Renesas Synergy

QEとSSP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、Synergy MCU グループ を使用した静電容量タッチセンシングを応用したアプリケーションの作成に必要な手順を説明します。

動作確認デバイス

静電容量式タッチセンサユニット(CTSU)をサポートする Renesas Synergy

目次

1. 概要 3

2. 関連ドキュメント 3

3. 開発手順の概要 3

4. 開発ツールとソフトウェアコンポーネント 3

5. アプリケーション例の概要 4

6. 静電容量タッチアプリケーション開発手順 4
6.1 プロジェクト作成 4
6.2 Synergyスマート・コンフィグレータによるモジュール追加 6
6.3 静電容量タッチインタフェース作成 19
6.4 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 24
6.5 QE for Capacitive Touch [RA,RL78,Synergy]を使用した静電容量タッチセンサ・チューニング 25
6.6 アプリケーションにqe_touch_main() を追加 29
6.7 [式]ウィンドウとQE for Capacitive Touch [RA,RL78,Synergy]によるモニタリング 31
6.8 シリアル通信を利用したQE for Capacitive Touch [RA,RL78,Synergy]によるモニタリング 39

7. 変更後のqe_touch_sample.cのリスト 42

ホームページとサポート窓口 44

改訂記録 45

1. 概要

本アプリケーションノートでは、S3A7 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方法を以下の手順を説明します。

- S3A7 MCU ボードを使用した Synergy 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 [RA,RL78,Synergy]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. 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、e² studio /Synergy スマート・コンフィグレータ、Synergy Software Package (SSP)のドライバ/ミドルウェア、QE for Capacitive Touch [RA,RL78,Synergy]のヘルプ(e² 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. 開発手順の概要

以下は、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です。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。

1. 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。
2. Synergy 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。
3. QE for Capacitive Touch [RA,RL78,Synergy]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。
4. QE for Capacitive Touch [RA,RL78,Synergy]を使用してプロジェクトをチューニングします。
5. 必要な SSP 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。
6. QE for Capacitive Touch [RA,RL78,Synergy]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。

4. 開発ツールとソフトウェアコンポーネント

このプロジェクトでは以下の開発環境を使用します。

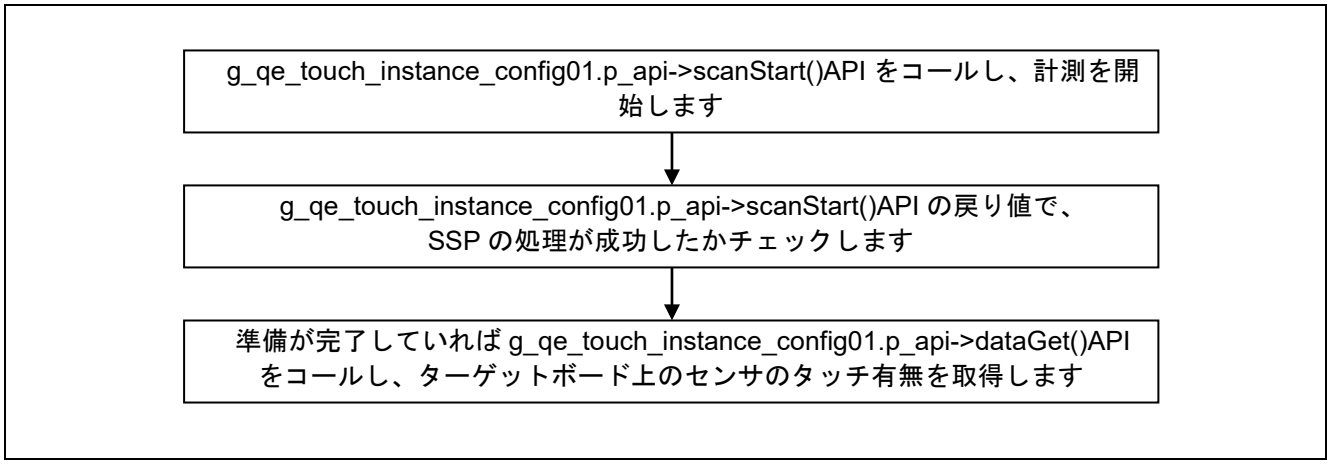
表 1　開発環境

開発ツール ソフトウェアコンポーネント	S3A7(CTSU)
評価キット・使用ボード	静電容量式タッチアプリケーション AE-CAP1 (YSAECAP1S11) ・AE-CAP1-S3 V1.1 ・Self-capacitance Button/Wheel/Slider Board (以降”BWS”という)
統合開発環境 e² studio	2021-04 以降
GNU ARM Embedded compiler	9.2.1.20191025 以降
Renesas QE for Capacitive Touch [RA,RL78,Synergy]	1.4.0 以降
Synergy Software Package (SSP)	2.0.0 以降

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。

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「7 変更後のqe_touch_sample.cのリスト」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手順

6.1 プロジェクト作成

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] - [Renesas Synergy] を選択し、新規プロジェクトの作成を開始します。

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

4. 次の[Project Configuration (Synergy C Executable Project)]ダイアログで、[Project name]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[Project Configuration (Synergy C Executable Project)]ダイアログでは、以下を選択します。

表 2　デバイス、ツールチェーン選択

項目	S3A7(CTSU)
SSP version	V2.0.0 以降
Board	Custom User Board (Any Device)
Device	R7FS3A77C3A01CFB
Toolchain	GNU ARM Embedded
Toolchain version	9.2.1.20191025 以降
Debugger	J-Link ARM

【注】：”Device”の選択には[…]ボタンの押下で表示されるデバイスを使用します。

今回使用する”Board”は、”S3” > “S3A7” > “S3A7 – 144Pin”から選択します。

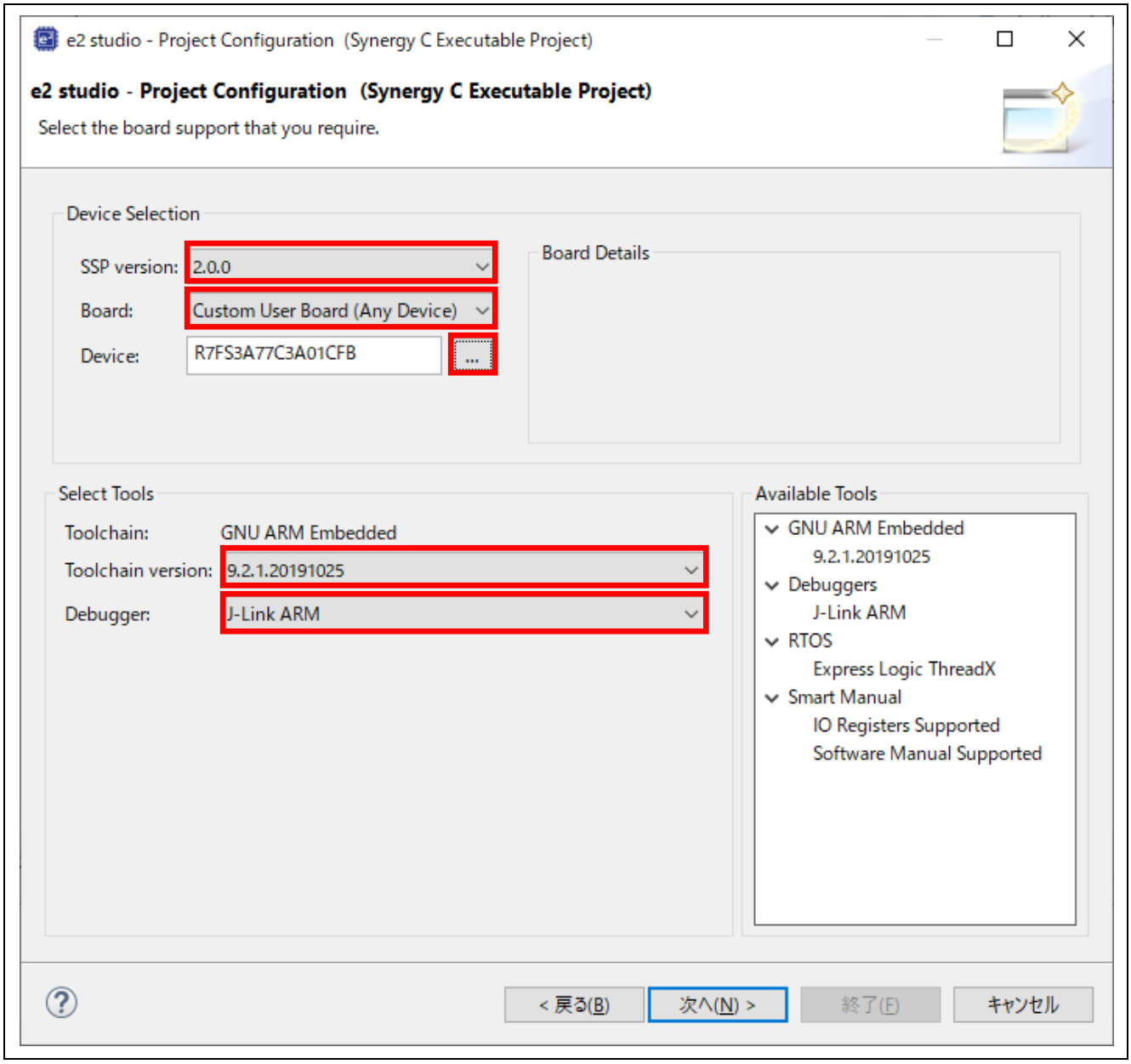


図1　デバイス・ツール設定

6. 完了したら、[次へ]をクリックします。

7. 次の[e² studio – Project Configuration (Synergy C Executable Project)]ダイアログが表示されたら、”BSP”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 Synergy スマート･コンフィグレータが表示され、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　Synergy スマート・コンフィグレータによるモジュール追加

1. e² studio の中央下部のタブから[BSP]タブを選択し、BSP 構成を表示します。

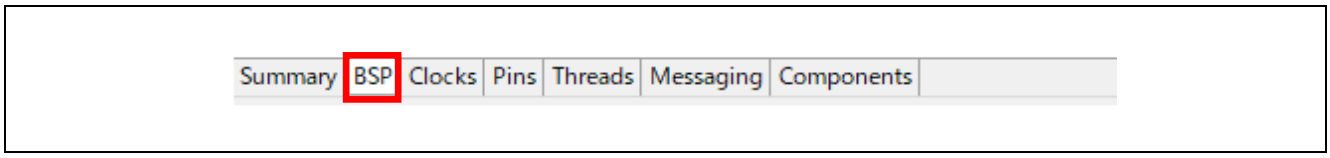


図 2　[BSP]タブ

2. 画面左下に表示される[プロパティ] – [Settings] – [Synergy Common] – [MCU Vcc (mV)]から電源供給電圧を設定します。このアプリケーション例では”3300” mV に設定しています。

