _0D_UARTA_OUTPUT_BAUDRATE;
77     ASIMA11 = _00_UARTA_PARITY_NONE | _18_UARTA_TRANSFER_LENGTH_8 | _0
78               _00_UARTA_DATA_NORMAL;
79     ASIMA10 = _02_UARTA_BUFFER_EMPTY | _01_UARTA_INTUR_OCCUR;
80     UTA0CK |= _20_UARTA_FSEL_SELECT_FIHP;
81     UTA1CK = _00_UARTA_CLKA1_OUTPUT_DISABLE | _03_UARTA1_SELECT_FSEL8;
```

図 7-6 Config_UARTA1.c

注 2.　ツールで設定する 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。例えば、UART1 を使用する場合、下記画像のように設定が異なります。

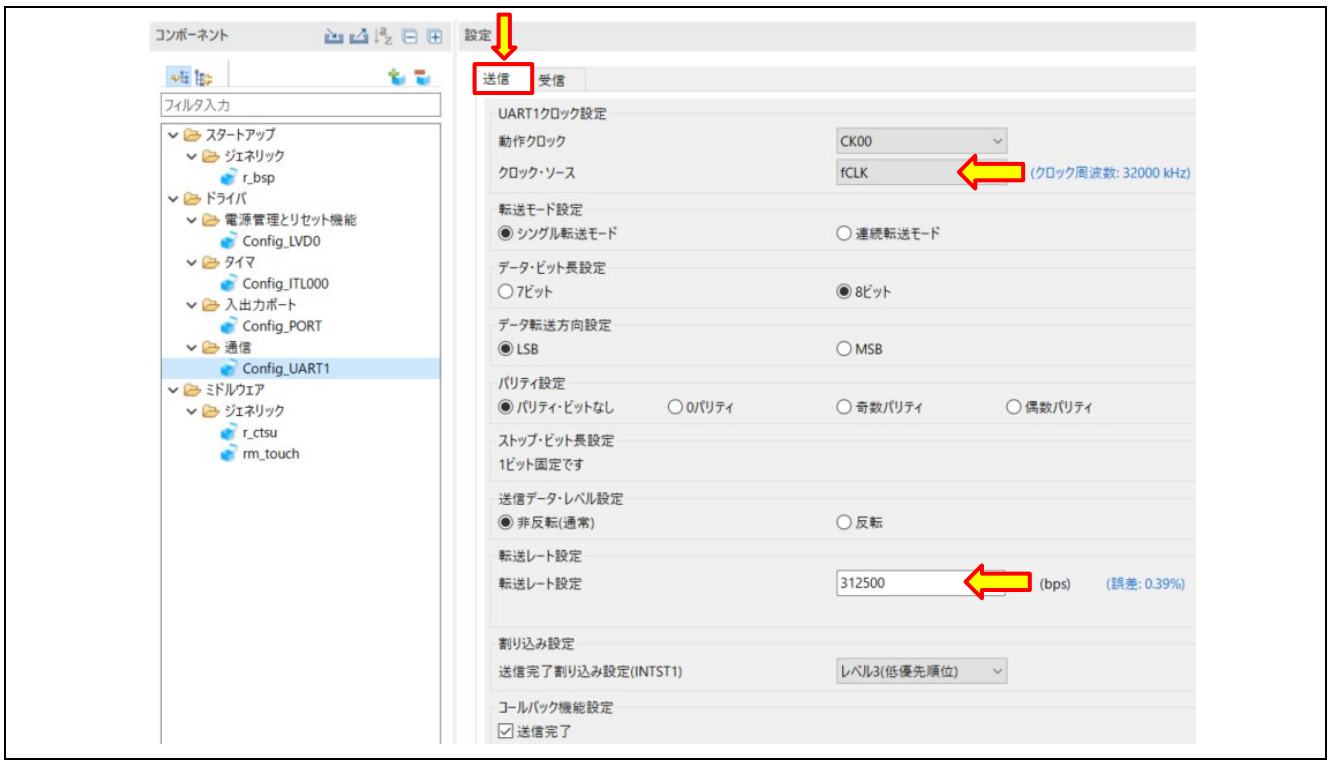

図 7-7 UART 通信コンポーネントの設定 (UART1 送信)

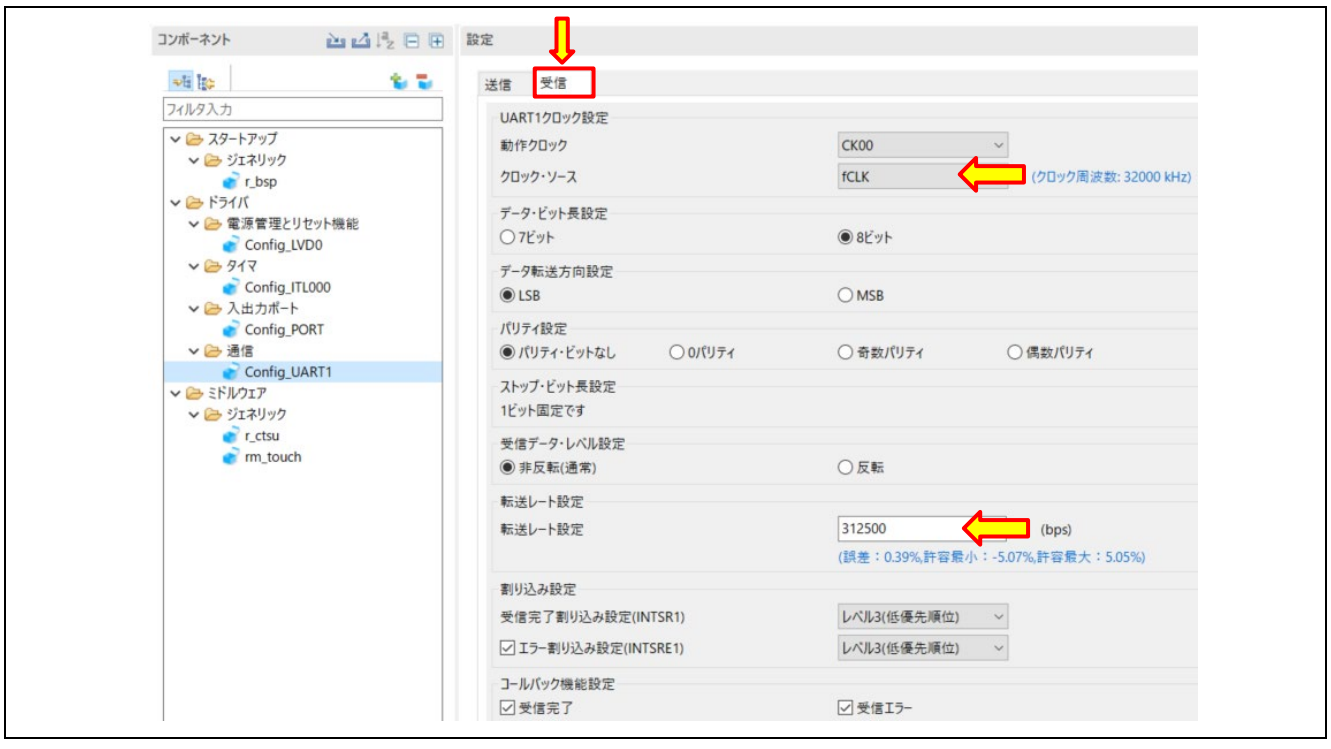

図 7-8 UART 通信コンポーネントの設定 (UART1 受信)

5. [**端子**]タブに移動します。

図 7-9 [端子]タブの選択

6. “**RxDA1**”機能を“**P33**”に、“**TxDA1**”機能を“**P34**”に割り当てます。

図 7-10 UART チャネル（UARTA1）の端子割り当て

注.　ツールで設定する 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。例えば、UART1 を使用する場合、下記画像のように設定が異なります。

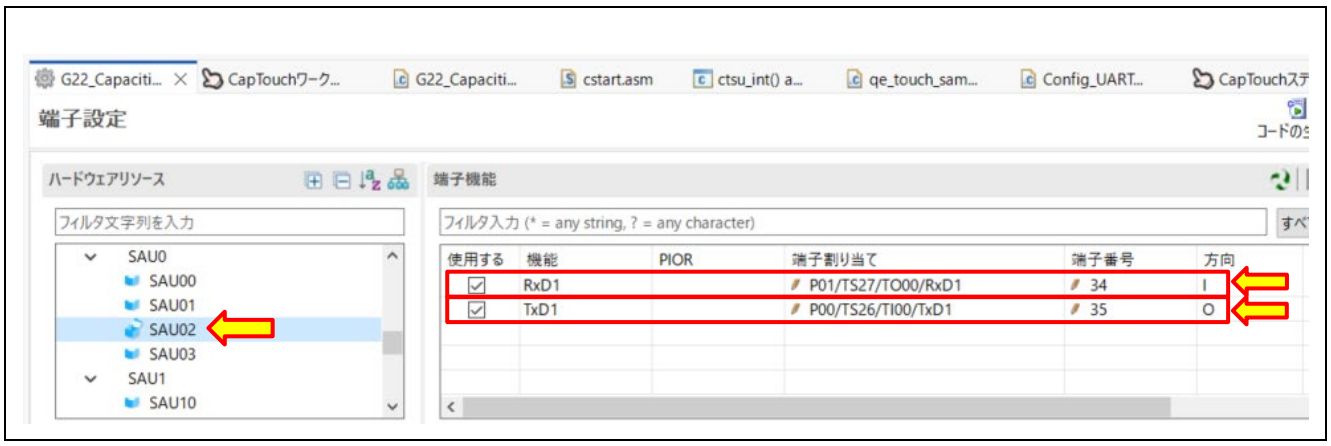

図 7-11 UART チャネル（UART1）の端子割り当て

7. スマート・コンフィグレータの右上の“コードの生成”アイコン をクリックして、プロジェクトに必要なコンポーネントのコードを追加します。

図 7-12 “コードの生成”アイコンの選択 (2)

8.　静電容量タッチインタフェース作成

1.　e² studio のメニュー[**Renesas Views**] - [**Renesas QE**] - [**CapTouch ワークフロー (QE)**]を選択し、プロジェクトに静電容量タッチの設定をするためのメインウィンドウを表示します。

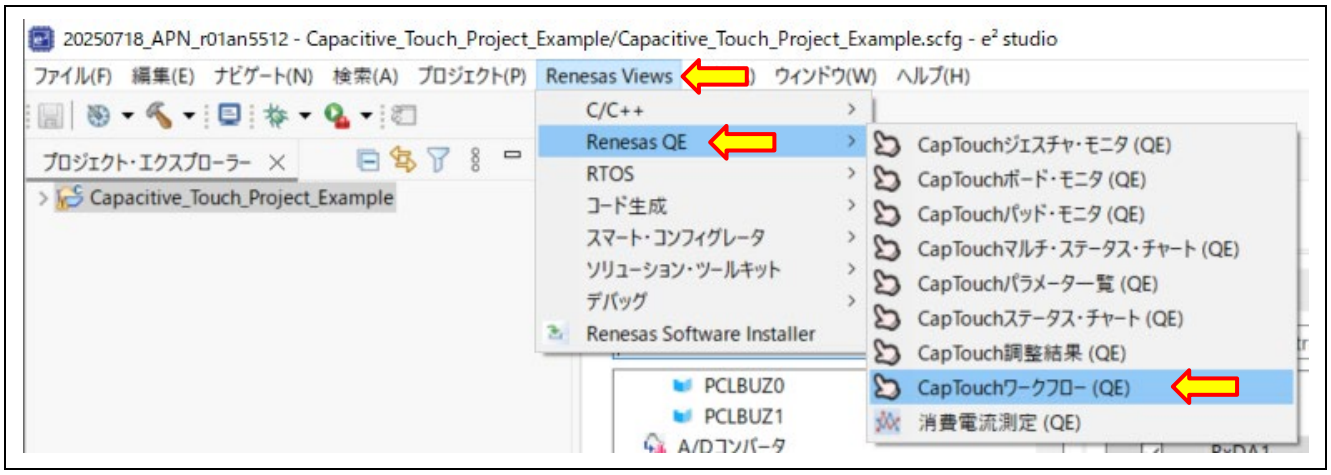

図 8-1 CapTouch ワークフロー (QE) の選択

注.　QE のバージョンによっては選択項目の表示名が異なります。
QE V3.1.0 以前では[CapTouch メイン (QE)]を選択してください。
QE V3.2.0 以降では[CapTouch ワークフロー (QE)]を選択してください。

2.　[**CapTouch ワークフロー (QE)**]の“**プロジェクトの選択**”のプルダウンメニューから“**Capacitive_Touch_Project_Example**”をクリックして、タッチインタフェースを設定するプロジェクトを選択します。

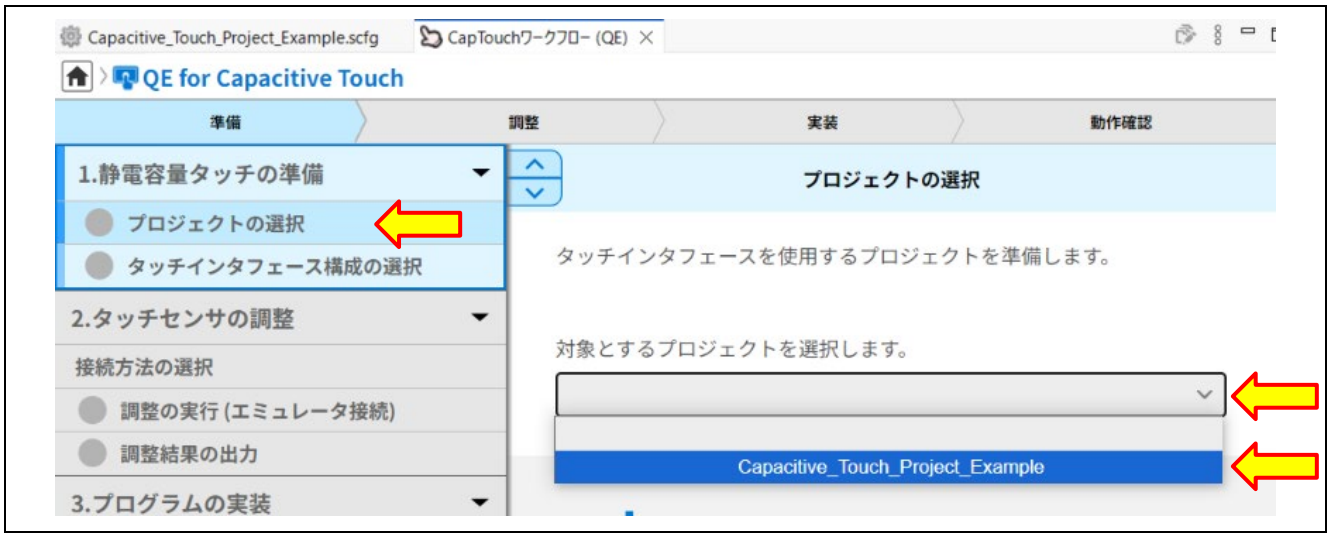

図 8-2　プロジェクトの選択

3. “**タッチインタフェース構成の選択**”のプルダウンメニューから“**タッチインタフェース構成の新規作成**”を選択します。

図 8-3 タッチインタフェース構成の新規作成

4. “**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。

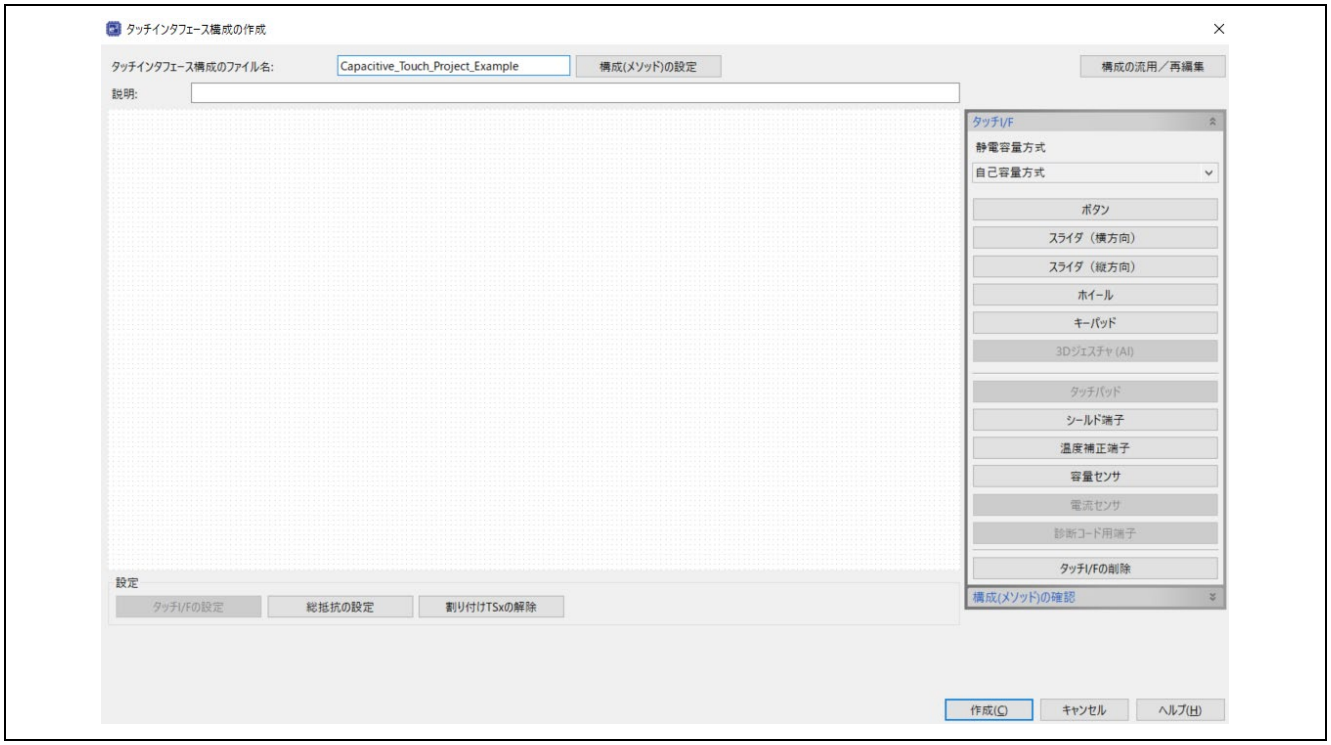

図 8-4 タッチインタフェース構成の作成 (1)

5. “**タッチインタフェース構成の作成**”ウィンドウの右側から[**ボタン**]を選択して、2 つのボタンを画面に追加してください。キーボードの[**Esc**]キーを押してタッチインタフェースの追加を終了すると以下のようになります。

図 8-5 タッチインタフェース構成の作成 (2)

6. “**Button00**”をダブルクリックし、“タッチインタフェースの設定”ダイアログを表示します。ここではプルダウンメニューから、このボタン“**Button00**”に **TS06** を割り当てます。

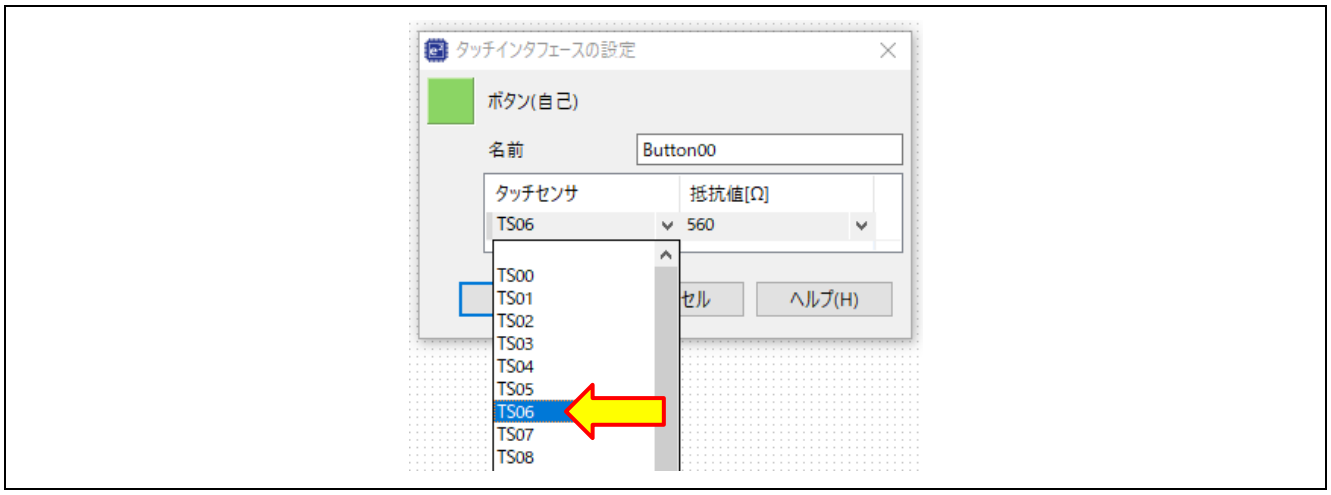

図 8-6 タッチインタフェース構成の作成 (3)

7. 前の手順(“**Button00**”)と同様に、“**Button01**”に **TS05** を割り当てます。タッチインタフェースは、以下のようになります。スマート・コンフィグレータで有効にしたセンサポートに従って、割り当てが正しく設定されると設定エラーの表示がなくなります。

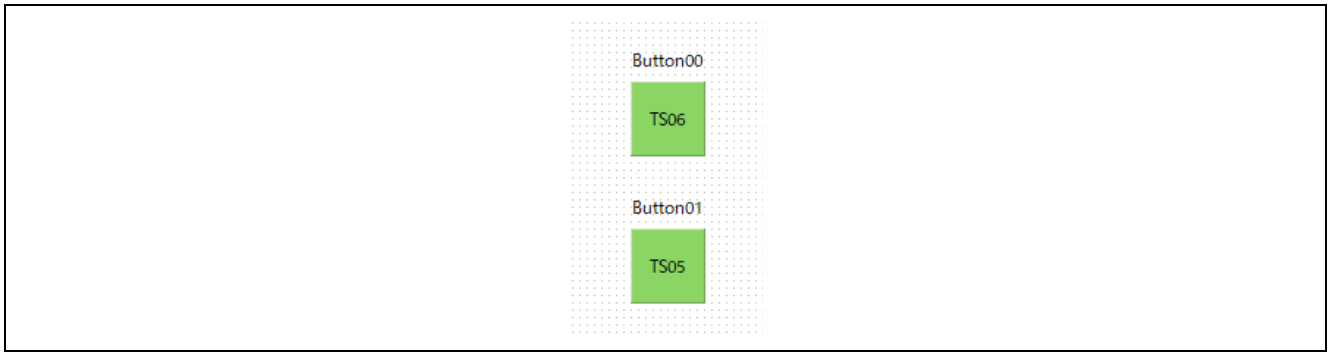

図 8-7 タッチインタフェース構成の作成 (4)

8. “タッチインタフェース構成の作成”ウィンドウの[**作成**]をクリックします。これでタッチインタフェースが設定されます。

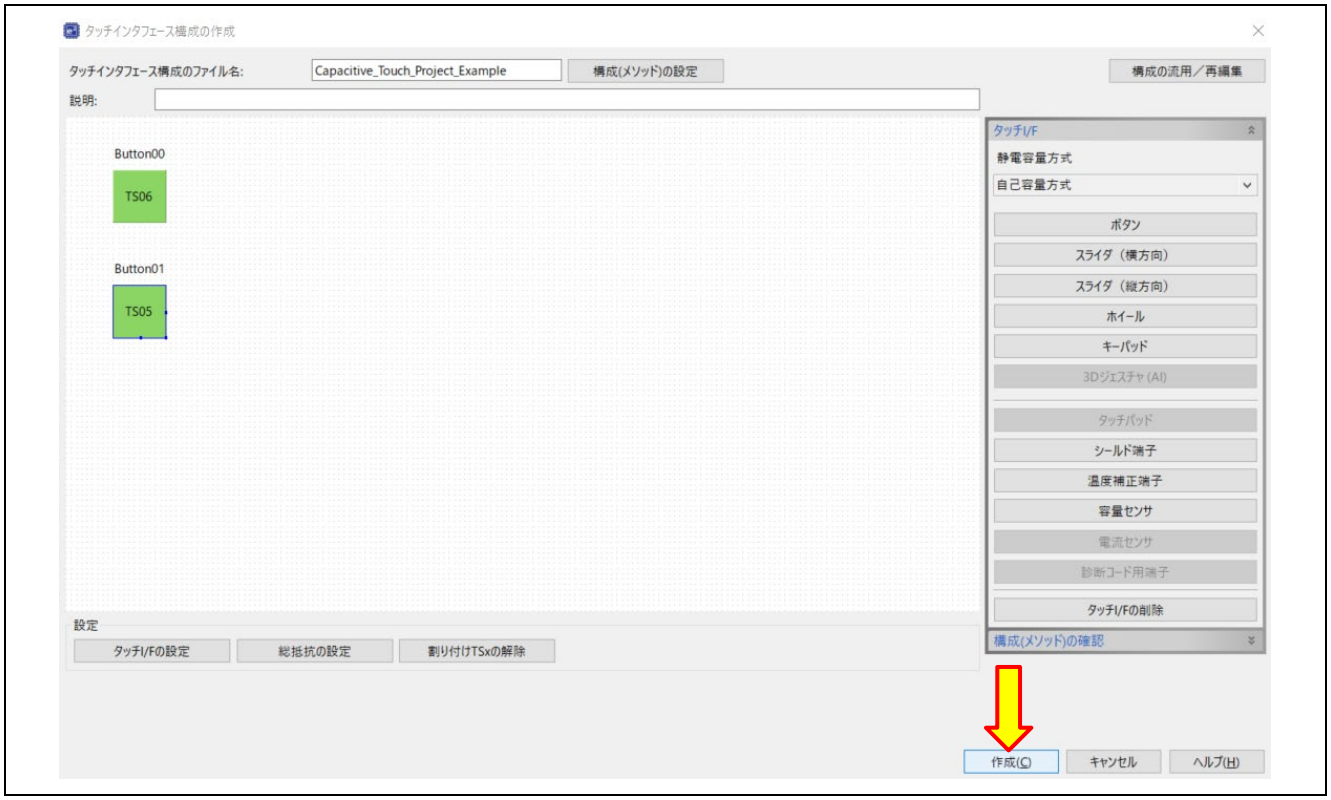

図 8-8 タッチインタフェース構成の作成 (5)

9. e² studio 左上のビルドアイコン をクリックしてビルドを開始します。“コンソール”ウィンドウで、ビルドした結果にエラーやワーニングがないことが確認できます。これで静電容量タッチインタフェースの作成は完了です。

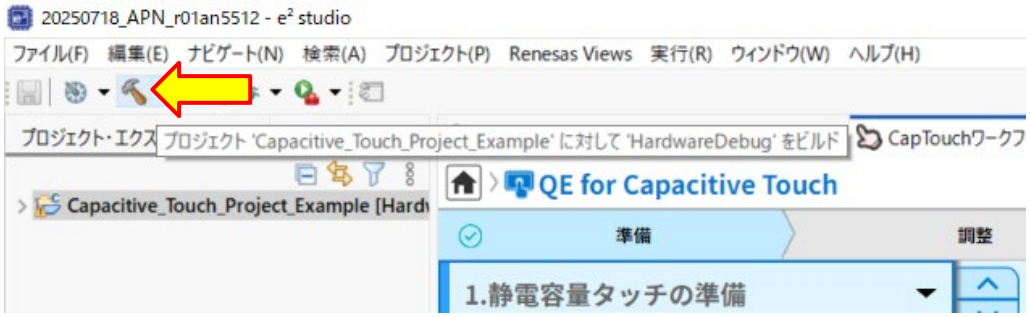

図 8-9 プロジェクトのビルド

9.　静電容量タッチセンサ・チューニング向けデバッグ構成の設定変更

