RX140 タッチキー評価キット

[QE CapTouch 編]
マニュアル

-本書を必ずよく読み、ご理解された上でご利用ください

株式会社北斗電子

REV.1.0.1.0

ー目　次ー

注意事項

本書を必ずよく読み、ご理解された上でご利用ください

【ご利用にあたって】

1. 本製品をご利用になる前には必ず取扱説明書をよく読んで下さい。また、本書は必ず保管し、使用上不明な点がある場合は再読し、よく理解して使用して下さい。
2. 本書は株式会社北斗電子製マイコンボードの使用方法について説明するものであり、ユーザシステムは対象ではありません。
3. 本書及び製品は著作権及び工業所有権によって保護されており、全ての権利は弊社に帰属します。本書の無断複写・複製・転載はできません。
4. 弊社のマイコンボードの仕様は全て使用しているマイコンの仕様に準じております。マイコンの仕様に関しましては製造元にお問い合わせ下さい。弊社製品のデザイン・機能・仕様は性能や安全性の向上を目的に、予告無しに変更することがあります。また価格を変更する場合や本書の図は実物と異なる場合もありますので、御了承下さい。
5. 本製品のご使用にあたっては、十分に評価の上ご使用下さい。
6. 未実装の部品に関してはサポート対象外です。お客様の責任においてご使用下さい。

【限定保証】

1. 弊社は本製品が頒布されているご利用条件に従って製造されたもので、本書に記載された動作を保証致します。
2. 本製品の保証期間は購入戴いた日から1年間です。

【保証規定】

保証期間内でも次のような場合は保証対象外となり有料修理となります

1. 火災・地震・第三者による行為その他の事故により本製品に不具合が生じた場合
2. お客様の故意・過失・誤用・異常な条件でのご利用で本製品に不具合が生じた場合
3. 本製品及び付属品のご利用方法に起因した損害が発生した場合
4. お客様によって本製品及び付属品へ改造・修理がなされた場合

【免責事項】

弊社は特定の目的･用途に関する保証や特許権侵害に対する保証等、本保証条件以外のものは明示・黙示に拘わらず一切の保証は致し兼ねます。また、直接的・間接的損害金もしくは欠陥製品や製品の使用方法に起因する損失金・費用には一切責任を負いません。損害の発生についてあらかじめ知らされていた場合でも保証は致し兼ねます。
ただし、明示的に保証責任または担保責任を負う場合でも、その理由のいかんを問わず、累積的な損害賠償責任は、弊社が受領した対価を上限とします。本製品は「現状」で販売されているものであり、使用に際してはお客様がその結果に一切の責任を負うものとします。弊社は使用または使用不能から生ずる損害に関して一切責任を負いません。
保証は最初の購入者であるお客様ご本人にのみ適用され、お客様が転売された第三者には適用されません。よって転売による第三者またはその為になすお客様からのいかなる請求についても責任を負いません。
本製品を使った二次製品の保証は致し兼ねます。

概要

本書は、
「RX140 タッチキー評価キット」
でタッチキーのルネサスエレクトロニクス製ミドルウェアである QE for Cap Touch の使い方について解説を行うものです。

RX140 マイコンはルネサスの新しいタッチキーユニットである CTSU2 を搭載しています。

本書では、キット付属の 2 種類のタッチキー基板、「自己容量タイプタッチキー基板(S16A)」及び「相互容量タイプタッチキー基板(D55A)」を、QE for Cap Touch で使う方法を示しています。

QE for Cap Touch は、複雑なタッチキーユニットの初期設定や、タッチした際の閾値のチューニングや測定値のモニタ等が行える高機能なソフトとなります。

タッチキー(CTSU2)のプログラムをスクラッチで書き下す場合は、本書とは別なマニュアルで、タッチキー(CTSU2)のソフトウェア編マニュアルがありますので、そちらも必要に応じて参照ください。

1. 本書で説明する範囲

1.1. CD フォルダ構成

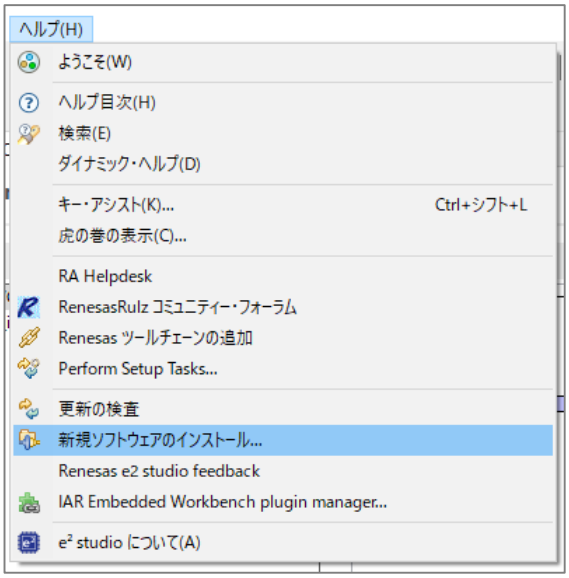
PROJECT/	RX/	RX140_QE_CAP_TOUCH_S.zip	RX140 向け e2studio プロジェクト （自己容量, S16A タッチパッド向け）
	RX/	RX140_QE_CAP_TOUCH_M.zip	RX140 向け e2studio プロジェクト （相互容量, D55A タッチパッド向け）

e2studio のプロジェクトのアーカイブとなっていますので、ワークスペースにインポートしてください。

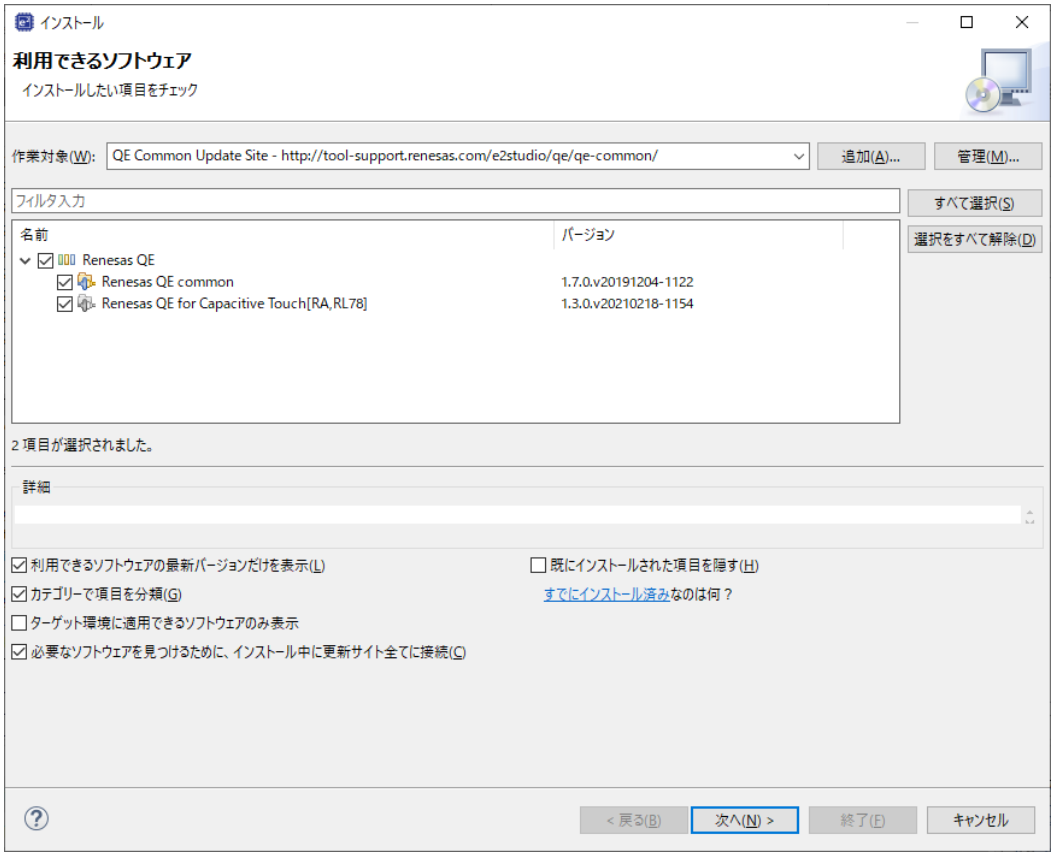
2. 初期設定

2.1. QE ツールのインストール

タッチキーは、Renesas QE Cap-touch でサポートされる機能となりますので、e2studio に、Renesas QE Cap-touch をインストールしてください。



ヘルプ－新規ソフトウェアのインストールから、



QE Common Update Site を選択して、

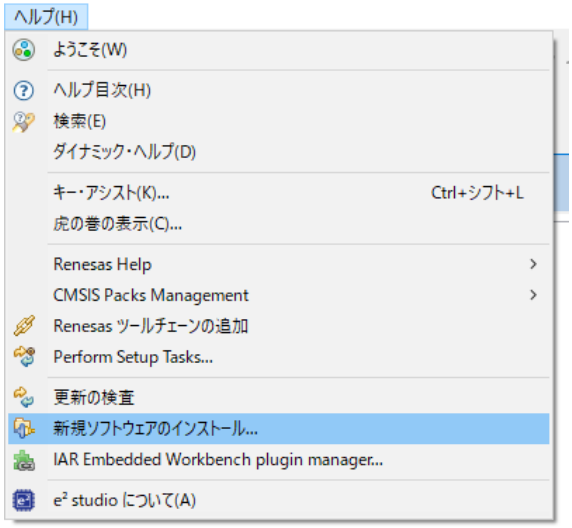
Renesas QE common

Resnesas QE for Capacitive Touch

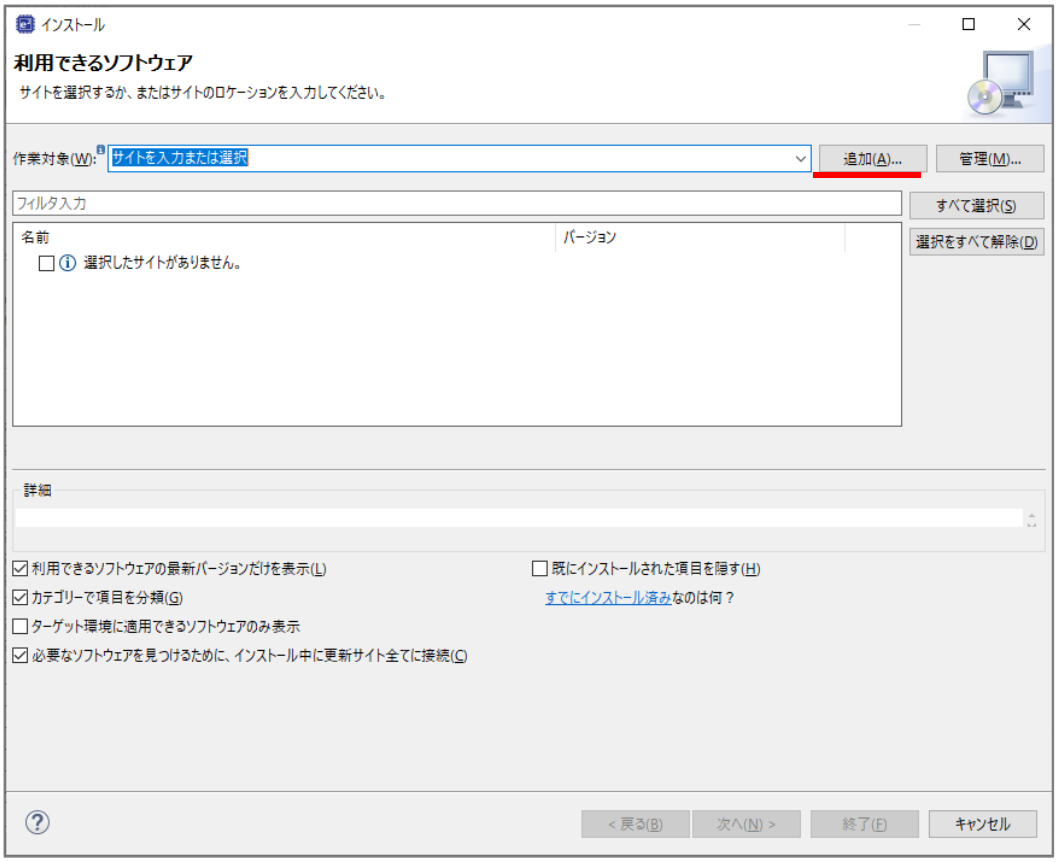
をインストールしてください。

・ローカルインストールの場合

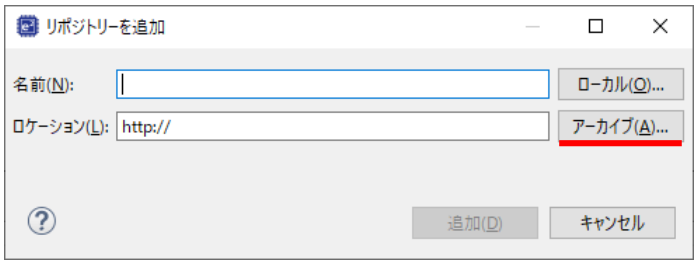
インターネットに直接接続していない場合は、ルネサスエレクトロニクスのサイトから zip ファイルをダウンロードして、ローカルファイルからインストールしてください。



ヘルプ－新規ソフトウェアのインストール

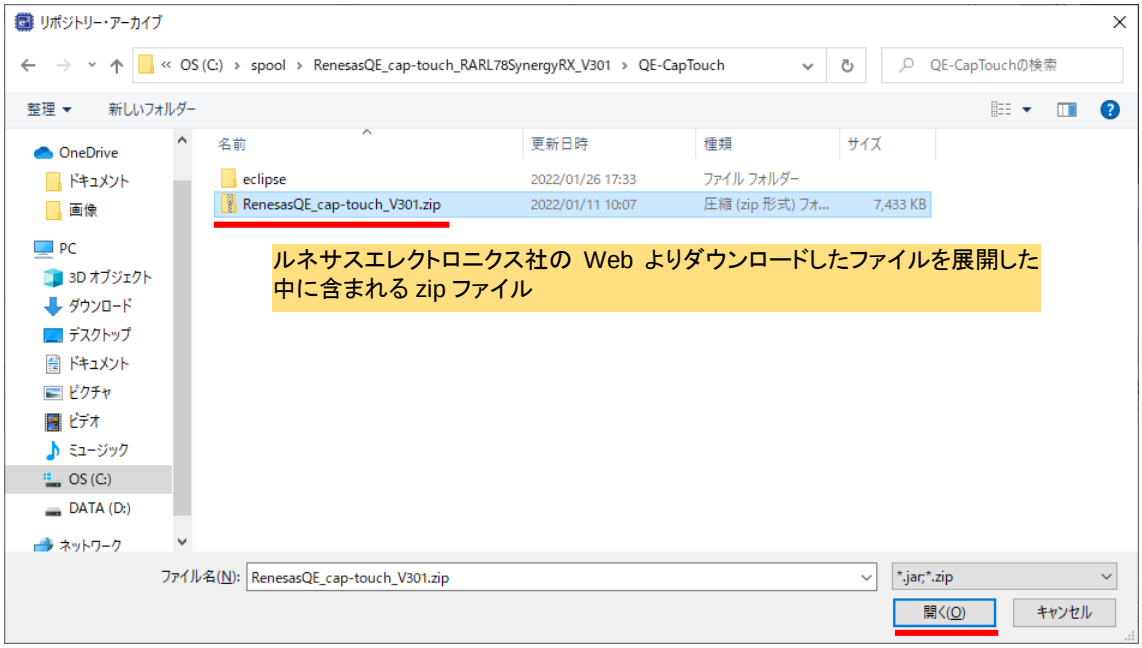


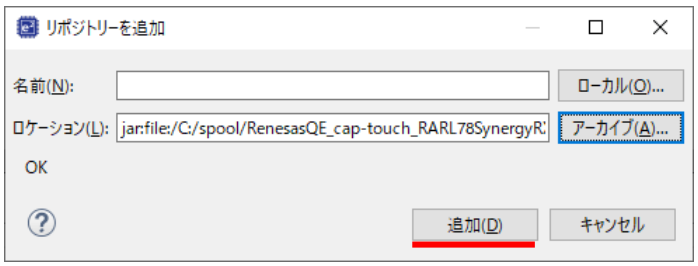
追加



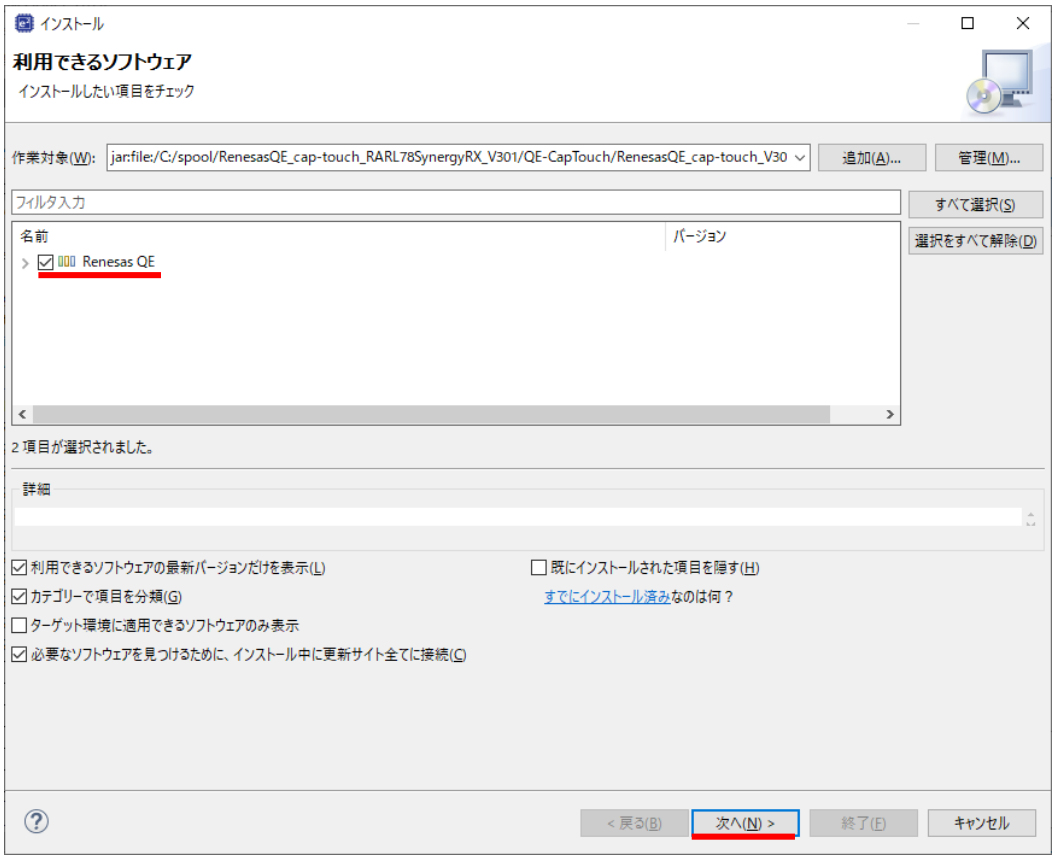
アーカイブで、QE Cap Touch のアーカイブファイル（.zip ファイル）を選択。

※RenesasQE_cap-touch_RARL78SynergyRX_V303.zip を展開し、展開したフォルダに含まれる RenesasQE_cap-touch_RARL78SynergyRX_V303¥QE-CapTouch¥**RenesasQE_cap-touch_V303.zip** 上記太字の zip ファイルを開く（V3.03 の場合、バージョンは読み替えてください）

