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

要旨

本アプリケーションノートでは、RL78 MCU の SNOOZE モード・シーケンサ (SMS) を使用した静電容量タッチ低消費電力アプリケーションの作成に必要な手順を説明します。

SMS を使用した自動判定計測を行うことで、低消費電力なタッチアプリケーションを実現できます。

動作確認デバイス

RL78/G22

RL78/G23

目次

1. システム概要

RL78 ファミリは SNOOZE モード・シーケンサ (SMS) を使用して静電容量タッチ機能 (CTSU2La) の自動判定計測動作(SMS を使用した自動判定計測) を実装可能です。本アプリケーションノートでは、SMS を使用した自動判定計測の作成手順について主に以下の項目に分けて説明します。

- RL78 ファミリ CPU ボード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

2. 関連ドキュメント

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を作成する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. SNOOZE モード・シーケンサ (SMS) を使用した自動判定計測方法

3.1 SMS 計測時に使用するモジュールのフロー

RL78/G22 と RL78/G23 で、SMS 計測時に使用するモジュールのフローが異なります。

- RL78/G22 の場合
  CTSU2La から DTC を使用してポート出力を行います注。ポートから出力された信号を用いて外部割り込み信号を発生させます。割り込み信号によってELCをトリガさせ SMS 処理を開始します。
  【注】 DTC 転送でポート・レジスタ (Pxx) を 8 ビット単位で書き換えます。
  したがって SMS を使用した自動判定計測処理の実行中は、DTC の転送先となるポート・レジスタ (Pxx) を他の機能で使用することはできません。他システムと競合しないように、使用するポート・レジスタ (Pxx) を選択してください。

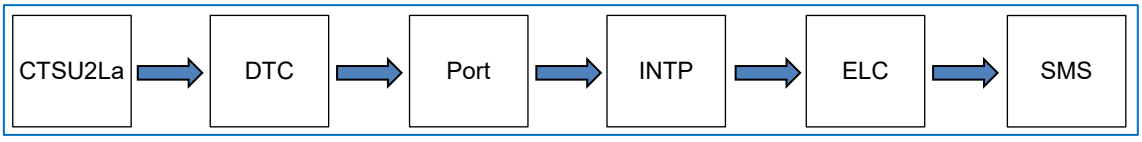


図 3-1 SMS 計測時に使用するモジュールのフロー(RL78/G22)

- RL78/G23 の場合
  CTSU2L から ELCL をトリガさせ SMS 処理を開始します。

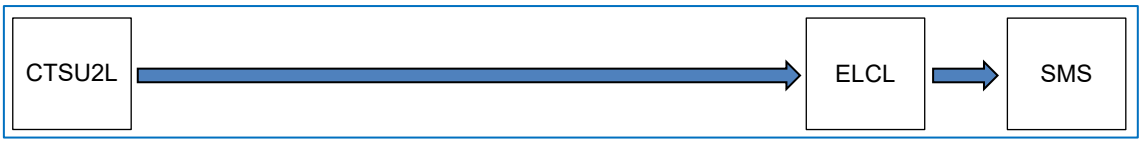


図 3-2 SMS 計測時に使用するモジュールのフロー(RL78/G23)

3.2　外部端子の結線 (RL78/G22 のみ)

RL78/G22 で SMS を使用した自動判定計測を行う場合は、上記フロー内で使用するポート端子 (Pxx) と外部割り込み端子 (INTPxx) を結線してください。なお RL78/G23 では本処理は不要です。

RL78/G22 静電容量タッチ評価システム(製品型名: RTK0EG0042S01001BJ)を使用した場合で説明します。

CPU ボード側のCN2の32 番ピン (P22/TS20) と16 番ピン (P16/INTP5/TS17) を以下のように結線します。

P22　　: CTSU2La から DTC を使用してポート出力するための端子です。
INTP5　: ポート端子から出力された信号を用いて割り込み信号を発生させるための端子です。

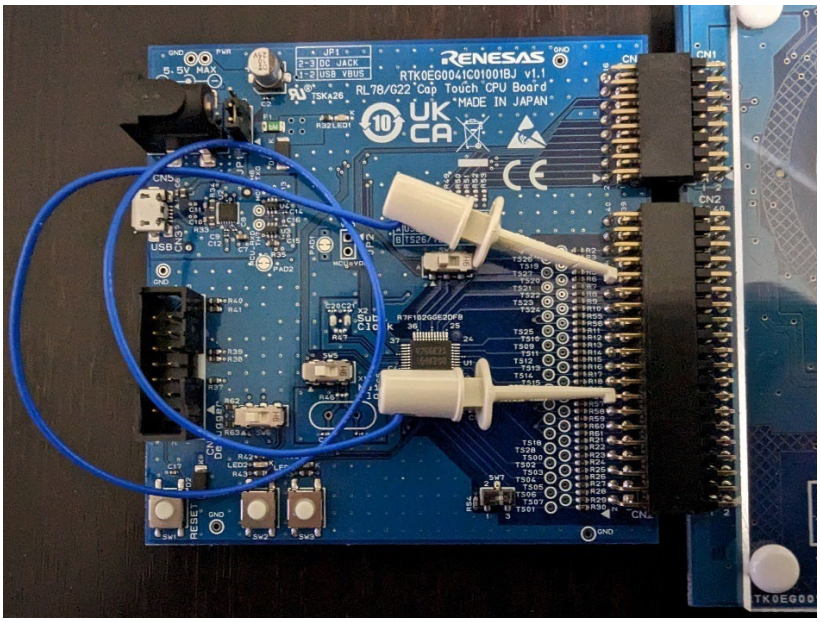


図 3-3　RL78/G22 で SMS 使用時のハードウェア結線パターン

4. 使用する周辺機能

表 4-1, 表 4-2 にサンプルコードで使用する周辺機能を示します。

表 4-1 周辺機能と用途 (RL78/G22)

周辺機能	用途
静電容量センサユニット (CTSU2La)	タッチ電極に発生する静電容量を計測
32 ビット・インターバル・タイマ (TML32)	STOP モードを解除し、SNOOZE モードへ遷移するためのタイマ (計測周期：20 ms)
データ・トランスファ・コントローラ (DTC)	DTC を使用してポート出力を行う。 ポートから出力された信号を用いて割り込み信号を発生させる。 割り込み信号によって ELC をトリガさせる。 そして、SMS 処理を開始する。
ポート	
割り込みコントローラ (INTP)	
イベント・リンク・コントローラ (ELC)	
SNOOZE モード・シーケンサ (SMS)	

表 4-2 周辺機能と用途 (RL78/G23)

周辺機能	用途
静電容量センサユニット (CTSU2L)	タッチ電極に発生する静電容量を計測
32 ビット・インターバル・タイマ (TML32)	STOP モードを解除し、SNOOZE モードへ遷移するためのタイマ (計測周期：20 ms)
ロジック&イベント・リンク・コントローラ (ELCL)	計測データ転送要求 (INTCTSURD) で ELCL をトリガし、SMS 処理を開始する。
SNOOZE モード・シーケンサ (SMS)	

5. 動作確認環境

表 5-1 にプロジェクトで使用する周辺環境を示します。

表 5-1 動作確認環境

項目	内容
使用マイコン	RL78/G22 (R7F102GGE2DFB)
	RL78/G23 (R7F100GSN2DFB)
動作電圧	3.3V LVD0 検出電圧： 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	高速オンチップ・オシレータ・クロック (f_{IH}) : 32 MHz 低速オンチップ・オシレータ・クロック (f_{IL}) : 32.768 kHz
ターゲットボード	RL78/G22 静電容量タッチ評価システム (製品型名：RTK0EG0042S01001BJ)
	RL78/G23 静電容量タッチ評価システム (製品型名：RTK0EG0030S01001BJ)
統合開発環境	e^2 studio (2024-04)
スマート・コンフィグレータ	V24.4.0
C コンパイラ	CC-RL V1.13.00 最適化レベルのオプション: -Odefault
静電容量式タッチセンサ対応開発支援ツール	QE for Capacitive Touch V3.4.0
エミュレータ	E2 エミュレータ Lite (RTE0T0002LKCE00000R)

6. ソフトウェア設定

6.1 オプション・バイト設定

表 6-1, 表 6-2 にオプション・バイトの設定を示します。

表 6-1 オプション・バイト設定 (RL78/G22)

アドレス	設定値	内容
000C0H / 020C0H	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 / 020C1H	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 / 020C2H	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 / 020C3H	1000 0100B (0x84)	オンチップ・デバッグ動作許可

表 6-2 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、「ユーザー・オプション・バイト値」及び「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。

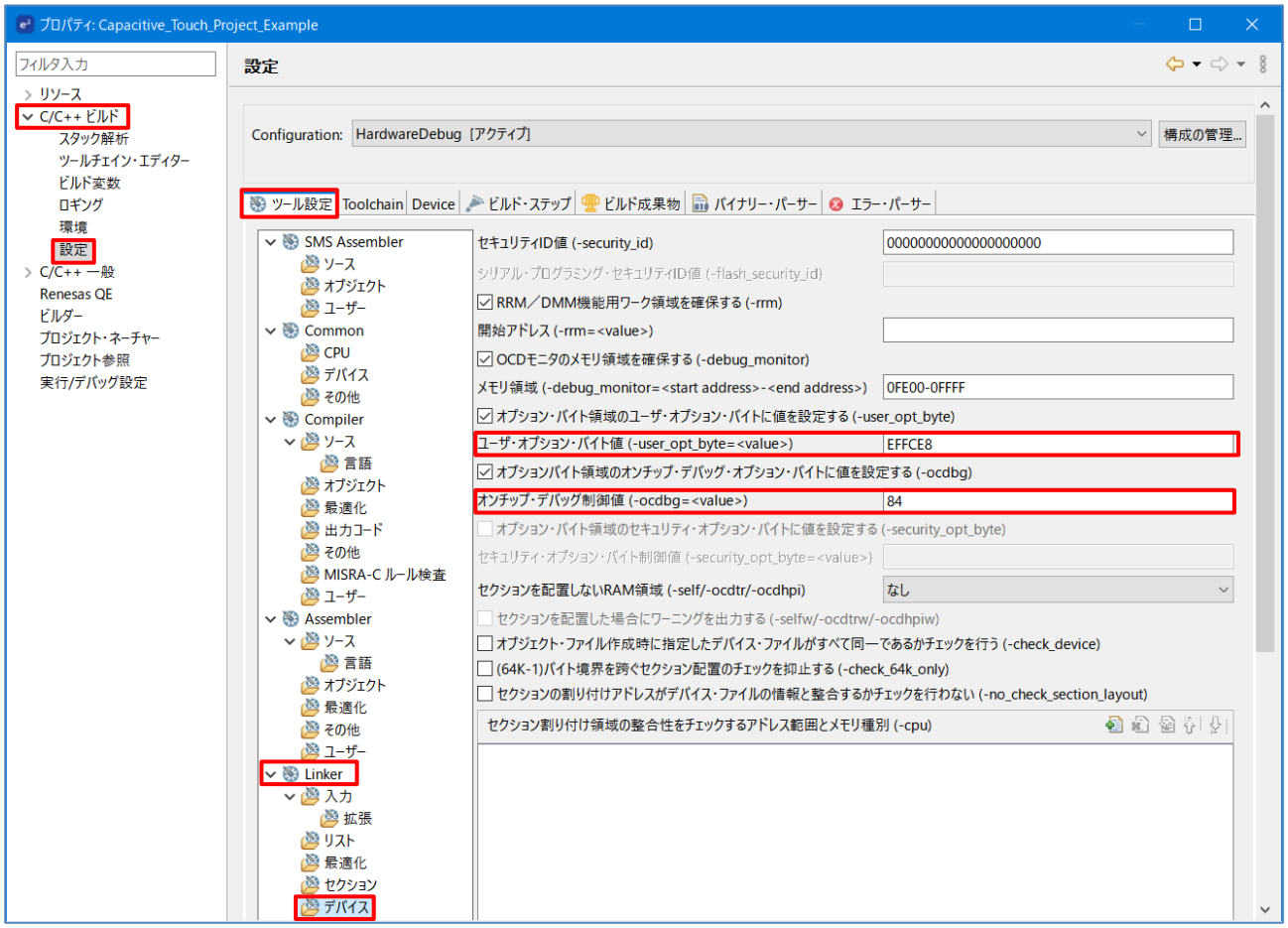


図 6-1 オプション・バイト設定

6.2　静電容量タッチ設定

本アプリケーション例での静電容量タッチ設定を示します。

RL78/G22 の場合は、静電容量タッチセンサ(CTSU2La) の複数電極接続 (MEC) 機能を SMS と併用することで、SMS のみでのタッチ計測動作に比べてさらに低消費電力で動作が可能です。

※RL78/G23 の静電容量タッチセンサは CTSU2L のため、MEC 機能は非搭載です。

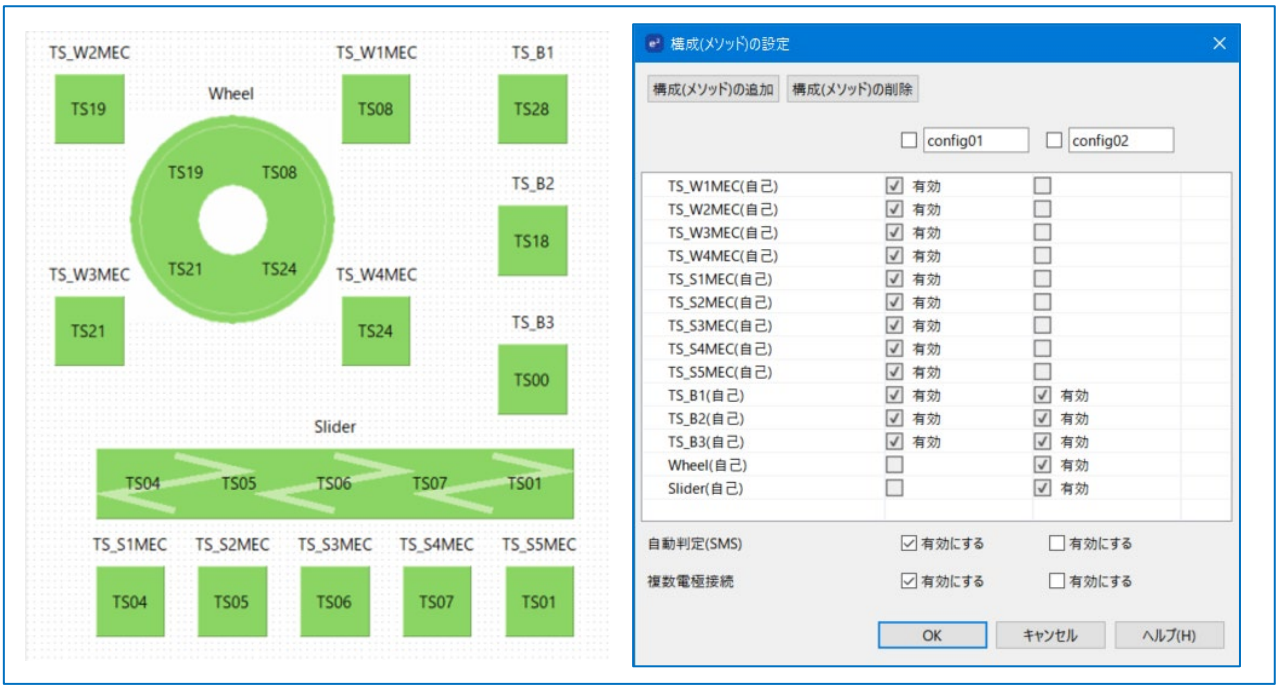


図 6-2 本アプリケーション例での静電容量タッチ設定 (RL78/G22)

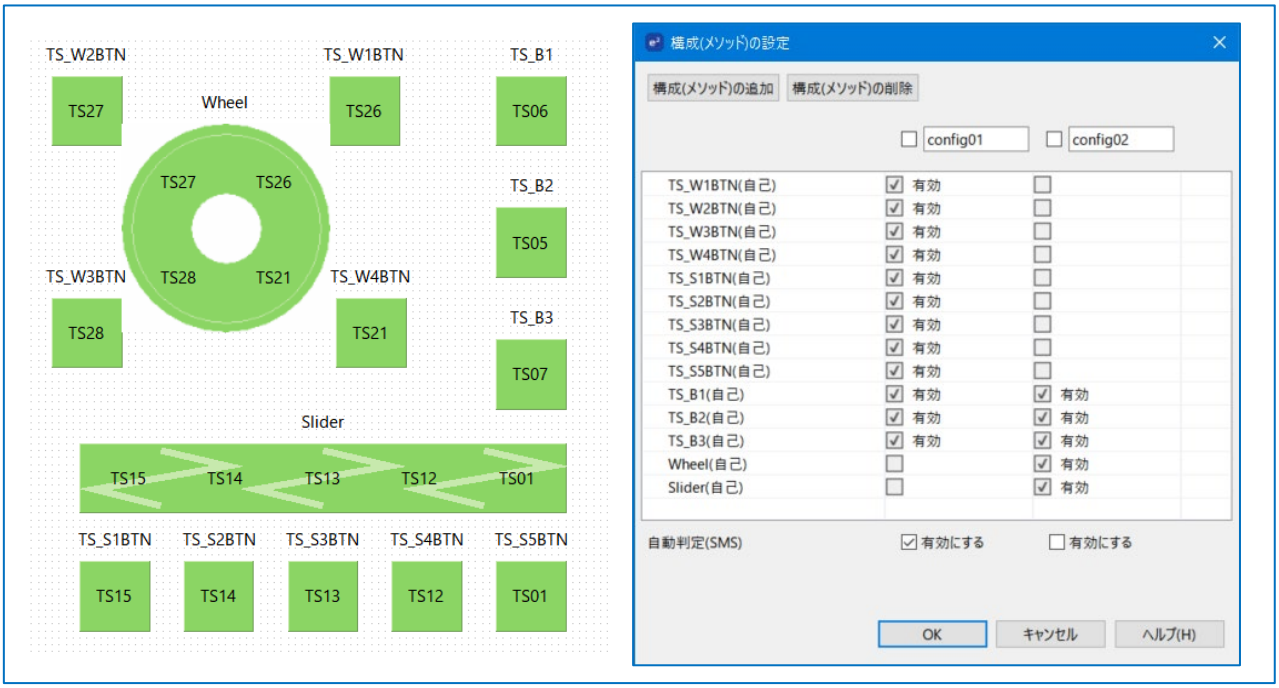


図 6-3 本アプリケーション例での静電容量タッチ設定 (RL78/G23)

6.3　使用コンポーネント

図 6-3, 図 6-4 にスマート・コンフィグレータで設定したコンポーネント及びモジュールを示します。

コンポーネント	バージョン	設定
Board Support Packages. - v1.62 (r_bsp)	1.62	r_bsp(使用中)
Capacitive Sensing Unit driver. (r_ctsu)	1.50	r_ctsu(使用中)
Touch middleware. (rm_touch)	1.50	rm_touch(使用中)
インターバル・タイマ	1.4.0	Config_ITL000(ITL000: 使用中)
ポート	1.4.1	Config_PORT(PORT: 使用中)
割り込みコントローラ	1.4.0	Config_INTC(INTC: 使用中)
電圧検出回路	1.3.0	Config_LVD0(LVD0: 使用中)

図 6-4 スマート・コンフィグレータで設定したコンポーネント及びモジュール (RL78/G22)

