Renesas Synergy

Using QE and SSP to Develop Capacitive Touch Applications

Introduction

This document will demonstrate the necessary steps for creating an application example that integrates capacitive touch sensing using Renesas Synergy Microcontrollers.

Target Device

Renesas Synergy family MCUs with Capacitive Touch Sensing Unit (CTSU)

Contents

1. Application Example Overview

This document will demonstrate how to implement capacitive touch sensing functions using Renesas S3A MCUs based on the following methods:

- Create a project using the Synergy Smart Configurator with the S3A7 MCU board
- Create a cap touch interface with QE for Capacitive Touch [RA, RL78, Synergy] for tuning and monitoring

2. Related Documents

This application example is intended to give the user a short introduction to creating a working RA capacitive touch sensing project. A thorough review of all the applicable documentation for the e2 studio/Synergy Smart Configurator, Synergy Software Package (SSP) drivers/middleware, and QE for Capacitive Touch [RA, RL78, Synergy] plug-in help (contained within the e2 studio IDE help index) is strongly suggested to answer any questions or for more details on usage of any of the tools utilized in this application example.

3. High Level Integration Steps

The following high-level steps will give the reader an overview of the steps required to integrate touch detection into this project. These same steps should apply to any typical user development application.

1. Create a new project with the e^2 studio project creation wizard.
2. Use the Synergy Smart Configurator to add the required modules to the created e^2 studio project.
3. Use the QE for Capacitive Touch [RA, RL78, Synergy] e^2 studio plug-in to create the capacitive touch interface.
4. Use the QE for Capacitive Touch [RA, RL78, Synergy] e^2 studio plug-in to tune the application project.
5. Add the needed SSP module API calls to the user project to enable capacitive touch sensing operations in the application.
6. Monitor the application project using QE for Capacitive Touch [RA, RL78, Synergy] e^2 studio plug-in to demonstrate capacitive touch sensing detection.

4. Required Development Tools and Software Components

This project utilizes the following development environment:

Table 1. Development Environment

Development Tool Software Components	S3A7(CTSU)
Board kit	Application Example for Capacitive Touch (AE-CAP1) (YSAECAP1S11) ・AE-CAP1-S3 V1.1 ・Self-capacitance Button/Wheel/Slider Board (referred to as “BWS” in this document)
Renesas e^2 studio Integrated Development Environment (IDE)	2021-04 or later
GNU ARM Embedded compiler	9.2.1.20191025 or later
Renesas QE for Capacitive Touch [RA, RL78, Synergy]	1.4.0 or later
Synergy Software Package (SSP)	2.0.0 or later

5. Application Example Overview

In the main loop of the application example, the following processing is performed.

A code listing of the completed application example after modifications is provided in "7. qe_touch_sample.c Listing After Modifications" for review.

A call to g_qe_touch_instance_config01.p_api->scanStart()API

starts the measurement (scan process).

↓

The scanStart()API return value from g_qe_touch_instance_config01.p_api
is checked to determine if the SSP processing was completed successfully.

↓

If the middleware is ready, a call from the g_qe_touch_instance_config01.p_api
to dataGet()API determines if there is a touch on the target sensor board.

6. Capacitance Touch Application Development Procedure

6.1 Project Creation

1. On your PC, start the e² studio IDE using the **Windows -> Start** menu or the icon on your desktop. When the dialog appears, create the Workspace wherever you like.
2. Start a new project by clicking **File -> New -> Renesas Synergy** from the e² studio menu.
3. When the **New Synergy C/C++ Project** dialog box opens, select "**Renesas Synergy C Executable Project**", then click **Next**
4. In the **Project Configuration (Synergy C Executable Project)** dialog, enter a project name in **Project Name** (any name can be used).
   The example here uses **Capacitive_Touch_Project_Example**. When you have entered your project name, click **Next**.
5. Next, in the **Project Configuration (Synergy C Executable Project)** dialog, select the following:

Table 2. Device and Toolchain Selection

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

Note: To select the Device, press the ellipsis (...) and chose from the displayed list.

Select the board you will be using from: S3 > S3A7 > S3A7 – 144Pin.

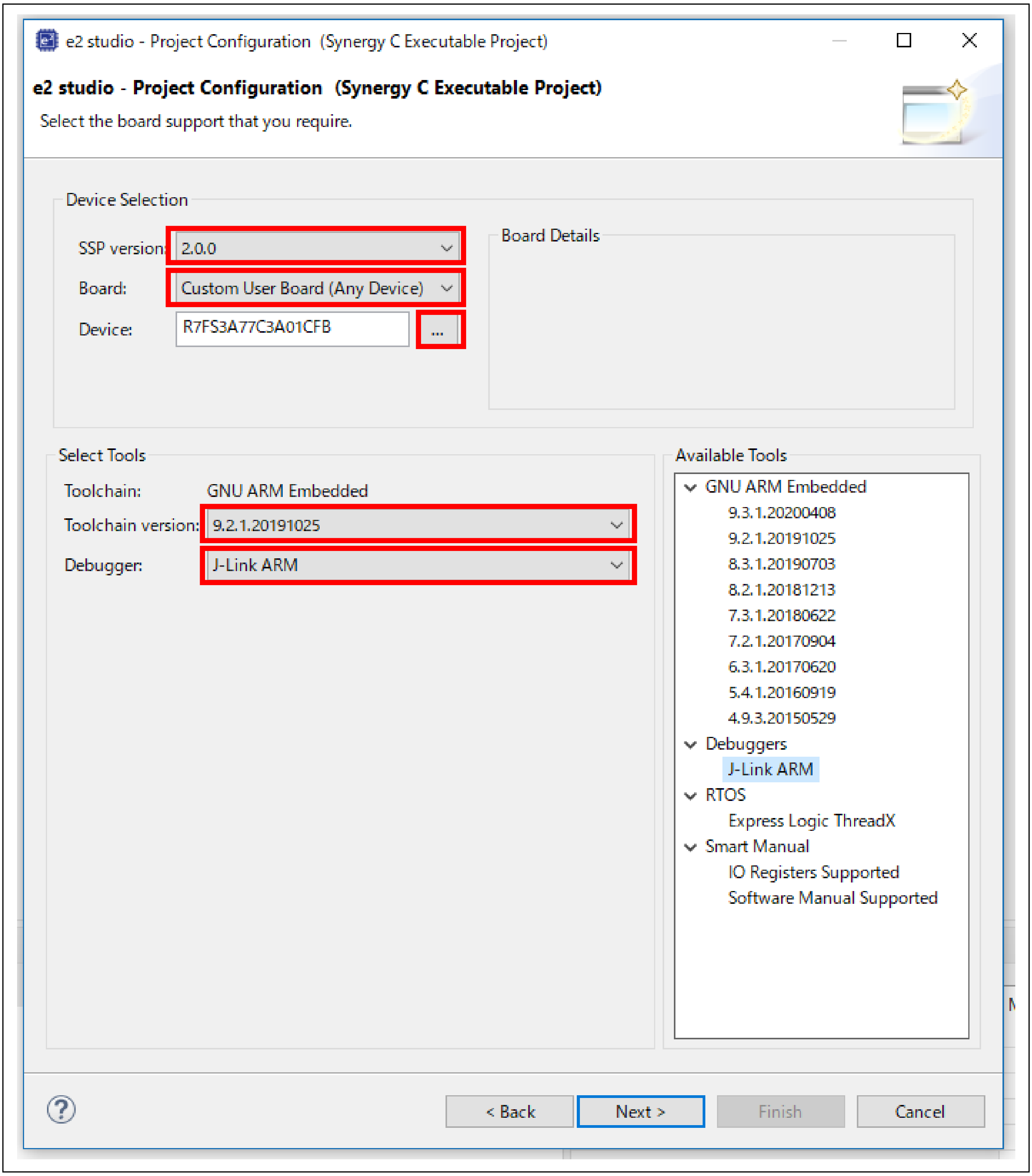

Figure 1. Device and Toolchain Selection

6. Once complete, click **Next.**
7. When the **e^2 studio – Project Configuration (Synergy C Executable Project)** dialog displays, check “BSP” and click **Finish**.

Once complete, the Synergy Smart Configurator perspective will appear in the e^2 studio default window, ready for project configuration. This completes the new project creation process.

6.2 Using Synergy Smart Configurator to Add Modules

1. Using the tabs in the lower-middle pane of the e^2 studio, select the **BSP** tab to display the Board Support Package (BSP) configuration.

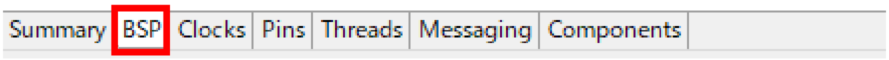

Figure 2. BSP Tab

2. To set the power supply, select the following from the lower-left of the e^2 studio screen: **Properties -> Settings -> Synergy Common -> MCU Vcc (mV)**. For this example, the MCU Vcc (mV) will be set to 3300mV.

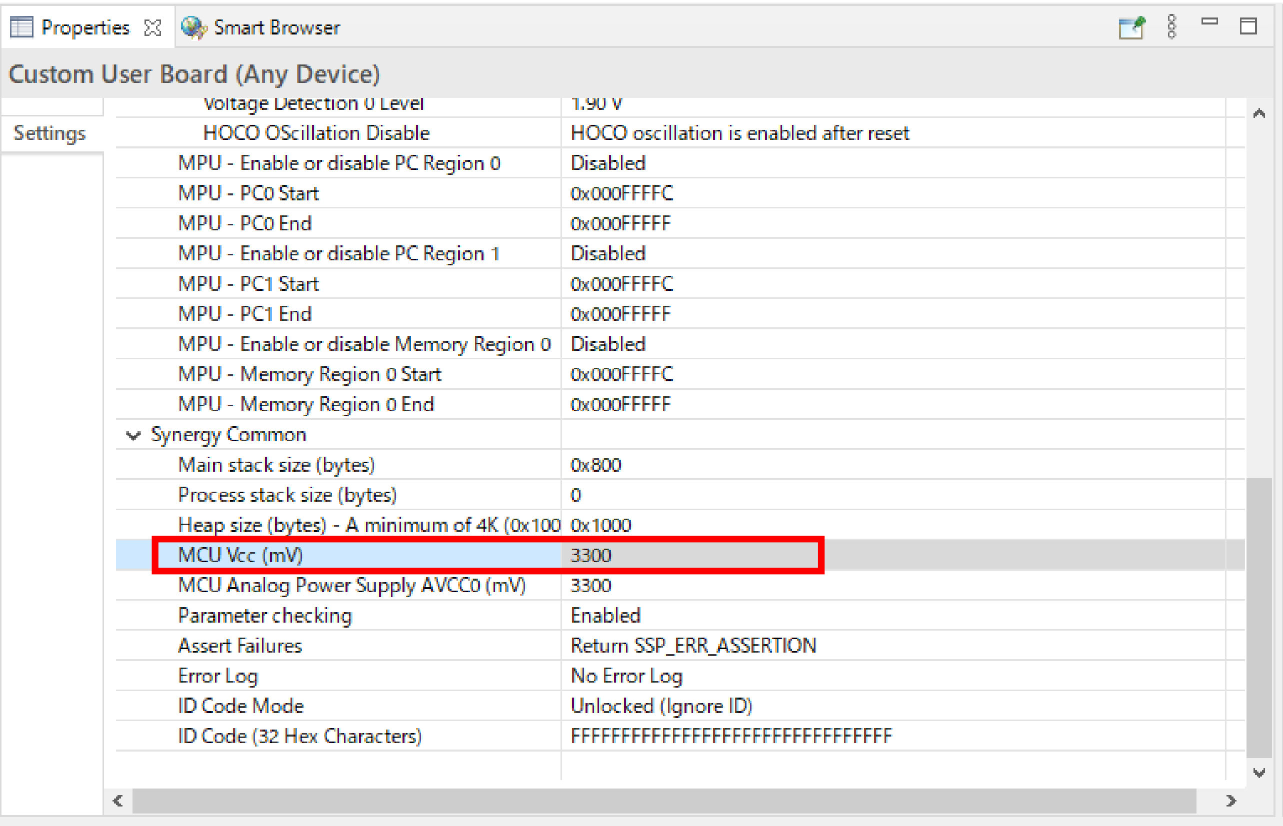

Figure 3. BSP Tab - Power Supply Voltage Setting

3. Select the **Clocks** tab from the lower-middle middle pane of e² studio to display the clock settings.

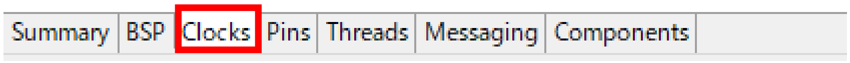

Figure 4. Clocks Tab

4. For this example, the following settings are used.

Table 3. Clocks Configuration

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

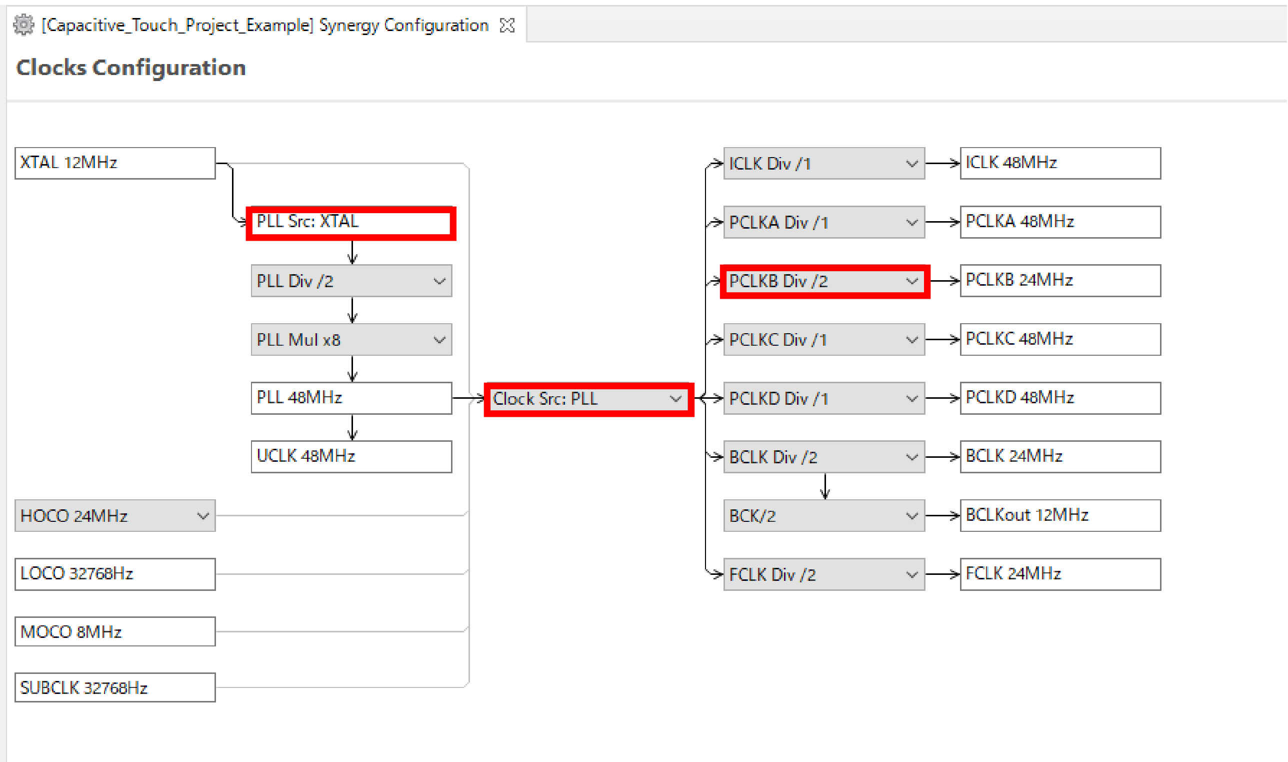

Figure 5. Clocks Tab – Clocks Configuration

5. Next, select the **Pins** tab. Assign the pins to connect the sensor port of the MCU to the BWS Board.

   Note: Depending on the selections made in Device Selection -> Board when creating the project, the sensor port may be assigned to the TS pin by default.

