RA2L1 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.1.0.0.0

ー目　次ー

注意事項 **1**

概要 **2**

1.　本書で説明する範囲 **3**

1.1.　CD フォルダ構成 3

2.　初期設定 **4**

2.1.　QE ツールのインストール 4

2.2.　プロジェクトの新規作成 11

2.3.　FSP の設定 15

2.3.1.　Clocks タブ 15

2.3.2.　Stacks タブ 16

2.3.3.　Pins タブ 20

2.4.　エミュレータ（デバッガ）の接続設定 23

3.　QE CapTouch の使用 **25**

3.1.　CapTouch の起動 25

3.2.　タッチキーインタフェースの作成 26

3.2.1.　自己容量基板(S16A) 26

3.2.2.　相互容量基板(D55A) 29

3.3.　タッチキーインタフェースの調整 31

3.3.1.　自己容量基板(S16A) 32

3.3.2.　相互容量基板(D55A) 33

3.4.　ソースコードの変更 34

3.4.1.　自己容量基板(S16A) 36

3.4.2.　相互容量基板(D55A) 38

3.5.　ビルド・実行 40

3.6.　モニタリング 42

4.　まとめ **46**

5.　付録 **47**

取扱説明書改定記録 47

お問合せ窓口 47

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、
「RA2L1 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。

RA2L1 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/	RA/	RA2L1_QE_CAP_TOUCH_S.zip	RA2L1 向け e2studio プロジェクト （自己容量, S16A タッチパッド向け）
	RA/	RA2L1_QE_CAP_TOUCH_M.zip	RA2L1 向け e2studio プロジェクト （相互容量, D55A タッチパッド向け）

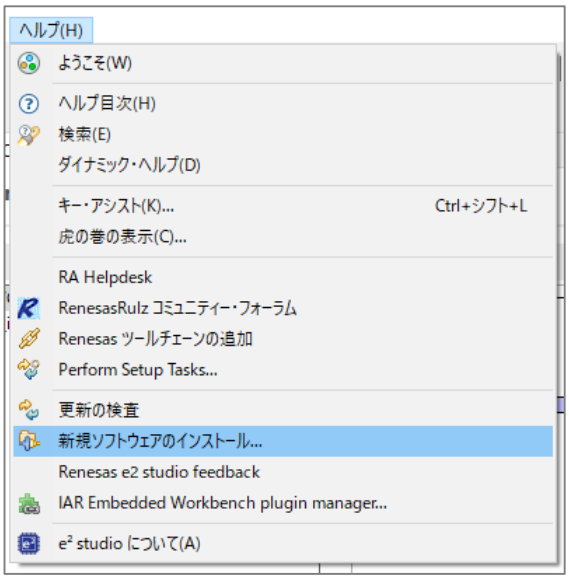
e2studio のプロジェクトのアーカイブとなっていますので、ワークスペースにインポートしてください。

2. 初期設定

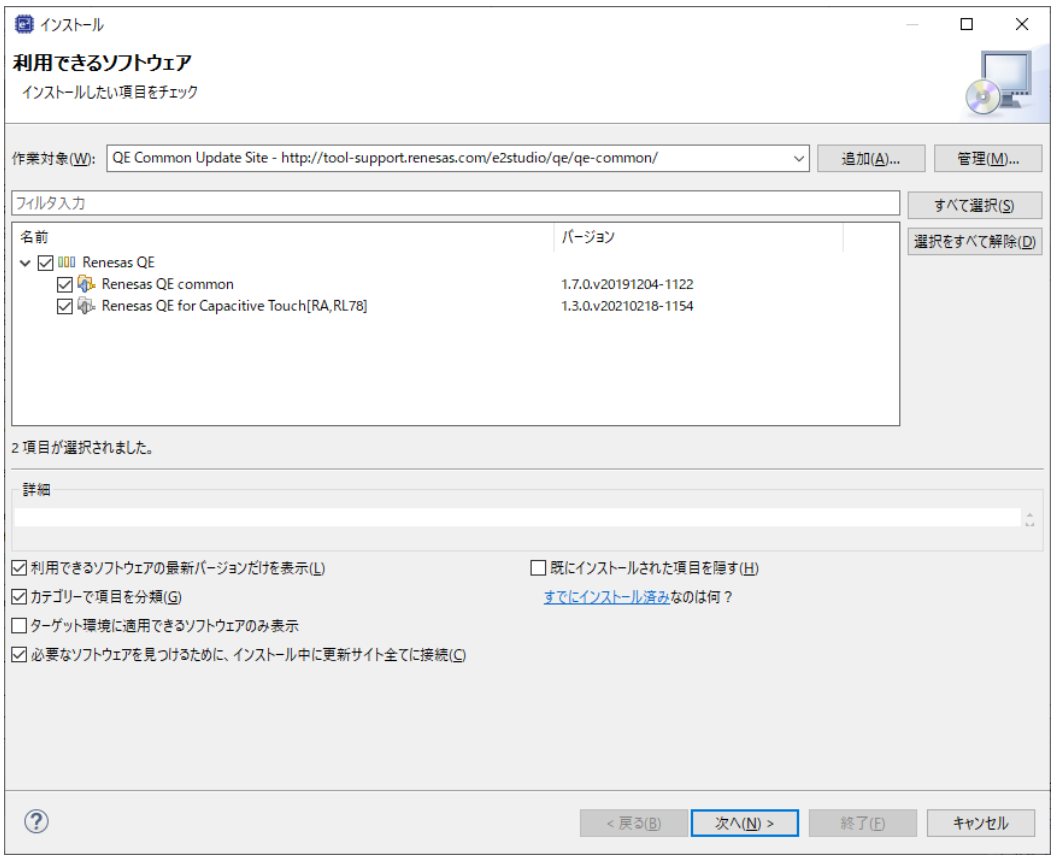
2.1. QE ツールのインストール

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。

※[重要] Renesas QE Cap-touch を使って PC からタッチキー動作を見る場合は、エミュレータ（ルネサス E2Lite もしくは E2）が必要です （エミュレータがない場合は、CD に格納されている作成済みのプロジェクトを開き、タッチ動作を確認することができます。新規にプロジェクトを作成する場合はデバッガが必須となります。）



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、

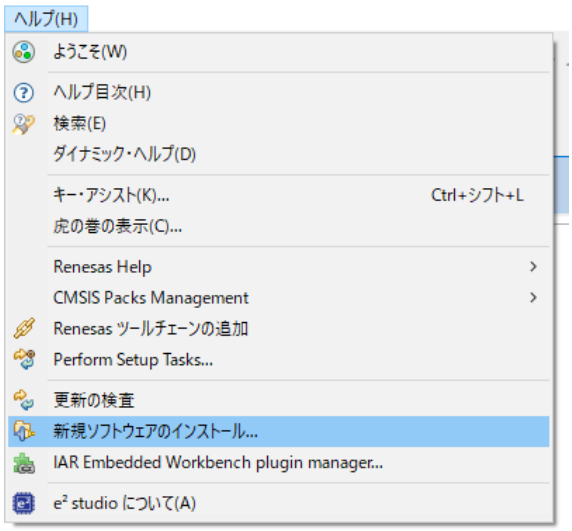
Renesas QE common

Resnesas QE for Capacitive Touch

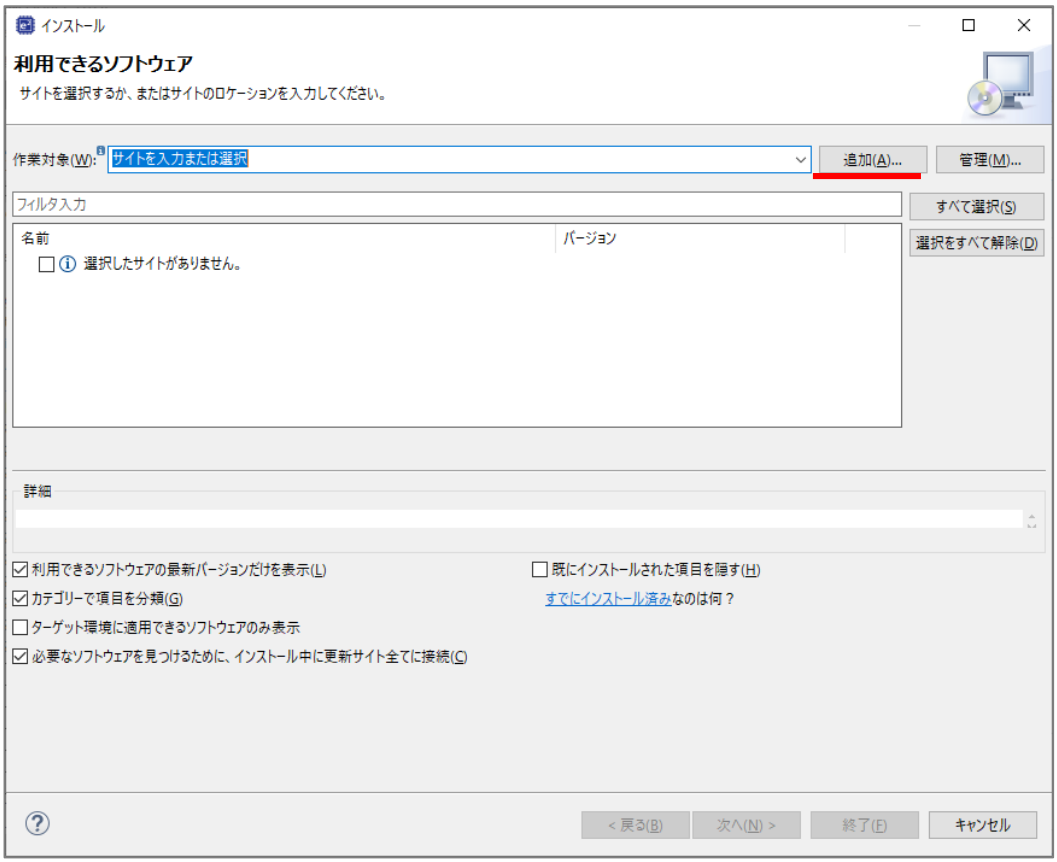
をインストールしてください。

・ローカルインストールの場合

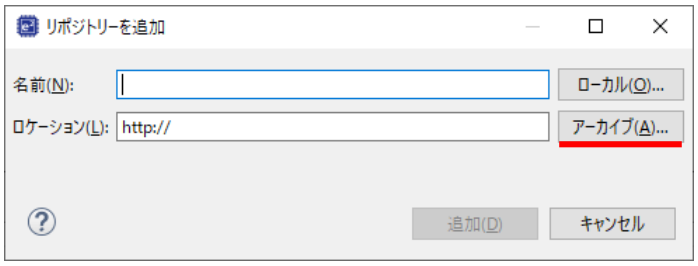
インターネットに直接接続していない場合は、ルネサスエレクトロニクスのサイトから zip ファイルをダウンロードして、ローカルファイルからインストールしてください。



ヘルプ－新規ソフトウェアのインストール

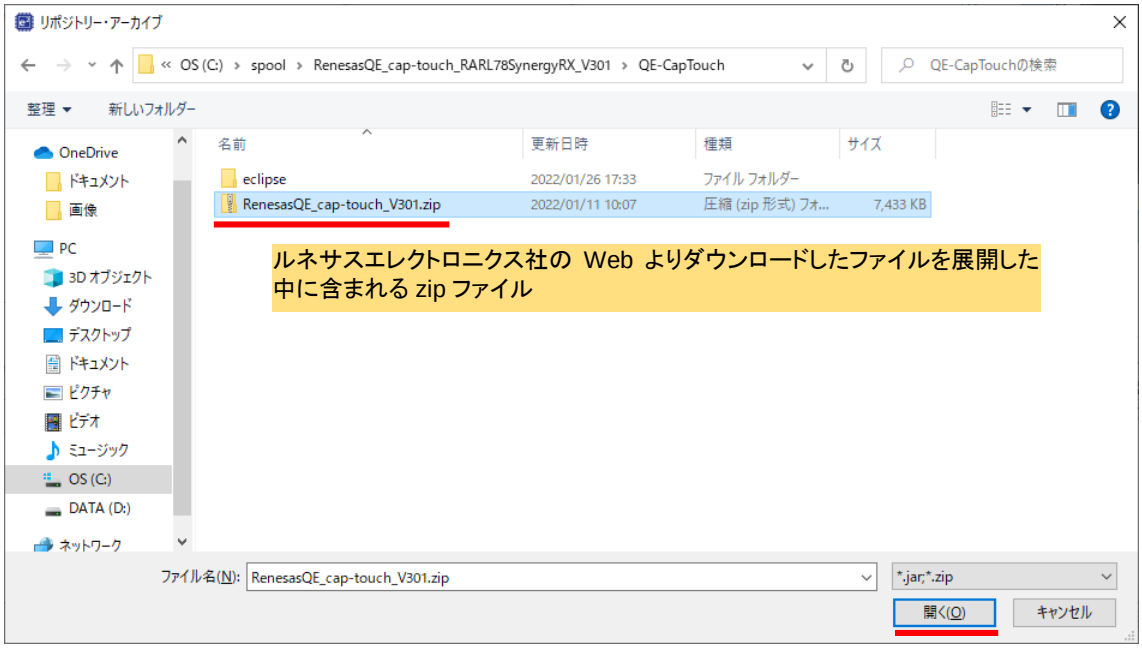


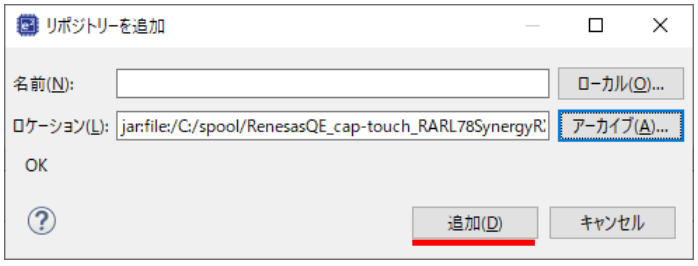
追加



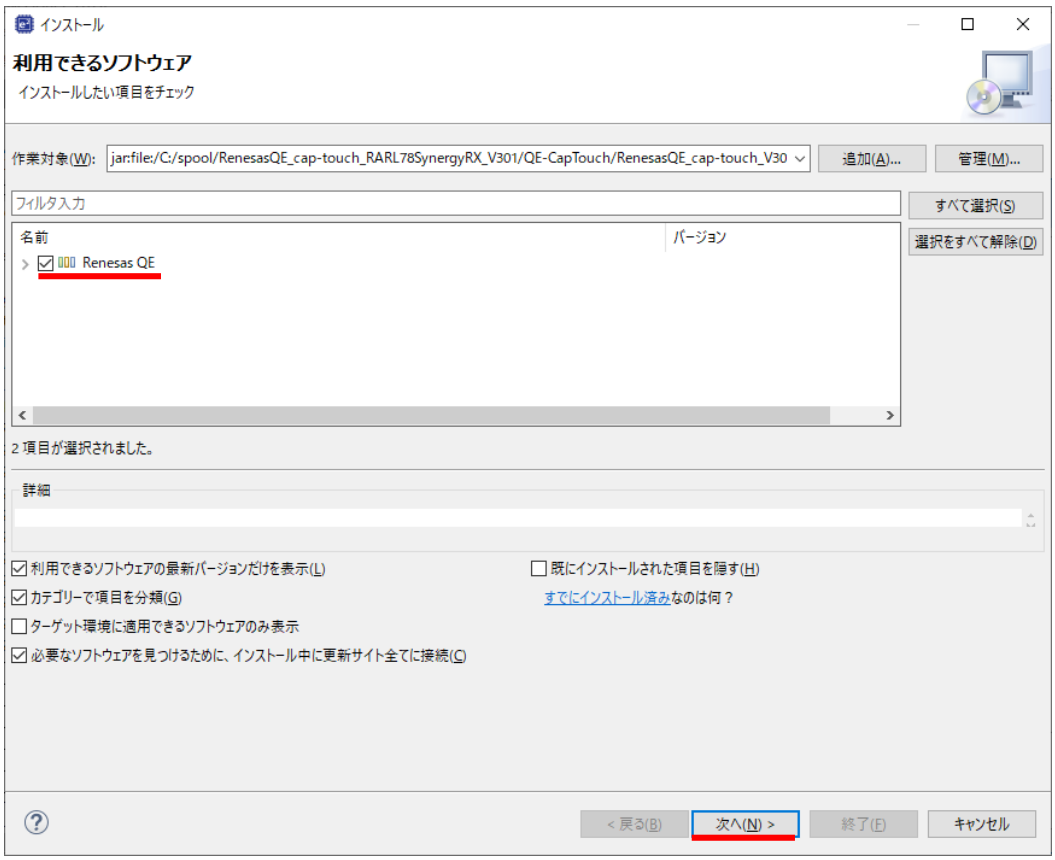
アーカイブで、QE Cap Touch のアーカイブファイル（.zip ファイル）を選択。

※RenesasQE_cap-touch_RARL78SynergyRX_V301.zip を展開し、展開したフォルダに含まれる
RenesasQE_cap-touch_RARL78SynergyRX_V301¥QE-CapTouch¥**RenesasQE_cap-touch_V301.zip**
上記太字の zip ファイルを開く（V3.01 の場合、バージョンは読み替えてください）

