

## RL78 Family

### Using QE and SIS 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 RL78 Microcontrollers.

This application note is a guide to the development of capacitive touch applications by using a combination of e<sup>2</sup> studio, the e<sup>2</sup> studio plug-in version of the Smart Configurator, and the plug-in version of QE for Capacitive Touch.

QE for Capacitive Touch is a development tool for supporting initial settings and sensitivity adjustment of touch interfaces that are required in the development of embedded systems that use capacitive touch sensors.

#### Operation Confirmation Device

RL78/G23

#### Target Device

RL78/G22

RL78/G23

RL78/L23

RL78 family with Capacitive Sensing Unit (CTSU)

When applying the sample program covered in this application note to another microcomputer, modify the program according to the specifications for the target microcomputer and conduct an extensive evaluation of the modified program.

**Note:** RL78/G16 does not support the following contents of this application note.

- “9. Changing the Debug Configuration for Capacitive Touch Sensor Tuning”
- “10. Capacitive Touch Sensor Tuning with QE for Capacitive Touch”
- Steps 9 to 13 of “13. Monitoring Touch Performance using e<sup>2</sup> studio Expressions Window and QE for Capacitive Touch” (Explanation of monitoring via emulator.)

RL78/G16 only supports tuning and monitoring via serial (UART).

Therefore, when using RL78/G16, also refer to the application note “RL78 Family Using the standalone version of QE to Develop Capacitive Touch Applications (R01AN6574)”.

---

**Contents**

1. Overview .....	4
2. About Development Tools.....	4
3. Operation Confirmation Environment.....	5
3.1 Hardware Settings .....	5
3.2 Software Settings .....	6
3.3 Operation Confirmation Conditions .....	7
3.3.1 Option Byte Setting .....	7
4. Capacitive Touch Application Development Procedure Overview .....	9
5. Application Example Overview .....	9
6. From Starting Capacitive Touch Application Development to Adding Modules .....	10
6.1 Creating a new Project .....	10
6.2 Adding Modules with the Smart Configurator.....	13
6.2.1 Setting the CTSU Component.....	21
6.2.2 Setting the Touch Component.....	23
6.2.3 Setting the PORT Component.....	24
6.2.4 Setting the LVD Component.....	26
6.2.5 Setting the BSP Component .....	26
6.3 Generating Code .....	27
7. [Additional function] Setting the serial communication monitor using UART (1/3) .....	28
7.1 Setting the Touch Component (for serial communication monitoring) .....	28
7.2 Setting the UART Communications Component.....	29
8. Creating the Capacitive Touch Interface .....	36
9. Changing the Debug Configuration for Capacitive Touch Sensor Tuning.....	41
10. Capacitive Touch Sensor Tuning with QE for Capacitive Touch .....	45
11. Adding API Call for rm_touch Middleware.....	51
12. [Additional function] Setting the serial communication monitor using UART (2/3) .....	54
13. Monitoring Touch Performance using e <sup>2</sup> studio Expressions Window and QE for Capacitive Touch .....	56
14. [Additional function] Setting the serial communication monitor using UART (3/3) .....	67
15. qe_touch_sample.c (Example of Using a Software Timer).....	73
16. [Practical applications] Touch Measurement by Hardware Timer.....	75
16.1 Using the Smart Configurator to Make Settings (Hardware Timer) .....	75

16.2	Flowcharts (Example of Using a Hardware Timer)	79
16.3	qe_touch_sample.c (Example of Using a Hardware Timer)	81
17.	Documents for Reference	83
	Revision History	84

## 1. Overview

This application note describes the following procedures for using a device of the RL78 family to embed a capacitive touch function in a system.

- Creating a project for using the Capacitive Touch Evaluation System for RL78/G23 by using Smart Configurator.
- Creating, tuning, and monitoring touch interfaces by using QE for Capacitive Touch.

## 2. About Development Tools

This application note introduces the brief procedure required to create a workable application. For questions regarding tools used in the application example in this document or for more detailed instructions, refer to the following documents: e<sup>2</sup> studio / Smart Configurator, Software Integration System (SIS) Driver / Middleware, help function for Renesas Code Generator and QE for Capacitive Touch (included in e<sup>2</sup> studio help).

### 3. Operation Confirmation Environment

This chapter describes the operation confirmation environment of the sample code.

#### 3.1 Hardware Settings

Table 3-1 shows the operating confirmation conditions (hardware).

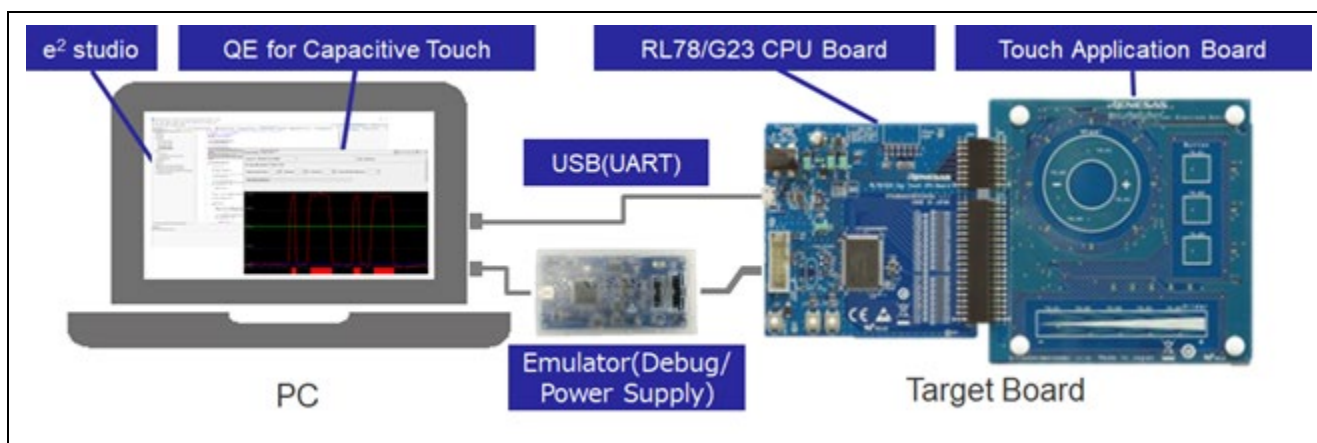
**Table 3-1 Operation Confirmation Environment (Hardware)**

Item	Description
MCU	RL78/G23 (R7F100GSN2DFB)
Target board	RL78/G23 Capacitive touch evaluation system (Product model: RTK0EG0030S01001BJ)
Emulator	E2 emulator Lite (RTE0T0002LKCE00000R)

Table 3-2 shows the jumper settings on the target board for this application. Power supply for the target board uses the emulator. Then see the circuit of the target board, and set switches or jumpers as necessary. Connect the target board to the PC through E2 emulator Lite and a USB cable as shown in Figure 3-1.

**Table 3-2 Jumper Settings on the Target Board**

Reference	Circuit Group	Jumper Setting	Description
JP1	Power supply	Pins 1-2 open (Changed from the default setting)	This jumper setting does not supply USB power to board device. (Because the power is supplied from the emulator.)
JP2		Closed (Default Setting)	Supplies board device power to MCU
JP3		Pins 1-2 closed (Default Setting)	Supplies JP1 power to board devices
JP4			



**Figure 3-1 Connection of Target Board with PC**

### 3.2 Software Settings

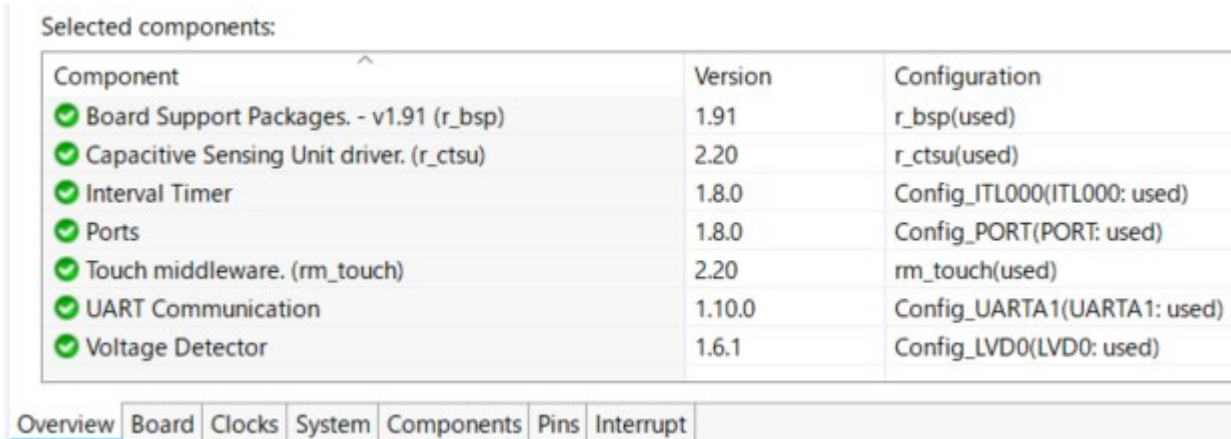
Table 3-3 shows the operation confirmation environment (software).

**Table 3-3 Operation Confirmation Environment (Software)**

Item	Description	Version
Integrated Development Environment (IDE)	e <sup>2</sup> studio	Version 2025-07 (25.7.0) (Version 2025-07 (25.7.0) or later)
Compiler	CC-RL	V1.15.00 (V1.12.00 or later)
Development assistance tool for capacitive touch sensors	e <sup>2</sup> studio plug-in version of QE for Capacitive Touch	V4.2.0 (V4.2.0 or later)
Smart Configurator	e <sup>2</sup> studio plug-in version of Smart Configurator (Included in e <sup>2</sup> studio)	V1.14.0 (V1.5.00 or later)
Software integration system (SIS) modules	<ul style="list-style-type: none"> <li>Capacitive sensing unit driver (r_ctsu)</li> <li>Touch middleware (rm_touch)</li> </ul>	V2.20 (V2.20 or later)

Note: When the free evaluation edition of CC-RL V1.12.00 or a later version is to be used for compilation during the tuning of touch sensors, set the optimization level of the compiler for building to "Debug precedence (-onothing)".

Figure 3-2 shows components and modules set by the Smart Configurator.



The screenshot shows the 'Selected components:' section of the Smart Configurator. It contains a table with three columns: Component, Version, and Configuration. The components listed are Board Support Packages, Capacitive Sensing Unit driver, Interval Timer, Ports, Touch middleware, UART Communication, and Voltage Detector, all with green checkmarks indicating they are selected. Below the table is a navigation bar with tabs for Overview, Board, Clocks, System, Components, Pins, and Interrupt.

Component	Version	Configuration
Board Support Packages. - v1.91 (r_bsp)	1.91	r_bsp(used)
Capacitive Sensing Unit driver. (r_ctsu)	2.20	r_ctsu(used)
Interval Timer	1.8.0	Config_ITL000(ITL000: used)
Ports	1.8.0	Config_PORT(PORT: used)
Touch middleware. (rm_touch)	2.20	rm_touch(used)
UART Communication	1.10.0	Config_UARTA1(UARTA1: used)
Voltage Detector	1.6.1	Config_LVD0(LVD0: used)

Overview Board Clocks System Components Pins Interrupt

**Figure 3-2 Components and Modules Set by Smart Configurator**

### 3.3 Operation Confirmation Conditions

Table 3-4 shows the operation confirmation conditions used in the project.

**Table 3-4 Operation Confirmation Conditions**

Item	Description
Operating voltage	5.0V (Operable within the range of 2.7 V to 5.5 V.) LVD0 detection voltage: Reset mode Rising edge: TYP. 2.67V (2.59V to 2.75V) Falling edge: TYP. 2.62V (2.54V to 2.70V)
Operating frequency	<ul style="list-style-type: none"> <li>Main system clock High-speed on-chip oscillator clock (<math>f_{IH}</math>): 32 MHz</li> <li>CPU/peripheral hardware clock (<math>f_{CLK}</math>): 32 MHz</li> <li>Subsystem clock<sup>Note</sup> Low-speed on-chip oscillator (<math>f_{IL}</math>): 32.768 kHz Low-speed peripheral clock frequency (<math>f_{SXP}</math>): 32.768 kHz</li> </ul>

Note: Subsystem clock is used for touch measurement using a hardware timer.  
(It is not used for touch measurement using a software timer.)

#### 3.3.1 Option Byte Setting

Table 3-5 shows the option byte settings in this sample application.

**Table 3-5 Option Byte Settings (RL78/G23)**

Address	Setting Value	Description
000C0H / 040C0H	1110 1111B (0xEF)	Disables the watchdog timer. (Counting stopped after reset)
000C1H / 040C1H	1111 1100B (0xFC)	LVD0 detection voltage: Reset mode Rising edge: 2.67V (TYP) (2.59V to 2.75V) Falling edge: 2.62V (TYP) (2.54V to 2.70V)
000C2H / 040C2H	1110 1000B (0xE8)	HS (high-speed main) mode, High-speed on-chip oscillator clock ( $f_{IH}$ ): 32MHz
000C3H / 040C3H	1000 0100B (0x84)	Enables on-chip debugging

To confirm the option byte setting, open the project property after the code is generated,

then select **[C/C++ Build] - [Settings] - [Tool Settings] - [Linker] - [Device]**. The setting can be confirmed in “User option byte value” and “On-chip debug control value”.

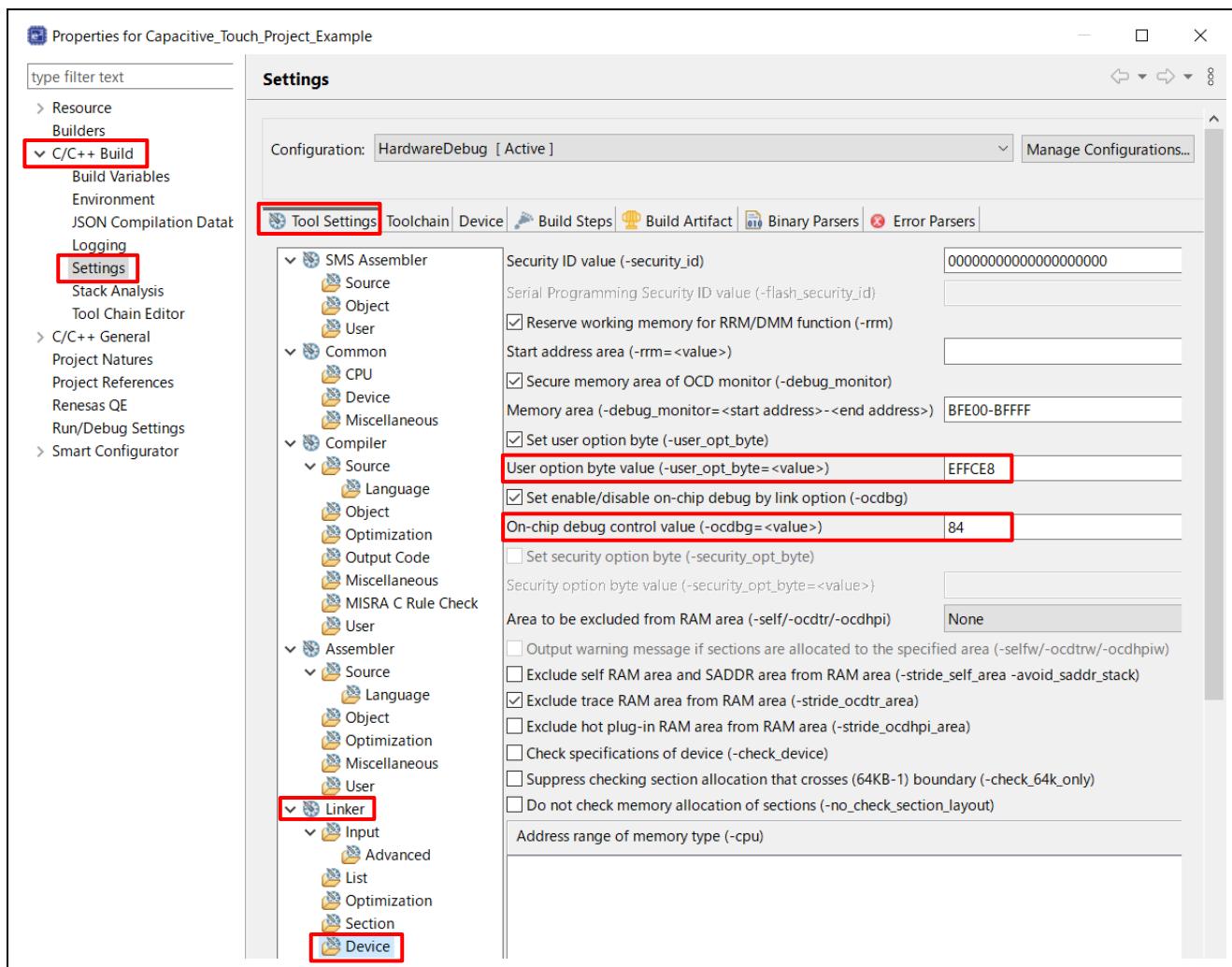


Figure 3-3 Setting of Option Bytes



## 4. Capacitive Touch Application Development Procedure Overview

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<sup>2</sup> studio project creation wizard
- Use the Smart Configurator to add the required modules to the created e<sup>2</sup> studio project
- Use the QE for Capacitive Touch e<sup>2</sup> studio plug-in to create the capacitive touch interface
- Use the QE for Capacitive Touch e<sup>2</sup> studio plug-in to tune the application project
- Add the needed SIS application code function calls to the user project to enable capacitive touch sensing operations in the application project
- Monitor the application project using QE for Capacitive Touch e<sup>2</sup> studio plug-in to demonstrate capacitive touch sensing detection

## 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 “15. qe\_touch\_sample.c (Example of Using a Software Timer)” for review.

## 6. From Starting Capacitive Touch Application Development to Adding Modules

### 6.1 Creating a new Project

This section describes how to create a new project using the e<sup>2</sup> studio.

1. On the PC start the IDE (e<sup>2</sup> studio) using the Windows -> Start menu or the e<sup>2</sup> studio icon on the desktop. When the dialog appears, create the Workspace at anywhere.
2. Start a new project by clicking File->New->**C/C++ Project**.
3. At the dialog box that opens, select "**Renesas RL78**" and highlight with a single-click "**Renesas CC-RL 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-RL
- Toolchain Version: v1.15.00
- Target Board: Custom
- Target Device: R7F100GSNxFB (Refer to Figure 6-2 for the device selection method.)
- Configurations: Check “**Create Hardware Debug Configuration**”,  
then select “E2 Lite (RL78)” from the pulldown menu.

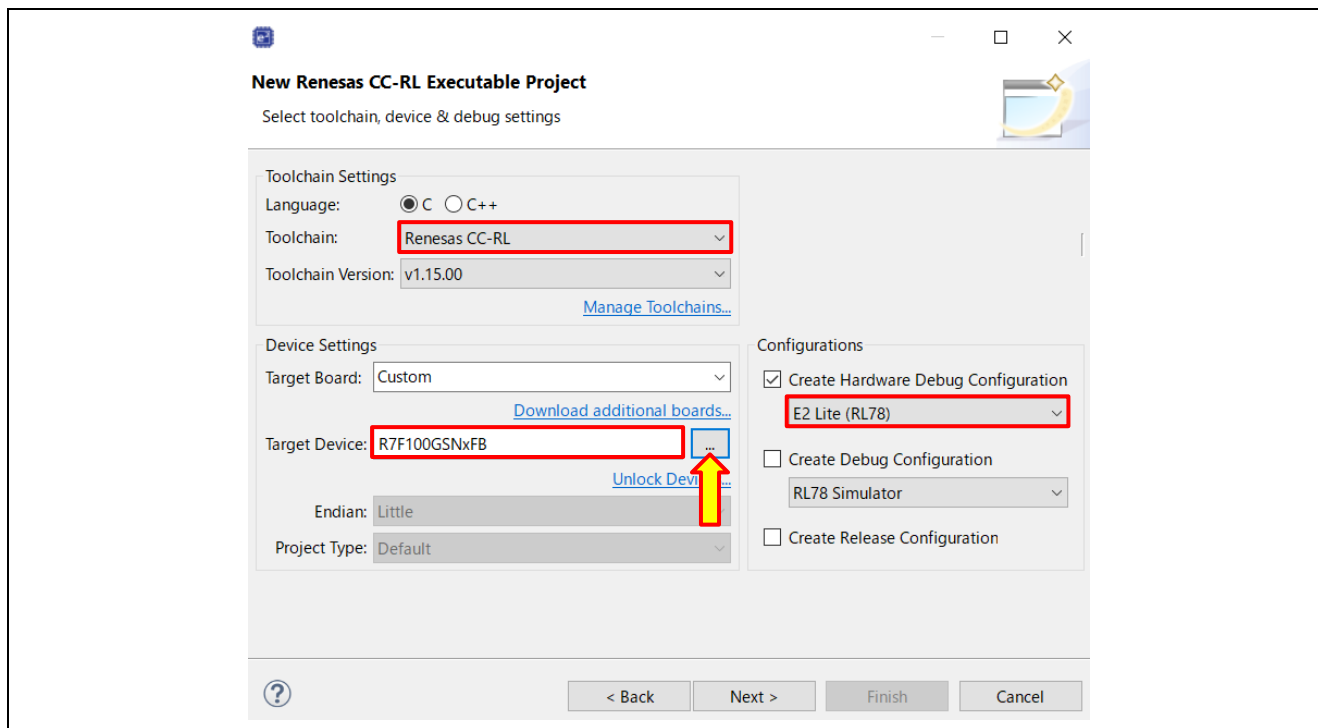
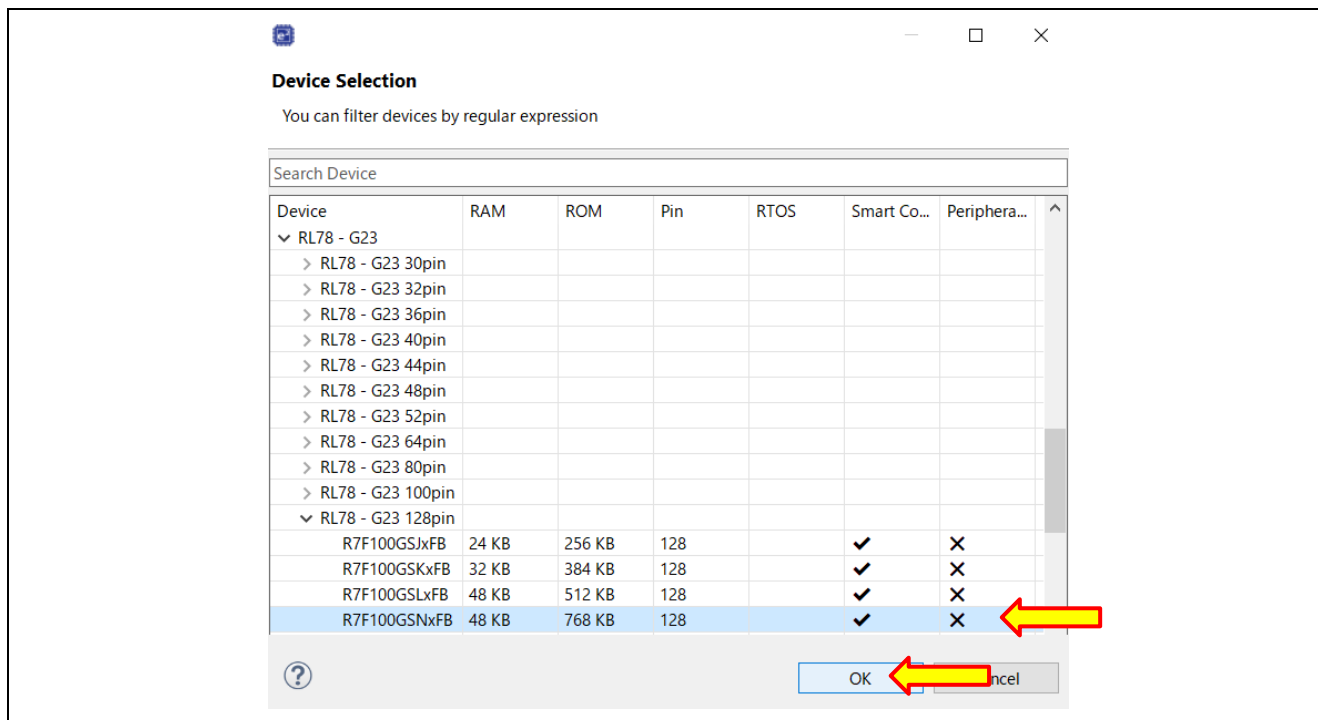


Figure 6-1 Toolchain, Device, and Debug Setting

To select the target device, click [...] next to the "Target Device" field. Select the device from the list shown in the "Device Selection" window.



**Figure 6-2 Target Device Selection**

When performing operation checks using the RSSK of other RL78 products, select the target device as follows.

- RL78/G22 (Model No. RTK0EG0042S01001BJ): R7F102GGExFB
- RL78/L23 (Model No. RTK0EG0063S01001BJ): R7F100LPLxFB
- RL78/G16 (Model No. RTK0EG0047S01001BJ): R5F121BCxFP

6. After completing your selections, click **[Next]**.
7. When the next dialog box appears, check **"Use Smart Configurator"**, then click **[Finish]**.
8. When the settings are completed, the Smart Configurator perspective will appear in the e<sup>2</sup> studio default window. The new project is now created and ready for configuration.

## 6.2 Adding Modules with the Smart Configurator

This section describes how to add source files of the necessary modules using the Smart Configurator.

1. Select the [**Clocks**] tab at the lower-middle of e<sup>2</sup> studio and display the clock tree for the RL78 MCU.

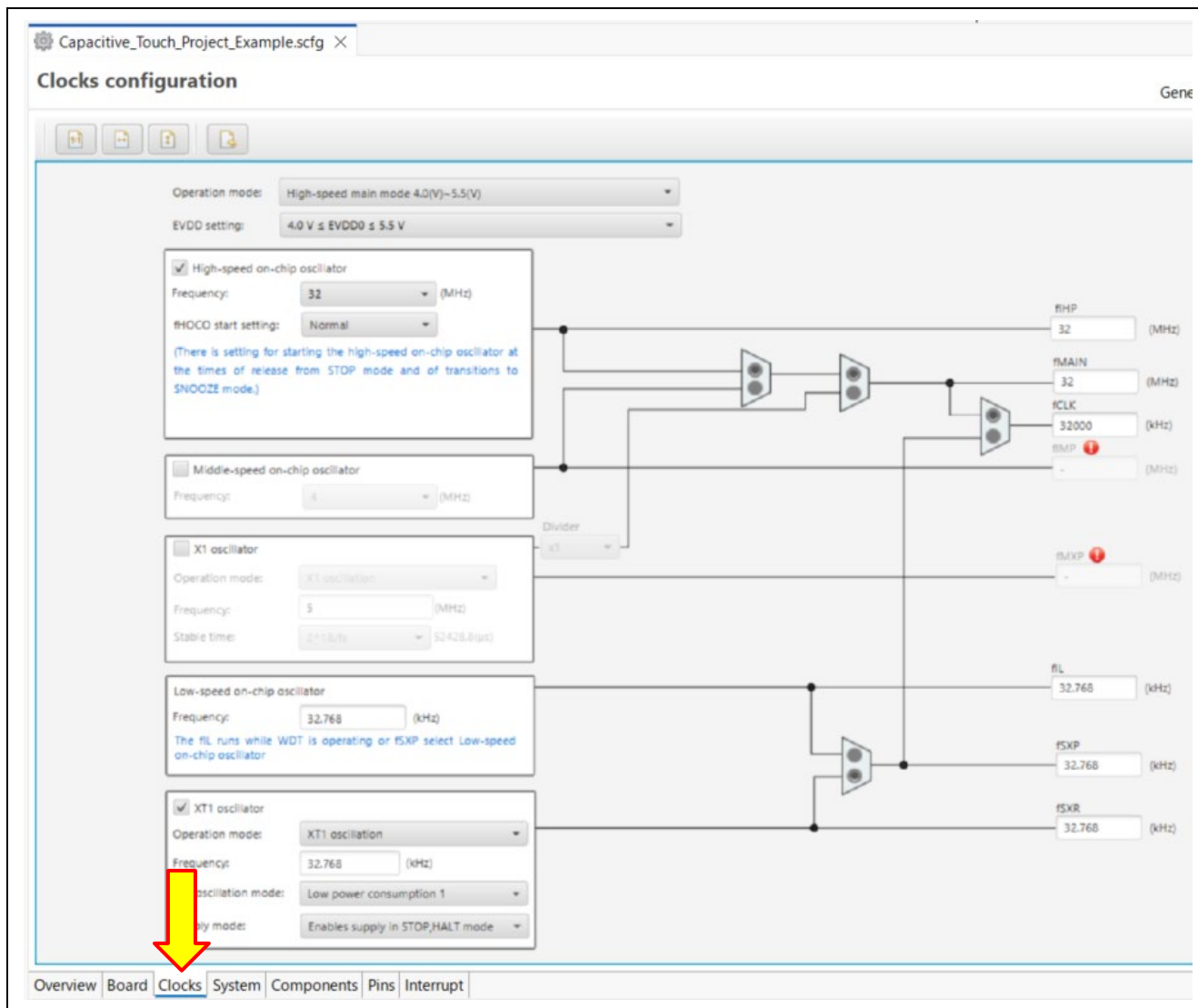


Figure 6-3 [Clocks] tab Selection

- This application example uses the MCU in the high-speed mode (HS mode) with operation voltage between 2.7V to 5.5V. Select the appropriate "Operation mode" as shown in the figure below.

Operation mode: High-speed main mode 2.7(V)~5.5(V)

EVDD setting:  $2.7\text{ V} \leq \text{EVDD0} \leq 5.5\text{ V}$

Figure 6-4 Operating Mode and EVDD Setting

## [Reference information]

When checking operation with other MCU,  
whether or not EVDD settings exist depends on the MCU used.

- The setting of the clock configuration (default setting) that is used for this application note is shown below.

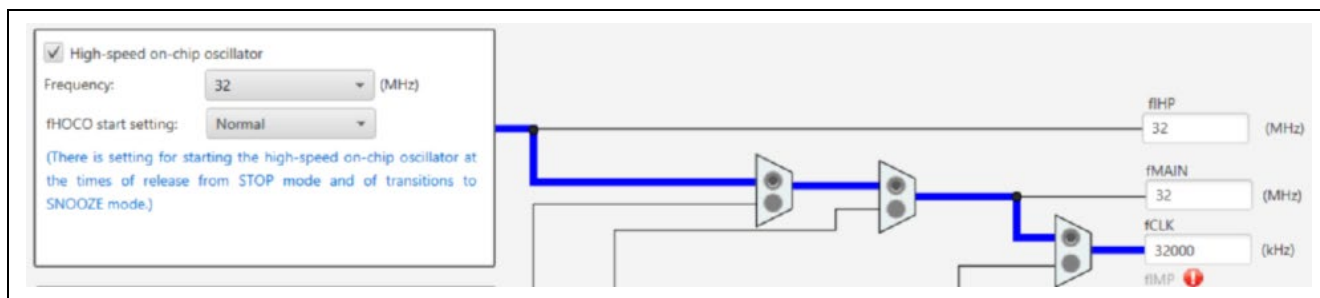


Figure 6-5 Clock Configuration (1)

- Uncheck "XT1 oscillator" as shown below.

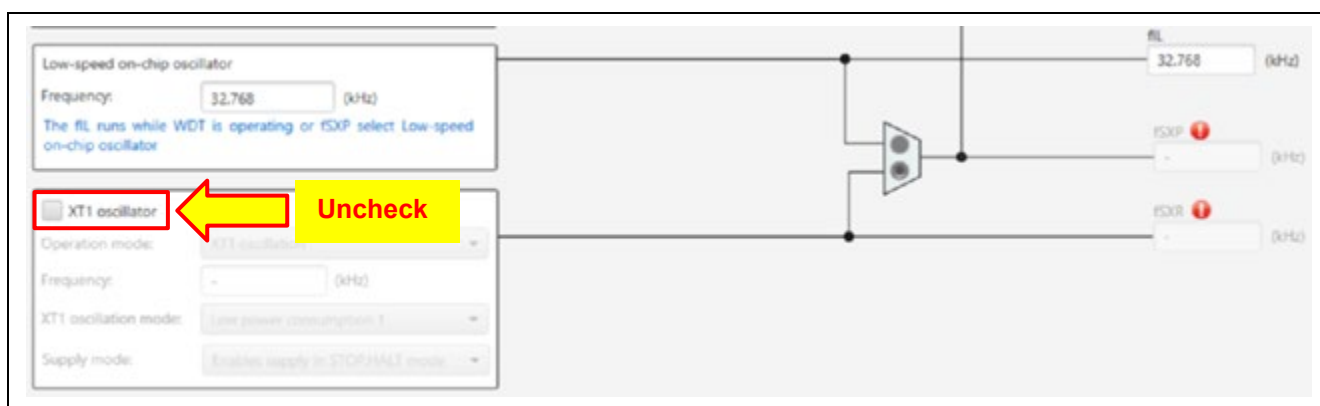


Figure 6-6 Clock Configuration (2)

5. Move to the **[System]** tab by selecting it at the bottom of the pane.

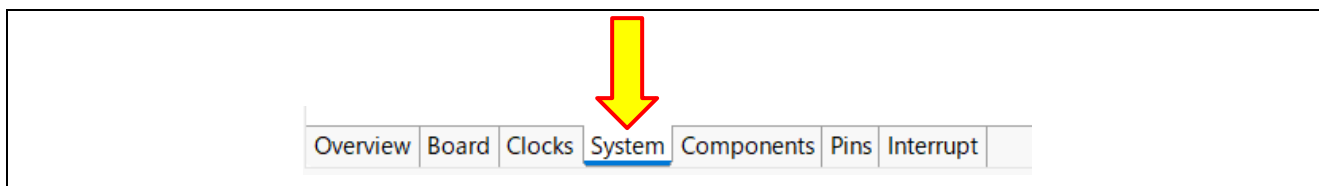


Figure 6-7 [System] tab Selection

6. Select "Use emulator" and "E2 Lite" and uncheck "Use security ID", as shown in the figure below.

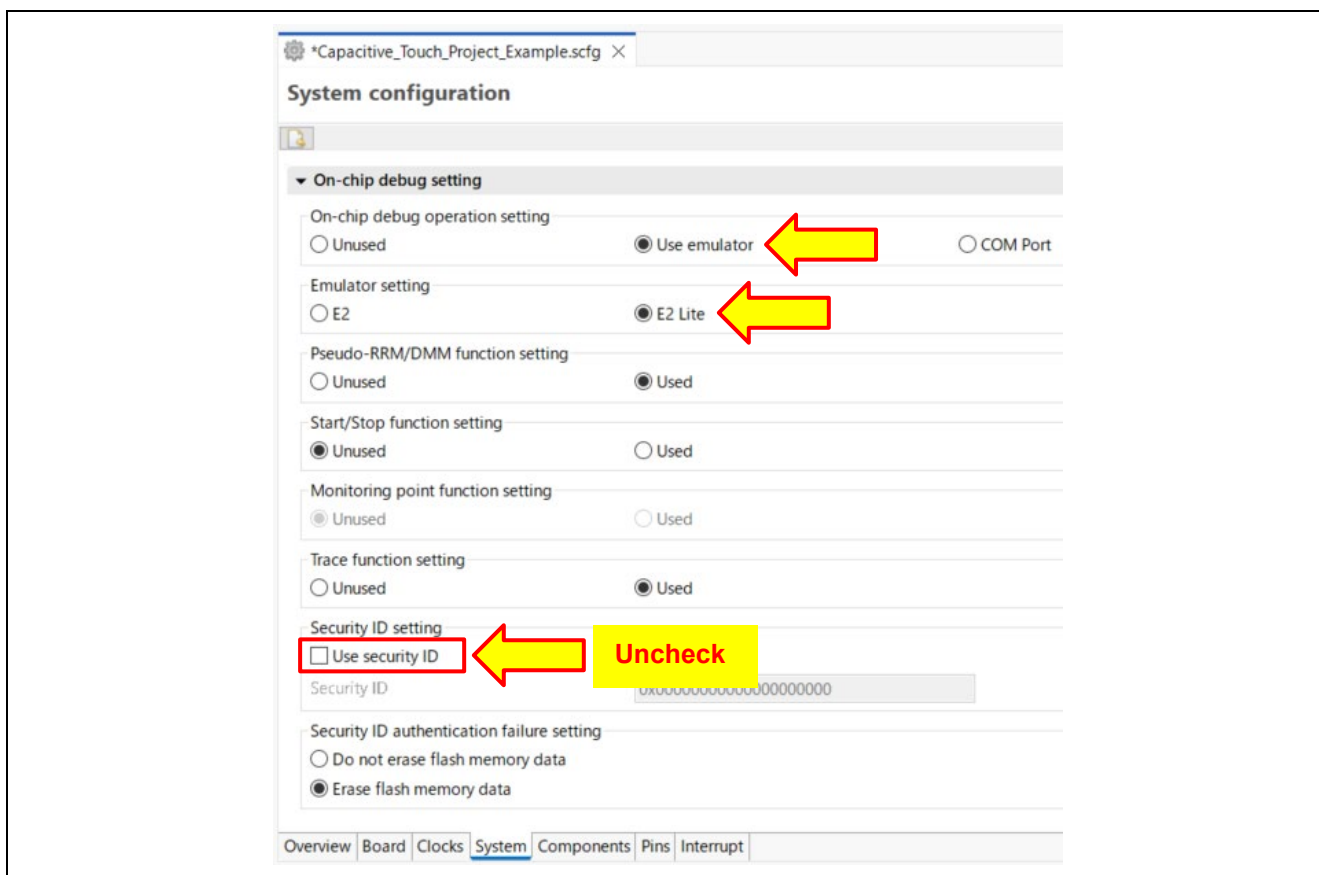


Figure 6-8 System Configuration

7. Move to the **[Components]** tab by selecting it at the bottom of the pane.

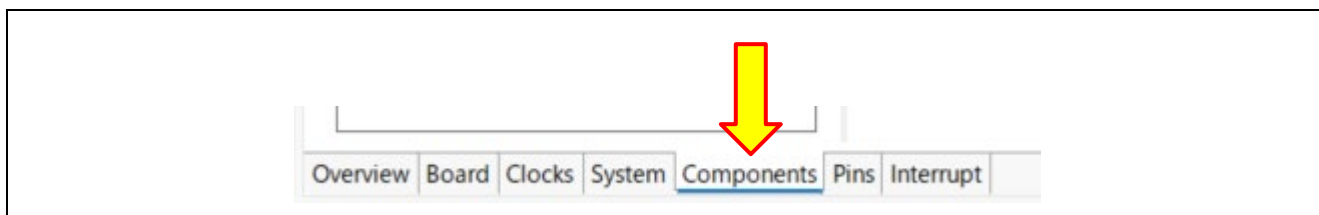


Figure 6-9 [Components] tab Selection

8. Add the SIS modules and components to the project.

Click the “Add component” button (marked with red square in Figure 6-10) and select components to be added from the list.

