RA Family

Using QE and FSP to Develop Capacitive Touch Applications

Introduction

This document demonstrates the necessary steps for creating an application example that integrates capacitive touch sensing using Renesas RA Microcontrollers.

Target Device

RA family with Capacitive Touch Sensing Unit (CTSU, CTSU2)

Contents

1. Application Example Overview

This document demonstrates how to implement capacitive touch sensing functions using Renesas RA MCUs using the following procedures:

- Create a project using the RA Smart Configurator with the RA6M2 MCU board or RA2L1 MCU board
- Create a capacitive touch interface with QE for Capacitive Touch 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 e^2 studio/RA Smart Configurator, Flexible Software Package (FSP) drivers/middleware, and QE for Capacitive Touch plug-in help (contained within the e^2 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 give the reader an overview of how 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 RA Smart Configurator to add the required modules to the created e^2 studio project.
3. Use the QE for Capacitive Touch e^2 studio plug-in to create the capacitive touch interface.
4. Use the QE for Capacitive Touch e^2 studio plug-in to tune the application project.
5. Add the needed FSP 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 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	RA6M2 (CTSU)	RA2L1 (CTSU2)
Board kit	RA6M2 MCU Group Evaluation Kit (EK-RA6M2)	RA2L1-embedded CapTouch CPU Board (RTK0EG0022S01001BJ)
Renesas e^2 studio Integrated Development Environment (IDE)	2023-07 or later	
GCC ARM Embedded Compiler	12.2.1.20230214 or later	
Renesas QE for Capacitive Touch	3.3.0 or later	
Flexible Software Package (FSP)	4.6.0 or later	

5. Application Example Overview

In the main loop of the application example, the following processing is performed as shown in Figure 1.

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

A call to the RM_TOUCH_ScanStart()API starts the measurement (scan process).

↓

The RM_TOUCH_ScanStart()API return value is checked to determine if the rm_touch middleware is ready for processing.

↓

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

Figure 1. Application Example Overview

6. Capacitance Touch Application Development Procedure

6.1 Project Creation

1. On your PC, start the e^2 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 C/C++ Project > Renesas RA** from the e^2 studio menu.
3. When the **New C/C++ Project** dialog box opens, select "**Renesas RA CC++ Project**", then click **Next**.
4. In the **Renesas RA C/C++ Project** wizard, go to the **Project Name and Location** page and enter a **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 **Device and Tools Selection** dialog box, select the following:

Table 2. Device and Toolchain Selection

Item	**RA6M2 (CTSU)**	**RA2L1 (CTSU2)**
FSP Version	4.6.0	
Board	EK-RA6M2	RSSK-RA2L1
Device	R7FA6M2AF3CFB	R7FA2L1AB2DFP
Toolchains	GNU ARM Embedded 12.2.1.20230214	
Debugger	J-Link ARM	J-Link ARM, E2 (ARM) OR E2 Lite (ARM)

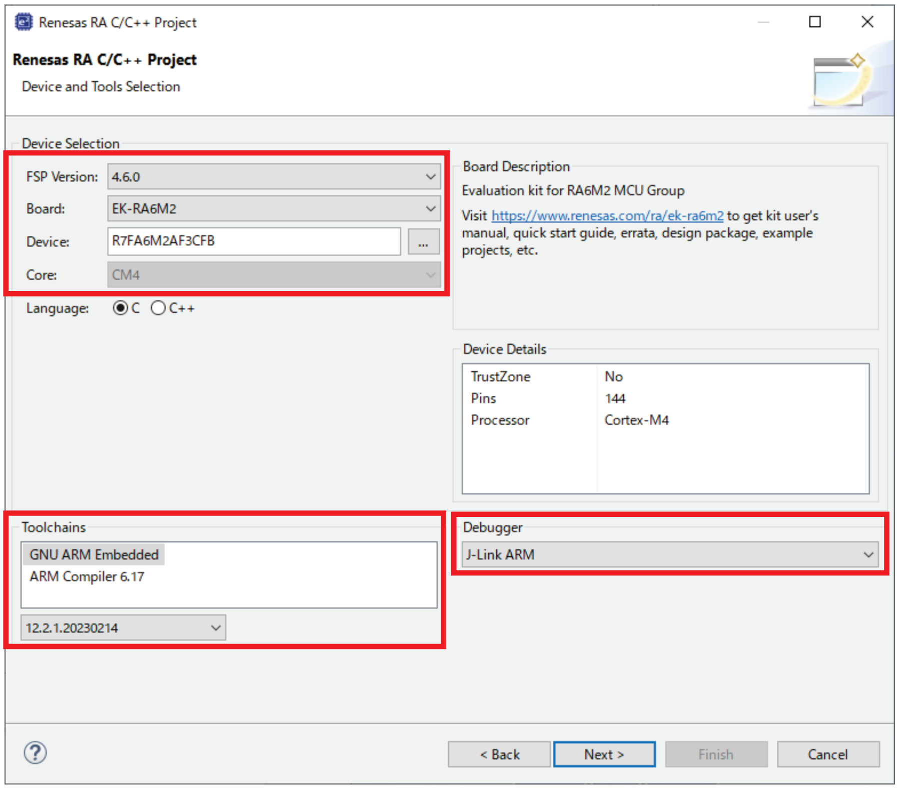

Figure 2. Device and Toolchain (RA6M2)

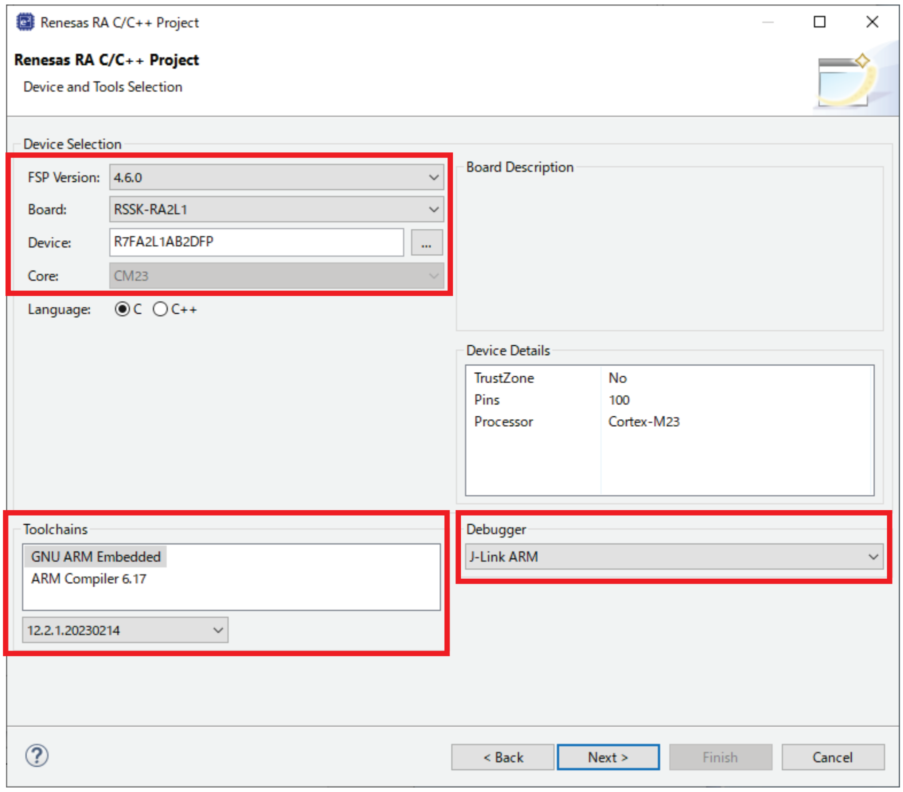

Figure 3. Device and Toolchain (RA2L1)

6. Once complete, click **Next**.

7. In the **Build Artifact and RTOS Selection** dialog box, select the following:

Table 3. Build and RTOS Selection

Item	RA6M2 (CTSU)	RA2L1 (CTSU2)
Build Artifact Selection	Executable	
RTOS Selection	No RTOS	

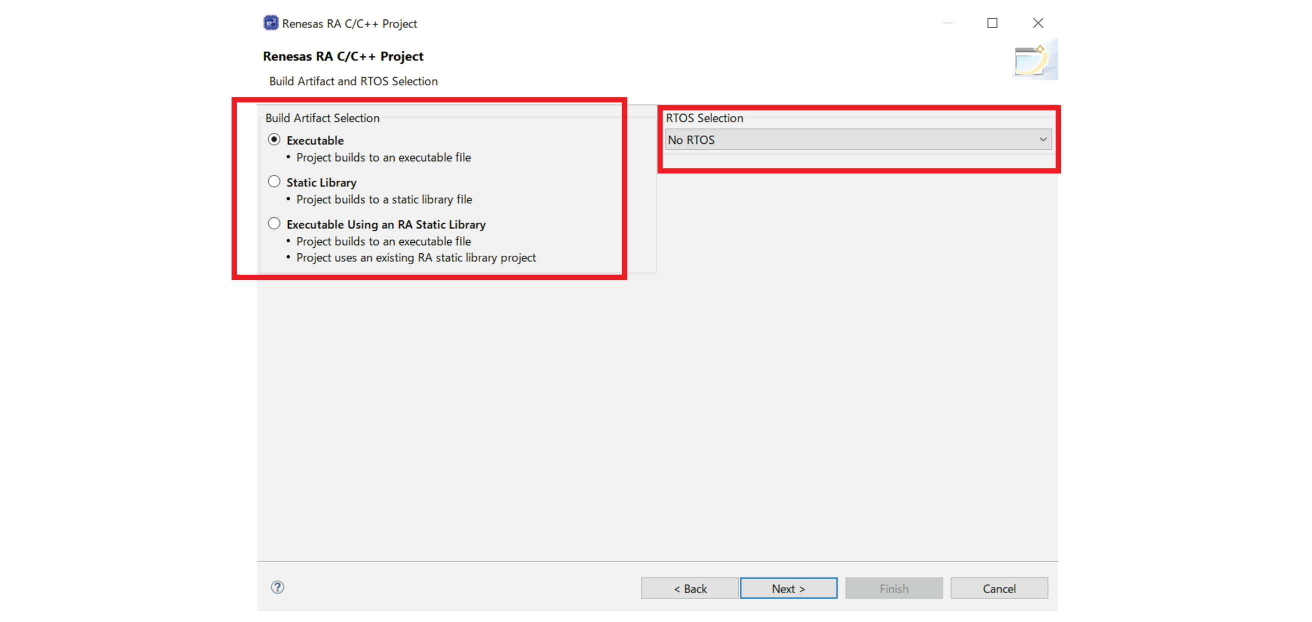

Figure 4. Build and RTOS Selection

8. Once complete, click **Next**.

   In the next dialog box, **Project Template Selection**, select "Bare Metal – Minimal", then click **Finish**.

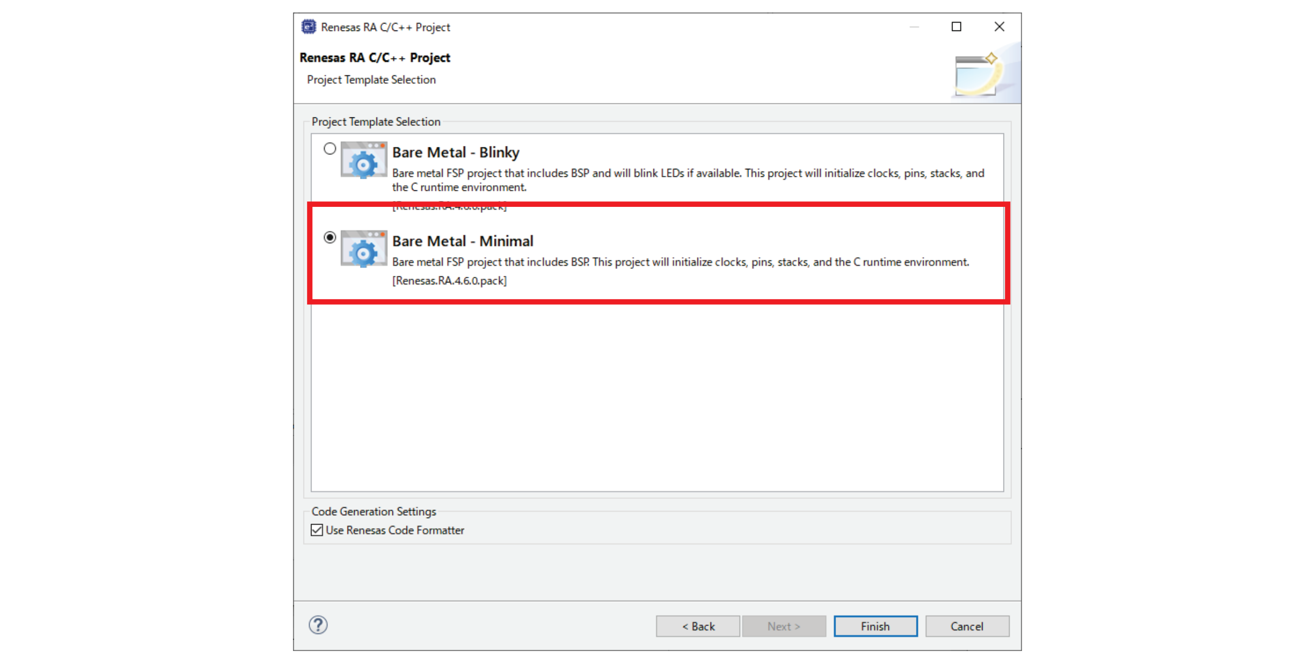

Figure 5. Project Template Selection

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

Using RA Configurator to Add Modules

6.2 Using RA 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.

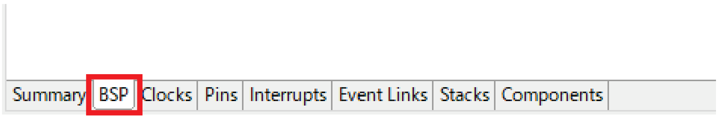

Figure 6. BSP Tab

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

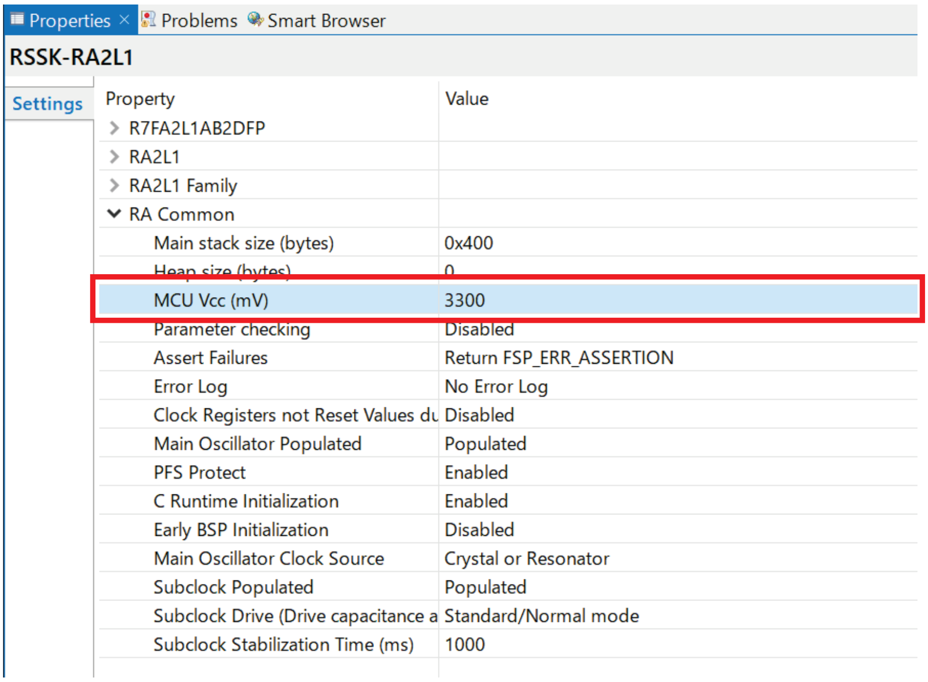

Figure 7. Power Supply Voltage Setting

3. Select the **Clocks** tab to configure the clocks.

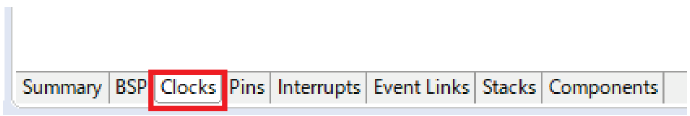

Figure 8. Clocks Tab

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

Table 4. Clocks Configuration

Clock	RA6M2 (CTSU)	RA2L1 (CTSU2)
PLL Src	XTAL	- (no setting)
Clock Src	PLL	HOCO
PCLKB	PCLKB Div /4	PCLKB Div /2

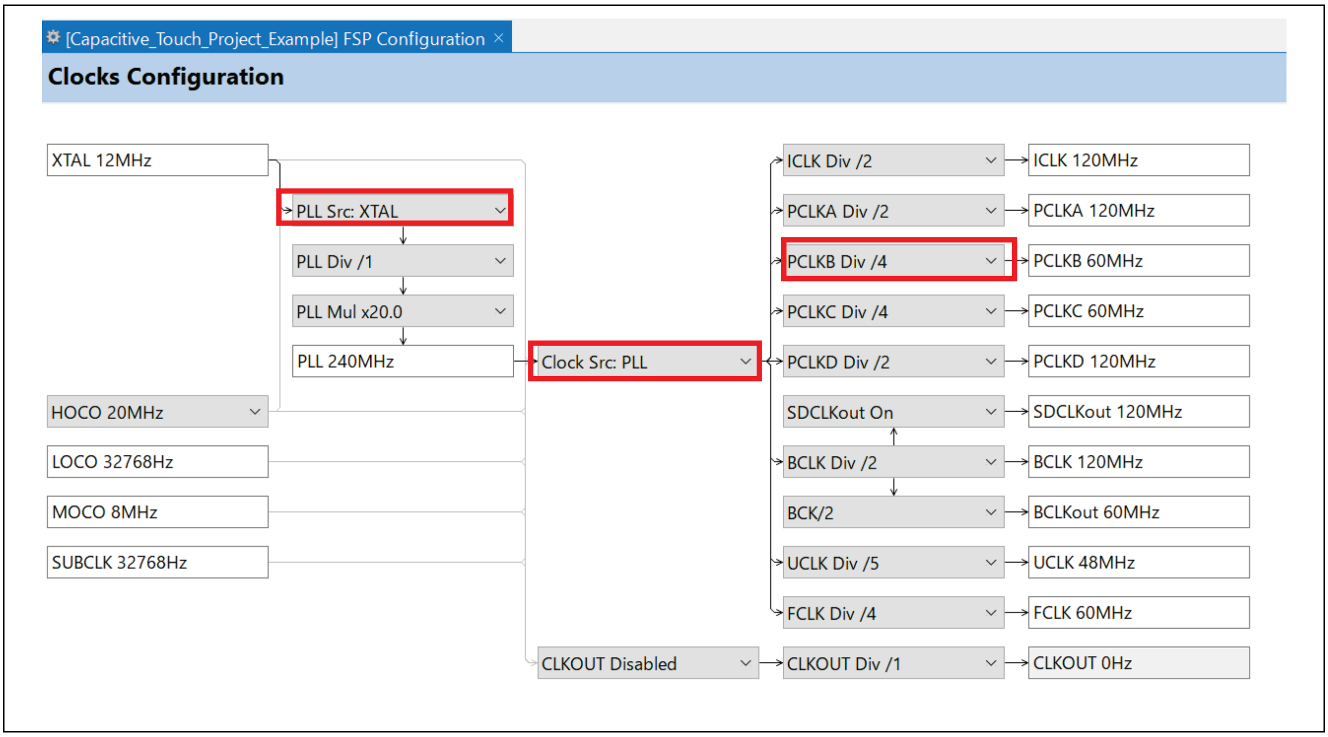

Figure 9. Clocks Configuration (RA6M2)

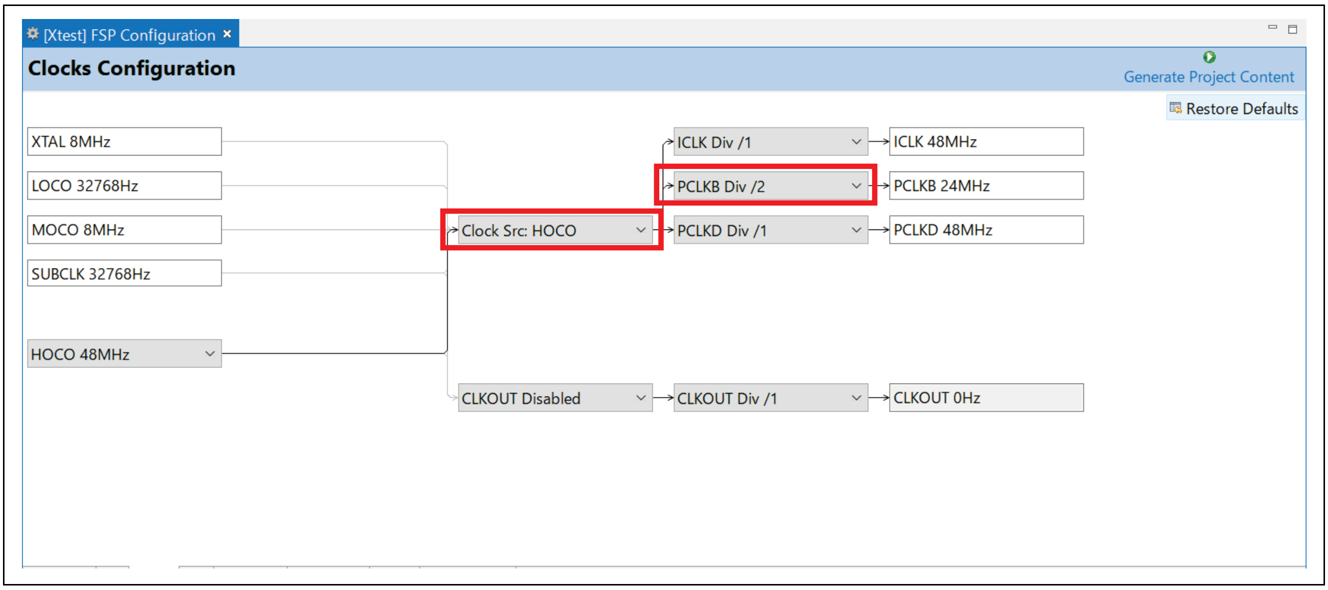

Figure 10. Clocks Configuration (RA2L1)

5. Next, select the **Pins** tab. Assign the pins in order to connect the sensor port of the MCU to the Capacitive Touch Application 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.