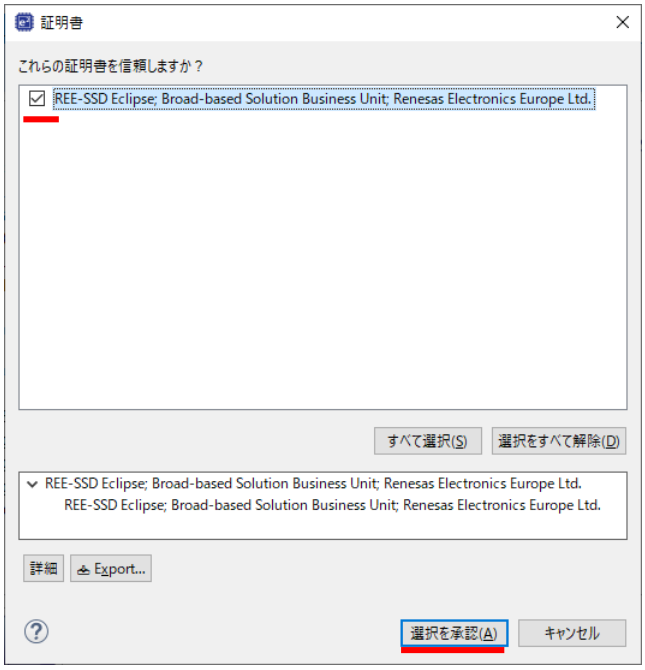




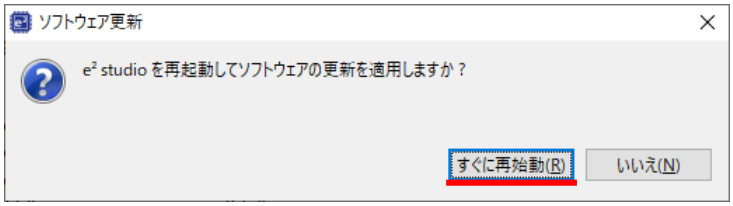
追加



チェックを入れて次へ



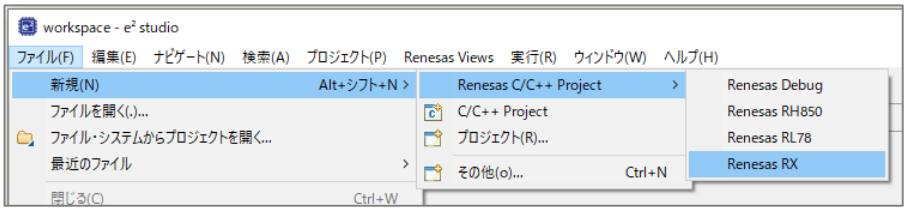
証明書関連は承認してください。



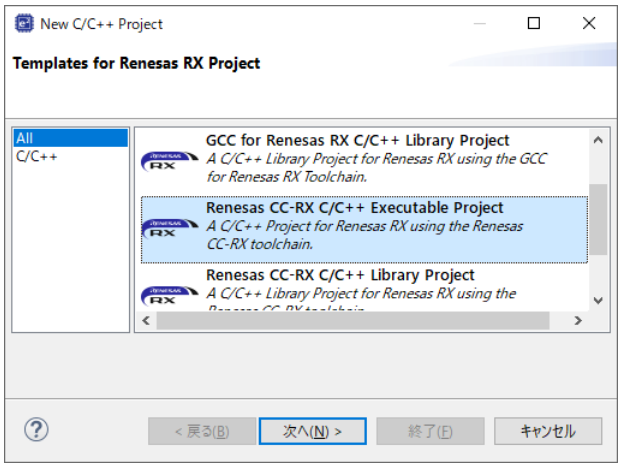
一度 e2studio を再起動して、インストールを有効化してください。

2.2. プロジェクトの新規作成

e2studio（+FSP 環境）上で

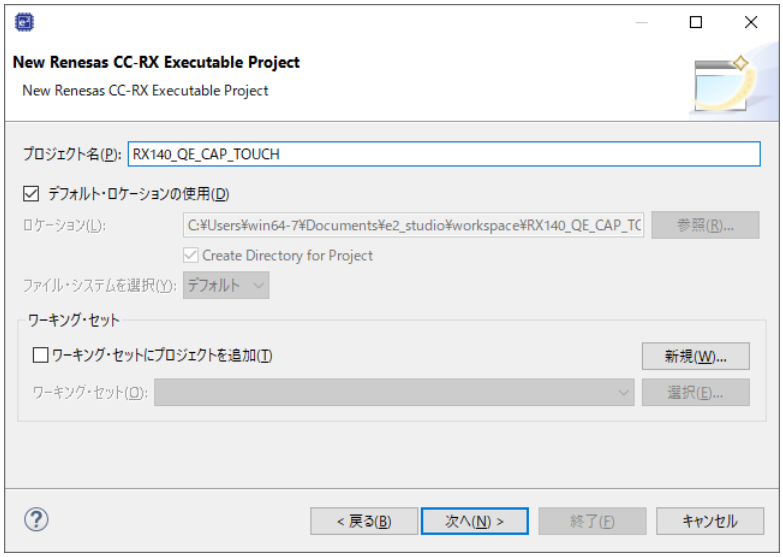


ファイル - 新規 - Renesas C/C++ Project - Renesas RX
を選択。

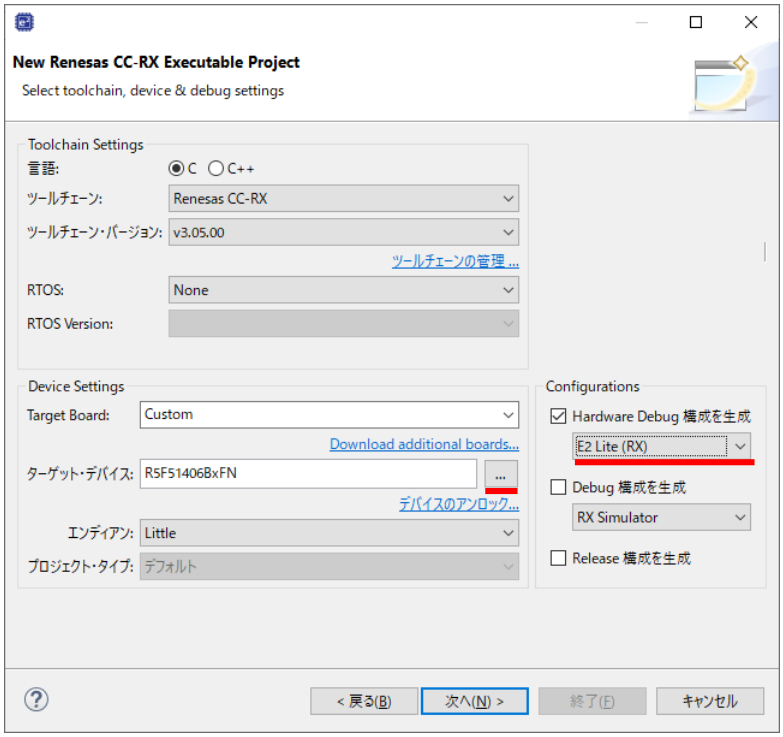


Renesas CC-RX C/C++
Executable Project を選択

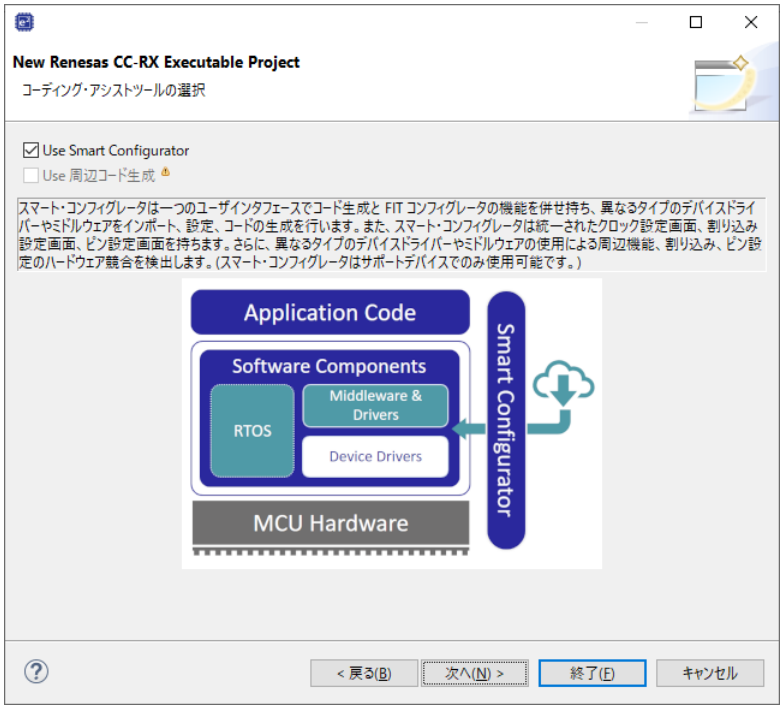
次へ。



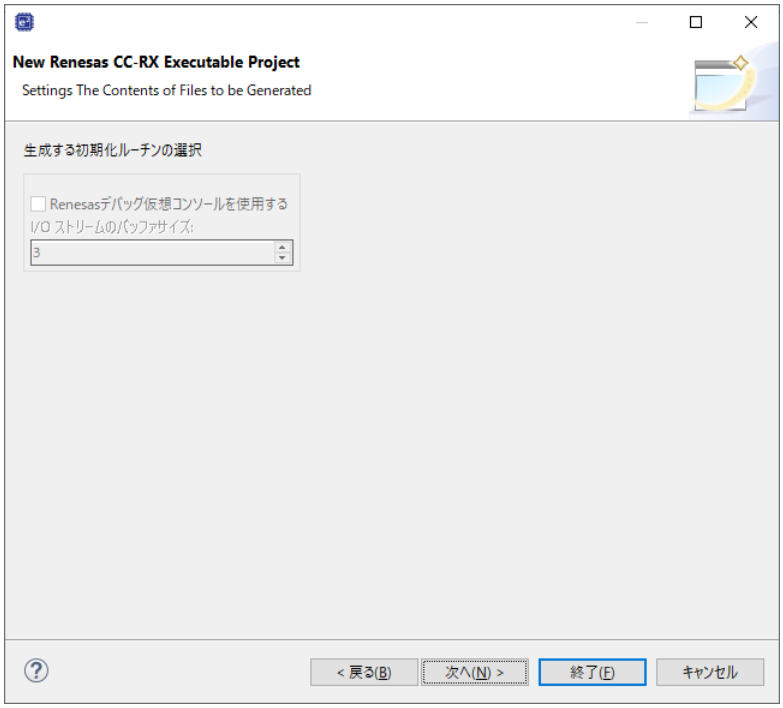
任意のプロジェクト名を入力。次へ。



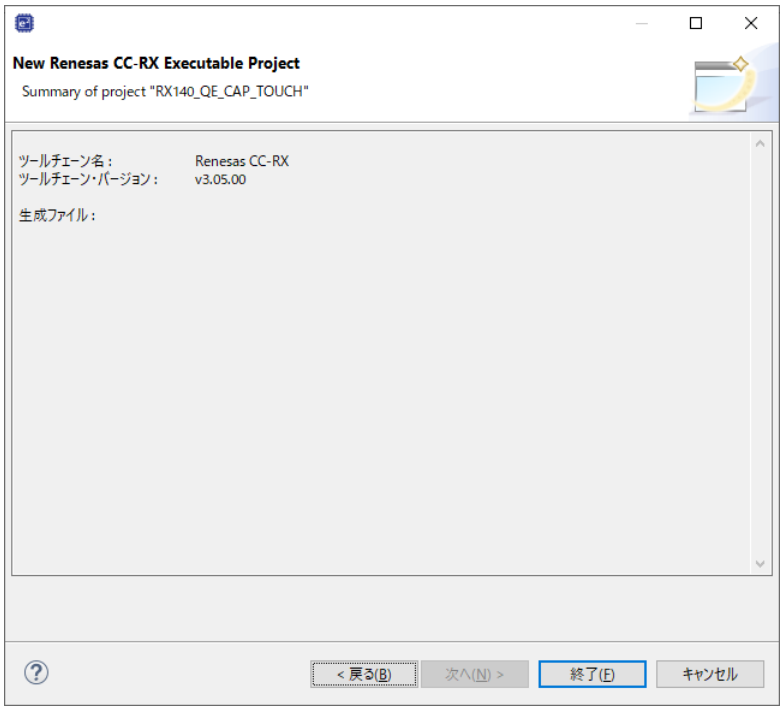
ターゲット・デバイスの右側のボタンを押し、デバイス選択画面を開き
RX100 - RX140 - RX140 - 80pin - R5F51406BxFN を選択。
お手持ちのデバッガを選択（ここでは E2 Lite を選択）
次へ。



Use Smart Configurator にチェックが入った状態（デフォルト）で、次へ。

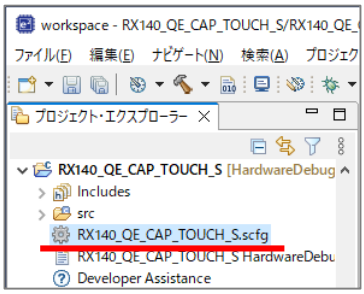

New Renesas CC-RX Executable Project
Settings The Contents of Files to be Generated
生成する初期化ルーチンの選択
Renesasデバッグ仮想コンソールを使用する
I/O ストリームのバッファサイズ:
3
< 戻る(B)
次へ(N) >
終了(F)
キャンセル

次へ。

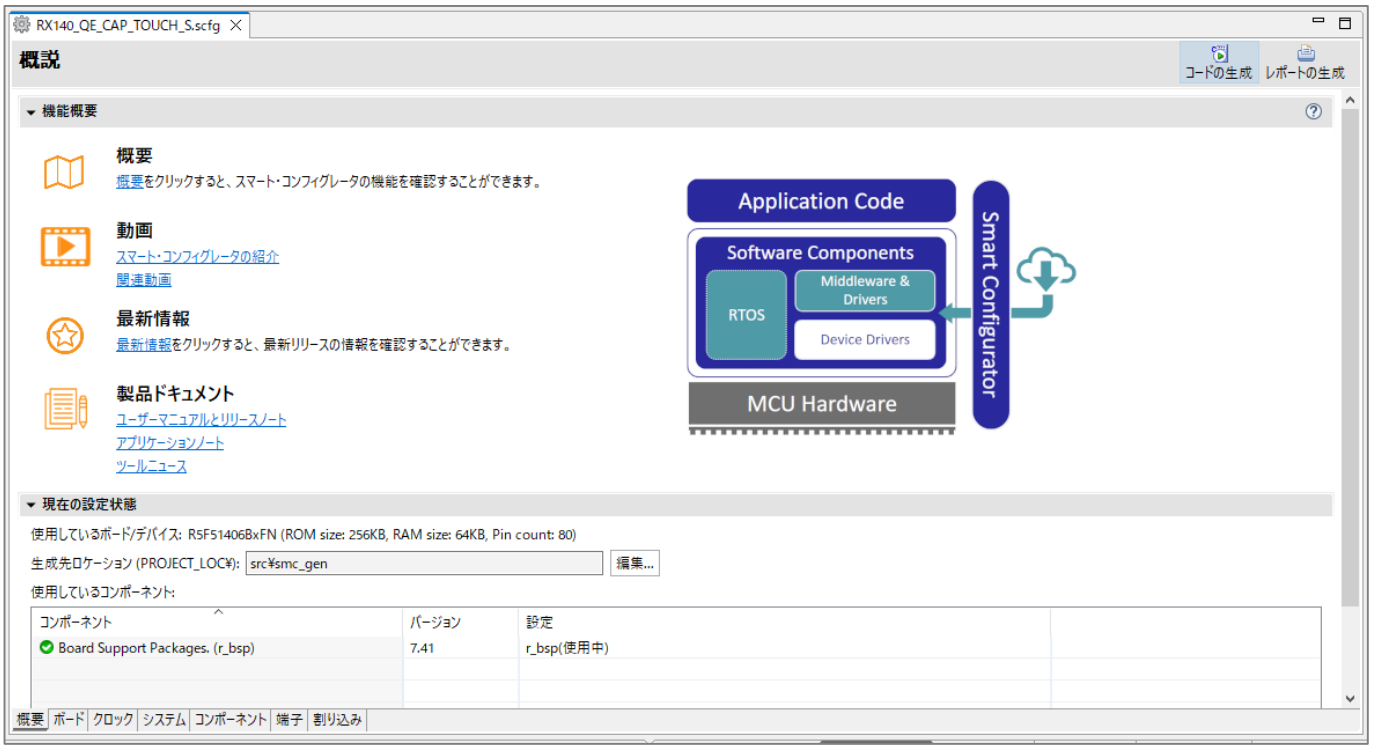

New Renesas CC-RX Executable Project
Summary of project "RX140_QE_CAP_TOUCH"
ツールチェーン名：
Renesas CC-RX
ツールチェーン・バージョン：
v3.05.00
生成ファイル：
< 戻る(B)
次へ(N) >
終了(F)
キャンセル