デバッグセッション開始後にチューニングカーネルを MCU の RAM にダウンロードできるように、デバッグ構成を変更します。

1. アイコン横の▼をクリックし、タブから“**デバックの構成**”を選択してください。

図 9-1 "デバックの構成"の選択

2. 表示されたパネルから“使用するデバッグ構成(例: xxxxx HardwareDebug)”をクリックし、[**Debugger**]タブおよび[**Connection Settings**]タブを選択します。このアプリケーション例では、ターゲットボードの電源はエミュレータの電源から供給します。“**エミュレータから電源供給(最大 200mA)**”と“**供給電圧[V]**”が以下の赤枠部分のように設定されていることを確認します。

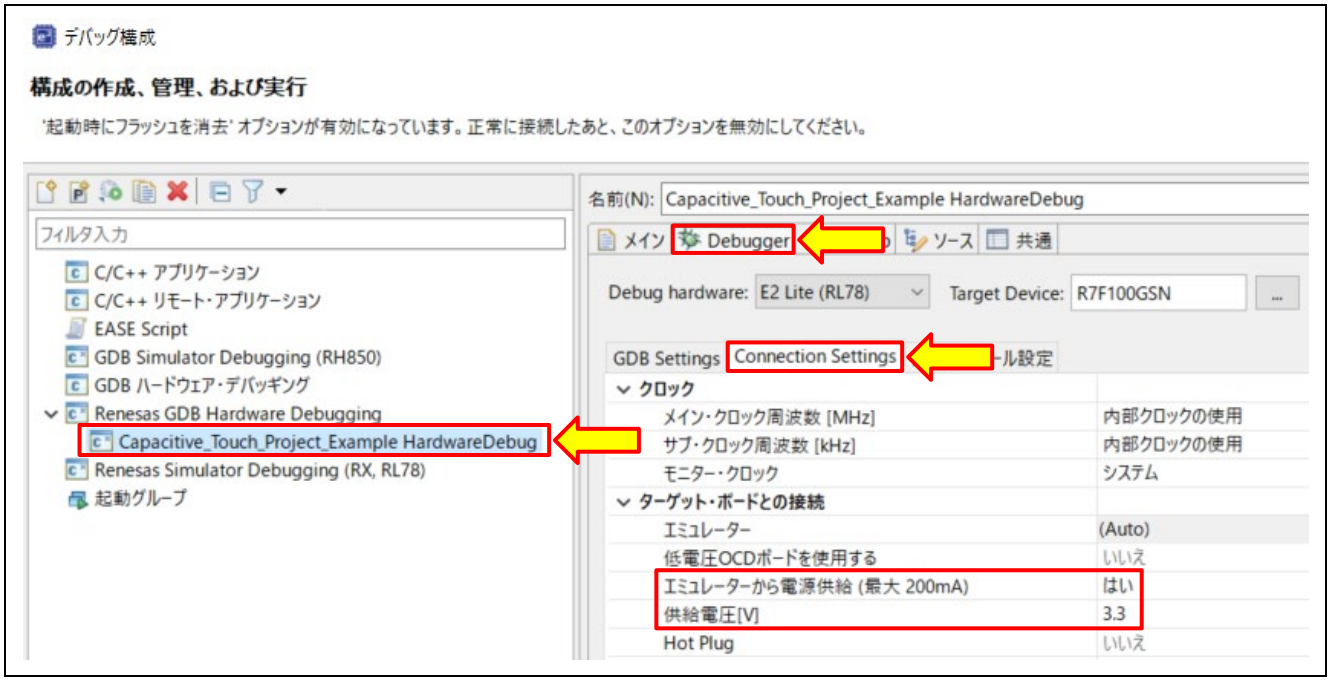

図 9-2　デバック構成の設定 (E2 Lite)

注 1.　動作確認を簡単に行うために、このアプリケーション例では、ターゲットボードの電源はエミュレータの電源から供給します。PC の USB ポートから、E2 emulator Lite を経由してターゲットボードに電源を供給することは可能ですが、ルネサスではターゲットボードで生成する電源を使用することを推奨しています。

注 2.　どのデバッグ方法が使用可能かは、ターゲットボードの仕様によります。使用するデバッグ方法に応じて“Debug hardware:”を選択し、項目を設定してください。例えば、COM port デバッグを行う場合、下記画像のように設定が異なります。

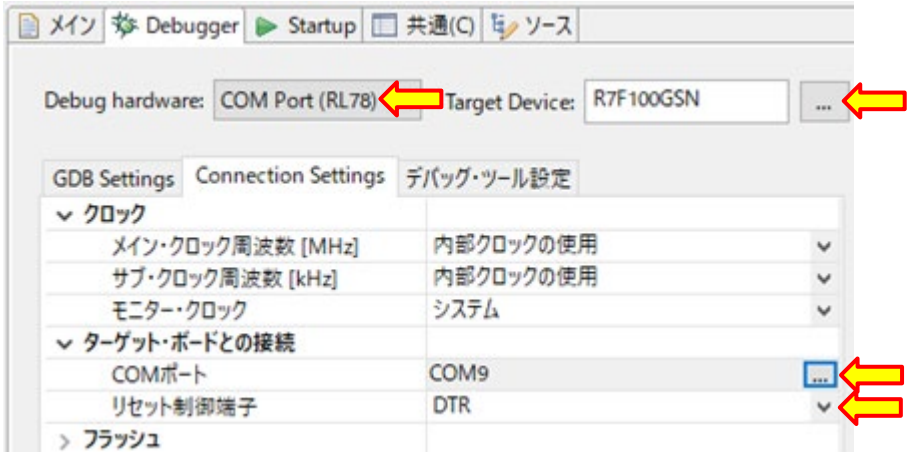

図 9-3 "デバックの構成"の選択（COM port デバッグ）

3. [**デバック・ツール設定**]タブを選択します。"メモリー"の"実行を一時停止してメモリアクセスする"を"はい"に設定してください。

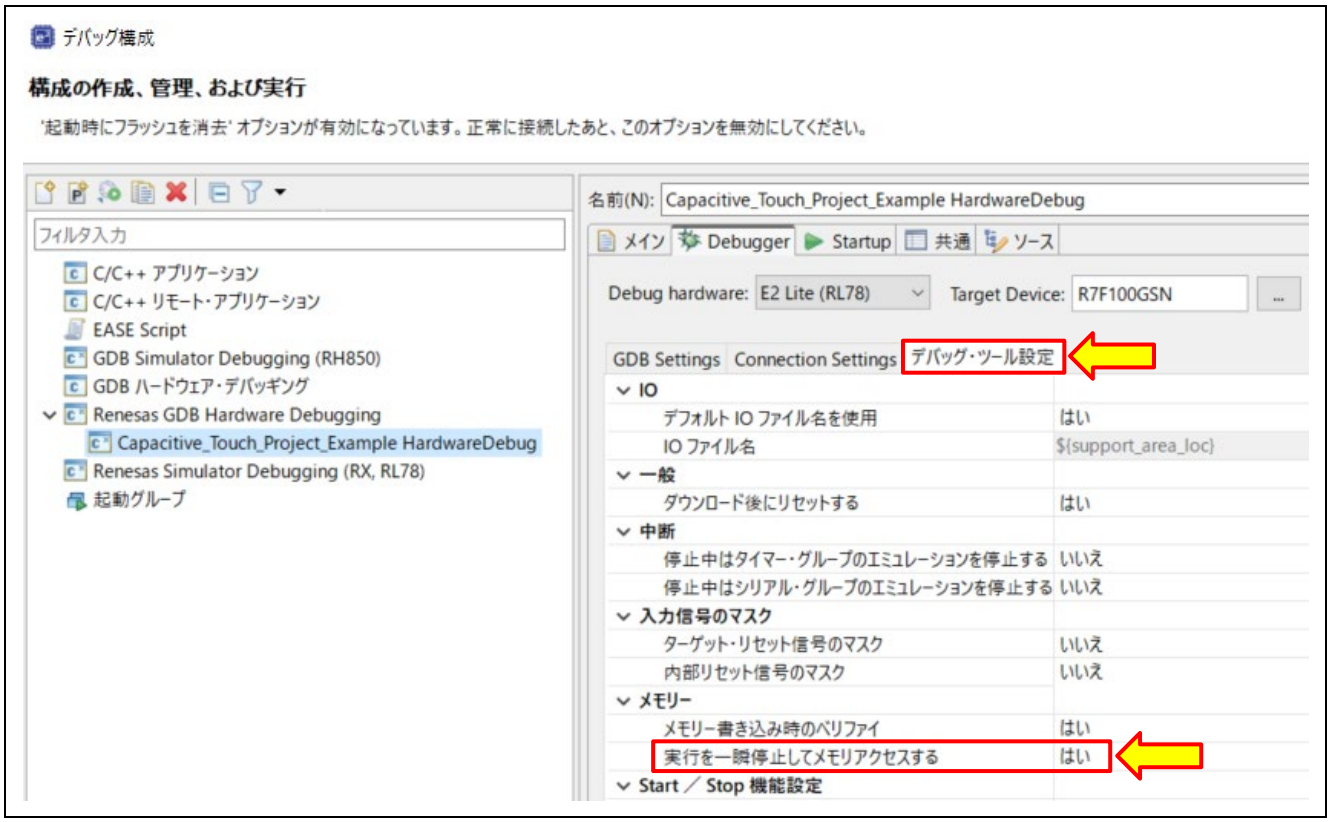

図 9-4 デバック・ツール設定

4. [**Startup**]タブを選択します。“**ブレークポイント設定先**:”と“**再開**”のチェックボックスが以下のようにチェックされていることを確認し、[**適用**]および[**閉じる**]をクリックします。これでチューニングのためのデバック構成の設定は完了です。

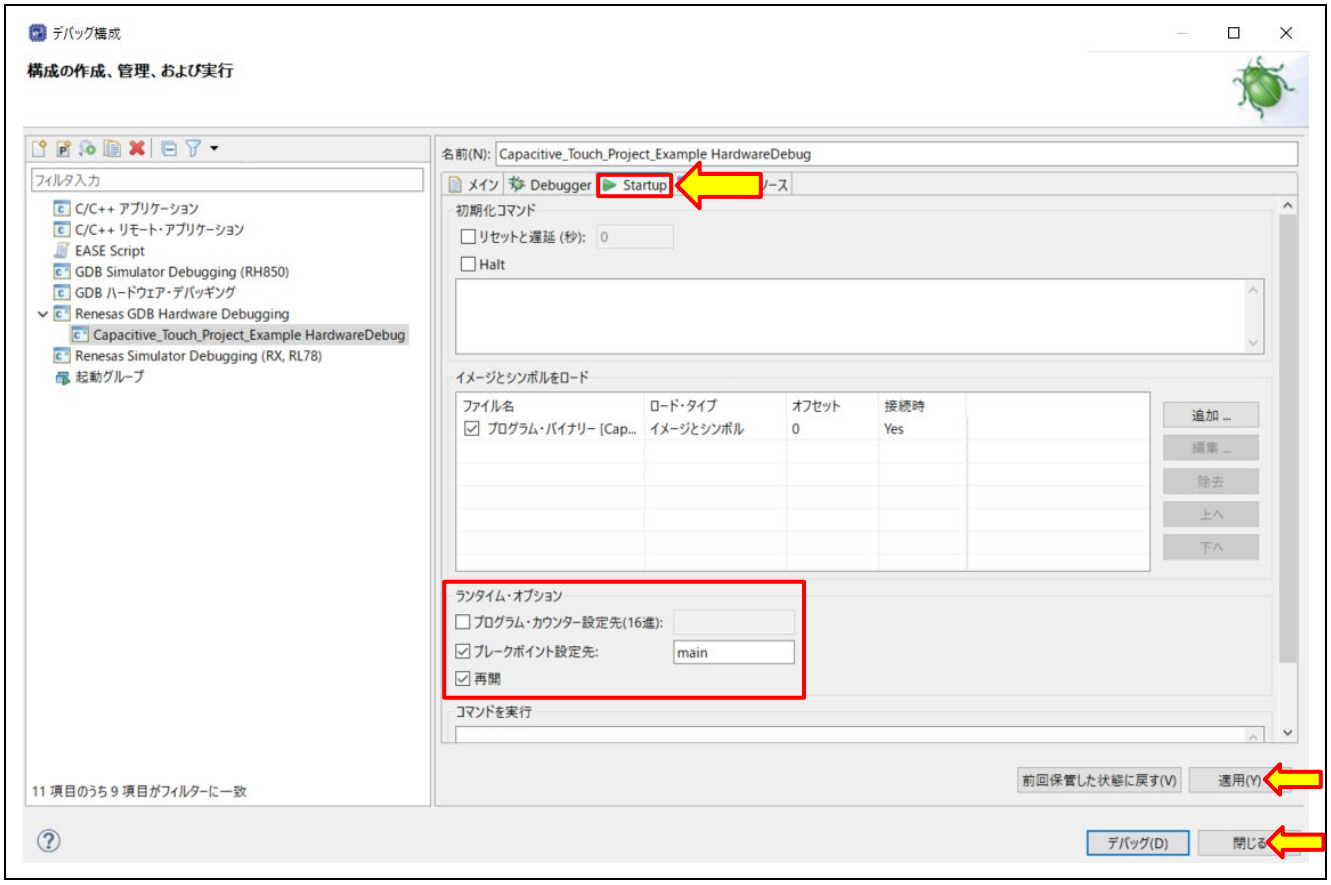

図 9-5 ランタイム・オプションの設定

10. QE for Capacitive Touch を使用した静電容量タッチセンサ・チューニング

QE for Capacitive Touch を使用してプロジェクトをチューニングします。以下に手順を示します。

1. [**CapTouch ワークフロー (QE)**]の[**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。

注.　動作確認を簡単に行うために、このアプリケーション例では、ターゲットボードの電源はエミュレータの電源から供給します。PC の USB ポートから、E2 emulator Lite を経由してターゲットボードに電源を供給することは可能ですが、ルネサスではターゲットボードで生成する電源を使用してチューニングすることを推奨しています。

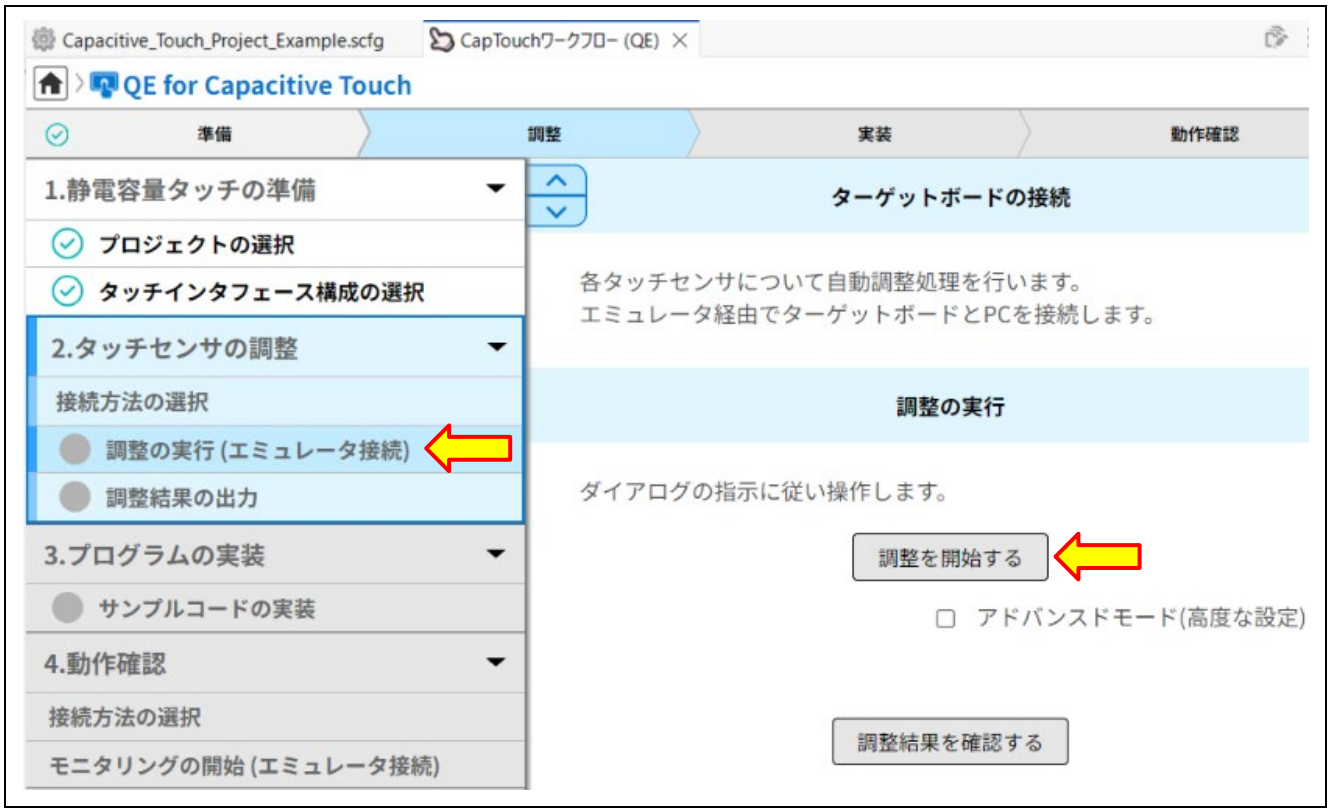

図 10-1 自動チューニングの開始

2. マイコンへ供給する電圧値[V]と同じ値を以下のように設定し、[**OK**]をクリックします。

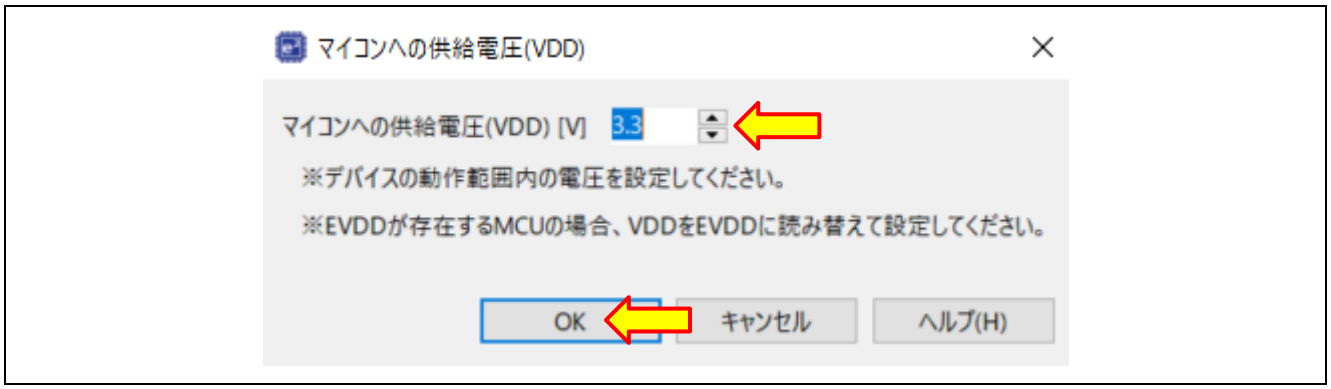

図 10-2 マイコンへ供給する電圧値の設定(チューニング用設定)

注.　このアプリケーション例では、E2 emulator Lite から電源供給します。そのため、“マイコンへの供給電圧(VDD) [V]”は、3.3 V です。

3. デバッグセッションの開始時、e² studio はデバッグ・パースペクティブに切り替える旨のメッセージを表示することがあります。[**常にこの設定を使用する(R)**]をチェックし、[**切り替え**]をクリックしてデバッグセッションと QE for Capacitive Touch の自動チューニングを続行してください。

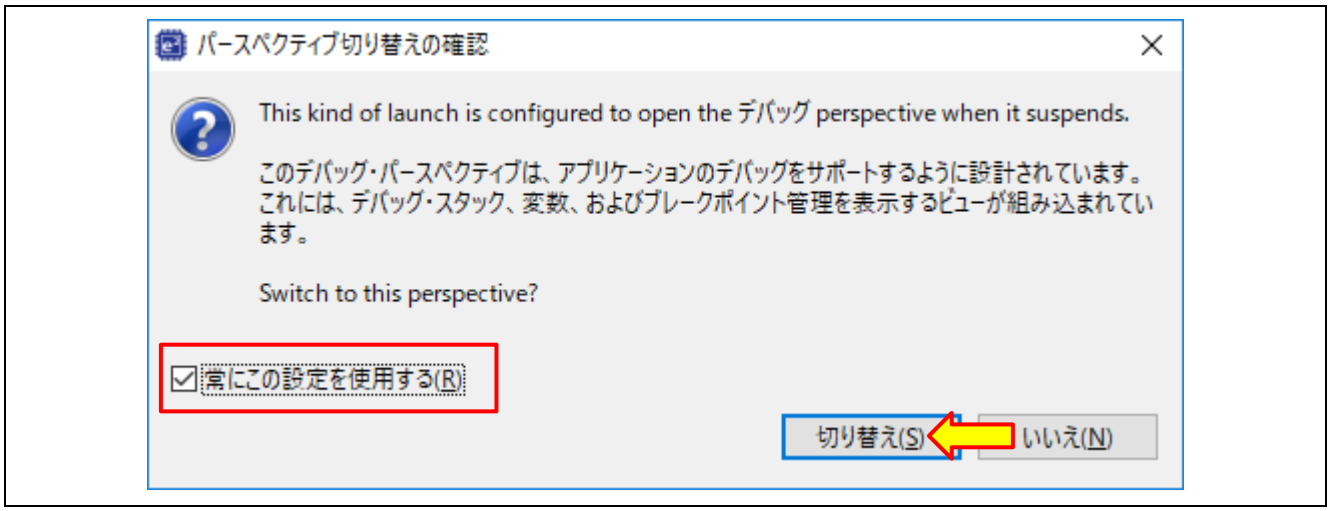

図 10-3 デバック・パースペクティブの切り替え

4. 以下のメッセージが表示されます。[はい]をクリックします。

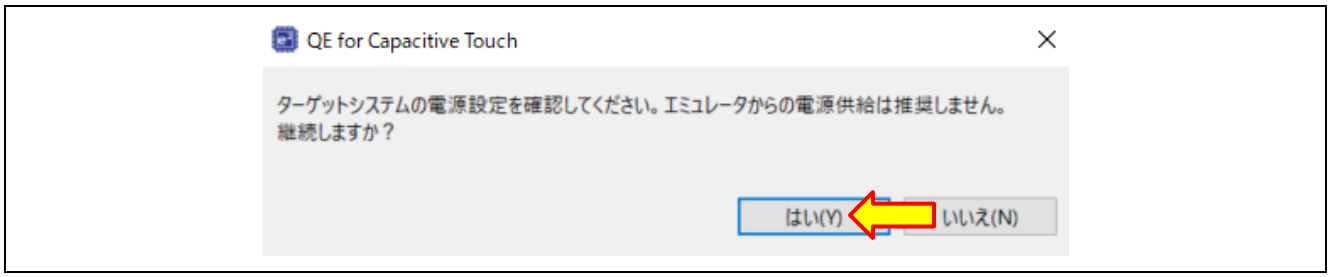

図 10-4 ターゲットシステムの電源設定に関する確認

5. QE for Capacitive Touch の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、確認してください。表示例を以下に示します。通常、初期のチューニングプロセス中は操作を必要としません。

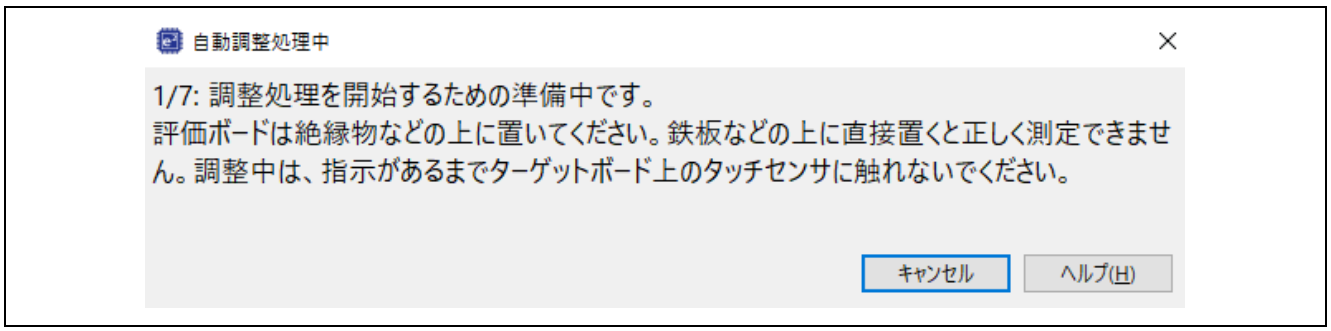

図 10-5 "自動調整処理中”ダイアログ(初期チューニングプロセス中)

いくつかの工程を経て、以下のようなダイアログが表示されます。ここではチューニングプロセスにおけるタッチ感度の計測をします。ダイアログで表示されているセンサ(**Button01**/**TS05**)を通常の圧力でタッチします。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増えます。センサに触れたまま、PC の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。

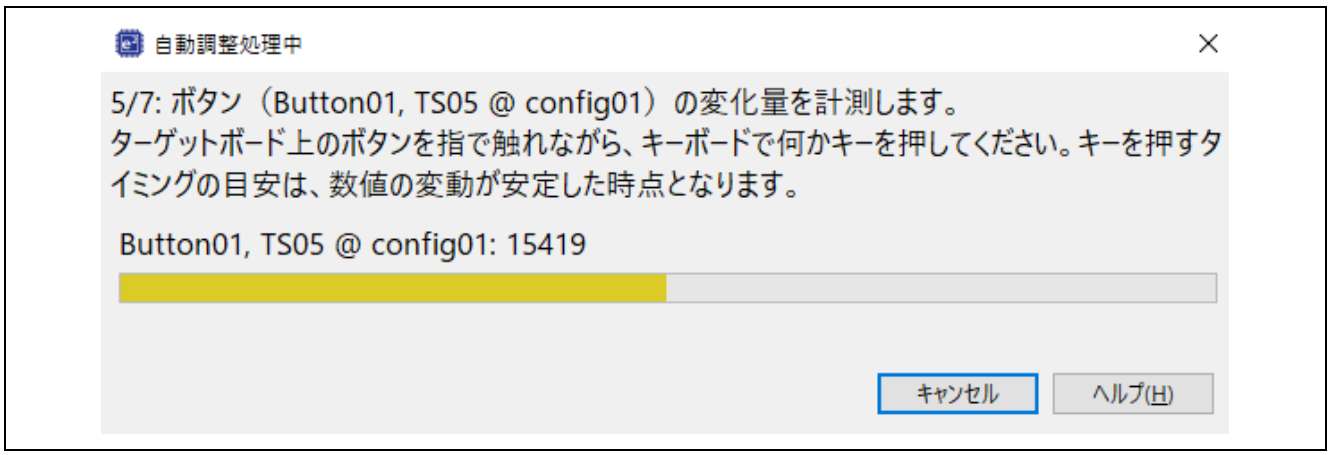

図 10-6 "自動調整処理中”ダイアログ(タッチ感度の計測中)

6. **Button00/TS06** に対して、前の手順を繰り返します。