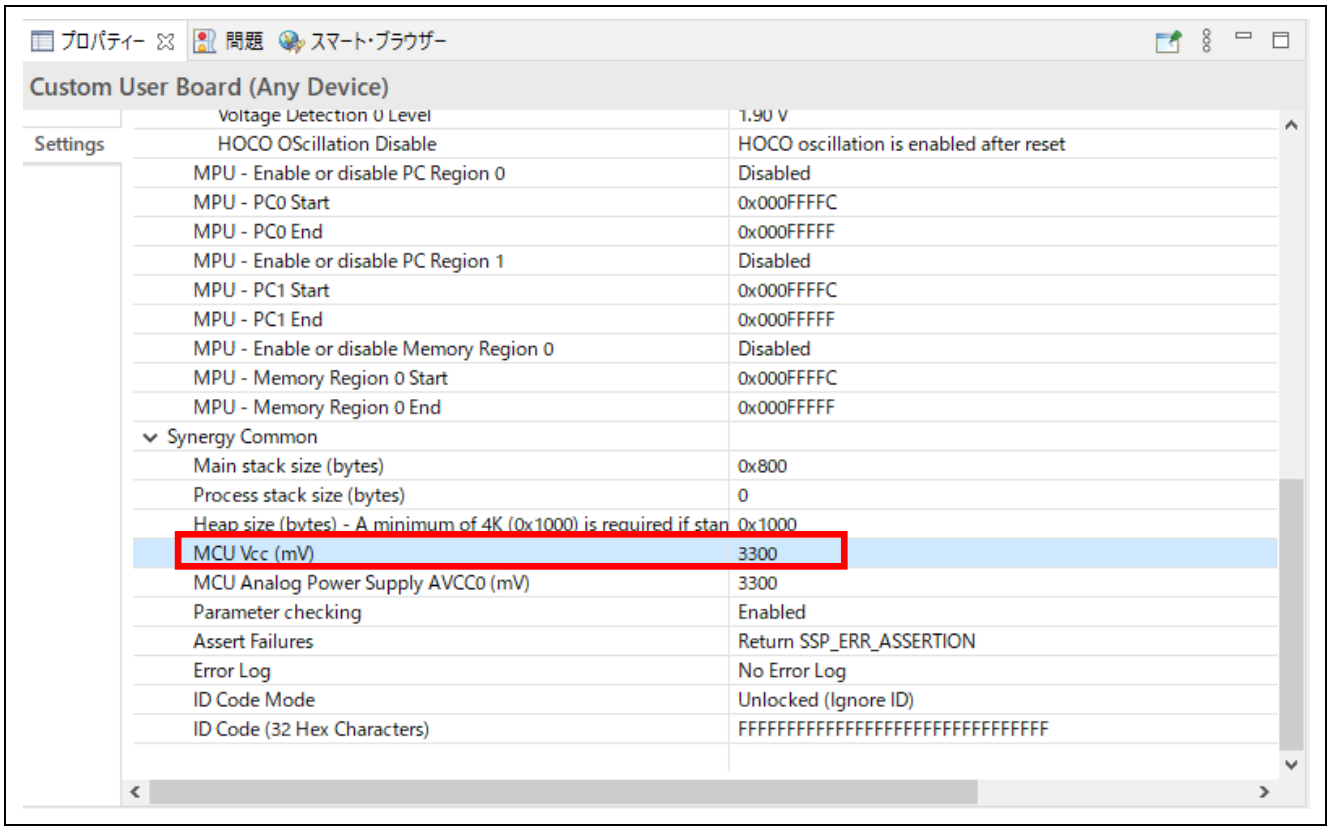


図 3　[BSP]タブ 電源供給電圧

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

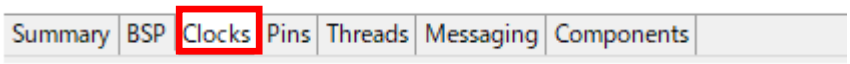


図 4　[Clocks]タブ

4. このアプリケーション例では、以下を選択します。

表 3　クロック設定

項目	S3A7 (CTSU)
PLL Src	XTAL
Clock Src	PLL
PCLKB	PCLKB Div /2

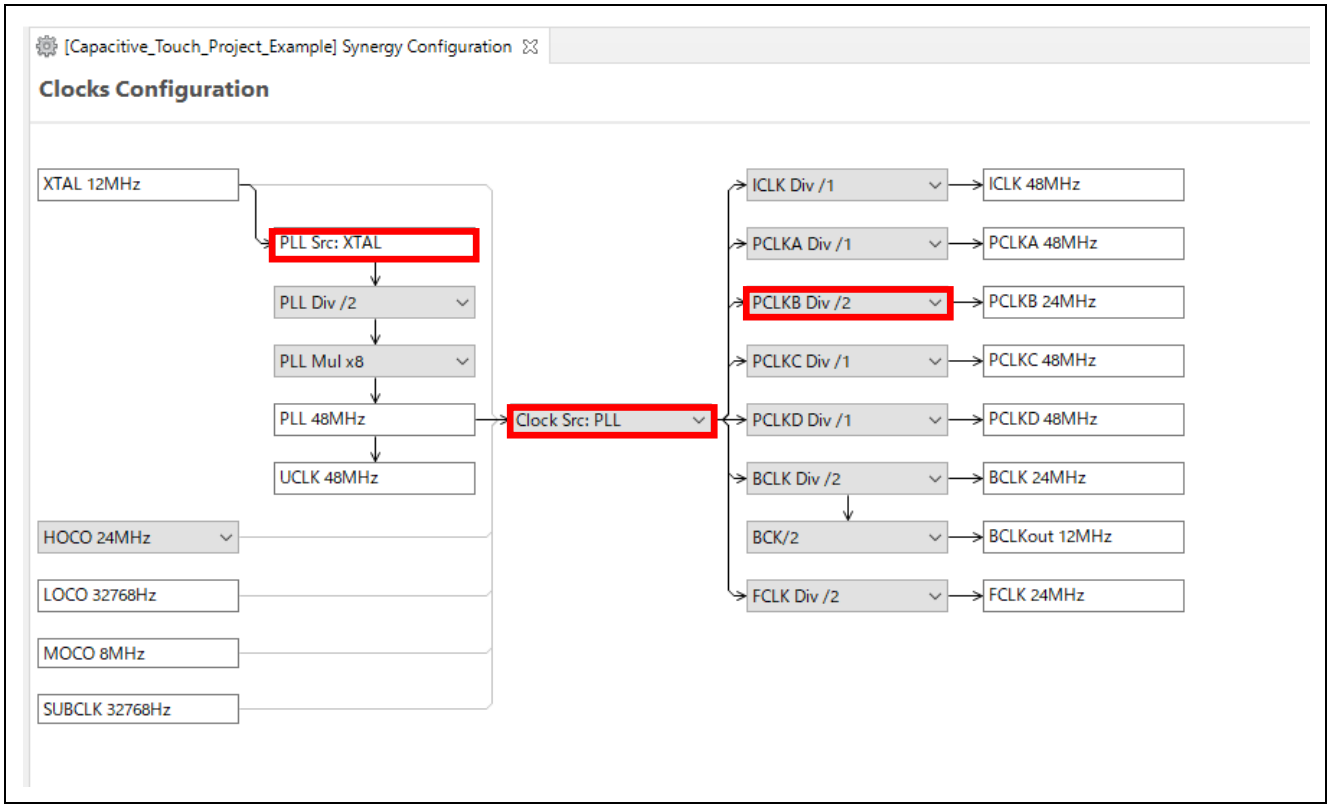


図 5　[Clocks]タブ クロック設定

5. 次に[Pins]タブに移動します。MCU のセンサポートと”BWS”ボードを接続するための割り当てを行います。

【注】：プロジェクト作成時に選択した[Device Selection]の[Board]の項目によっては、デフォルトで TS 端子が設定済みです。

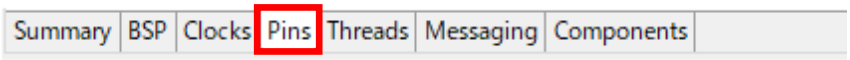


図 6　[Pins]タブ

6. [Pin Selection] - [Peripherals] - [Input:CTSU]を開き、[CTSU0]を選択します。

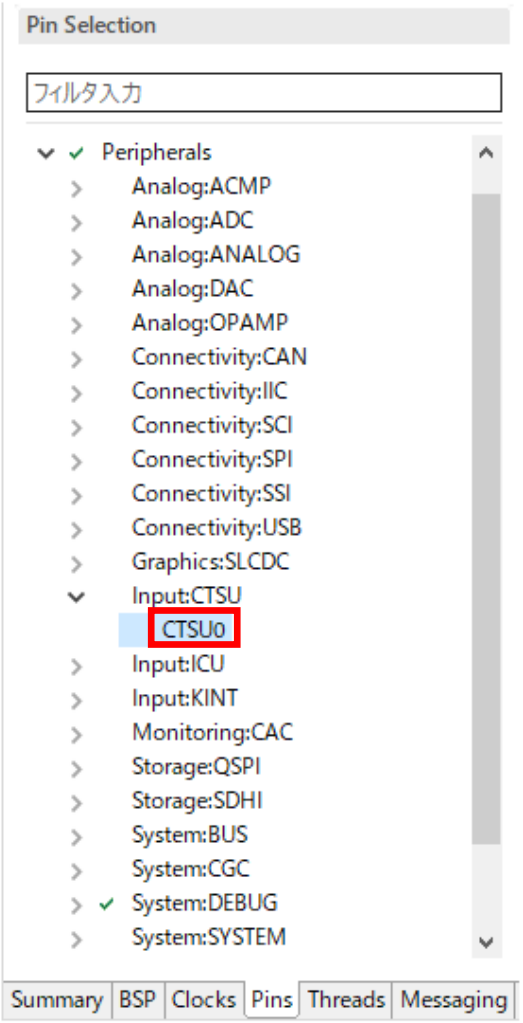


図 7　[Pins]タブ Peripherals 選択

7. [Pin Configuration] - [Operation mode]を”Disabled”から”Enabled”に変更します。

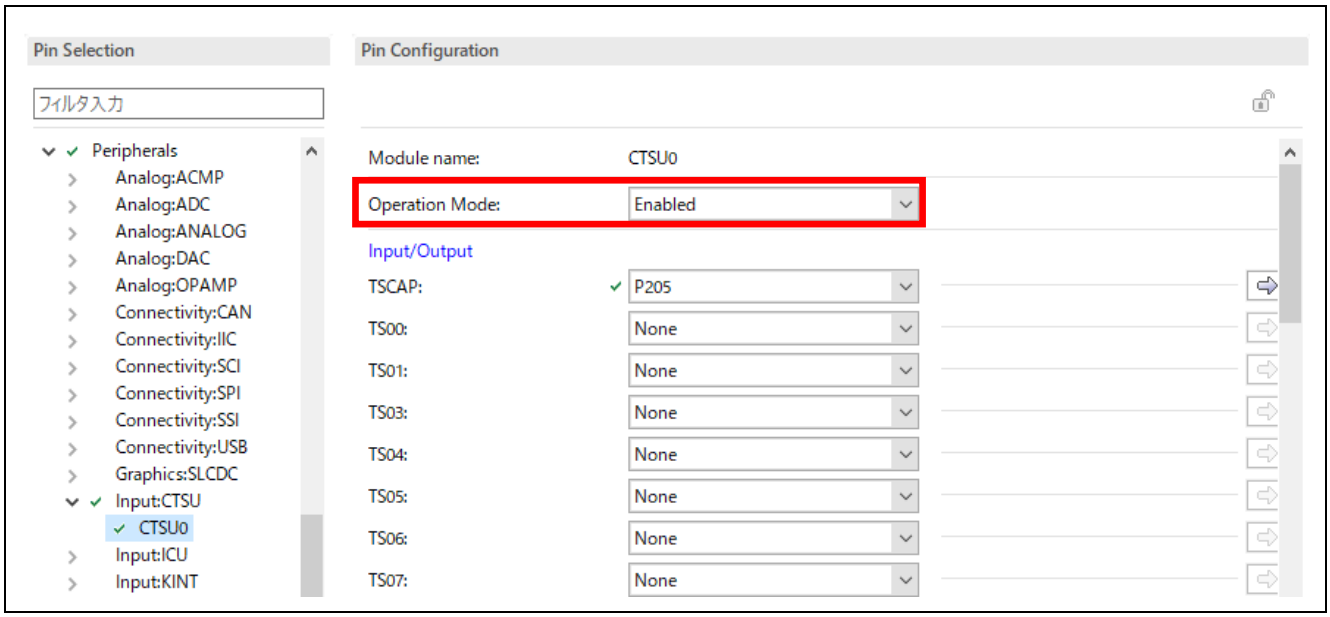


図 8　[Pins]タブ [Operation Mode]

8. アプリケーション例では、ボタン 1 つを作成します。このときに使用する静電容量タッチインタフェースに使用する以下の TS 端子設定を”None”から変更し、プルダウンメニューからポートを指定します。

- TSCAP 端子
- TS31 端子

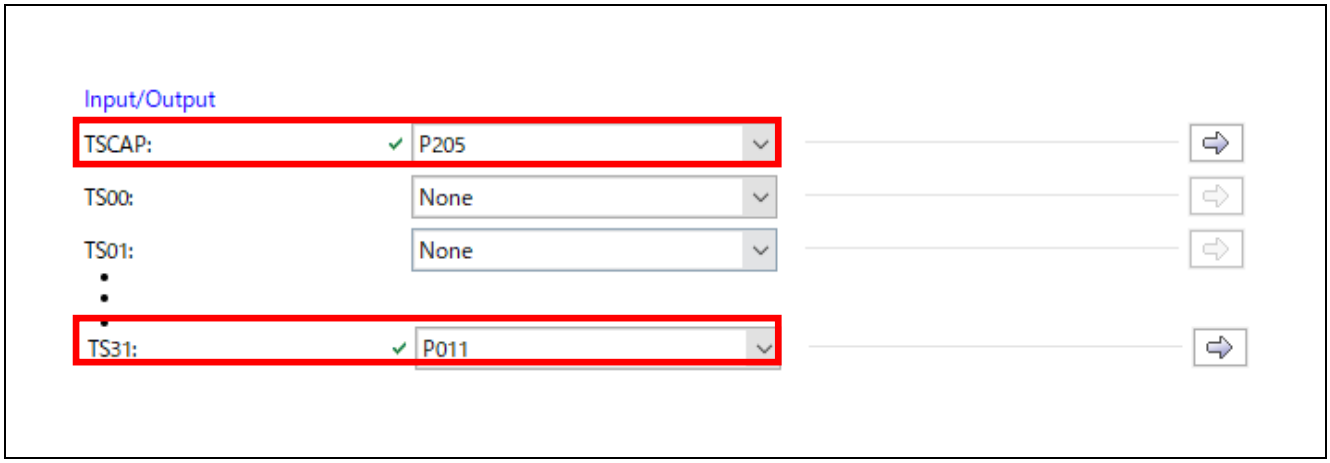


図 9　[Pins]タブ TS 端子設定

9. [Pin Selection] - [Peripherals] - [Connectivity:SCI]を開き、[SCI9]を選択します。

【注】： 9～10項はシリアル通信によるモニタリングを使用する場合のみ設定してください。シリアル通信によるモニタリングを使用しない場合は11項へ進んでください。

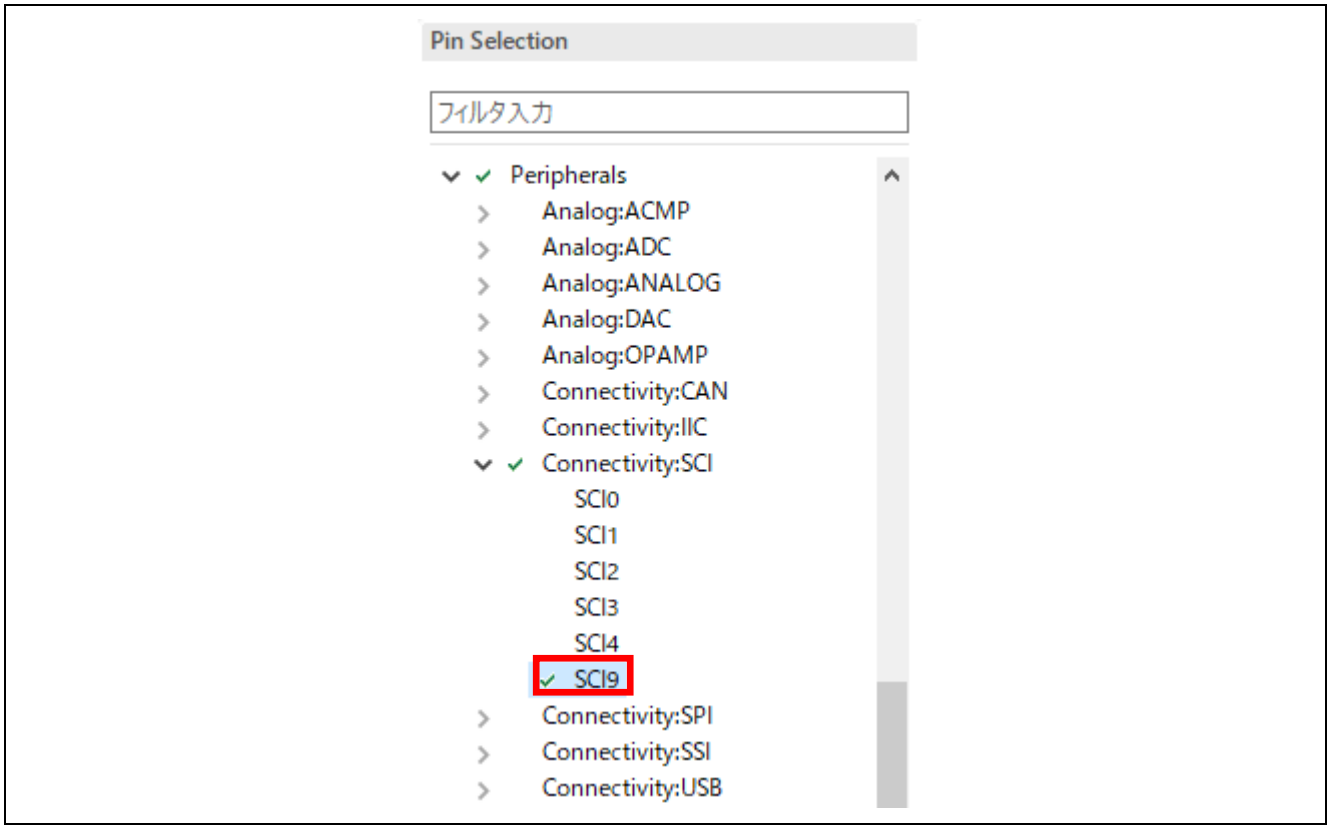


図 10　[Pins]タブ SCI9 選択

10. [Pin Configuration] – [Operation mode]を”Disabled”から”Asynchronous UART”に変更します。また、[TXD]に”P203”、[RXD]に”P202”が選択されていることを確認します。

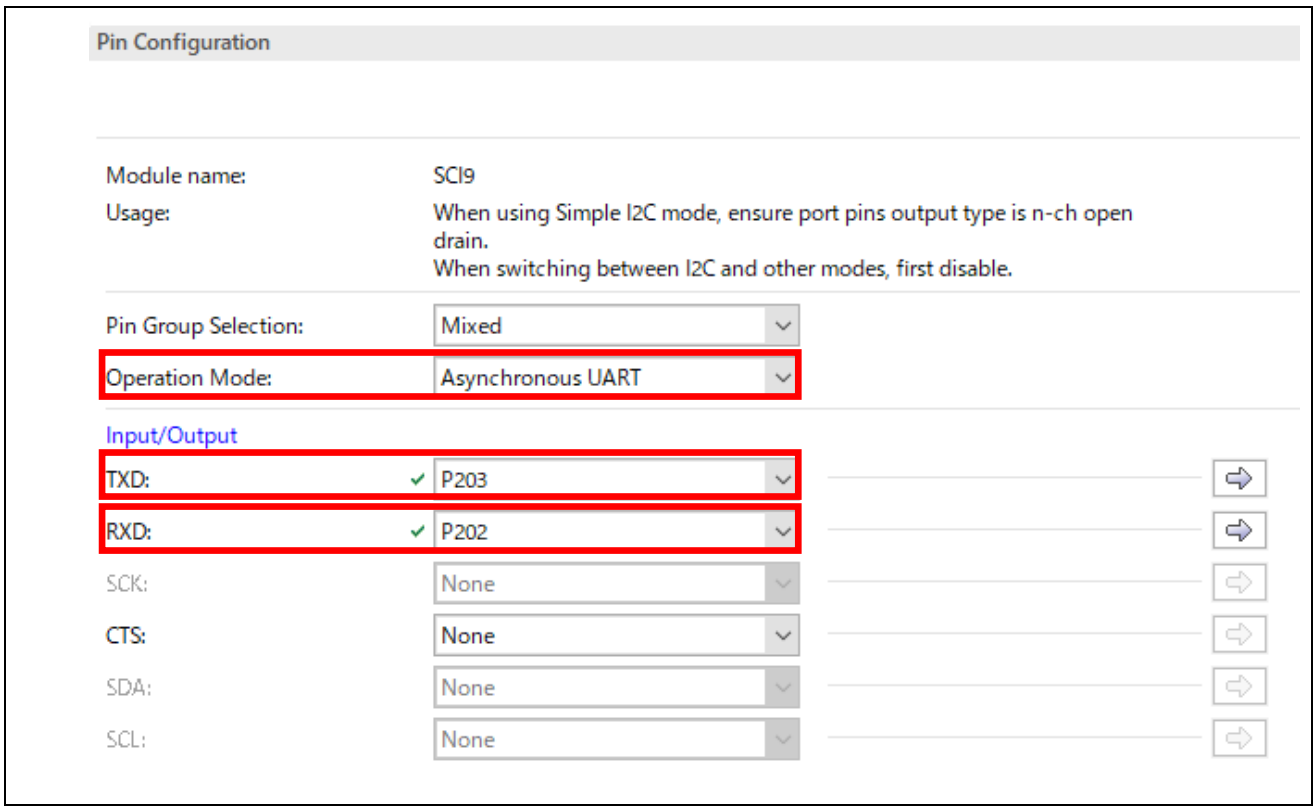


図 11　[Pins]タブ Operation Mode

11. 次に[Threads]タブに移動します。

図 12　[Threads]タブ

12. [New Stack]を選択し、[Framework] -> [input] -> [Cap Touch Framework on sf_touch_ctsuv2]を選択します。CTSU ドライバと DTC ミドルウェアが追加されます。