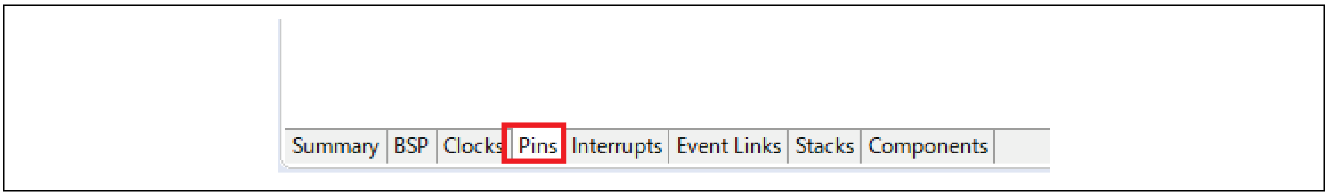

Figure 11. Pins Tab

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

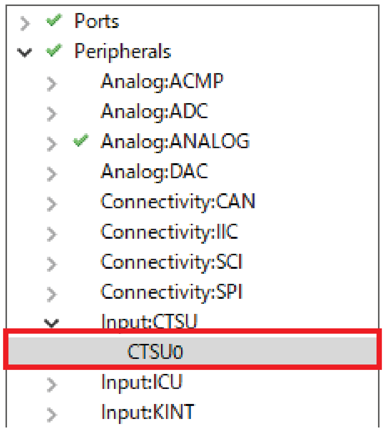

Figure 12 Select Peripherals (CTSU0)

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

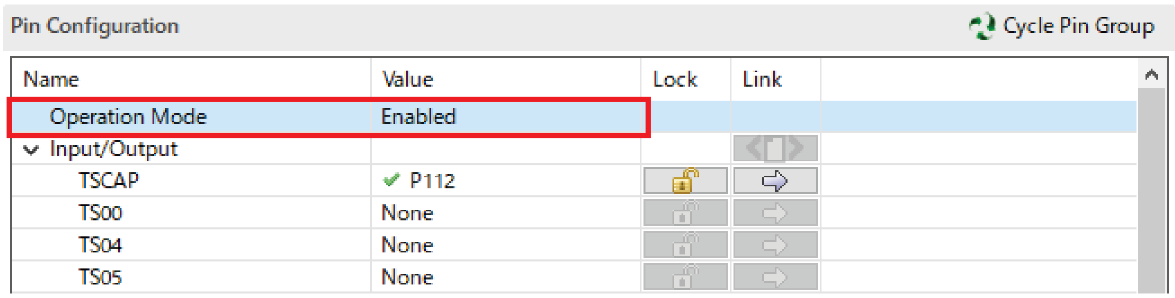

Figure 13. Operation Mode (CTSU0)

8. Enable the TS pins that you will be using. Since RA2L1 (CTSU2) can control the output of non-measurement pins at once, all TS pins should be enabled. In this application example, the following TS pins are assigned for use in capacitive touch interface, and so forth.

Table 5. TS Pins Used in Application Example

Item	RA6M2 (CTSU)	RA2L1 (CTSU2)
TS pins used in the application example	● TSCAP pin ● TS2 pin	● TSCAP pin ● TS0 pin ● TS11-CFC pin

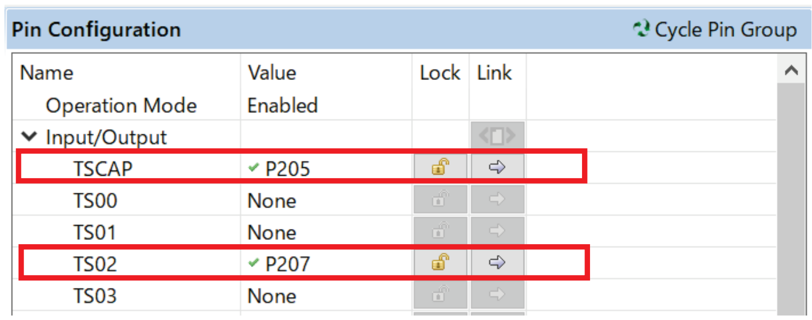

Figure 14. TS Pin Configurations (RA6M2)

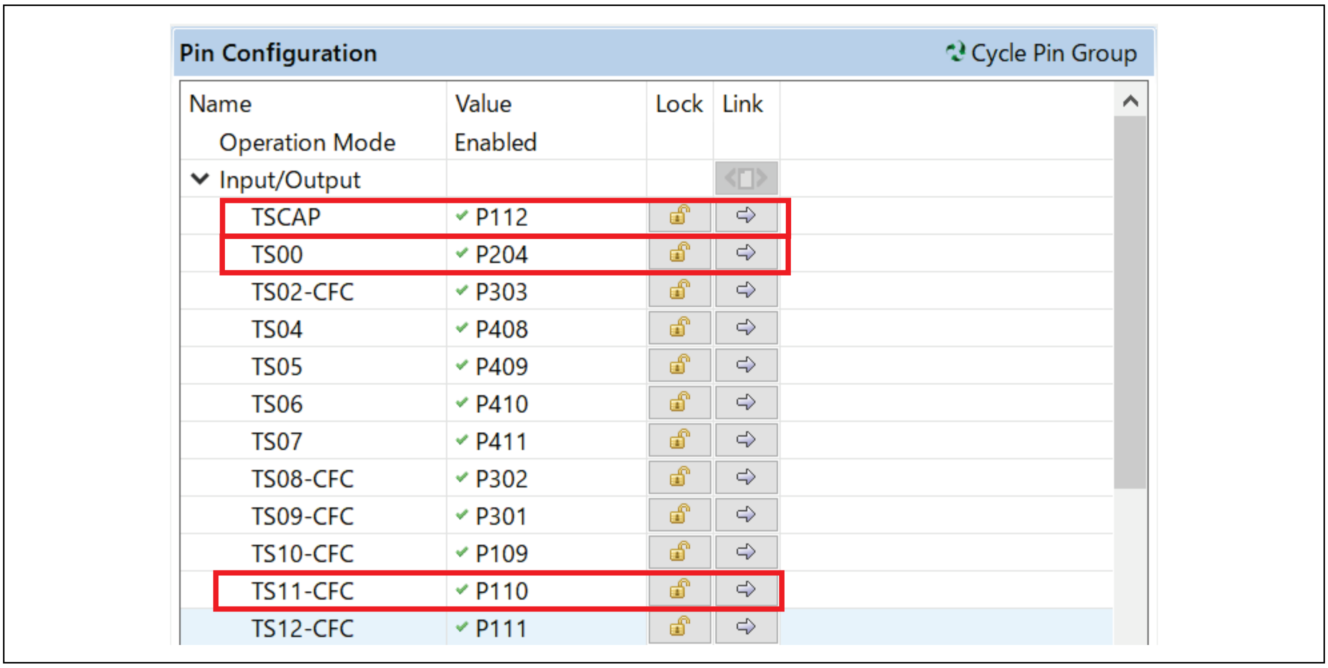

Figure 15. TS Pin Configurations (RA2L1)

9. For RA6M2 (CTSU), we recommend that the touch sensor pins that are NOT being used by the application be setup so that they are driven to 'low-level output' at startup.
To do so, expand the **Ports** tree node in **Pin Selection**. Expand the **P2** sub-node and select "P206". Change the **Mode** setting of the pin configuration from "Disable" to "Output" mode (Initial Low)".

Note: Only one port is setup here as a usage example. Set all unused sensor ports in the same manner.

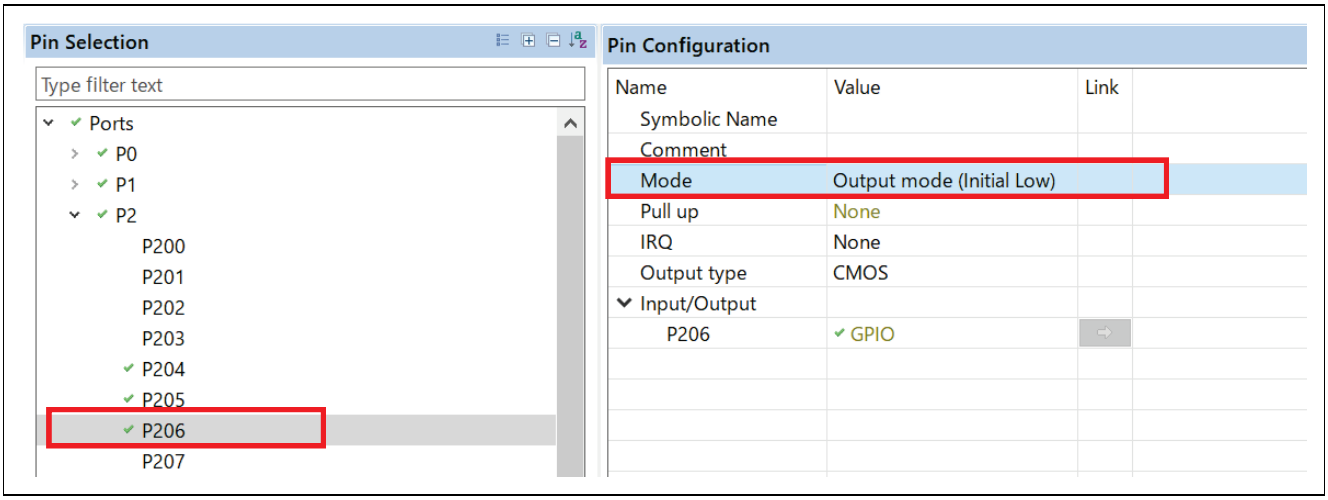

Figure 16. Low Output Drive Setting

10. Select **Pin Selection > Peripherals**, then open the **Connectivity:SCI** tree and select "SCI9."

    Note: Steps 10 and 11 should only be set when using serial communication for monitoring. Otherwise, skip to Step 12.

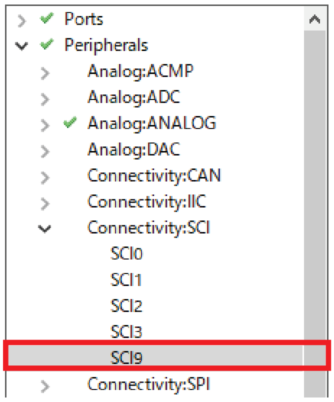

Figure 17. Select Peripherals (SCI9)

11. 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".

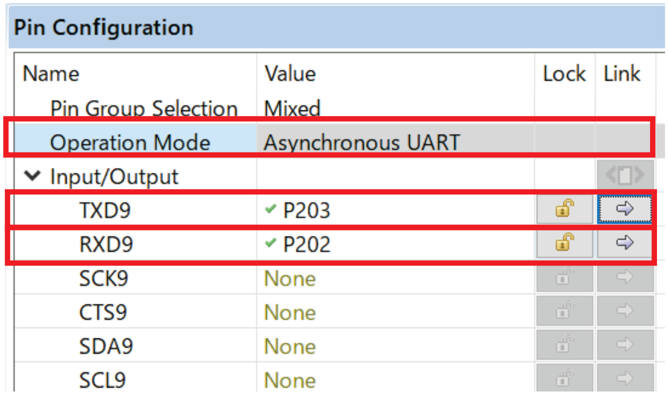

Figure 18. Operation Mode (SCI9)

12. Next, select the **Stacks** tab.

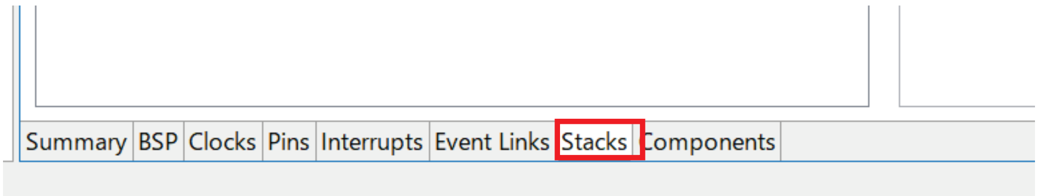

Figure 19. Stacks Tab

13. Select **New Stack > CapTouch > Touch (rm_touch)** to add the CTSU driver and middleware.

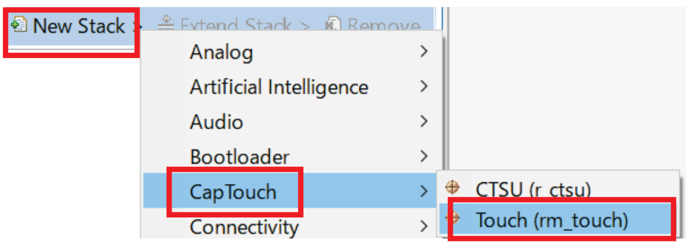

Figure 20. Add CTSU Driver and Middleware

14. **HAL/Common Stacks** appear as shown in the following figure.

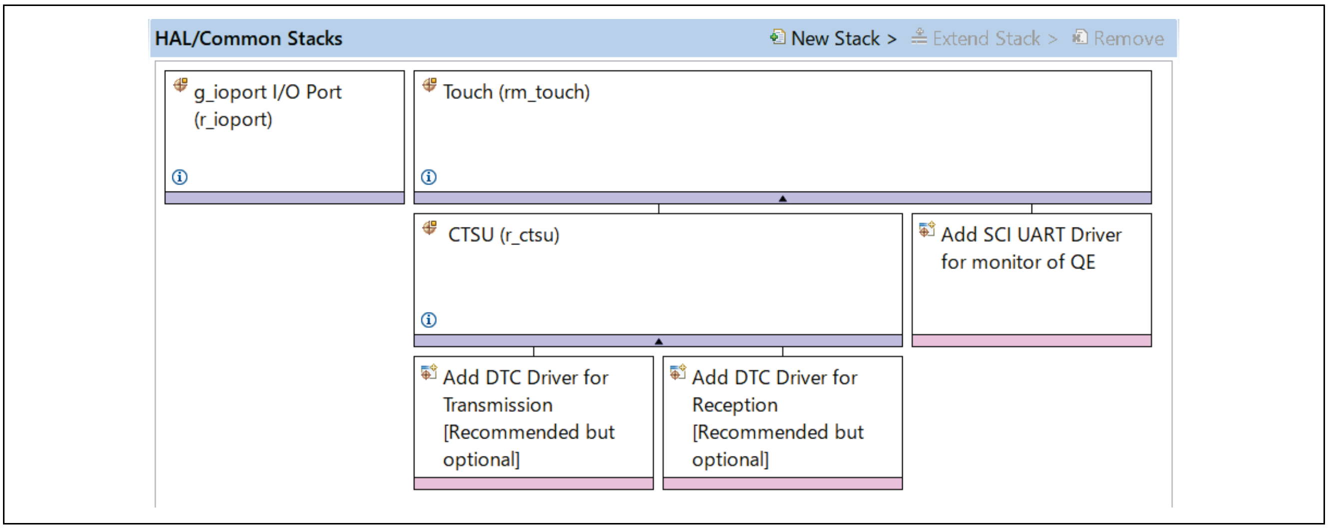

Figure 21. HAL/Common Stacks (after CTSU is added)

15. Next, click the **CTSU Driver on r_ctsu** module to display **Properties**.

Note: Steps 15 to 18 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 19.

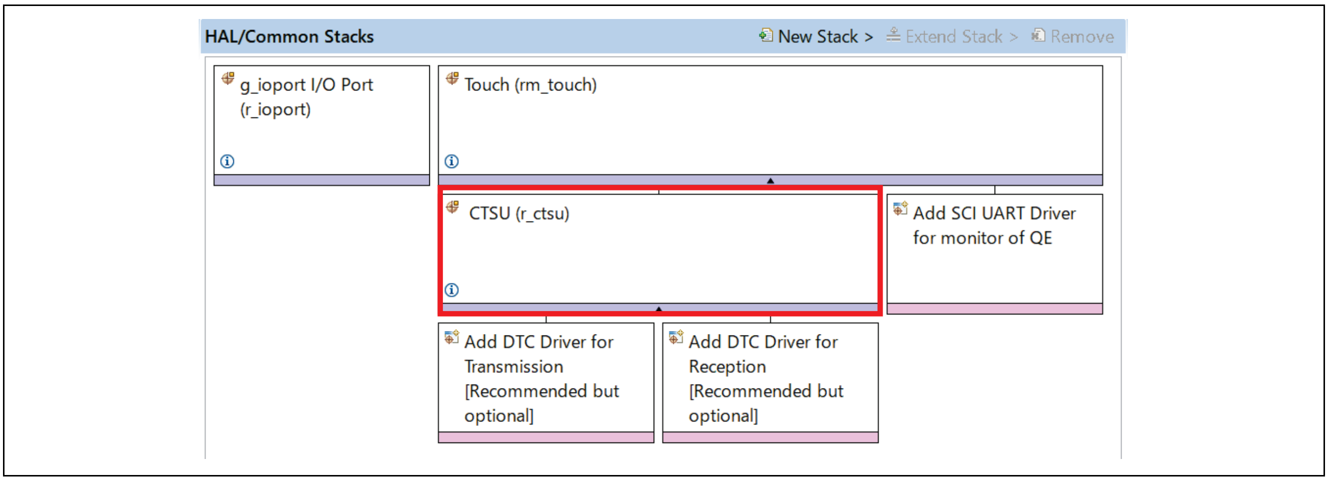

Figure 22. CTSU Driver on r_ctsu Module

16. Set the DTC from **Properties** at the bottom-left of the screen. Select **Settings > Common > Support for using DTC** and change the setting from “Disabled” to “Enabled”.

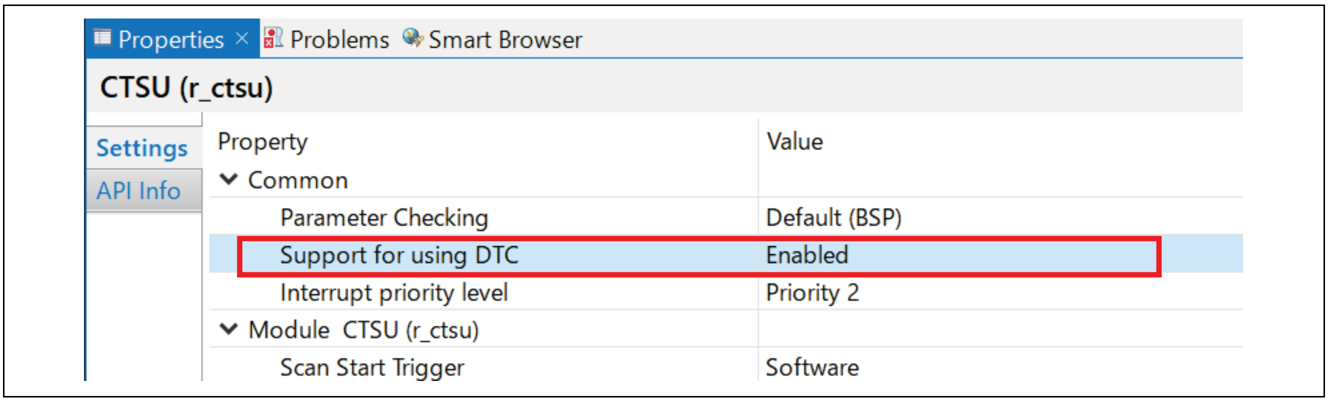

Figure 23. Support for Using DTC

17. Click the **Add DTC Driver for Transmission** module and select the **New > Transfer (r_dtc)** to add the DTC driver for transmission.