7. 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しきい値を確認できます。このしきい値はミドルウェアでタッチのイベント判定に使用されます。

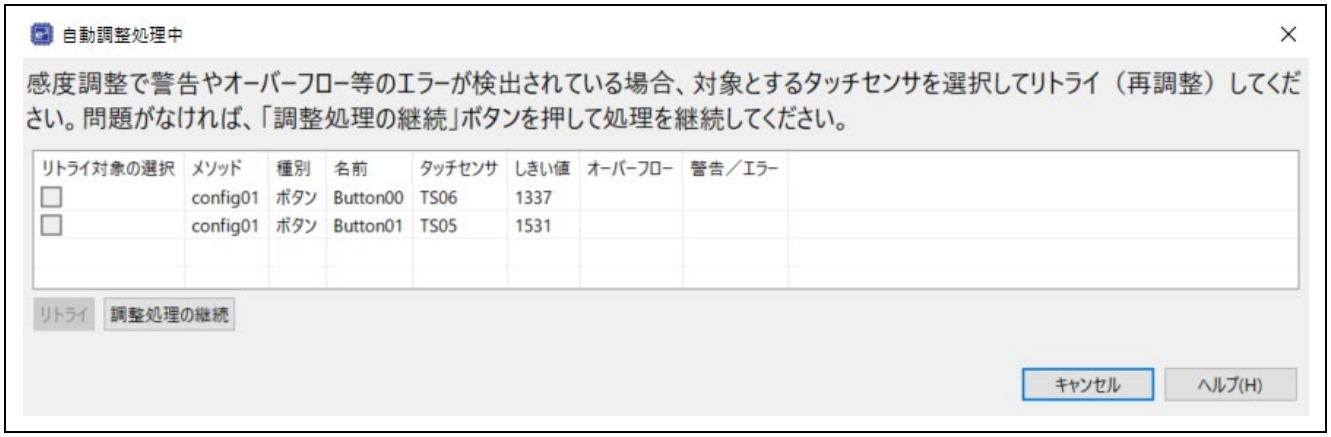

図 10-7 調整結果の表示

8. 表示されたダイアログの[**調整処理の継続**]をクリックします。これでチューニングプロセスは終了し、ターゲットボードとのデバッグセッションを切断します。[**CapTouch ワークフロー (QE)**]に戻ります。

図 10-8 調整処理の継続

9. [**CapTouch ワークフロー (QE)**]の[**調整結果を確認する**]をクリックすると、[**CapTouch 調整結果 (QE)**]が表示され詳細なチューニング結果を確認できます。

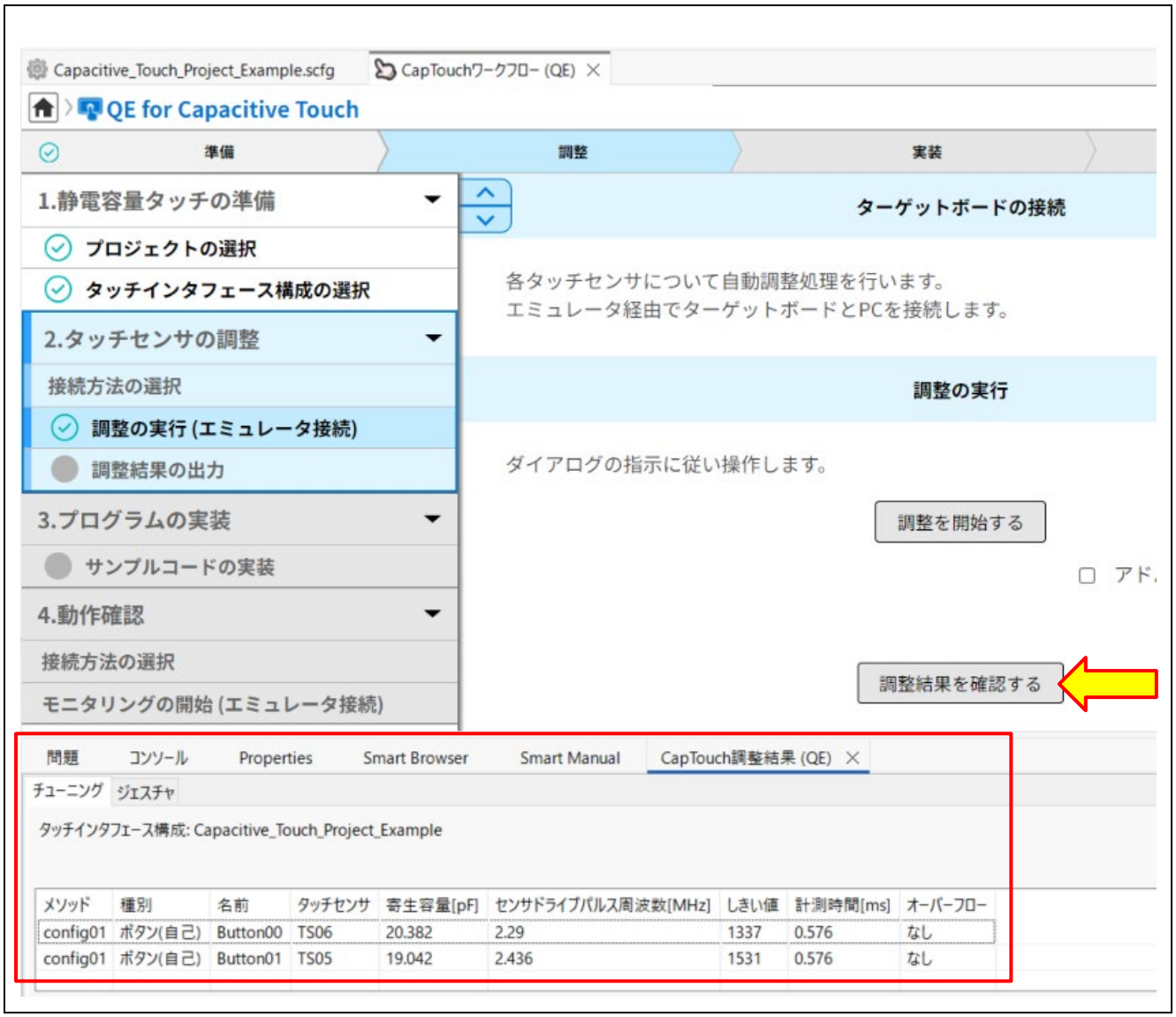

図 10-9 CapTouch 調整結果 (QE)

10. チューニングされたパラメータファイルの出力を行います。[**ファイルを出力する**]をクリックします。

図 10-10 パラメータファイルの出力

11. “**プロジェクト・エクスプローラー**”ウィンドウで **qe_touch_config.c** と **qe_touch_config.h**、**qe_touch_define.h** が追加されたことを確認できます。これらのファイルにはドライバを使用したタッチ検出を有効にするために必要なチューニング情報が含まれています。

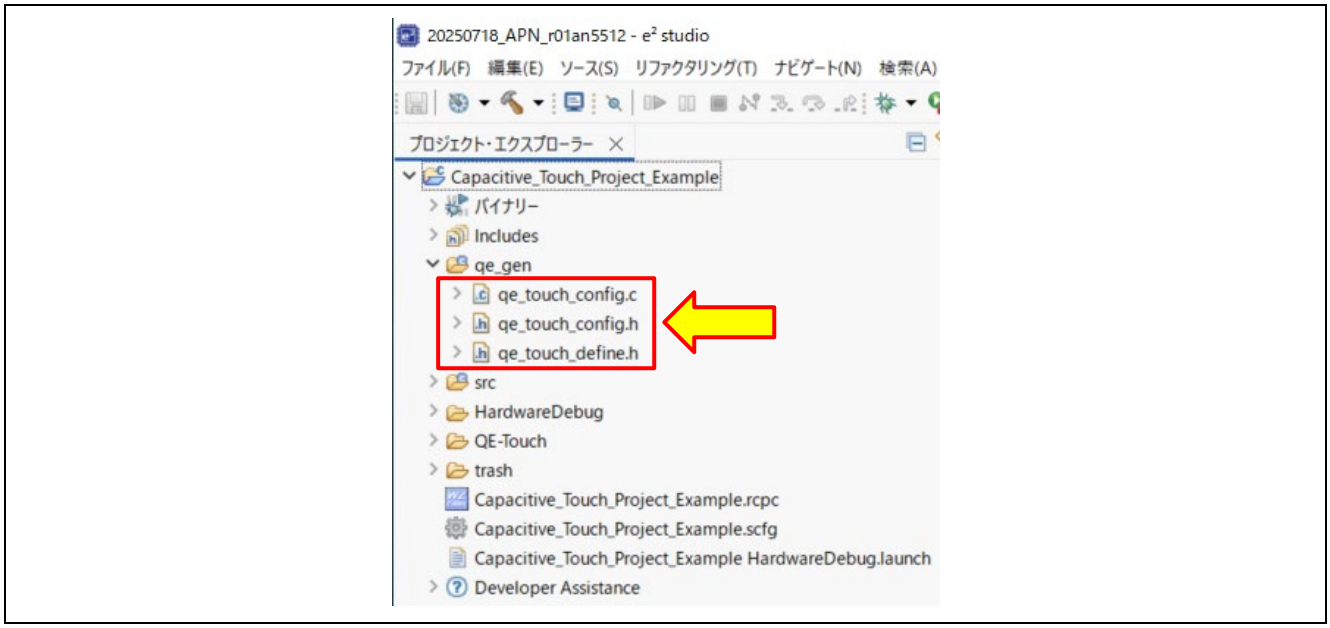

図 10-11 パラメータファイルの出力

12. e² studio の左上の アイコンをクリックしてプロジェクトをビルドします。“コンソール”ウィンドウで、ビルドした結果にエラーがないことが確認できます。

    これで QE for Capacitive Touch を使用した静電容量タッチセンサ・チューニングは完了です。

11. アプリケーションに rm_touch ミドルウェアの API コールを追加

rm_touch ミドルウェアの API コールをプロジェクトに追加し、静電容量タッチ制御を有効にします。

以下に手順を示します。

1. タッチセンサの状態をスキャンするプログラムを実装するために、[**CapTouch ワークフロー (QE)**]の[**例を表示する**]をクリックします。

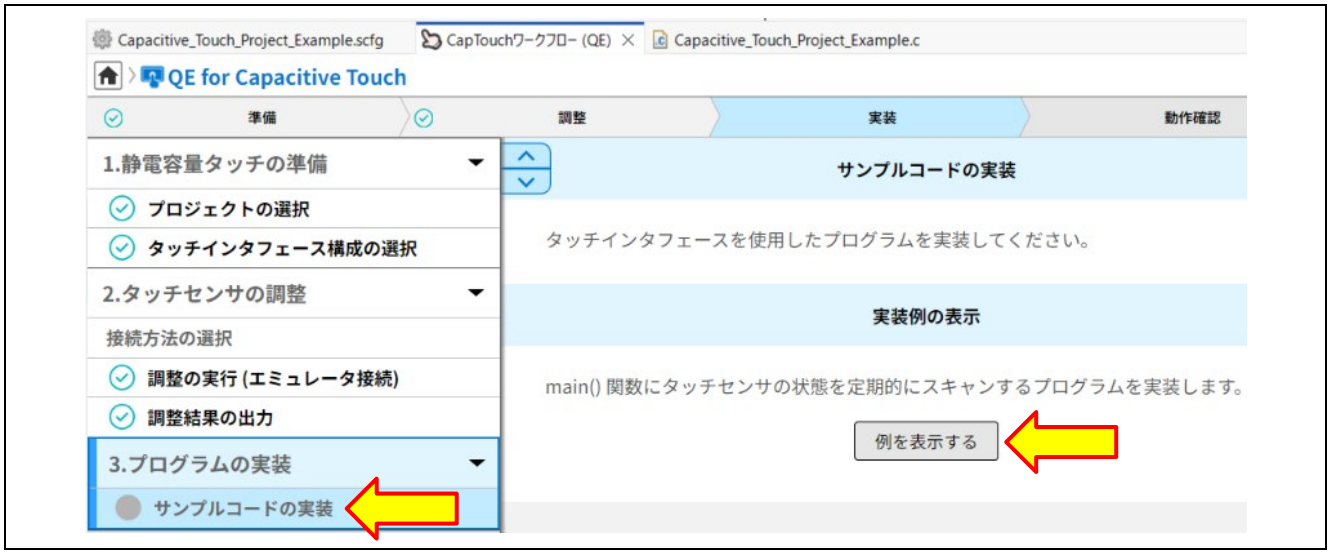

図 11-1 サンプルコード例の表示

2. “サンプルコードの表示”ウィンドウが開き、サンプルコードが表示されます。サンプルコードを出力するために[**ファイルに出力**]および[**OK**]をクリックしてください。

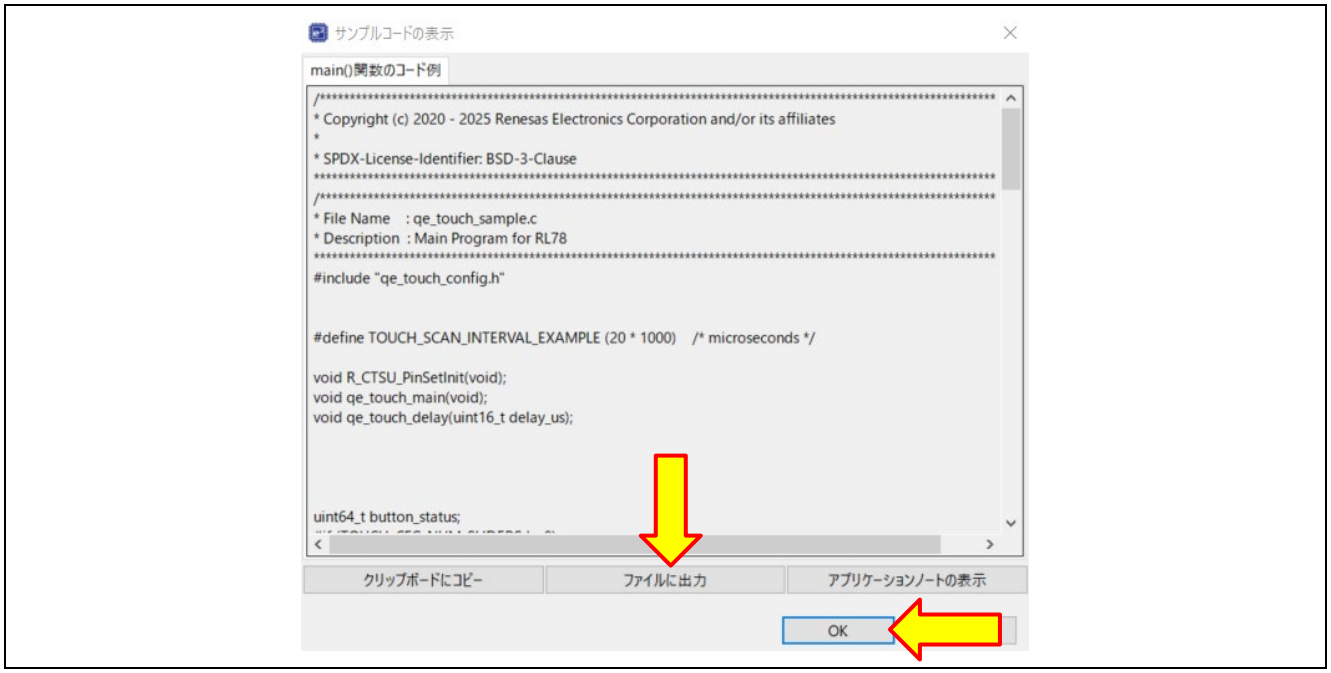

図 11-2 サンプルコードのファイル出力

3. “プロジェクト・エクスプローラー”より“**qe_touch_sample.c**”ファイルが生成されたことを確認してください。

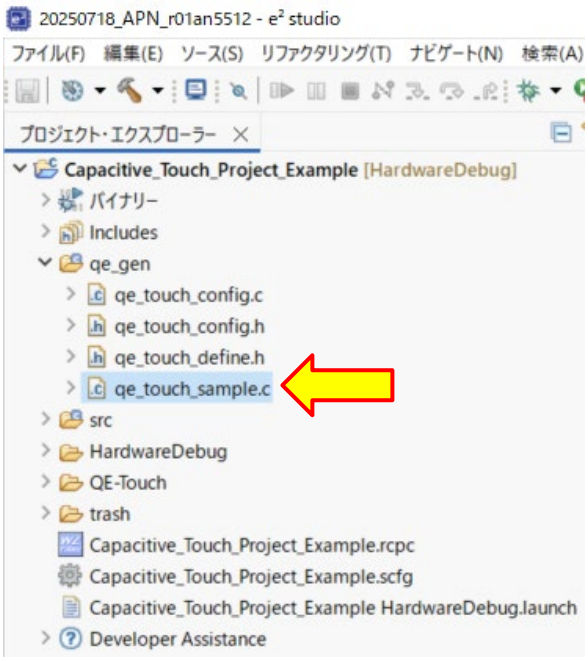

図 11-3 QE 出力コード (qe_touch_sample.c) の生成

4. “**Capacitive_Touch_Project_Example.c**”ファイルを開きます。

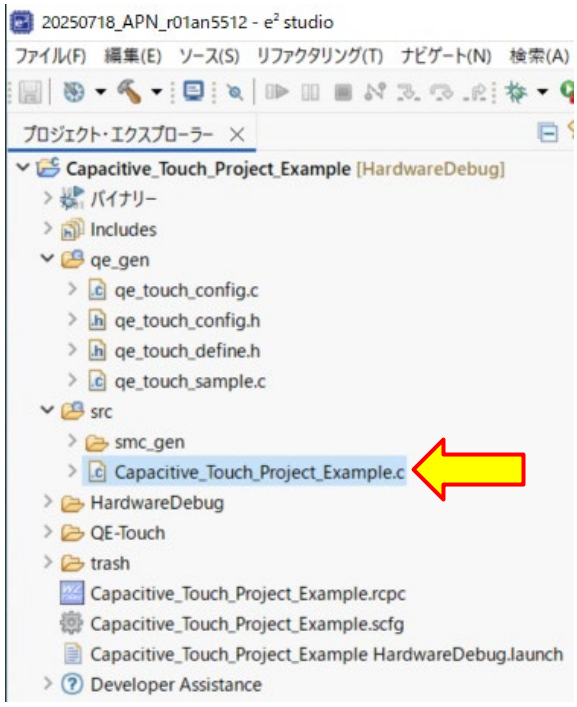

図 11-4 Capacitive_Touch_Project_Example.c の選択

5. main()関数から qe_touch_main()関数をコールします。“**Capacitive_Touch_Project_Example.c**”ファイルへ下記画像のように赤枠部分のコード(“**void qe_touch_main(void);**”および“**qe_touch_main();**”)を追加してください。

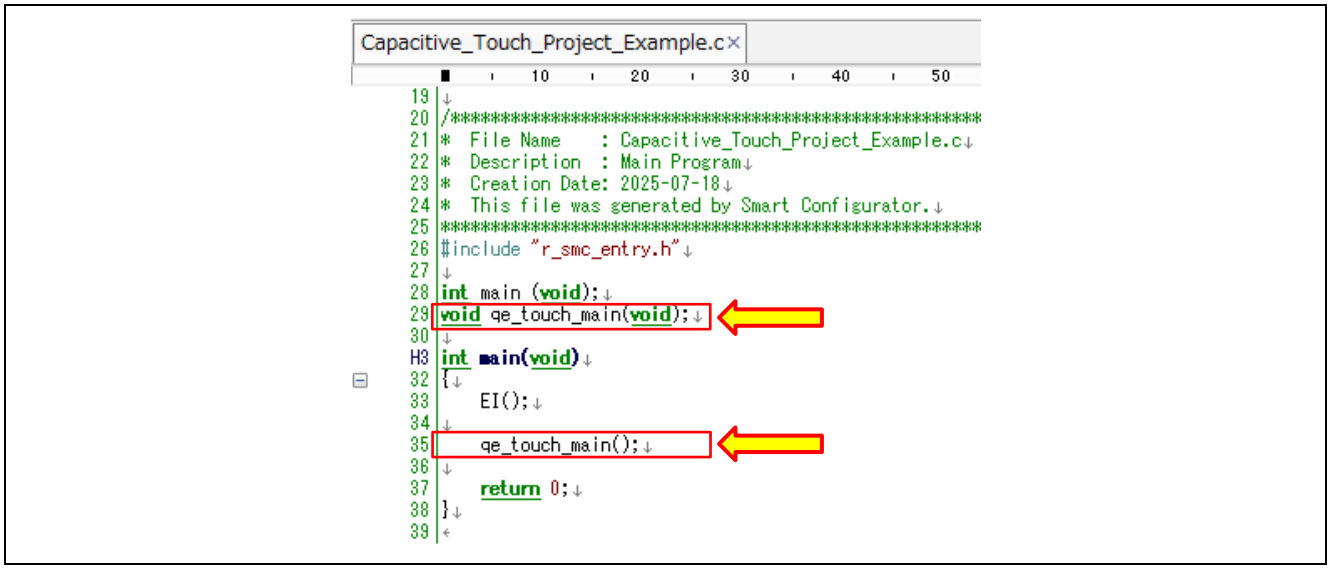

図 11-5 qe_touch_main()関数の呼び出し設定

6. e² studio 左上のビルドアイコン をクリックしてビルドを開始します。
“コンソール”ウィンドウで、ビルドした結果にエラーやワーニングがないことが確認できます。
これで、アプリケーション例に必要なコードの変更はすべて終了です。

12. [追加機能] UART を使用したシリアル通信モニタの設定 (2/3)

注.　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。
また一方で、タッチ性能のモニタリングは、シリアル通信を介して行うこともできます。 したがって、スムーズにモニタリングを行いたい場合は、シリアル通信を介したモニタリング機能を追加してください(推奨設定)。

以下に示す 7 章、12 章および 14 章(本章を含む)では、UART を使用したシリアル通信モニタの設定について説明します。

- 7. [追加機能] UART を使用したシリアル通信モニタの設定 (1/3)
- 12. [追加機能] UART を使用したシリアル通信モニタの設定 (2/3)
- 14. [追加機能] UART を使用したシリアル通信モニタの設定 (3/3)

1. “**Config_UARTA1_user.c**”ファイルを開きます。

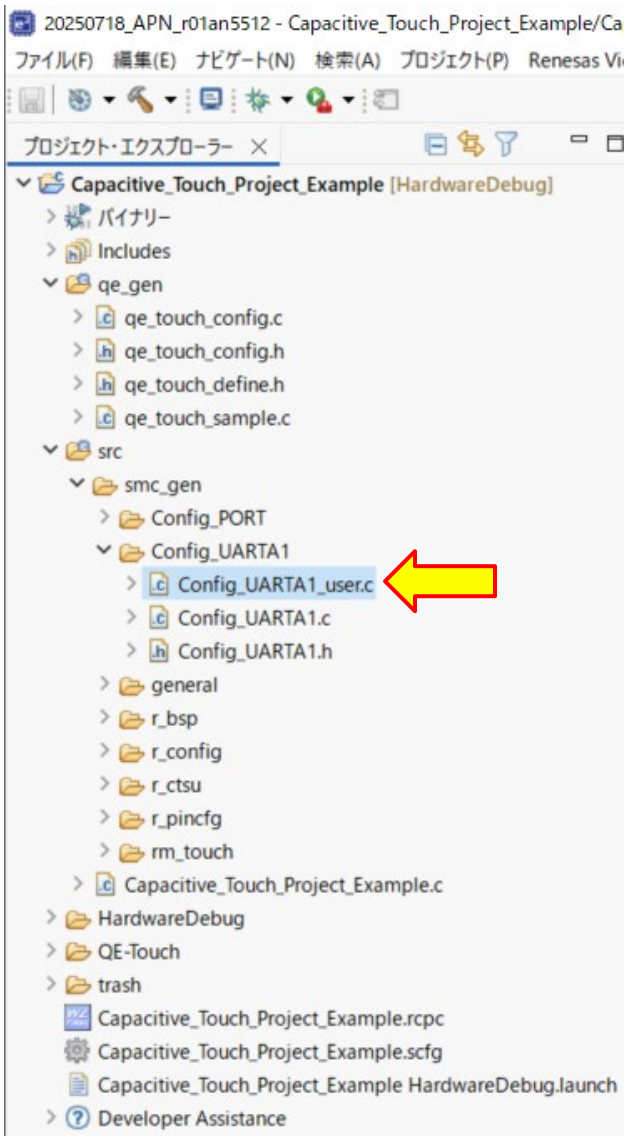

図 12-1 Config_UARTA1_user.c の選択

2. “**Config_UARTA1_user.c**”ファイルへ下記画像のように赤枠部分のコード“**extern void touch_uart_callback(uint16_t event);**”を追加してください。

```
Config_UARTA1_user.c×
43 /***********************************************************************
44 Global variables and functions↓