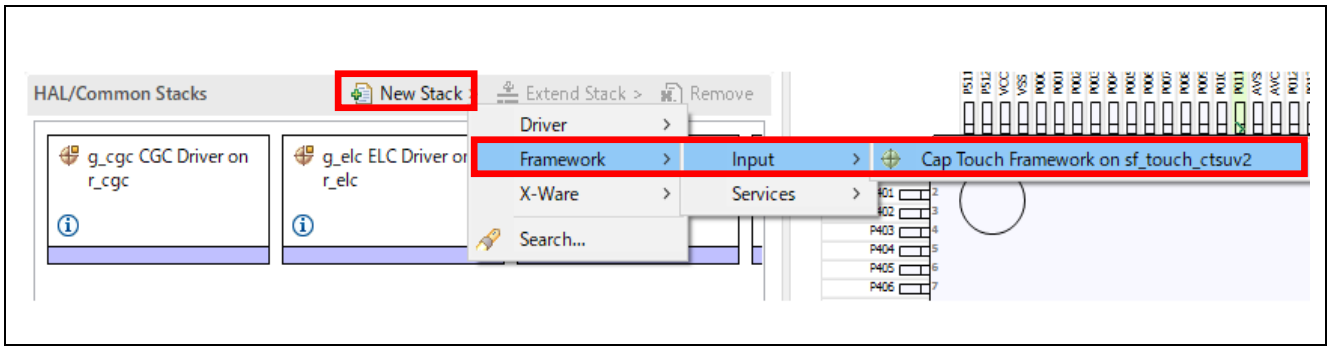


図 13　[Threads]タブ CTSU ドライバ、ミドルウェア追加

13. 以下の図のように、[HAL/Common Stacks]が表示されます。選択された Stacks はグレーになります。

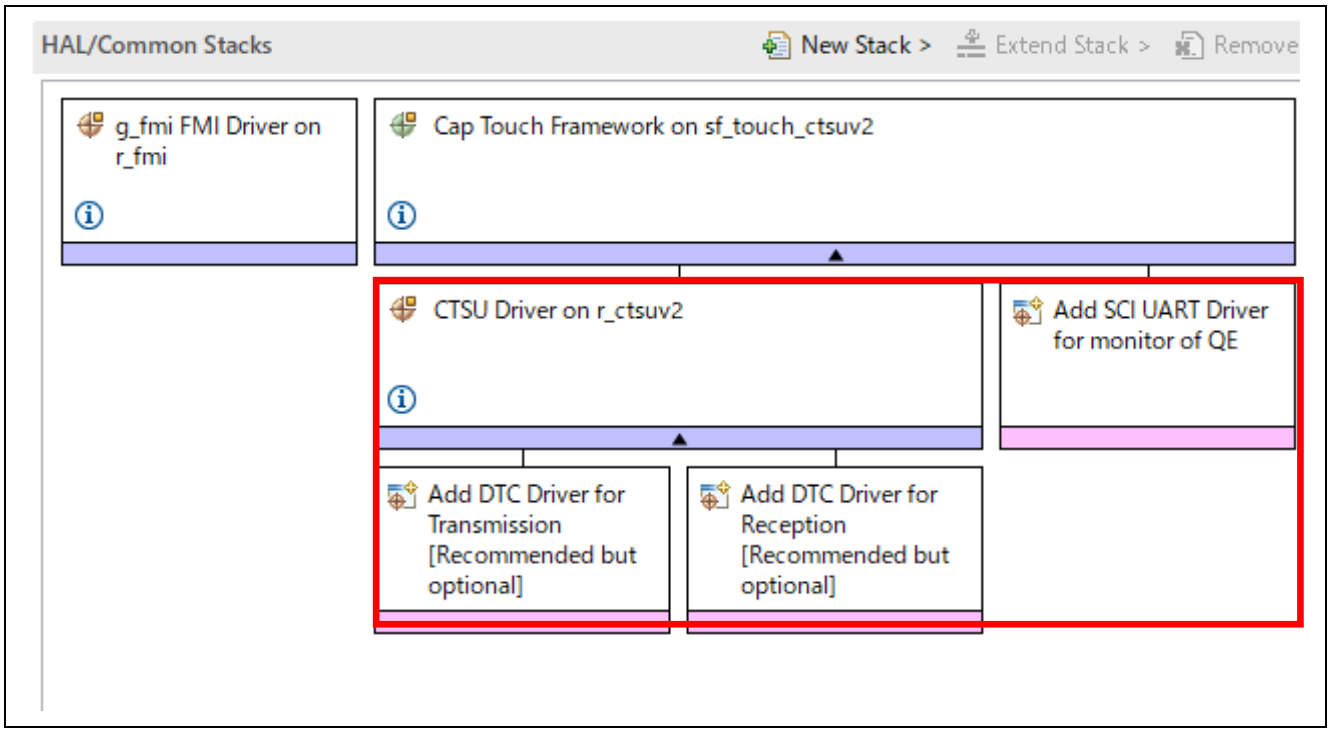


図 14　CTSU 追加後

14. 次に[CTSU Driver on r_ctsuv2]を選択し、”プロパティ”を表示します。

【注】：14～17項は DTC を使用する場合のみ設定してください。DTC により CPU を経由せずにメモリ、レジスタ間でデータを転送します。DTC を使用しない場合は18項へ進んでください。

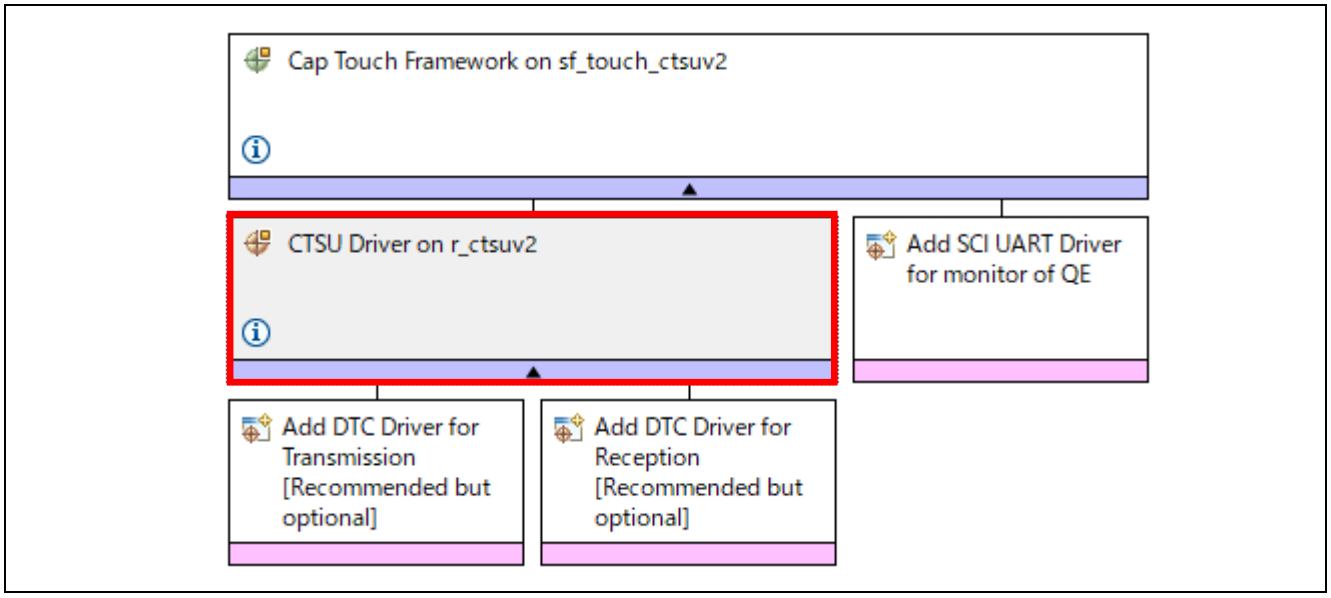


図 15　[CTSU Driver on r_ctsuv2]モジュール

15. 画面左下に表示される[プロパティ] - [Support for using DTC]を”Disabled”から”Enabled”に変更してください。

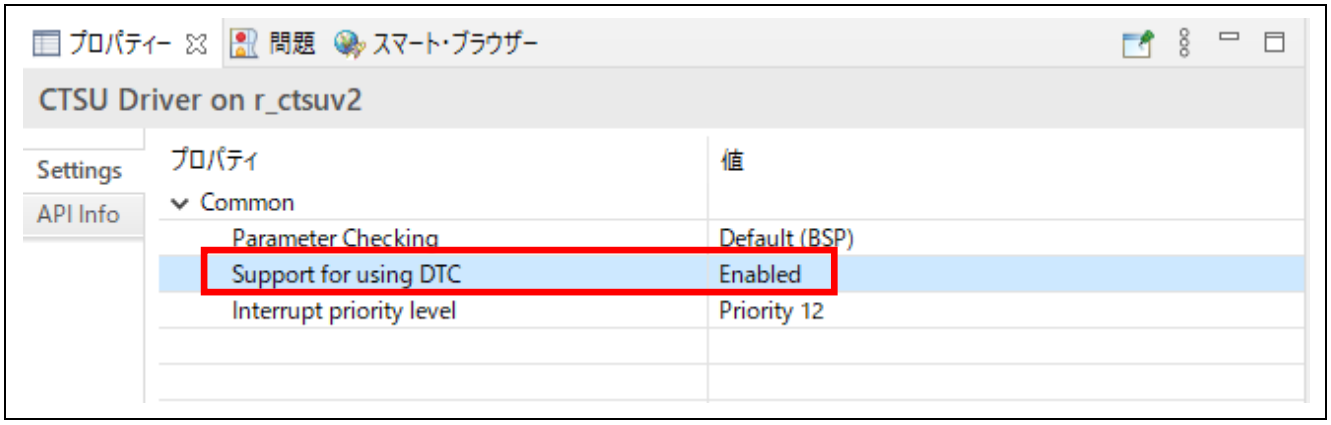


図 16　Support for using DTC

16. [Add DTC Driver for Transmission]を選択し、[New]->[Transfer Driver on r_dtc]から送信用の DTC ドライバを追加します。

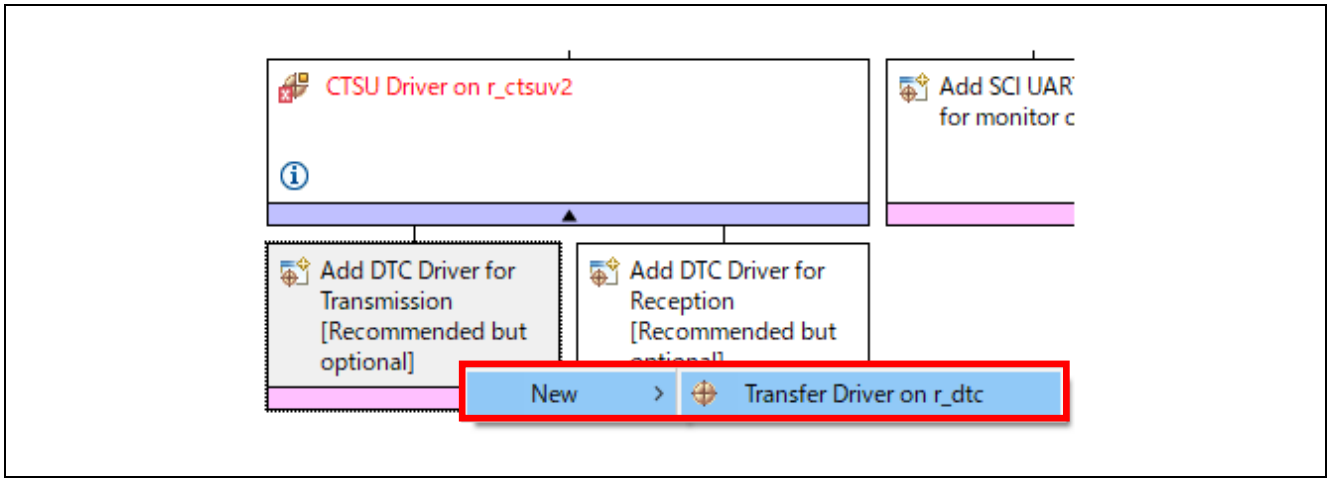


図 17　送信用 DTC ドライバ追加 (CTSU)

17. 同様に[Add DTC Driver for Reception]を選択し、[New]->[Transfer Driver on r_dtc]から受信用の DTC ドライバを追加します。

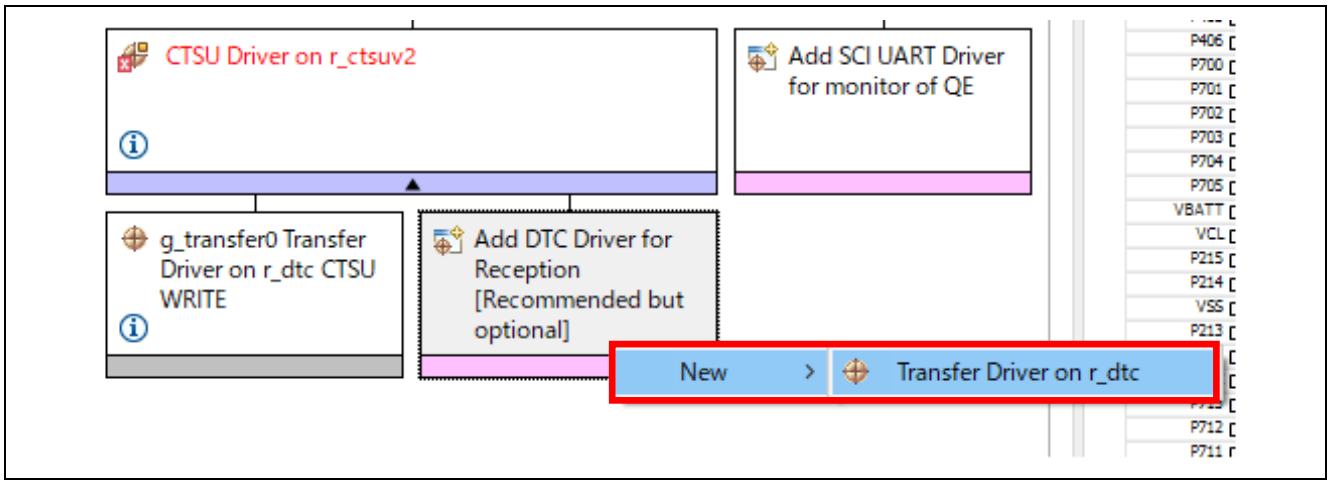


図 18　受信用 DTC ドライバ追加 (CTSU)

18. [Cap Touch Framework on sf_touch_ctsuv2]を選択して[プロパティ]を表示します。

【注】：18~23項はシリアル通信によるモニタリングを使用する場合のみ設定してください。
シリアル通信によるモニタリングを使用しない場合は、24項へ進んでください。

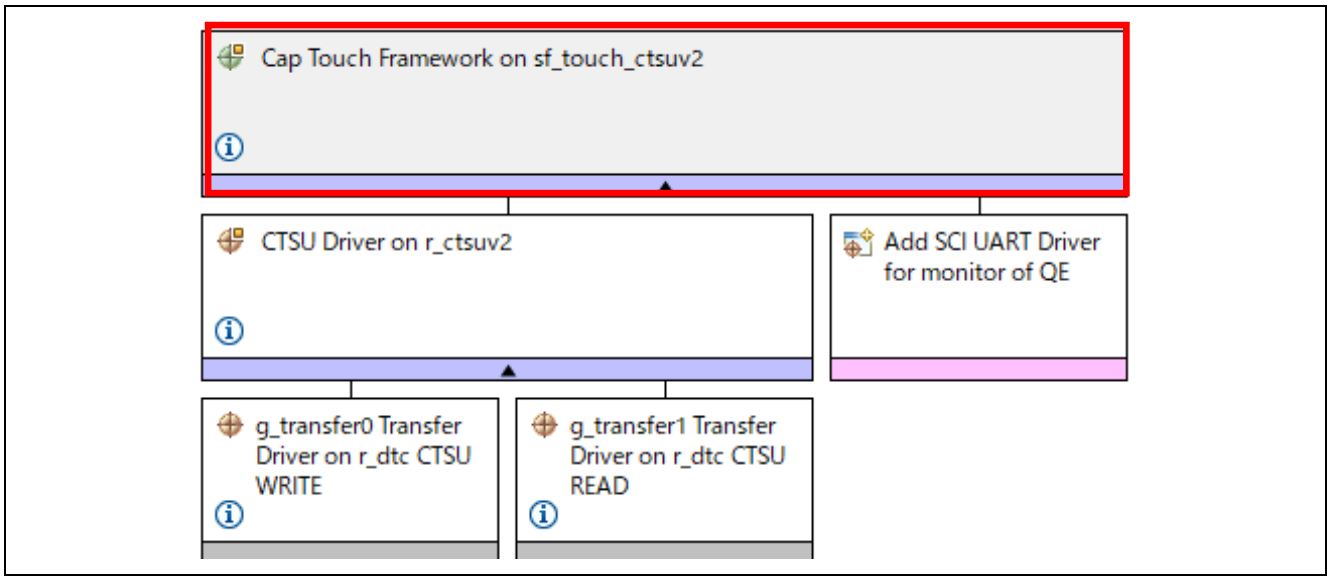


図 19　[Cap Touch Framework on sf_touch_ctsuv2]モジュール

19. 画面左下に表示される[プロパティ] – [Settings] – [Common] – [Support for QE monitoring using UART]を”Disabled”から”Enabled”に変更してください。