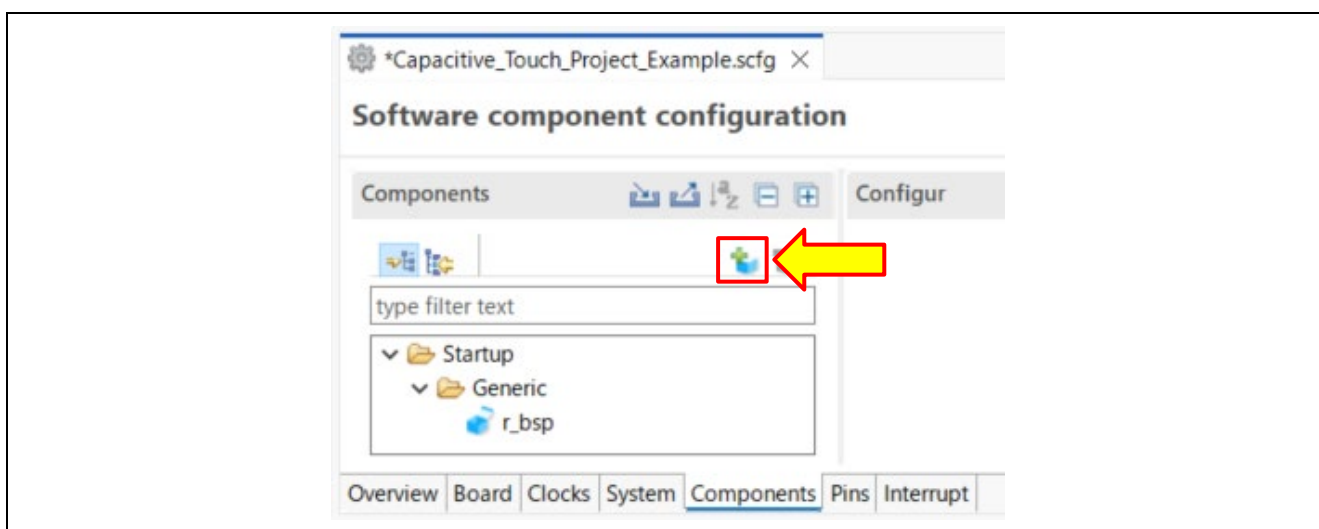


Figure 6-10 Add Components (1)



9. The “New Component” dialog box appears with the list of available modules to add to the project.

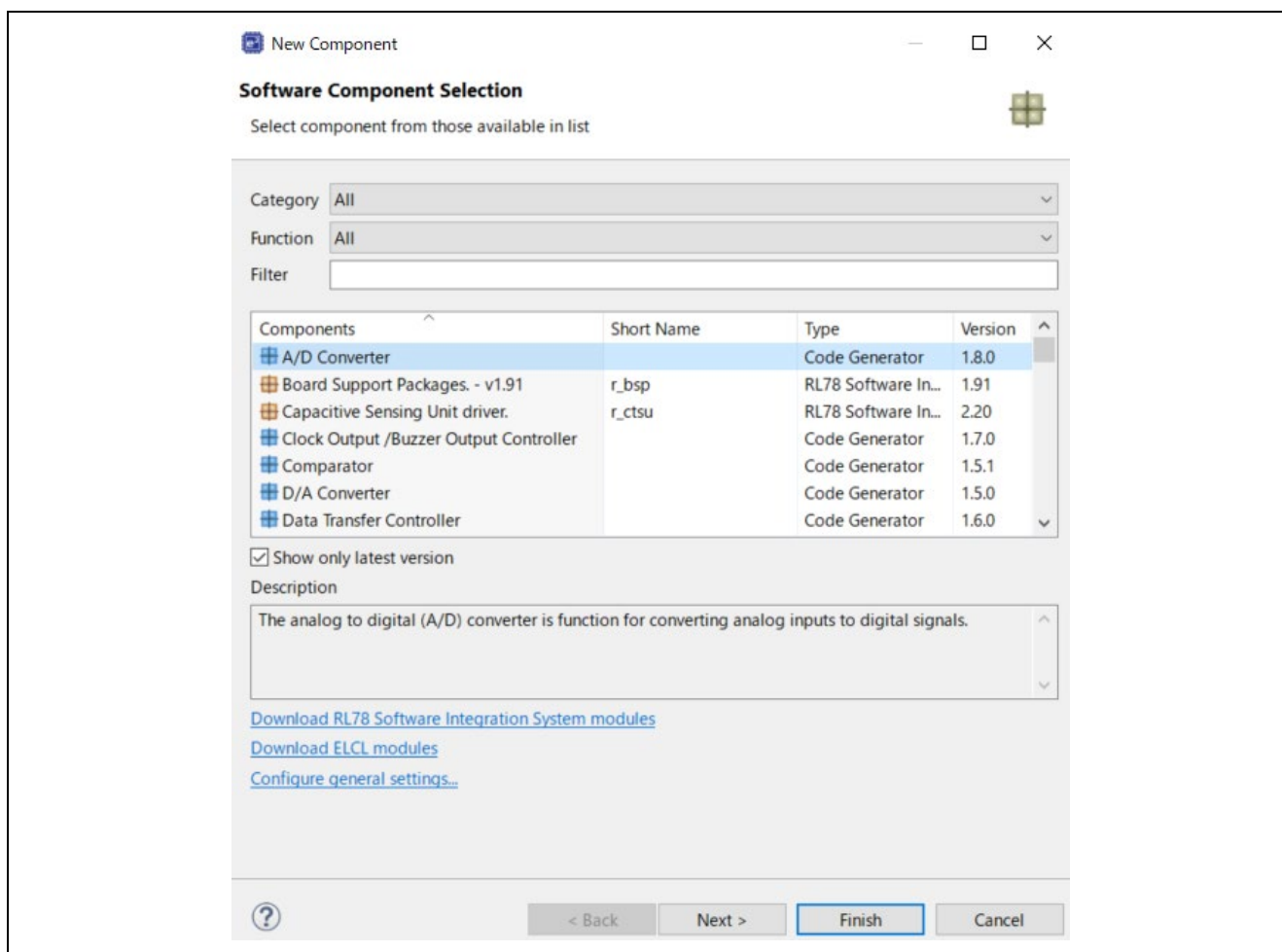


Figure 6-11 “New Component” Dialog Box

10. If you are using SIS (Software Integration System) modules for the first time, download the modules using steps (1) to (4) in Figure 6-12 to Figure 6-14 below. If they have already been installed, skip this step.

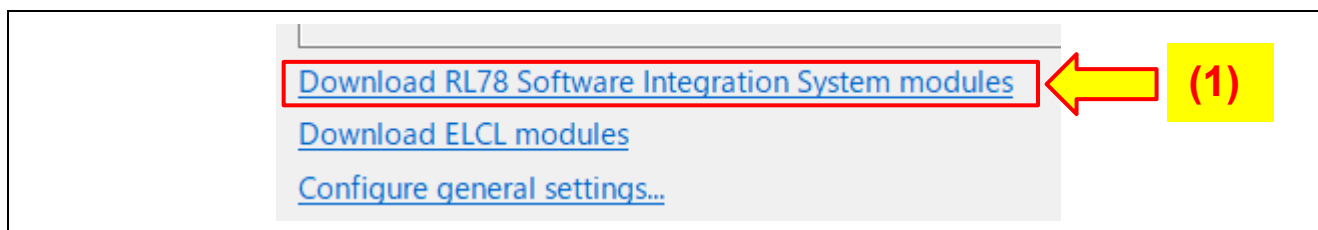


Figure 6-12 Download SIS Modules Link

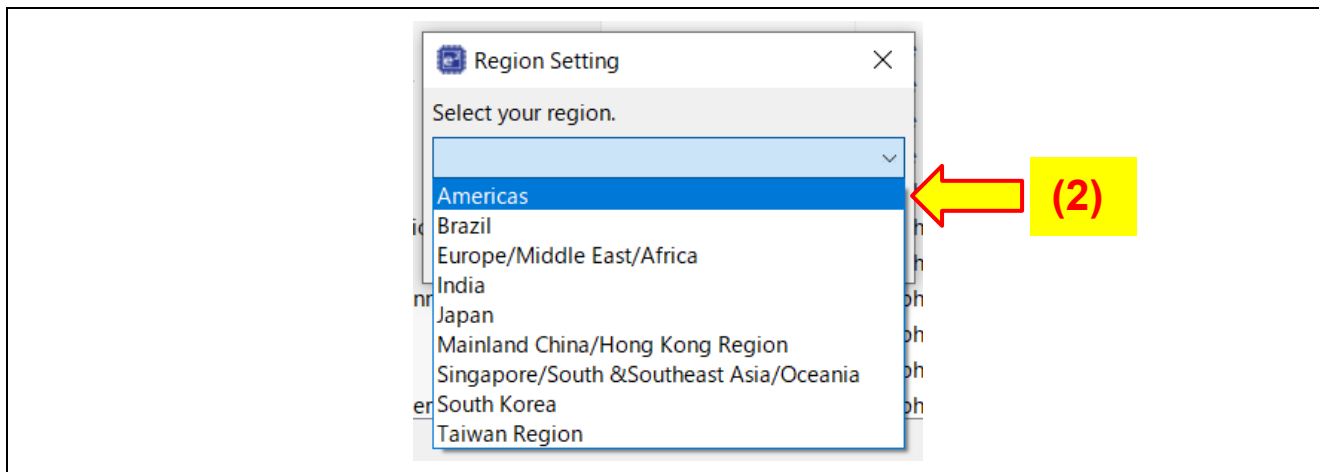


Figure 6-13 Select Region

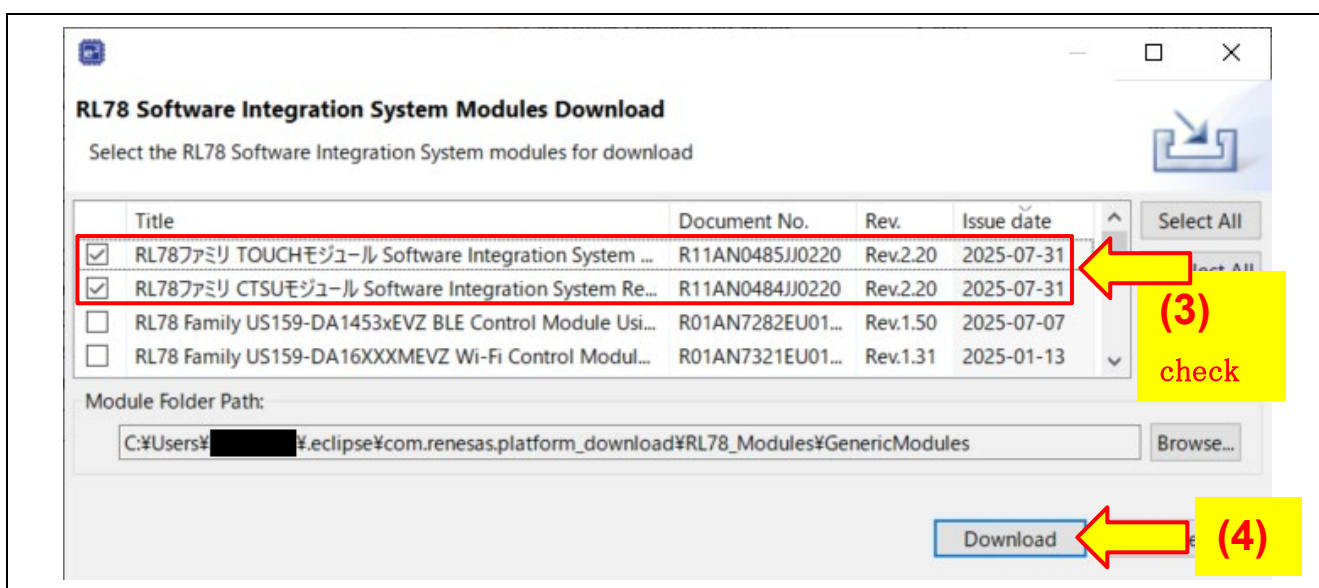


Figure 6-14 Download SIS Modules

**Note:** If the TOUCH module or CTSU module does not appear in the above dialog box for downloading, download them by using the procedure below.  
Download the modules from the Renesas Web site and use the procedures described on the site to add the files to the folder for storing the downloaded SIS modules.

For the web pages of the individual modules and how to download and display the modules in the dialog box, refer to the following.

- Web pages for downloading the CTSU module and TOUCH module  
 RL78 Family CTSU Module Software Integration System  
[RL78 Family CTSU Module Software Integration System Rev.x.xx - Sample Code](#)  
  
 RL78 Family TOUCH Module Software Integration System  
[RL78 Family TOUCH Module Software Integration System Rev.x.xx - Sample Code](#)
- How to download and use the modules  
[How to Download and Use a SIS Module](#)

11. Select the components shown in the figure below in the Smart Configurator.

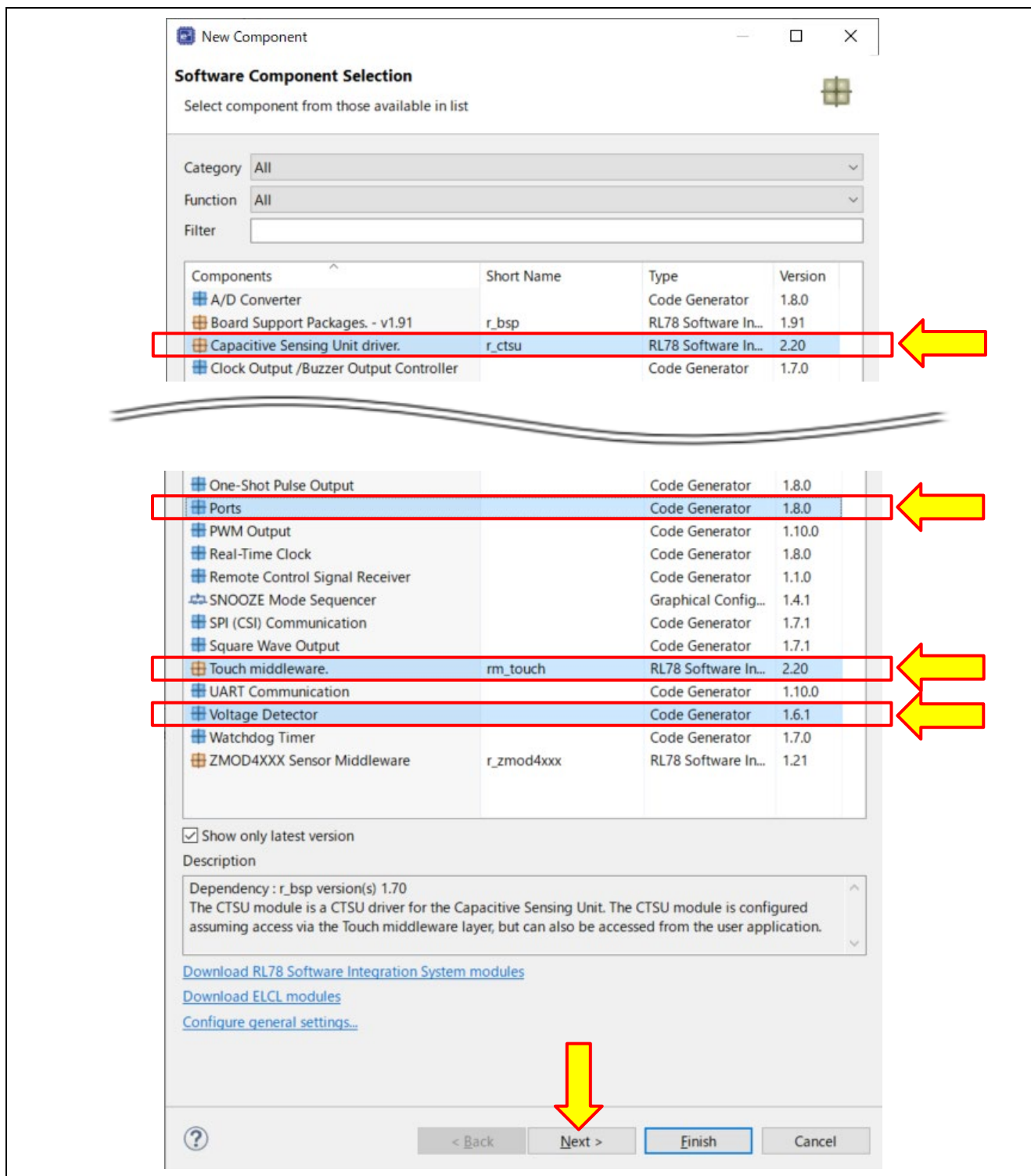


Figure 6-15 Software Component Settings (1)

12. Specify the resources for the selected components. The settings for the application example are shown below.

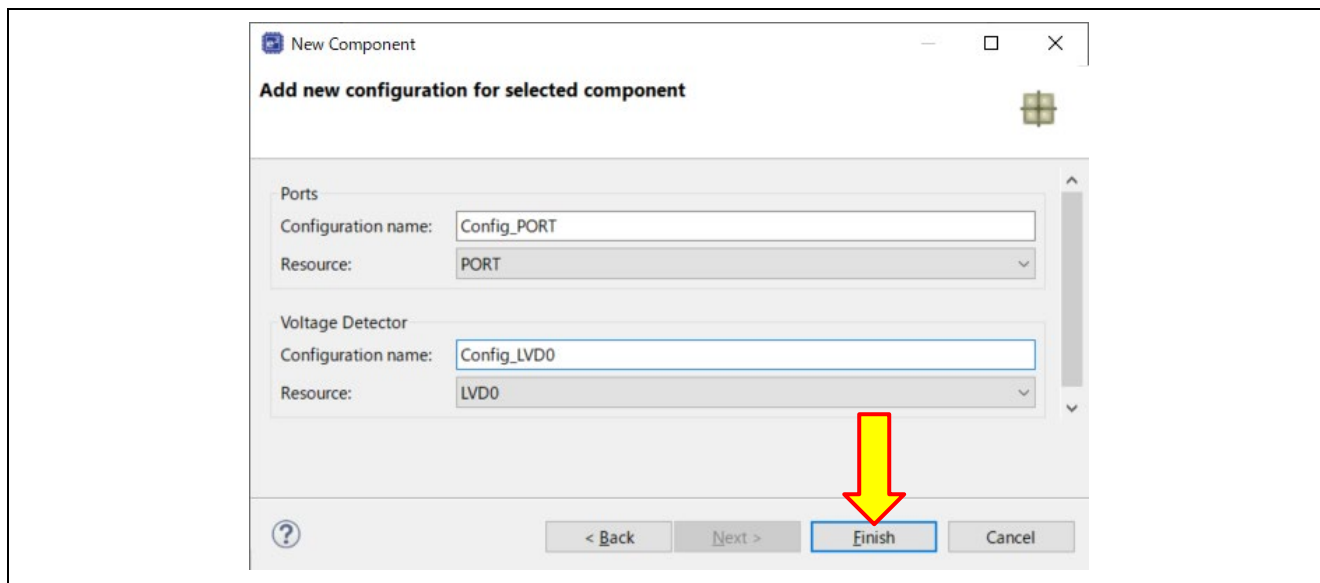


Figure 6-16 Component Resource Settings (1)

13. Components are added as shown below.

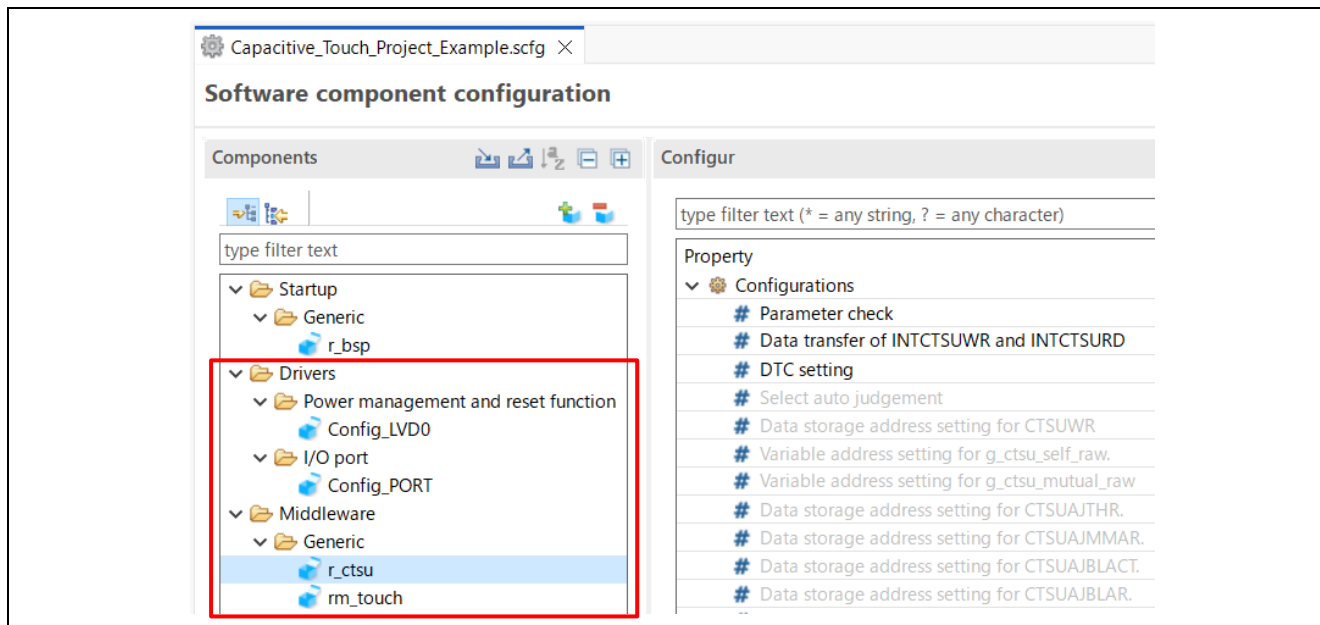


Figure 6-17 Software Component Settings (after components added)

### 6.2.1 Setting the CTSU Component

Click on the "r\_ctsu" module and enable the TSxx pins to be used by this sample application. This application example will assign the two buttons (**TS05** and **TS06**).

Click the following pins in the Configure pane so they can be used in the project.

- TSCAP Pin
- TS05 Pin
- TS06 Pin

For the correspondence between the TS pins and touch sensors, refer to the user's manual of the target board you are using.

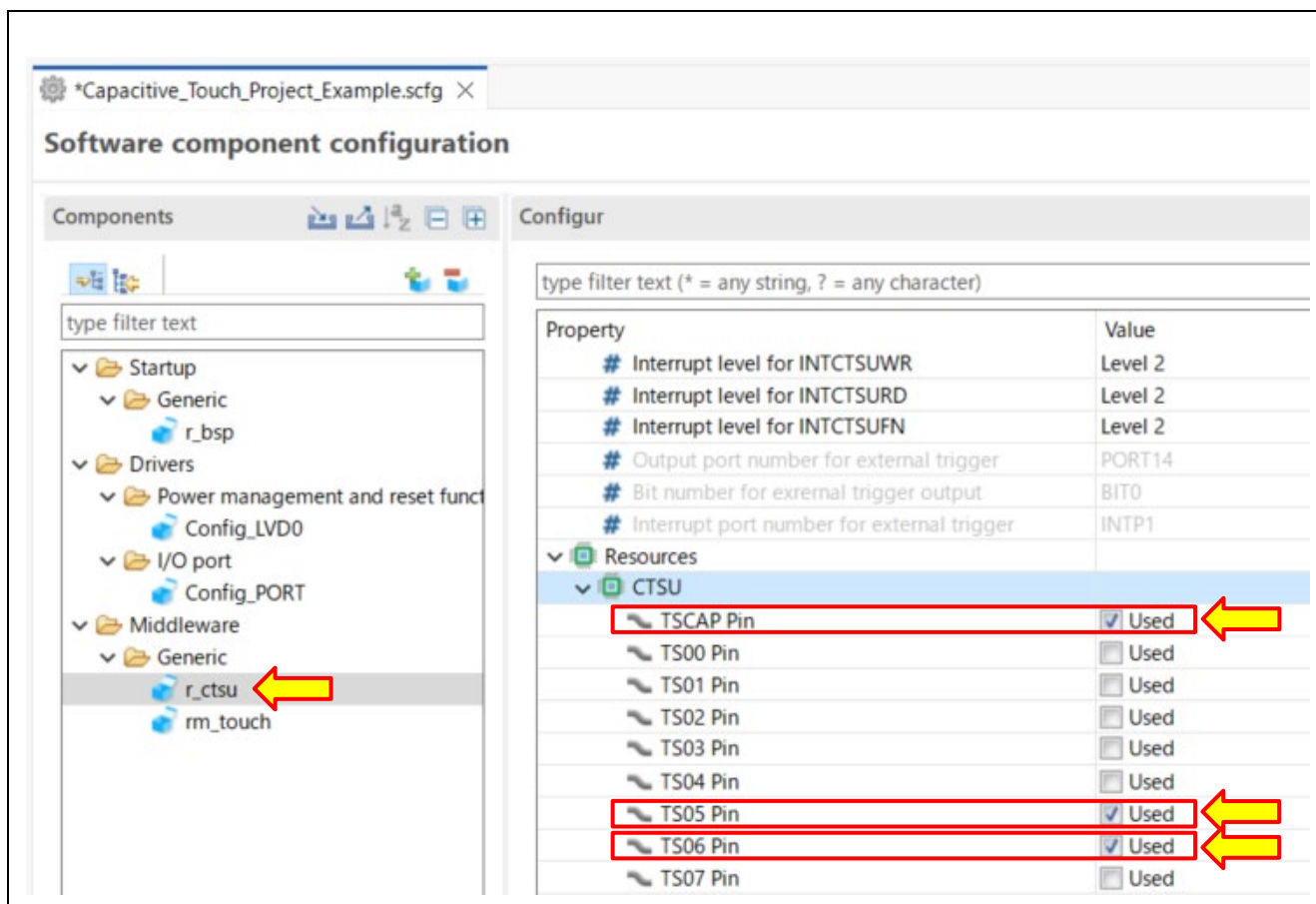


Figure 6-18 Enabling the TSCAP Pin and TS Pins to be Used by the Application

**Note:** The TSxx pins that are not being used by the touch application are recommended to be setup such that they are driven to low-level output. In the case of CTSU2, If the TSxx pin that is not being used in the touch application is enabled by checking "☐ Used", the TSxx pin is set to low-level output as non-measurement pin. When actually designing your circuit, make sure the design includes sufficient pin processing and meets electrical characteristic requirements.

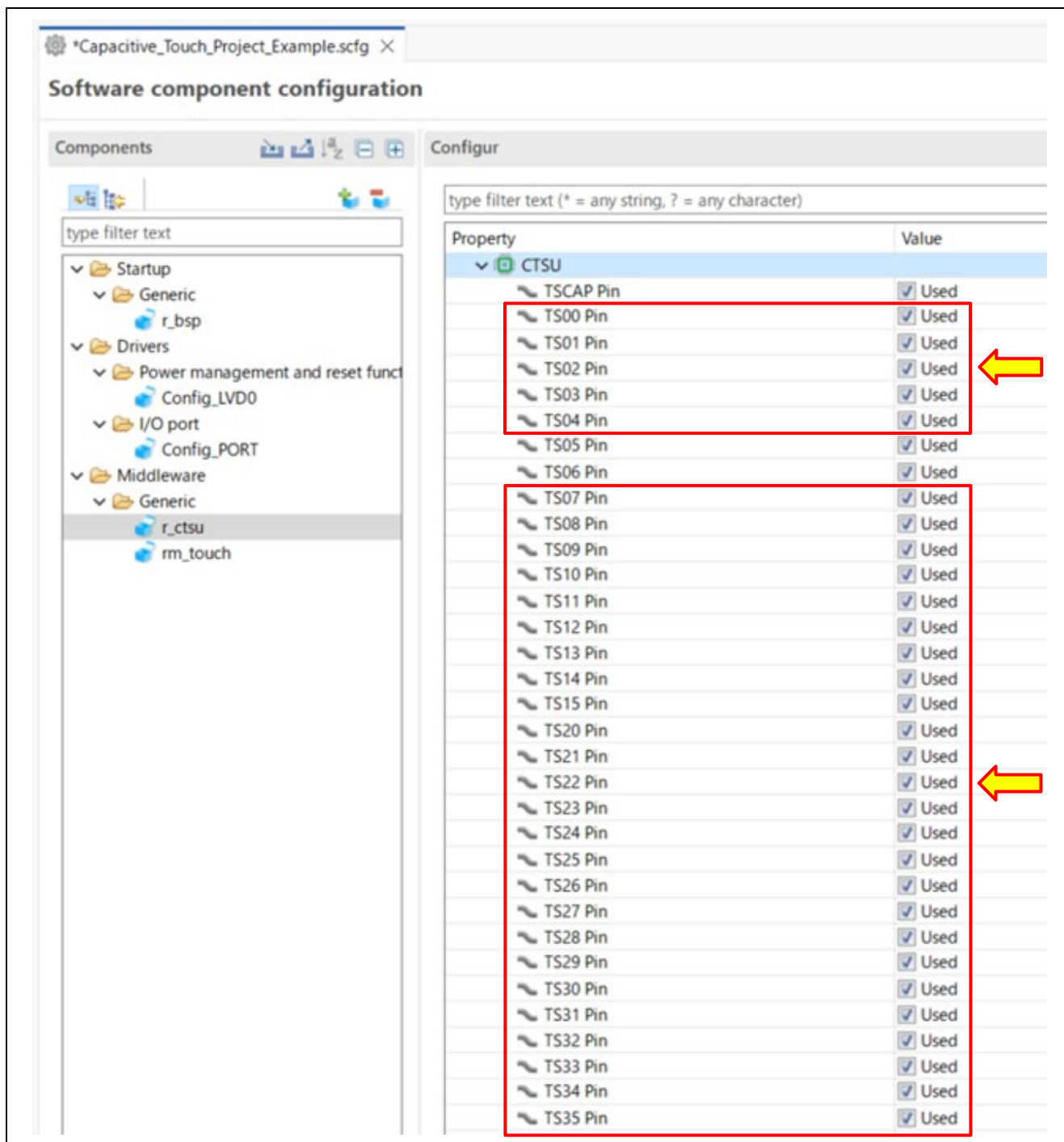


Figure 6-19 Enabling the TS Pins Not to be Used by the Application



6.2.2 Setting the Touch Component

Use the default setting as is.

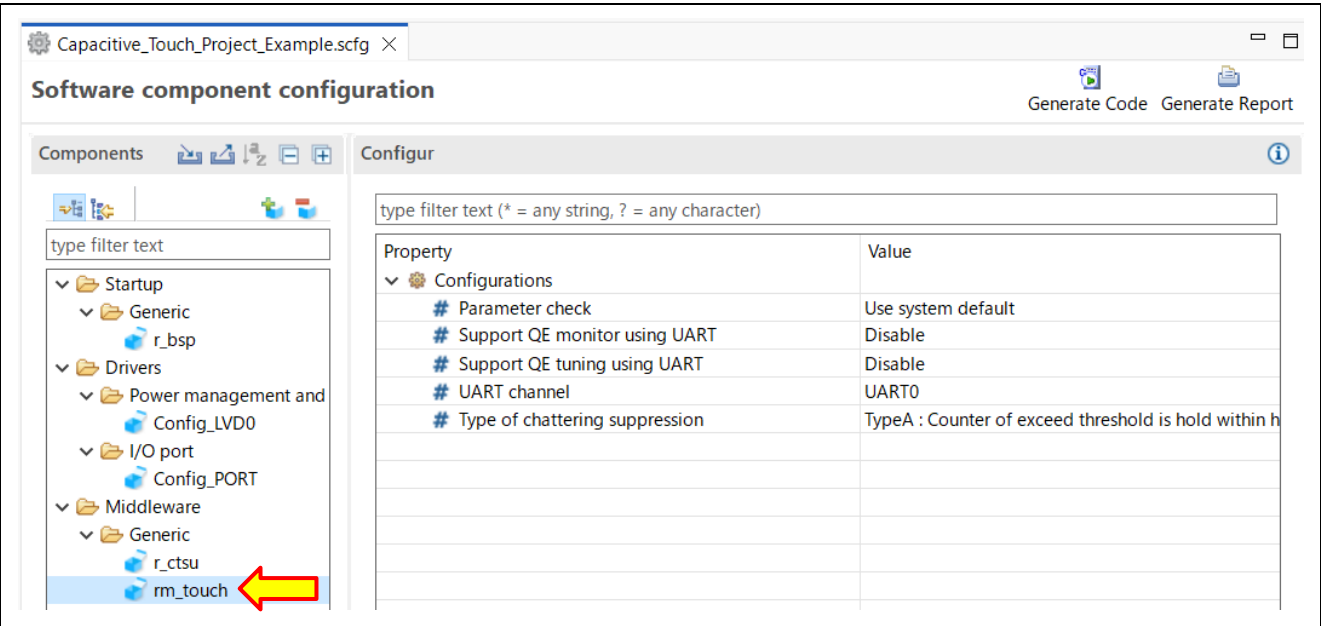


Figure 6-20 Touch Component Setting

### 6.2.3 Setting the PORT Component

Configure the port settings. In this section, configure the settings for unused pins.

1. The ports that are not being used by the touch application are recommended to be setup such that they are driven to low-level output at startup. Click the “**Config\_PORT**” module, then click the **PORT0** checkbox.

**Note:** Only one port is setup here as an example of the usage. When actually designing your circuit, make sure the design includes sufficient pin processing and meets electrical characteristic requirements.

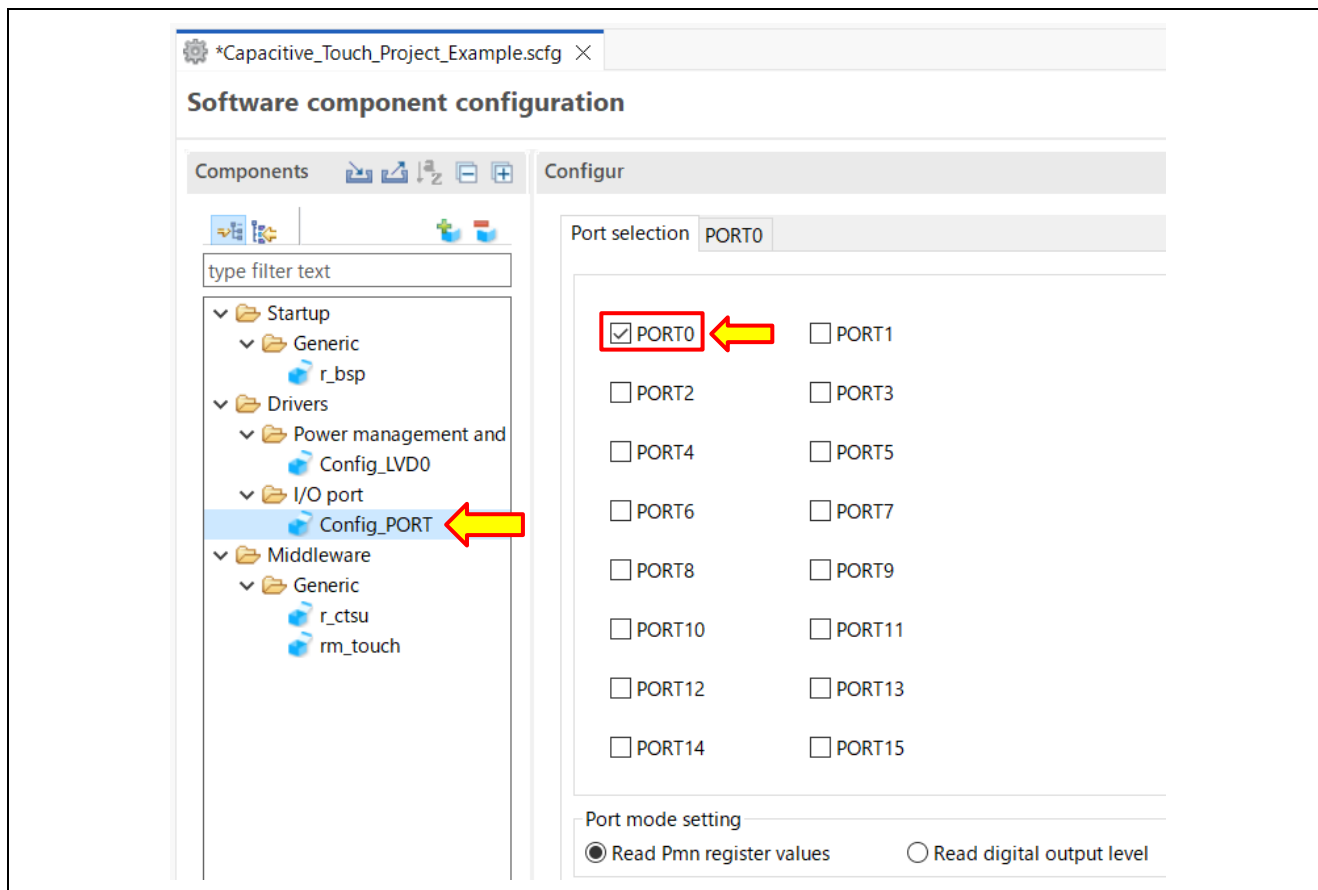


Figure 6-21 Component Setting (Port selection tab)



2. Select the [PORT0] tab and select “Out” for P07. The tab should look like the picture below.

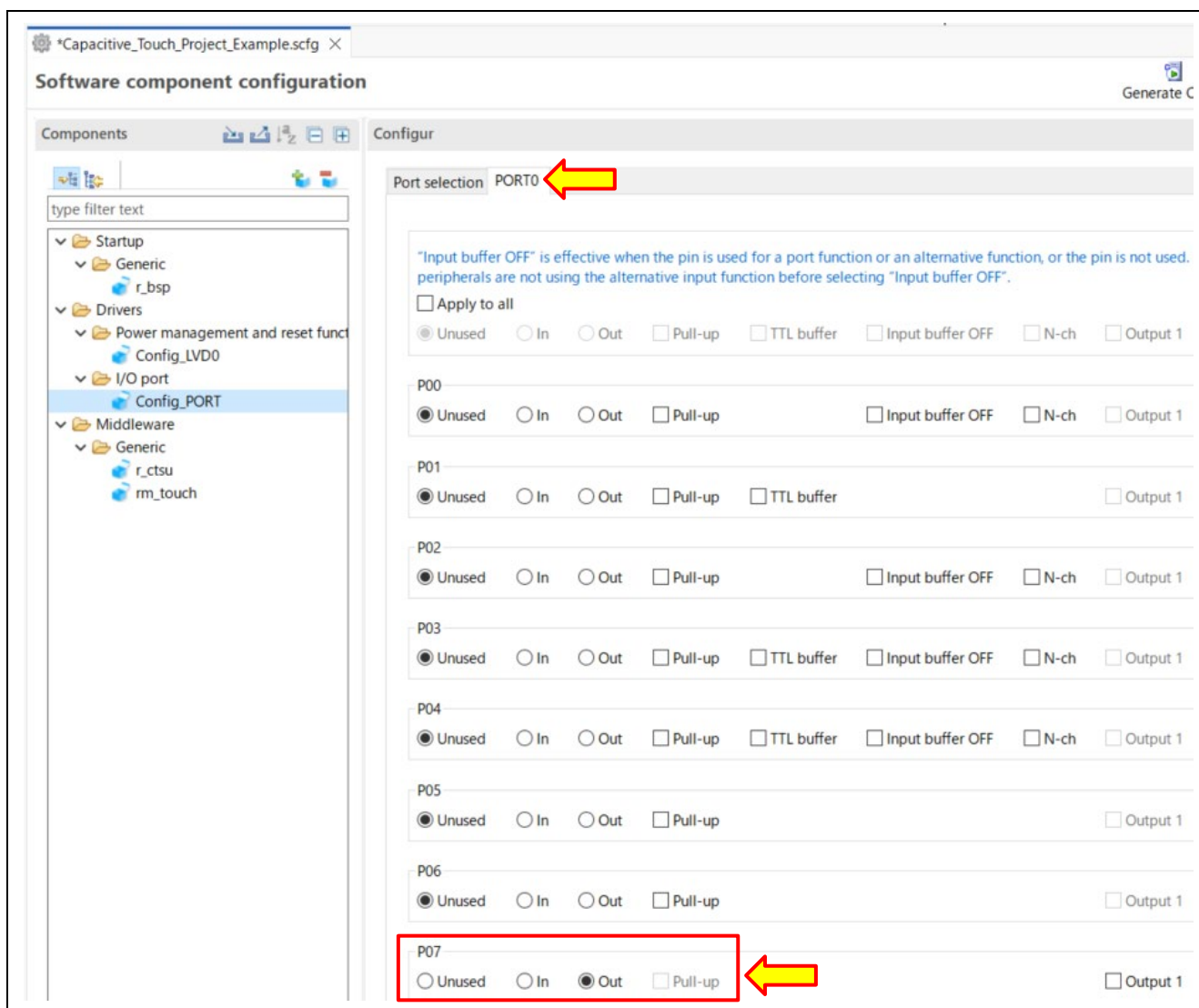


Figure 6-22 PORT Component Setting (PORT0 tab)

### 6.2.4 Setting the LVD Component

Set up the user option byte for voltage detector 0 (LVD0).

Click on the "**Config\_LVD0**" module and specify the operating mode and voltage to be detected.

Set the reset generation level (VLVD0) to 2.62 V.

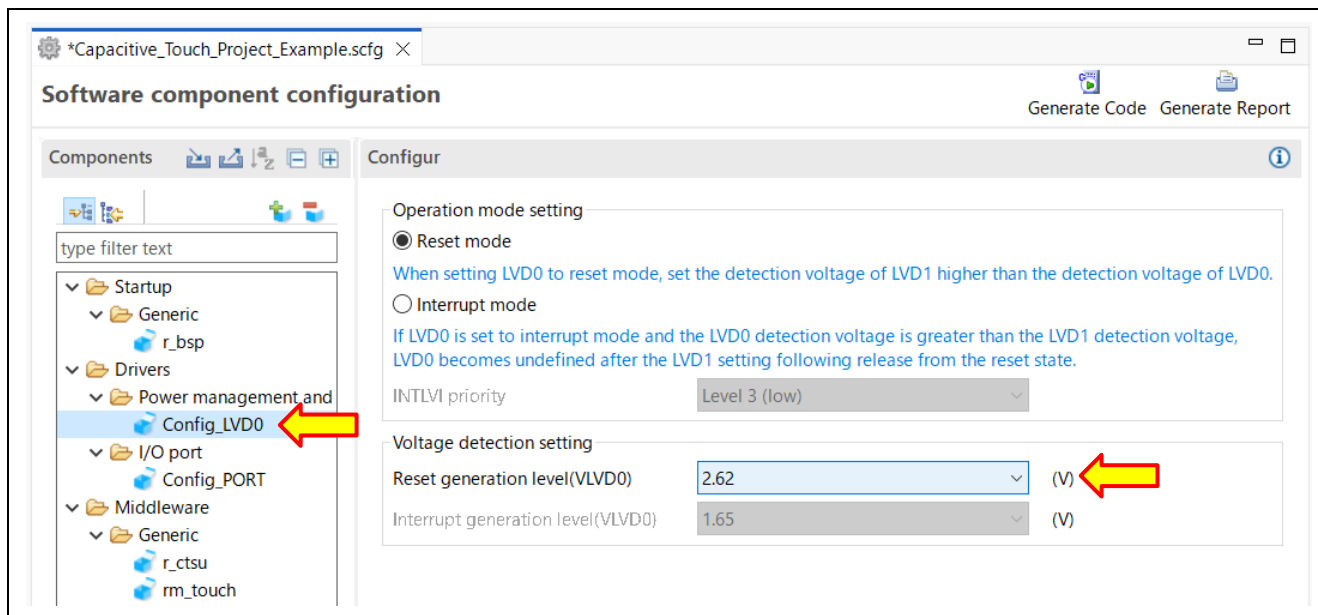


Figure 6-23 Setting the LVD Component (LVD0)

### 6.2.5 Setting the BSP Component

Click on the "**r\_bsp**" module and check that "Initialization of peripheral functions by Code Generator/Smart Configurator" is set to "Enable".