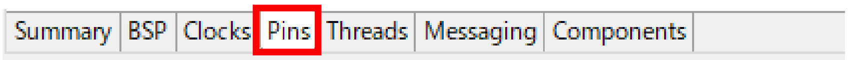

Figure 6. Pins Tab

6. Under **Pin Selection**, expand **Peripherals**. Open **Input:CTSU** and select **CTSU0**.

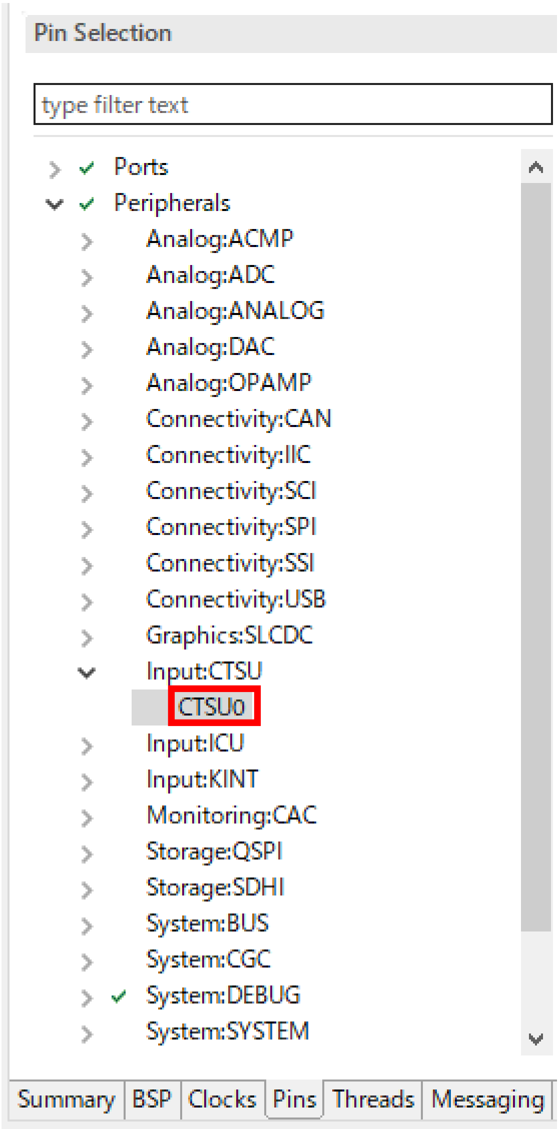

Figure 7. Pins Tab - Select Peripherals

7. In **Pin Configuration**, change the **Operation Mode** from “Disabled” to “Enabled”.

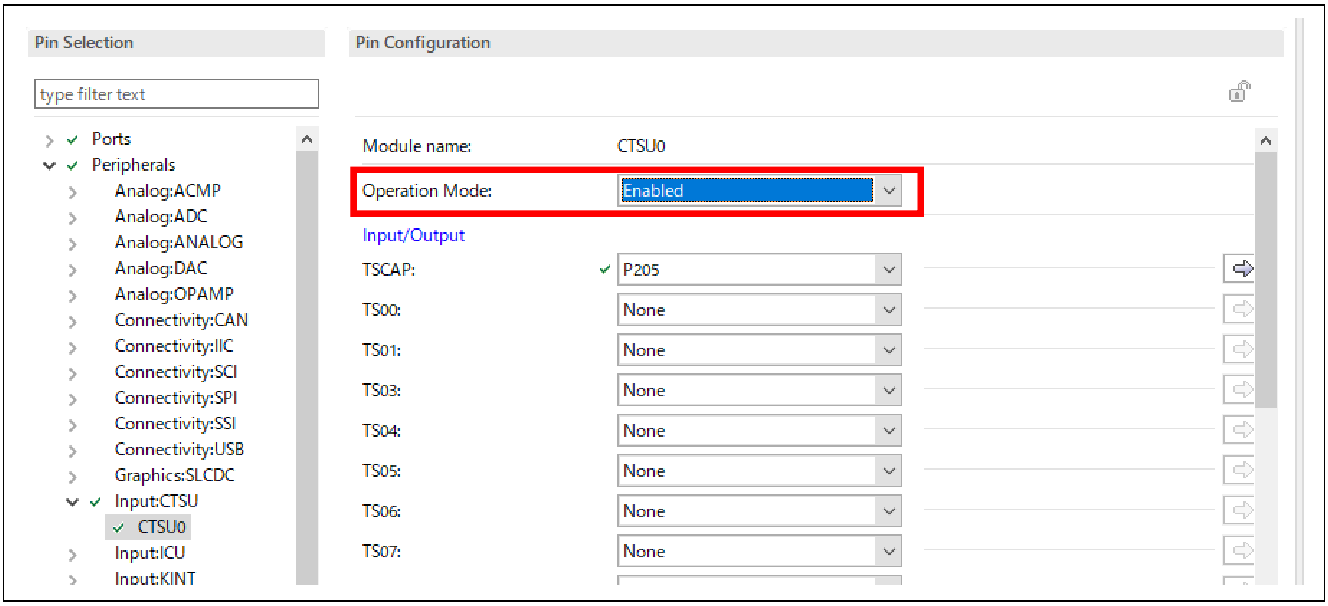

Figure 8. Pins Tab - Operation Mode

8. In this application example, only one button is created. In this case, the setting for the TS pins listed below, which will be used for cap touch interface, must be changed from “None”, and the port set from the pull-down menu.
   - TSCAP pin
   - TS31pin

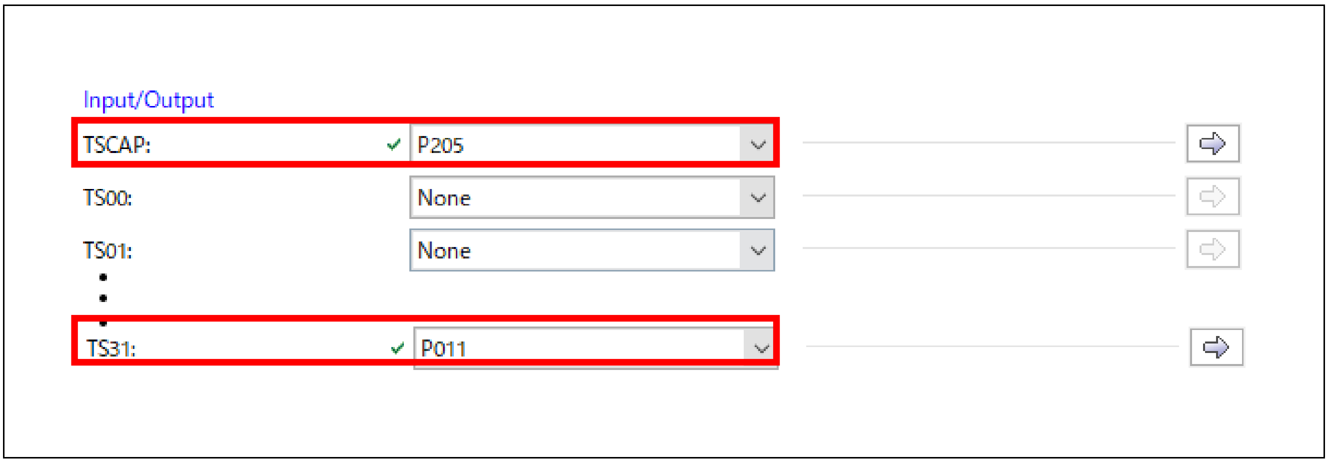

Figure 9. TS Pin Configurations

9. Click **Pin Selection** --> **Peripherals** --> **Connectivity:SCI**, and select **SCI9**.

   NOTE: Steps 9 and 10 should only be set when using serial communication for monitoring. Otherwise, skip to Step 11.

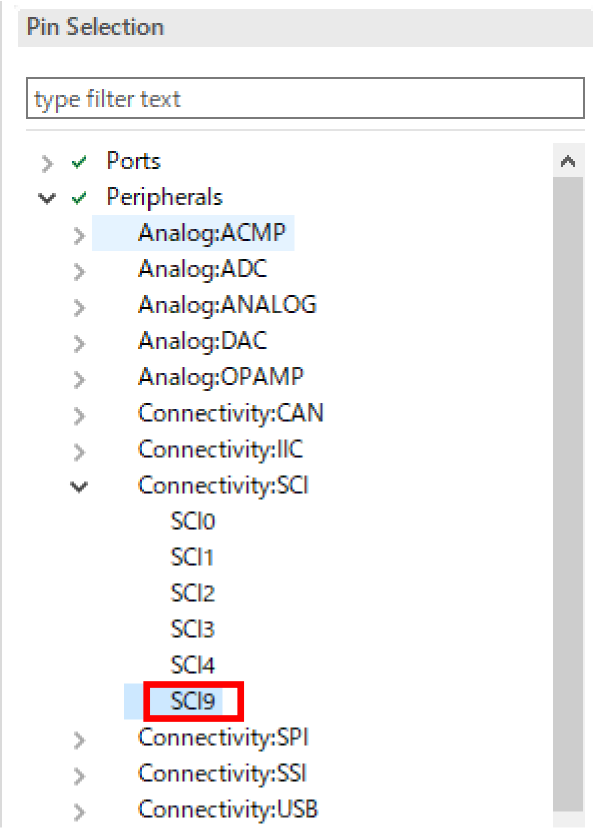

Figure 10. Pins Tab – Select Peripheral SCI9

10. Select **Pin Configuration -> Operation Mode**, then change “Disabled” to “Asynchronous UART”. Also make sure **TXD** is set to “P203” and **RXD** is set to “P202”.

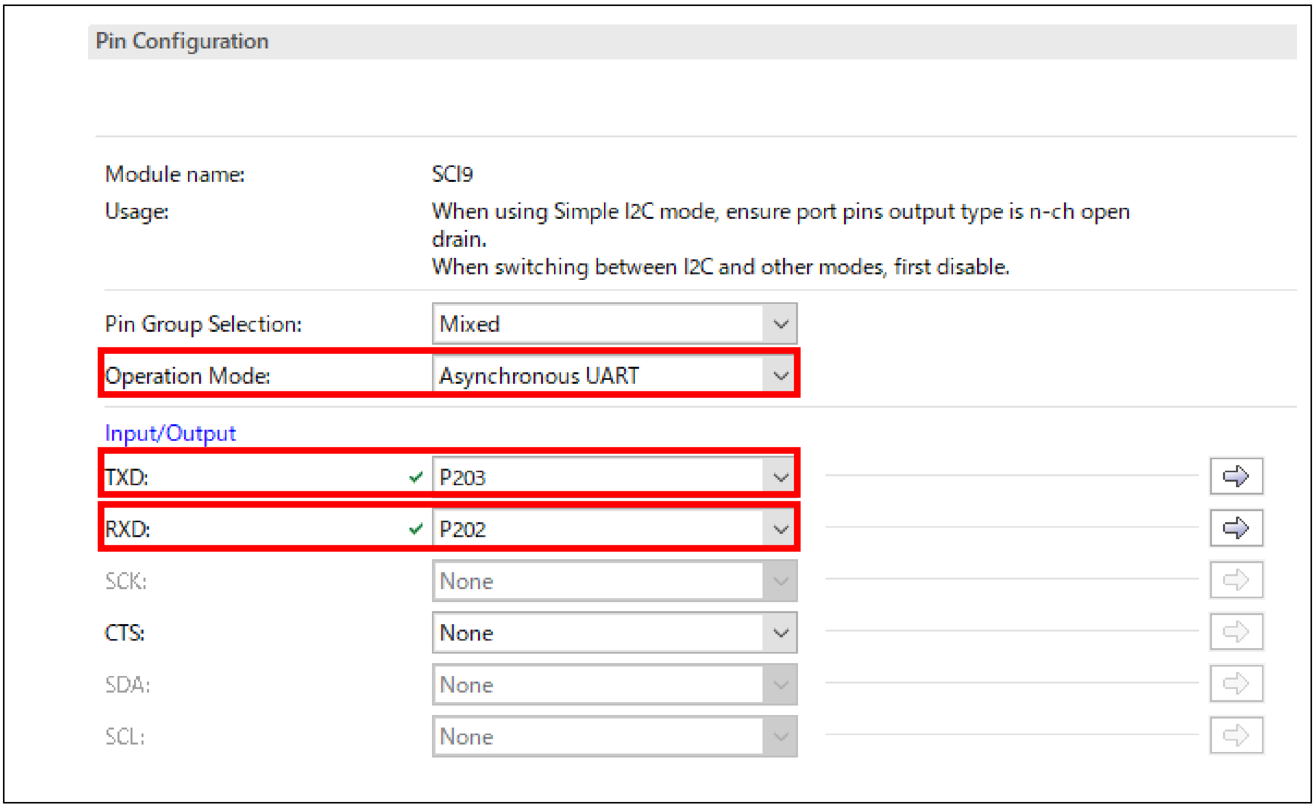

Figure 11. Pins Tab - Operation Mode

11. Next, move on to the **Threads** tab.

Figure 12. Threads Tab

12. Select **New Stack**, then **Framework -> input -> Cap Touch Framework on sf_touch_ctsuv2**. This will add the CTSU driver and DTC middleware

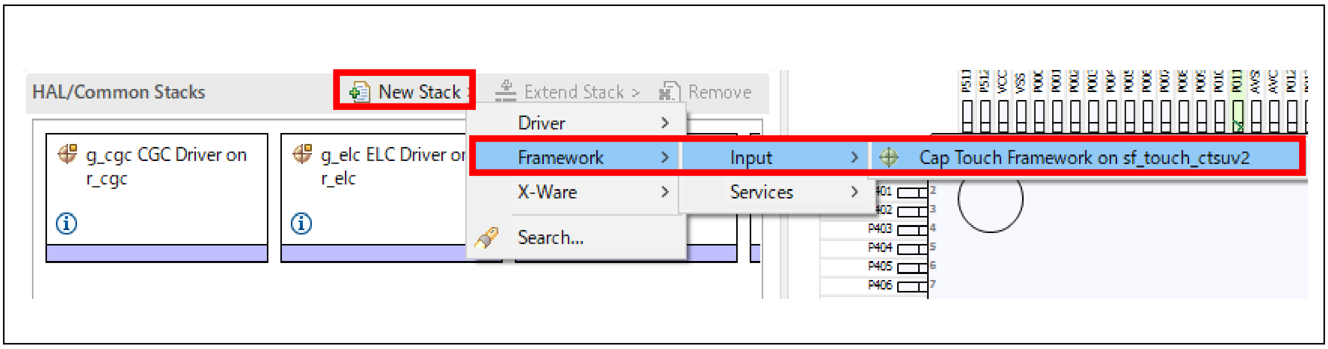

Figure 13. Threads Tab - Add CTSU Driver and Middleware

13. **HAL/Common Stacks** will appear as shown in the figure below. The selected Stacks will be highlighted in gray.

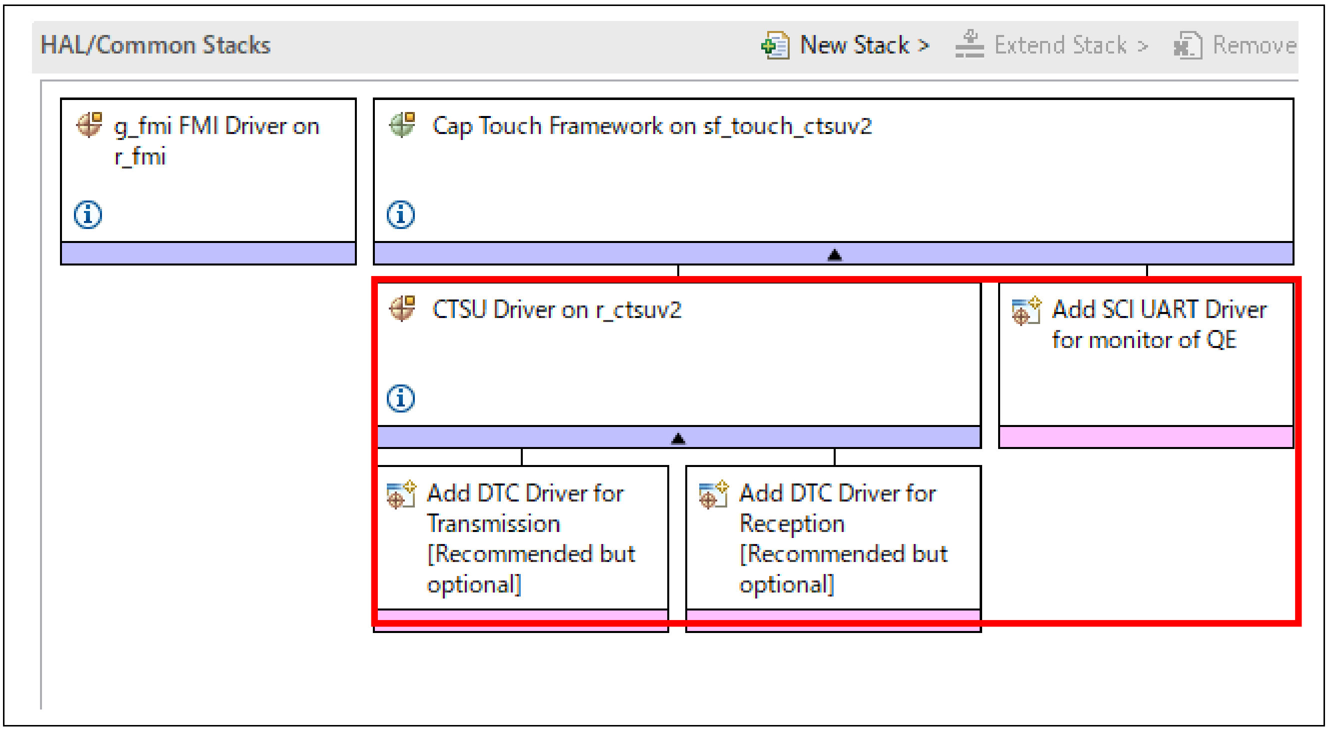

Figure 14. HAL/Common Stacks (after CTSU is added)

14. Next, click the **CTSU Driver on r_ctsu2** module to display **Properties**.

NOTE: Steps 14 to 17 should only be set when using the Data Transfer Controller (DTC). Use the DTC to transfer data between memory and registers without going through the CPU. When not using the DTC, skip to Step 18.

