RX Family

Using QE and FIT to Develop Capacitive Touch Applications

R01AN4516EU0300
Rev.3.00
Sep.26.2023

Introduction

This document will demonstrate the needed steps to create an application example that integrates capacitive touch sensing using Renesas RX microcontrollers.

Target Device

RX671, RX140, RX231, RX230, RX130, RX113 with Capacitive Touch Sensing Unit (CTSU)

Contents

1. Application Example Overview

This document will demonstrate how to implement capacitive touch sensing using Renesas RX microcontrollers. Using these steps, the user will be shown the process of connecting the RX140 MCU board, using Smart Configurator for project creation, and QE for Capacitive Touch to create, tune and monitor a Capacitive Touch project.

2. Related Documents

- Firmware Integration Technology User's Manual (R01AN1833EU)
- Board Support Package Firmware Integration Technology Module (R01AN1685EU)
- Adding Firmware Integration Technology Modules to Projects (R01AN1723EU)
- QE CTSU Module Using Firmware Integration Technology (R01AN4469EU0100)
- QE Touch Module Using Firmware Integration Technology (R01AN4470EU0100)

This application example is intended to give a user a short introduction to creating a working example. A thorough review of all the applicable documentation for the e² studio IDE/Smart Configurator, Firmware Integration Technology (FIT) drivers/middleware, Renesas Code Generator, and QE for Capacitive Touch 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 needed to integrate touch detection into this project. These same steps should apply to any typical user development application.

- Create the initial project using the e² studio project creation wizard
- Use Smart Configurator to add the needed modules to the created e² studio project
- Use the QE for Capacitive Touch e² studio plug-in to create the capacitive touch interface
- Use the QE for Capacitive Touch e² studio plug-in to tune the application project
- Add the needed FIT application code function calls to the user project to enable capacitive touch operations in the application project
- Monitor the application project using QE for Capacitive Touch e² studio plug-in to demonstrate capacitive touch detection

4. Required Development Tools and Software Components

The project used for this example is a simplistic example application that performs touch detection. The project uses API calls from the FIT modules described above in the c.

The project utilizes the following development environment:

- RX140 Capacitive Touch Evaluation System (RTK0EG0039S01001BJ)
- Renesas e² studio IDE, V2023-07 or later
- Renesas CC-RX compiler (v3.05.00 or later) installed
- Renesas E2 emulator Lite (RTE0T0002LKCE00000R)
- Renesas QE for Capacitive Touch V3.3.0 or later
- Renesas Firmware Integration Technology (FIT) and Renesas Code Generator
- QE CTSU FIT module (r_ctsu_qe) v2.20
- QE Touch FIT module (rm_touch_qe) v2.20
- Renesas Smart Configurator for RX
- Renesas Code Generator

4.1 Application Example Overview

In the main loop of the application example created:

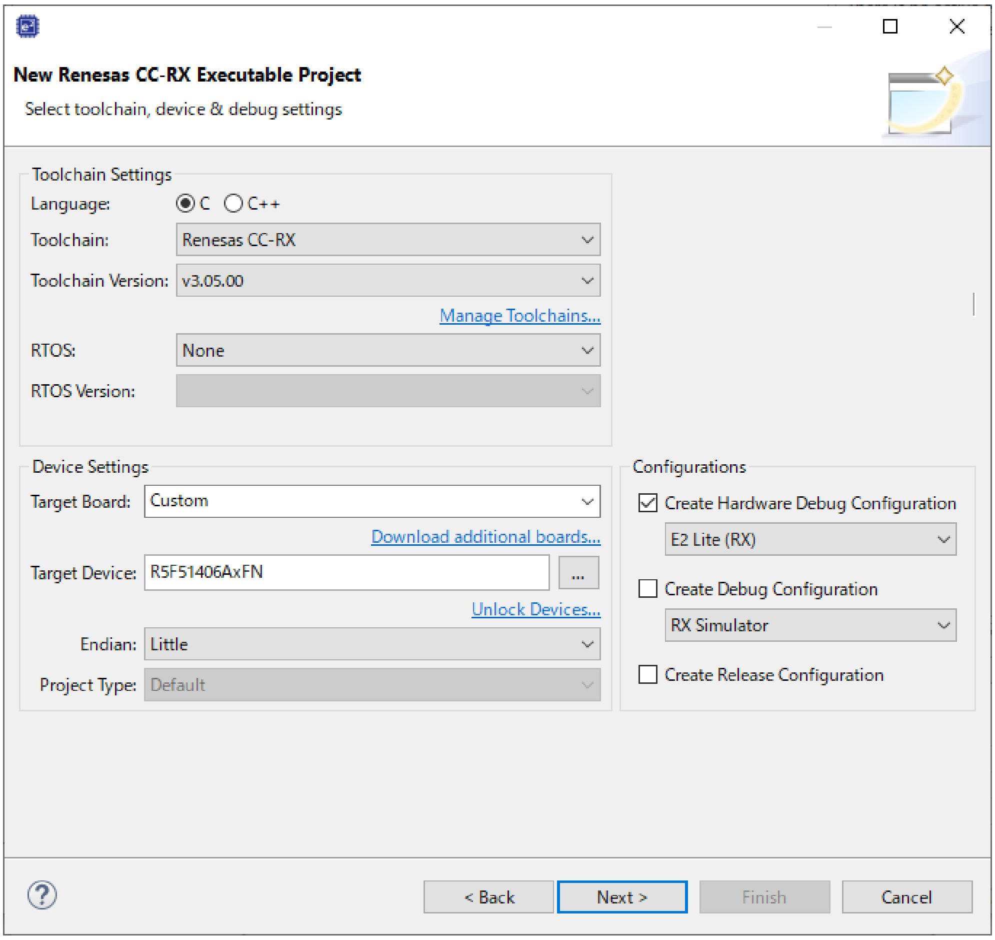
- The global flag used to determine if the rm_touch_qe module is ready to be processed is checked
    - If the flag is TRUE (ready to process)
        - The global flag is reset to FALSE
        - A call to the rm_touch_qe FIT module processes data from the previous completed scan, updates needed data, then starts the next touch scan process
        - A call to the rm_touch_qe FIT module populates a user created global variable with the binary determination of a touch on the sensor board

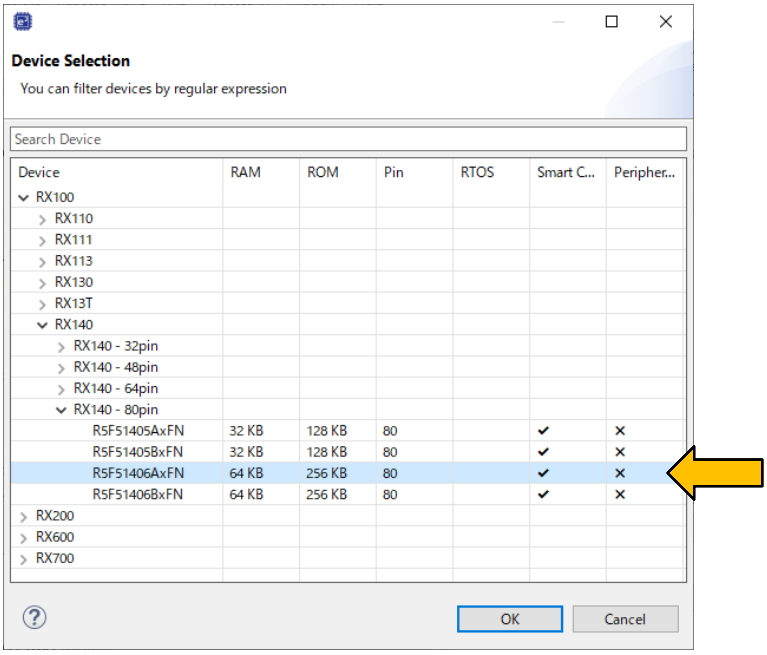
A code listing of the completed application example is in “14. qe_touch_sample.c Listing” for review.

5. Project Creation

1. On the PC start the IDE using the Windows->Start menu or the icon on the desktop. When the dialog appears, create the Workspace at: C:\Workspace\Capacitive_Touch_Project_Example.
2. Start a new project by clicking File->New->C/C++ Project
3. At the dialog box that opens, select Renesas RX and highlight with a single-click **Renesas CC-RX C/C++ Executable Project**, then click Next
4. In the next dialog box, enter a Project name-this can be any name desired. The example here uses **Capacitive_Touch_Project_Example**. When complete, click Next
5. In the next dialog box, ensure the following is selected:
   - Language: C
   - Toolchain: Renesas CC-RX
   - Toolchain Version: v3.05.00
   - Endian: Little
   - Create Hardware Debug Configuration is CHECKED
   - E2 Lite (RX) is selected in the pull-down
   - Target Device: R5F51406AxFN

   Note: Use the '…' to select the proper device using the menu that appears



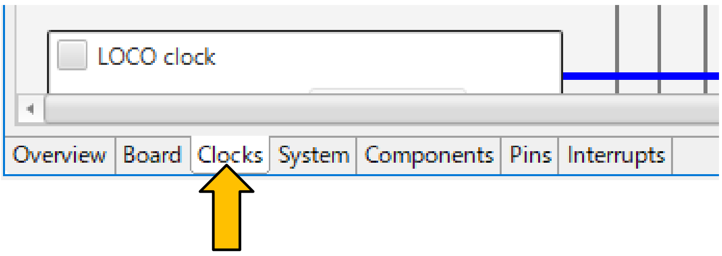


6. Once complete, click Next.
7. In the next dialog box that appears, check the box for “Use Smart Configurator”, then click Finish

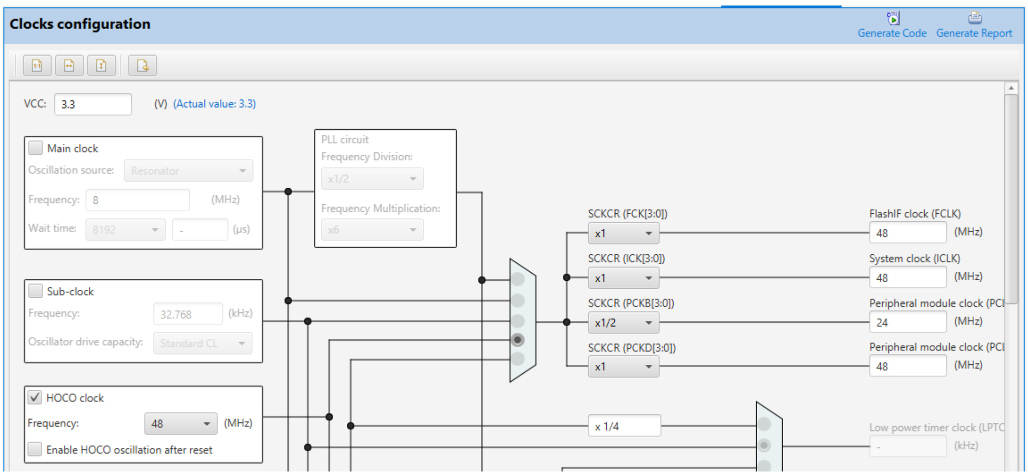
Once complete, a default window will open for the IDE with the Smart Configurator perspective open and ready for project configuration. This completes the project creation.

6. Using Smart Configurator to Add Modules

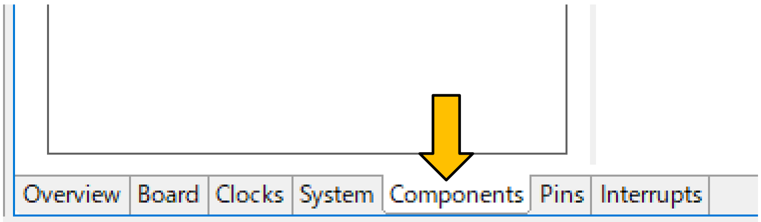
1. Using the tabs in the lower-middle pane of the IDE, select the Clocks tab to display the clock tree for the RX140 MCU.



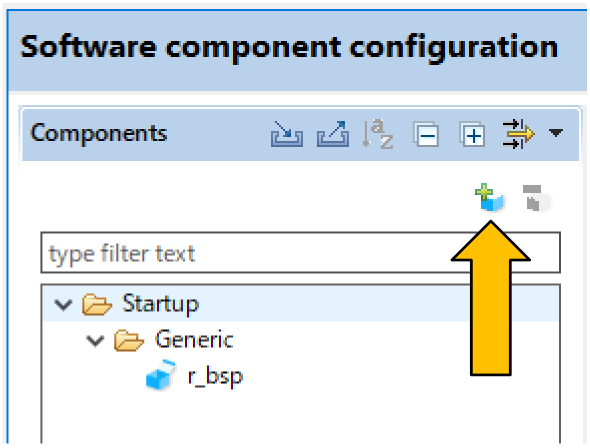
2. The setting of the clock configuration that is used for this application note is shown below.



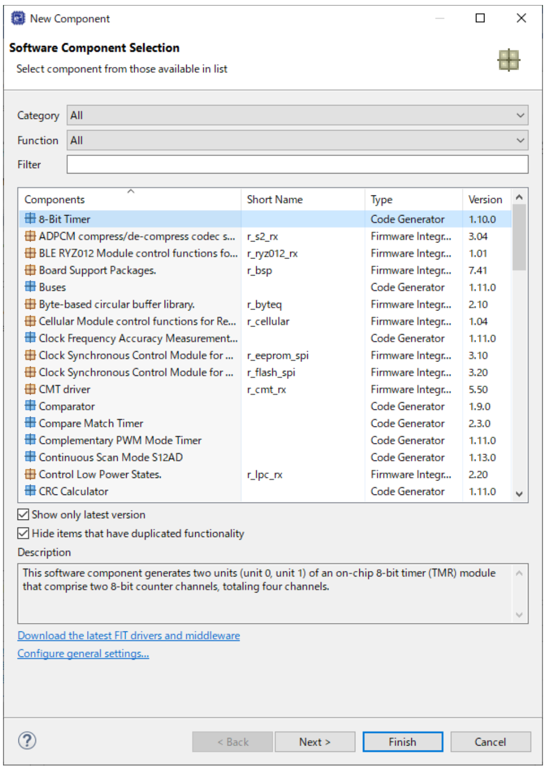
3. Next, move to the Components tab by selecting it at the bottom of the pane



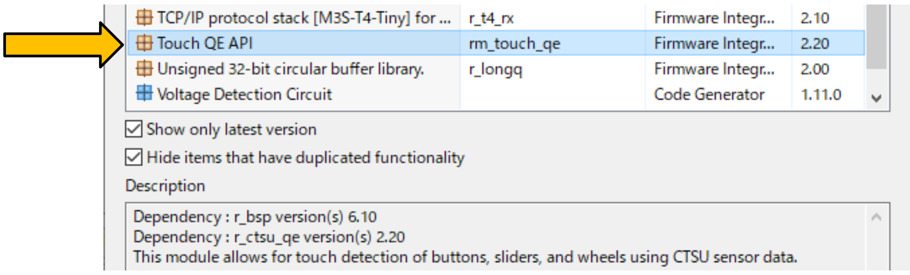
4. Modules now need to be added to the project for application use. Add a module by clicking on the '+' sign in the Components tab



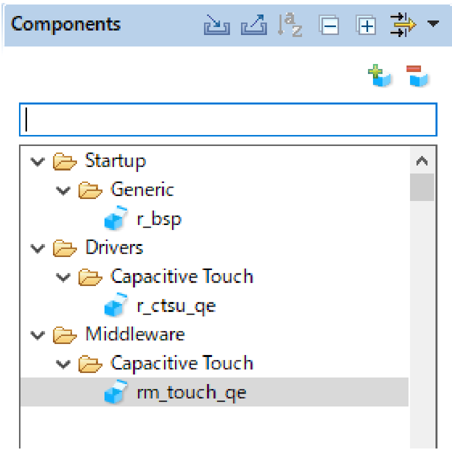
5. A new dialog window will open showing all the available modules that can be added into the project. It will appear similar to the below picture.



6. Scroll down the list until rm_touch_qe is shown. Single-click this entry and click Finish in the lower part of that open dialog box.