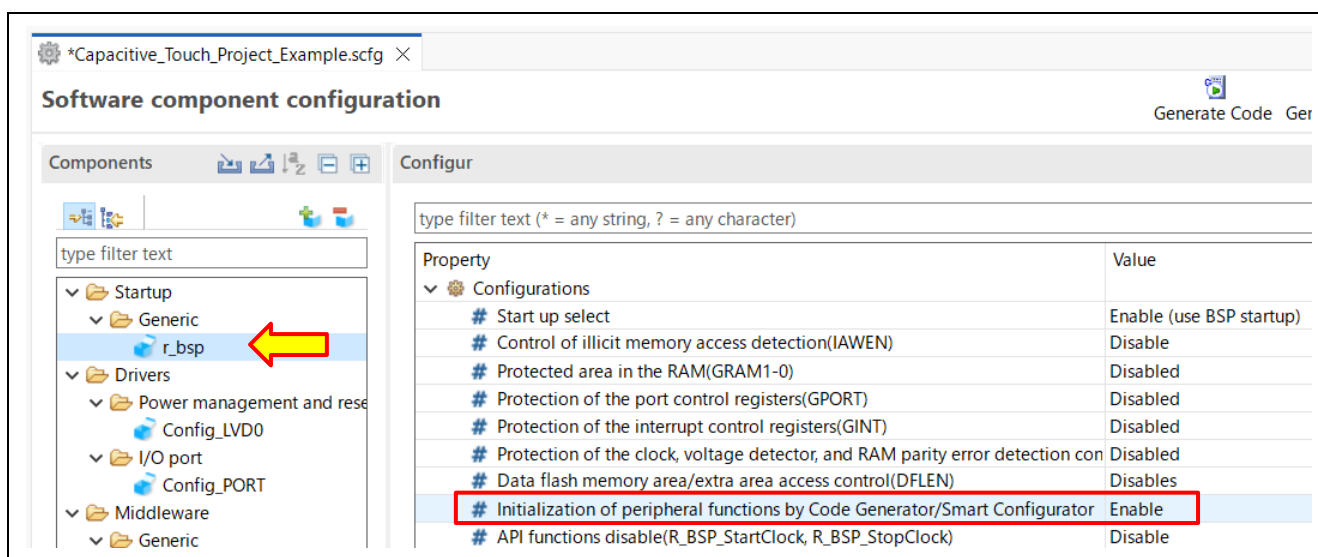



Figure 6-24 Setting r\_bsp

### 6.3 Generating Code

Click the **"Generate Code"** icon  located on the top right of the Smart Configurator to generate the code for components necessary for the project. This completes the process for adding components to the project.

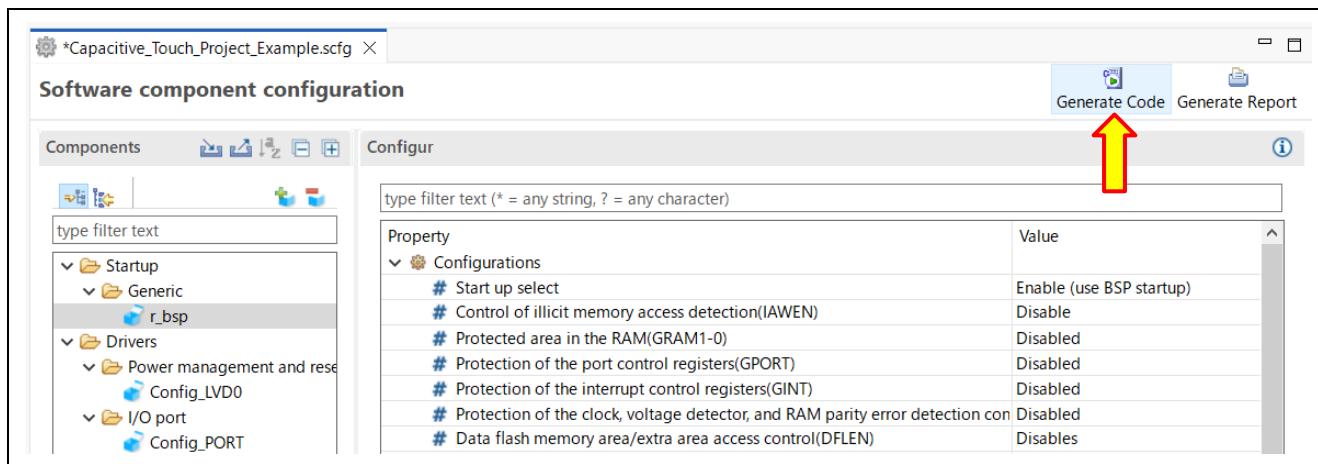


Figure 6-25 "Generating Code" icon Selection

**Note:** If the settings for on-chip debugging or option bytes have been changed, the following message may appear. Confirm the changes and click on the [OK] button.

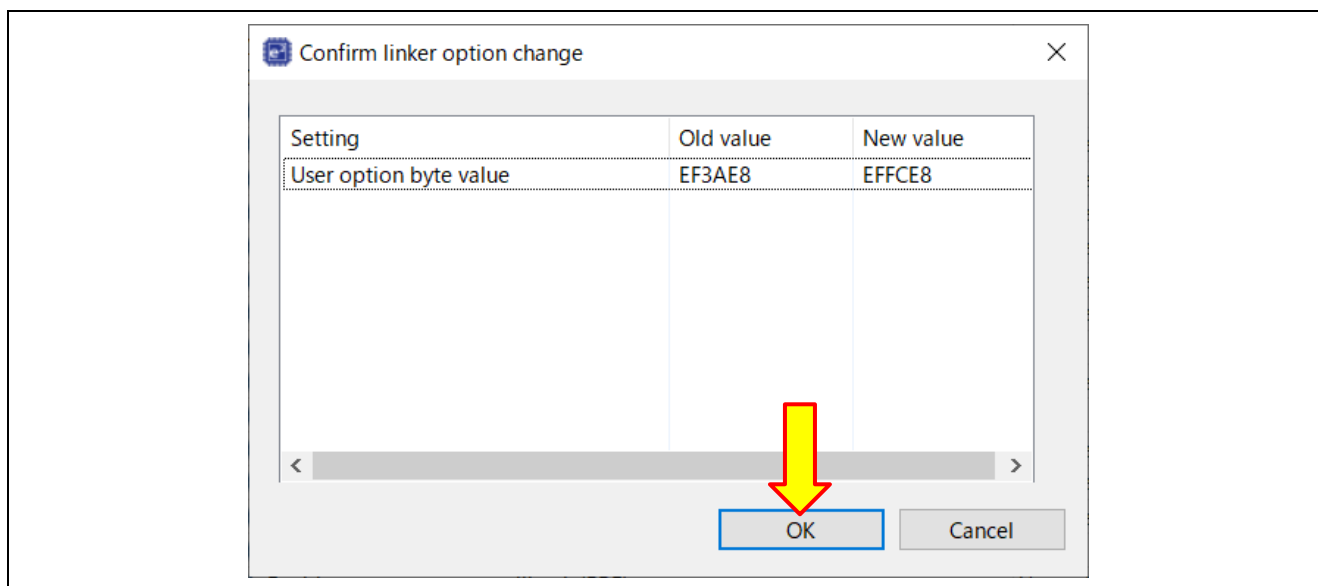


Figure 6-26 Confirm Linker Option Change

## 7. [Additional function] Setting the serial communication monitor using UART (1/3)

**Note:** Monitoring touch performance for touch applications can be confirmed by communication via the OCD (On-Chip Debugging) emulator. However, RL78 family case, monitoring performance is limited by the OCD function of the RL78 family.

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 (This is the recommended setting.).

Chapters 7, 12 and 14 (including this chapter) shown below describe setting the serial communication monitor using UART.

- “7. [Additional function] Setting the serial communication monitor using UART (1/3)”
- “12. [Additional function] Setting the serial communication monitor using UART (2/3)”
- “14. [Additional function] Setting the serial communication monitor using UART (3/3)”

### 7.1 Setting the Touch Component (for serial communication monitoring)

In the [Components] tab, select the “**rm\_touch**” module, set “Support QE monitor using UART” to “**Enable**” and “UART channel” to “**UARTA1**” as shown below.

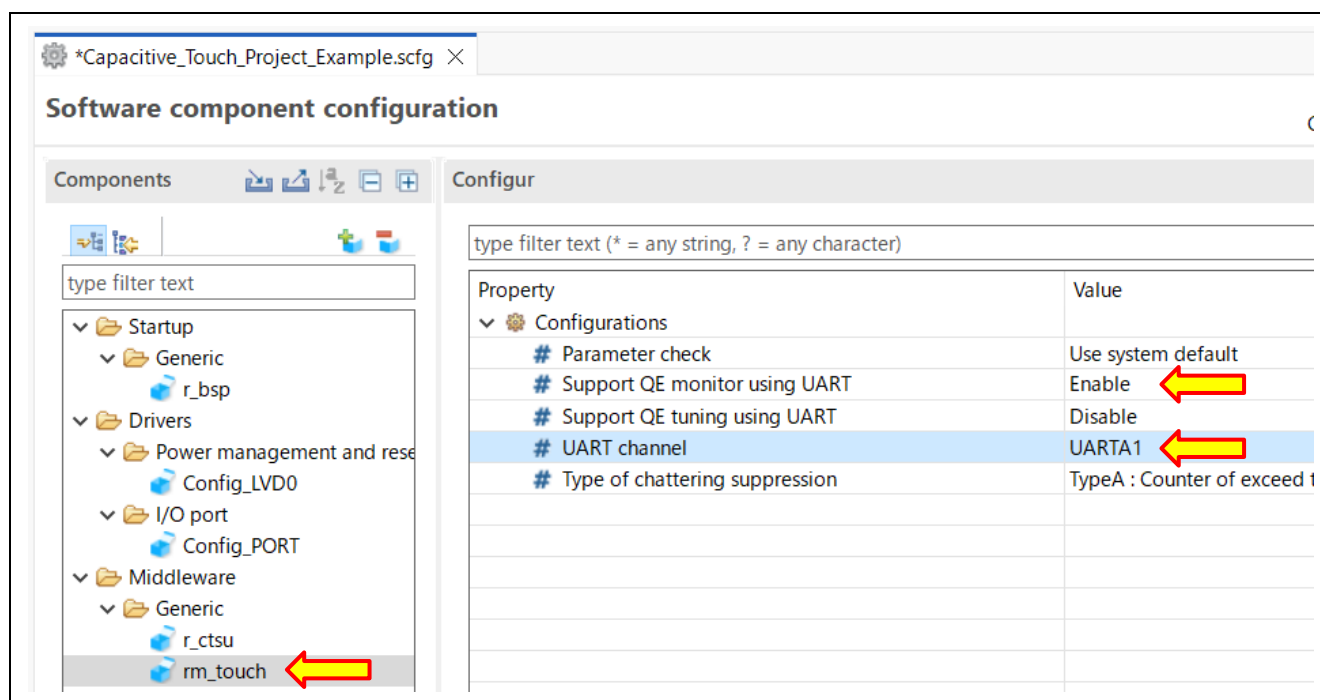


Figure 7-1 Touch Component Setting (for serial communication monitoring)

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

## 7.2 Setting the UART Communications Component

1. Click the “Add component” button (marked with red square in Figure 7-2).

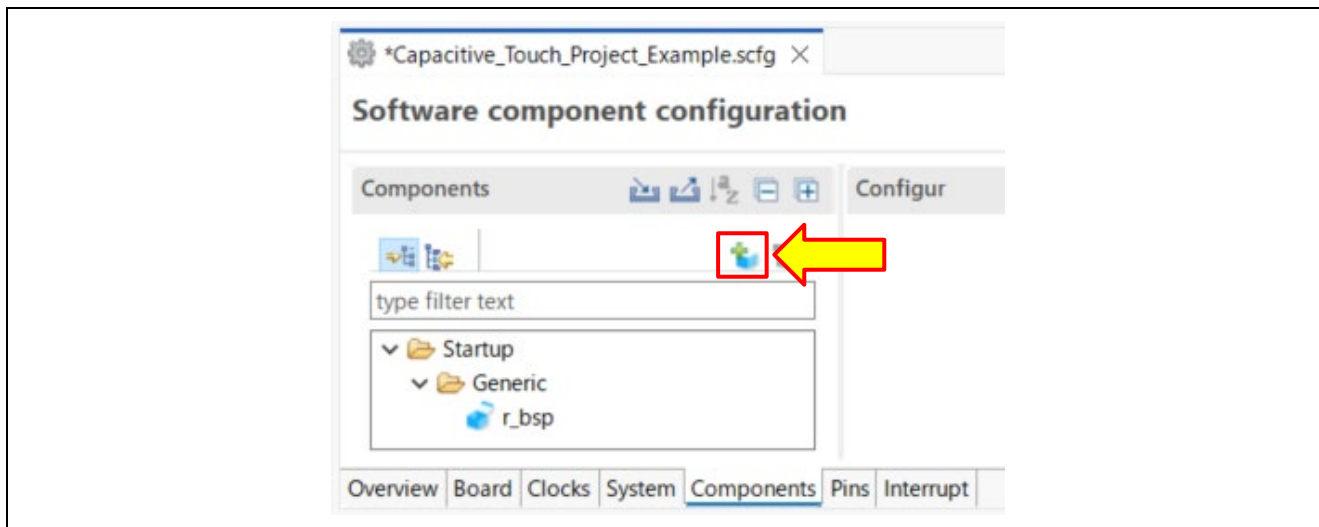


Figure 7-2 Add Components (2)

2. Click the “UART Communication” module and click [Next] in the lower part of that open dialog box.

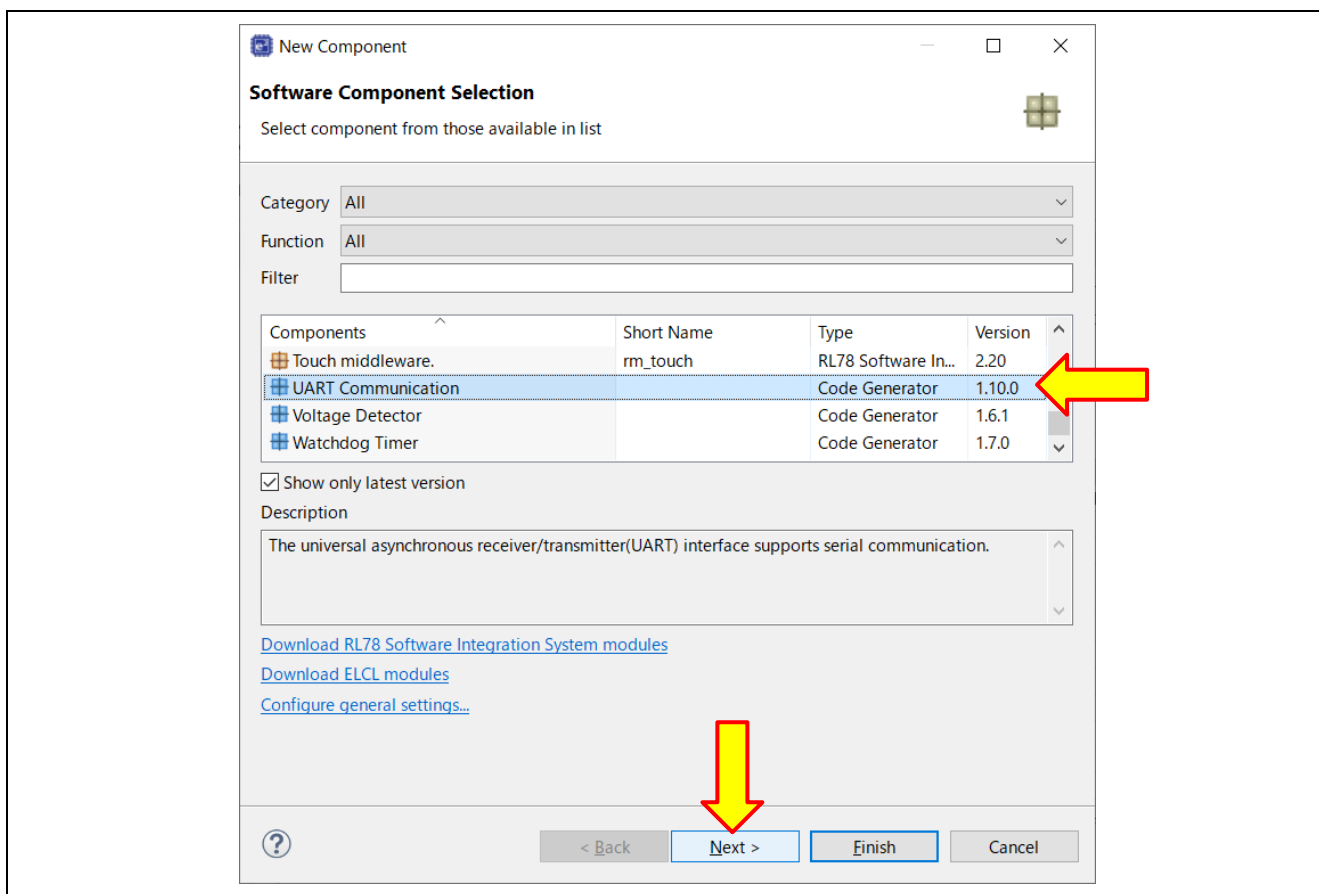


Figure 7-3 Software Component Settings (2)

- Set “Operation” mode to “**Transmission/reception**”, “Resource” to “**UARTA1**”, and click [**Finish**] in the lower part of that open dialog box.

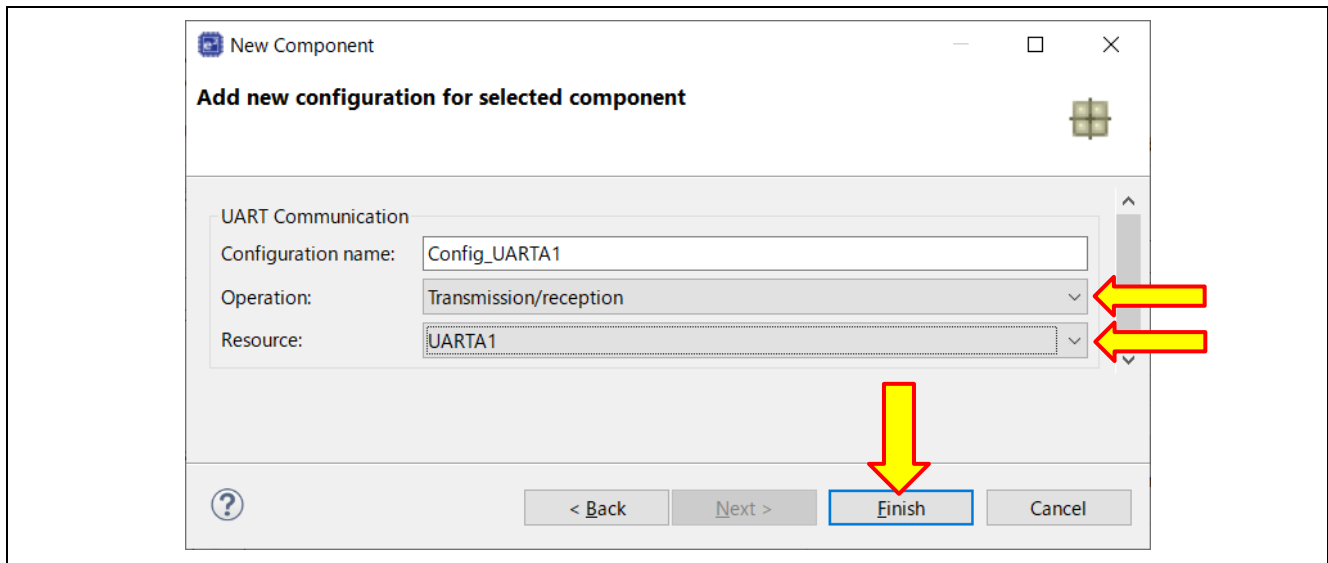


Figure 7-4 Component Resource Settings (2)

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

4. In the [Components] tab, select the “**Config\_UARTA1**” module.

Set the “operation clock” to “**fSEL/8**” and “**fSEL clock select fIHP**”, select “**Continuous transmit by interrupt**”, and the transfer rate setting to “**153600**” (bps).

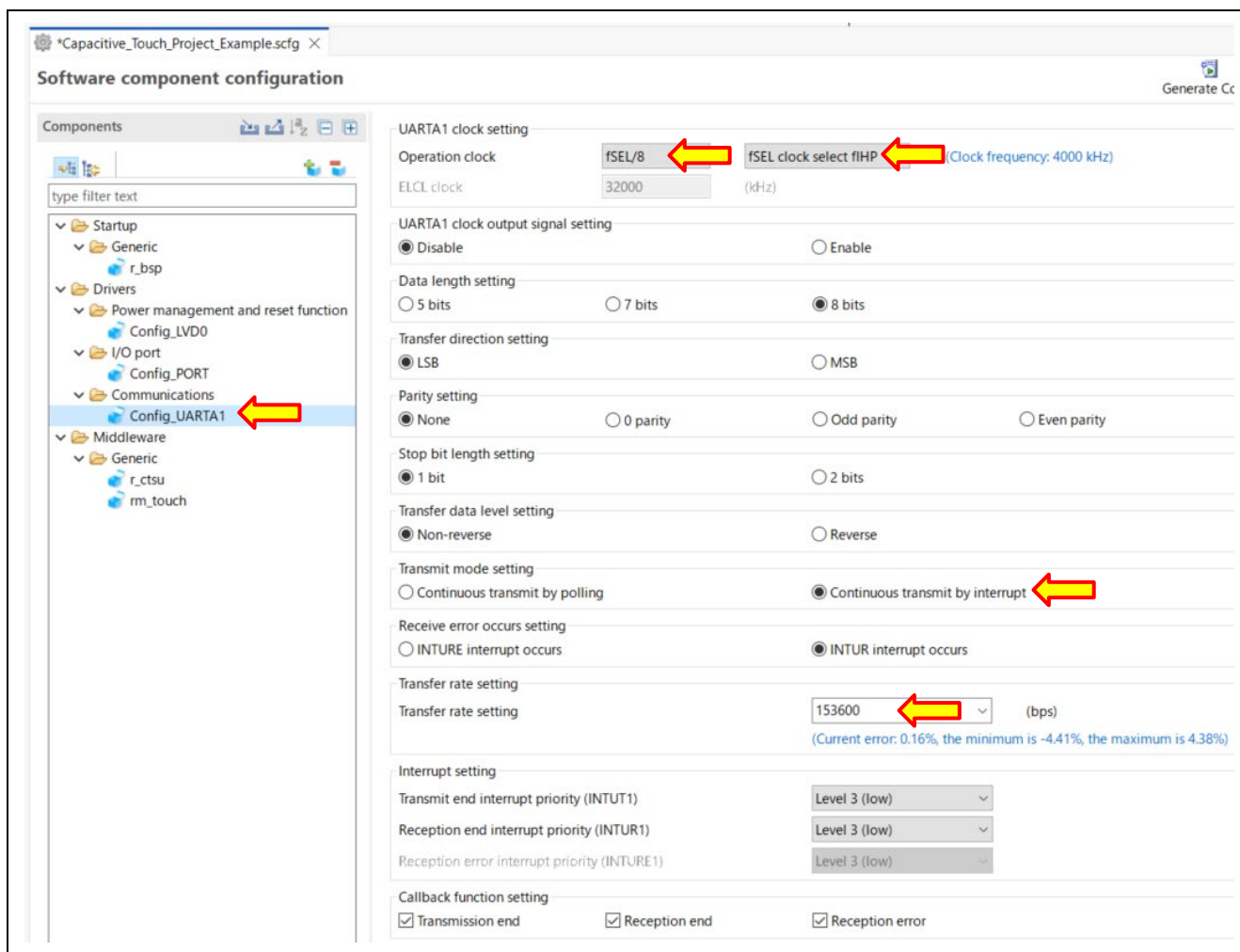


Figure 7-5 Setting the UART Communications Component (UARTA1)

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

When using UARTx instead of UARTAx, refer to Note 2 on the following pages.

**Note1.** Depending on the version of the Smart Configurator, an error may occur regarding the transfer rate setting. If this error occurs, change the program in the red frame below to set the transfer rate after generating the code.

```

Config_UARTA1.c
10 20 30 40 50 60 70
59 void R_Config_UARTA1_Create(void)
60 {
61     UARTEAEN1 = 0U;
62     UTMK1 = 1U; /* disable INTUT1 interrupt */
63     UTIF1 = 0U; /* clear INTUT1 interrupt flag */
64     URMK1 = 1U; /* disable INTUR1 interrupt */
65     URIF1 = 0U; /* clear INTUR1 interrupt flag */
66     UREMK1 = 1U; /* disable INTURE1 interrupt */
67     UREIF1 = 0U; /* clear INTURE1 interrupt flag */
68     /* Set INTUT1 low priority */
69     UTPR11 = 1U;
70     UTPR01 = 1U;
71     /* Set INTUR1 low priority */
72     URPR11 = 1U;
73     URPR01 = 1U;
74     /* Set INTURE1 low priority */
75     UREPR11 = 1U;
76     UREPR01 = 1U;
77     BRGCA1 = 00_UARTA_OUTPUT_BAUDRATE;
78     ASIMA11 = 00_UARTA_PARITY_NONE | 18_UARTA_TRANSFER_LENGTH_8 | 01
79     00_UARTA_DATA_NORMAL;
80     ASIMA10 = 02_UARTA_BUFFER_EMPTY | 01_UARTA_INTUR_OCCUR;
81     UTA0CK |= 20_UARTA_FSEL_SELECT_FIHP;
82     UTA1CK = 00_UARTA_CLKA1_OUTPUT_DISABLE | 03_UARTA1_SELECT_FSEL8;
83 }

```

Figure 7-6 Config\_UARTA1.c



**Note2.** The UART channels and ports used by this tool depend on your target board. For example, when using UART1, the settings are different as shown in the image below.

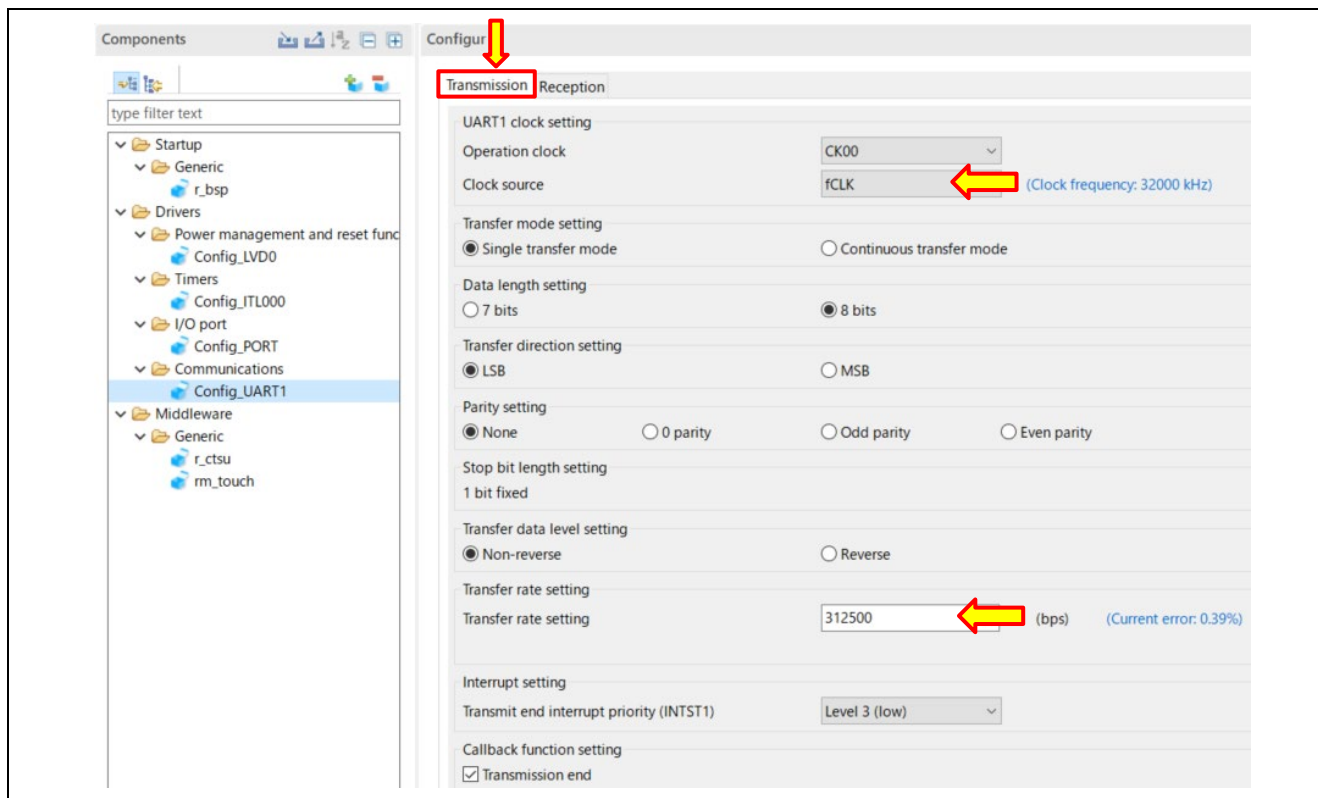


Figure 7-7 Setting the UART Communications Component (UART1 Transmission)

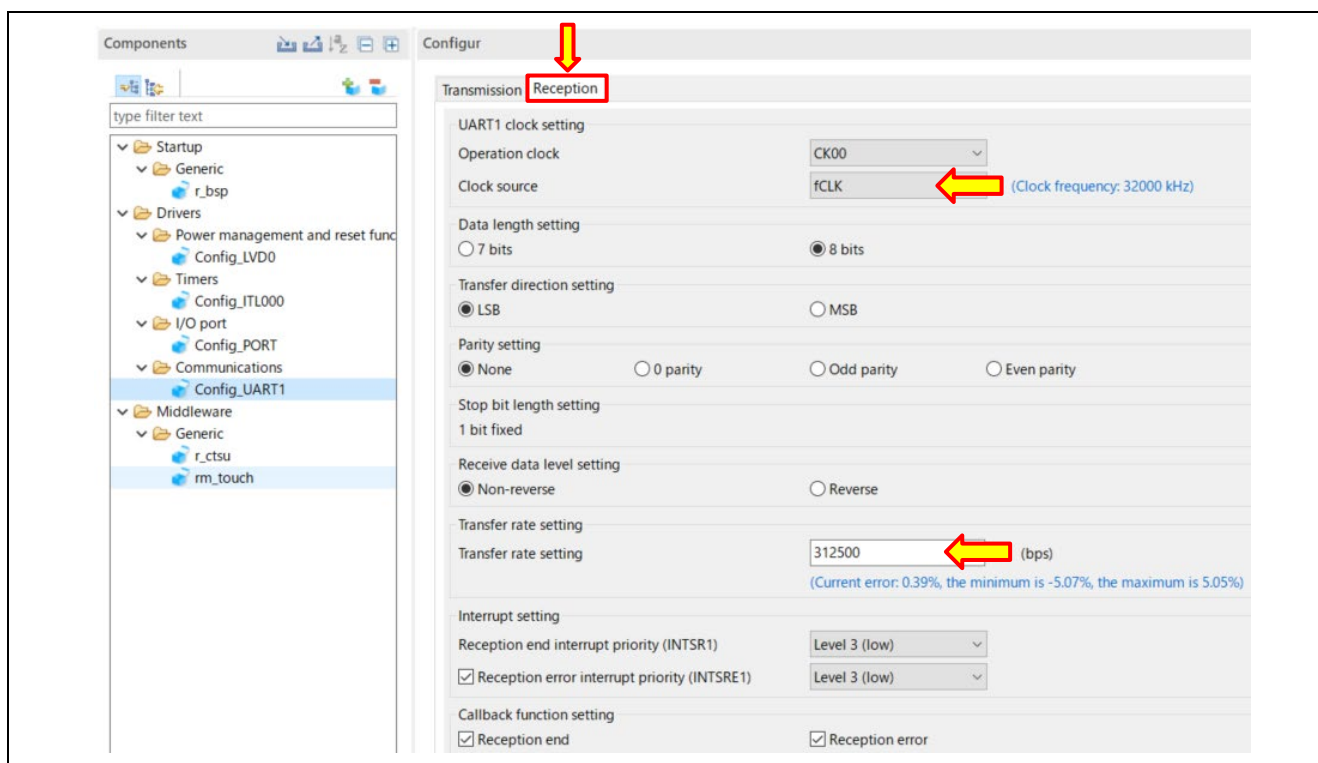


Figure 7-8 Setting the UART Communications Component (UART1 Reception)

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

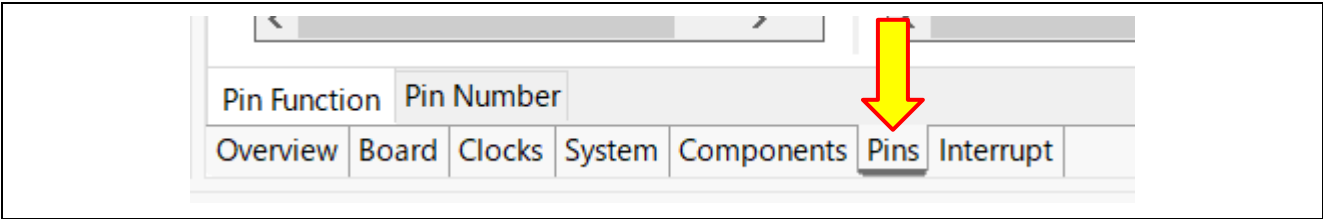


Figure 7-9 [Pins] tab Selection

6. Assign **RxDA1** function to **P33** and **TxDA1** function to **P34**.

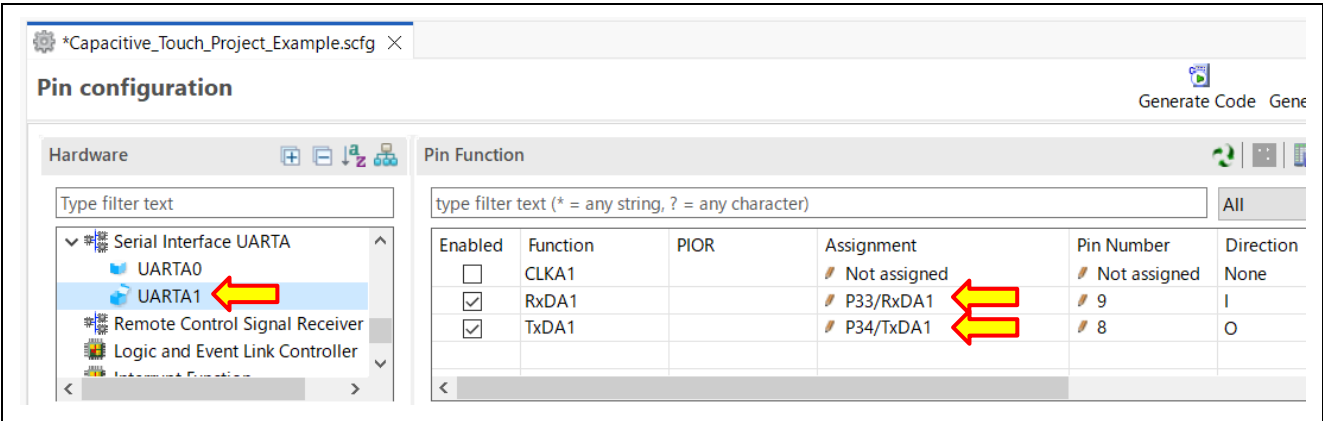


Figure 7-10 Assigning Pins to the UART Channel (UARTA1)

**Note:** The UART channels and ports used by this tool depend on your target board. For example, when using UART1, the settings are different as shown in the image below.

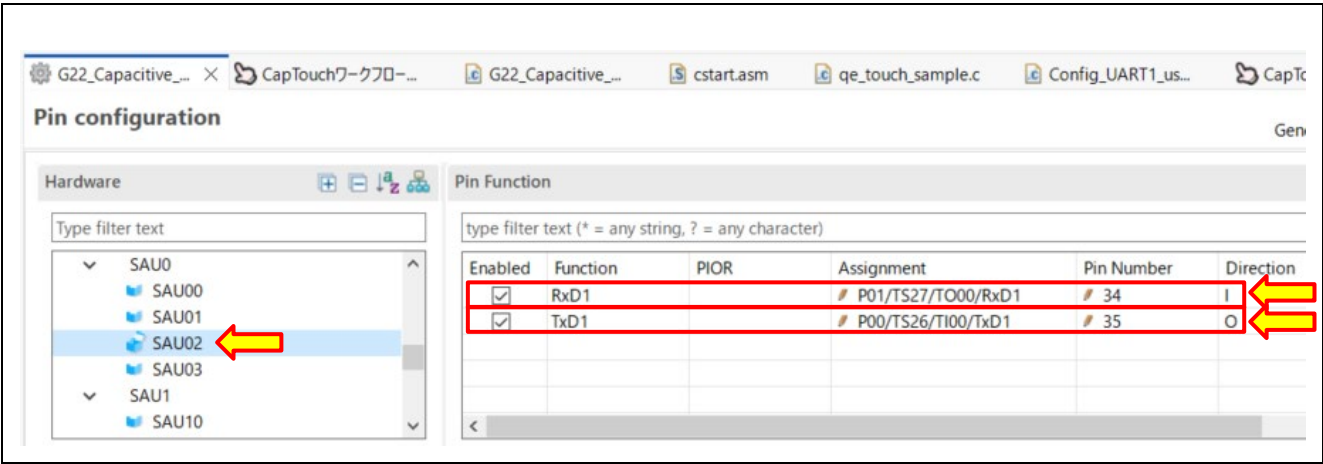



Figure 7-11 Assigning Pins to the UART Channel (UART1)

- Click the **"Generate Code"** icon  located on the top right of the Smart Configurator to generate the code for components necessary for the project.

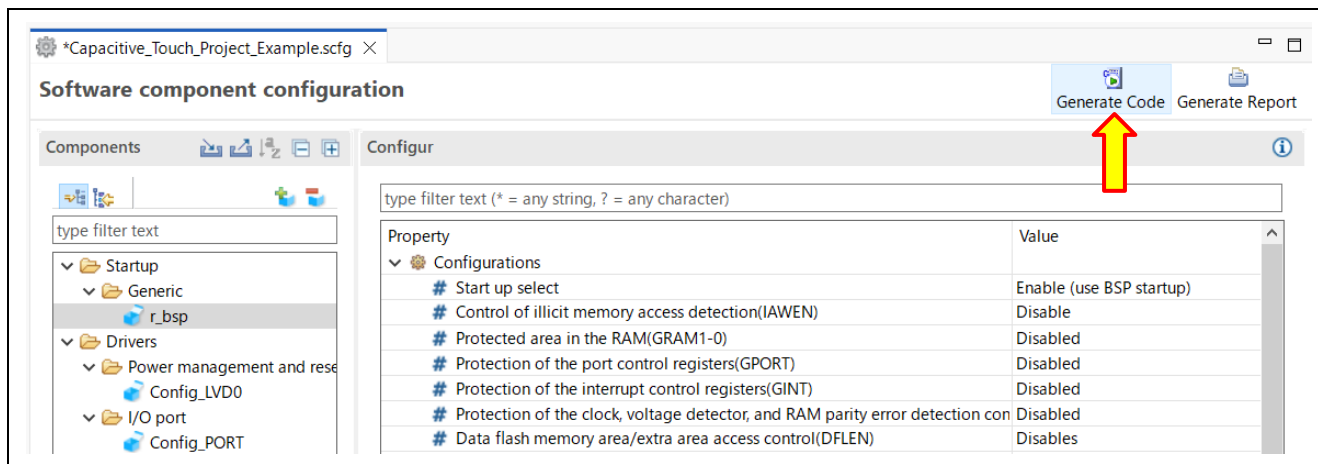


Figure 7-12 "Generating Code" icon Selection (2)

## 8. Creating the Capacitive Touch Interface

1. From the e<sup>2</sup> studio IDE, use **Renesas Views -> Renesas QE -> CapTouch Workflow (QE)** to open the main perspective for configuring capacitive touch to the project.

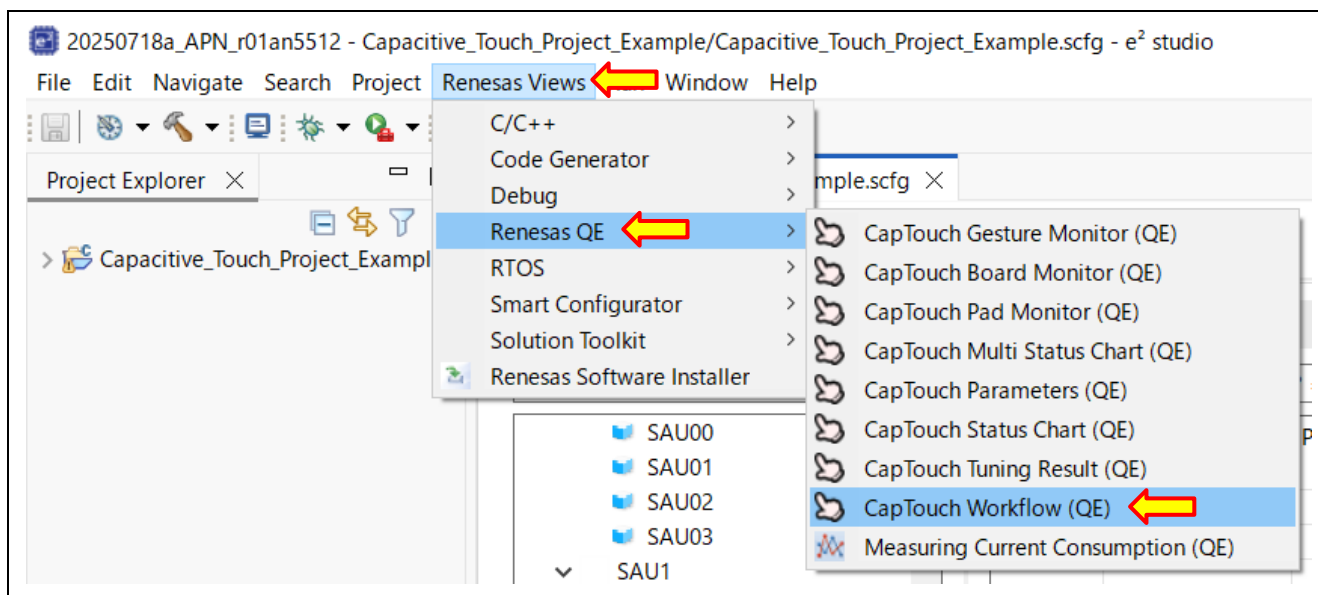


Figure 8-1 Cap Touch Workflow (QE) Selection

**Note:** The label on the selection button differs depending on the version of QE.

For versions earlier than QE V3.1.0, select [CapTouch Main (QE)].