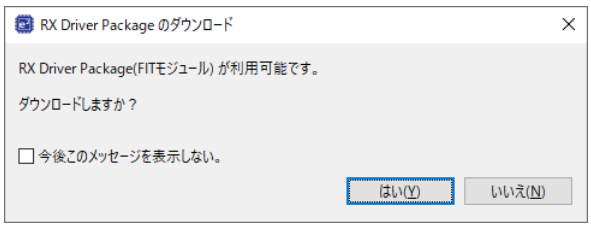
終了。

2.3. スマートコンフィグレータ上の設定



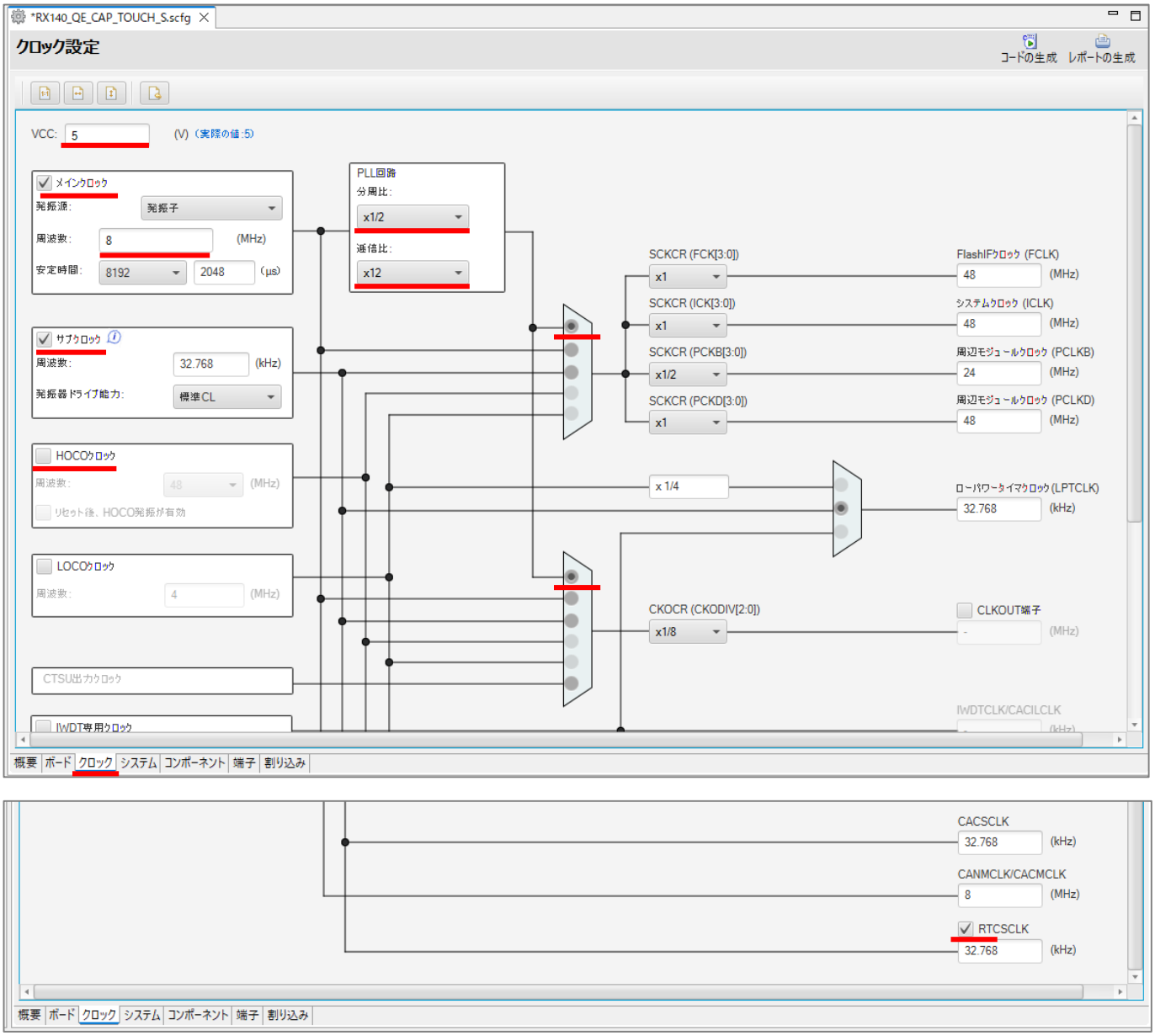
プロジェクト名.scfg がスマートコンフィグレータの設定となります。





上記メッセージが出た場合は、FIT モジュールのダウンロード、インストールを行っておいてください。

2.3.1. クロックタブ



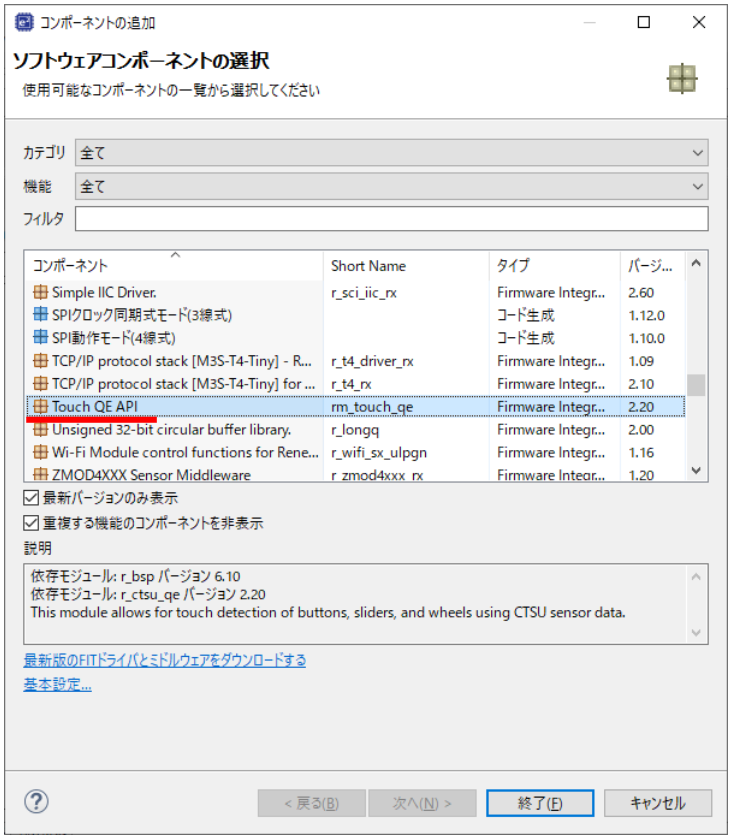
クロックの設定を変更

（メインクロック(8MHz)を使用する設定、サブクロックを有効化、HOCO を止める）

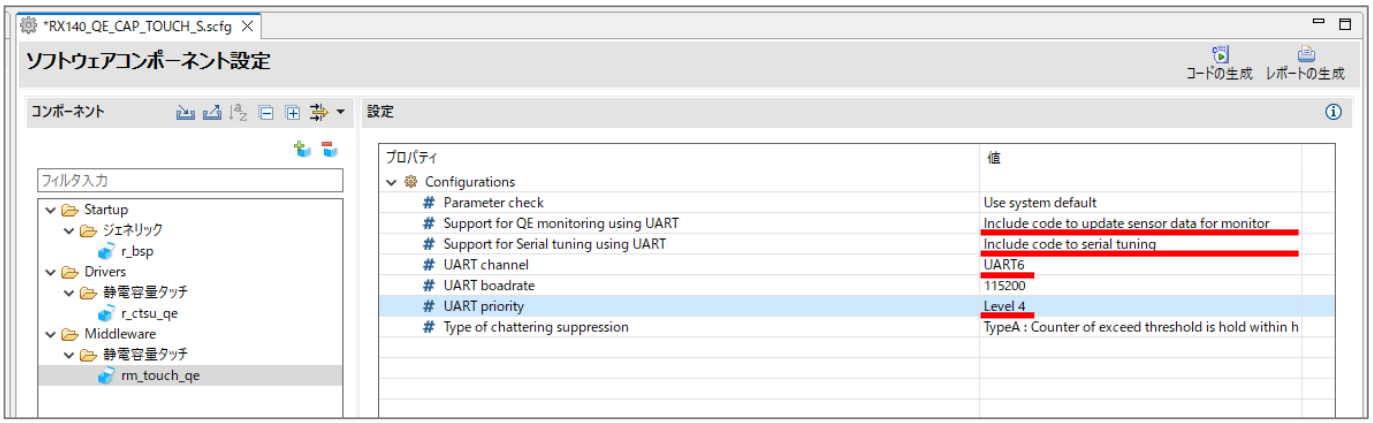
※上記は一例です、クロックの設定は任意です

2.3.2. コンポーネントタブ

コンポーネントの追加アイコンを押す。

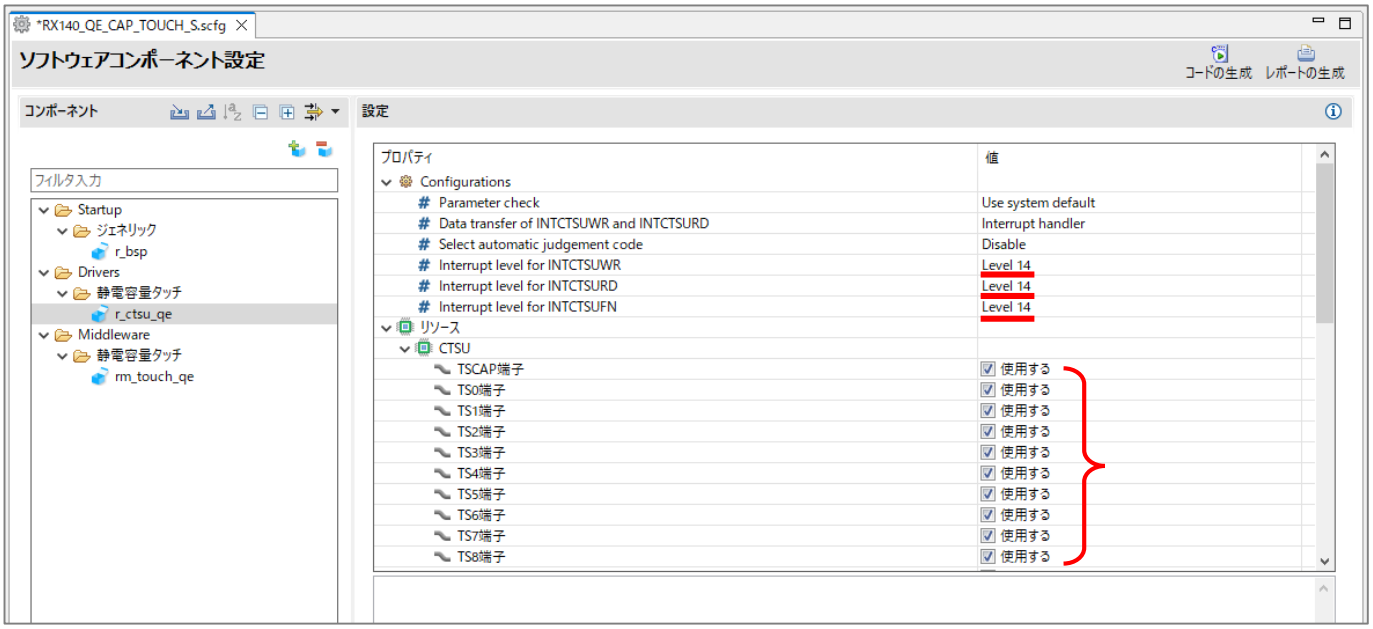


Touch QE API を選択、「終了」を押す。
（Touch QE API が表示されない場合は、「最新版の FIT ドライバとミドルウェアをダウンロードする」を押して、FIT の追加モジュールをダウンロード、インストールしてください。



rm_touch_qe コンポーネントの設定は、RX140 タッチキー評価キットの、LCD 接続基板に付いている UART のコネクタ(J5)に、USB-Serial 変換機器（USB-1S(JST)等）を接続する場合は、「モニタで UART を使用する」「チューニングで UART を使用する」項目の設定を有効化してください。割り込み優先度は、1（最低）～15(最高）となりますので、適当に選択してください。

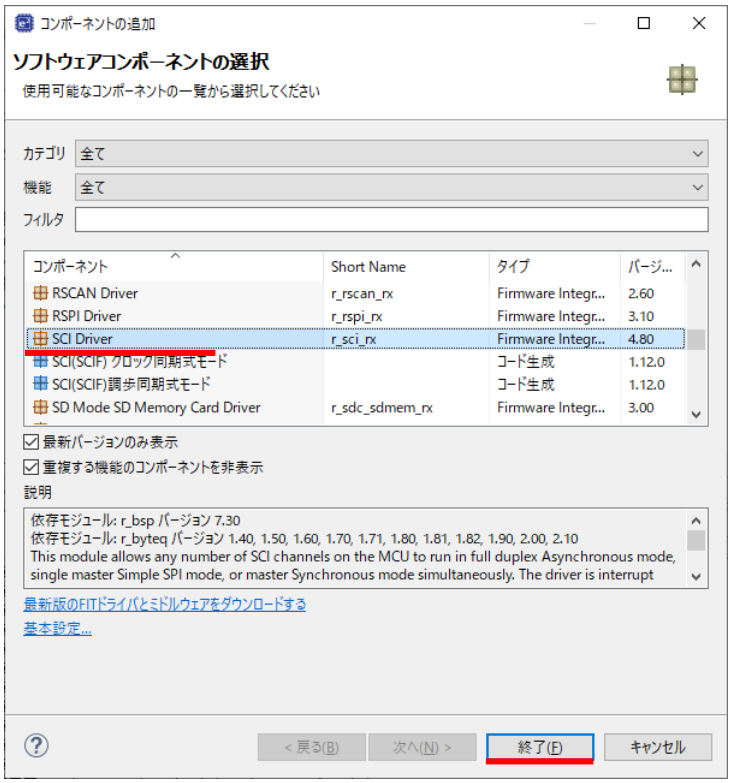
UART を使用しない場合は、デフォルトから変更の必要はありません。



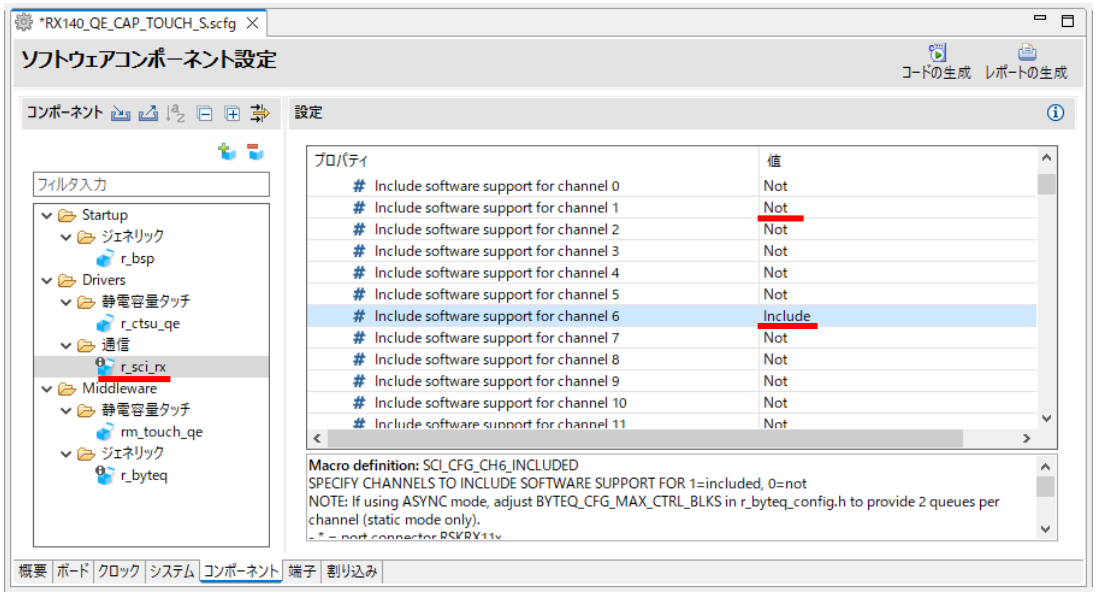
r_ctsu_qe コンポーネントの設定は、割り込み優先度はデフォルト 1（最低）になっているので、レベルを上げておきます（任意）。TSCAP 端子は、「使用する」にチェックを入れます。

自己容量基板(S16A)を使用する場合は、TS0~TS15 の端子を「使用する」にチェックを入れます。

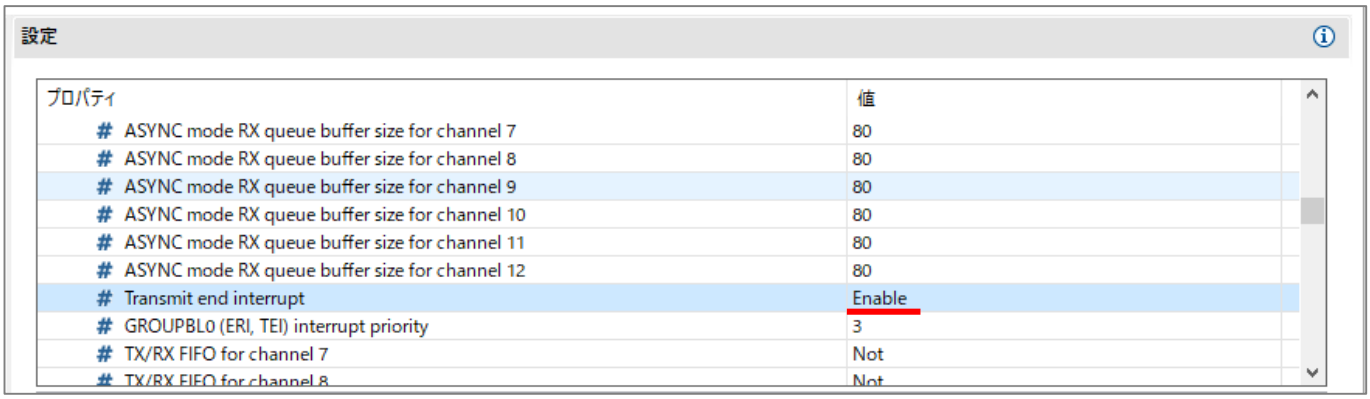
相互容量基板(D55A) を使用する場合は、TS0~TS9 の端子を「使用する」にチェックを入れます。



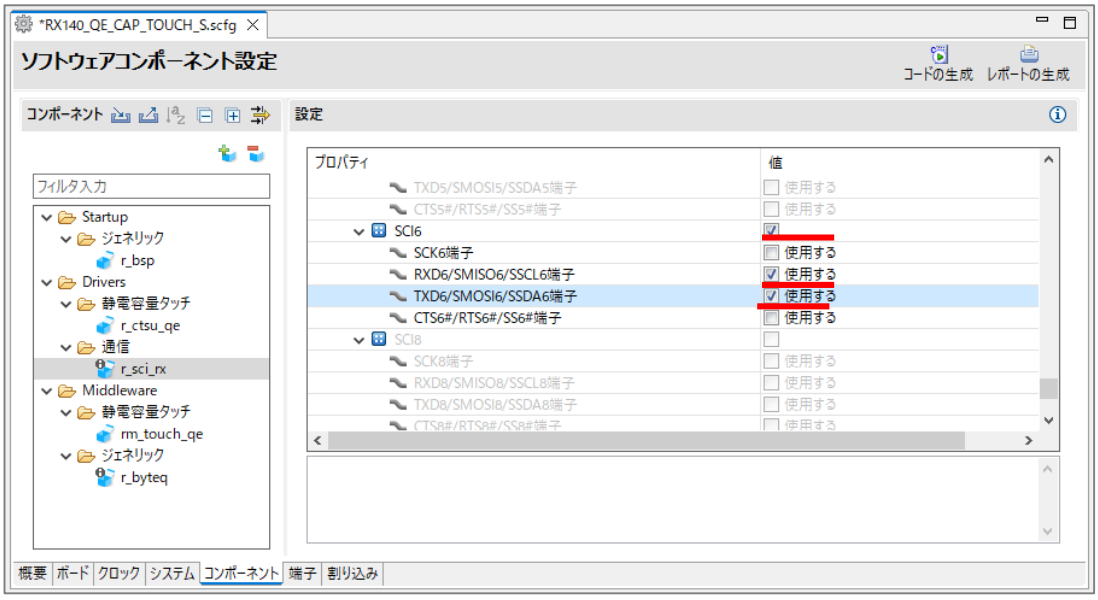
SCI Driver のコンポーネントを追加。終了。



デフォルトでは、channel1 が選択されていますので、channel1 を「Not」に変更。channel6 を「Include」に変更。

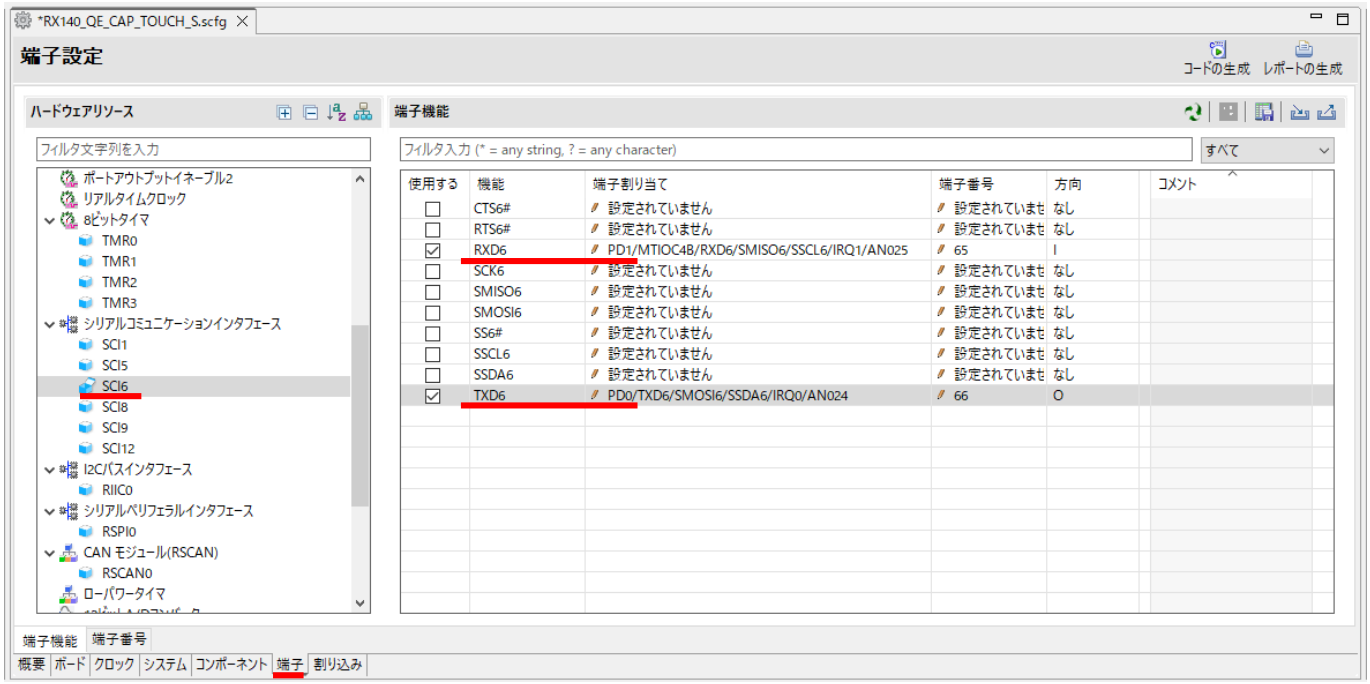


Transmit end interrupt を Enable に変更。



SCI6 にチェック。RXD6, TXD6 を「使用する」にチェック。

2.3.3. 端子タブ

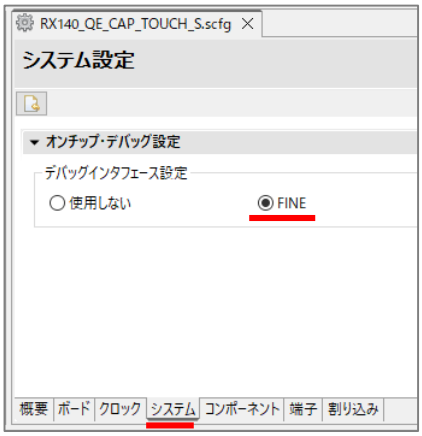


端子タブを選択。SCI6 の端子設定を変更する。

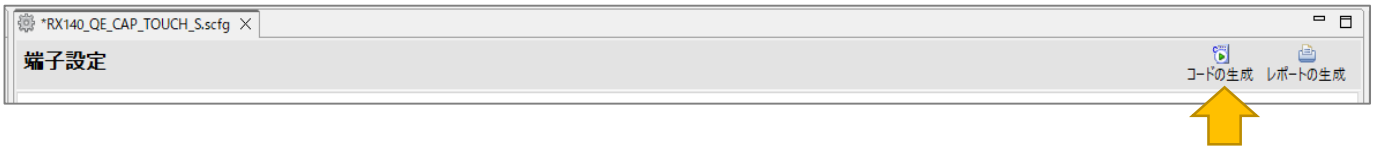
RXD6：PD1 を選択
TXD6：PD0 を選択

（※LCD 接続基板上の J5 コネクタを使用して PC と通信を行う場合に上記設定とする）
（基板上の拡張 I/O 端子から信号引き出しを行う場合は、使用する信号端子に応じて設定）

