RA Family

Using QE and FSP to Develop Capacitive Touch Applications

Introduction

This document will demonstrate the needed steps to create an application example that integrates capacitive touch sensing using Renesas RA Microcontrollers

Target Device

RA family with Capacitive Touch Sensing Unit (CTSU)

Contents

1. Application Example Overview 2

2. Related Documents 2

3. High Level Integration Steps 2

4. Required Development Tools and Software Components 2

5. Application Example Overview 2

6. Project Creation 3

7. Using RA Configurator to Add Modules 4

8. Creating the Capacitive Touch Interface 10

9. Modifying the e^2 studio Project Debug Session for Capacitive Touch Tuning 13

10. Tuning the Capacitive Touch Interface Using QE for Capacitive Touch [RA] Plug-in 14

11. Adding rm_touch FSP Function Calls to Application Example 17

12. Monitoring Touch Performance using e^2 studio Expressions Window and QE for Capacitive Touch [RA] 19

13. qe_touch_sample.c Listing AFTER Modifications 25

Revision History 28

1. Application Example Overview

This document will demonstrate how to implement capacitive touch sensing using Renesas RA Microcontrollers. Using these steps, the user will be shown the process of connecting the RA6M2 MCU board, using the RA Configurator for project creation, and QE for Capacitive Touch [RA] to create, tune and monitor a Capacitive Touch project.

2. Related Documents

This application example is intended to give a user a short introduction to creating a working RA capacitive touch project. A thorough review of all the applicable documentation for the e^2 studio IDE/RA Configurator, Flexible Software Package (FSP) drivers/middleware, Renesas Code Generator, 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 will give the reader an overview of the steps required integrate touch detection into this project. These same steps should apply to any typical user development application.

- Create the initial project using the e^2 studio project creation wizard
- Use the RA Configurator to add the required modules to the created e^2 studio project
- Use the QE for Capacitive Touch [RA] e^2 studio plug-in to create the capacitive touch interface
- Use the QE for Capacitive Touch [RA] e^2 studio plug-in to tune the application project
- Add the needed FSP 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 [RA] e^2 studio plug-in to demonstrate capacitive touch detection

4. Required Development Tools and Software Components

The project utilizes the following development environment:

- EVALUATION Kit (EK-RA6M2)
- Renesas e^2 studio IDE, Version 7.8.0 or later
- GCC ARM Embedded compiler (9.2.1.20191025 or later) installed
- Renesas QE for Capacitive Touch [RA] e^2 studio plug-in, version: 1.1.0 or later
- Flexible Software Package (FSP) version: 1.0.0 or later and Renesas Code Generator

5. Application Example Overview

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

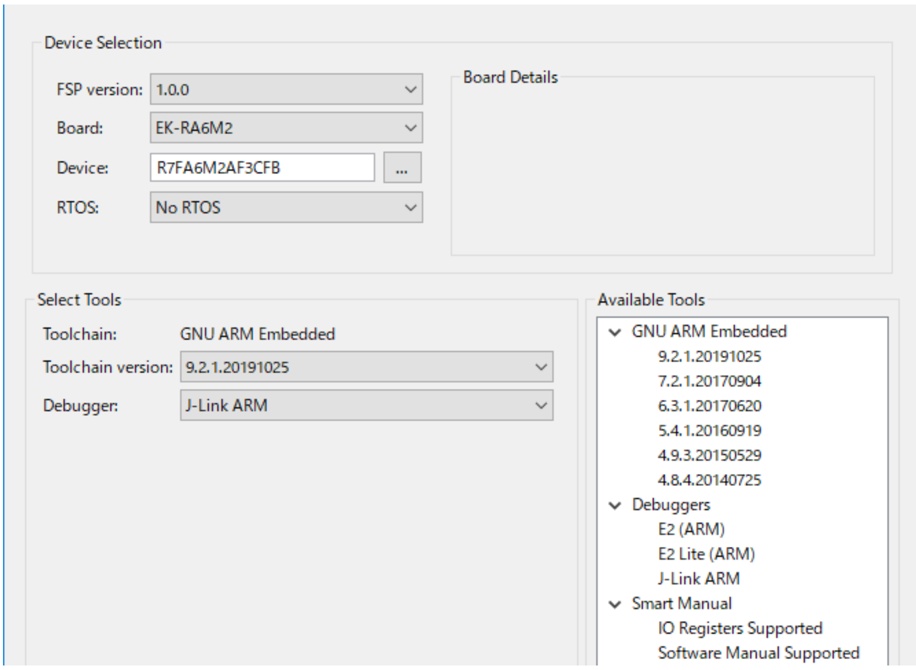
- The global flag used to determine if the rm_touch middleware is ready to be processed is checked
    - If the flag is set (ready to process)
        - The global flag is reset to 0
        - A call to the rm_touch middleware processes data from the previous completed scan, updates needed data, then starts the next touch scan process
        - A call to the rm_touch middleware 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 Appendix A for review.

6. 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 anywhere
2. Start a new project by clicking File->New->RA C/C++ Project
3. At the dialog box that opens, select Renesas RA and highlight with a single-click **Renesas RA 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:
   - FSP version: 1.0.0 or later
   - Board: EK-RA6M2
   - Target Device: R7FA6M2AF3CFB
   - RTOS: No RTOS
   - Toolchain: GCC ARM Embeded
   - Toolchain Version: 9.2.1.20191025
   - Debugger: J-Link ARM

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

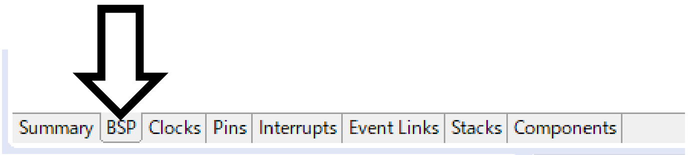


6. Once complete, click Next.
7. In the next dialog box that appears, select Bare Metal - Minimal, then click Finish

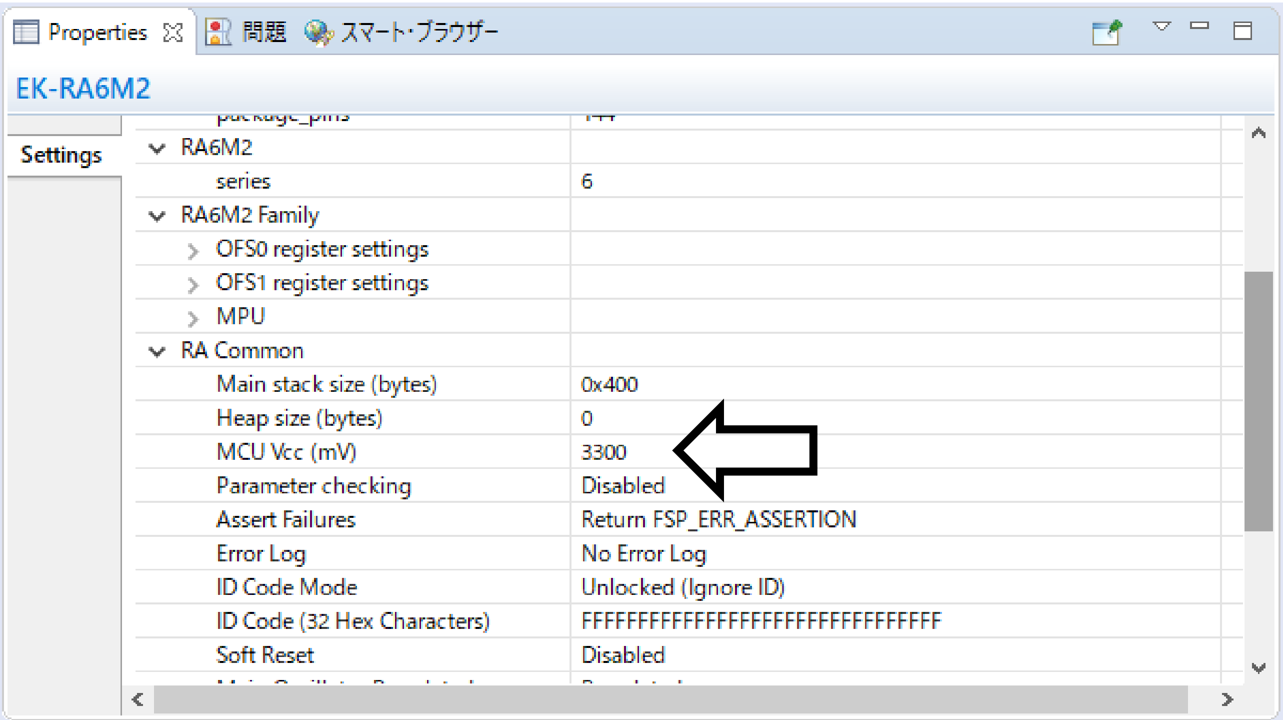
Once complete, a default window will open for the IDE with the RA Configurator perspective open and ready for project configuration. This completes the project creation.

7. Using RA Configurator to Add Modules

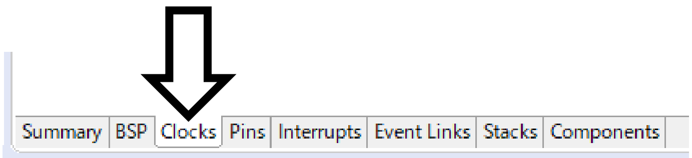
1. Using the tabs in the lower-middle pane of the IDE, select the BSP tab to display the board support package configuration.



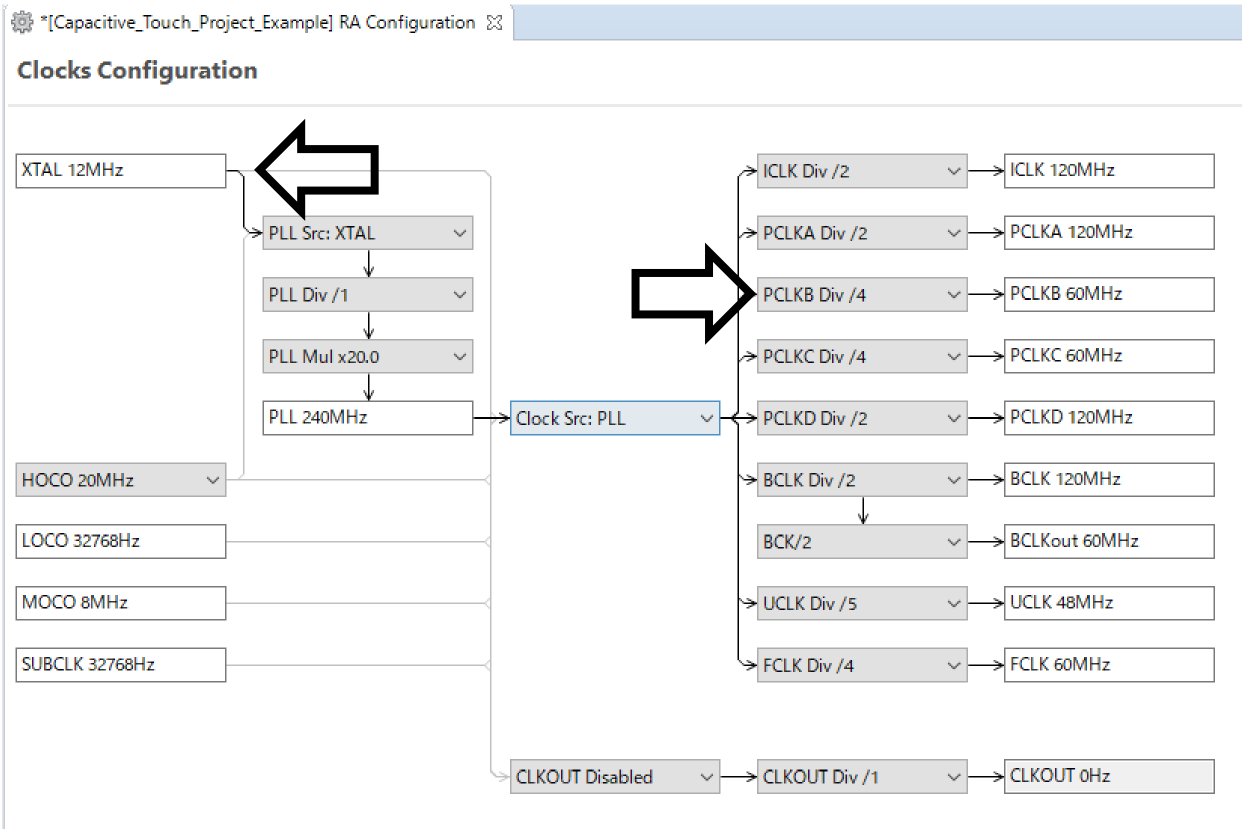
2. Set the power supply voltage with the property displayed at the bottom left of the screen. For this example, the MCU Vcc (mV) will be set to 3300mV.



