RL78 Family

Using the standalone version of QE to Develop Capacitive Touch Applications

Introduction

This application note describes the steps for creating an example of a capacitive touch sensing application on the Capacitive Touch Evaluation System for RL78/G23.

This application note is a guide to the development of capacitive touch applications by using a combination of CS+, the standalone version of the Smart Configurator, and the standalone version of QE for Capacitive Touch. The standalone version of QE for Capacitive Touch enables development independently of the device or integrated development environment (IDE) by using serial communications between the PC and board.

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.

Target Device

RL78/G23

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.

Contents

1. Overview 4
2. Operating Environment 5
2.1 Functions of QE for Capacitive Touch 6
3. Configuring the Development Environment 7
3.1 Installing Development Tools 7
3.1.1 Procedure for Installing the CS+ Integrated Development Environment 7
3.1.2 Procedure for Installing the Standalone Version of the Smart Configurator 7
3.1.3 Procedure for Installing the Standalone Version of QE for Capacitive Touch 7
3.2 Hardware Settings 8
4. Procedure for Application Development 9
5. Example of Application 11
5.1 Application Example Overview 11
5.2 List of Pins Used 12
6. Creating a new Project 13
7. Settings of Smart Configurator 14
7.1 Starting the Smart Configurator 14
7.2 Setting the Clock and System 15
7.3 Downloading Software Integration System (SIS) Modules 16
Adding Components 18
7.4 18
7.5 Modifying the Component Settings in the Smart Configurator 20
7.5.1 Setting the CTSU Component 20
7.5.2 Setting the Touch Component 22
7.5.3 Setting the UART Communications Component 23
7.5.4 Setting the LVD Component 24
7.5.5 Setting the PORT Component 25
7.5.6 Checking the Board Support Package (BSP) Settings 26
7.6 Setting Unused Pins 27
7.7 Generating Code 28
8. Settings of QE for Capacitive Touch 29
8.1 Starting QE for Capacitive Touch 29
8.2 Preparing a Project 30
8.3 Configuring the Touch Interface 32
8.4 Tuning 41
8.5 Coding and Monitoring 46

8.5.1 Monitoring 46
8.6 Flowchart (Software Timer) 54

9. Another Implementation of the Sample Application 55
9.1 Touch Measurement by Hardware Timer 55
9.1.1 Using the Smart Configurator to Make Settings (Hardware Timer) 55
9.1.2 Flowchart (Hardware Timer) 58
9.1.3 Sample Code (Hardware Timer) 59

10. Documents for Reference 61

Revision History 62

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 the standalone version of Smart Configurator
- Creating, tuning, and monitoring touch interfaces by using the standalone version of QE for Capacitive Touch

The application note describes the procedures for using the Capacitive Touch Evaluation System for RL78/G23, but the procedures can also be applied to other devices of the RL78 family that incorporate the capacitive touch IP.

2. Operating Environment

Table 2-1 and Table 2-2 list the elements of the environment used in development for this application note.

The sample code attached to this application note was developed with the versions of tools listed in Table 2-1. This application note also supports development with the versions in parentheses ( ) in the table.

The program generated by the standalone version of QE is written to the RL78/G23 and then executed on CS+ and E2 emulator Lite.

Table 2-1 Operating Environment (Software)

Items	Contents	Version
Integrated development environment (IDE)	CS+ for CC	V8.14.00 (V8.09.00 or later)
Compiler	CC-RL	V1.15.01 (V1.12.00 or later)
Development assistance tool for capacitive touch sensors	Standalone version of QE for Capacitive Touch	V4.2.0
Smart Configurator	RL78 Smart Configurator	V1.14.00 (V1.5.00 or later)
Software integration system (SIS) modules	• Capacitive sensing unit driver (r_ctsu) • Touch middleware (rm_touch)	V2.20

Caution: 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)".

Table 2-2 Development Environment (Hardware)

Items	Contents
Target MCU	RL78/G23 (R7F100GSN2DFB)
Target Board	Capacitive Touch Evaluation System • RL78/G23 CPU Board – RTK0EG0029C01001BJ • Touch Application Board – RTK0EG0019B01002BJ
Emulators	E2 emulator Lite

Operation of the sample code attached to this application note was verified under the following conditions.

Table 2-3 Conditions for Verifying Operation

Item	Description
Operating voltage	5.0 V Level of voltage detection by LVD0 in reset mode For rising: 2.67 V typ. (2.59 V to 2.75 V) For falling: 2.62 V typ. (2.54 V to 2.70 V)
Operating frequency	High-speed on-chip oscillator clock (f_{IH}): 32 MHz

2.1 Functions 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.

The following shows the main functions of QE for Capacitive Touch.

- Creating touch interface configurations
  This allows the visual placement of touch-interface elements such as buttons and assignment of touch sensors (electrodes) to the elements.

- Tuning
  This allows automatic offset and sensitivity tuning of the touch interface.

- Monitoring operation and adjusting parameters
  This allows monitoring of the operation of the touch interface and the fine adjustment of parameters.

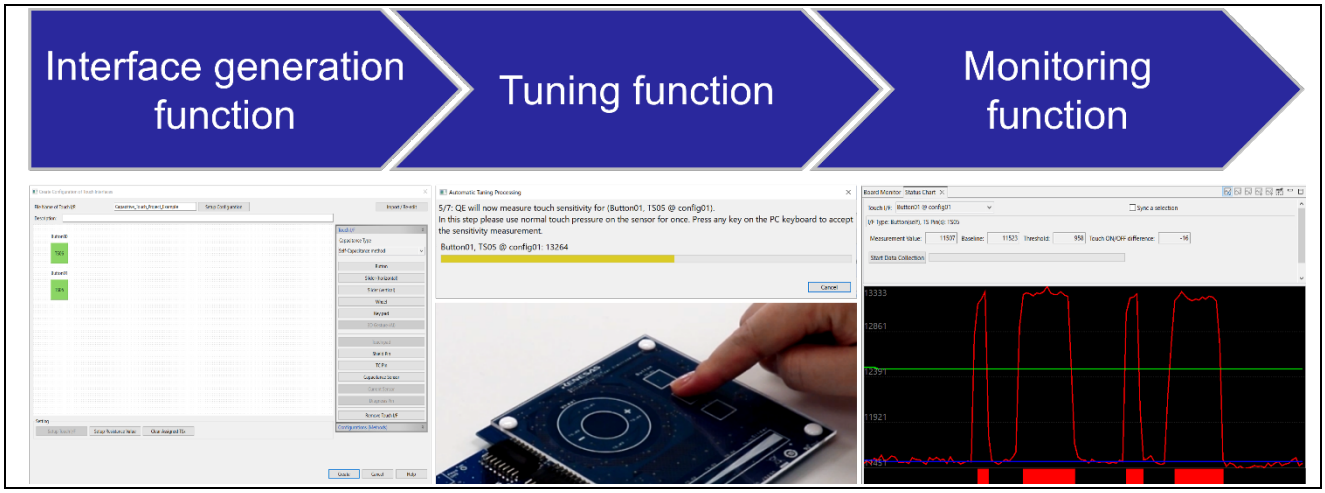


Figure 2-1 Main Functions of QE for Capacitive Touch

3. Configuring the Development Environment

This chapter describes the procedures for installing the development environment and setting up the hardware.

3.1 Installing Development Tools

The following tools are used with this sample application.

- CS+
- Standalone version of Smart Configurator
- Standalone version of QE for Capacitive Touch

If these tools have already been installed, skip the procedures in the rest of section 0.

3.1.1 Procedure for Installing the CS+ Integrated Development Environment

1. Download the installer for the latest version of the CS+ for CC integrated development environment from the following link.
   CS+ IDE and Coding Tool | Renesas
2. Decompress the downloaded zip file and execute the installer file.
3. Click on "Begin CS+ Setup".
4. Check that "Tools for RL78 family" is selected.
5. After installation has been completed, click on the [Finish] button.