2.3.1. システムタブ



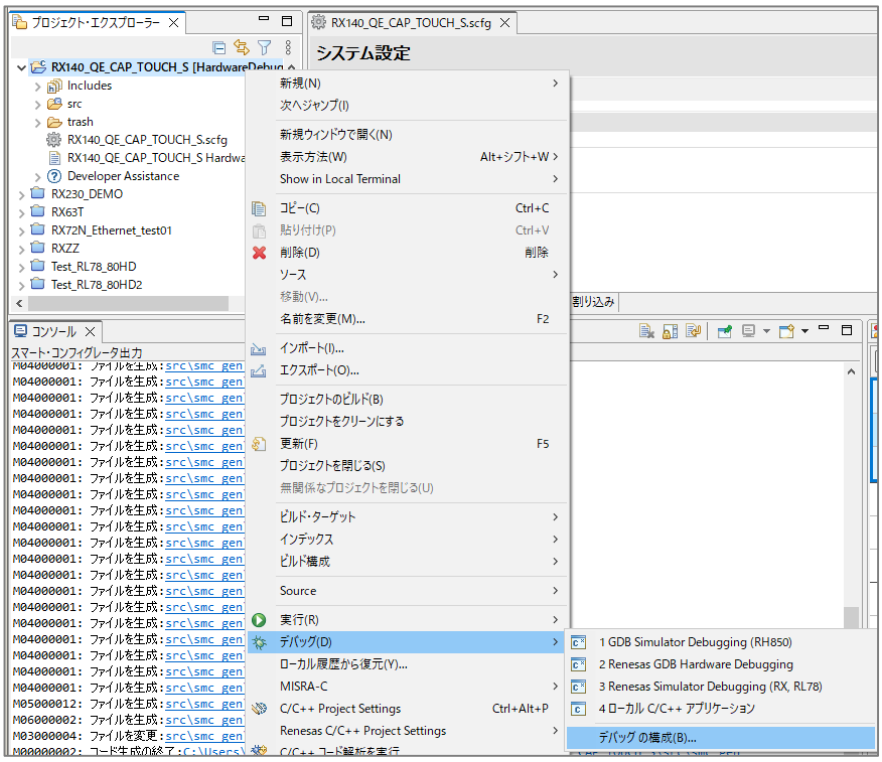
「システム」タブのデバッグインタフェース設定は「FINE」を選択してください。



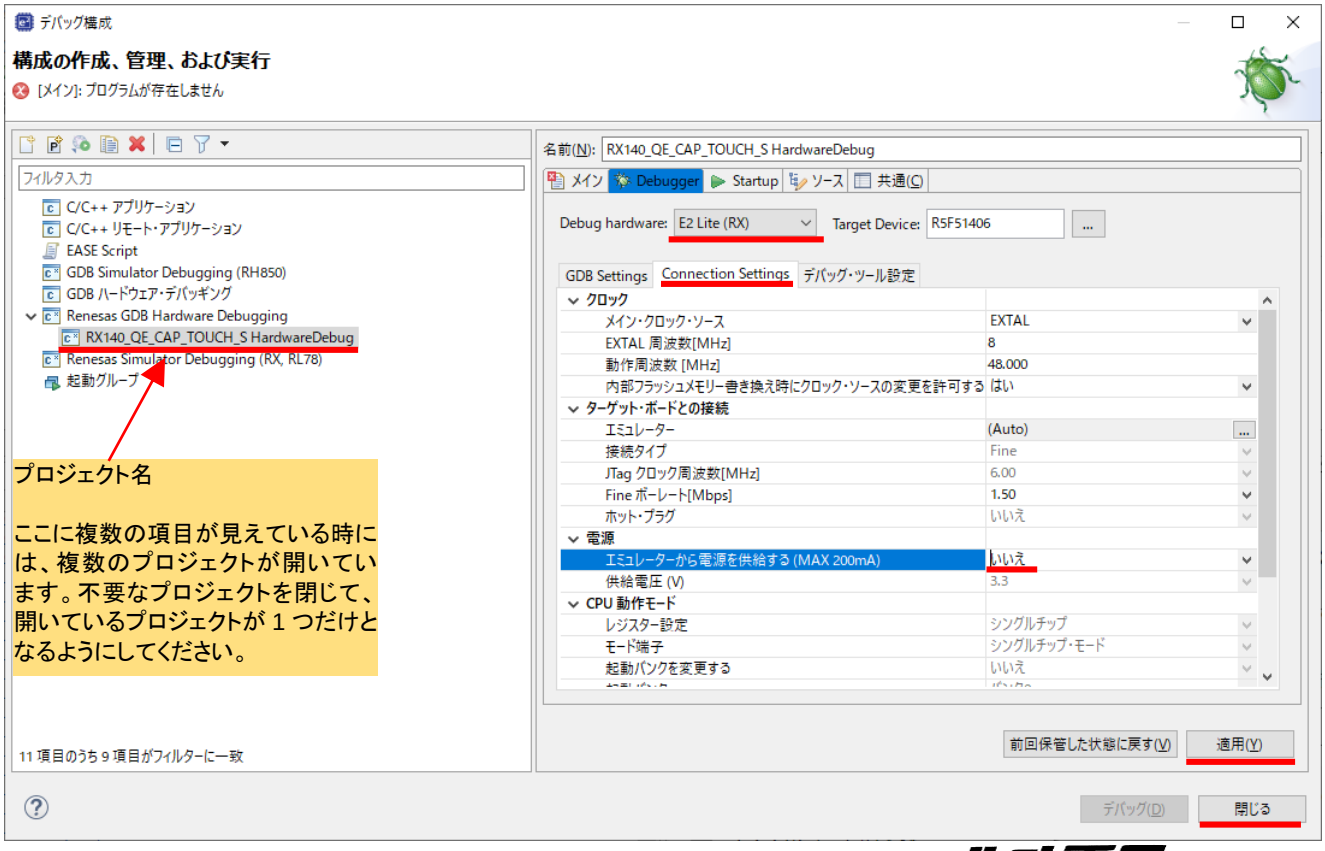
一連の設定が終わったら、「コード生成」のボタンを押す。
（何か設定を変えた場合も、「コード生成」のボタンを押してください。）

2.4. エミュレータ（デバッガ）の接続設定

プロジェクトエクスプローラーから、対象プロジェクトを右クリック。



デバッグ – デバッグの構成



プロジェクト名 HardwareDebug の Debugger タブ
Debug hardware 使用するデバッガを指定
Connection Settings タブ 電源－エミュレータから電源を供給する 「いいえ」
「適用」「閉じる」

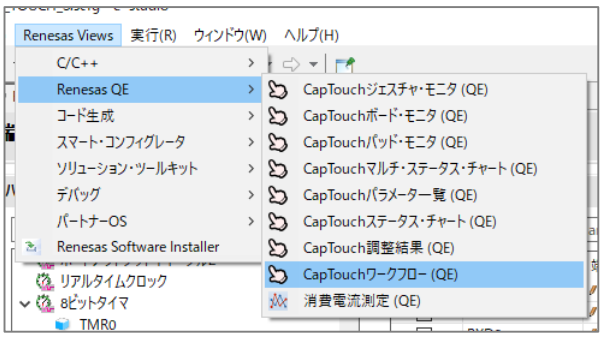
※エミュレータから電源を供給する事も可能ですが、その場合は
・別な箇所（電源コネクタ等）から電源を供給しない
様にしてください。

※なお、E2Lite をお使いの場合は、エミュレータから供給する電圧は、3.3V しか選択できません
（E2 では、5V と 3.3V が選択可能です）

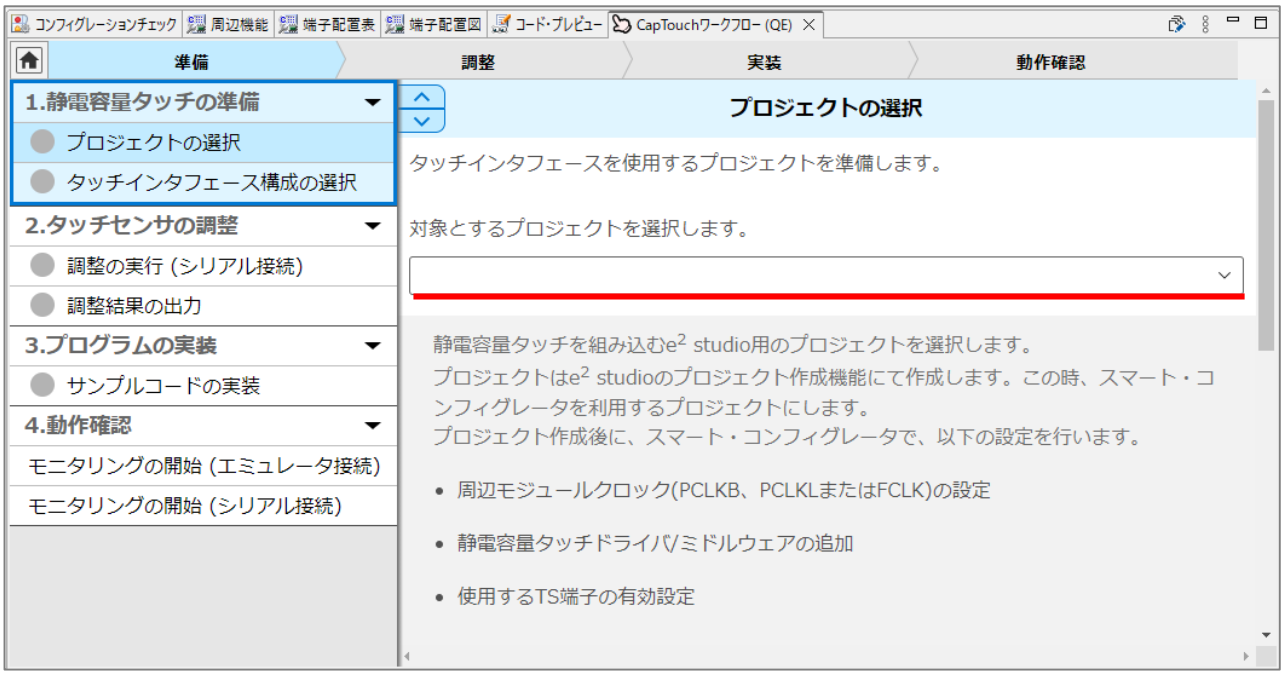
※「エミュレータ」と「デバッガ」、「Debugger」は同義です

3. QE CapTouch の使用

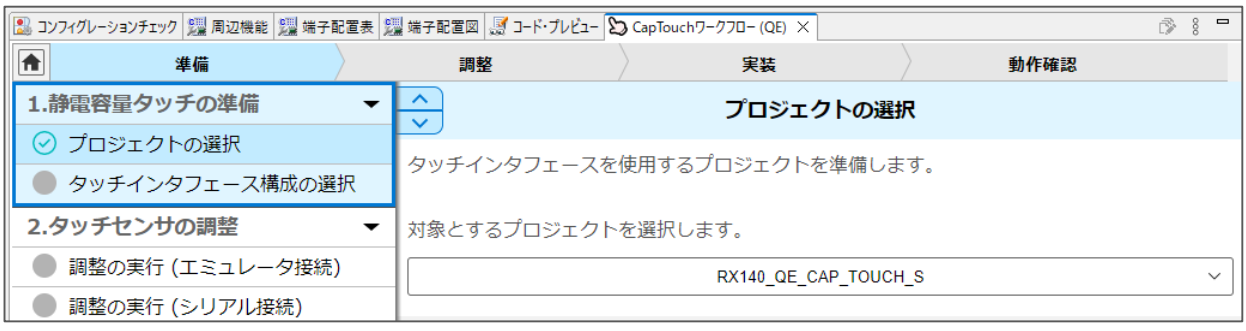
3.1. CapTouch の起動

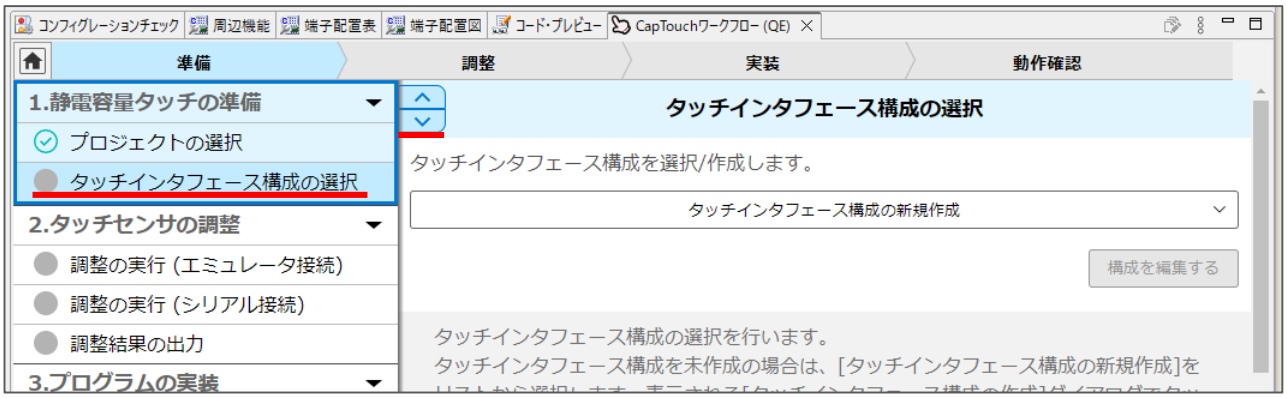


Renesas Views -「CapTouch ワークフロー」を実行します。



作成したプロジェクトを、プルダウンメニューで選択してください。

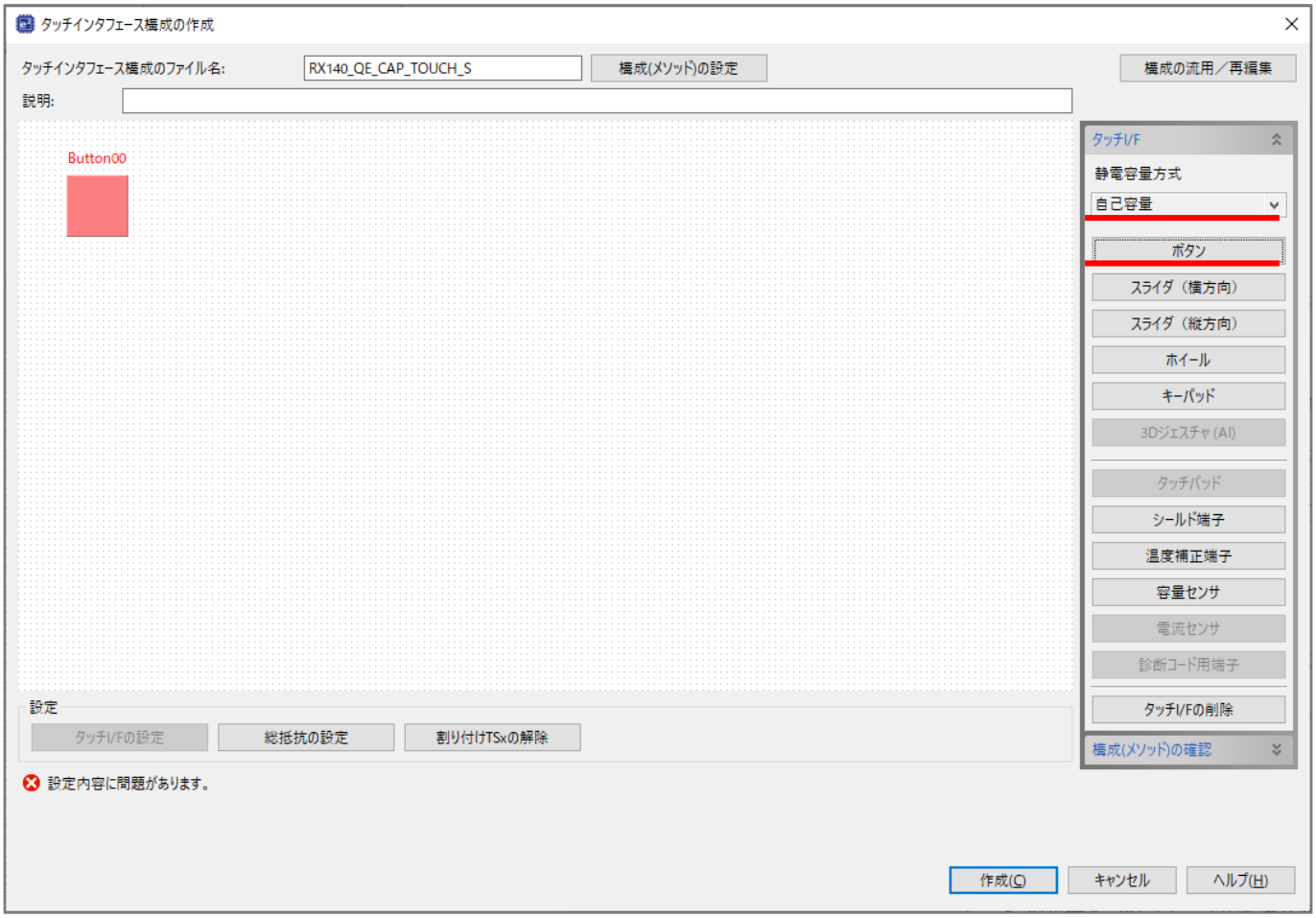




タッチインタフェース構成の選択（もしくは、下矢印）で次の画面に移り、「タッチインタフェース構成の新規作成」を選択。

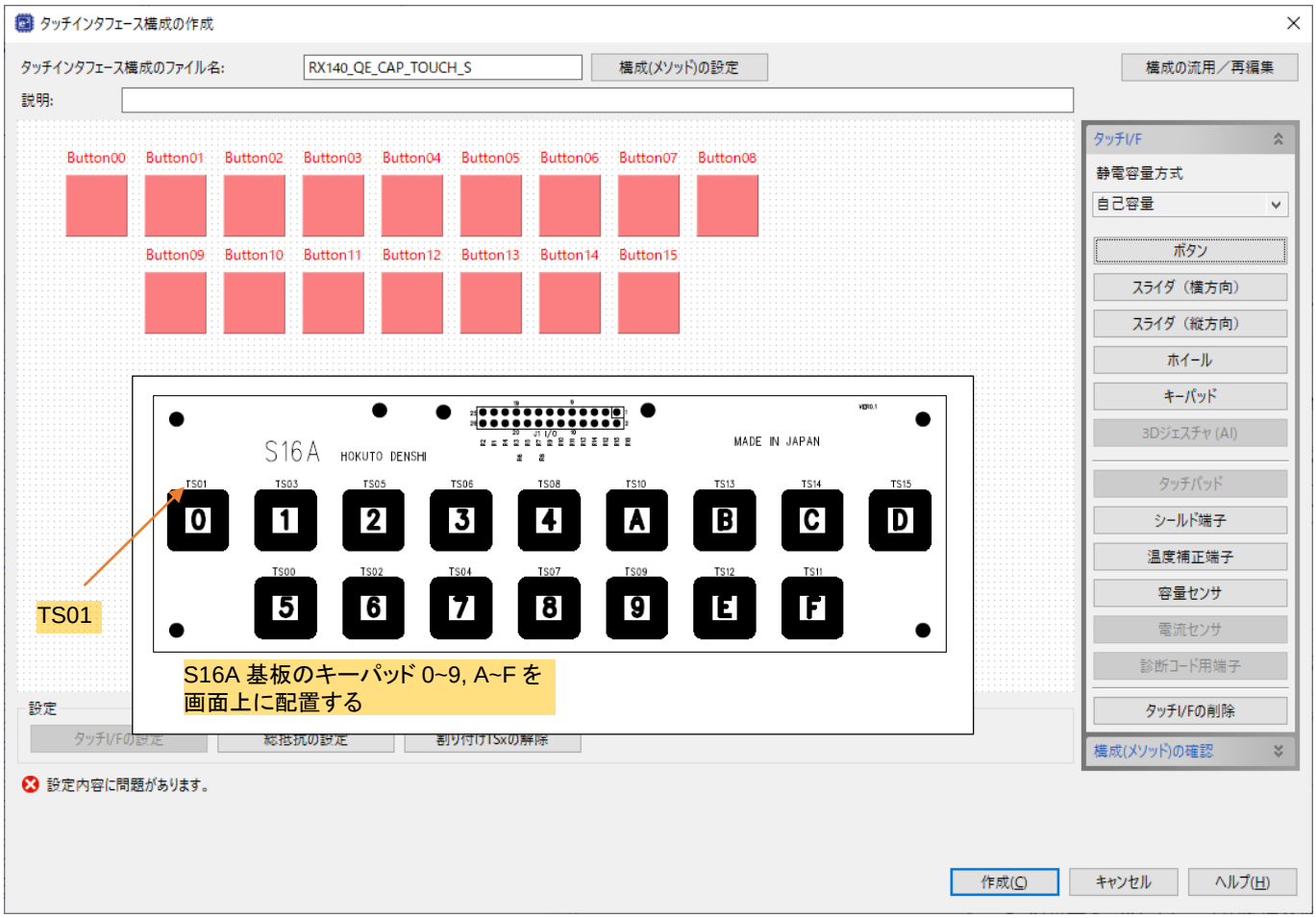
3.2. タッチキーインタフェースの作成

3.2.1. 自己容量基板(S16A)

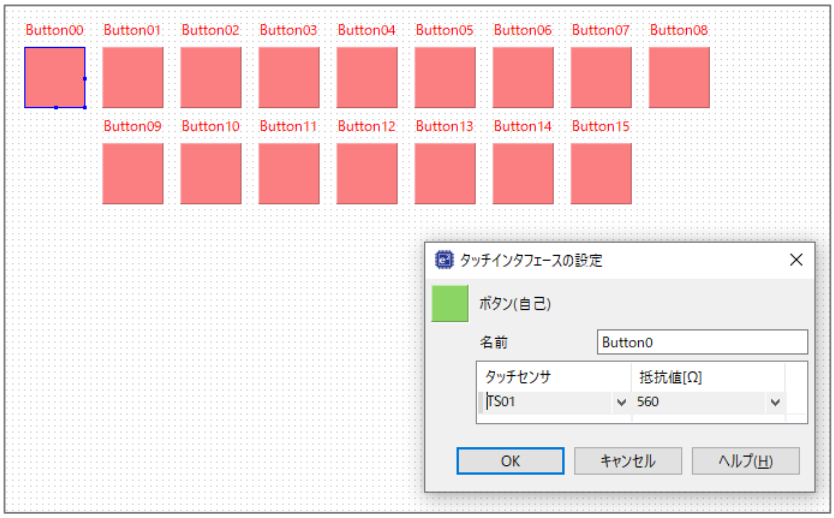


静電容量方式「自己容量」を選択。

「ボタン」を押し、ボタンをレイアウト上に配置していきます。



S16A の基板に合わせて、0~9,A~F の 15 個のキーパッドを、基板と同じような画面イメージとなる様にボタンを画面上に配置します。



Button00 をダブルクリックして、このボタンに関する設定を行います。

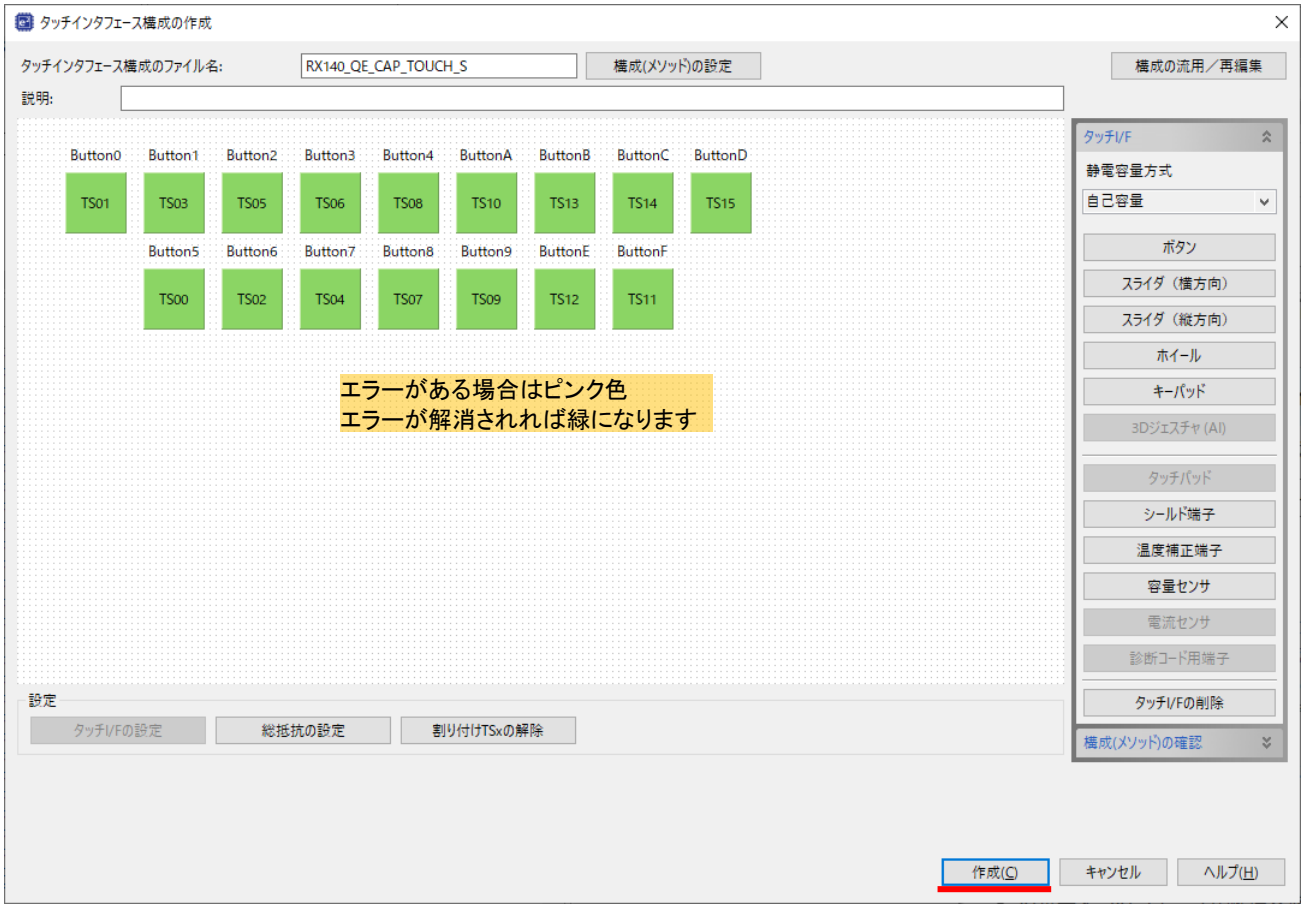
接続されている TS 端子は、TS01 なので、TS01 を選択します。抵抗値は、560Ω(デフォルト値)のままとします。(S16A 基板上に、各キーパッドに対し、560Ωの抵抗が搭載されています)