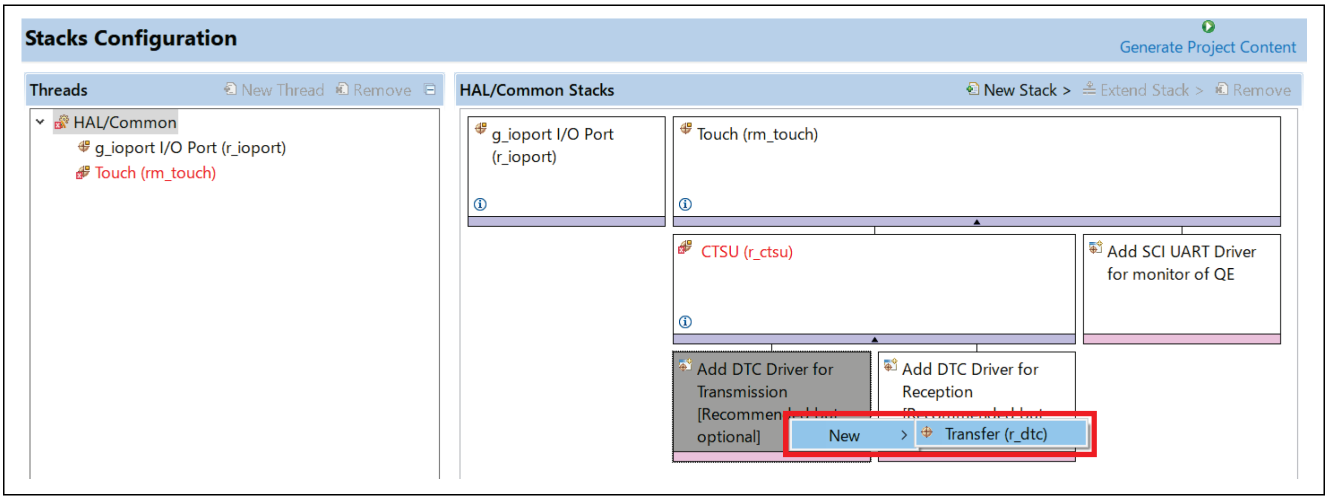

Figure 24. Add DTC Driver for Transmission

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

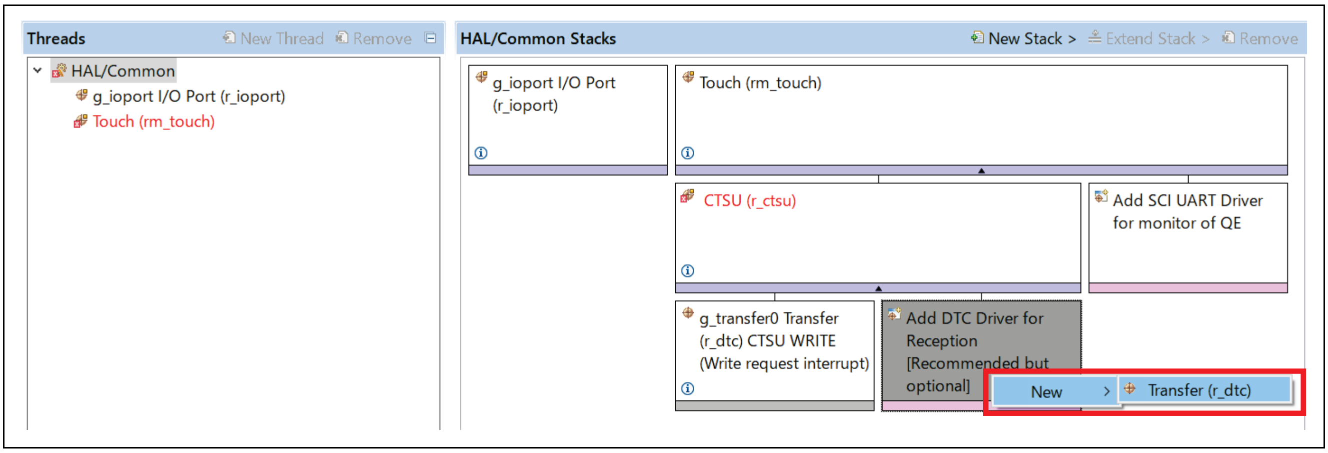

Figure 25. Add DTC Driver for Reception

19. Click the **TOUCH (rm_touch)** module to display **Properties**.

Note: Steps 19 to 26 should only be set when using serial communication for monitoring. Otherwise, skip to Step 27.

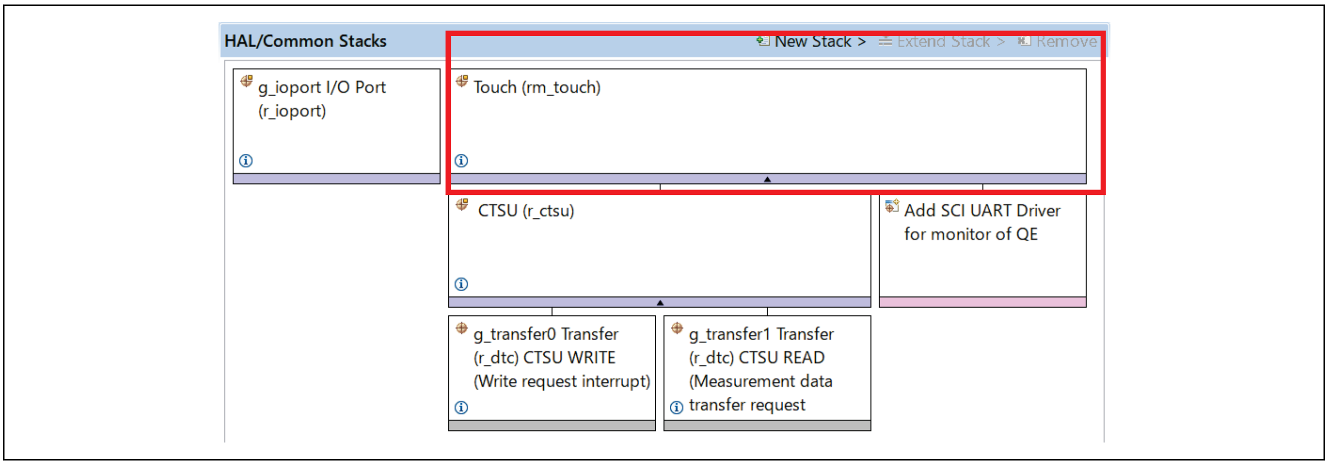

Figure 26. TOUCH Driver on rm_touch Module

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

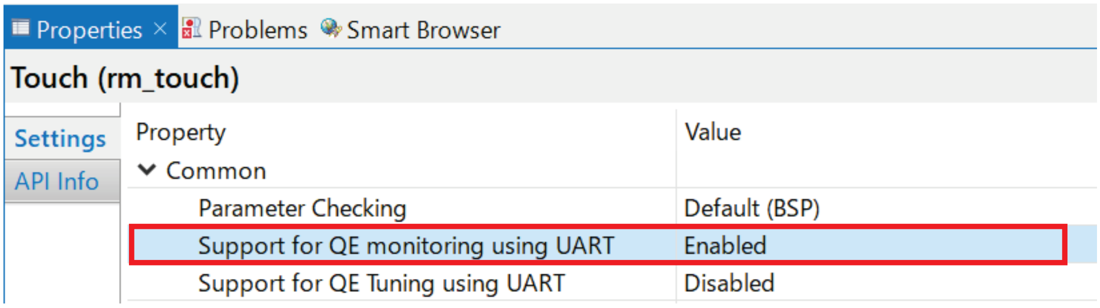

Figure 27. Support for QE Monitoring using UART

21. Click the **Add SCI UART Driver** module and select **New > UART (r_sci_uart)** to add the UART driver.

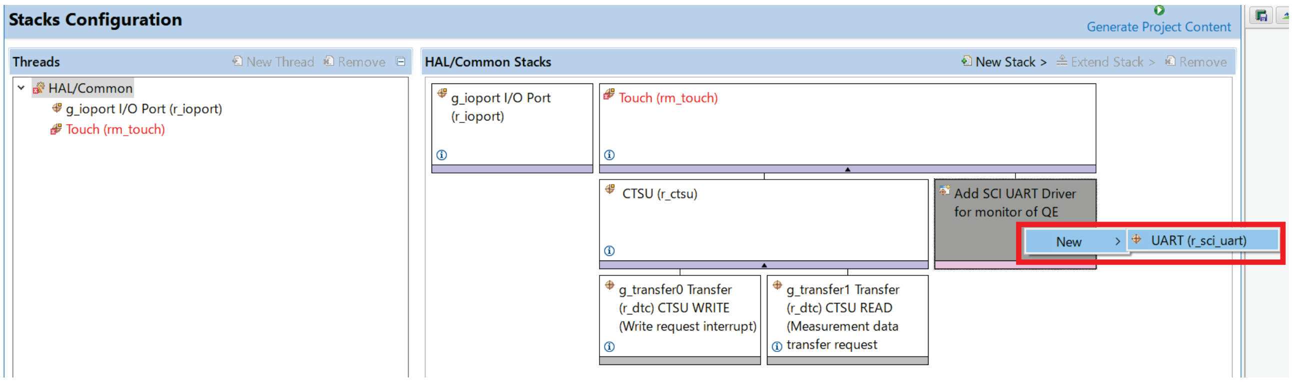

Figure 28. Add UART Driver

22. Click the **g_uart_qe UART Driver on r_sci_uart** module to display **Properties**.

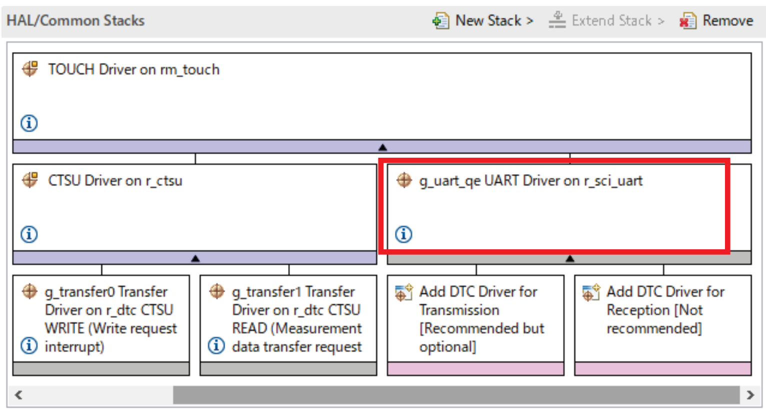

Figure 29. g_uart_qe UART Driver on r_sci_uart Module

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

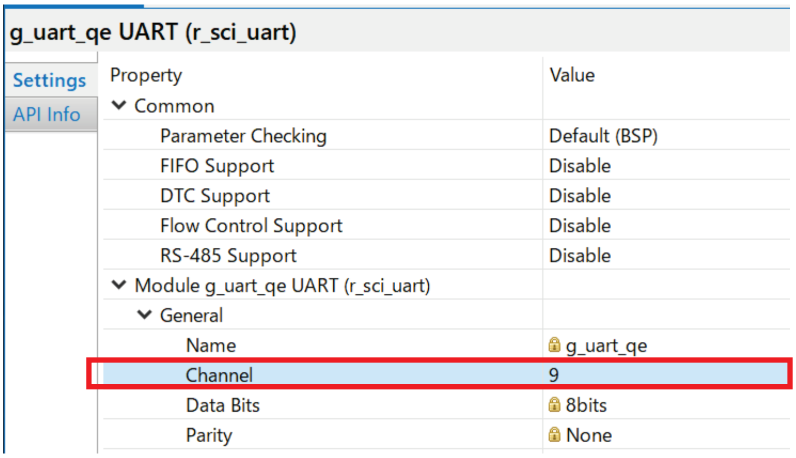

Figure 30. Channel

24. From **Properties** set the DTC as well. Select **Common > DTC Support** and change “Disable” to “Enable”.

    Note: Steps 24 to 26 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 27.

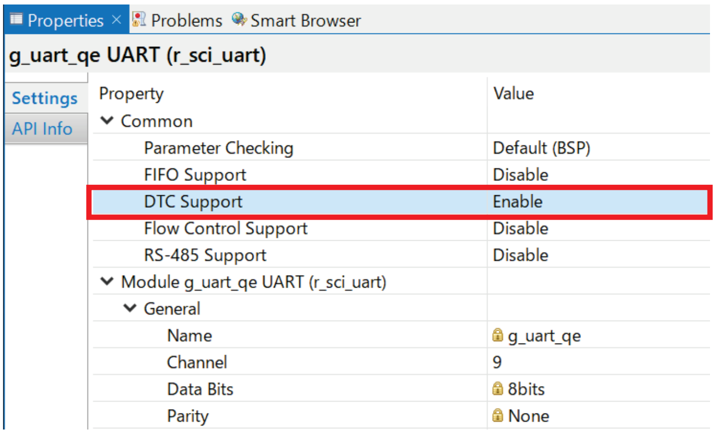

Figure 31. DTC Support

25. Click the **Add DTC Driver for Transmission** module and select **New > Transfer (r_dtc)** to add the DTC driver for transmission.

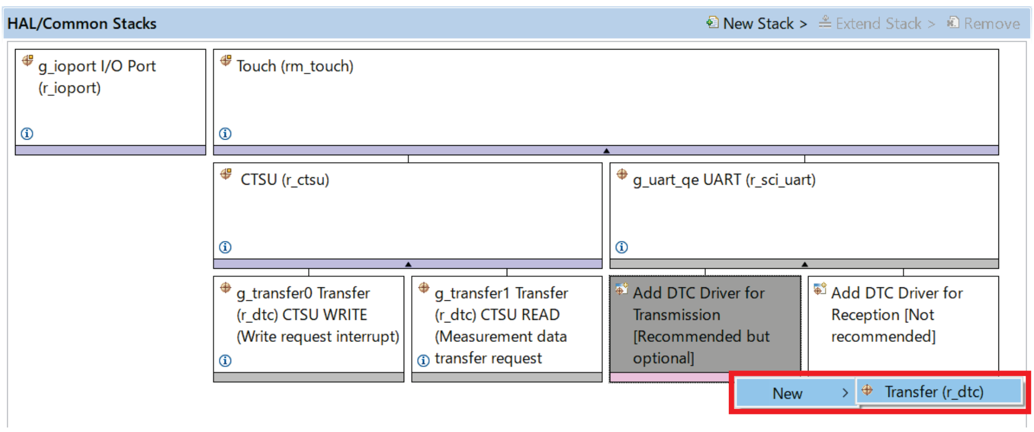

Figure 32. Add DTC Driver for Transmission (SC19)

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

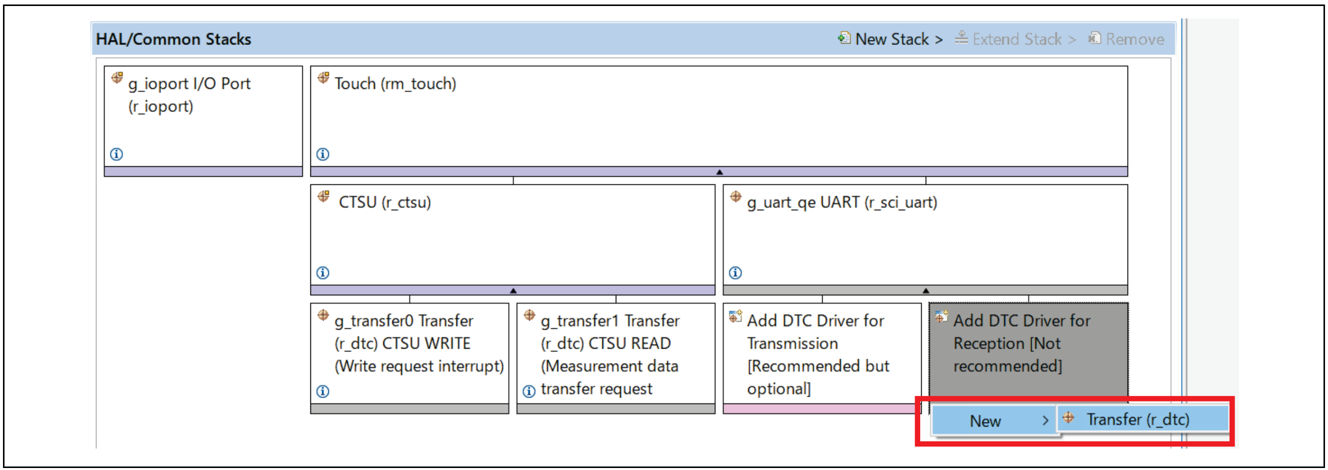

Figure 33. Add DTC Driver for Reception (SC19)

27. Select **New Stack >Timers > Timer, Low-Power (r_agt)** to add the AGT driver.

Note: Steps 27 to 33 should only be set when using an external trigger as the CTSU measurement start trigger. Otherwise (for example, when using a software trigger), skip to Step 34.

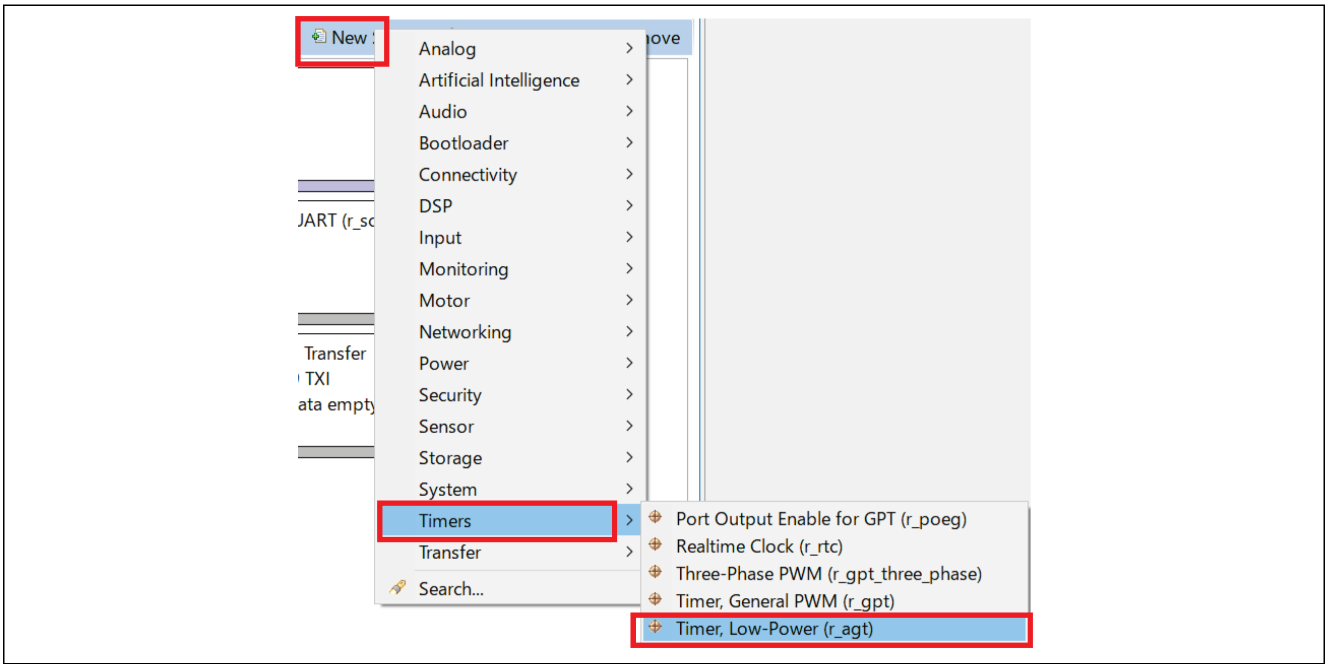

Figure 34. Add AGT Driver

28. Click the **g_timer0 Timer,Low-Power (r_agt)** module to display **Properties**.

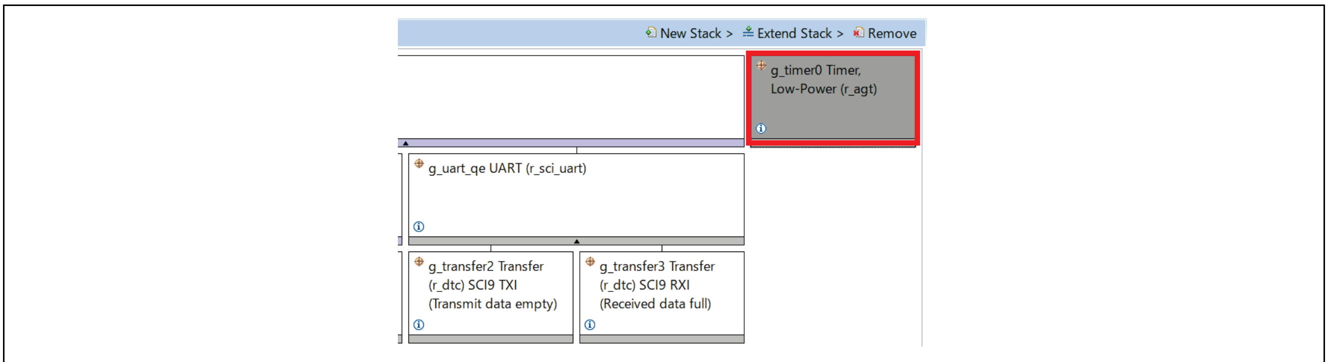

Figure 35. g_timer0 Timer Driver on r_agt Module

29. Set the AGT. At the bottom left of the screen, select **Properties > Settings > Module g_timer0 Timer Driver on r_agt > General > Period.** Enter “100” as the **Period** setting and select milliseconds as the unit.