3.1.2 Procedure for Installing the Standalone Version of the Smart Configurator

1. Download the installer for the latest version of the RL78 Smart Configurator from the following link.
   RL78 Smart Conifgurator | Renesas
2. Execute the downloaded EXE file to start the installer.
3. After the installer has started, follow the instructions for installation as they appear on the screen.

3.1.3 Procedure for Installing the Standalone Version of QE for Capacitive Touch

1. Download the installer for the latest version of QE for Capacitive Touch, a development assistance tool for capacitive touch sensors, from the following link.
   QE for Capacitive Touch: Development Assistance Tool for Capacitive Touch Sensors | Renesas
2. The downloaded zip file contains both the plug-in and standalone versions of QE. Extract the contents of the zip file and install the standalone version.

Caution: Extract the contents in a location as close as possible to the root of a drive so that the pathname does not exceed the limit on the number of characters (260) in a pathname for Windows.

Example of a suitable location: Under the "C:\Renesas" folder

Do not specify the Windows folder, the Program Files folder, or a folder that has a name which includes white space.

3.2 Hardware Settings

This section describes the hardware settings and connection of the target board. Table 3-1 shows the jumper settings on the target board for this sample application. Power supply for the target board uses USB. 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 3-1 Jumper Settings on the Board

Jumper	Circuit Group	Setting	Description
JP1	Power supply	Pins 1-2 closed	Supplies USB power to board USB
JP2		Closed	Supplies board device power to MCU
JP3		Pins 1-2 closed	Supplies JP1 power to board devices
JP4			

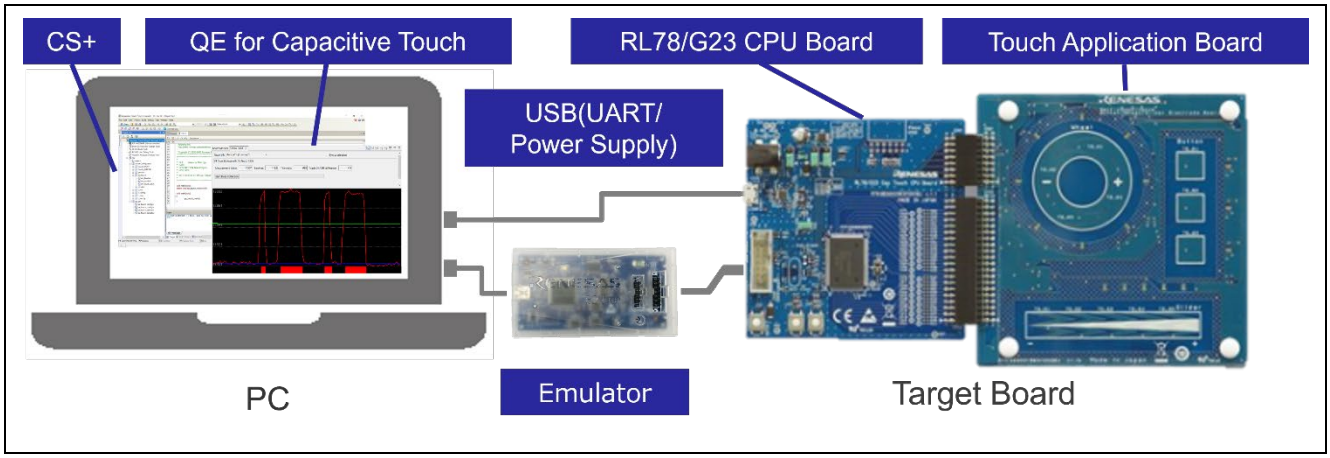


Figure 3-1 Connection of Target Board with PC

4. Procedure for Application Development

This chapter describes how to develop an application.

Follow the steps in the workflow of QE for Capacitive Touch.

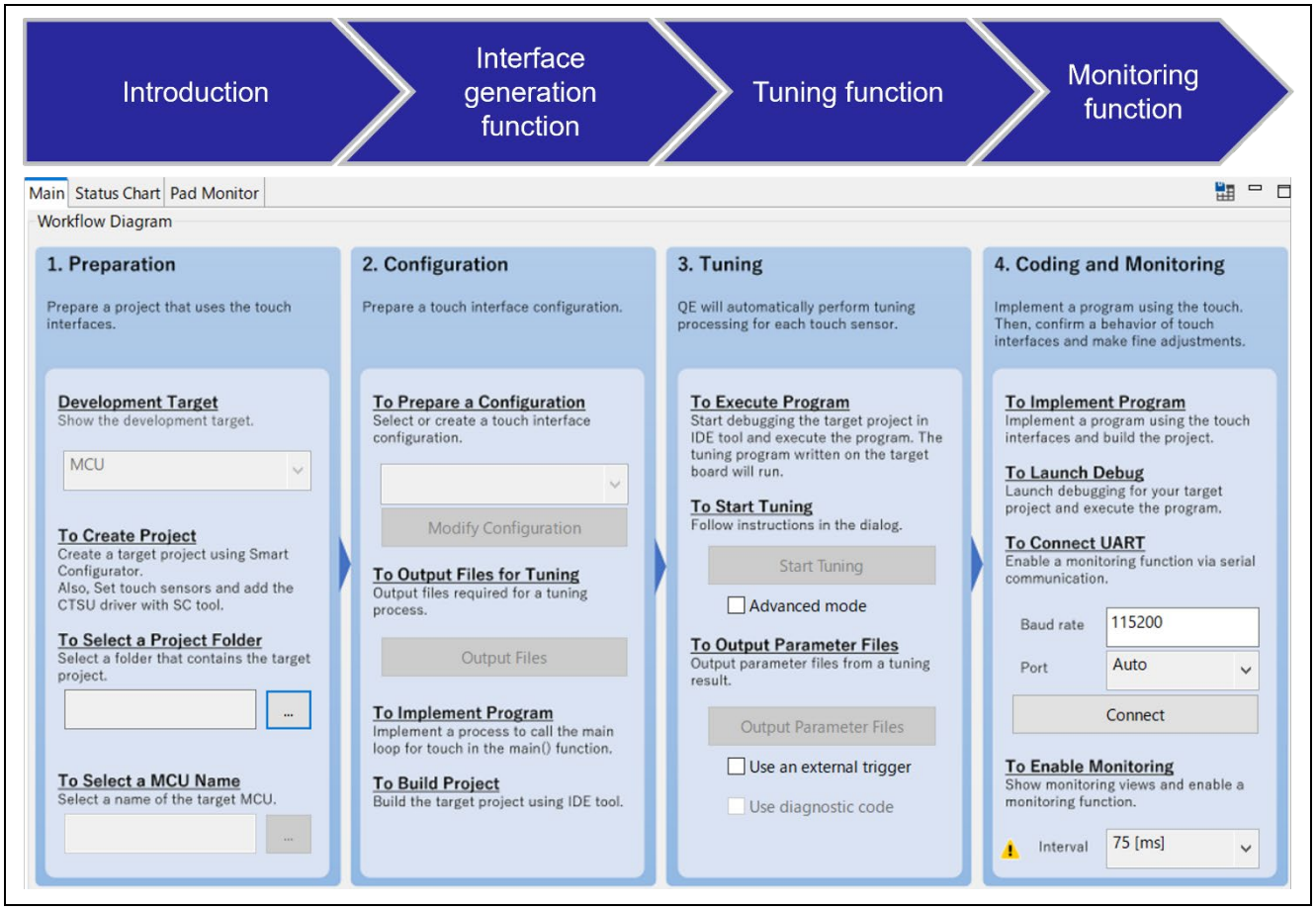


Figure 4-1 Procedure for Application Development

Figure 4-1 shows the steps of workflow. Chapter numbers of the Table link to each chapter page. Click each chapter number of the Table and see how to use it. IDE and Smart Configurator is used for project creation and coding, build of project, debug.