TS の番号は、S16A の基板上に記載されているものと合わせてください。また、名前を Button00 から Button0 に変更します。（名前は任意です。数字で始まる名前やスペースは許されていませ。）

・キーパッドと TS の対応

S16A（自己容量基板） キーパッド	TSxx 番号
0	TS01
1	TS03
2	TS05
3	TS06
4	TS08
5	TS00
6	TS02
7	TS04
8	TS07
9	TS09
A	TS10
B	TS13
C	TS14
D	TS15
E	TS12
F	TS11

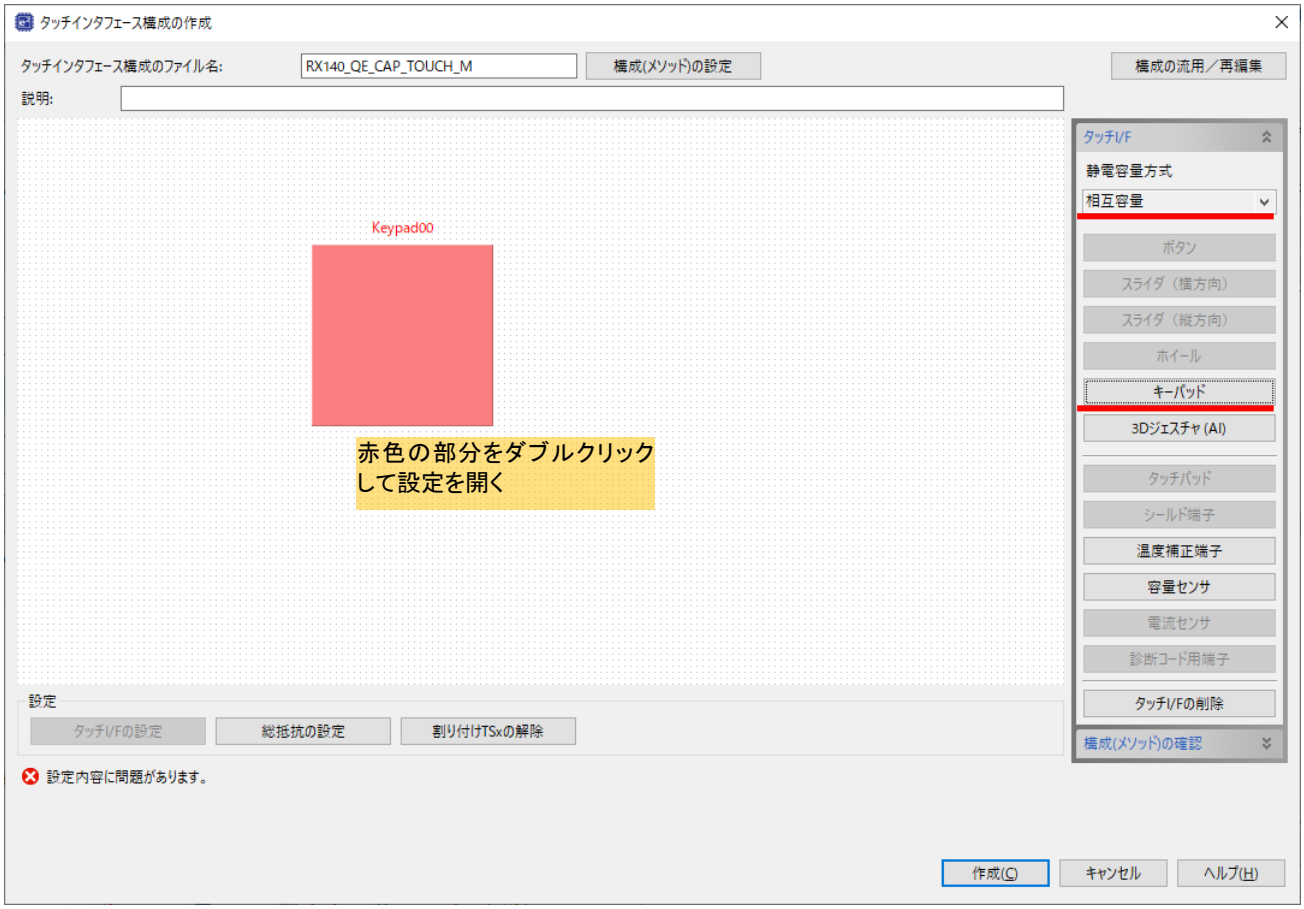
上記表に従い、キーパッドに TSxx の割り当てを行います。



作成を押す。

3.2.2. 相互容量基板(D55A)

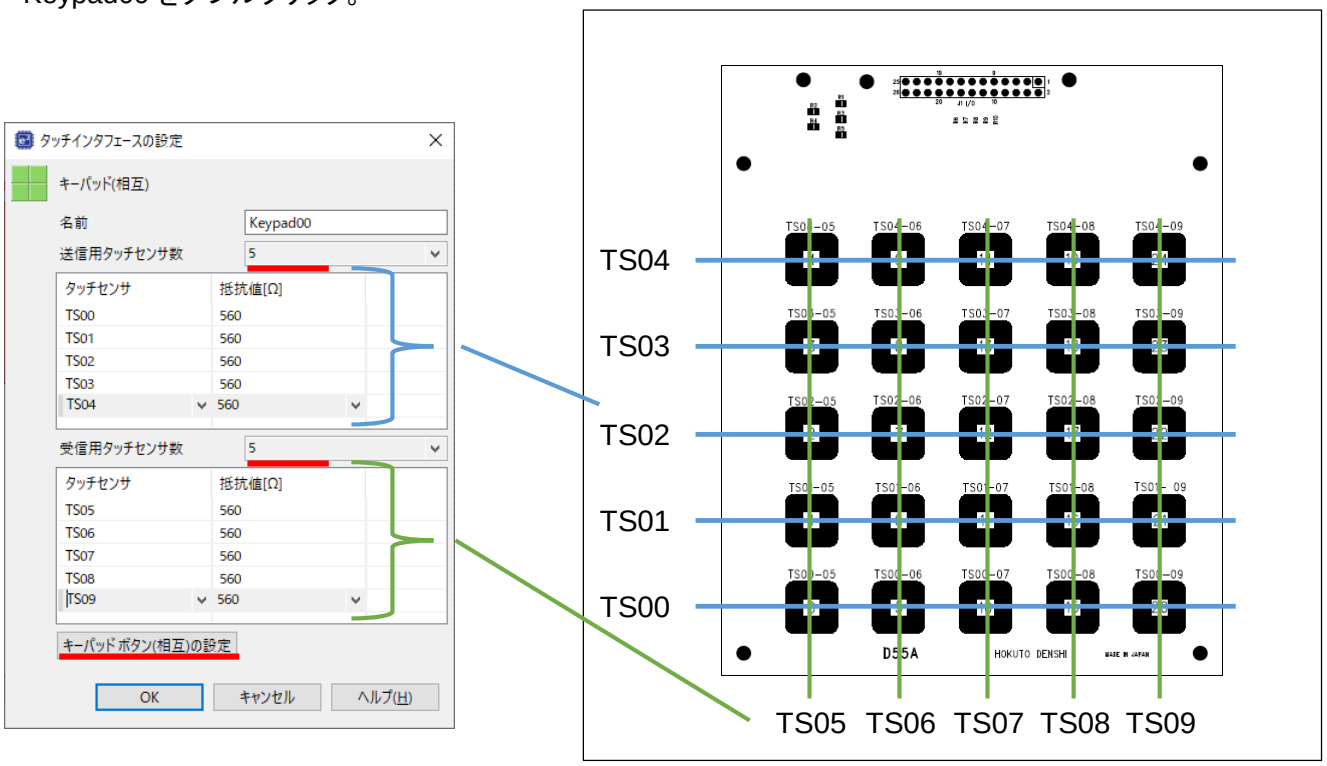
相互容量基板を用いる際のタッチインタフェースの作成に関しては、以下の様になります。



静電容量方式 ： 相互容量

「キーパッド」を選択して配置

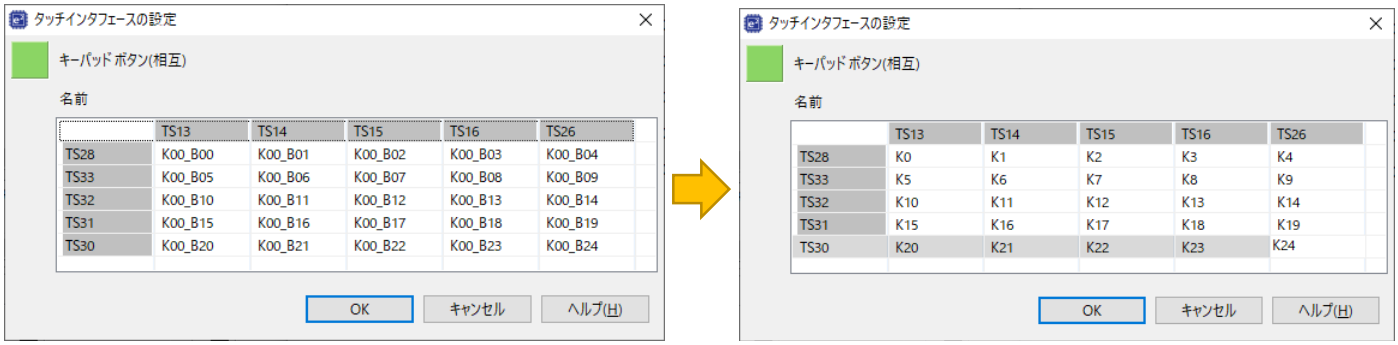
Keypad00 をダブルクリック。



送信用タッチセンサ数 5
受信用タッチセンサ数 5

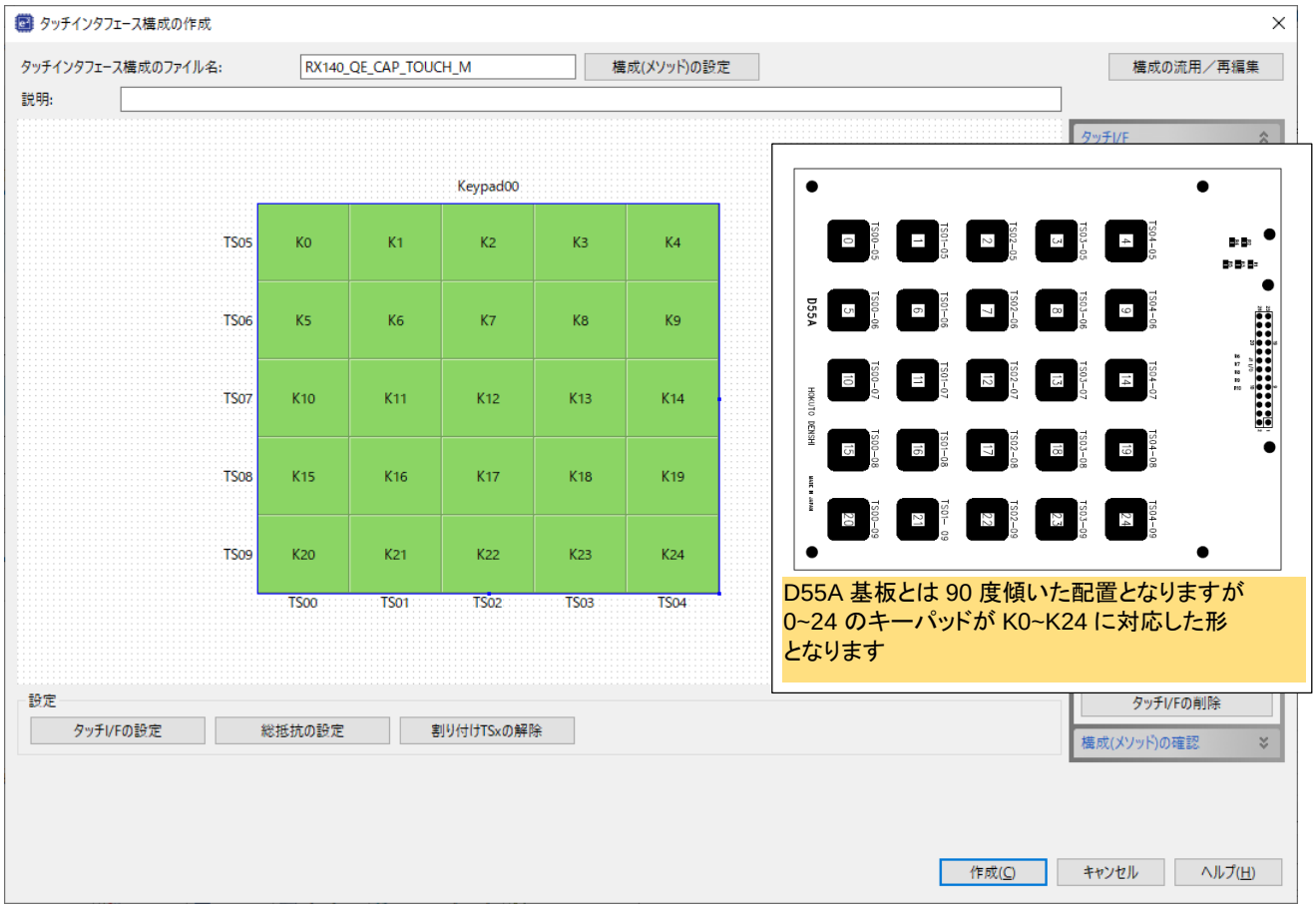
送信用のところに、TS00~TS04 を設定。受信用に、TS05~TS09 を設定する。抵抗値は、560Ω(デフォルトのまま)としてください。

「キーパッドボタン(相互)の設定」を押す。



キーの名前を、K0~K24(基板に記載されている 0~24 に対応)に変更。

※数字から始まる名前に設定できないので Kn としています

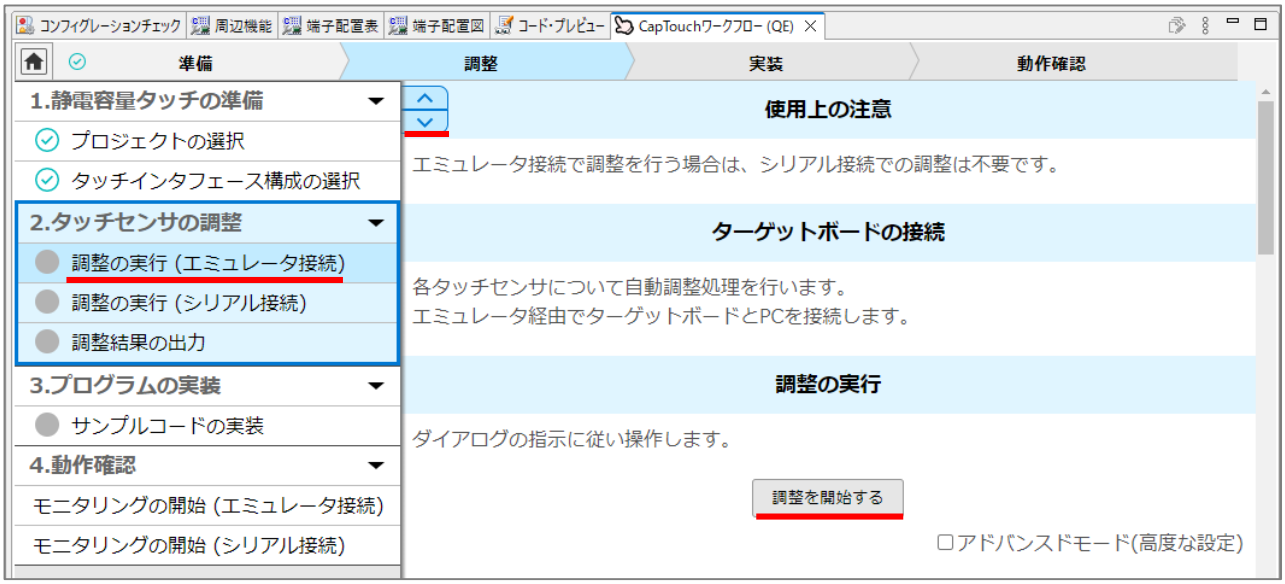


D55A 基板とは 90 度傾いた配置となりますが
0~24 のキーパッドが K0~K24 に対応した形
となります

3.3. タッチキーインタフェースの調整

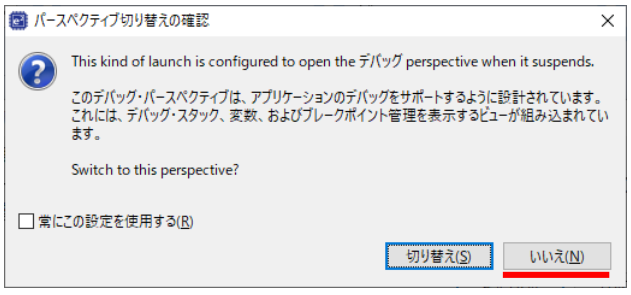
マイコンボードとタッチキーボード基板（S16A または D55A）を接続。

(3)調整

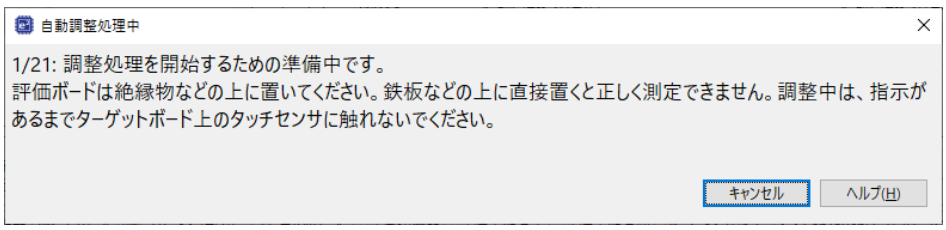


「調整を開始する」を押す。

プログラムのビルドとデバッガへのダウンロードが実行されます。

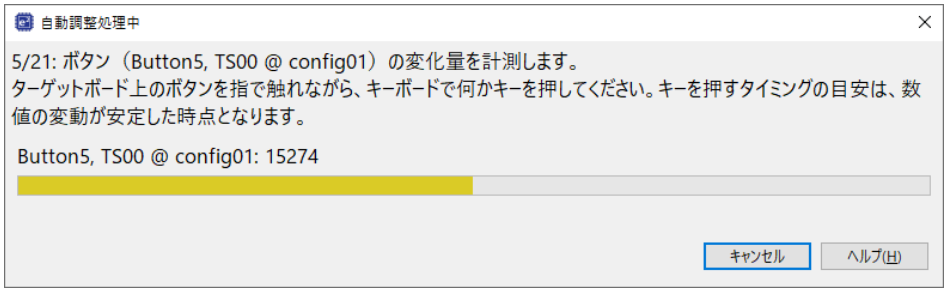


上記メッセージが出た場合は、（どちらでも構いませんが）ここでは「いいえ」を押します。



上記のメッセージが出力され、1/21, 2/21, ...の様に処理が進んでいきます。

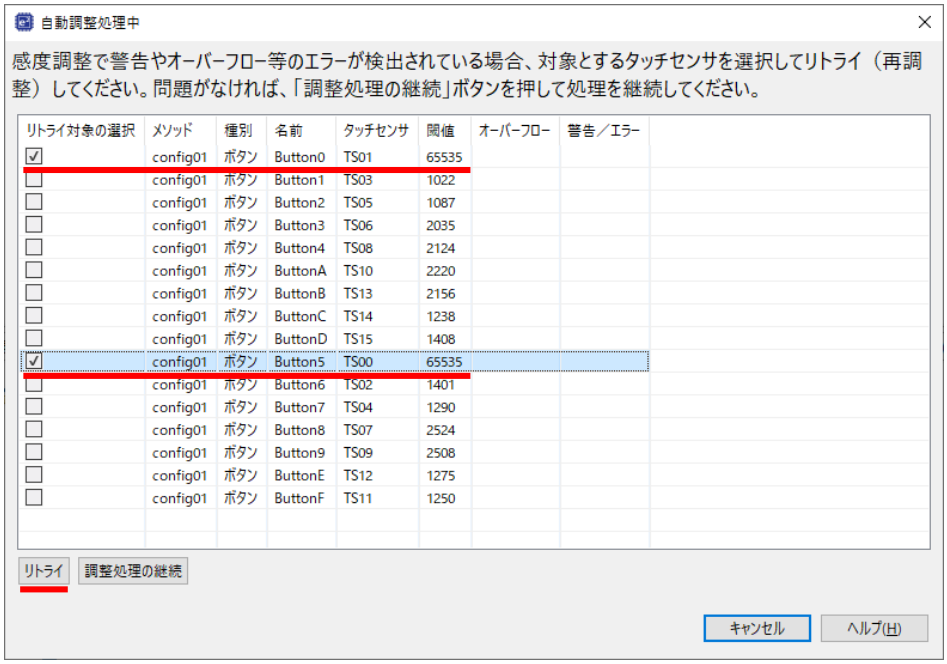
3.3.1. 自己容量基板(S16A)