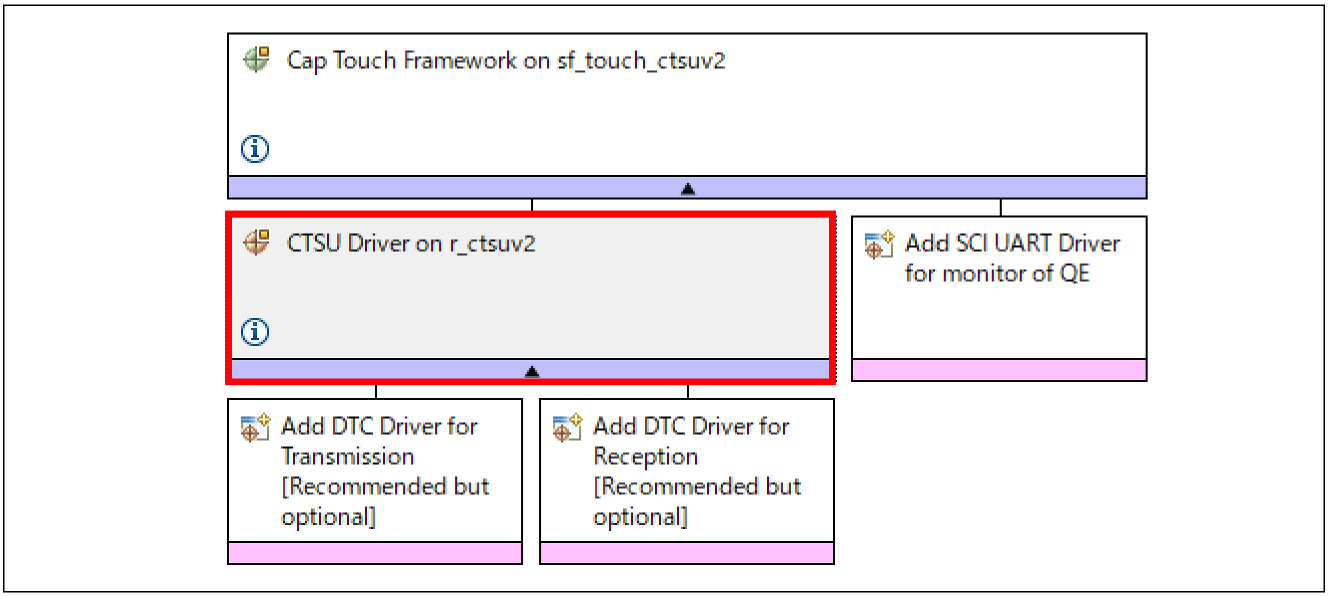

Figure 15. CTSU Driver on r_ctsu2 Module

15. Set the DTC from **Properties** at the bottom-left of the screen. Change the **Support for using DTC** setting from “Disabled” to “Enabled”.

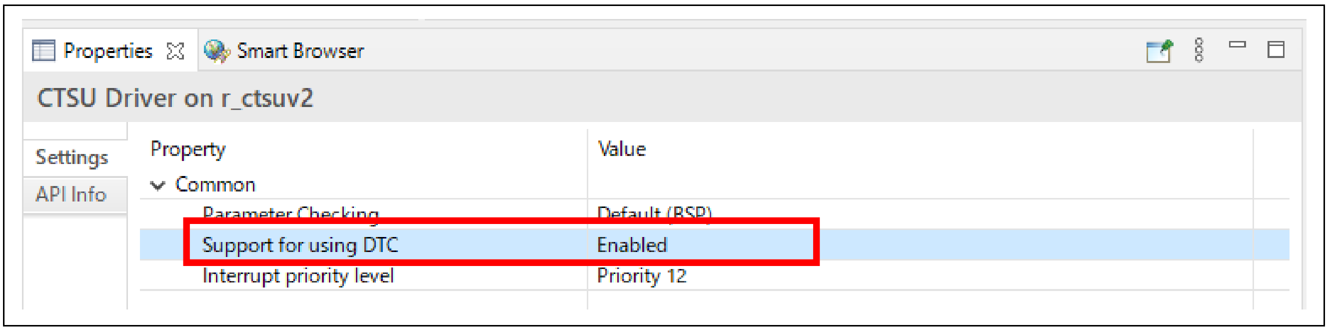

Figure 16. Support for using DTC

16. Click the **Add DTC Driver for Transmission** module and select the **New -> Transfer Driver on r_dtc** to add the DTC driver for transmission.

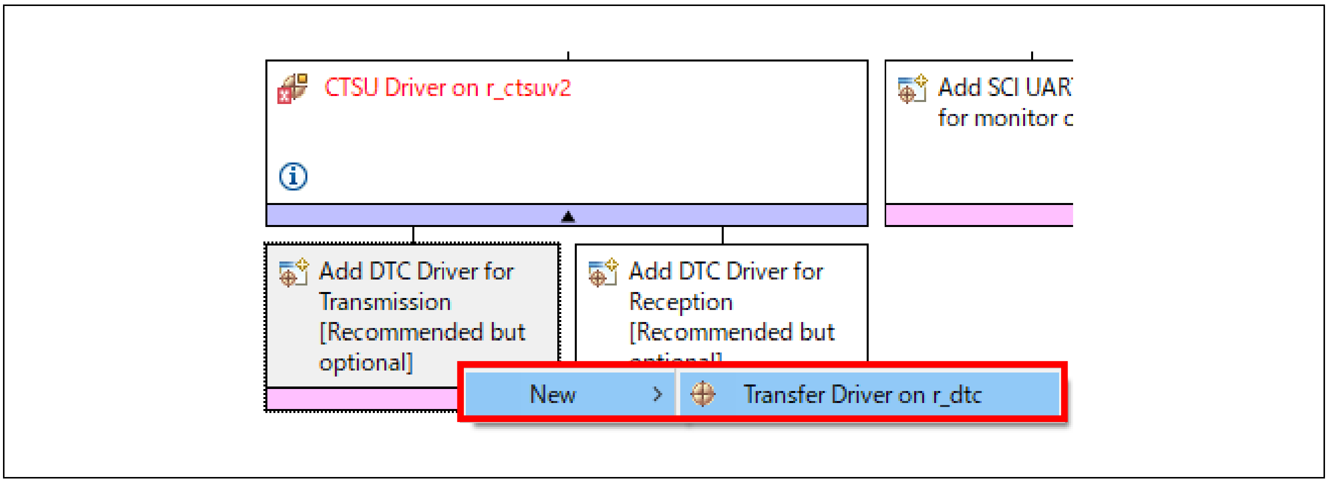

Figure 17. Add DTC Driver for Transmission (CTSU)

17. In the same manner, click the **Add DTC Driver for Reception** module and select the **New -> Transfer Driver on r_dtc** to add the DTC driver for reception.

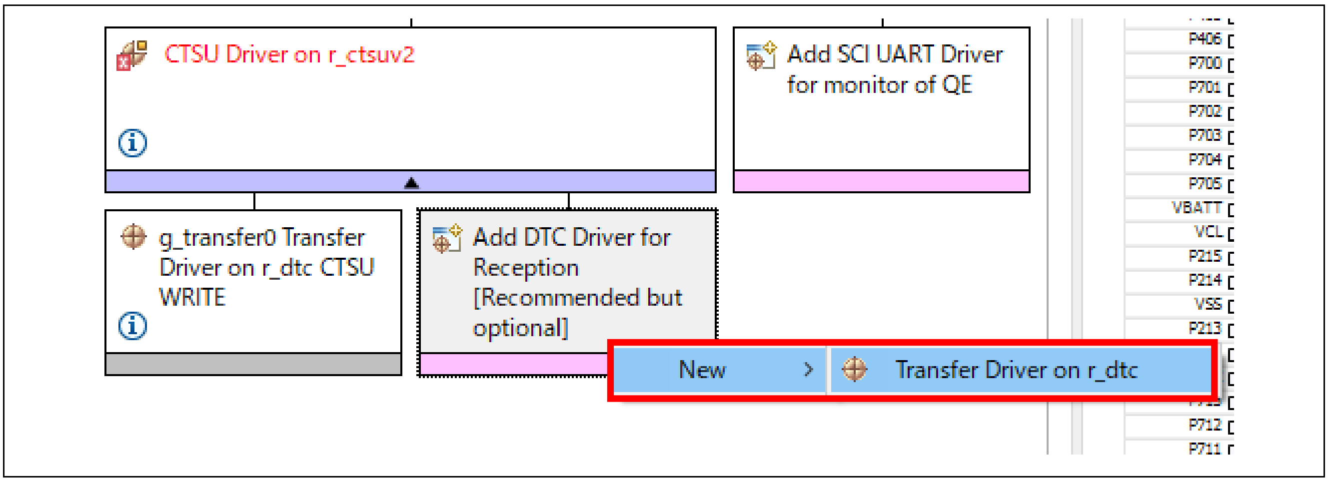

Figure 18. Add DTC Driver for Reception (CTSU)

18. Select **Cap Touch Framework on sf_touch_ctsuv2** to display **Properties**.

    NOTE: Steps 18 to 23 should only be set when using serial communication for monitoring. Otherwise, skip to Step 24.

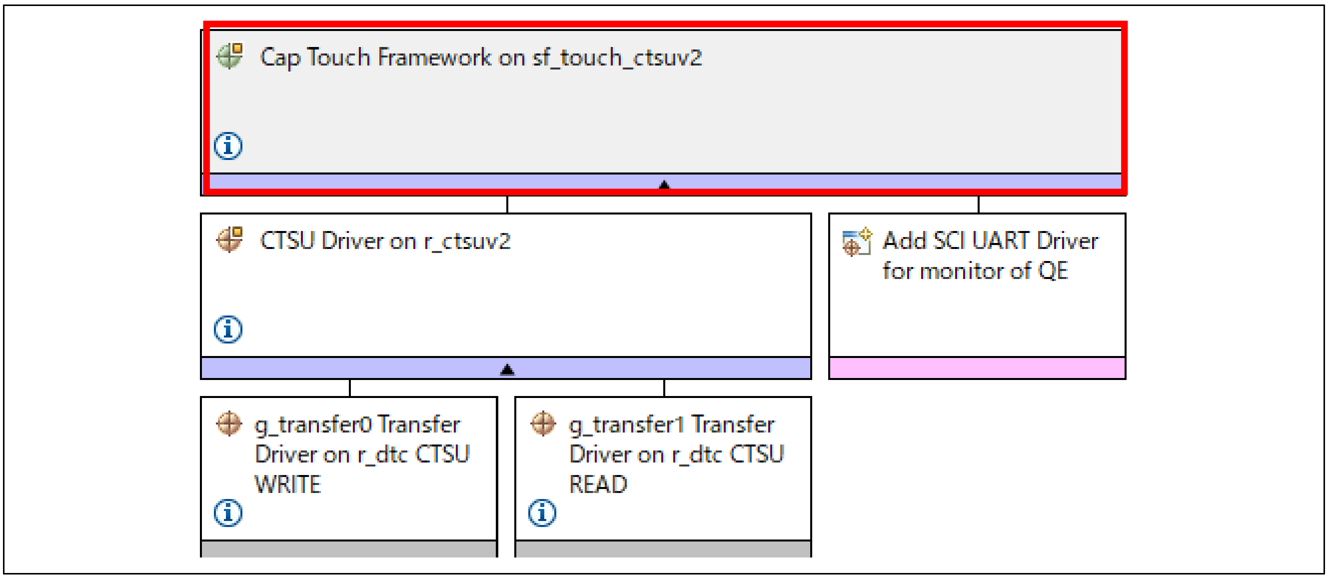

Figure 19. Cap Touch Framework on sf_touch_ctsuv2 Module

19. At the bottom left of the screen, select **Properties -> Settings -> Common -> Support for QE monitoring using UART** and change "Disabled" to "Enabled".

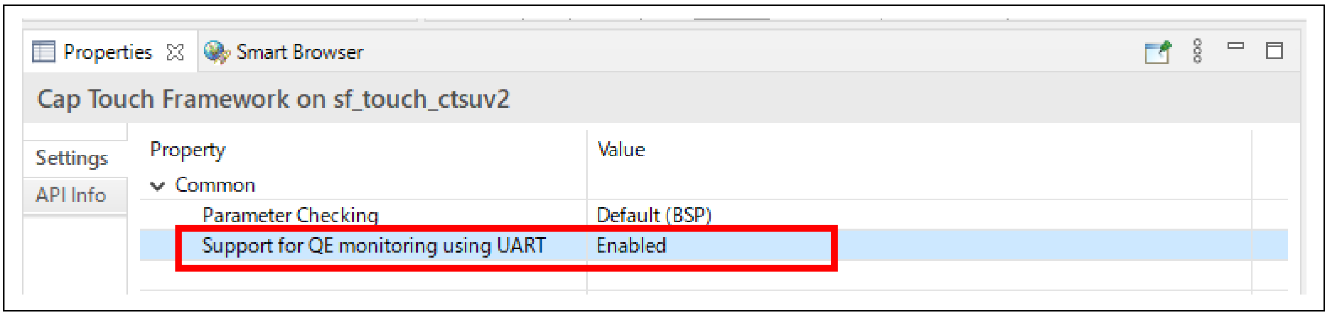

Figure 20. Support for QE monitoring using UART

20. Click **Add SCI UART Driver monitor of QE** and select **New -> UART Driver on r_sci_uart** to add the UART driver.

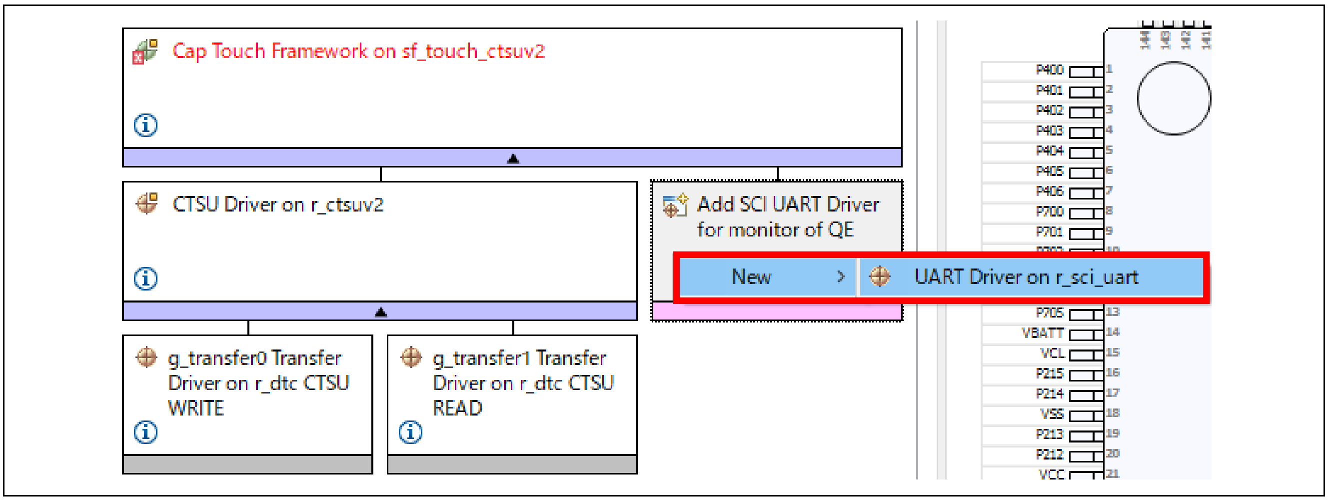

Figure 21. Add UART Driver

21. Click **g_uart_qe UART Driver on r_sci_uart** to display **Properties**.

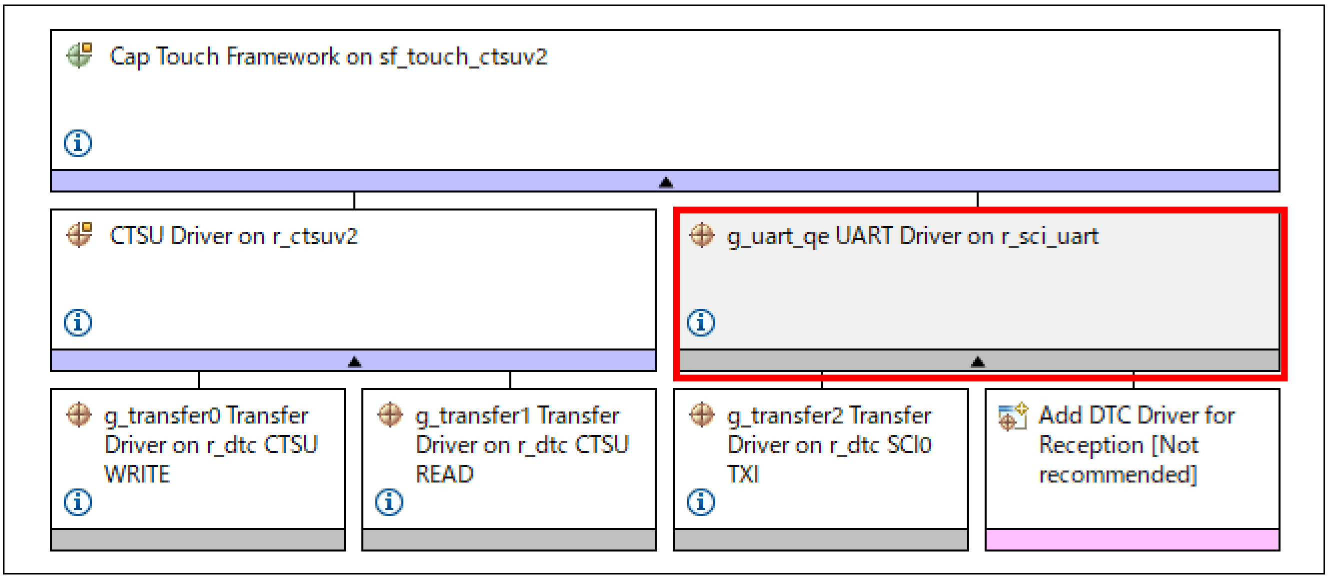

Figure 22 g_uart_qe UART Driver on r_sci_uart Module

22. Set the SCI channel. At the bottom-left of the screen, select **Properties -> Settings -> Module g_uart_qe UART Driver on r_sci_uart] -> Channel** and change the channel setting from “0” to “9”.