Table 4-1 Workflow of Development Using QE for Capacitive Touch

Item				Section Number
Preparation	To Create Project	Creation of a project by using the IDE		6
		Settings made by the Smart Configurator		7
			Clock and system	7.2
			CTSU driver	7.5.1
			Touch middleware	7.5.2
			Serial interface (UART)	7.5.3
			Voltage detector (LVD)	7.5.4
			Port functions (PORT)	7.5.5
			Board support package	7.5.6
			Unused pins	7.6
	To Select a Project Folder			8.2
	To Select an MCU Name			
Configuration	To Prepare a Configuration			8.3
	To Output Files for Tuning			
	To Implement Program			
	To Build Project			
Tuning	To Execute Program			8.4
	To Start Tuning			
	To Output Parameter Files			
Coding and Monitoring	To Implement Program			8.5
	To Launch Debug			
	To Connect UART			
	To Enable Monitoring			

5. Example of Application

5.1 Application Example Overview

This application note describes an example of an application that uses two buttons. This example also involves tuning and monitoring the touch performance through serial communications.

The method of creating an application that uses two buttons and monitoring whether a button is being touched are described in chapter 6 and subsequent sections.

Remark: Communications for checking the touch performance of a touch application can also be handled by an on-chip debugging (OCD) emulator. For devices of the RL78 family, however, note that the on-chip debugging functionality of the device limits performance in monitoring.
Using serial communications enables smooth monitoring of the touch performance. Serial communications can also be used in tuning for the touch sensors.

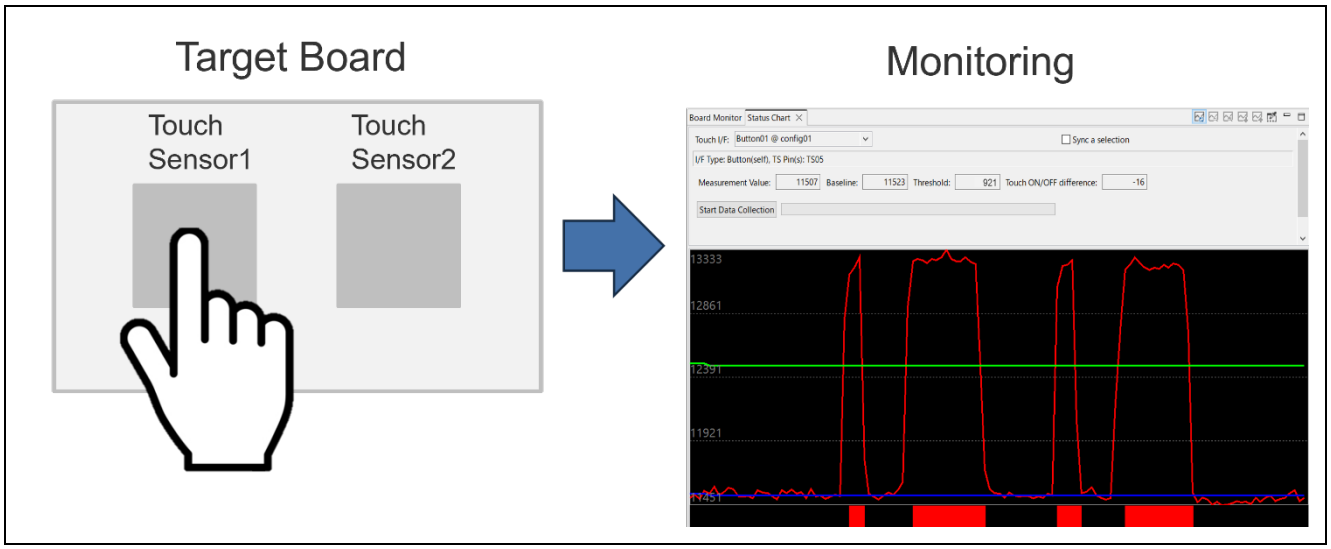


Figure 5-1 Sample Application

This application note is provided with the two sets of sample code listed in Table 5-1. Both have the same procedure for development but differ in terms of some settings by the Smart Configurator and the code to be added to the qe_touch_sample.c file. Chapters 6 to 8 describe the procedure for development by taking the touch application implemented with the use of a software timer as an example. For implementation of the application with the use of a hardware timer and the control of LEDs to indicate sensor operation, see chapter 9, Another Implementation of the Sample Application

Table 5-1 Overview of the Attached Sample Code

File Name	Timer for Generating Cycles of Touch Measurement	LED Control
Capacitive_Touch_Project_Example	Software timer	None
Capacitive_Touch_Project_HardwareTimer_Example	Hardware timer	Included

5.2 List of Pins Used

Table 5-1 lists the pins used in this sample application.

UART communications and touch sensors should be set up according to the specifications of the target board in use.

Table 5-1 List of Pins Used in the Sample Application

Items	Pins	uses
UART Communications	RxDA1/P33	For tuning and monitoring
	TxDA1/P34	
Touch Sensor 1	TS06/P74	For detecting touching of the TS_B1 button
Touch Sensor 2	TS05/P73	For detecting touching of the TS_B2 button

Figure 5-2 shows positions of the touch sensors used for this application example.

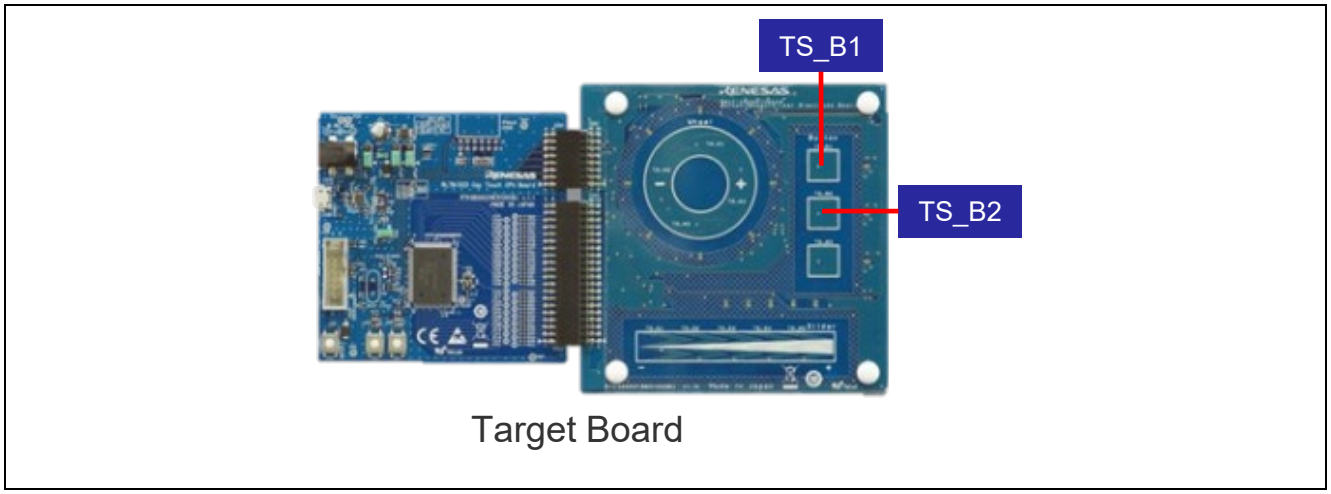


Figure 5-2 Locations of the Touch Sensors

6. Creating a new Project

Start CS+ and create a new project.

For this sample application, make the following selections in the [Create Project] dialog box.