コンポーネント	バージョン	設定
Board Support Packages. - v1.62 (r_bsp)	1.62	r_bsp(使用中)
Capacitive Sensing Unit driver. (r_ctsu)	1.50	r_ctsu(使用中)
Touch middleware. (rm_touch)	1.50	rm_touch(使用中)
インターバル・タイマ	1.4.0	Config_ITL000(ITL000: 使用中)
電圧検出回路	1.3.0	Config_LVD0(LVD0: 使用中)

図 6-5 スマート・コンフィグレータで設定したコンポーネント及びモジュール (RL78/G23)

7.　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。

===== SMS、MEC を使用したタッチ計測 (低消費モード) 注 ==========

① イニシャルオフセットチューニング後、RM_TOUCH_SmsSet 関数を実行することで SMS を設定します。
② RM_TOUCH_ScanStart 関数の実行で、タッチ計測設定および SNOOZE 機能を有効にする設定を行い、外部トリガ待ち状態にする。
③ TML32 のタイマカウントを開始します。(計測周期：20ms)
④ STOP 命令の実行で STOP モードへ遷移します。
⑤ TML32 の割り込み要求が発生することで、ELC (G23 の場合は ELCL) からの外部トリガにより、SNOOZE モードへ遷移してタッチ計測を開始します。
⑥ 計測終了割り込みが発生すると SNOOZE モードのまま SMS によりタッチオン/オフ判定を行います。
⑦ タッチオン判定時に通常モードに遷移します。タッチオフ判定時は④に戻ります。

=====ボタン/スライダ/ホイールのタッチ計測 (通常モード) ==========

⑧ RM_TOUCH_ScanStart 関数を実行して外部トリガ待ち状態にします。
⑨ TML32 のタイマカウントを開始します。(計測周期：20 ms)
⑩ TML32 のタイマカウントが 20ms 経過で、ELC (G23 の場合は ELCL) からの外部トリガにより、タッチ計測を行います。
⑪ タッチ計測終了後、タッチオン/オフ判定を実行します。なお、本アプリケーション例では判定結果に関わらず①に遷移します。

【注】RL78/G23 は SMS を使用したタッチ計測 (低消費モード) になります。

8. 静電容量タッチアプリケーション開発手順

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開発手順を以下に示します。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作成した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 のモジュールの 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。

次項から各手順について説明します。

なお特に説明のない限り、RL78/G22 での設定手順を記載しています。

8.1　プロジェクト作成

e² 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。

以下に手順を示します。

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 CCRL
- ツールチェーン・バージョン: v1.13.00
- Target Board: Custom
- ターゲット・デバイス: RL78/G22 (R7F102GGExFB)
  RL78/G23 (R7F100GSNxFB)
- “**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。

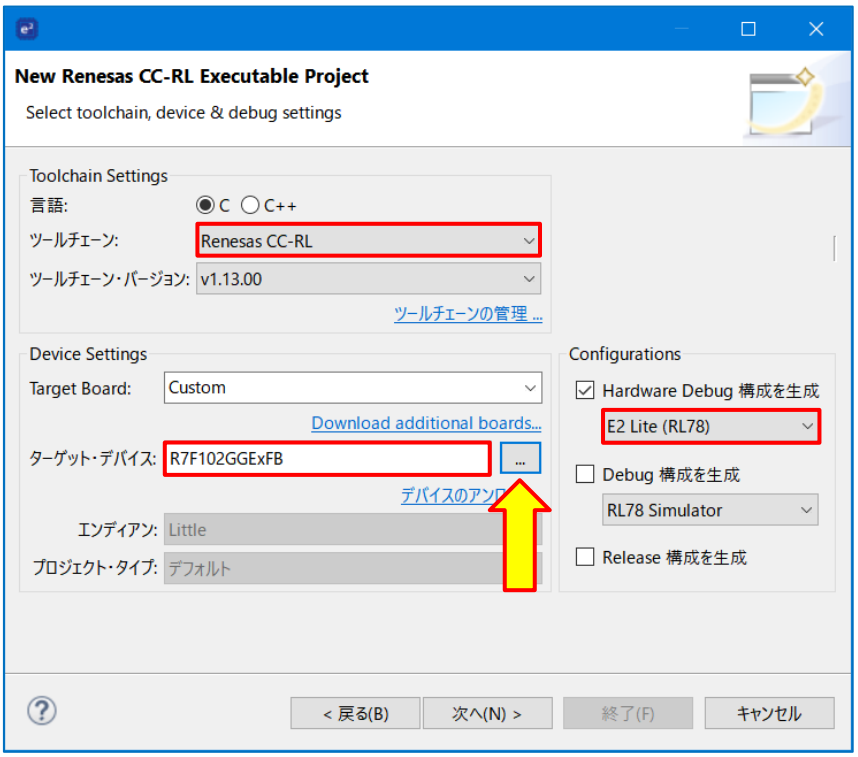


図 8-1 ツールチェーン，デバイス，デバッグ設定の選択

“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。

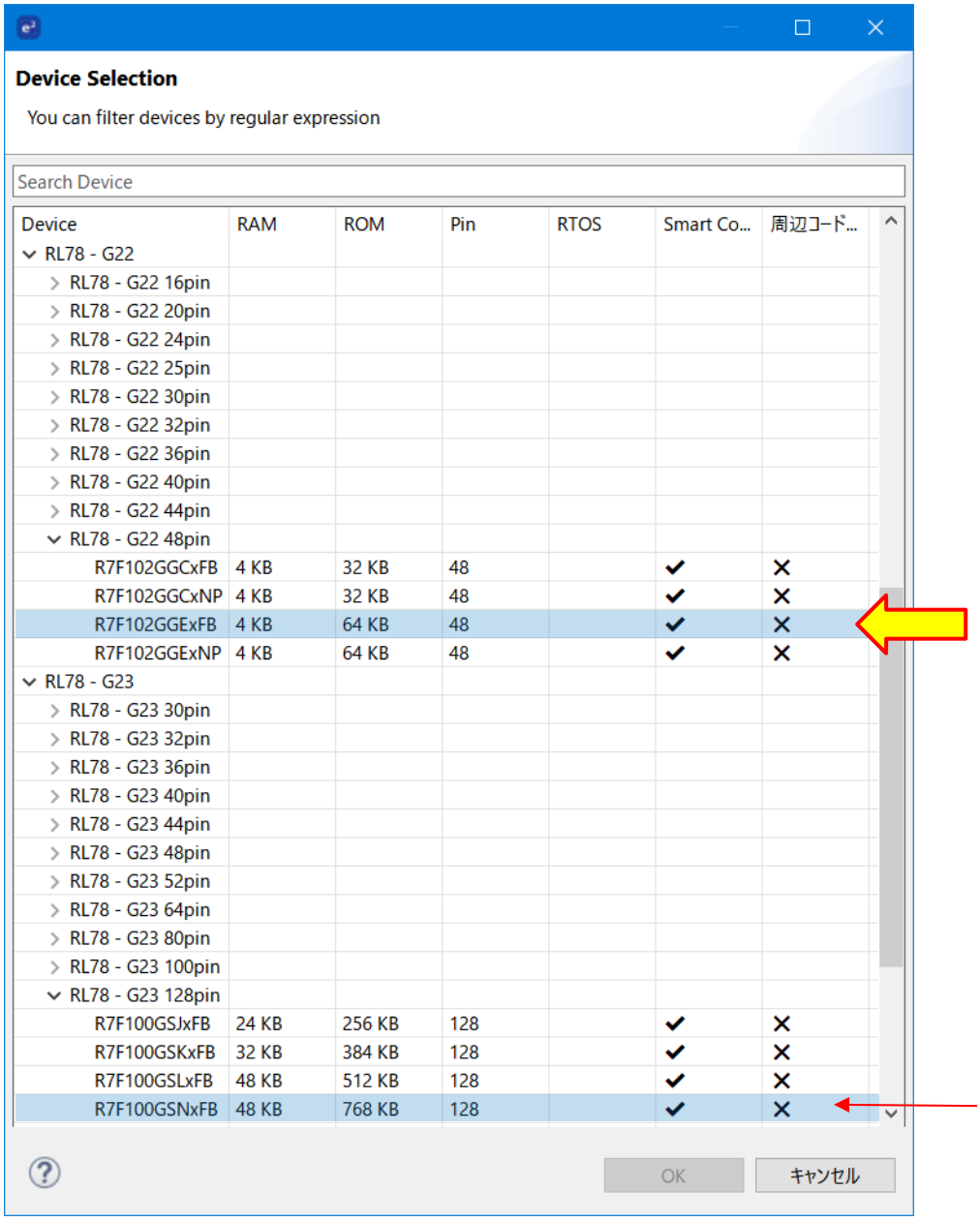


図 8-2 ターゲット・デバイスの選択

【備考】RL78/G23 RSSK (RTK0EG0030S01001BJ) で動作確認する場合は、ターゲット・デバイスに「R7F100GSNxFB」を選択してください。

6. 完了したら、[**OK**]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。

完了すると e² 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。

8.2　スマート・コンフィグレータによるモジュール作成

8.2.1　スマート・コンフィグレータによるコンポーネント追加

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。以下に手順を示します。

RL78/G22 と RL78/G23 では一部設定内容が異なる箇所があります。

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 のクロック設定を下記のように行います。本アプリケーション例では低速周辺クロック (fSXP) に低速オンチップ･オシレータを使用します。下図のとおり、XT1 発振回路のチェックを外し、fSXP に"低速オンチップ ･オシレータ"を選択します。

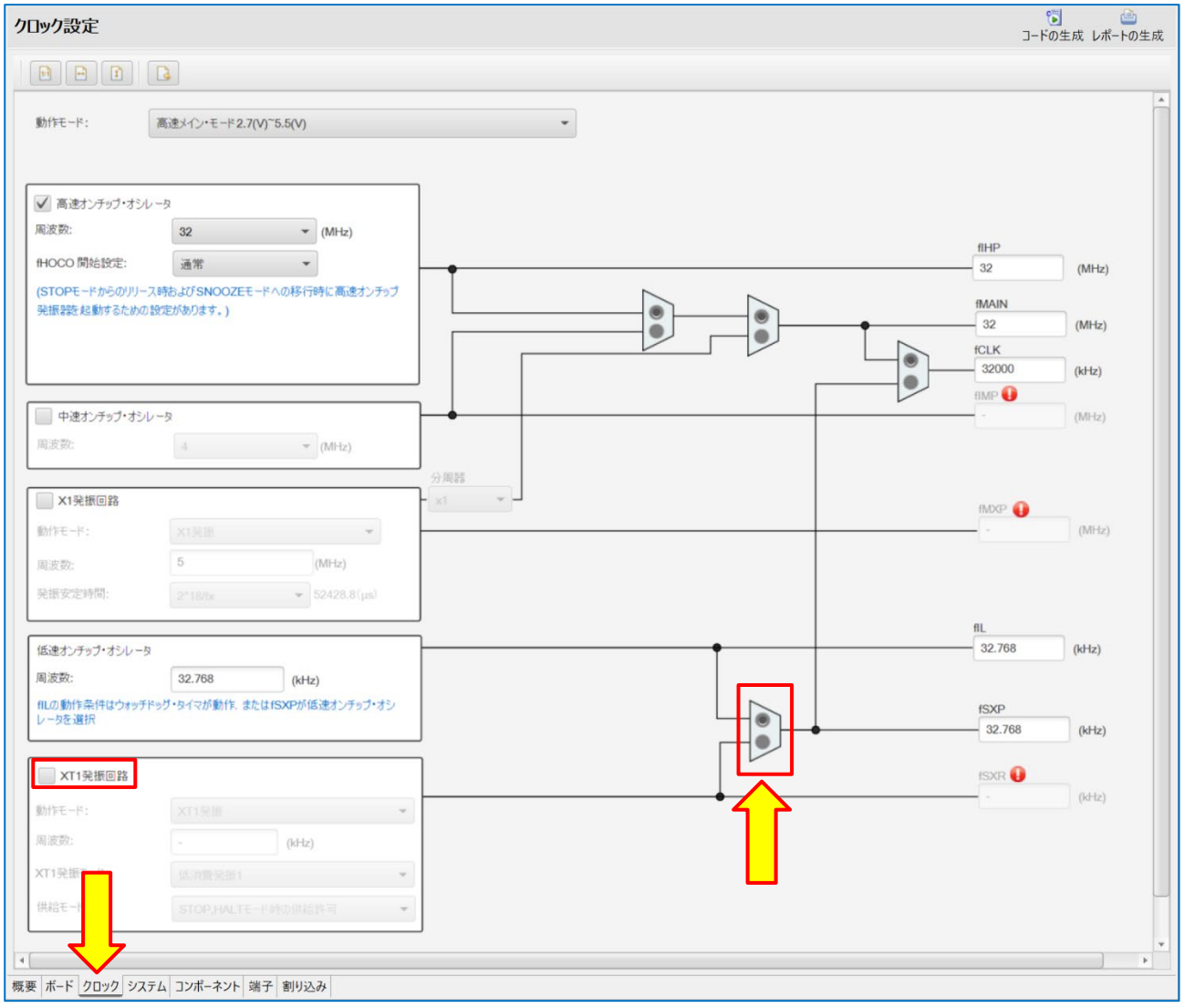


図 8-3 クロック設定

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。
   RL78/G23 の場合は EVDD 設定も設定する必要があります。

図 8-4 動作モード設定 (RL78/G22)

図 8-5 動作モードおよび EVDD 設定 (RL78/G23)

3. [**システム**]タブに移動します。
   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 8-6 オンチップ・デバック設定

4. [**コンポーネント**]タブを選択し、r_bsp のバージョンを確認します。
   r_bsp を選択して右クリックし、"**バージョンの変更**"をクリックします。

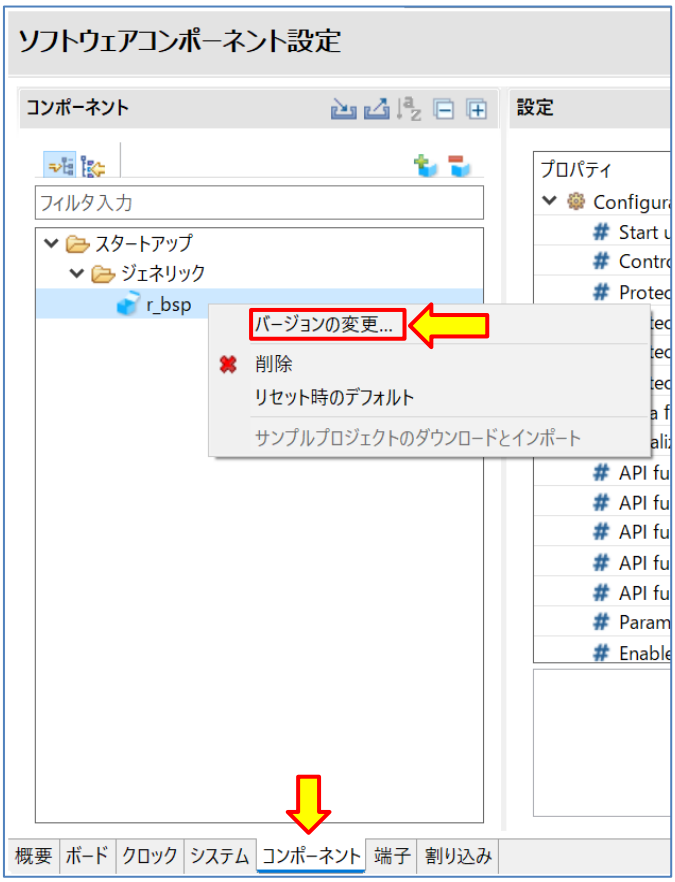


図 8-7 r_bsp のバージョン変更

“現在のバージョン” が 1.62 以降と異なる場合は “変更後のバージョン” で 1.62 以降を選択して[**次へ**]をクリックでバージョンを変更してください。

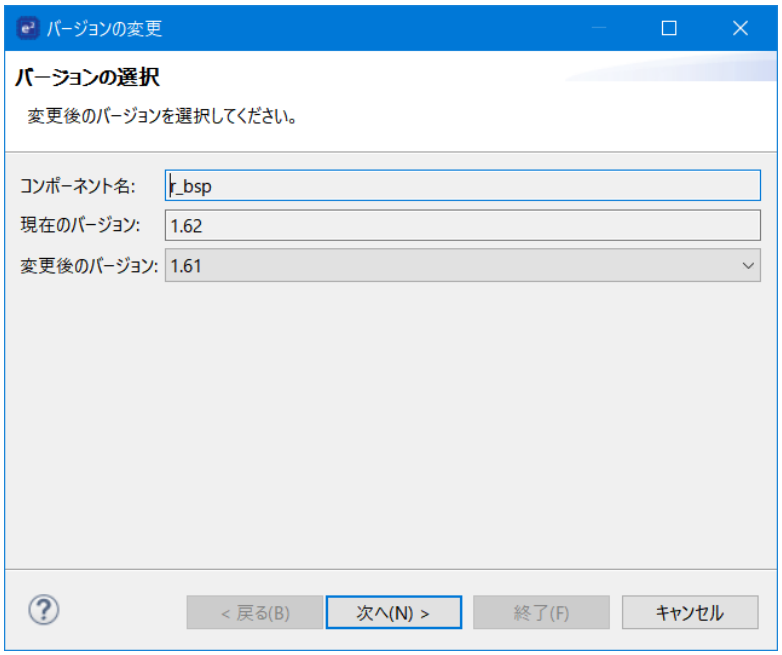


図 8-8 r_bsp のバージョン選択

5. SIS モジュールと各種コンポーネントをプロジェクトに追加します。
コンポーネントの追加ボタン (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。

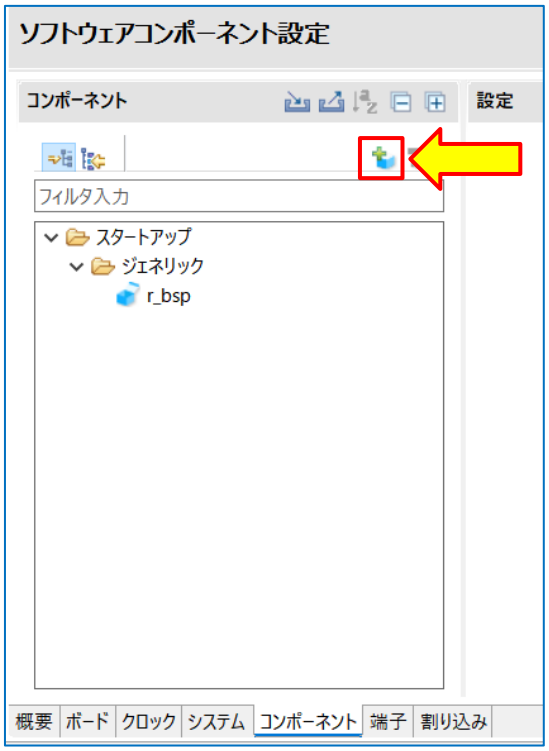


図 8-9 コンポーネントの追加

6. “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。

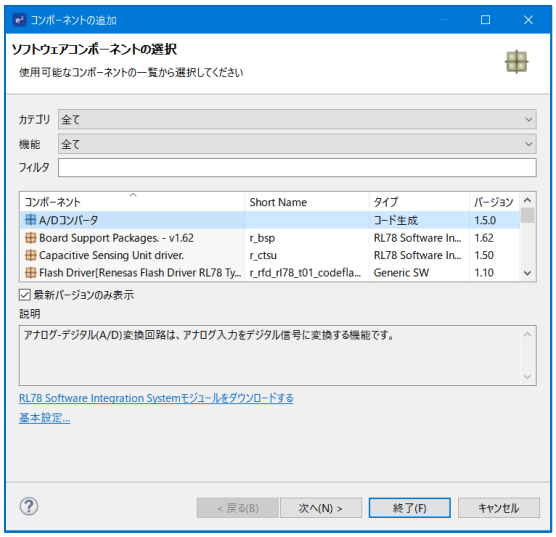


図 8-10 “コンポーネントの追加”ダイアログ

【注意】 SIS (Software Integration System) モジュールを初めて使用する場合は、以下の①～④の順に選択してダウンロードしてください。