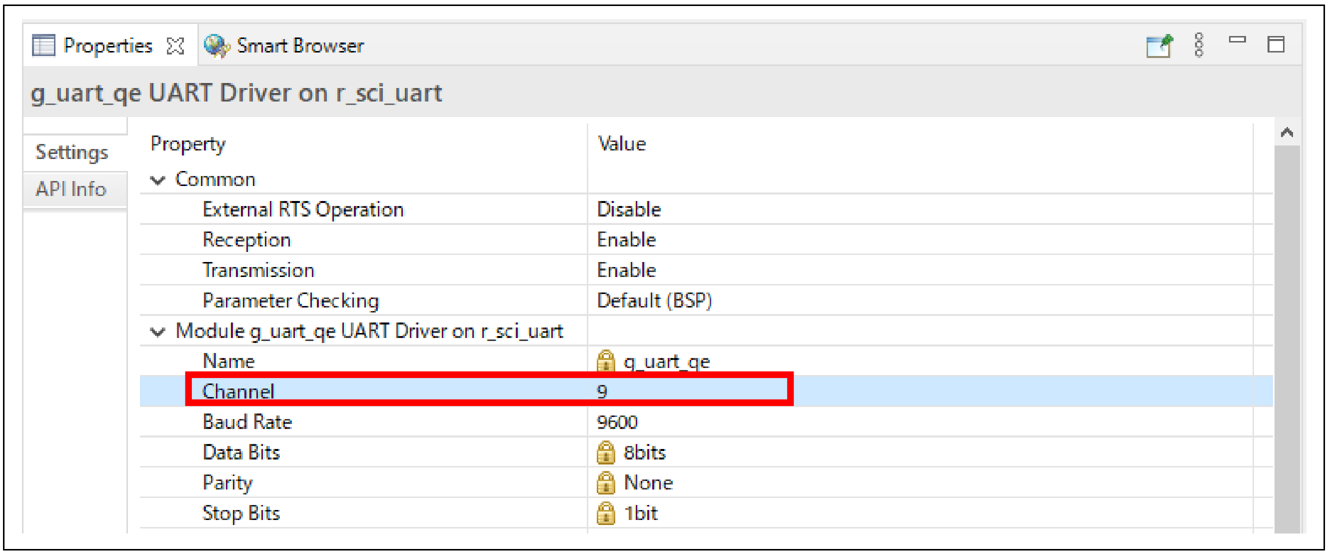

Figure 23. Channel

23. Set the DTC. Click **Add DTC Driver for Reception** and select **New -> Transfer Driver on r_dtc** to add the DTC driver for reception.

NOTE: Only carry out this step when using the Data Transfer Controller (DTC). Use the DTC to transfer data between memory and registers without going through the CPU.

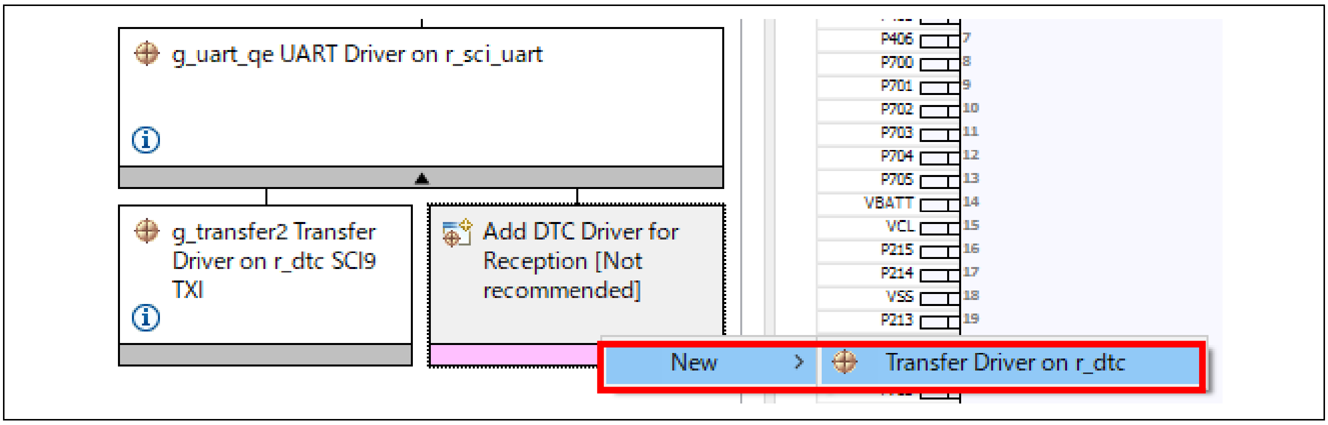

Figure 24. Add DTC Driver for Reception (UART)

24. At this point, all application modules necessary for capacitive touch operations have been added. The final step is to generate the application code modules necessary for the project. Click **Generate Project Content** icon at the upper-right of the screen.

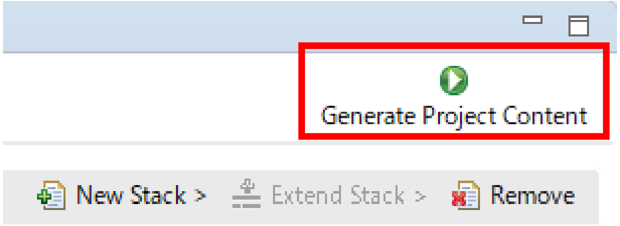

Figure 25. Generate Project Content Button

6.3 Creating the Capacitive Touch Interface

1. From the e² studio menu, use **Renesas Views -> Renesas QE -> CapTouch Main / Sensor Tuner RA, RL78, Synergy (QE)** to open the main perspective for configuring capacitive touch to the project.

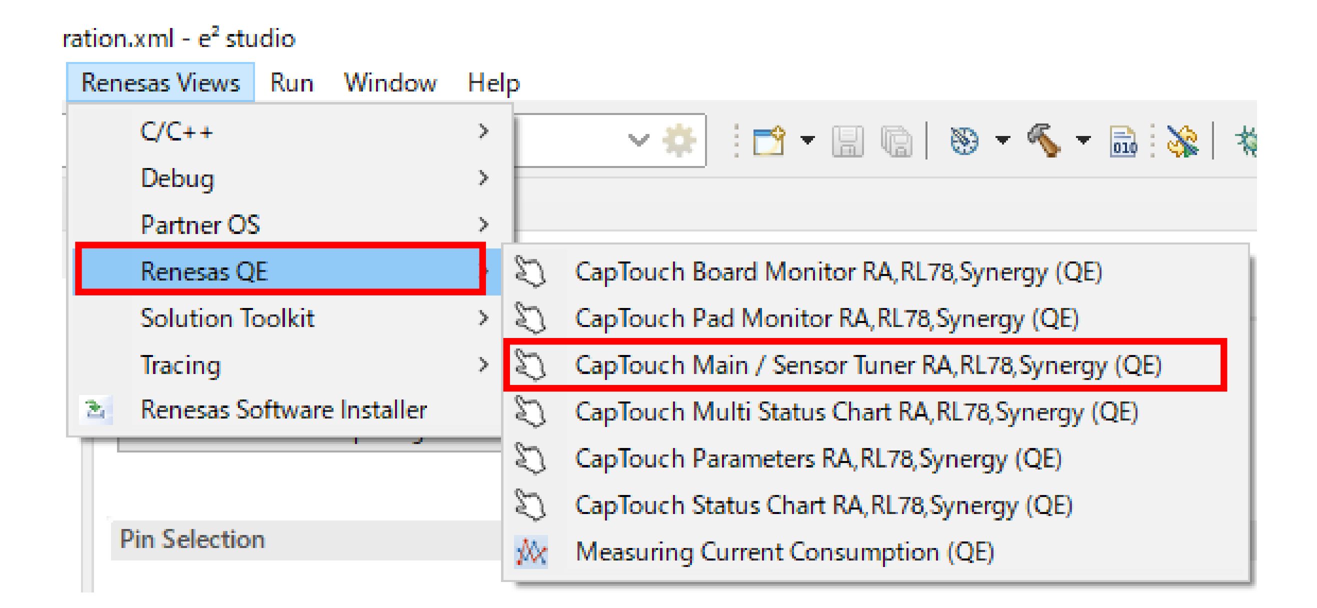

Figure 26. CapTouch Main / Sensor Tuner RA, RL78, Synergy (QE) Menu

2. In the **CapTouch Main / Sensor Tuner RA, RL78, Synergy (QE)** pane, select the project you want to configure for touch interface by using the pull-down tab under **To Select Project** and selecting "Capacitive_Touch_Project_Example" as shown below.

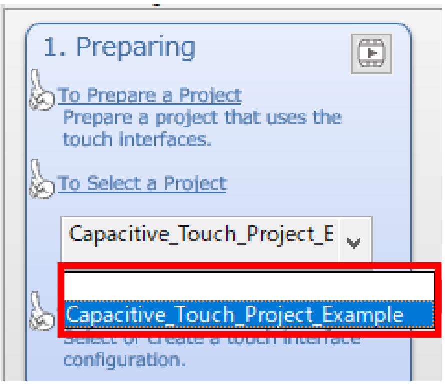

Figure 27 Select a Project

3. Next, create a new touch interface configuration by using the **To Prepare a Configuration** pull-down menu and selecting **Create a new configuration**.

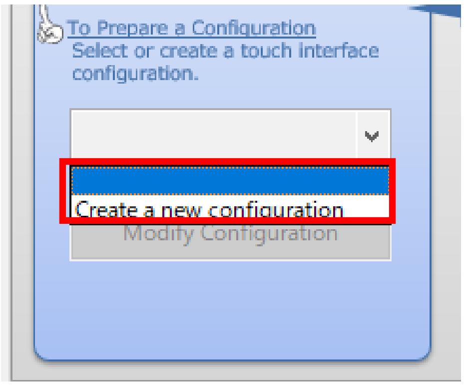

Figure 28. Create Configuration of Touch Interfaces Dialog Box

4. The **Create Configuration of Touch Interfaces** dialog box will open, showing the area for positioning the touch interface (default blank canvas).

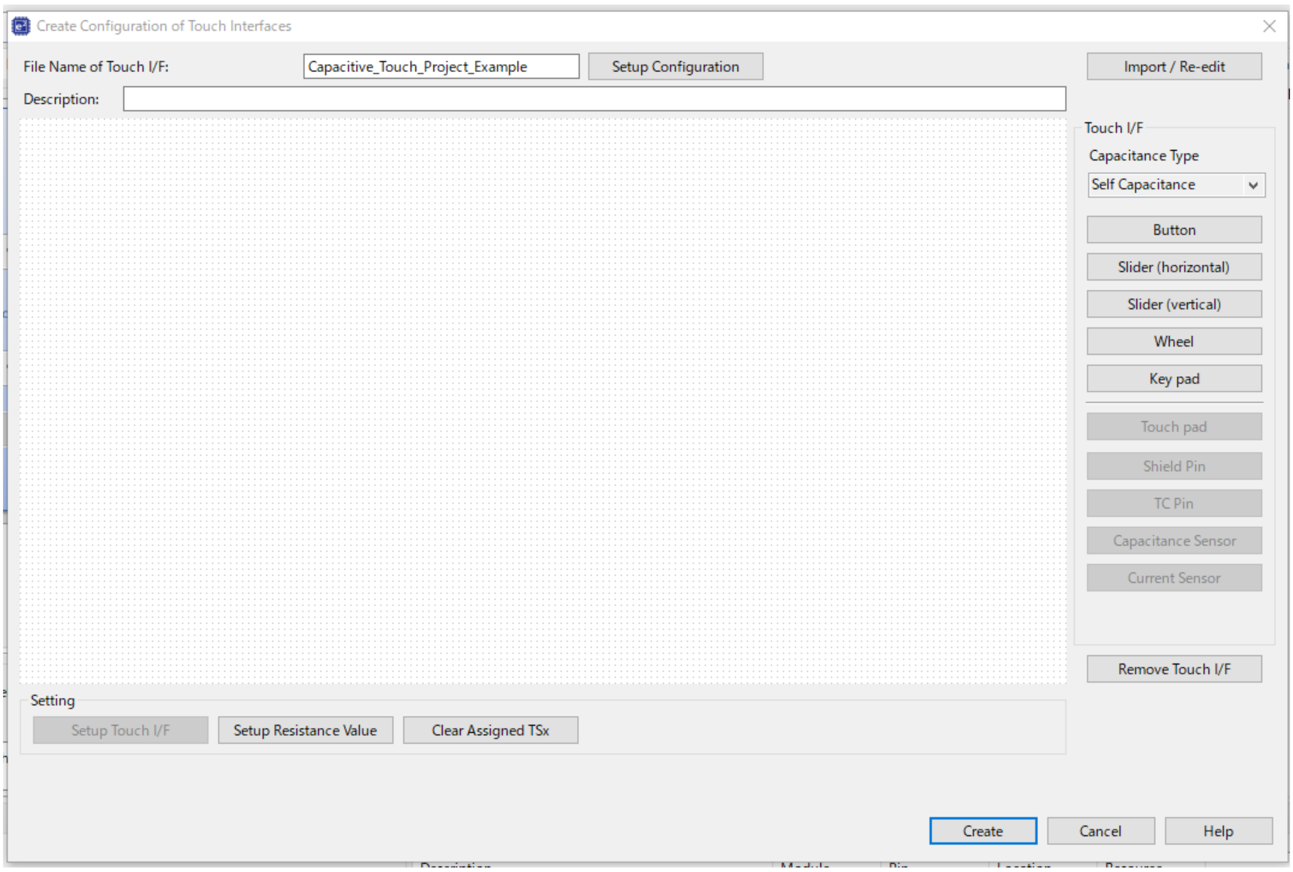

Figure 29. Create Configuration of Touch Interfaces Dialog Box

5. Add a button to the touch interface area by selecting **Button** on the right side of the **Create Configuration of Touch Interfaces** dialog box, and then clicking on the blank canvas.

   Press the ESC key on your keyboard or select **Button** again to complete the step. The added button will remain red (indicating setting error), as shown in the figure below, as it has not been assigned a touch sensor yet.

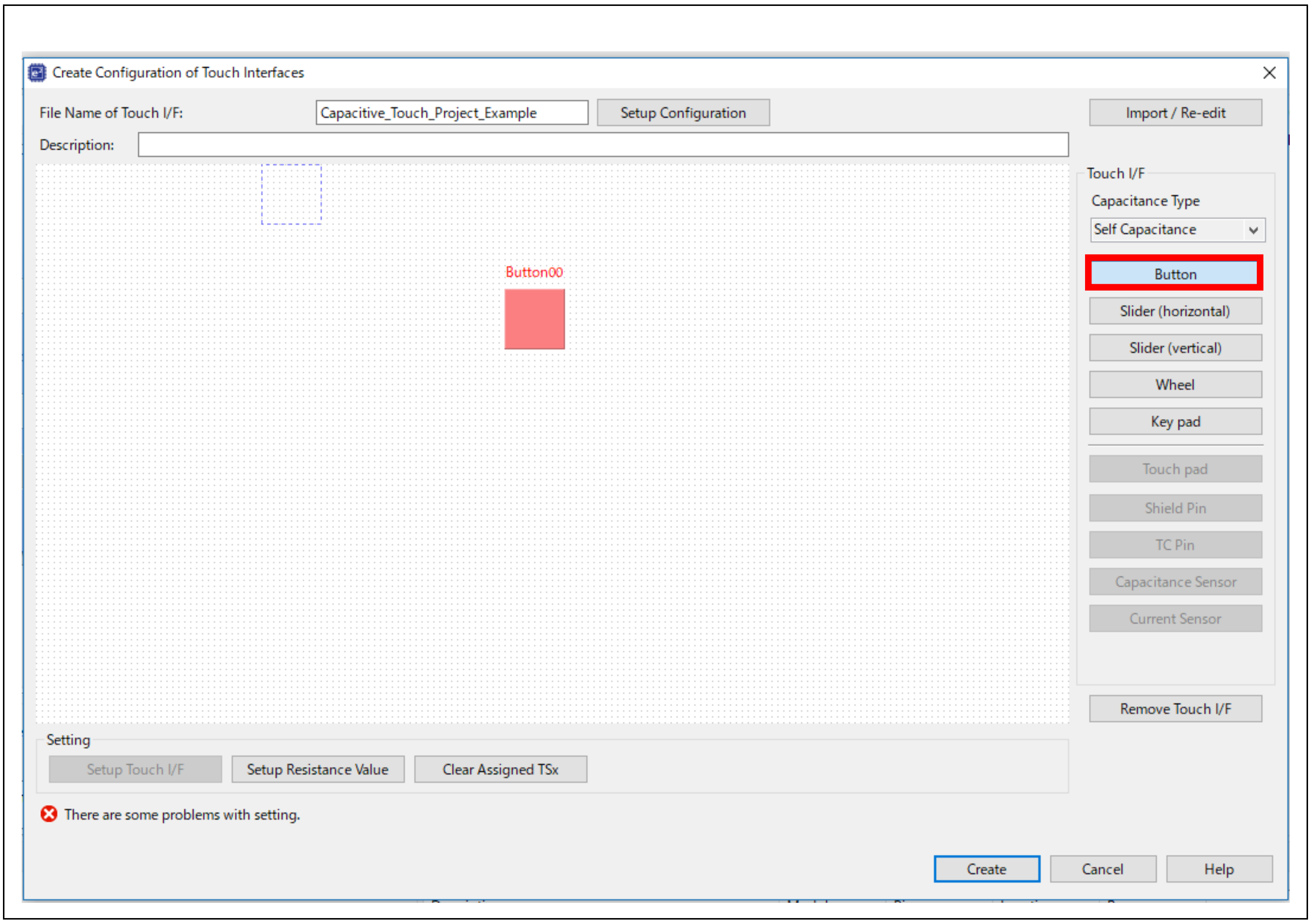

Figure 30. Add a Button

6. Double click “Button00” in the touch interface canvas to display the **Setup Touch Interface** dialog box. From the pull-down menu, select MCU sensor port TS31 to assign to this button.