45 ***********************************************************************
46 extern volatile uint16_t g_uarta1_rx_total_num;↓
47 extern volatile uint8_t * gp_uarta1_rx_address;↓
48 extern volatile uint16_t g_uarta1_rx_num;↓
49 extern volatile uint8_t * gp_uarta1_tx_address;↓
50 extern volatile uint16_t g_uarta1_tx_count;↓
51 /* Start user code for global. Do not edit comment generated here */↓
52 extern void touch_uart_callback(uint16_t event);↓
53 /* End user code. Do not edit comment generated here */↓
```

図 12-2 Config_UARTA1_user.c (1/3)

3. “**Config_UARTA1_user.c**”ファイルへ下記画像のように赤枠部分のコード“**touch_uart_callback(0);**”を追加してください。

```
Config_UARTA1_user.c×
67 /***********************************************************************************************************************
68 * Function Name: r_Config_UARTA1_callback_sendend↓
69 * Description  : This function is a callback function when UARTA1 finishes transmission.↓
70 * Arguments    : None↓
71 * Return Value : None↓
72 ***********************************************************************************************************************
H3 static void r_Config_UARTA1_callback_sendend(void)↓
74 {↓
75     /* Start user code for r_Config_UARTA1_callback_sendend. Do not edit comment generated here */↓
76     touch_uart_callback(0);↓
77     /* End user code. Do not edit comment generated here */↓
78 }↓
```

図 12-3 Config_UARTA1_user.c (2/3)

4. “**Config_UARTA1_user.c**”ファイルへ下記画像のように赤枠部分のコード“**touch_uart_callback(1);**”を追加してください。

```
Config_UARTA1_user.c×
80 /***********************************************************************************************************************
81 * Function Name: r_Config_UARTA1_callback_receiveend↓
82 * Description  : This function is a callback function when UARTA1 finishes reception.↓
83 * Arguments    : None↓
84 * Return Value : None↓
85 ***********************************************************************************************************************
H3 static void r_Config_UARTA1_callback_receiveend(void)↓
87 {↓
88     /* Start user code for r_Config_UARTA1_callback_receiveend. Do not edit comment generated here */↓
89     touch_uart_callback(1);↓
90     /* End user code. Do not edit comment generated here */↓
91 }↓
```

図 12-4 Config_UARTA1_user.c (3/3)

5. e² studio の左上の アイコンをクリックしてプロジェクトをビルドします。“コンソール”ウィンドウで、ビルドした結果にエラーがないことが確認できます。

13. “式”ウィンドウと QE for Capacitive Touch によるモニタリング

1. e² studio の左上にある アイコンをクリックしてデバッグセッションを開始します。

図 13-1 デバッグアイコンの選択

2. デバッグセッションは **main()**関数コールでストップします。

3. **qe_touch_main()**関数の宣言を開きます。

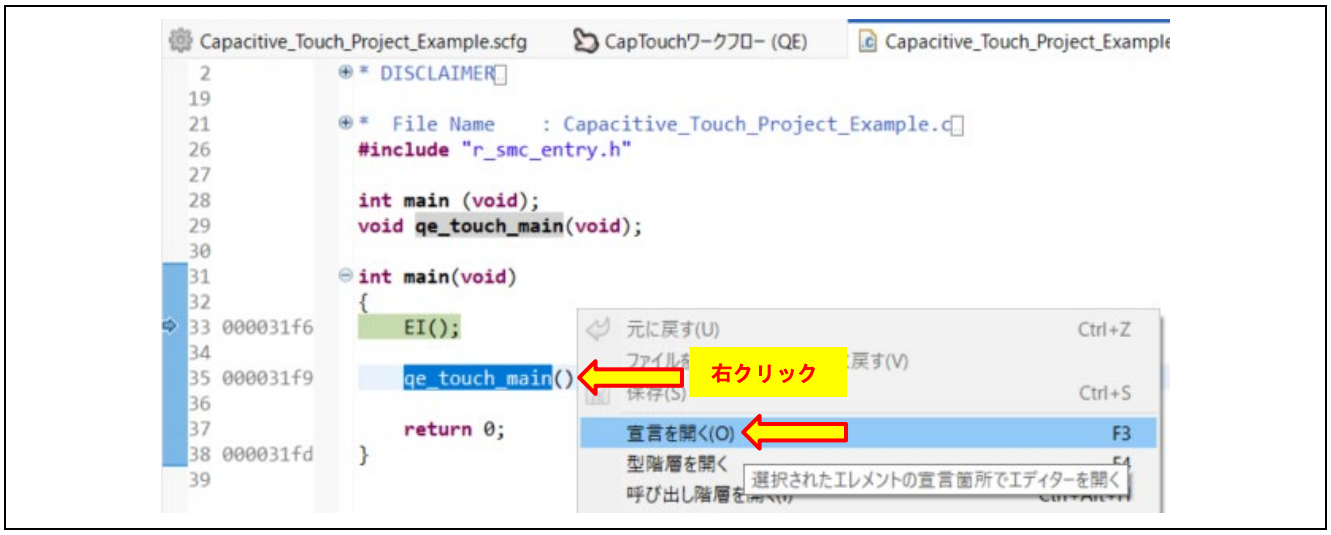

図 13-2 “宣言を開く”の選択

4. **qe_touch_sample.c** ファイルを下にスクロールして while(true)ループの **RM_TOUCH_DataGet()**関数を表示し、“式”ウィンドウに変数 **button_status** を追加します。