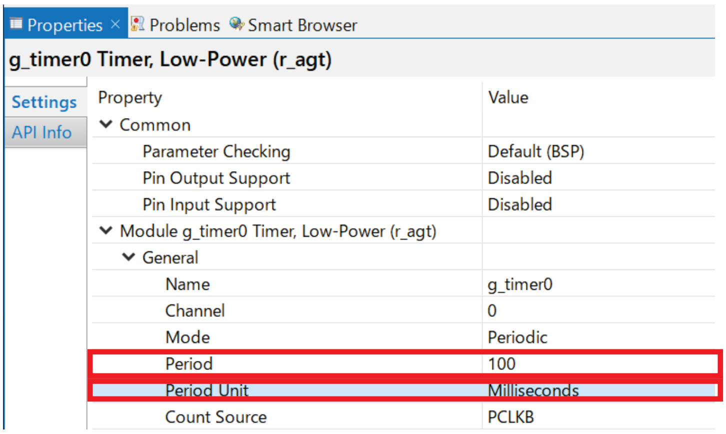

Figure 36. Period / Period Unit

30. Now set the AGT interrupts by selecting **Module g_timer0 Timer Driver on r_agt > Interrupts > Underflow Interrupt Priority** and setting the priority to anything other than “Disabled”. In the example, Priority 2 is selected, which generates an AGT0 interrupt every100 ms.

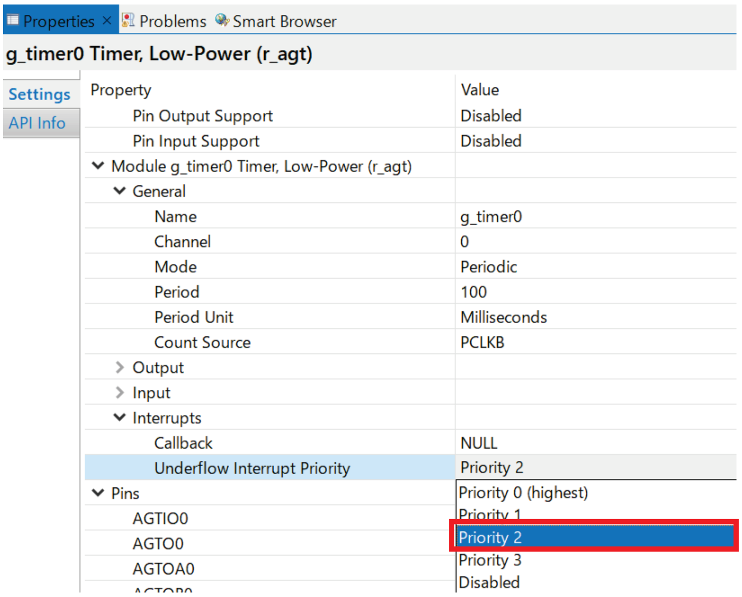

Figure 37. Underflow Interrupt Priority

31. Select **New Stack > System > Event Link Controller (r_elc)** to add the ELC Driver.

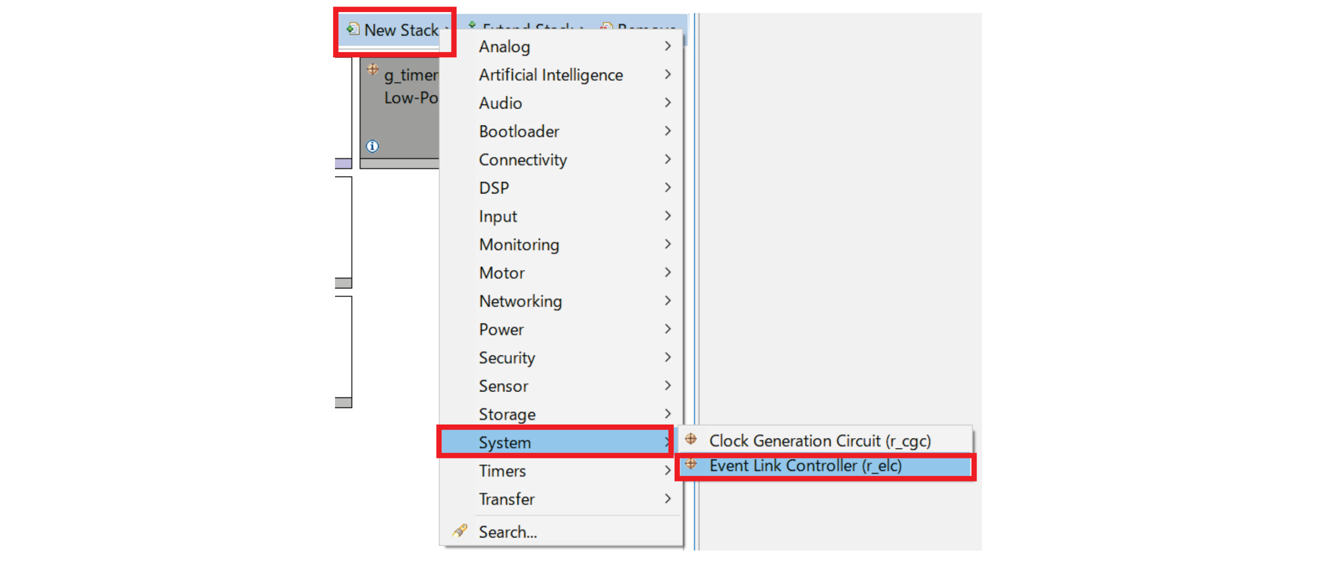

Figure 38. Add ELC Driver

32. Click the **CTSU Driver on r_ctsu** module to display **Properties**.

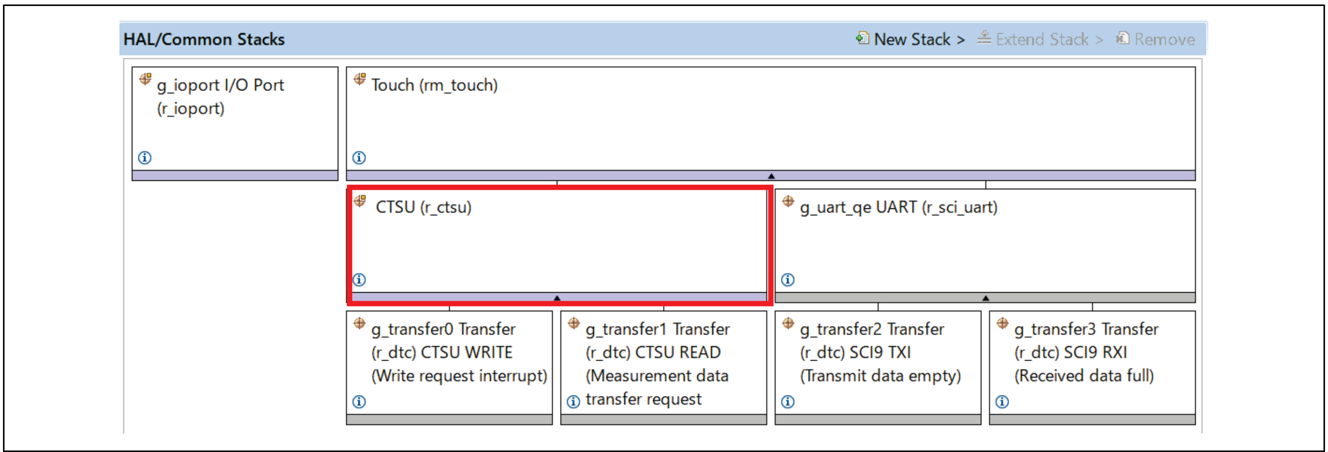

Figure 39. CTSU Driver on r_ctsu Module

33. Set the CTSU measurement start trigger. At the bottom-left of the screen, select **Properties > Settings -> Module CTSU Driver on r_ctsu > Scan Start Trigger** and change the trigger from "Software" to "AGT0 INT (AGT interrupt)". This setting makes the CTSU start the measurement with an AGT0 interrupt.

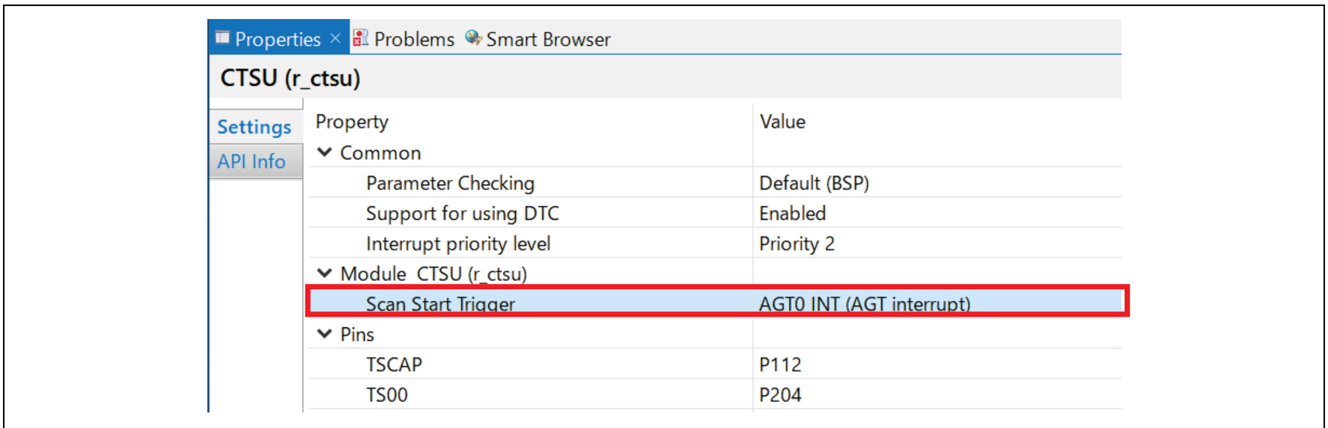

Figure 40. Scan Start Trigger

34. 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 by clicking **Generate Project Content** icon at the upper-right of the screen.

Figure 41. Generate Project Content

6.3 Creating the Capacitive Touch Interface

1. From the e2 studio IDE, use Renesas Views > Renesas QE > CapTouch Workflow (QE) to open the main perspective for configuring capacitive touch to the project.

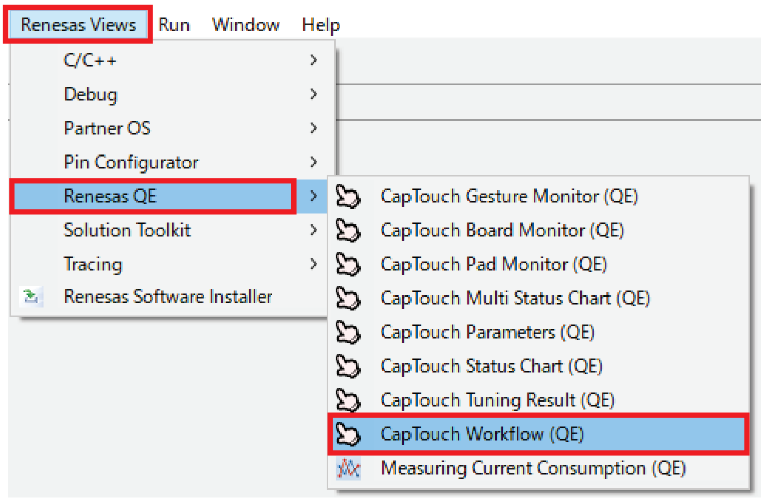

Figure 42. CapTouch Workflow (QE) Menu

2. In the **CapTouch Workflow (QE)** pane, select the project you want to configure for touch interface by using the pull-down tab and selecting “Capacitive_Touch_Project_Example” as shown below.

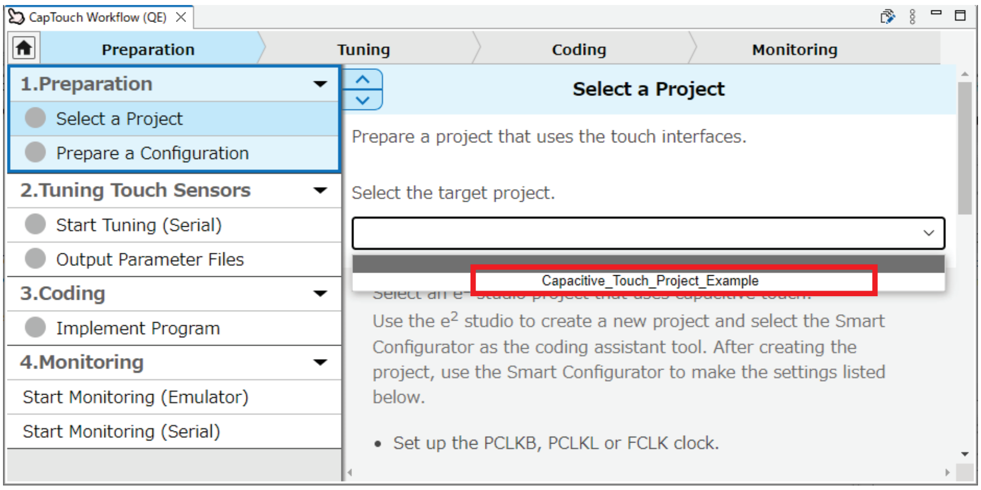

Figure 43. Select a Project

3. Next, select **Prepare a Configuration** in the menu on the left-hand side of "CapTouch Workflow (QE)" to display the items for setting. Select **Create a new configuration** from the pull-down menu to generate a new configuration for a touch interface.

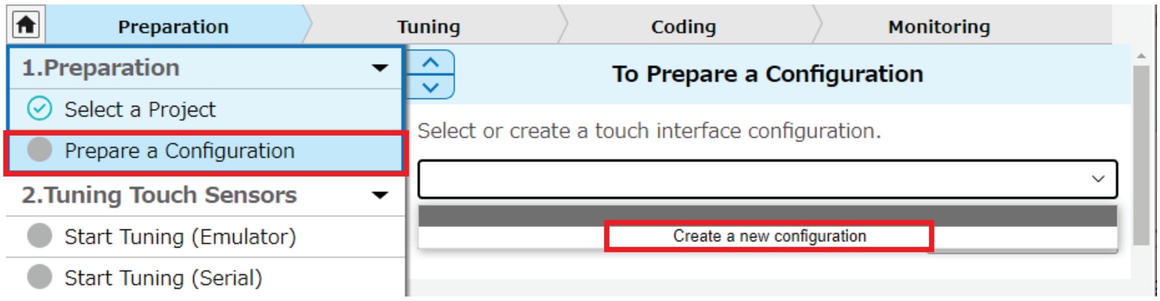

Figure 44. Select a Configuration

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

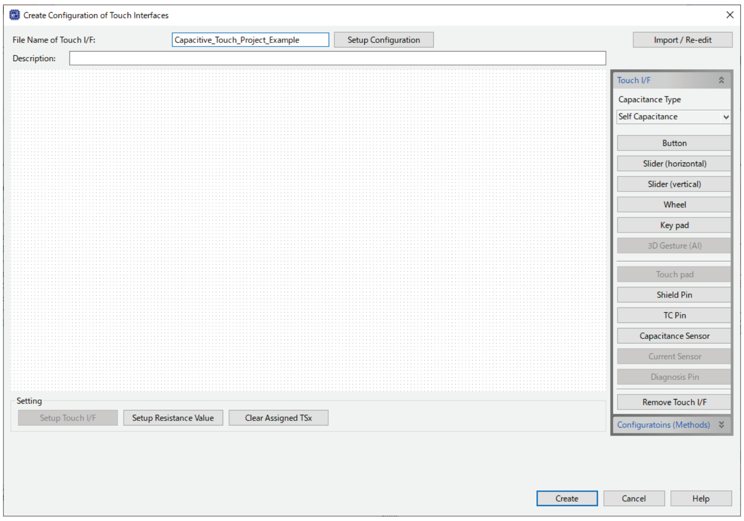

Figure 45. Create Configuration of Touch Interfaces Dialog Box

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 remains red, as shown in the figure below, indicating that it has not been assigned a touch sensor yet.

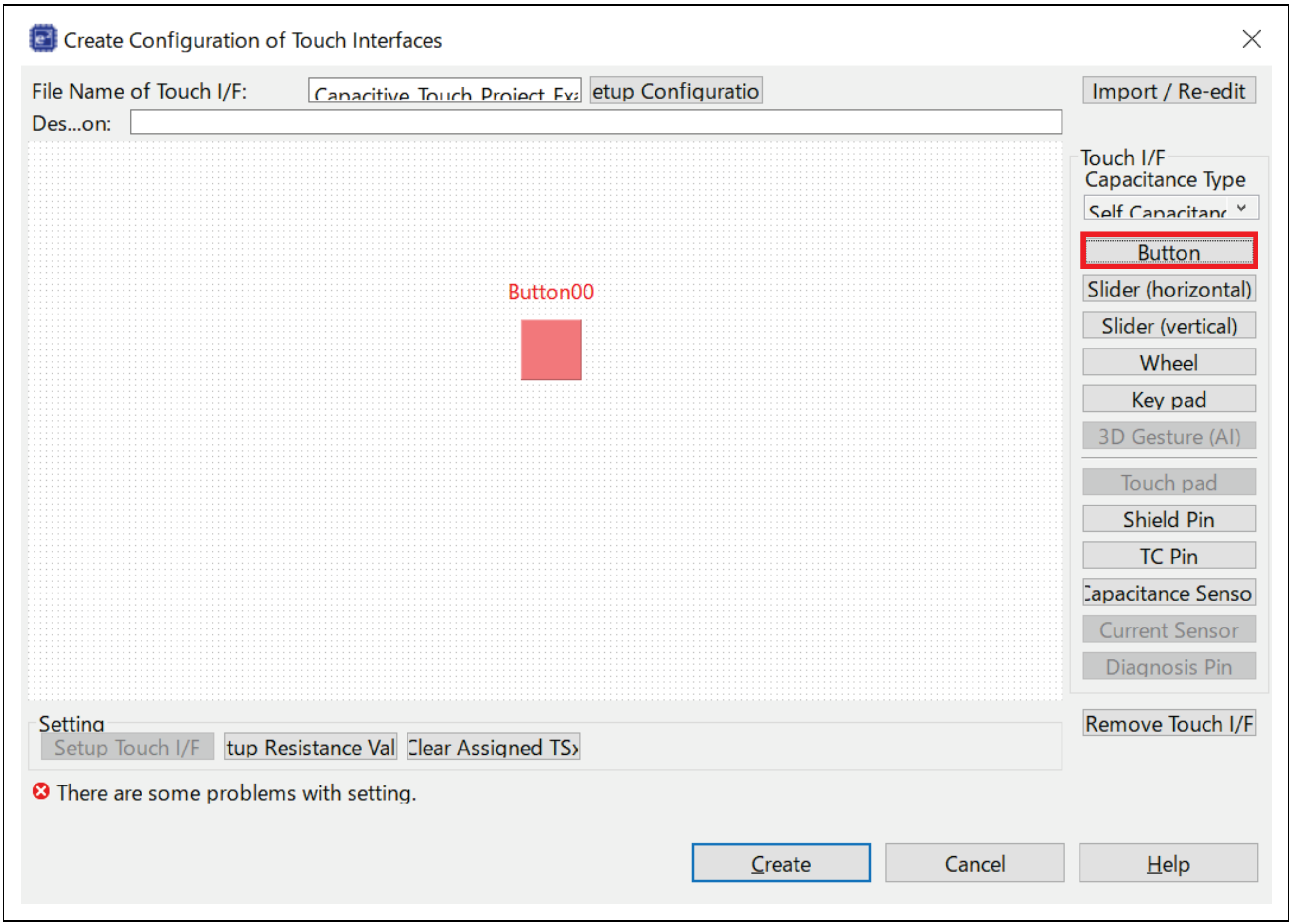

Figure 46. Add a Button

6. Double click **Button00** in the touch interface canvas to display the **Setup Touch Interface** dialog box. From the pull-down menu, select the MCU sensor port you want to assign to this button. When a sensor port that has been enabled by the RA Smart Configurator is correctly assign to the button, the setting error disappears and the button turns green, indicating that it has been set.