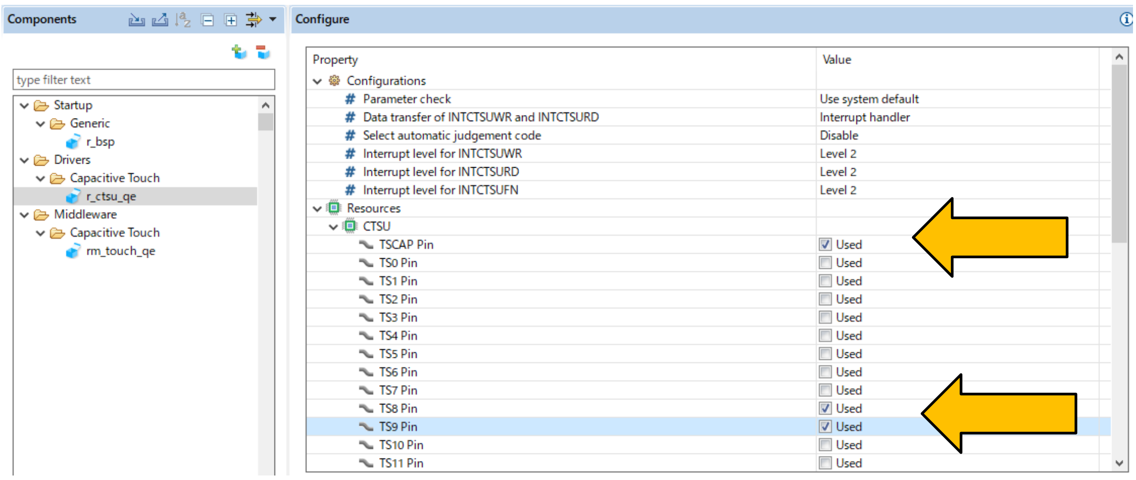


7. The module will appear in the project tree as shown in the picture below:

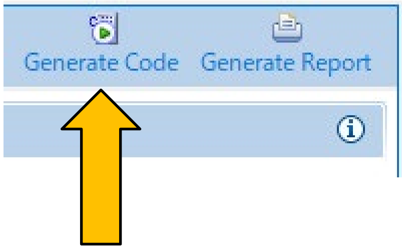


8. Next, assignment between the sensor pads on the Buttons/Wheels/Slider board needs to be made to connect them to the MCU pins. On the Buttons/Wheels/Slider board, the specific MCU sensor pins are labeled for easy identification on the board silkscreen. To do this assignment, select the r_ctsu_qe module and in the Configure pane a list of pins associated with the module will appear. This application example will only assign the two buttons on the upper right-hand side of the Buttons/Wheels/Slider board (TS8 and TS9). Click the following pins in the Configure pane so they can be used in the project:

- TSCAP Pin
- TS8 Pin
- TS9 Pin



9. At this point, all needed application modules for capacitive touch operations have been added. The final step is to generate the needed application code modules for the project. Do this by clicking the Generate Code icon in the upper-right of the Smart Configure pane as shown in the picture below



7. [Additional function] Setting the serial communication monitor using UART (1/2)

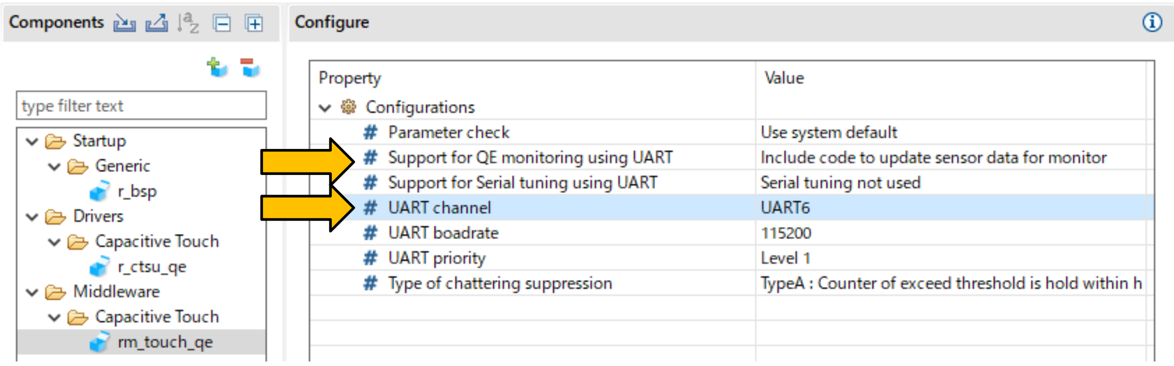
Note: Monitoring touch performance for touch applications can be confirmed by communication via the emulator.

On the other hand, monitoring touch performance can also be achieved via serial communication. Therefore, if you want to monitor smoothly, please add the monitoring function via serial communication.

Chapters 7 and 13 (including this chapter) shown below describe setting the serial communication monitor using UART.

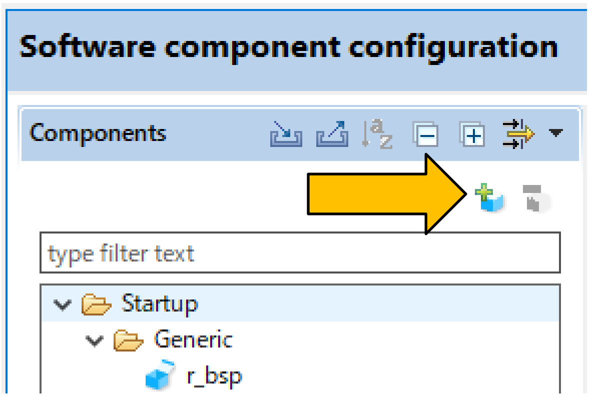
- "7. [Additional function] Setting the serial communication monitor using UART (1/2)"
- "13. [Additional function] Setting the serial communication monitor using UART (2/2)"

1. In the Components tab, select the rm_touch module, set **Support QE monitor using UART** to **Include code to update sensor date for monitor** and **UART channel** to **UARTA6** as shown below

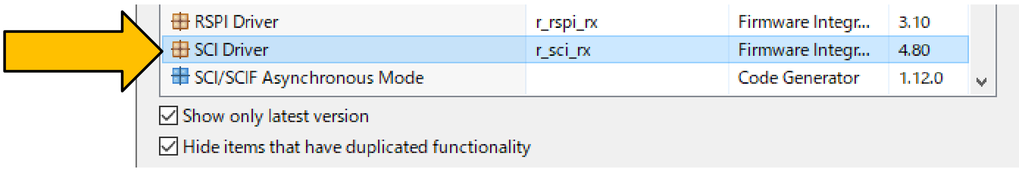


Note: The UART channels and ports used by this tool depend on your target board.

2. Add a module by clicking on the '+' sign in the Components tab.

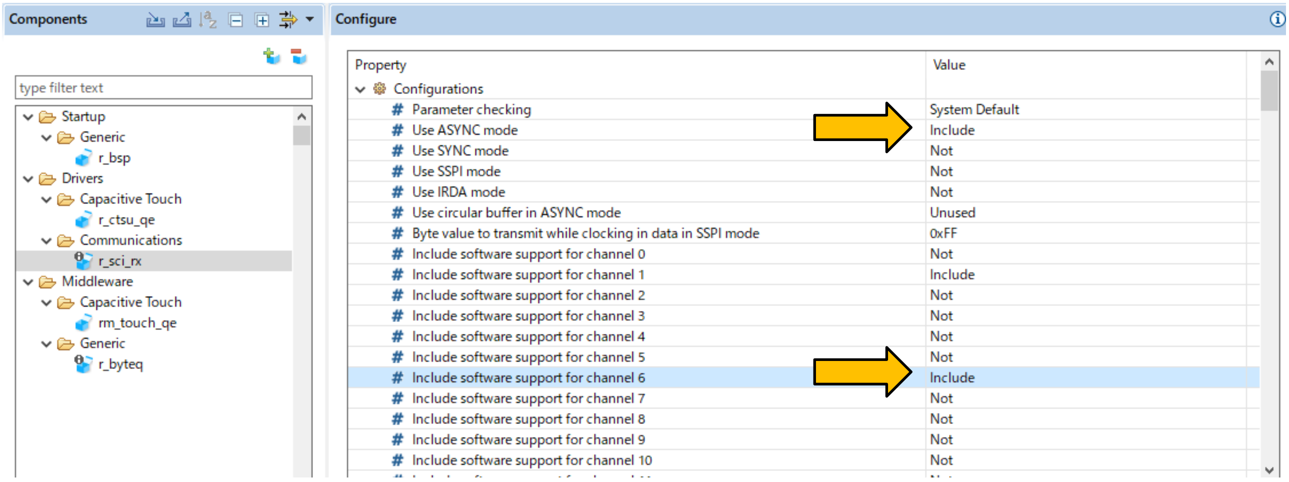


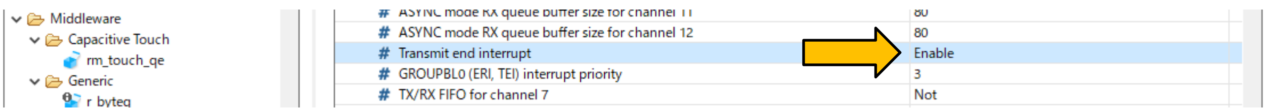
3. Add the code generation component **r_sci_rx** in the same way as the previous **rm_touch_qe**. Single-click this entry and click Finish in the lower part of that open dialog box.

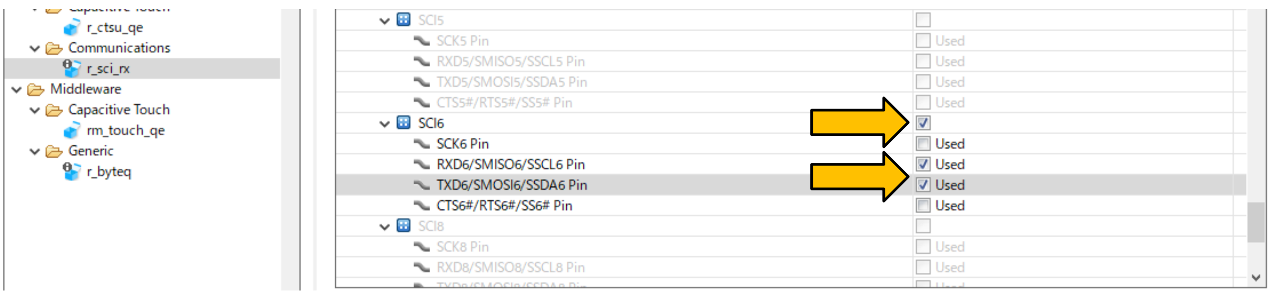


4. Next, select **r_sci_rx** and display the ports associated with this module in the configuration panel and configure as follows.

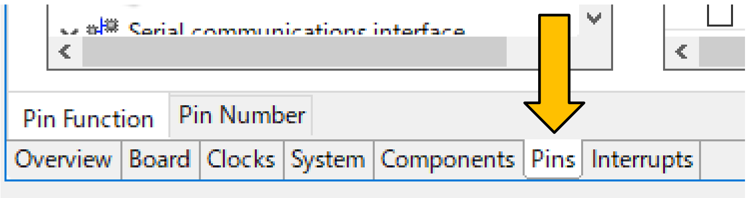
Note: The UART channels and ports used by this tool depend on your target board.





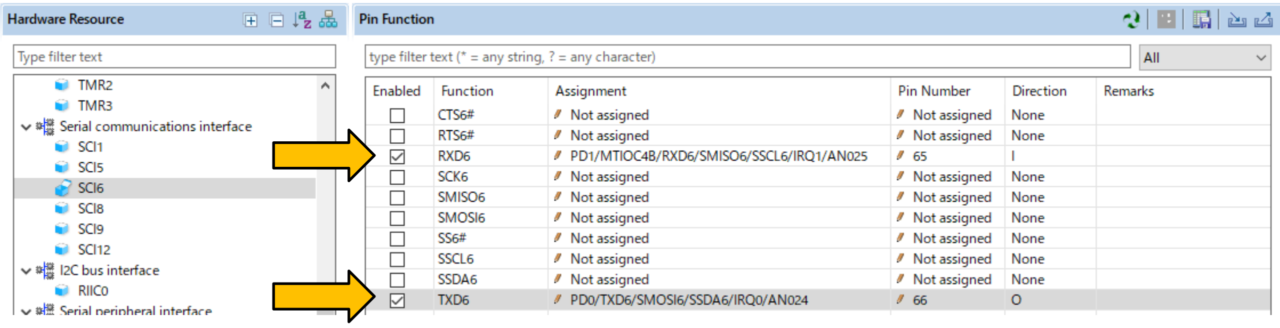


5. Move to the Pins tab by selecting it at the bottom of the pane.

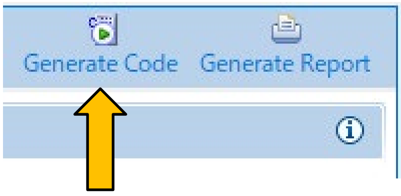


6. Assign RXD6 function to PD1 and TXD1 function to PD0.

Note: The UART channels and ports used by this tool depend on your target board.

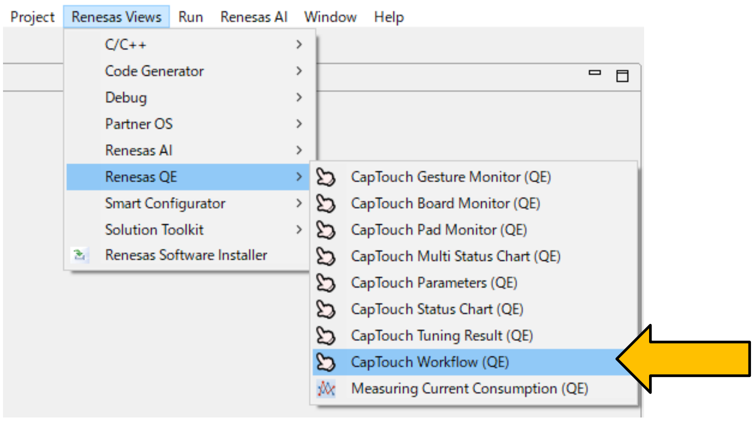


7. Generate the needed application code modules for the project. Do this by clicking the Generate Code icon in the upper-right of the Smart Configurator pane as shown in the picture below.

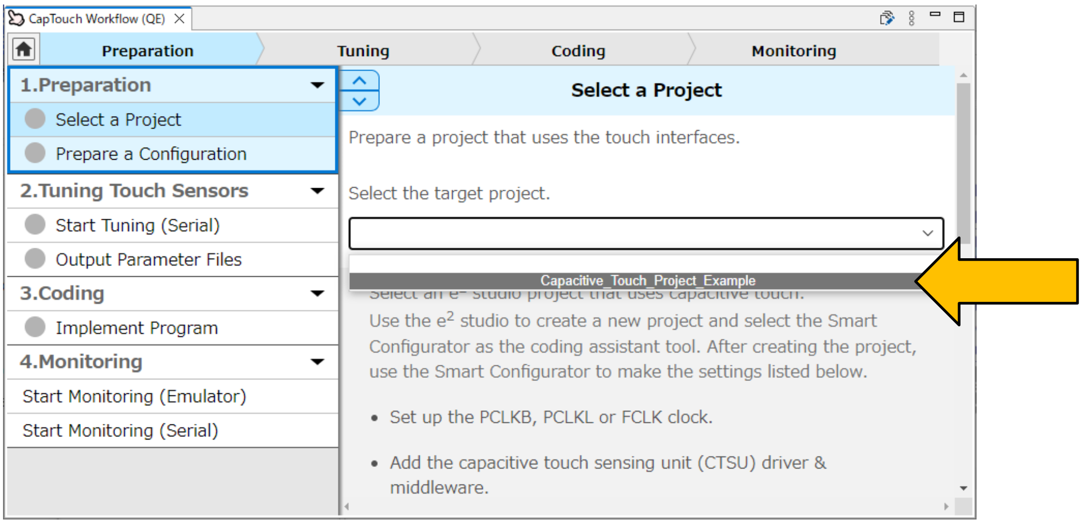


8. Creating the Capacitive Touch Interface

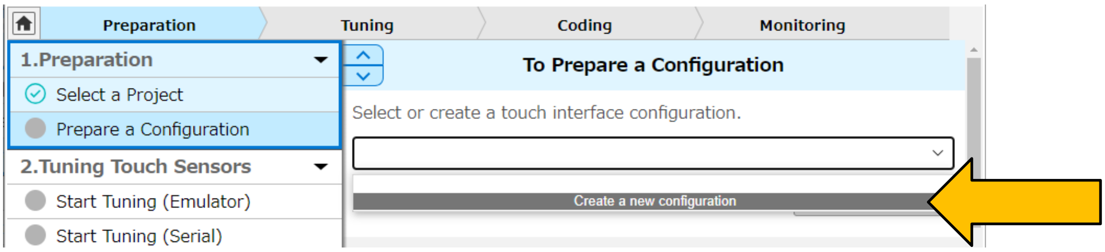
1. From the e² studio IDE, use Renesas Views->Renesas QE->CapTouch Workflow (QE) to open the main perspecitve for configuring capacitive touch to the project



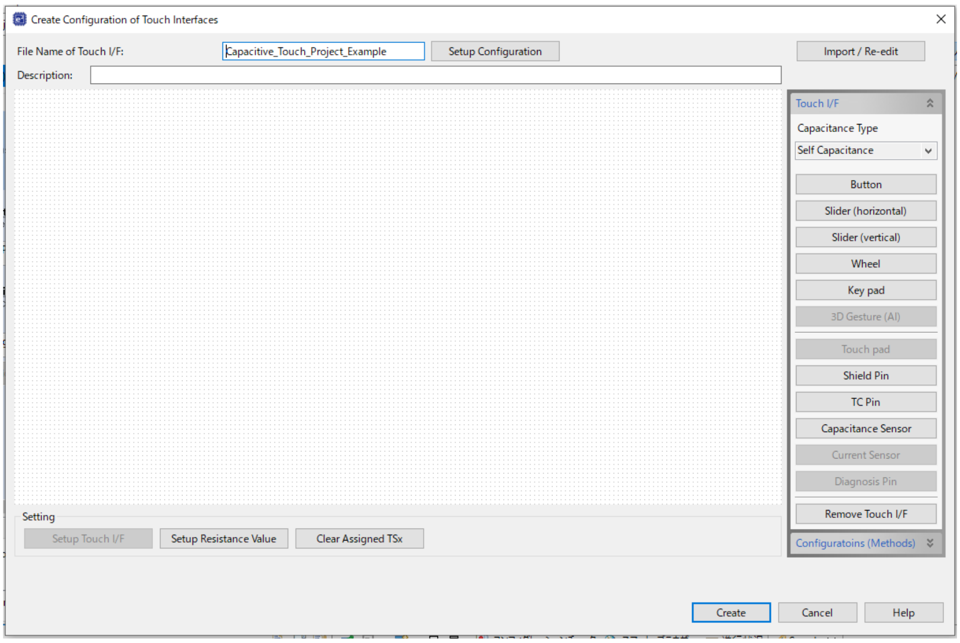
2. In the CapTouch Workflow (QE) pane, select the project to configure the touch interface for by using the pull-down tab and selecting the **Capacitive_Touch_Project_Example** project as shown below.