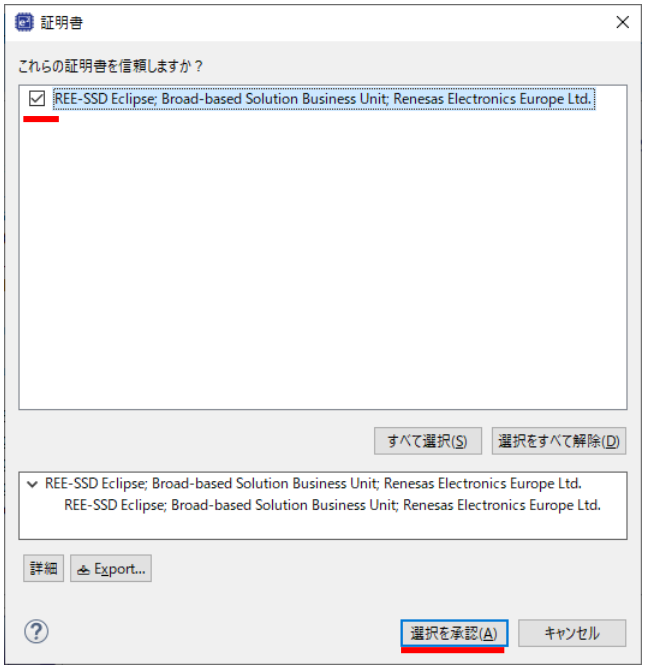




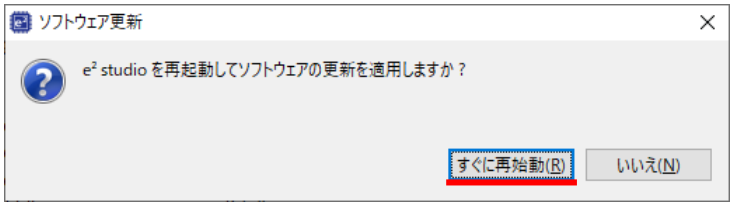
追加



チェックを入れて次へ



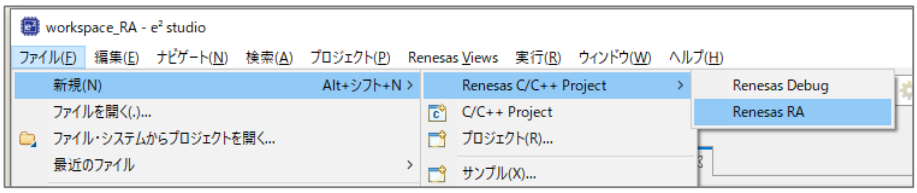
証明書関連は承認してください。



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。

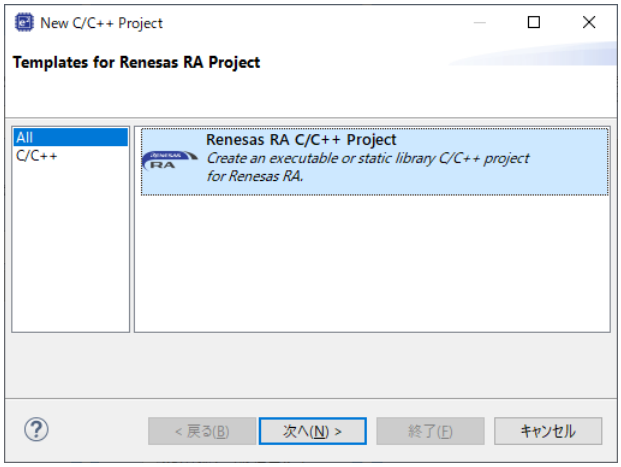
2.2. プロジェクトの新規作成

e2studio（+FSP 環境）上で

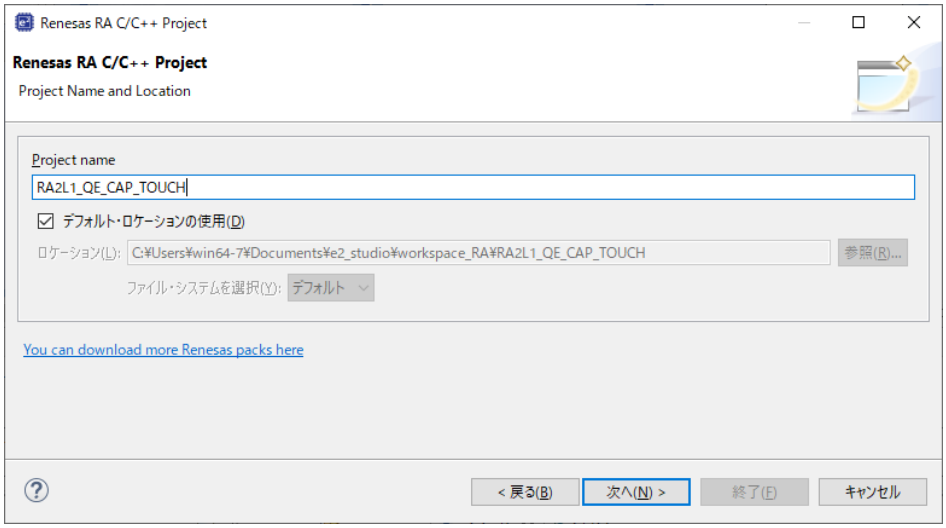


ファイル - 新規 - Renesas C/C++ Project - Renesas RA

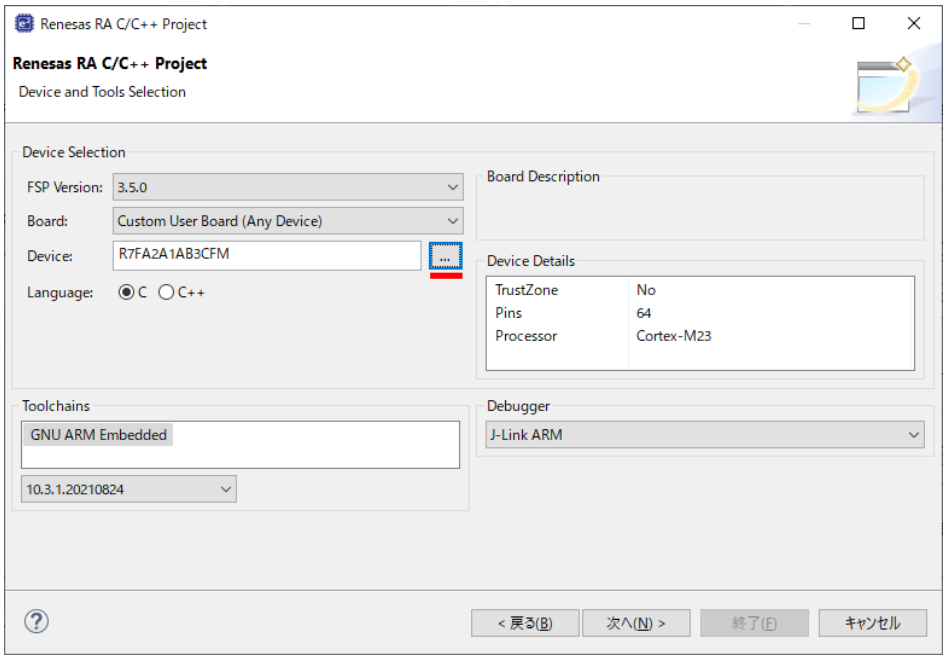
を選択。



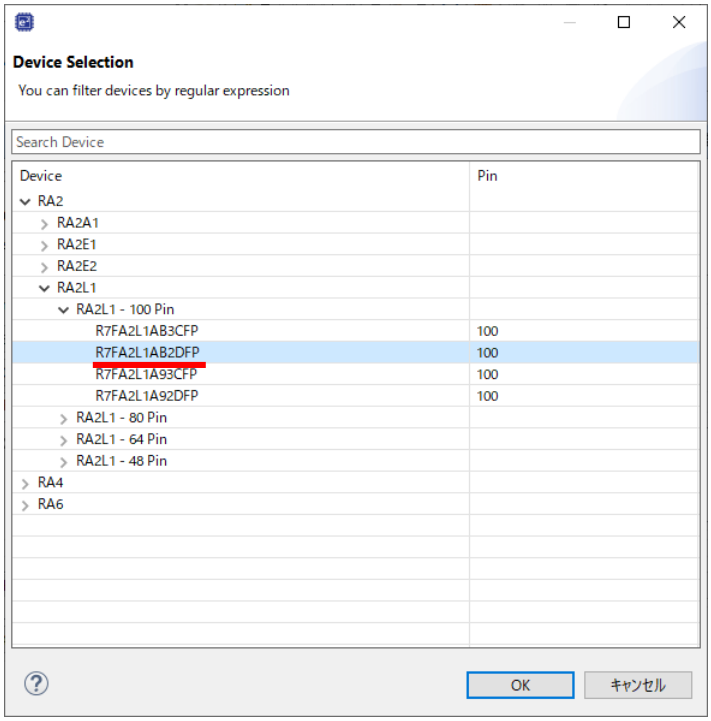
次へ。



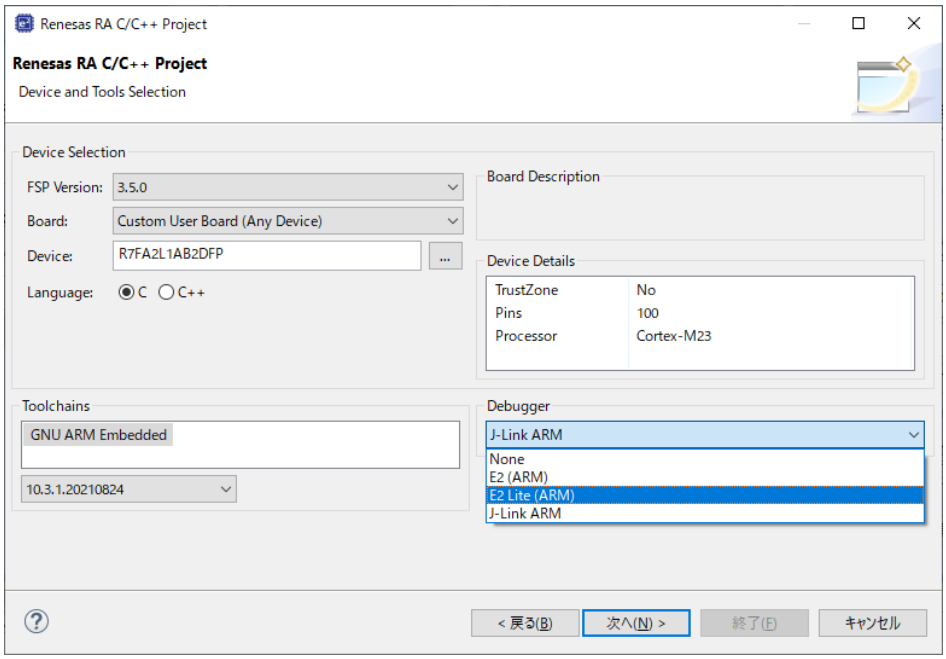
プロジェクト名を入力。



Device の右側のボタンを押し、デバイス選択画面を開く。

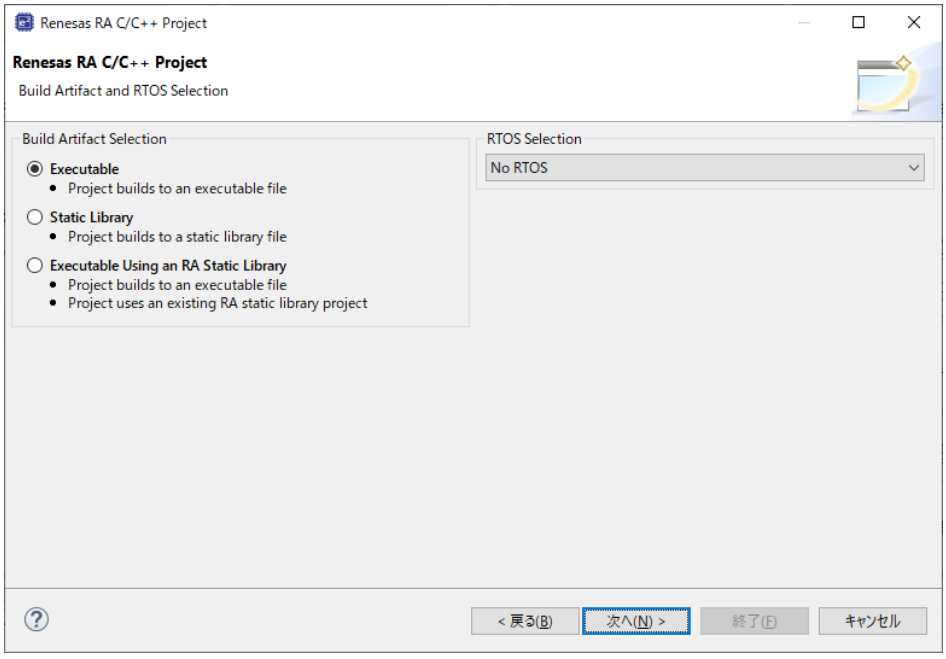


R7FA2L1AB2DFP を選択して「OK」。

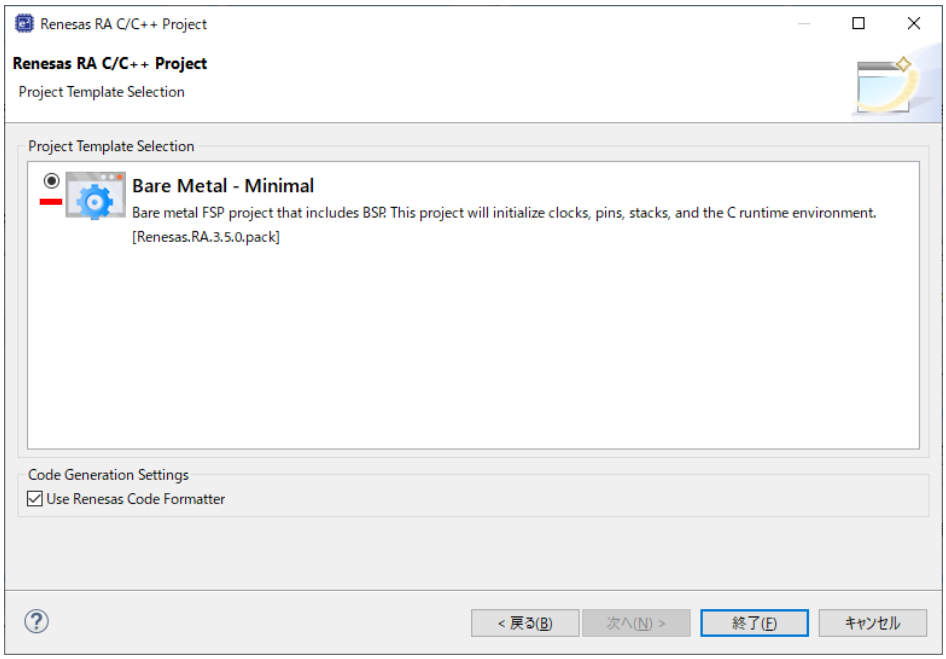


Debugger を選択。（ここでは、E2 Lite(ARM)を選びます）

※QE Cap Tocuh で新規にプロジェクトを作成する場合、エミュレータ（デバッガ）が必要です
（作成済みのプロジェクトを使って動作を見る場合は、UART(SCI)経由で情報のモニタが行えます→エミュレータは不要です）

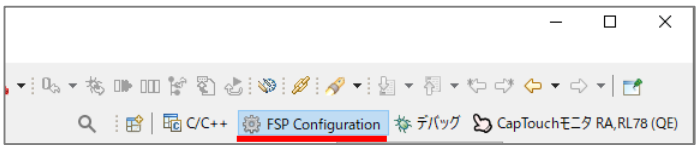


デフォルトのまま「次へ」。



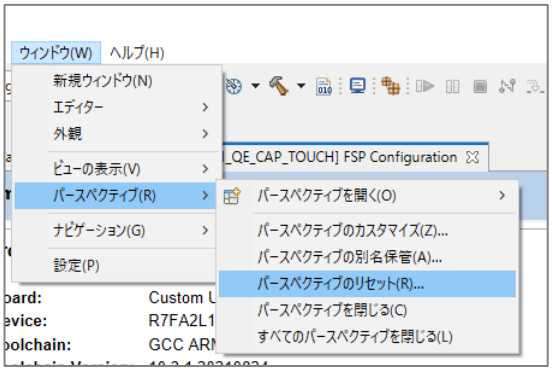
Bare Metal - Minimal にチェックを入れ「終了」。

2.3. FSP の設定



FSP の設定を行う場合は、e2studio ウィンドウ右上のビュー設定を、「FSP Configuration」を選択してください。
※他のビュー設定の場合は、プロパティが表示されません