- Microcontroller : RL78
- Using microcontroller : R7F100GSNxFB (128pin)
- Kind of project : Application (CC-RL)
- Project name : (Desired project name)
- Place : (Desired place)

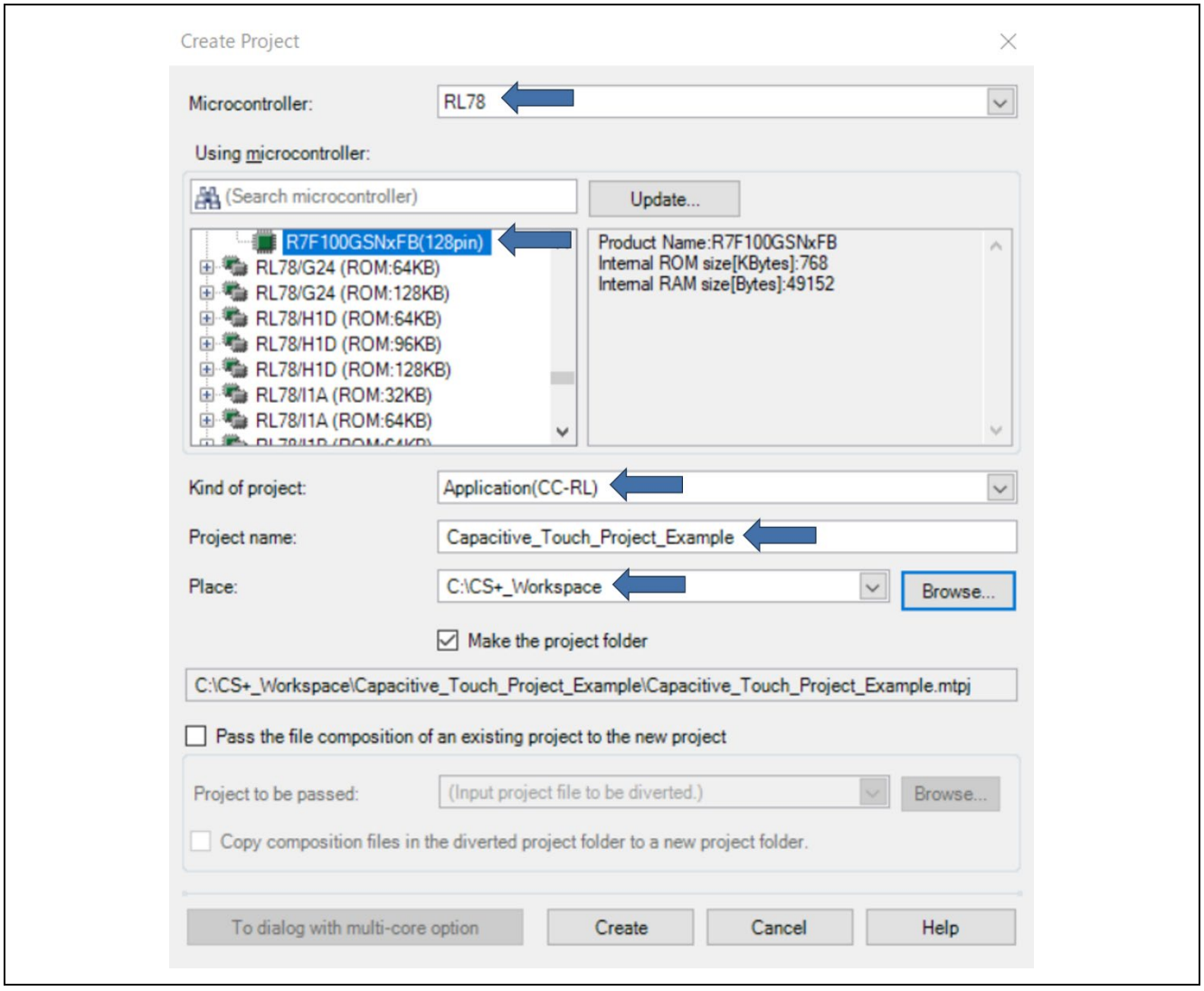


Figure 6-1 Creating a New Project

7. Settings of Smart Configurator

This chapter describes the procedures for using the Smart Configurator to make settings. The settings required for this sample application are listed below.

- Clock and system
- CTSU driver
- Touch middleware
- Serial interface (UART communications)
- Voltage detector (LVD)
- Port functions (PORT)

7.1 Starting the Smart Configurator

Double-click on "Smart Configurator" in [Project Tree] of CS+ to start the Smart Configurator.

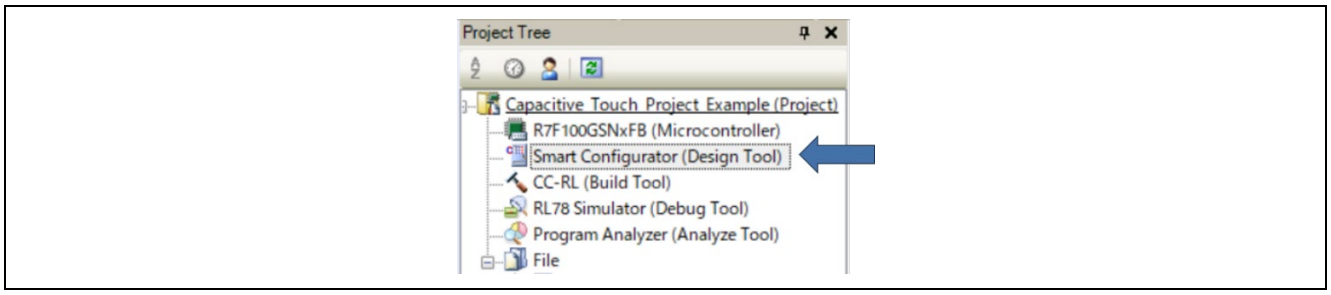


Figure 7-1 Starting the Smart Configurator

If the Smart Configurator does not start, check that the following two items are correctly specified.

- A correct file path is specified in [Property] of the Smart Configurator.
- "Smart Configurator for RL78 Communication Plug-in" is selected in the [Plug-in Manager] dialog box that can be opened from the [Tool] menu.
-

Figure 7-2 Path to the Smart Configurator File

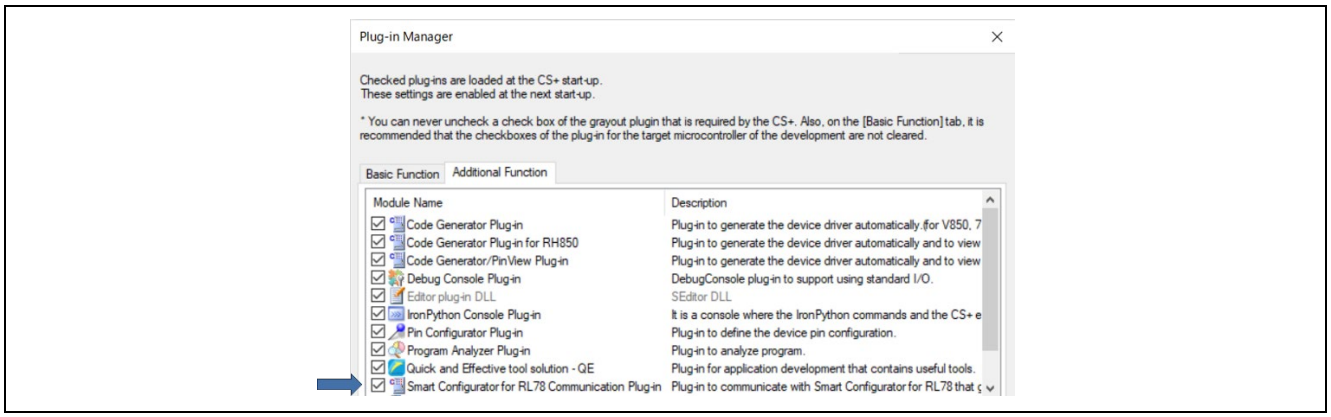


Figure 7-3 Plug-in Manager

7.2 Setting the Clock and System

This section describes the procedure for setting up the clock and system.

1. After starting the Smart Configurator, select the [Clocks] tab at the bottom of the Smart Configurator view and set up the clock. If the target MCU requires the use of EVDD, select an appropriate value for "EVDD setting" according to the operating mode.