3. Using the tabs in the lower-middle pane of the IDE, select the Clocks tab to display the clock tree.

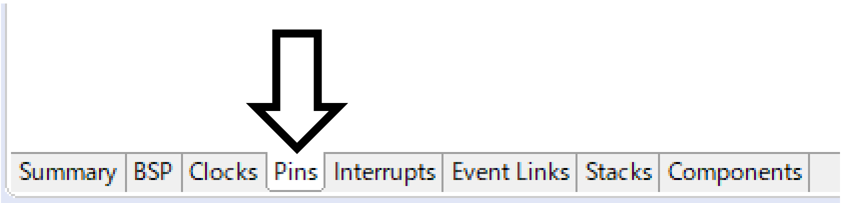


4. For this example, the XTAL will be used as the system clock.

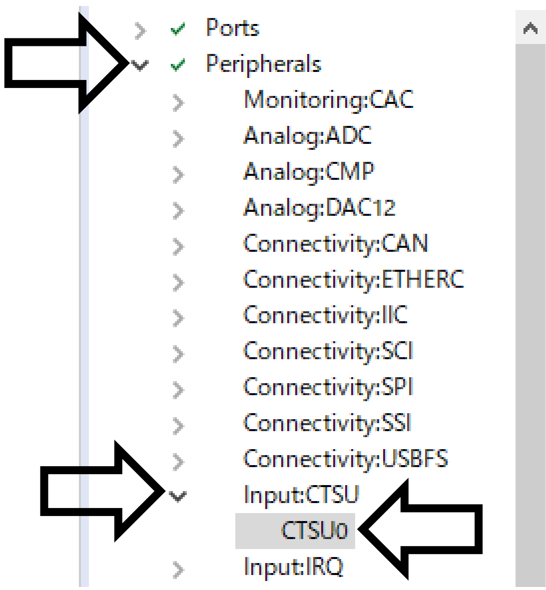


5. Next, Assigns the sensor port of the MCU to connect the button / wheel / slider board. Move to the Pins tab by selecting it at the bottom of the pane.

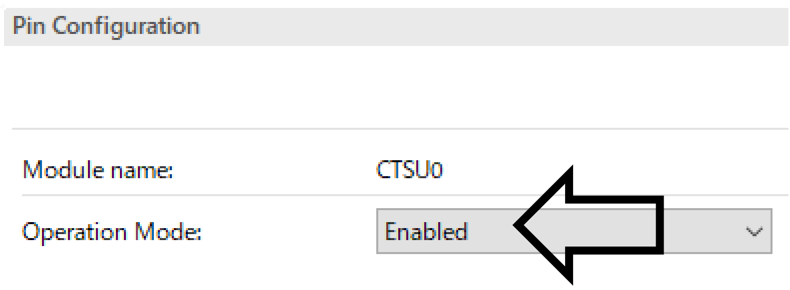
   Note:If you select the EK-RA6M2 board when creating a project, the sensor port is assigned TS2 Pin by default.



6. The pin to use must be selected. Expand the [Peripherals] tree, then expand the [Input:CTSU] entry and select the "CTSU0".

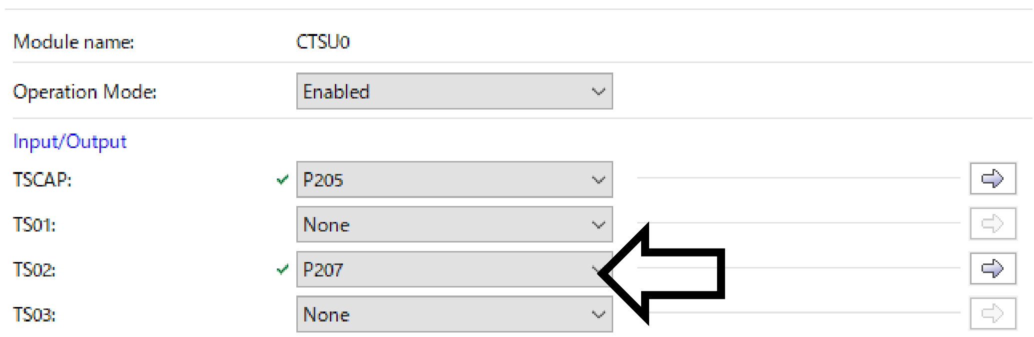


7. Change the Operation Mode setting of the pin configuration from "Disable" to "Enabled".



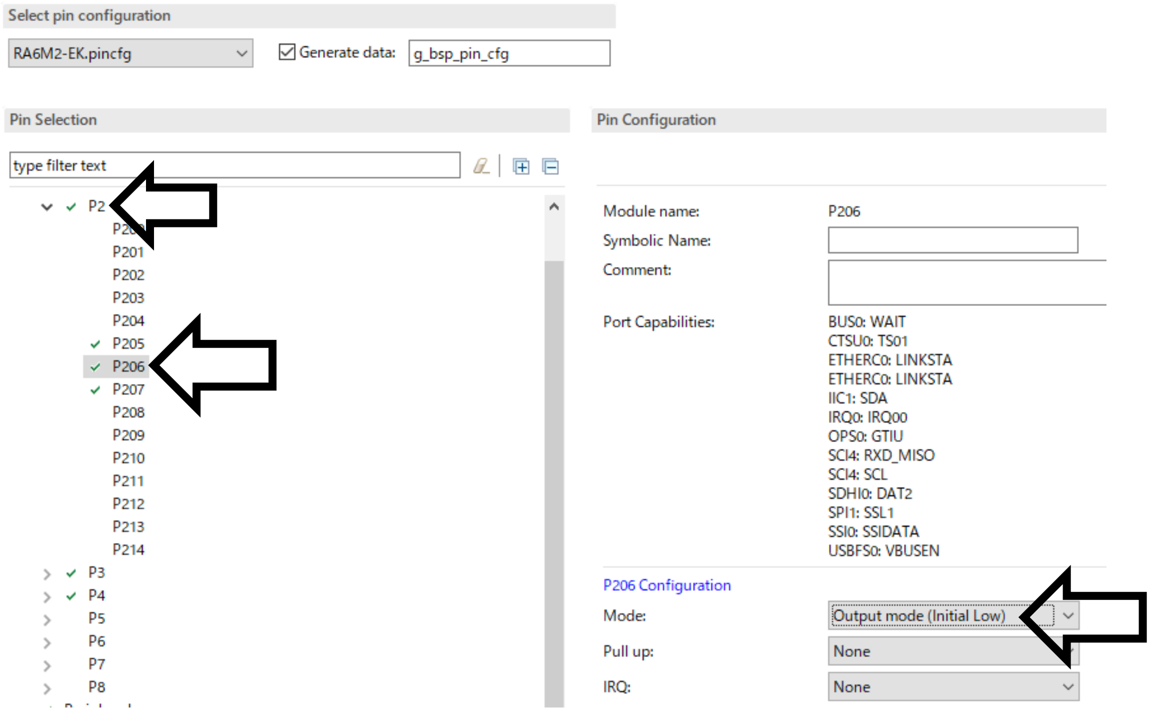
8. Enable the TS pins that you use. This application example will only assign TS2.

- TSCAP Pin
- TS2 Pin

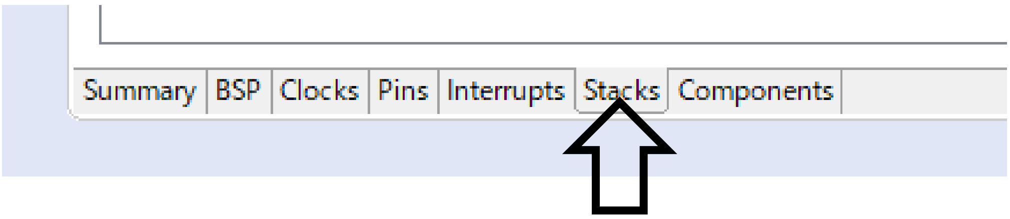


9. Next, the touch sensor pins that are NOT being used by the application are recommended to be setup such that they are driven to ‘OUTPUT’ and ‘LOW’ at startup. Expand the [Ports] tree node. Expand the [P2] sub-node and select the “P206”. Change the Mode setting of the pin configuration from “Disable” to “Output mode (Initial Low)”.

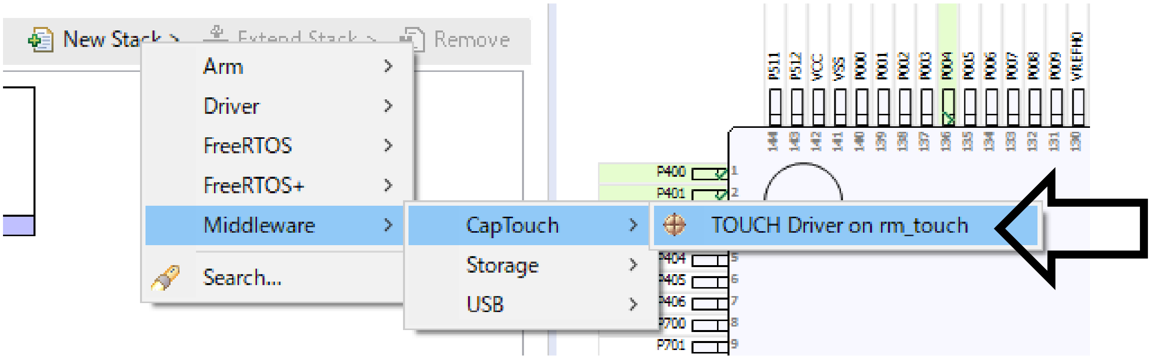
   NOTE: Only one port is setup here as an example of the usage.