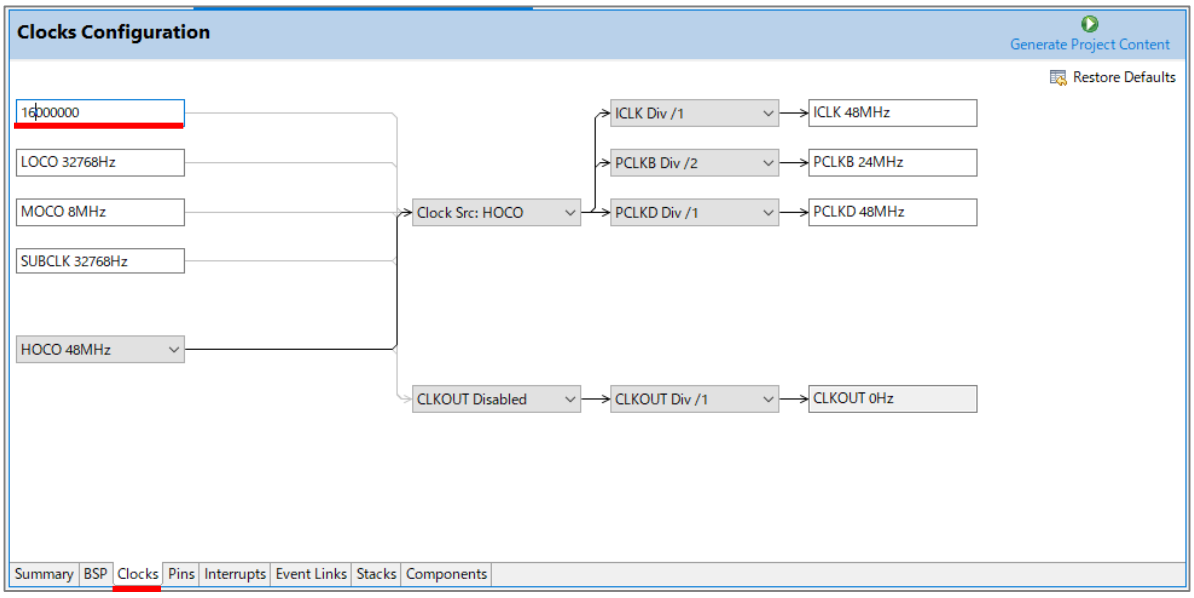
※画面レイアウトが崩れて、どこに何があるか判らなくなった場合は、



ウィンドウ - パースペクティブ - パースペクティブのリセット

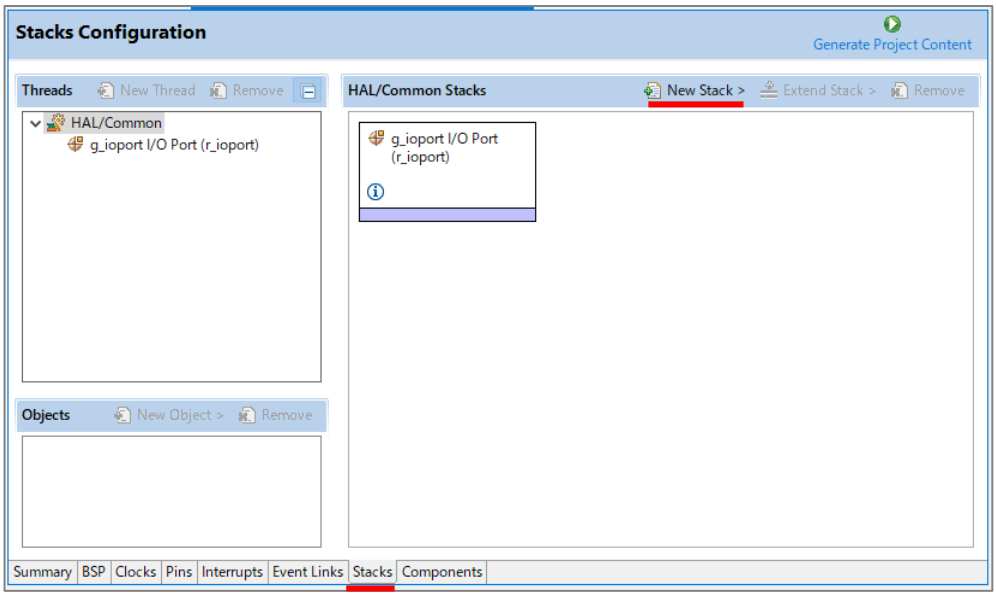
を実行して、画面レイアウトを初期状態に戻してください。

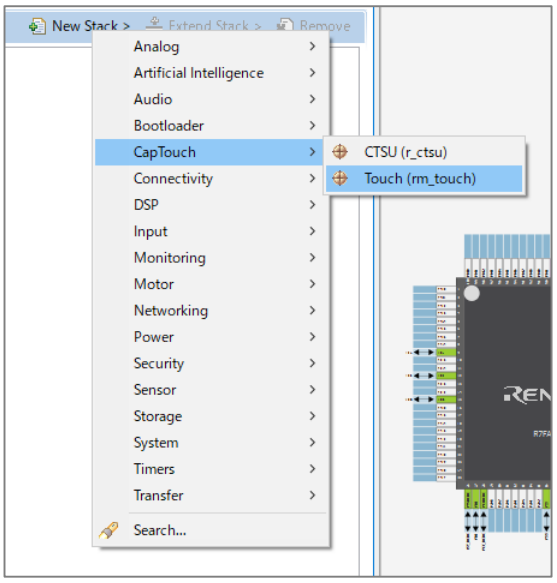
2.3.1. Clocks タブ



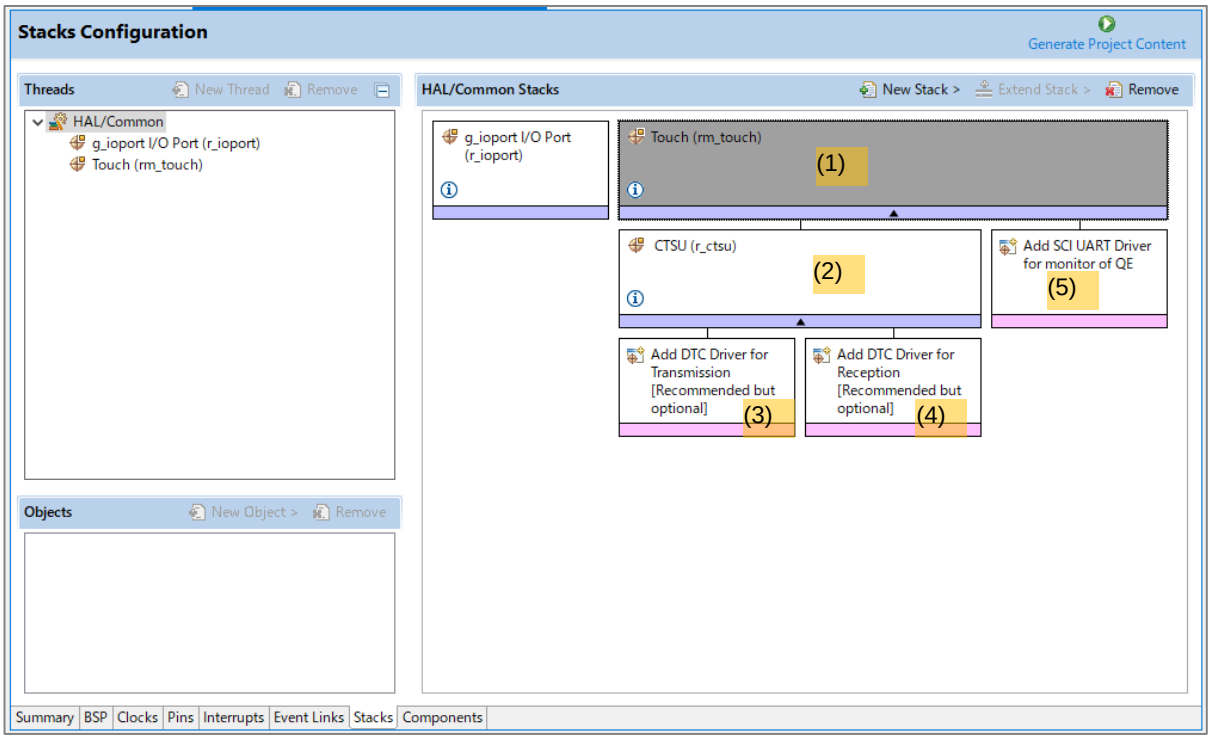
XTAL の値を 16MHz(16000000)に変更する（※タッチキーでは XTAL は使っていないので必須ではありません）。

2.3.2. Stacks タブ



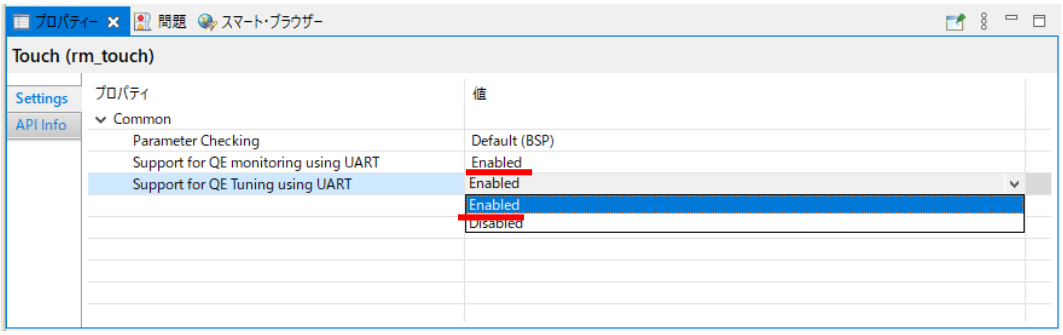


New Stack – CapTouch – Touch(rm_touch)
を追加。



(1)~(5)のスタックが追加されます。

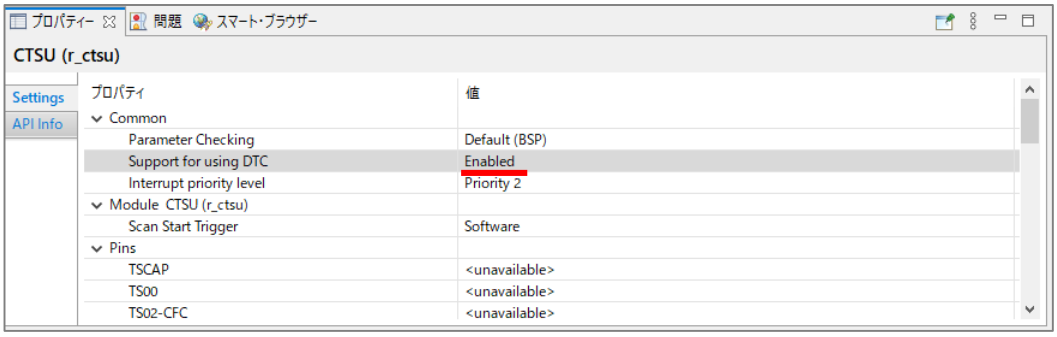
(1)Touch(rm_touch)の設定



Support for QE monitoring using UART : Enabled に変更

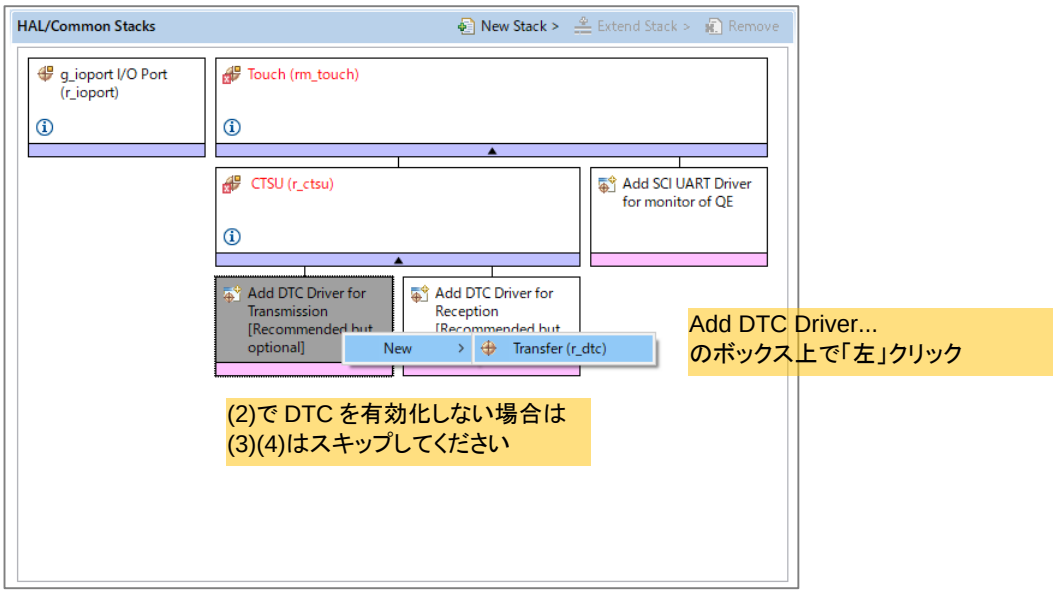
Support for QE Tuning using UART : Enabled に変更

(2)CTSU(r_ctsu)の設定

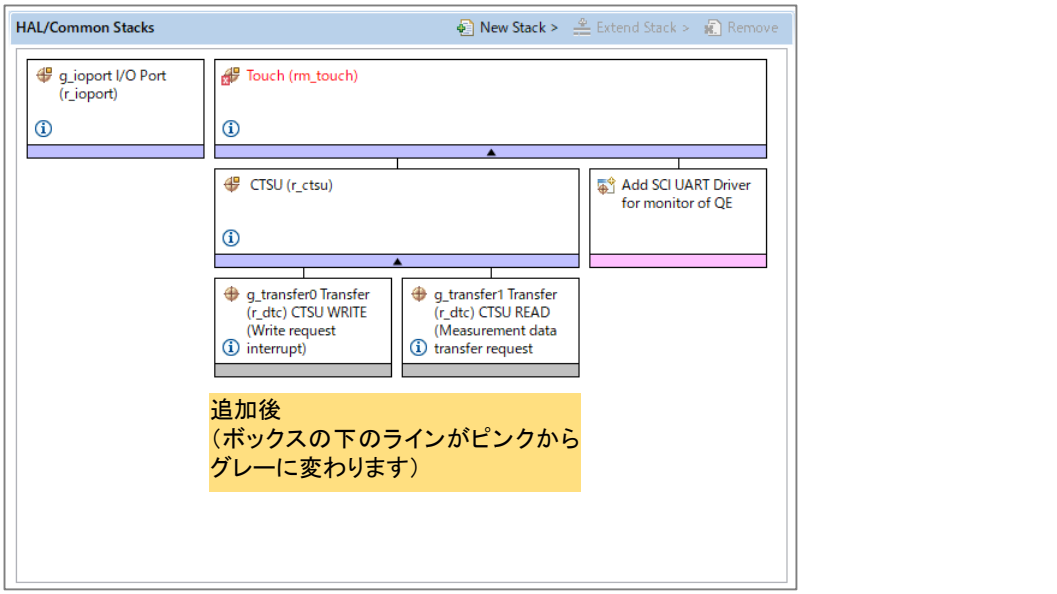


Support for using DTC : Enabled に変更（※DTC を使うかは任意です）

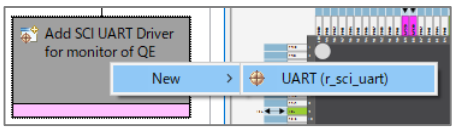
(3)(4)DTC 追加（DTC を使う場合）



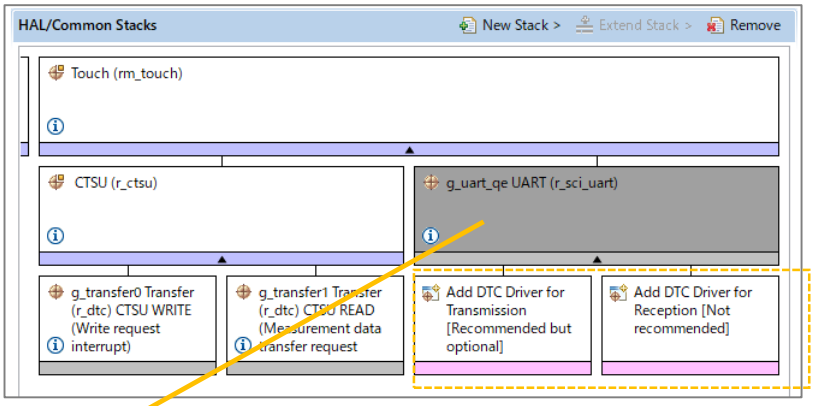
CTSU の下にぶら下がっているスタックに New - Transfer(r_dtc)を追加