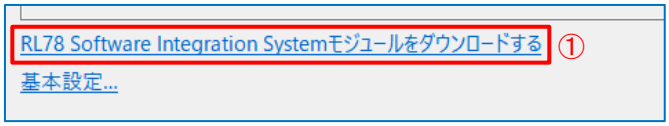


図 8-11 SIS モジュールのダウンロードリンク

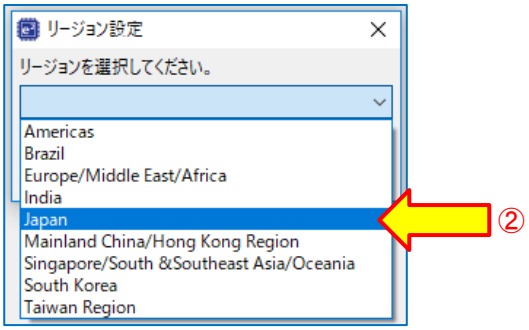


図 8-12 リージョン選択

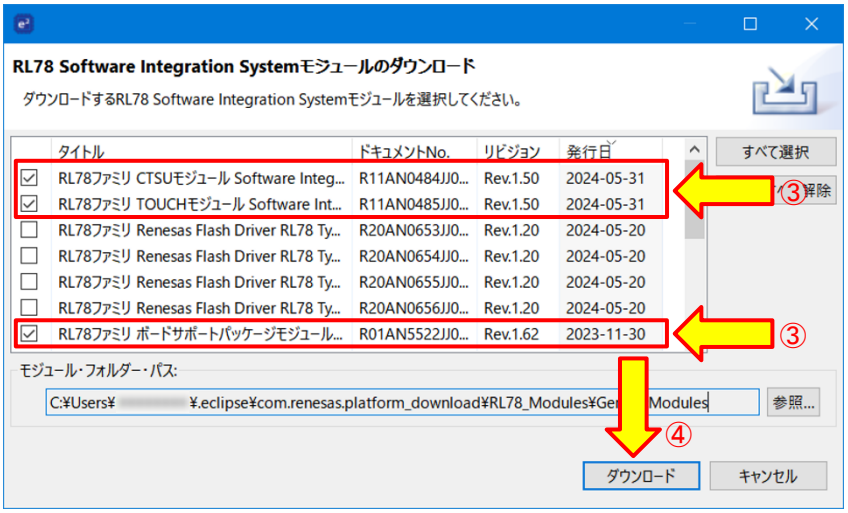


図 8-13 SIS モジュールのダウンロード

7. 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。
【注】RL78/G23 の場合、SMS を使用した自動判定計測の動作に「ポート」「割り込みコントローラ」の設定は不要です。

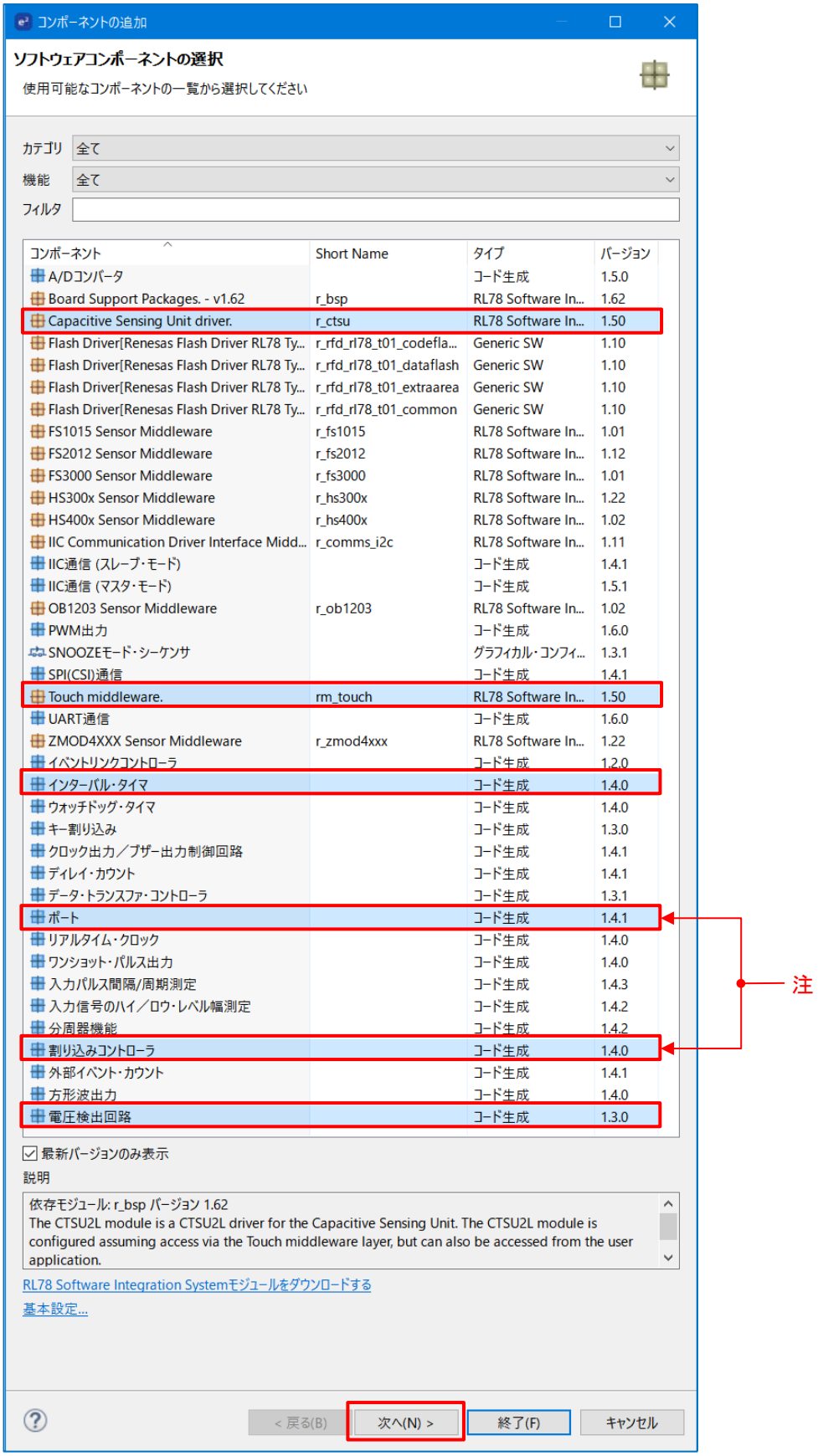


図 8-14 ソフトウェアコンポーネントの選択

8. 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。
【注】RL78/G23 の場合、SMS を使用した自動判定計測の動作に「ポート」「割り込みコントローラ」の設定は不要です。コンポーネントの追加を行わないため、表示されません。

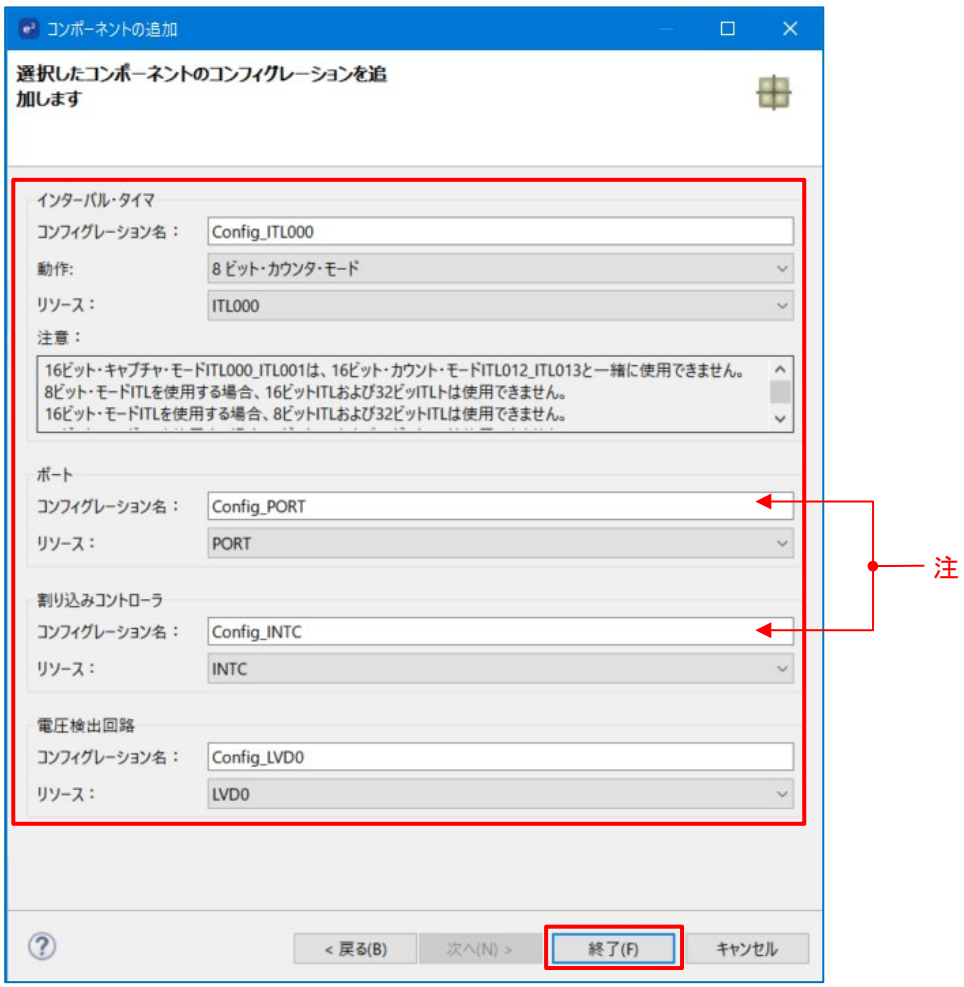


図 8-15 コンポーネントのリソース設定

以下に示すようにコンポーネントが追加されます。

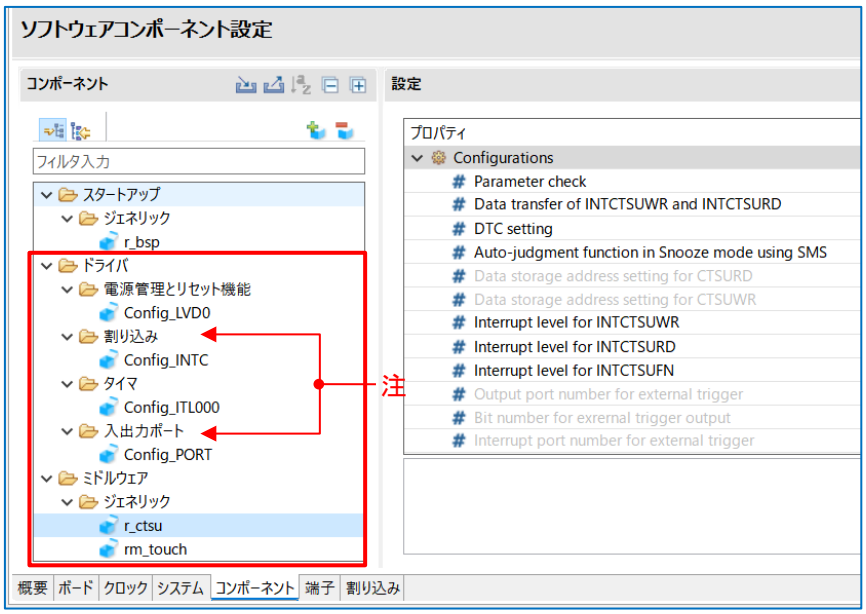


図 8-16 ソフトウェアコンポーネント設定 (コンポーネント追加後)

8.2.2　スマート・コンフィグレータによるコンポーネント設定の変更

追加した各コンポーネントについて以下のように設定を行います。

- CTSU コンポーネント設定
  RL78/G22 と RL78/G23 では設定内容が異なります。

  ① “Data transfer of INTCTSUWR and INTCTSURD”の設定は“DTC”を選択します。

  ② “Data storage address setting for CTSURD”の設定は、デバイスによって異なります。
  RL78/G22 の場合は、「0xFEF00」となります。
  RL78/G23 の場合は、「0xFF500」となります。

  ③ RL78/G22 では本設定で、CTSU2La から DTC を介して伝達されるタッチ計測完了信号を出力するためのポート端子を選択します。RL78/G23 では設定不要のため、表示されません。
  - Output port number for external trigger: 使用するポートグループを選択します。
  - Bit number for external trigger output: 上記ポートグループのうち、使用するビット番号を選択します。

  【注意】DTC 転送でポート・レジスタ (Pxx) を 8 ビット単位で書き換えます。
  したがって SMS を使用した自動判定計測処理の実行中は、DTC の転送先となるポート・レジスタ (Pxx) を他の機能で使用することはできません。
  他システムと競合しないように使用するポート・レジスタ (Pxx) を選択してください。

  ④ RL78/G22 ではポート端子からの出力信号を入力する外部割り込み端子を設定します。
  RL78/G23 では設定不要のため、表示されません。

  ⑤ タッチ計測に使用する TS 端子を有効にします。
  【注意】RL78/G22 では、SMS を使用した自動判定計測で使用するポート端子および外部割り込み端子との、兼用機能となる TS 端子は使用できません。本アプリケーション例では[P22/TS20]と[P16/INTP5/TS17]を使用するため、TS17 端子と TS20 端子のチェックを外します。
  RL78/G23 では任意の TS 端子を選択可能です。