キーパッドにタッチする事が求められますので、表示に従い S16A ボードの 5 のキーパッドにタッチして、(PC の)キーボードを叩いてください。

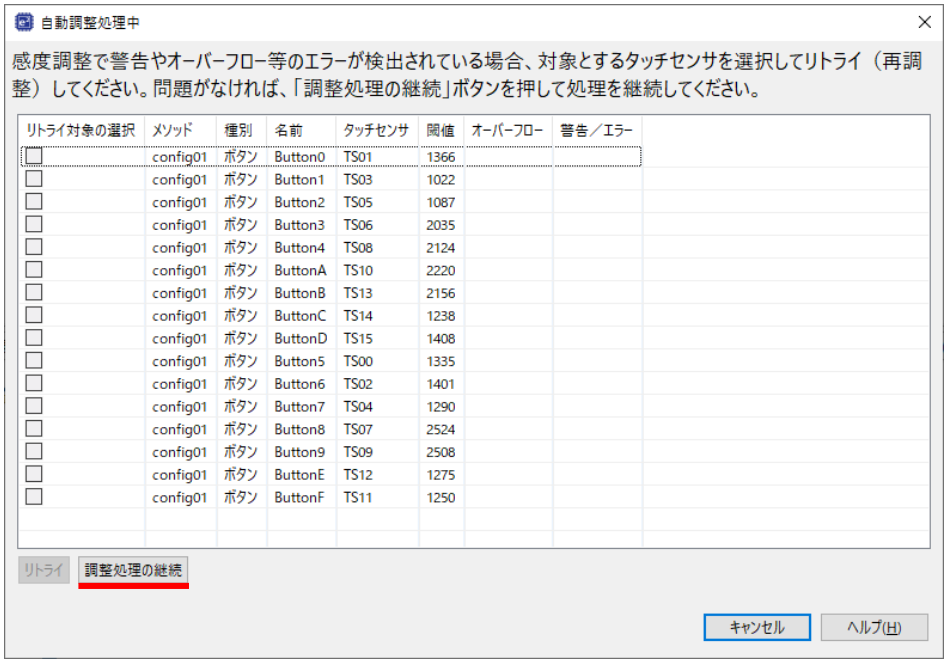
順次他のキーパッドに関しても、同様にタッチ＆キーボードを叩くという事を繰り返してください。

※キーパッドは TS 順の入力となりますので、5→0→…の様に 0 からの順番ではない事に注意願います

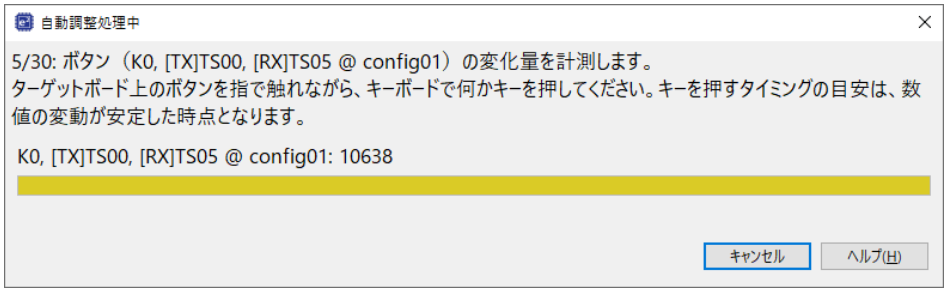


「閾値」の値が極端に小さいボタンや、65535（多分計算結果が負数となり表示上、65535 になっていると思われます）になっている場合タッチに失敗しています。「リトライ対象の選択」にチェックを入れ、「リトライ」を行ってください。

値が問題なさそうであれば、「調整処理の継続」を押して次に進んでください。

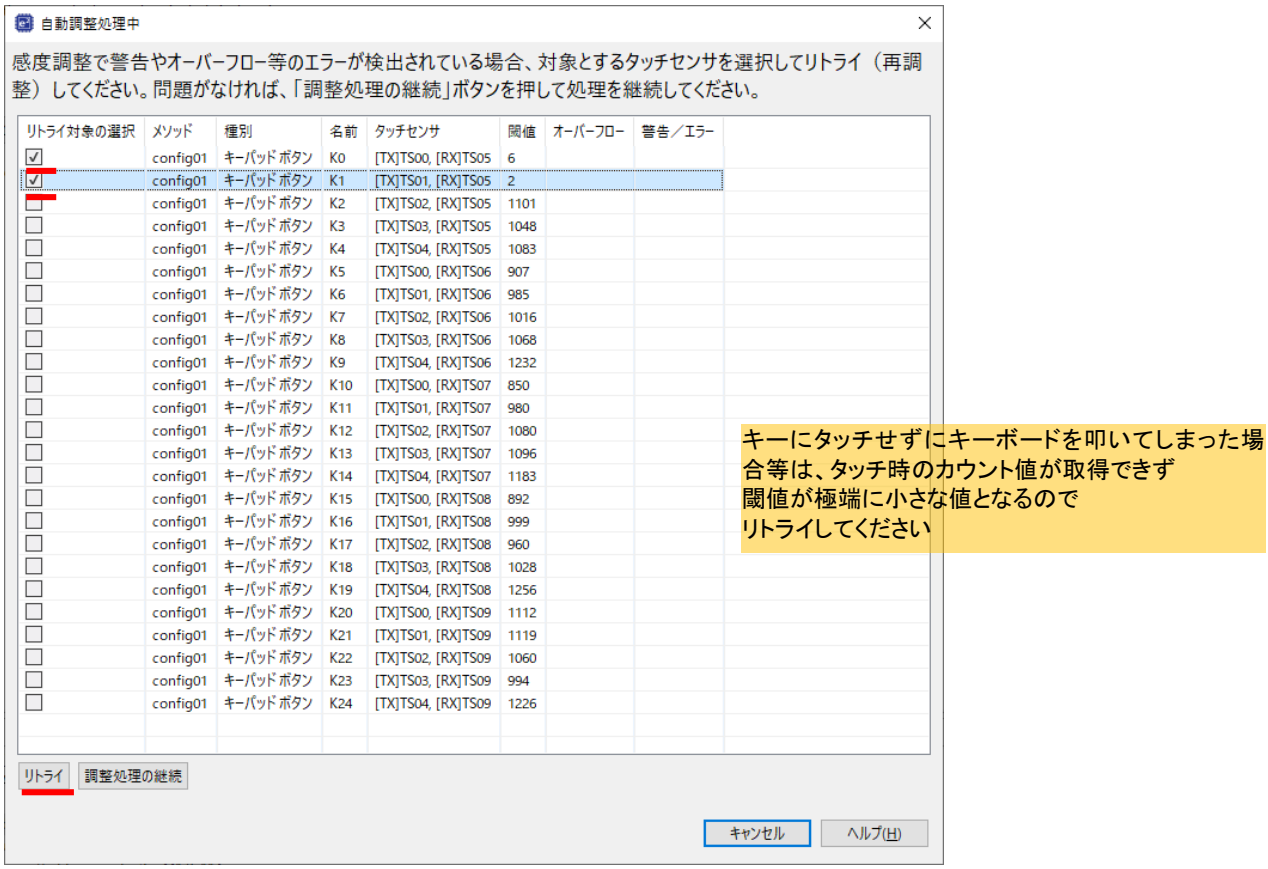

自動調整処理中
感度調整で警告やオーバーフロー等のエラーが検出されている場合、対象とするタッチセンサを選択してリトライ（再調整）してください。問題がなければ、「調整処理の継続」ボタンを押して処理を継続してください。
リトライ対象の選択
メソッド
種別
名前
タッチセンサ
閾値
オーバーフロー
警告／エラー
config01 ボタン Button0 TS01 1366
config01 ボタン Button1 TS03 1022
config01 ボタン Button2 TS05 1087
config01 ボタン Button3 TS06 2035
config01 ボタン Button4 TS08 2124
config01 ボタン ButtonA TS10 2220
config01 ボタン ButtonB TS13 2156
config01 ボタン ButtonC TS14 1238
config01 ボタン ButtonD TS15 1408
config01 ボタン Button5 TS00 1335
config01 ボタン Button6 TS02 1401
config01 ボタン Button7 TS04 1290
config01 ボタン Button8 TS07 2524
config01 ボタン Button9 TS09 2508
config01 ボタン ButtonE TS12 1275
config01 ボタン ButtonF TS11 1250
リトライ
調整処理の継続
キャンセル
ヘルプ(H)

3.3.2. 相互容量基板(D55A)



キーパッドにタッチする事が求められますので、表示に従い D55A ボードの 0 のキーパッドにタッチして、（PC の）キーボードを叩いてください。

順次、他のキーパッドに関しても、同様にタッチ＆キーボードを叩くという事を繰り返してください。

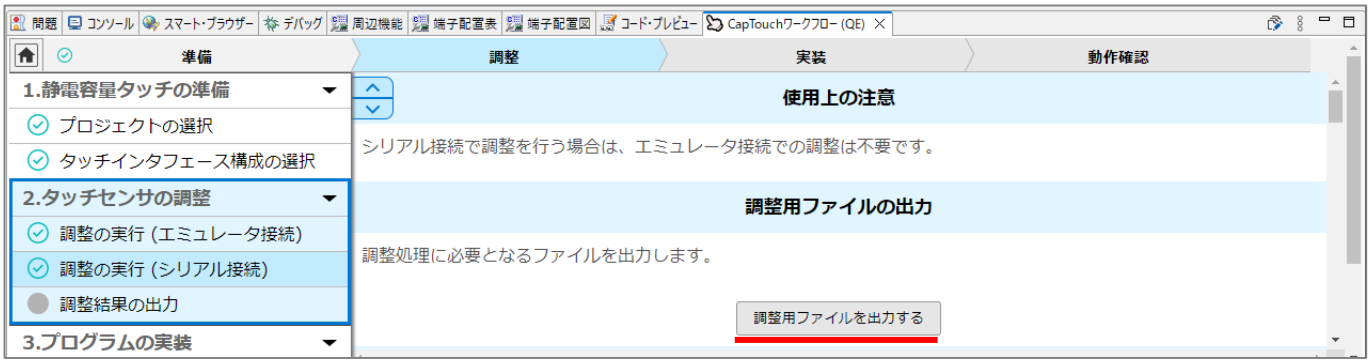


キーにタッチせずにキーボードを叩いてしまった場合等は、タッチ時のカウント値が取得できず閾値が極端に小さな値となるのでリトライしてください

閾値が他より極端に小さい、65535 になっている、オーバフローやエラーがない場合は「調整処理の継続」で次に進んでください。変な値がある場合は、リトライ対象の選択にチェックを入れリトライを行ってください。