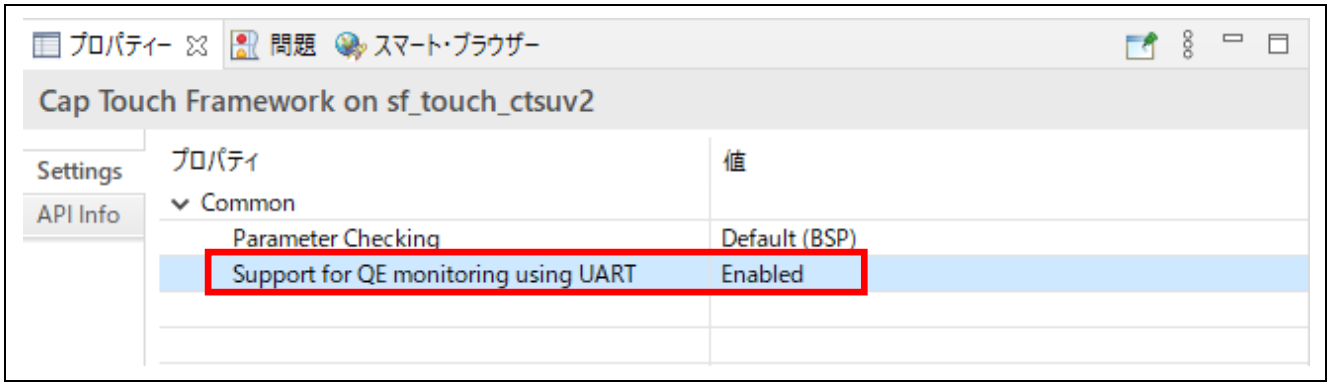


図 20　Support for QE monitoring using UART

20. [Add SCI UART Driver for monitor of QE]をクリックし、[New]->[UART Driver on r_sci_uart]から UART ドライバを追加します。

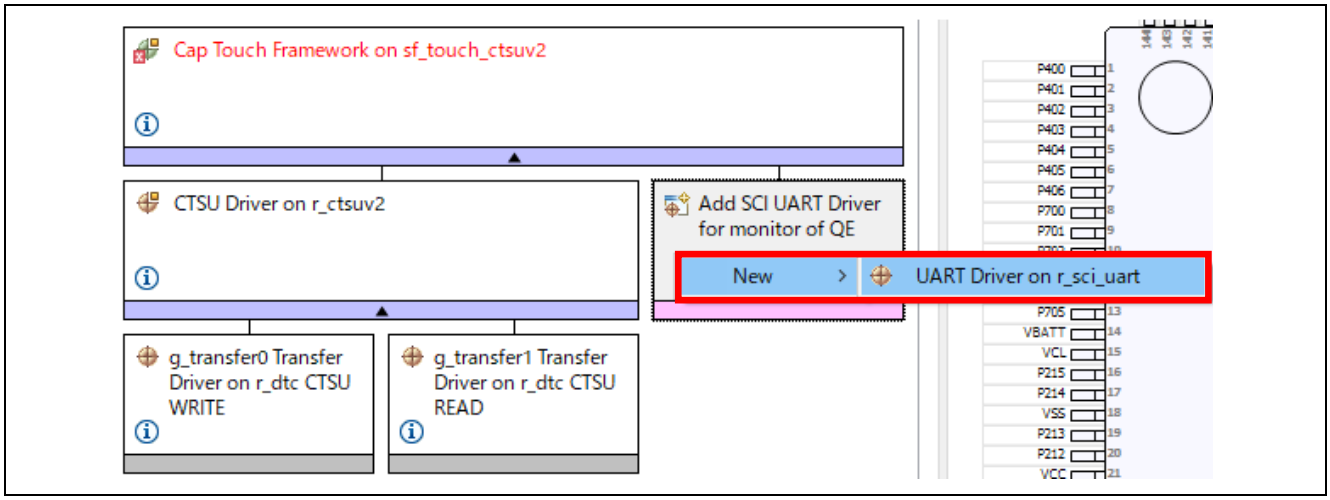


図 21　UART ドライバ追加

21. 次に[g_uart_qe UART Driver on r_sci_uart]をクリックして[プロパティ]を表示します。

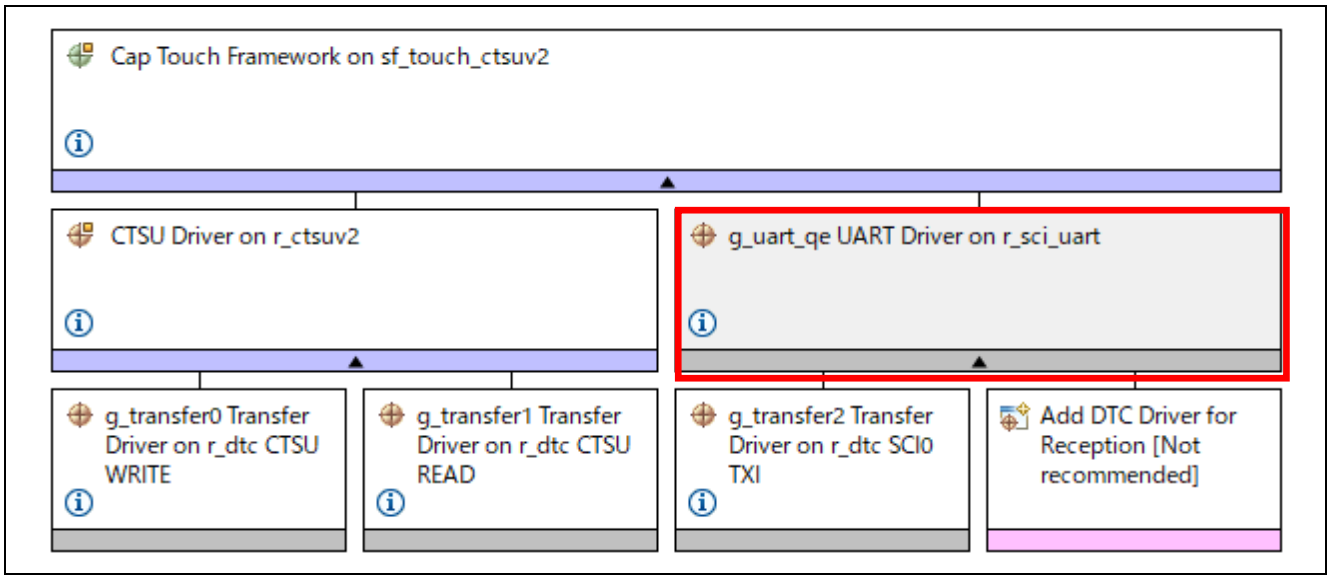


図 22　[g_uart_qe UART Driver on r_sci_uart]モジュール

22. 画面左下に表示される[プロパティ] – [Settings] – [Module g_uart_qe UART Driver on r_sci_uart] – [Channel]を”0”から”9”に変更してください。

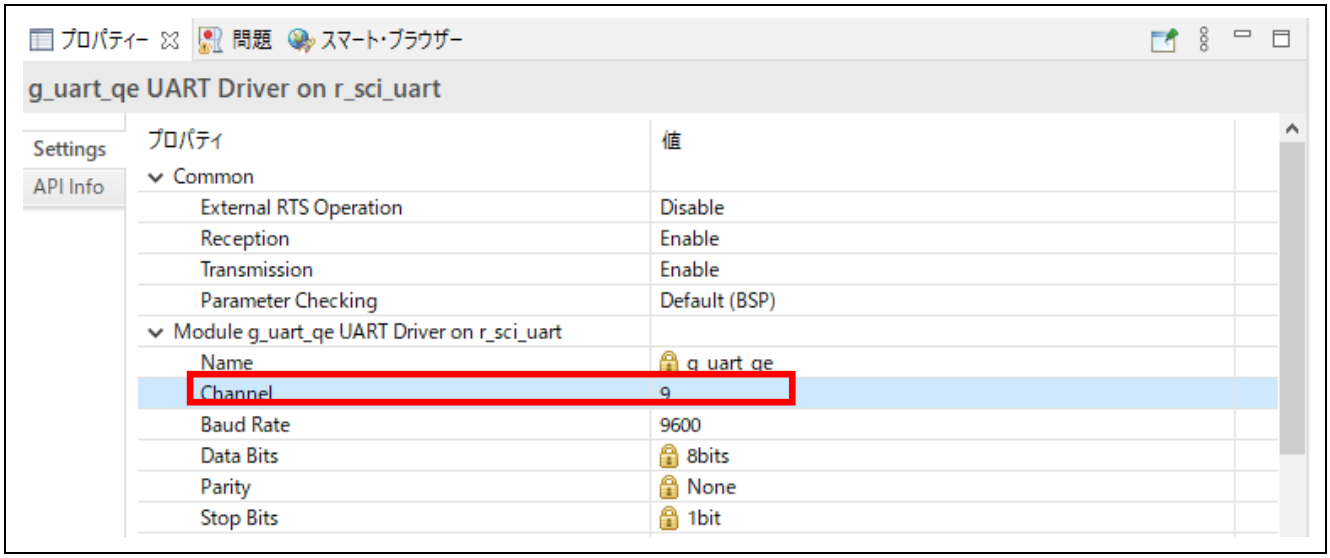


図 23　Channel

23. DTC の設定をします。[Add DTC Driver for Reception]をクリックし、[New]->[Transfer Driver on r_dtc]から受信用の DTC ドライバを追加します。

【注】：本項は DTC を使用する場合のみ設定してください。DTC により CPU を経由せずにメモリ、レジスタ間でデータを転送します。

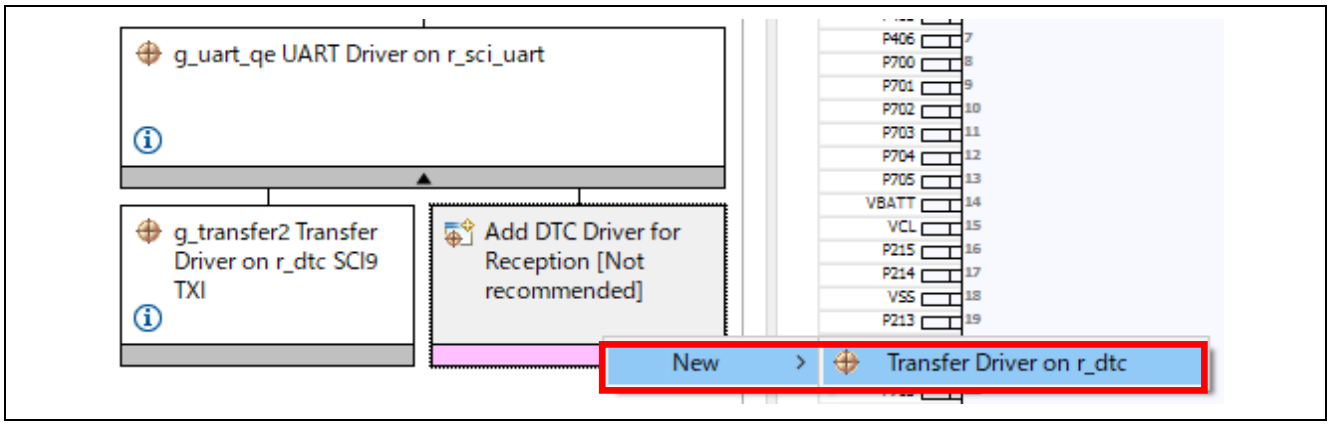


図 24　受信用 DTC ドライバ追加 (UART)

24. これで静電容量タッチに必要なモジュールが追加されました。最後にプロジェクトに必要なモジュールのコードを生成します。以下の図のように、画面右上にある[Generate Project Content]をクリックします。

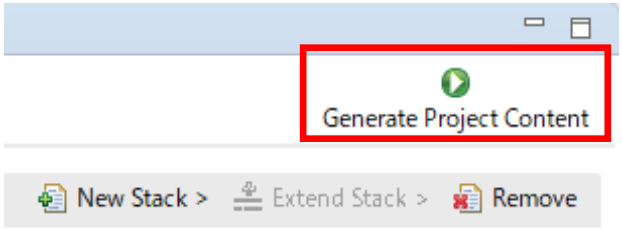


図 25　コード生成ボタン

6.3　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 メイン/センサ・チューナ RA,RL78,Synergy (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。

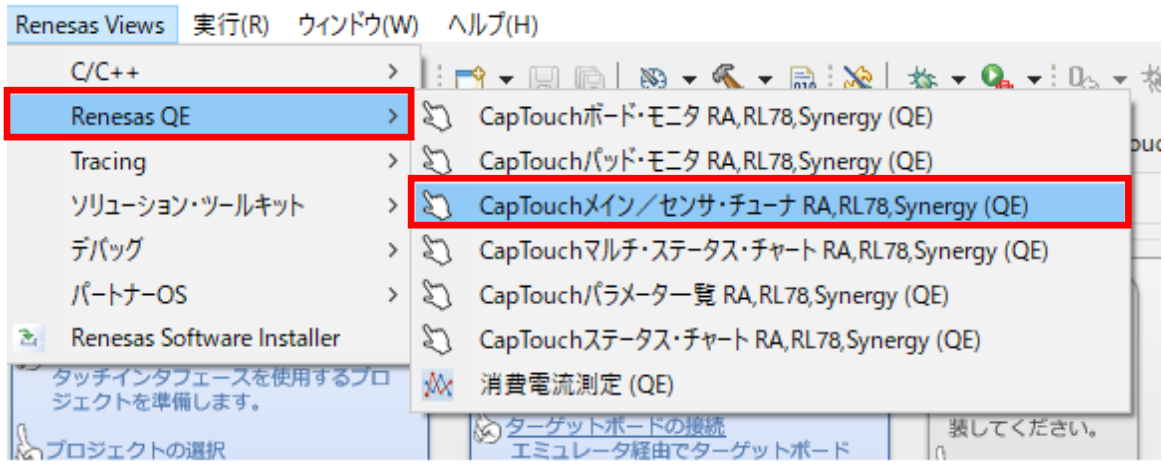


図 26　[CapTouch メイン/センサ・チューナ RA,RL78,Synergy (QE)]メニュー

2. [CapTouch メイン/センサ・チューナ RA,RL78,Synergy (QE)] の[プロジェクトの選択]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 27　プロジェクトの選択