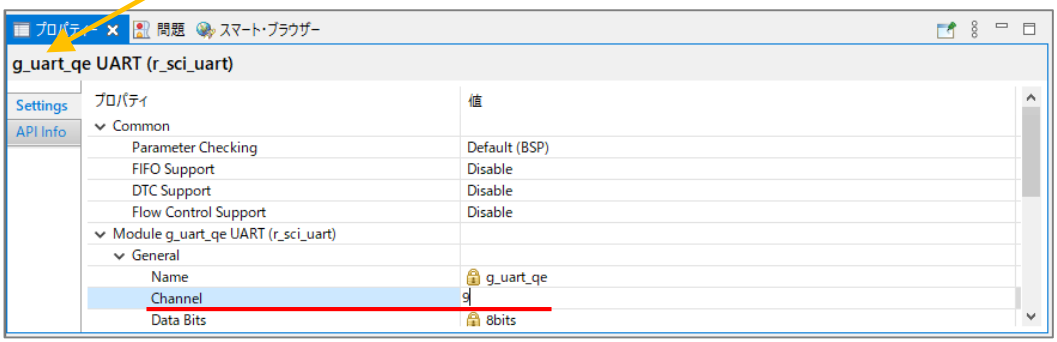
(5)UART の追加



Add SCI UART Driver... のボックスで「左クリック」New - UART(r_sci_uart)を追加。

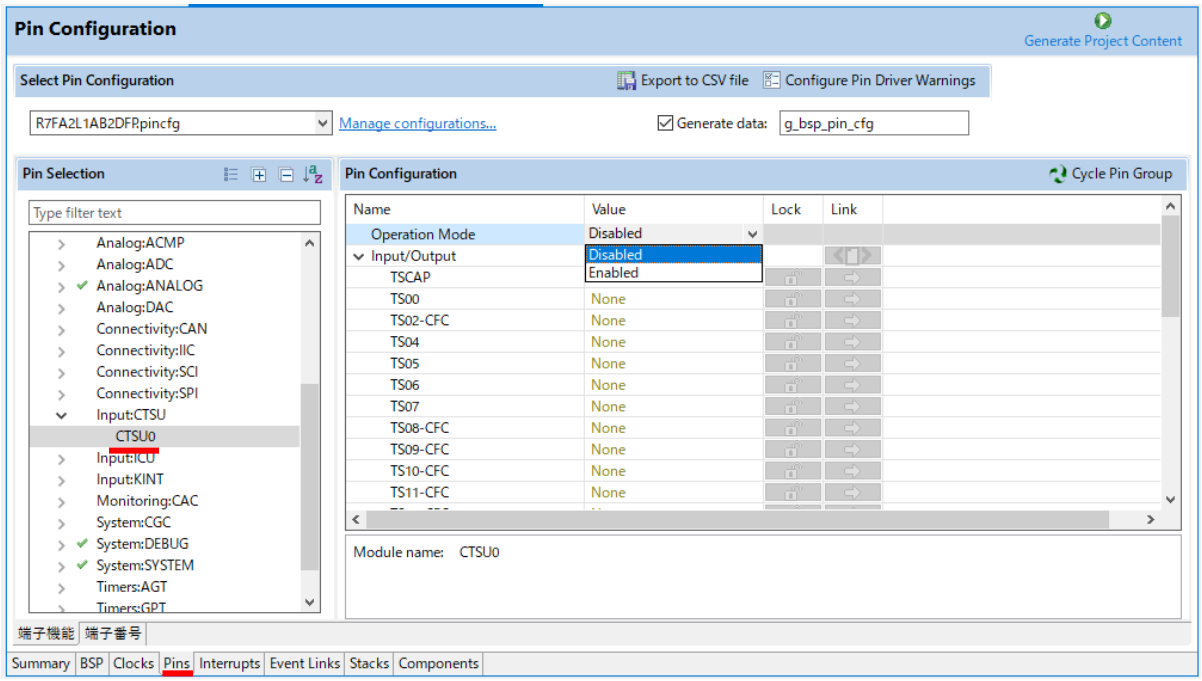


※UART でも DTC 追加は可能です（サンプルプログラムでは追加はしていません）



General - Channel : 9 に変更

2.3.3. Pins タブ



Input:CTSU - CTSU0

Operation Mode : Enabled に変更

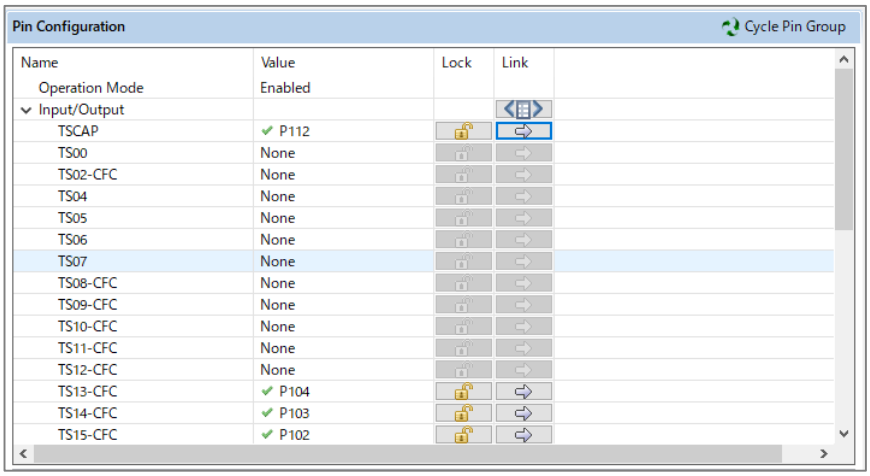
ここで、タッチキーで使用する端子を設定します。

	割り当てる端子	S16A（自己容量基板）	D55A（相互容量基板）
TSCAP	P112		
TS13-CFC	P104	5	TX-0
TS14-CFC	P103(*1)	0	TX-1
TS15-CFC	P102(*2)	6	TX-2
TS16-CFC	P101	1	TX-3
TS26-CFC	P100	7	TX-4
TS28-CFC	P015	2	RX-0
TS30-CFC	P010	9	RX-4
TS31-CFC	P011	4	RX-3
TS32-CFC	P012	8	RX-2
TS33-CFC	P013	3	RX-1

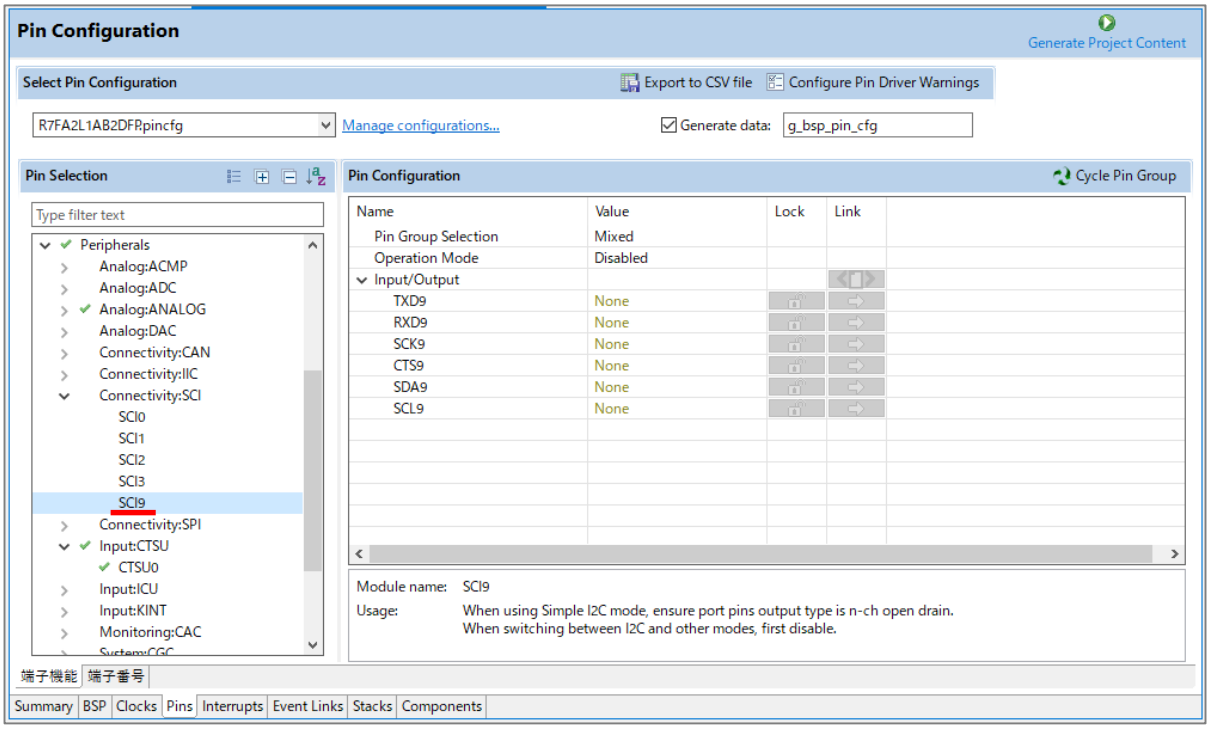
上記の TSxx を Enabled に変更します。

※CTSU, TSxx は端子は複数からは選べません（有効にするか無効にするかの選択です）

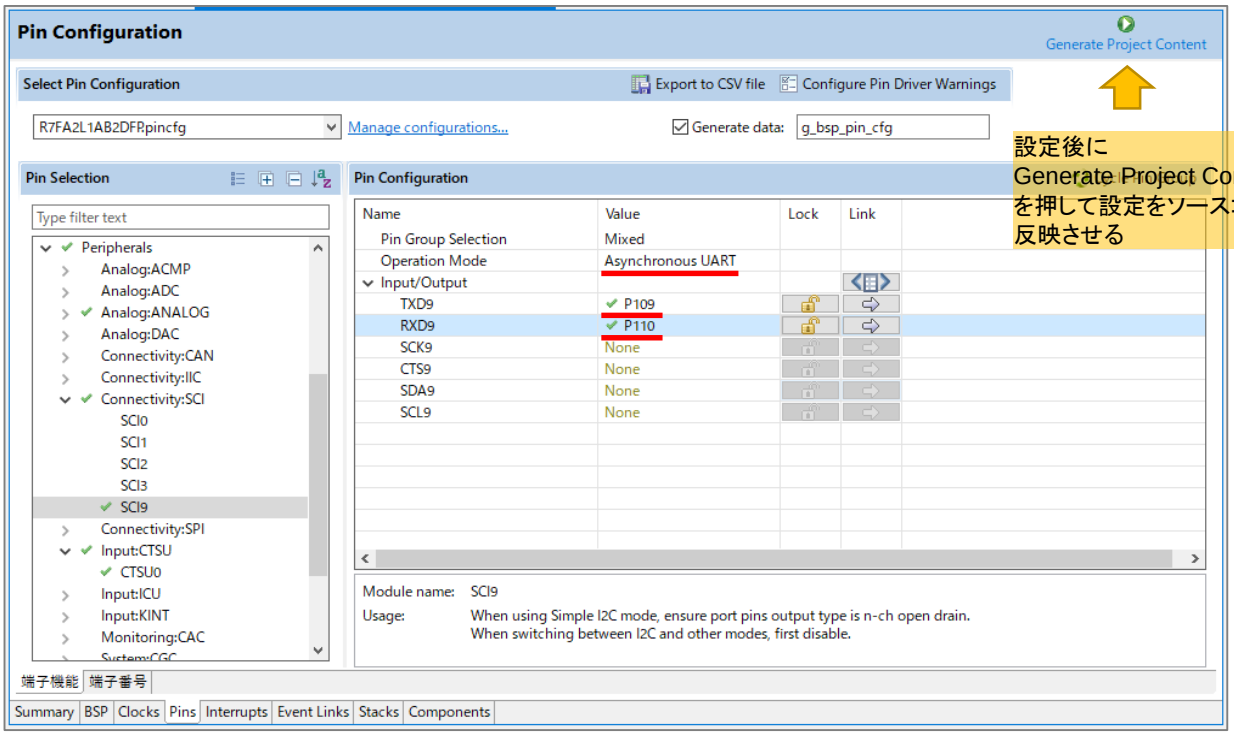
(*1)マイコンボード(HSBRA2L1F100)側で CAN のポートに割り当てられていますので、マイコンボード上の J14 のジャンパ 2 本を抜いて、P103, P102 の端子を CAN の回路と切り離してください。



設定後は、TSCAP, TSxx の端子の一部が有効化された状態となります。



Connectivity:SCI - SCI9 の設定を変更します。



Operation Mode Asyncronous : UART

TXD9 : P109

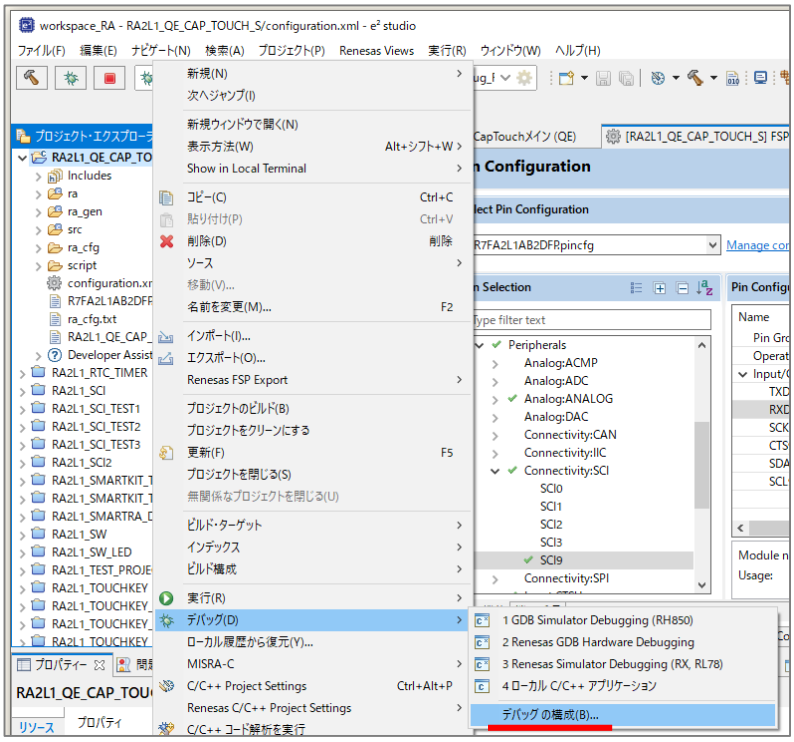
RXD9 : P110

を選択

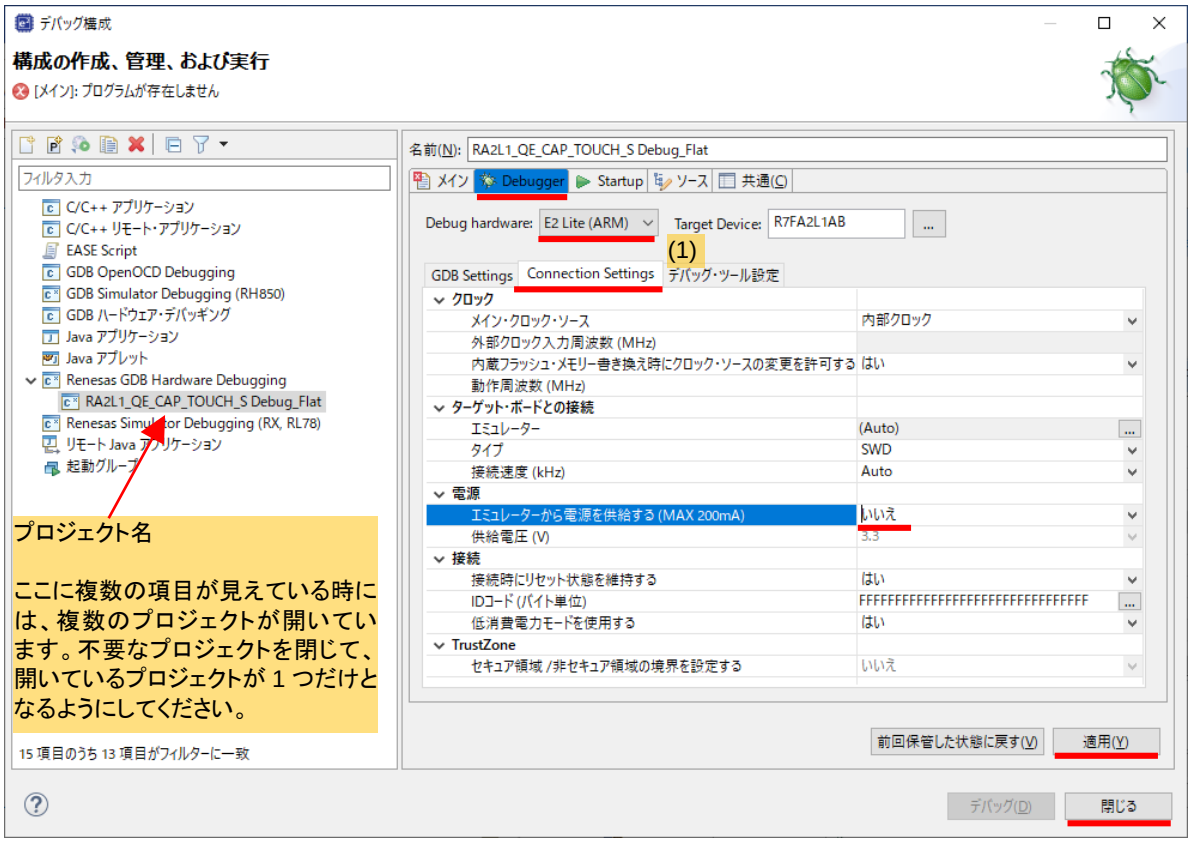
一通り設定が終わったら、「Generate Project Content」のボタンを押してソースコードを出力します。