図 13-3 “監視式を追加”の選択

5. “式”ウィンドウで追加した変数のリアルタイム・リフレッシュを有効に設定します。

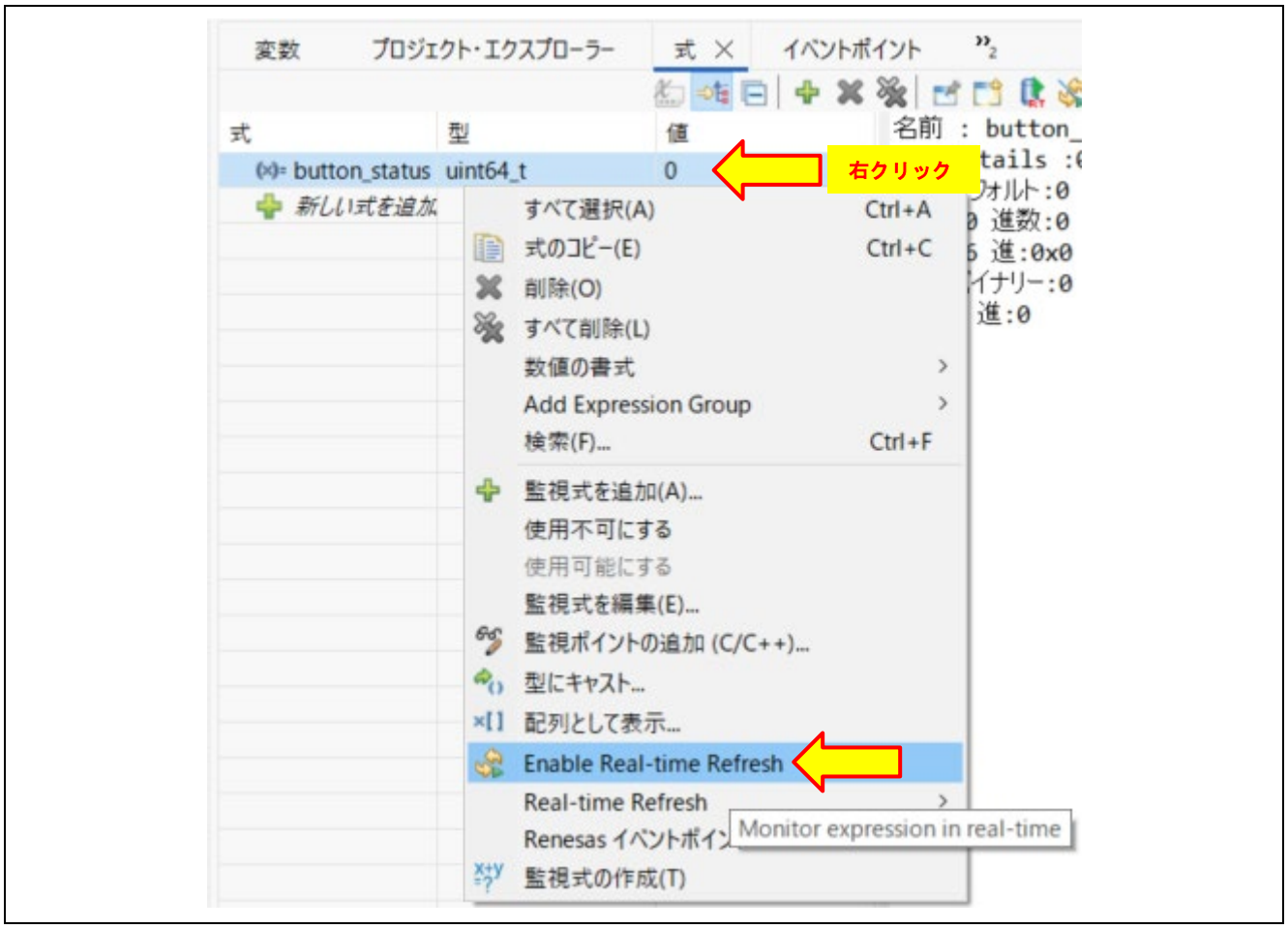

図 13-4 リアルタイム・リフレッシュの選択

6. 変数のリアルタイム・リフレッシュが有効に設定されていることを確認します。

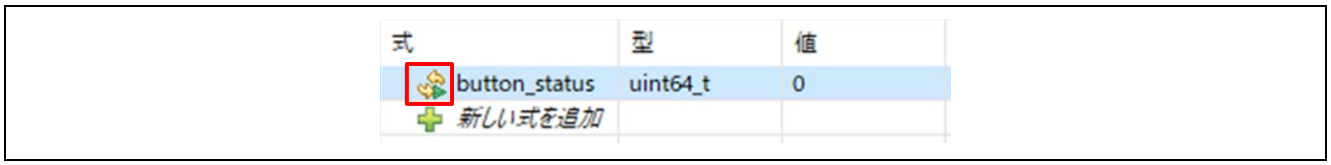

図 13-5 リアルタイム・リフレッシュの有効設定確認

7. e² studio のツールバーにある“**再開**”ボタンをクリックしプログラム実行を続行します。

図 13-6 “再開”ボタンの選択

8. このアプリケーションノートの「**8. 静電容量タッチインタフェース作成**」章で“**Button01**”として定義した、ボード上の **TS05** をタッチします。“**Button01**”をタッチすると、“式”ウィンドウの **button_status** の値が 1 になることを確認できます。

図 13-7 “button_status”の確認

9. [**CapTouch ワークフロー (QE)**]から“動作確認”の“**接続方法の選択**”をクリックし、[**エミュレータ**]を選択します。

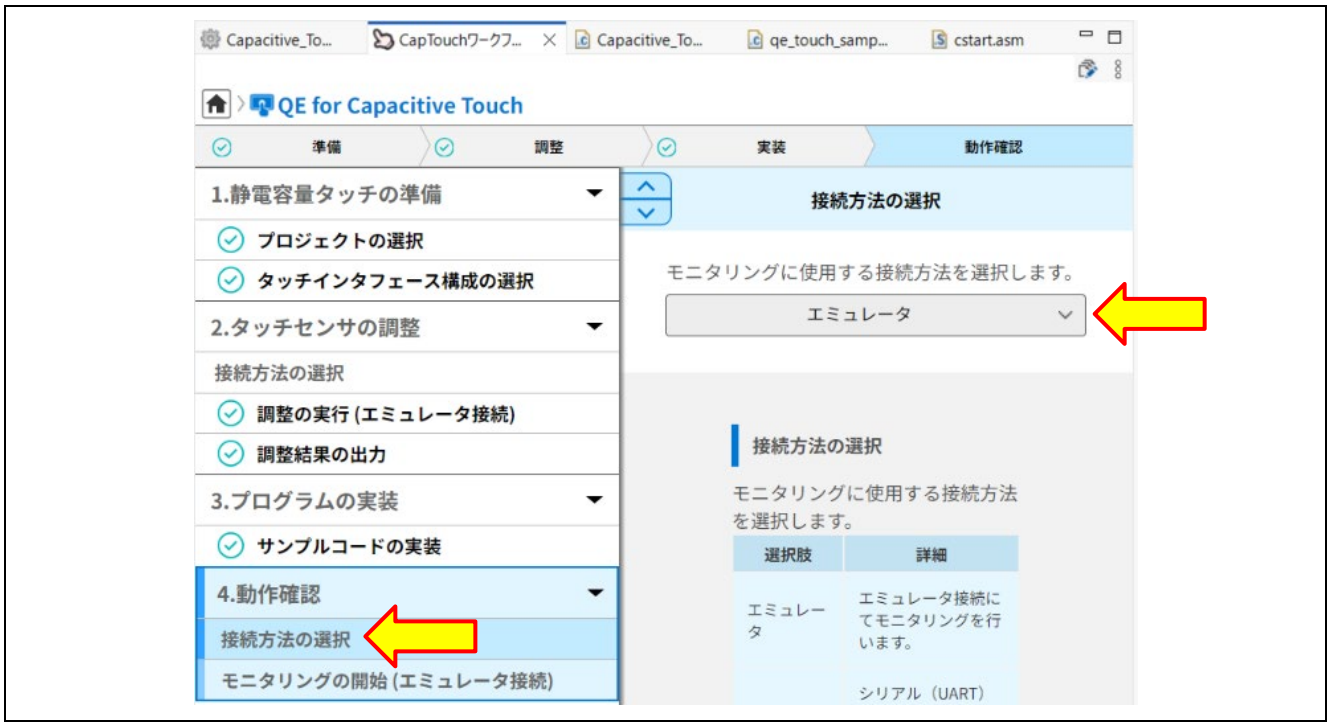

図 13-8 接続方法の選択(エミュレータ)

10. “**モニタリングの開始 (エミュレータ接続)**”をクリックし、[**ビューを開く**]を選択します。

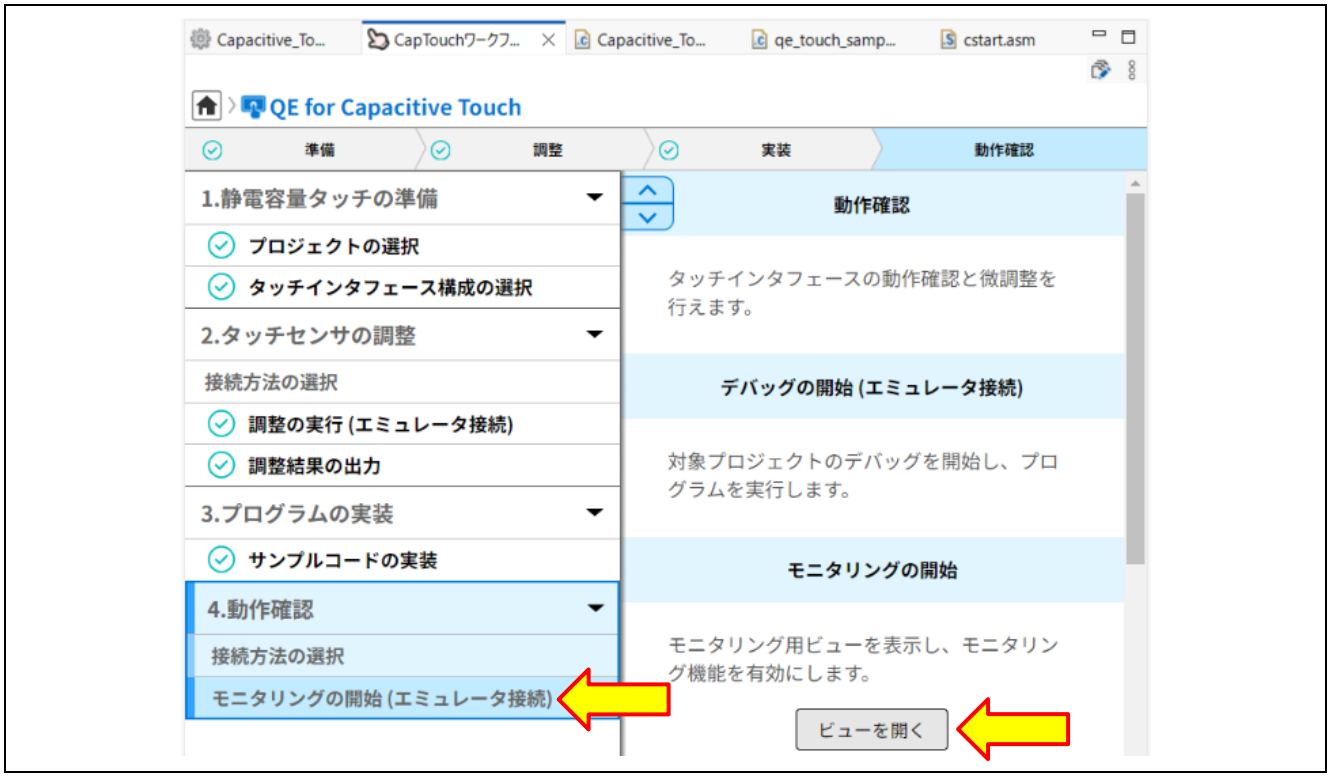

図 13-9 [ビューを開く]の選択(エミュレータ接続)

11. 見やすくするためウィンドウをドラッグする必要があるかもしれませんが、“**CapTouch ボード・モニタ (QE)**”は以下のように表示されます。

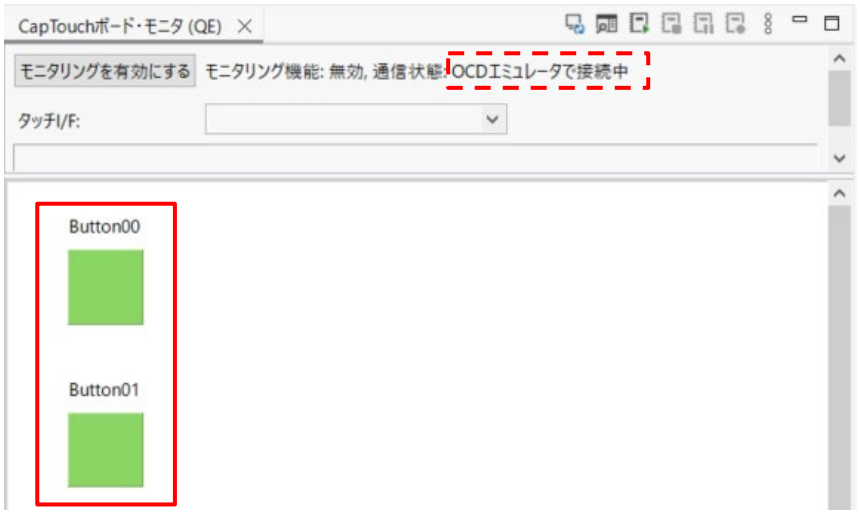

図 13-10 “CapTouch ボード・モニタ (QE)”画面(エミュレータ接続)

12. [**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。

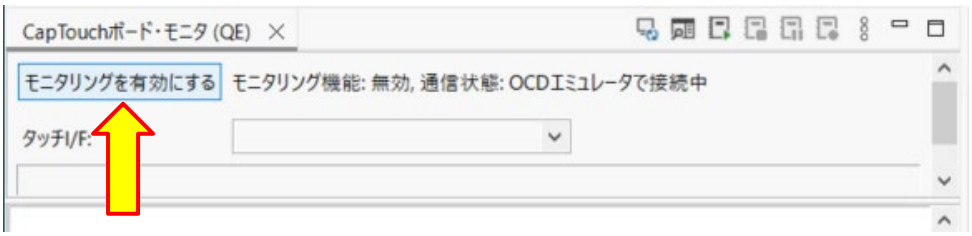

図 13-11 [モニタリングを有効にする]の選択(エミュレータ接続)

13. ターゲットボード上のボタン **TS06** をタッチします。“**CapTouch ボード・モニタ (QE)**”では、以下のように、タッチした状態をボタン上の指アイコンで表します。

図 13-12 タッチ状態の表示 (1)

14. タッチカウント値をグラフィカルに表示するには、“**CapTouch ステータス・チャート (QE)**”を起動します。

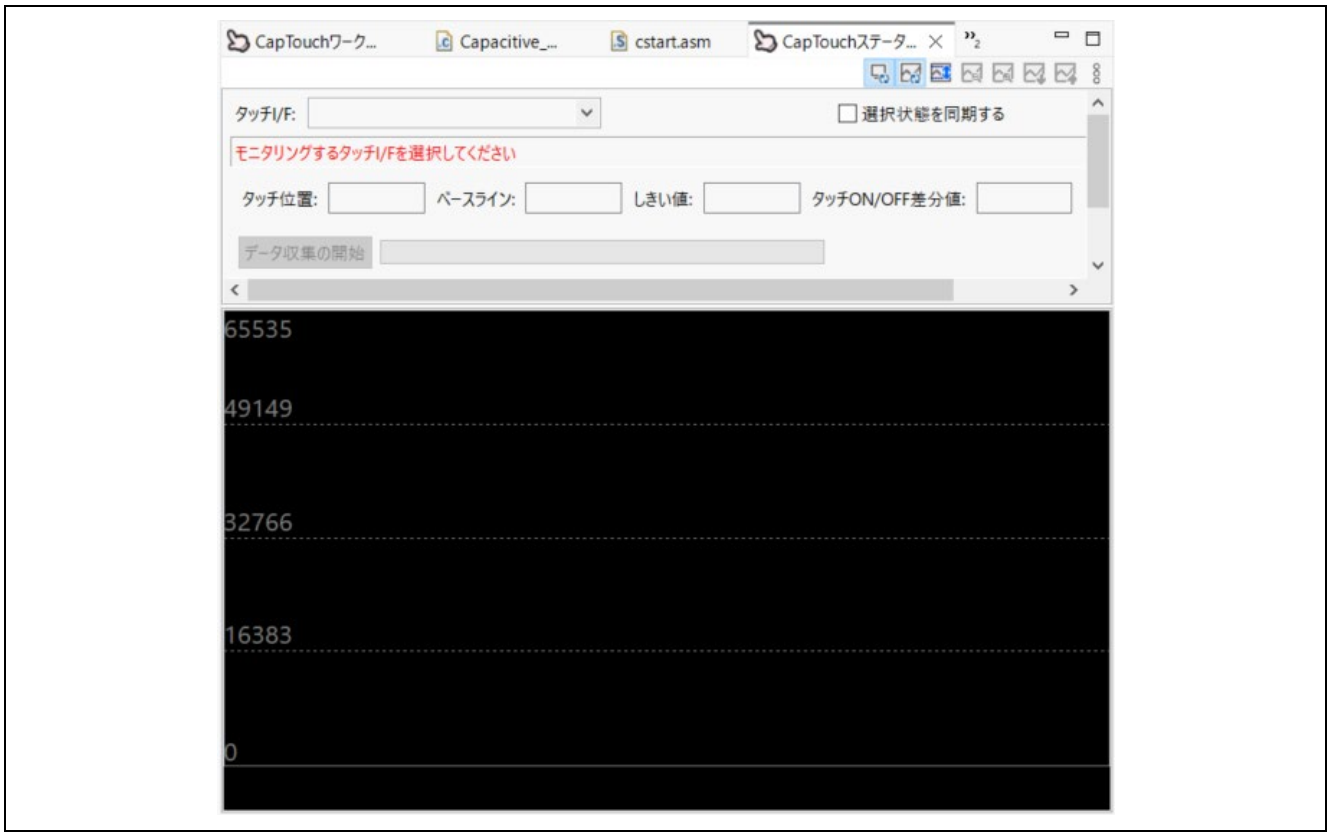

図 13-13 “CapTouch ステータス・チャート (QE)”画面 (1)

15. プルダウンメニューから“**Button00 @ config01**”を選択します。

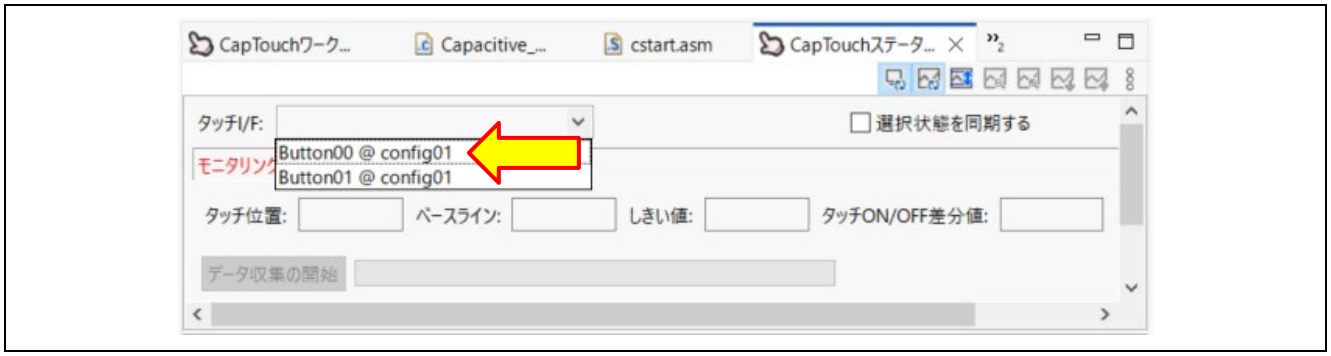

図 13-14 タッチ I/F の選択

16. グラフには実行中のタッチカウント値が表示されます。ボード上の **TS06** をタッチすると、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。緑のラインはしきい値を表し、**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しきい値を超えてタッチが検出されたことを表示しています。

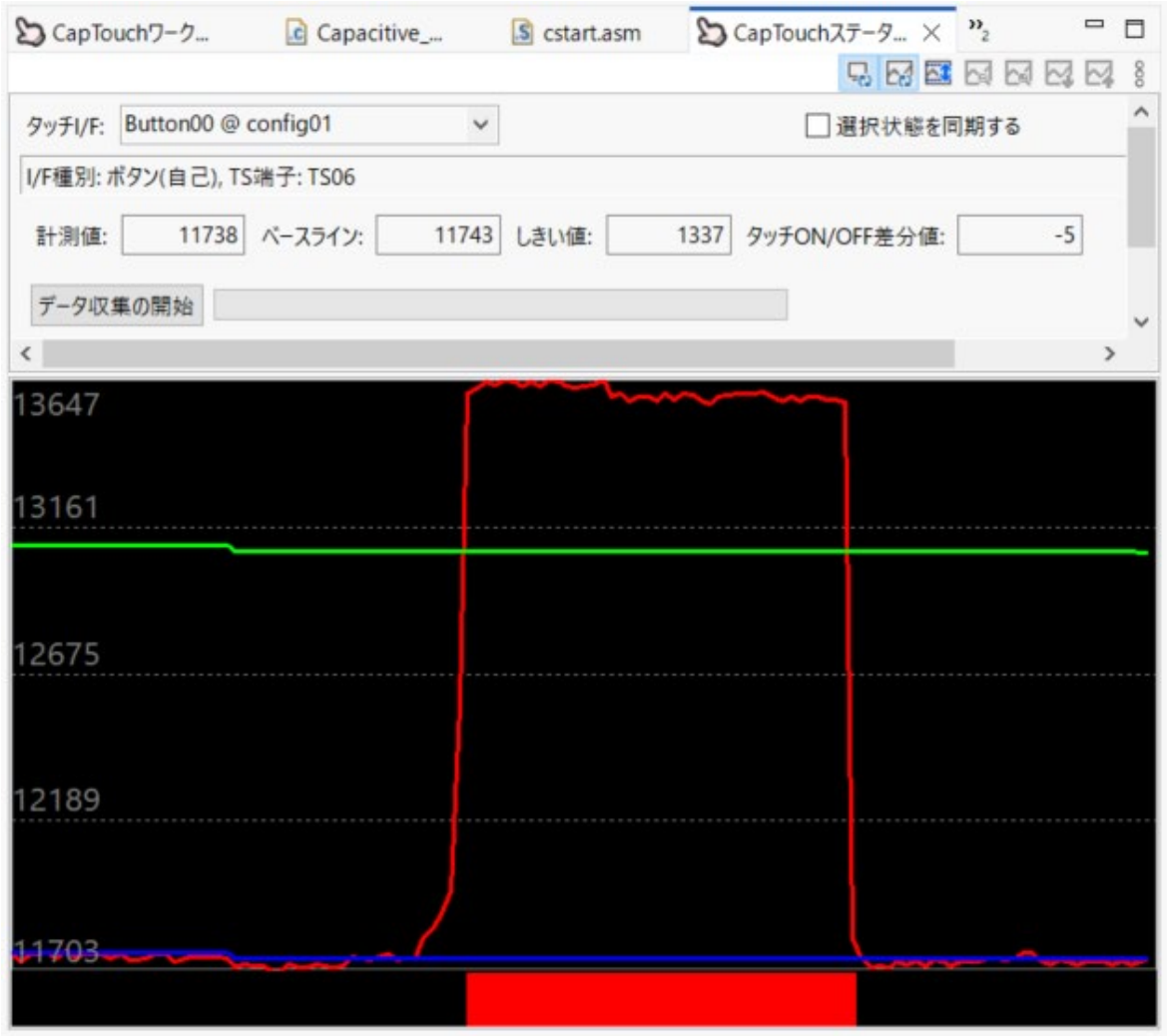

図 13-15 タッチカウント値のグラフ表示(1)

注.　必要に応じて、SNR 値を測定します。
手順 17～20 は、SNR 値を測定する際に設定する必要があります。

17. “**CapTouch ステータス・チャート (QE)**”の[**データ収集の開始**]をクリックします。

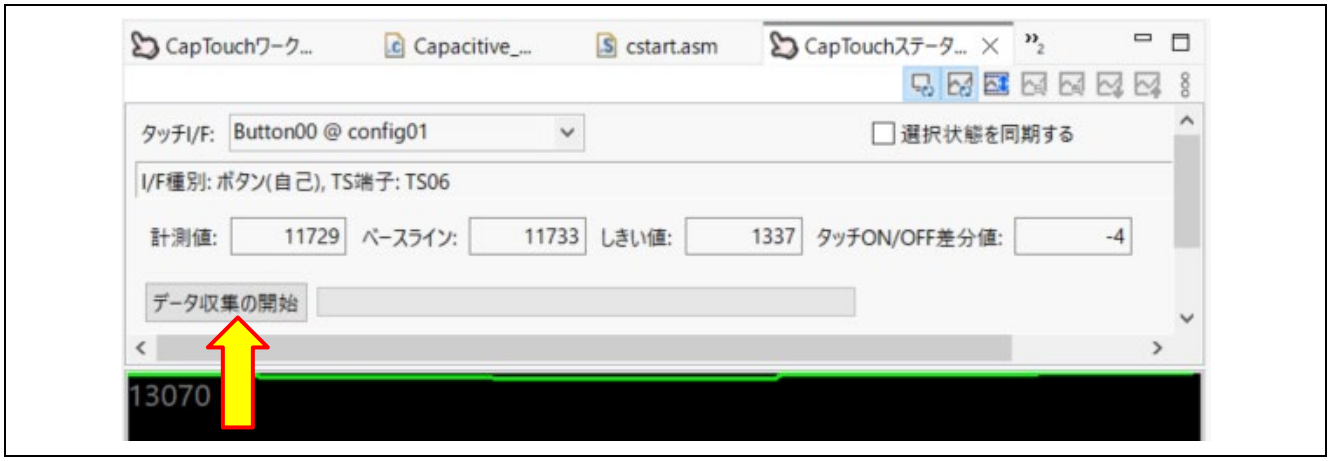

図 13-16 “データ収集の開始”ボタン

18. データ収集設定を行い、[**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。また、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。データの収集率は 100%となりタッチオフ状態の収集は完了します。

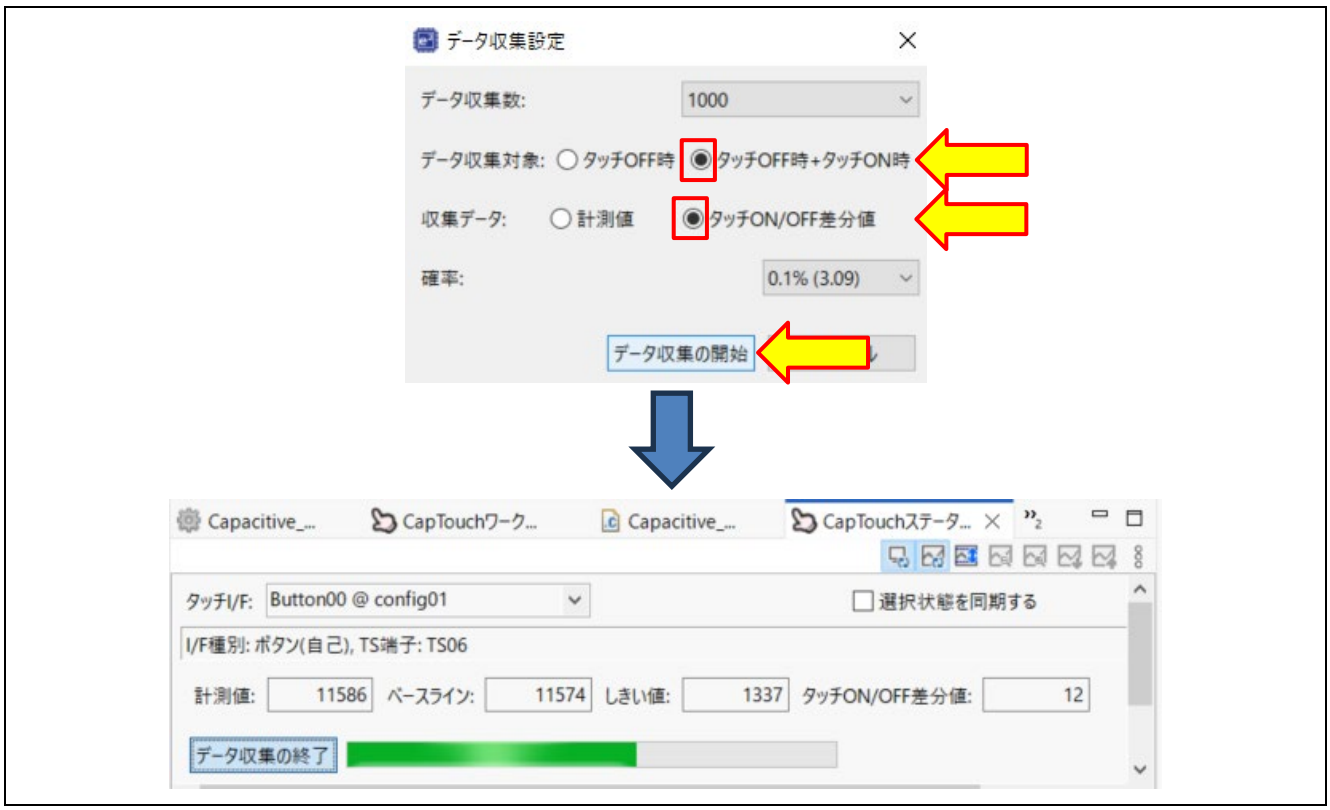

図 13-17　タッチ OFF 時のデータ収集の開始

19. 同様の手順でタッチ ON 時のデータを収集します。指が電極に触れていることを確認してから、”データ収集の開始”をクリックします。緑色のバーが右端まで達するとタッチオン状態の収集は完了します。

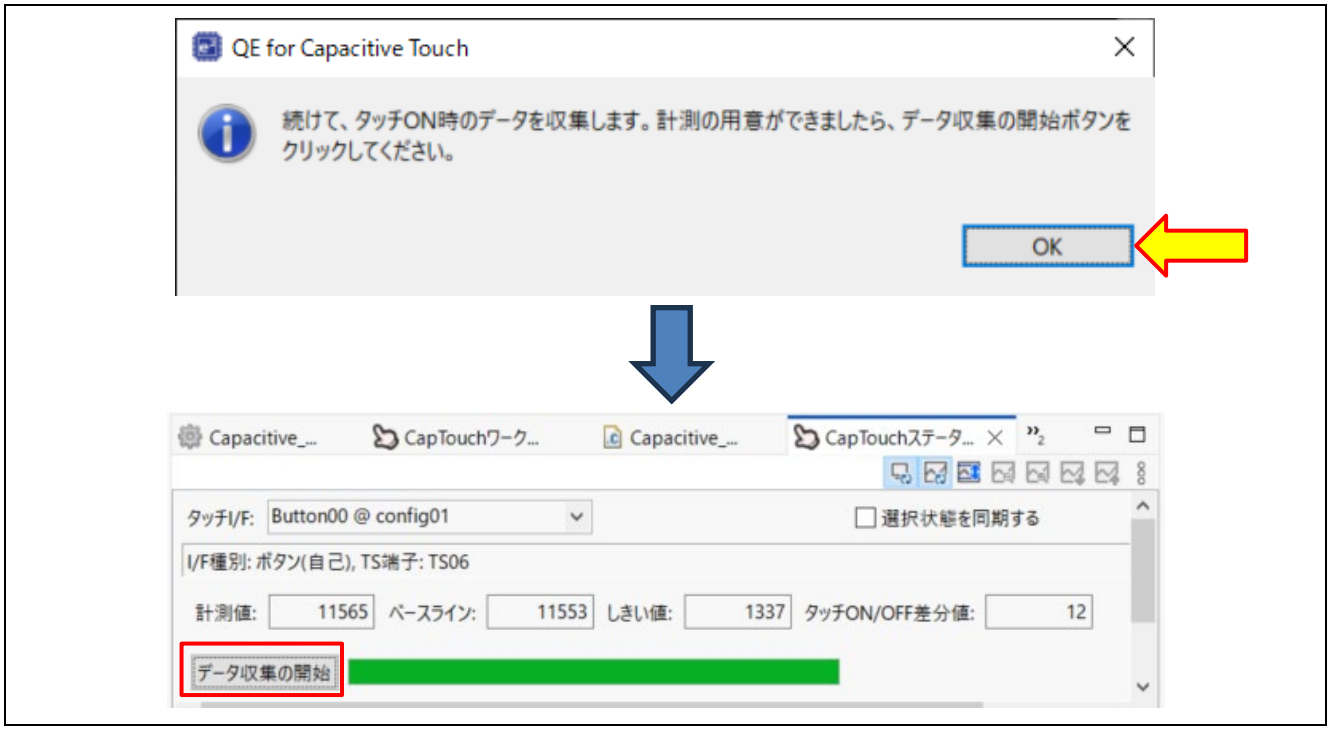

図 13-18 タッチ ON 時のデータ収集の開始

20. データ収集が完了すると SNR 値が表示されます。

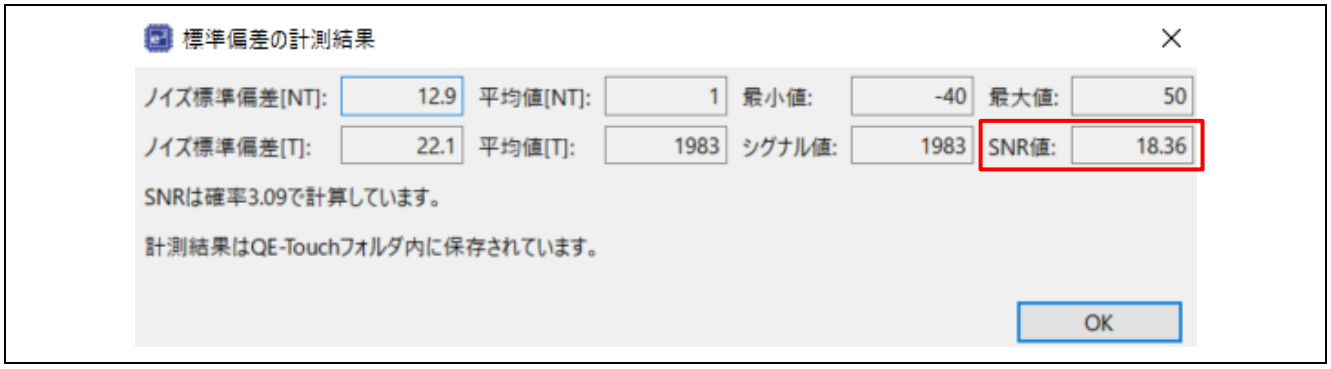

図 13-19 SNR 値

21. 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するには、“**CapTouch マルチ・ステータス・チャート (QE)**”を使用します。

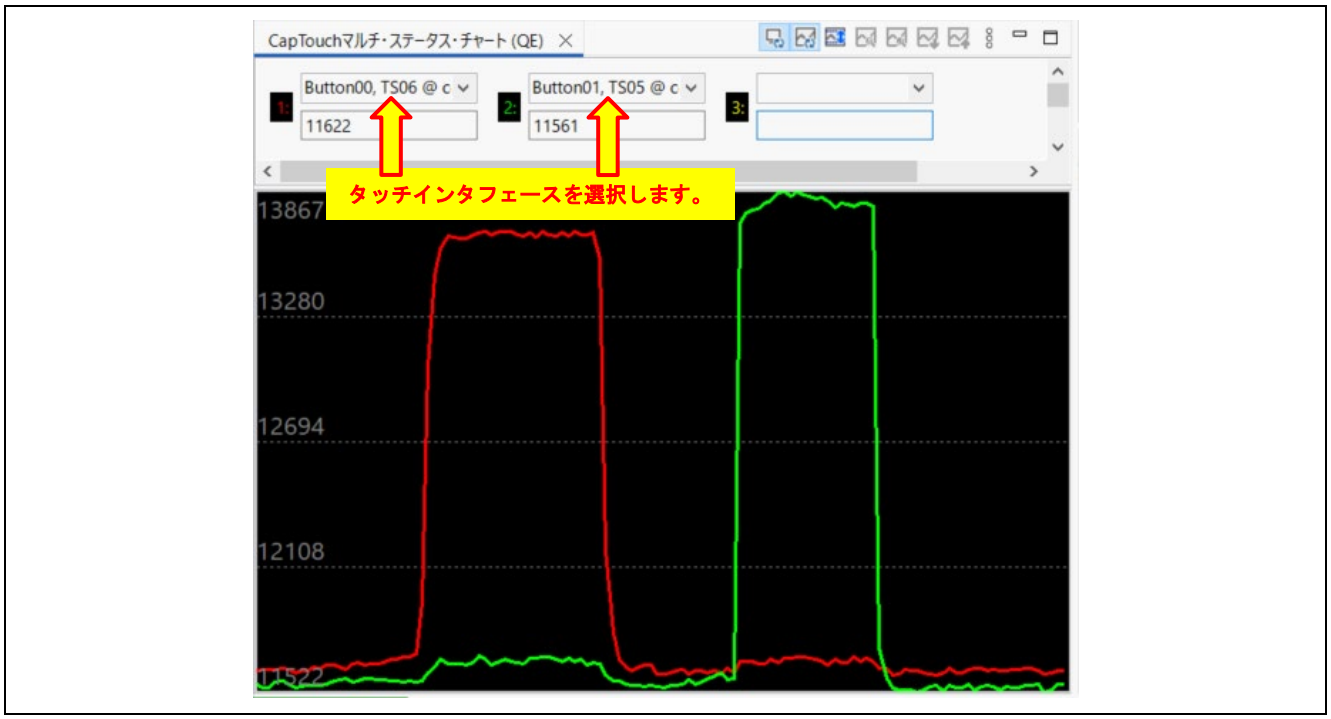

図 13-20 マルチ・ステータス・チャート

22. タッチパラメータを確認および調整する場合は、“**CapTouch パラメーター覧 (QE)**”を使用します。