Item	RA6M2 (CTSU)	RA2L1 (CTSU2)
Touch sensor to be selected	TS02	TS11

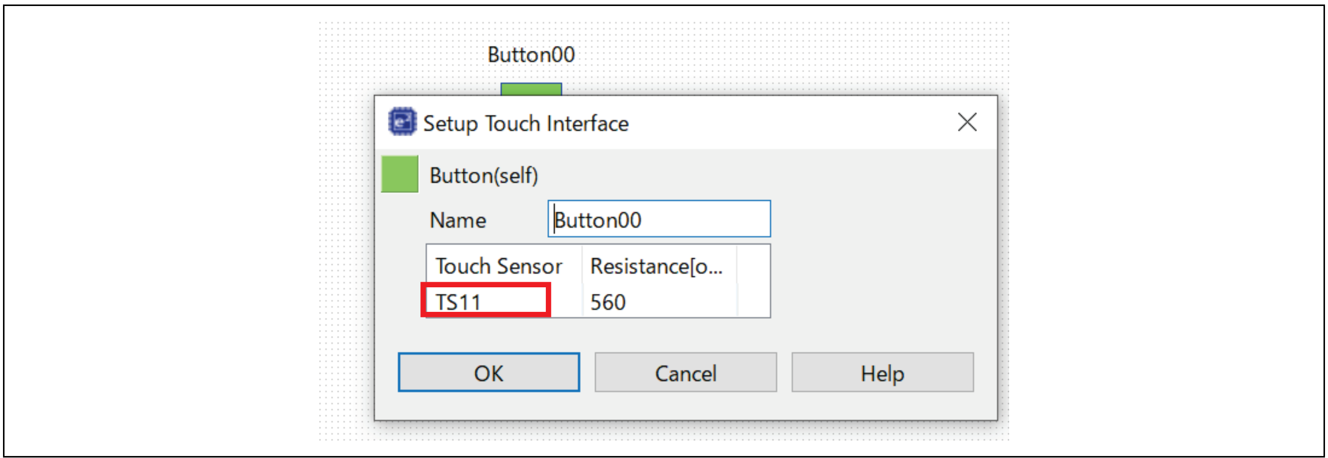

Figure 47. Setup Touch Interface (set button)

7. Next, on the right side of the **Create Configuration of Touch Interfaces** dialog box, select **Shield Pin** and then click on the canvas to add a shield pin. Press the ESC key on your keyboard or select **Shield Pin** again to complete the step. The added shield pin remains red, as shown in the figure below, indicating it has not been assigned a touch sensor yet.

   Note: When creating a project, if "EK-RA6M2" (RA6M2 MC Group Evaluation Kit) is selected in Device Selection > Board, the Shield Pin is disabled and cannot be selected.

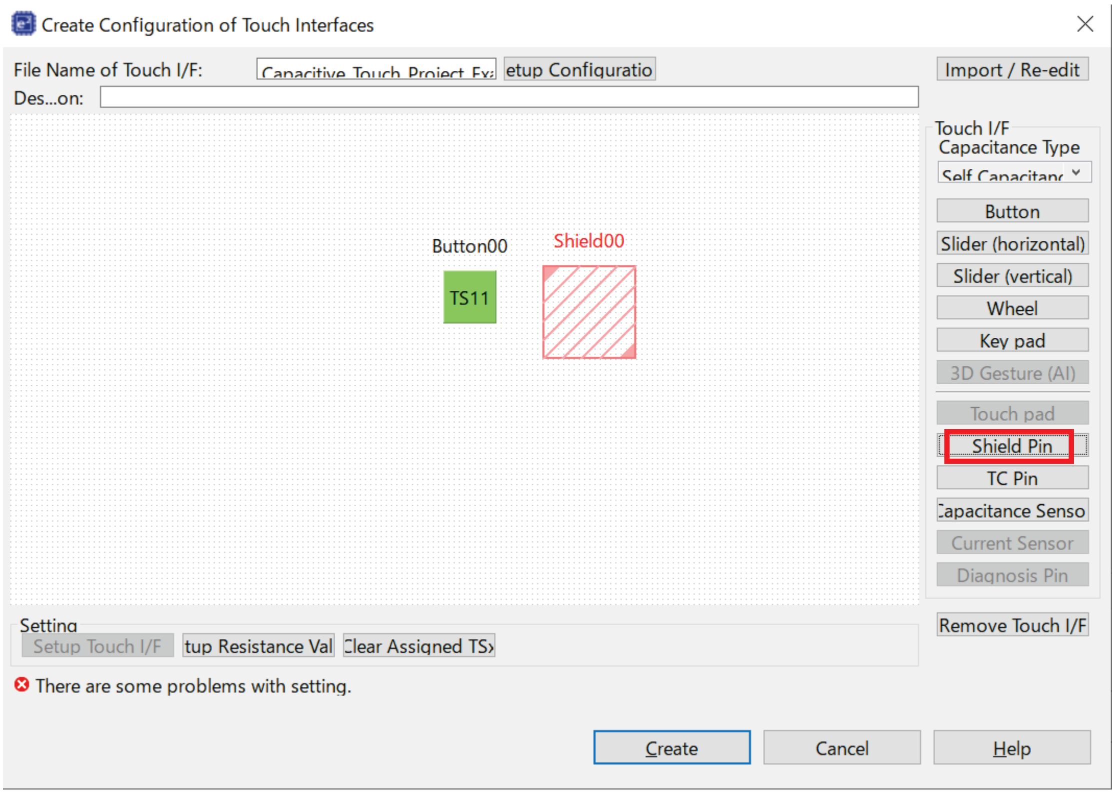

Figure 48. Add Shield Pin

8. Double click "Shield00" in the touch interface canvas to open the Setup Touch Interface dialog box. From the pull-down menu, select the MCU sensor port to assign to this shield pin. When a sensor port that has been enabled by the RA Smart Configurator is correctly assign to the pin, the setting error disappears and the button turns green, indicating that it has been set.

Item	RA6M2 (CTSU)	RA2L1 (CTSU2)
Touch sensor to be selected	- (Cannot be set for EK-RA6M2)	TS00

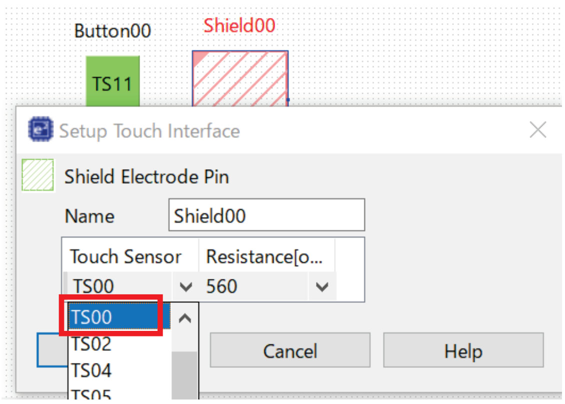

Figure 49. Setup Touch Sensor (Shield Pin)

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

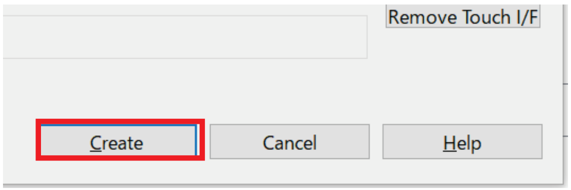

Figure 49. Create Button

10. Select **Renesas Views** >**Renesas QE** >**CapTouch Tuning Result (QE)** from the menu of the e² studio and open the **CapTouch Tuning Result (QE)** view to display the configuration of the touch interface in the **Tuning** panel.

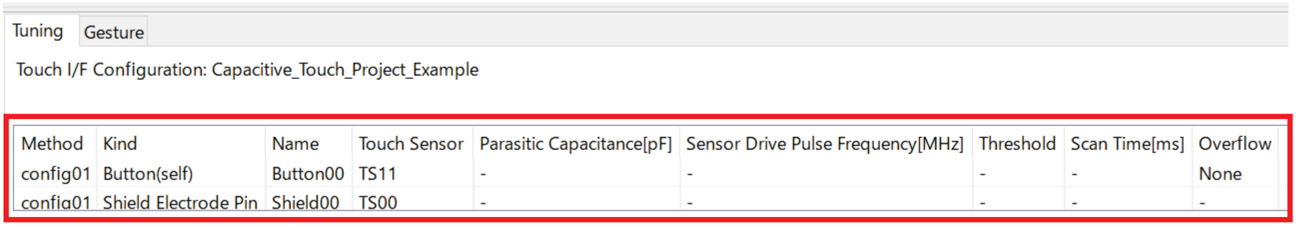

Figure 50. Tuning Pane

11. Build the project using the hammer icon in the upper left-hand side of the e² studio screen. The project should build without any errors or warnings.

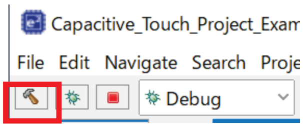

Figure 51. Build Button

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 drop-down list button of the bug icon from the e² Studio Tools button.

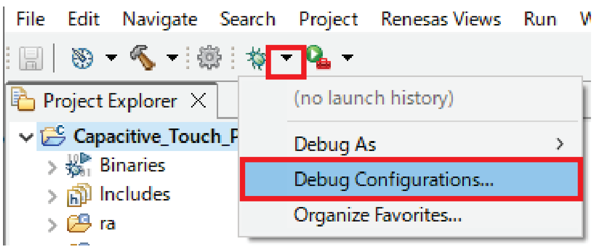

Figure 52. Debug Configuration Editing Button

2. Select the **Startup** tab.

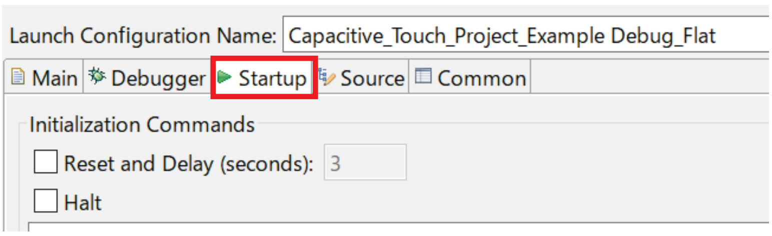

Figure 53 Startup Tab

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

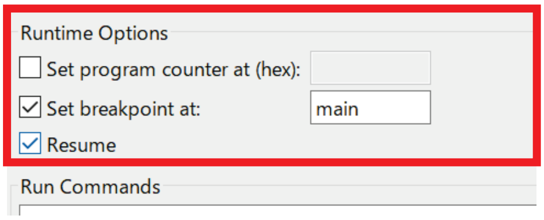

Figure 54. Runtime Option

4. Click **Apply** 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 Plug-in

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

2. Select **Start Tuning (Emulator)** in the menu on the left-hand side of "CapTouch Workflow (QE)" to open the settings of "Tuning Touch Sensors". Click on **Start Tuning** to start automatic tuning.

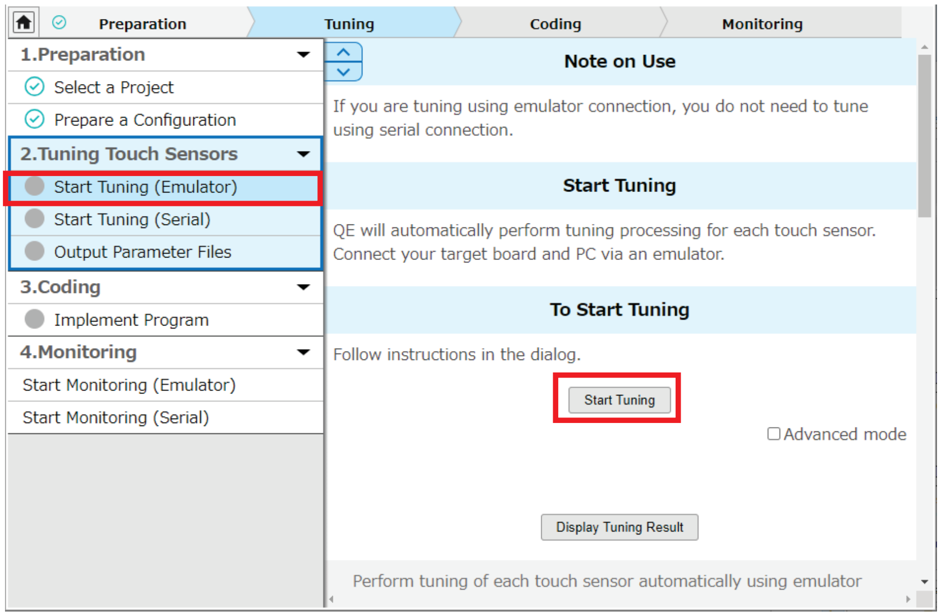

Figure 55. Start Tuning

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 automatic tuning.

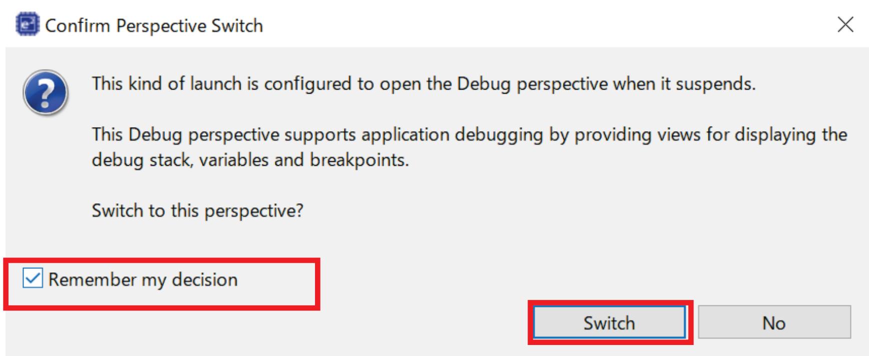

Figure 56. Confirm Perspective Setting

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

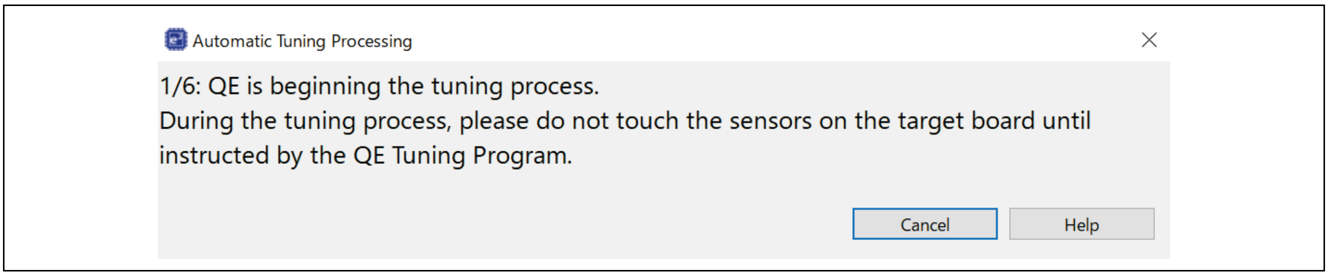

Figure 57. Automatic Tuning Processing Dialog (now preparing)

5. After several automated steps, a dialog box with information similar to what is shown below appears. This is the touch sensitivity measurement step of the tuning process. As the first interactive step of the tuning process, press “Button00” (sensor) using normal touch pressure as indicated in the dialog box (RA6M2: TS02, RA2L1: TS11-CFC). When pressing the sensor, the bar graph increases to the right and the touch counts increase numerically. While continuing to apply pressure to the sensor, press any key on the PC keyboard to accept the measurement.

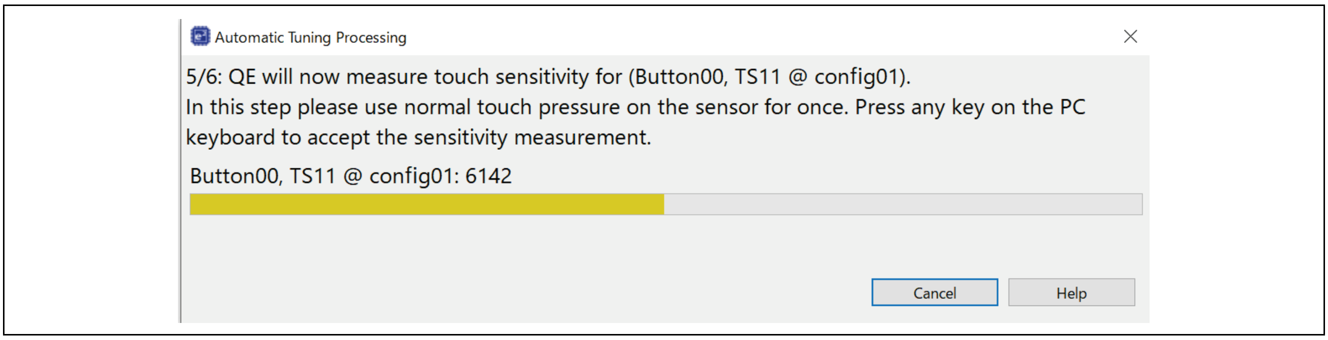

Figure 58. Automatic Tuning Processing Dialog (now measuring)

6. Once sensitivity measurement for the button is complete, a dialog like the following appears allowing threshold confirmation. This is the detection threshold that is used by the middleware to determine if a touch event has occurred.

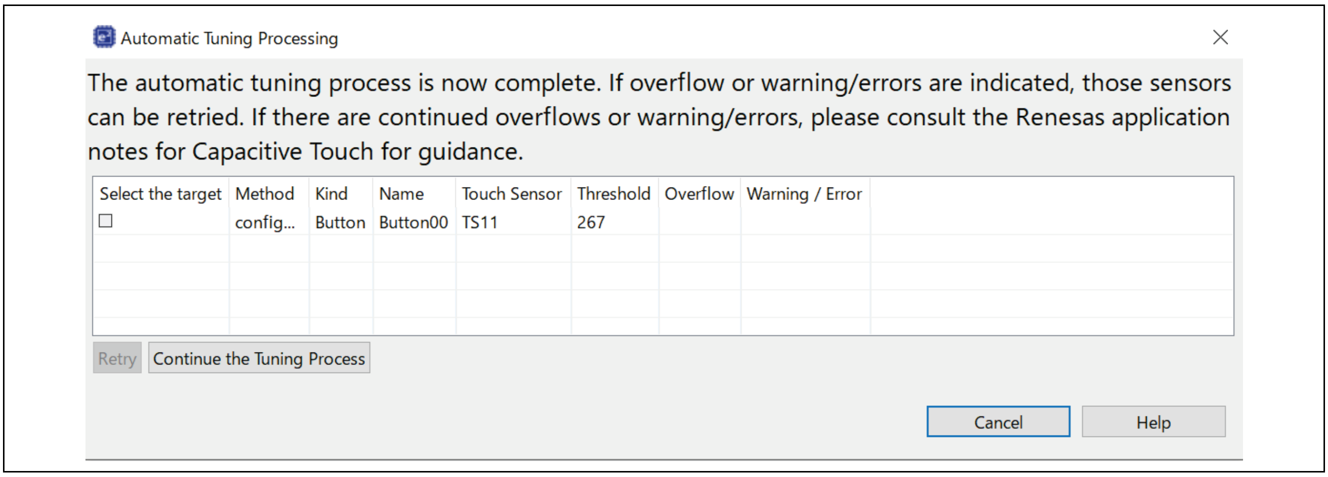

Figure 59 Automatic Tuning Processing (tuning complete)

7. Click the **Continue the Tuning Process** button in the dialog box shown. This exits the tuning process and disconnects from the Debug session on the target board. This should return you to the default **Cap Touch Workflow (QE)** screen in the e² studio IDE.

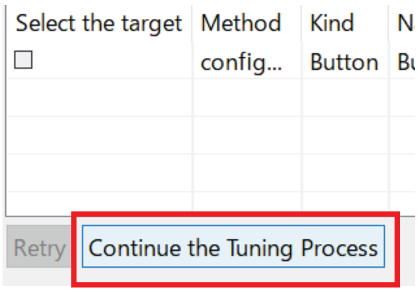

Figure 60. Continue the Tuning Process Button

8. The only remaining step is the output of the tuned parameter files. Select **Output Parameter Files** from the menu on the left-hand side of "CapTouch Workflow (QE)", and click on **Output Parameter Files**.