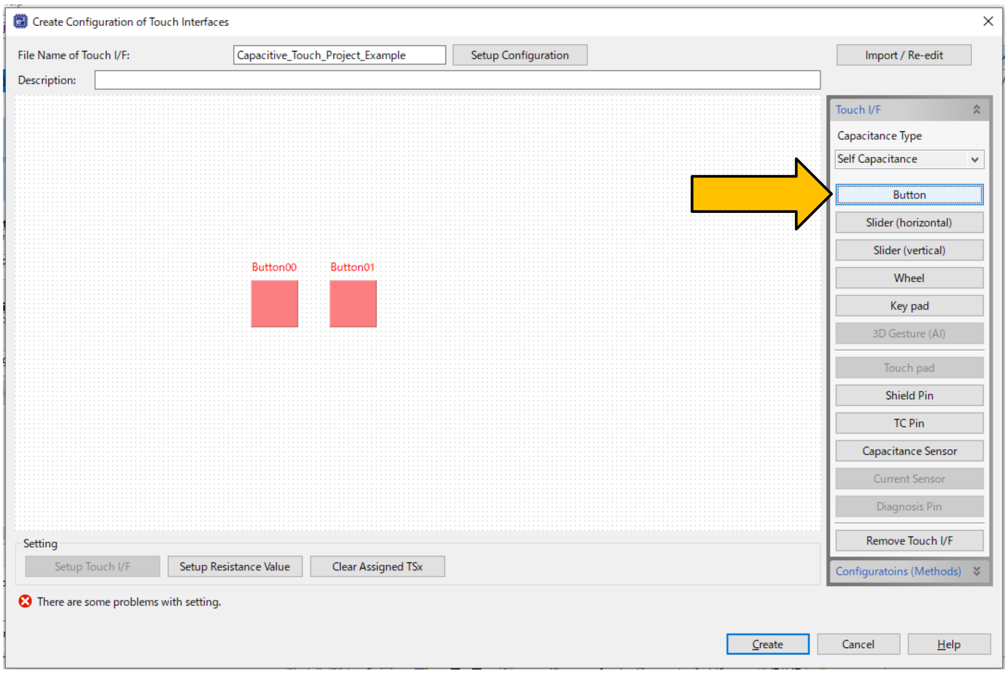
3. Next, select [Prepare a Configuration] in the menu on the left-hand side of "CapTouch Workflow (QE)" to display the items for setting. Select [Create a new configuration] from the pull-down menu to generate a new configuration for a touch interface.



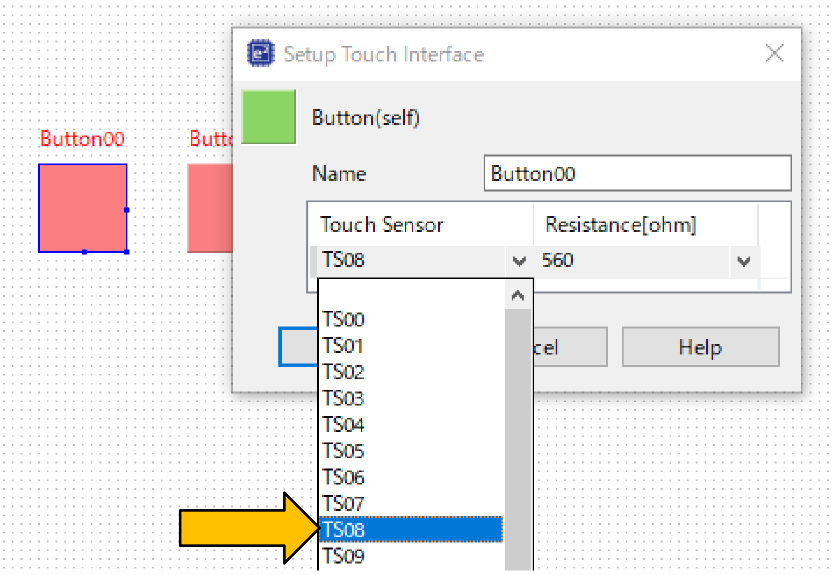
4. A new menu window will open with shows the default blank canvas for creating the touch interface.



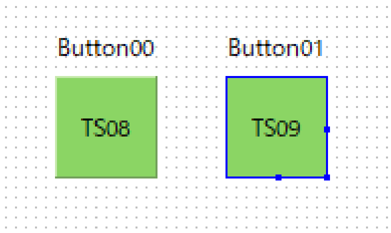
5. Add 2 buttons to the canvas by selecting the **Button** menu item from the right-hand side and adding two (2) buttons to the canvas. Press the **ESC** key to exit once two buttons are added. The canvas will look similar to below.



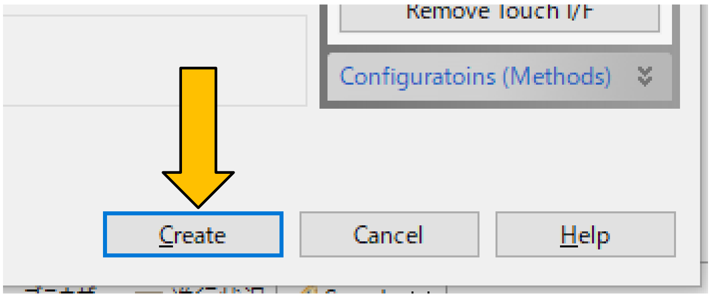
6. To make this connection, double-click on **Button00** and a dialog box will appear. In this case, using the pull-down, select **TS08** as the MCU sensor to assign to this button.



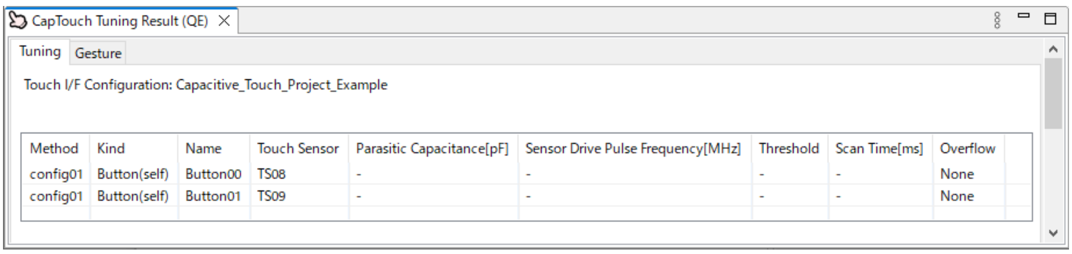
7. Perform the same operation as the previous step for **Button01** and assign it to **TS09**. The canvas should look similar to below. Note also, the indication of a configuration error will go away once all assignment are made properly and correspond to the enabled channels in the Smart Configurator.



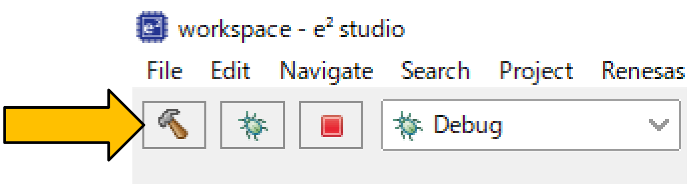
8. Click **Create** in the dialog box. This will setup the Touch Interface.



9. Select [Renesas Views] – [Renesas QE] – [CapTouch Tuning Result (QE)] from the menu of the e^2 studio and open the [CapTouch Tuning Result (QE)] view to display the configuration of the touch interface in the [Tuning] panel.

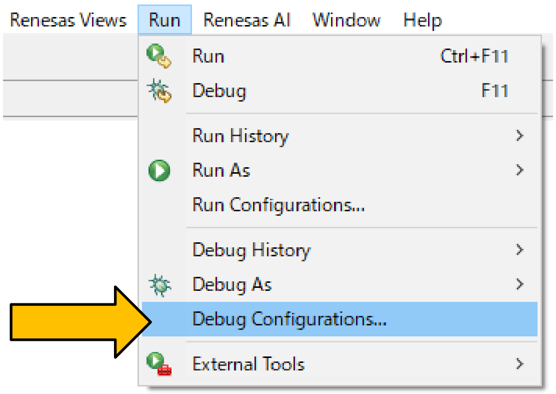


10. Build the project using the hammer icon in the upper left-hand side of the IDE. The project should build without any errors or warnings.

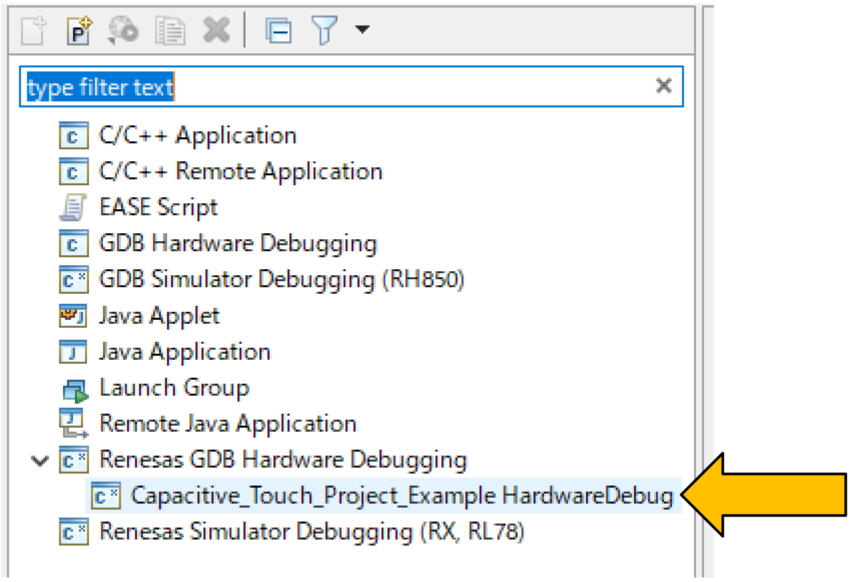


9. Modifying the e² studio Project Debug Session for Capacitive Touch Tuning

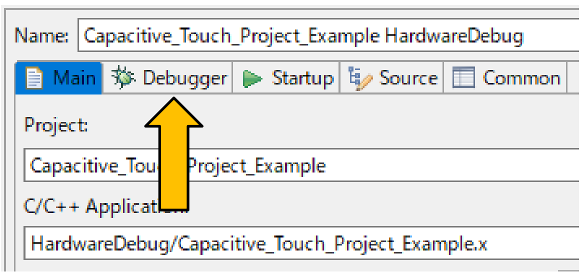
1. The debug session needs to be modified slightly so that a special tuning kernel can be downloaded into the MCU RAM after the debug session starts. Enter the Debug Configuration by clicking **Run->Debug Configurations…**



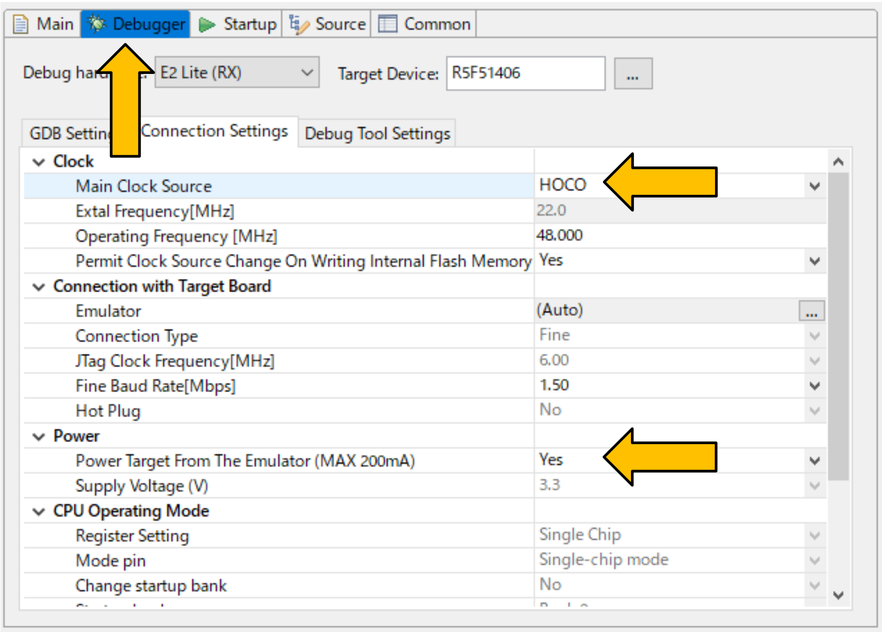
2. A dialog box will now open. In the list in the left-hand pane, open the **Renesas GDB Hardware Debugging** entry and select the named project *HardwareDebug* configuration, in this example: **Capacitive_Touch_Project_Example HardwareDebug**



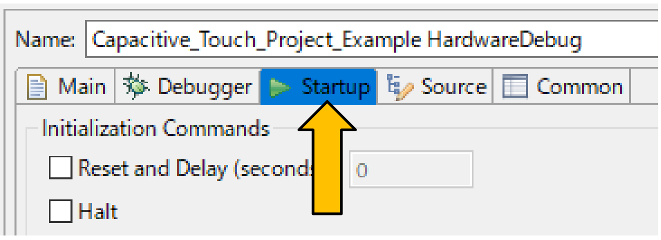
3. In the pane that opens, select the **Debugger** tab.



4. Select the **Connection Settings** tab. For this application example, the power for the target board is supplied from the emulator power supply. **Main Clock Source**, Ensure Power Target From The Emulator (MAX 200mA) and Supply Voltage [V] are set as shown below.