For versions QE V3.2.0 or later, select [CapTouch Workflow (QE)].

2. [CapTouch Workflow (QE)] pane, "Select a Project" to configure the touch interface for by using the pull-down tab and selecting the "Capacitive\_Touch\_Project\_Example" project as shown below.

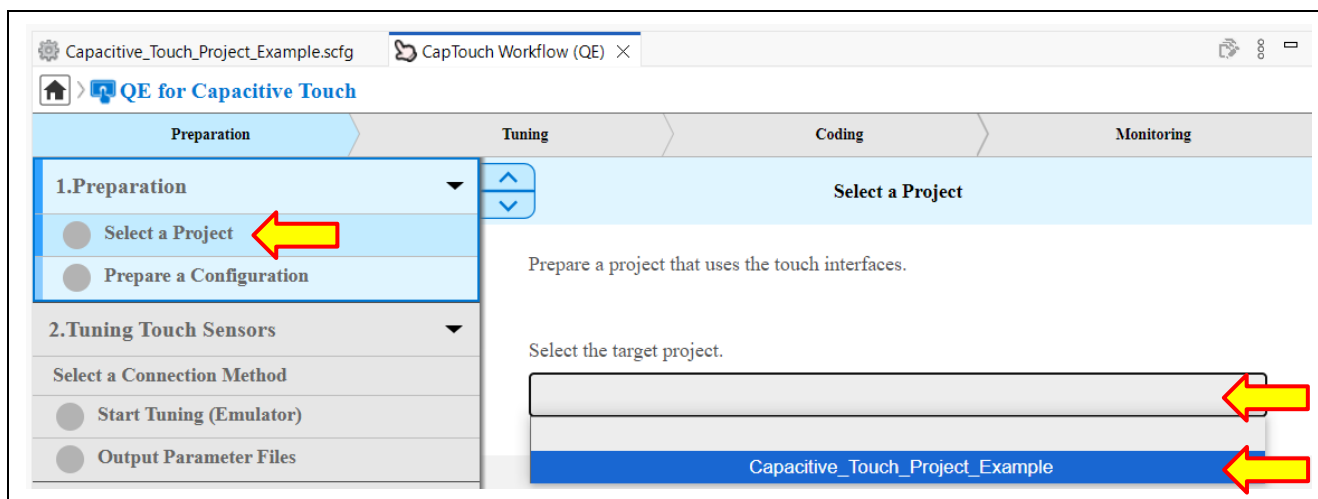


Figure 8-2 Project Selection

3. Select **[Create a new configuration]** from the pulldown menu in **“Prepare a Configuration”** to generate a new touch interface configuration.

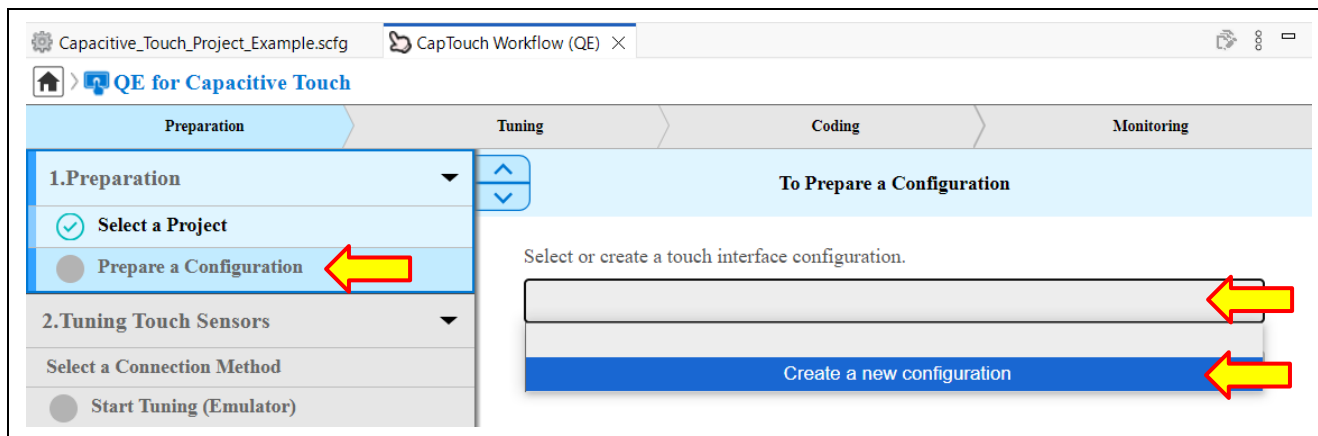


Figure 8-3 Create a new configuration

4. **“Create Configuration of Touch Interfaces”** window opens, providing an area to layout the touch interface.

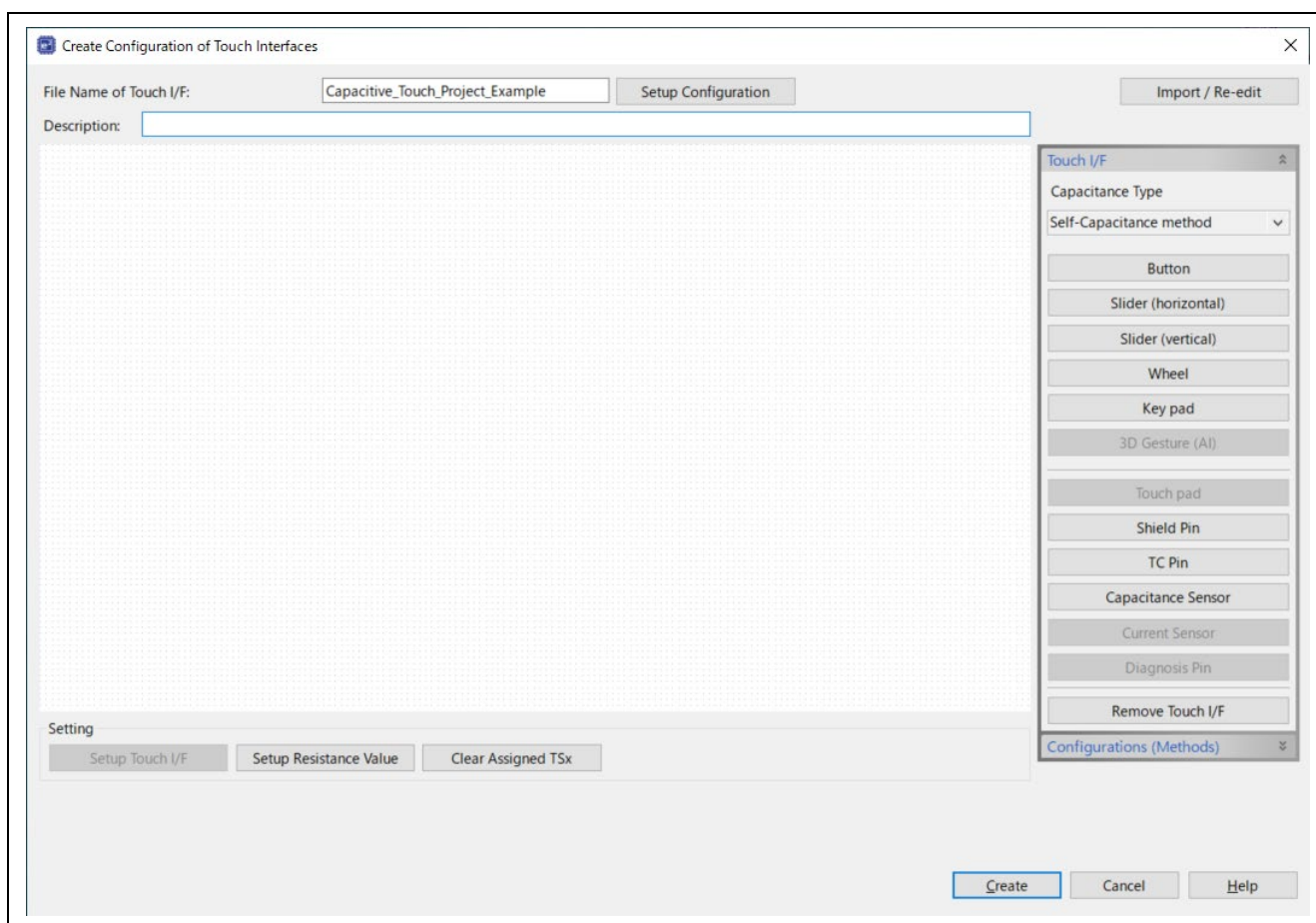


Figure 8-4 Create Configuration of Touch Interfaces (1)

5. Select **[Button]** from “Touch I/F” on the right side of in the “**Create Configuration of Touch Interfaces**” window. Add two buttons to the layout area.

Press **[Esc]** on your keyboard to finish adding the touch interface. The layout should now look like the figure below.

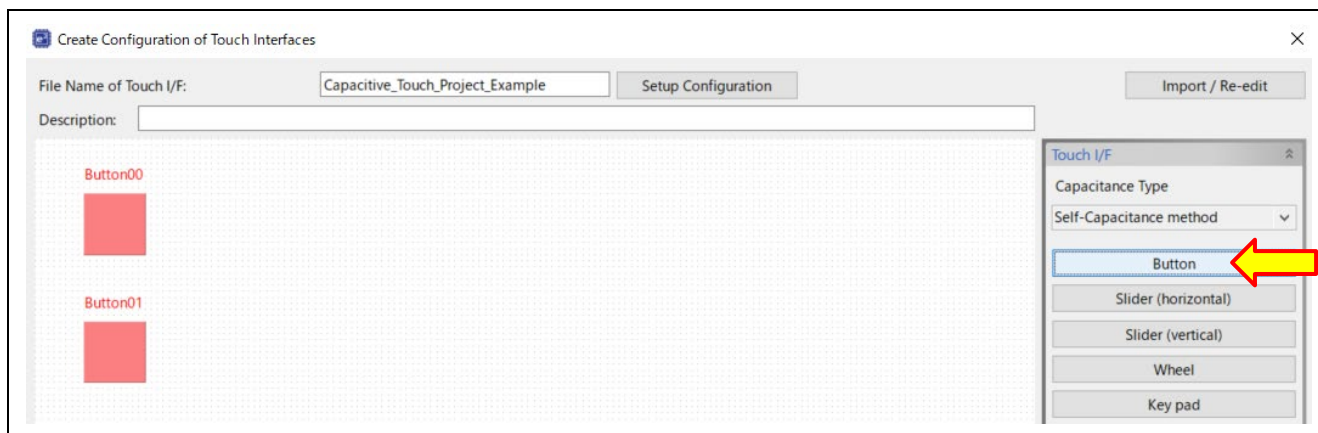


Figure 8-5 Create Configuration of Touch Interfaces (2)

6. Double click “**Button00**” to display the “Setup Touch Interface” dialog box. In this case, using the pull-down, select **TS06** as the touch sensor to assign to this button.

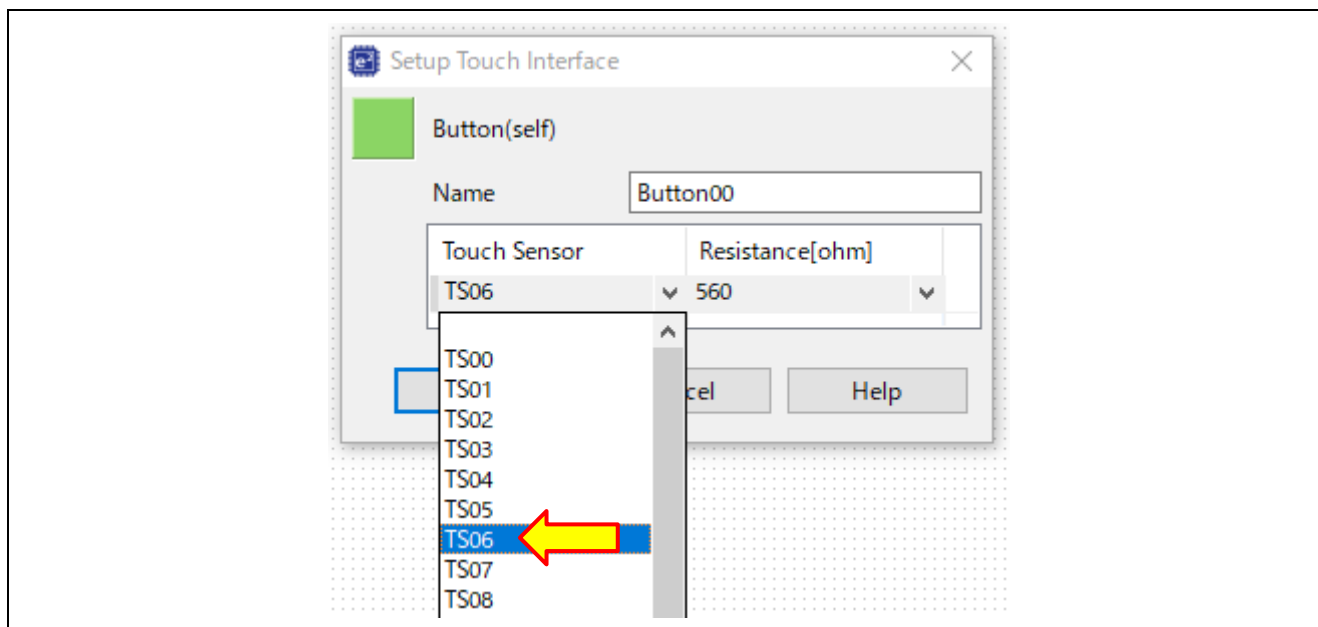


Figure 8-6 Create Configuration of Touch Interfaces (3)

7. Perform the same operation as the previous step for **Button01** and assign it to **TS05**. 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.

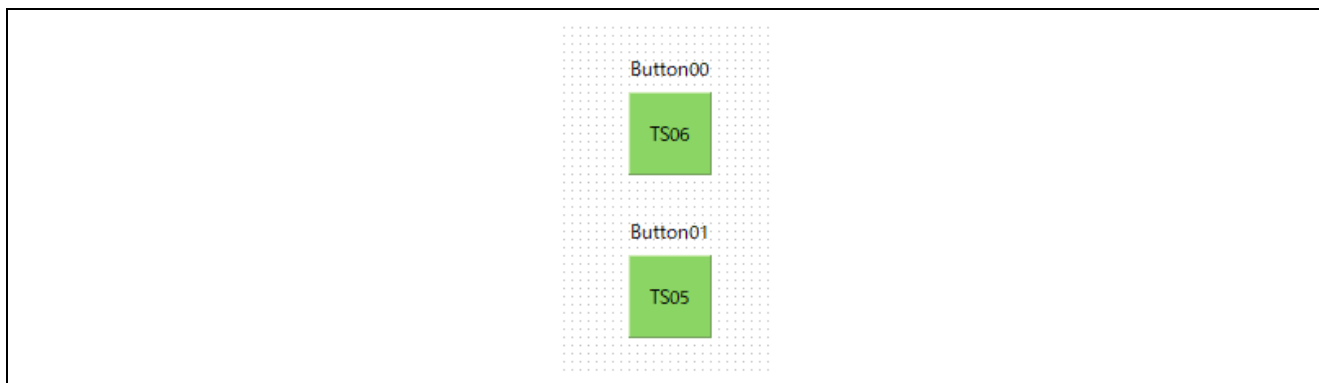


Figure 8-7 Create Configuration of Touch Interfaces (4)

8. Click [**Create**] in the “Create Configuration of Touch Interface” window.  
This completes the touch interface settings.

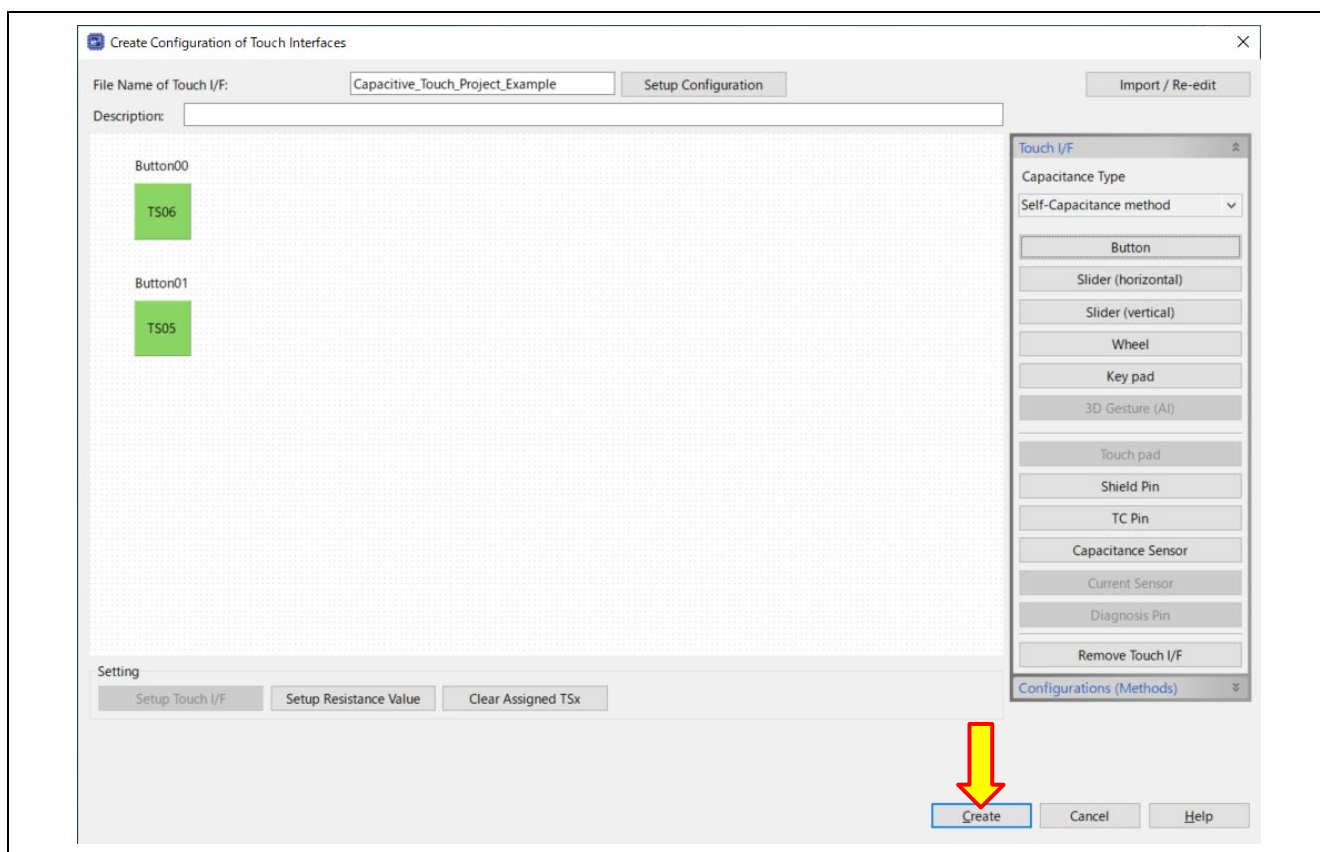



Figure 8-8 Create Configuration of Touch Interfaces (5)



9. Click the hammer icon  on the top left in e<sup>2</sup> studio to build the project.

Confirm that there is no error displayed in the “Console” window.

This completes the creation of the capacitive touch interface.

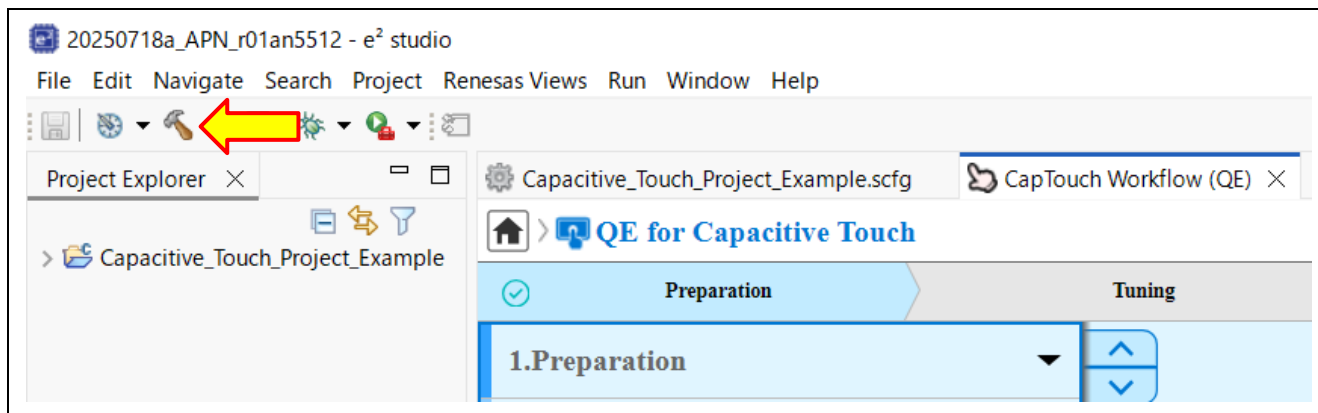



Figure 8-9 Project Build



## 9. Changing the Debug Configuration for Capacitive Touch Sensor Tuning

The debug configuration needs to be changed so that the tuning kernel can be downloaded to the MCU RAM after the debug session is started.

1. Click ▼ next to  and select "Debug Configurations" from the pulldown menu.

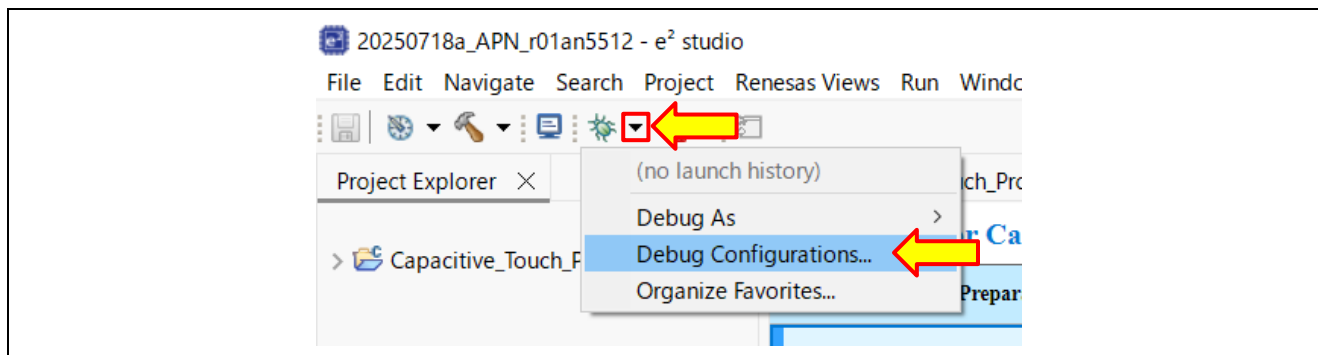


Figure 9-1 Select "Debug Configurations"

2. Click "Debug Configuration (e.g., xxxxx HardwareDebug)" on the panel that appears, and select the [Debugger] tab and [Connection Settings] tab. For this application example, the power for the target board is supplied from the emulator power supply. Ensure "Power Target From The Emulator (MAX 200mA)" and "Supply Voltage [V]" are set as shown in the red frame below.

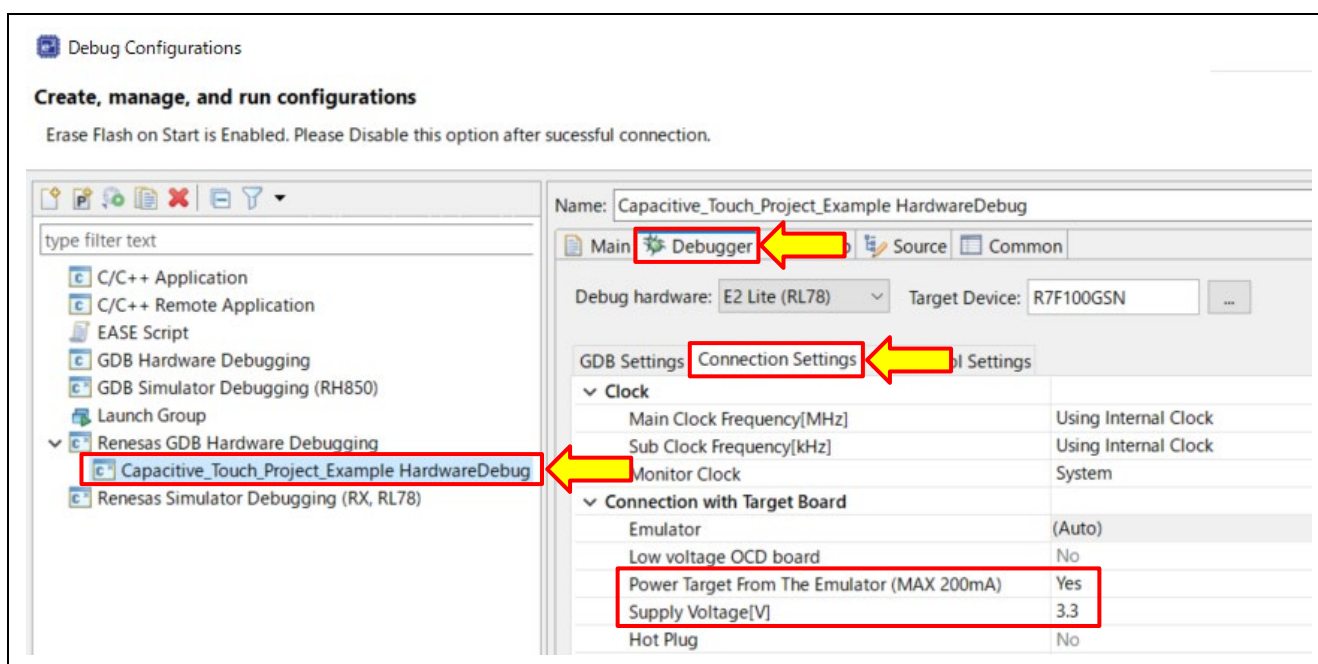


Figure 9-2 Debug Configuration (E2 Lite) Settings

- Note1.** In the application example, power is supplied to the target board from the emulator for easy confirmation of the operation. The power can also be supplied to the target board via E2 emulator Lite from the USB port on the PC, however we recommend using power generated by the target board.
- Note2.** Available debugging methods vary depending on the specifications of the target board. Set up the required items by selecting “Debug hardware:” according to the debug method you are using.  
For example, COM port debugging requires a different setting.

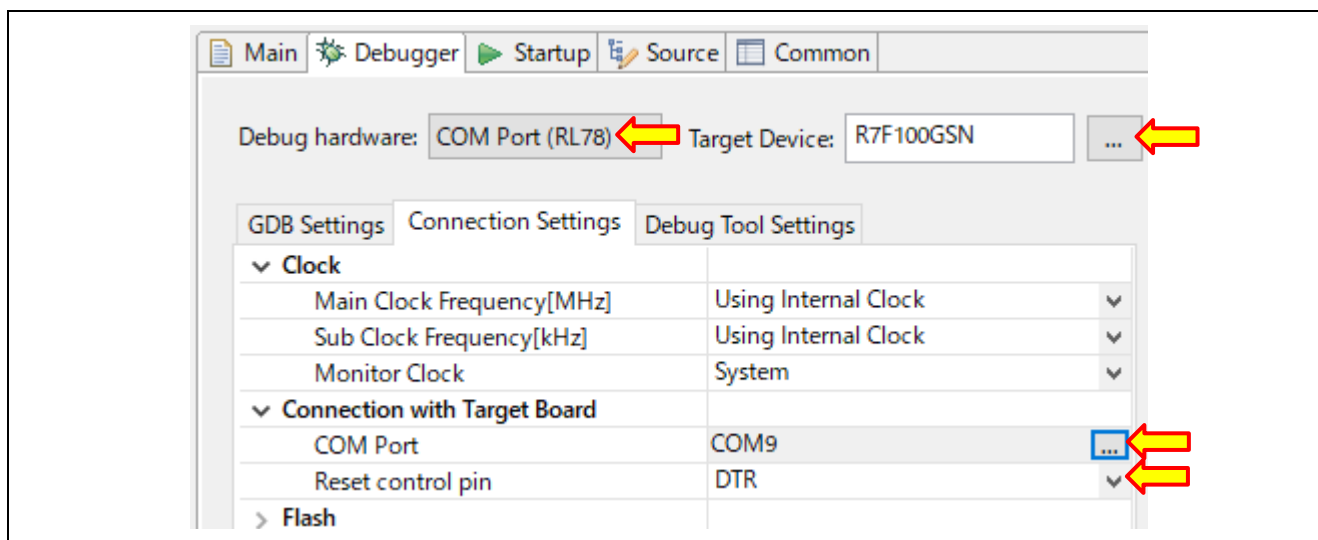


Figure 9-3 Debug Configuration (COM port) Settings

3. Select the [**Debug Tool Settings**] tab. Set "Allow to access by stopping execution while running" under "Memory" to "Yes".

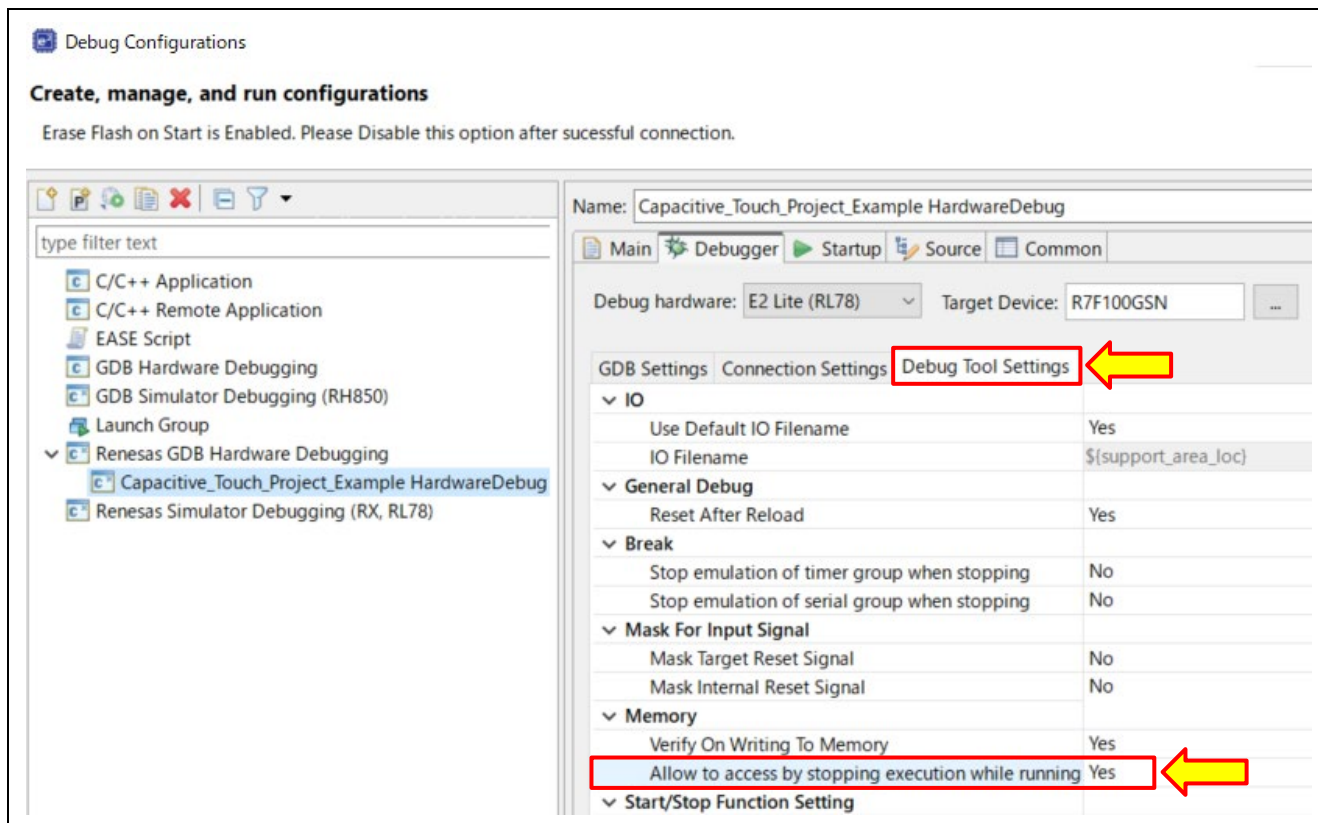


Figure 9-4 Debug Tool Settings

4. Select the **[Startup]** tab. Confirm that “**Set breakpoint at:**” and “**Resume**” are selected as shown in the figure below, then click **[Apply]** and **[Close]**. This completes the debug configuration for tuning.

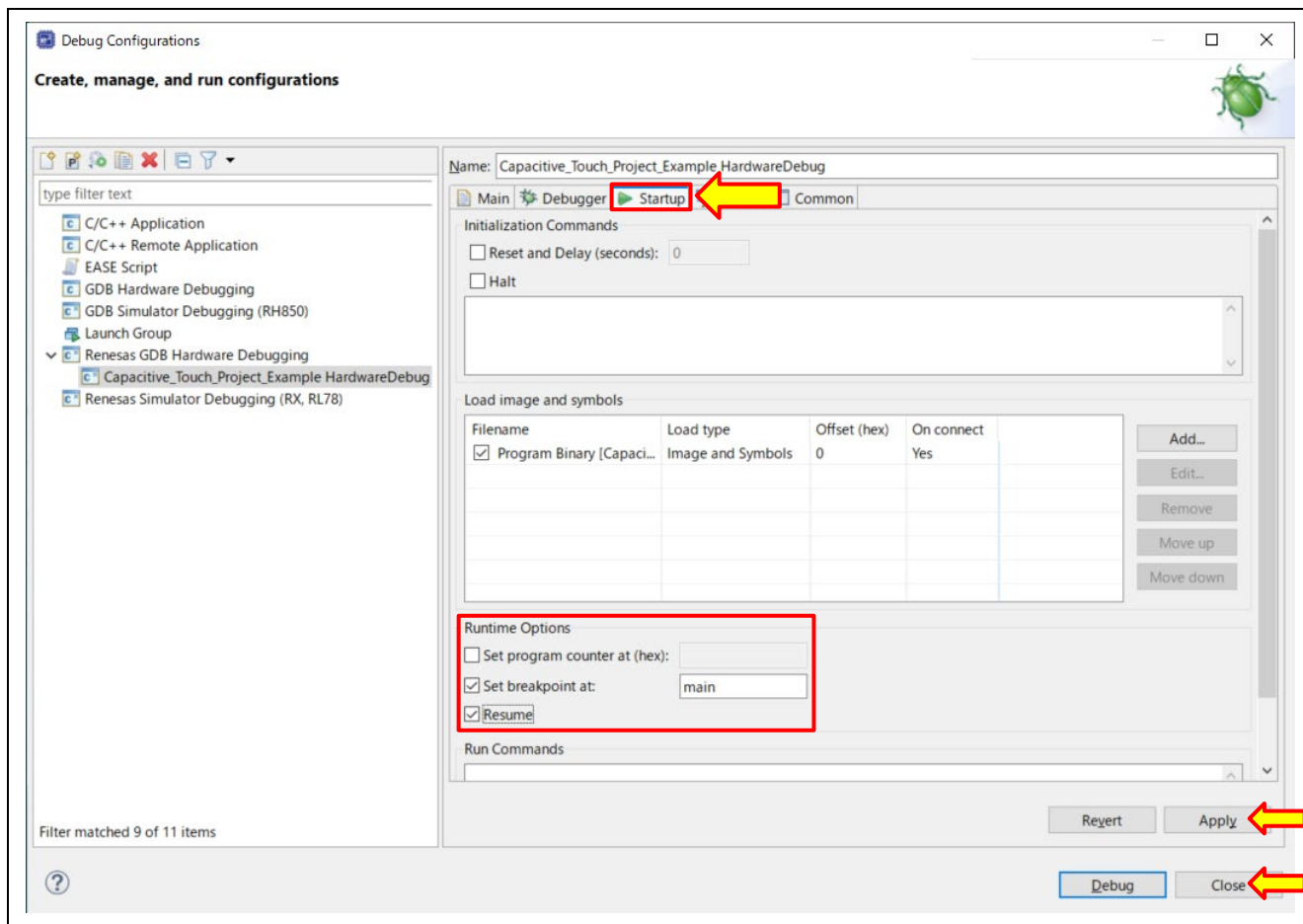


Figure 9-5 Runtime Options Settings

## 10. Capacitive Touch Sensor Tuning with QE for Capacitive Touch

This section describes how to tune the project using QE for Capacitive Touch.

1. Click **[Start Tuning]** in **[CapTouch Workflow (QE)]** to start automatic tuning.

**Note:** In the application example, power is supplied to the target board from the emulator for easy confirmation of the operation. The power can also be supplied to the target board via E2 emulator Lite from the USB port on the PC, however we recommend using power generated by the target board for tuning.

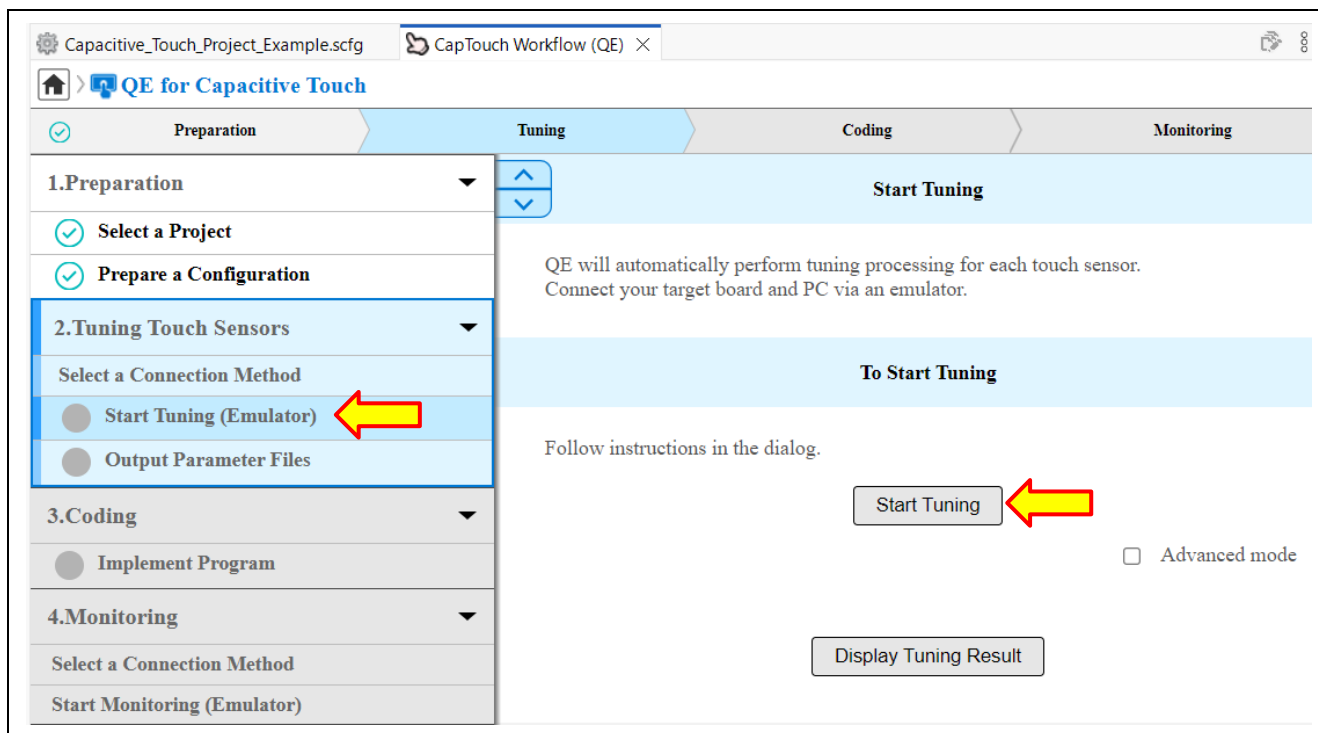


Figure 10-1 Start Automatic Tuning

- Set the same value as the voltage [V] supplied to the MCU as shown below, and click [OK].