   Note: Carry out this step if you are NOT using an external trigger (that is, when you are using a software trigger) to start the CTSU measurement.

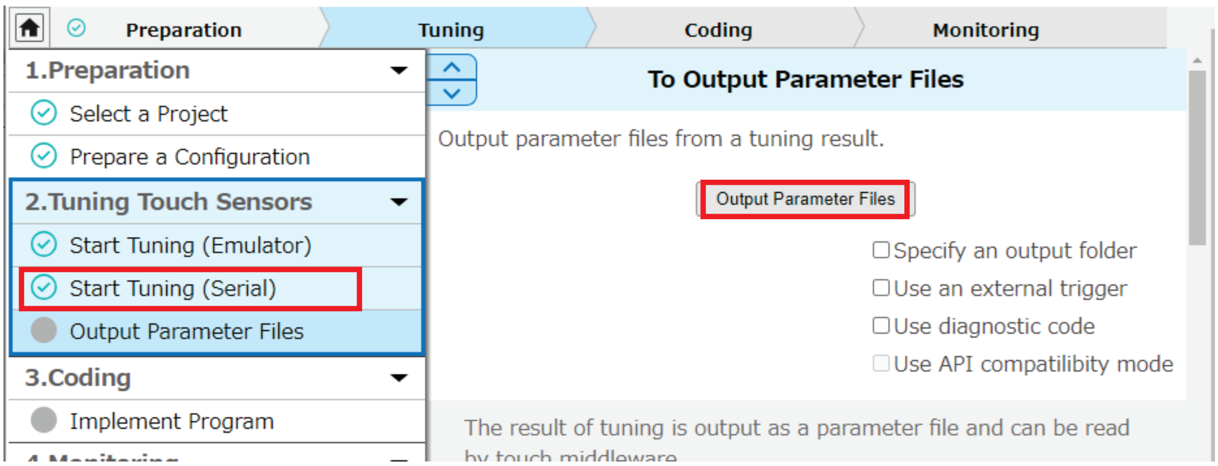

Figure 61. Output Parameter Files

9. When using an external trigger to start the CTSU measurement, Check the **Use an external trigger** box and click on **Output Parameter Files**.

   Note: Carry out this step if you are using an external trigger to start the CTSU measurement process.

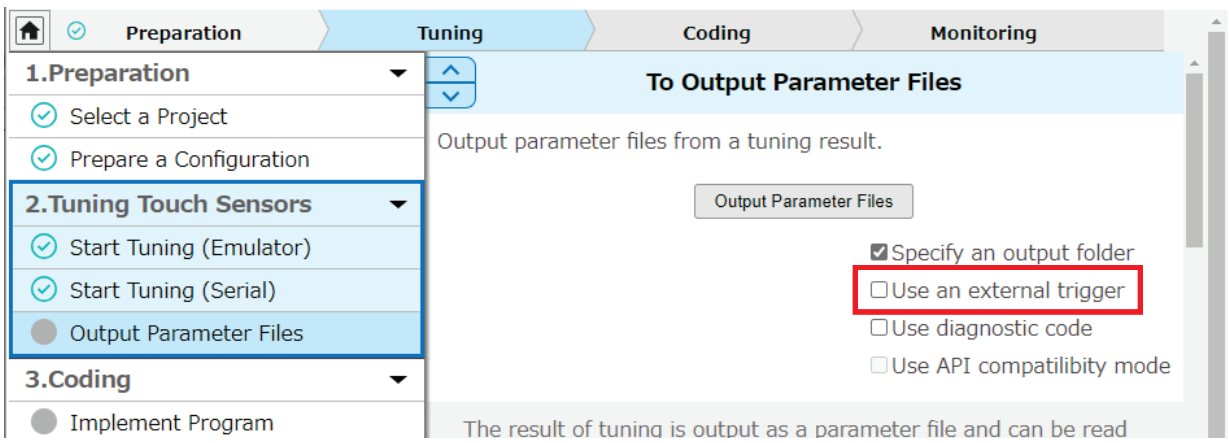

Figure 62. Parameter File Output Settings

10. In the **Project Explorer** window, 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.

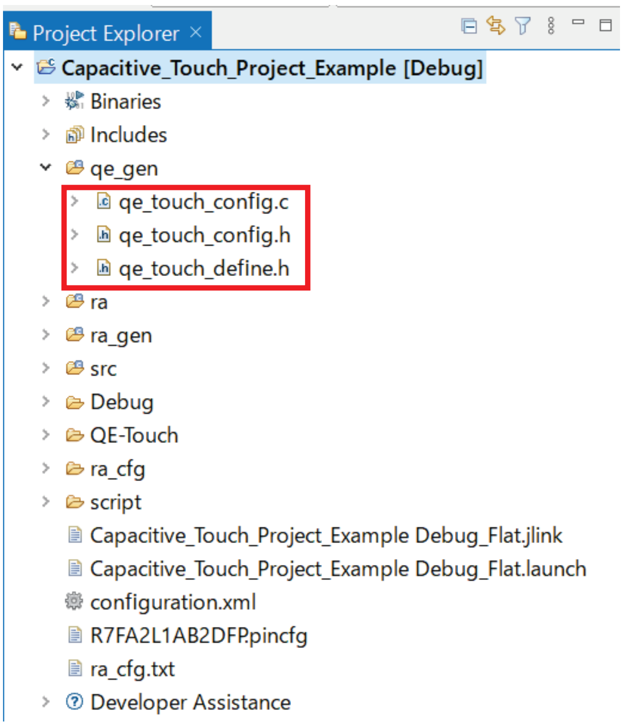

Figure 63. Parameter Files

11. 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.

Note: A build error is generated if monitoring via serial communication is enabled in Flexible Software Package (FSP) V3.0.0. See https://github.com/renesas/fsp/issues/78 for more details.

6.6 Adding rm_touch middleware API Calls to Application Example

1. To implement a program (code) to scan the states of the touch sensors, select **Implement Program** in the menu on the left-hand side of "CapTouch Workflow (QE)", and then click on **Show Sample**.

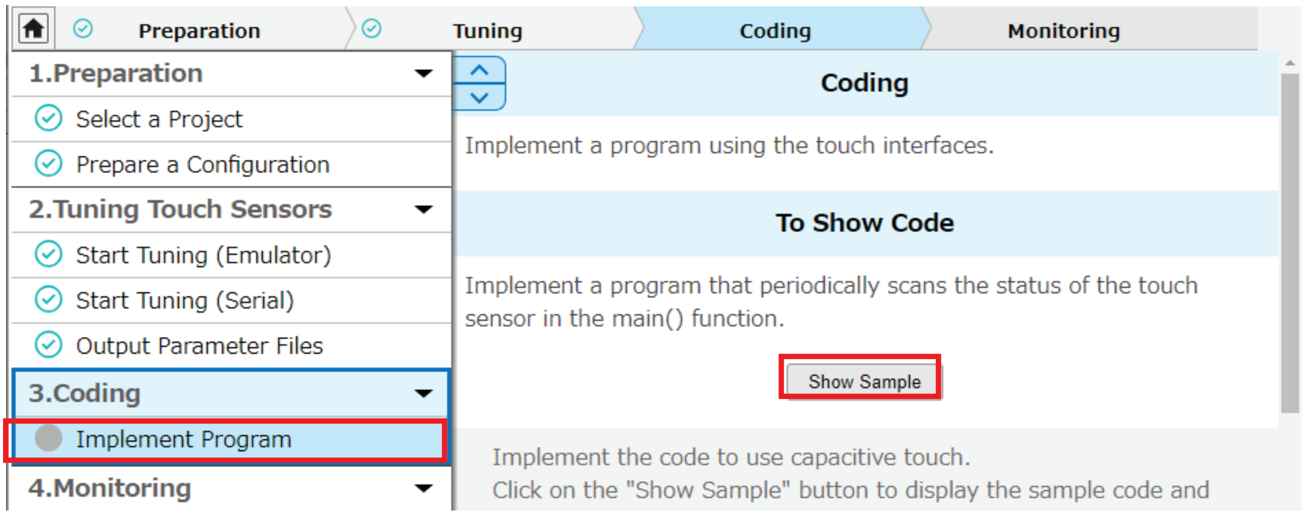

Figure 64. Show Sample Code

2. The **Show Sample Code** window opens and 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.

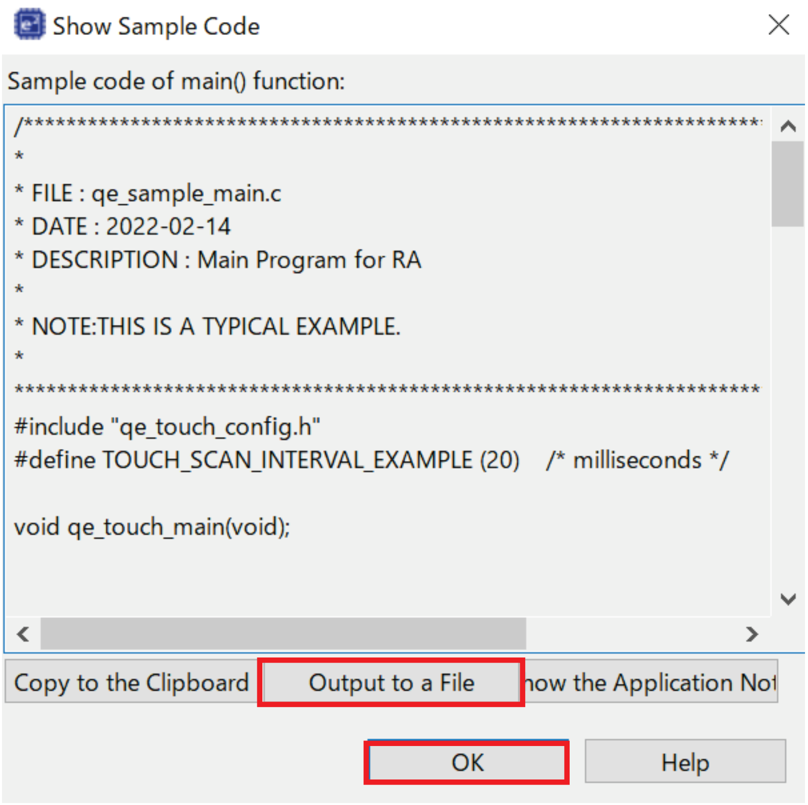

Figure 65. Show Sample Code Window

3. You can confirm that the new qe_touch_sample.c file has been created in **Project Explorer**.

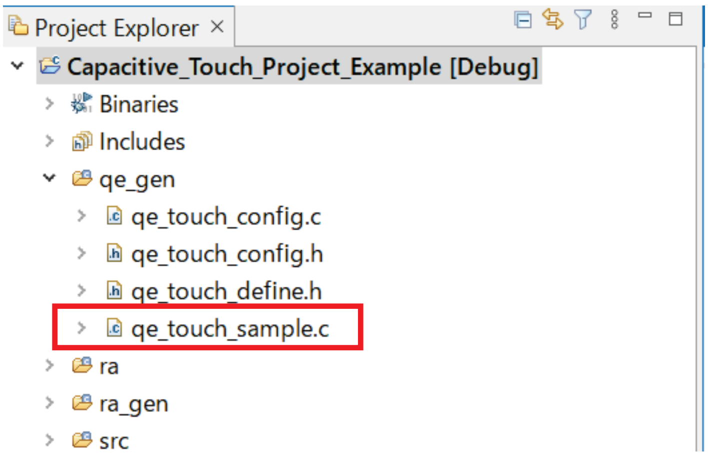

Figure 66. qe_touch_sample.c

4. In **Project Explorer**, select **ra_gen** and double click on main.c to open the file. Make sure **hal_entry** is selected in the main() function and right-click on **Open Declaration**. This opens the hal_entry() function declaration.

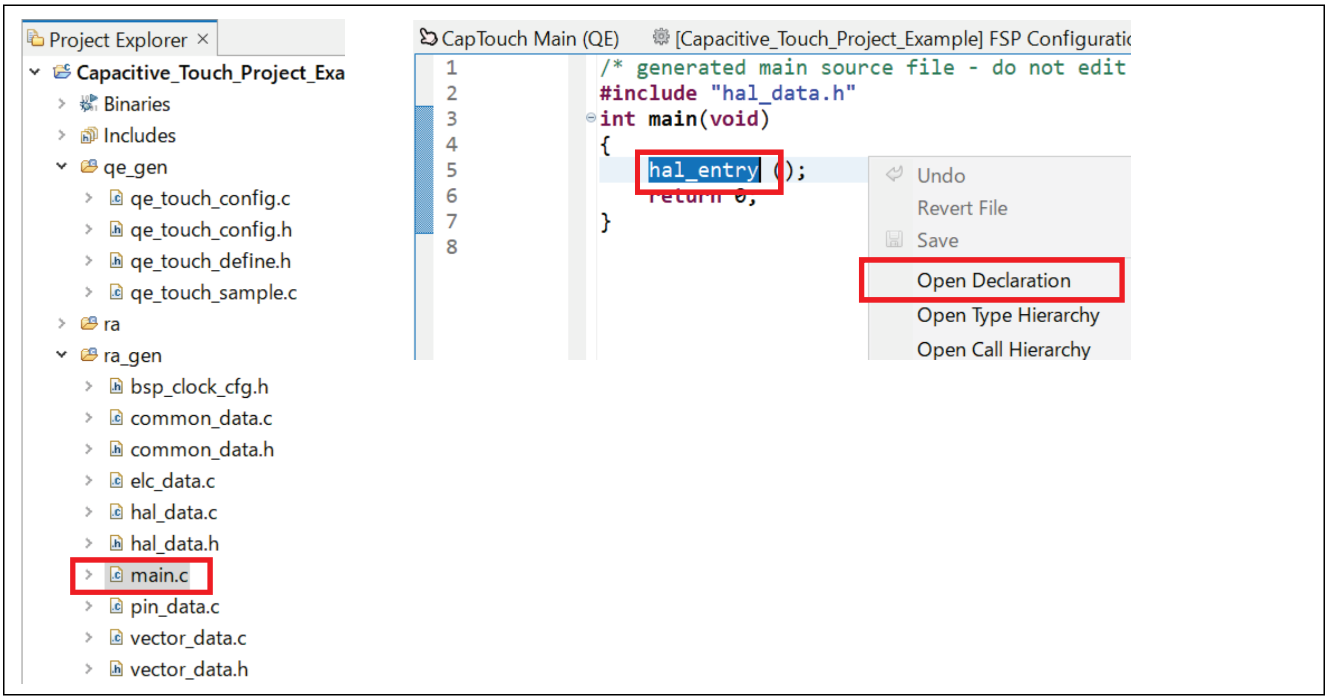

Figure 67. Open Declaration (hal_entry function)

5. In the **hal_entry()** function, add the code to call the **qe_touch_main()** function and the prototype declaration as shown in the image below.

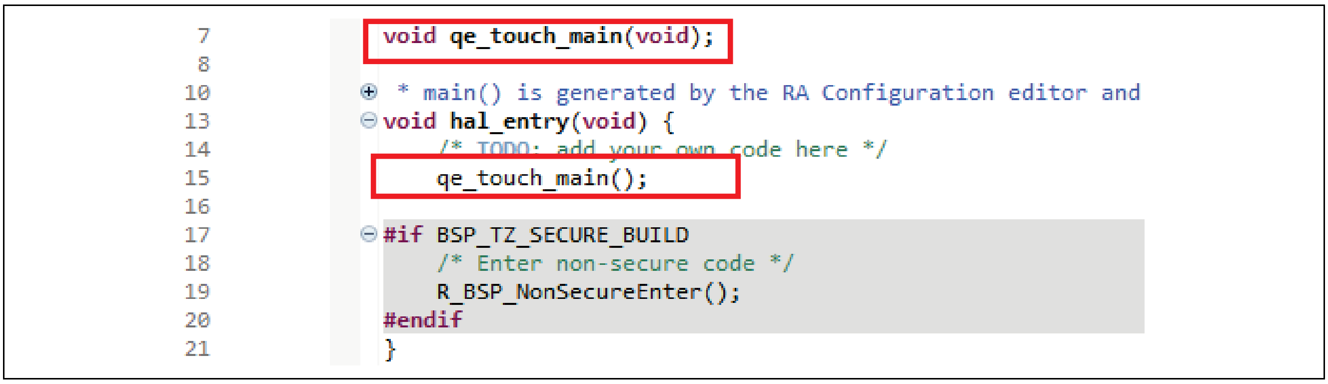

Figure 68. Function Call (qe_touch_main function)

6. In the **hal_entry()** function, select the **qe_touch_main()** function, then click on **Open Declaration** to open the qe_touch_main () declaration.

   Note: Carry out Steps 6 and 7 only if you are using an external trigger to start the CTSU measurement process. If you are NOT using an external trigger (that is, when you are using a software trigger), skip to Step 8.

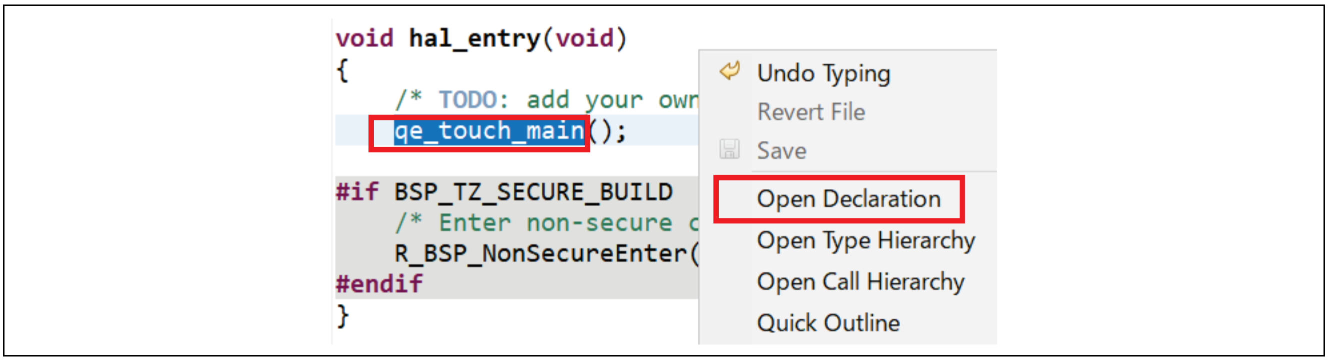

Figure 69. Open Declaration (qe_touch_main function)

7. When using an external trigger to start the CTSU measurement process, enable the code after commenting out the API that controls the AGT driver of the qe_touch_main().