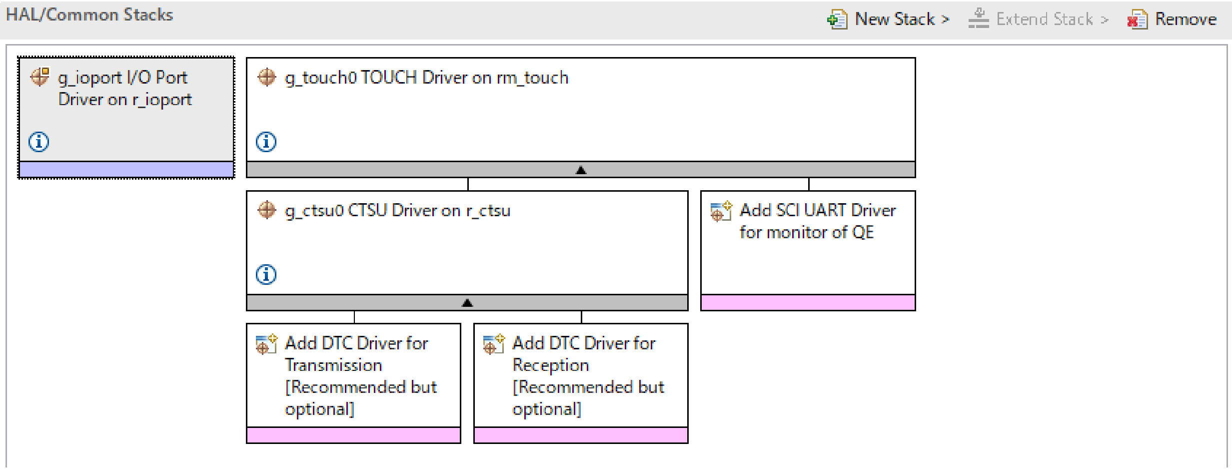
10. Next, move to the Stacks tab by selecting it at the bottom of the pane



11. Select the [New Stack]->[Middleware]->[CapTouch]->[Touch Driver on rm_touch] to add the CTSU driver & middleware

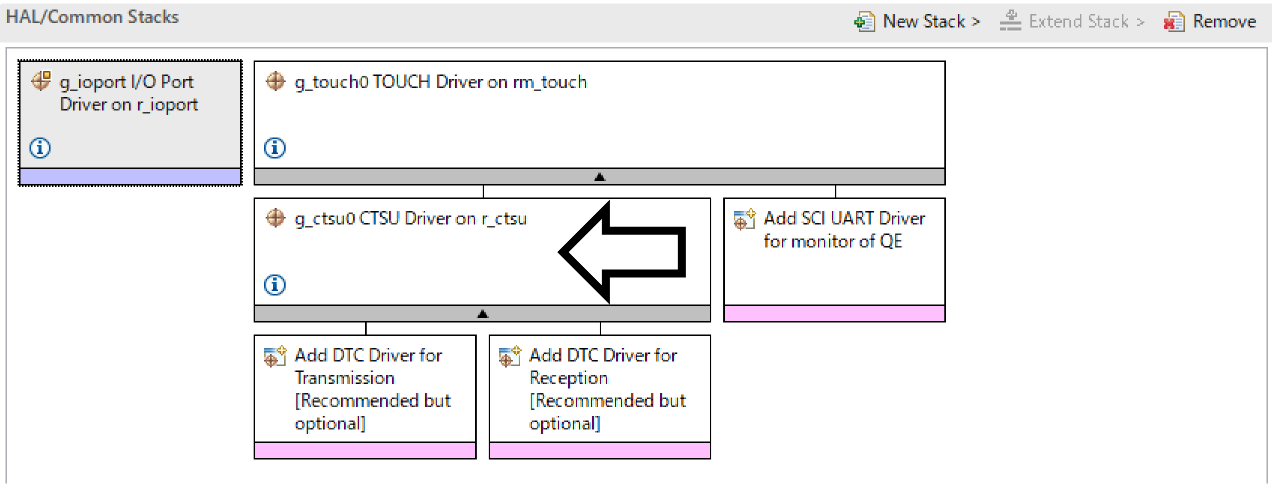


12. HAL/Common Stack will appear as shown in the picture below:

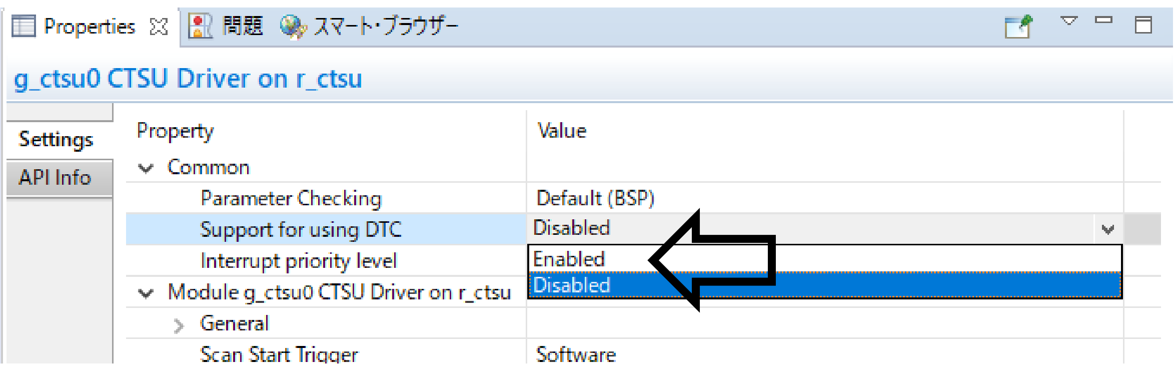


13. Next, click [g_ctsu0 CTSU Driver on r_ctsu] to display properties

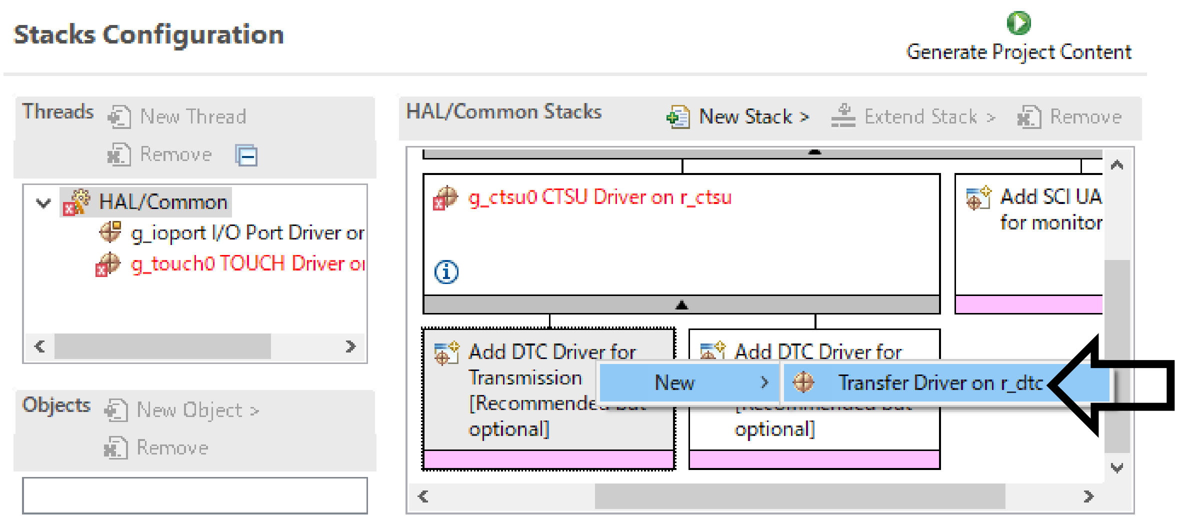
NOTE: Sections 13 to 16 should be set only when using the Data transfer controller（DTC). Transfer data between memory and registers without going through CPU by DTC



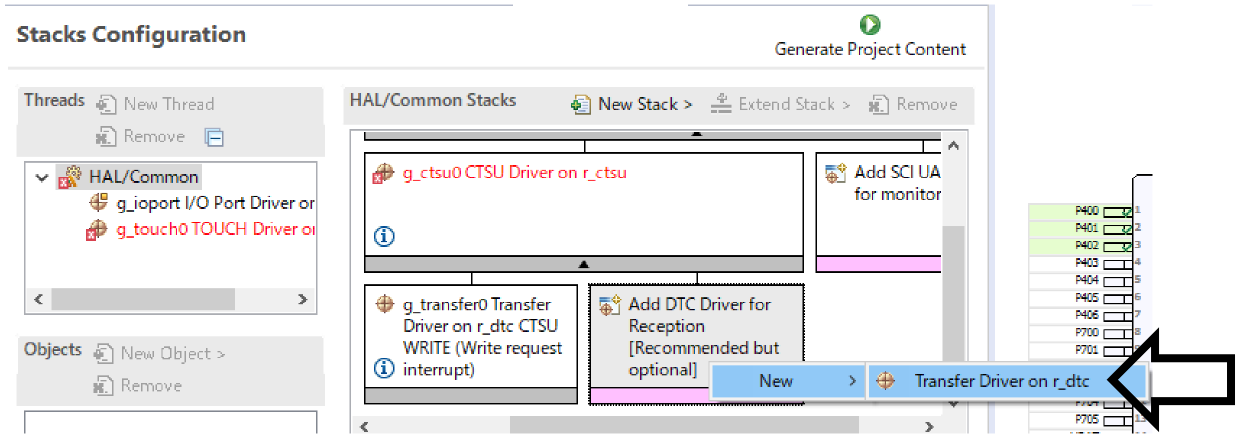
14. Set the DTC with the property displayed at the bottom left of the screen. Change the Support for using DTC setting from “Disable” to “Enable”.



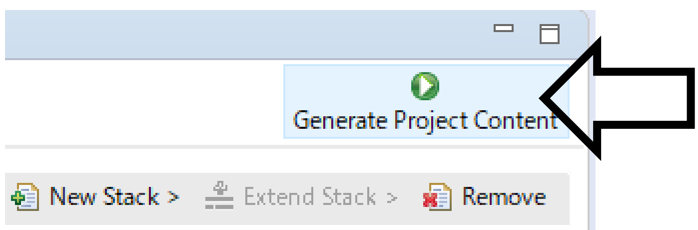
15. Click [Add DTC Driver for Transmission] to select the [New]->[Transfer Driver on r_dtc] to add the DTC driver for Transmission



16. Click [Add DTC Driver for Reception] to select the [New]->[Transfer Driver on r_dtc] to add the DTC driver for Reception

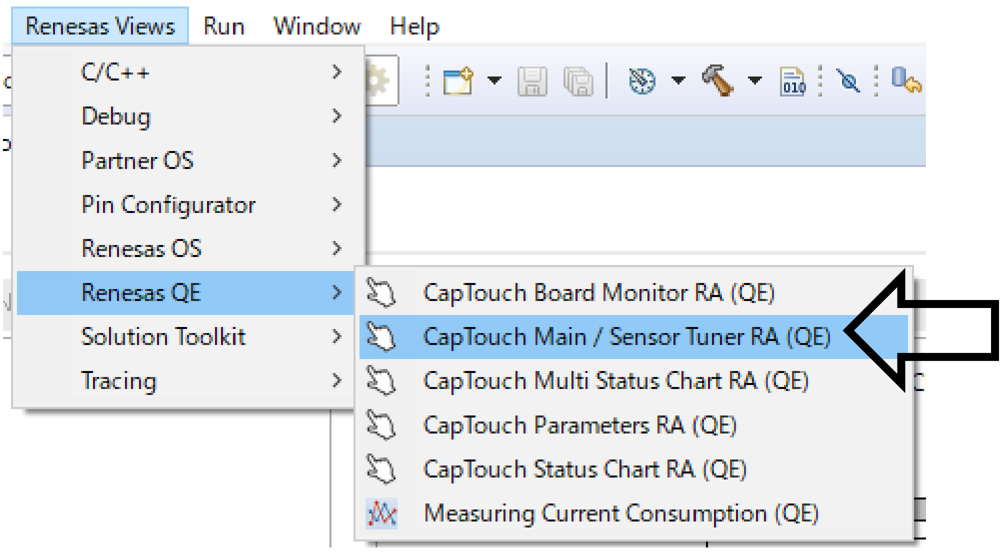


17. At this point, all needed application modules for capacitive touch operations have been added. The final step is to generate the needed application code modules for the project. Do this by clicking the Generate Project Content icon in the upper-right of the Smart Configure pane as shown in the picture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