3. 次に、[構成の選択]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。

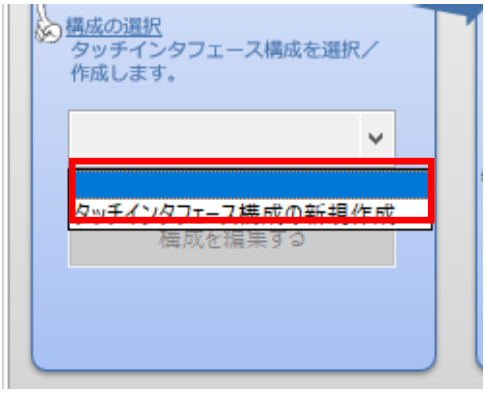


図 28　タッチインタフェース構成の選択

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。

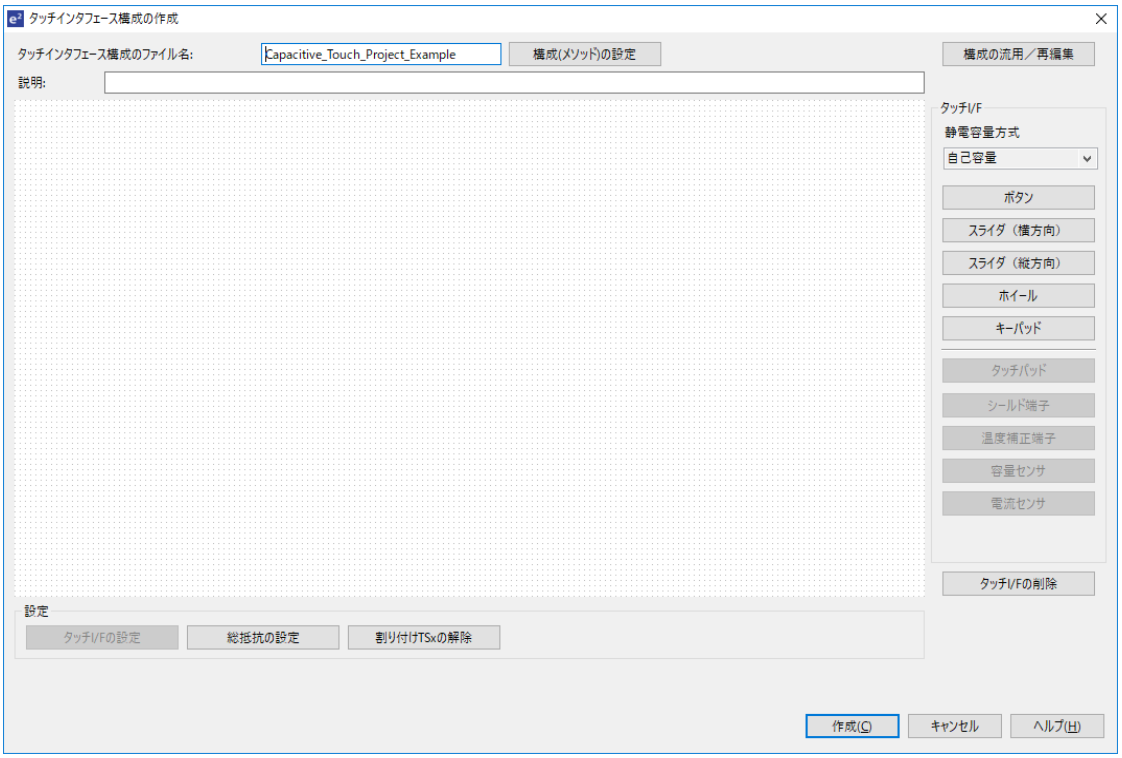


図 29　[タッチインタフェース構成の作成]ウィンドウ

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、タッチインタフェースの配置領域上にボタン 1 つを追加します。

   選択を解除するには、キーボードの ESC キーの押下、もしくは、再度[ボタン]を選択します。このボタンには TS 番号の振り分けがまだ設定されていないので、設定エラー（赤色）のままです。

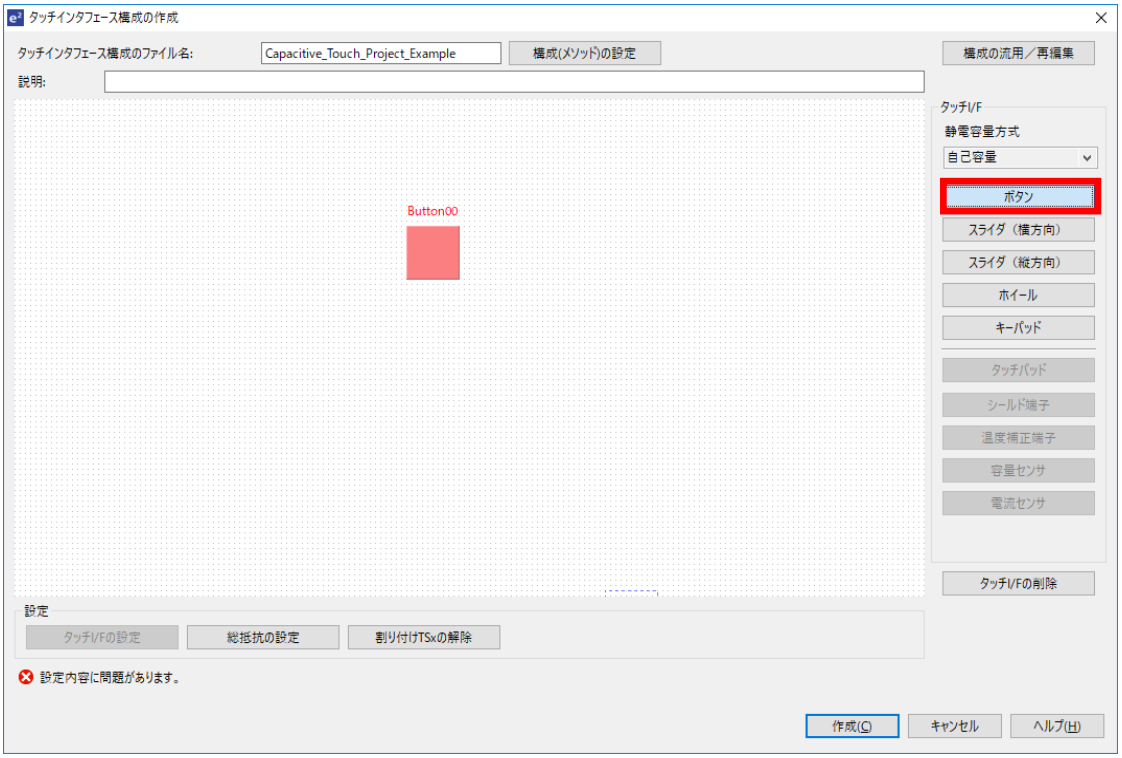


図 30　ボタン追加

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、このボタンに割り当てる MCU のセンサポートに"TS31"を選択します。

   Synergy 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、割り付けが正しく設定されると設定エラーの表示がなくなり、ボタンの表示色が緑色になります。

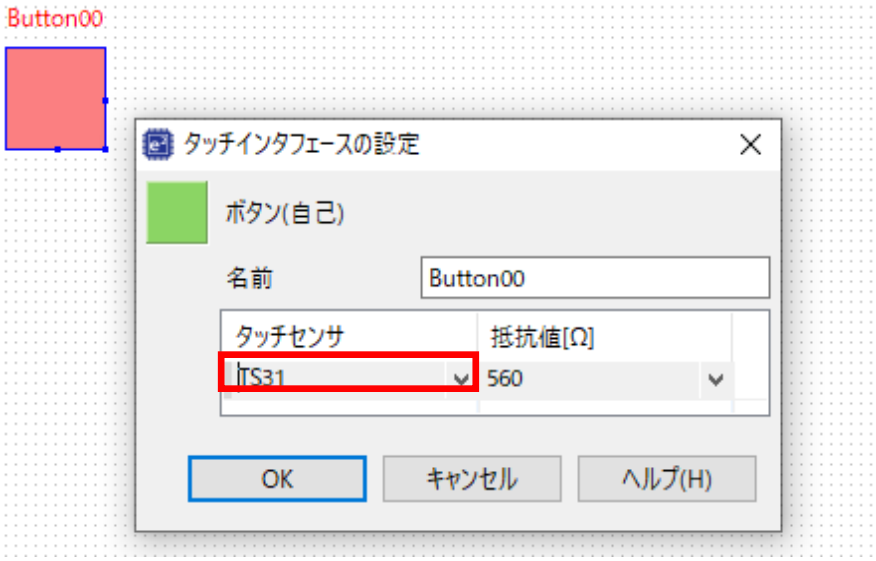


図 31　タッチセンサの設定（ボタン）

7. [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。

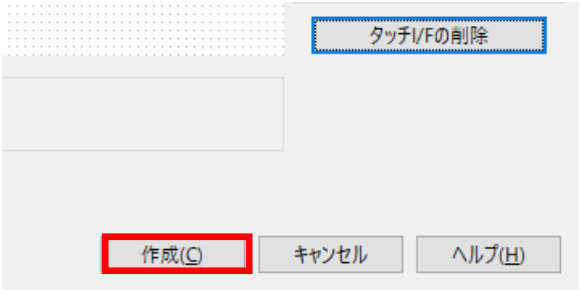


図 32　[作成]ボタン

8. [CapTouch メイン/センサ・チューナ RA,RL78,Synergy (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。

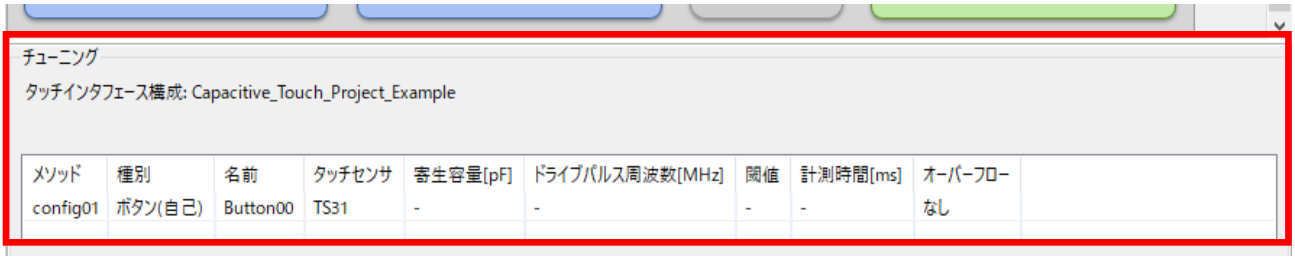


図 33　チューニングパネル

9. e² studio 左上の アイコンをクリックしてビルドを開始します。プロジェクトはエラーやワーニングなしでビルドされます。

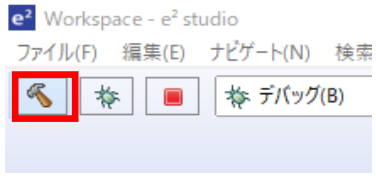


図 34　ビルドボタン

6.4　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

1. 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する必要があります。アイコンをクリックし、デバッグ構成を選択してください。

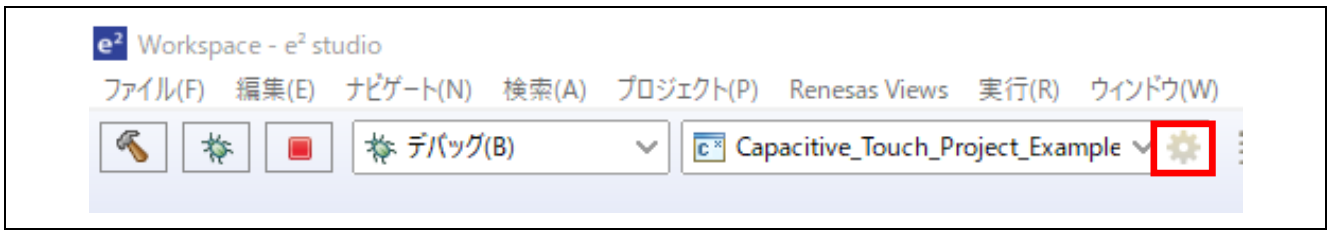


図 35　デバック構成の編集ボタン

2. [Startup]タブを選択します。

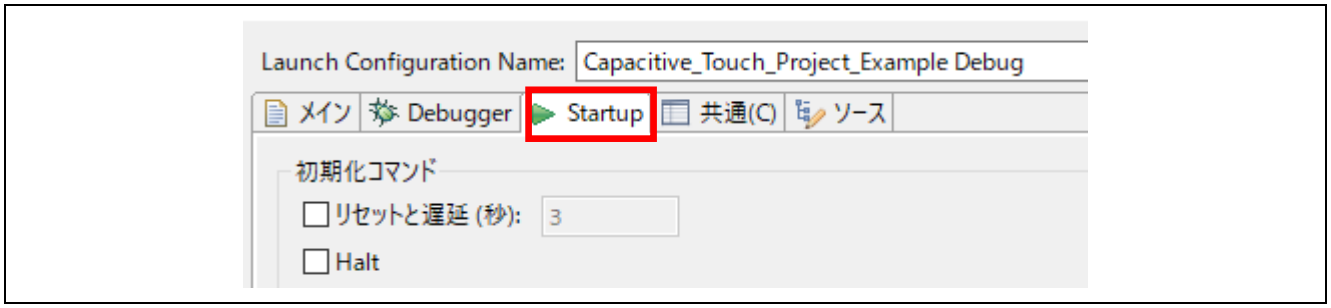


図 36　[Startup]タブ

3. [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ます。これらのチェックボックスを表示するにはダイアログを下にスクロールする必要があります。

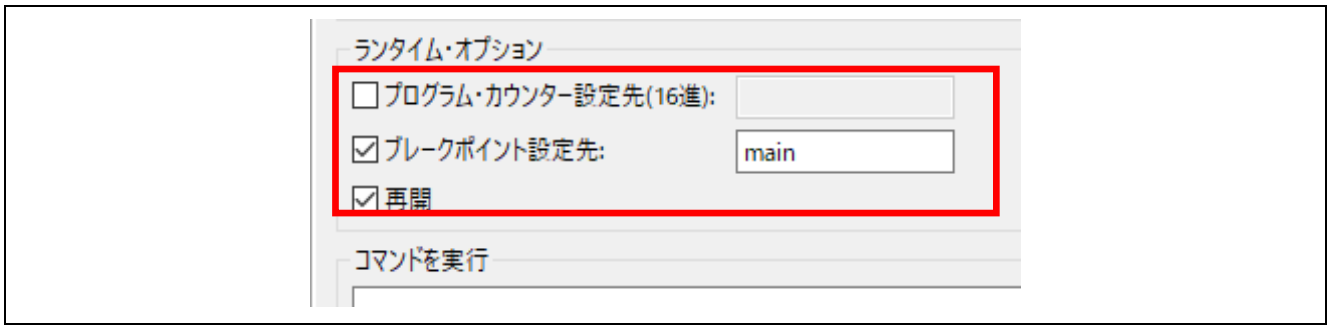


図 37　ランタイム・オプション

4. [OK]をクリックし、変更した設定を有効にします。これでチューニングのためのプロジェクトの設定とデバッグ構成の設定は終了です。

6.5 QE for Capacitive Touch [RA,RL78,Synergy]を使用した静電容量タッチセンサ・チューニング

1. エミュレータがボードと PC に正しく接続されていることを確認します。

2. [CapTouch メイン/センサ・チューナ RA,RL78,Synergy (QE)]の[チューニングを開始する]をクリックし、自動チューニングを開始します。

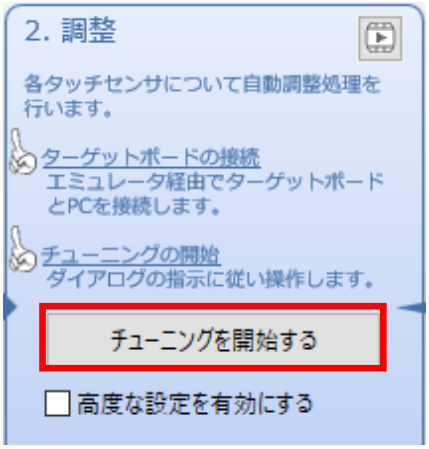


図 38　チューニングの開始

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]をチェックし、[はい]をクリックしてデバッグセッションと QE for Capacitive Touch [RA,RL78,Synergy]の自動チューニングを続行してください。

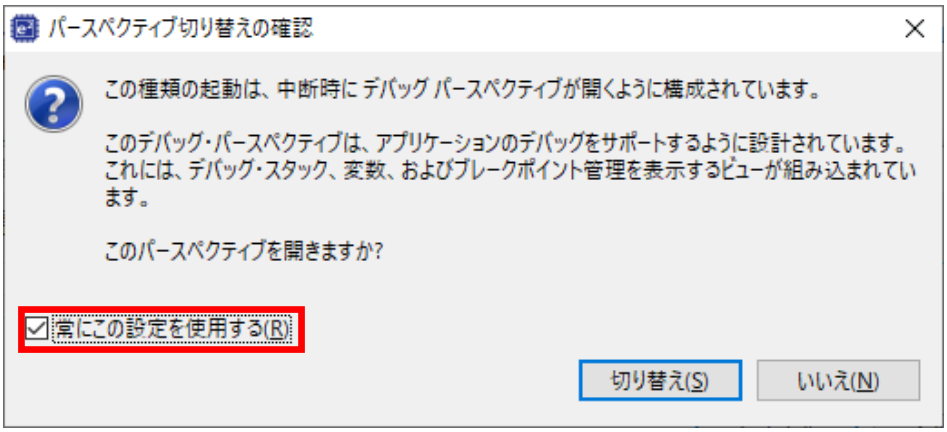


図 39　パースペクティブ設定

4. QE for Capacitive Touch [RA,RL78,Synergy]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。

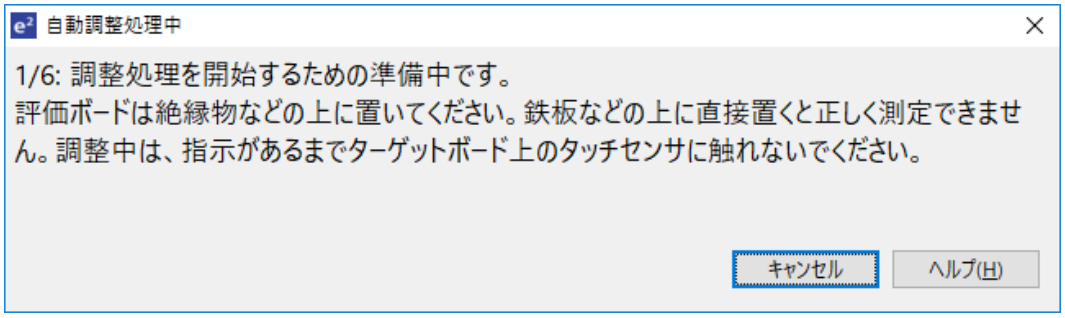


図 40　[自動調整処理中]ダイアログ（準備中）

5. いくつかの自動工程を経て、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(Button00/TS31)を通常の圧力でタッチします。自己容量方式のタッチセンサに触れているとき、バーグラフは右に増加し、数値で示すタッチカウント値が増えます。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。