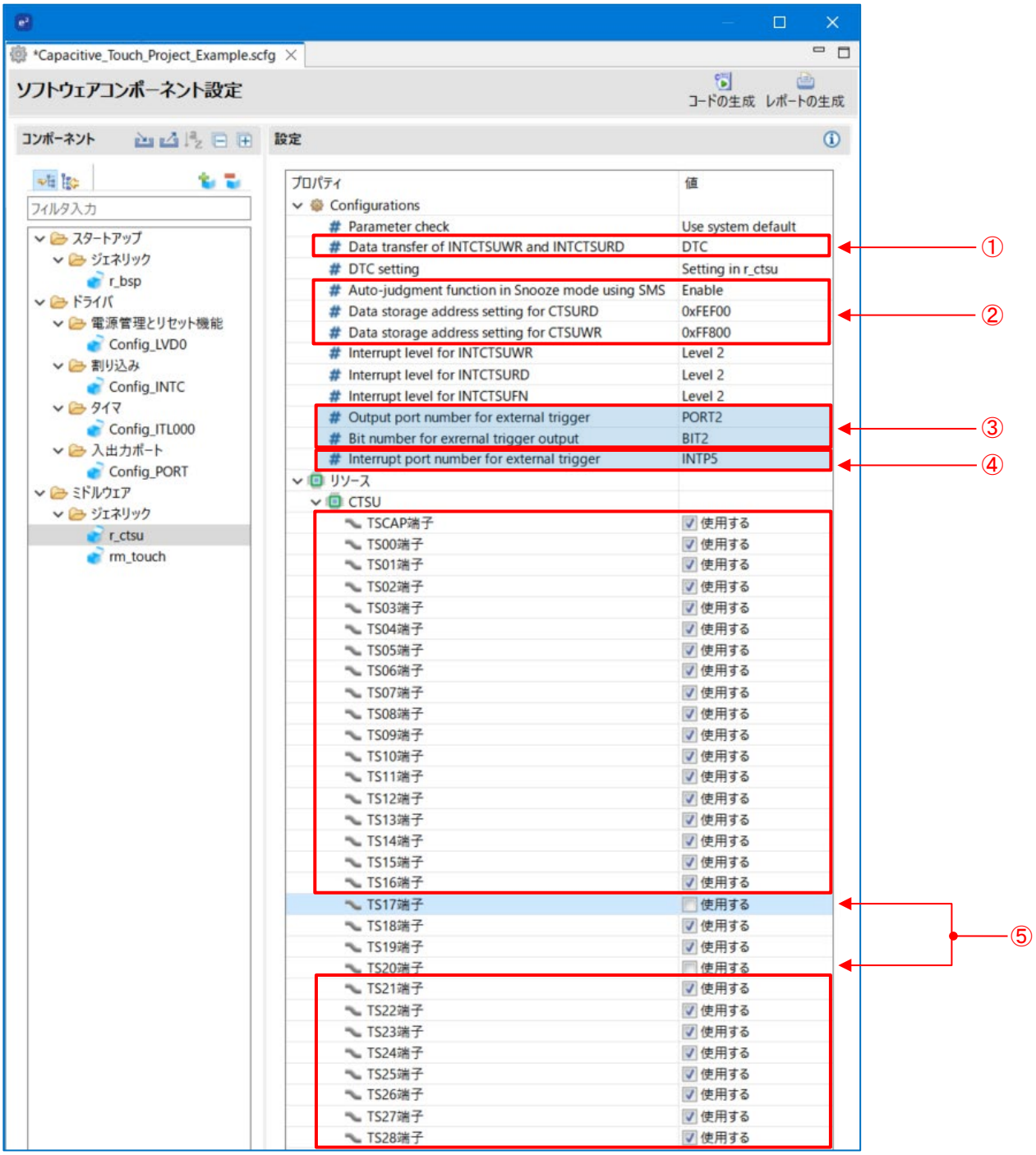


図 8-17 CTSU コンポーネント設定

- 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。

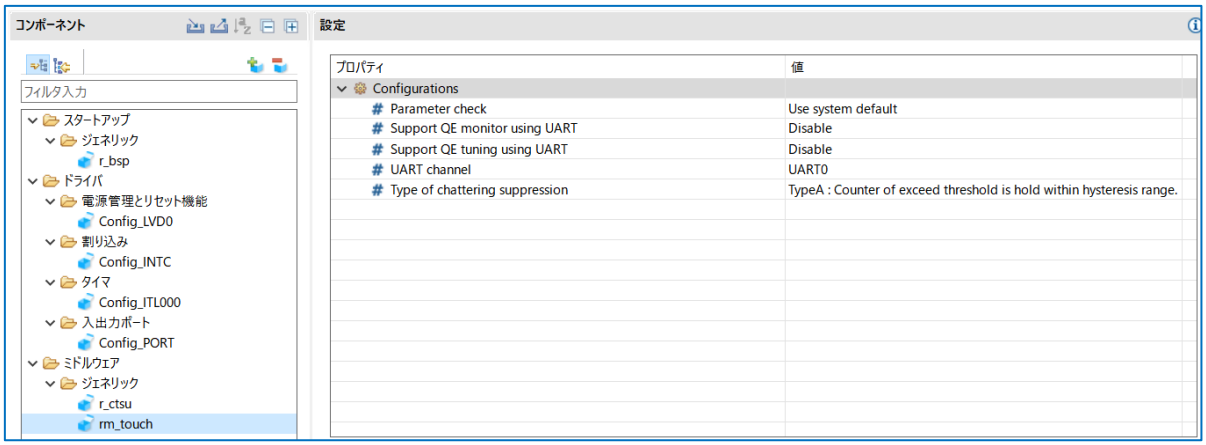


図 8-18 Touch コンポーネント設定

- INTC コンポーネント設定
  RL78/G23 の場合、SMS を使用した自動判定計測の動作に以下の設定は不要です。

図 8-19 INTC コンポーネント設定

- ITL コンポーネント設定

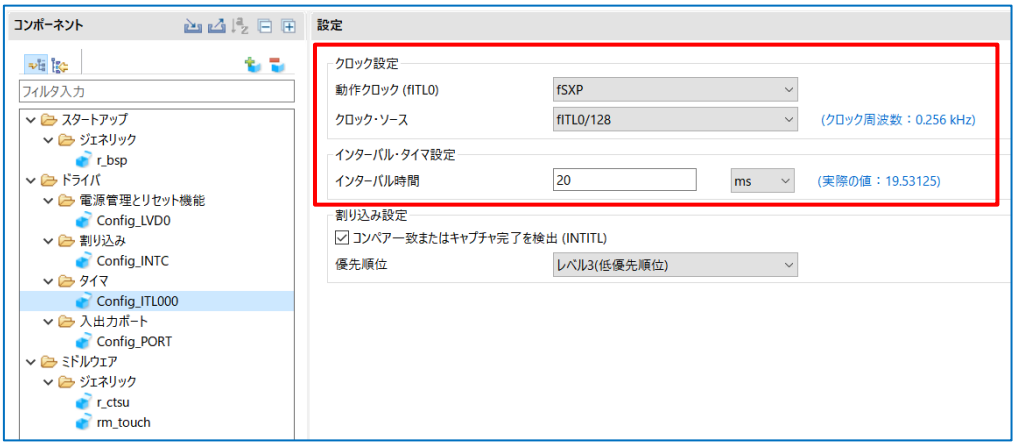


図 8-20 ITL コンポーネント設定

- PORT コンポーネント設定
  RL78/G23 の場合、SMS を使用した自動判定計測の動作に以下の設定は不要です。

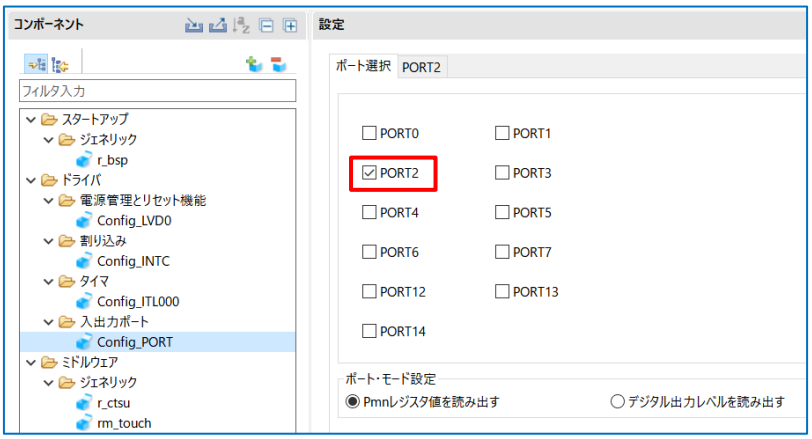


図 8-21 PORT コンポーネント設定 (ポート選択タブ)

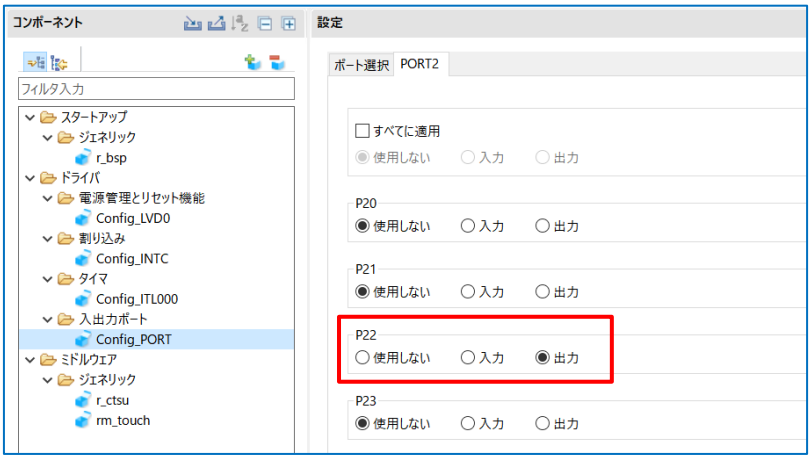


図 8-22 PORT コンポーネント設定 (PORT2 タブ)

- LVD コンポーネント設定

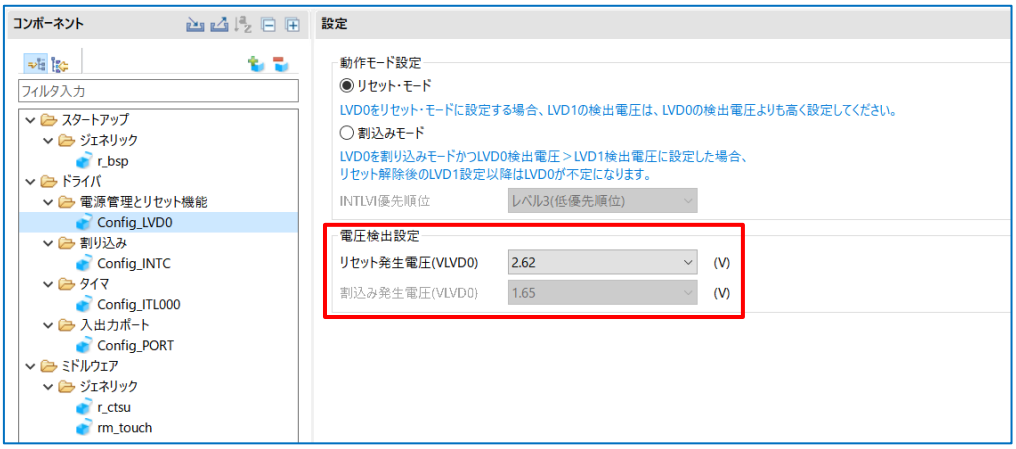


図 8-23 LVD コンポーネント設定

- BSP コンポーネント設定
  "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。

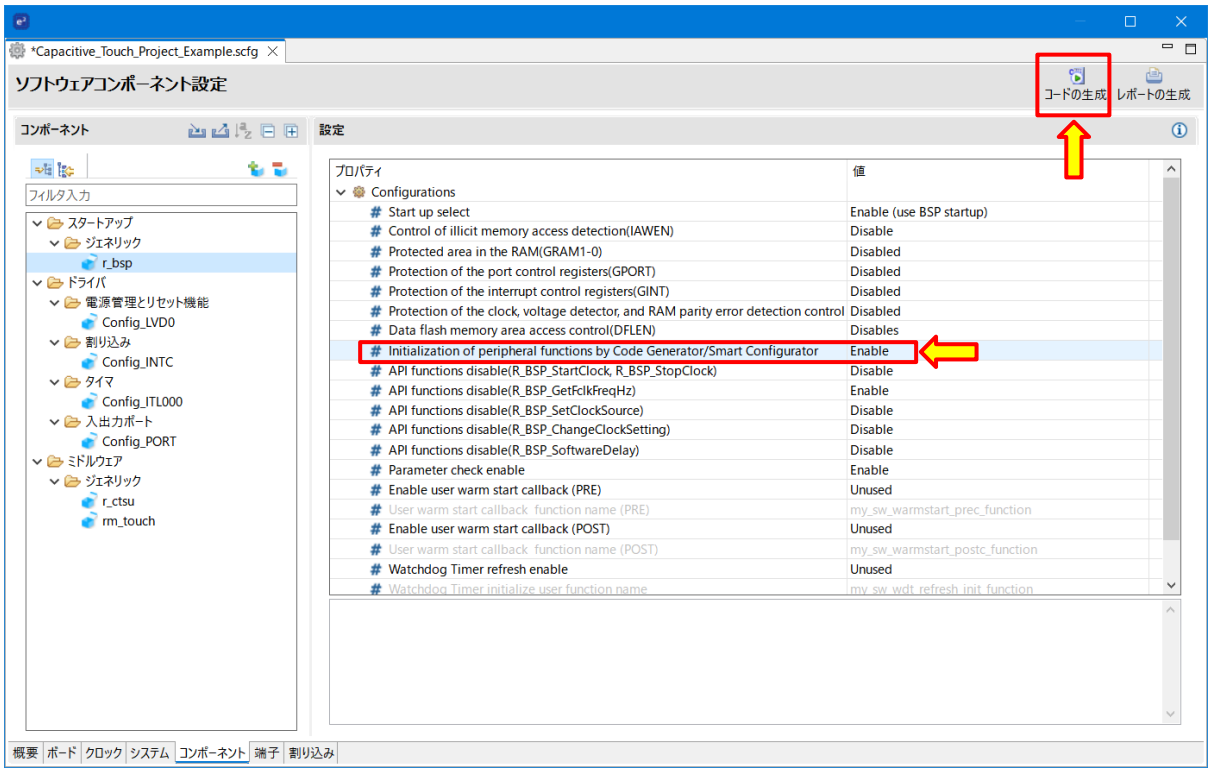


図 8-24 BSP コンポーネント設定

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生成します。以上でコンポーネントの追加は完了です。

8.3　静電容量タッチインタフェース作成

QE for Capacitive Touch を使用して静電容量タッチインタフェースの設定を作成します。

以下に手順を示します。

なお RL78/G22 と RL78/G23 では一部設定内容が異なります。

1. e² studio のメニュー[**Renesas Views**] - [**Renesas QE**] - [**CapTouch メイン (QE)**]／QE V3.2.0 以降では[**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。

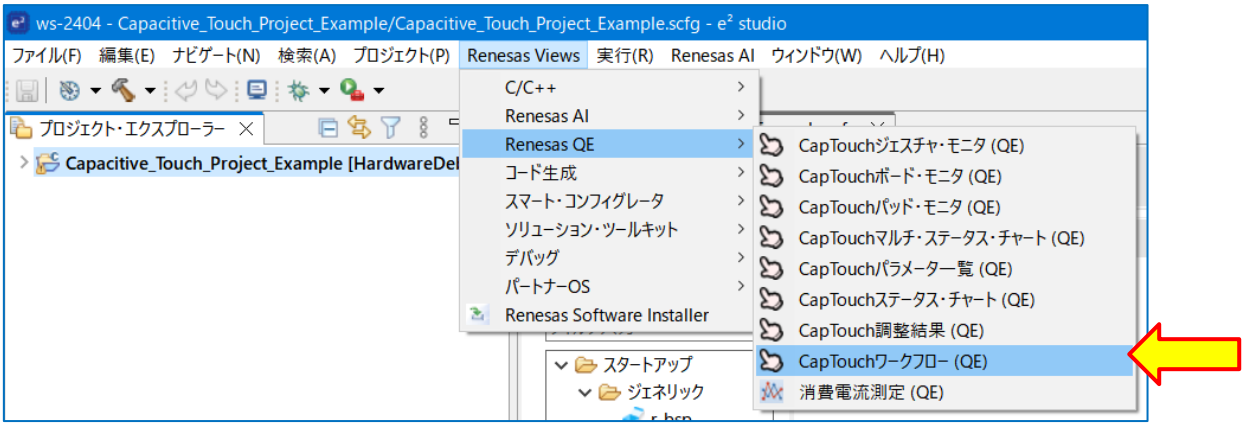


図 8-25 Cap Touch ワークフロー（QE）の選択

2. “**CapTouch メイン (QE)**／QE V3.2.0 以降では[**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**”を選択し、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。

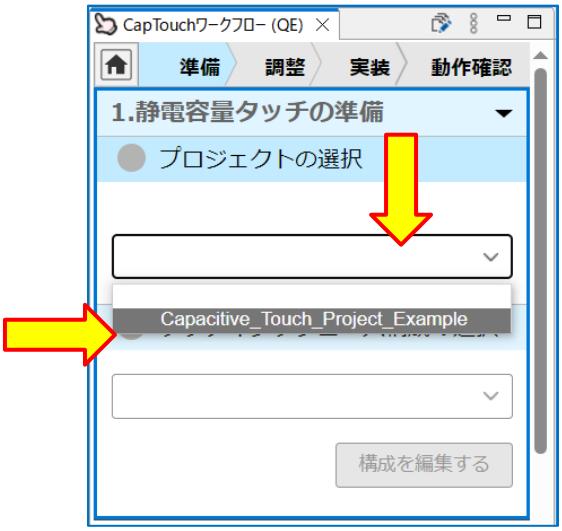


図 8-26 プロジェクトの選択

3. “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、新しいタッチインタフェース構成を生成します。

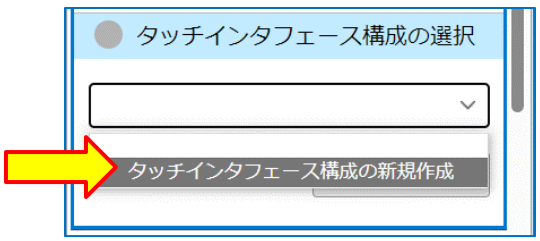


図 8-27 タッチインタフェース構成の新規作成

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。

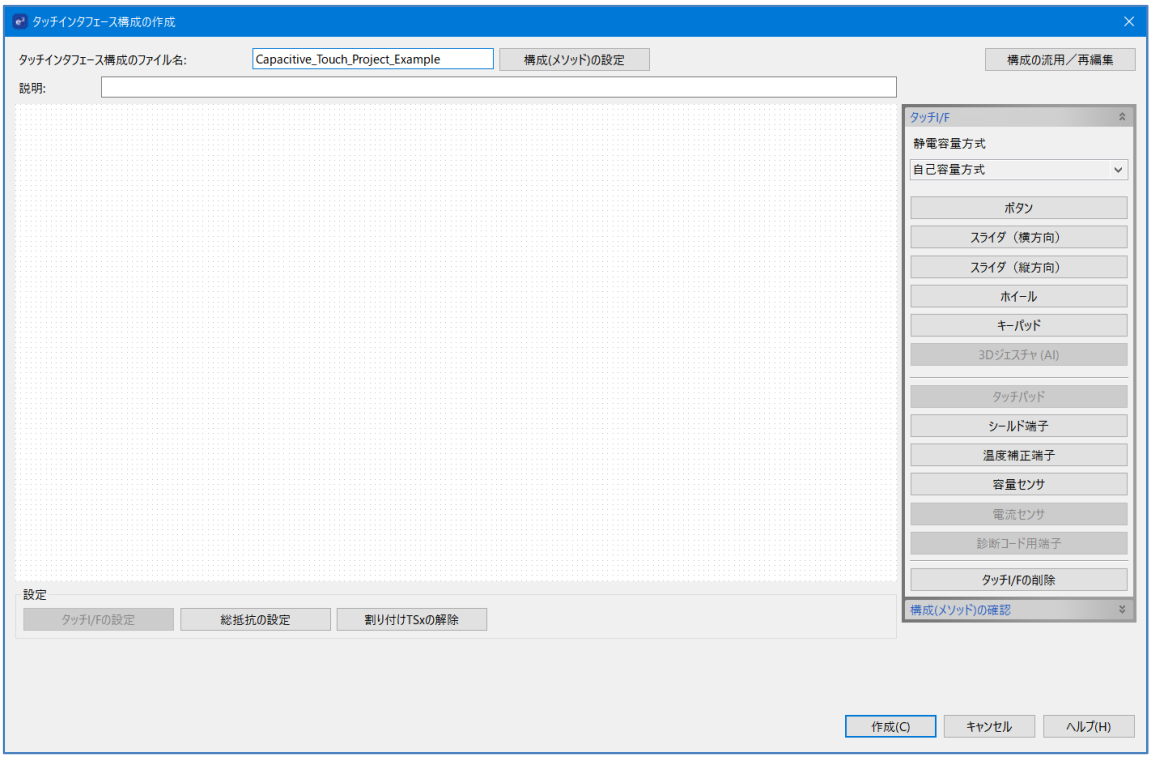


図 8-28 タッチインタフェース配置領域

5. ウィンドウの右側から[**ボタン**]を選択して 3 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状態になります。

【備考】次の手順で各ボタンへタッチセンサが問題なく割り当てられると「設定内容に問題があります。」というエラー表示は消えます。
そのため、このまま次の手順に進むことができます。

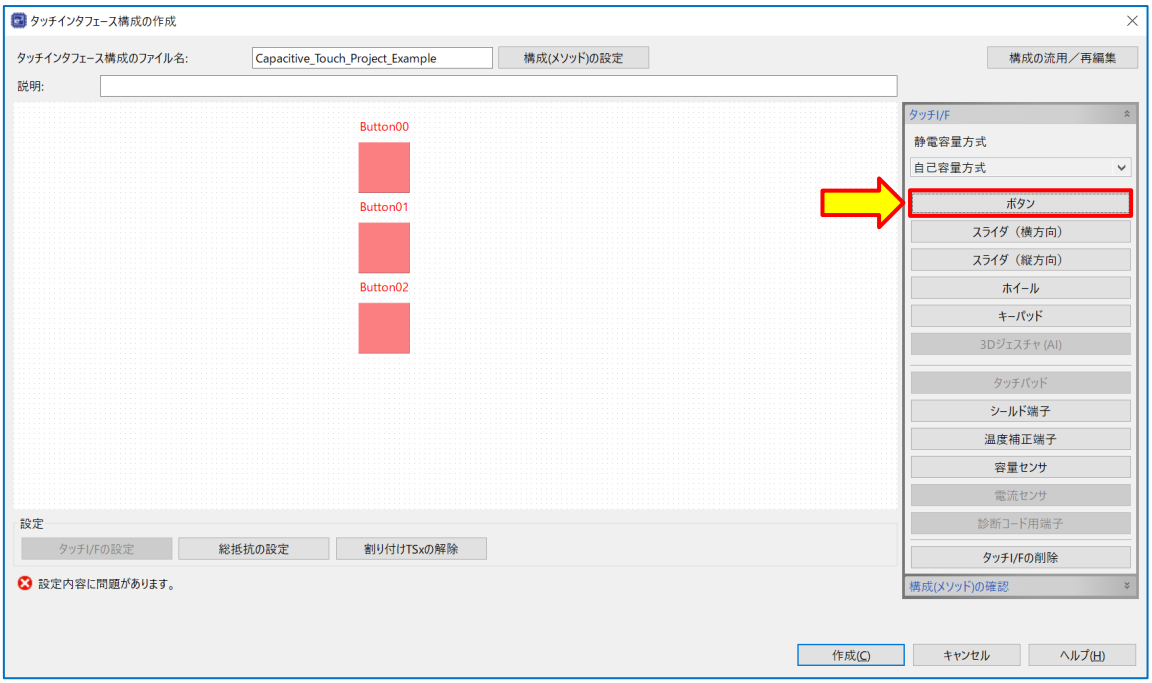


図 8-29 タッチインタフェース (ボタン 3 個) の追加

6. 各ボタンの名前とタッチセンサの割り当てを行います。
“**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。ここでは名前とタッチセンサ (赤枠) を変更して[**OK**]をクリックします。
RL78/G23 の場合は、Button00 に TS06 を割り当てます。