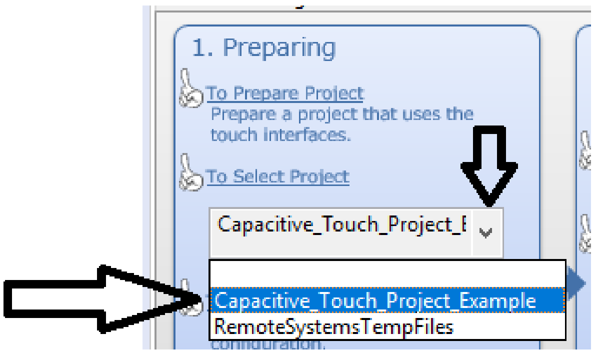


8. Creating the Capacitive Touch Interface

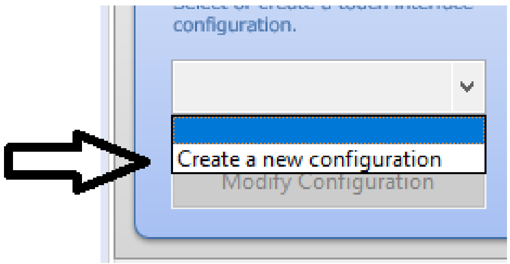
1. From the e[2] studio IDE, use **Renesas Views->Renesas QE->CapTouch Main / Sensor Tuner RA (QE)** to open the main perspecitve for configuring capacitive touch to the project



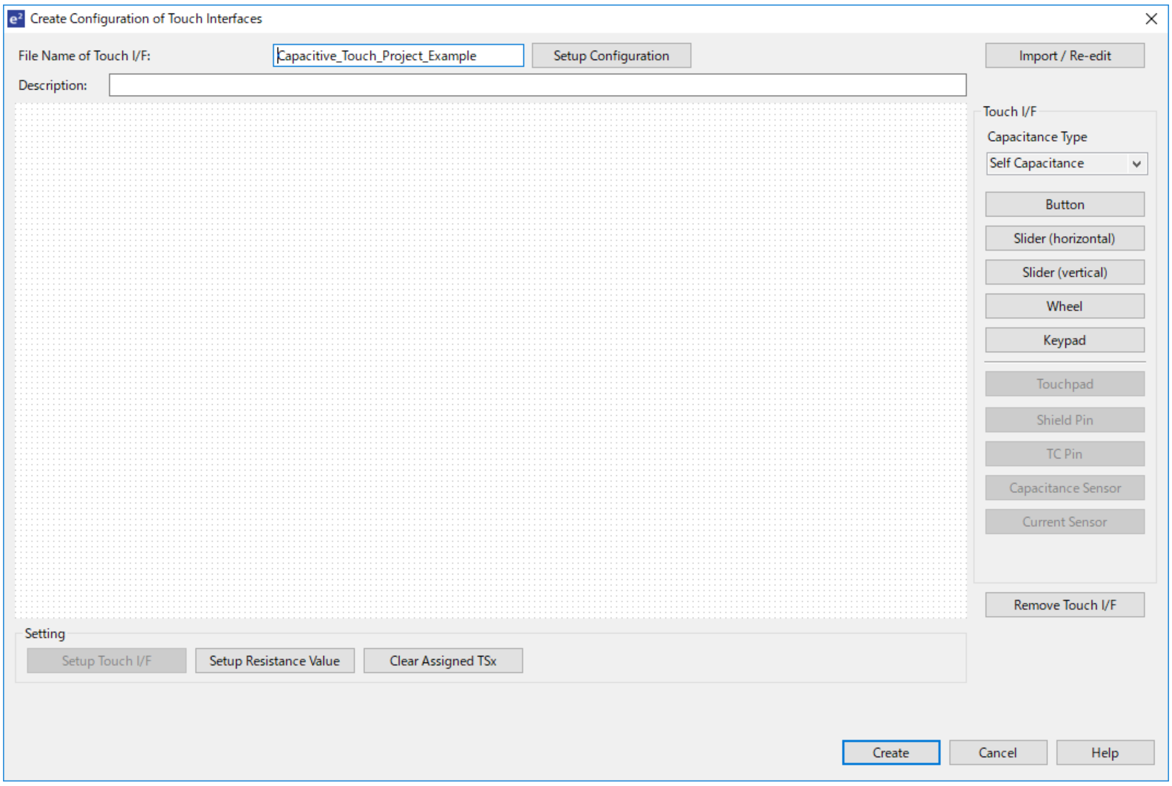
2. In the CapTouch Main / Sensor Tuner RA (QE) pane, select the project to configure the touch interface for by using the pull-down tab and selecting the **Capacitive_Touch_Project_Example** project as shown below



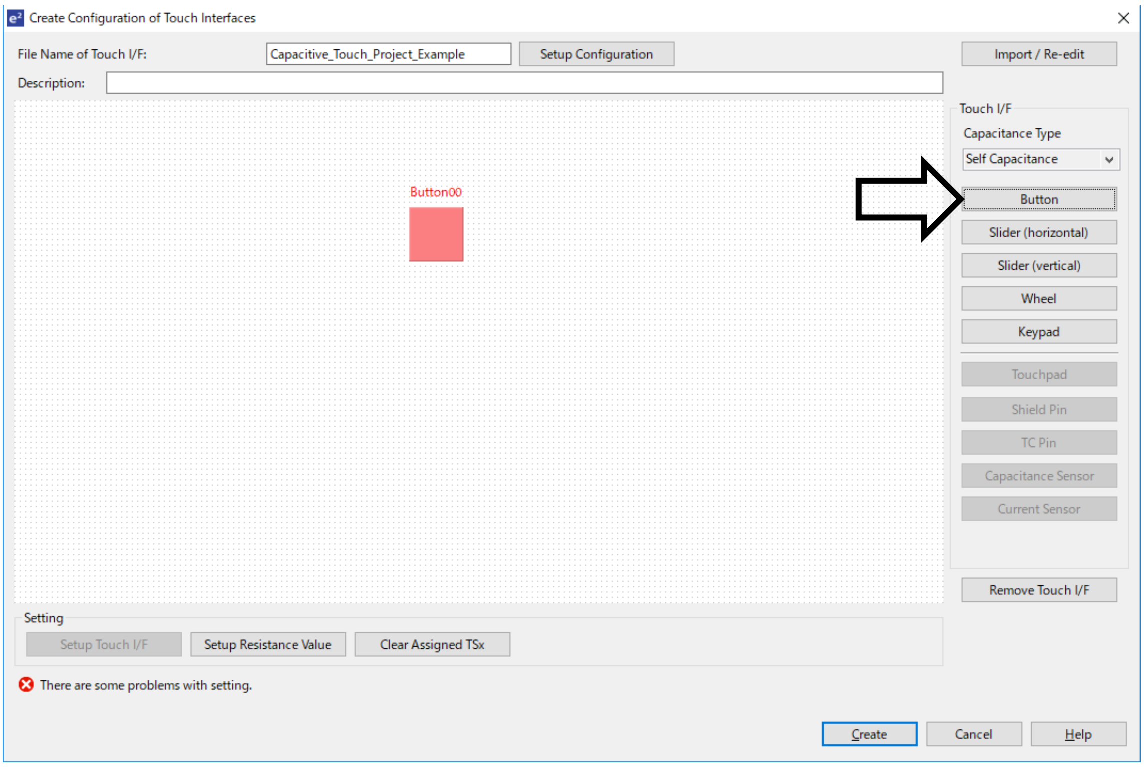
3. Next, create a new touch configuration by using the lower pull-down and selecting **Create a new configuration**.



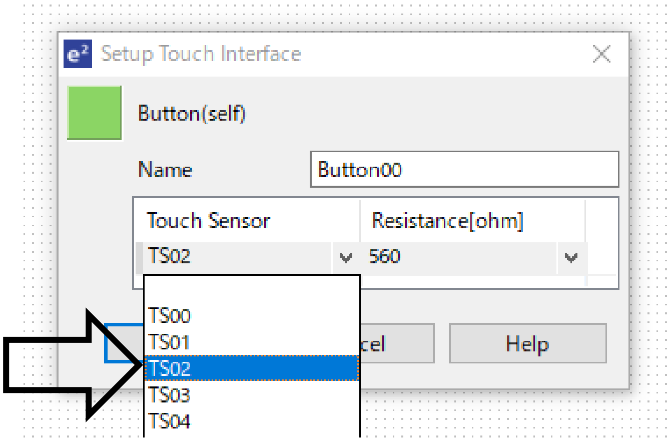
4. A new menu window will open with shows the default blank canvas for creating the touch interface



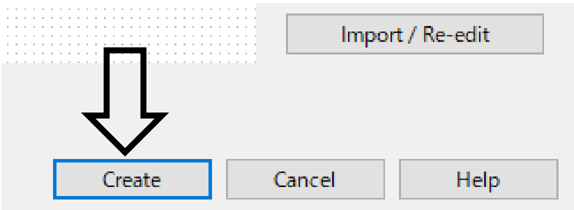
5. Add 1 buttons to the canvas by selecting the **Button** menu item from the right-hand side and adding one(1) button to the canvas. Press the **ESC** key to exit once one button is added. The canvas will look similar to below



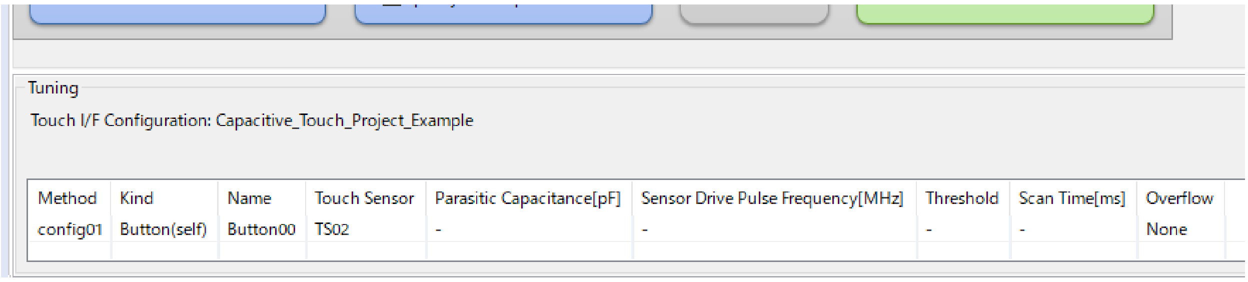
6. To make this connection, double-click on **Button00** and a dialog box will appear. In this case, using the pull-down, select **TS02** as the MCU sensor to assign to this button. Note also, the indication of a configuration error will go away once all assignment are made properly and correspond to the enabled channels in the RA Configurator



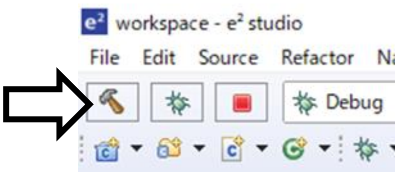
7. Click **Create** in the dialog box. This will setup the Touch Interface.