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



(1)使用エミュレータ

使用するエミュレータを選択してください。

※E2 をマイコンボード 14P コネクタで使用する場合は J8 ジャンパを 1-2 側に設定。E2Lite を使用する場合は、2-3 側に設定してください。（オプションの 20P コネクタを使用する場合はジャンパの設定は不要です。）

Connection Settings - エミュレータから電源を供給する ： いいえ

を選んでください。

「適用」、「閉じる」で設定画面を閉じてください。

※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。

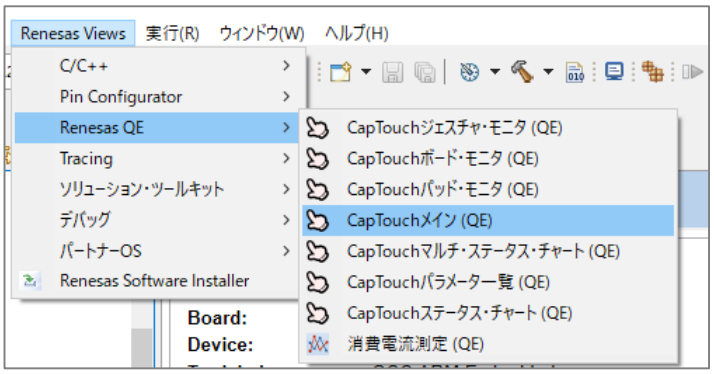
※なお、E2Lite をお使いの場合は、エミュレータから供給する電圧は、3.3V しか選択できません
（E2 では、5V と 3.3V が選択可能です）

※「エミュレータ」と「デバッガ」、「Debugger」は同義です

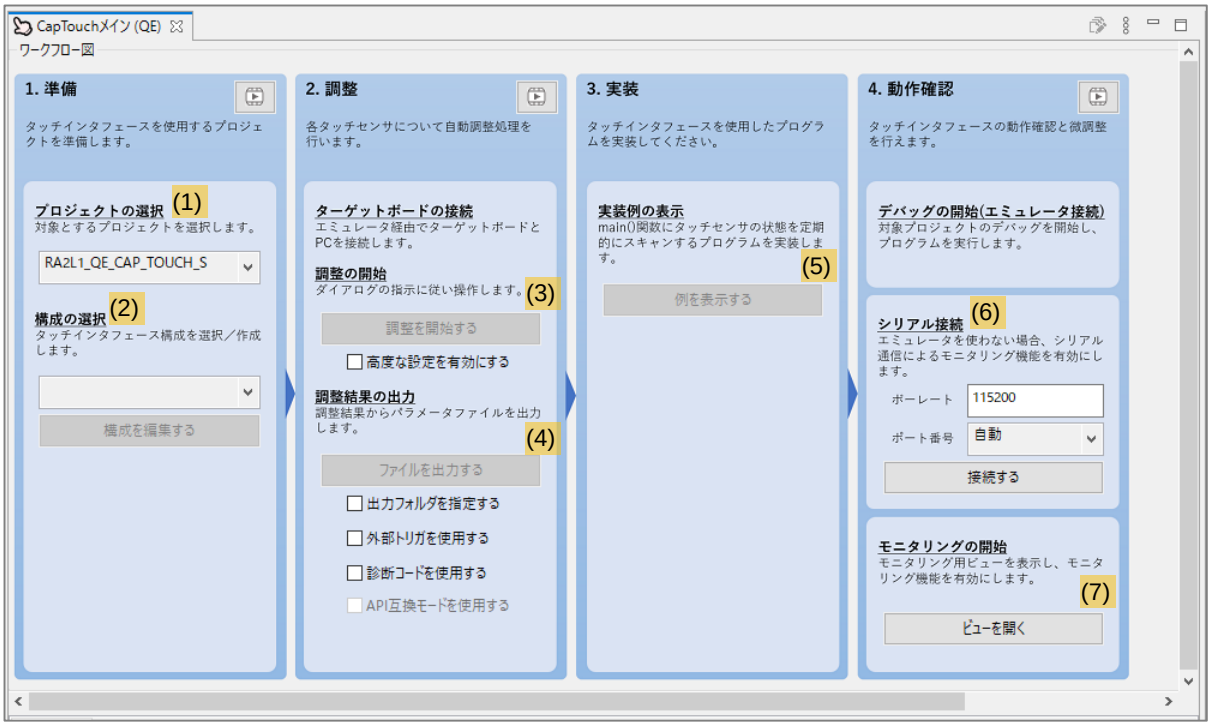
3. QE CapTouch の使用

3.1. CapTouch の起動

e2studio + FSP 環境でプロジェクトを作成し、



Renesas Views - CapTouch メインを実行します。

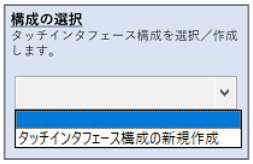


全体の流れ（ワークフロー）が表示されます。

(1)プロジェクトの選択

新しく作成したプロジェクトを指定します。

(2)構成の選択

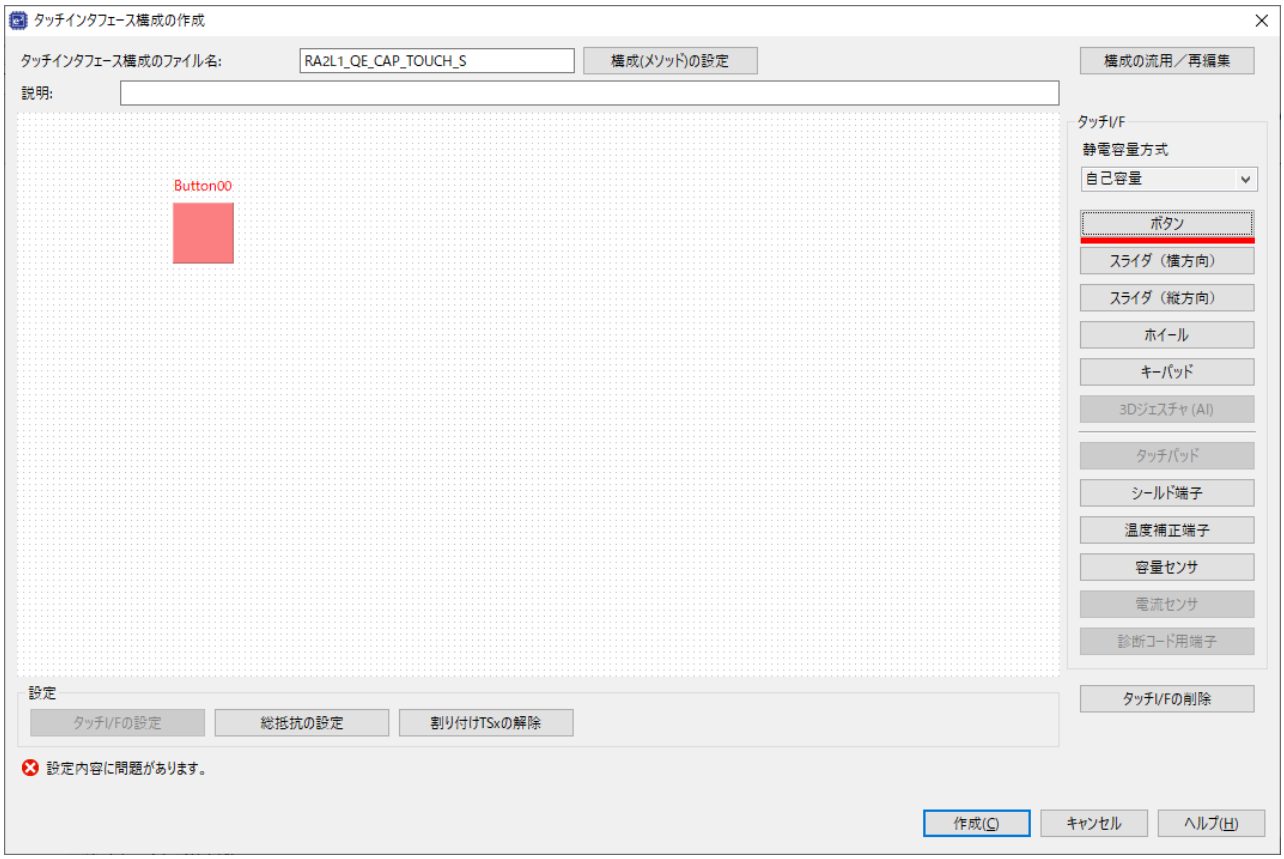


タッチインタフェース構成の新規作成

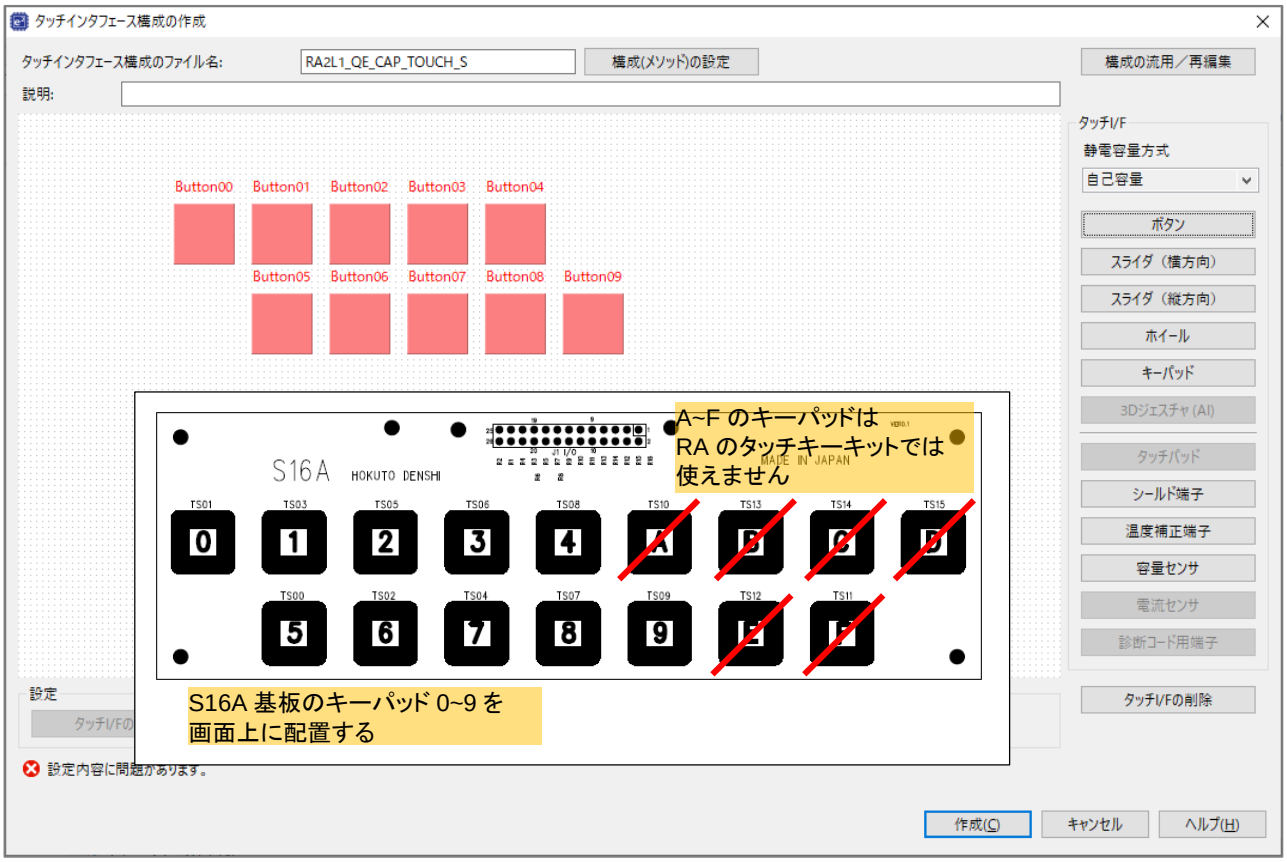
を選びます。

3.2. タッチキーインタフェースの作成

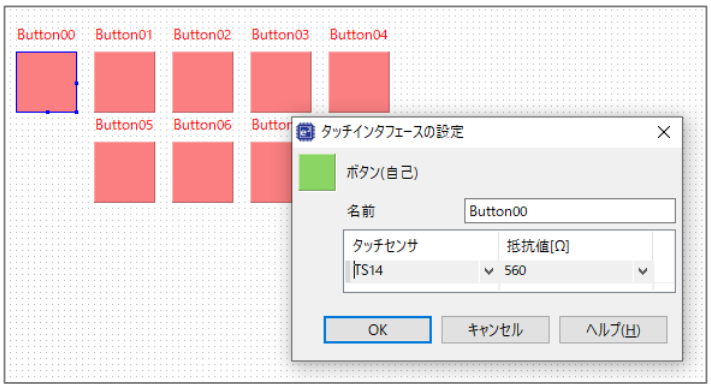
3.2.1. 自己容量基板(S16A)



「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。



RA2L1 のタッチキーキットでは、0~9 の 10 個のキーパッドが使用可能なので、同じような画面イメージとなる様に 10 個の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、TS14 なので、TS14 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)

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ん。)

・キーパッドと TS の対応

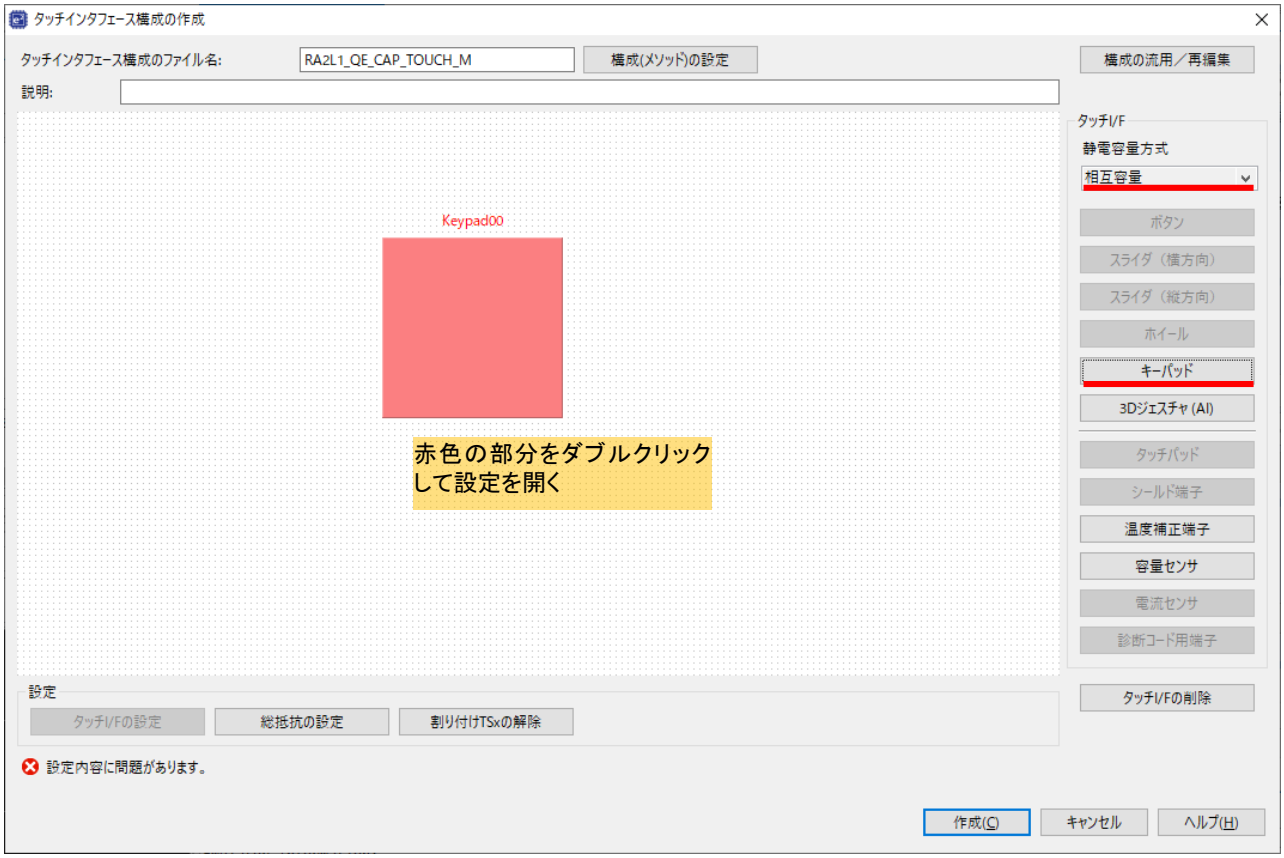
S16A（自己容量基板） キーパッド	TSxx 番号
0	TS14
1	TS16
2	TS28
3	TS33
4	TS31
5	TS13
6	TS15
7	TS26
8	TS32
9	TS30