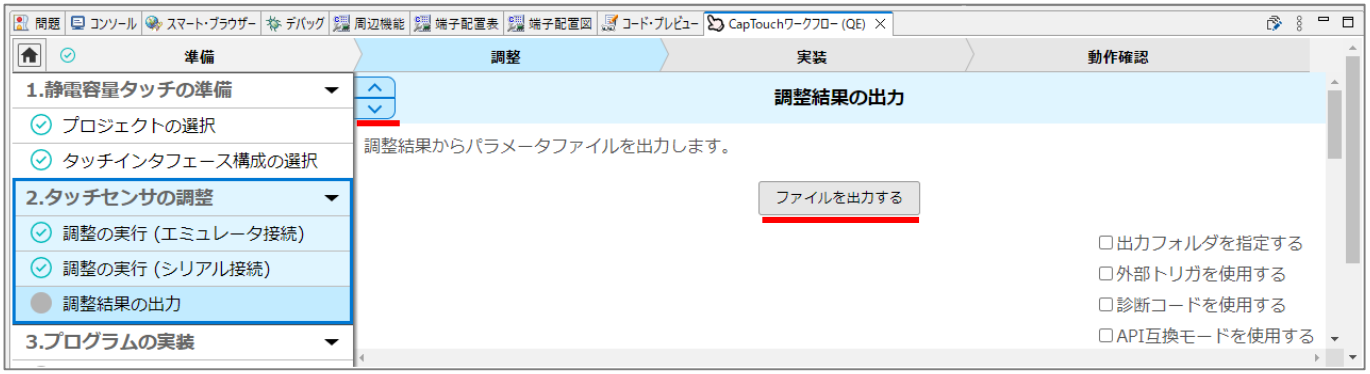
3.4. ソースコードの変更

(4)パラメータファイルの出力

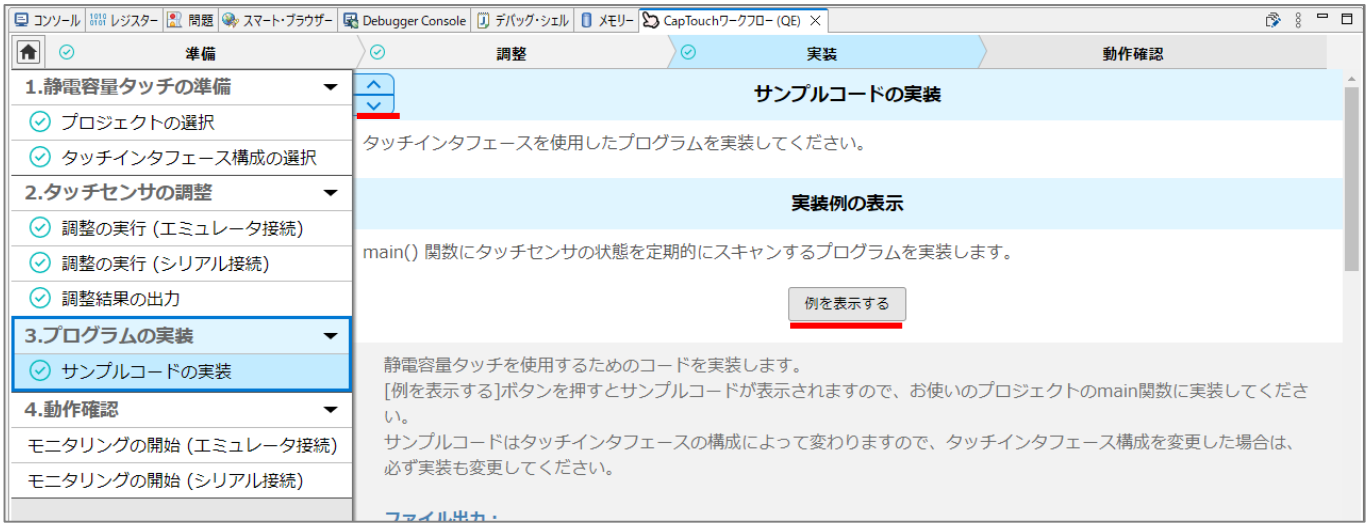


「調整用ファイルを出力する」。

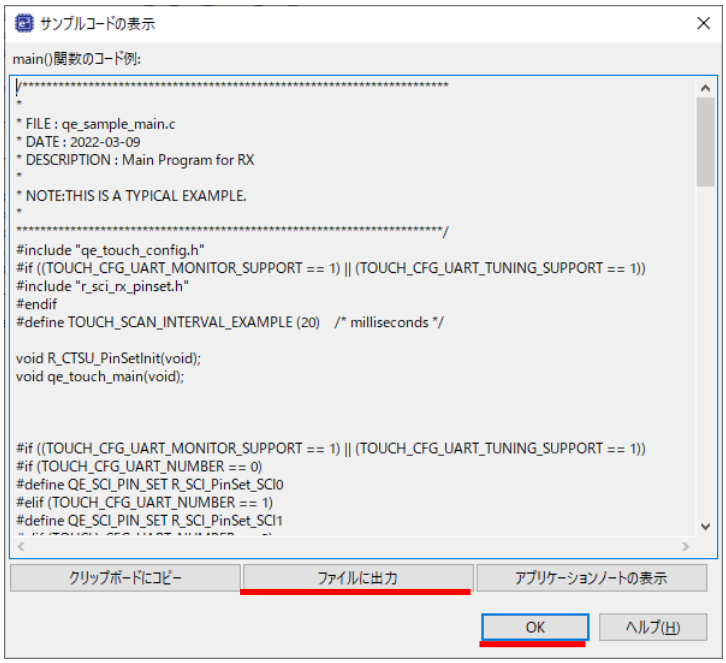
下矢印で次のページに進み、



「ファイルを出力する」。

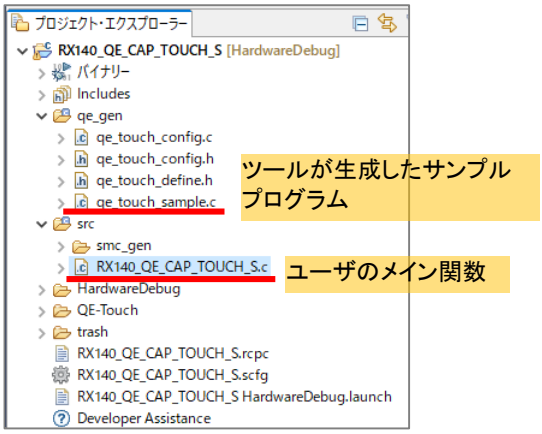


サンプルコードの実装、「例を表示する」。

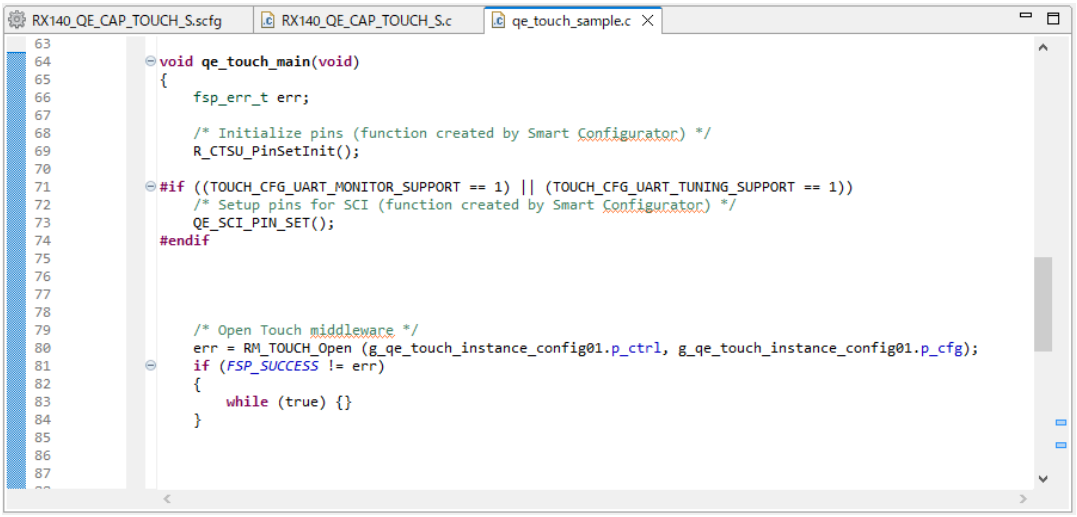


「ファイルに出力」「OK」

(5)ユーザプログラム内にタッチキーの判定プログラムを取り込む

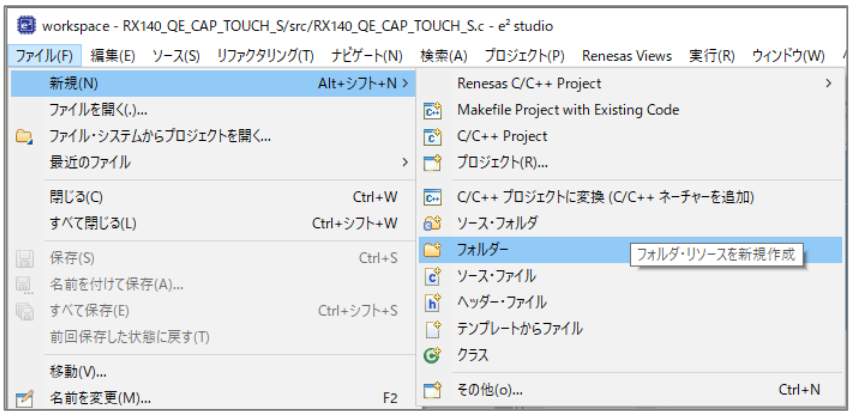


・qe_touch_sample.c

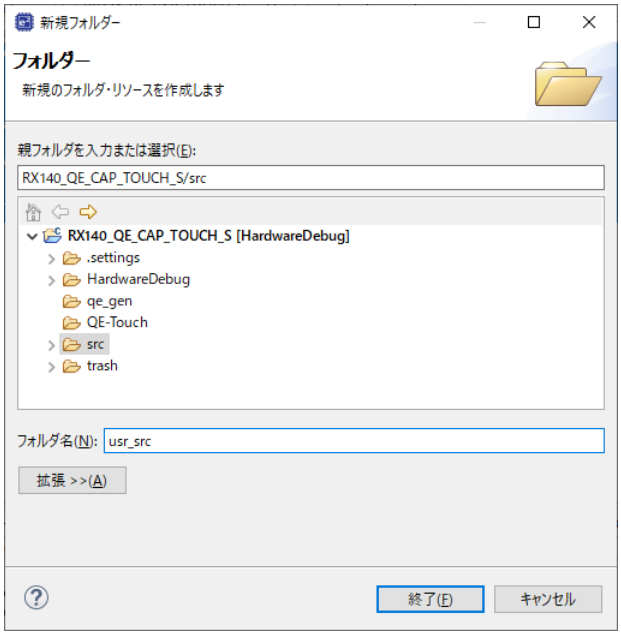


ツールが生成したサンプルプログラム（qe_touch_main()関数が定義されている）。

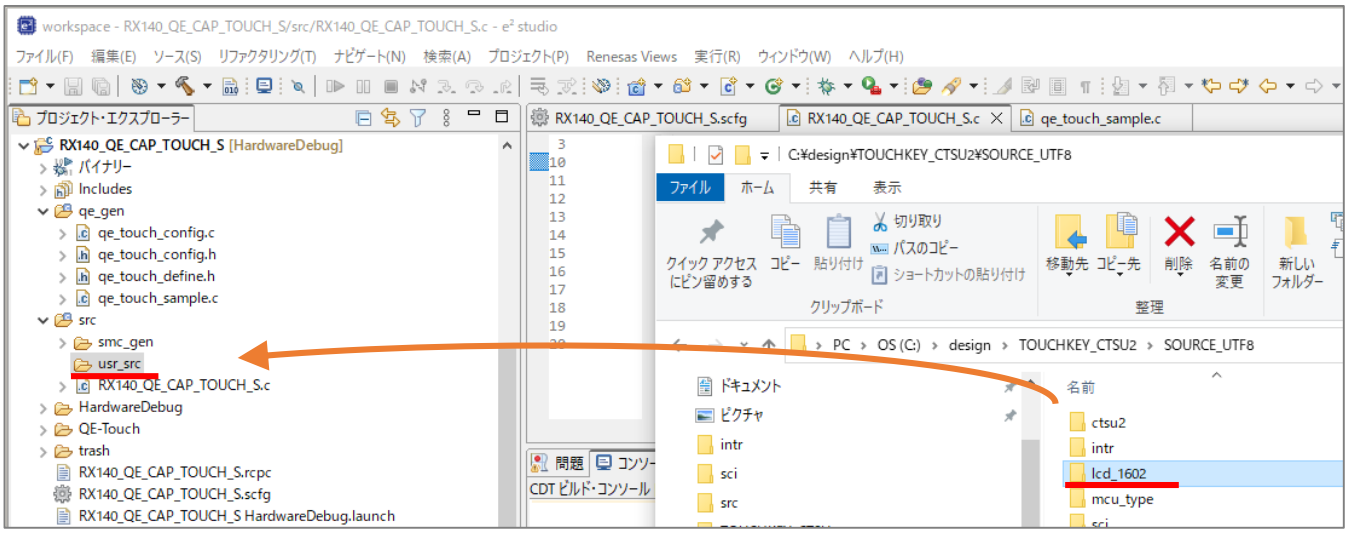
ここで、タッチしているキーを LCD に表示させるプログラムを追加する事とします。



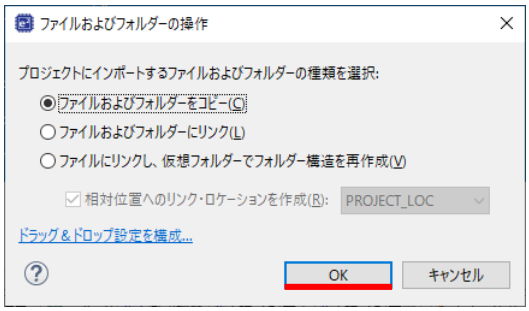
ファイル - 新規 - フォルダ



usr_src フォルダ（名称は任意）を、src 以下に追加します。

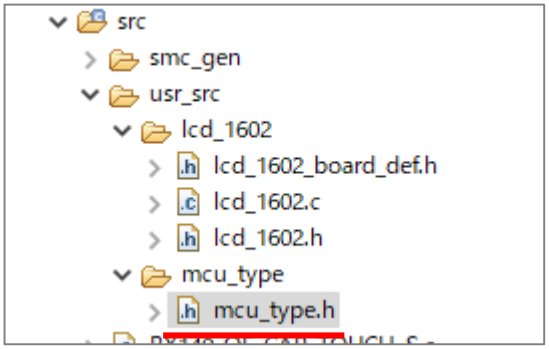


CD の SOURCE_UTF8 以下の lcd_1602 フォルダを usr_src 以下にドラッグ&ドロップします。



ファイル及びフォルダをコピーを選択して「OK」。

mcu_type フォルダも同様にコピー




```
/*-----------------------------------------------------------------------
マイコン種別（いずれか1つを選択）
-----------------------------------------------------------------------*/

#define MCU_TYPE_RX
//#define MCU_TYPE_RL78
//#define MCU_TYPE_RA
```

mcu_type.h 内で
#define MCU_TYPE_RX が
有効になる様に設定してください

3.4.1. 自己容量基板(S16A)

・src/*RX140_QE_CAP_TOUCH_S*.c（斜体部は、作成したプロジェクト名に応じて変わります）

```
#include "r_smc_entry.h"
#include "usr_src/mcu_type/mcu_type.h"
#include "usr_src/lcd_1602/lcd_1602.h"

extern void qe_touch_main(void);

void main(void);

void main(void)
{
    lcd_init();
    lcd_clear();
    lcd_hs1();
    //            1234567890123456
    lcd_write_str("RX140 SelfCap QE");
    R_BSP_SoftwareDelay(1000, BSP_DELAY_MILLISECS);

    qe_touch_main();
}
```

LCD の初期化
LCD 画面に初期化メッセージを表示
1 秒ウェイト（LCD の表示が見えるように）

黄色部分を追加。

・qe_gen/qe_sample.c

```
#include "qe_touch_config.h"
#if ((TOUCH_CFG_UART_MONITOR_SUPPORT == 1) || (TOUCH_CFG_UART_TUNING_SUPPORT == 1))
#include "r_sci_rx_pinset.h"
#endif
#include "../src/usr_src/mcu_type/mcu_type.h"
#include "../src/usr_src/lcd_1602/lcd_1602.h"

#define TOUCH_SCAN_INTERVAL_EXAMPLE (20)    /* milliseconds */
（中略）
```

```
void qe_touch_main(void)
{
    fsp_err_t err;

    int i;
    unsigned long x;

    //                        TS   0  1  2  3  4  5  6  7  8  9  10  11  12  13  14  15
    unsigned char key_table[16] = {5, 0, 6, 1, 7, 2, 3, 8, 4, 9, 10, 15, 14, 11, 12, 13};
    //測定chとキーパッド順のテーブル
    char result[16];

    lcd_hs1();
    //            1234567890123456
    lcd_write_str("0123456789ABCDEF");
```

・qe_gen/qe_sample.c（続き）

```
    /* 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. */
            lcd_hs2();
             x = 0x1UL;
             for(i=0; i<16; i++)
             {
                 if((x & button_status) == 0)
                 {
                     result[key_table[i]] =  '-';    //-(not touch)
                 }
                 else
                 {
                     result[key_table[i]] = 'o';     //o(touch)
                 }
                 x <<= 1UL;
             }
             //結果のLCD表示
             lcd_write_str(result);
        }

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

LCD 画面に、タッチしているキーの情報を表示
（タッチ：o，タッチしていない:-）
ー
表示例ー

```
0123456789ABCDEF
--o---o---
```

※複数のキー同時押しを認識

/* TODO: Add your own code here. */
の部分に、タッチしているキーを LCD に表示させるプログラムコードを追加。

3.4.2. 相互容量基板(D55A)

・src/*RX140_QE_CAP_TOUCH_M*.c（斜体部は、作成したプロジェクト名に応じて変わります）

```
#include "r_smc_entry.h"
#include "usr_src/mcu_type/mcu_type.h"
#include "usr_src/lcd_1602/lcd_1602.h"

extern void qe_touch_main(void);

void main(void);

void main(void)
{
    lcd_init();
    lcd_clear();
    lcd_hs1();
    //            1234567890123456
    lcd_write_str("RX140 MutuCap QE");
    R_BSP_SoftwareDelay(1000, BSP_DELAY_MILLISECS);

    qe_touch_main();
}
```

LCD の初期化
LCD 画面に初期化メッセージを表示
（1 秒間）

黄色部分を追加。

・qe_gen/qe_sample.c

```
#include "qe_touch_config.h"
#if ((TOUCH_CFG_UART_MONITOR_SUPPORT == 1) || (TOUCH_CFG_UART_TUNING_SUPPORT == 1))
#include "r_sci_rx_pinset.h"
#endif
#include "../src/usr_src/mcu_type/mcu_type.h"
#include "../src/usr_src/lcd_1602/lcd_1602.h"

#define TOUCH_SCAN_INTERVAL_EXAMPLE (20)    /* milliseconds */
 （中略）
```

```
void qe_touch_main(void)
{
    fsp_err_t err;

    int i;
    unsigned long x;
    unsigned char key_table[25] = {0, 1, 2, 3, 4,
    5, 6, 7, 8, 9,
    10, 11, 12, 13, 14,
    15, 16, 17, 18, 19,
    20, 21, 22, 23, 24};//測定chとキーパッド順のテーブル
    int result;

    lcd_hs1();
    //            1234567890123456
    lcd_write_str("Touch key pad:  ");

    /* Initialize pins (function created by Smart Configurator) */
    R_CTSU_PinSetInit();
 （中略）
```

測定 ch の若い順ーキーパッドの対応
のテーブルを定義

・qe_gen/qe_sample.c（続き）

```
    /* 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. */
            lcd_hs2();
            lcd_write_str("0x");
            lcd_write_uint32_hex((unsigned long)button_status);
            lcd_write_str(" > ");

            x = 0x1UL;
            result = -1;
            for(i=0; i<25; i++)
            {
                if(button_status & x)
                {
                    result = i;
                    break;
                }
                x <<= 1UL;
            }

            if(result == -1)
            {
                lcd_write_str("-  ");
            }
            else
            {
                lcd_write_uint8(key_table[i]);
            }
        }
（後略）
```

LCD 画面に、タッチしているキーの情報を表示

表示例ー

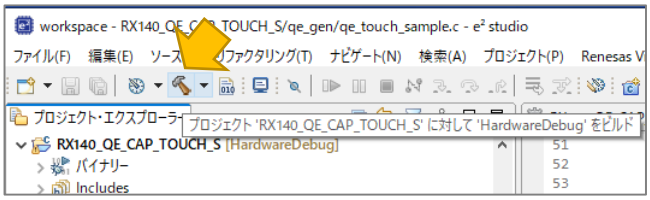
```
Touch Key pad:
0x00000010 > 4
```

button status　　タッチキーパッド番号

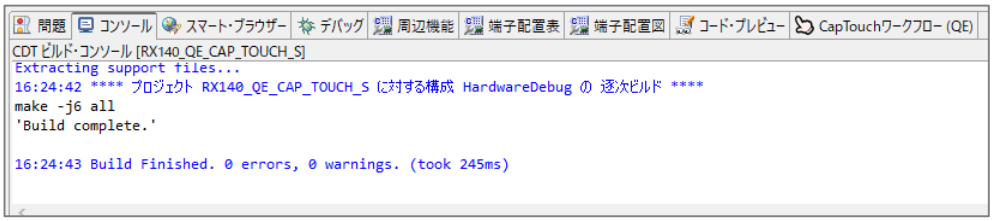
button status は、64bit 変数で、下位 32bit を 16 進数で表示

/* TODO: Add your own code here. */
の部分に、タッチしているキーを LCD に表示させるプログラムコードを追加。

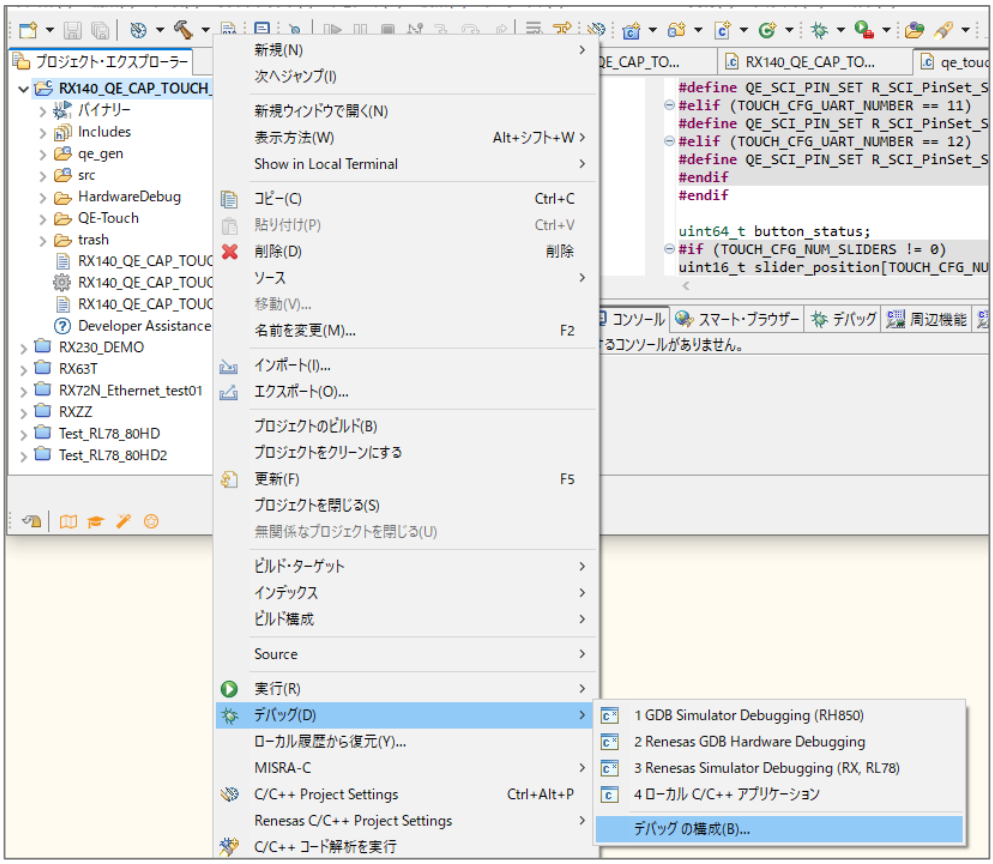
3.5. ビルド・実行



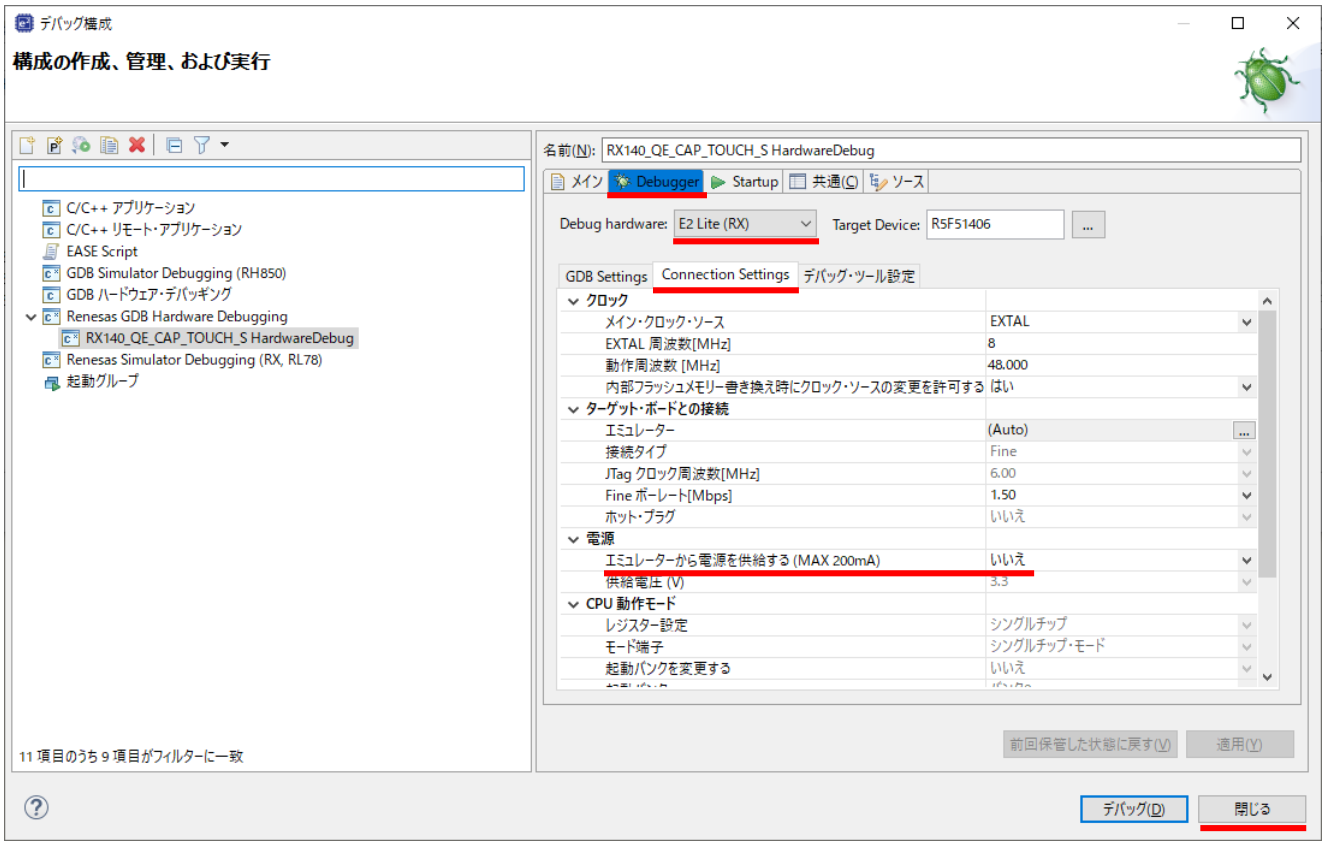
「ビルド」を実行。



エラーや意図しないワーニングがなければ問題ありません。



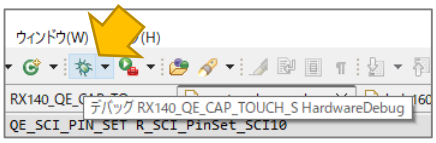
プロジェクト名を右クリック。デバッグ－デバッグの構成。



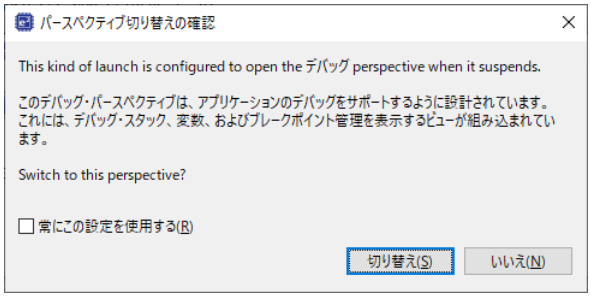
Debugger タブを選択し、
Debug hardware:「使用しているエミュレータを選択」
※E1(RX)の様に、RX と書いているものを選択してください

Connection タブ
エミュレータから電源を供給する:「いいえ」
※外部から電源供給を行わず「はい」を選択することも可能です。
プロジェクト作成後、5V を印加してキーのチューニングを行う。その後、3.3V を印加してプログラムを実行した場合、パラメータが最適値ではないという状況になりますので、チューニングとプログラム実行時の電圧は合わせる様にしてください。E2 Lite を使用した場合は、エミュレータから印加可能な電圧は 3.3V のみとなりますのでご注意ください。

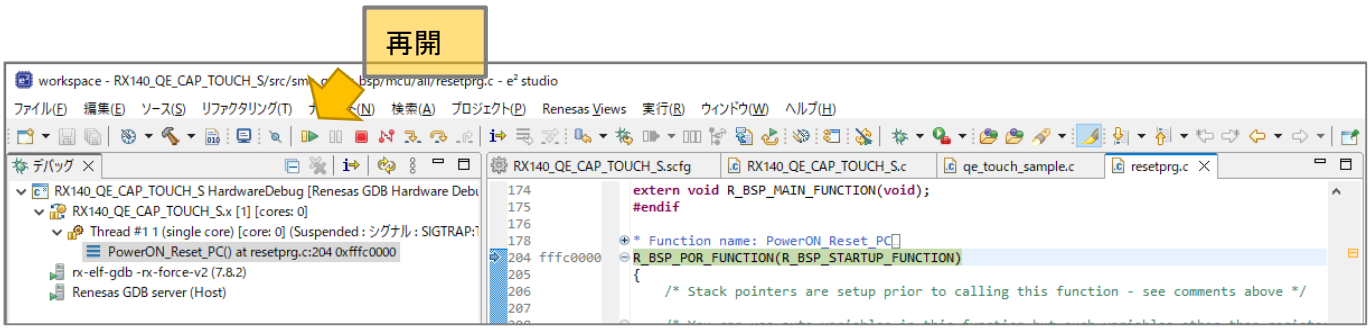
設定終了後「閉じる」。



虫のアイコンをクリックして「デバッグ」を実行。



上記メッセージが出た場合、「切り替え」。



ブレークポイントで止まりますので、「再開」ボタン(もしくは F8)で進めてください(2 回押す)。

自己容量(S16A)基板を使用している場合は、LCD に