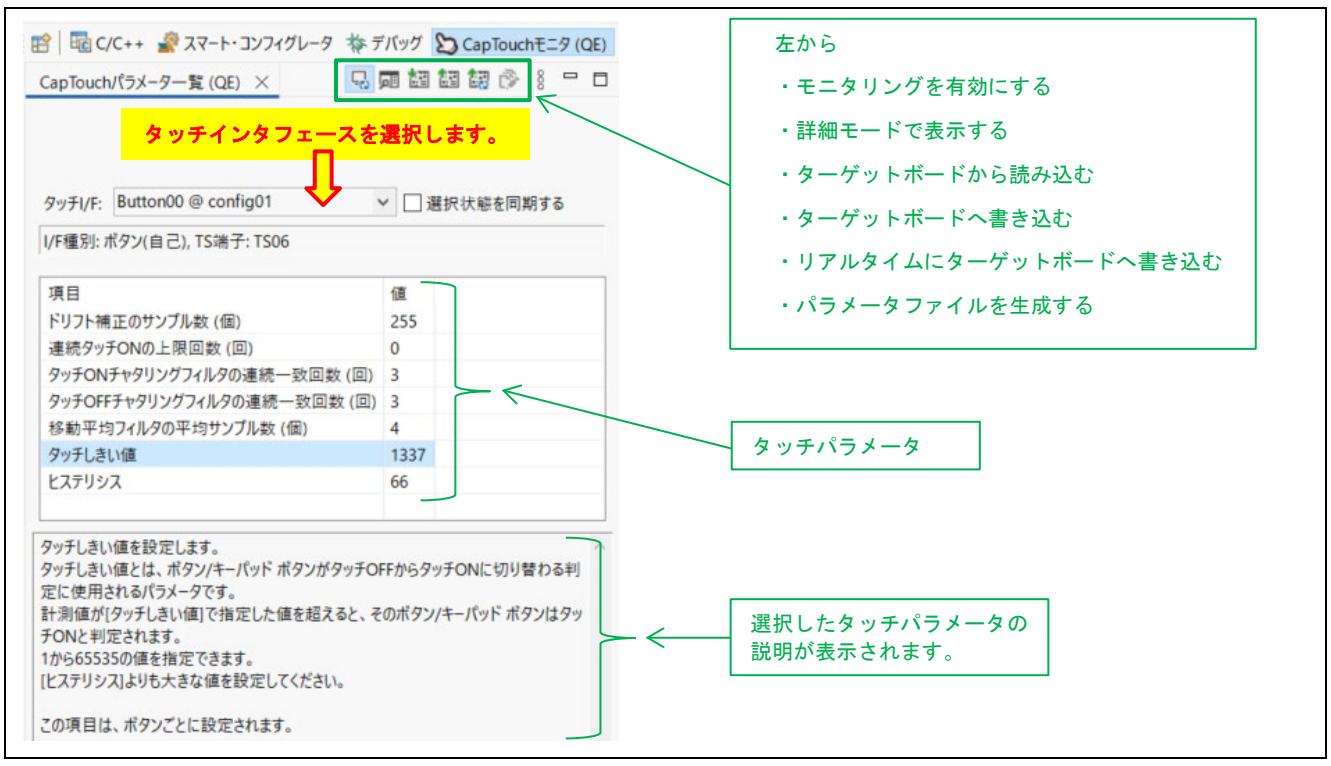

図 13-21 パラメータの調整

23. モニタリングを終了する場合は、[**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: 有効**”が“**モニタリング機能: 無効**”に切り替わります。

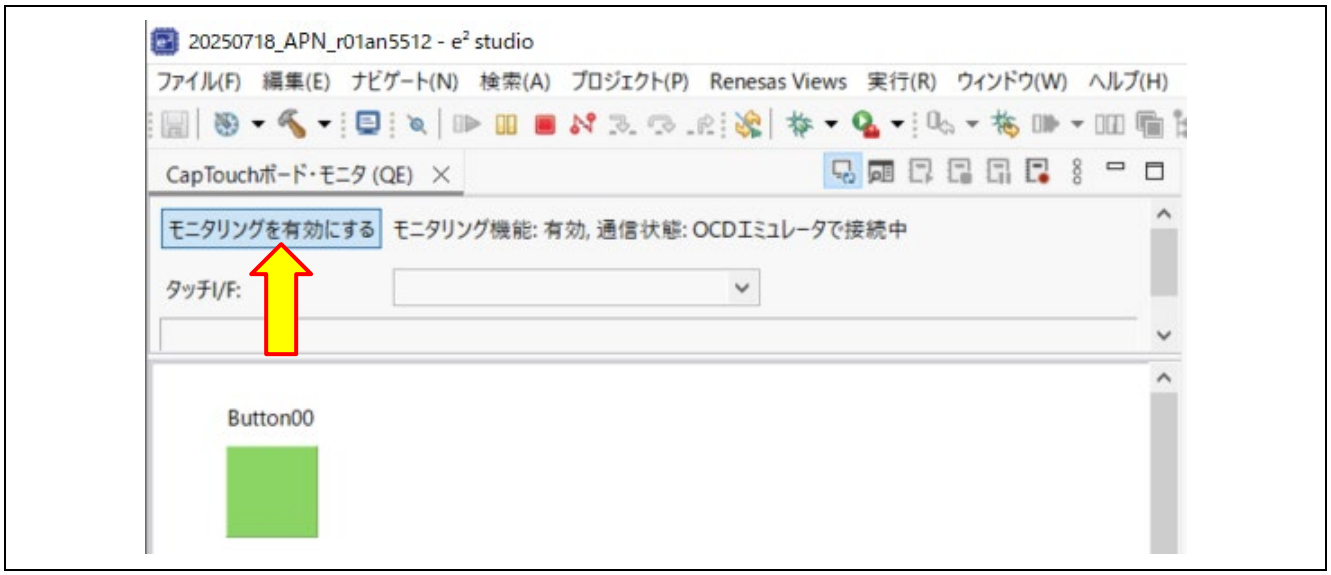

図 13-22 モニタリングを終了する(1)

24. デバッグセッションを終了する場合は、終了アイコンをクリックします。

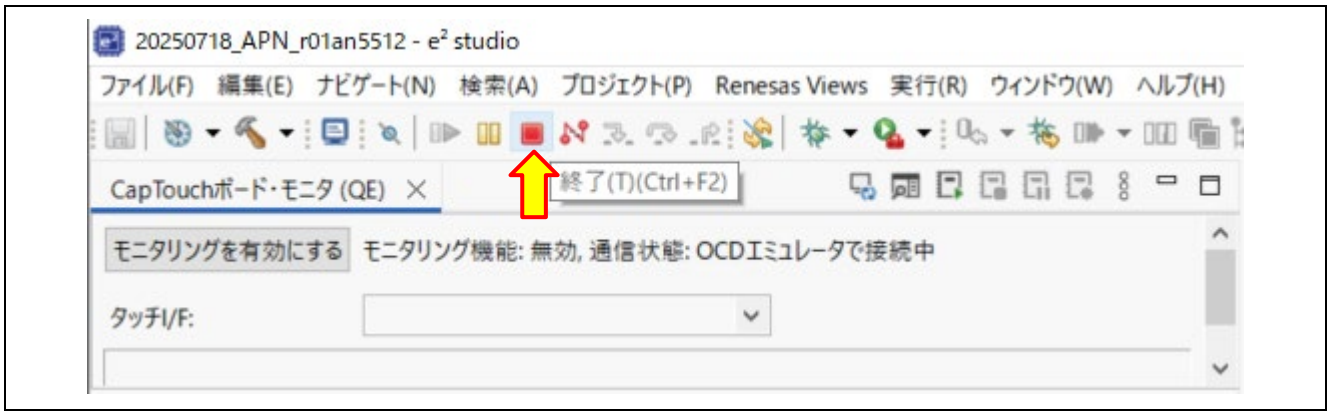

図 13-23 デバッグセッションを終了する

14. [追加機能] UART を使用したシリアル通信モニタの設定 (3/3)

注.　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。
また一方で、タッチ性能のモニタリングは、シリアル通信を介して行うこともできます。 したがって、スムーズにモニタリングを行いたい場合は、シリアル通信を介したモニタリング機能を追加してください(推奨設定)。

以下に示す 7 章、12 章および 14 章(本章を含む)では、UART を使用したシリアル通信モニタの設定について説明します。

- 7. [追加機能] UART を使用したシリアル通信モニタの設定 (1/3)
- 12. [追加機能] UART を使用したシリアル通信モニタの設定 (2/3)
- 14. [追加機能] UART を使用したシリアル通信モニタの設定 (3/3)

1. シリアル接続(UART/USB)を介してターゲットボードを PC に接続します。

2. ターゲットボードでタッチアプリケーションプログラムを実行します。

3. [**CapTouch ワークフロー (QE)**]パネルを開きます。
   プロジェクトフォルダ(**Capacitive_Touch_Project_Example**)および tifcfg ファイル(**Capacitive_Touch_Project_Example.tifcfg**)が図 14-1 のように設定されていることを確認します。

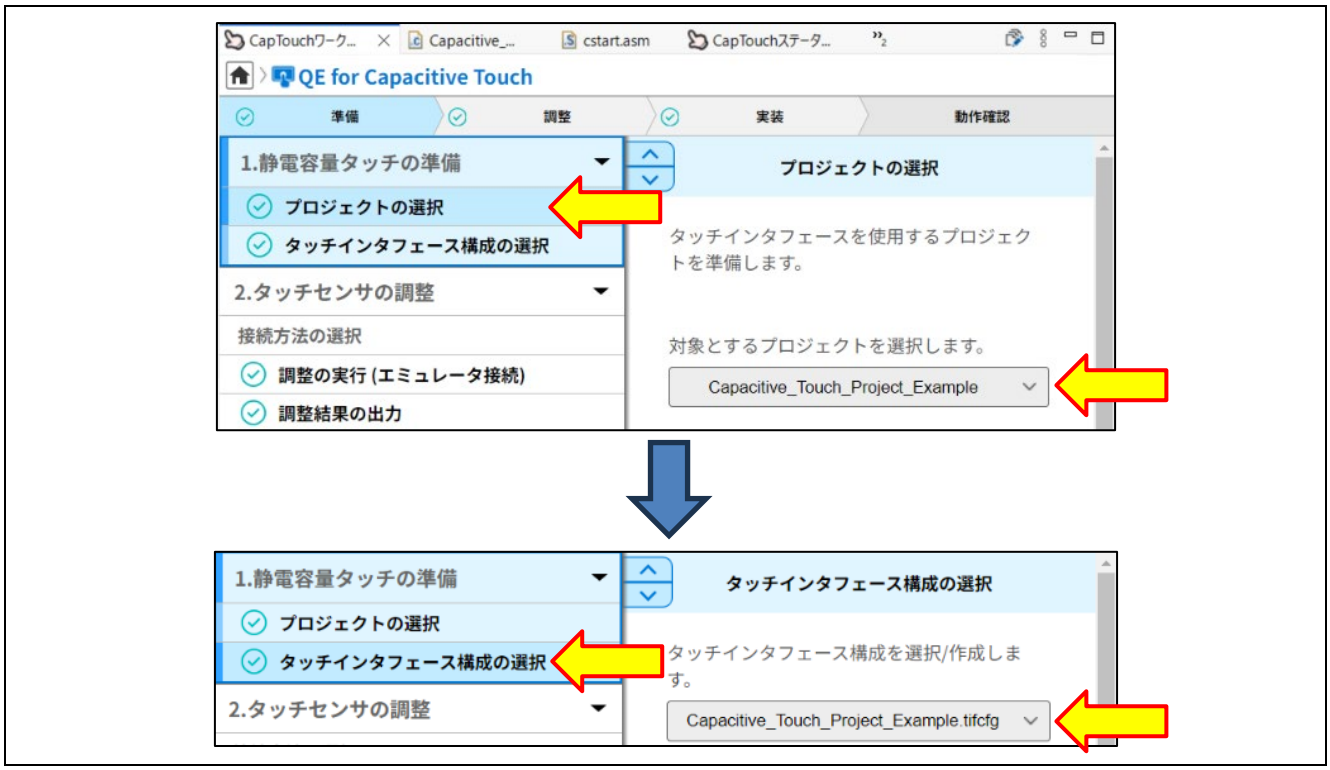

図 14-1 プロジェクトおよびタッチインタフェース構成の選択

4. 動作確認の“**接続方法の選択**”をクリックし、[**シリアル**]を選択します。

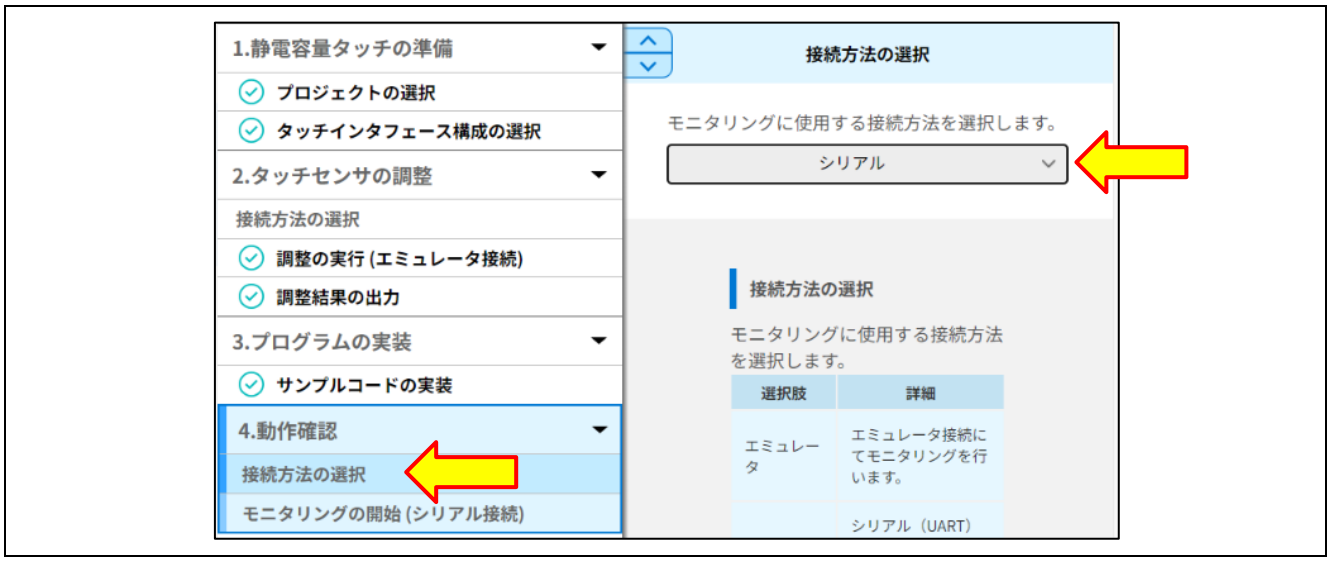

図 14-2 接続方法の選択(シリアル)

5. 動作確認の“**モニタリングの開始 (シリアル接続)**”をクリックし、“ボーレート”を“**153600**”(bps)に設定して[**接続する**]ボタンをクリックします。

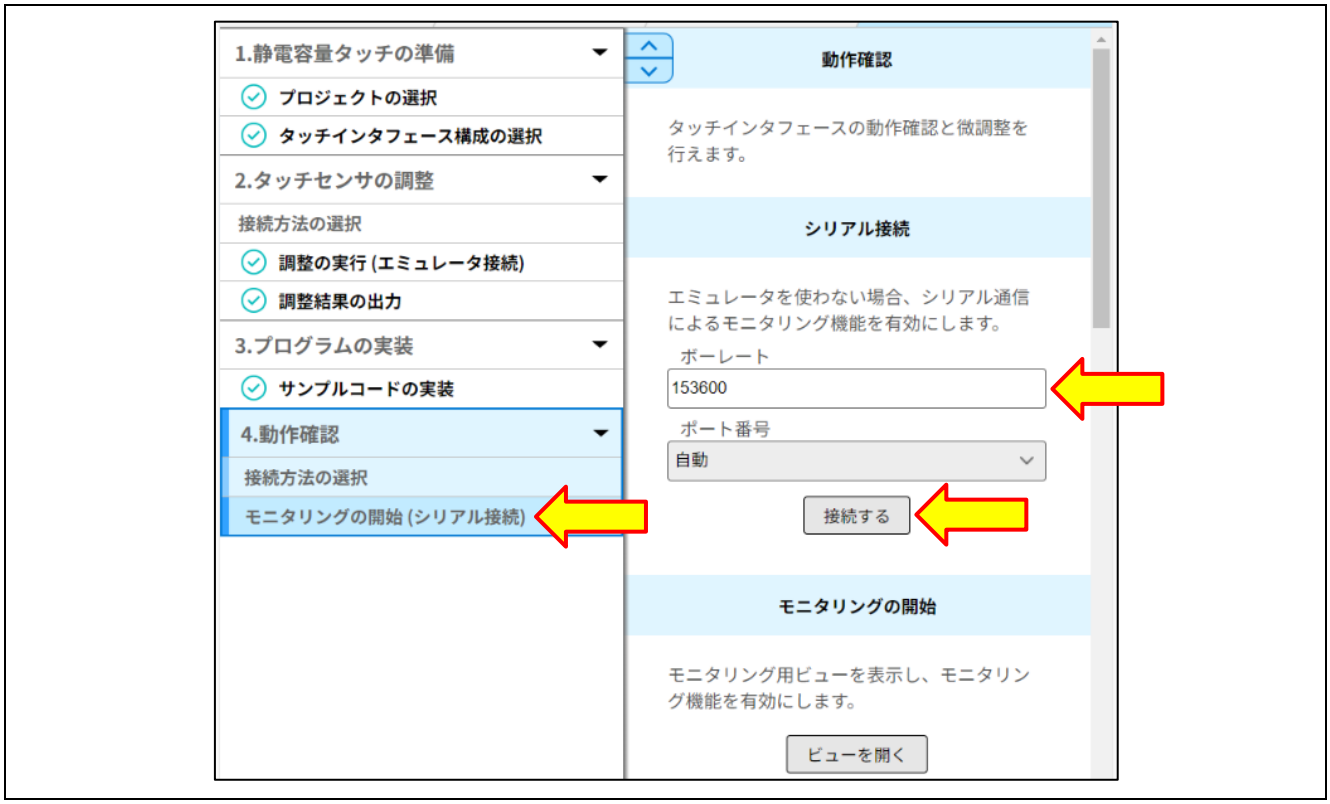

図 14-3 シリアル接続

注. ボーレート(bps)の設定値は、「7. [追加機能] UART を使用したシリアル通信モニタの設定 (1/3)」章の手順 4 で設定したボーレート(bps)を入力してください。

6. シリアル接続を実行すると、コンソール画面に以下のメッセージが表示されます。その後、[**ビューを開く**]を選択します。

図 14-4 [ビューを開く]の選択(シリアル接続)

7. “**CapTouch ボード・モニタ (QE)**”は以下のように表示されます。

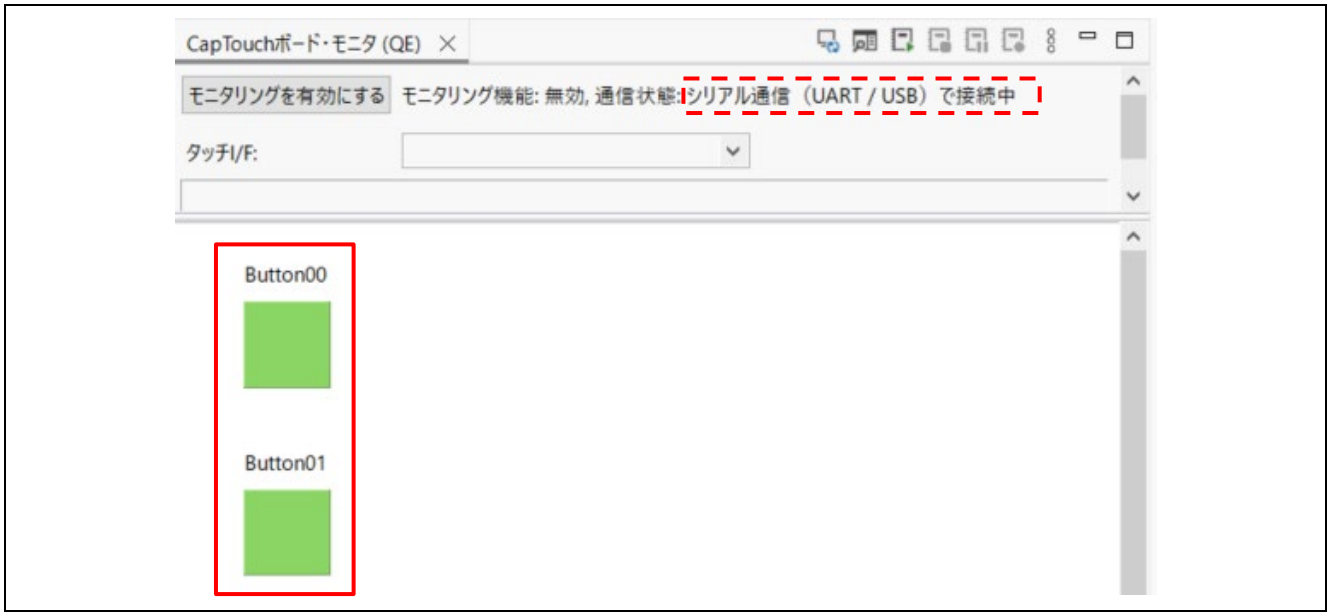

図 14-5 “CapTouch ボード・モニタ (QE)”画面(シリアル接続)

8. [**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。

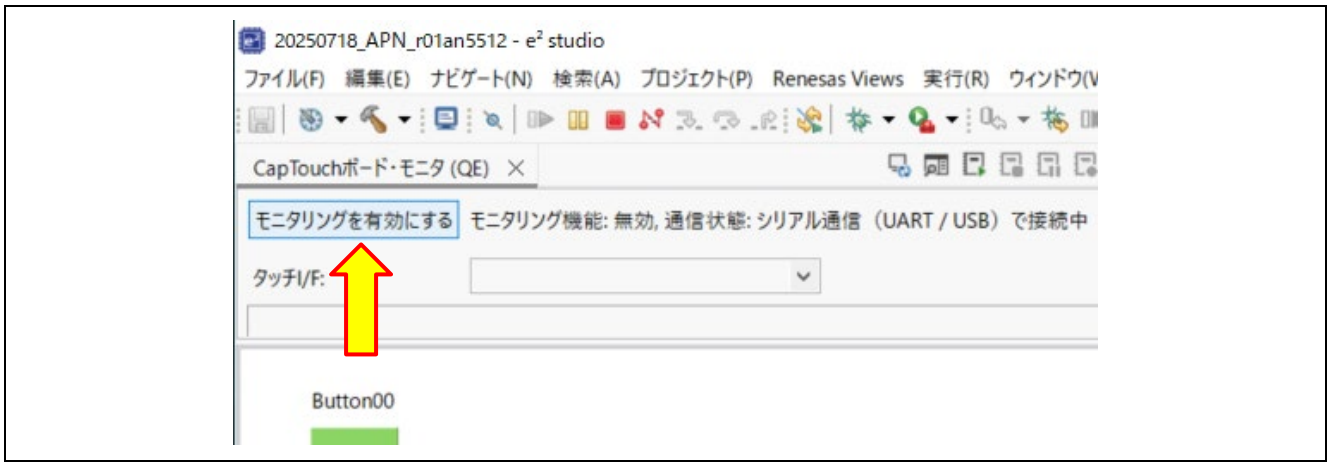

図 14-6 [モニタリングを有効にする]の選択(シリアル接続)

9. ターゲットボード上のボタン **TS06** をタッチします。“**CapTouch ボード・モニタ (QE)**”では、以下のように、タッチした状態をボタン上の指アイコンで表します。

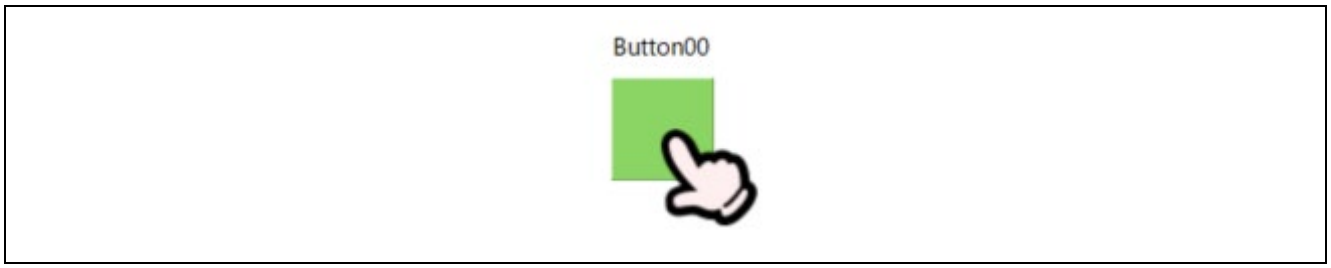

図 14-7 タッチ状態の表示 (2)

10. タッチカウント値をグラフィカルに表示するには、“**CapTouch ステータス・チャート (QE)**”を起動します。プルダウンメニューから“**Button00 @ config01**”を選択します。

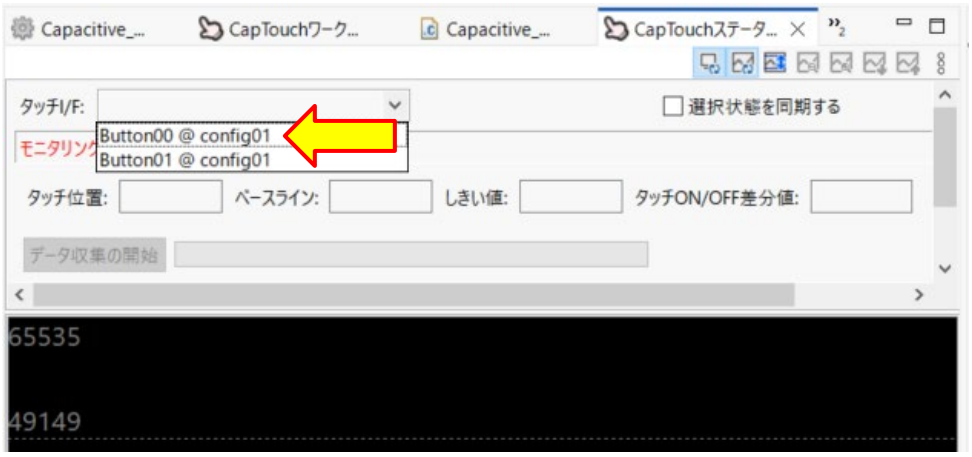

図 14-8 “CapTouch ステータス・チャート (QE)”画面 (2)

11. グラフには実行中のタッチカウント値が表示されます。ボード上の **TS06** をタッチすると、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。

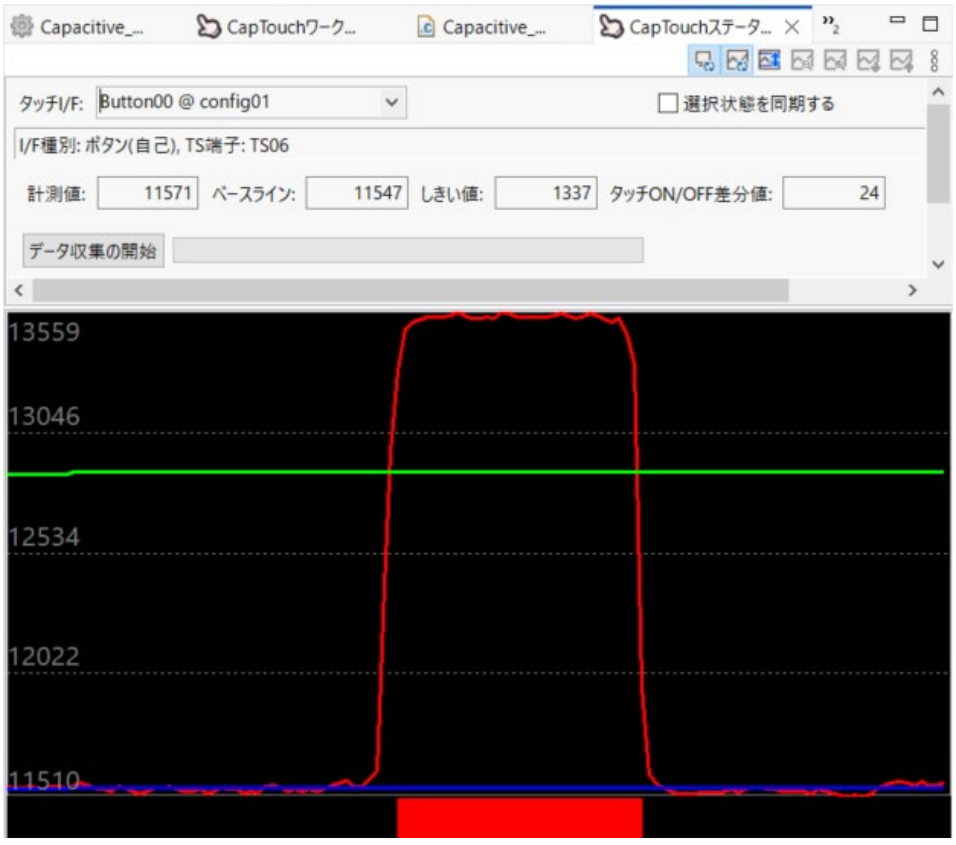

図 14-9 タッチカウント値のグラフ表示(2)

12. モニタリングを終了する場合は、[**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: 有効**”が“**モニタリング機能: 無効**”に切り替わります。

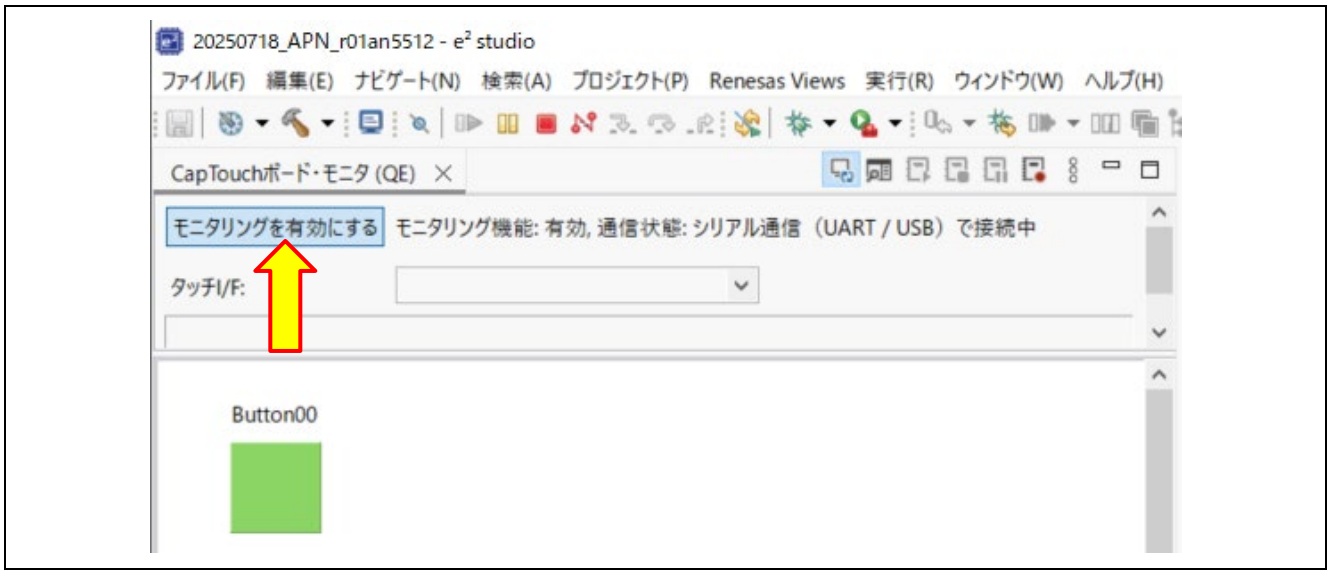

図 14-10 モニタリングを終了する(2)

13. シリアル接続を終了する場合は、[**切断する**]ボタンをクリックします。

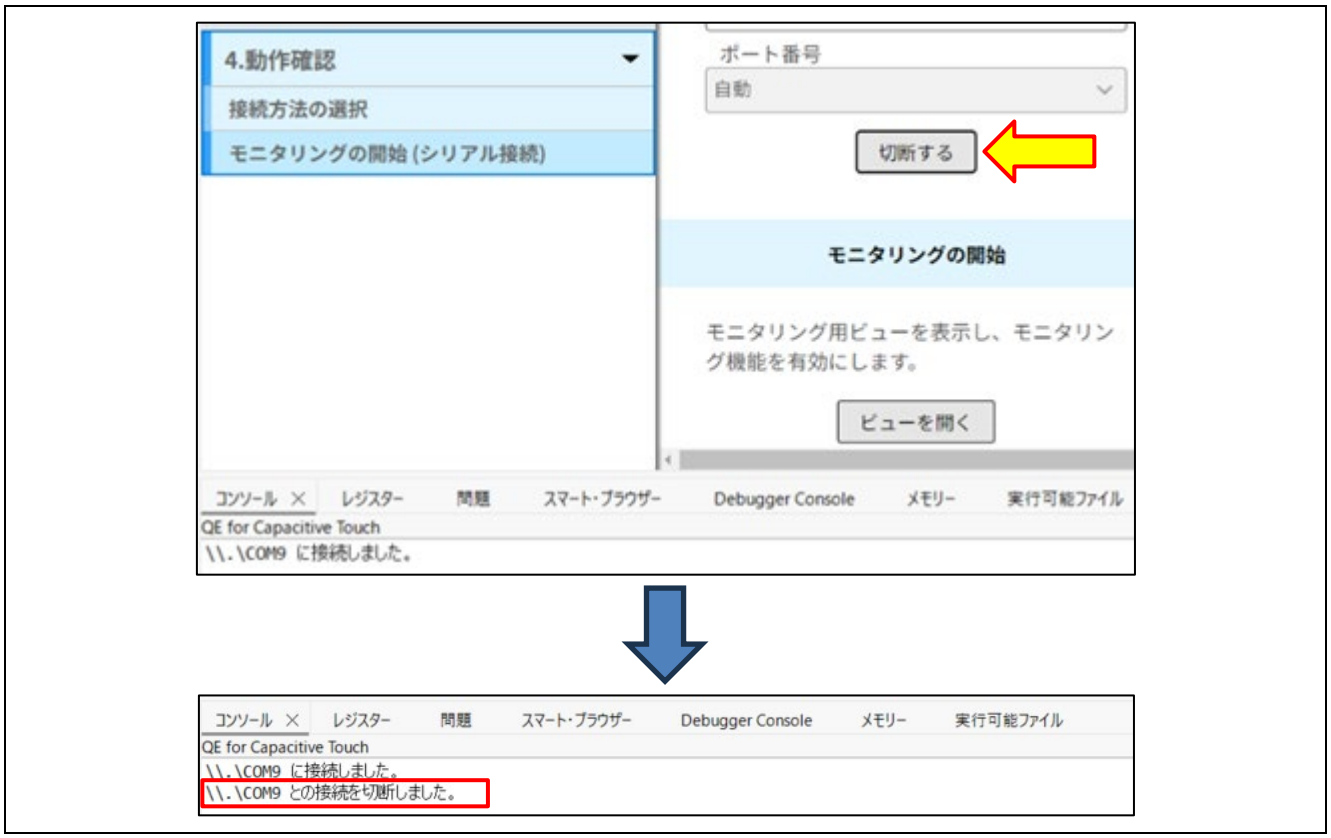

図 14-11 [切断する]ボタンの選択(シリアル接続)

15. qe_touch_sample.c (ソフトウェアタイマ使用例)

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