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

    /* Initializes the software and sets the links defined in the control structure. */
    R_ELC_Open(g_elc.p_ctrl, g_elc.p_cfg);
    /* Globally enable event linking in the ELC. */
    R_ELC_Enable(g_elc.p_ctrl);

    /* 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) {}
    }

    /* Open AGT driver */
    R_AGT_Open(g_timer0.p_ctrl, g_timer0.p_cfg);
    /* Start AGT. */
    R_AGT_Start(g_timer0.p_ctrl);

    /* Main loop */
    while (true)
    {
```

Figure 70. Code Modifications

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

6.7 Monitoring Touch Performance Using e^{2}studio Expressions Window and QE for Capacitive Touch

1. Start a Debug session by clicking the Bug icon in the upper left-hand corner of e^2 studio.
2. The debugger stops at the hal_entry () function call. This is the first code point in the main() function.
3. Open the declaration of hal_entry () function.

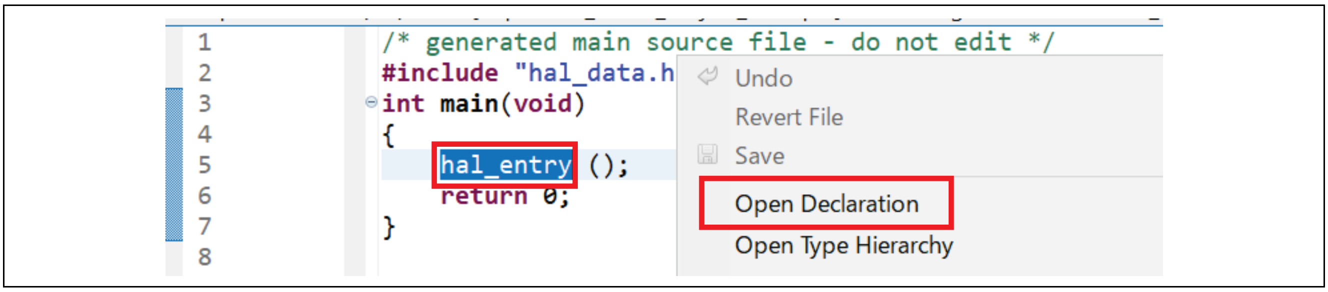

Figure 71. Open Declaration (hal_entry function)

4. Open the declaration of the **qe_touch_main()** function.

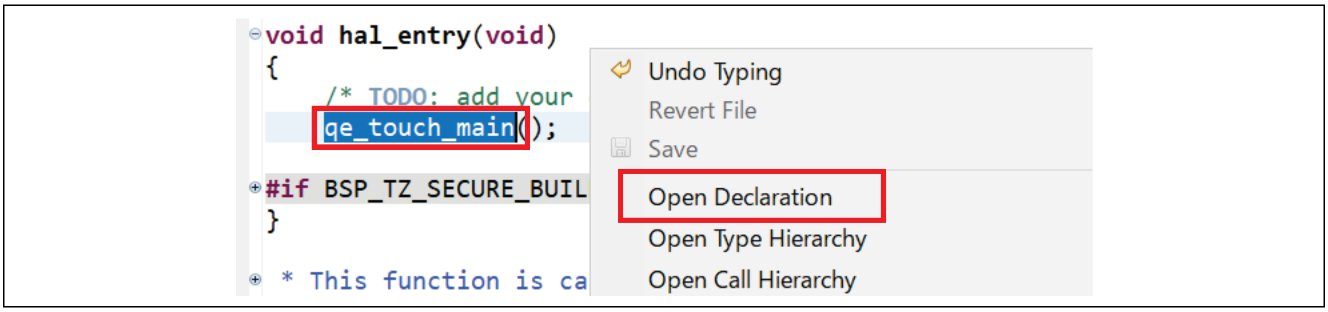

Figure 72. Open Declaration (qe_touch_main function)

5. Scroll down in the qe_touch_main.c file to the **RM_TOUCH_DataGet ()** function in the **while (true) loop**. With the argument "**button_status**" selected, right click and select **Add Watch Expression...** This opens the **Add Watch Expression** dialog box, where you can add the variable "**button_status**" to the expressions window.

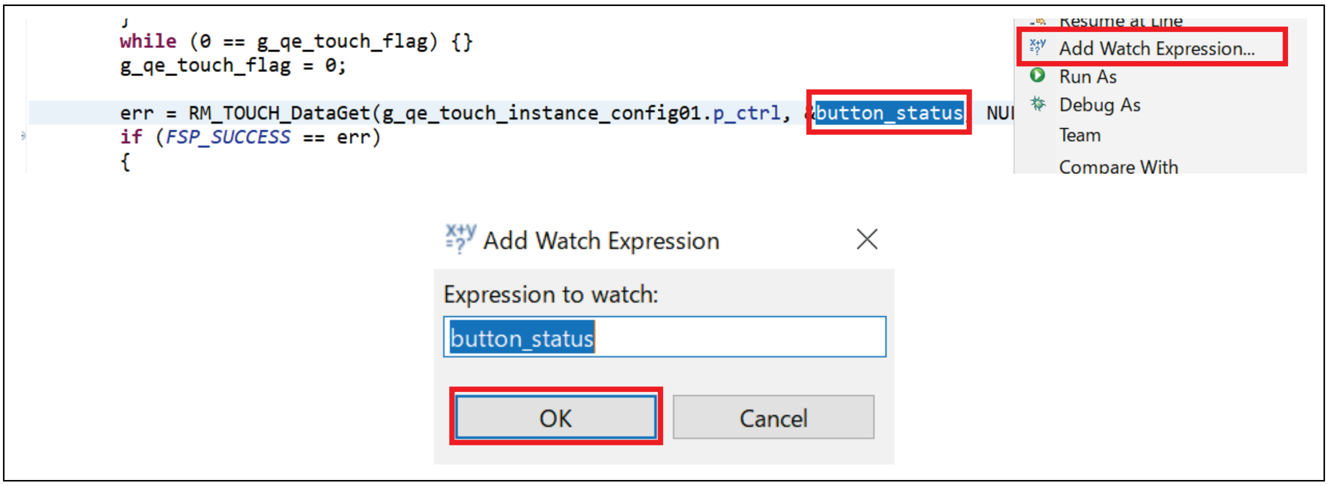

Figure 73. Add Watch Expression... Menu / Dialog

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

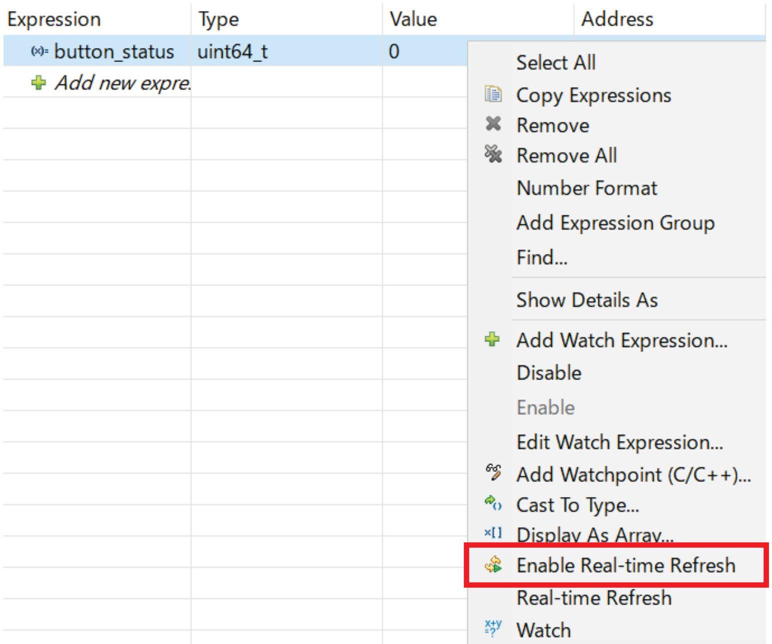

Figure 74. Enable Real-time Refresh Menu

7. Click the **Resume** (green arrow) button, located near the middle of the e^2 studio tool bar, to continue code execution.

8. Press the sensor on the board (RA6M2: TS02, RA2L1: TS11-CFC) which was configured as "Button00" in section 6.3, **Creating Capacitive Touch Interface**. When pressed, the "button_status" value in the Expressions window becomes '1', confirming the touch with a binary indication.

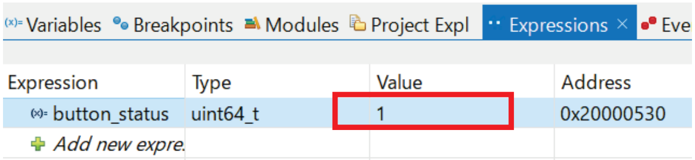

Figure 75. Touch Status Confirmation in Expressions Window

9. Select **Start Monitoring (Emulator)** in the menu on the left-hand side of "CapTouch Workflow (QE)". Click on **Show Views** to start **CapTouch Board Monitor (QE)**.

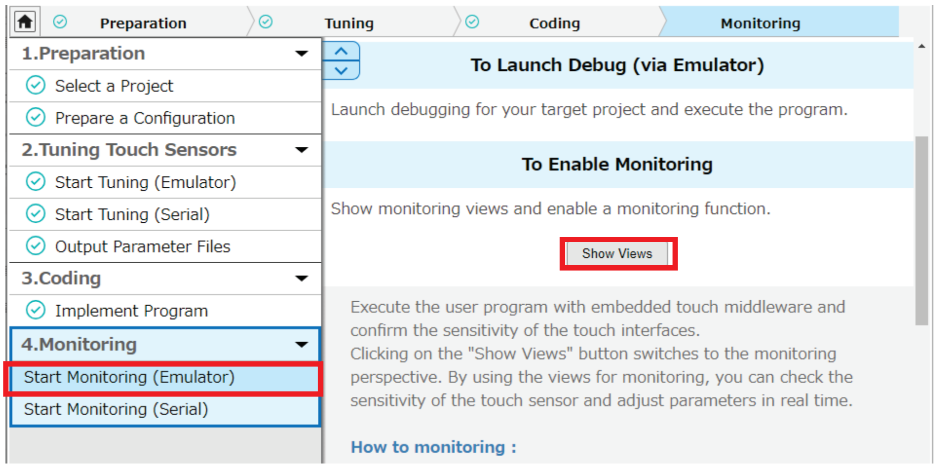

Figure 76. Start Monitoring

10. The **CapTouch Board Monitor** pane should appear as shown in the image below.

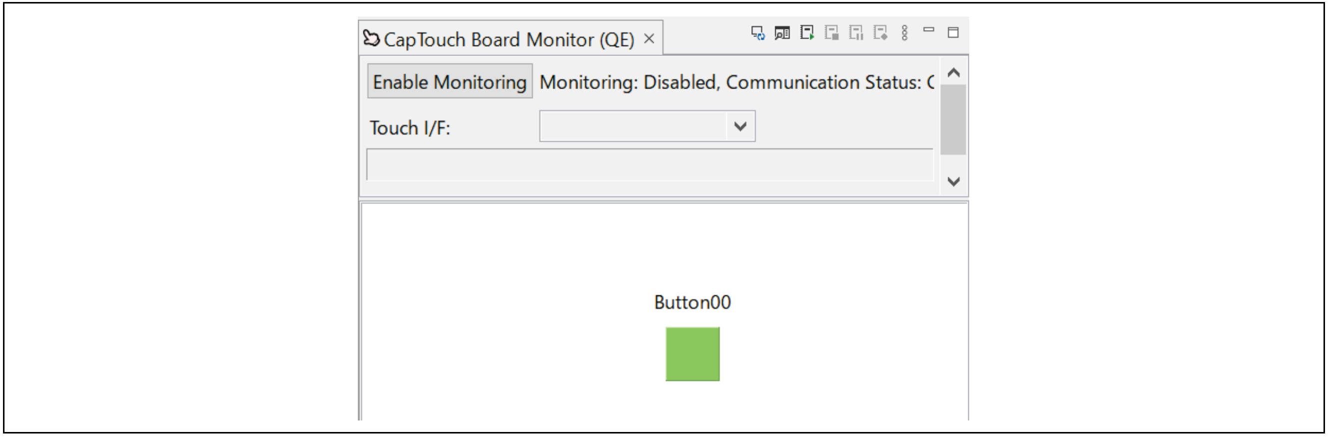

Figure 77. CapTouch Board Monitor Window (RA6M2)

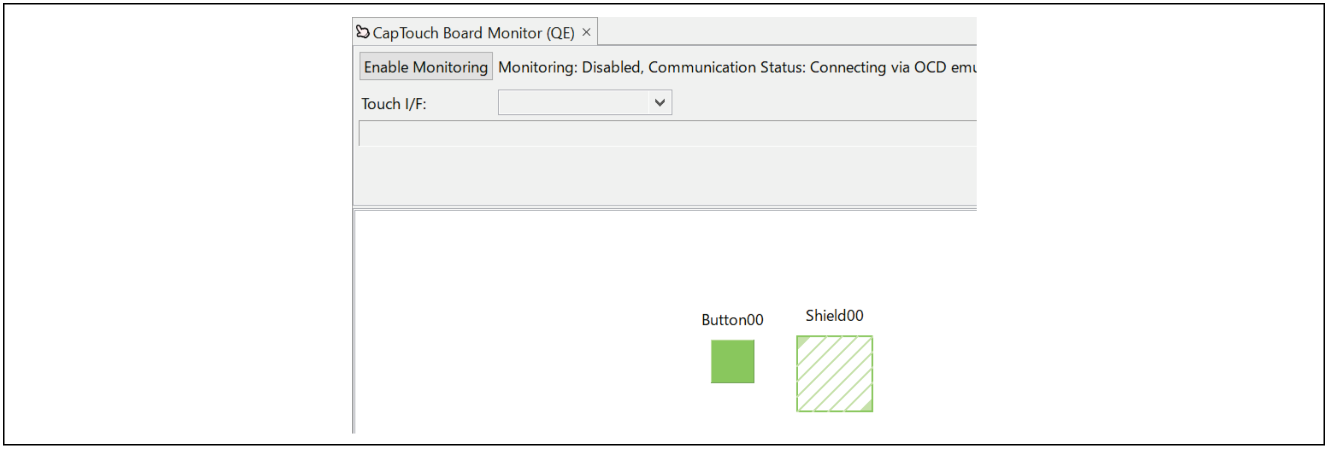

Figure 78. CapTouch Board Monitor Window (RA2L1)

11. Click the **Enable Monitoring** button. The dialog text changes from "Monitoring: Disabled" to "Monitoring: Enabled."

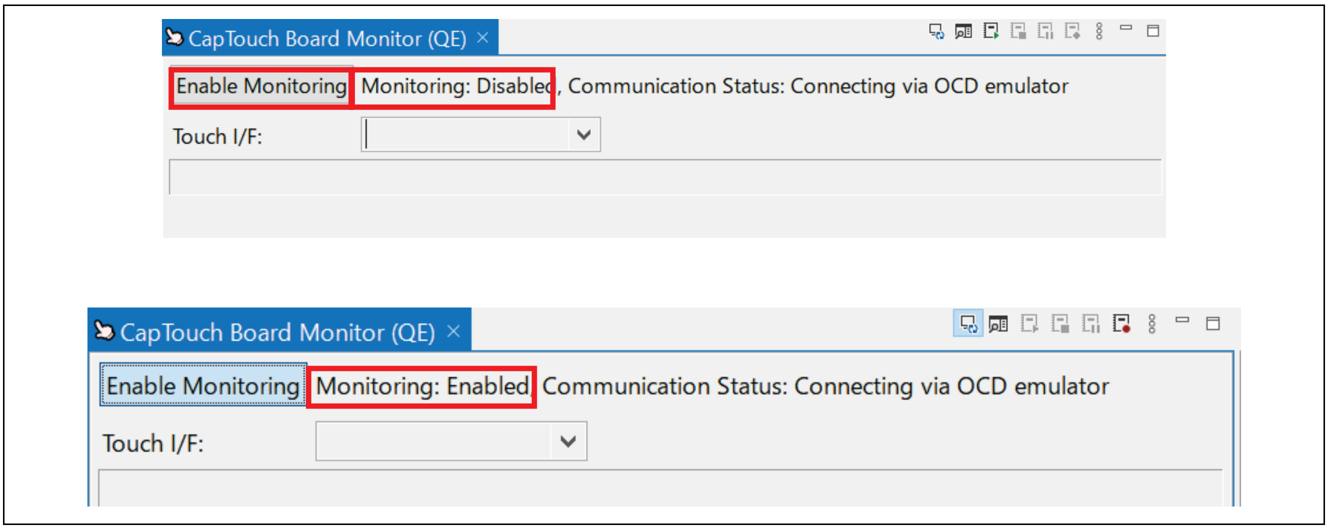

Figure 79. Enable Monitoring Function

12. Touch "**Button00**" (RA6M2: TS02, RA2L1: TS11-CFC) on the Capacitive Touch Application board. The **CapTouch Board Monitor** shows a touch with a finger image on the button like the following image.

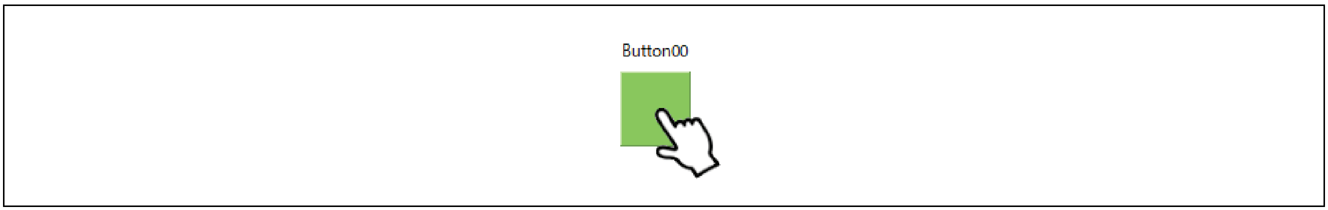

Figure 80. Touch Status Confirmation in CapTouch Board Monitor Window

13. To see a graphical representation of the 'touch counts' from the board, use the **CapTouch Status Chart.**

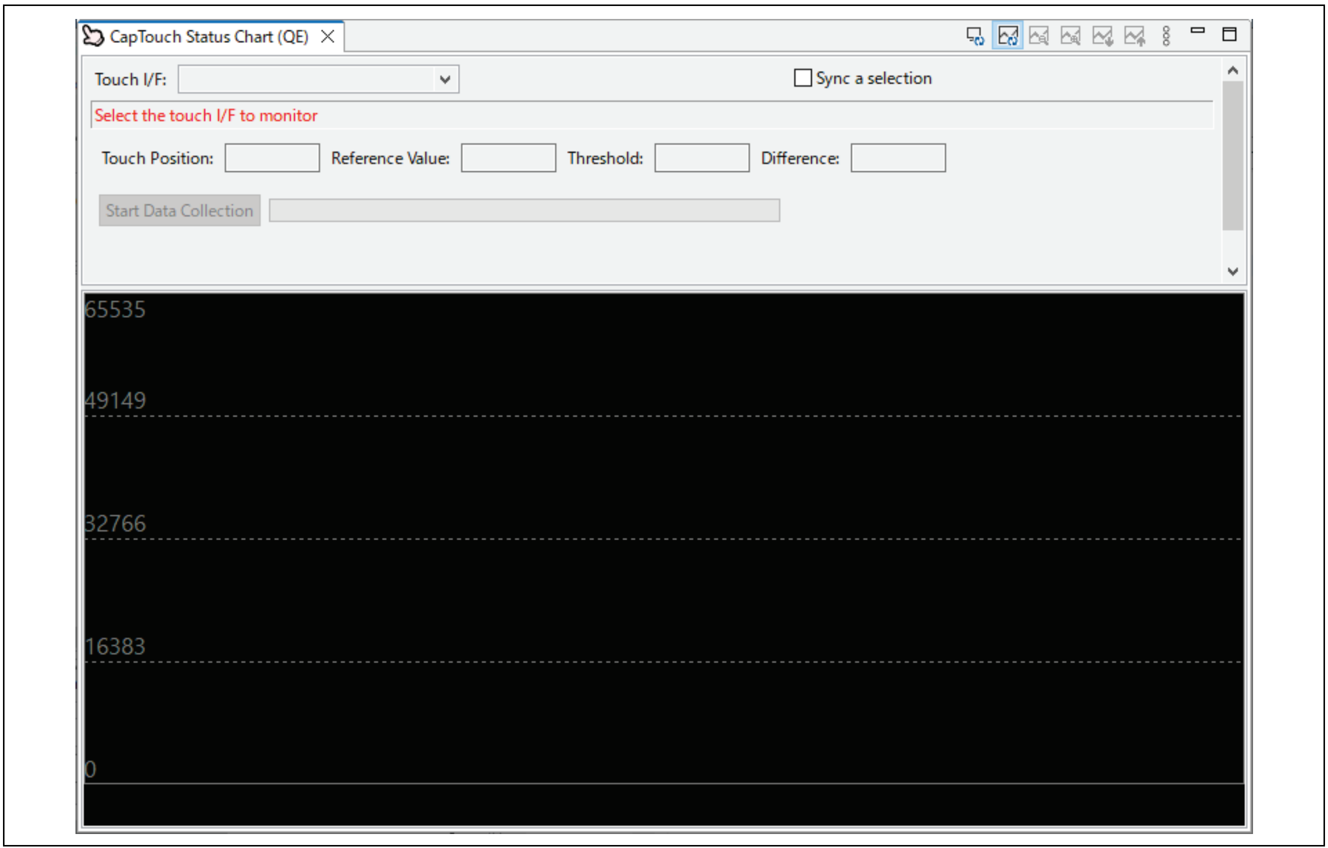

Figure 81. CapTouch Status Chart Window

14. Using the **Touch I/F** pull-down menu, select "**Button00 @ config01**".

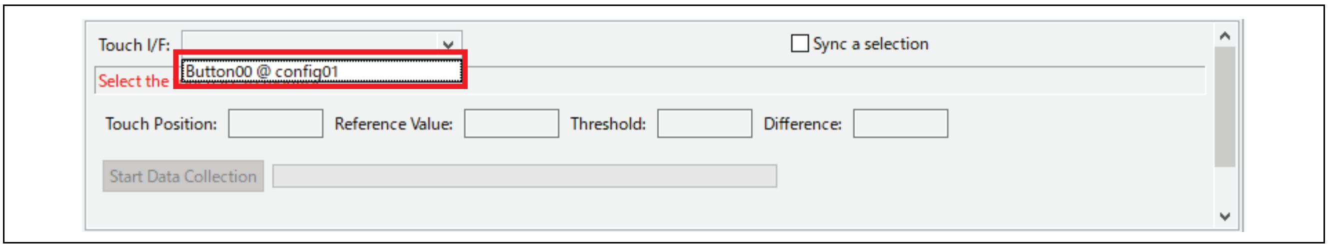

Figure 82. Select Touch I/F

15. The graph begins to display running values. Touch “Button00” (RA6M2: TS02, RA2L1: TS11-CFC) 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 rm_touch 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 83. Confirm Touch Status in CapTouch Status Chart Window

Note: Steps 16 to 19 must be set when displaying and measuring standard deviation.

16. Next, measure the standard deviation. Click on [Start Data Collection] displays the [Data Collection Settings] dialog box. Select the number of data to be collected from the pull-down menu in this dialog box. Select [Non Touch + Touch] for [Data collection target] and click on [Start Data Collection].

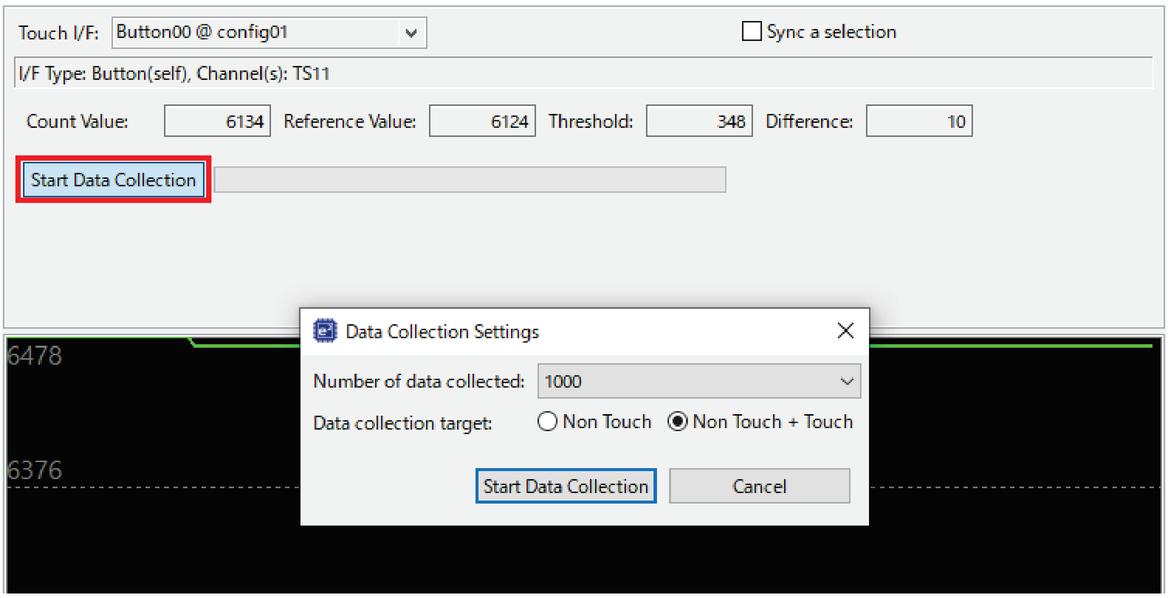

Figure 84. Start Data Collection Button (touch-off state)

17. When the green bar reaches the right edge, collection of the touch-off data is completed and a dialog box is displayed. Click on [OK] to close the dialog box.

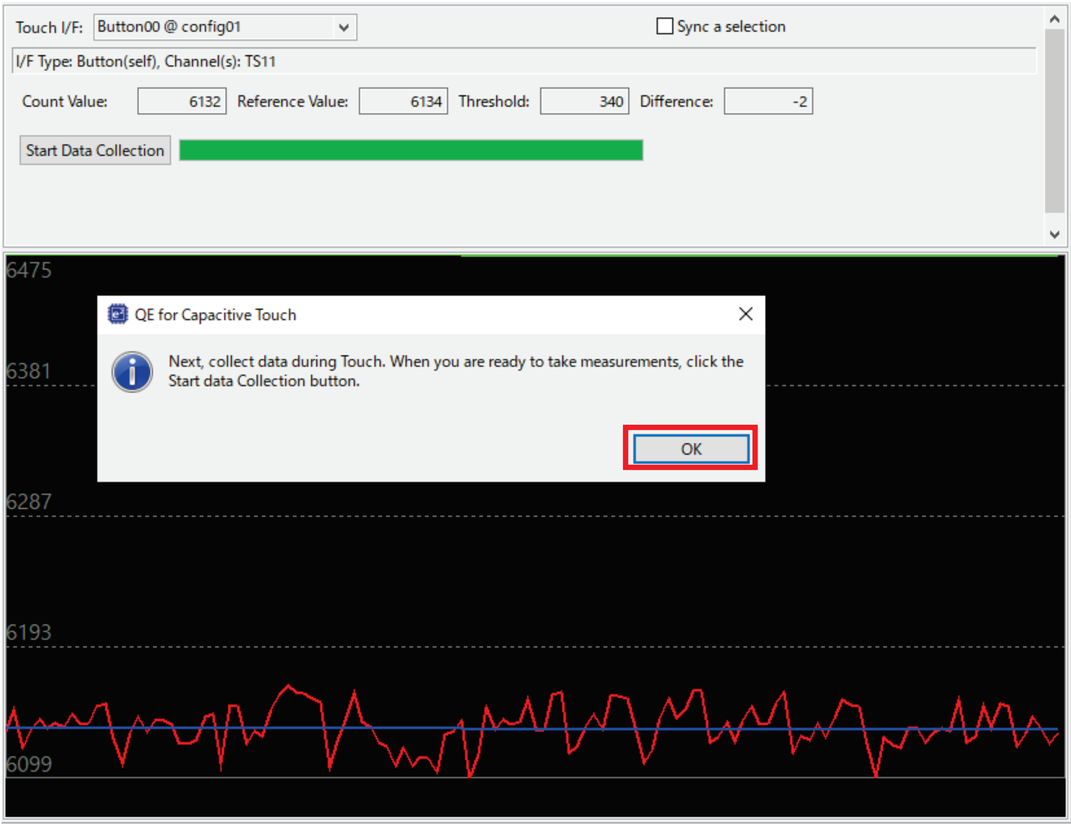

Figure 85. Stop Data Collection Button (touch-off state)

18. After that, touch the electrodes to collect touch-on data. Click on [Start Data Collection] while touching the electrodes. Continue to touch the electrodes until the data collection is complete.

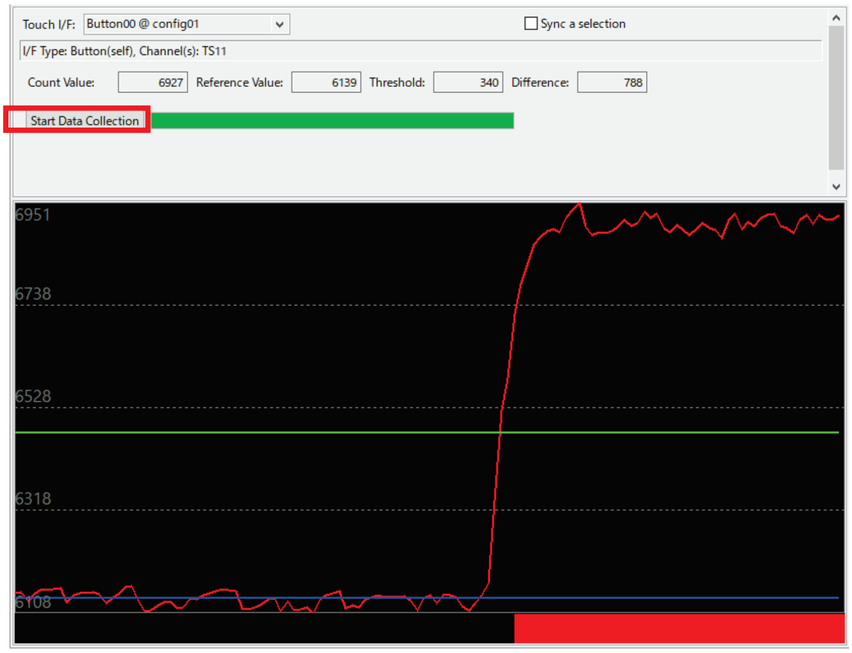

Figure 86. Start Data Collection Button (touch-on state)

19. When the green bar reaches the right edge to indicate the completion of data collection, the [Standard Deviation Measurement Result] dialog box appears. This dialog box shows the standard deviations of noise and the SNRs.

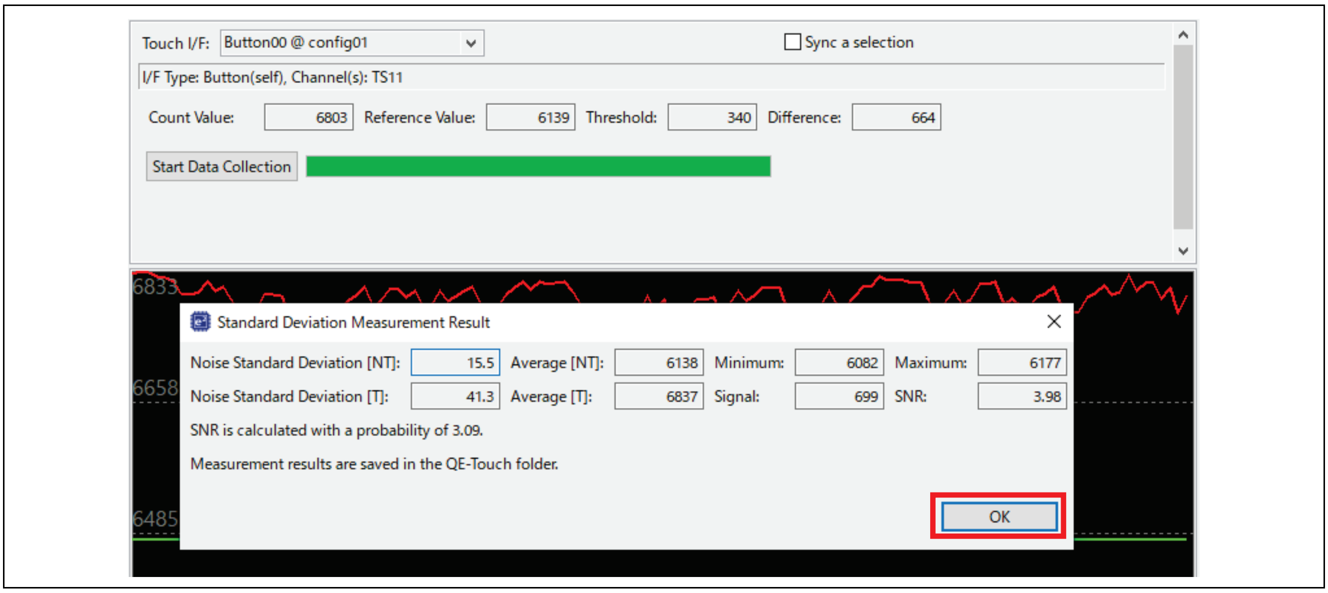

Figure 87. Displaying the Measured Results

6.8 Monitoring Touch Performance with QE for Capacitive Touch Using Serial Communication

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

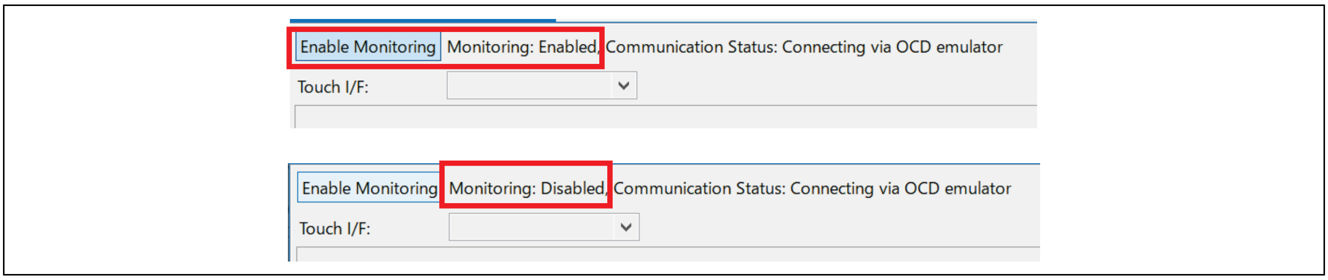

Figure 88. Disable Monitoring Function

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 target 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 board by pressing the **RESET** switch.

5. Open the **CapTouch Workflow** pane and make sure the following folder/file are selected:
   To Select a Project: “Capacitive_Touch_Project_Example”
   To Prepare a Configuration: “Capacitive_Touch_Project_Example.tifcfg”.

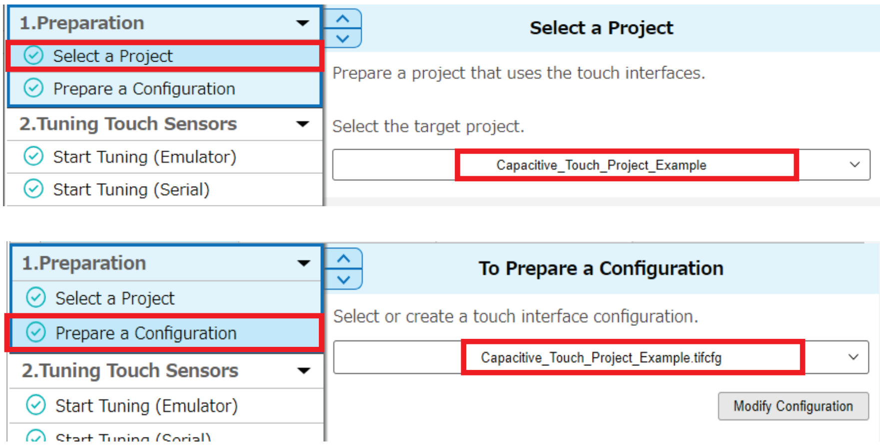

Figure 89. Select Project / Configuration

6. Then select [Start Monitoring (Serial)] in the menu on the left-hand side of “CapTouch Workflow (QE)”. Select [Connect] to enable the serial communication monitoring function.

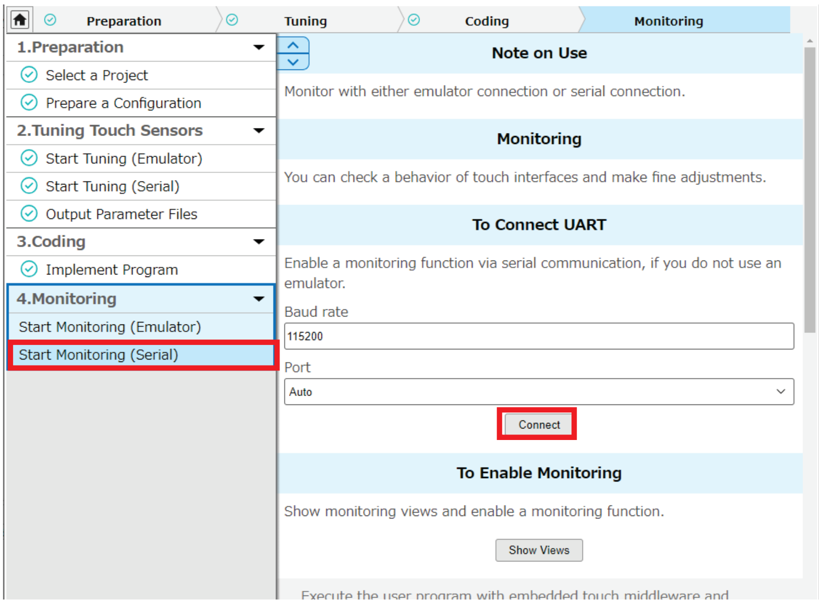

Figure 90. Connect Button

7. “Connected to COMn” 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 91. Console Window Output

8. The rest of the process is the same as Step 9 on in Section 6.7 Monitoring Touch Performance Using e²studio Expressions Window and QE for Capacitive Touch .

7. qe_touch_sample.c Listing After Modifications

7.1 When Using a Software Trigger to Start the CTSU Measurement Process