   When a sensor port that has been enabled by the Synergy Smart Configurator is correctly assign to the button, the setting error will disappear and the button will turn green, indicating it has been set.

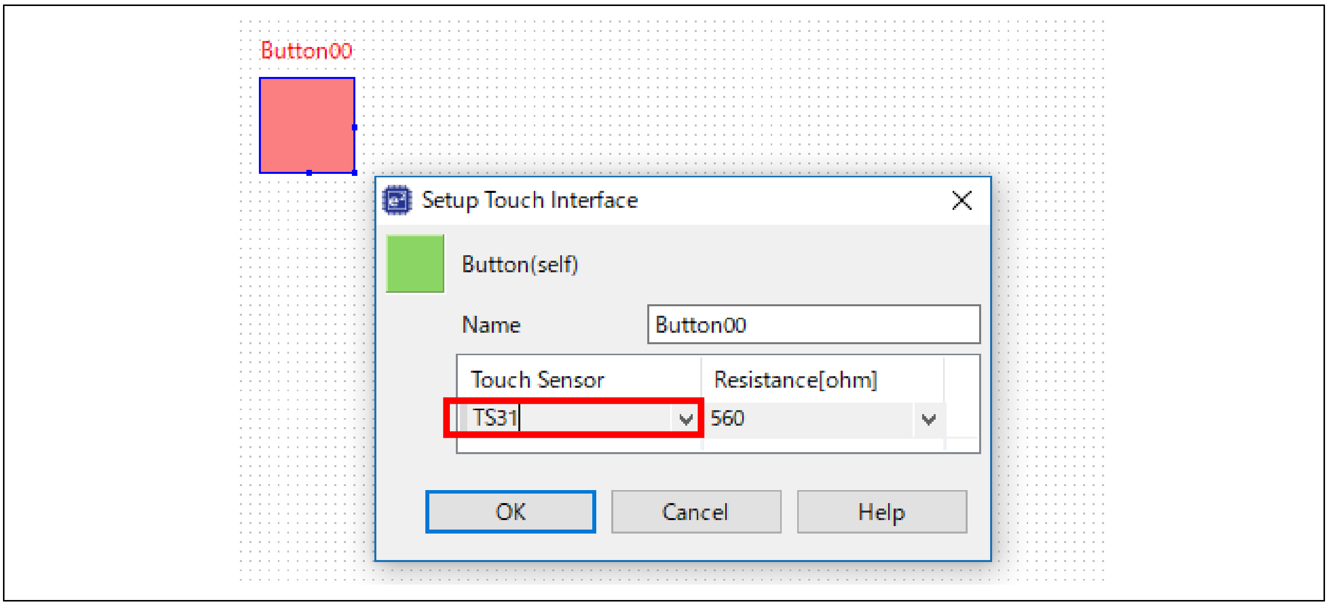

Figure 31. Setup Touch Interface (button)

7. Click the **Create** button in the **Create Configuration of Touch Interfaces dialog box** to complete the touch interface settings.

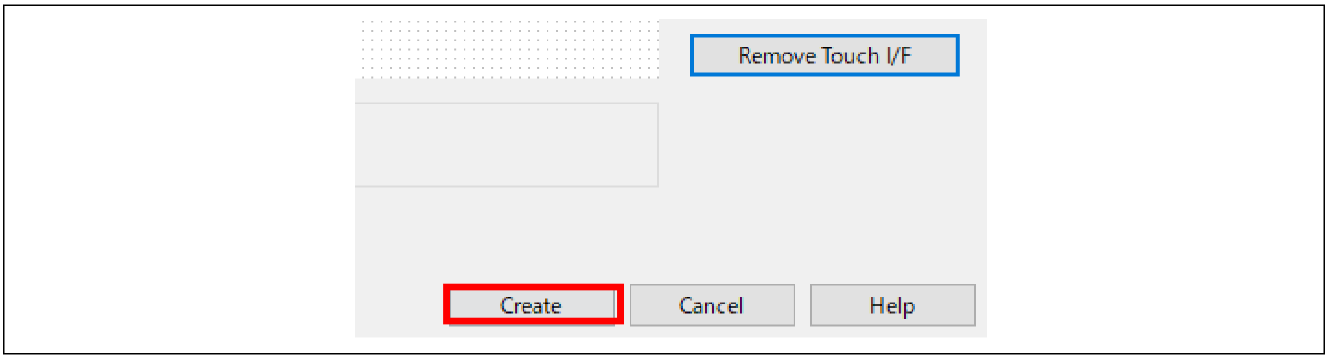

Figure 32. Create Button

8. The **CapTouch Main / Sensor Tuner RA, RL78, Synergy (QE)** window will now display the configuration of the touch interface in the **Tuning** pane.

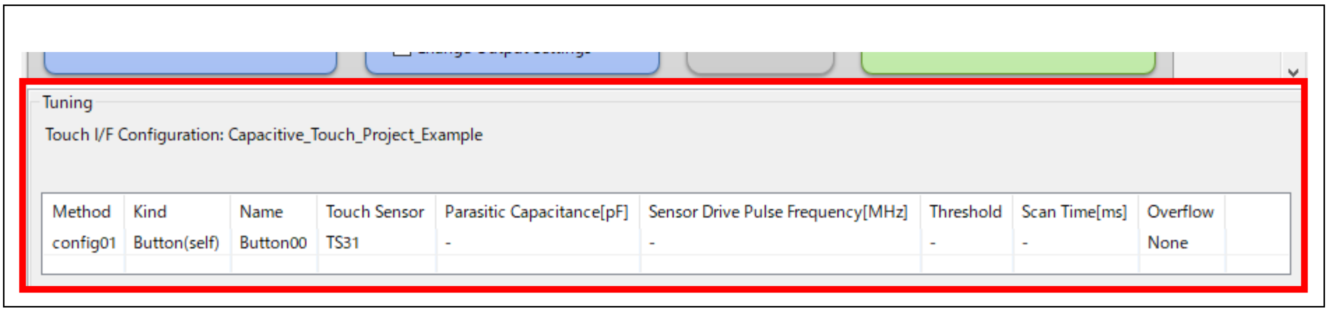

Figure 33. Tuning Pane

9. Build the project using the hammer icon in the upper left-hand side of the e^2 studio screen. The project should build without any errors or warnings.

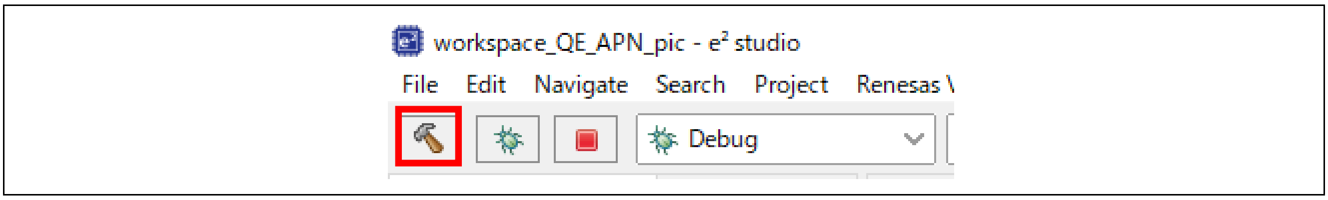

Figure 34. Build Button

6.4 Modifying Debug Session for Capacitive Touch Tuning

1. The debug configuration needs to be modified slightly so that a special tuning kernel can be downloaded into the MCU RAM after the debug session starts. Enter the Debug Configuration by clicking the **Gear** icon in the upper right-hand side of the Workspace

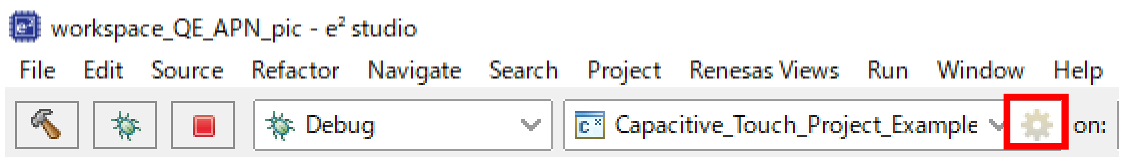

Figure 35. Debug Configuration Editing Button

2. Select the **Startup** tab.

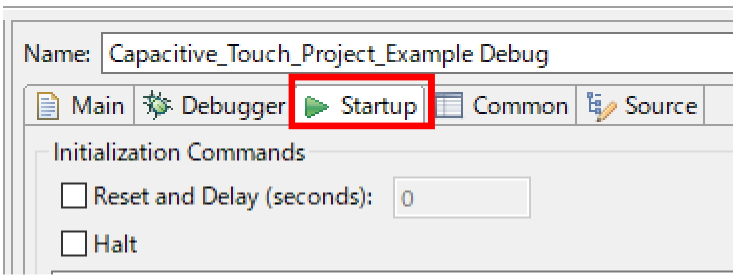

Figure 36. Startup Tab

3. Ensure the two check boxes **Set breakpoint at:** and **Resume** are checked and look as follows in the Runtime Options. You may need to scroll down in the dialog box to see these check boxes

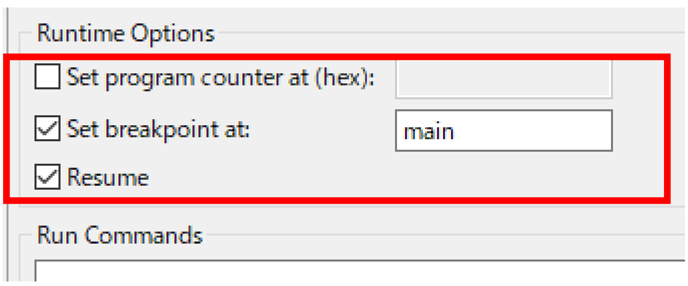

Figure 37. Runtime Option

4. Click **OK** to use these modified settings. This completes the project configuration and debug setup for tuning.

6.5 Tuning the Capacitive Touch Interface Using QE for Capacitive Touch [RA, RL78, Synergy] Plug-in

1. Make sure the emulator is connected correctly to the board and PC.

2. To start the automatic tuning process, click the **Start Tuning** button in the **CapTouch Main / Sensor Tuner RA, RL78, Synergy (QE)** e^2 studio IDE.

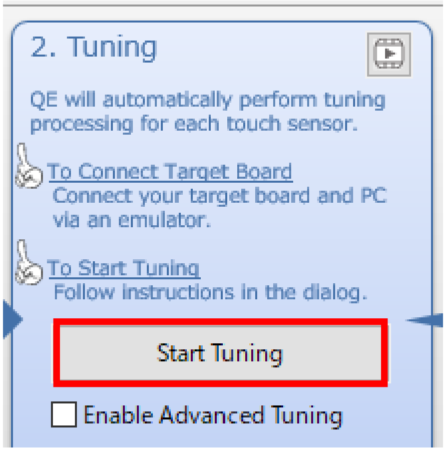

Figure 38. Start Tuning

3. At the start of the first debug session, e^2 studio may display a message regarding a switch to the Debug perspective. Click the **Remember my decision** check box and **Yes** to continue the Debug session and the QE for Capacitive Touch [RA, RL78, Synergy] automatic tuning.

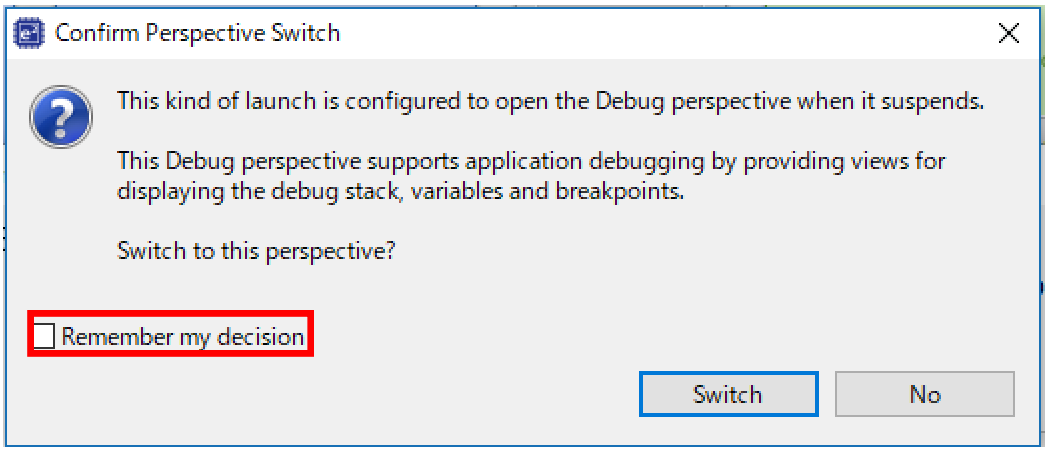

Figure 39. Confirm Perspective Setting

4. QE for Capacitive Touch [RA, RL78, Synergy] automatic tuning will now begin. Please carefully read the **Automatic Tuning Processing** dialog windows as they will guide you through the tuning process. An example screen is shown below. Typically, no interaction is required during the initial tuning process steps.

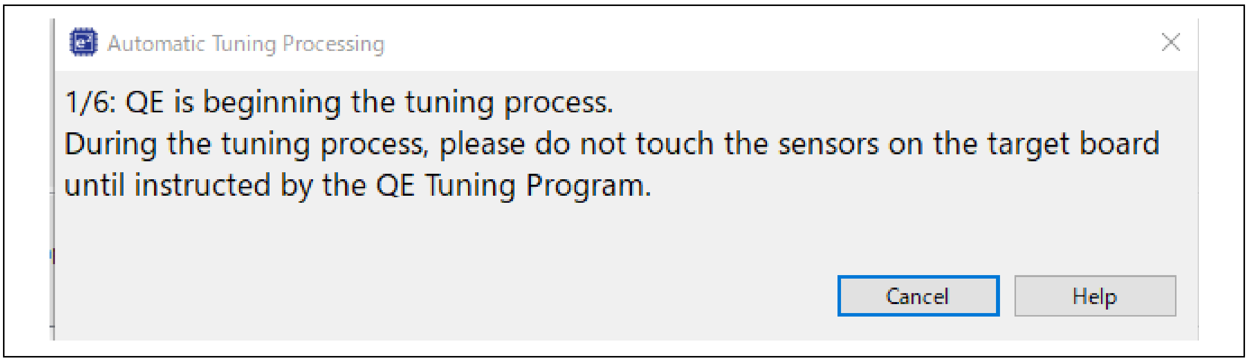

Figure 40. Automatic Tuning Processing Dialog (now preparing)

5. After several automated steps, a dialog box with information similar to what is shown below will appear. This is the touch sensitivity measurement step of the tuning process.

As the first interactive step of the tuning process, press “Button00/TS31” (sensor) using normal touch pressure as indicated in the dialog box. When pressing the self-capacitance sensor, the bar graph will increase to the right and the touch counts will increase numerically. While continuing to apply pressure to the sensor, press any key on the PC keyboard to accept the measurement

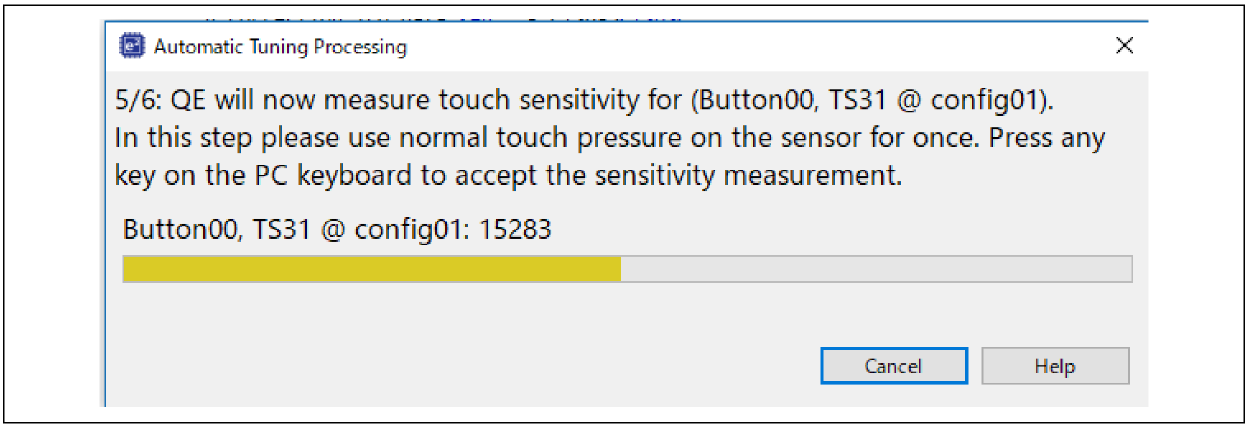

Figure 41. Automatic Tuning Processing Dialog (now measuring)

6. Once tuning is complete, a dialog like the following will appear allowing threshold confirmation. This is the detection threshold that is used by the middleware to determine if a touch event has occurred.