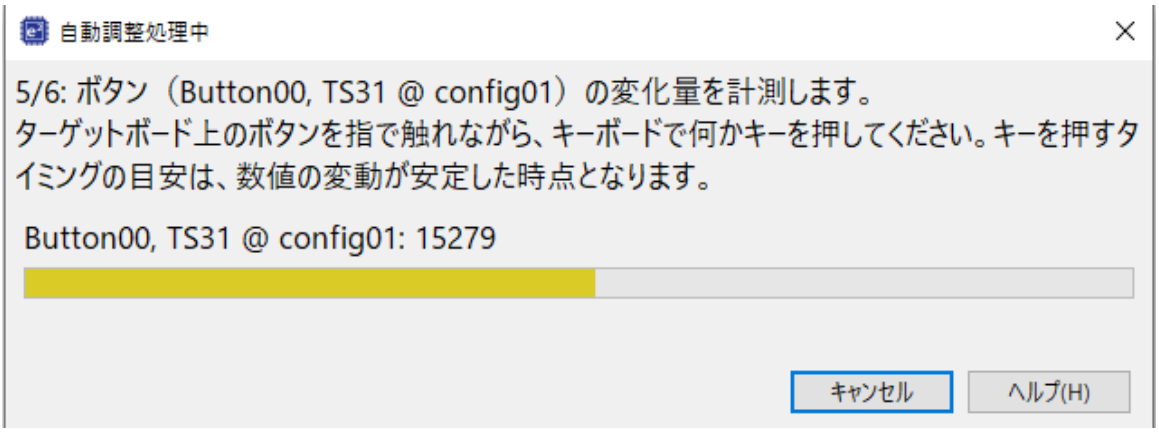


図 41　[自動調整処理中]ダイアログ（計測中）

6. チューニングが完了すると、以下のようなダイアログが表示され閾値を確認できます。この閾値はミドルウェアでタッチのイベント判定に使用されます。

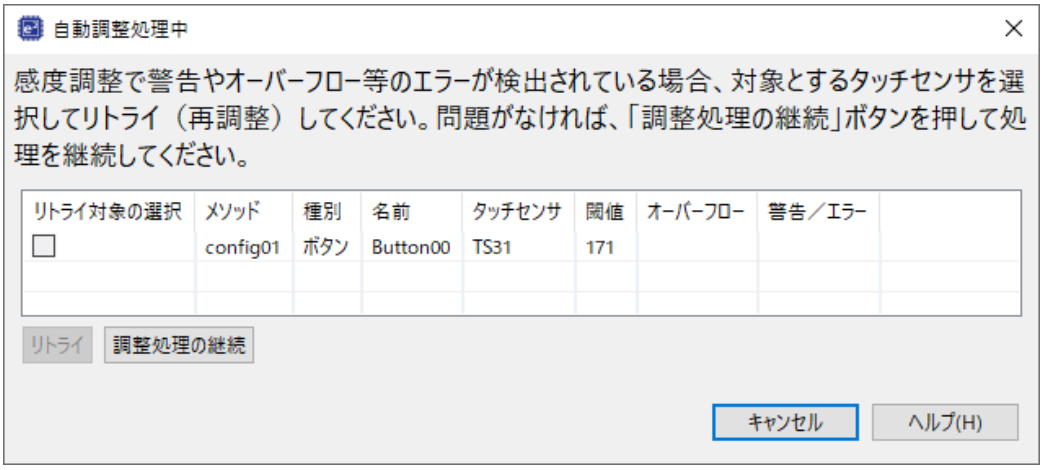


図 42　[自動調整処理中]ダイアログ（調整完了）

7. 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/センサ・チューナ RA,RL78,Synergy (QE)]に戻ります。

図 43　[調整処理の継続]ボタン

8. 残る手順はチューニングされたパラメータファイルの出力だけです。[ファイルを生成する]をクリックします。

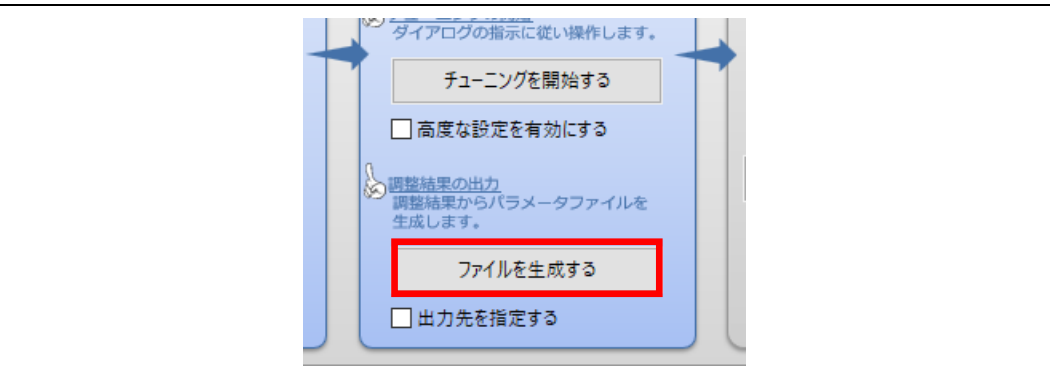


図 44　調整結果の出力

9. [プロジェクト・エクスプローラー]ウィンドウで[qe_gen]フォルダ下に、qe_touch_config.c, 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。

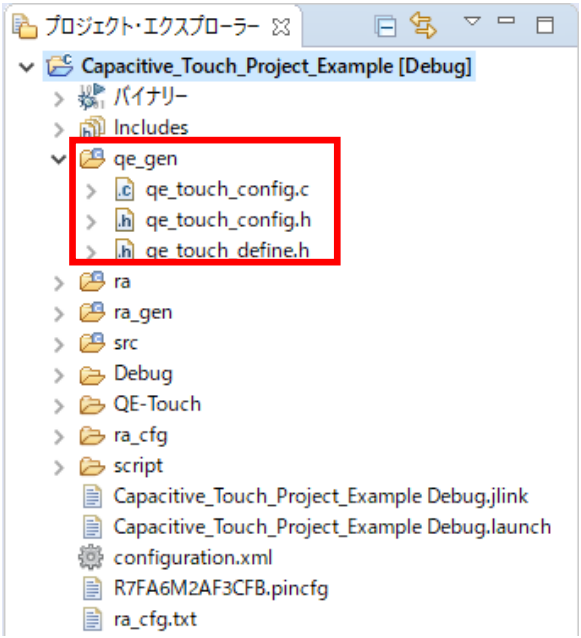


図 45　生成されたパラメータファイル

10. 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]ウィンドウにビルドした結果、エラーがないことが確認できます。

6.6　アプリケーションに qe_touch_main() を追加

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/センサ・チューナ RA,RL78,Synergy (QE)]の[例を表示する]をクリックします。

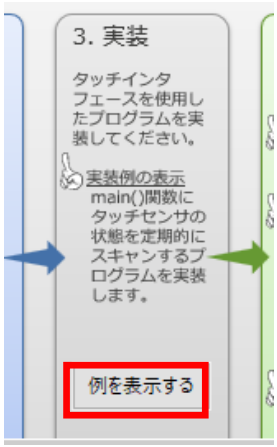


図 46　実装例の表示

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。次に[OK]をクリックして、ウィンドウを閉じます。

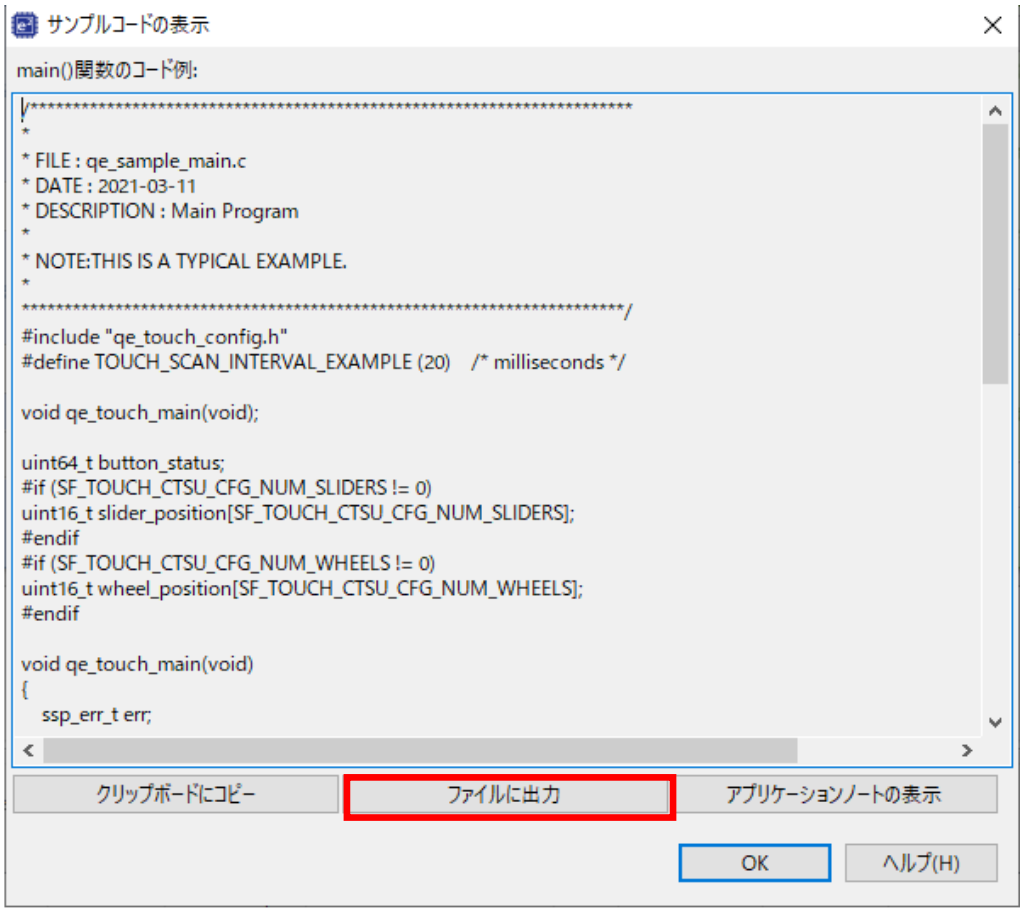


図 47　[サンプルコードの表示]ウィンドウ

3. [プロジェクト・エクスプローラー]の[qe_gen]フォルダ下に qe_touch_sample.c が生成されたことを確認してください。

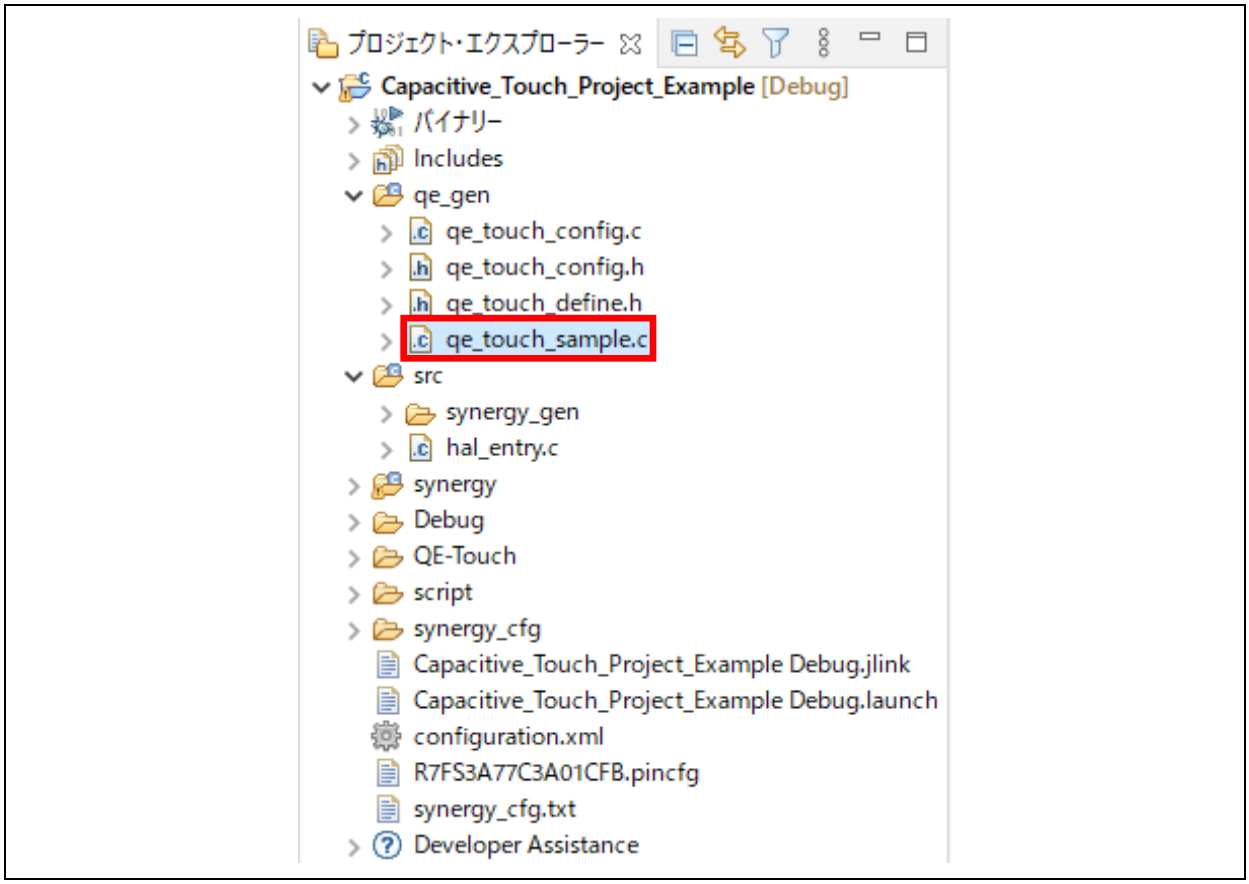


図 48　qe_touch_sample.c

4. [src]フォルダ下の hal_entry.c を開きます。

   qe_touch_main() 関数の宣言と hal_entry() 関数内に qe_touch_main() 関数のコールを追記します。

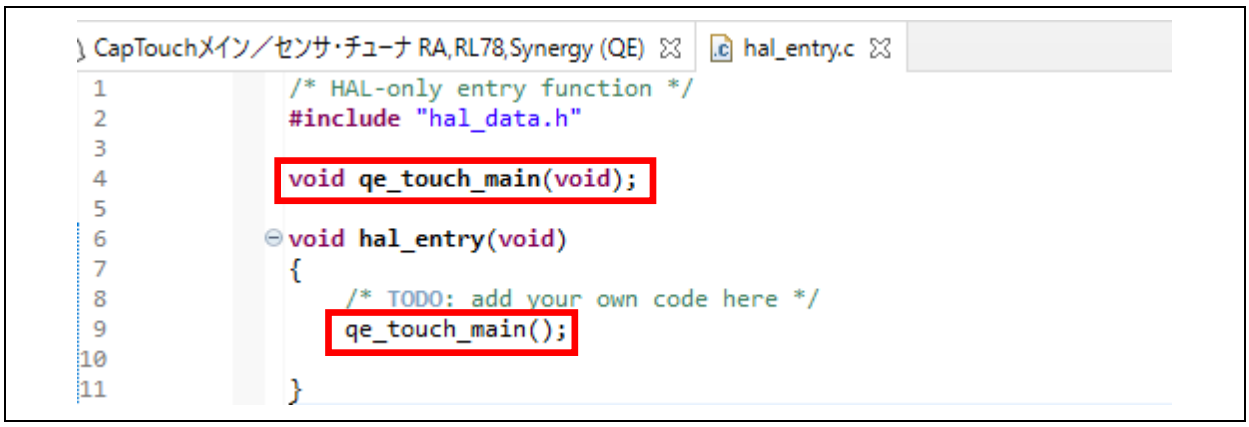


図 49　hal_entry.c への追記

5. これで、アプリケーション例で必要なコードの変更はすべて終了です。アプリケーション例のプロジェクトをビルドすると、エラーまたはワーニングなしで終了することを確認できます。