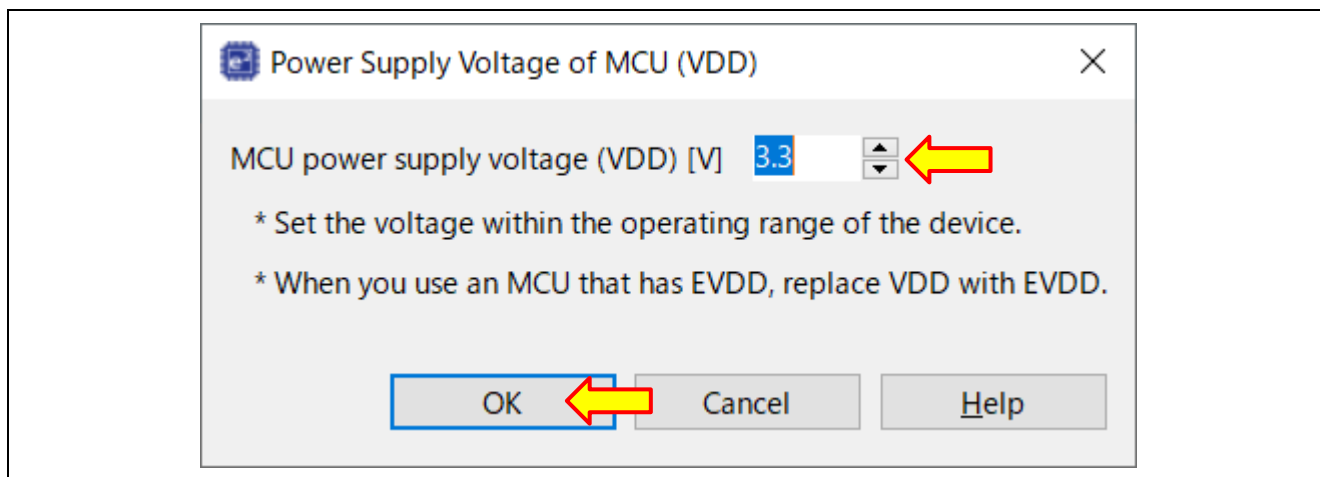


Figure 10-2 Setting the Voltage Value Supplied to MCU (tuning setting)

**Note:** In this application example, power is supplied from the E2 emulator Lite. Therefore, the “MCU power supply voltage (VDD) [V]” is 3.3 V.

- When the debug session is started, e<sup>2</sup> studio may display the message to switch to the debug perspective. If the message is displayed, check [Remember my decision(R)], then click [Switch] to continue the debug session and automatic tuning for QE for Capacitive Touch.

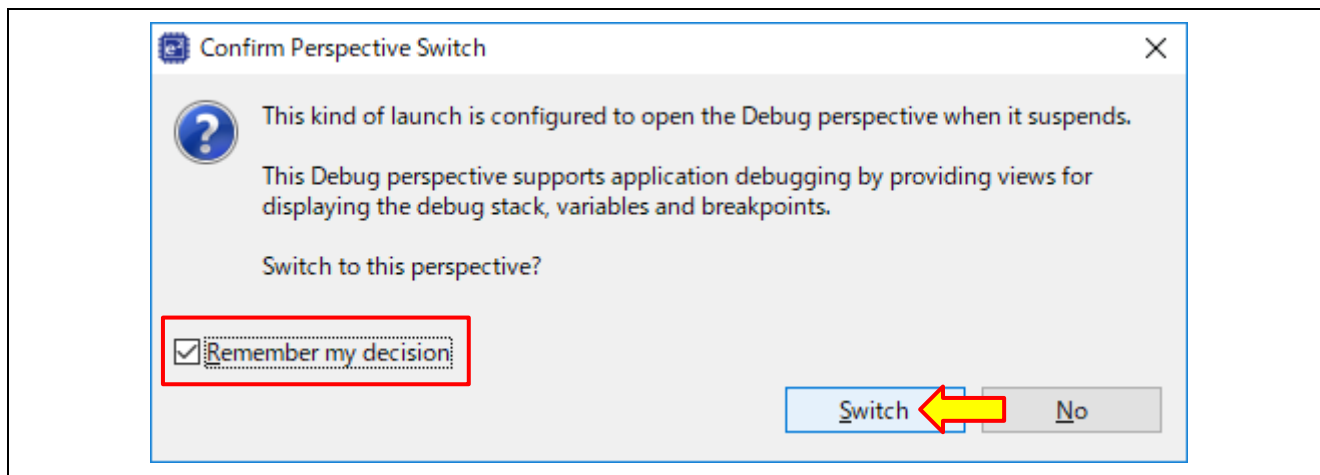


Figure 10-3 Confirm Perspective Switch

4. The following message will appear. Click [Yes].

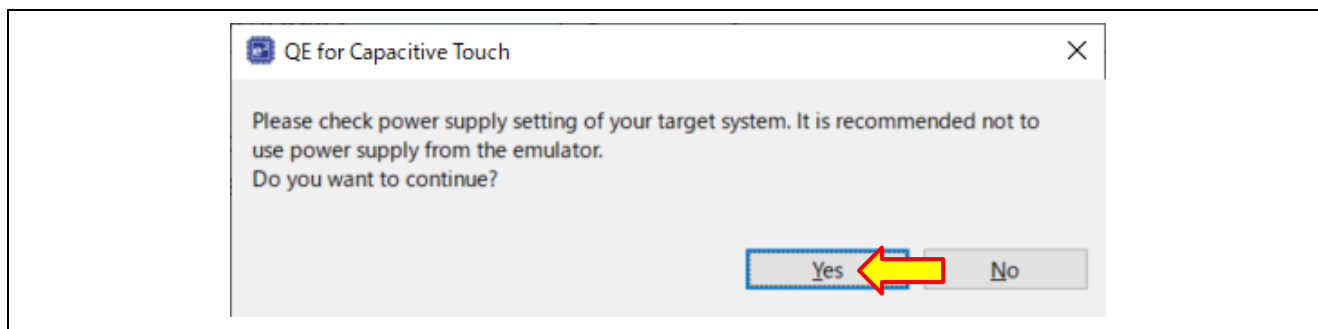


Figure 10-4 Confirmation of Power Settings for Target System

5. Automatic tuning for QE for Capacitive Touch is started. Continually confirm the "Automatic Tuning Processing" dialog box that guides the tuning process.

The following is an example of the dialog box. Normally, the initial tuning processing will not require any action on the part of the user.

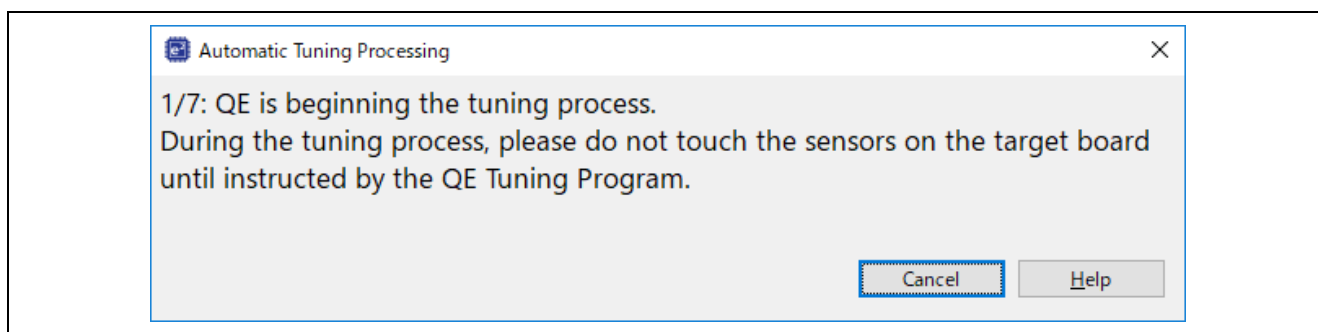


Figure 10-5 "Automatic Tuning Processing" Dialog Box (during initial tuning process)

The dialog box as shown below is displayed after several steps in the process. At this point, touch sensitivity in the tuning processing is measured. Touch the sensor (**Button01/TS05**) indicated in the dialog box with normal pressure. When touching the sensor, the bar graph increases to the right and the number indicating the touch count value increases. Press any key on your PC keyboard while touching the sensor to confirm the measurement.

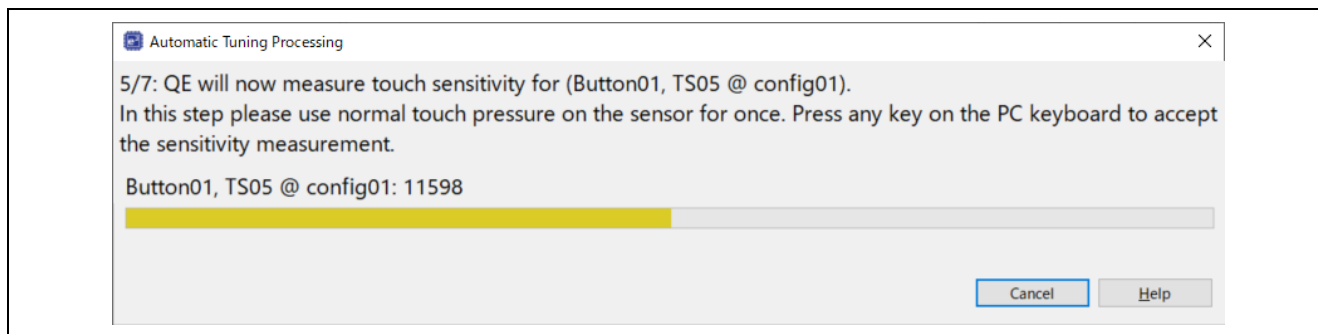


Figure 10-6 "Automatic Tuning Processing" Dialog Box (while measuring touch sensitivity)

6. Repeat the same procedure for **Button00/TS06**.
7. When the tuning is completed, the following dialog box appears showing the threshold values. These threshold values are used for touch event judgement in the middleware.

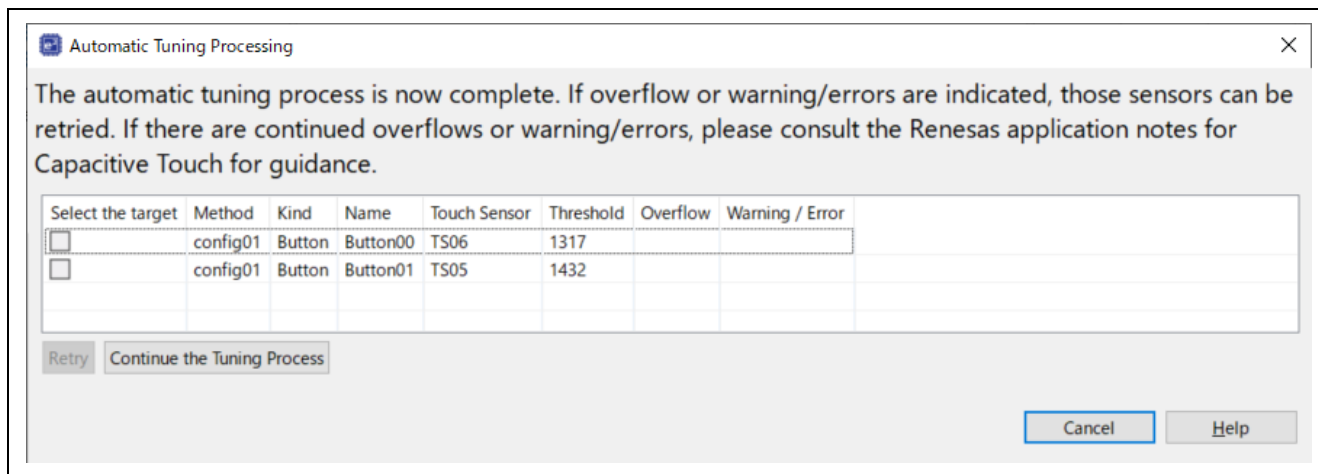


Figure 10-7 Tuning Results Display

8. Click [**Continue the Tuning Process**] in the displayed dialog box. The tuning process finishes and the target board and debugging session are disconnected. The process then goes back to [**CapTouch Workflow (QE)**].

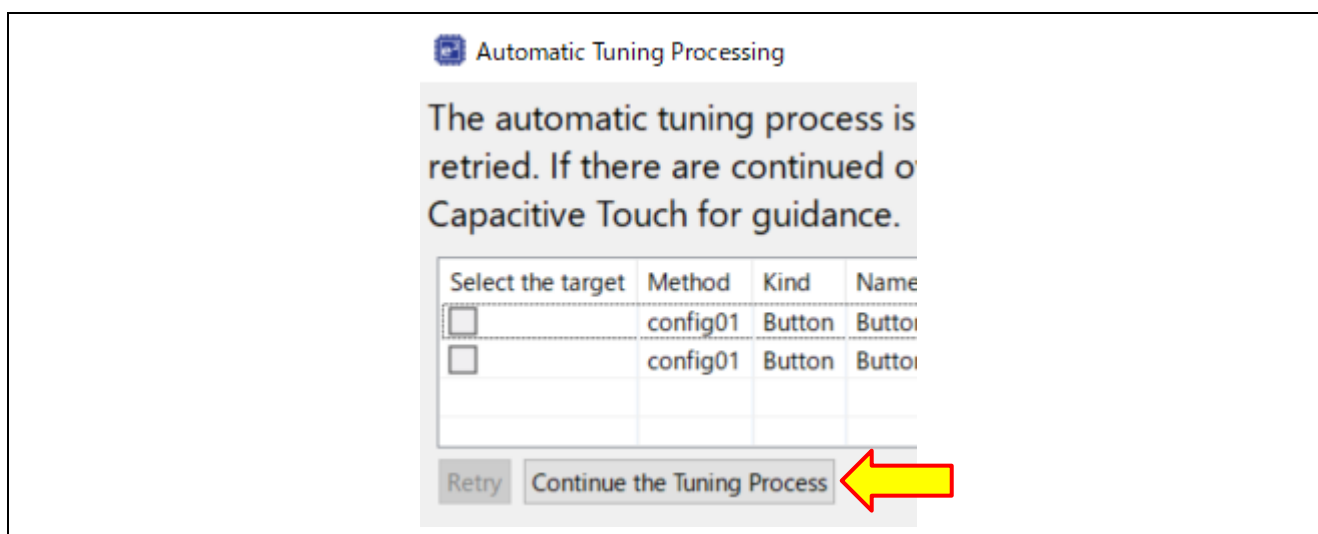
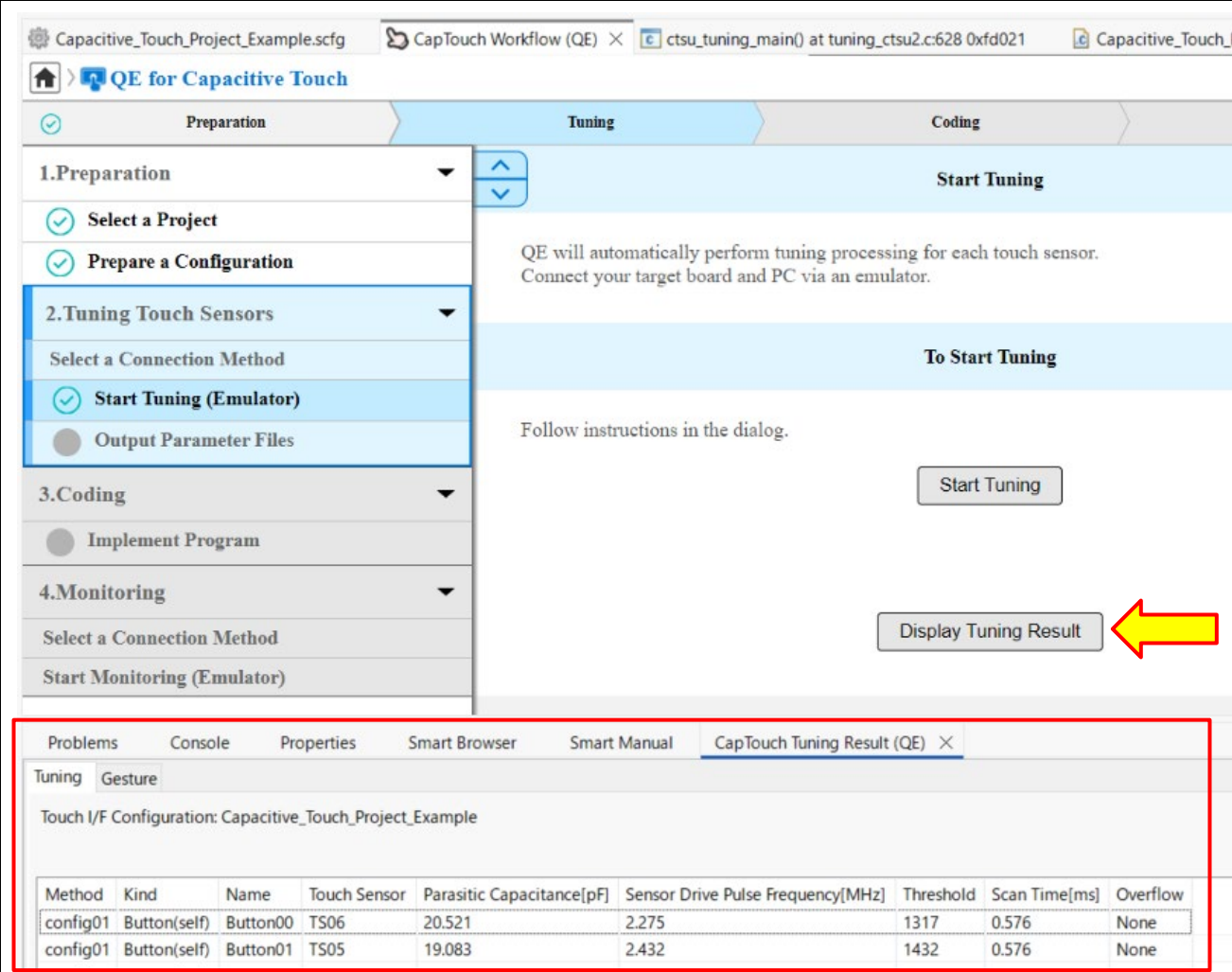


Figure 10-8 Continue the Tuning Process



9. Click **[Display Tuning Result]** in **[CapTouch Workflow (QE)]** to display **[CapTouch Tuning Result (QE)]** and check the detailed tuning results.



The screenshot shows the 'CapTouch Workflow (QE)' interface. The 'Tuning' stage is active, and the 'Display Tuning Result' button is highlighted with a red arrow. Below the workflow, the 'CapTouch Tuning Result (QE)' window is open, displaying a table of tuning results for two touch sensors.

Method	Kind	Name	Touch Sensor	Parasitic Capacitance[pF]	Sensor Drive Pulse Frequency[MHz]	Threshold	Scan Time[ms]	Overflow
config01	Button(self)	Button00	TS06	20.521	2.275	1317	0.576	None
config01	Button(self)	Button01	TS05	19.083	2.432	1432	0.576	None

Figure 10-9 CapTouch Tuning Result (QE)

10. Click [Output Parameter Files] as shown below to output the tuned parameter files.

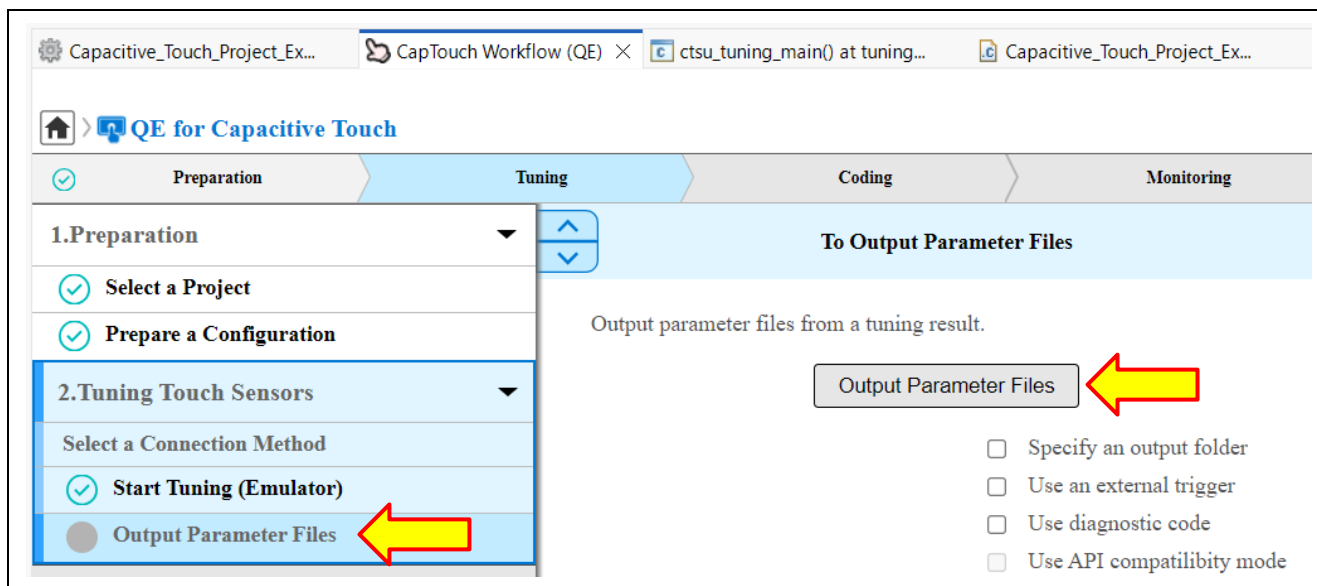


Figure 10-10 Output Parameter Files

11. Confirm that **qe\_touch\_config.c**, **qe\_touch\_config.h**, and **qe\_touch\_define.h** have been added in the "Project Explorer" window.

These files include the tuning information required to enable touch detection using the driver.

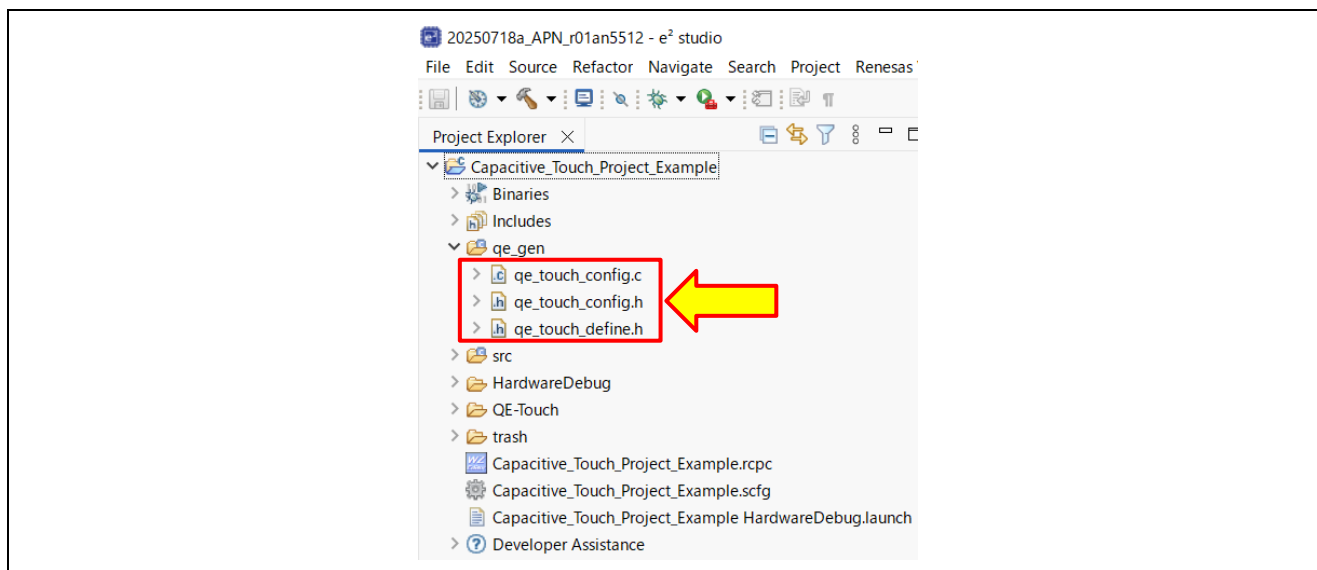



Figure 10-11 Tuning Parameter File Output

12. Click  on the top left of e<sup>2</sup> studio to build the project. In the "Console" window, confirm that no error has occurred during the build.

This completes the tuning of the capacitive touch sensor using QE for Capacitive Touch.

## 11. Adding API Call for rm\_touch Middleware

This section describes how to add the API call for the rm\_touch middleware to the project and enable capacitive touch control.

1. Click **[Show Sample]** in **[CapTouch Workflow (QE)]** to implement the program that scans the touch sensor.

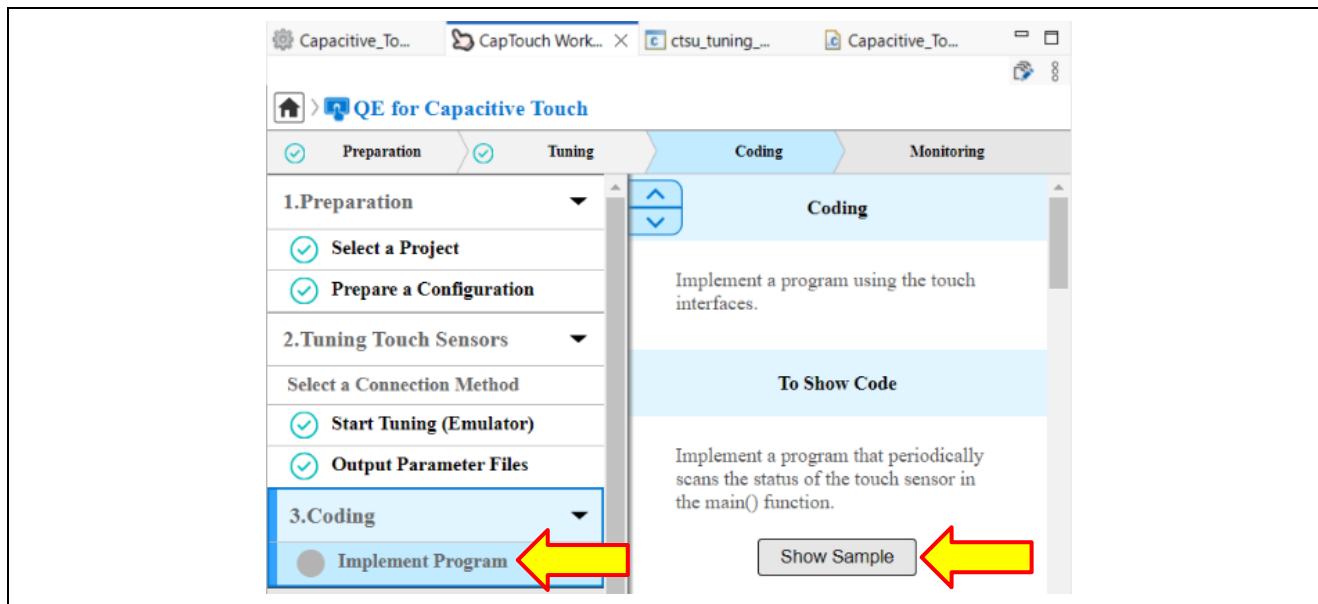


Figure 11-1 Sample Code Example Display

2. The "Show Sample Code" window opens and displays the sample code. Click **[Output to a File]** and **[OK]** to output the sample code.

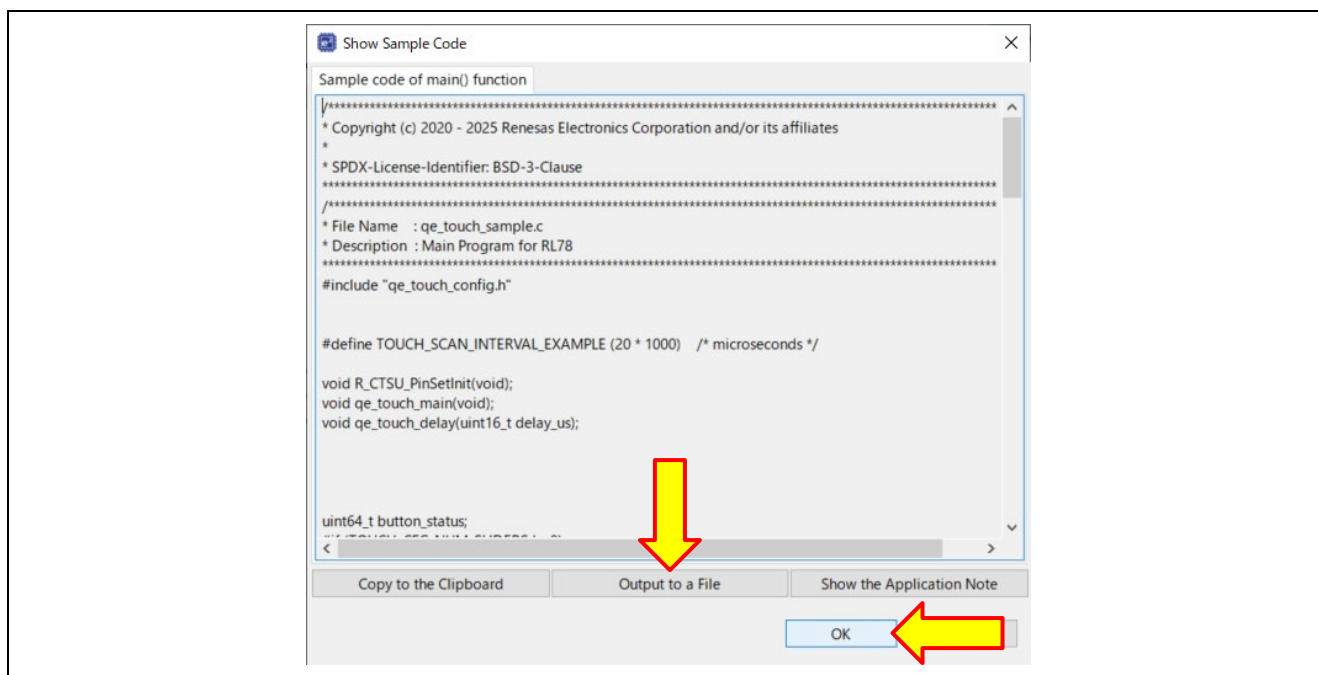


Figure 11-2 Sample Code Output to File

3. Confirm that “**qe\_touch\_sample.c**” file is generated in “Project Explorer”.

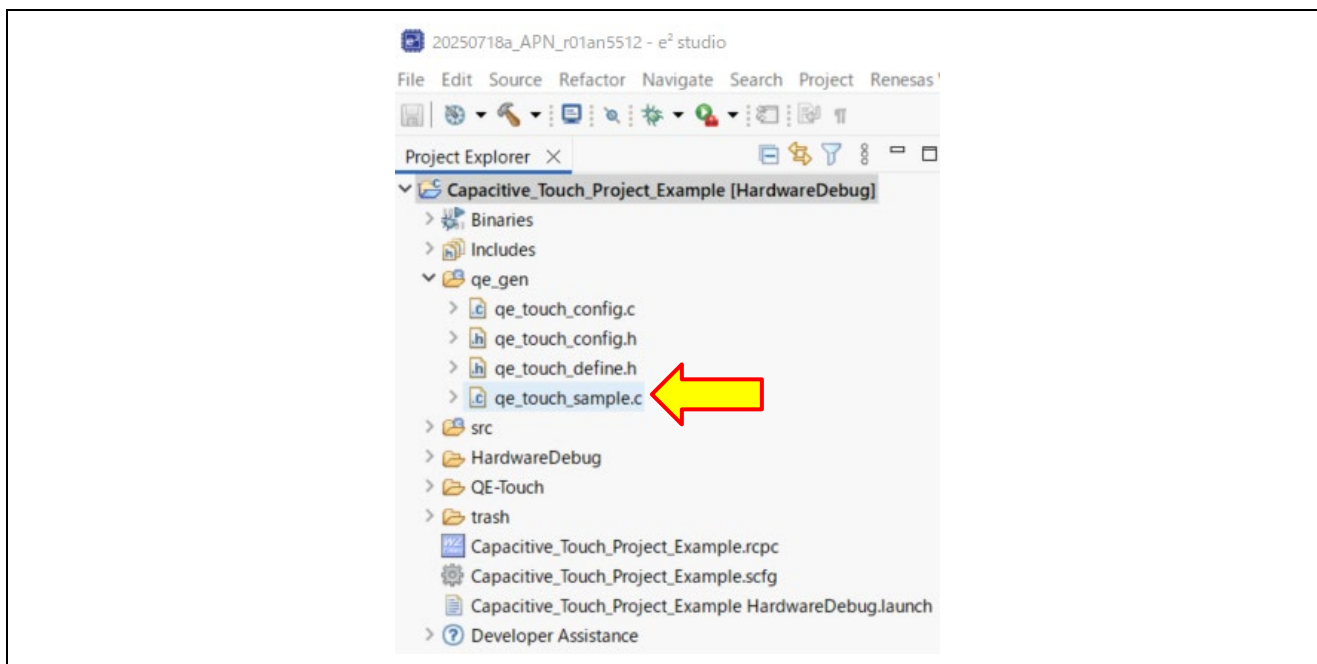


Figure 11-3 QE Output Code (qe\_touch\_sample.c) Generation Confirmation

4. Open “**Capacitive\_Touch\_Project\_Example.c**”.

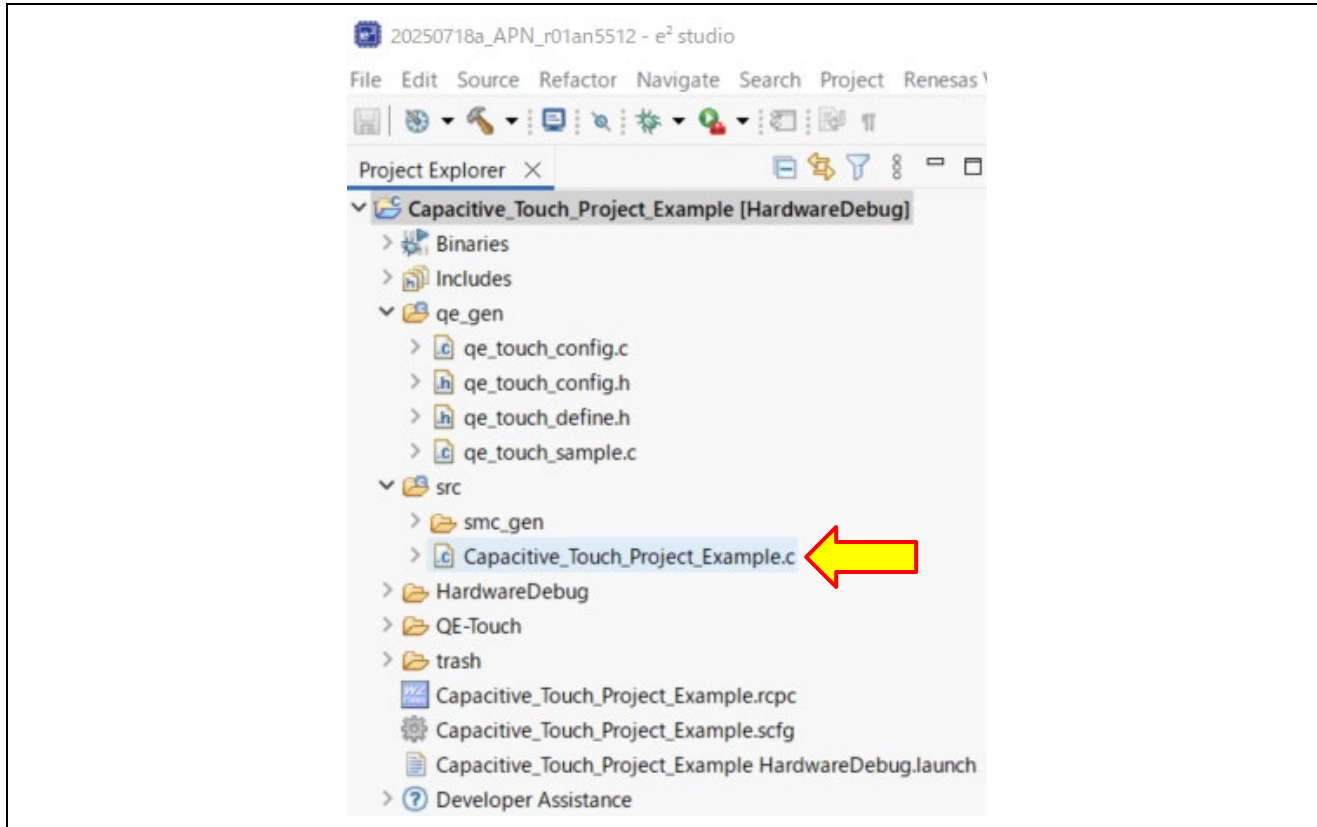


Figure 11-4 Select Capacitive\_Touch\_Project\_Example.c

5. Add codes “**void qe\_touch\_main(void);**” and “**qe\_touch\_main();**” (marked with red in the figure) into the “Capacitive\_Touch\_Project\_Example.c” file to call the qe\_touch\_main() function form the main() function.

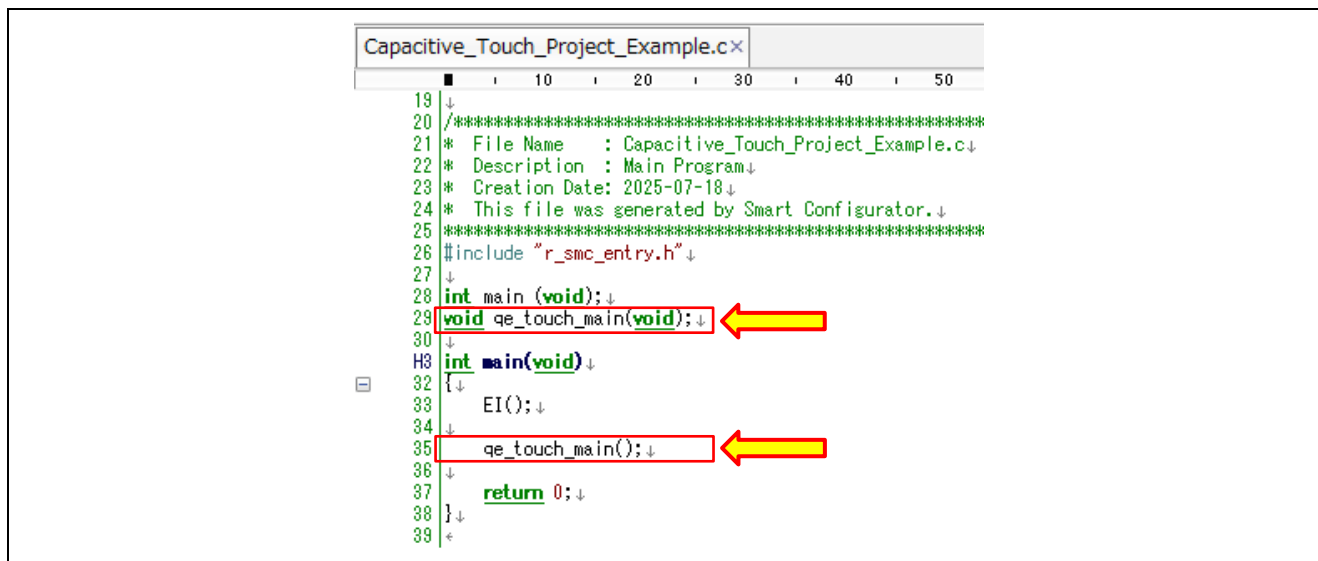



Figure 11-5 Set Call for qe\_touch\_main() Function

6. Click the hammer icon  on the top left in e<sup>2</sup> studio to build the project.  
Confirm that there is no error displayed in the “Console” window.  
This completes all the needed code modifications required for this simple application example.

## 12. [Additional function] Setting the serial communication monitor using UART (2/3)

**Note:** Monitoring touch performance for touch applications can be confirmed by communication via the OCD (On-Chip Debugging) emulator. However, RL78 family case, monitoring performance is limited by the OCD function of the RL78 family.

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 (This is the recommended setting.).

Chapters 7, 12 and 14 (including this chapter) shown below describe setting the serial communication monitor using UART.