【注意】1. SMS 起動用の外部トリガとして使用するポート端子と端子機能を兼用するタッチセンサ (TSxx) は使用できません。
2. ボタン名称は任意に設定できますが、文字数制限によりエラーが表示される場合があります。

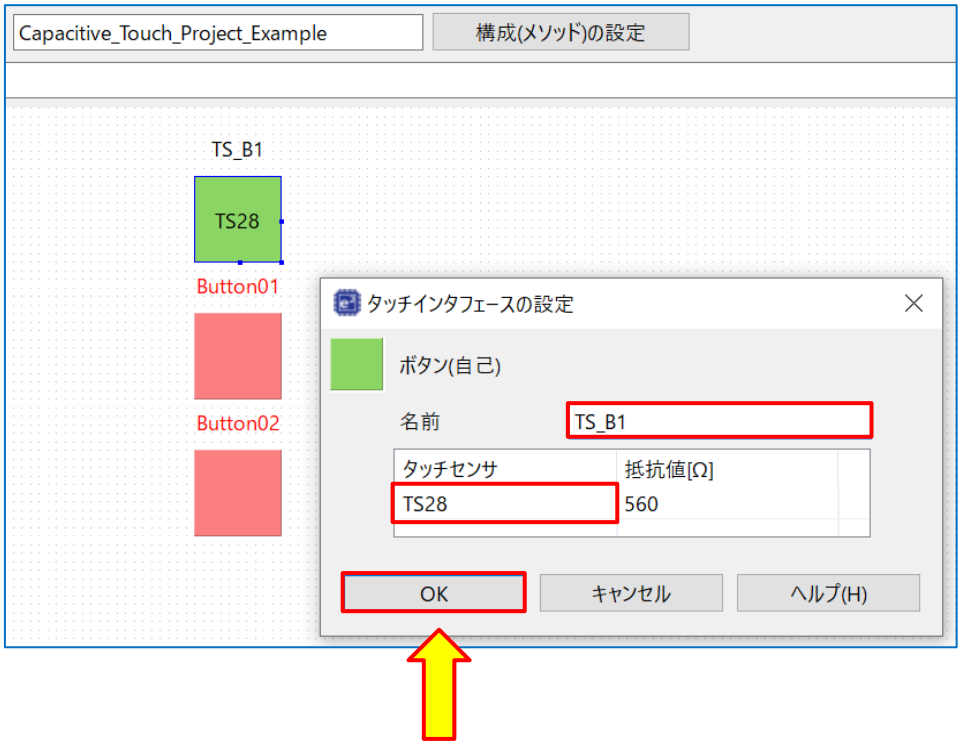


図 8-30 タッチインタフェースの設定

7. “Button00(TS_B1)”と同様に、Button01 と Button02 の設定を行います。
   タッチインタフェースの設定に問題が無い場合はエラー表示が消えます。
   RL78/G23 の場合は、Button01 に TS05、Button02 に TS07 を割り当てます。

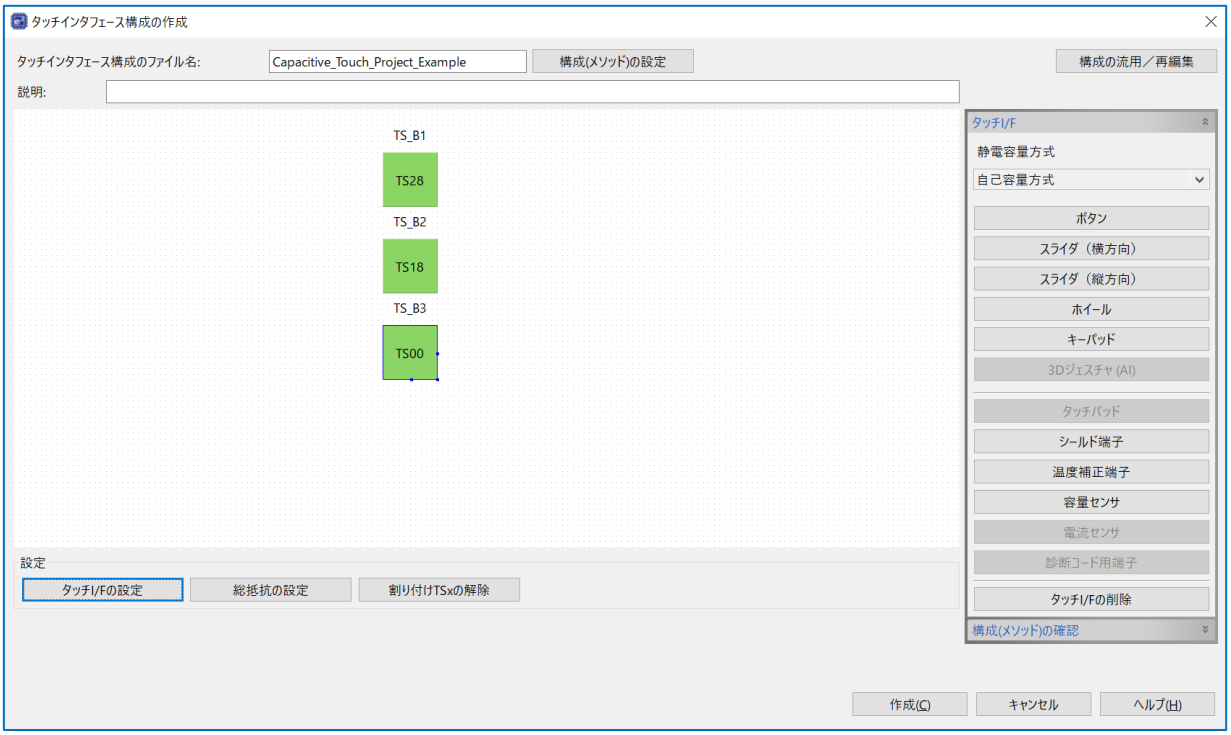


図 8-31 設定後のタッチインタフェース(ボタン 3 個)

8. ボタンと同様にホイールとスライダを設定します。
   本アプリケーション例では以下の設定で使用します。
   すべてのタッチインタフェースの配置が完了したら、[**構成(メソッド)の設定**]をクリックします。

   【注意】「設定内容に問題があります。」というエラーが再度表示されますが、手順 9 を実行すると
   エラー表示は消えます。
   【備考】本アプリケーション例では、スライダおよびホイールに割り当てたタッチセンサを、低消費モー
   ド中は MEC 機能ボタンへと切り替えて使用します。
   SMS、MEC を使用したタッチ計測(低消費モード)で使用するタッチインタフェースは
   「TS_xxMEC」と命名しています。

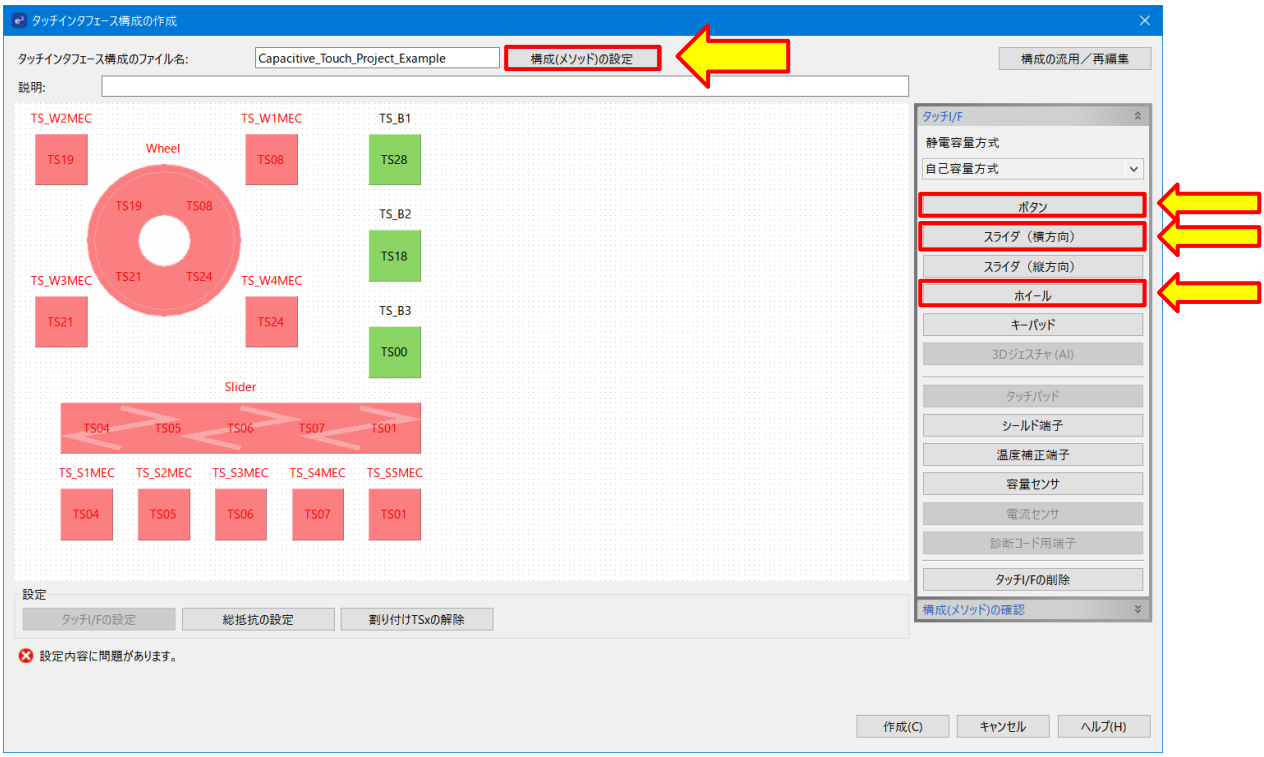


図 8-32 タッチインタフェース（ホイール，スライダ）の追加（RL78/G22）

9. “構成（メソッド）の設定”ウィンドウで自動判定機能と複数電極接続機能の設定をします。
SMS による自動判定計測を使用する場合は"自動判定（SMS)”を、複数電極接続（MEC）機能を使用する場合は"複数電極接続”を、それぞれ"**有効にする**”に設定します。

本アプリケーション例では、通常モードと低消費モードを実装するためにタッチインタフェース構成（メソッド）を分けています。[**構成（メソッド）の追加**]をクリックし、config01 の横に config02 を追加します。
各コンフィグに対して設定を行い、[**OK**]をクリックします。

— config01：低消費モードでのタッチインタフェース構成（メソッド）の機能設定
— config02：通常モードでのタッチインタフェース構成（メソッド）の機能設定

【注意】CTSU モジュールの V1.50 を使用する場合は、config01 にのみ自動判定（SMS）機能を適用できます。

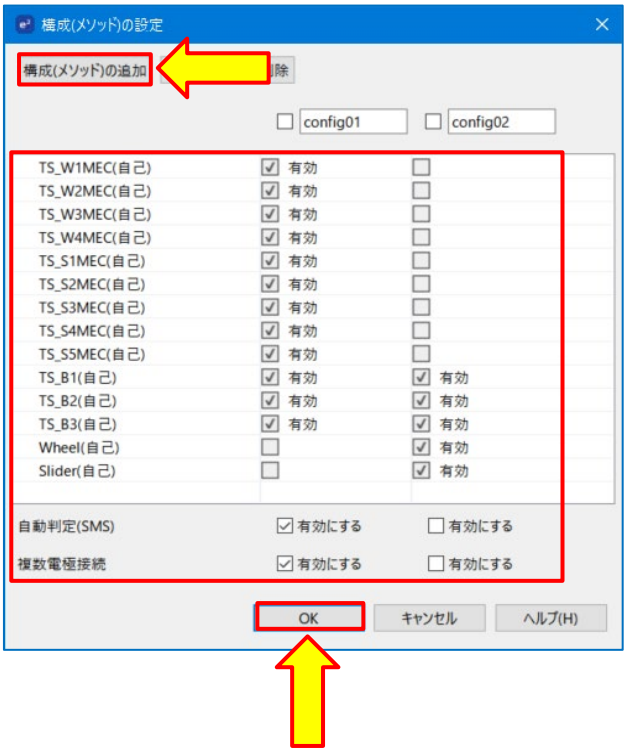


図 8-33 構成（メソッド）の設定（RL78/G22）

10.エラーが表示されないことを確認して"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、RL78/G22 の場合のタッチインタフェースの作成は完了です。

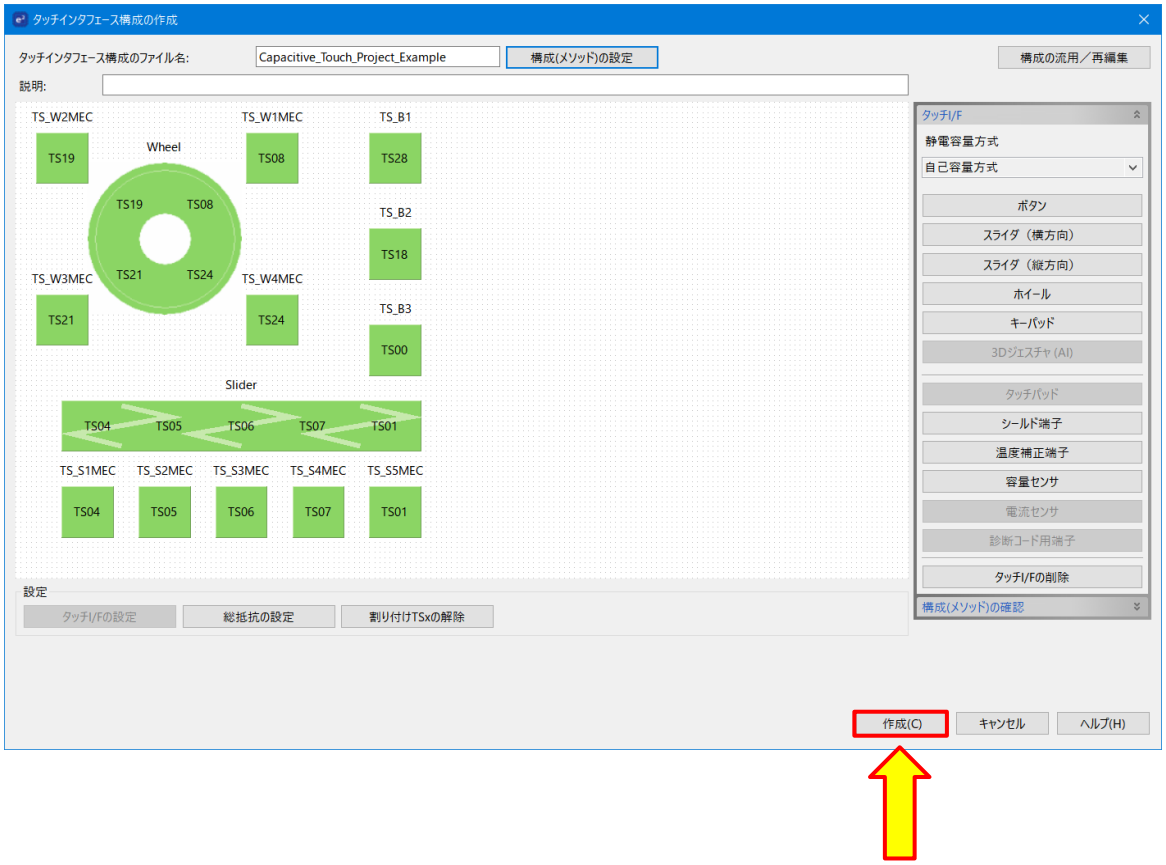


図 8-34 設定後のタッチインタフェース構成図(RL78/G22)

11.以降では RL78/G23 のタッチインタフェース構成例を示します。
基本的な設定手順および諸注意は RL78/G22 と同様です。

【注意】RL78/G22 と RL78/G23 では電極に割り当てるタッチセンサが異なります。
【備考】SMS を使用したタッチ計測（低消費モード）で使用するタッチインタフェースは「TS_xxBTN」と命名しています。

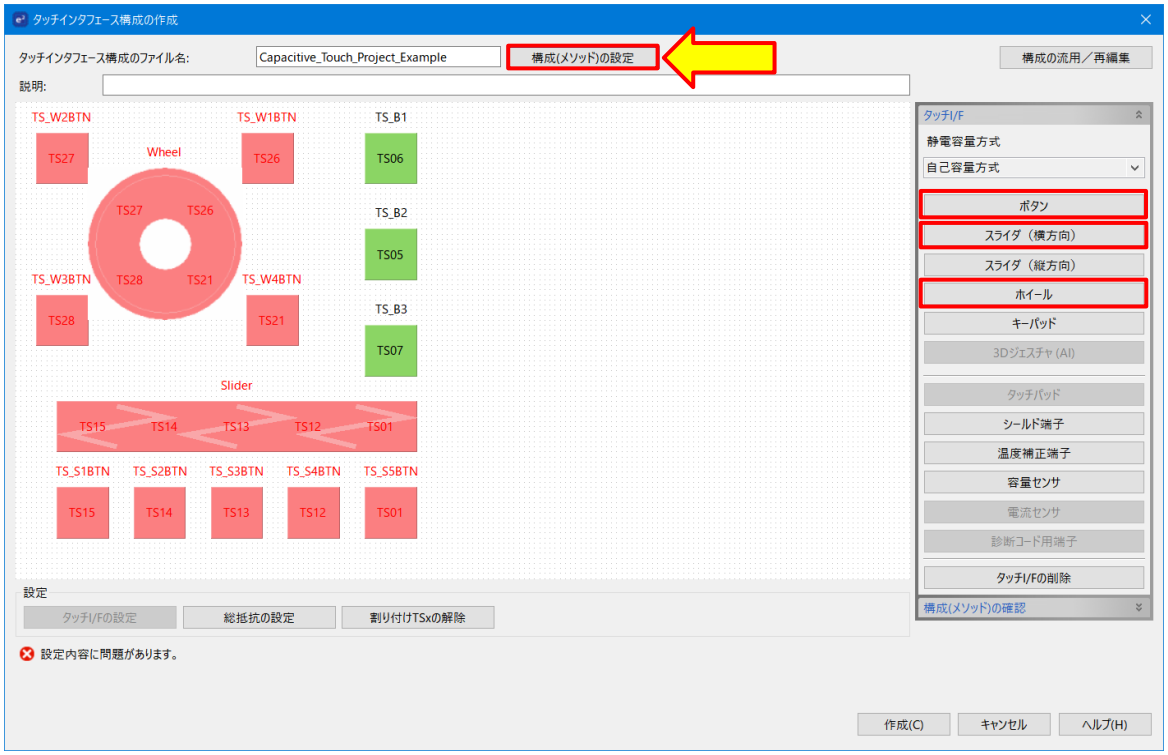


図 8-35 タッチインタフェース（ホイール，スライダ）の追加（RL78/G23）

12."構成（メソッド）の設定"ウィンドウで自動判定機能の設定をします。
本アプリケーション例では下記設定で使用します。

— config01：低消費モードでのタッチインタフェースの機能設定
— config02：通常モードでのタッチインタフェースの機能設定

【注意】CTSU モジュールの V1.50 を使用する場合は、config01 にのみ自動判定（SMS）機能を適用できます。

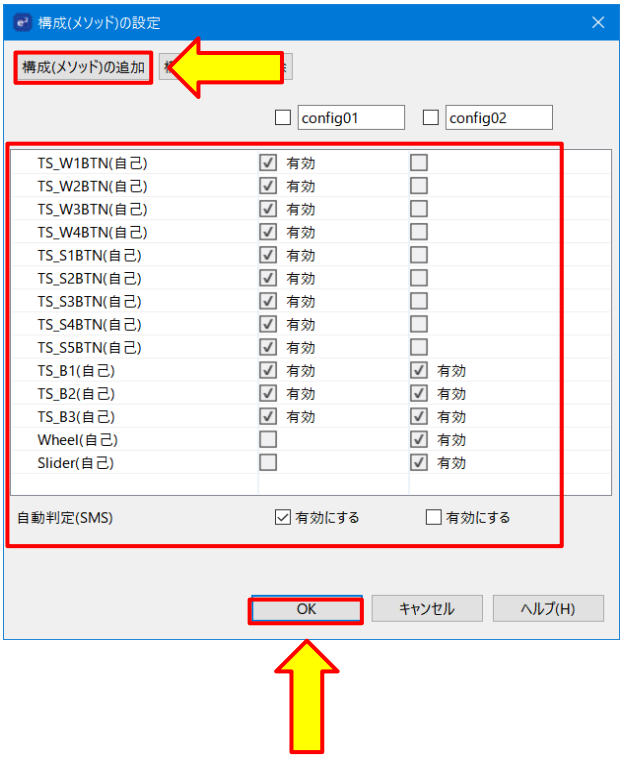


図 8-36 構成（メソッド）の設定（RL78/G23）

13.エラーがないことを確認して"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、RL78/G23 の場合のタッチインタフェースの作成は完了です。