6.7　[式]ウィンドウと QE for Capacitive Touch [RA,RL78,Synergy]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。

   デバッグセッションは、main.c の hal_entry() 関数コールでストップします。これは main() 関数の最初のエントリーポイントです。

2. hal_entry() 関数を選択し、宣言を開きます。

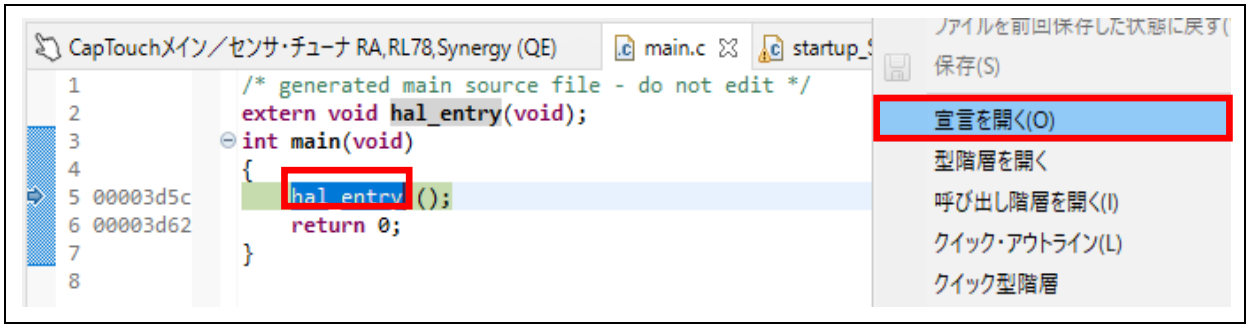


図 50　宣言を開く（hal_entry 関数）

3. 次に、qe_touch_main() 関数を選択し、宣言を開きます。

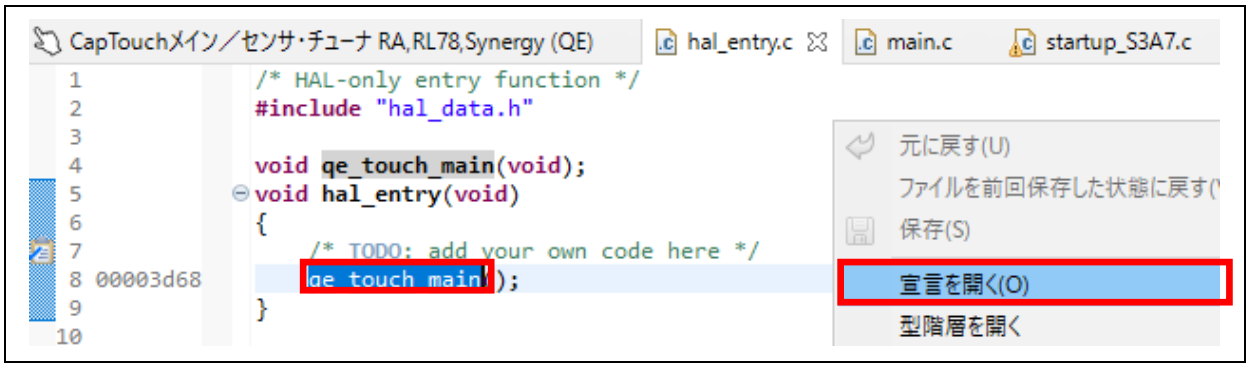


図 51　宣言を開く（qe_touch_main 関数）

4. qe_touch_main.c を下にスクロールして、while(true)ループの g_qe_touch_instance_config01.p_api->dataGet() を表示します。”button_status”を選択し、監視式に変数を追加します。

図 52　[監視式を追加]メニュー

5. [式]ウィンドウで右クリックし、[Enable Real-time Refresh]を選択すると、追加した変数のリアルタイムリフレッシュが有効になります。

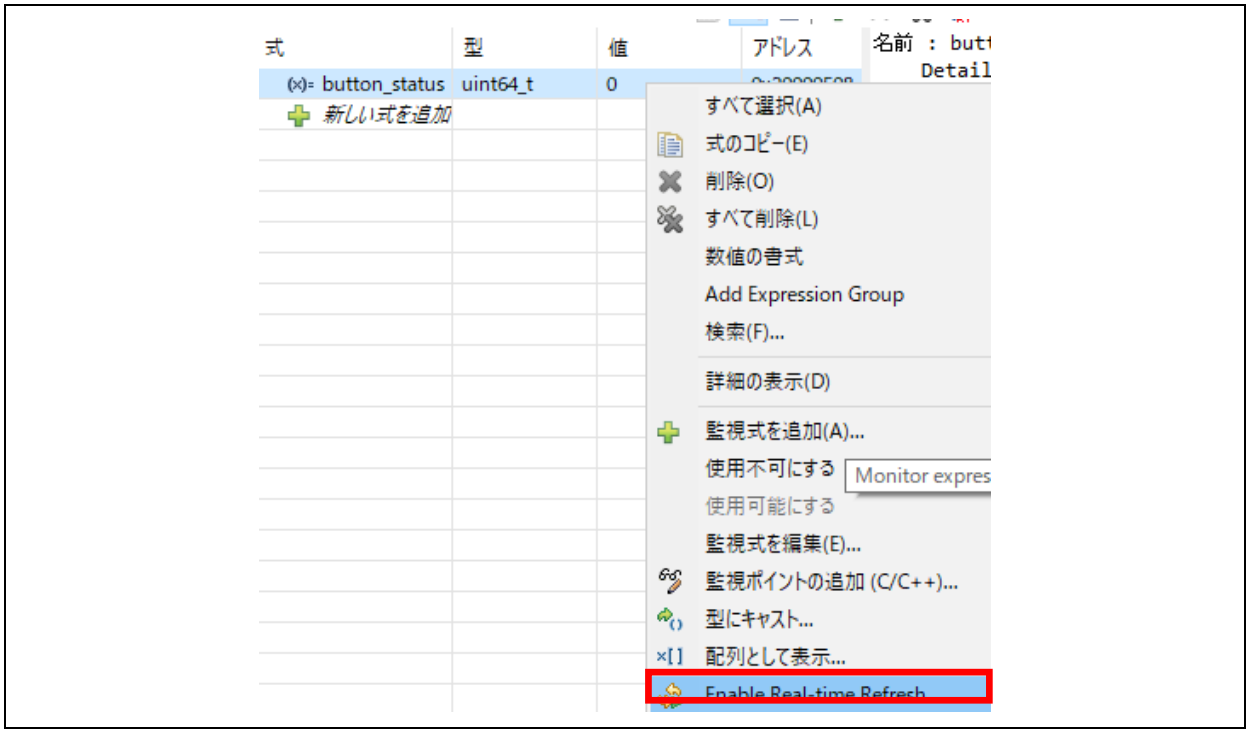


図 53　[Enable Real-time Refresh]メニュー

6. e² studio のツールバーボタンのほぼ中央にある アイコンをクリックし、プログラム実行を続行します。

7. 「6.3 静電容量タッチインタフェース作成」の6項で”Button00”として定義した、ボード上の”TS31”をタッチします。”Button00”をタッチすると[式]ウィンドウの”button_status”の値が”0”から”1”になることを確認できます。

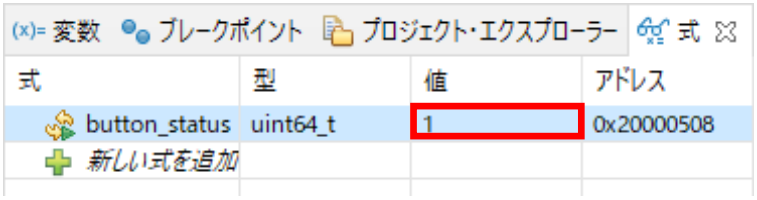


図 54　[式]ウィンドウでのタッチ状態の確認

8. [CapTouch メイン/センサ・チューナ RA,RL78,Synergy (QE)]から[ビューを開く]を選択し、[CapTouch ボード・モニタ RA,RL78,Synergy (QE)]を起動します。

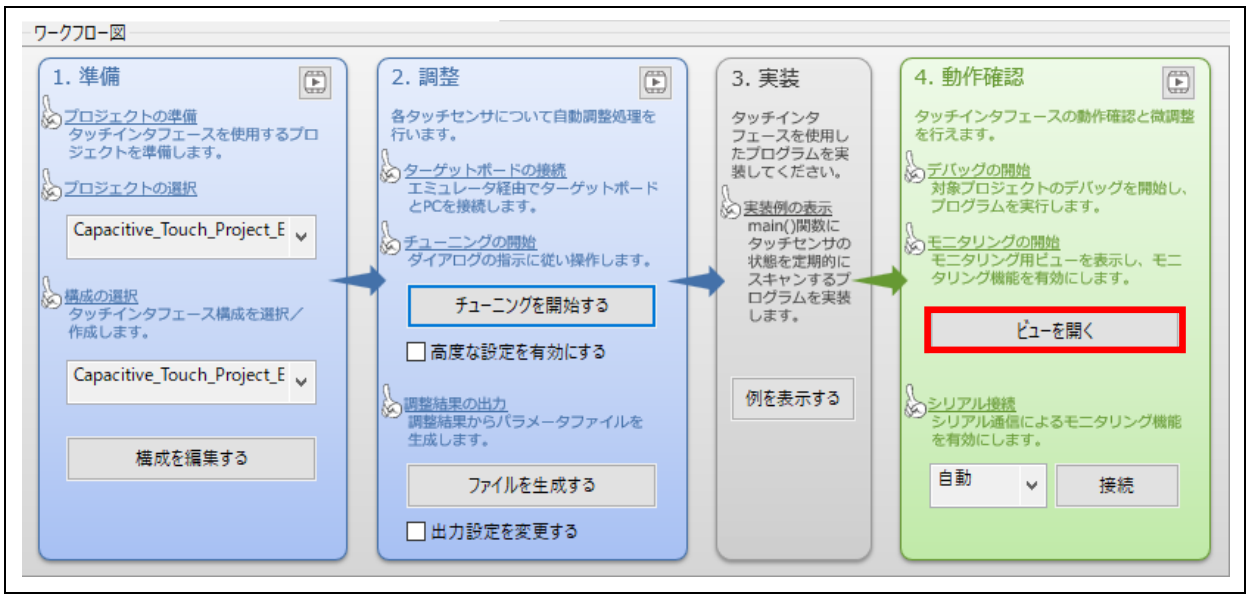


図 55　[ビューを開く]ボタン

9.　[CapTouch ボード・モニタ　RA,RL78,Synergy (QE)]は以下のように表示されます。

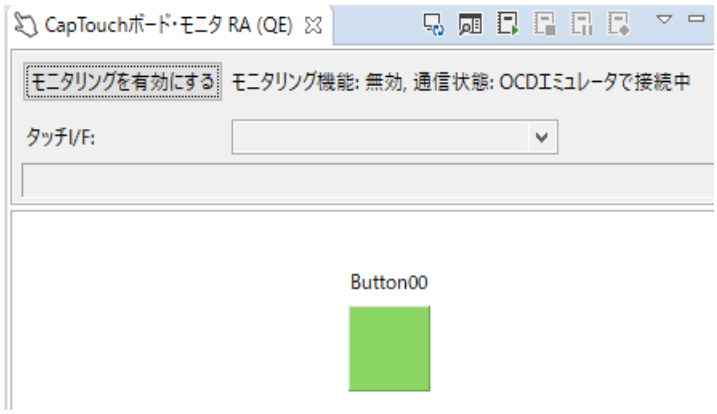


図 56　[CapTouch ボード・モニタ　RA,RL78,Synergy (QE)]ウィンドウ

10. [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。

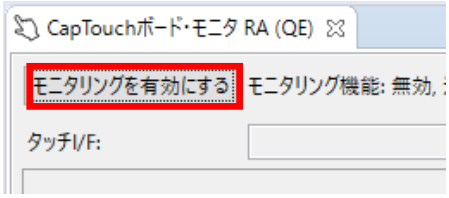


図 57　モニタリング有効化ボタン

11. “BWS”上のボタン”TS31”をタッチします。[CapTouch ボード・モニタ　RA,RL78,Synergy (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 58　[CapTouch ボード・モニタ　RA,RL78,Synergy (QE)]ウィンドウでのタッチ状態の確認

12. 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 RA,RL78,Synergy (QE)]を起動します。

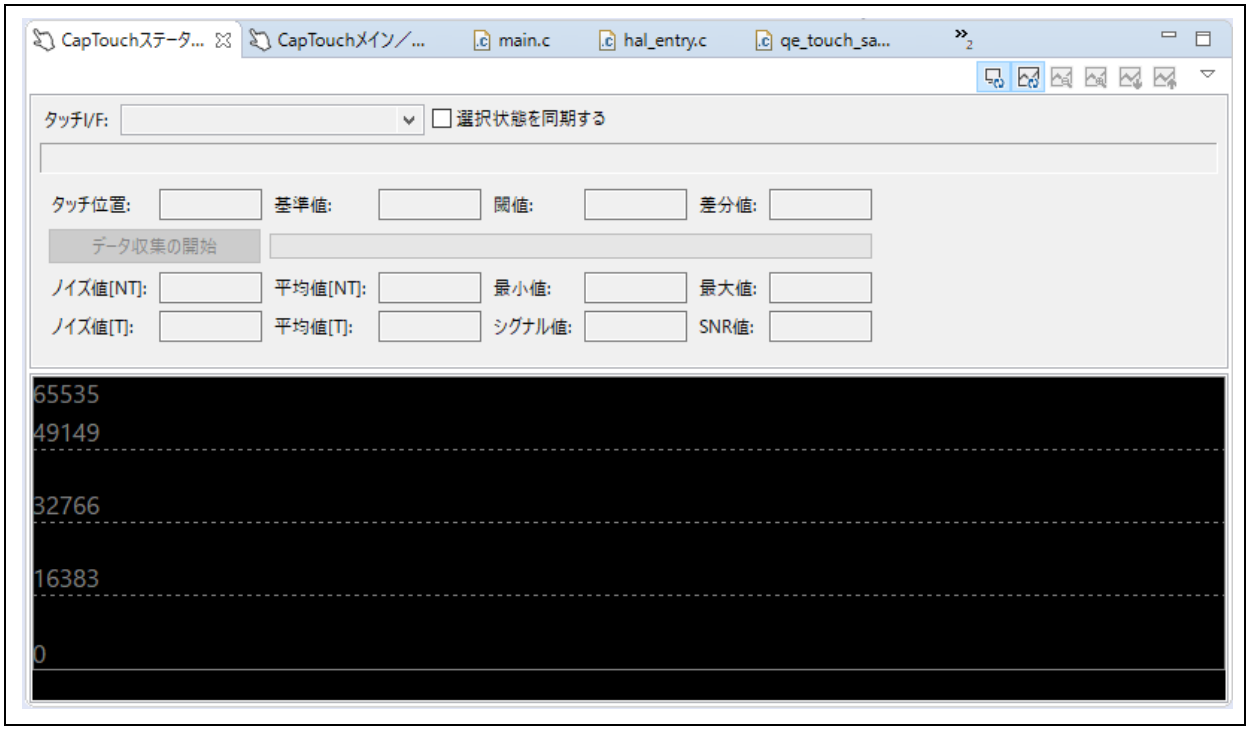


図 59　[CapTouch ステータス・チャート RA,RL78,Synergy (QE)]ウィンドウ

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

図 60　タッチ I/F の設定

14. グラフには実行中の値が表示されます。ボード上の”TS31”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。緑のラインは閾値を表し、SSP はボタンが操作/タッチされているかどうかを判断するために使用されます。グラフ下部の赤い矩形はタッチカウント値が閾値を超えてタッチが検出されたことを表示しています。