8. The CapTouch Main / Sensor Tuner RA (QE) window will now display the configuration of the touch interface in the main view pane

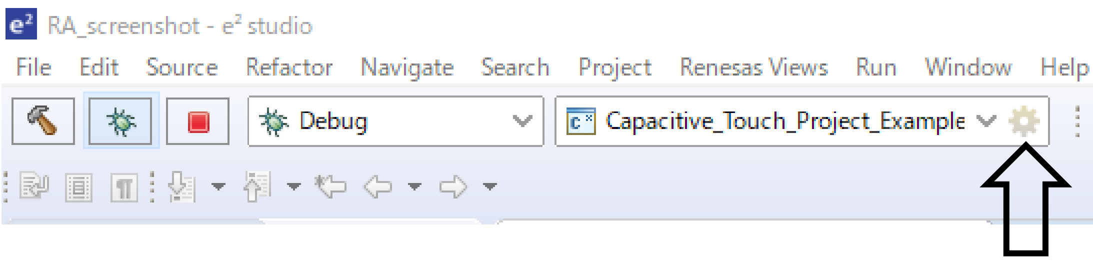


9. Build the project using the hammer icon in the upper left-hand side of the IDE. The project should build without any errors or warnings

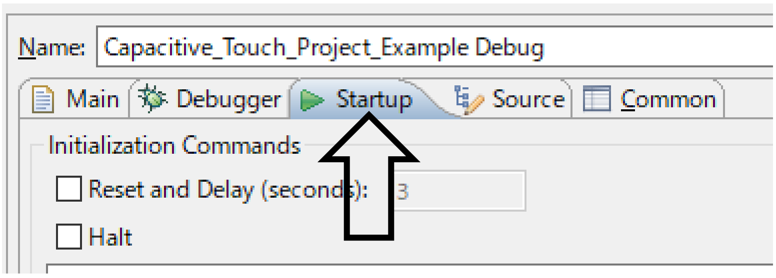


9. Modifying the e² studio Project Debug Session for Capacitive Touch Tuning

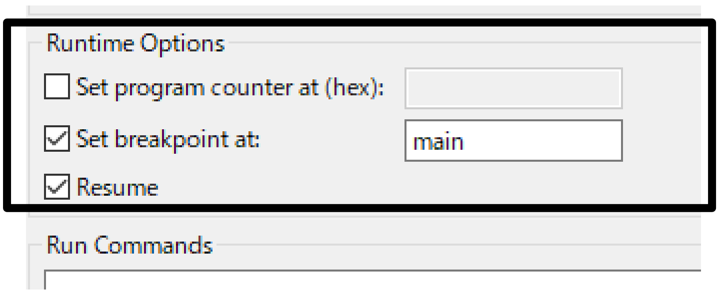
1. The debug session needs to be modified slightly so that a special tuning kernel can be downloaded into the MCU RAM after the debug session starts. Enter the Debug Configuration by clicking the gear icon in the upper left-hand side of the IDE.



2. Select the **Startup** tab



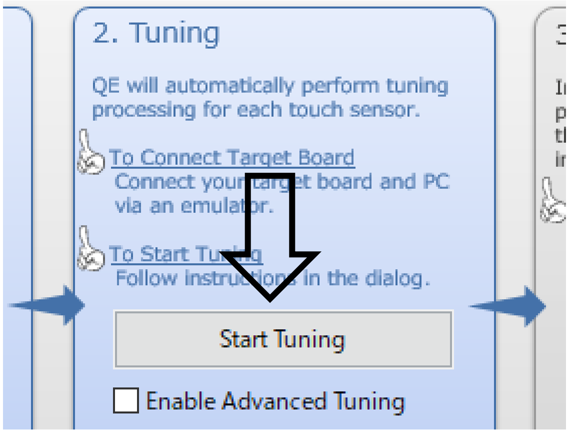
3. Ensure the two check boxes **Set breakpoint at: AND Resume** are checked and look as follows. You may need to scroll down in the dialog box to see these check boxes



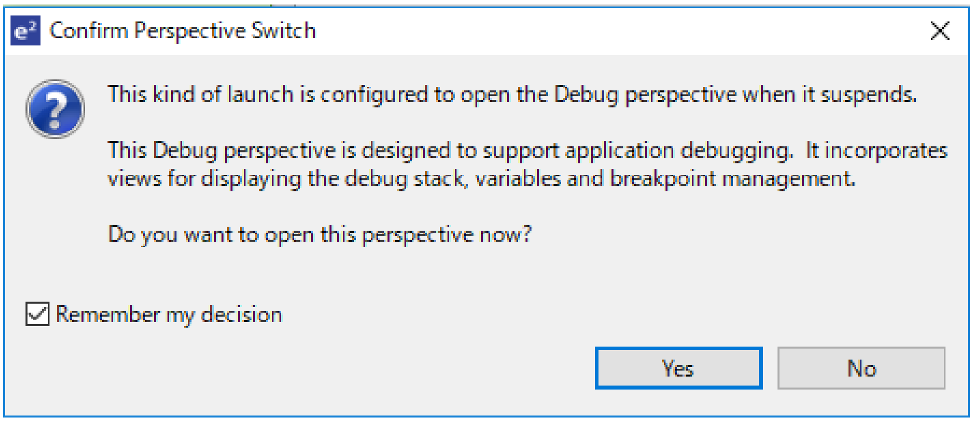
4. Click **Apply** then **Close** to use these modified settings. This completes the project configuration and debug setup for tuning

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

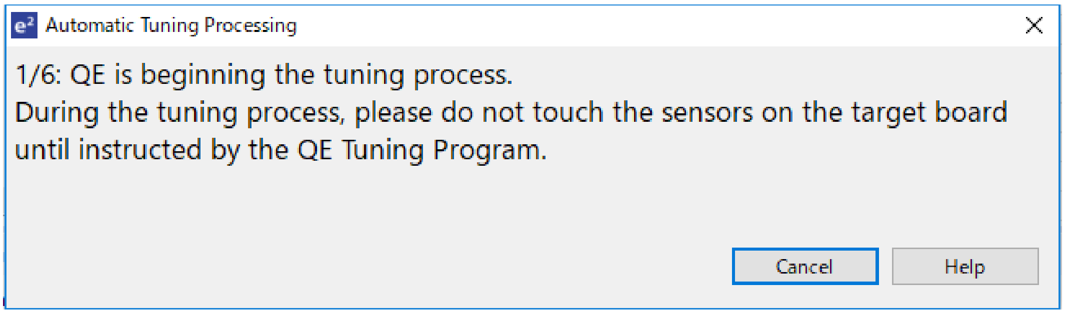
1. To start the automatic tuning process, click the button **Start Tuning** in the **CapTouch Main / Sensor Tuner RA (QE)** e^2 studio IDE.



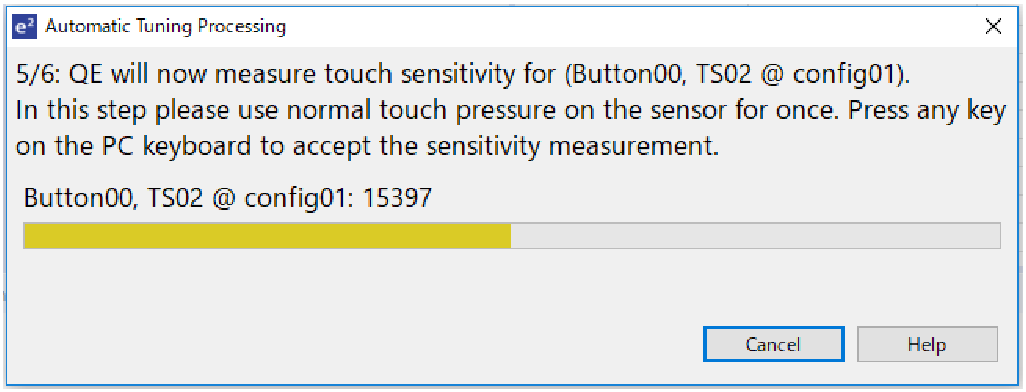
2. At the start of the first debug session, e^2 studio **MAY** display a message indicating the Debug perspective will be switched to. Click the **Remember my decision** check box and **Yes** to continue the Debug process and the QE for Capacitive Touch [RA] automatic tuning



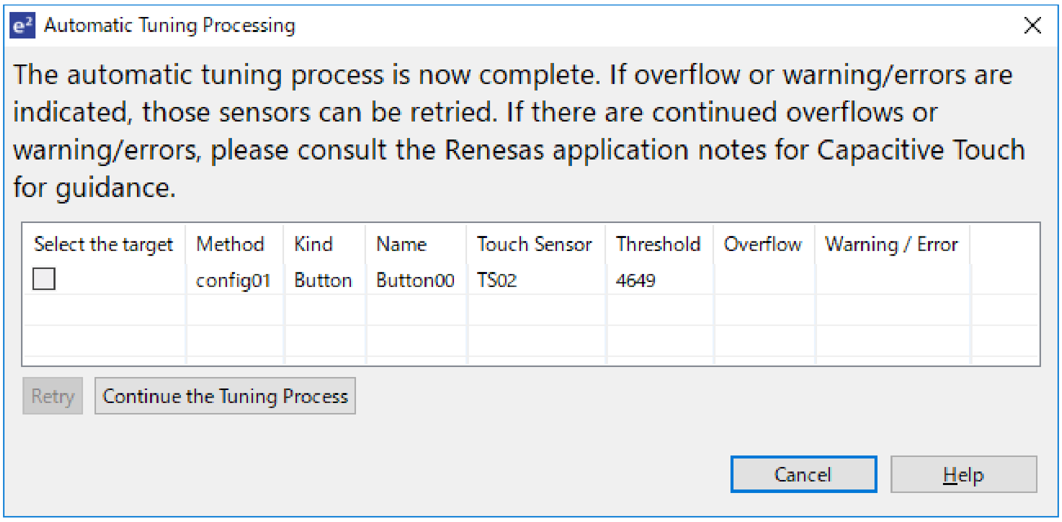
3. QE for Capacitive Touch [RA] automatic tuning will now begin. Please carefully read the tuning dialog windows as they will guide you thru the tuning process. An example screen is shown below. Typically, no interaction is required during the initial tuning process steps



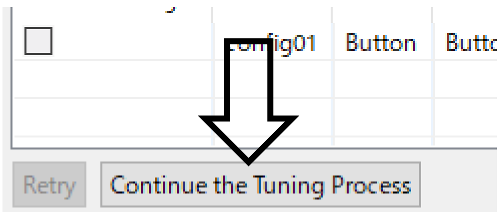
4. After several automated steps, a dialog box with information similar to what is shown below will appear. This is the **touch sensitivity measurement** step of the tuning process. As the first 'interactive' step of the tuning process, press using **normal touch pressure** on the sensor being indicated in the dialog box (Button00/TS02). 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.



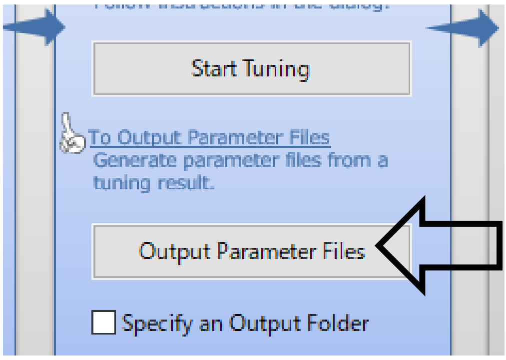
5. Once sensitivity measurement for the buttons is complete, you will see a screen like what is shown below. This is the detection threshold that is used by the middleware to determine if a touch event has occurred.



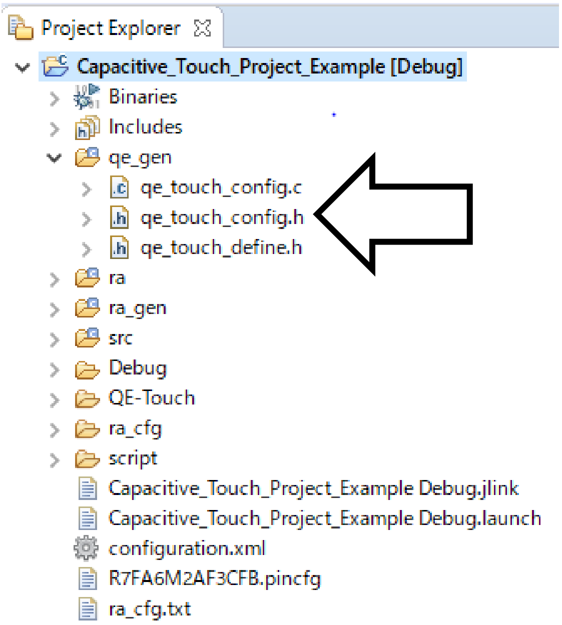
6. Click the **Continue the Tuning Process** button in the dialog box shown. This will exit the tuning process and disconnect from the Debug session on the target. You should return to the default Cap Touch Main / Sensor Tuner RA (QE) screen in the e^2 studio IDE.