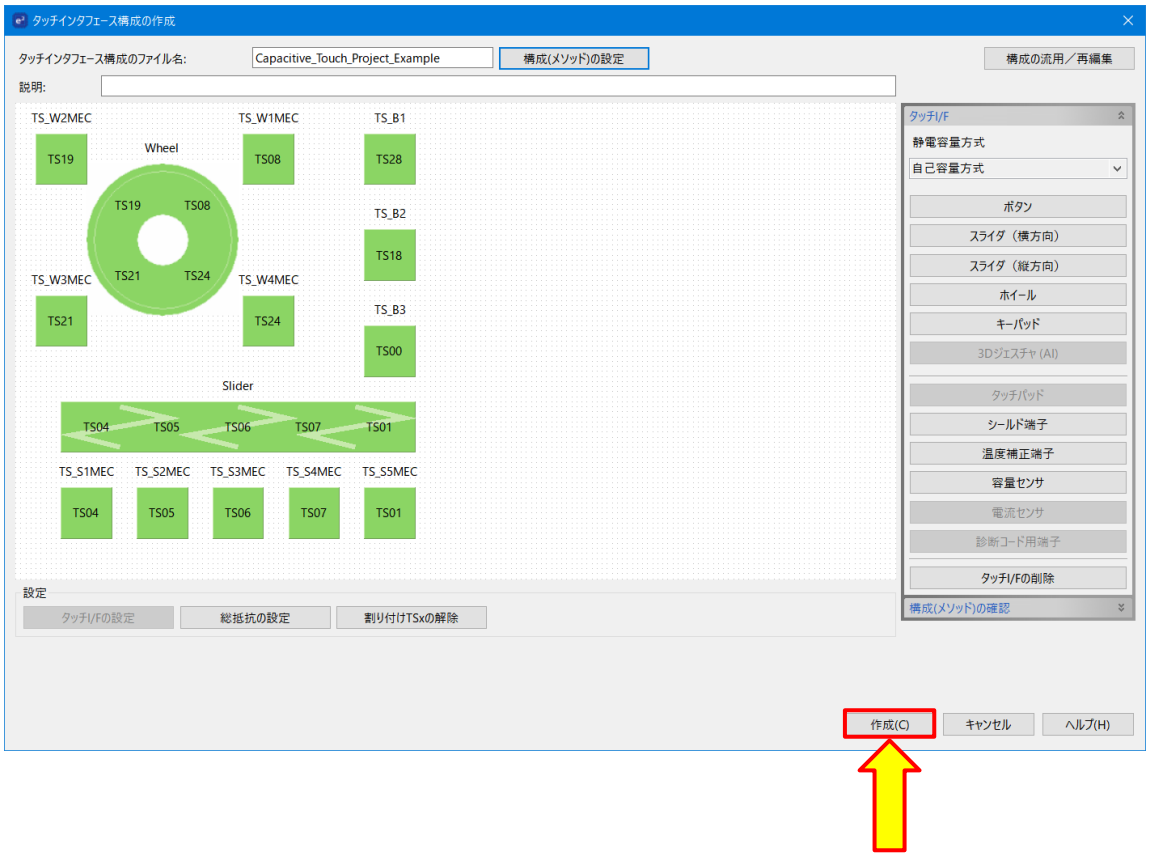
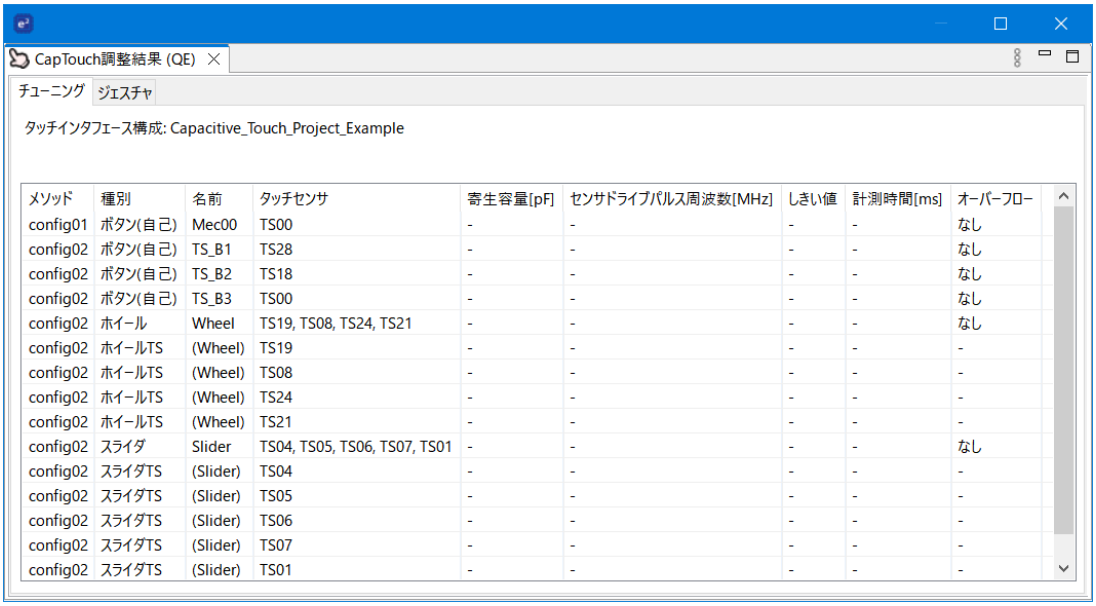


図 8-37 設定後のタッチインタフェース構成図 (RL78/G23)

14. "**CapTouch メイン (QE)**／QE V3.2.0 以降では[**CapTouch 調整結果 (QE)**]"の"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。



メソッド	種別	名前	タッチセンサ	寄生容量[pF]	センサドライブパルス周波数[MHz]	しきい値	計測時間[ms]	オーバーフロー
config01	ボタン(自己)	Mec00	TS00	-	-	-	-	なし
config02	ボタン(自己)	TS_B1	TS28	-	-	-	-	なし
config02	ボタン(自己)	TS_B2	TS18	-	-	-	-	なし
config02	ボタン(自己)	TS_B3	TS00	-	-	-	-	なし
config02	ホイール	Wheel	TS19, TS08, TS24, TS21	-	-	-	-	なし
config02	ホイールTS	(Wheel)	TS19	-	-	-	-	-
config02	ホイールTS	(Wheel)	TS08	-	-	-	-	-
config02	ホイールTS	(Wheel)	TS24	-	-	-	-	-
config02	ホイールTS	(Wheel)	TS21	-	-	-	-	-
config02	スライダ	Slider	TS04, TS05, TS06, TS07, TS01	-	-	-	-	なし
config02	スライダTS	(Slider)	TS04	-	-	-	-	-
config02	スライダTS	(Slider)	TS05	-	-	-	-	-
config02	スライダTS	(Slider)	TS06	-	-	-	-	-
config02	スライダTS	(Slider)	TS07	-	-	-	-	-
config02	スライダTS	(Slider)	TS01	-	-	-	-	-

図 8-38 [CapTouch 調整結果 (QE)] の"チューニング"パネル

15. e² studio 左上の  アイコンをクリックしてプロジェクトのビルドを開始します。

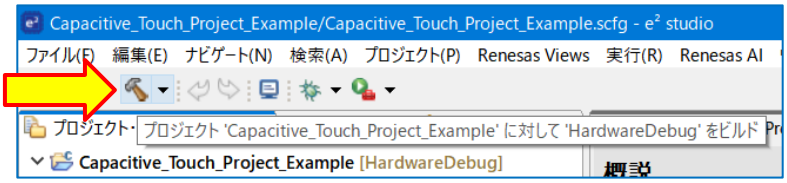


図 8-39 プロジェクトのビルド

“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。
これで静電容量タッチインタフェースの作成は完了です。

【注意】プロジェクトをビルドした際、コンソールに以下のエラー(E0562310) が出た場合は、作成したプロジェクトのプロパティから [**C/C++ ビルド**] - [**設定**] - [**Linker**] - [**入力**]を開き、ランタイム・ライブラリを使用するにチェックを入れて再ビルドしてください。

```
E0562310:Undefined external symbol "__COM_lshr" referenced in ".\src\smc_gen\r_ctsu\r_ctsu.obj"
E0562310:Undefined external symbol "__COM_llshl" referenced in ".\src\smc_gen\rm_touch\rm_touch.obj"
E0562310:Undefined external symbol "__COM_llmul" referenced in ".\src\smc_gen\r_ctsu\r_ctsu.obj"
E0562310:Undefined external symbol "__COM_slldiv" referenced in ".\src\smc_gen\r_ctsu\r_ctsu.obj"
E0562310:Undefined external symbol "__COM_sldiv" referenced in ".\src\smc_gen\r_ctsu\r_ctsu.obj"
E0562310:Undefined external symbol "__COM_sldiv" referenced in ".\src\smc_gen\rm_touch\rm_touch.obj"

Renesas Optimizing Linker Abort
makefile:113: recipe for target 'G22_RSSK_SMS_MEC_test.abs' failed
make: *** [G22_RSSK_SMS_MEC_test.abs] Error 1
"make -j8 all" terminated with exit code 2. Build might be incomplete.

13:18:09 Build Failed. 8 errors, 0 warnings. (took 18s.976ms)
```

図 8-40 コンソールウィンドウ (エラー: E0562310 発生時)

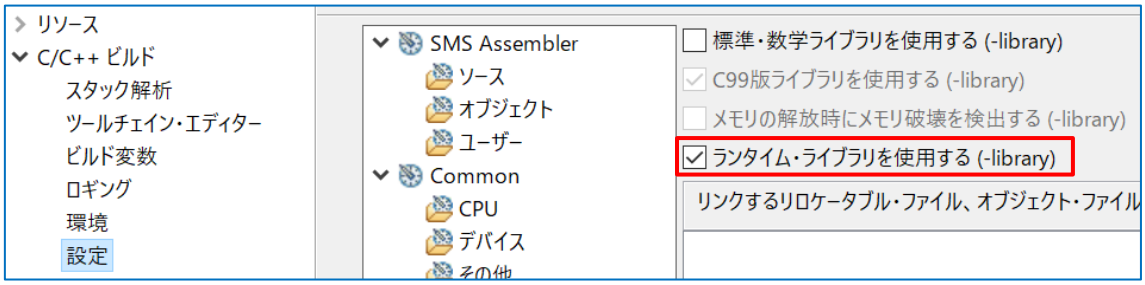


図 8-41 ランタイム・ライブラリの設定

8.4　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。

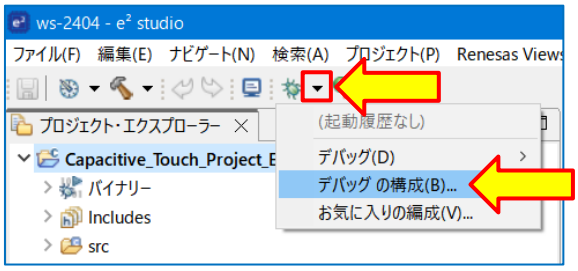


図 8-42 "デバックの構成"の選択

2. デバッグ構成ウィンドウで[**Debugger**] - [**Connection Settings**]とタブを選択します。
以下のようにターゲット・ボードとの接続の設定を行ってください。

【注意】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。なお 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。
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は、設定が異なります。

— デバック方法：E2 Lite

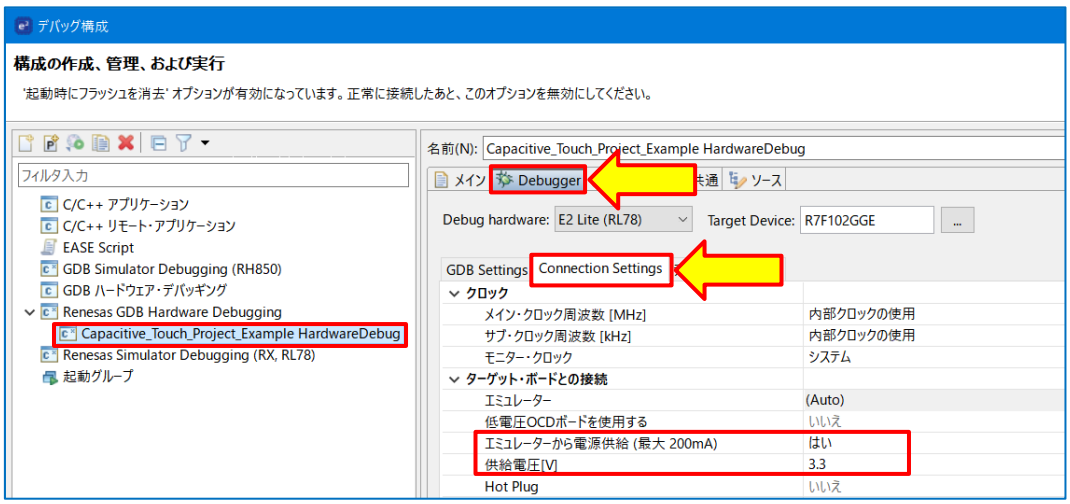


図 8-43 デバック構成の設定 (E2 Lite)

— デバック方法：COM port

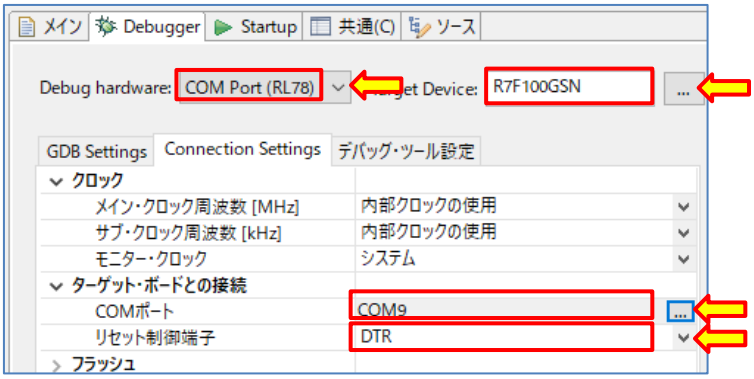


図 8-44 デバック構成の設定 (COM port)

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。

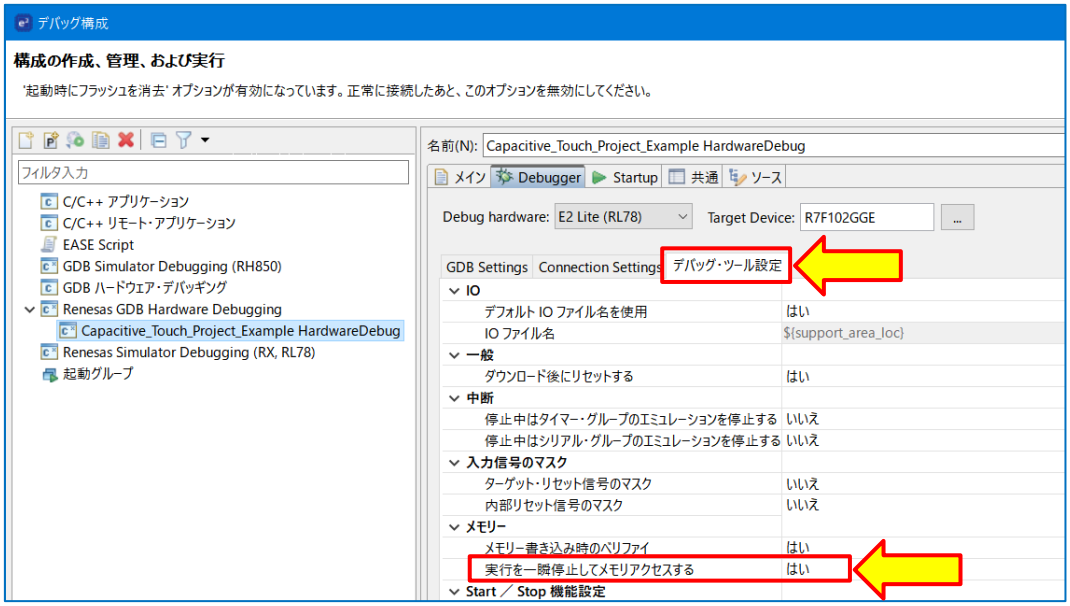


図 8-45 デバック・ツール設定

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**]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。