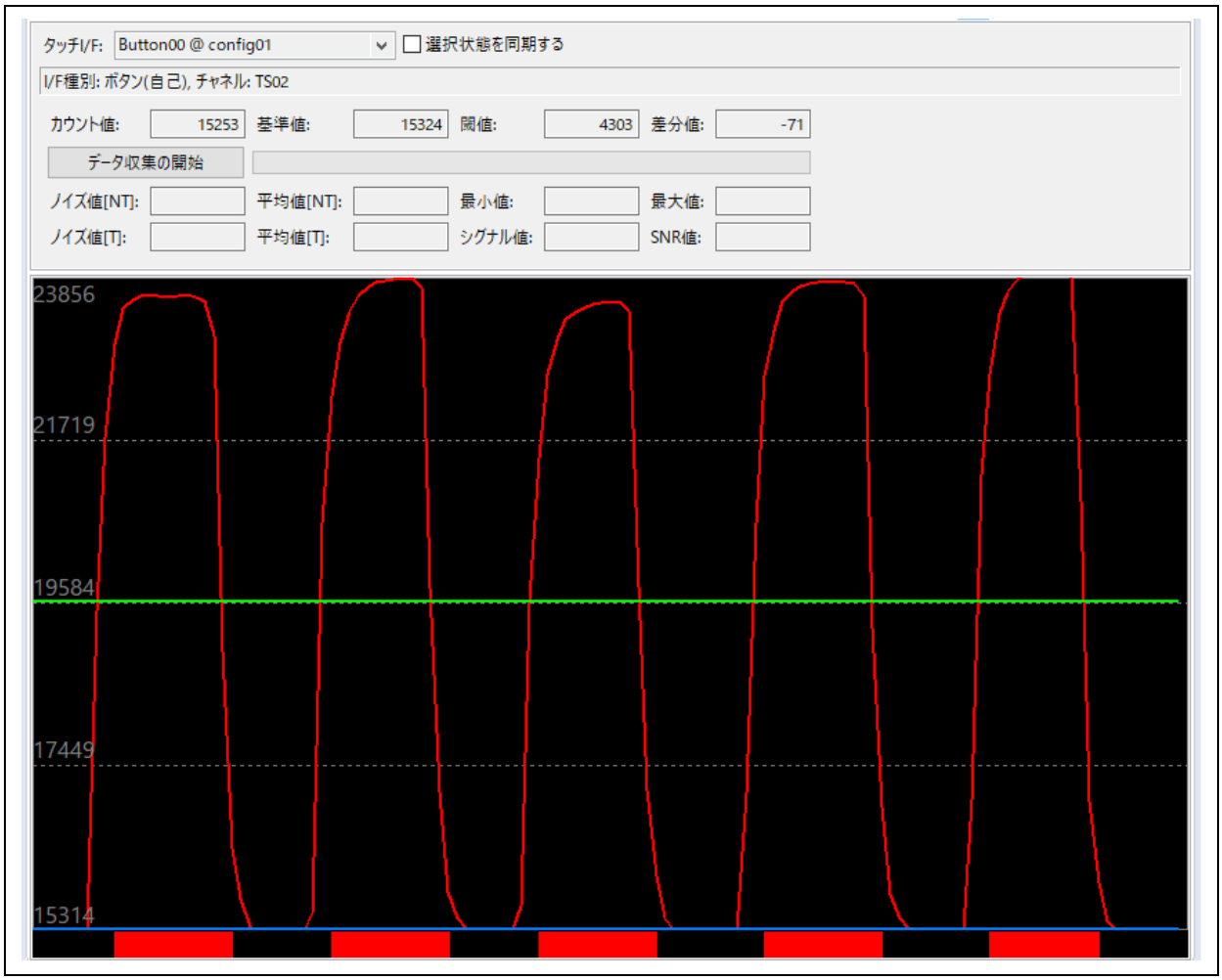


図 61　[CapTouch ステータス・チャート RA,RL78,Synergy (QE)]ウィンドウでのタッチ状態の確認

【注】：15～18項は標準偏差の表示および測定する際に設定する必要があります。

15. 次に標準偏差の測定方法についてです。[データ収集の開始]をクリックします。タッチオフ状態を収集している間は電極に触れないでください。緑色のバーはデータ収集率を表しています。緑色のバーが右端まで達するとタッチオフ状態のデータ収集は完了します。

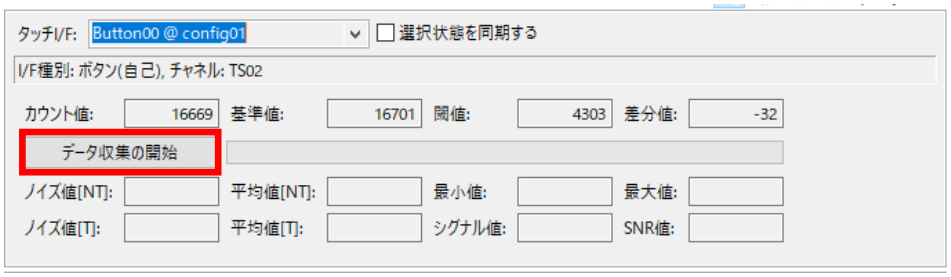


図 62　[データ収集の開始]ボタン　（タッチオフ状態）

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

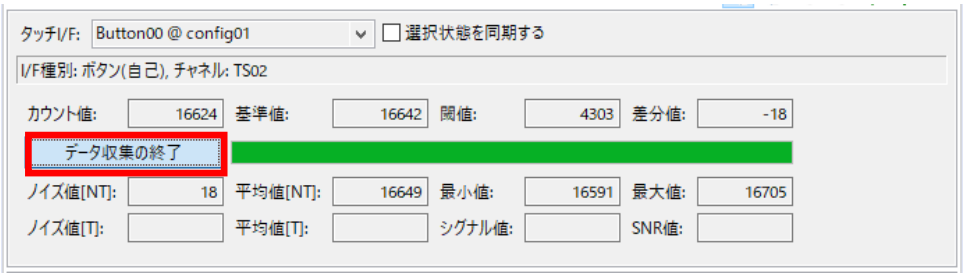


図 63　[データ収集の終了]ボタン（タッチオフ状態）

17. 次にタッチオン状態のデータを収集するために電極に触れてください。電極に触れている状態で[データ収集の開始]をクリックしてください。

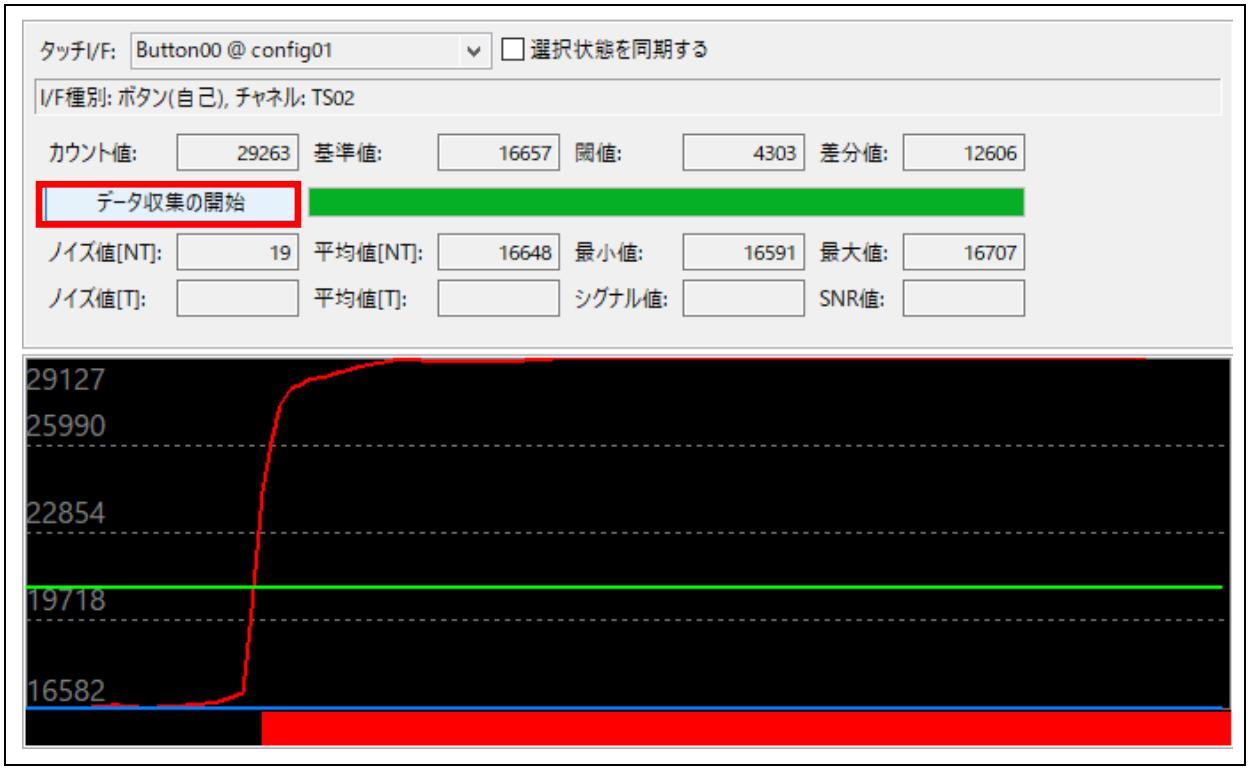


図 64　[データ収集の開始]ボタン　（タッチオン状態）

18. 緑色のバーが右端まで達したときに[データ収集の終了]をクリックしてください。データ収集が完了すると SNR が表示されます。

図 65　[データ収集の終了]ボタン（タッチオン状態）

6.8 シリアル通信を利用した QE for Capacitive Touch [RA,RL78,Synergy]によるモニタリング

1. モニタリングを実行中の場合は、選択中の[モニタリングを有効にする]をクリックします。”モニタリング機能：有効”が”モニタリング機能：無効”に切り替わります。

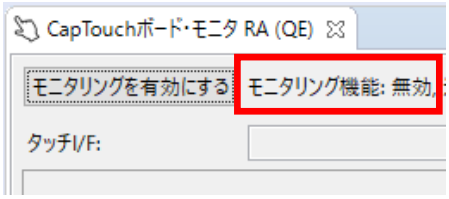


図 66　モニタリング無効化

2. デバッグセッションを終了します。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終了します。

3. エミュレータを PC と AE-CAP1-S3 V1.1 ボードから外します。USB ケーブルが PC とボードに正しく接続されていることを確認します。

   【注】：エミュレータを接続した状態でも動作可能ですが、ここではエミュレータなしでモニタリングが可能なことを確認するため、エミュレータを外します。

4. AE-CAP1-S3 V1.1 ボード上の RESET スイッチを押して、リセットします。

5. [CapTouch メイン/センサ・チューナ RA,RL78,Synergy (QE)]を表示し、[プロジェクトの選択]で"Capacitive_Touch_Project_Example"、[構成の選択]で"Capacitive_Touch_Project_Example.tifcfg"が選択されていることを確認します。

図 67　プロジェクトの選択と構成の選択

6. [CapTouch メイン/センサ・チューナ RA,RL78,Synergy (QE)]から動作確認の[接続]を選択し、シリアル通信によるモニタリング機能を有効にします。

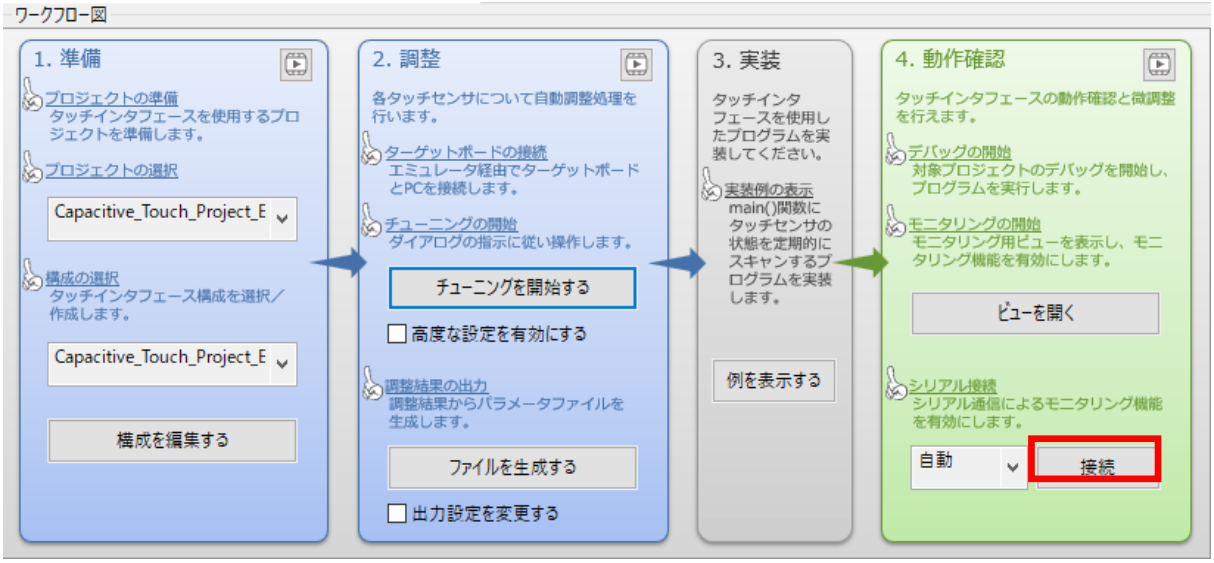


図 68　[接続]ボタン

7. 画面下の[コンソール]ウィンドウに、”COM*n* に接続しました。” のメッセージが表示されることを確認します。

【注】：接続先の COM ポートの番号は、PC の環境によって異なります。

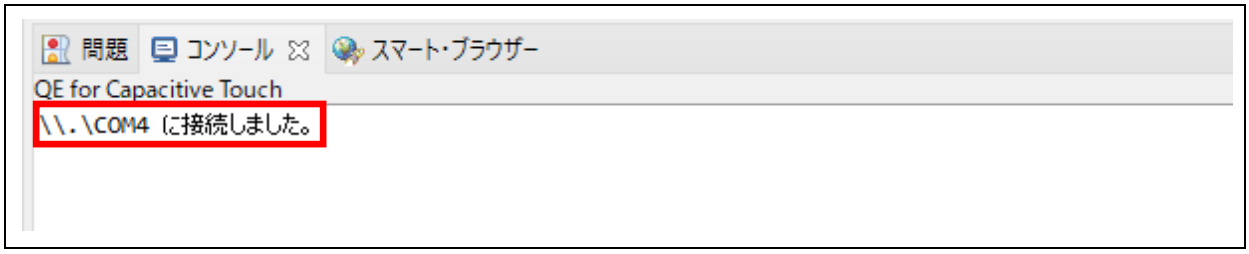


図 69　[コンソール]ウィンドウの出力

8. 以降の手順は、「6.7 [式]ウィンドウとQE for Capacitive Touch [RA,RL78,Synergy]によるモニタリング」の9項以降と同様です。

7.　変更後の qe_touch_sample.c のリスト

```
/*********************************************************************
*
* FILE : qe_sample_main.c
* DATE : 2021-03-11
* DESCRIPTION : Main Program
*
* NOTE:THIS IS A TYPICAL EXAMPLE.
*
*********************************************************************/
#include "qe_touch_config.h"
#define TOUCH_SCAN_INTERVAL_EXAMPLE (20)       /* milliseconds */

void qe_touch_main(void);

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

void qe_touch_main(void)
{
    ssp_err_t err;

    /* Open Touch middleware */
    err = g_qe_touch_instance_config01.p_api->open(g_qe_touch_instance_config01.p_ctrl,
g_qe_touch_instance_config01.p_cfg);
    if (SSP_SUCCESS != err)
    {
        while (true) {}
    }

    /* Main loop */
    while (true)
    {
        /* for [CONFIG01] configuration */
        err = g_qe_touch_instance_config01.p_api->scanStart(g_qe_touch_instance_config01.p_ctrl);
```

```
        if (SSP_SUCCESS != err)
        {
            while (true) {}
        }
        while (0 == g_qe_touch_flag) {}
        g_qe_touch_flag = 0;

        err = g_qe_touch_instance_config01.p_api->dataGet(g_qe_touch_instance_config01.p_ctrl,
&button_status, NULL, NULL);
        if (SSP_SUCCESS == err)
        {
            /* TODO: Add your own code here. */
        }



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

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

https://www.renesas.com/

静電容量式タッチセンサユニット関連ページ

https://www.renesas.com/solutions/touch-key
https://www.renesas.com/ssp
https://www.renesas.com/qe-capacitive-touch

お問い合わせ

https://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	Jun.23.21	-	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、ノイズ輻射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

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。

https://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。

© 2021 Renesas Electronics Corporation. All rights reserved.