5. Select the **Startup** tab.

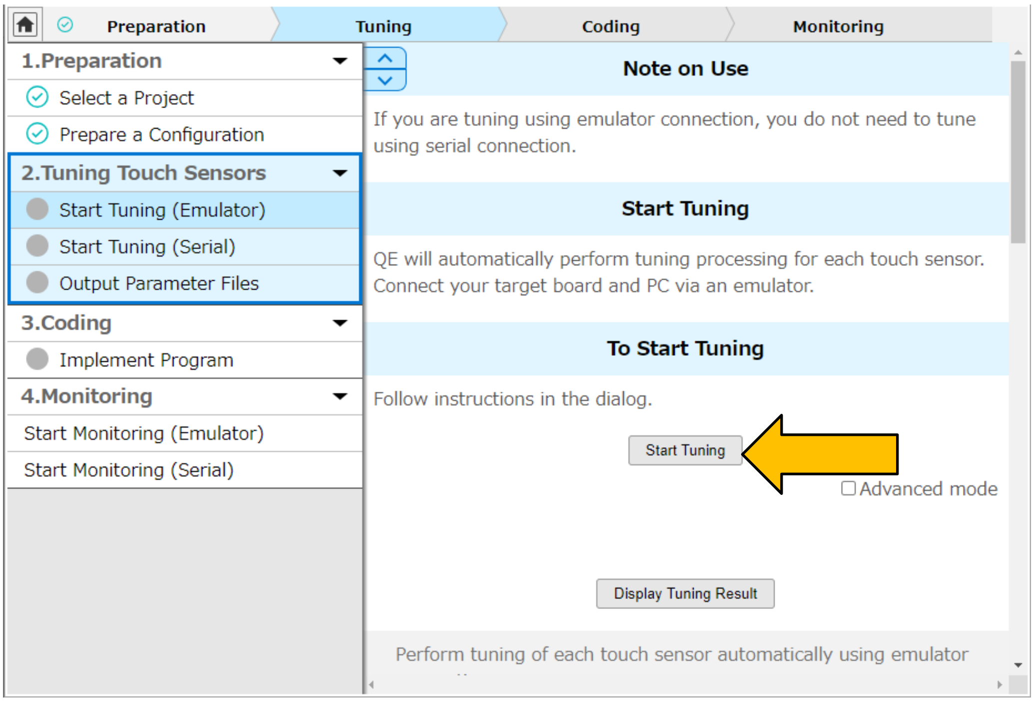


6. Ensure the two check boxes **Set breakpoint at: AND Resume** are checked and look as follows. You may need to scroll down in the dialog box to see these check boxes.

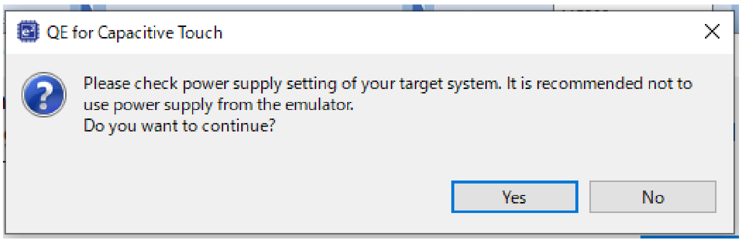
7. Click **Apply** then **Close** to use these modified settings. This completes the project configuration and debug setup for tuning.

10. Tuning the Capacitive Touch Interface Using QE for Capacitive Touch Plug-in

1. 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.

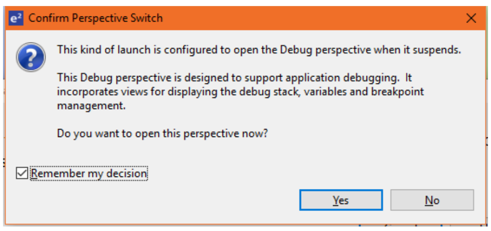


2. If the E1 debugger is supplying power to the target board, the following message will be displayed. Click **YES** to continue the tuning process.

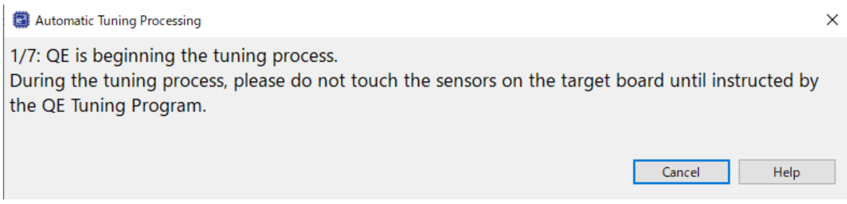


Note: For sake of simplicity, in this application example, the power for the target board is supplied from the emulator power supply. Renesas recommends that tuning be done with the end application power supply to alleviate any voltage deviations that might occur from using the power supplied thru the E2 emulator Lite/PC USB port.

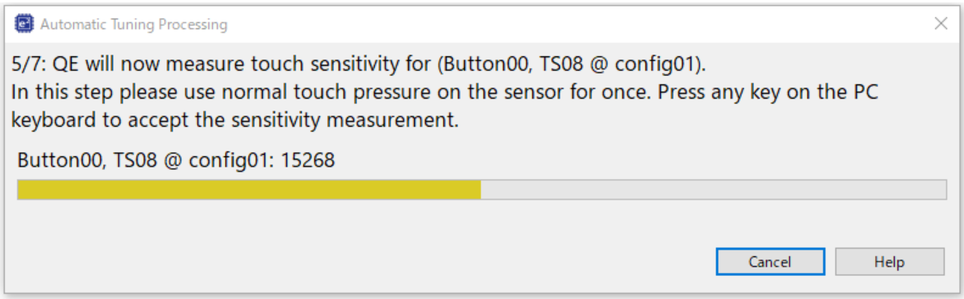
3. At the start of the first debug session, e² studio **MAY** display a message indicating the debug perspective will be switched to. Click the **Remember my decision** check box and **Yes** to continue the debug process and the QE for Capacitive Touch automatic tuning.



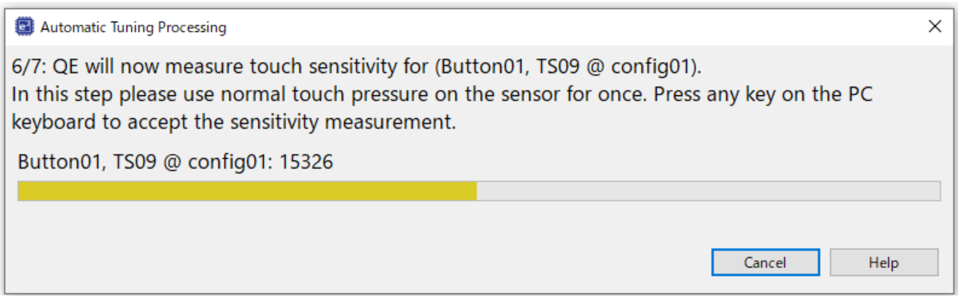
4. QE for Capacitive Touch automatic tuning will now begin. Please carefully read the tuning dialog windows as they will guide you through the tuning process. An example screen is shown below. Typically no interaction is required during the initial tuning process steps.



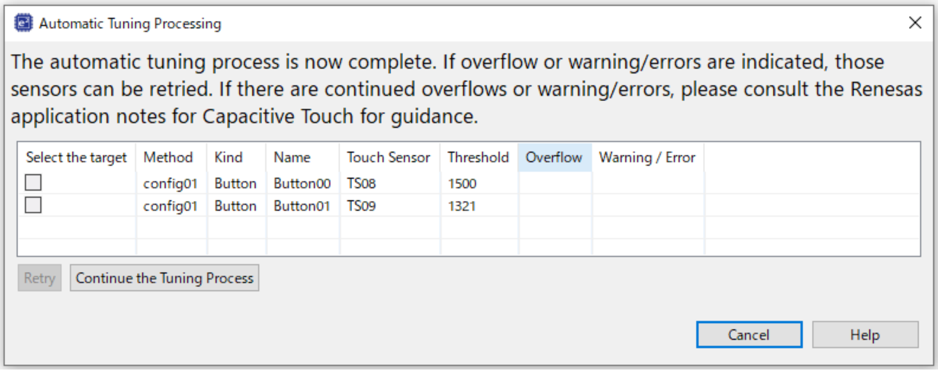
5. After a number of automated steps, a dialog box with information similar to what is shown below will appear. This is the **touch sensitivity measurement** step of the tuning process. As the first ‘interactive’ step of the tuning process, press using **normal touch pressure** on the sensor being indicated in the dialog box (Button00/TS08). When pressing, the bar graph will increase to the right and the touch counts go numerically **UP.** While holding that pressure, **press any key on the PC keyboard** to accept the measurement.



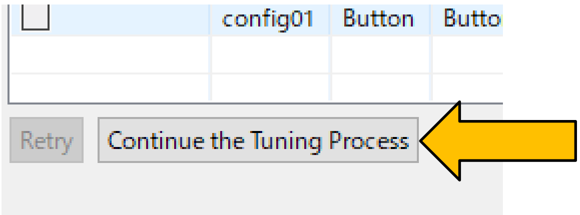
6. Repeat the previous steps for Button01/TS09.



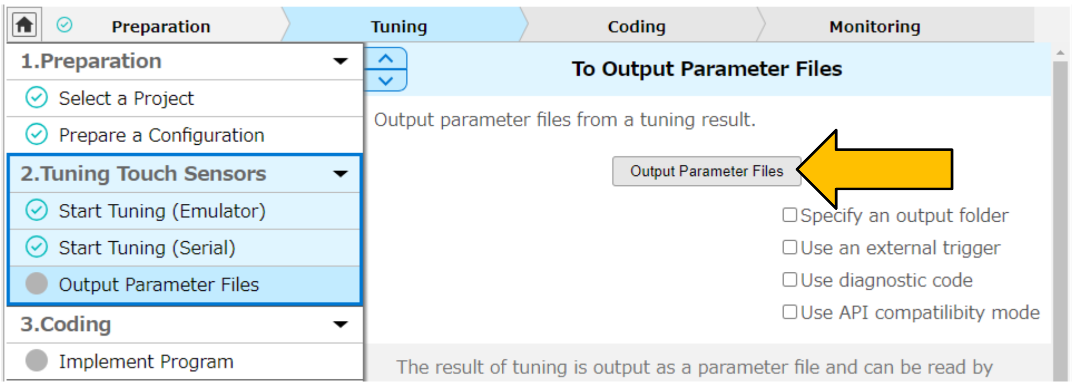
7. Once both buttons are complete, you will see a screen similar to what is shown below. This is the detection threshold that is used by the middleware to determine if a touch event has occurred.



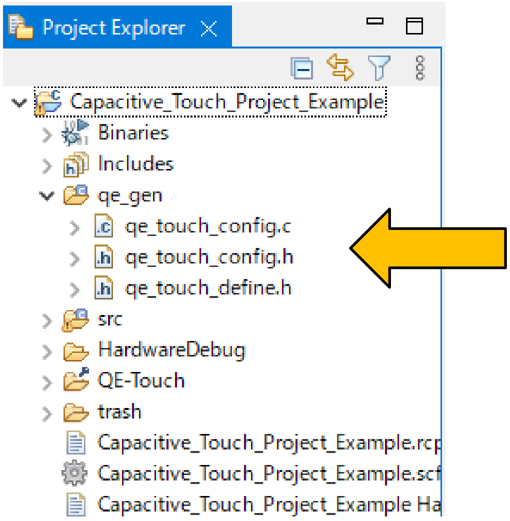
8. Click the **Continue the Tuning Process** button in the dialog box shown. This will exit the tuning process and disconnect from the Debug session on the target. You should return to the default CapTouch Workflow (QE) screen in the e² studio IDE.



9. 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].



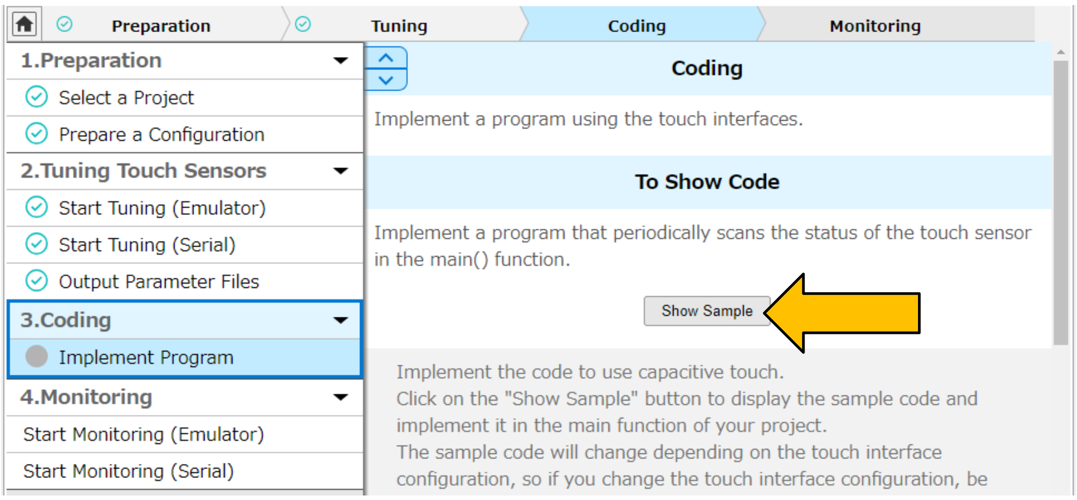
10. In the **Project Explorer** window and you will see that a **qe_touch_config.c, qe_touch_config.h and qe_touch_define.h** file pair have been added. These contain the needed tuning information to enable touch detection using the FIT API middleware.



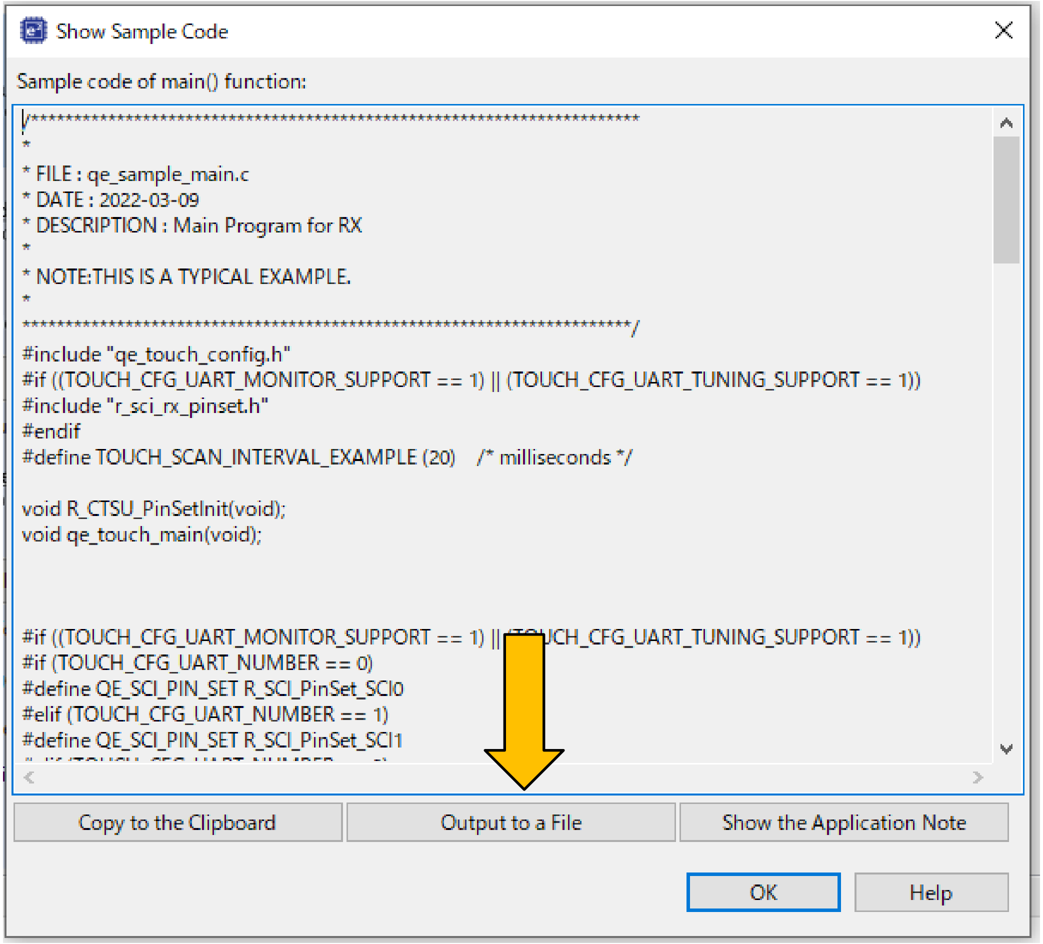
11. Build the project using the hammer icon in the upper left-hand side of the IDE. The Console output showing build results should show no errors.

11. Adding rm_touch_qe FIT Module Calls to Application Example

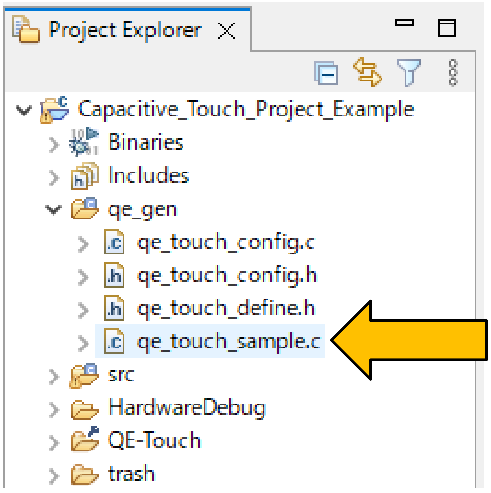
1. To implement a program to scan the states of the touch sensors, select [Implement Program] in the menu on the left-hand side of "CapTouch Workflow (QE)", and then click on [Show Sample].