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。

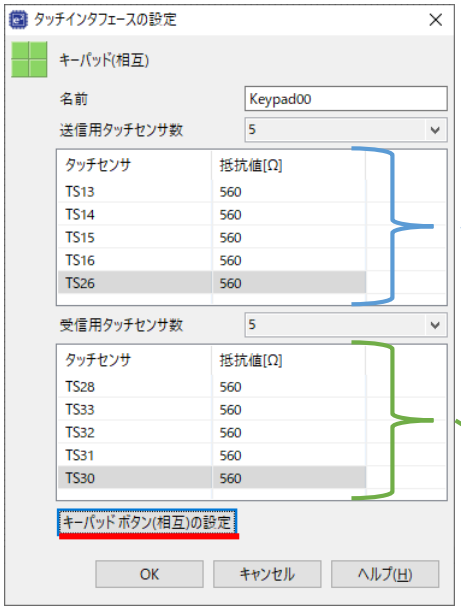
3.2.2. 相互容量基板(D55A)

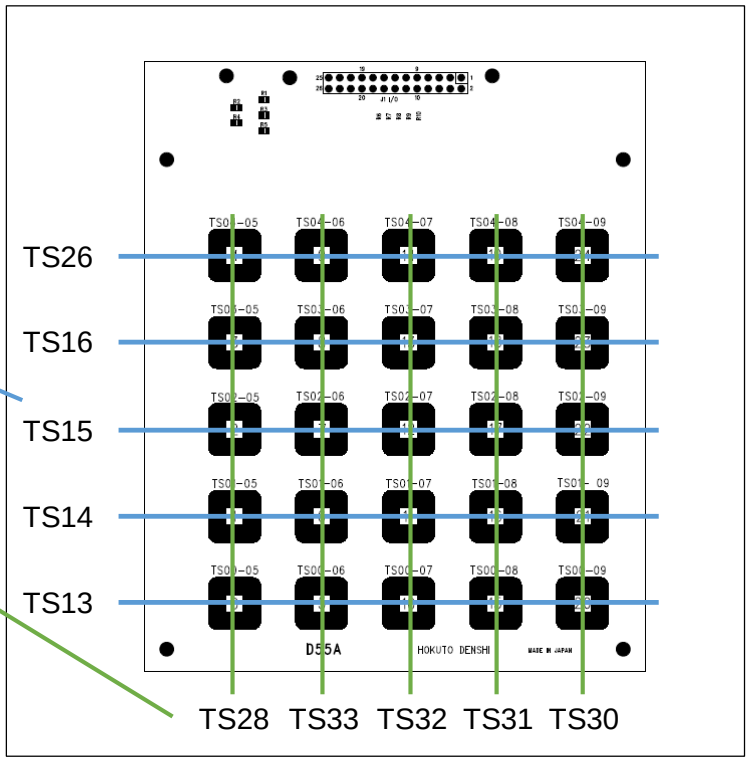
相互容量基板を用いる際のタッチインタフェースの作成に関しては、以下の様になります。



静電容量方式 ： 相互容量

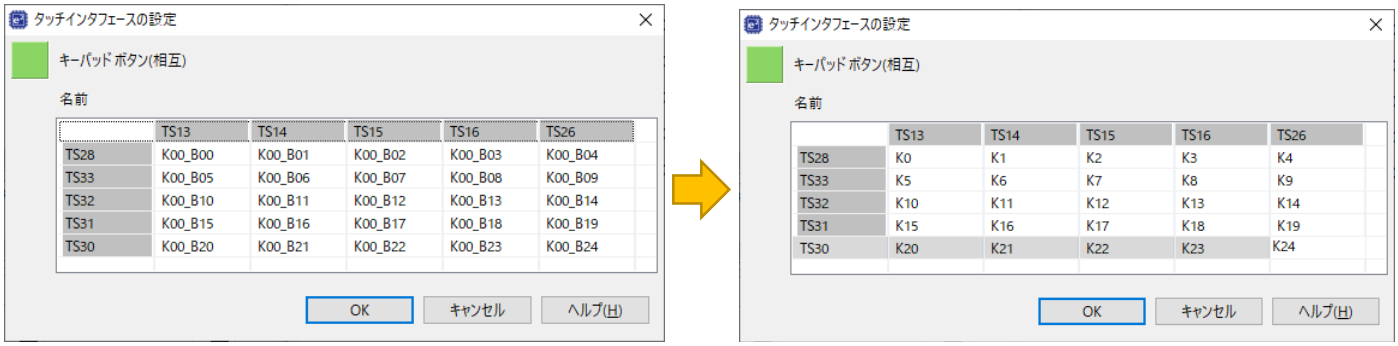
「キーパッド」を選択して配置。





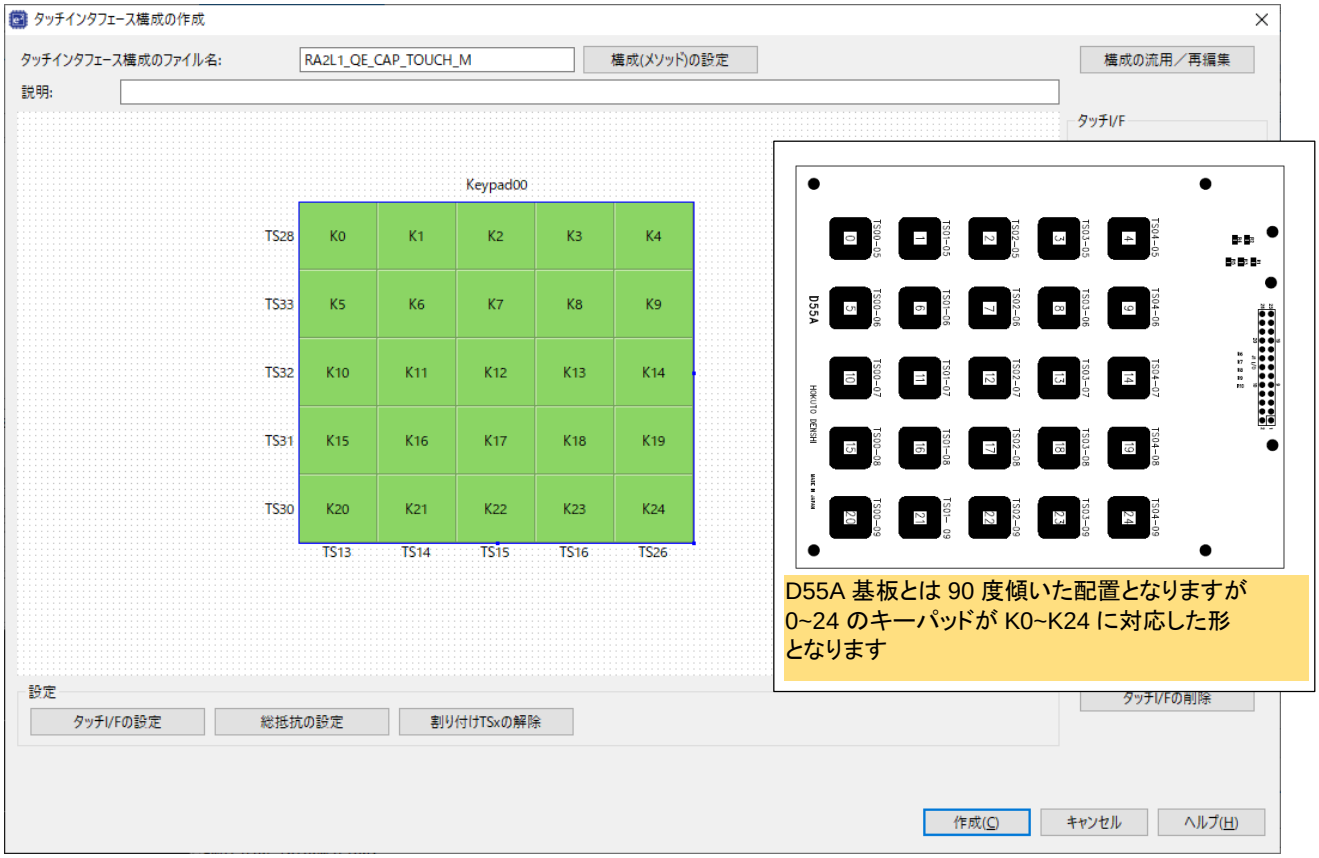
送信用のところに、TS13~TS26 を設定。受信用に、TS28~TS30 を設定する。抵抗値は、560Ω(デフォルトのまま)としてください。

「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。

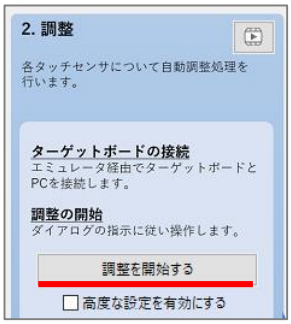
※数字から始まる名前に設定できないので Kn としています



3.3. タッチキーインタフェースの調整

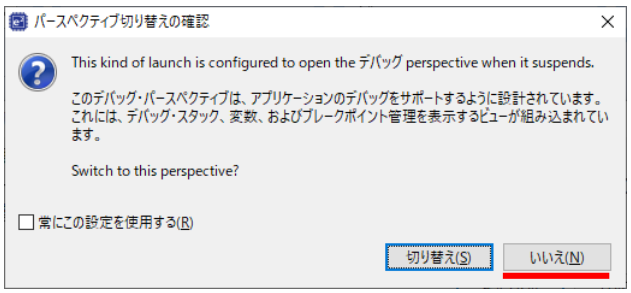
マイコンボードとタッチキーボード基板（S16A または D55A）を接続。

(3)調整

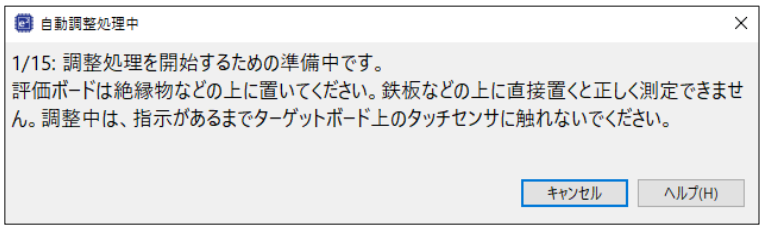


「調整を開始する」を押す。

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。

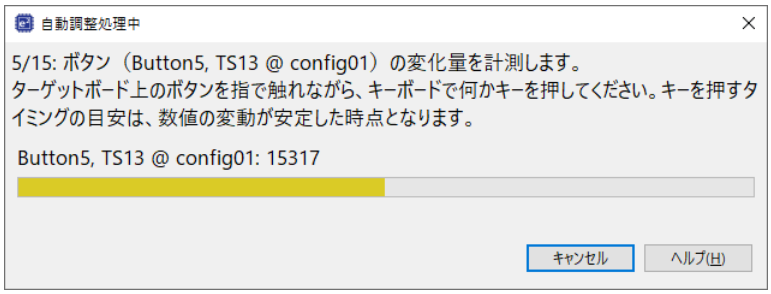


上記メッセージが出た場合は、（どちらでも構いませんが）ここでは「いいえ」を押します。



上記のメッセージが出力され、1/15, 2/15, ...の様に処理が進んでいきます。

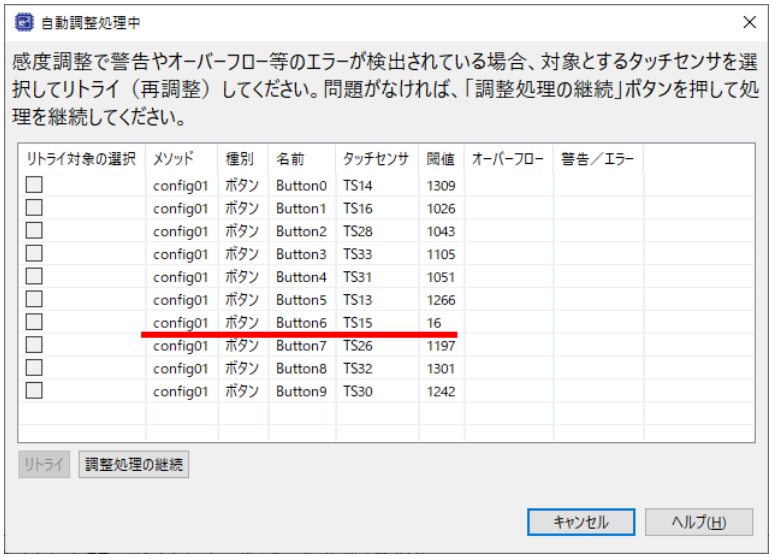
3.3.1. 自己容量基板(S16A)



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

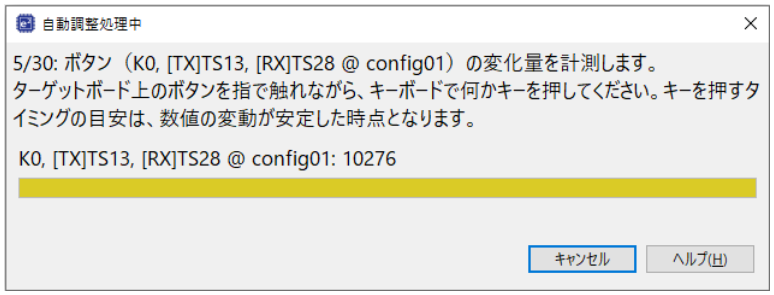


上記の様に「閾値」の値が極端に小さいボタンがあった場合タッチに失敗しています。「リトライ対象の選択」にチェックを入れ、「リトライ」を行ってください。

※閾値が 65535 になっている場合も NG です（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）

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。

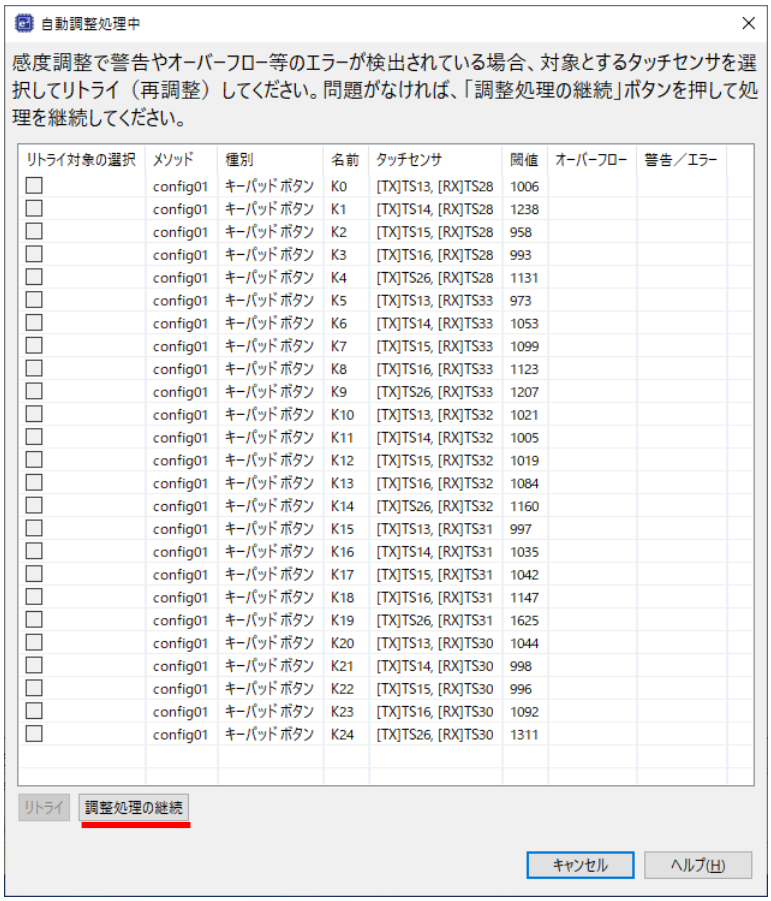
3.3.2. 相互容量基板(D55A)



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。

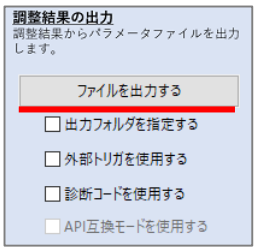
※キーの順番ですが、0→1→2→3→4→20→…と 5 個のキー（列方向）は順番ですが、行はばらばらの順となる事に注意ください