```
0123456789ABCDEF
--o----o--------
```

-:非タッチ

o:タッチ

の表示が出るはずです。(2, 7 にタッチした場合)

相互容量(D55A)基板を使用している場合は、LCD に

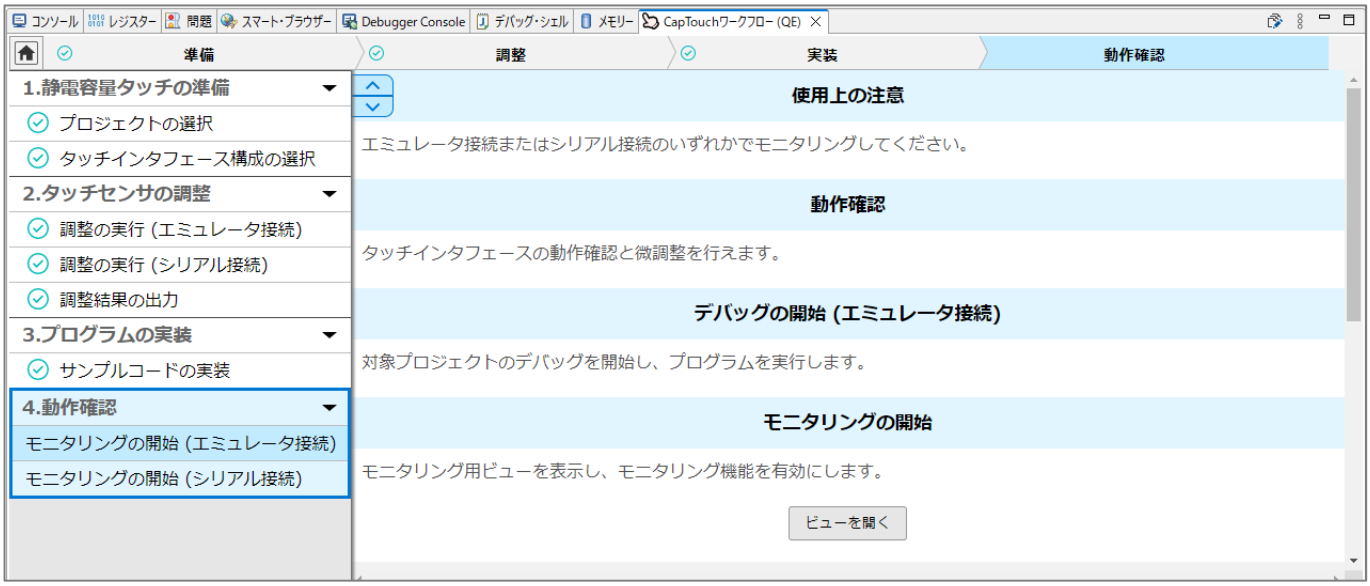
```
Touch key pad:
0x00000002 > 1
```

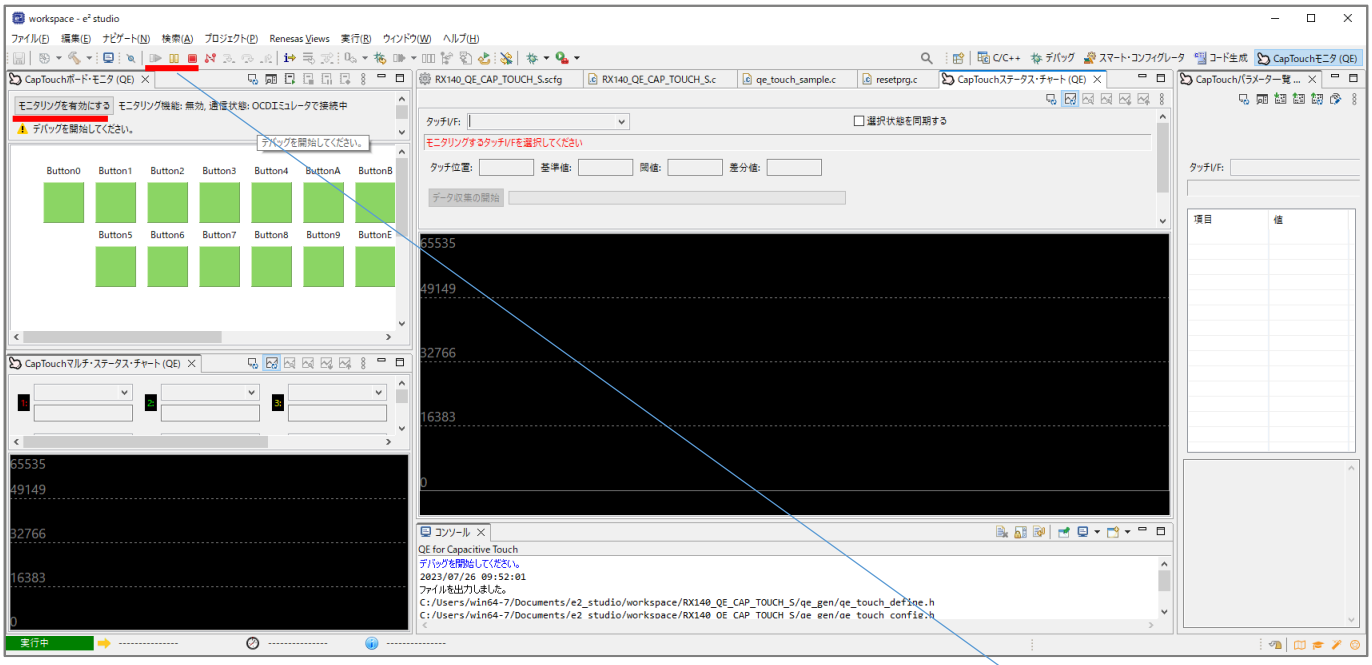
の表示が出るはずです。(1 にタッチした場合)

※3.4 で LCD を動かすプログラムを追加した場合。

※LCD のプログラムを追加していない場合でも、モニタリングツールでモニタは可能です。

3.6. モニタリング（エミュレータ接続）





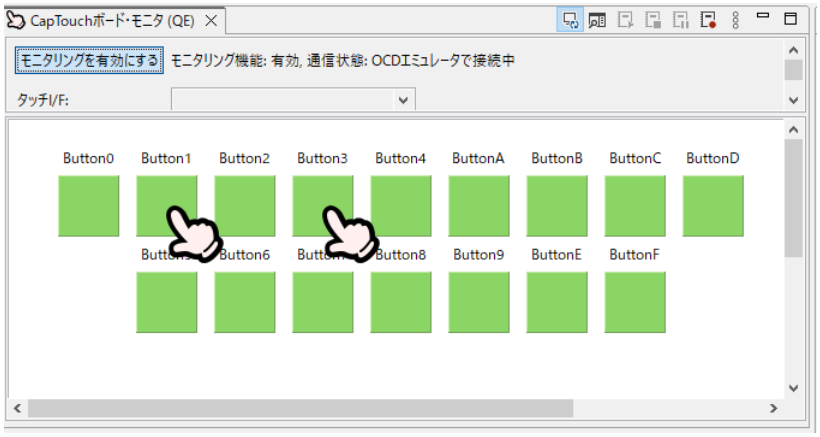
※3.5 章の、エミュレータを使用してのプログラムのダウンロードと実行を行い、プログラムが実行されている状態である事が必要です。

※緑の三角のアイコンがグレーアウトしていない場合

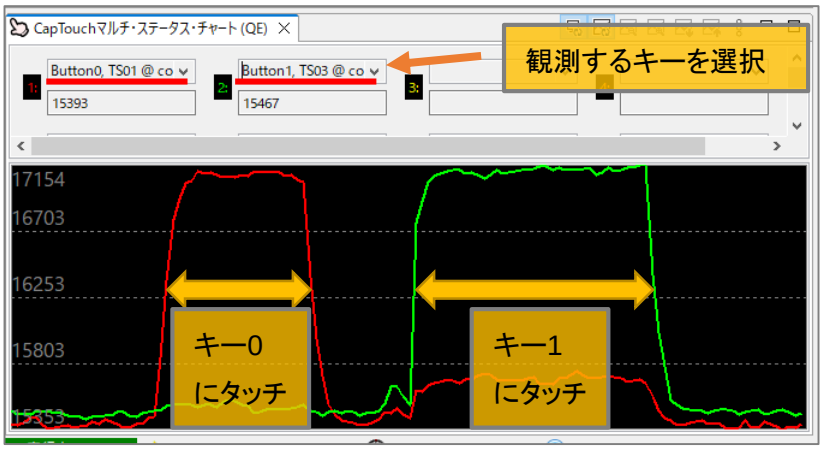
プログラムがブレークポイントで停止した状態なので、緑の三角のアイコンをクリックしてプログラムを実行状態にしてください。

「モニタリングを有効にする」を押す。

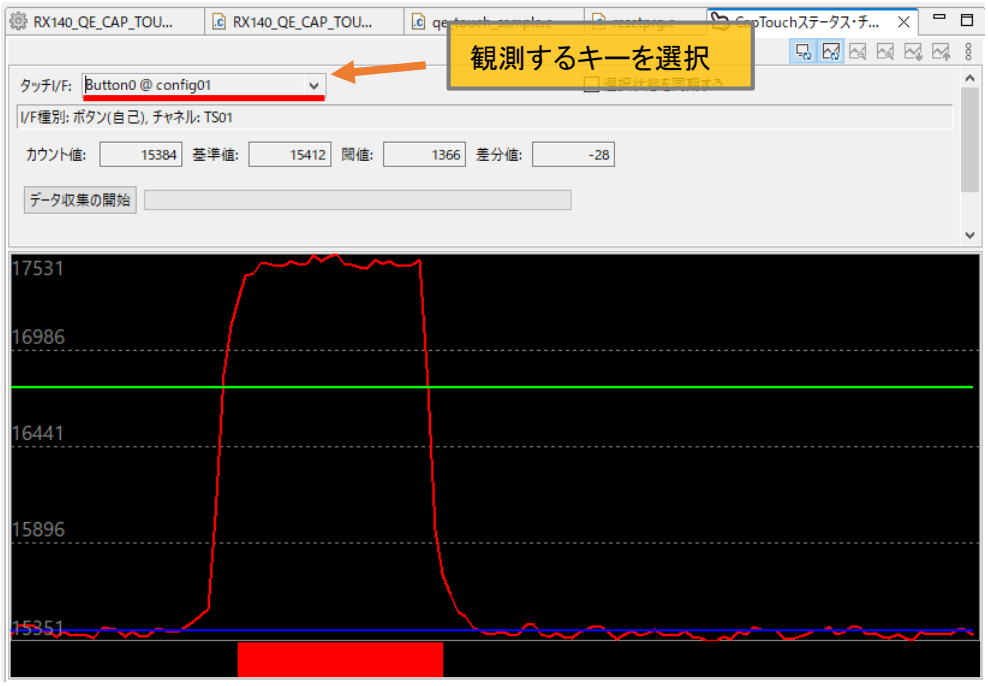
キーにタッチすると、



タッチしたキーに指のアイコンが表示されるはずです。

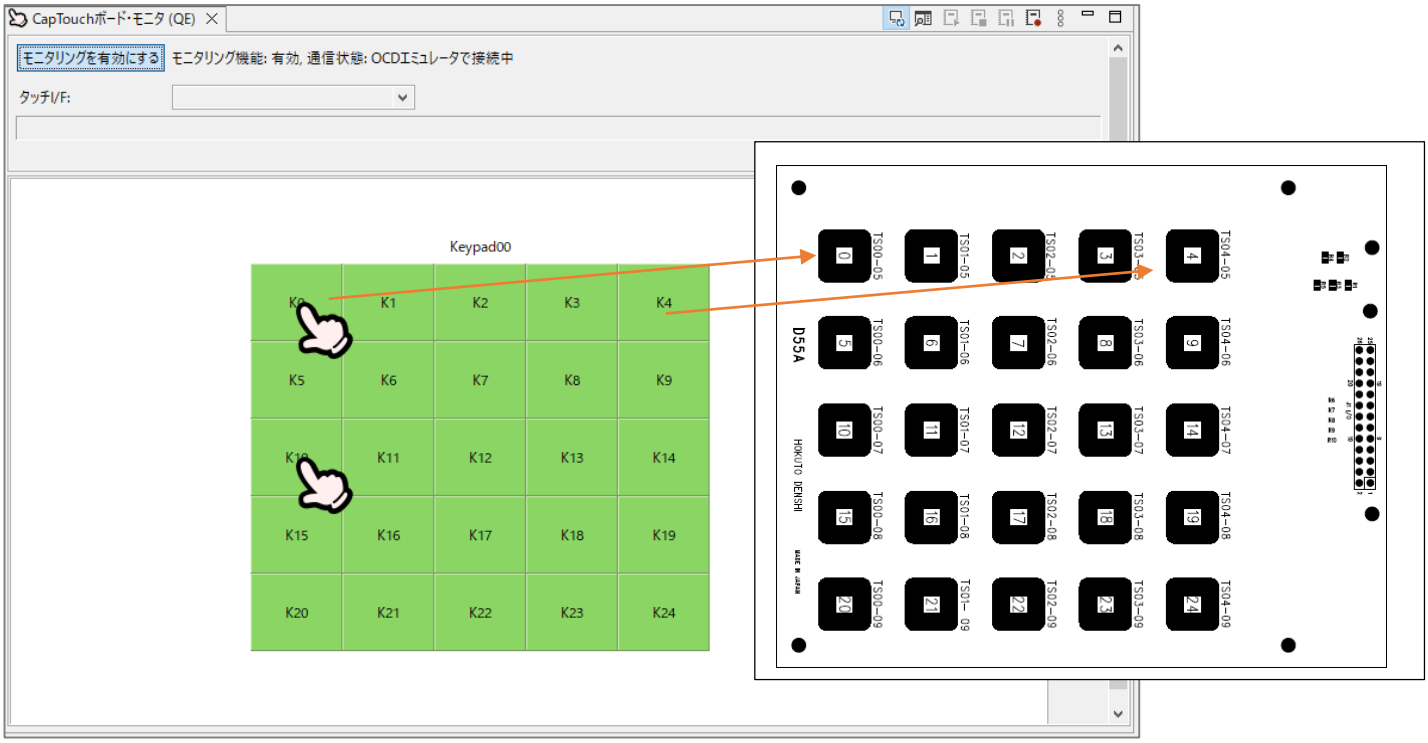


マルチ・ステータス・チャートでは、プルダウンからモニタしたいキーを選択すると、横軸－時間、縦軸－測定値のグラフを見ることが出来ます。

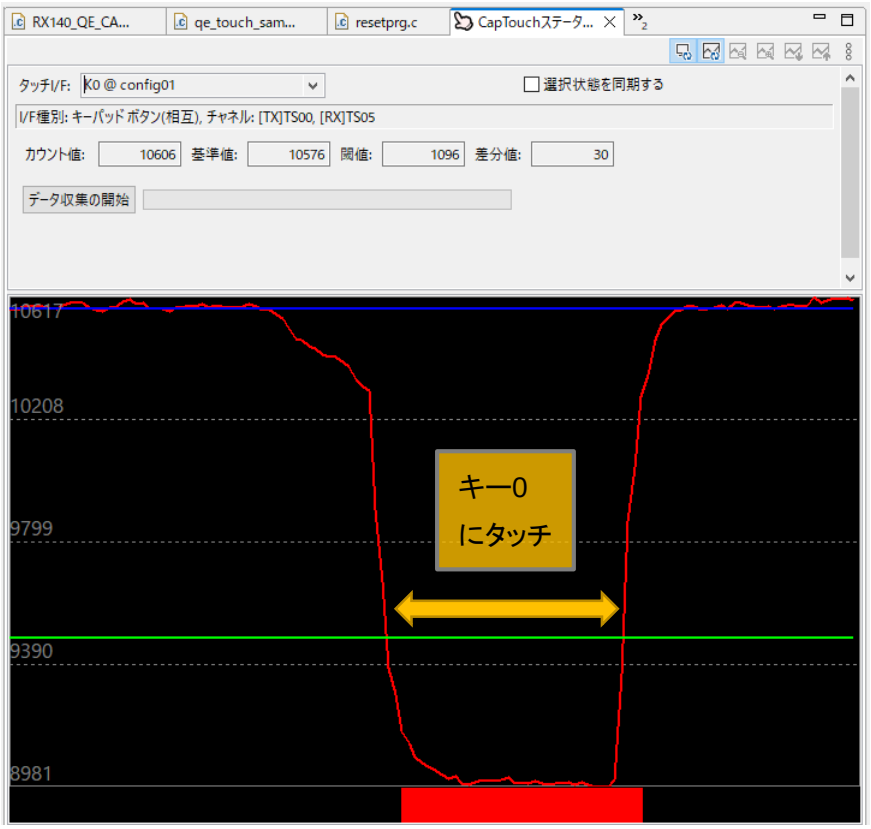


ステータス･チャートでは、閾値を含めた詳しい情報の表示が可能です（緑のラインが閾値で、カウント値がこのラインより上の場合タッチしていると見なされます。

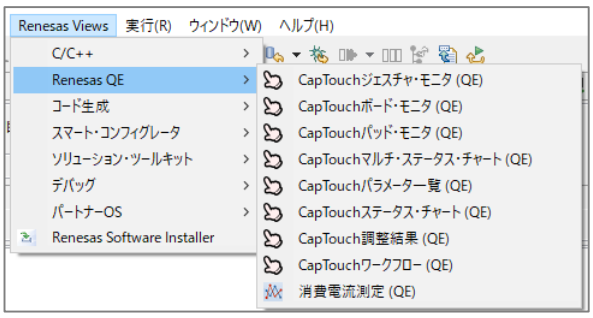
－相互容量基板(D55A)の場合－



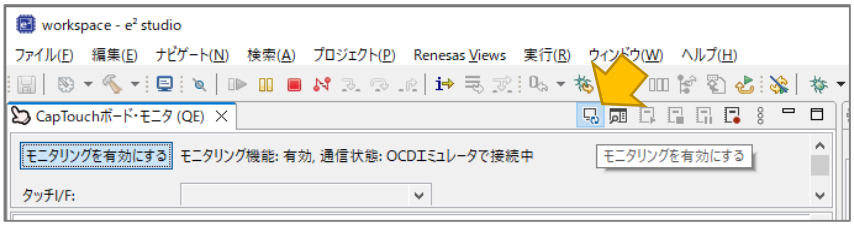
相互容量の場合は、D55A 基板と表示されるキーの向きは 90° 回転した方向になりますが、タッチしているキーの番号は変わりません（なお、本ツールでは複数タッチにも応答するはずです）。



相互容量の場合は、タッチ時にカウント値が減少する動作となります。

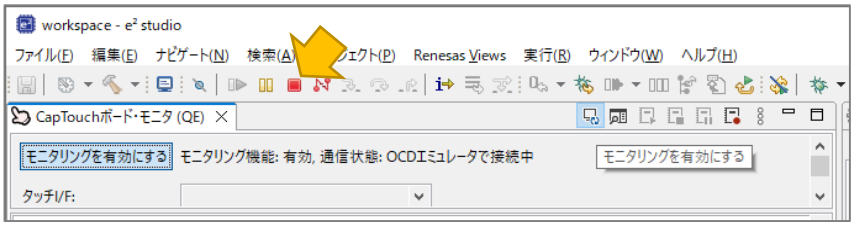


タッチ状態を示す指のアイコンや、カウント値のグラフ以外でも、CapTouch には色々な情報をモニタできるツールがあるので、色々試してみてください。



モニタリングを停止する場合は矢印で示しているアイコンを押してください。

エミュレータ接続を切断する場合は、



赤の□のボタンを押してください。

4. まとめ

QE CapTouch を使用すると、タッチキーインタフェース（キーパッドに合わせたレイアウト）の作成を GUI で行う事ができ、ソースコードの生成をツールが行ってくれますので、タッチキー(CTSU2)のレジスタ構成や、仕組みを理解していなくても、タッチキーを使用したアプリケーションの構築が可能です。

5. 付録

取扱説明書改定記録

バージョン	発行日	ページ	改定内容
REV.1.0.0.0	2023.7.27	—	初版発行
REV.1.0.1.0	2023.9.26	40	プログラムコードのバグ修正

お問合せ窓口

最新情報については弊社ホームページをご活用ください。

ご不明点は弊社サポート窓口までお問合せください。

株式会社 北斗電子

〒060-0042 札幌市中央区大通西 16 丁目 3 番地 7

TEL 011-640-8800　FAX 011-640-8801

e-mail：support@hokutodenshi.co.jp (サポート用)、order@hokutodenshi.co.jp (ご注文用)

URL:https://www.hokutodenshi.co.jp

ルネサス エレクトロニクス RX グループマイコン搭載
HSB シリーズマイコンボード向けキット

RX140 タッチキー評価キット
[QE CapTouch 編] マニュアル

株式会社 北斗電子
 2023 年 9 月 26 日改訂 REV.1.0.1.0 (230926)