```
/***********************************************************************
*
* FILE : qe_sample_main.c
* DATE : 2020-09-10
* 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 (TOUCH_CFG_NUM_SLIDERS != 0)
uint16_t slider_position[TOUCH_CFG_NUM_SLIDERS];
#endif
#if (TOUCH_CFG_NUM_WHEELS != 0)
uint16_t wheel_position[TOUCH_CFG_NUM_WHEELS];
#endif





void qe_touch_main(void)
{
    fsp_err_t err;



    /* 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. */
        }



        /* 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);
    }
}
```

7.2 When Using an External Trigger to Start the CTSU Measurement Process

```
/***********************************************************************
*
* FILE : qe_sample_main.c
* DATE : 2020-09-10
* 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 (TOUCH_CFG_NUM_SLIDERS != 0)
uint16_t slider_position[TOUCH_CFG_NUM_SLIDERS];
#endif
#if (TOUCH_CFG_NUM_WHEELS != 0)
uint16_t wheel_position[TOUCH_CFG_NUM_WHEELS];
#endif





void qe_touch_main(void)
{
    fsp_err_t err;

    /* Initializes the software and sets the links defined in the control
structure. */
    R_ELC_Open(g_elc.p_ctrl, g_elc.p_cfg);
    /* Globally enable event linking in the ELC. */
    R_ELC_Enable(g_elc.p_ctrl);

    /* 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) {}
    }

    /* Open AGT driver */
    R_AGT_Open(g_timer0.p_ctrl, g_timer0.p_cfg);
    /* Start AGT. */
    R_AGT_Start(g_timer0.p_ctrl);
```

```
    /* 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. */
        }



        /* 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

RA Product Information	renesas.com/ra
RA Product Support Forum	renesas.com/ra/forum
RA Flexible Software Package	renesas.com/FSP
Renesas Support	renesas.com/support
Capacitive Touch Sensing Unit	renesas.com/solutions/touch-key
	renesas.com/qe-capacitive-touch

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Feb.28.20	-	First edition issued
1.10	Apr.06.20	2	Overview of a simplified example application
		3	Change file generation location
		3	Change project configuration settings
		4	Addition of power supply voltage setting method
		5	Change clock frequency setting method
		5	Change port number not used for touch sensor
		8,9	Added DTC setup instructions
		13	Change in debugging method
		14	Omission of tuning process
		18	Change the method of adding code to the application example
		19,20	Changed the way the monitoring view opens.
		23	Addition of standard deviation measurement method
		25,26	Sample Code Update
2.00	Jun.07.22	Entire document	Added RA2L1 (CTSU2) environment and process steps. Updated IDE version and name Added numbers to figures and table. Changed black arrows in figures to red outlines
		3	Chapter 2/4: Removed Renesas Code Generator Chapter 3: Changed title
		4	Chapter 5: Changed implementation description to figure Chapter 6: Changed title, moved remaining development steps from chapters to sections
		5	Added Fig. 3 Device and Toolchain (RA2L1)
		6	Section 6.1: Added Step 7 for building an artifact and RTOS selection
		7	Added Fig. 5 Project Template Selection
		10	Added Fig. 10 Clocks Configuration (RA2L1)
		12	Added Fig. 15 TS Pin Configurations (RA2L1)
		13	Section 6.2: Added Steps 10 and 11 for when using serial communication for monitoring
		17～21	Section 6.2: Added Steps 19 to 26 for when using serial communication for monitoring
		21～24	Section 6.2: Added Steps 27 to 33 for when using an external trigger to start CTSU measurement
		28	Section 6.3: Added Steps 7 and 8 for adding a shield pin.
		31	Section 6.4: Added Step 1 for confirming emulator connection.
		35	Section 6.5: Added steps for when using an external trigger to start CTSU measurement Section 6.5: Added note regarding build error to Step 11
		38, 39	Section 6.6: Added Steps 6 and 7 for when using a software trigger to start the CTSU measurement
		41	Added Fig. 75 Enable Real-time Refresh Menu
		43	Added Fig. 79 CapTouch Board Monitor RA, RL78, Synergy (QE) Window (RA2L1)
		49～51	Added Section 6.8 Monitoring Touch Performance with QE for Capacitive Touch [RA, RL78, Synergy] Using Serial Communication

		52, 53	Section 7.1: Updated source Code
		54, 55	Added Section 7.2 When Using an External Trigger to Start the CTSU Measurement Process
		56	Changed URL
3.00	Jan.11.24	3～7	Updated the development environment.(Upgrade of development tools and software components)
		25, 26, 29, 32, 34～36, 42, 50	Changes due to the upgrade of QE for Capacitive Touch.(workflow renewal)
		31	Changed toolbar with upgrade of e² studio
		45～48	Changes due to the upgrade of QE for Capacitive Touch.(How to measure the standard deviation)
		56	Added URL

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

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/.