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」で次に進む。

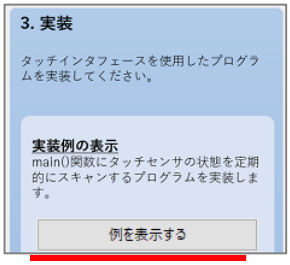
3.4. ソースコードの変更

(4)パラメータファイルの出力

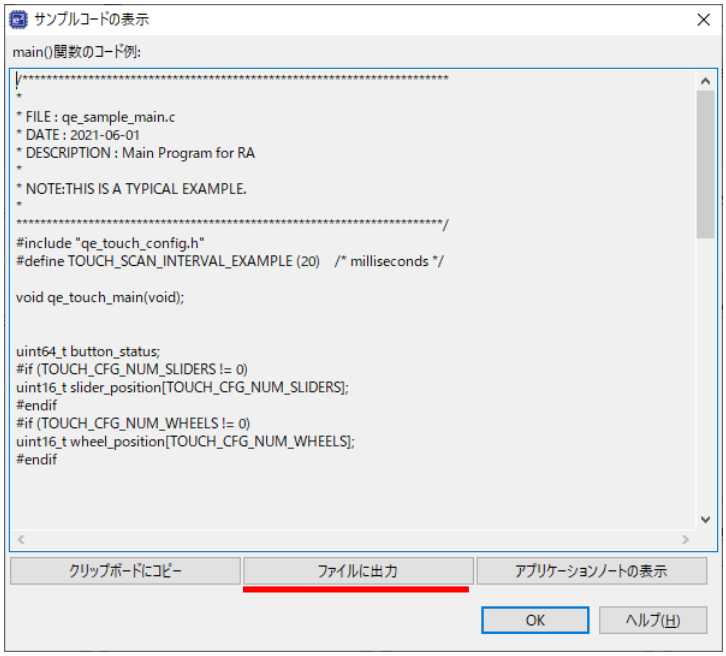


「ファイルを出力する」。

(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

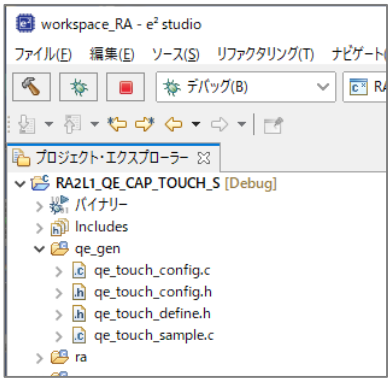


「例を表示する」。



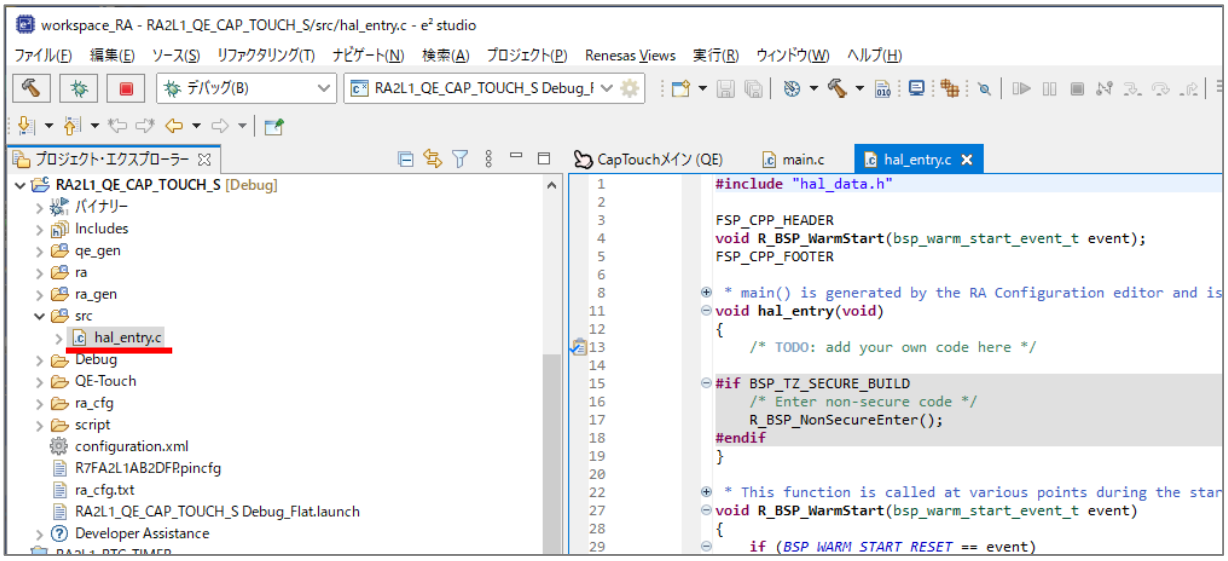
ユーザプログラム内に記載するテンプレートが出力されるので、この記述を参考に、
src/hal_entry.c （ユーザプログラムの main 関数に相当する）
に、タッチキーのプログラムコードを追加してください。

ここでは、ファイルに出力を押して、QE Cap Touch が生成したプログラムをファイルに保存します。



上記に保存されます。

サンプルプロジェクト、
RA2L1_QE_CAP_TOUCH_S（自己容量）
RA2L1_QE_CAP_TOUCH_M（相互容量）
には、LCD を使った表示例が、hal_entry.c 内に記載されています。



上記ファイルが、一般的な main 関数に相当する hal_entry()（FSP が呼び出すユーザプログラムのメイン関数）を含むファイルですので、このファイルにユーザプログラムを記載、または hal_entry()から別ファイルに記載したプログラムを呼び出す様にしてください。

3.4.1. 自己容量基板(S16A)

・src/hal_entry.c

```
#include "hal_data.h"
#include "./lcd_1602/lcd_1602.h"

extern void qe_touch_main(void);

FSP_CPP_HEADER
void R_BSP_WarmStart(bsp_warm_start_event_t event);
FSP_CPP_FOOTER

/*******************************************************************************************************************
**************//**
 * main() is generated by the RA Configuration editor and is used to generate threads if an RTOS is
used.  This function
 * is called by main() when no RTOS is used.

*******************************************************************************************************************
****************/
void hal_entry(void)
{
    /* TODO: add your own code here */

    lcd_init();
    lcd_clear();
    lcd_hs1();
    //            1234567890123456
    lcd_write_str("RA2L1 SelfCap QE");

    R_BSP_SoftwareDelay(1000, BSP_DELAY_UNITS_MILLISECONDS);

    qe_touch_main();

#if BSP_TZ_SECURE_BUILD
    /* Enter non-secure code */
    R_BSP_NonSecureEnter();
#endif
}
```

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

```
/***********************************************************************
*
* FILE : qe_sample_main.c
* DATE : 2021-06-01
* DESCRIPTION : Main Program for RA
*
* NOTE:THIS IS A TYPICAL EXAMPLE.
*
***********************************************************************/
#include "qe_touch_config.h"
#include "../src/lcd_1602/lcd_1602.h"

（中略）
```

・qe_gen/qe_sample.c（続き）

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

    int i;
    unsigned long x;

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

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

    /* Open Touch middleware */
    err = RM_TOUCH_Open(g_qe_touch_instance_config01.p_ctrl, g_qe_touch_instance_config01.p_cfg);
    if (FSP_SUCCESS != err)
    {
        while (true) {}
    }

    /* Main loop */
    while (true)
    {

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

        err = RM_TOUCH_DataGet(g_qe_touch_instance_config01.p_ctrl, &button_status, NULL, NULL);
        if (FSP_SUCCESS == err)
        {

            /* TODO: Add your own code here. */
            lcd_hs2();
            x = 0x1UL;
            for(i=0; i<10; 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);
        }
（後略）
```

測定 ch の若い順ーキーパッドの対応
のテーブルを定義

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---

3.4.2. 相互容量基板(D55A)

・src/hal_entry.c

```
#include "hal_data.h"
#include "./lcd_1602/lcd_1602.h"

extern void qe_touch_main(void);

FSP_CPP_HEADER
void R_BSP_WarmStart(bsp_warm_start_event_t event);
FSP_CPP_FOOTER

/*******************************************************************************************************************
**************//**
 * main() is generated by the RA Configuration editor and is used to generate threads if an RTOS is
used.  This function
 * is called by main() when no RTOS is used.

*******************************************************************************************************************
***************/
void hal_entry(void)
{
    /* TODO: add your own code here */

    lcd_init();
    lcd_clear();
    lcd_hs1();
    //            1234567890123456
    lcd_write_str("RA2L1 MutCap QE");

    R_BSP_SoftwareDelay(1000, BSP_DELAY_UNITS_MILLISECONDS);

    qe_touch_main();

#if BSP_TZ_SECURE_BUILD
    /* Enter non-secure code */
    R_BSP_NonSecureEnter();
#endif
}
```

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

```
/***********************************************************************
*
* FILE : qe_sample_main.c
* DATE : 2021-06-01
* DESCRIPTION : Main Program for RA
*
* NOTE:THIS IS A TYPICAL EXAMPLE.
*
***********************************************************************/
#include "qe_touch_config.h"
#include "../src/lcd_1602/lcd_1602.h"