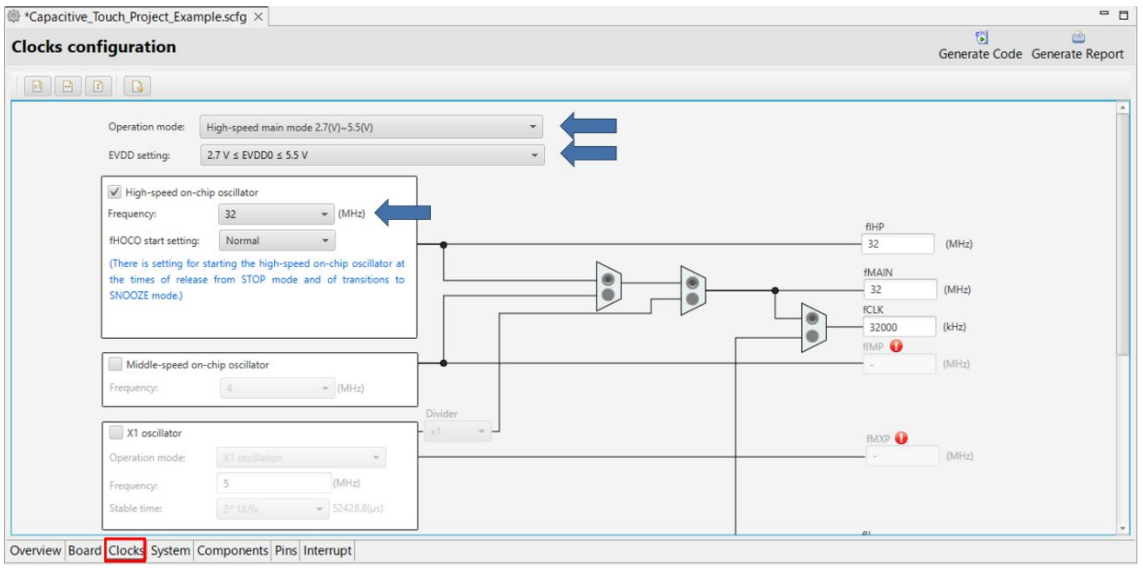


Figure 7-4 Setting the Clocks

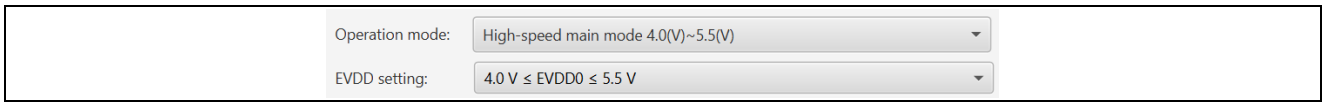


Figure 7-5 Setting EVDD

2. Select the [System] tab and set up the debugging environment.

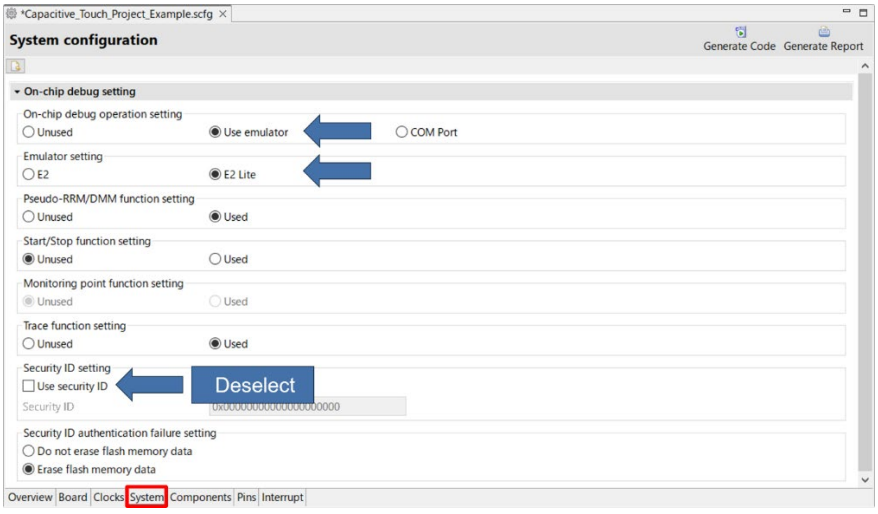


Figure 7-6 Settings for Debugging

7.3 Downloading Software Integration System (SIS) Modules

This section describes how to download two SIS modules, the CTSU driver and touch middleware, which are necessary to implement a touch application. If they have already been installed, skip the steps in the rest of this section.

1. Select the [Components] tab and click on the icon.

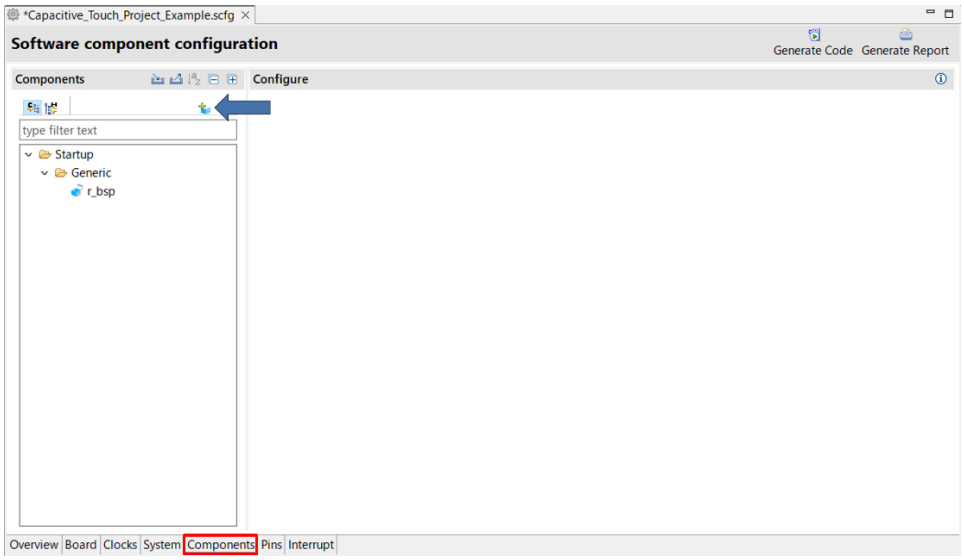


Figure 7-7 [Software component configuration] View

2. Click on “Download RL78 Software Integration System modules” at the bottom of the [New Component] dialog box.

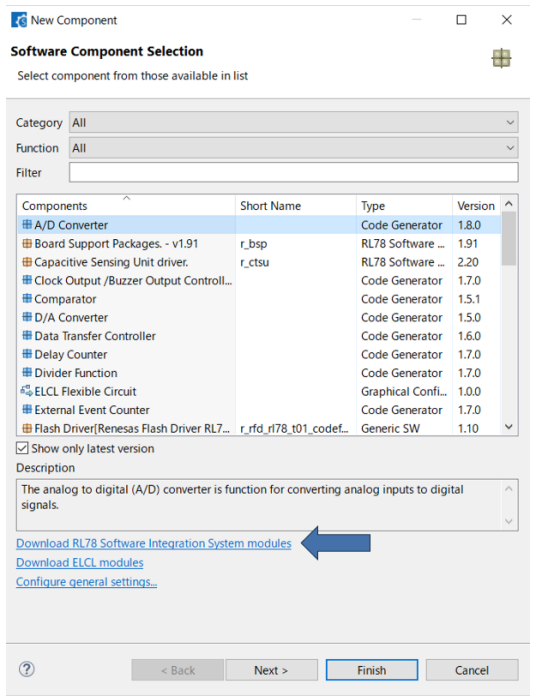


Figure 7-8 [New Component] Dialog Box

3. A dialog box will open. Select the following items and click on the [Download] button.
   — RL78 Family CTSU Module Software Integration System
   — RL78 Family TOUCH Module Software Integration System