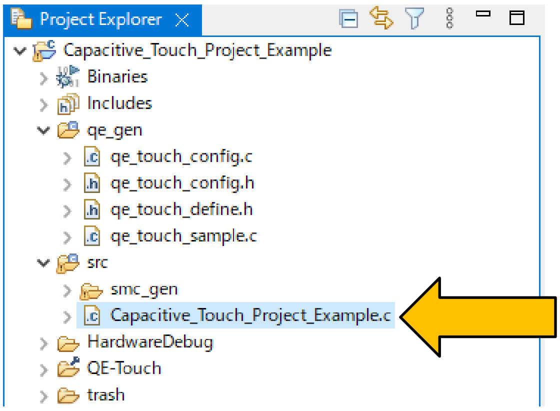
2. A new menu window will open showing the sample code in a text. Click the button **Output to a File**.



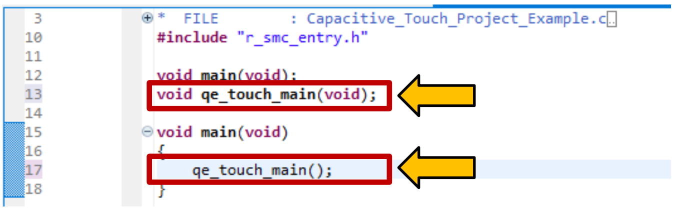
3. Confirm that the qe_touch_sample.c file has been generated in the [Project Explorer] window.



4. Open the **Capacitive_Touch_Project_Example.c** file.



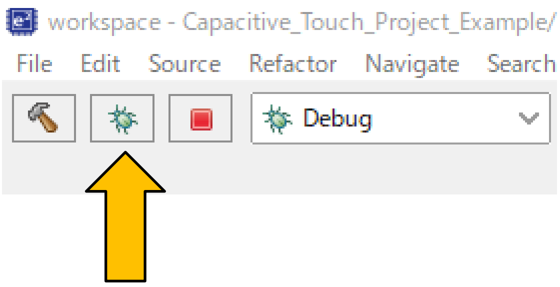
5. In the main() function, call the qe_touch_main(). Add the code (“**void qe_touch_main(void);**” and ”**qe_touch_main();**”) in the red frame to the **Capacitive_Touch_Project_Example.c** file as shown in the image below.



6. This completes all the needed code modifications required for this simple application example. Building the code should result in no errors or warnings for this simplified application example.

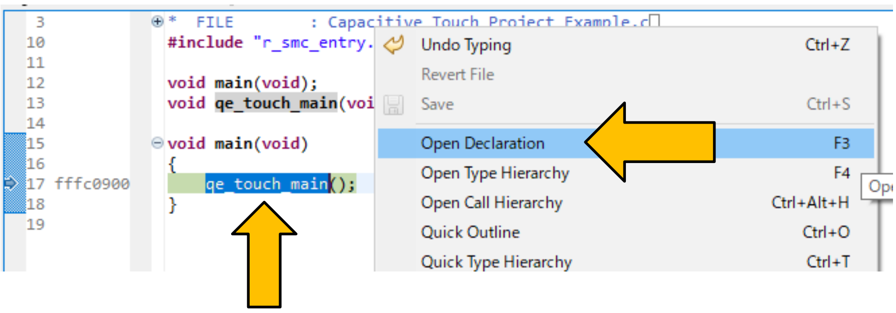
12. Monitoring Touch Performance using e² studio Expressions Window and QE for Capacitive Touch

1. Start a debug session by clicking the **Bug** icon in the upper left hand corner of the e² studio. A debug session will start.

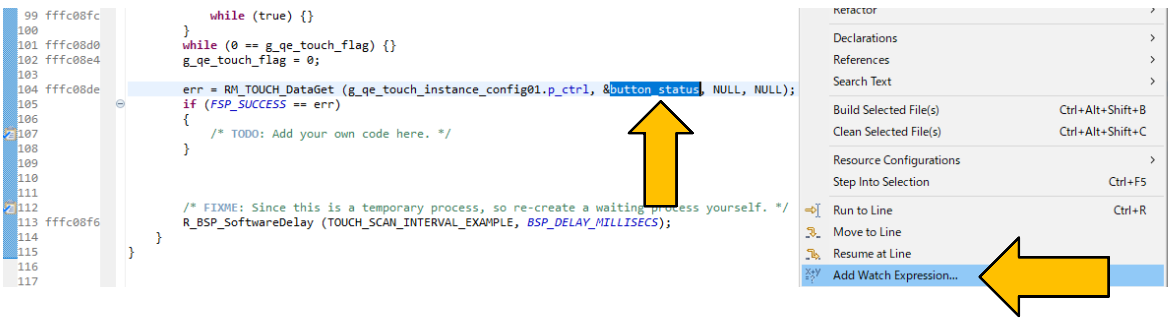


2. The debugger will stop at the **qe_touch_main()** function call. This is the first code point in the **main()** function.

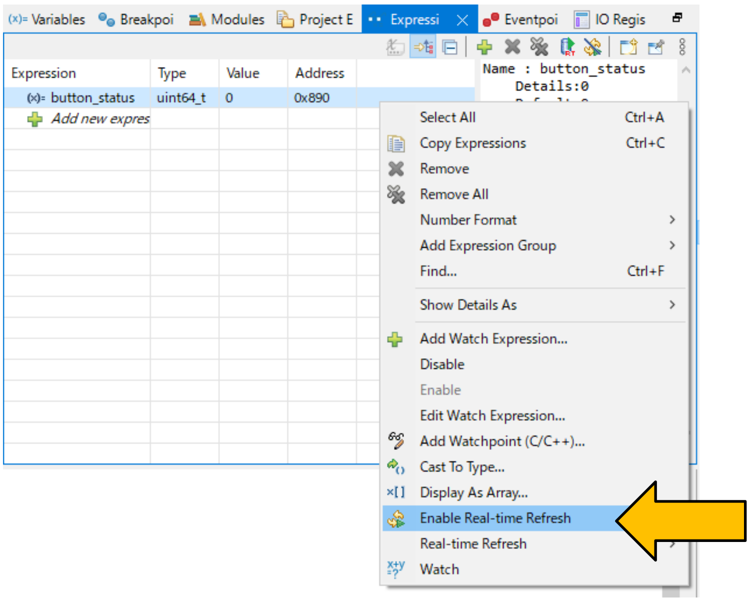
3. Open the declaration of the **qe_touch_main**() function.



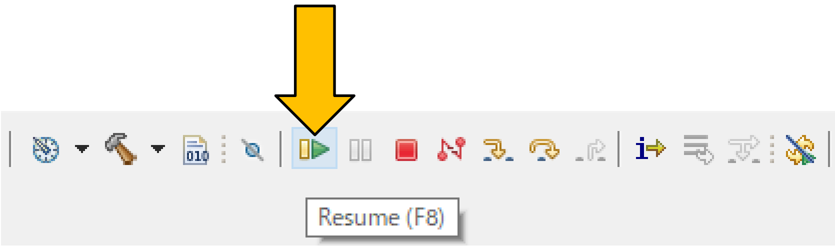
4. Scroll down in the **qe_touch_sample.c** file to the **RM_TOUCH_DataGet**() function in the while (true) loop. Add the variable **button_status** to the expressions window.



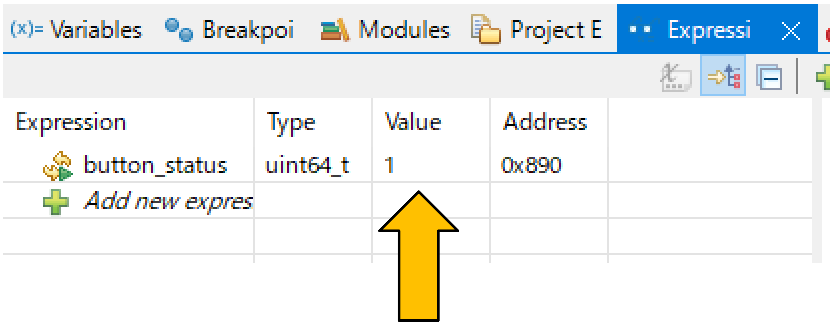
5. Enable Real-time Refresh on the variable in the Expressions window.



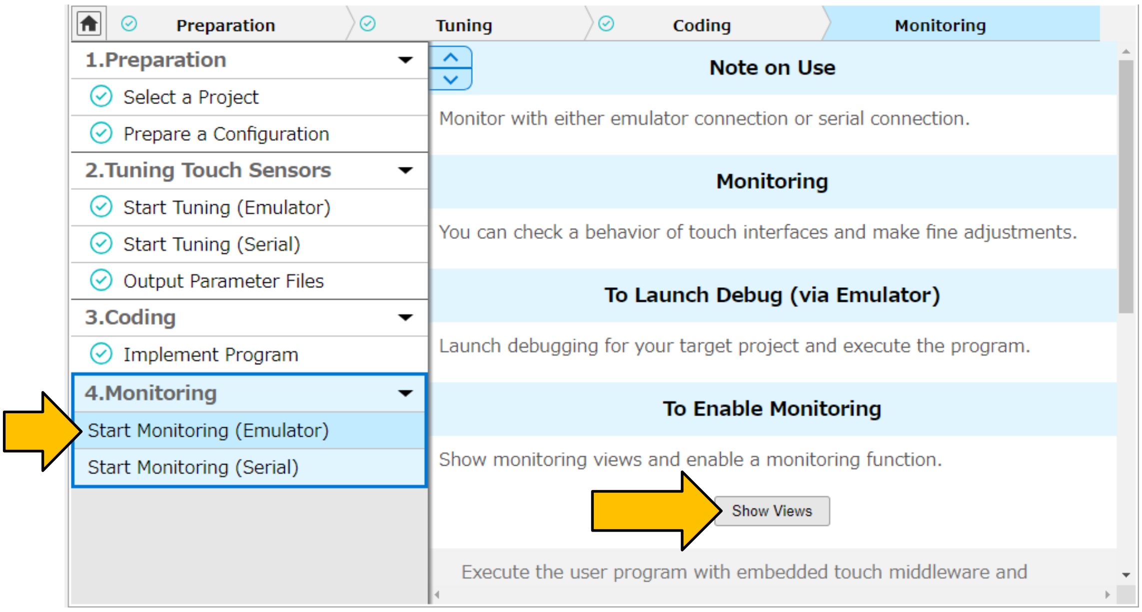
6. Click the '**Resume**' button located approximately in the middle of the e² studio menu bar to continue code execution.



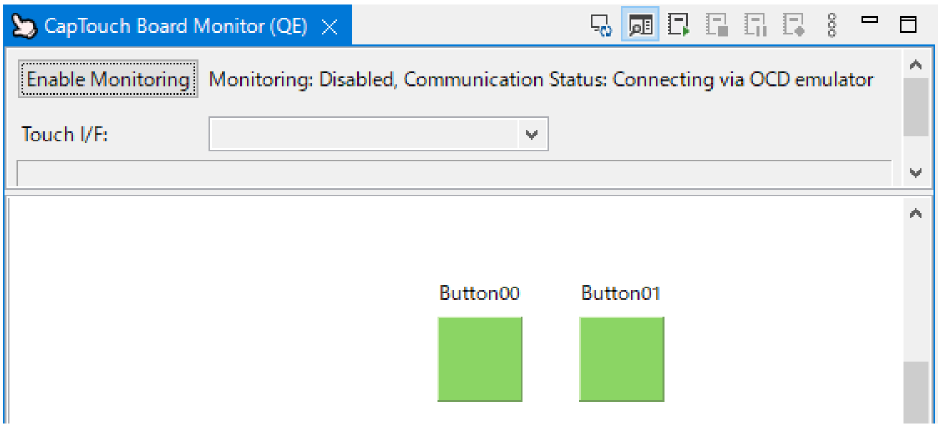
7. Press **TS-B1** on the board, which was configured as **Button00** in Chapter "**8. Creating the Capacitive Touch Interface**" of this application guide. When pressed, a '1' will appear for **button_status** in the Expression window, indicating a binary indication of touch.



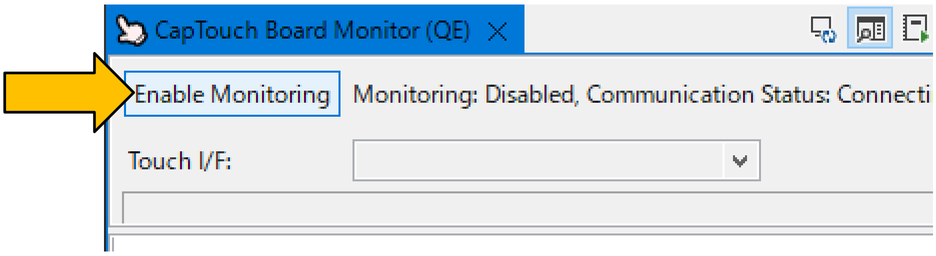
8. Select [Start Monitoring (Emulator)] in the menu on the left-hand side of "CapTouch Workflow (QE)". Click on [Show Views] to start [CapTouch Board Monitor (QE)].



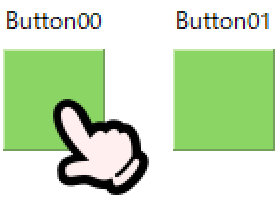
9. It may be necessary to drag the pane up for better viewing, however you should see the **CapTouch Board Monitor (QE)** pane appear similar to the image below.



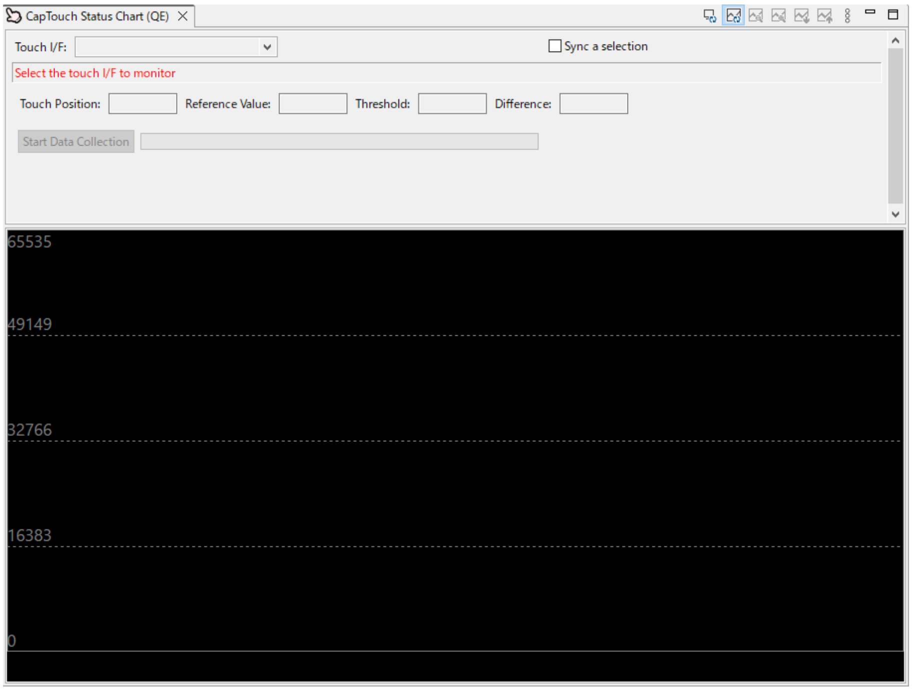
10. Click the **Enable Monitoring** button. The dialog text will change to **Monitoring: Enabled.**



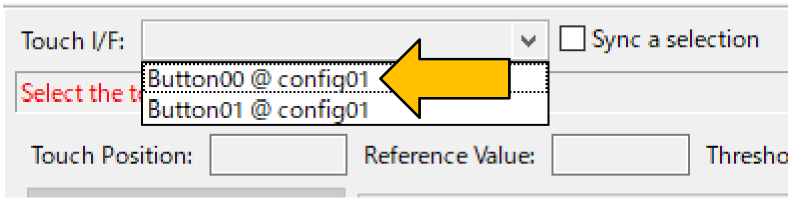
11. T ouch the button **TS-B1** on the target board. The **CapTouch Board Monitor (QE)** will show a touch with a finger image on the button like the image below.