7. In this example, all that is left is to output the tuning parameter files. Click the button **Output Parameter Files**



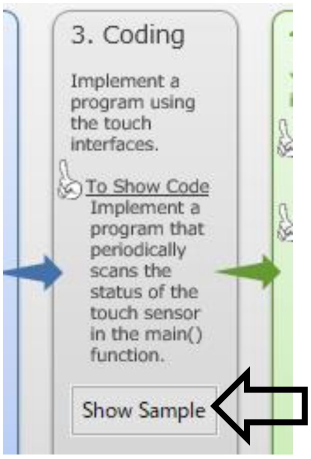
8. In the **Project Explorer** window and you will see that **qe_touch_config.c, qe_touch_config.h and qe_touch_define.h** files have been added. These contain the needed tuning information to enable touch detection using the module of FSP



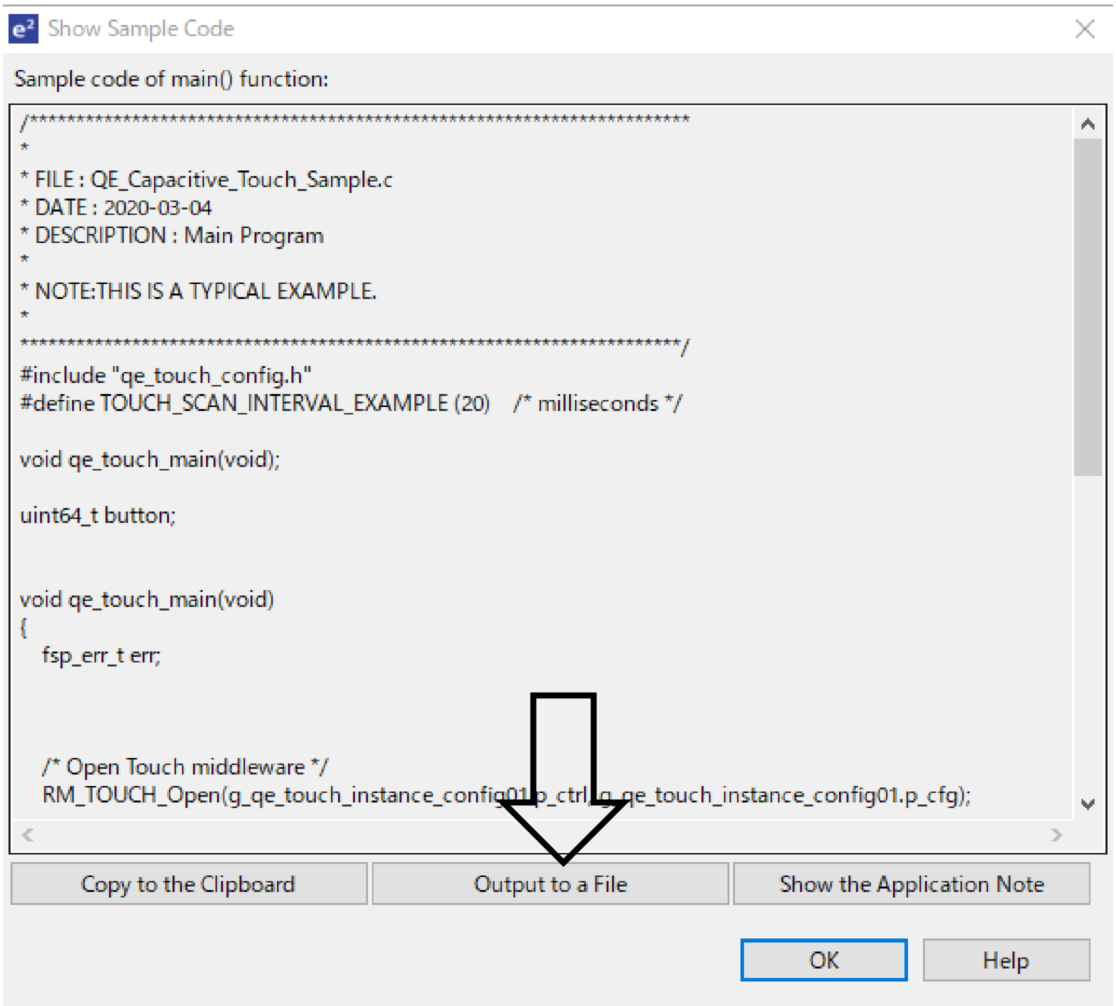
9. Build the project using the hammer icon in the upper left-hand side of the IDE. The Console output showing build results should show no errors

11. Adding rm_touch FSP Function Calls to Application Example

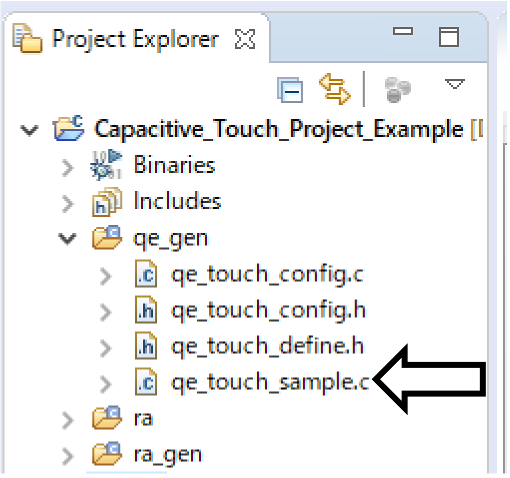
1. To implement application code to scan and report the state of the the touch sensor, click the button **Show Sample** in the **CapTouch Main / Sensor Tuner RA (QE)** e² studio IDE



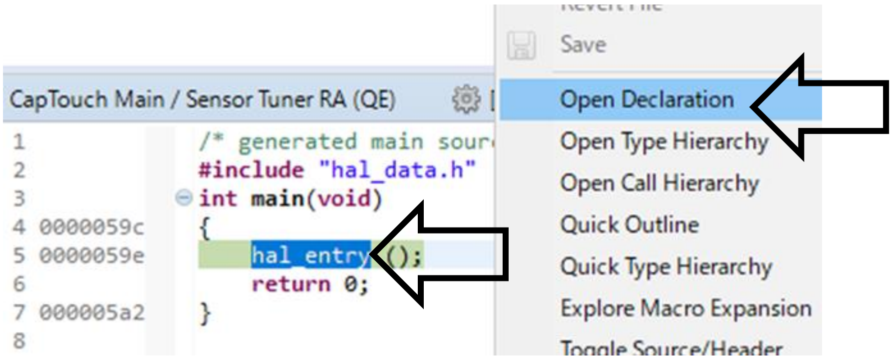
2. A new menu window will open with shows the sample code in text. Click the button **Output to a File**



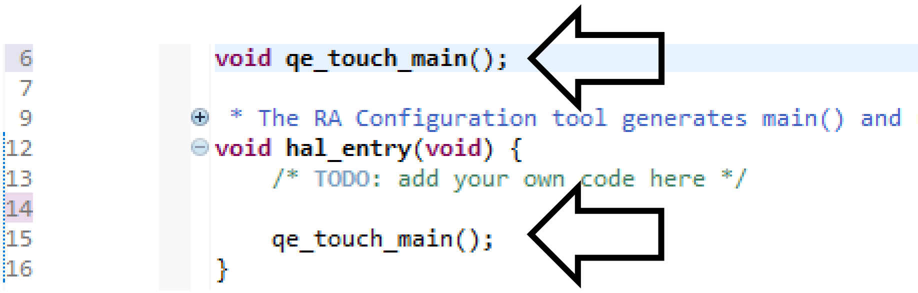
3. Created a new project file that describes the sample code. In the Project Explorer window and you will see that qe_touch_sample.c files have been added.



4. Open the declaration of **hal_entry** () function that is called within the **main()** function



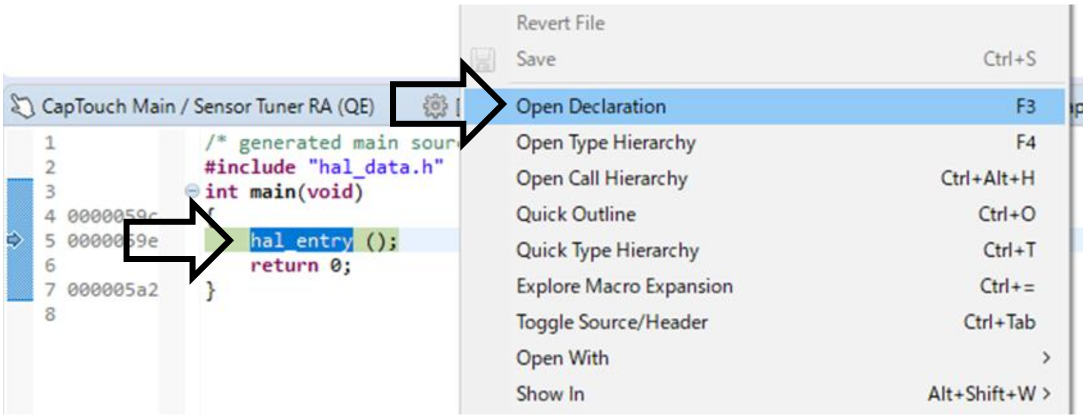
5. In the **hal_entry**(), call the **qe_touch_main**() with main program



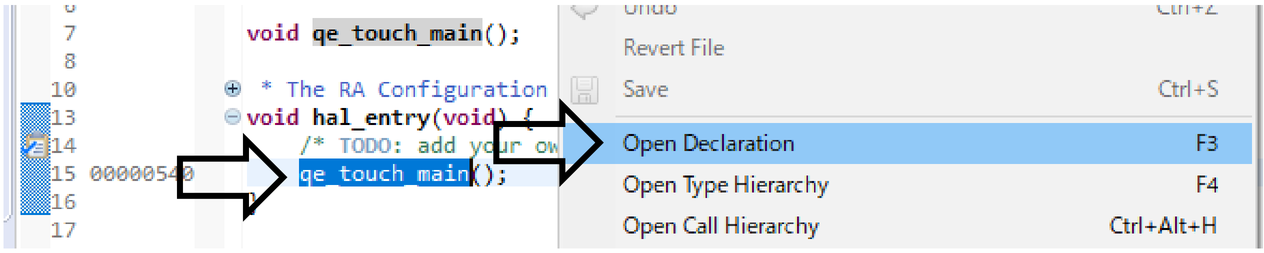
6. This completes all the needed code modifications required for this simple application example. Building the code should result in no errors or warnings for this simplified application example.