（中略）
```

・qe_gen/qe_sample.c（続き）

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

    int i;
    unsigned long x;
    unsigned char key_table[25] = {0, 1, 2, 3, 4, 20, 21, 22, 23, 24, 15, 16, 17, 18, 19,
                10, 11, 12, 13, 14, 5, 6, 7, 8, 9};   //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:  ");

    /* Open Touch middleware */
    err = RM_TOUCH_Open(g_qe_touch_instance_config01.p_ctrl, g_qe_touch_instance_config01.p_cfg);
    if (FSP_SUCCESS != err)
    {
        while (true) {}
    }

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

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

測定 ch の若い順ーキーパッドの対応のテーブルを定義

LCD 画面に、タッチしているキーの情報を表示

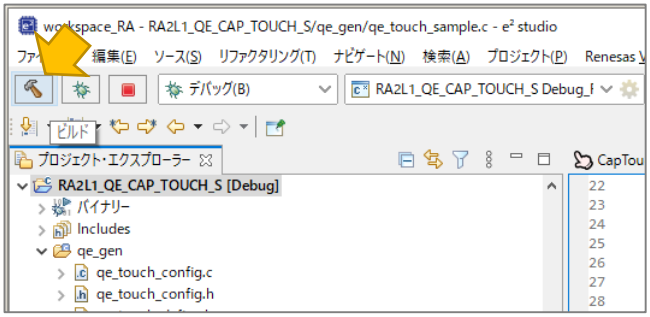
表示例ー

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

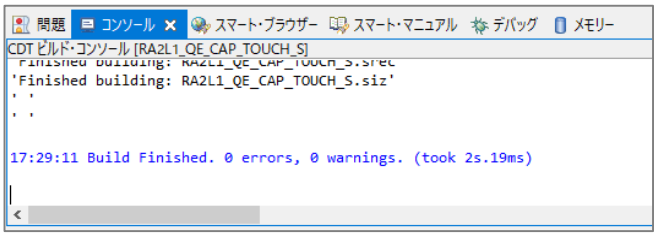
↑ button status　　↑ タッチキーパッド番号

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

3.5. ビルド・実行



「ビルド」を実行。



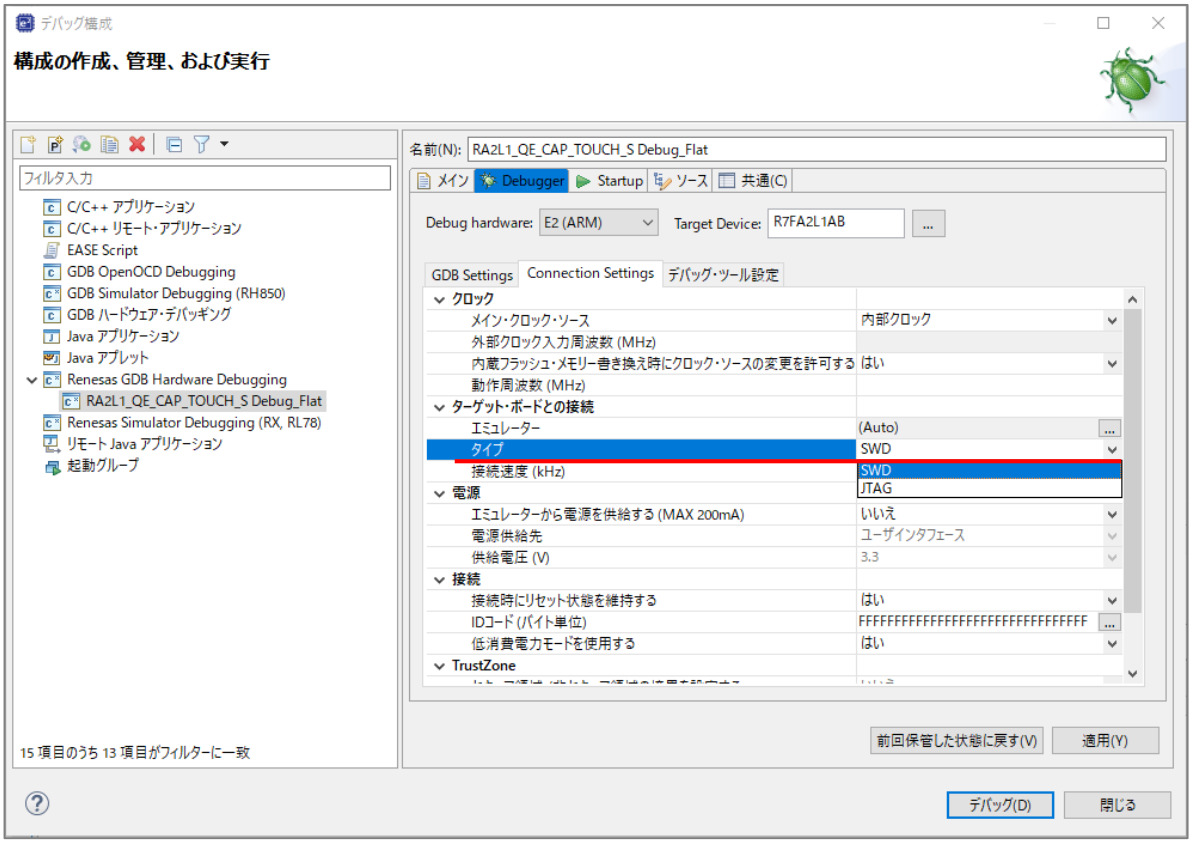
エラーや意図しないワーニングがなければ問題ありません。

・エミュレータ接続（E2, E2Lite 等をお持ちの場合）
・マイコンにプログラムを書き込む（デバッガをお持ちでない場合）
（ハードウェア編のマニュアルを参照し、製品付属の USB-ADAPTER-RX14 でプログラムの書き込みを行ってください）

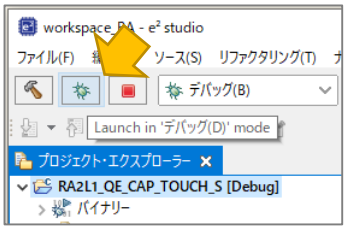
どちらかの方法で作成したプログラムを RA2L1 のマイコンに書き込みを行ってください。

以下は、エミュレータ接続のケースで説明します。

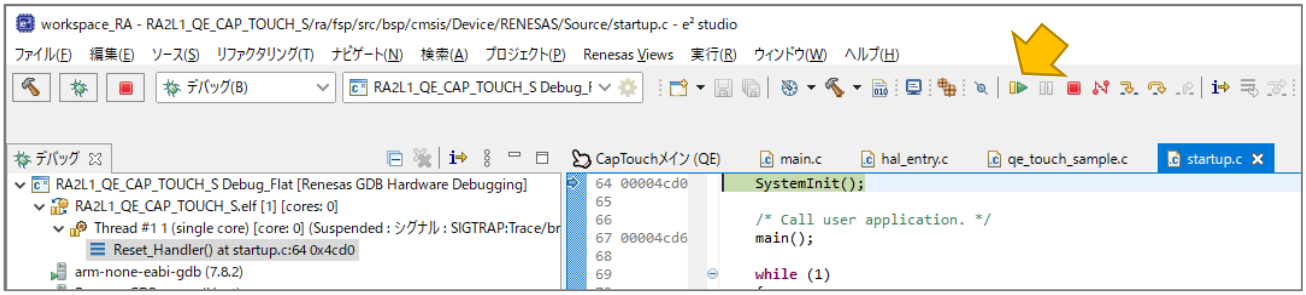
※E2 エミュレータを使用する場合



タイプとして「SWD」を選択してください（デフォルト）。



「デバッグ」を実行。

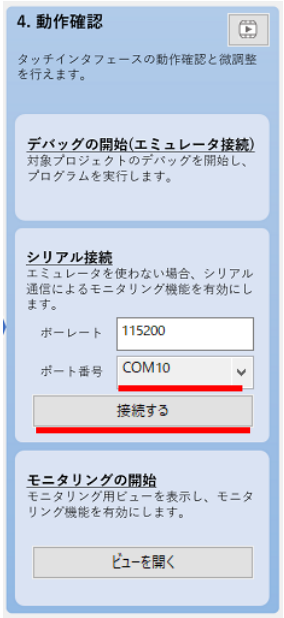


ブレークポイントで止まりますので、「再開」ボタン（もしくは F8）で進めてください（2 回押す）。

3.6. モニタリング

CapTouch モニタのビューを有効化。

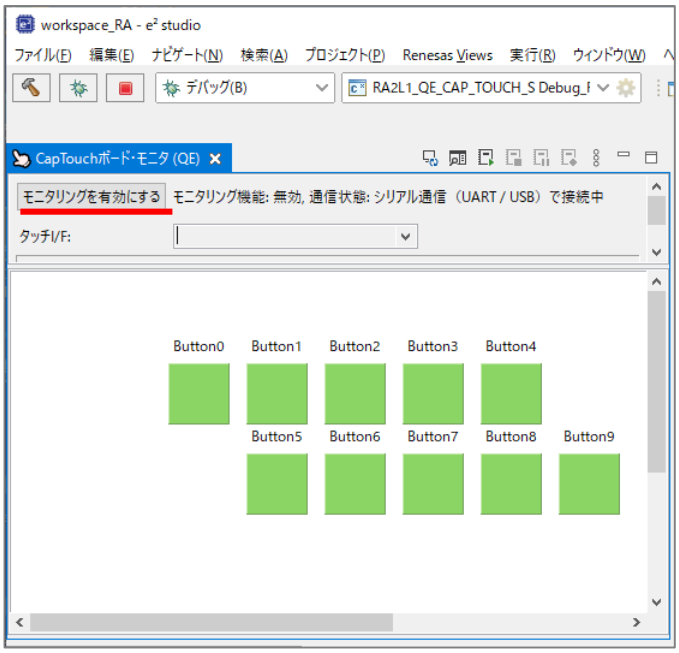
(6)シリアル接続



モニタリングは、
「エミュレータ接続」
「シリアル接続」
の 2 種類の接続方法があります。

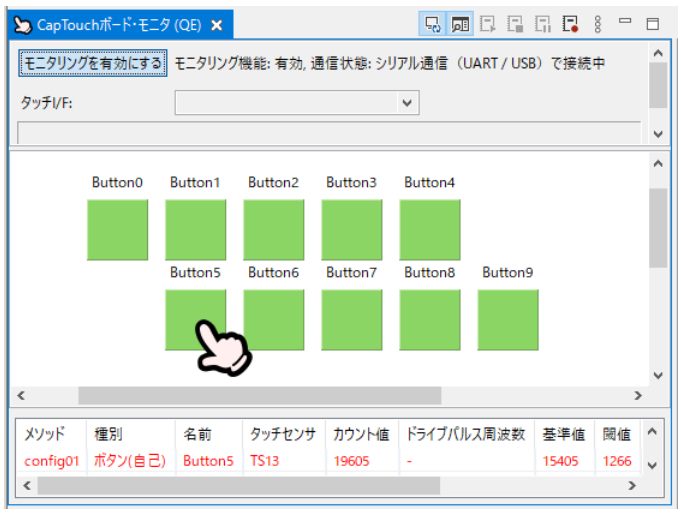
エミュレータをお持ちでない場合も、「シリアル接続」は実行できます。

ポート番号を、USB-ADAPTER-RX14 の接続ポートの番号を選択して「接続する」。



「モニタリングを有効にする」。

－自己容量タイプキーパッド(S16A)の場合－

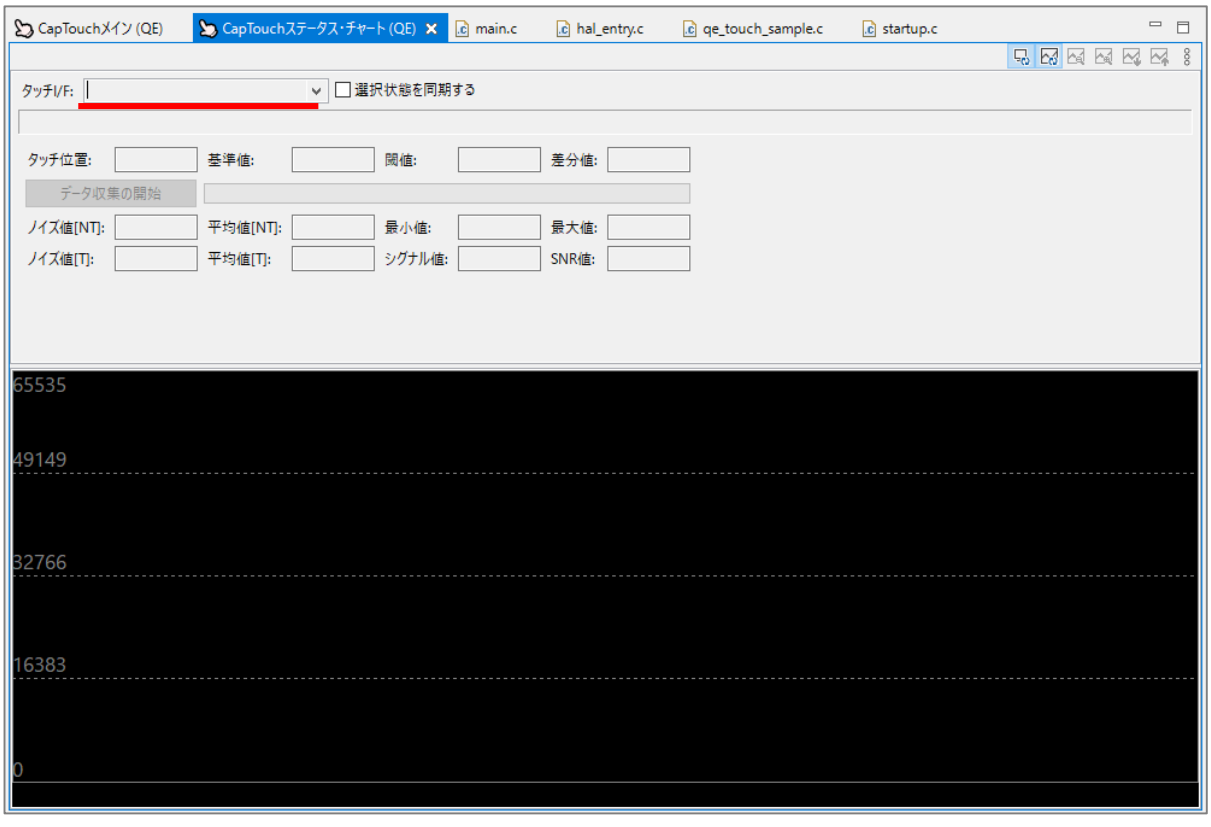


5 のタッチパッドにタッチした場合

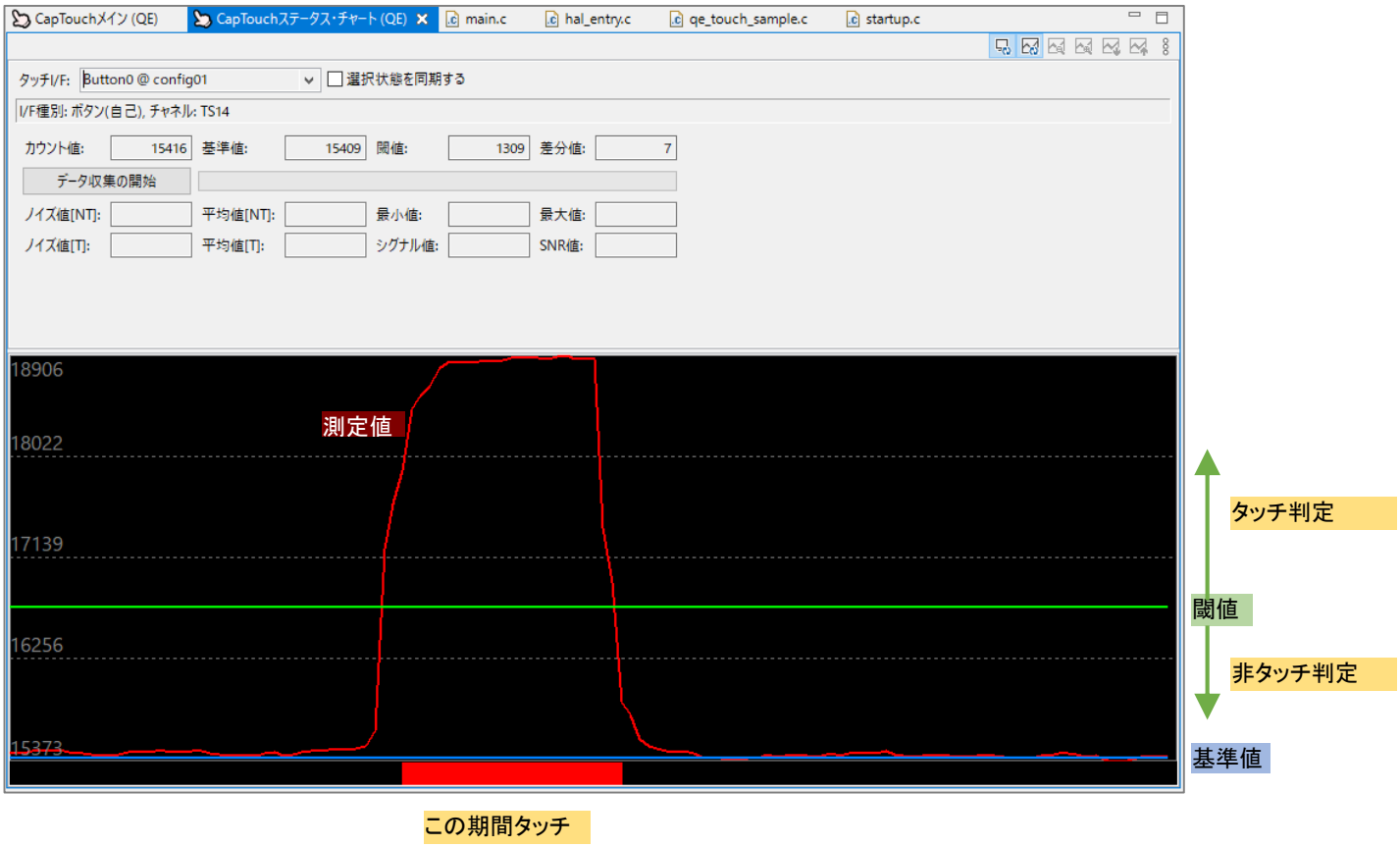
・指のアイコンが出現
・カウント値が基準値＋閾値を超えているとき、赤字になりタッチ判定される

となります。

(7)ビューを開く



CapTouch ステータスチャートが開きますので、タッチ I/F を観測したいキーパッド（ここでは、キーパッド 0;button0 とします）を選択。



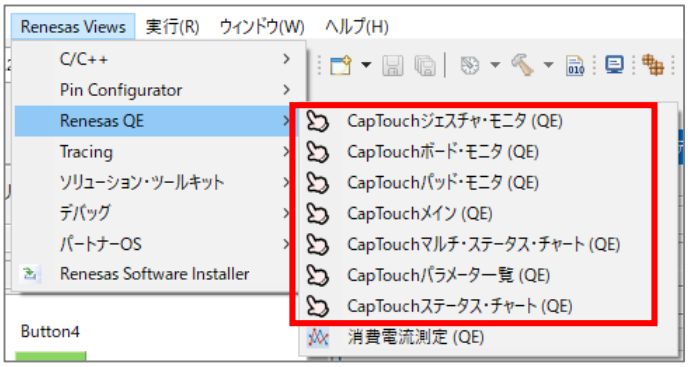
横軸時間、縦軸測定値

青線：基準値（非タッチ時の測定値）

赤線：測定値

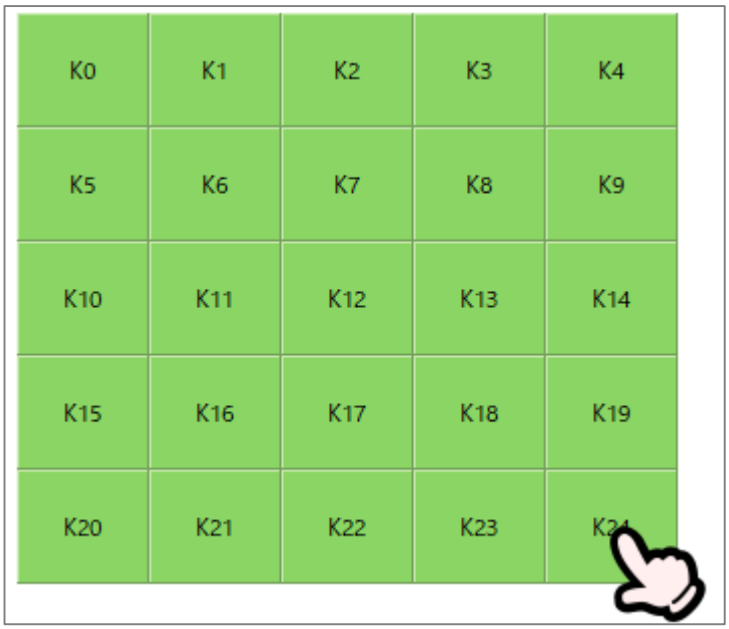
緑線：閾値

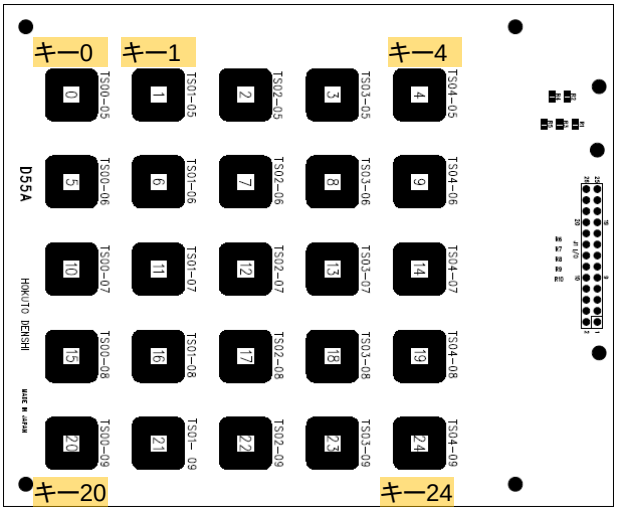
上記の他にも



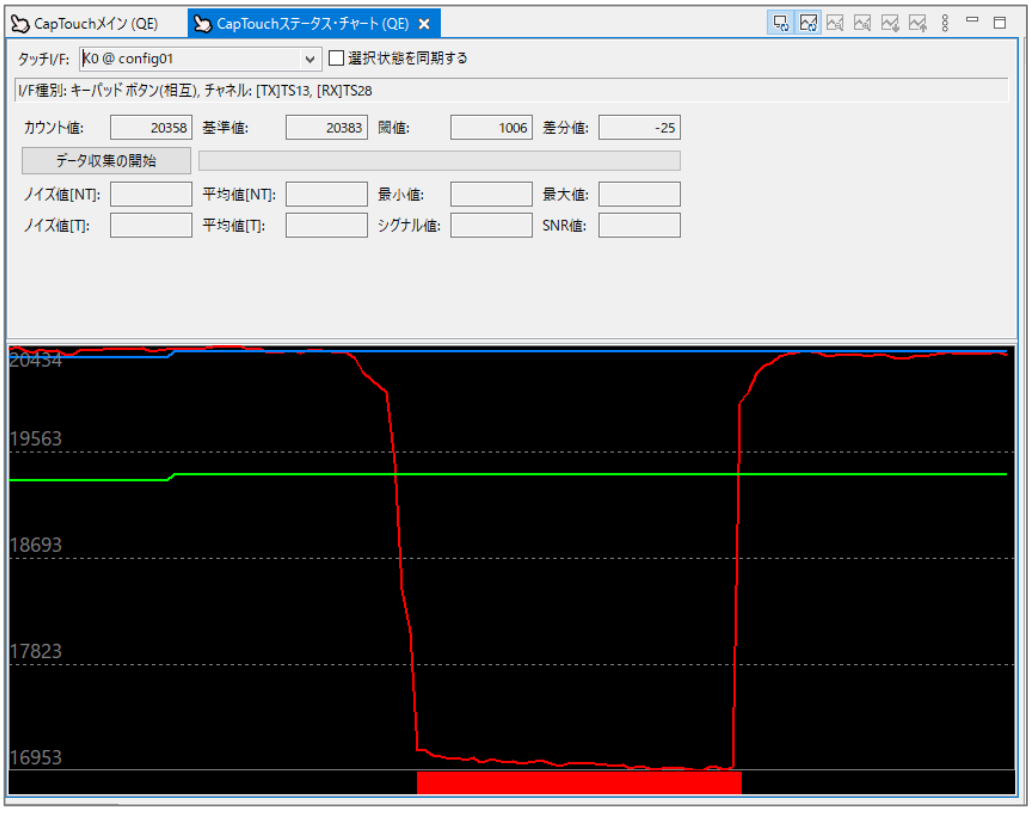
観測できる項目がありますので、見てみてください。

－相互容量タイプキーパッド(D55A)の場合－





D55A 基板を 90 度回転させた場合、CapTouch ボード・モニタの表示と対応します



相互容量タイプの場合、
自己容量とは逆にタッチしている期間数値が減少となります

この期間タッチ

QE CapTouch では、数値の変化がリアルタイム、グラフィカルに表示されますので、数値の変化が掴みやすいかと思います。

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	2022.2.9	—	初版発行

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:http://www.hokutodenshi.co.jp

ルネサス エレクトロニクス RA 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

RA2L1 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
 2022 年 2 月 9 日改訂 REV.1.0.0.0 (220209)