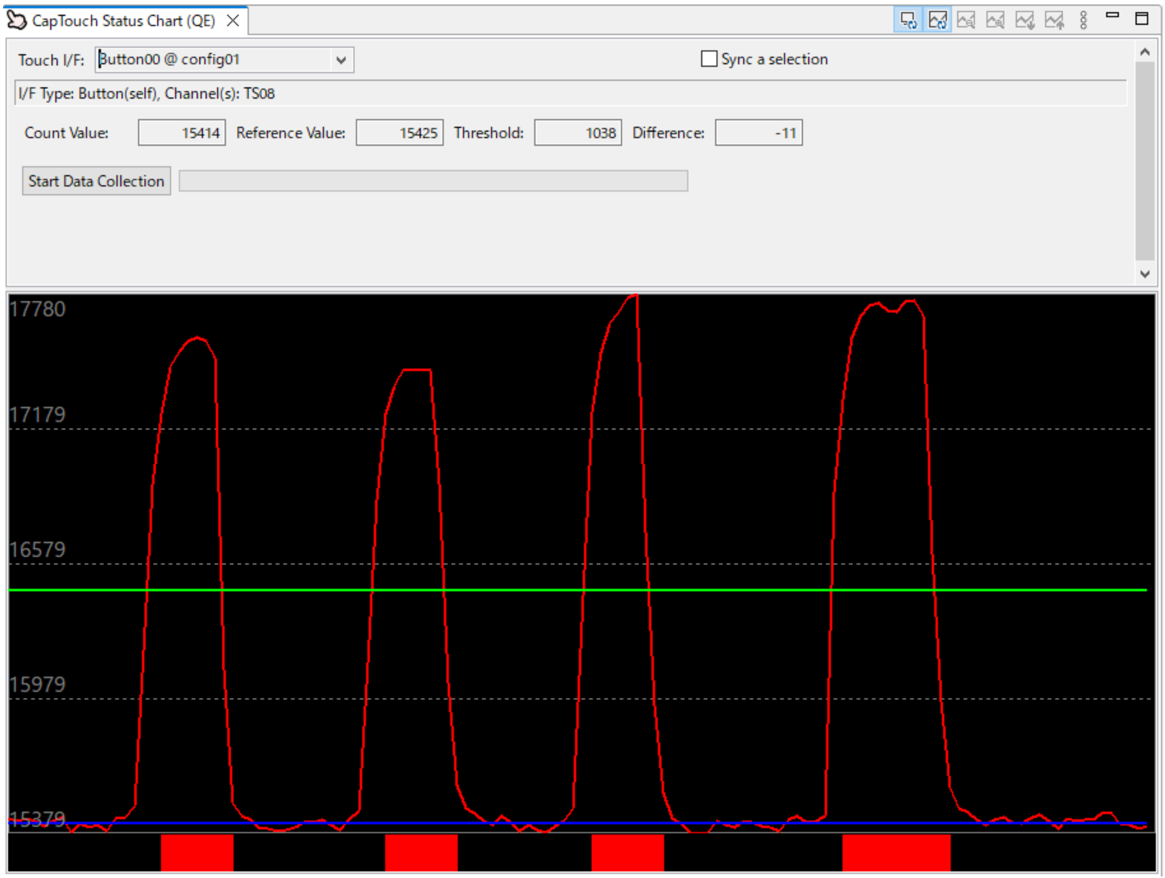
12. To see a graphical representation of the ‘touch counts’ from the board, use the **CapTouch Status Chart (QE)**.



13. Using the pulldown, select **Button00 @ config01**

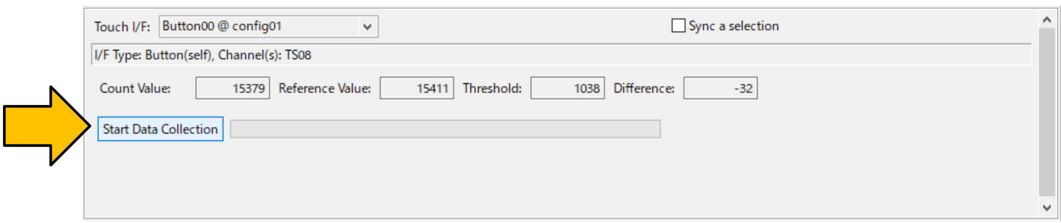


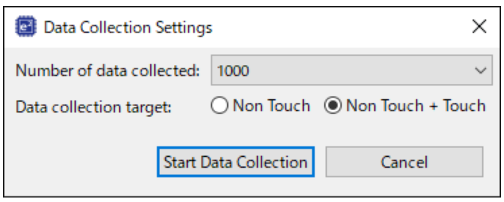
14. The graph will begin to display running values. Touch **TS-B1** on the board and you should see the 'Count Value' show as a step change on the running graph. The **GREEN** line is the touch **'Threshold'**, which the middleware uses to determine whether a button is actuated/touched. The **RED BELT** at the bottom of the graph is a visual indication to the user that the 'Count Value' have crossed above the threshold and a touch is detected.



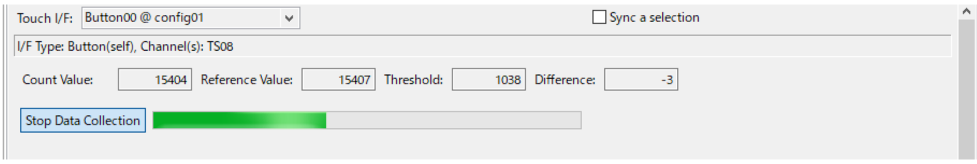
Note: Sections 15 to 18 should only be set when displaying and measuring standard deviation.

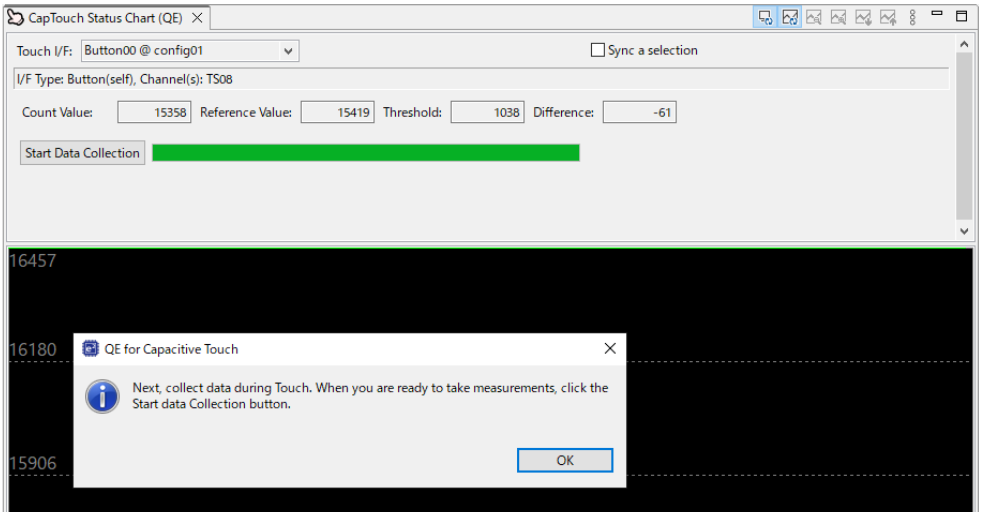
15. Clicking on [Start Data Collection] displays the [Data Collection Settings] dialog box. Select the number of data to be collected from the pull-down menu in this dialog box. Select [Non Touch + Touch] for [Data collection target] and click on [Start Data Collection]. Do not touch the electrodes while collecting the touch-off state.



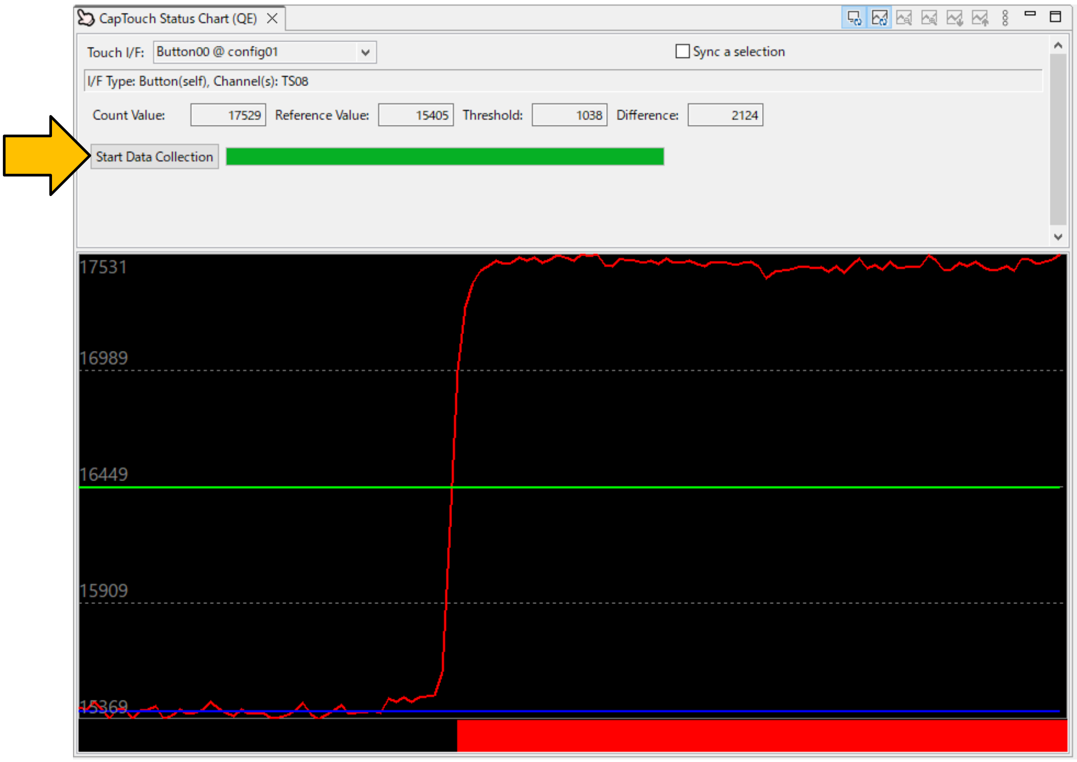


16. When the green bar reaches the right edge, collection of the touch-off data is completed and a dialog box is displayed. Click on [OK] to close the dialog box.

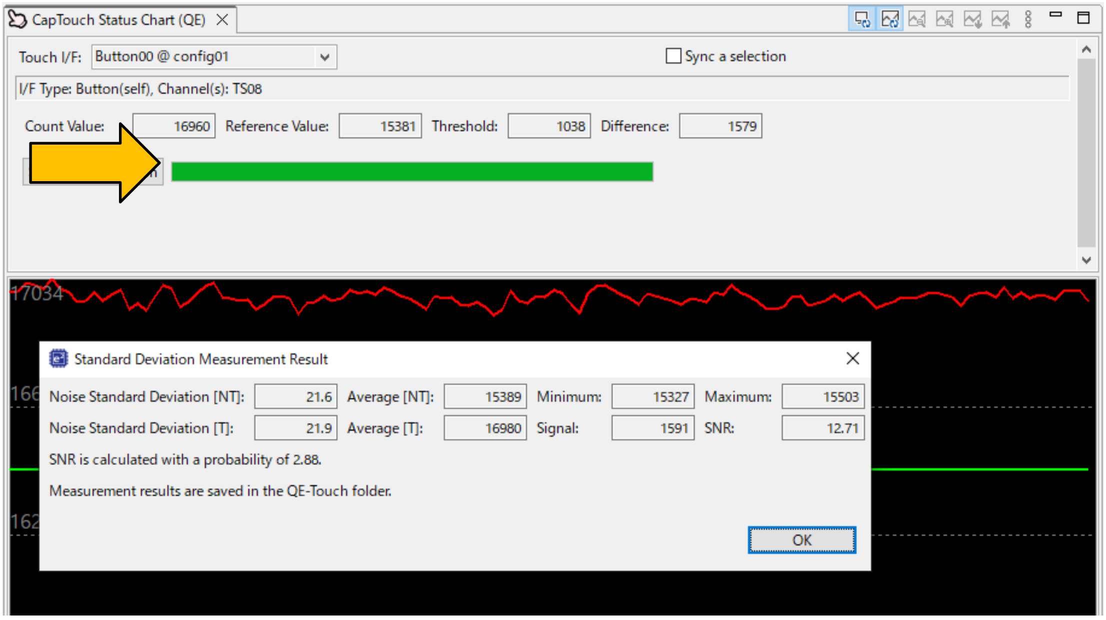




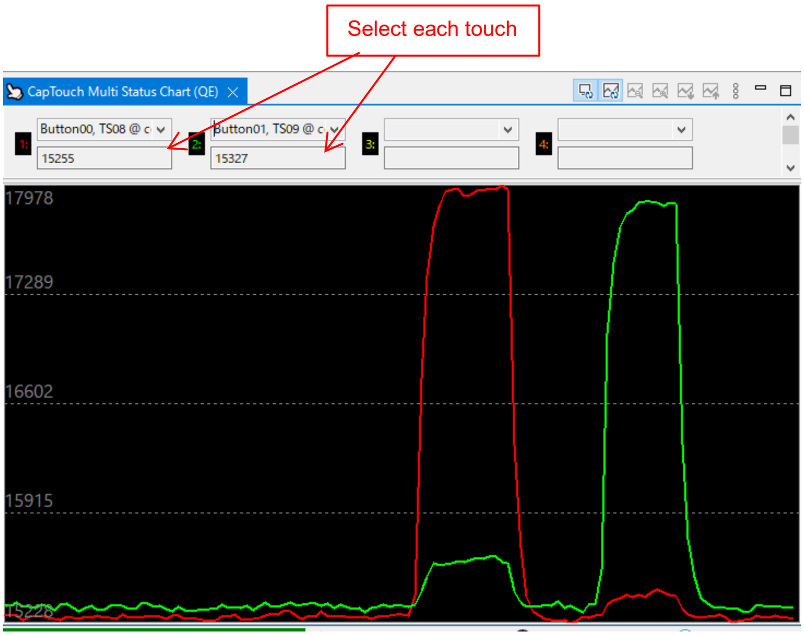
17. After that, touch the electrodes to collect touch-on data. Click on [Start Data Collection] while touching the electrodes. Continue to touch the electrodes until the data collection is complete.



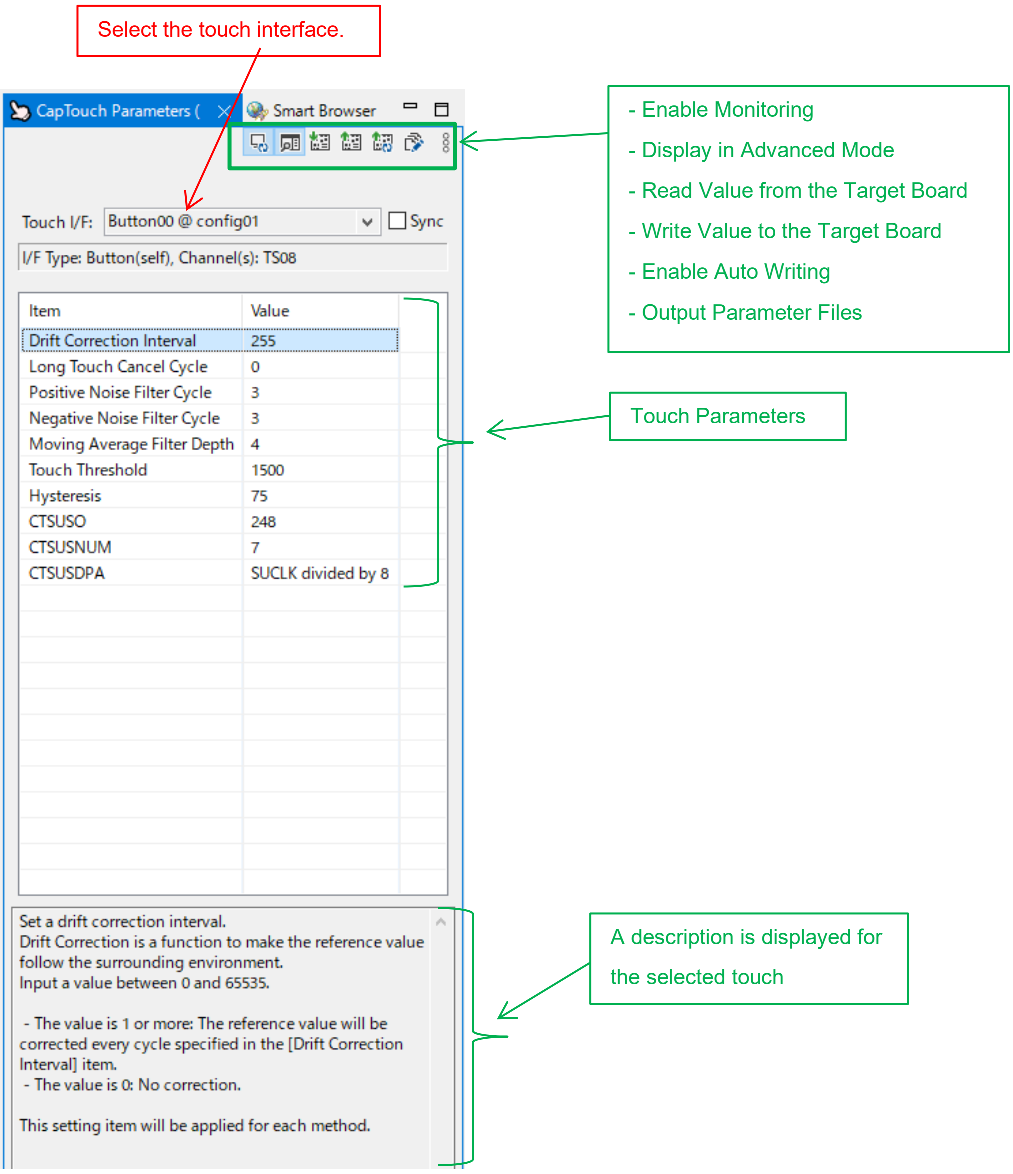
18. When the green bar reaches the right edge to indicate the completion of data collection, the [Standard Deviation Measurement Result] dialog box appears. This dialog box shows the standard deviations of noise and the SNRs.