- “7. [Additional function] Setting the serial communication monitor using UART (1/3)”
- “12. [Additional function] Setting the serial communication monitor using UART (2/3)”
- “14. [Additional function] Setting the serial communication monitor using UART (3/3)”

1. Open the “**Config\_UARTA1\_user.c**” file.

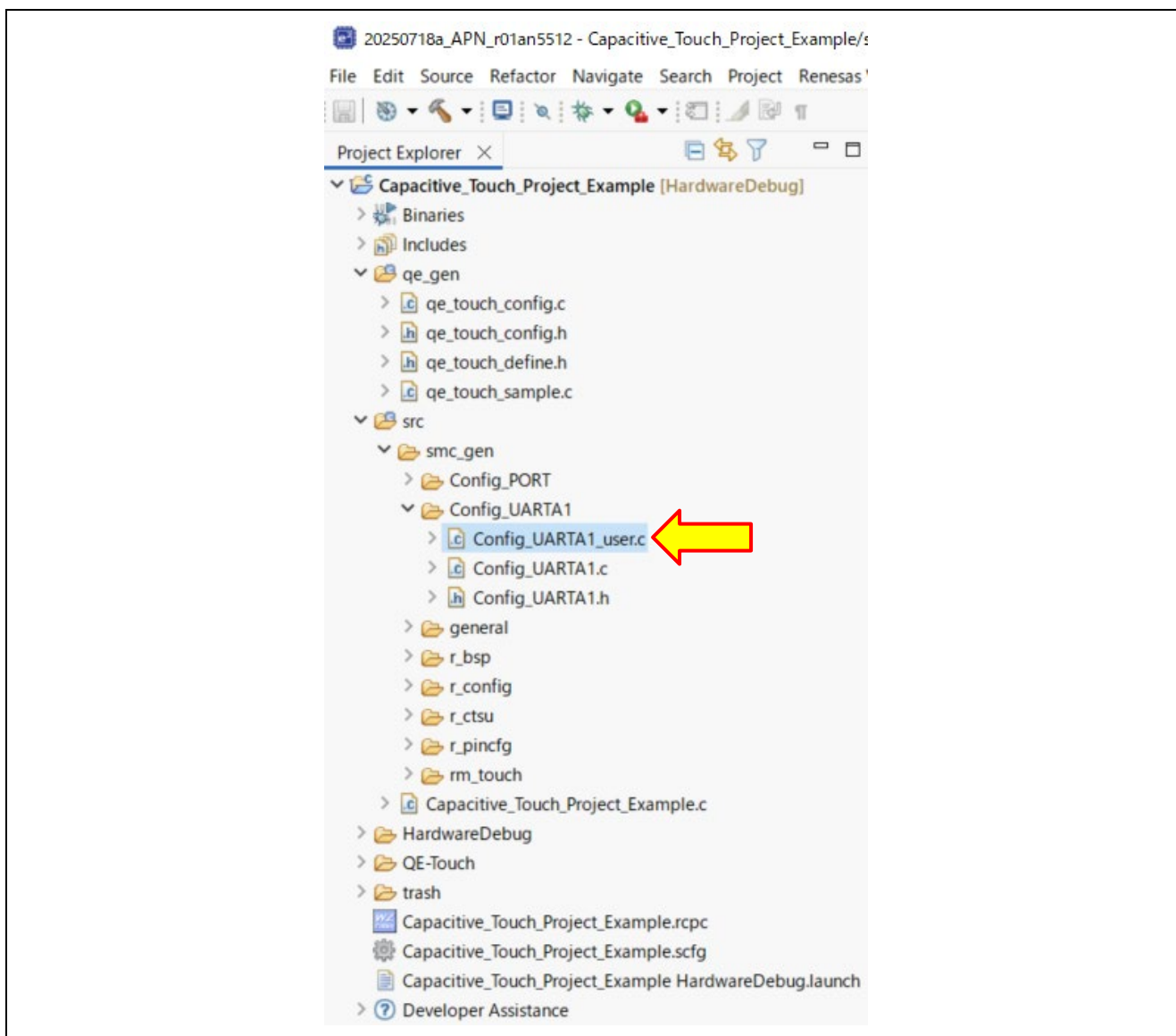


Figure 12-1 Select Config\_UARTA1\_user.c



2. Add the code “**extern void touch\_uart\_callback(uint16\_t event);**” in the red frame to **Config\_UARTA1\_user.c** file as shown in the image below.

```

Config_UARTA1_user.c
10 20 30 40 50 60
43 /*****
44 Global variables and functions↓
45 *****/
46 extern volatile uint16_t g_uartal_rx_total_num;↓
47 extern volatile uint8_t * gp_uartal_rx_address;↓
48 extern volatile uint16_t g_uartal_rx_num;↓
49 extern volatile uint8_t * gp_uartal_tx_address;↓
50 extern volatile uint16_t g_uartal_tx_count;↓
51 /* Start user code for global. Do not edit comment generated here */↓
52 extern void touch_uart_callback(uint16_t event);
53 /* End user code. Do not edit comment generated here */↓
  
```

Figure 12-2 Config\_UARTA1\_user.c (1/3)

3. Add the code “**touch\_uart\_callback(0);**” in the red frame to **Config\_UARTA1\_user.c** file as shown in the image below.

```

Config_UARTA1_user.c
10 20 30 40 50 60 70 80 90
67 /*****
68 * Function Name: r_Config_UARTA1_callback_sendend↓
69 * Description : This function is a callback function when UARTA1 finishes transmission.↓
70 * Arguments : None↓
71 * Return Value : None↓
72 *****/
H3 static void r_Config_UARTA1_callback_sendend(void)↓
74 {↓
75 /* Start user code for r_Config_UARTA1_callback_sendend. Do not edit comment generated here */↓
76 touch_uart_callback(0);
77 /* End user code. Do not edit comment generated here */↓
78 }↓
  
```


Figure 12-3 Config\_UARTA1\_user.c (2/3)

4. Add the code “**touch\_uart\_callback(1);**” in the red frame to **Config\_UARTA1\_user.c** file as shown in the image below.

```

Config_UARTA1_user.c
10 20 30 40 50 60 70 80 90 100
80 /*****
81 * Function Name: r_Config_UARTA1_callback_receiveend↓
82 * Description : This function is a callback function when UARTA1 finishes reception.↓
83 * Arguments : None↓
84 * Return Value : None↓
85 *****/
H3 static void r_Config_UARTA1_callback_receiveend(void)↓
87 {↓
88 /* Start user code for r_Config_UARTA1_callback_receiveend. Do not edit comment generated here */↓
89 touch_uart_callback(1);
90 /* End user code. Do not edit comment generated here */↓
91 }↓
  
```

Figure 12-4 Config\_UARTA1\_user.c (3/3)

- Click the hammer icon  on the top left in e<sup>2</sup> studio to build the project.  
Confirm that there is no error displayed in the “Console” window.

### 13. Monitoring Touch Performance using e<sup>2</sup> studio Expressions Window and QE for Capacitive Touch

- Start a Debug session by clicking the Bug icon  in the upper left-hand corner of e<sup>2</sup> studio. A Debug session will commence.

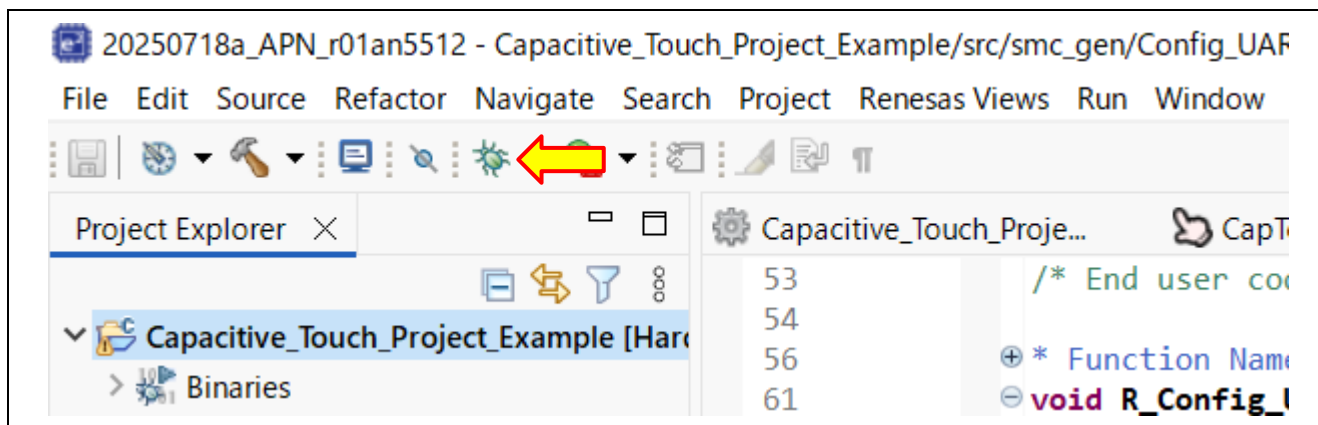


Figure 13-1 Debug icon Selection

- The debugger will stop at the **main()** function call.
- Open the declaration of **qe\_touch\_main()** function.

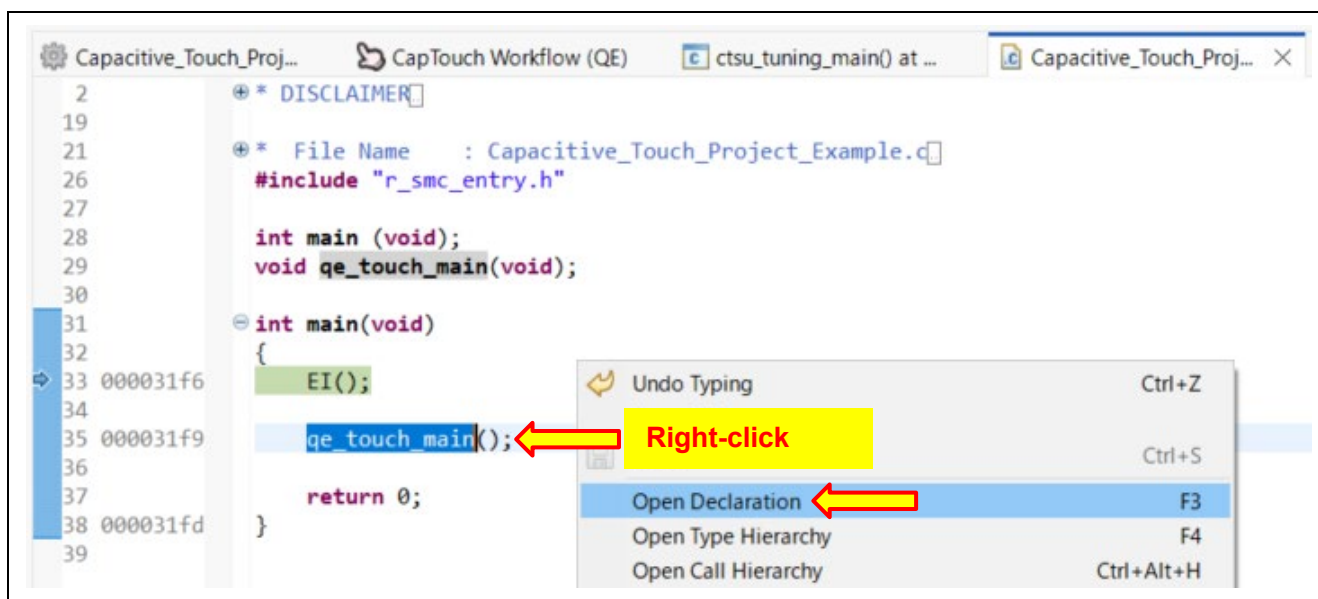


Figure 13-2 “Open Declaration” Selection



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.

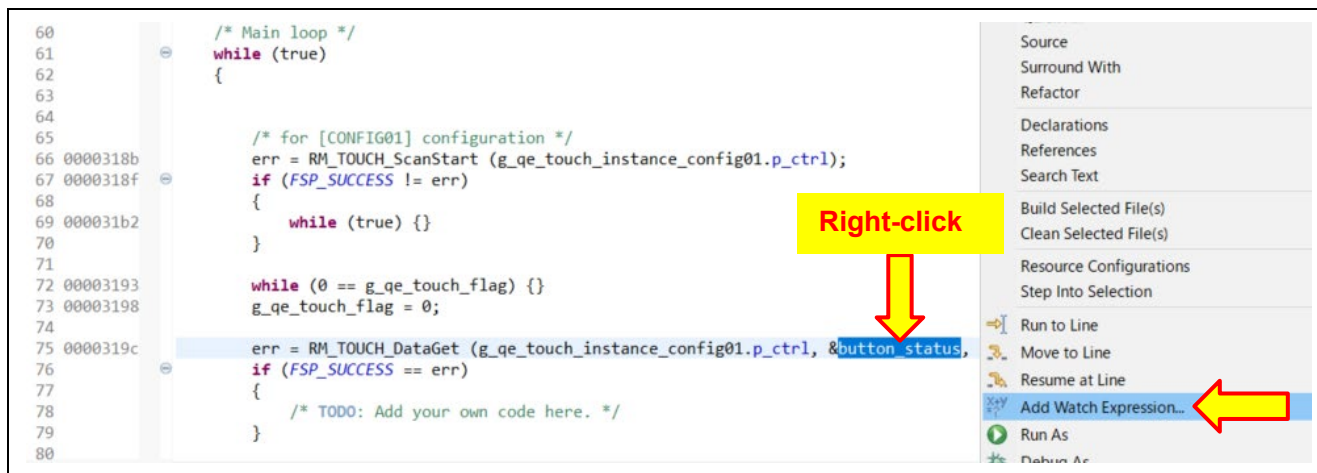


Figure 13-3 “Add Watch Expression” Selection

5. Enable Real-time Refresh for variables added to the “Expressions” window.

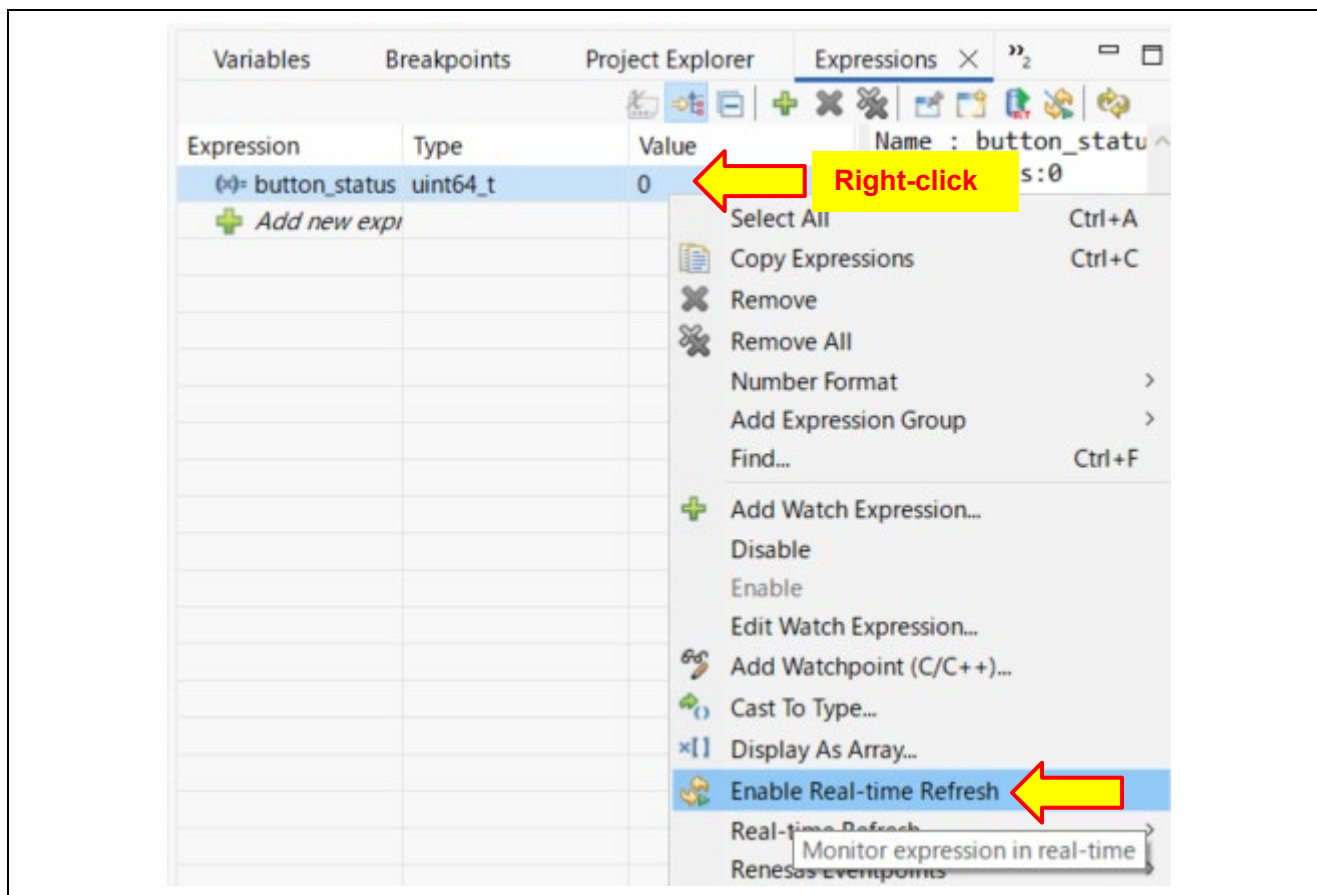


Figure 13-4 “Real-time Refresh” Selection

6. Check that **“Real-time Refresh”** for variable is enabled.

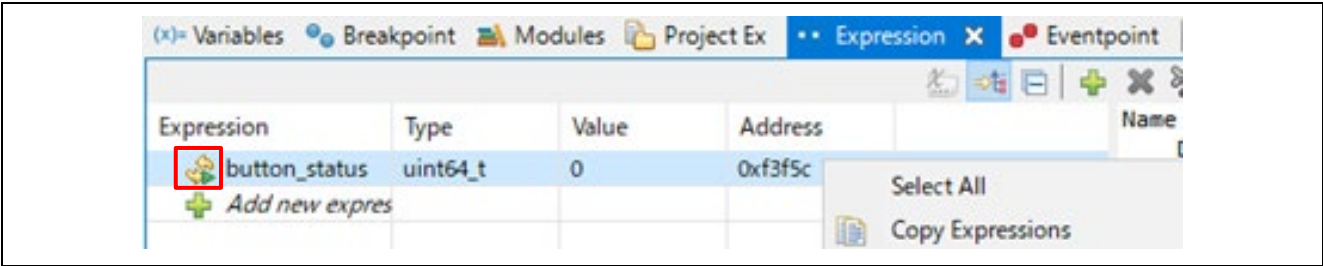


Figure 13-5 Check the enabled setting for “Real-time Refresh”

7. Click the **“Resume”** button located approximately of the e2 studio menu bar to continue code execution.

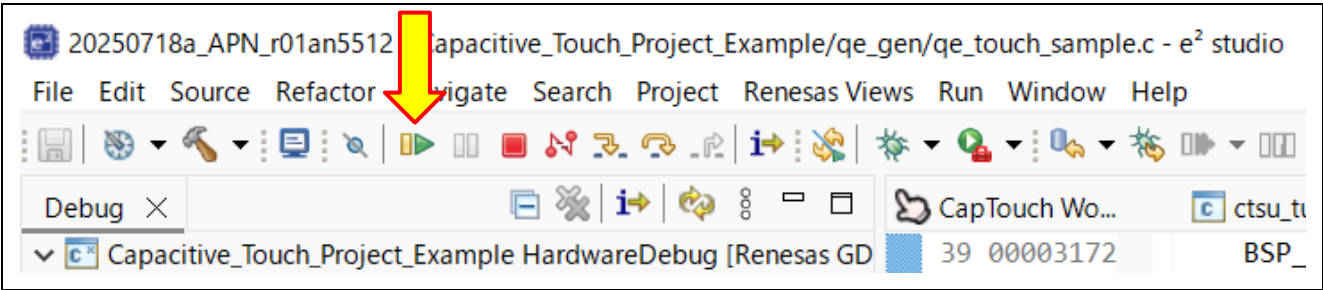


Figure 13-6 “Resume” button Selection

8. Press **TS05** on the board, which was configured as **Button01** 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.

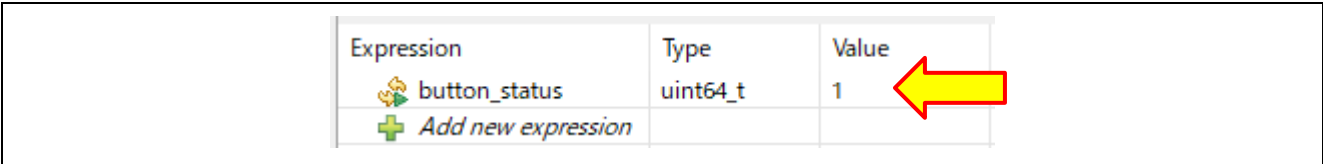


Figure 13-7 Check “button\_status”

9. From [CapTouch Workflow (QE)] pane, click “**Select a Connection Method**” under “Monitoring” and select [Emulator].

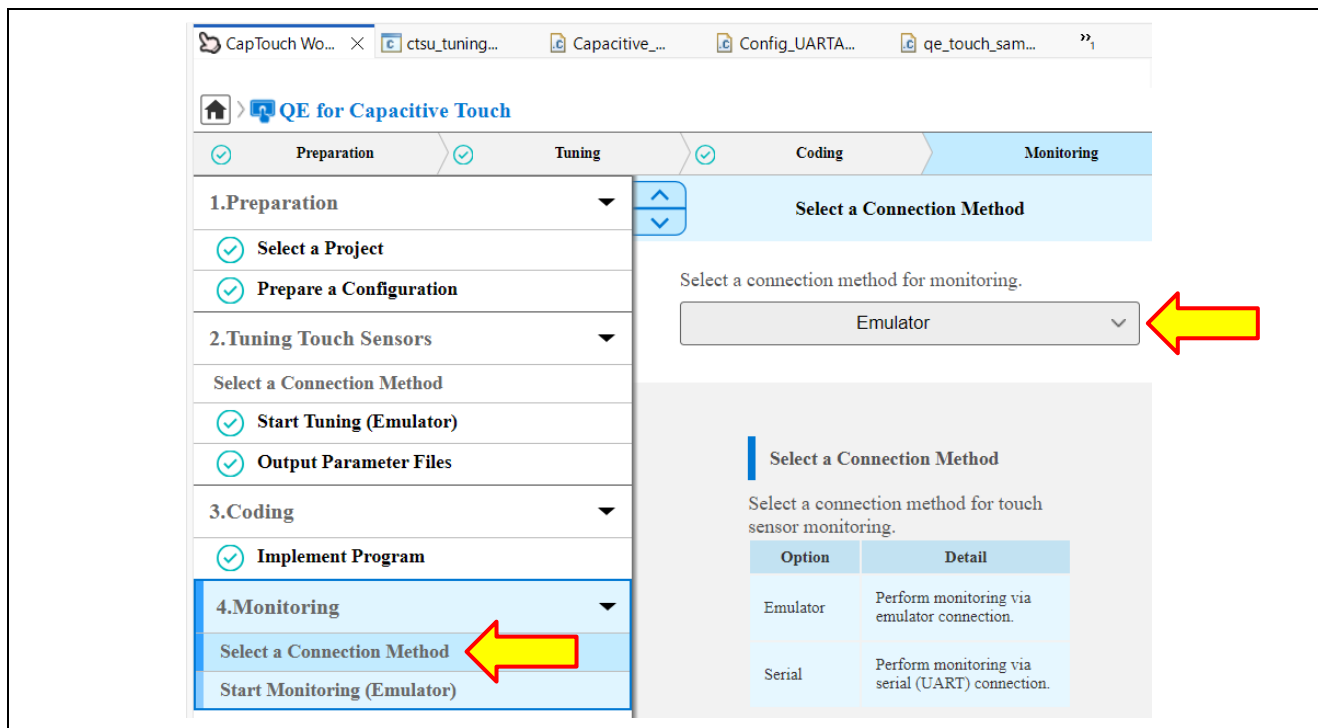


Figure 13-8 Select a Connection Method (Emulator Connection)

10. Click “**Start Monitoring (Emulator)**” and select “**Show Views**”.

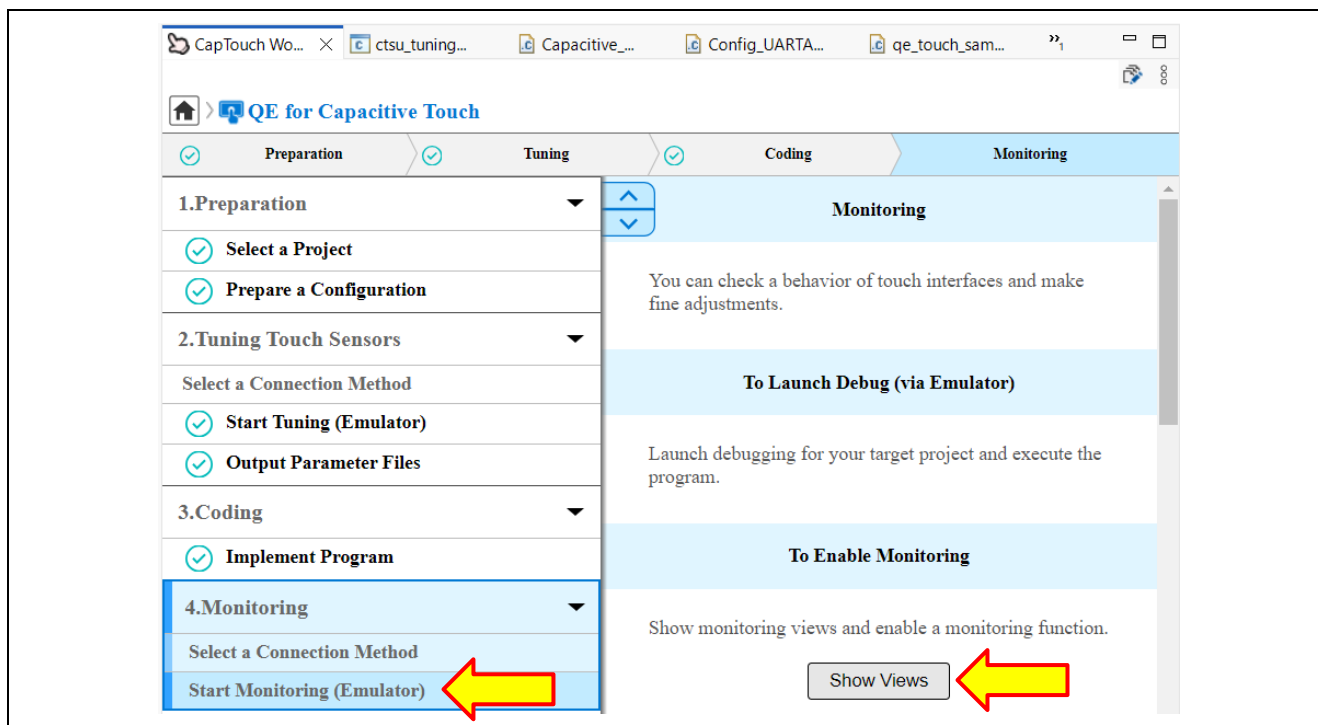


Figure 13-9 “Show Views” Selection (Emulator Connection)

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

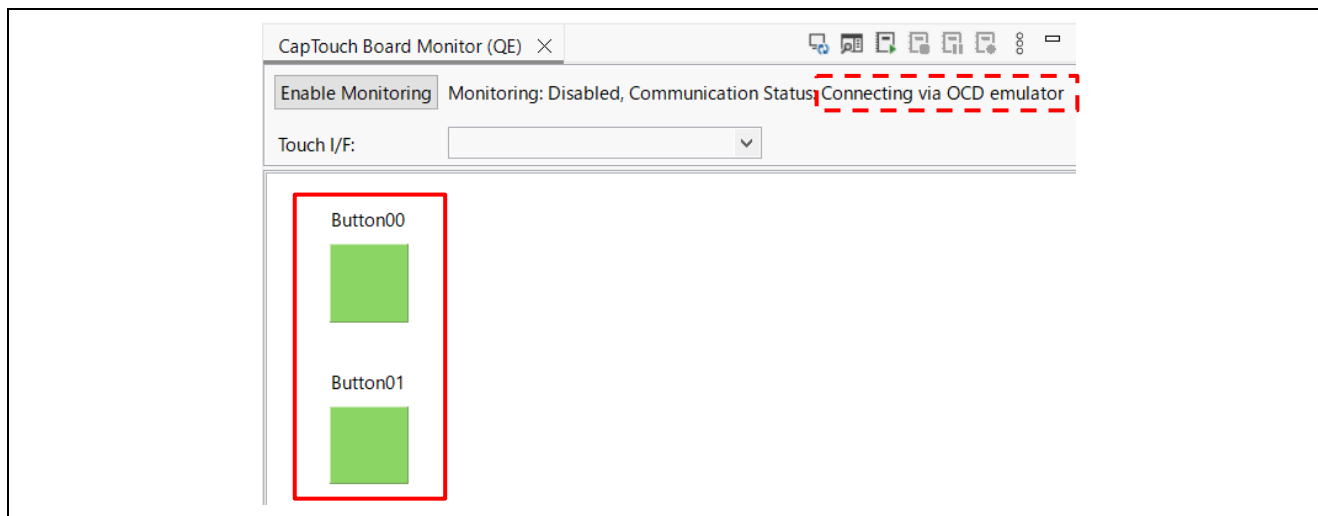


Figure 13-10 “CapTouch Board Monitor (QE)” (Emulator Connection)

12. Click the [Enable Monitoring] button. The dialog text will change to “**Monitoring: Enabled**”.

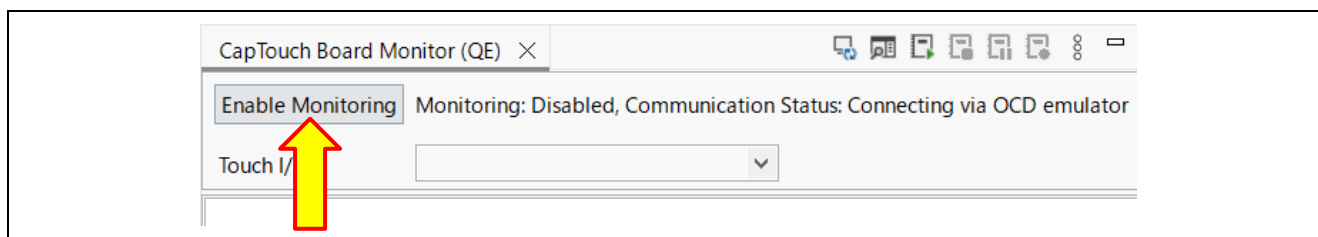


Figure 13-11 [Enable Monitoring] Selection (Emulator Connection)

13. Touch the button **TS06** on the target board. The “**CapTouch Board Monitor (QE)**” will show a touch with a finger image on the button like the below image.

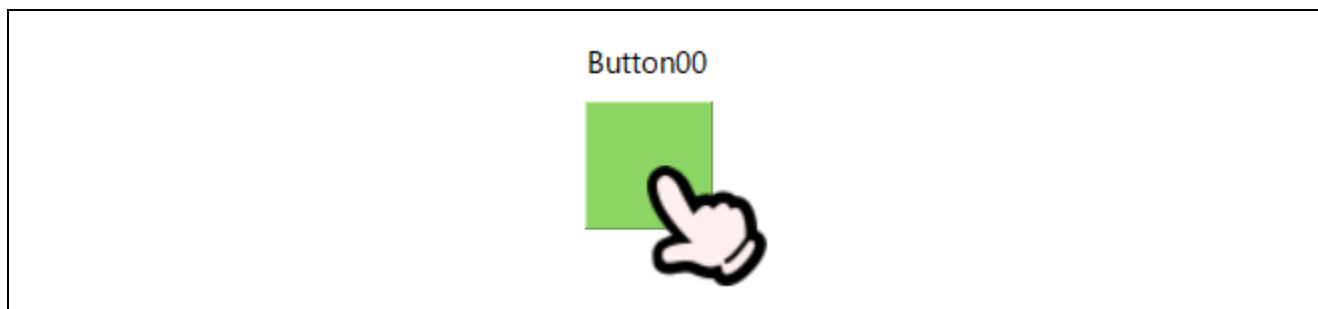


Figure 13-12 Touch Status Display (1)

14. To see a graphical representation of the 'touch counts' from the board, use the [CapTouch Status Chart (QE)].

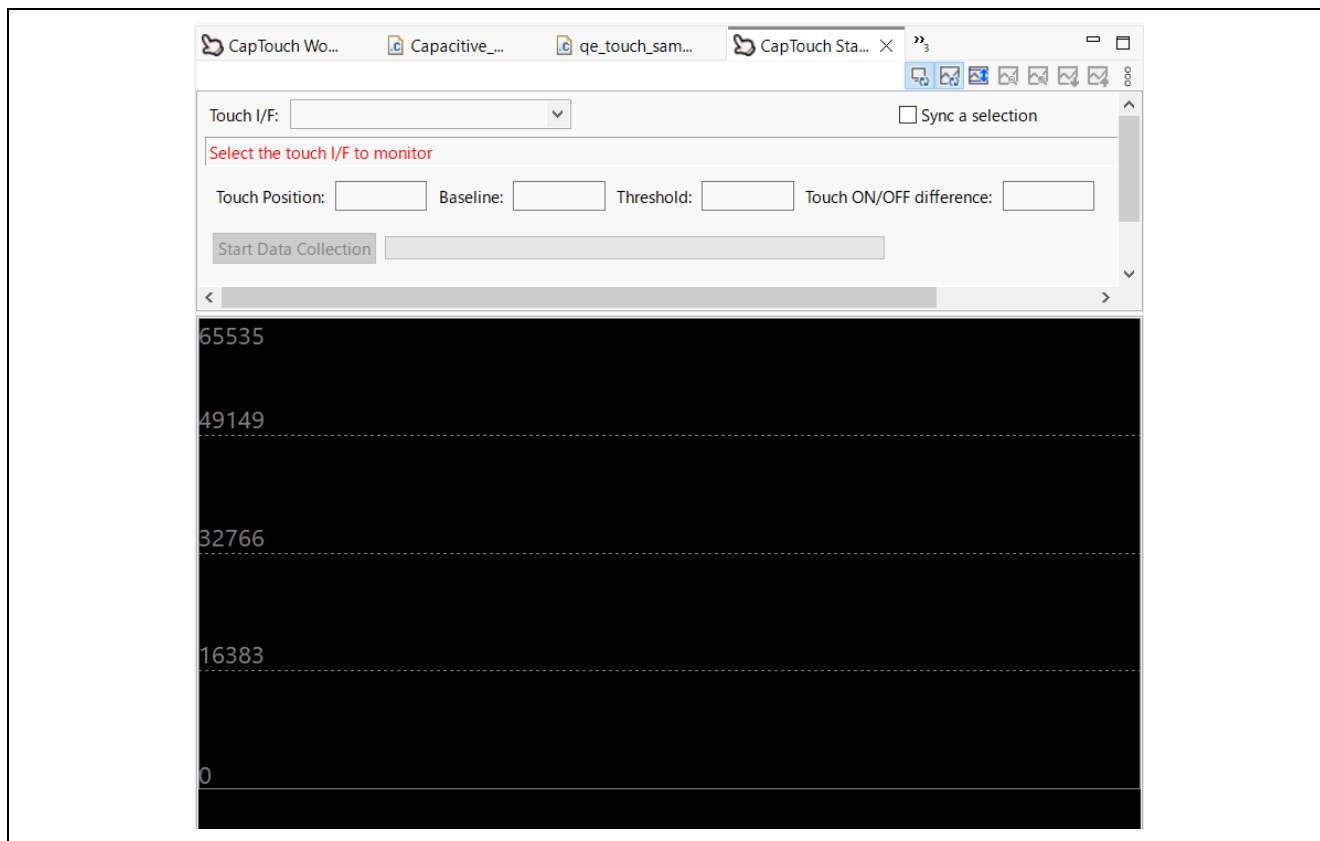


Figure 13-13 “CapTouch Status Chart (QE)” (1)

15. Using the pulldown, select “**Button00 @ config01**”.

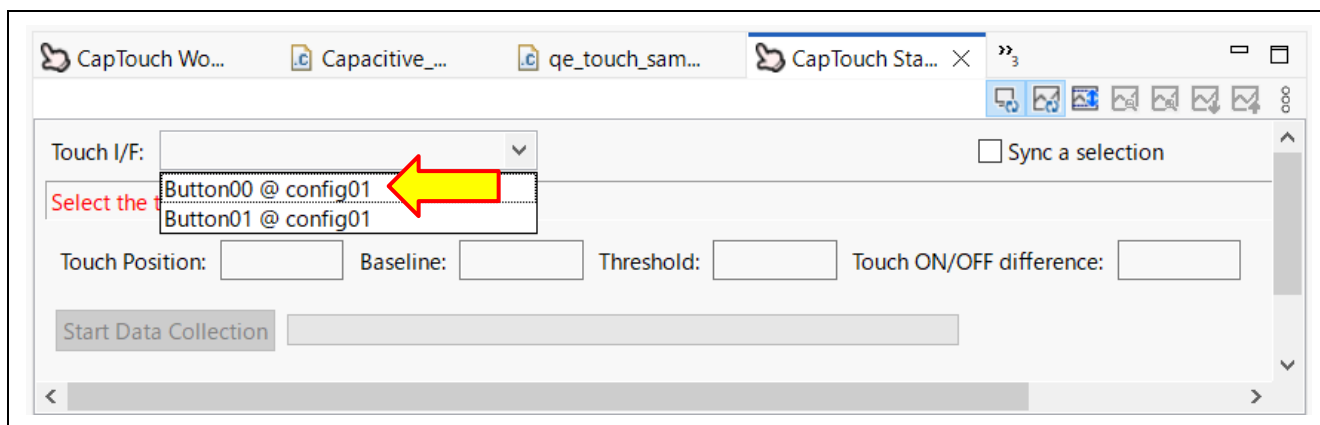


Figure 13-14 Touch I/F Selection

16. The graph will begin to display running values. Touch **TS06** 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.

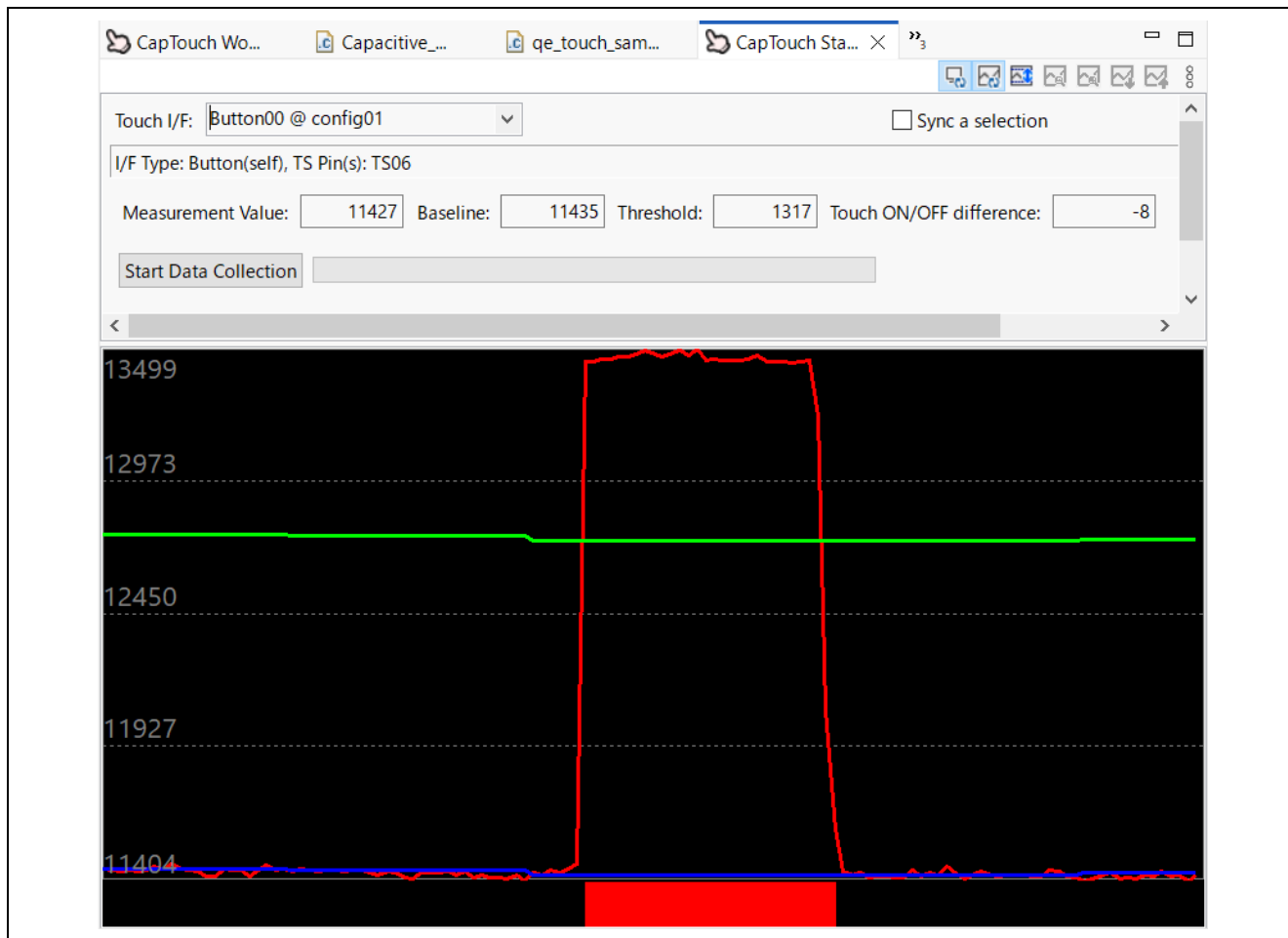
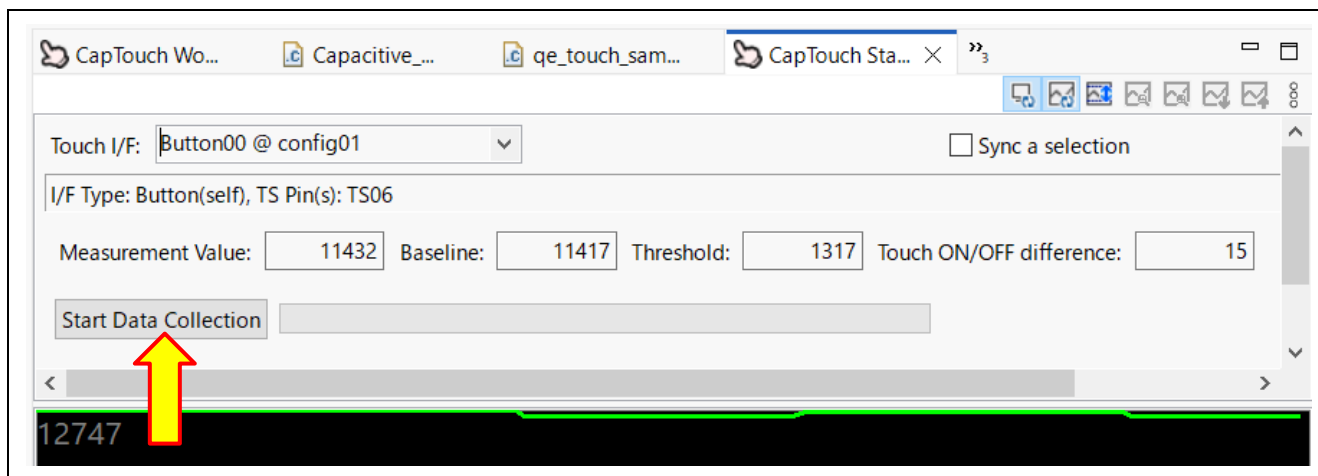


Figure 13-15 Graph of the Touch Counting Value (1)

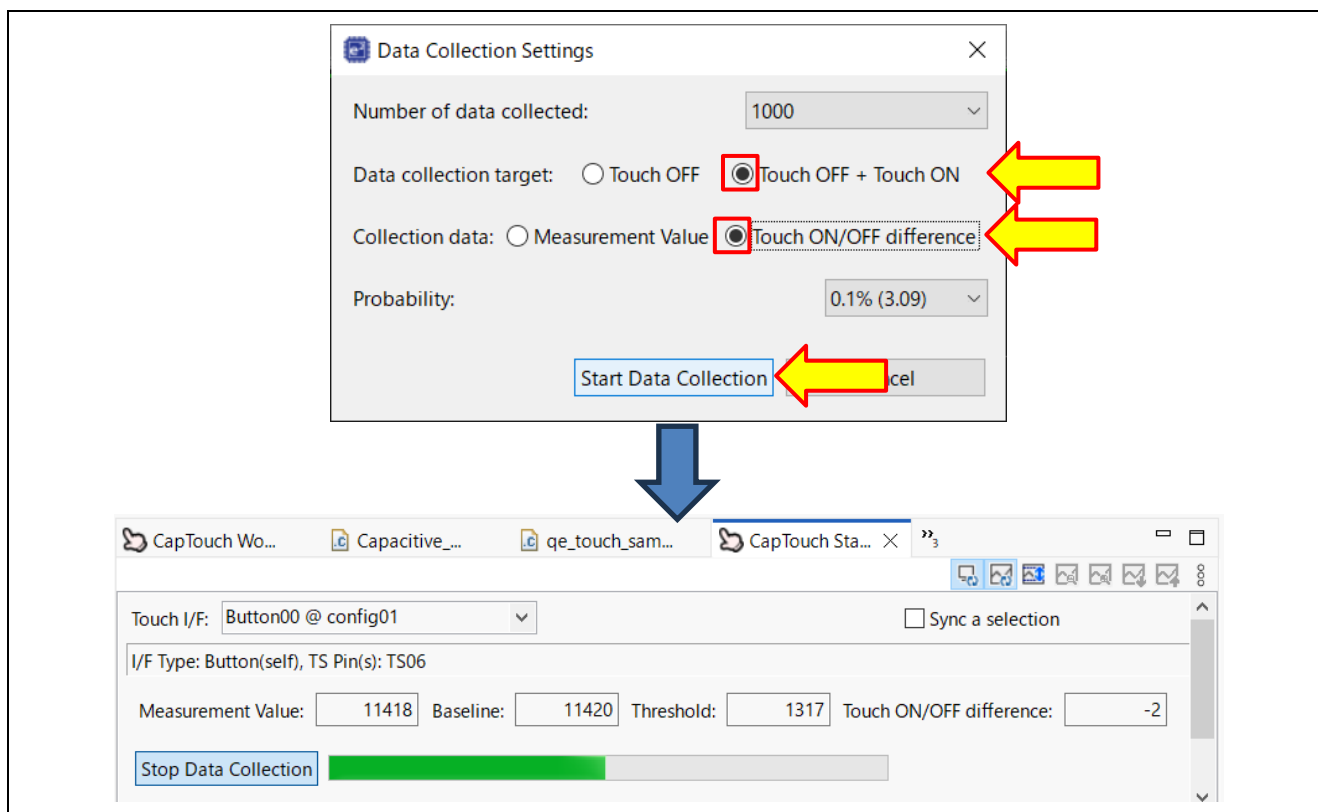
**Note: Measure the Signal-to-Noise Ratio (SNR) values as required.**  
**When measuring the SNR value, perform steps 17 to 20.**

17. Click on [Start Data Collection] on the [CapTouch Status Chart (QE)].



**Figure 13-16 [Start Data Collection] button**

18. Make settings for data collection as shown in the figure and click on [Start Data Collection]. Do not touch the sensor while collection of data in the touch-off state is in progress. The green bar indicates progress in data collection. When the green bar reaches the right end, the ratio of data collection is 100% so data collection in the touch-off state is completed.