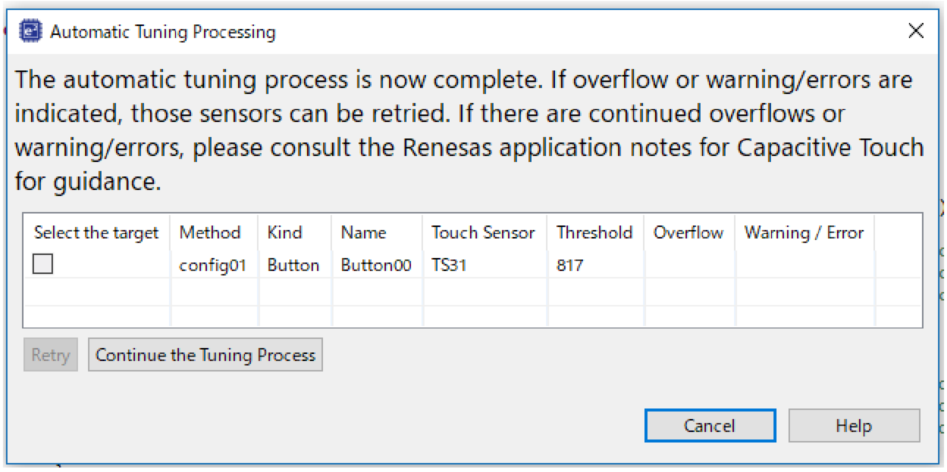

Figure 42. Automatic Tuning Processing (tuning complete)

7. Click the **Continue the Tuning Process** button in the dialog box shown. This will exit the tuning process and disconnect from the Debug session on the target board. This should return you to the default **Cap Touch Main / Sensor Tuner RA, RL78, Synergy (QE)** screen in the e^2 studio IDE.

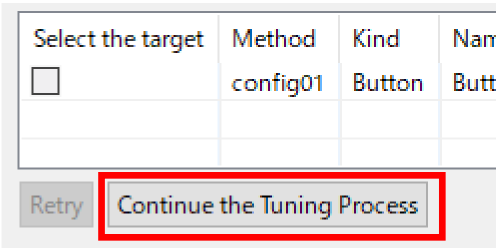

Figure 43. Continue the Tuning Process Button

8. In this example, the final step is to output the tuning parameter files by clicking the **Output Parameter Files** button.

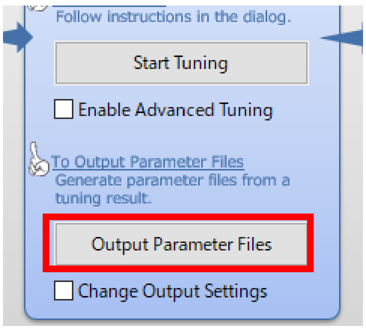

Figure 44. Output Parameter Files

9. In the **Project Explorer** window under folder **qe_gen**, confirm that **qe_touch_config.c**, **qe_touch_config.h** and **qe_touch_define.h** files have been added. These contain the tuning information necessary for enabling touch detection with drivers.

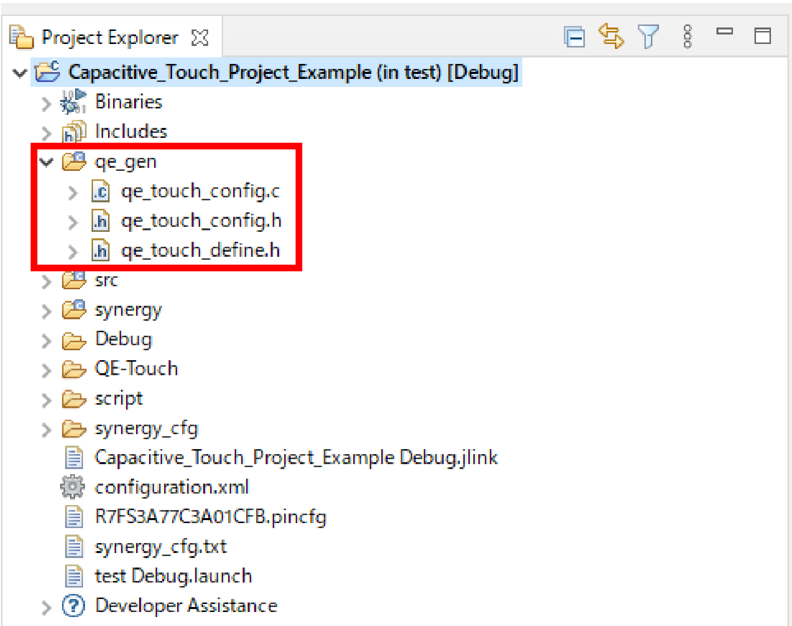

Figure 45. Generated Parameter Files

10. Build the project using the hammer icon in the upper left-hand side of the e^2 studio IDE. Confirm that the build results shown in the **Console** window do not show any errors.

6.6 Adding qe_touch_main() middleware to the Application Example

1. To implement a program (code) that will scan and report the state of the touch sensor, click the **Show Sample** button in the **CapTouch Main / Sensor Tuner RA, RL78, Synergy (QE)** e² studio IDE.

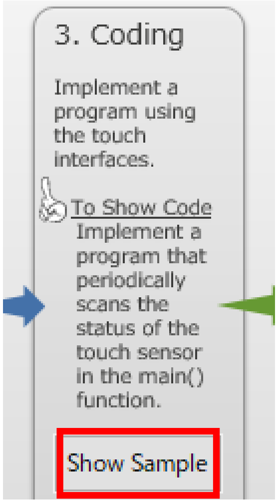

Figure 46. Show Sample Code

2. The **Show Sample Code** window will open which shows the sample code in text. Click the **Output to a File** button to output the sample code. Finally, click **OK** to close the window.

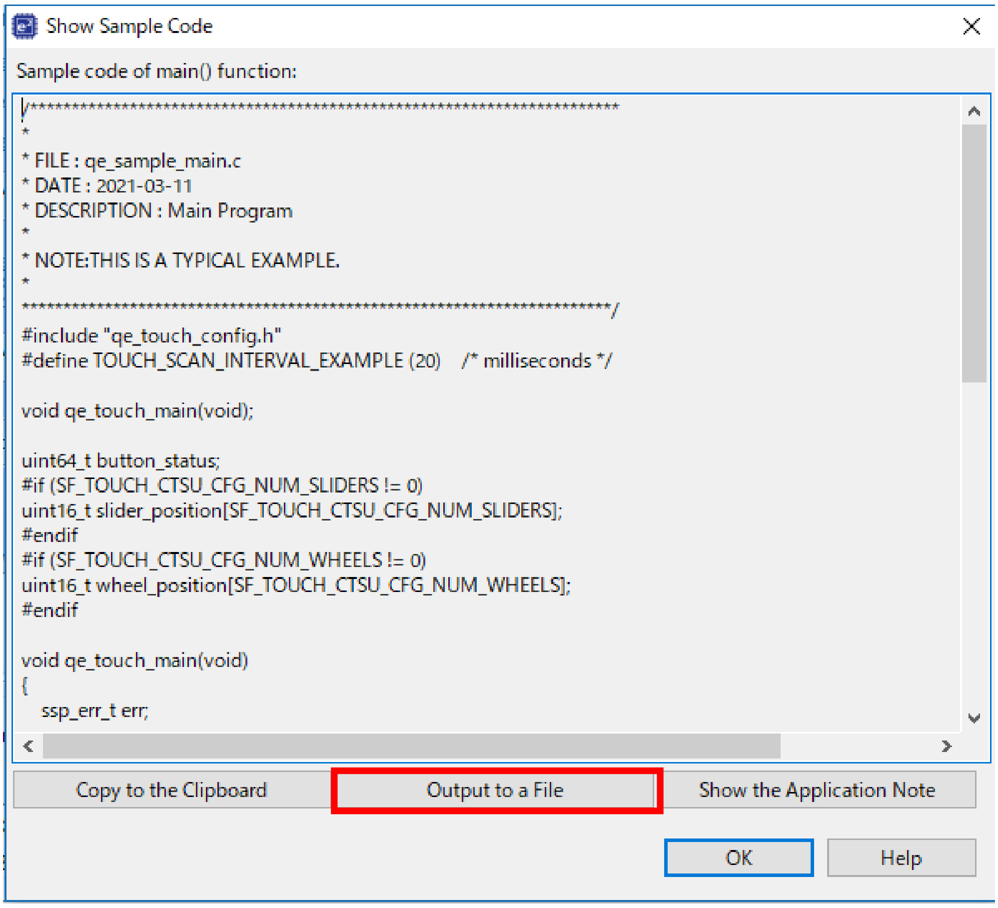

Figure 47. Show Sample Code Window

3. Confirm that the new qe_touch_sample.c file has been created in **Project Explorer** in the **qe_gen** folder.

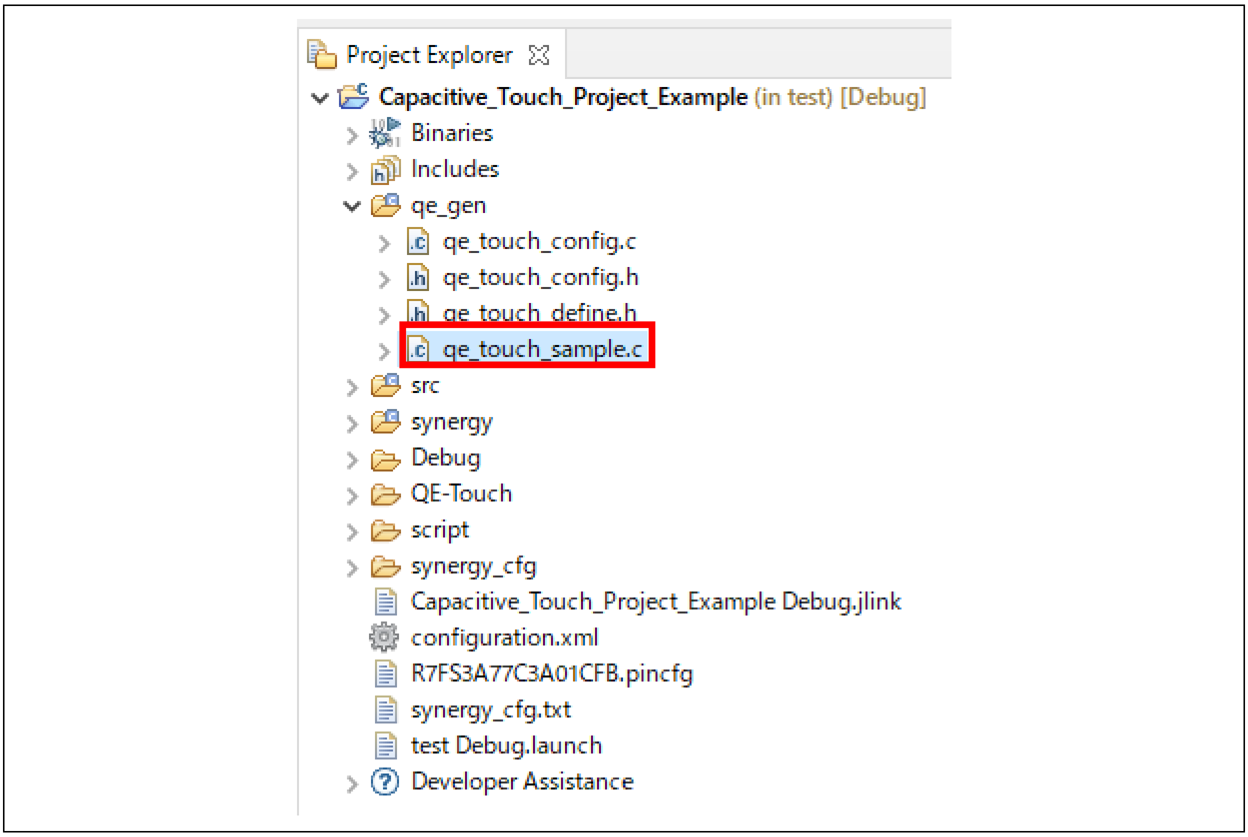

Figure 48. qe_touch_sample.c

4. In the **src** folder, open hal_entry.c.

   Add the code to call the **qe_touch_main()** function in the qe_touch_main() function declaration and in the hal_entry() function.

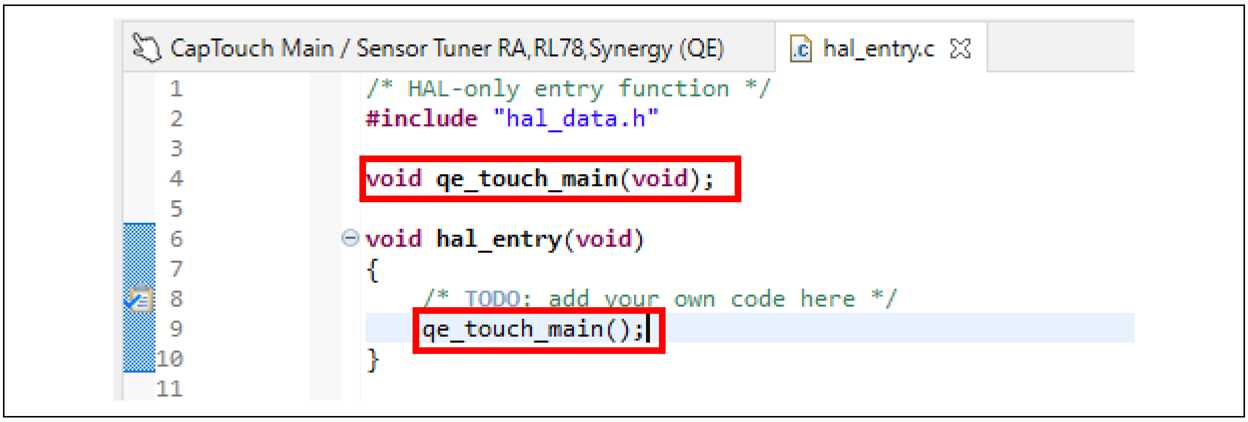

Figure 49. Add Call to hal_entry.c

5. This completes all the necessary code modifications required for this application example. Build the code to confirm that there are no errors or warning.

6.7 Monitoring Touch Performance Using e^{2}studio Expressions Window and QE for Capacitive Touch [RA, RL78, Synergy]

1. Start a Debug session by clicking the **Bug** icon in the upper left-hand corner of e^2 studio.

   The debugger session will stop at the hal_entry () function call. This is the first code entry point in the main() function

2. Select **hal_entry () function** to open the declaration.

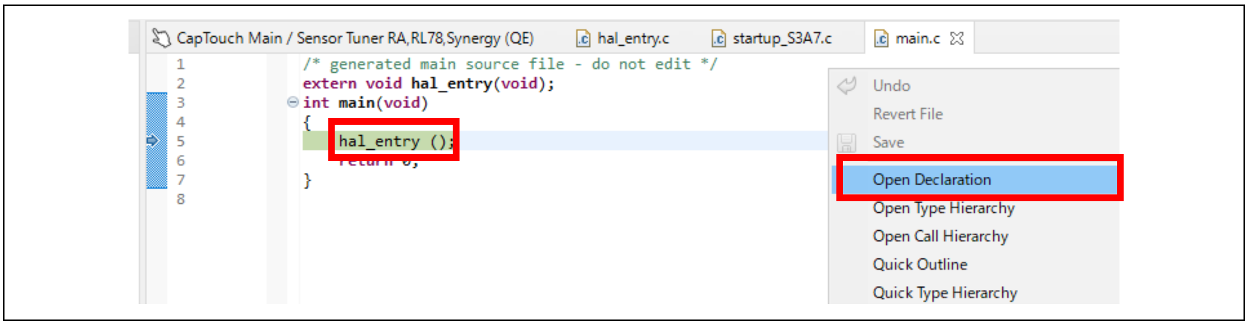

Figure 50. Open Declaration (hal_entry function)

3. In the hal_entry(), select the **qe_touch_main()** function and open the declaration.

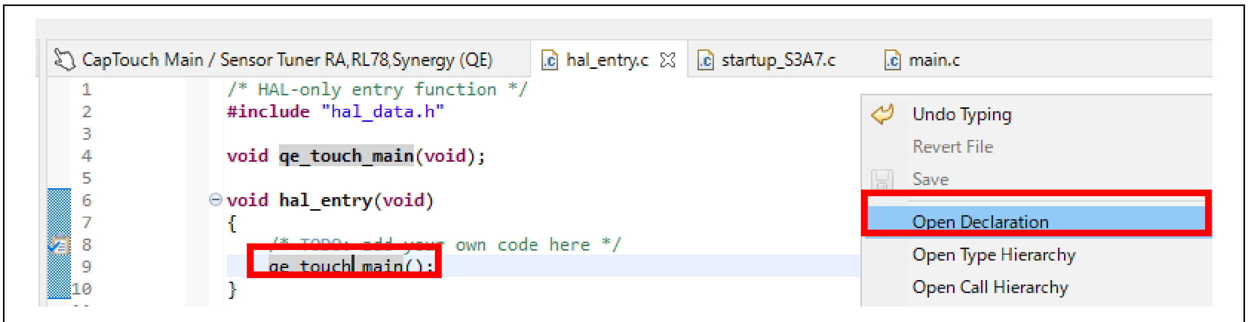

Figure 51. Open Declaration (qe_touch_main function)

4. Scroll down the qe_touch_main.c file in the **while (true) loop** and display **g_qe_touch_instance_config01.p_api->dataGet()**. With the argument "**button_status**" selected, right click and select **Add Watch Expression.**

Figure 52. Add Watch Expression Menu / Dialog

5. Right click in the **Expressions** window to select **Enable Real-time Refresh** and enable real-time refresh in the added variable.