```
/***********************************************************************************************************************
* Copyright (c) 2020 - 2025 Renesas Electronics Corporation and/or its affiliates
*
* SPDX-License-Identifier: BSD-3-Clause
***********************************************************************************************************************/
/***********************************************************************************************************************
* File Name    : qe_touch_sample.c
* Description  : Main Program for RL78
***********************************************************************************************************************/
#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, NULL, 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();
    }
}
```

16. [応用例] ハードウェアタイマでのタッチ計測

本章では、ハードウェアタイマを使用したタッチ計測周期の実装例を説明します。本アプリケーション例では、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ます。また、動作確認のため、タッチセンサ (ボタン) でのタッチ判定結果に応じてターゲットボード上の LED を点灯あるいは消灯させます。本アプリケーション例では、タッチセンサ① (TS_B1: Button00/TS06) または、タッチセンサ② (TS_B2: Button01/TS05)に指が触れ、タッチ判定結果が ON になると、LED1 が点灯します。

アプリケーションノート本編の「6. 静電容量タッチアプリケーション開発の開始～モジュール追加」～「15. qe_touch_sample.c (ソフトウェアタイマ使用例)」の内容に加えて、以下の設定を行ってください。

16.1 スマート・コンフィグレータの設定 (ハードウェアタイマ)

1. スマート・コンフィグレータの [クロック] タブを選択し、タッチ計測周期に使用するクロックを以下のように設定します。

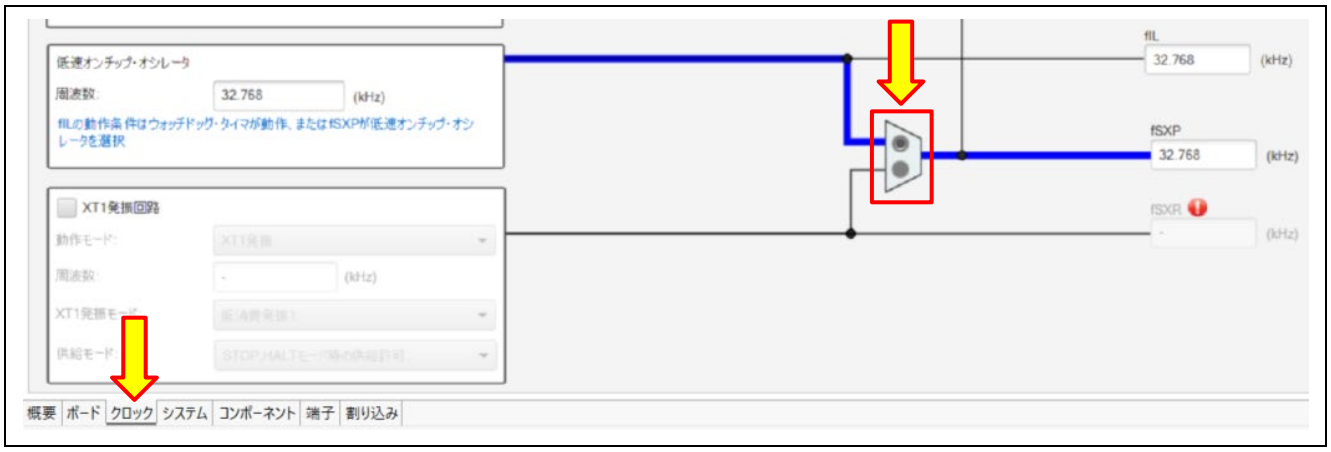

図 16-1 クロック設定(ハードウェアタイマ用)

2. ソフトウェアコンポーネント設定で“**インターバル・タイマ**”モジュールを選択し、ダイアログ下部の[**次へ**]をクリックします。

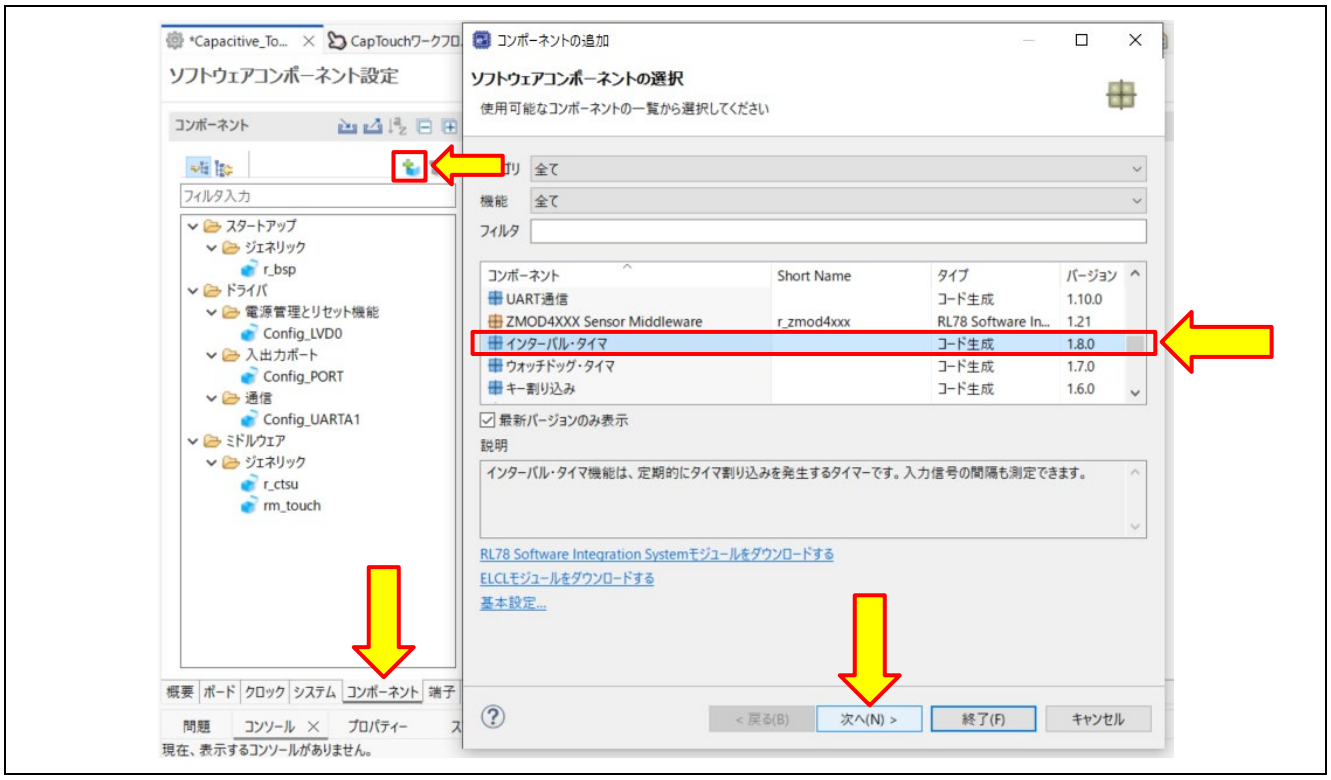

図 16-2 “インターバル・タイマ”モジュールの選択

3. インターバル・タイマの構成を以下のように設定し、ダイアログ下部の[**終了**]をクリックします。

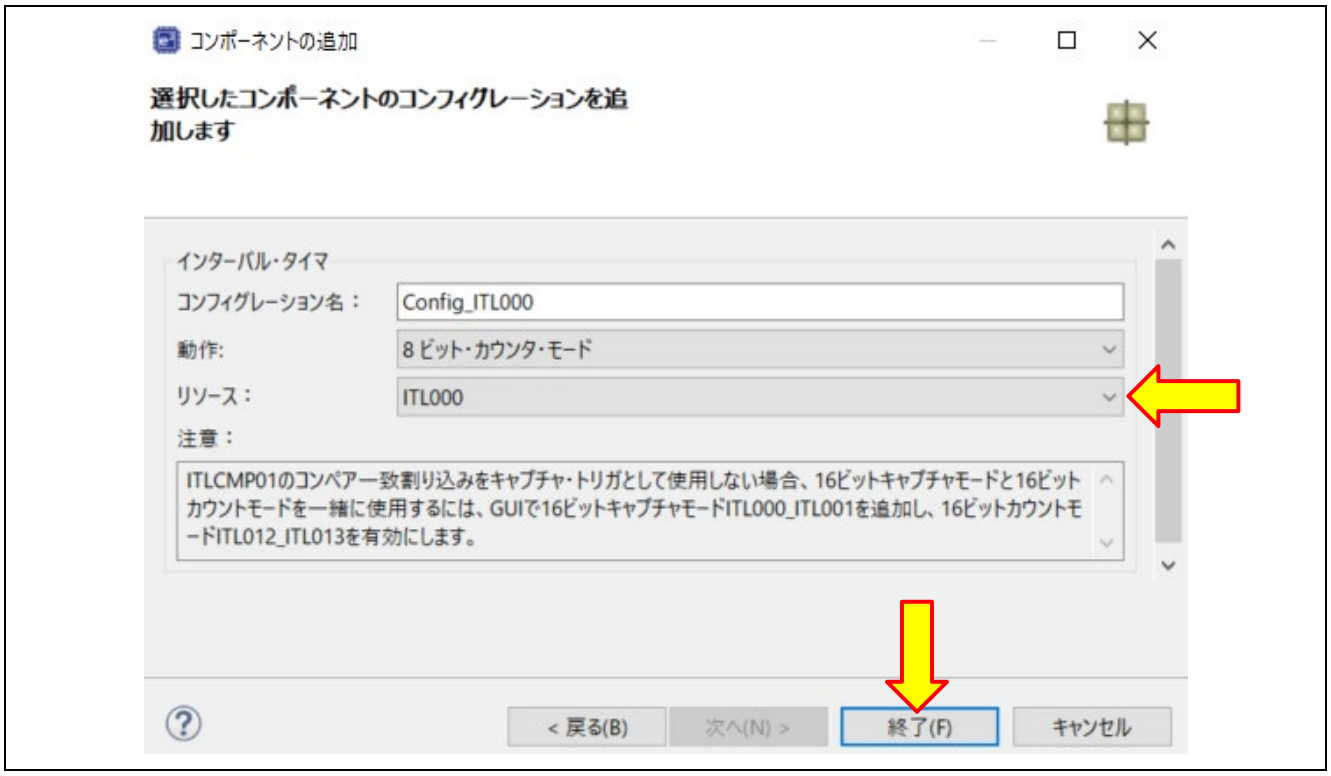

図 16-3 “インターバル・タイマ”モジュールのリソース設定

4. インターバル・タイマを以下のように設定します。

図 16-4 Config_ITL000 の設定

5. 動作確認のための LED 点灯/消灯に使用するポートを以下のように設定します。

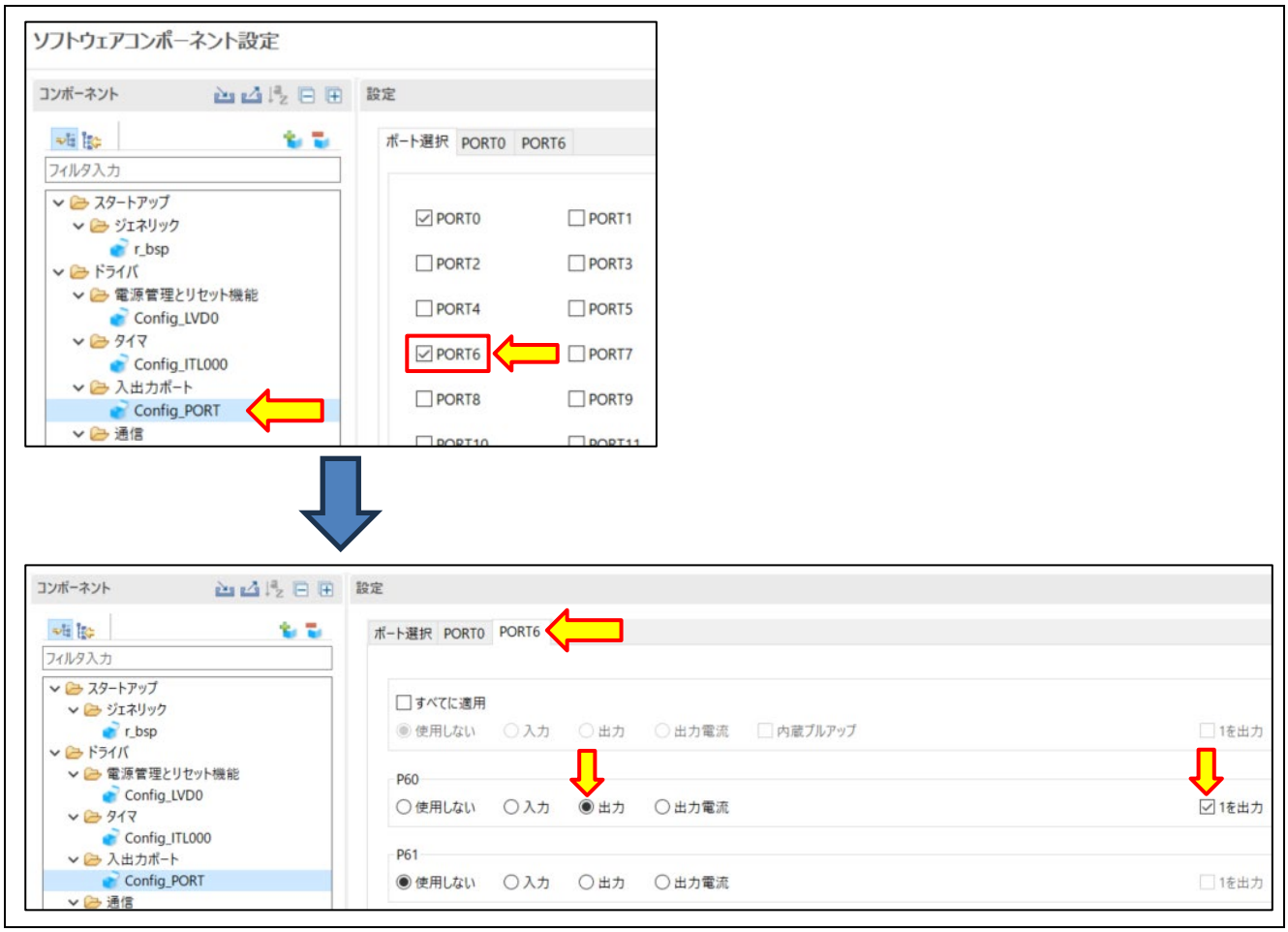

図 16-5 PORT コンポーネント設定(LED 点灯用)

6. スマート・コンフィグレータの右上の“コードの生成”アイコン をクリックして、プロジェクトに必要なコンポーネントのコードを追加します。

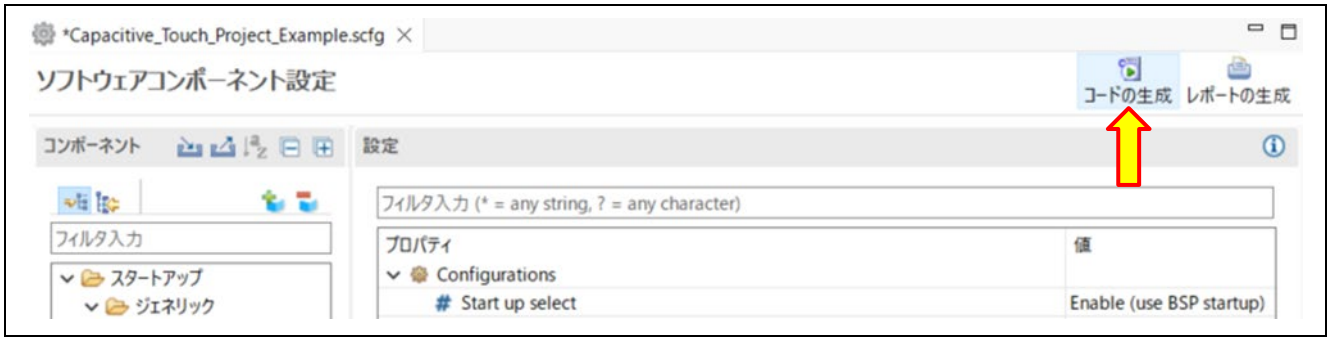

図 16-6 “コードの生成”アイコンの選択 (3)

7. プログラムの実装例は、「16.3 qe_touch_sample.c (ハードウェアタイマ使用例)」を参照してください。

16.2 フローチャート(ハードウェアタイマ使用例)

図 16-7 および図 16-8 にハードウェアタイマを使用したタッチ計測制御処理のフローチャートを示します。

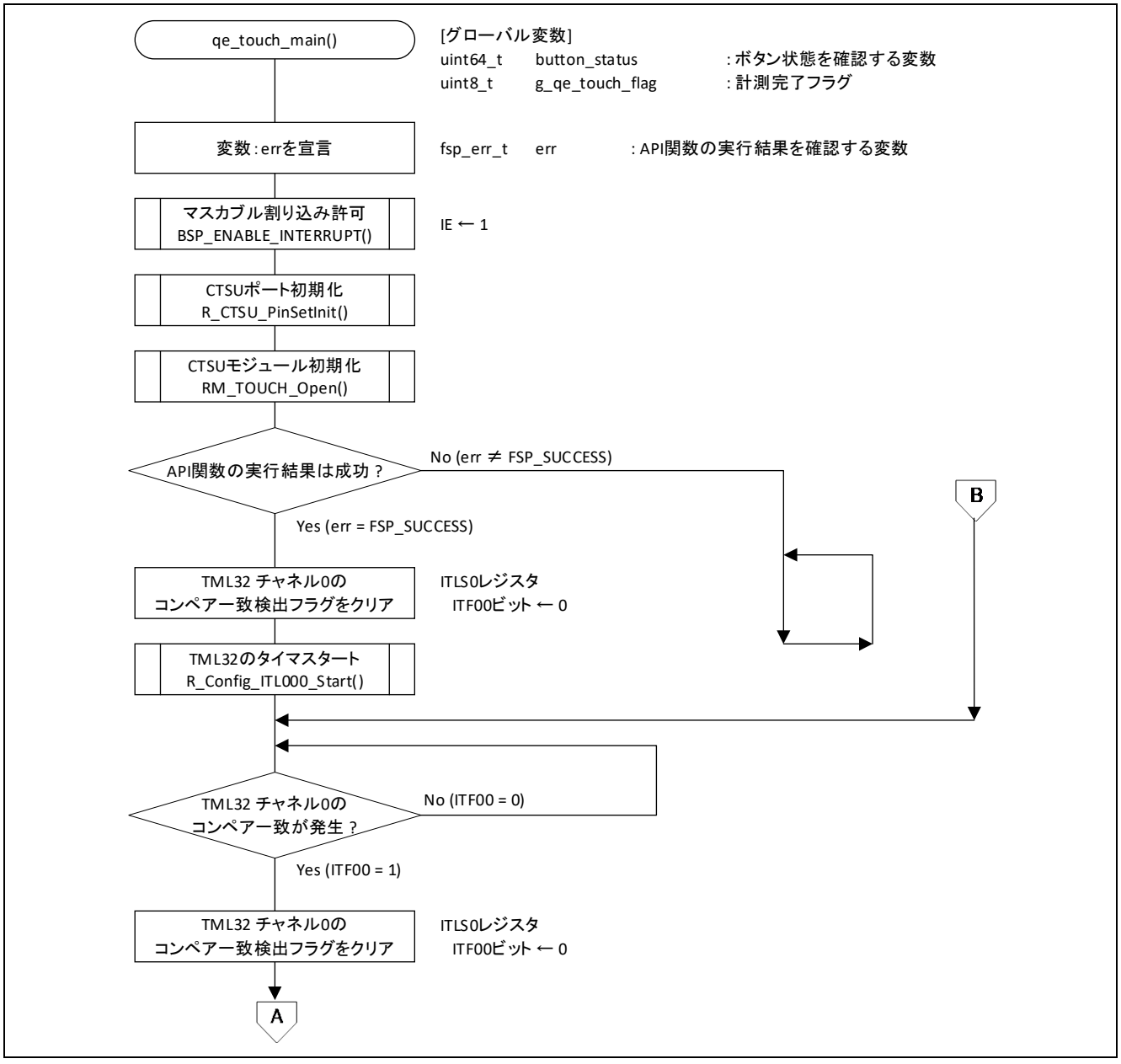

図 16-7 ハードウェアタイマを使用したタッチ計測制御処理(1/2)

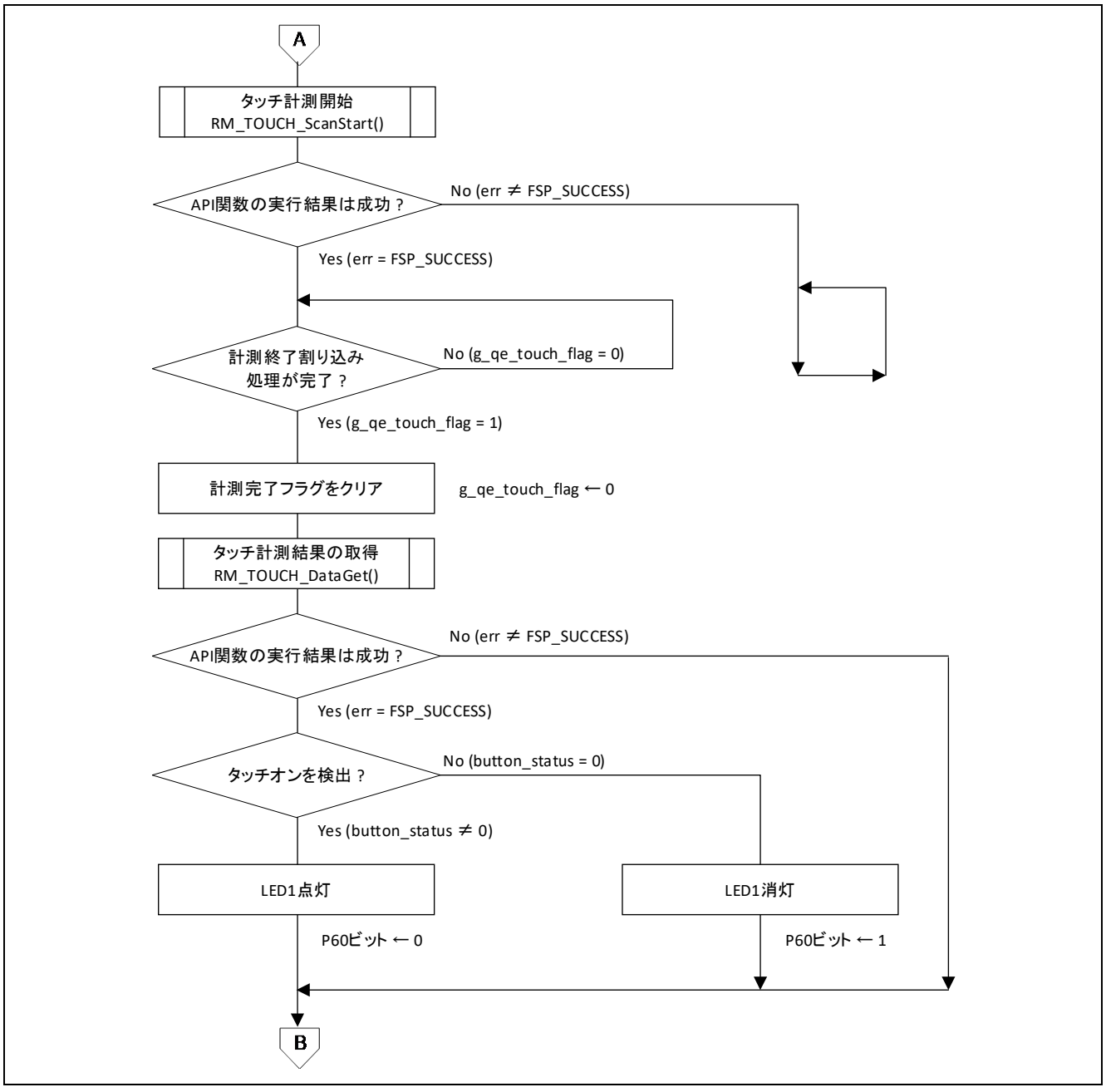

図 16-8 ハードウェアタイマを使用したタッチ計測制御処理(2/2)

16.3 qe_touch_sample.c (ハードウェアタイマ使用例)

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を以下に示します。

QE で自動生成されるコードに以下の赤文字で示した新規コードを追加しています。

```
/***********************************************************************************************************************
* Copyright (c) 2020 - 2025 Renesas Electronics Corporation and/or its affiliates
*
* SPDX-License-Identifier: BSD-3-Clause