**Figure 13-17 Starting Data Collection (Touch-off State)**

19. Collect data in the touch-on state in the same way. Make sure that one of your fingers is touching the sensor then click on **[Start Data Collection]**. When the green bar reaches the right end, data collection in the touch-on state is completed.

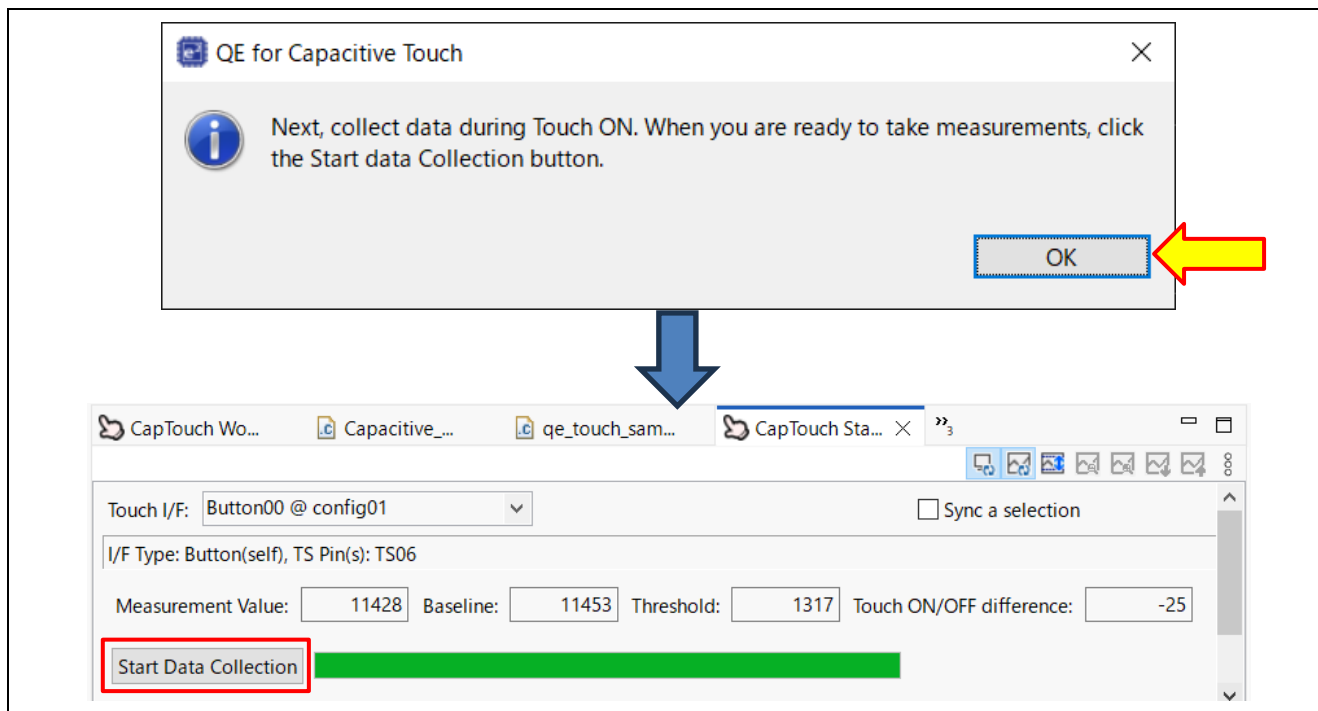


Figure 13-18 Starting Data Collection (Touch-on State)

20. After data collection is completed, the SNR value will be displayed.

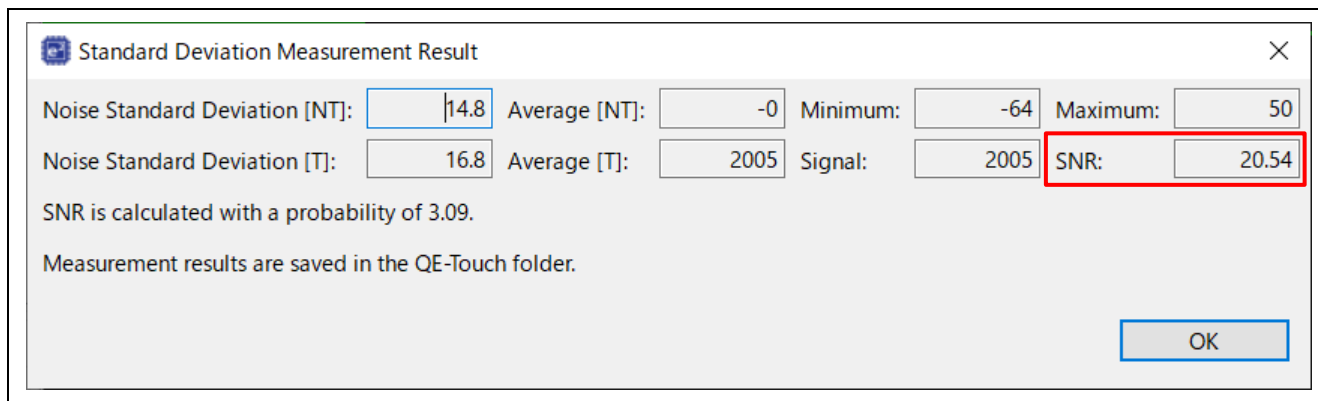


Figure 13-19 SNR Value



21. Display graphs of touch counting values for multiple touch sensors in the multi-status chart.

Select the touch sensors for which values are to be displayed on the [CapTouch Multi Status Chart (QE)] in the lower left part of the QE window.

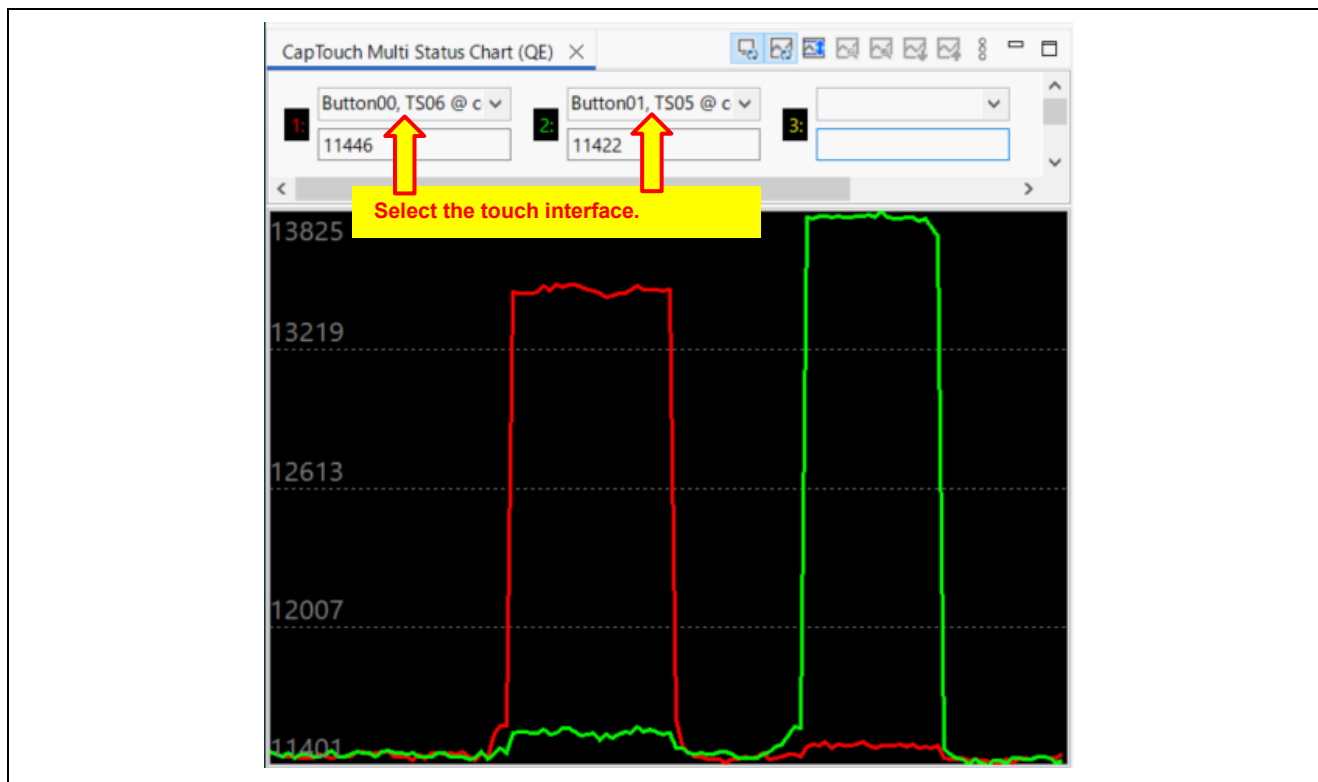


Figure 13-20 CapTouch Multi Status Chart (QE)

22. If you will check and adjust the touch parameters, use the “CapTouch Parameters (QE)”.

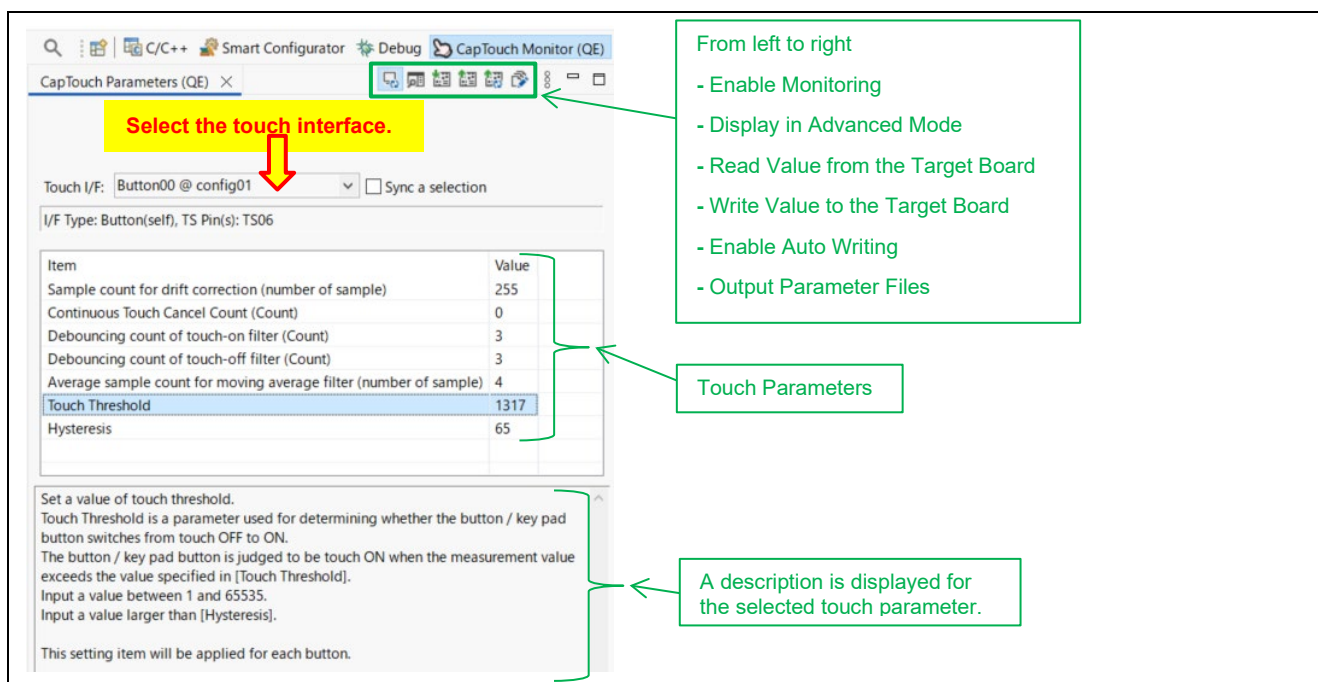


Figure 13-21 Adjusting Parameters

23. If you will finish monitoring, click the **[Enable Monitoring]** button. The dialog text will change to “**Monitoring: Disabled**”.

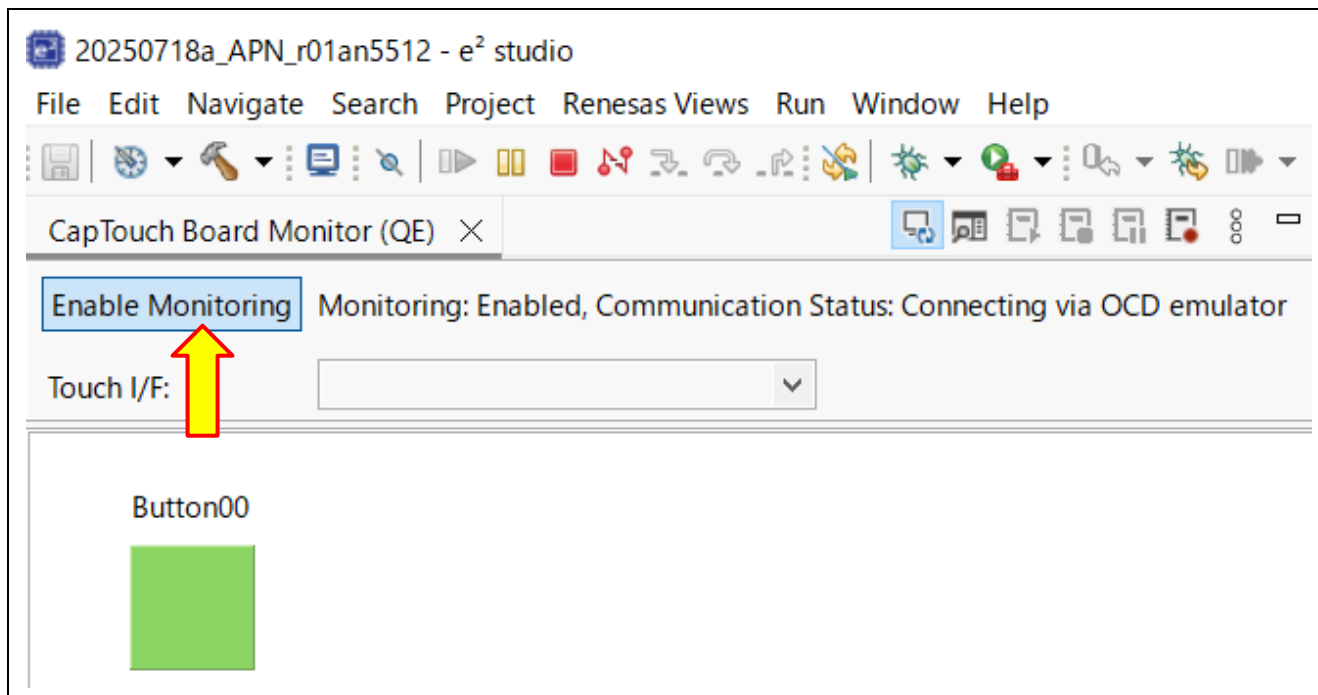


Figure 13-22 Disable monitoring (1)

24. If you will finish the debug session, click the “**Terminate**” icon to end the debug session.

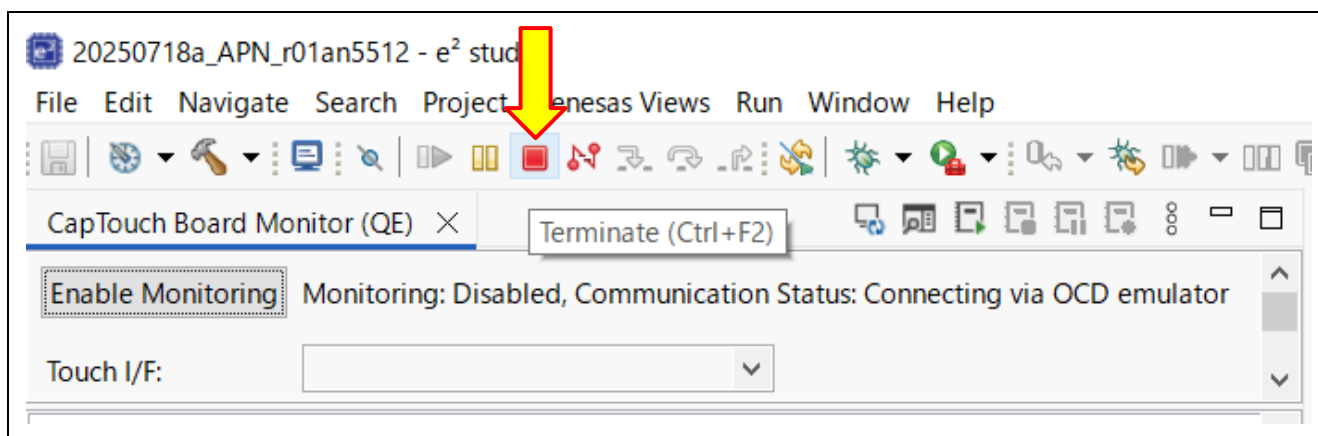


Figure 13-23 “Terminate” icon

## 14. [Additional function] Setting the serial communication monitor using UART (3/3)

**Note:** Monitoring touch performance for touch applications can be confirmed by communication via the OCD (On-Chip Debugging) emulator. However, RL78 family case, monitoring performance is limited by the OCD function of the RL78 family.

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 (This is the recommended setting.).

Chapters 7, 12 and 14 (including this chapter) shown below describe setting the serial communication monitor using UART.

- “7. [Additional function] Setting the serial communication monitor using UART (1/3)”
- “12. [Additional function] Setting the serial communication monitor using UART (2/3)”
- “14. [Additional function] Setting the serial communication monitor using UART (3/3)”

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)] pane. Ensure the project folder (Capacitive\_Touch\_Project\_Example) and the tificfg file (Capacitive\_Touch\_Project\_Example.tifcfg) are set as shown below.

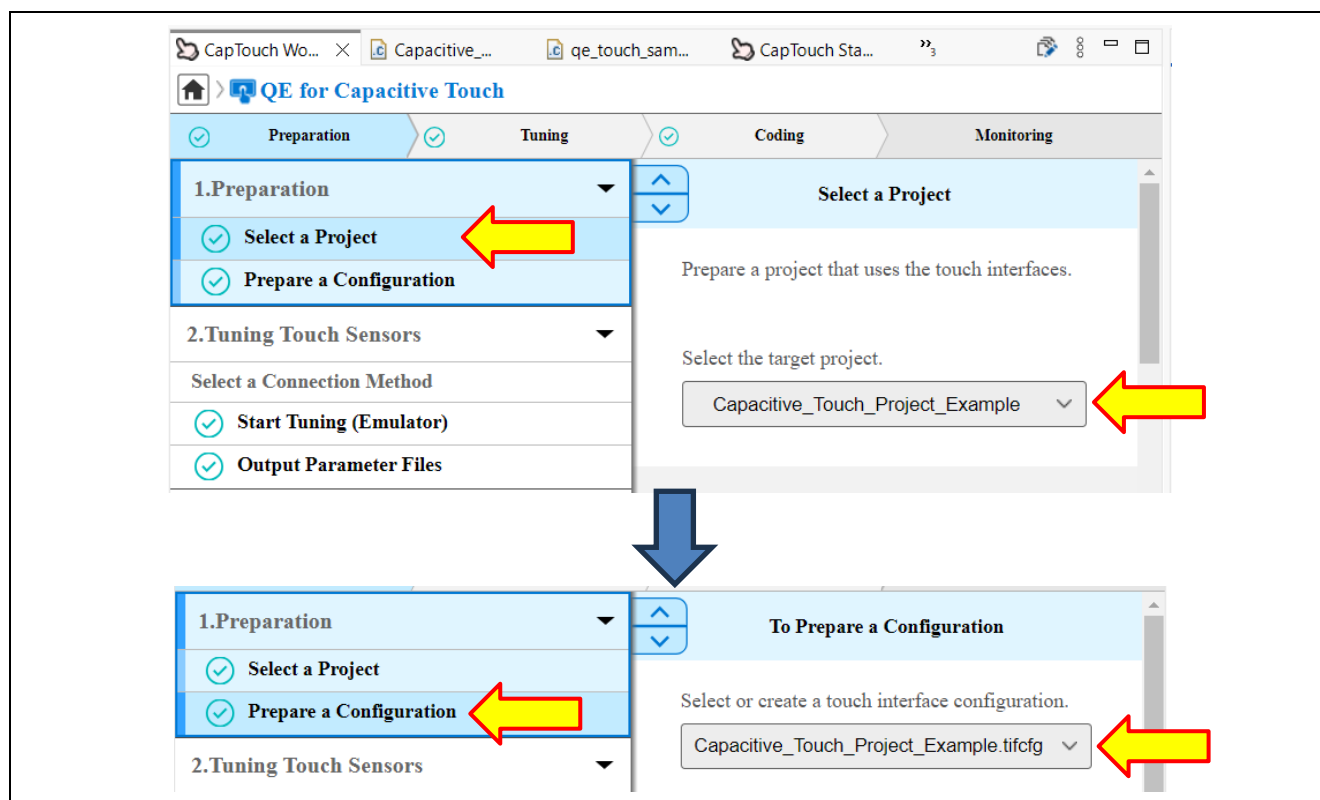


Figure 14-1 Selecting the project and touch interface configuration

4. Click “**Select a Connection Method**” under “Monitoring” and select [Serial].

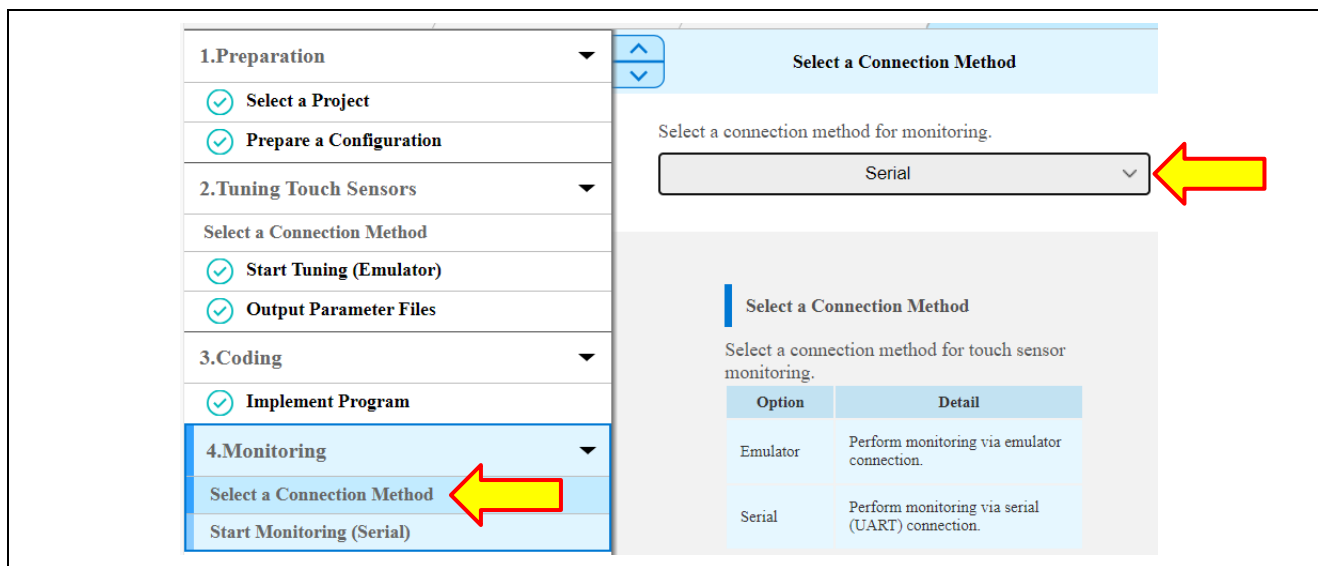


Figure 14-2 Select a Connection Method (Serial Connection)

5. Click “**Start Monitoring (Serial)**” under “Monitoring,” set the “Baud rate” to “**153600**” (bps), and click the [Connect] button.

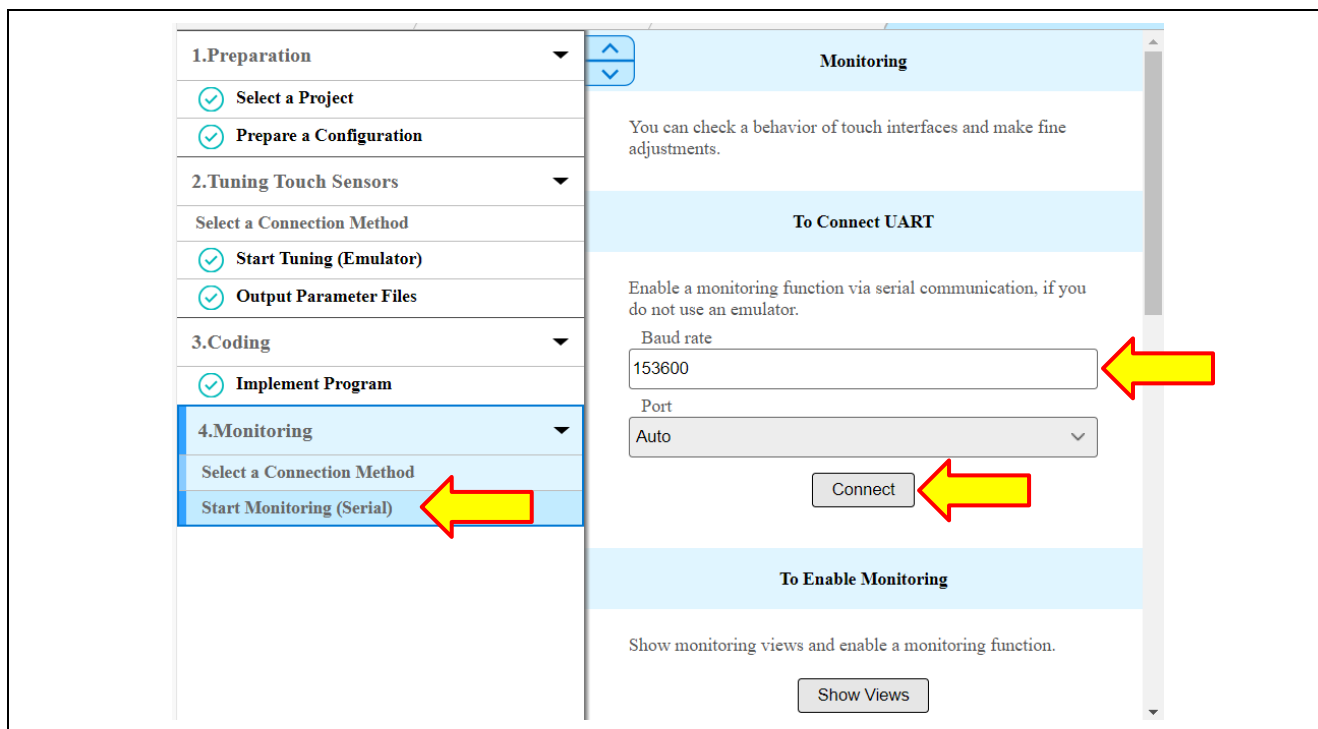


Figure 14-3 Serial Connection

**Note:** For the baud rate (bps) setting value, use the baud rate (bps) set in step 4 of chapter "7. [Additional function] Setting the serial communication monitor using UART (1/3)".

6. When serial connection is executed, the following message will appear in the console window. Then select **[Show Views]**.

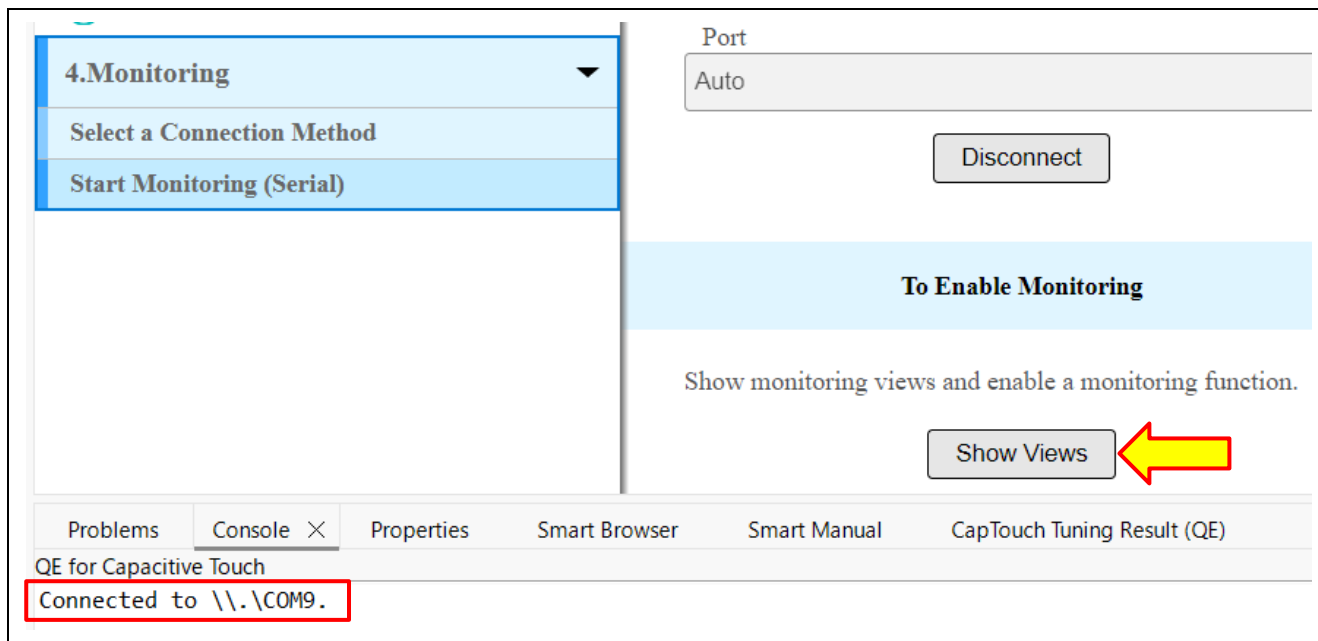


Figure 14-4 “Show Views” Selection (Serial Connection)

7. **[CapTouch Board Monitor (QE)]** pane will appear as shown below.

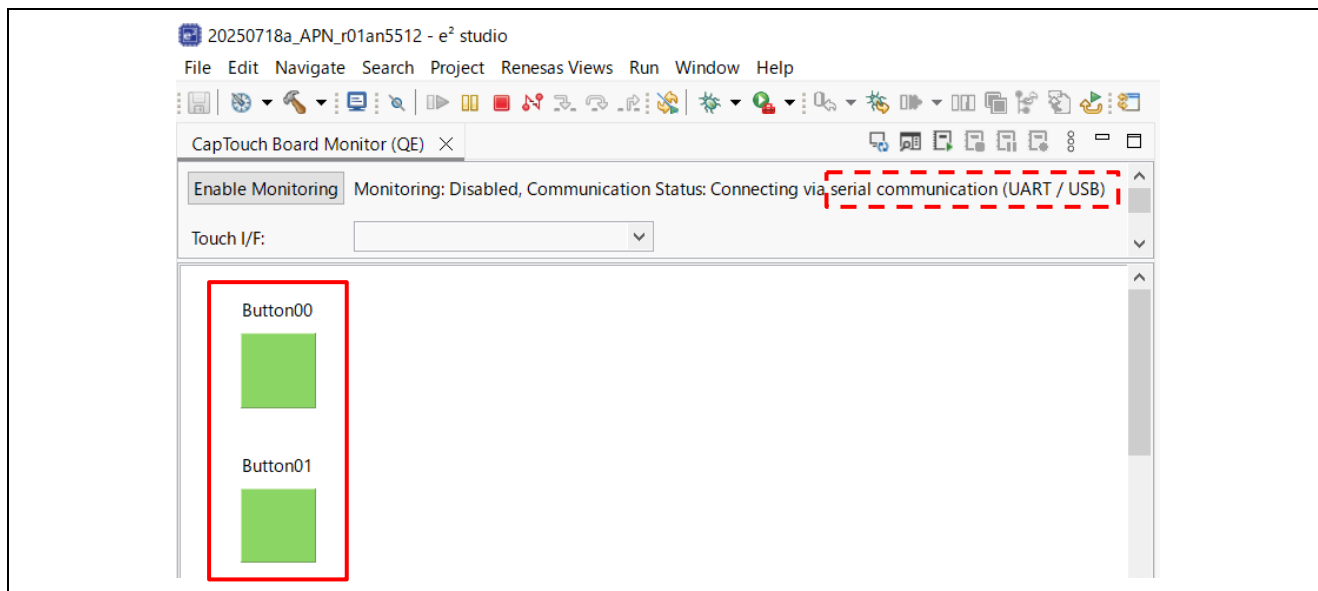


Figure 14-5 “CapTouch Board Monitor (QE)” (Serial Connection)

- Click the **[Enable Monitoring]** button. The dialog text will change to “**Monitoring: Enabled**”.

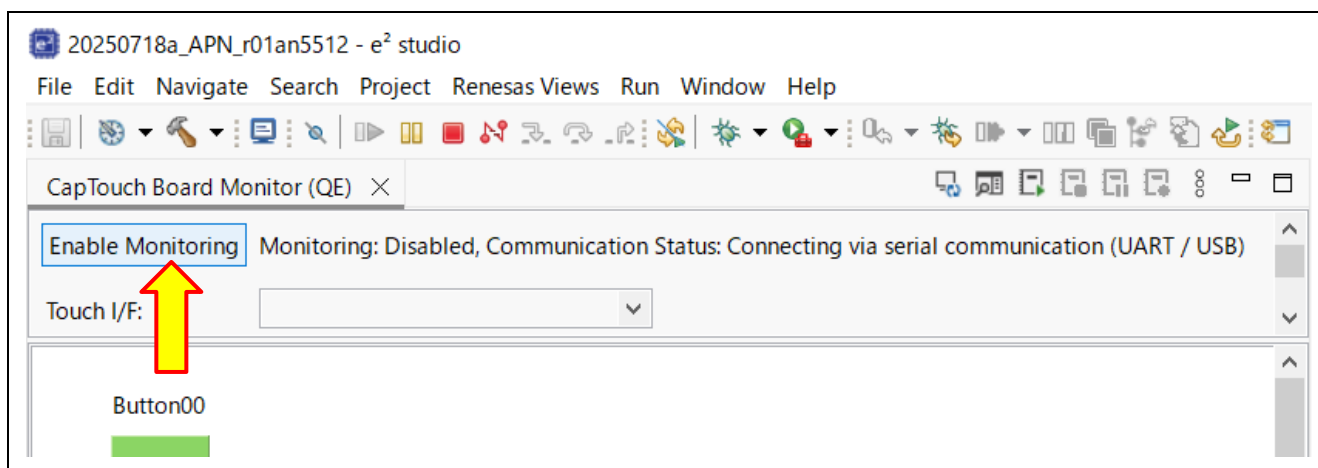


Figure 14-6 [Enable Monitoring] Selection (Serial Connection)

- Touch the button **TS06** on the target board. The **CapTouch Board Monitor (QE)** will show a touch with a finger image on the button like the below image.



Figure 14-7 Touch Status Display (2)

10. To see a graphical representation of the 'touch counts' from the board, use the **CapTouch Status Chart (QE)**. Using the pulldown, select **Button00 @ config01**.

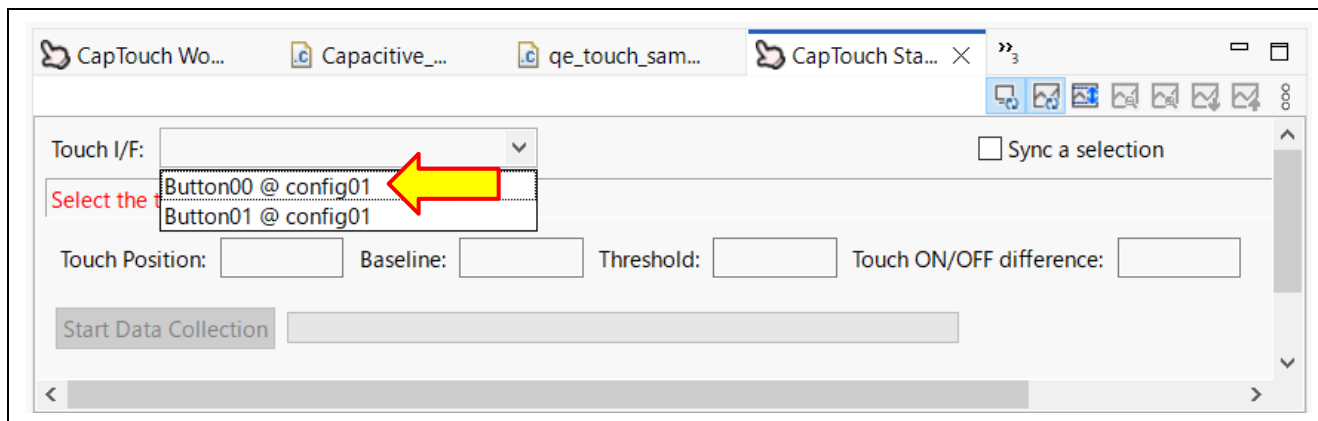


Figure 14-8 “CapTouch Status Chart (QE)” (2)

11. The graph will begin to display running values. Touch **TS06** on the board and you should see the touch count value show as a step change on the running graph.