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

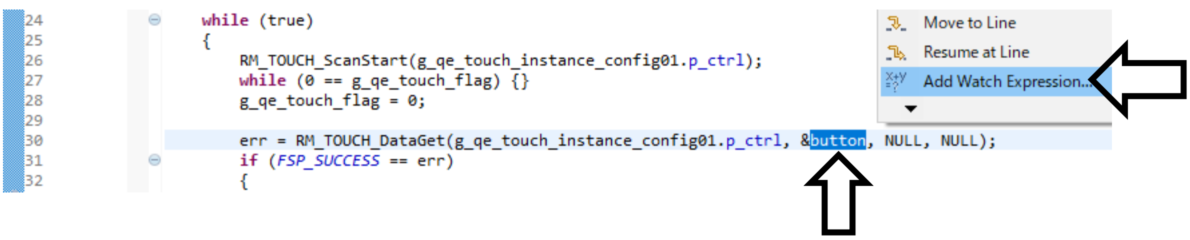
1. Start a Debug session by clicking the **Bug** icon in the upper left-hand corner of e² studio. A Debug session will commence
2. The debugger will stop at the **hal_entry ()** function call. This is the first code point in the **main()** function.
3. Open the declaration of **hal_entry** () function.



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

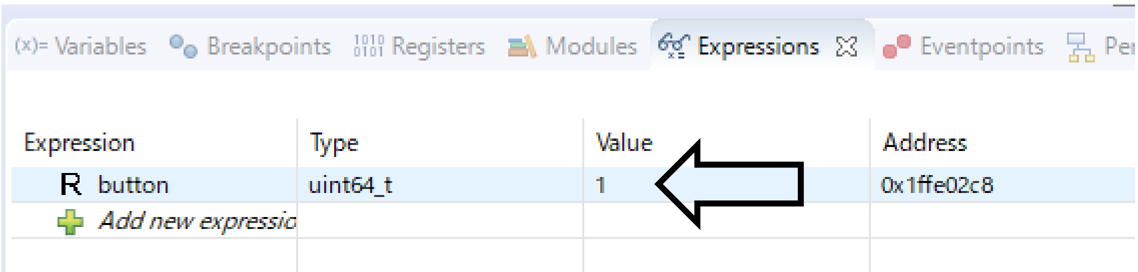


5. Scroll down in the hal_entry.c file to the **RM_TOUCH_DataGet ()** function in the **while (true)** loop. Add the variable **button** to the expressions window

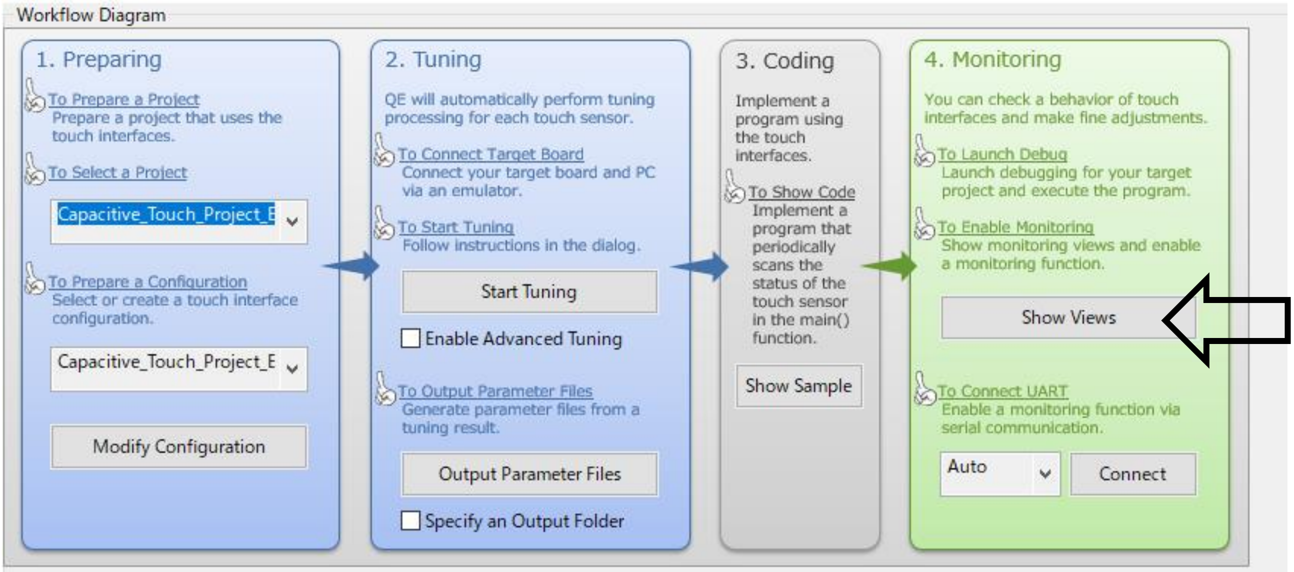


6. Enable **Real-Time Refresh** on the variable in the Expressions window
7. Click the '**Resume**' button located approximately in the middle of the e² studio menu bar to continue code execution

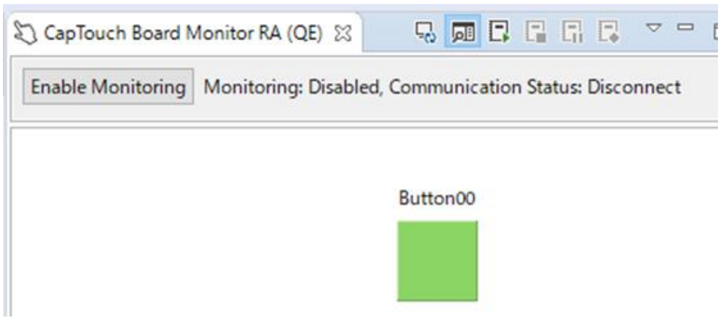
8. Press TS2 on the board, which was configured as Button00 in Section 7 of this application guide. When pressed, a ‘1’ will appear for **button** in the Expression window, indicating a binary indication of touch



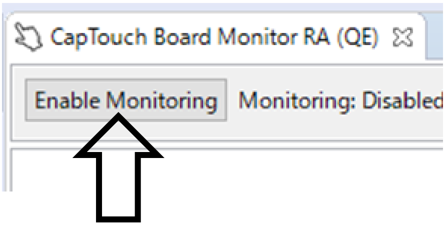
9. Open the Monitoring view from **Show Views** of **CapTouch Main / Sensor Tuner RA (QE)**



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

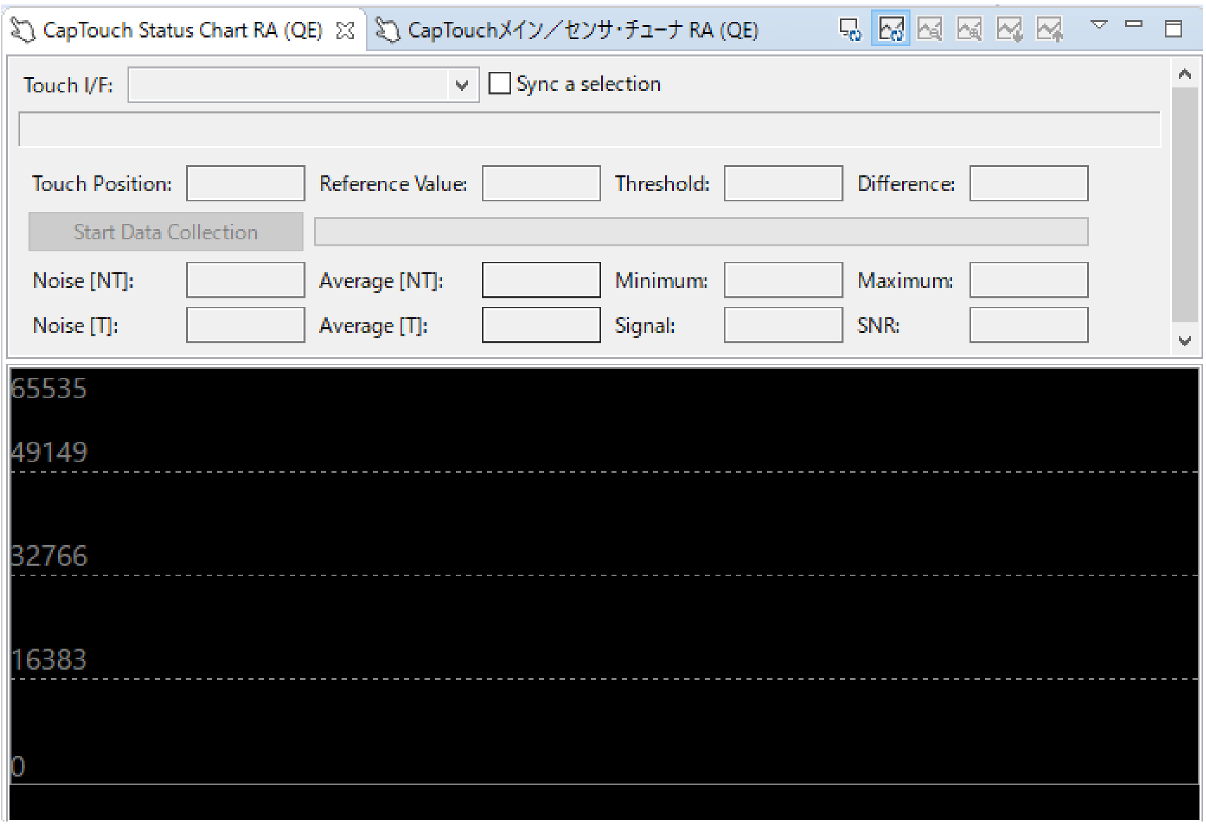


11. Click the **Enable Monitoring** button. The dialog text will change to **Monitoring: Enabled**

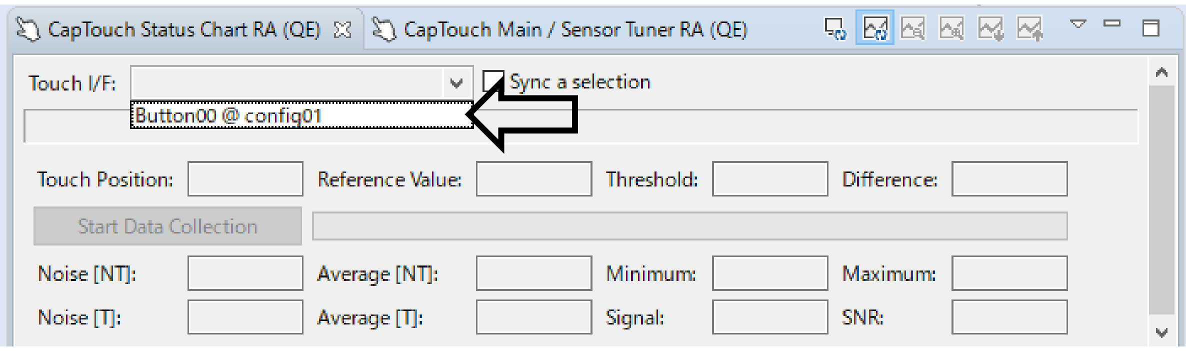


12. Touch the button **TS2** on the EK-RA6M2 board. The **CapTouch Board Monitor RA (QE)** will show a touch with a finger image on the button like the below image.