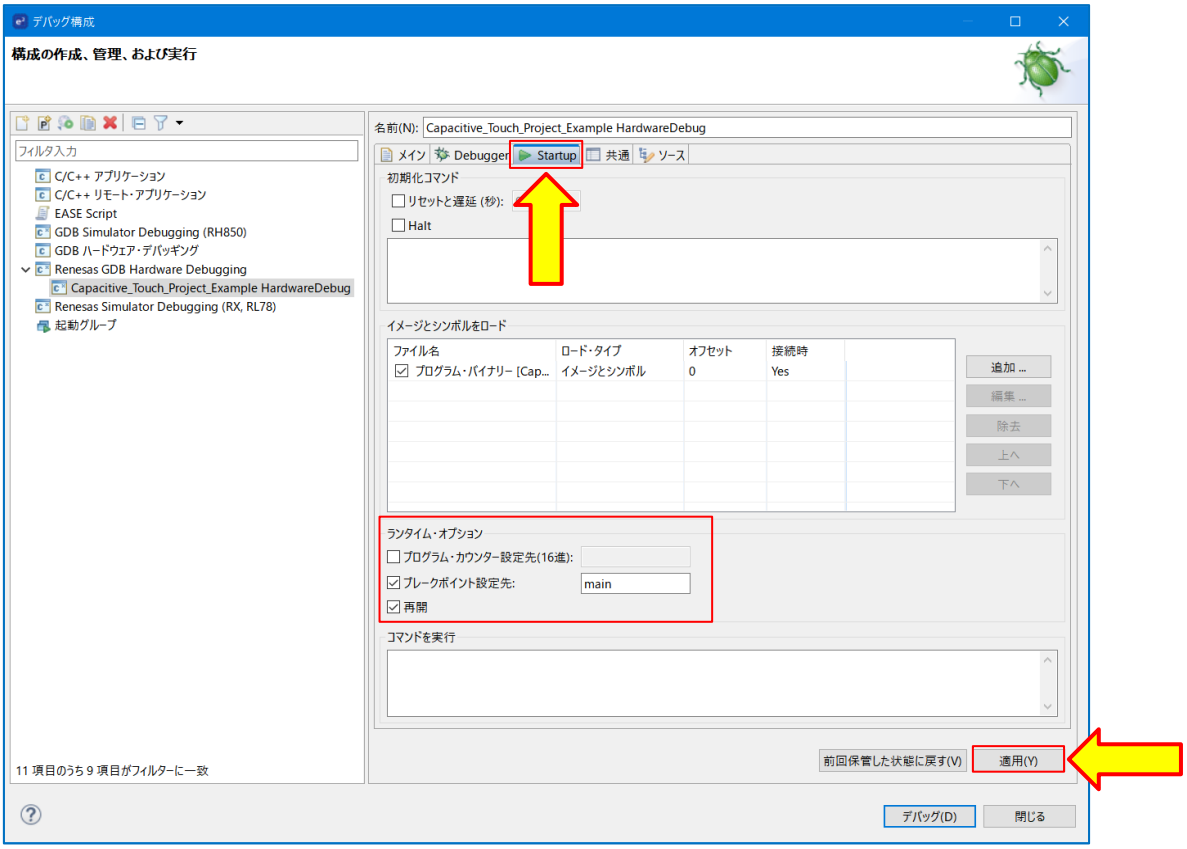


図 8-46 ランタイム・オプションの設定

8.5　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 メイン (QE)**／QE V3.2.0 以降では[**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。

【注意】 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。

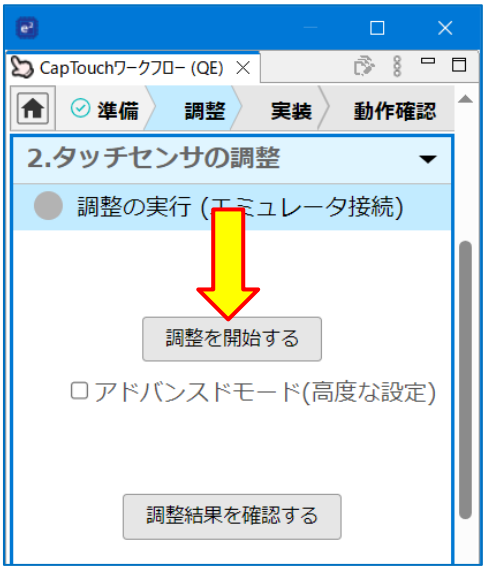


図 8-47 自動チューニングの開始

2. 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。

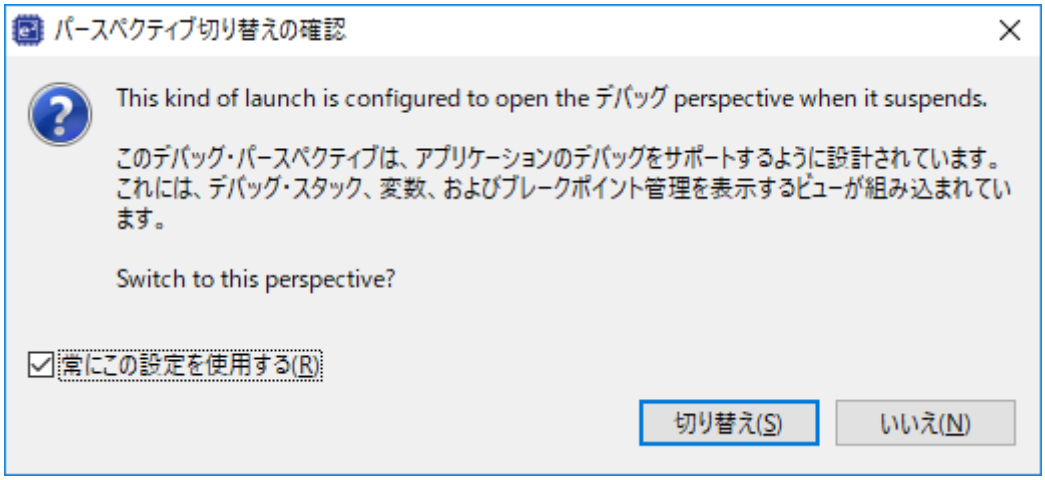


図 8-48 デバック・パースペクティブの切り替え

3. 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。

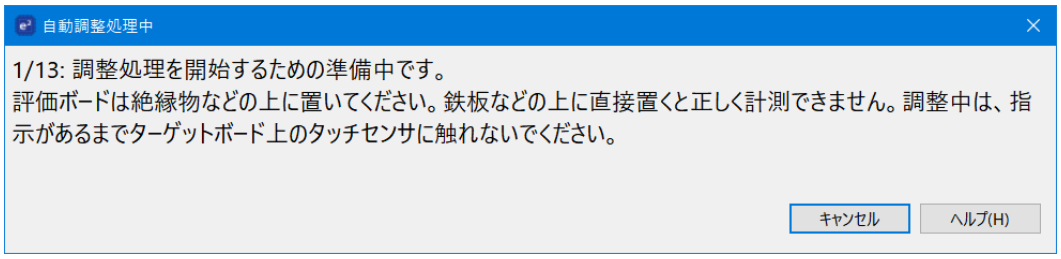


図 8-49 "自動調整処理中”ダイアログ (初期チューニングプロセス中)

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 (**Mec00**,**TS00**) 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。

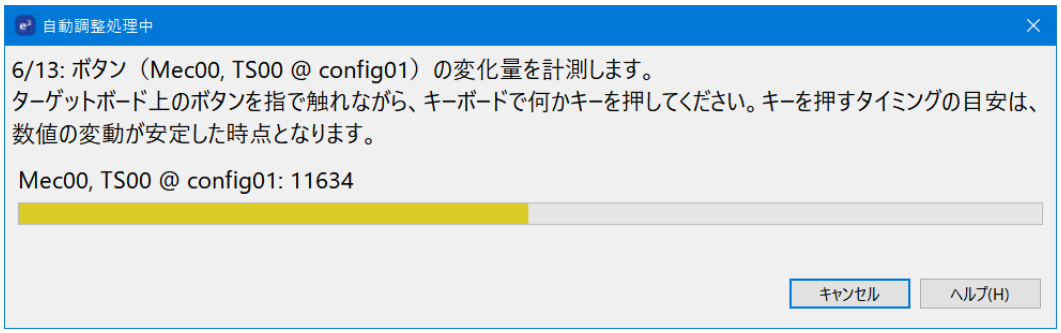


図 8-50 "自動調整処理中”ダイアログ (タッチ感度の計測中)

設定したすべてのボタンに対して、前の手順を繰り返します。

4. 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。

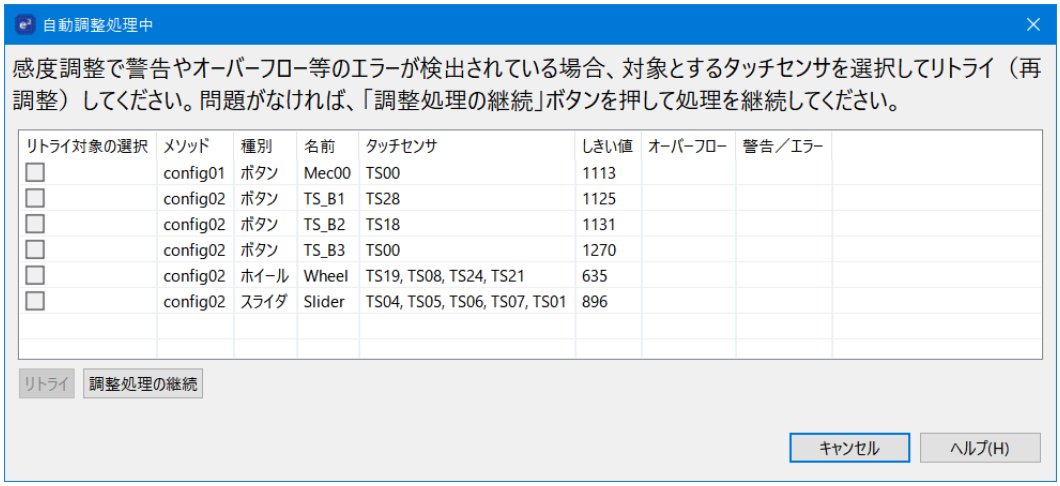


図 8-51 調整結果の表示

5. 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 メイン (QE)**／QE V3.2.0 以降では[**CapTouch ワークフロー (QE)**]”に戻ります。

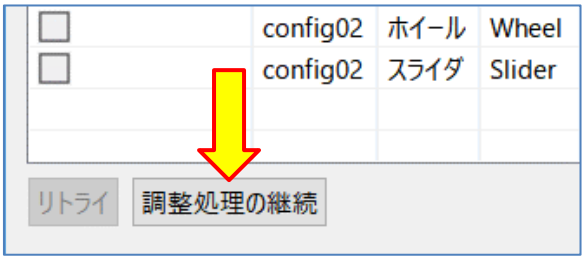


図 8-52 調整処理の継続

6. 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。以下のように外部トリガの設定を行い、[**ファイルを出力する**]をクリックします。

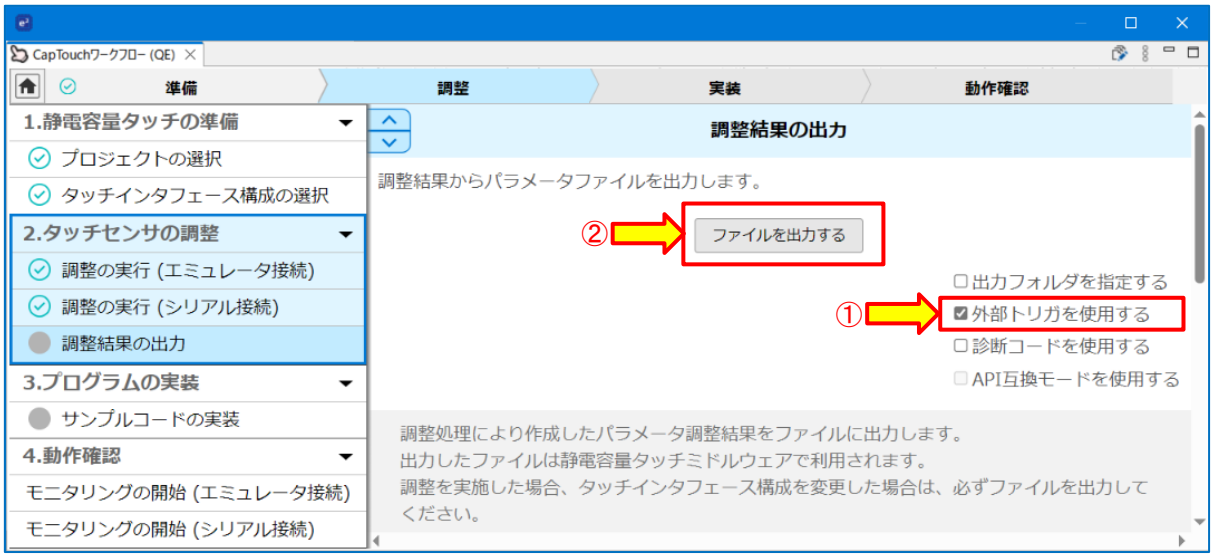


図 8-53 パラメータファイルの出力

7. “**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。

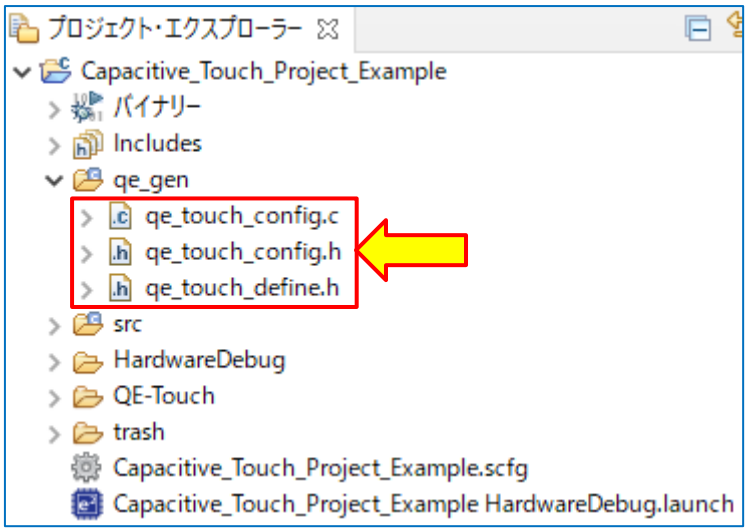


図 8-54 チューニング・パラメータのファイル出力

8. 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。

8.6　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 メイン (QE)**／QE V3.2.0 以降では[**CapTouch ワークフロー (QE)**]”の[**例を表示する**]をクリックします。