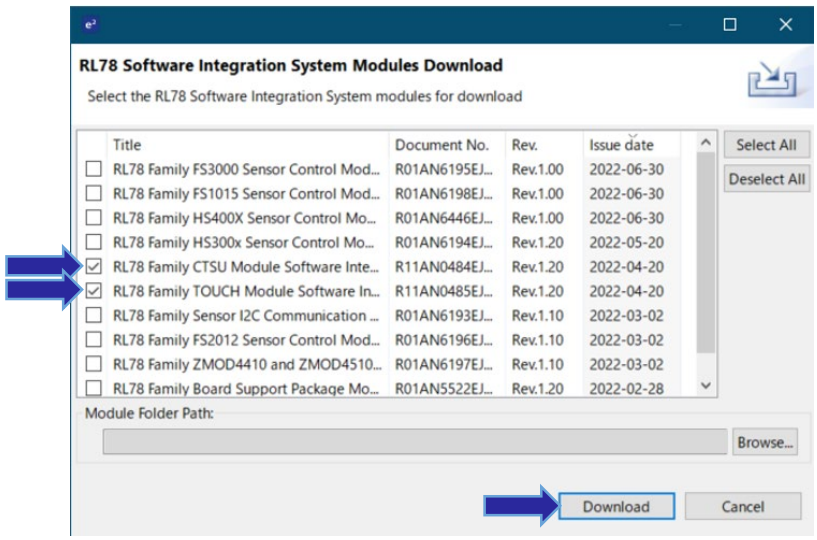


Figure 7-9 Download SIS Modules

Caution: 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.2.20 - Sample Code | Renesas

  RL78 Family TOUCH Module Software Integration System
  RL78 Family TOUCH Module Software Integration System Rev.2.20 - Sample Code | Renesas

- How to download and use the modules
  How to Download and Use a SIS Module

7.4 Adding Components

1. Select the components shown below in the Smart Configurator.

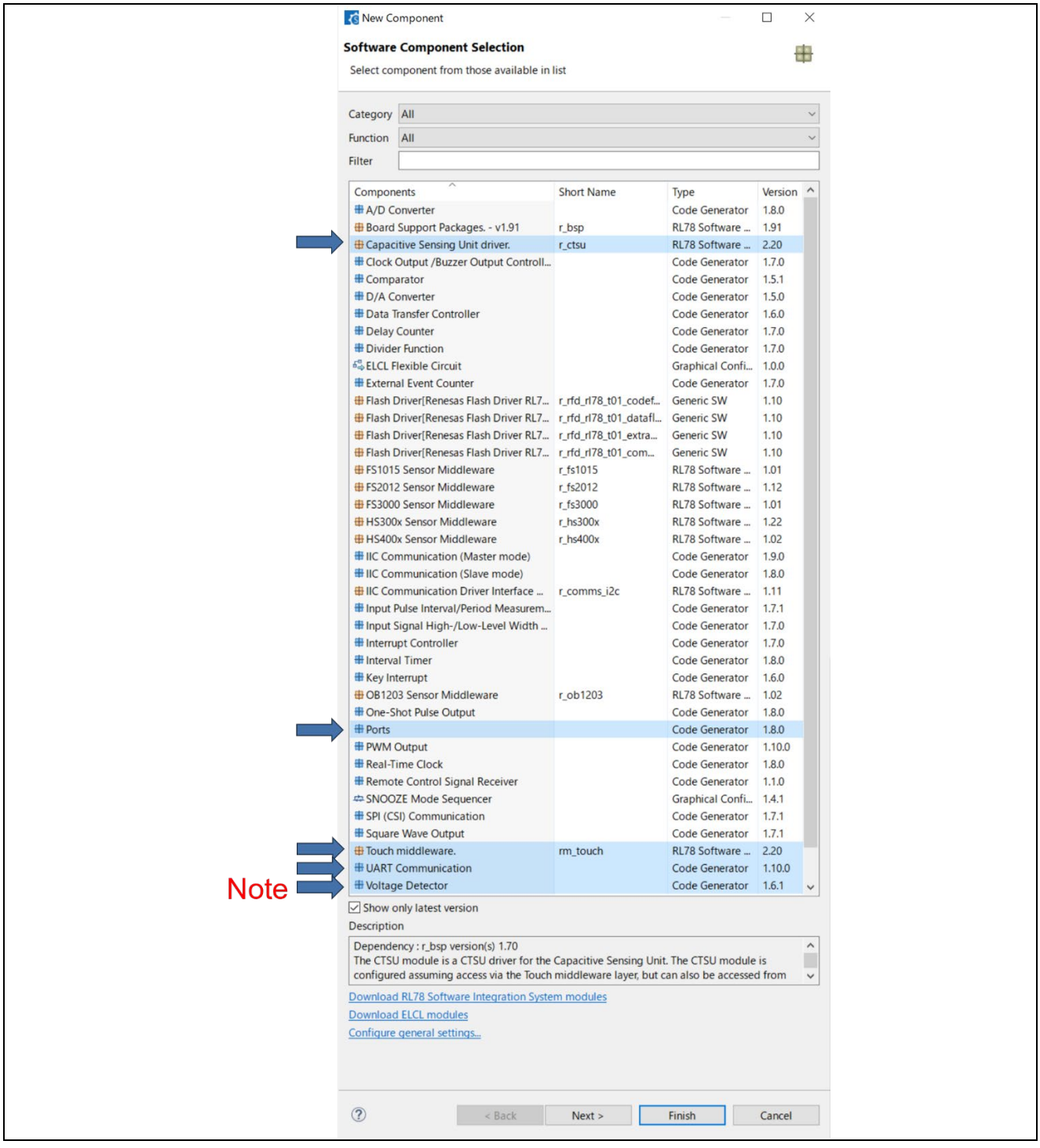


Figure 7-10 Software Component Selection

Note: For the RL78/G16, this component does not require addition in this way because the method of setting the voltage detector function differs from that for the other devices.
For the method of voltage detector setting for the RL78/G16, see section.