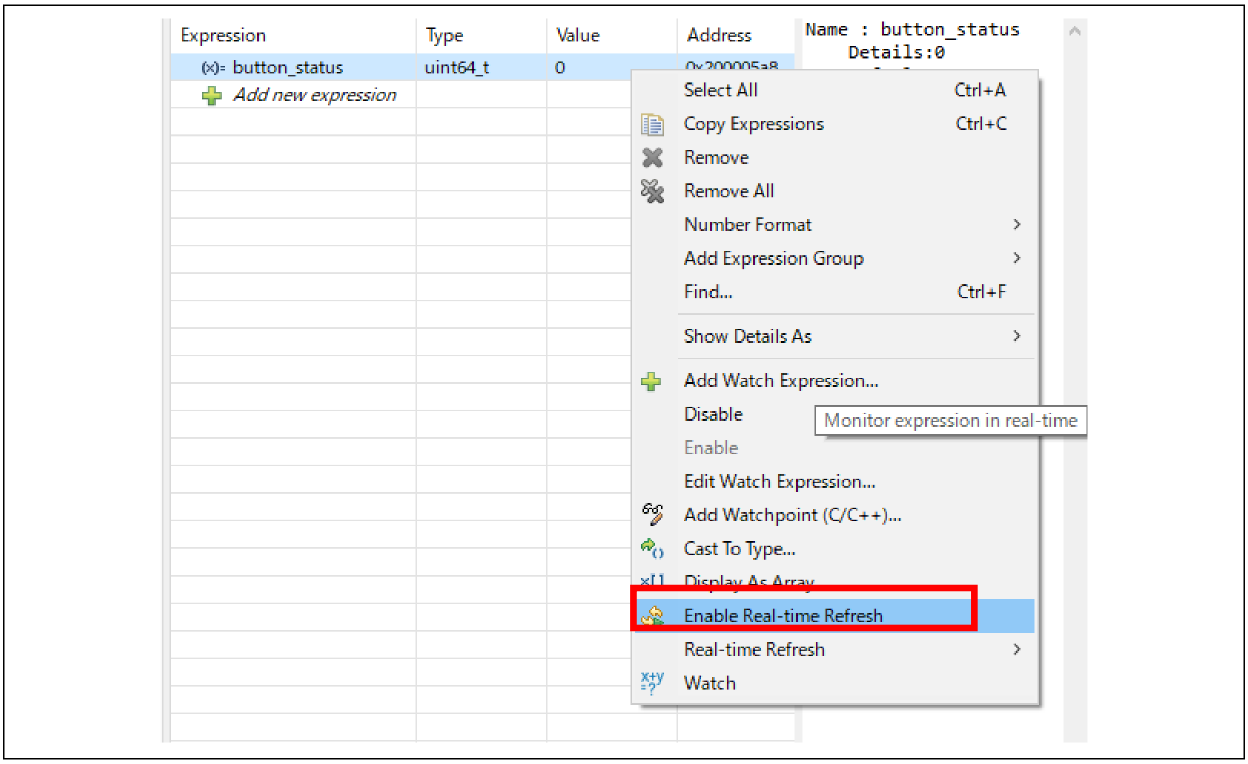

Figure 53. Enable Real-time Refresh Menu

6. Click the **Resume** (green arrow) button, located near the middle of the e² studio tool bar, to continue code execution.

7. Press the TS31 sensor on the board which was configured as "Button00" in Section 6.3 **Creating Capacitive Touch Interface**. When pressed, the "button_status" value in the Expressions window will change from '0' to '1', confirming the touch with a binary indication

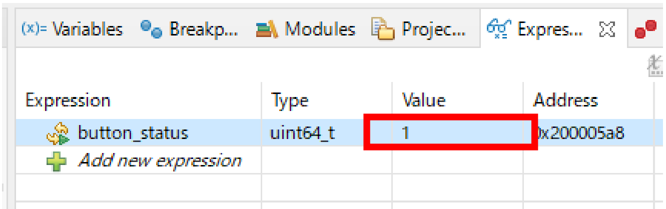

Figure 54. Touch Status Confirmation in Expressions Window

8. Click the **Show Views** button in the **Monitoring** pane of **CapTouch Main / Sensor Tuner RA, RL78, Synergy (QE)** to startup the **CapTouch Board Monitor RA, RL78, Synergy (QE)** pane

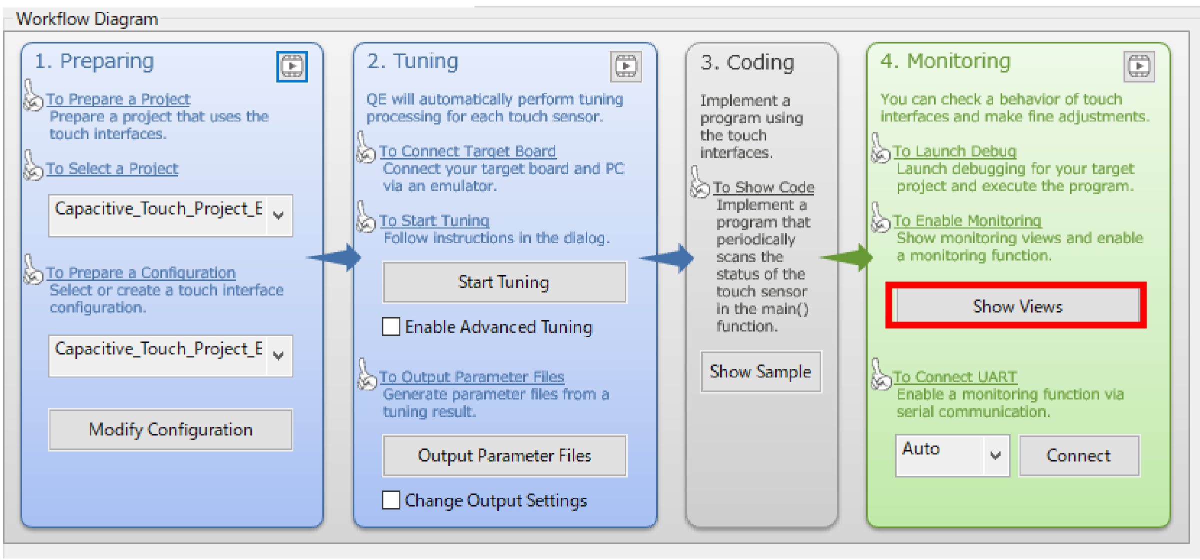

Figure 55. Show Views Button

9. The **CapTouch Board Monitor RA, RL78, Synergy (QE)** pane should appear like the image below.

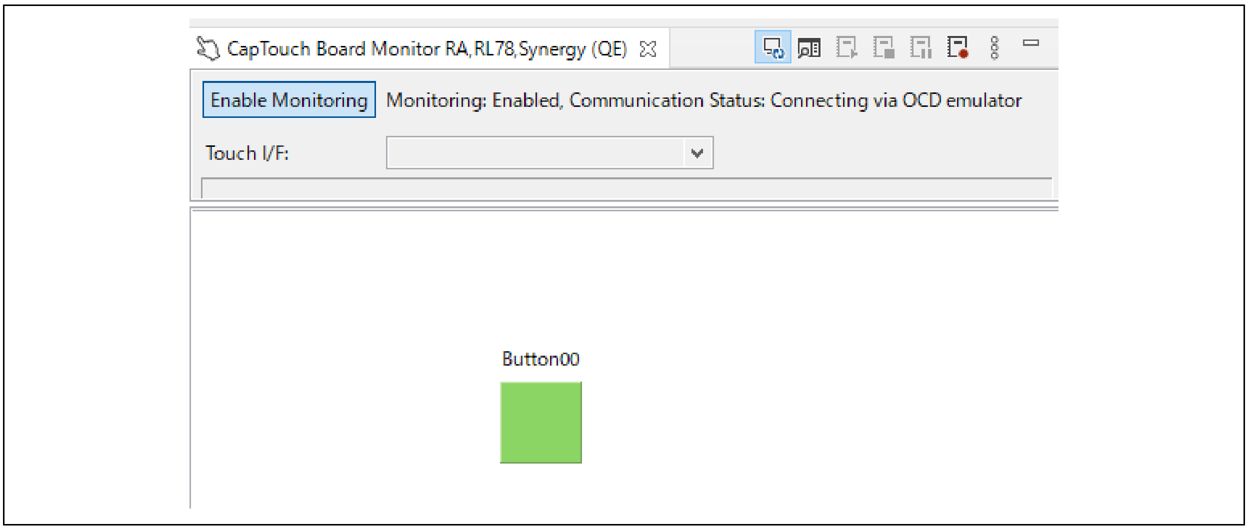

Figure 56. CapTouch Board Monitor RA, RL78, Synergy (QE) Window

10. [Click the **Enable Monitoring** button. The dialog text will change from “Monitoring: Disabled” to “Monitoring: Enabled.

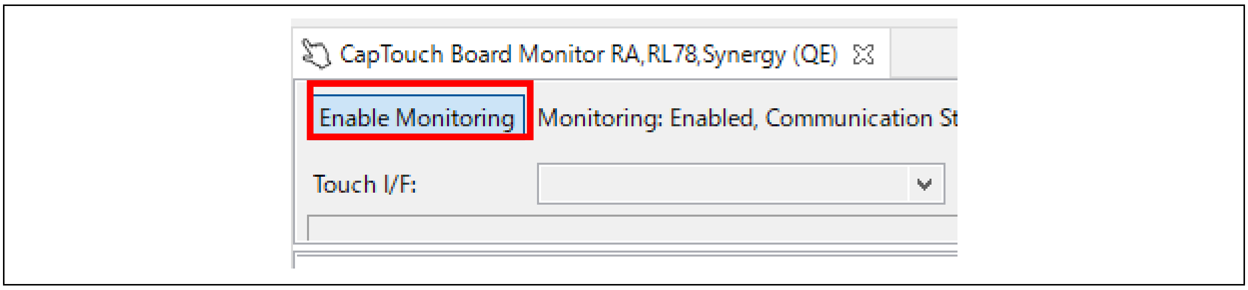

Figure 57. Enable Monitoring Function

11. Touch “Button00” (“TS31”) on the **BSW**. The **CapTouch Board Monitor RA, RL78, Synergy (QE)** will show a touch with a finger image on the button like the below image.

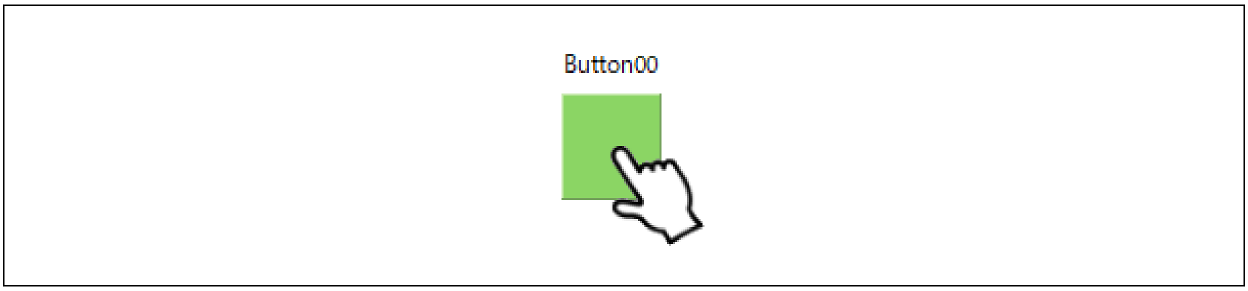

Figure 58. Touch Status Confirmation in CapTouch Board Monitor RA, RL78, Synergy (QE) Window

12. To see a graphical representation of the 'touch counts' from the board, open the **CapTouch Status Chart RA, RL78, Synergy (QE)** window.

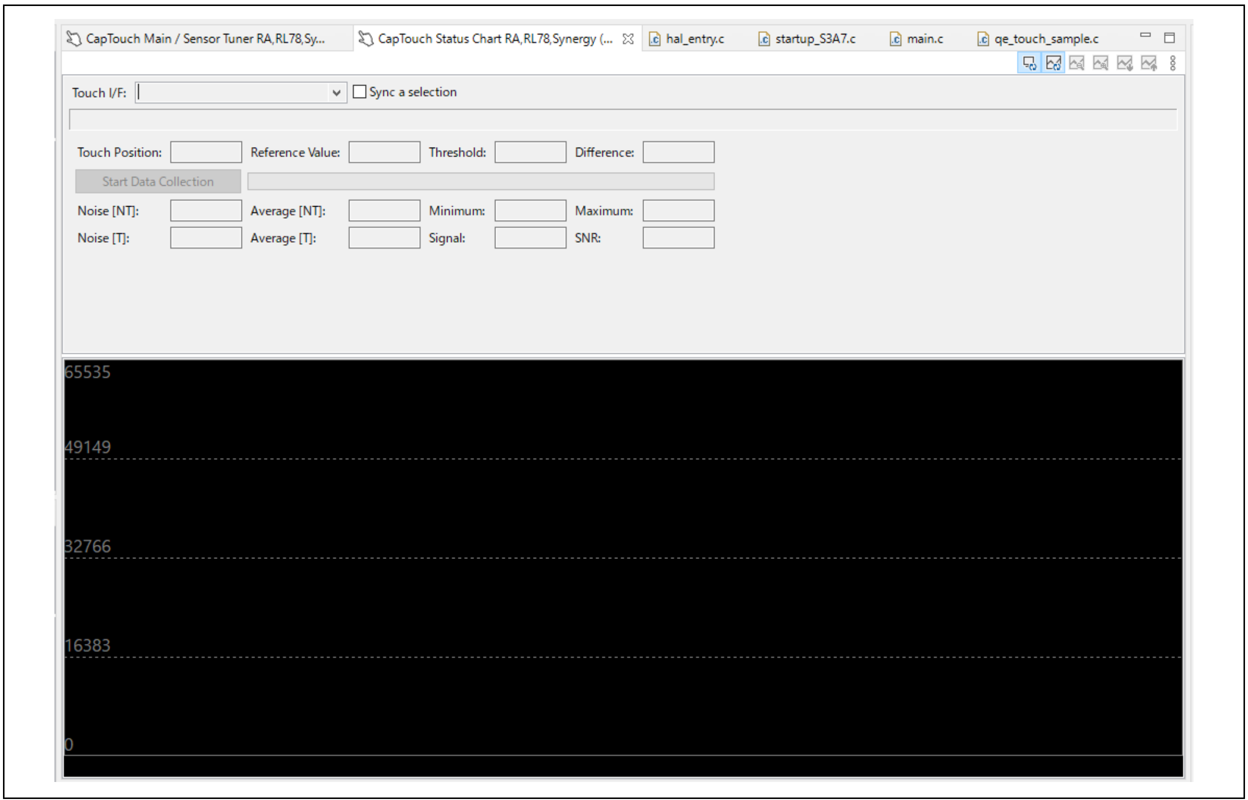

Figure 59. CapTouch Status Chart RA, RL78, Synergy (QE) Window

13. Using the **Touch I/F** pulldown, select "**Button00 @ config01**".

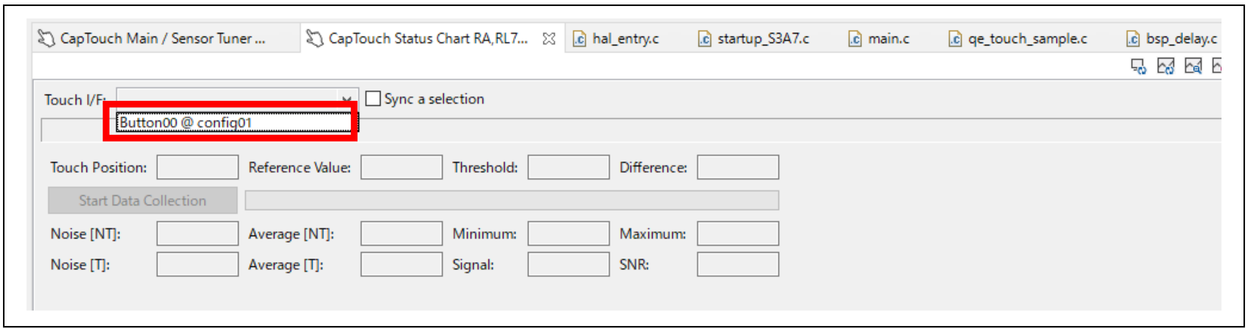

Figure 60. Touch I/F Configuration

14. The graph will begin to display running values. Touch “Button00” (TS31) on the board to view the touch counts shown as a step change on the running graph. The green line is the touch threshold, which the SPP middleware uses to determine whether a button is actuated/touched. The red blocks at the bottom of the graph are a visual indication to the user that the touch counts have exceeded the threshold and a touch is detected.