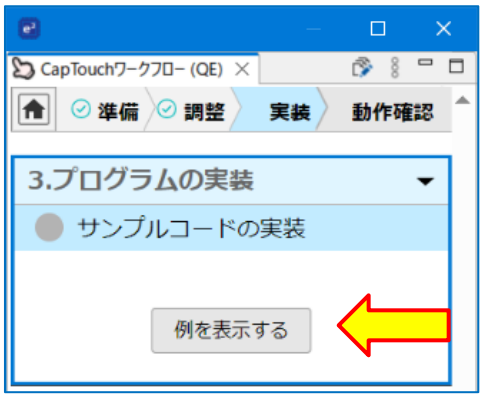


図 8-55 サンプルコード例の表示

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**]をクリックしてください。

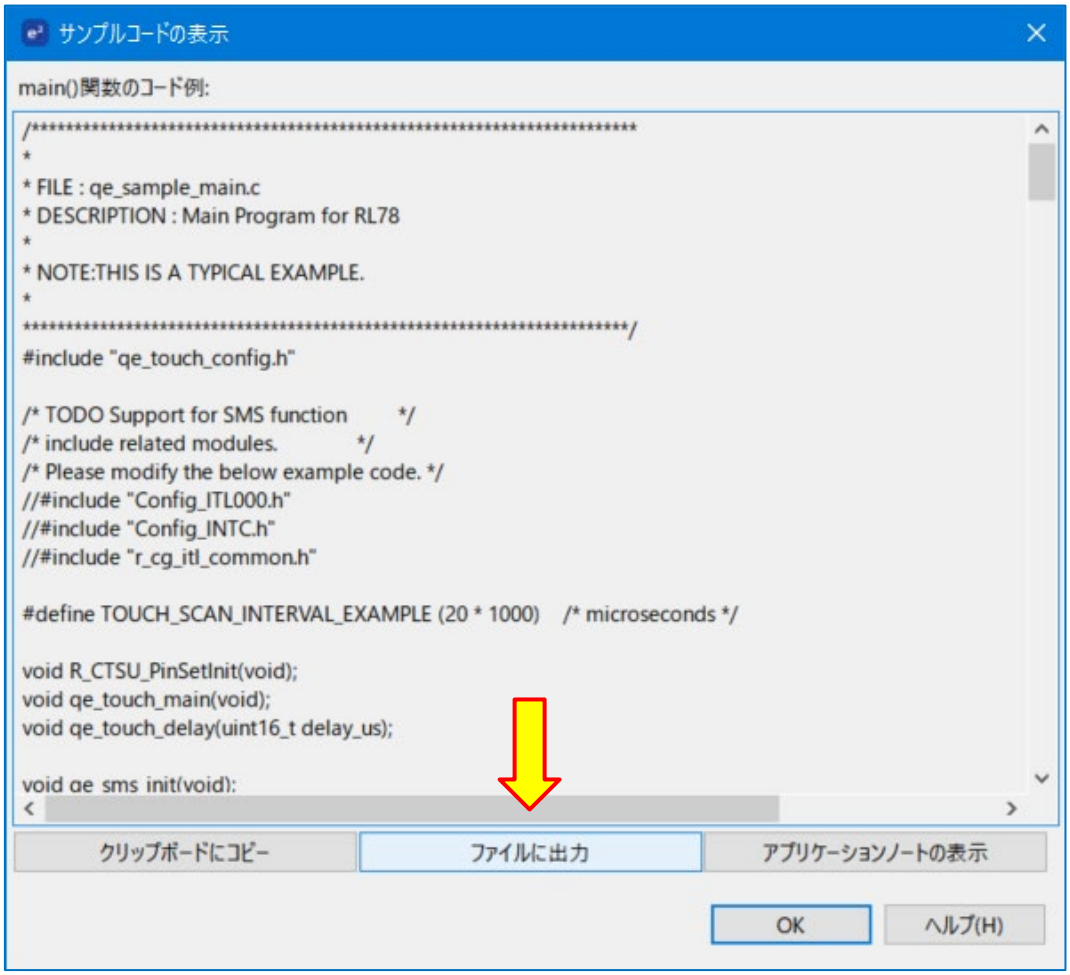


図 8-56 サンプルコードのファイル出力

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。

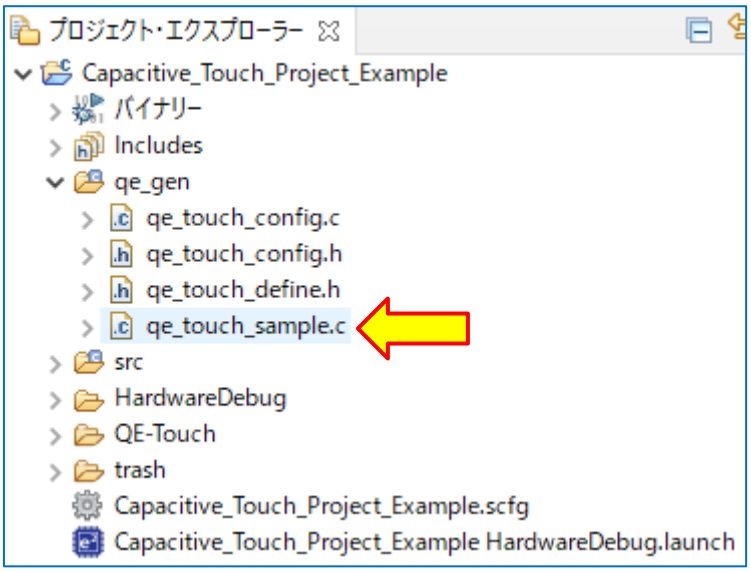


図 8-57 QE 出力コード（qe_touch_sample.c）の生成

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

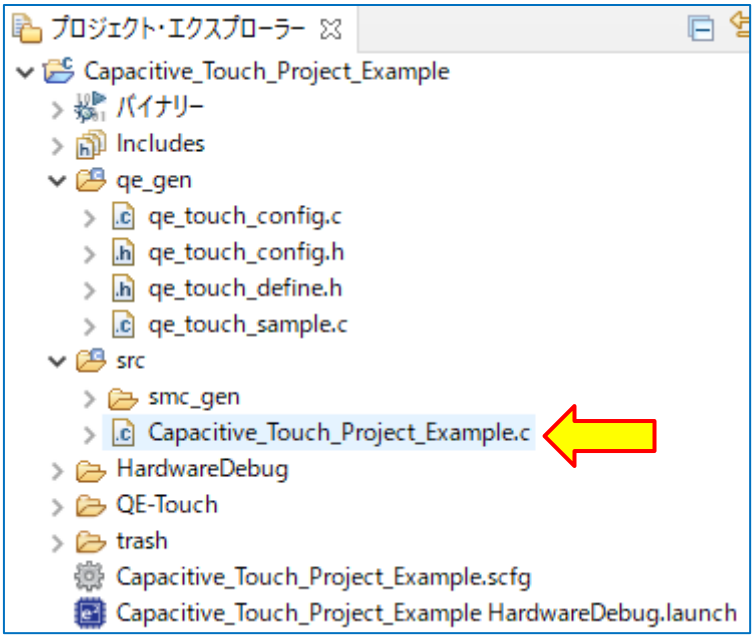


図 8-58 Capacitive_Touch_Project_Example.c の選択

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。

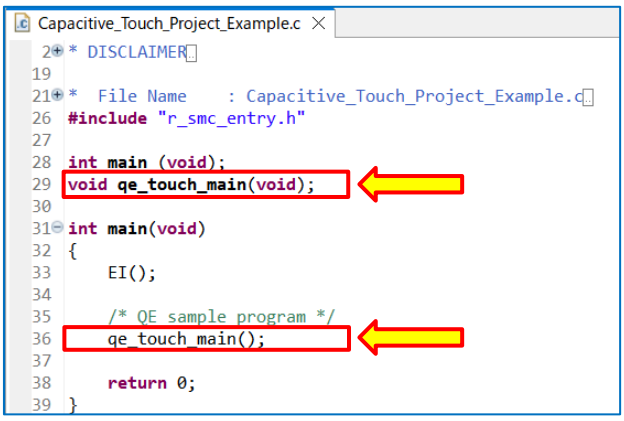


図 8-59 qe_touch_main()関数の呼び出し設定

6. QE 出力コード(qe_touch_sample.c)にインクルードファイルの設定を行います。
以下のとおりコメント設定を変更し、必要なヘッダファイルをインクルードしてください。

【注】RL78/G23 の場合、「#include "Config_INTC.h"」はコメント状態のままにしておいてください。

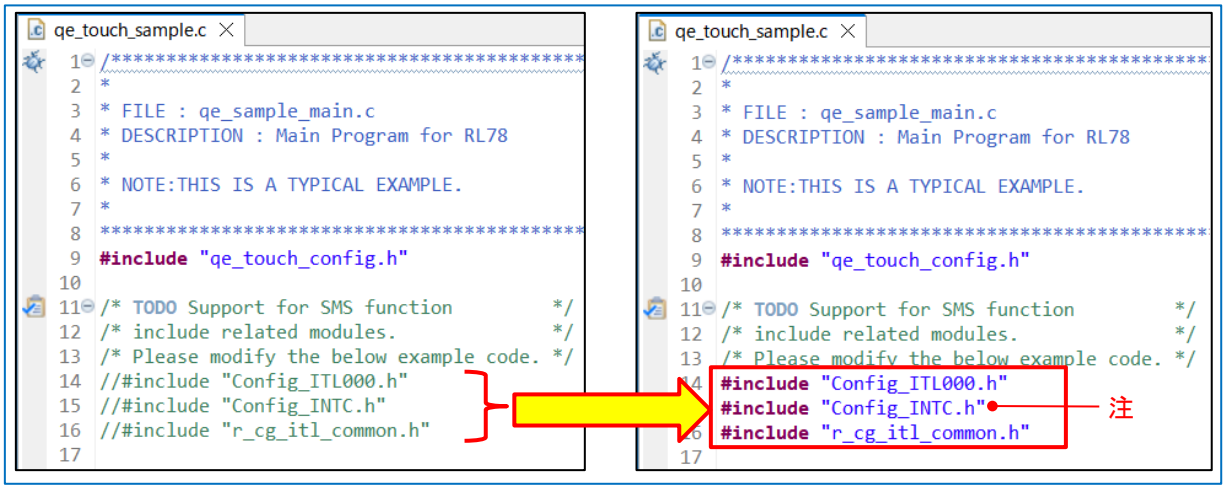
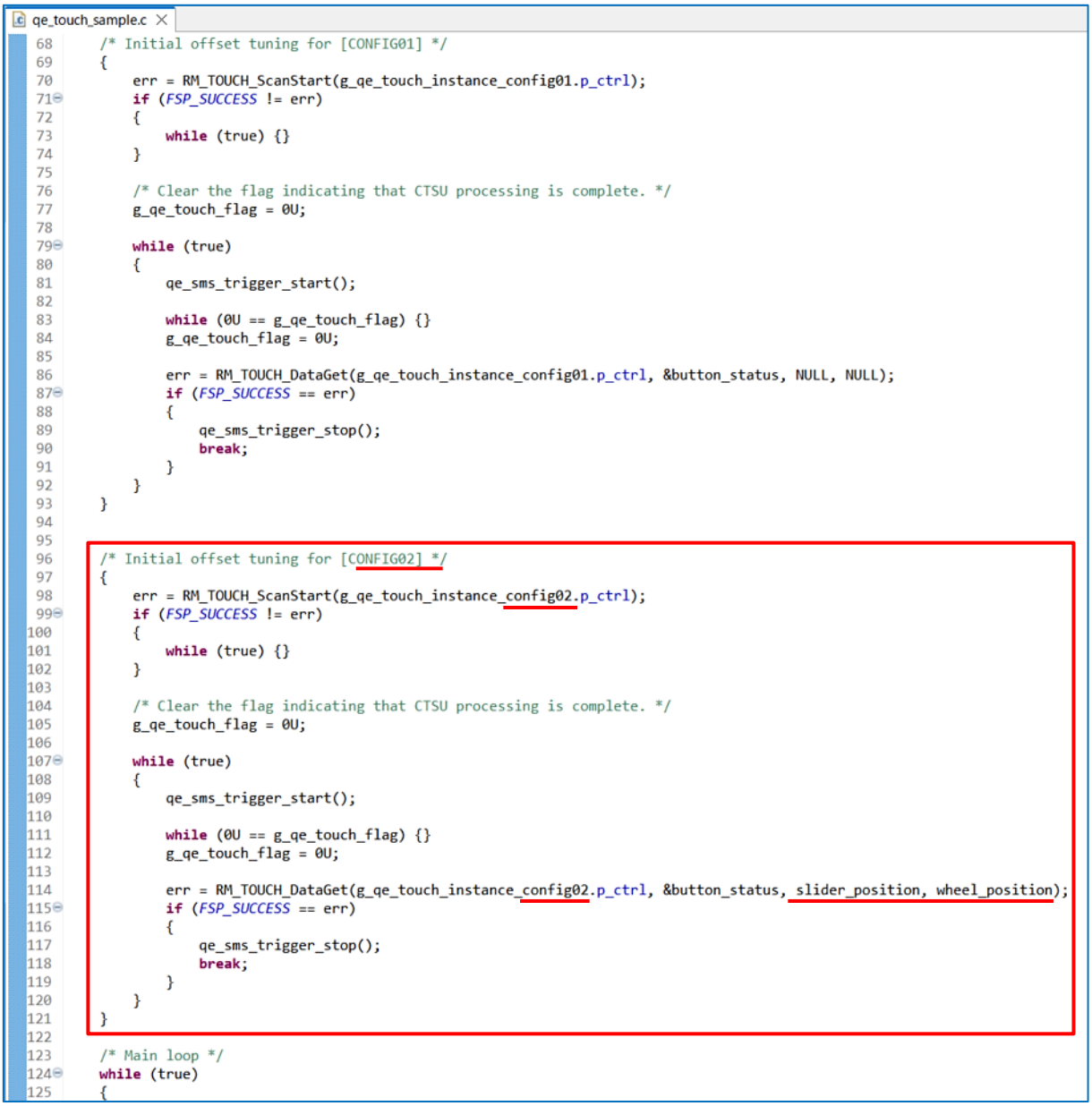


図 8-60 QE 出力コード（qe_touch_sample.c）のインクルードファイル設定

7. qe_touch_sample.c ファイルに Config2 を対象としたイニシャルオフセットチューニング処理を追加します。Config01 のコードをコピーして以下の赤枠のようにコードを追加し、赤線の箇所を Config02 用のコードに変更してください。



```
qe_touch_sample.c
68    /* Initial offset tuning for [CONFIG01] */
69    {
70        err = RM_TOUCH_ScanStart(g_qe_touch_instance_config01.p_ctrl);
71        if (FSP_SUCCESS != err)
72        {
73            while (true) {}
74        }
75
76        /* Clear the flag indicating that CTSU processing is complete. */
77        g_qe_touch_flag = 0U;
78
79        while (true)
80        {
81            qe_sms_trigger_start();
82
83            while (0U == g_qe_touch_flag) {}
84            g_qe_touch_flag = 0U;
85
86            err = RM_TOUCH_DataGet(g_qe_touch_instance_config01.p_ctrl, &button_status, NULL, NULL);
87            if (FSP_SUCCESS == err)
88            {
89                qe_sms_trigger_stop();
90                break;
91            }
92        }
93    }
94
95
96    /* Initial offset tuning for [CONFIG02] */
97    {
98        err = RM_TOUCH_ScanStart(g_qe_touch_instance_config02.p_ctrl);
99        if (FSP_SUCCESS != err)
100       {
101           while (true) {}
102       }
103
104       /* Clear the flag indicating that CTSU processing is complete. */
105       g_qe_touch_flag = 0U;
106
107       while (true)
108       {
109           qe_sms_trigger_start();
110
111           while (0U == g_qe_touch_flag) {}
112           g_qe_touch_flag = 0U;
113
114           err = RM_TOUCH_DataGet(g_qe_touch_instance_config02.p_ctrl, &button_status, slider_position, wheel_position);
115           if (FSP_SUCCESS == err)
116           {
117               qe_sms_trigger_stop();
118               break;
119           }
120       }
121   }
122
123   /* Main loop */
124   while (true)
125   {
```

図 8-61 Config02 を対象としたイニシャルオフセットチューニング処理

8. qe_touch_sample.c ファイルに Config02 計測用の外部トリガの設定を追加します。

qe_touch_sample.c

```
128         /* SMS setting for [CONFIG01] */
129         err = RM_TOUCH_SmsSet(g_qe_touch_instance_config01.p_ctrl);
130         if (FSP_SUCCESS != err)
131         {
132             while (true) {}
133         }
134
135         /* Scan start */
136         err = RM_TOUCH_ScanStart(g_qe_touch_instance_config01.p_ctrl);
137         if (FSP_SUCCESS != err)
138         {
139             while (true) {}
140         }
141
142         qe_sms_trigger_start();
143
144         /* Wait for CTSU to finish processing */
145         while (true)
146         {
147             __stop();
148
149             if (1U == g_qe_touch_flag)
150             {
151                 break;
152             }
153         }
154
155         g_qe_touch_flag = 0U;
156
157         qe_sms_trigger_stop();
158
159         err = RM_TOUCH_DataGet(g_qe_touch_instance_config01.p_ctrl, &button_status, NULL, NULL);
160         if (FSP_SUCCESS == err)
161         {
162             /* TODO: Add your own code here. */
163         }
164
165         /* for [CONFIG02] configuration */
166         err = RM_TOUCH_ScanStart(g_qe_touch_instance_config02.p_ctrl);
167         if (FSP_SUCCESS != err)
168         {
169             while (true) {}
170         }
171
172         qe_sms_trigger_start();
173
174         while (0 == g_qe_touch_flag) {}
175         g_qe_touch_flag = 0;
176
177         qe_sms_trigger_stop();
178
179         err = RM_TOUCH_DataGet(g_qe_touch_instance_config02.p_ctrl, &button_status, slider_position, wheel_position);
180         if (FSP_SUCCESS == err)
181         {
182             /* TODO: Add your own code here. */
183         }
184
185
```

図 8-62 Config02 計測用外部トリガの追加

9. qe_touch_sample.c ファイル内の qe_sms_init() 関数の設定をします。
   コメント設定を以下のように変更します。

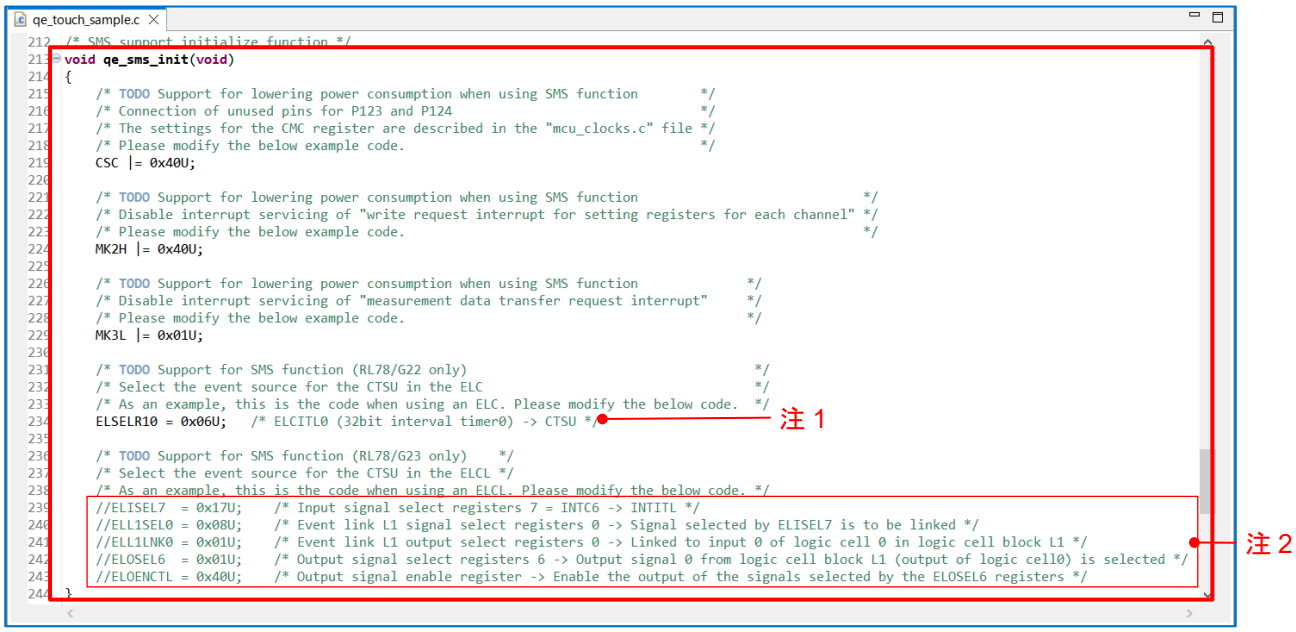


図 8-63 qe_sms_init 関数の設定

【注】 1. RL78/G23 の場合は、「ELSELR10 = 0x06U;」はコメント状態のままにしてください。
2. RL78/G23 の場合は、下記コードのコメントアウトを解除してください。
ELISEL7 = 0x17U;
ELL1SEL0 = 0x08U;
ELL1LNK0 = 0x01U;
ELOSEL6 = 0x01U;
ELOENCTL = 0x40U;

10. qe_touch_sample.c ファイル内の、qe_sms_trigger_start() 関数と qe_sms_trigger_stop() 関数の設定をします。
コメント設定を以下のように変更します。

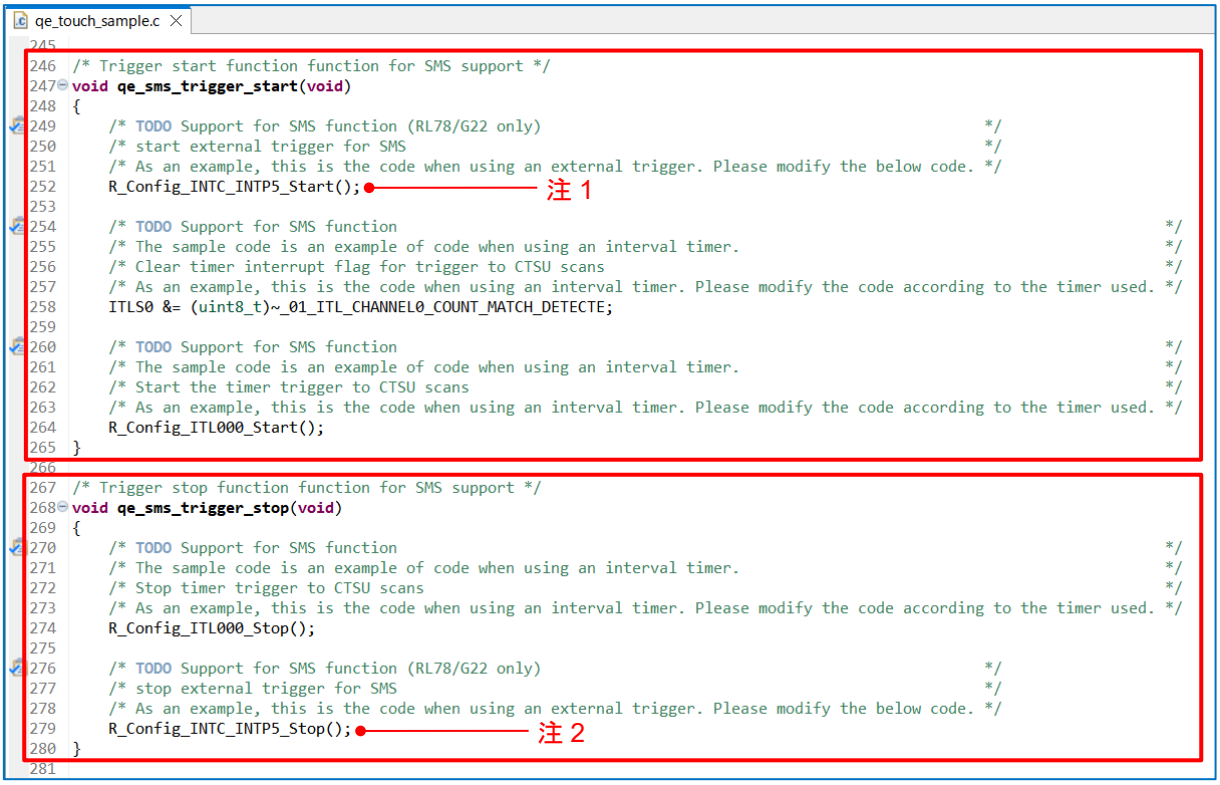


図 8-64 qe_sms_trigger_start 関数と qe_sms_trigger_stop 関数の設定

【注】1. RL78/G23 の場合、「R_Config_INTC_INTP5_Start();」はコメント状態のままにしておいてください。
2. RL78/G23 の場合、「R_Config_INTC_INTP5_Stop();」はコメント状態のままにしておいてください。

11.e^2 studio 左上の  アイコンをクリックしてプロジェクトの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がないことが確認できます。
SMS を使用した自動判定計測を活用した、静電容量タッチ低消費電力アプリケーションの開発に必要な手順は以上となります。

9. 参考ドキュメント

RL78/Gxx ユーザーズマニュアル ハードウェア編
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。)

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。)

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/G23 静電容量タッチ低消費電力ガイド(SMS 機能) (R01AN6670)
アプリケーションノート 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
https://www.renesas.com/qe-capacitive-touch

お問い合わせ
http://www.renesas.com/contact/

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2024.07.22	-	初版発行

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。