2. Assign resources to the selected components. Use the following settings for this sample application.

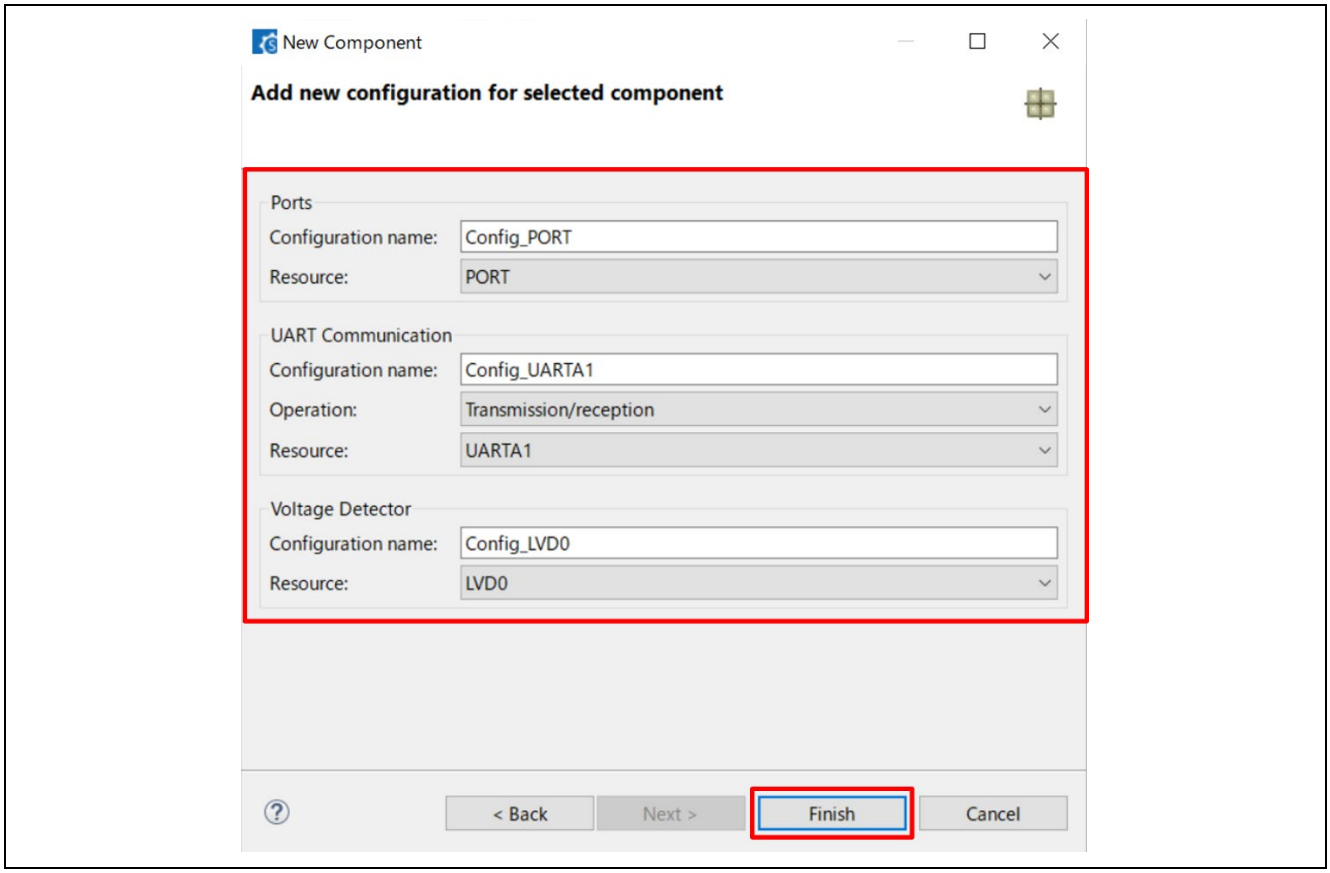


Figure 7-11 Assigning Resources to Components

The components are added as shown below.

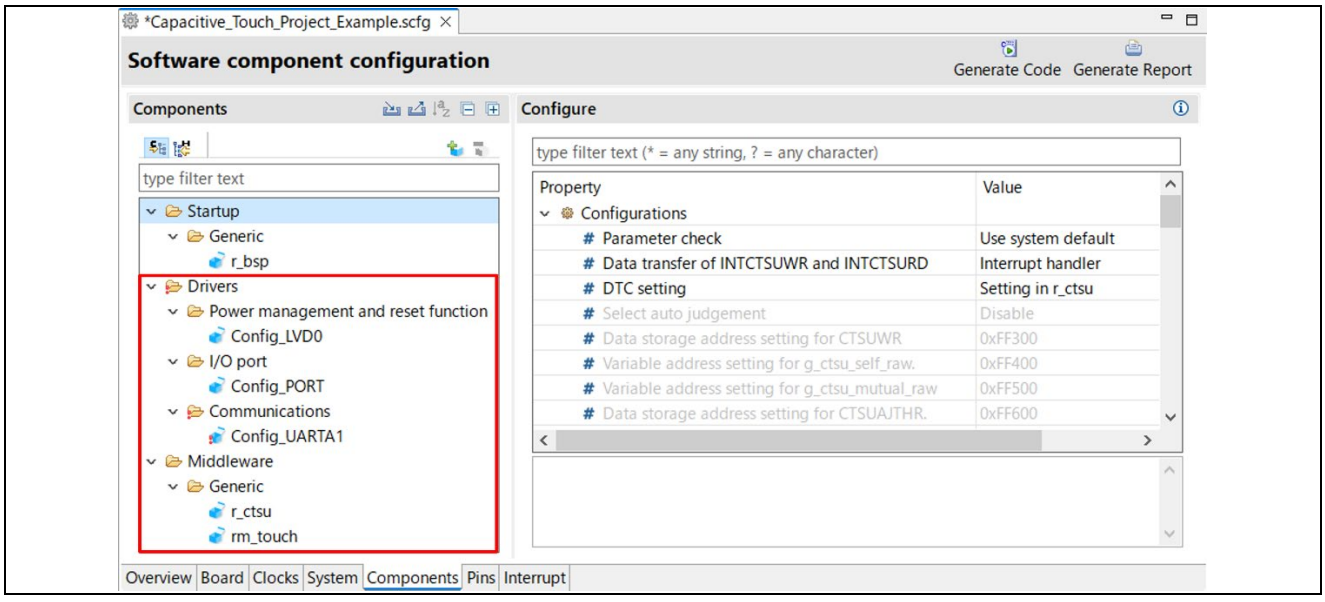


Figure 7-12 Software Component Configuration (after the Components are Added)

7.5 Modifying the Component Settings in the Smart Configurator

Set up the components added in the previous steps.

7.5.1 Setting the CTSU Component

Click on the "r_ctsu" module and enable the TSCAP pin and five TS pins to be used by this sample application. For the correspondence between the TS pins and touch sensors, refer to the user's manual of the target board you are using.

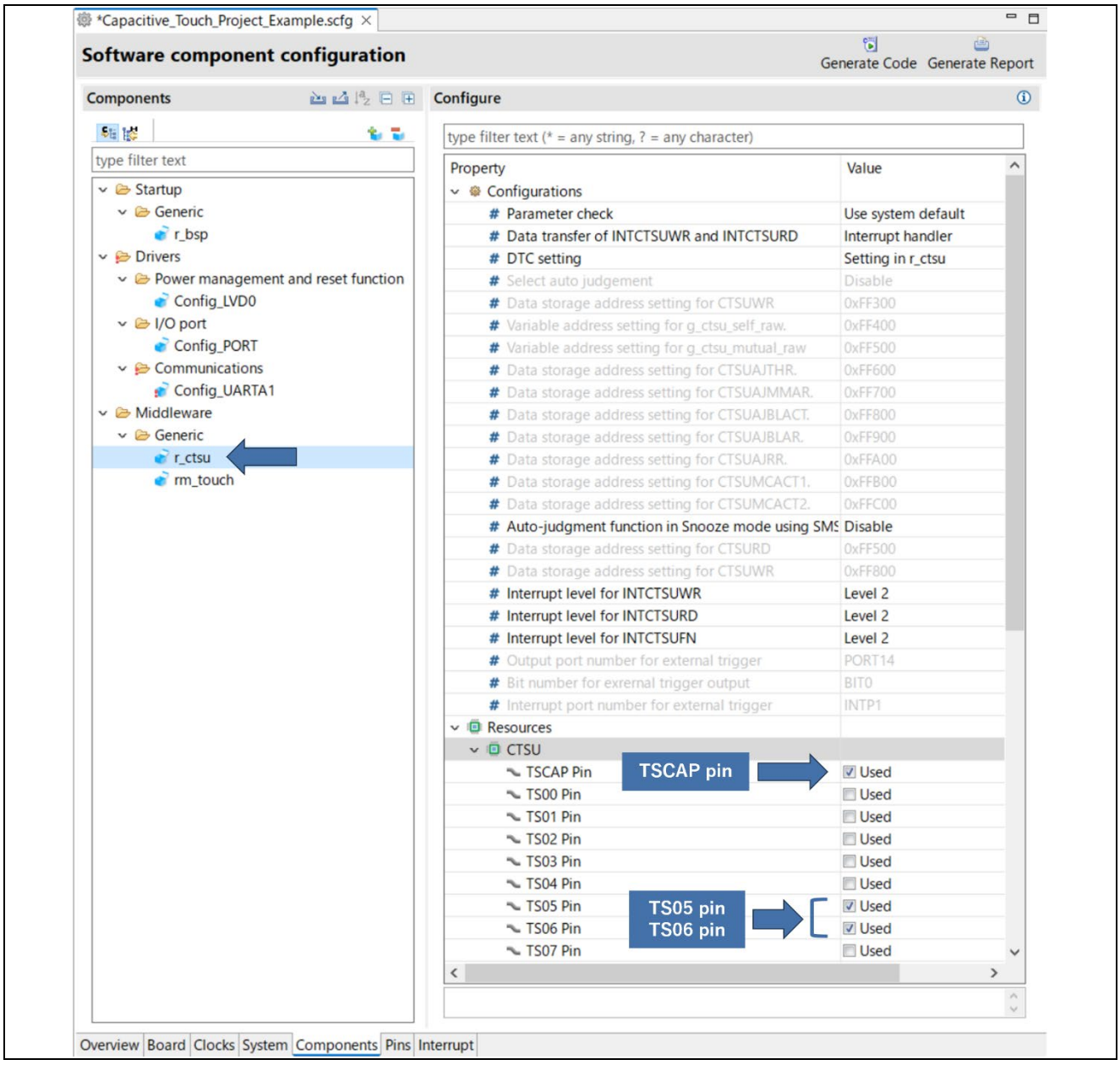


Figure 7-13 Enabling the TSCAP Pin and TS Pins to be Used by the Application

For the TS pins that are not to be used by the application, setting them as outputs of the low level is recommended. For the CTSU2 module, if the TS pins not to be used by the application are enabled, the Smart Configurator automatically handles them as non-measurement pins and sets them as outputs of the low level.