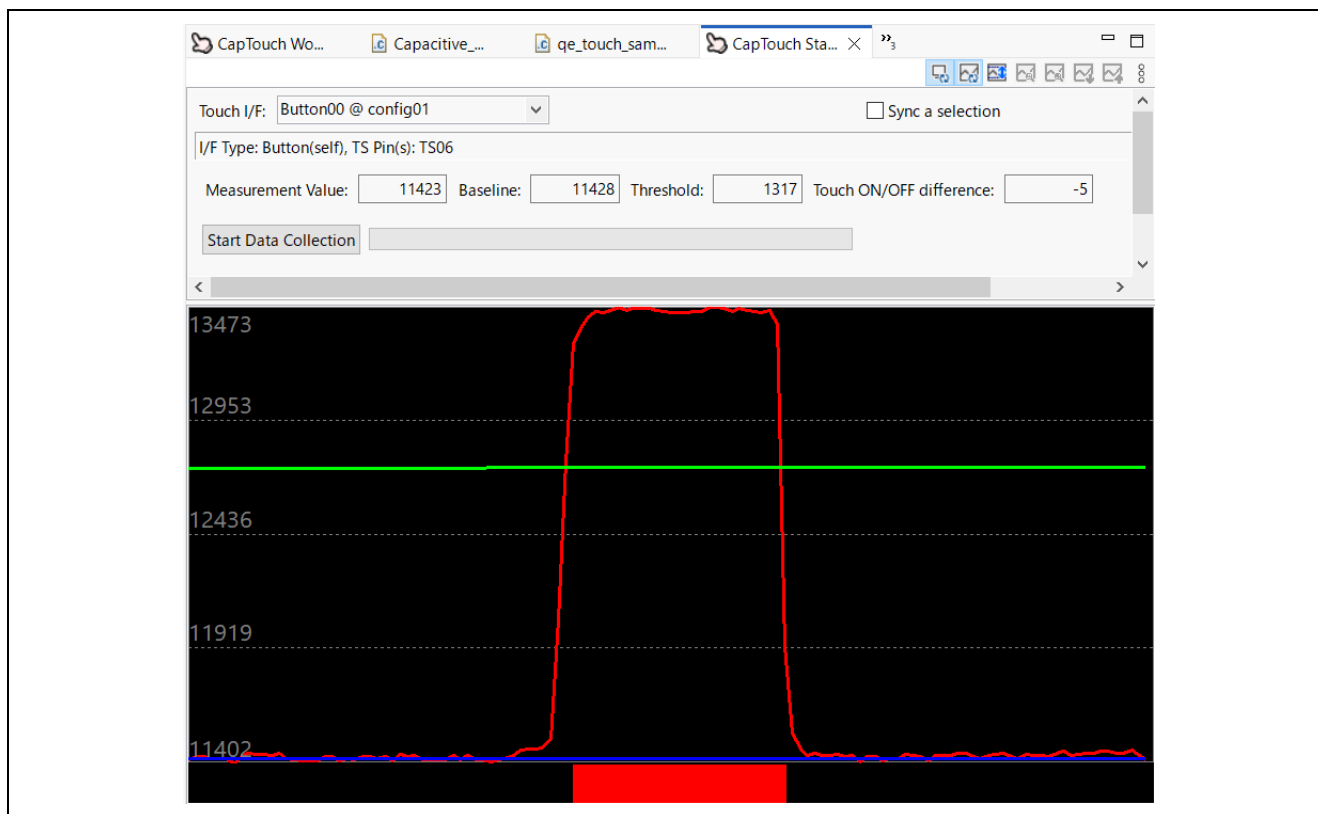


Figure 14-9 Graph of the Touch Counting Value (2)

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

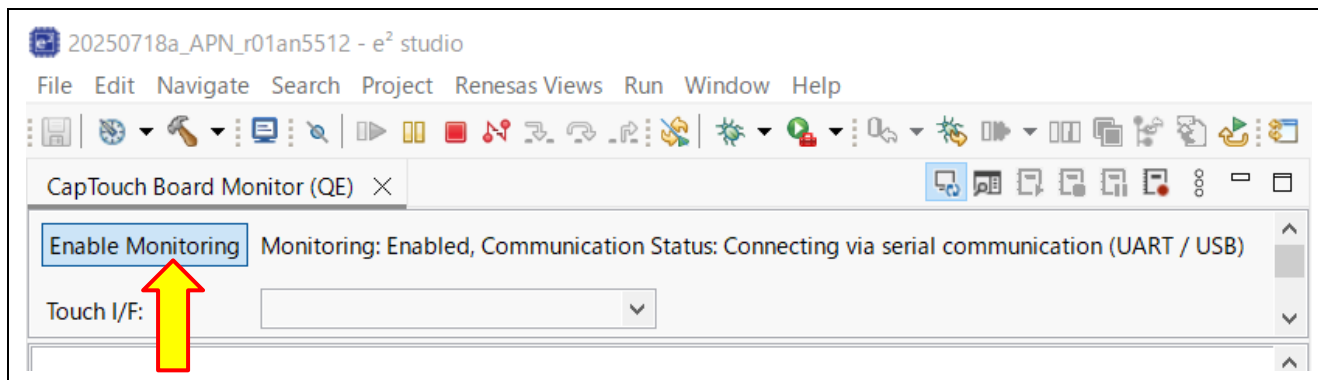


Figure 14-10 Disable monitoring (2)

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

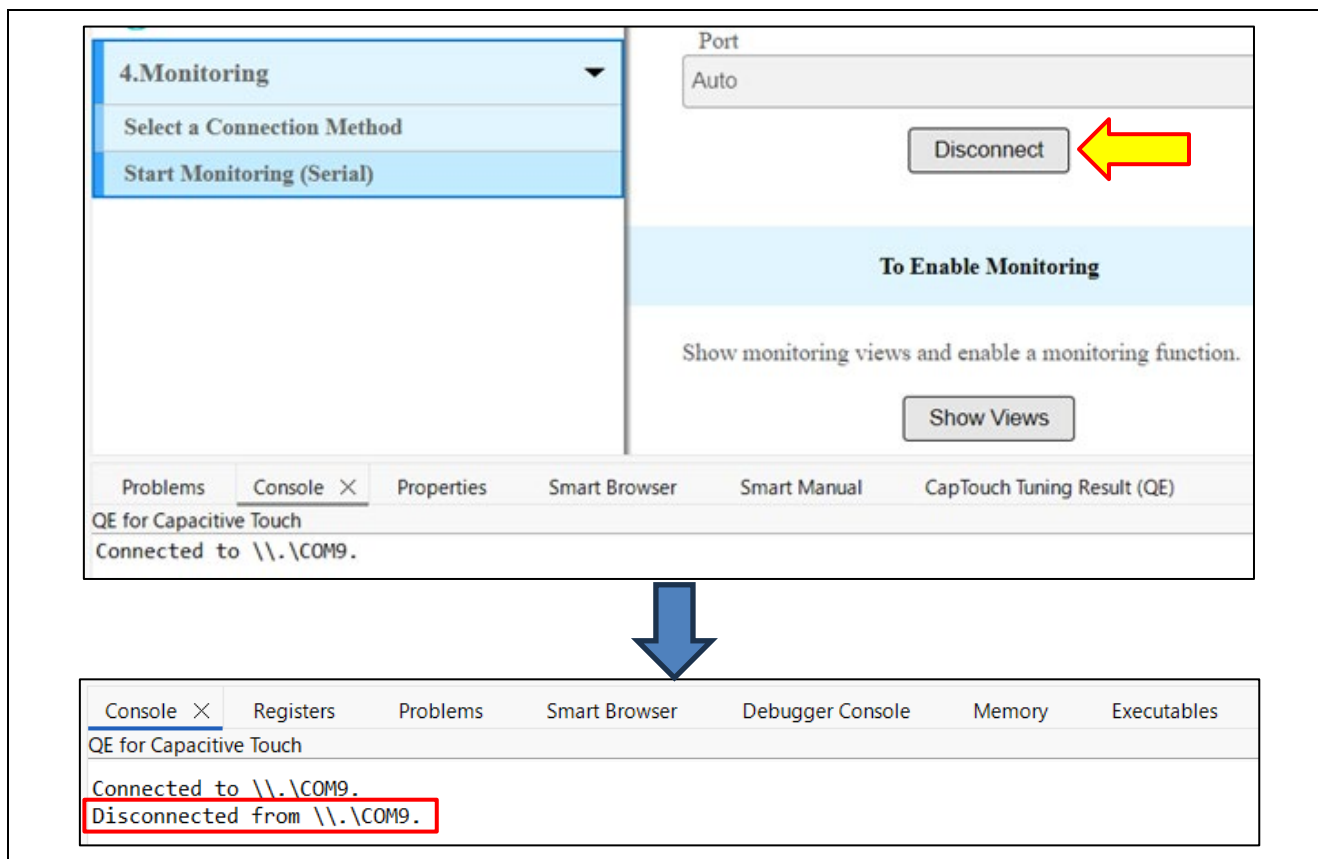


Figure 14-11 [Disconnect] button Selection (Serial Connection)



## 15. qe\_touch\_sample.c (Example of Using a Software Timer)

The sample program output from QE for Capacitive Touch is shown below.

```

/*****
 * Copyright (c) 2020 - 2025 Renesas Electronics Corporation and/or its affiliates
 *
 * SPDX-License-Identifier: BSD-3-Clause
 *****/
/*****
 * File Name      : qe_touch_sample.c
 * Description    : Main Program for RL78
 *****/
#include "qe_touch_config.h"

#define TOUCH_SCAN_INTERVAL_EXAMPLE (20 * 1000) /* microseconds */

void R_CTSU_PinSetInit(void);
void qe_touch_main(void);
void qe_touch_delay(uint16_t delay_us);

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;

    BSP_ENABLE_INTERRUPT();

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

    /* 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. */
    qe_touch_delay (TOUCH_SCAN_INTERVAL_EXAMPLE);
}

}

void qe_touch_delay(uint16_t delay_us)
{
    uint32_t i;
    uint32_t loops_required;
    uint16_t clock_mhz;

    clock_mhz = (uint16_t) (R_BSP_GetFclkFreqHz () / 1000000);
    if (0 == clock_mhz)
    {
        clock_mhz = 1;
    }

    loops_required = ((uint32_t) delay_us * (uint32_t) clock_mhz);
    loops_required /= 20;
    for (i = 0; i < loops_required; i++)
    {
        BSP_NOP();
    }
}
```

## 16. [Practical applications] Touch Measurement by Hardware Timer

This section describes an example of an implementation with the use of a hardware timer to generate the cycles of touch measurement. This example uses the interval timer function of the 32-bit interval timer in 8-bit counter mode. This example also provides a function for checking the touch sensor operation by turning an LED on the target board on or off according to the results of judging the state of touching of a sensor (a button). Specifically, LED1 is turned on when a finger touches touch sensor 1 (TS\_B1: Button00/TS06) or touch sensor 2 (TS\_B2: Button01/TS05) and the result of judgment becomes detection of the touch-on state.

Make the settings described in the following section in addition to the settings described in “6. From Starting Capacitive Touch Application Development to Adding Modules” to “15. qe\_touch\_sample.c (Example of Using a Software Timer)”.

### 16.1 Using the Smart Configurator to Make Settings (Hardware Timer)

1. Select the **[Clocks]** tab in the Smart Configurator view and set up the clock to be used for touch measurement cycles as shown below.

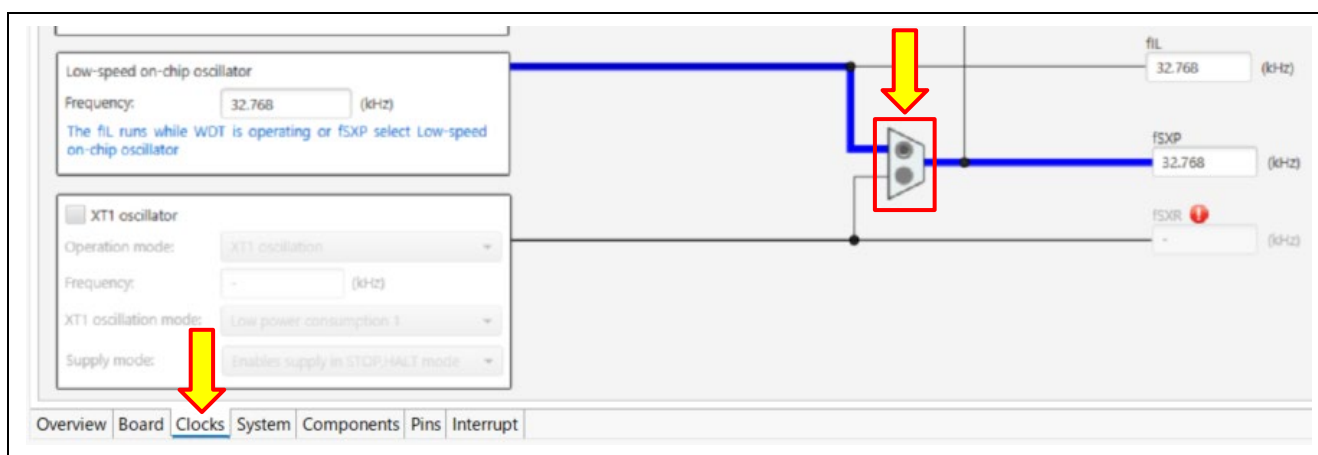


Figure 16-1 Setting the Clock (For Hardware Timer)

2. Select the “**Interval Timer**” module in the Software Component Selection and click [**Next**] at the bottom of the dialog box.

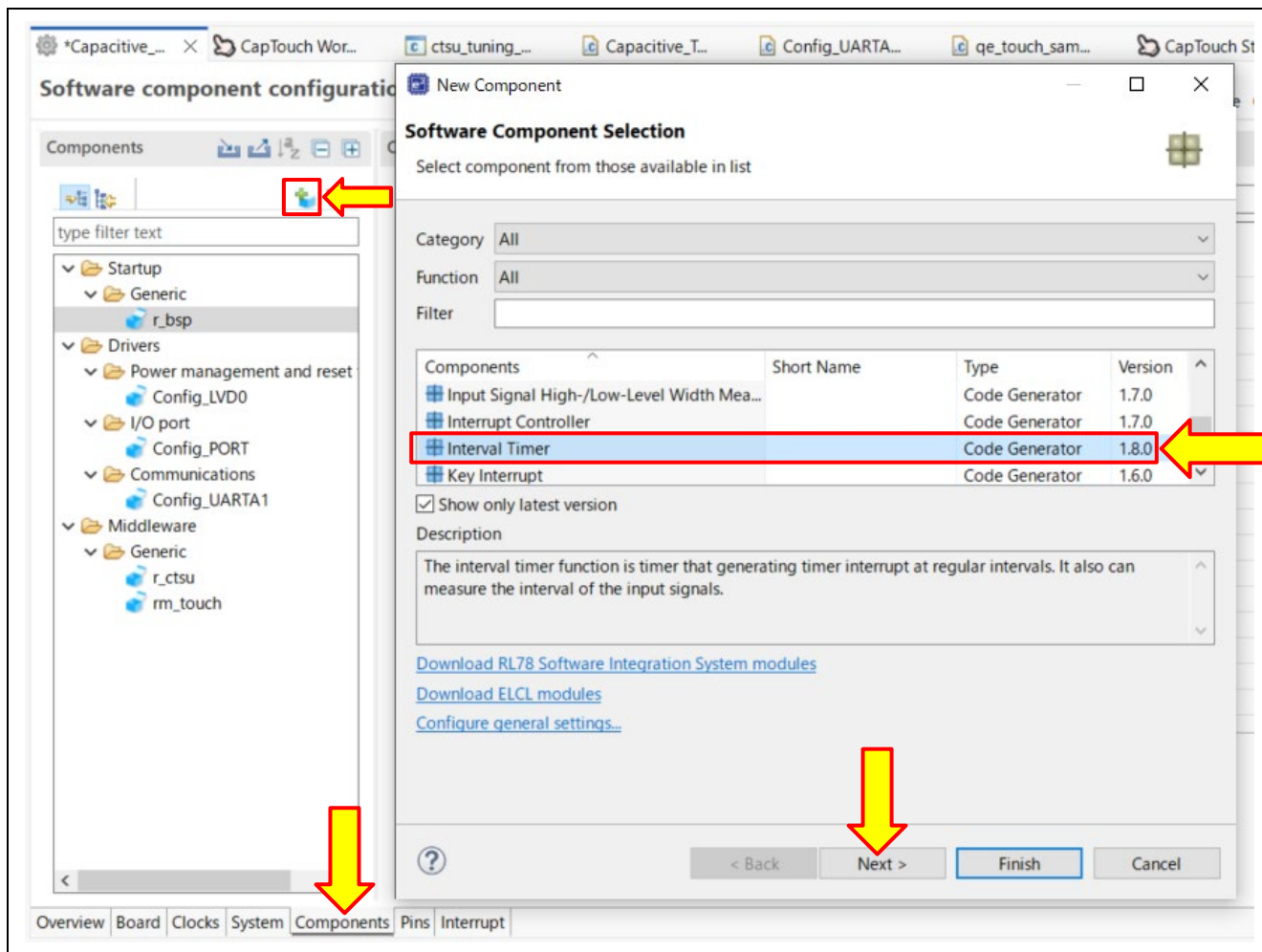
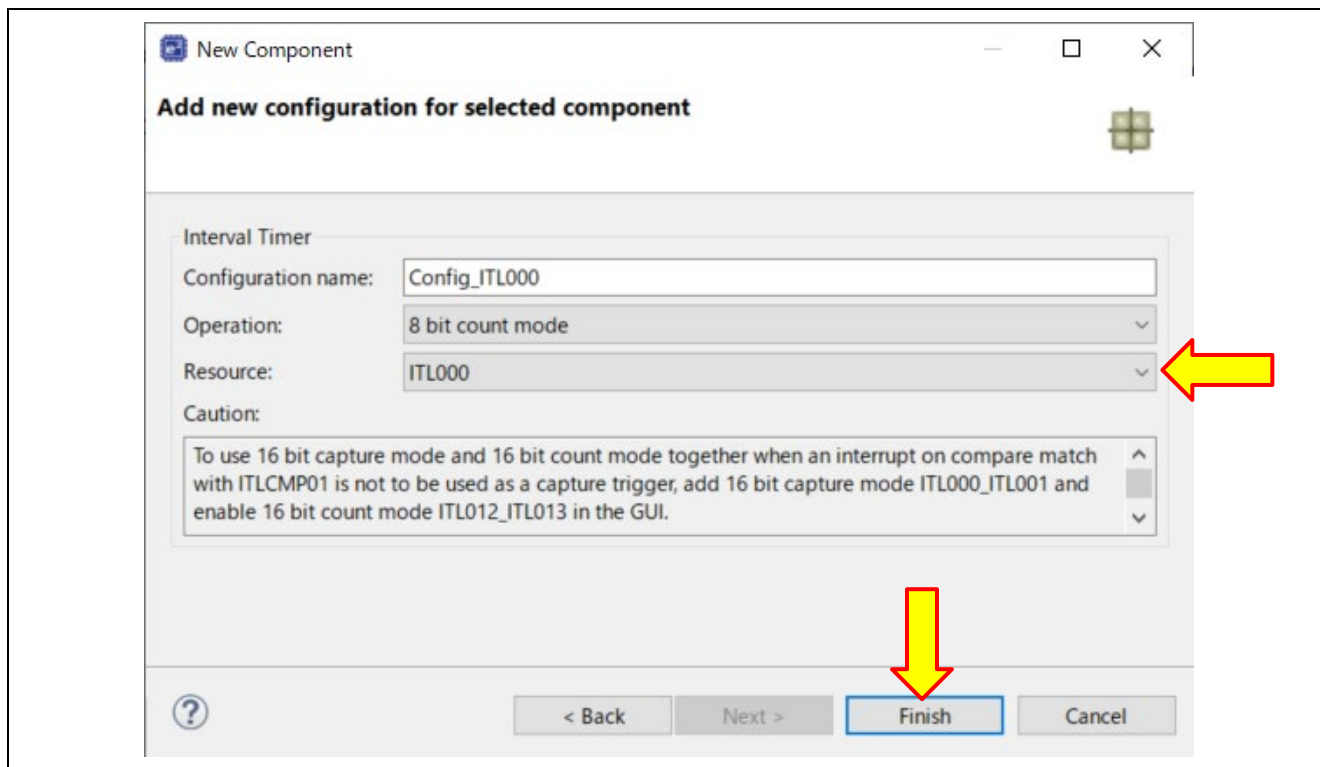


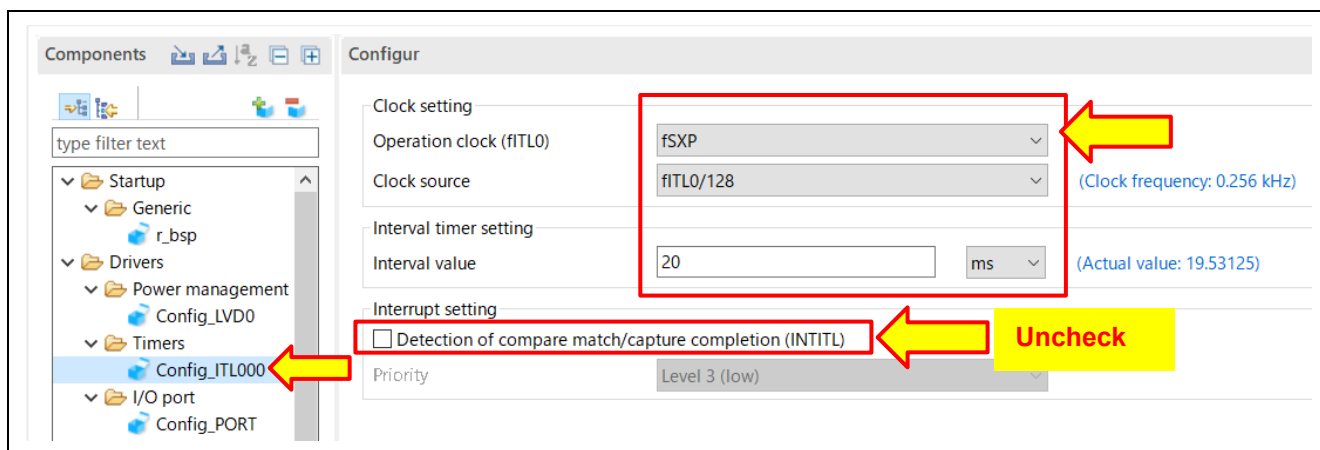
Figure 16-2 “Interval Timer” module Selection (For Hardware Timer)

- Set the interval timer configuration as shown below, and click [**Finish**] in the lower part of that open dialog box.



**Figure 16-3 “Interval Timer” module Resource Settings**

- Set the interval timer as shown below.



**Figure 16-4 Config\_ITL000 Setting**

5. Set the port to be used for turning on/off the LED for operation check as shown below.

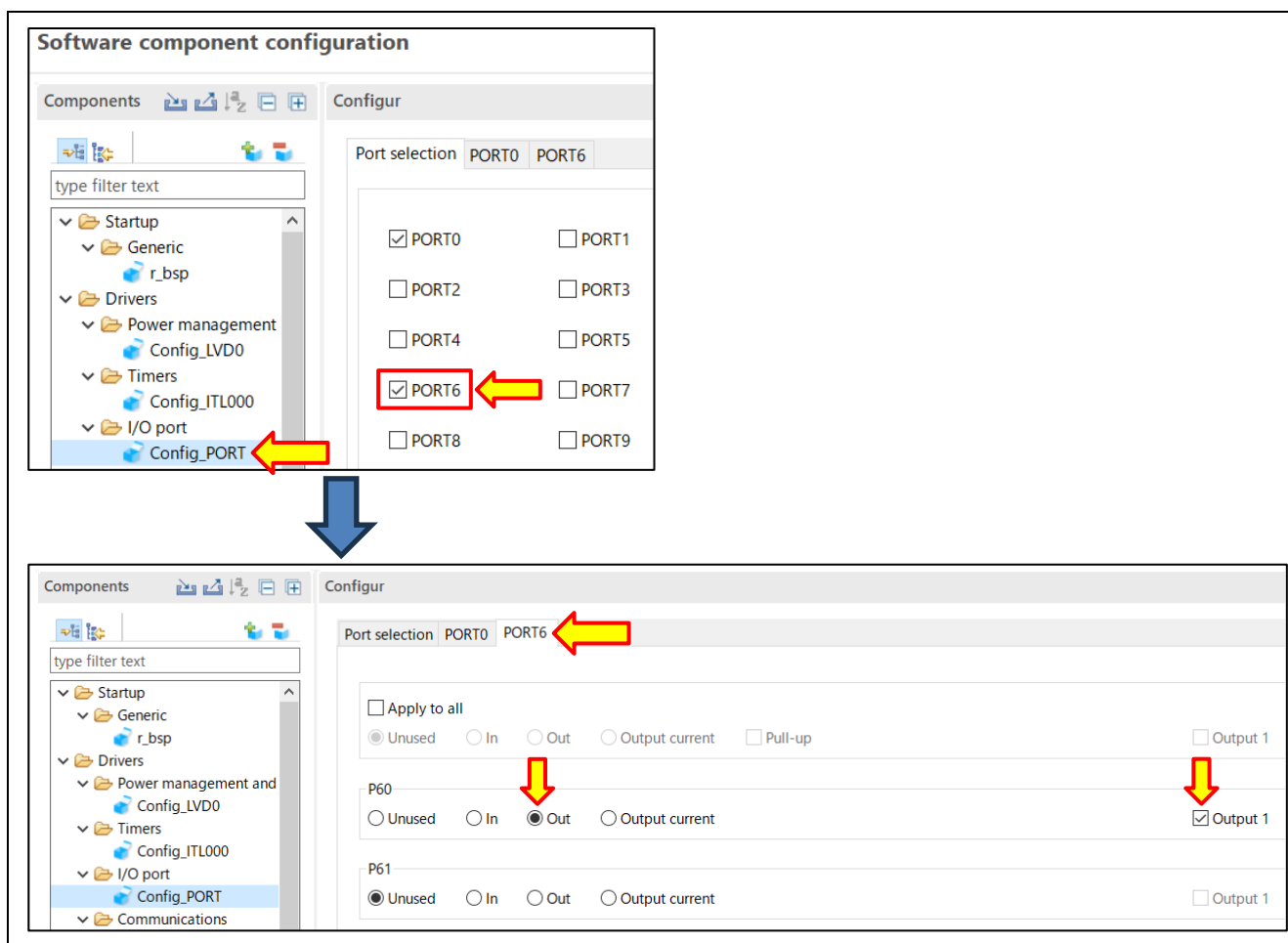



Figure 16-5 PORT Component Setting (For LED lighting)

6. Click the "Generate Code" icon  located on the top right of the Smart Configurator to generate the code for components necessary for the project.

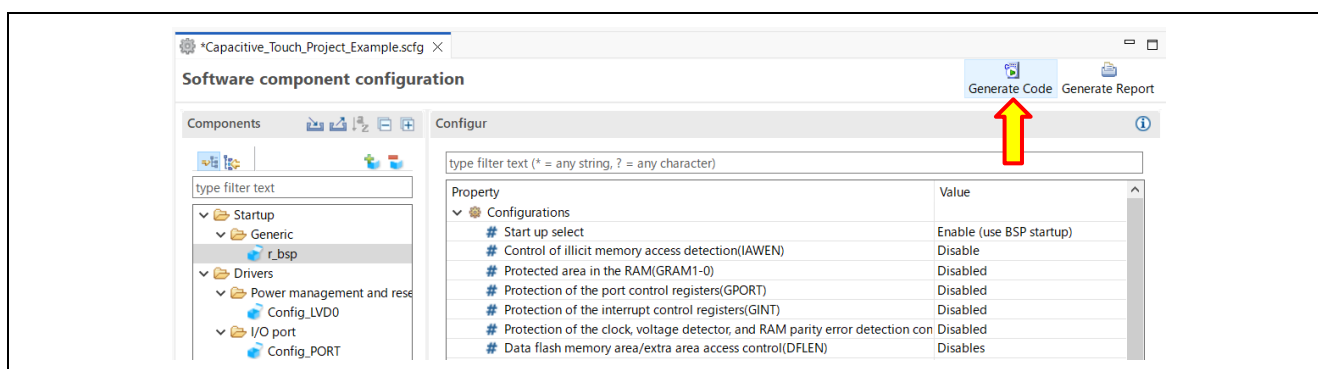


Figure 16-6 "Generating Code" icon Selection (3)

7. For an example of program implementation, please refer to "16.3 qe\_touch\_sample.c (Example of Using a Hardware Timer)".

## 16.2 Flowcharts (Example of Using a Hardware Timer)

Figure 16-7 and Figure 16-8 show flowcharts of the touch measurement control processing with the use of a hardware timer.

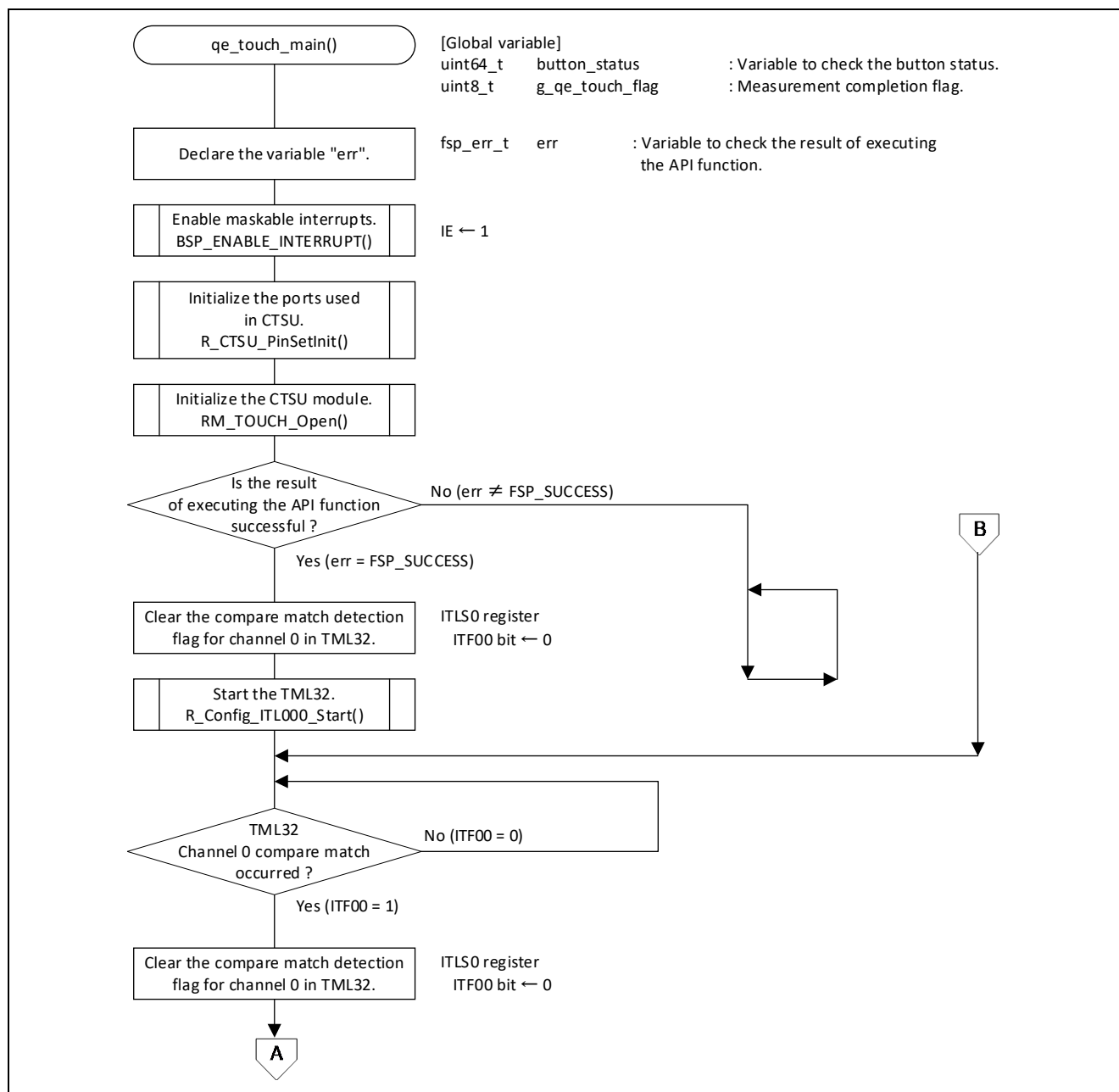


Figure 16-7 Touch Measurement Control Processing with the Use of a Hardware Timer (1/2)

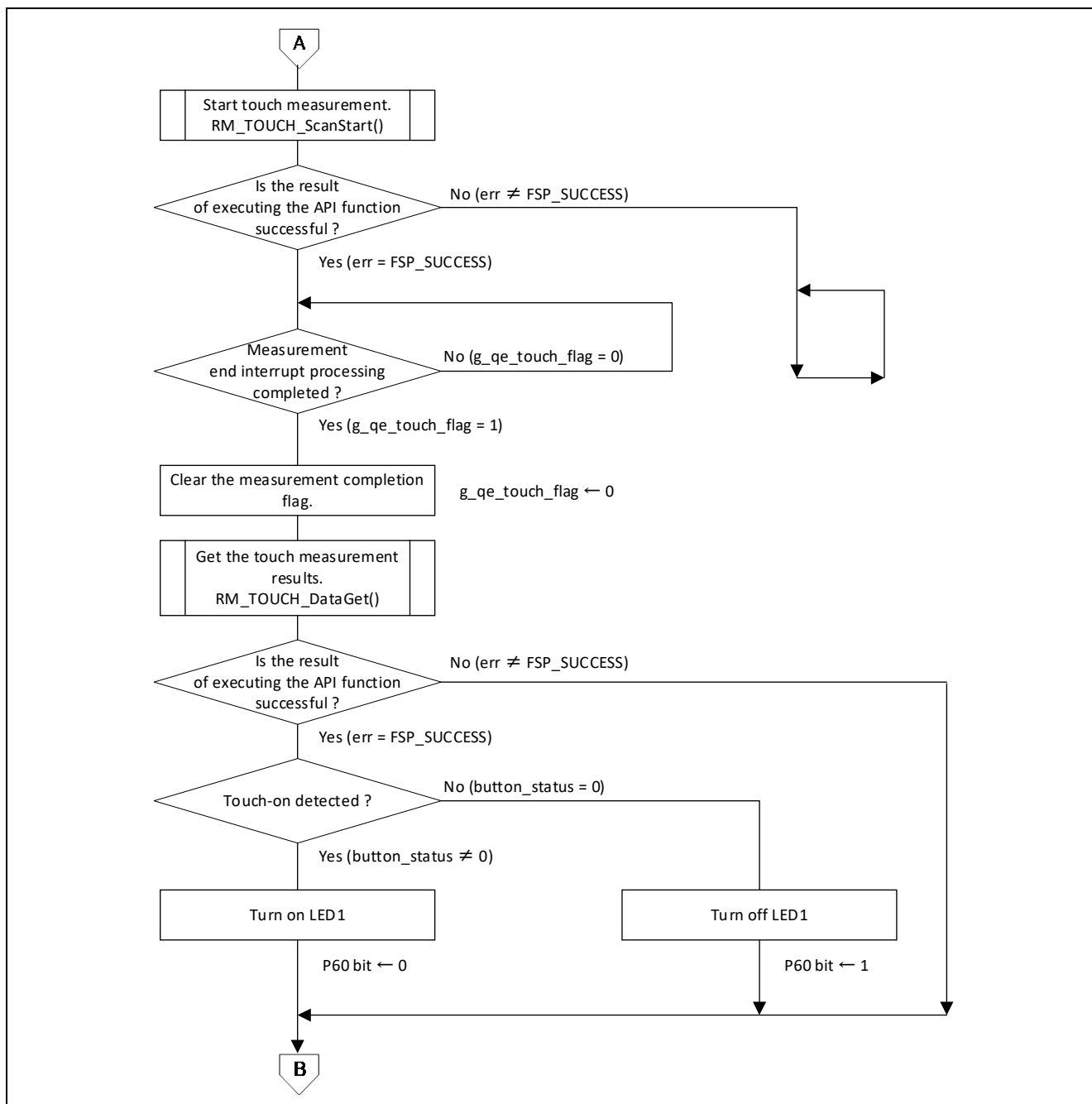


Figure 16-8 Touch Measurement Control Processing with the Use of a Hardware Timer (2/2)



### 16.3 qe\_touch\_sample.c (Example of Using a Hardware Timer)

The following sample code is an example of implementing touch measurement using a hardware timer.

The code shown in **red** below is being added to the code automatically generated by QE.

```

/*****
 * Copyright (c) 2020 - 2025 Renesas Electronics Corporation and/or its affiliates
 *
 * SPDX-License-Identifier: BSD-3-Clause
 *****/
/*****
 * File Name      : qe_touch_sample.c
 * Description    : Main Program for RL78
 *****/

#include "qe_touch_config.h"
#include "Config_ITL000.h"

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

uint64_t button_status;
#if (TOUCH_CFG_NUM_SLIDERS != 0)
uint16_t slider_position[TOUCH_CFG_NUM_SLIDERS];
#endif
#if (TOUCH_CFG_NUM_WHEELS != 0)
uint16_t wheel_position[TOUCH_CFG_NUM_WHEELS];
#endif

void qe_touch_main(void)
{
    fsp_err_t err;

    BSP_ENABLE_INTERRUPT();

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

    /* 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) {}
    }

    ITLS0 &= ~_01_ITL_CHANNEL0_COUNT_MATCH_DETECTE;

    R_Config_ITL000_Start();

```

```
/* Main loop */
while (true)
{
    while (_00_ITL_CHANNEL0_COUNT_MATCH_NOT_DETECTE == (ITLS0 & _01_ITL_CHANNEL0_COUNT_MATCH_DETECTE)) {}
    ITLS0 &= ~_01_ITL_CHANNEL0_COUNT_MATCH_DETECTE;

    /* 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. */
        if (0 != button_status)
        {
            P6_bit.no0 = 0;
        }
        else
        {
            P6_bit.no0 = 1;
        }
    }
}
}
```

## 17. Documents for Reference

### ○User's Manual

- RL78/Gxx User's Manual: Hardware  
— RL78/G23 User's Manual: Hardware (R01UH0896)
- RL78 Family User's Manual: Software (R01US0015)  
(The latest versions of the documents are available on the Renesas Electronics Website.)

### ○Technical Updates/Technical Brochures

(The latest versions of the documents are available on the Renesas Electronics Website.)

### ○User's Manual: Development Environment

- RL78/G23 Capacitive Touch Evaluation System User's Manual (R12UZ0095)  
(The latest versions of the documents are available on the Renesas Electronics Website.)

### ○Application Note

- Capacitive Sensor Microcontrollers CTSU Capacitive Touch Introduction Guide (R30AN0424)
- RL78 Family CTSU Module Software Integration System (R11AN0484)
- RL78 Family TOUCH Module Software Integration System (R11AN0485)
- Capacitive Sensor Microcontrollers CTSU Capacitive Touch Electrode Design Guide (R30AN0389)
- RL78 Family Using the standalone version of QE to Develop Capacitive Touch Applications (R01AN6574)
- RL78 Family Using the Standalone Version of QE to Develop Touch Applications for a Fast Prototyping Board (R01AN6741)
- RL78/G23 Capacitive Touch Low Power Guide (SNOOZE Mode Function) (R01AN5886)
- RL78/G22 Capacitive Touch Low Power Guide (SNOOZE function) (R01AN7413)
- RL78 Family Capacitive Touch Low Power Application Development using SMS (R01AN7261)
- RL78/G23 Capacitive Touch Low Power Guide (SMS function) (R01AN6670)
- RL78/G22 Capacitive Touch Low Power Guide (SMS / MEC function) (R01AN6847)
- (The latest versions of the documents are available on the Renesas Electronics Website.)

## Website and Support

- Renesas Electronics Website  
<http://www.renesas.com/>
- Capacitive Sensing Unit related page  
<https://www.renesas.com/solutions/touch-key>
- QE for Capacitive Touch related page  
<https://www.renesas.com/qe-capacitive-touch>

**Revision History**

Rev.	Date	Description	
		Page	Summary
1.00	Apr.30.21	-	First edition issued
2.00	Aug.31.21	-	Updated the development environment
2.10	May.20.22	-	Updated chapter "7. [Additional function] Setting the serial communication monitor using UART (1/3)". Updated chapter "14. [Additional function] Setting the serial communication monitor using UART (3/3)". Added chapter "16. [Practical applications] Touch Measurement by Hardware Timer".
3.00	May.22.23	-	Added chapter "18. [Applied Usage] Setting The Automatic Judgement (using SMS) and MEC functions".
4.00	Aug.27.25	-	Updated the development environment
		-	The overall chapter structure, descriptions, figures and tables in the application note have been revised to reflect the latest development environment (The flow of touch application development is the same as in previous versions of the application notes).
		1	Updated "Introduction".
		-	Chapters "1. Overview" to "3. Operation Confirmation Environment" have been updated with more detailed information on the requirements for developing touch applications using this application note.
		-	The section "18. [Applied Usage] Setting The Automatic Judgement (using SMS) and MEC functions" described in Rev. 3.00 has been deleted.
		83	Updated "17. Documents for Reference".

# 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 be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. 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.
6. 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.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION ("Vulnerability Issues"). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. 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.
9. 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.
10. 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.
11. 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.
12. 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.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. 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.5.0-1 October 2020)

## Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,  
Koto-ku, Tokyo 135-0061, Japan  
[www.renesas.com](http://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/](http://www.renesas.com/contact/).