19. To see a graphical representation of the 'touch counts' for multiple touch sensors, use the **CapTouch Multi Status Chart (QE)**.



20. If you will check and adjust the touch parameters, use the **CapTouch Parameters (QE)**.



13. [Additional function] Setting the serial communication monitor using UART (2/2)

Note: Monitoring touch performance for touch applications can be confirmed by communication via the emulator.

On the other hand, monitoring touch performance can also be achieved via serial communication. Therefore, if you want to monitor smoothly, please add the monitoring function via serial communication.

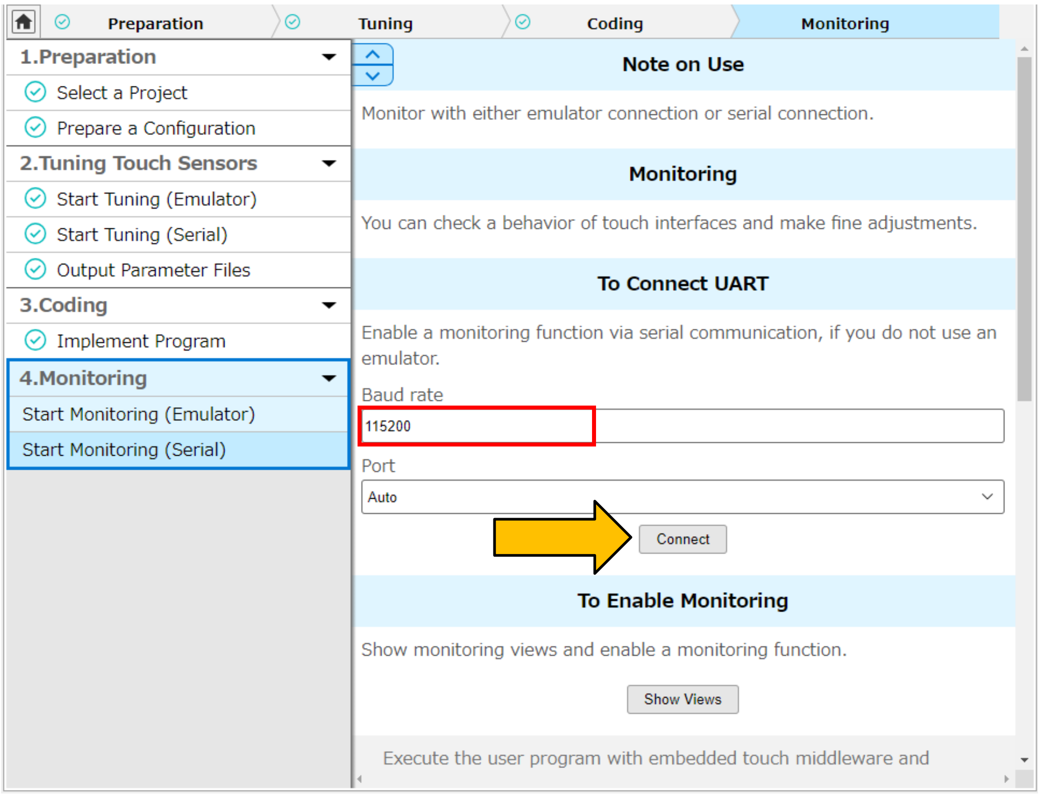
Chapters 7 and 13 (including this chapter) shown below describe setting the serial communication monitor using UART.

- "7. [Additional function] Setting the serial communication monitor using UART (1/2)"
- "13. [Additional function] Setting the serial communication monitor using UART (2/2)"

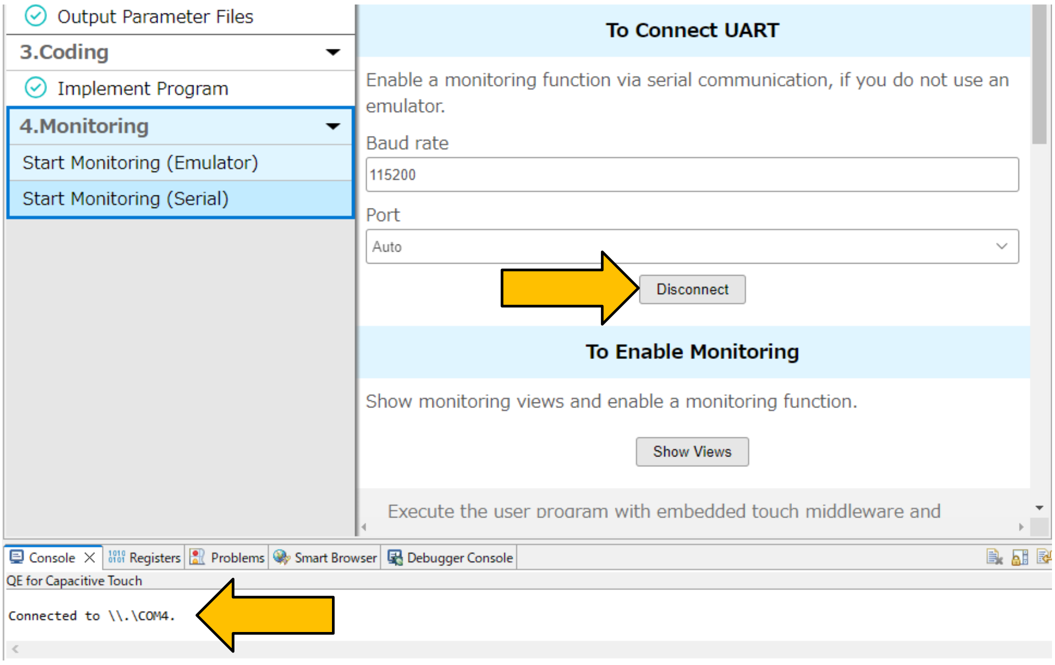
1. Connect the target board to the PC via serial connection (UART / USB).

2. Run the touch application program on the target board.

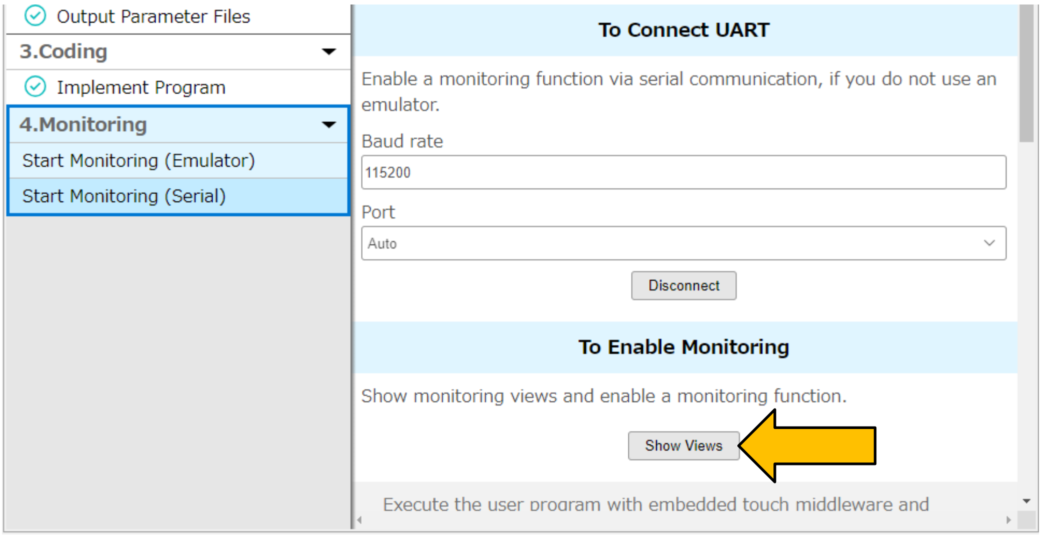
3. Open the "CapTouch Workflow (QE)" panel. Make sure that Capacitive_Touch_Project_Example and Capacitive_Touch_Project_Example.tifcfg have been respectively set in [To Select a Project] and [To Prepare a Configuration]. Then select [Start Monitoring (Serial)] in the menu on the left-hand side of "CapTouch Workflow (QE)". Set the baud rate which was set in Chapter 7 for [Baud rate], and click on the [Connect] button.



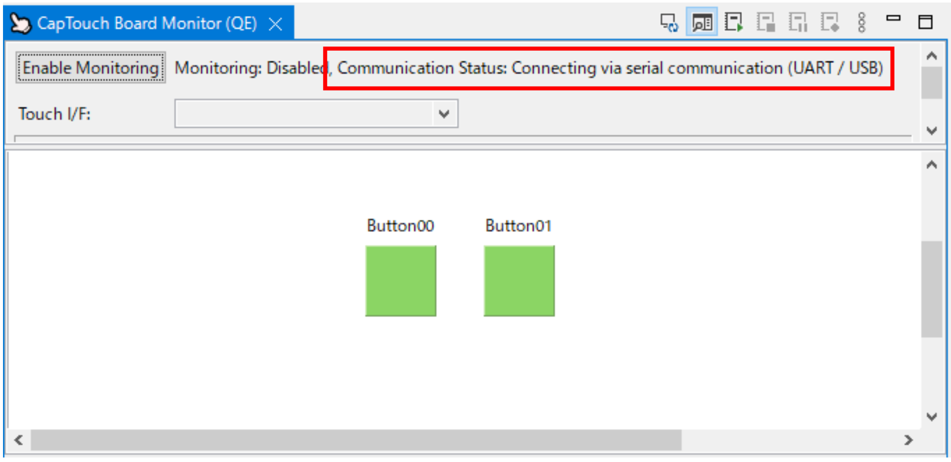
4. When serial connection is executed, the following display will be changed. Confirm a displaying message.



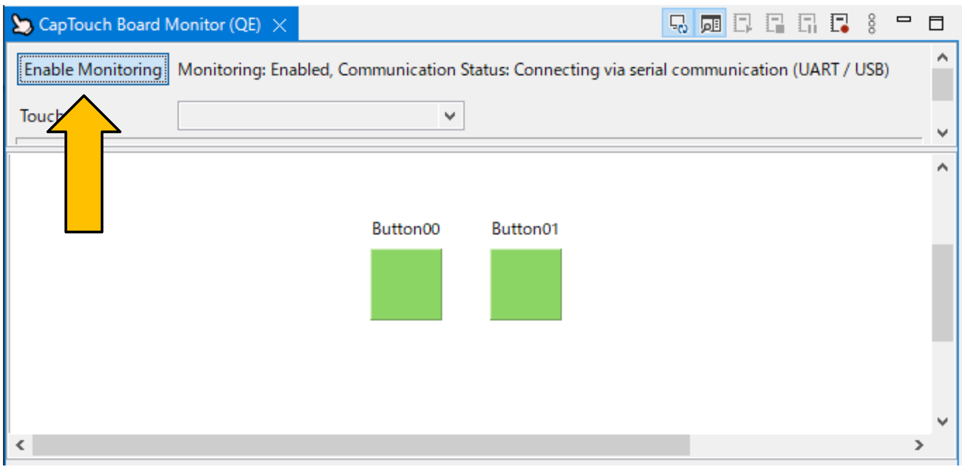
5. Open the Monitoring view from **Show Views** of **CapTouch Main (QE)**.



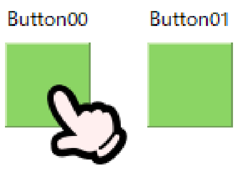
6. **CapTouch Board Monitor (QE)** pane will appear as shown below.



7. Click the **Enable Monitoring** button. The dialog text will change to **Monitoring: Enabled**.

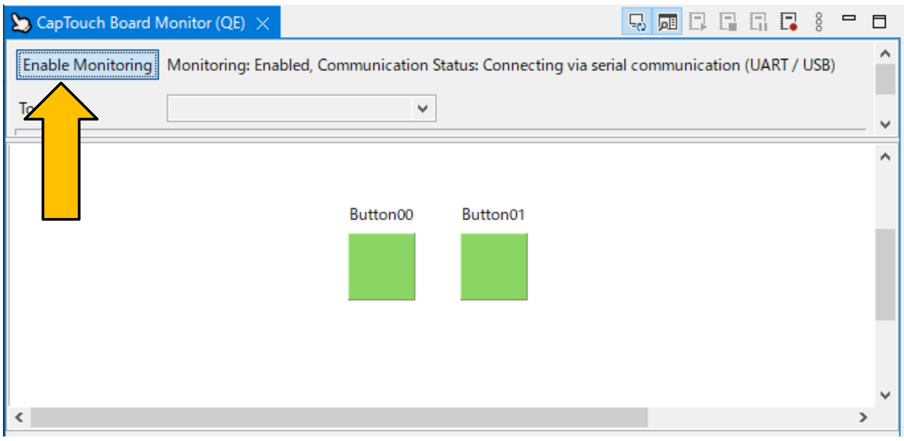


8. Touch the button **TS-B1** on the target board. **The CapTouch Board Monitor (QE)** will show a touch with a finger image on the button like the below image.

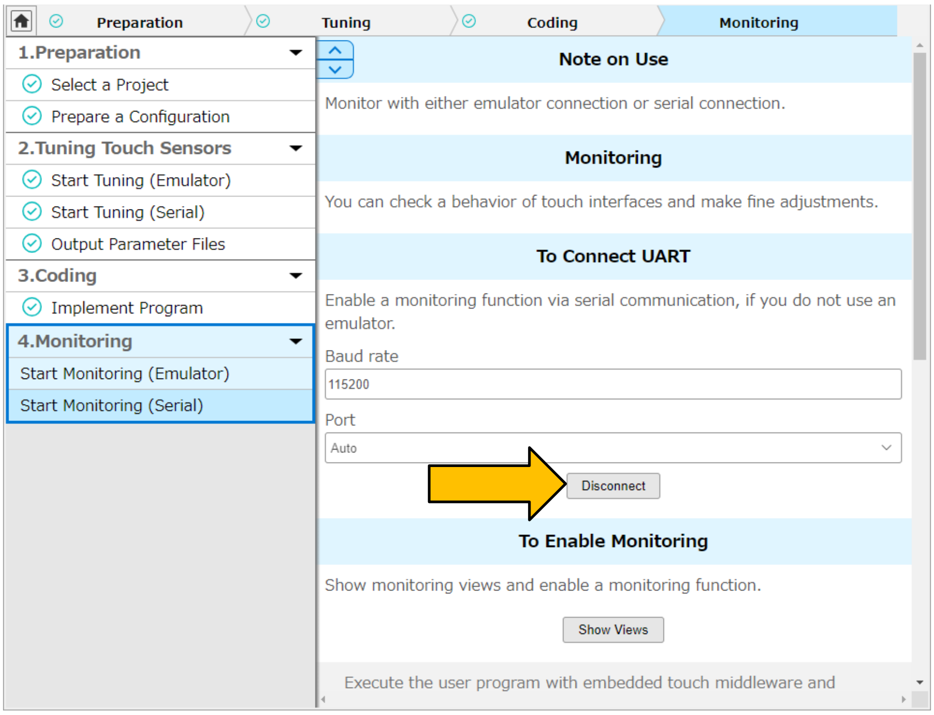


9. The same procedure as in Chapter 12, paragraphs 11-20, can be used to verify that the touch count value changes.

10. If you will finish monitoring, click the **Enable Monitoring** button. The dialog text will change to **Monitoring: Disabled**.