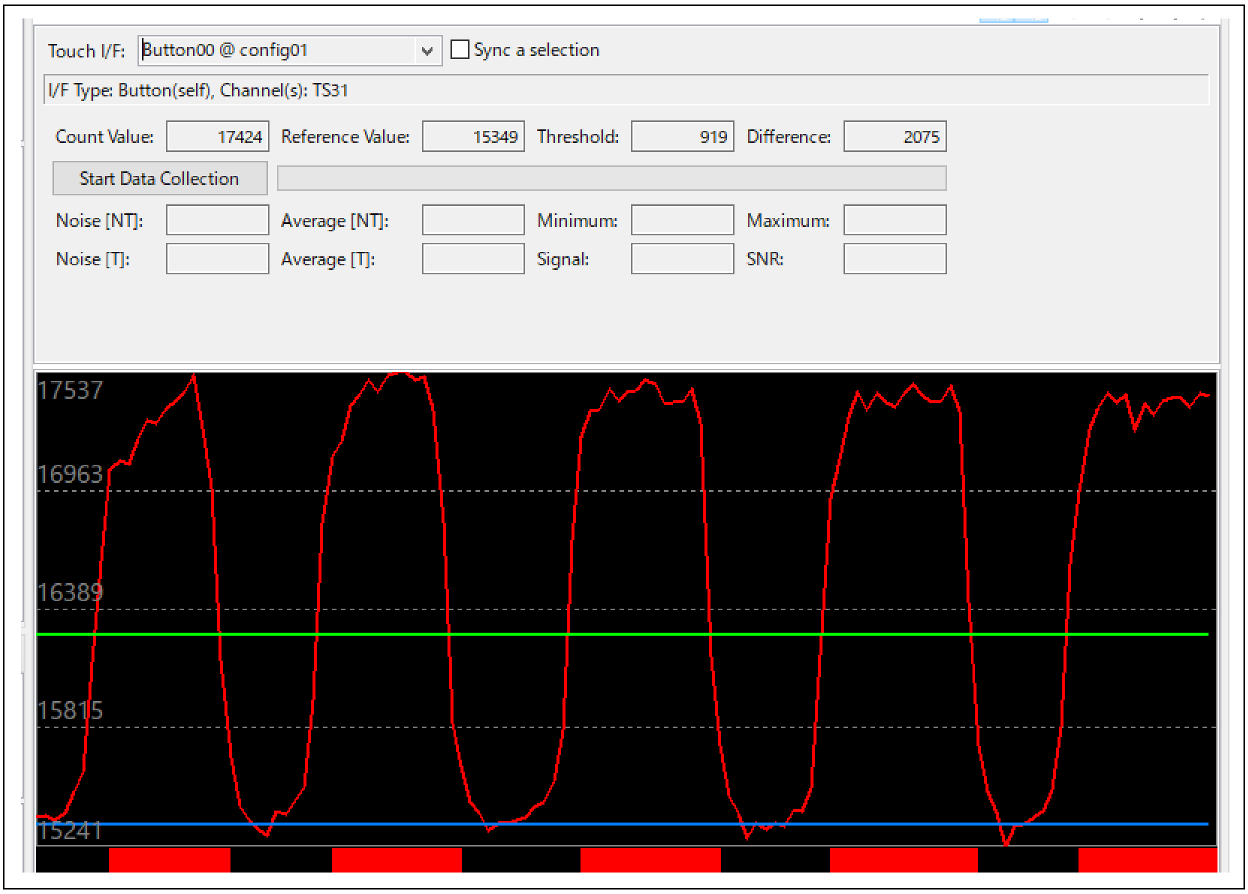

Figure 61. Confirm Touch Status in CapTouch Status Chart RA, RL78, Synergy (QE) Window

Note: Steps 15 to 18 must be set when displaying and measuring standard deviation.

15. Next, measure the standard deviation. Click the **Start Data Collection** button. While collecting data in the **touch-off state**, don't touch the electrode. The green bar is the data collection rate. When the green bar goes all the way to the right, the data collection in the touch-off state is complete.

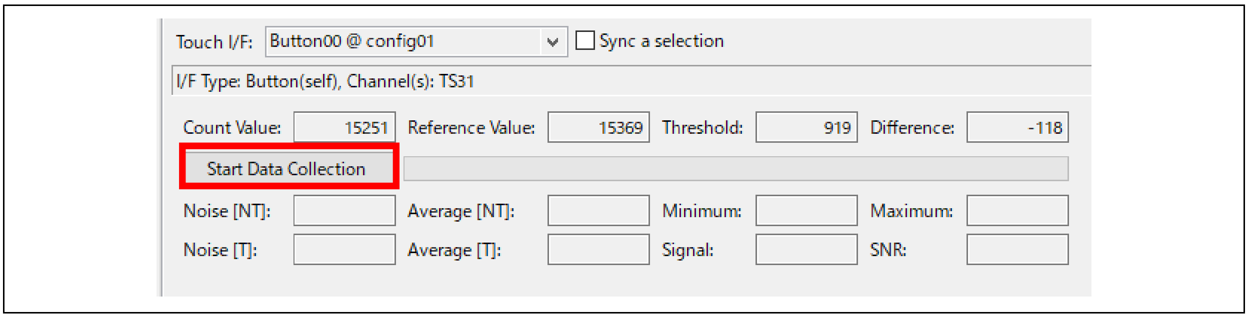

Figure 62. Start Data Collection Button (touch-off state)）

16. When the green bar goes all the way to the right, click the **Stop Data Collection** button.

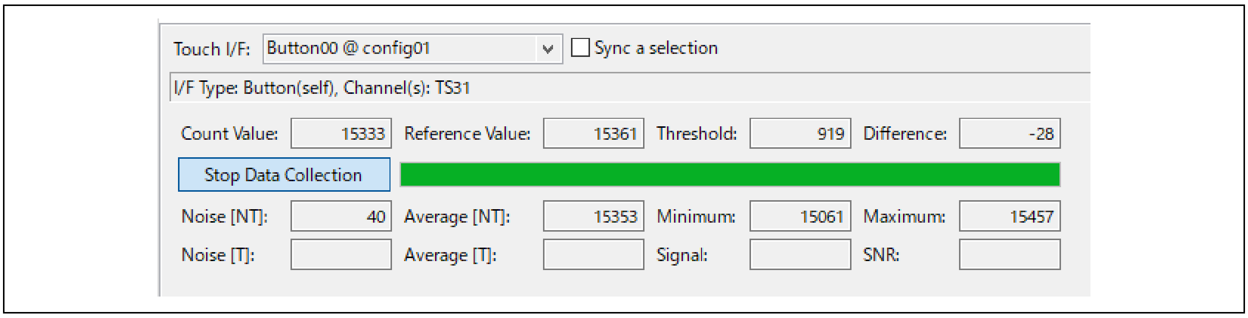

Figure 63. Stop Data Collection Button (touch-off state)

17. Next, touch the electrode in order to collect data in the touch-on state. Click the **Start Data Collection** button while touching the electrode.

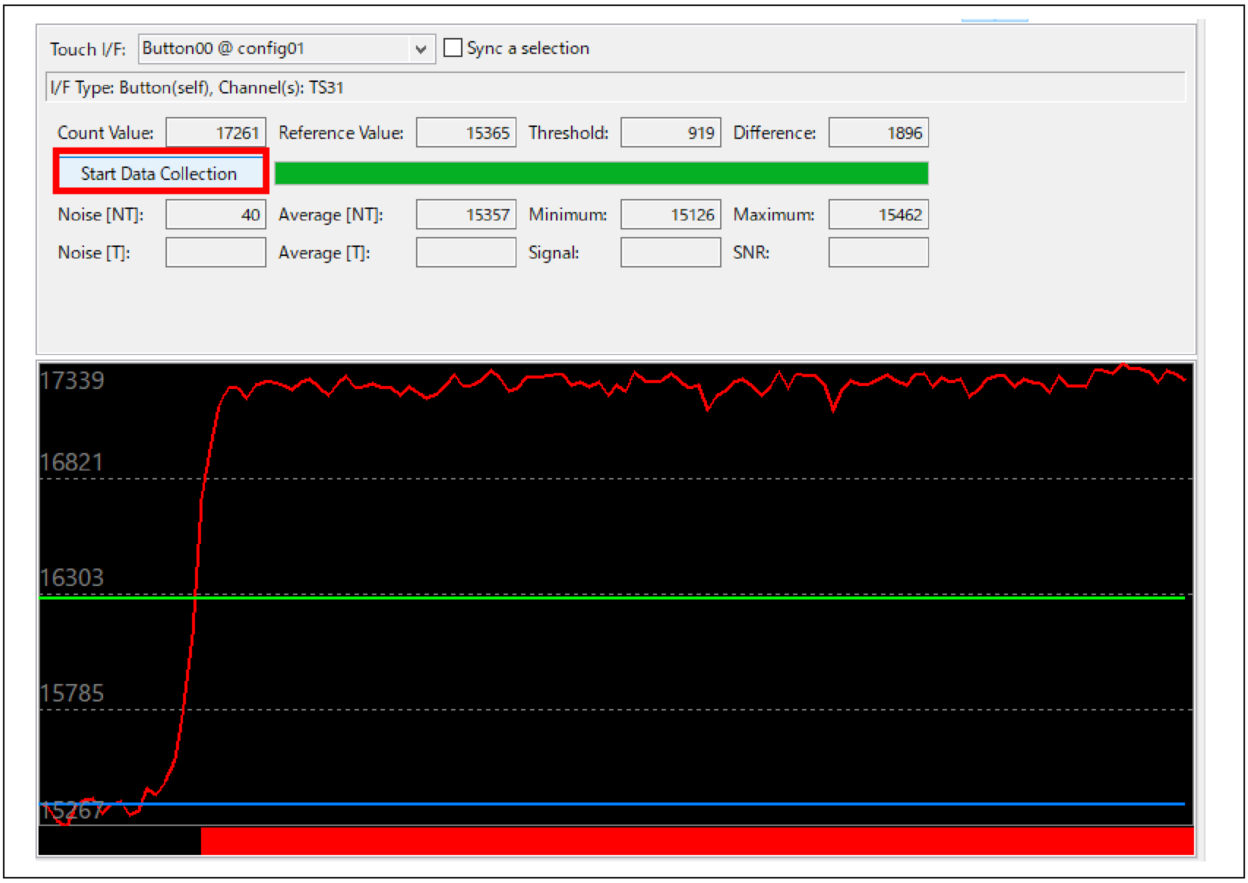

Figure 64. Start Data Collection Button (touch-on state)

18. When the green bar goes all the way to the right, click the **Stop Data Collection** button. The SNR is displayed when data collection is complete.

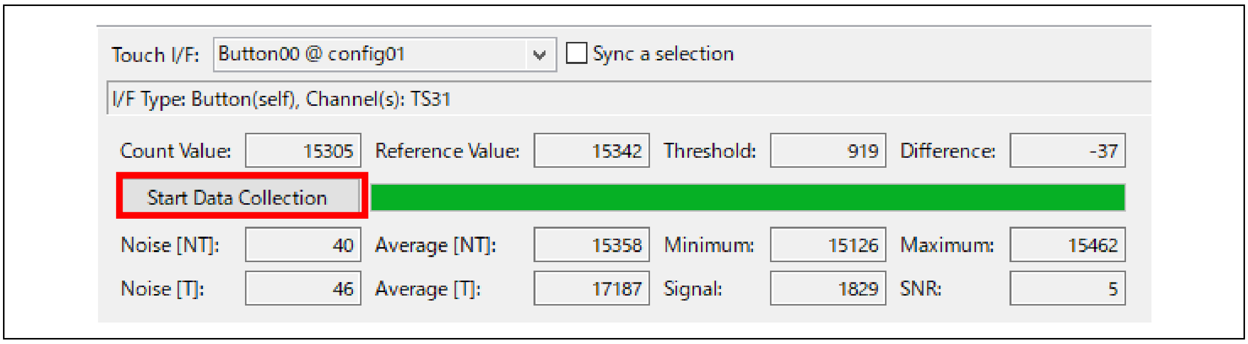

Figure 65. Stop Data Collection Button (touch-on state)

6.8 Monitoring Touch Performance with QE for Capacitive Touch [RA, RL78, Synergy] Using Serial Communication

1. When monitoring is in operation, click the **Enable Monitoring** button. The dialog text will change from “Monitoring: Enabled” to “Monitoring: Disabled”.

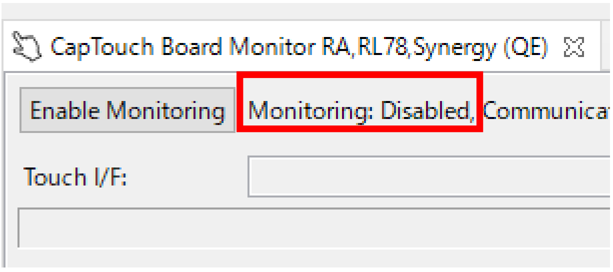

Figure 66. Disable Monitoring Function

2. To finish the debug session, click the **Stop** icon at the top-right of the e2 studio window.
3. Disconnect the emulator from the PC and the AE-CAP1-S3 V1.1 board. Confirm that the USB cable is correctly connected to the target board and PC.

   Note: Although operations can be carried out with the emulator connected, the emulator is removed in this step to confirm successful monitoring without the emulator.
4. Reset the AE-CAP1-S3 V1.1 board by pressing the **RESET** switch.

5. Open the **CapTouch Main / Sensor Tuner RA, RL78, Synergy (QE)** pane. Make sure following folder/file are selected:
 To Select a Project: “Capacitive_Touch_Project_Example”
 To Prepare a Configuration: “Capacitive_Touch_Project_Example.tifcfg”

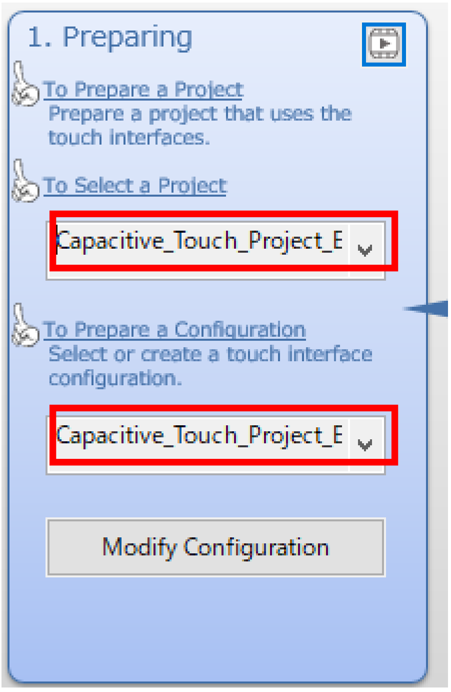

Figure 67. Select Project / Configuration

6. In the **CapTouch Main / Sensor Tuner RA, RL78 (QE)** pane under **Monitoring**, select **Connect** to enable monitoring using serial communication.

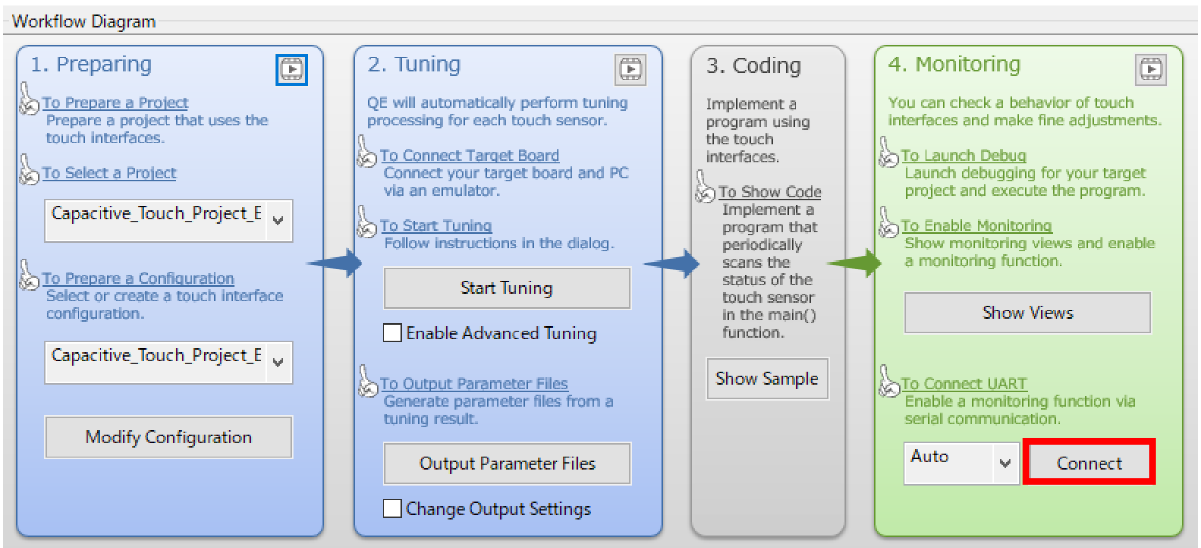

Figure 68. Connect Button

7. "Connected to COM*n*" should appear at the bottom of Console window. Confirm the message to make sure the connection is successful.

   Note: The COM port number for connection varies depending on the PC environment.

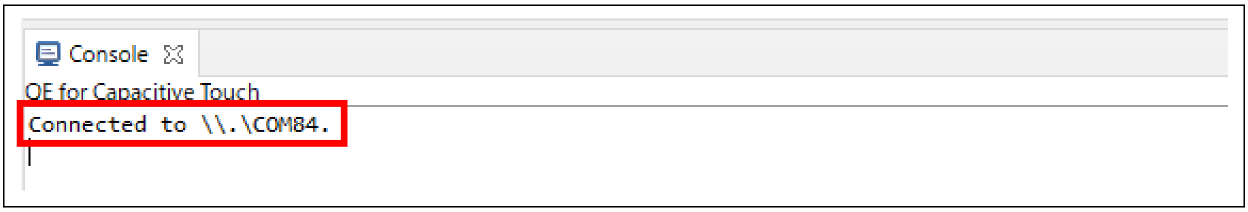

Figure 69. Console Window Output

8. The rest of the process is the same as Step 6.7 on in Section 6.7 Monitoring Touch Performance Using e²studio Expressions Window and QE for Capacitive Touch [RA, RL78, Synergy]

7. qe_touch_sample.c Listing After Modifications

```
/***********************************************************************
*
* 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);
    }
}
```

Website and Support

Renesas Electronics Website

https://www.renesas.com/

Capacitive Touch Sensing Unit related pages

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

Inquiries

https://www.renesas.com/contact/

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Jun.23.21	-	First edition issued

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)

   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on

   The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state

   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins

   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.
5. Clock signals

   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.
6. Voltage application waveform at input pin

   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).
7. Prohibition of access to reserved addresses

   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products

   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Trademarks

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit: www.renesas.com/contact/.