***********************************************************************************************************************/
/***********************************************************************************************************************
* File Name    : qe_touch_sample.c
* Description  : Main Program for RL78
***********************************************************************************************************************/
#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, NULL, NULL);
        if (FSP_SUCCESS == err)
        {

            /* TODO: Add your own code here. */
            if (0 != button_status)
            {
                P6_bit.no0 = 0;
            }
            else
            {
                P6_bit.no0 = 1;
            }
        }
    }
}
```

17. 参考ドキュメント

〇ユーザーズマニュアル

- RL78/Gxx ユーザーズマニュアル ハードウェア編
  — RL78/G23 ユーザーズマニュアル ハードウェア編 (R01UH0896)
- RL78 ファミリ ユーザーズマニュアル ソフトウェア編 (R01US0015)
  (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。)

〇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。)

〇ユーザーズマニュアル：開発環境

- 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。)

〇アプリケーションノート

- 静電容量センサマイコン 静電容量タッチ導入ガイド (R30AN0424)
- RL78 ファミリ CTSU モジュール Software Integration System (R11AN0484)
- RL78 ファミリ TOUCH モジュール Software Integration System (R11AN0485)

- 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)

- 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)
- 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 (R01AN6741)

- RL78/G23 静電容量タッチ低消費電力ガイド(SNOOZE モード機能) (R01AN5886)
- RL78/G22 静電容量タッチ低消費電力ガイド(SNOOZE 機能) (R01AN7413)

- RL78 ファミリ 静電容量タッチ低消費電力アプリケーション(SMS 使用)の開発 (R01AN7261)
- RL78/G23 静電容量タッチ低消費電力ガイド(SMS 機能) (R01AN6670)
- RL78/G22 静電容量タッチ低消費電力ガイド(SMS/MEC 機能) (R01AN6847)
  (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。)

ホームページとサポート窓口

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

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

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

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2021.4.30	-	初版発行
2.00	2021.8.31	-	開発環境を更新
2.10	2022.5.20	-	「7. [追加機能] UART を使用したシリアル通信モニタの設定(1/3)」章を更新。 「14. [追加機能] UART を使用したシリアル通信モニタの設定(3/3)」章を更新。 「16. [応用例] ハードウェアタイマでのタッチ計測」章を追加。
3.00	2023.5.22	-	「18. [応用例] 自動判定機能(SMS 使用)と MEC 機能の設定」章を追加。
4.00	2025.8.27	-	開発環境を更新
		-	最新の開発環境に合わせてアプリケーションノート全体の章構成、説明文章および図表を更新 (タッチアプリケーションの開発方法の流れとしては、以前のバージョンのアプリケーションノートと同様)。
		1	要旨を更新
		-	「1. 概要」章～「3. 動作確認環境」章に、本アプリケーションノートでタッチアプリケーションを開発するための条件に関して、より詳細な情報を追加。
		-	Rev. 3.00 で記載していた「18. [応用例] 自動判定機能(SMS 使用)と MEC 機能の設定」章を削除。
		83	「17. 参考ドキュメント」章を更新。

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。

© 2025 Renesas Electronics Corporation. All rights reserved.