11. If you will finish serial connection, click the **Disconnect** button to disconnect from the serial port.



14. qe_touch_sample.c Listing

```
/***********************************************************************
*
* FILE : qe_sample_main.c
* DATE : 2022-03-09
* DESCRIPTION : Main Program for RX
*
* NOTE:THIS IS A TYPICAL EXAMPLE.
*
***********************************************************************/
#include "qe_touch_config.h"
#if ((TOUCH_CFG_UART_MONITOR_SUPPORT == 1) || (TOUCH_CFG_UART_TUNING_SUPPORT == 1))
#include "r_sci_rx_pinset.h"
#endif
#define TOUCH_SCAN_INTERVAL_EXAMPLE (20)    /* milliseconds */

void R_CTSU_PinSetInit(void);
void qe_touch_main(void);



#if ((TOUCH_CFG_UART_MONITOR_SUPPORT == 1) || (TOUCH_CFG_UART_TUNING_SUPPORT == 1))
#if (TOUCH_CFG_UART_NUMBER == 0)
#define QE_SCI_PIN_SET R_SCI_PinSet_SCI0
#elif (TOUCH_CFG_UART_NUMBER == 1)
#define QE_SCI_PIN_SET R_SCI_PinSet_SCI1
#elif (TOUCH_CFG_UART_NUMBER == 2)
#define QE_SCI_PIN_SET R_SCI_PinSet_SCI2
#elif (TOUCH_CFG_UART_NUMBER == 3)
#define QE_SCI_PIN_SET R_SCI_PinSet_SCI3
#elif (TOUCH_CFG_UART_NUMBER == 4)
#define QE_SCI_PIN_SET R_SCI_PinSet_SCI4
#elif (TOUCH_CFG_UART_NUMBER == 5)
#define QE_SCI_PIN_SET R_SCI_PinSet_SCI5
#elif (TOUCH_CFG_UART_NUMBER == 6)
#define QE_SCI_PIN_SET R_SCI_PinSet_SCI6
#elif (TOUCH_CFG_UART_NUMBER == 7)
#define QE_SCI_PIN_SET R_SCI_PinSet_SCI7
#elif (TOUCH_CFG_UART_NUMBER == 8)
#define QE_SCI_PIN_SET R_SCI_PinSet_SCI8
#elif (TOUCH_CFG_UART_NUMBER == 9)
#define QE_SCI_PIN_SET R_SCI_PinSet_SCI9
#elif (TOUCH_CFG_UART_NUMBER == 10)
#define QE_SCI_PIN_SET R_SCI_PinSet_SCI10
#elif (TOUCH_CFG_UART_NUMBER == 11)
#define QE_SCI_PIN_SET R_SCI_PinSet_SCI11
#elif (TOUCH_CFG_UART_NUMBER == 12)
#define QE_SCI_PIN_SET R_SCI_PinSet_SCI12
#endif
#endif

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;

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

#if ((TOUCH_CFG_UART_MONITOR_SUPPORT == 1) || (TOUCH_CFG_UART_TUNING_SUPPORT == 1))
    /* Setup pins for SCI (function created by Smart Configurator) */
    QE_SCI_PIN_SET();
#endif




    /* 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_MILLISECS);
    }
}
```

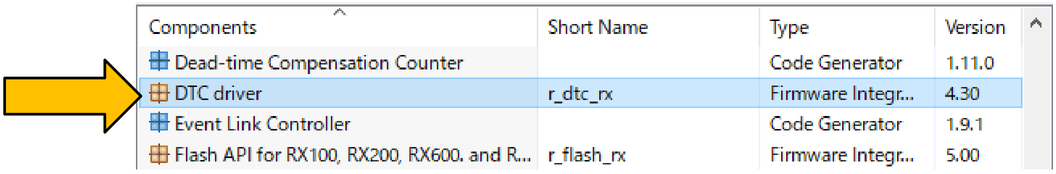
15. [Additional function] Setting The Automatic Judgement and MEC functions

This Chapters shown below describe setting The Automatic Judgement and MEC (Multi Electrode Connection) functions.

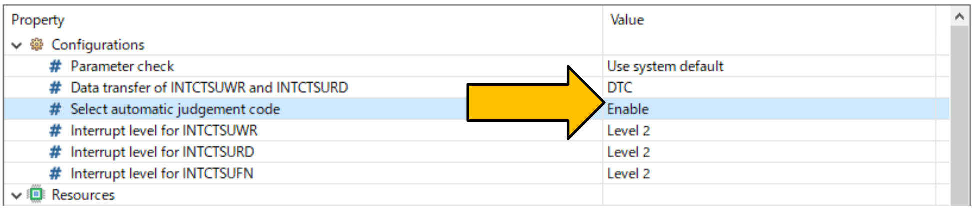
Note: The Automatic Judgement and MEC functions are supported by products equipped with CTSU2SL.

1. Create a project and add modules following the same procedure as described in section 6.11.

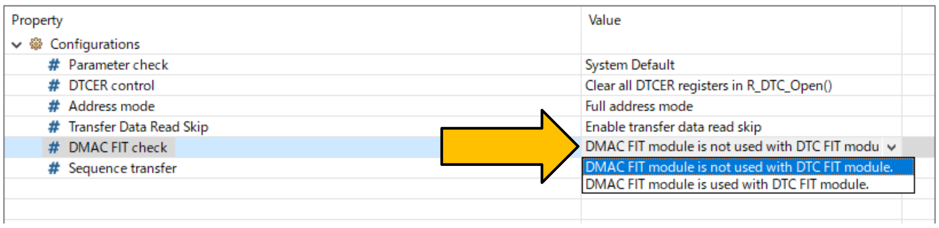
2. Add the code generation component **r_dtc_rx** in the same way as the **rm_touch_qe** as described in section 6.6. Select **r_dtc_rx** and click **Finish**.



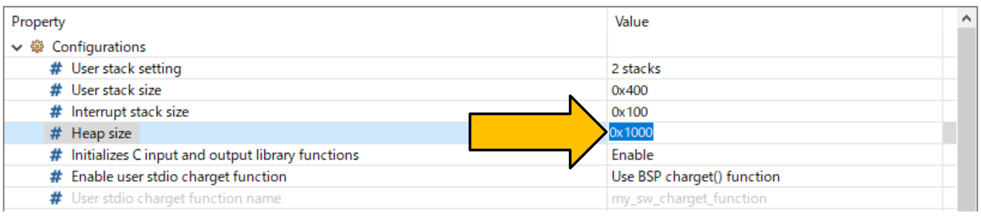
3. Then select **r_ctsu_qe** and display the ports associated with this module in the configuration panel. Set **Data transfer of INTCTSUWR and INTCTSURD**"to **DTC** and **Select automatic judgement code** to **Enable**.



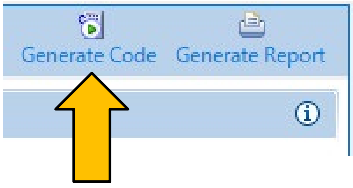
4. Next, select **r_dtc_rx** in the **Components** tab and display the ports associated with this module in the configuration panel. Set **DMAC FIT check** to **DMAC FIT module is not used with DTC FIT module**.



5. Next, select **r_bsp** in the **Components** tab and set **Help size** to **0x1000**.

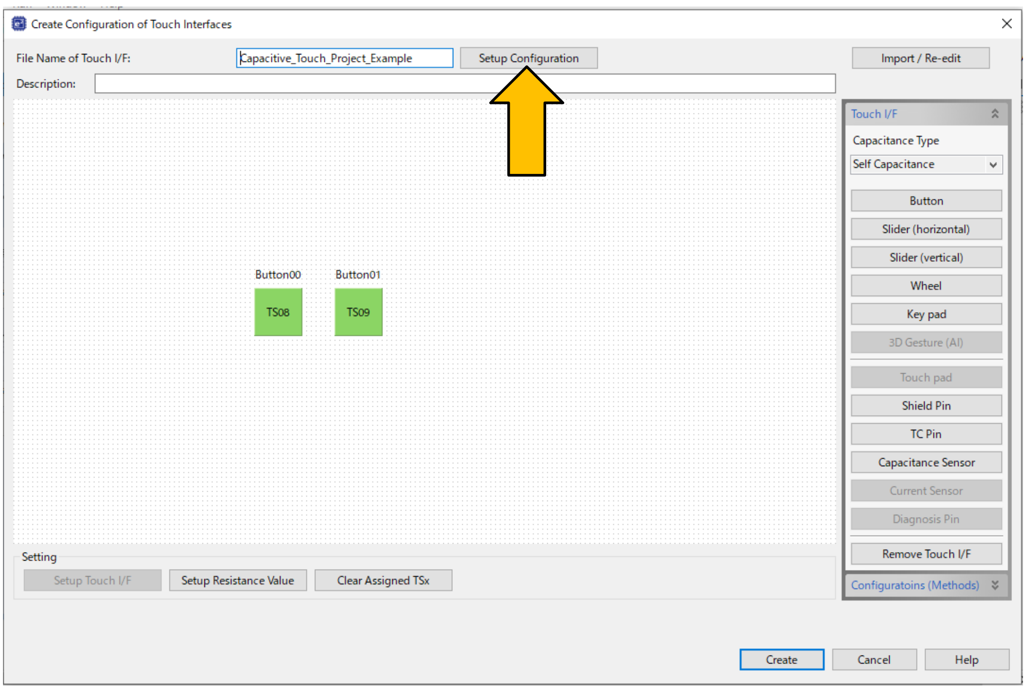


6. Generate the needed application code modules for the project. Do this by clicking the Generate Code icon in the upper-right of the Smart Configurator pane as shown in the picture below.

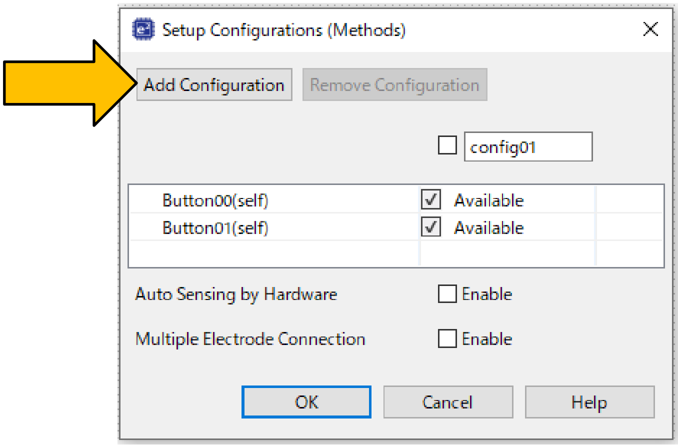


7. Follow the same procedure as in Chapter 8.1 through Chapter 8.7 to set up the touch interface..

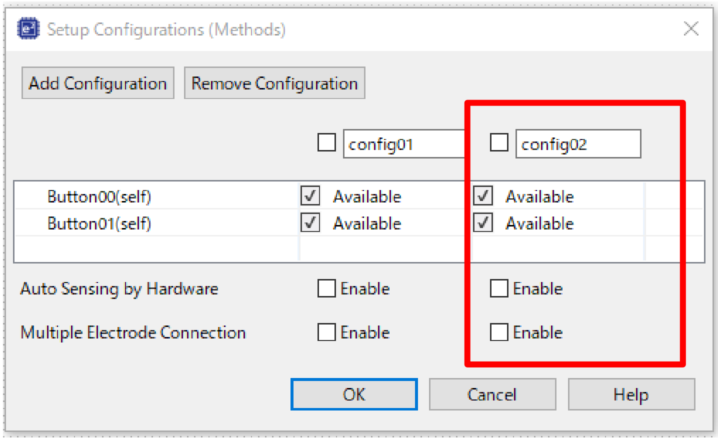
8. Click on **Setup Configuration**.



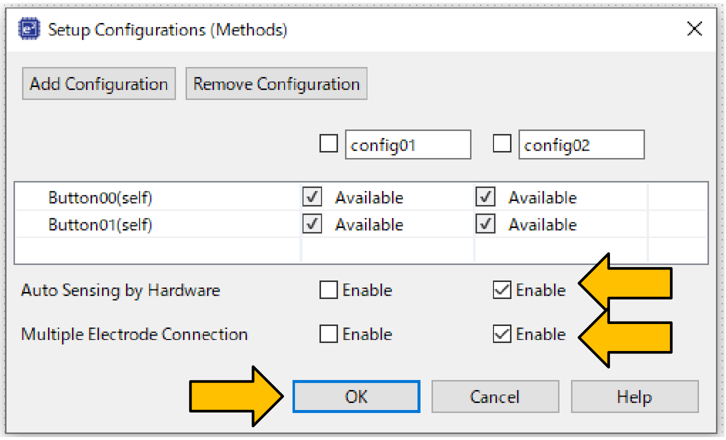
9. The **Setup Configuration (Method)** will appear. Select **Add Configuration** and add a method.



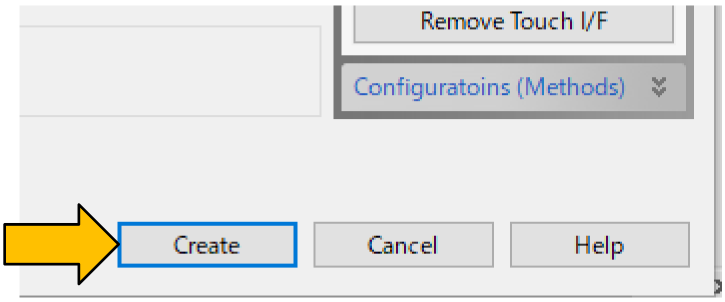
10. The method **config02** is added.



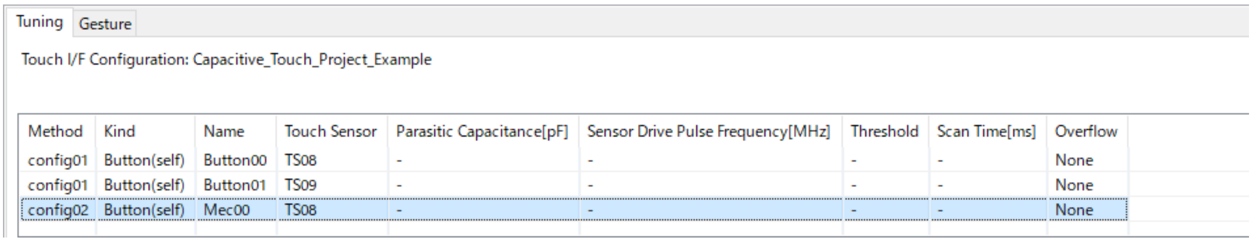
11. Check **Enable** for **Auto Sensing by Hardware** and **Multiple electrode connection** in **config02** and click **OK.**



12. Click **Create** in the **Create Configuration of Touch Interface** window. This will setup the Touch Interface.

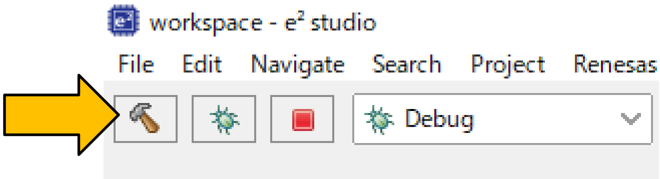


13. You can check the configuration of the touch interface by selecting [Renesas Views] – [Renesas QE] – [CapTouch Tuning Result (QE)] from the menu of the e^2 studio. Open the [CapTouch Tuning Result (QE)] view to display the [Tuning] panel.



Method	Kind	Name	Touch Sensor	Parasitic Capacitance[pF]	Sensor Drive Pulse Frequency[MHz]	Threshold	Scan Time[ms]	Overflow
config01	Button(self)	Button00	TS08	-	-	-	-	None
config01	Button(self)	Button01	TS09	-	-	-	-	None
config02	Button(self)	Mec00	TS08	-	-	-	-	None

14. Build the project using the hammer icon in the upper left-hand side of the IDE. The project should build without any errors or warnings.



15. Thereafter, capacitive touch sensor tuning and monitoring can be performed using the same procedure as in Chapter 9 and beyond.

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/qe-capacitive-touch

Technical Support Contact Form

https://www.renesas.com/support

Inquiries

http://www.renesas.com/contact/

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Oct.17.2018	-	Initial Revision
2.00	Aug.26.2022	-	Updated the development environment
3.00	Sep.26.2023	-	Updated the development environment

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 1Discharge (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/.