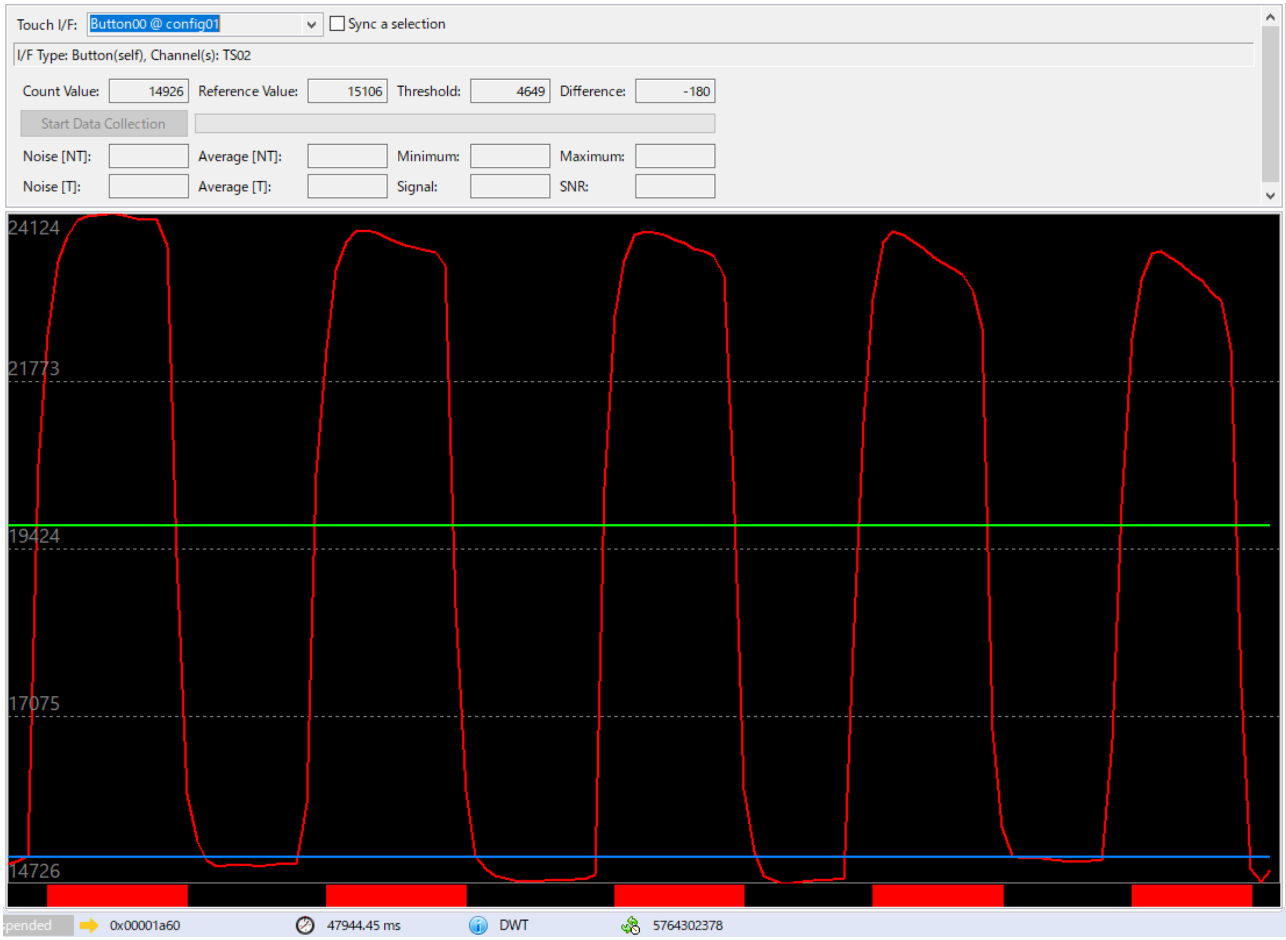
13. To see a graphical representation of the ‘touch counts’ from the board, use the **CapTouch Status Chart RA (QE)**



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

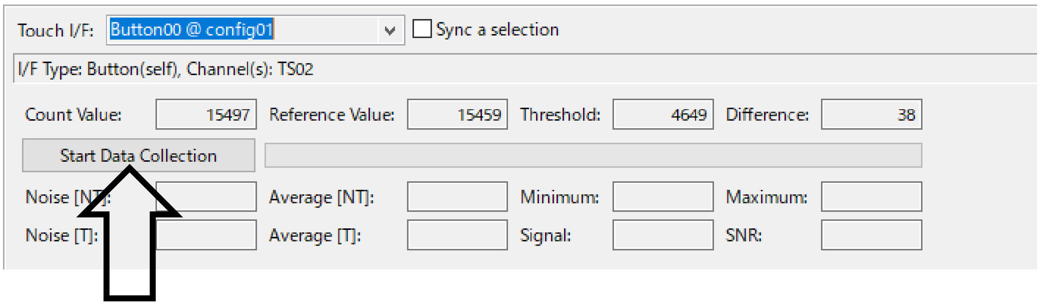


15. The graph will begin to display running values. Touch **TS2** on the board and you should see the 'touch counts' show as a step change on the running graph. The **GREEN** line is the touch '**Threshold'**, which the middleware uses to determine whether a button is actuated/touched. The **RED BELT** at the bottom of the graph is a visual indication to the user that the 'touch counts' have crossed above the threshold and a touch is detected**.**

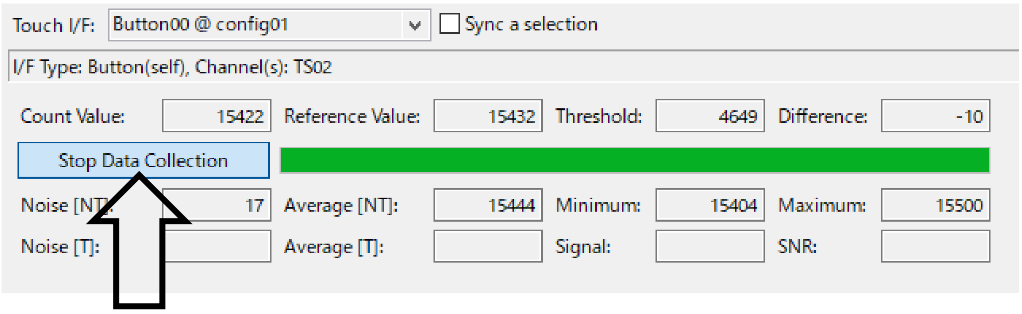


Note: Sections 16 to 19 should only be set when displaying and measuring standard deviation.

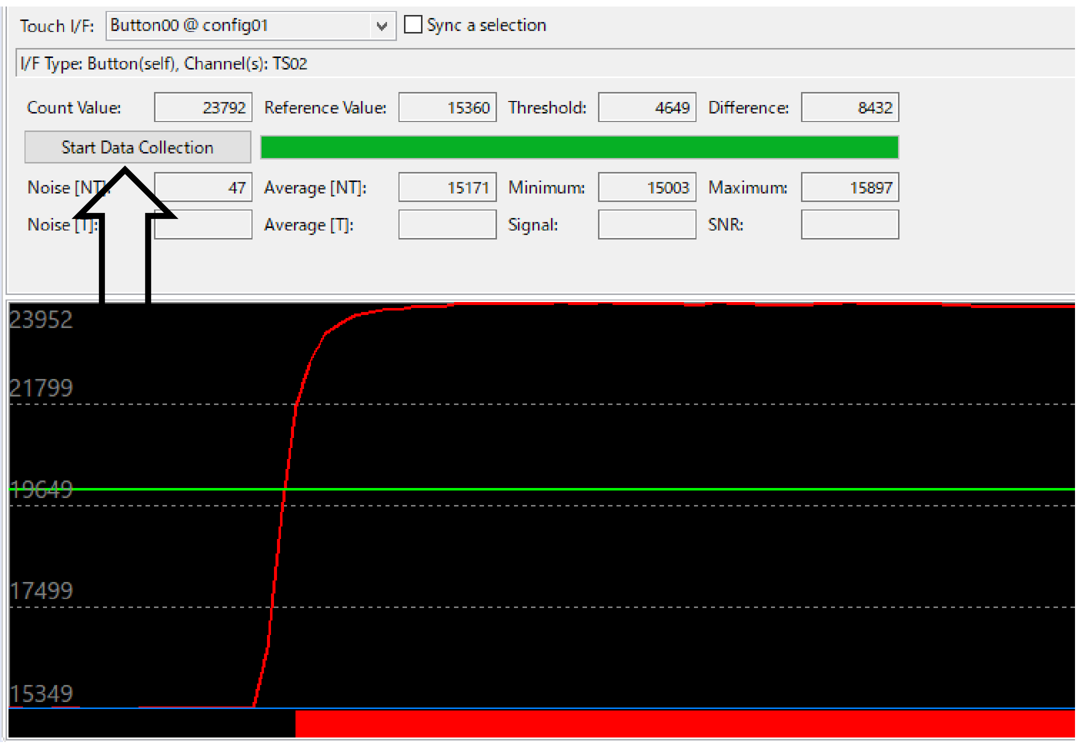
16. Next, measure standard deviation. Click the **Start Data Collection** button. Don't touch the electrode as this will collect data of **touch-off** state. The green bar is the data collection rate. When the green bar goes all the way to the right, the data collection of **touch-off** state is complete.



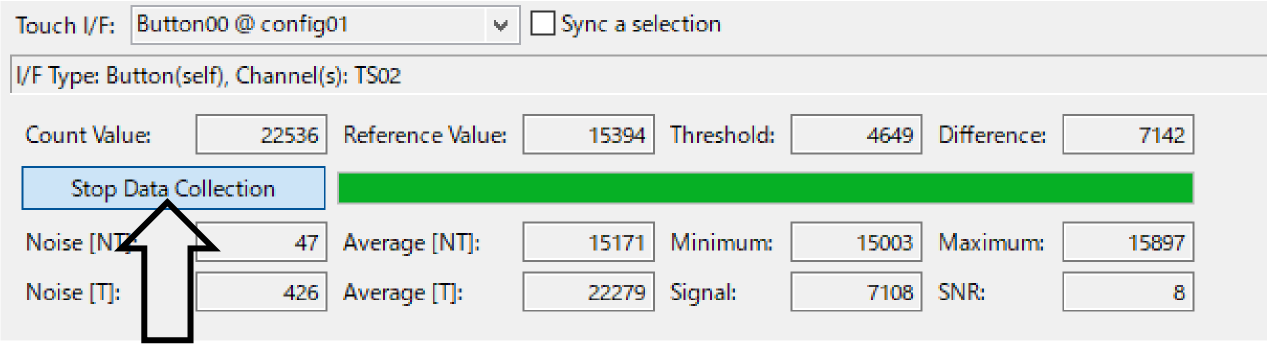
17. Click the **Stop Data Collection** button, when the green bar goes all the way to the right



18. Next, Touch the electrode as this will collect data of **touch-on** state. Click the **Start Data Collection** button while touching the electrode.



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



13. qe_touch_sample.c Listing AFTER Modifications

```
/***********************************************************************
*
* FILE : QE_Capacitive_Touch_Sample.c
* DATE : 2020-03-04
* 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;



void qe_touch_main(void)
{
    fsp_err_t err;




    /* Open Touch middleware */
    RM_TOUCH_Open(g_qe_touch_instance_config01.p_ctrl,
g_qe_touch_instance_config01.p_cfg);




    /* Main loop */
    while (true)
    {

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

        err = RM_TOUCH_DataGet(g_qe_touch_instance_config01.p_ctrl, &button, 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

Renesas Electronics Website
http://www.renesas.com/

Capacitive Touch Sensing Unit related page
https://www.renesas.com/solutions/touch-key
https://www.renesas.com/fsp
https://www.renesas.com/qe-capacitive-touch

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

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Feb/28/2020	-	Initial Revision
1.10	Apr/6/2020	2	SImplified application Example Overview.
		3	Changed the location where the file is created.
		3	Changed Setting of project configuration.
		4	Added power supply voltage setting method
		5	Changed the setting of clock frequency.
		5	Changed port number of unused touch sensor.
		8, 9	Added setting method of DTC
		13	Changed debugging session change method.
		14	Omitted the tuning process.
		18	Change the method of adding to the application example.
		19, 20	Changed how to open the monitoring view.
		23	Added standard deviation measurement method
		25, 26	Update to sample code.

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.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
5. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.
   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
6. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.4.0-1 November 2017)

Corporate Headquarters

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

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

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/.