Therefore, in the attached two sets of sample code, all TS pins are enabled even if they are not to be used by the application.

Caution: In designing circuits for a user board, appropriately handle the pins such that the electrical characteristics are satisfied.

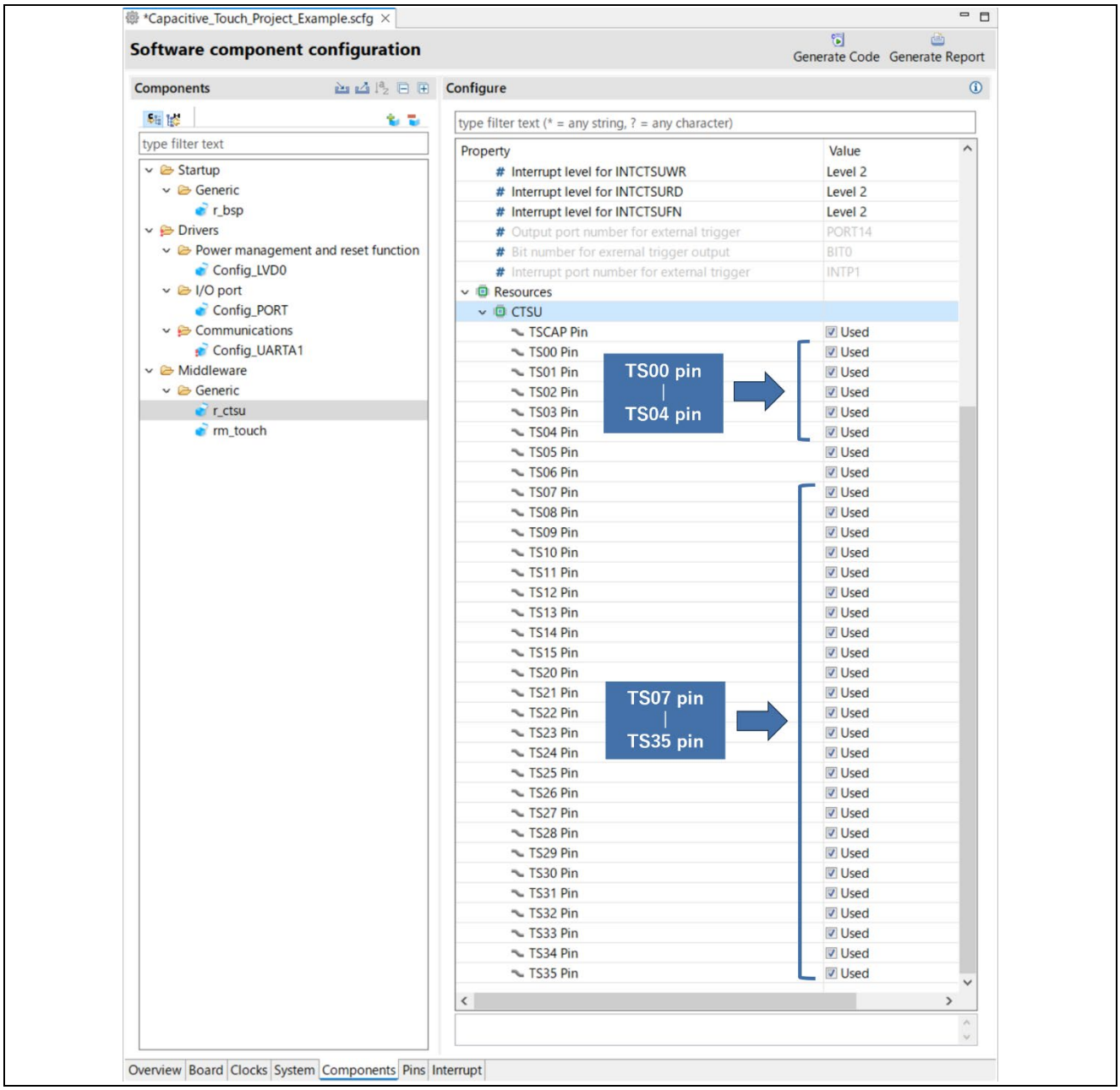


Figure 7-14 Enabling the TS Pins Not to be Used by the Application

7.5.2 Setting the Touch Component

Click on the "rm_touch" module and make the following settings.

- Support QE monitor using UART: Enable
- Support QE tuning using UART: Enable
- UART channel: UART0

The UART channel to be used differs with the target board. For details, refer to the circuit diagram of the target board you are using.

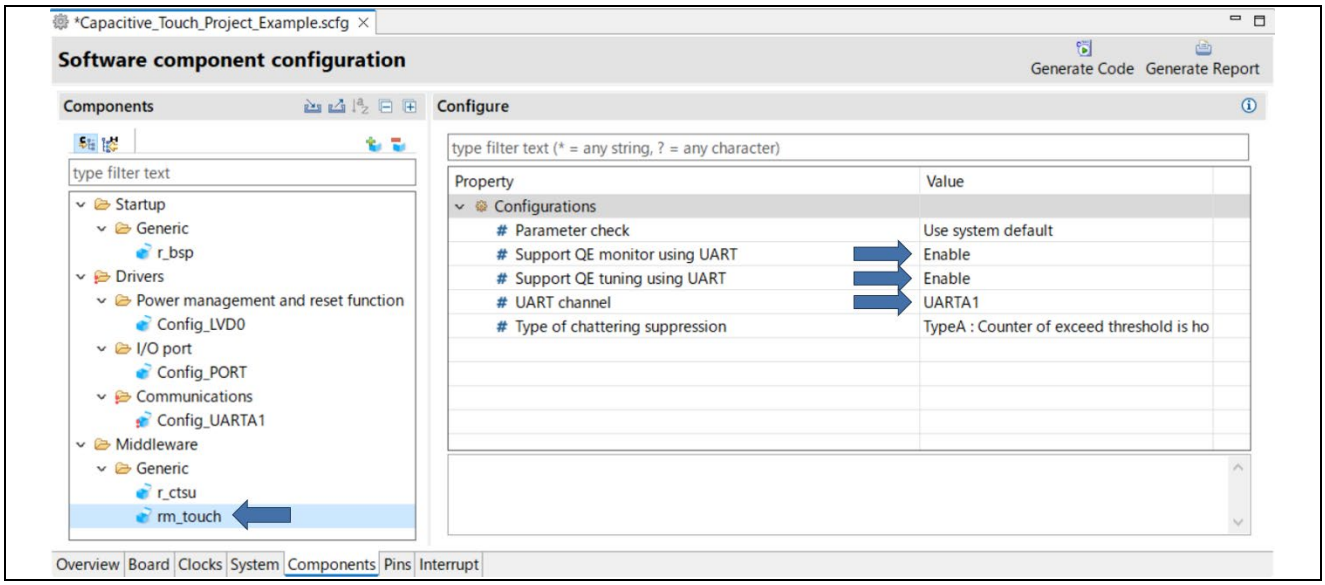


Figure 7-15 Setting rm_touch

7.5.3 Setting the UART Communications Component

This section describes the procedures for setting up the UART to be used for tuning and monitoring the touch sensors.

The UART channel and port to be used differ with the target board. Click on the added UART communications module and select the operating clock and transfer rate according to the target transfer rate.

Make the specifications shown in the figure below for this sample application.

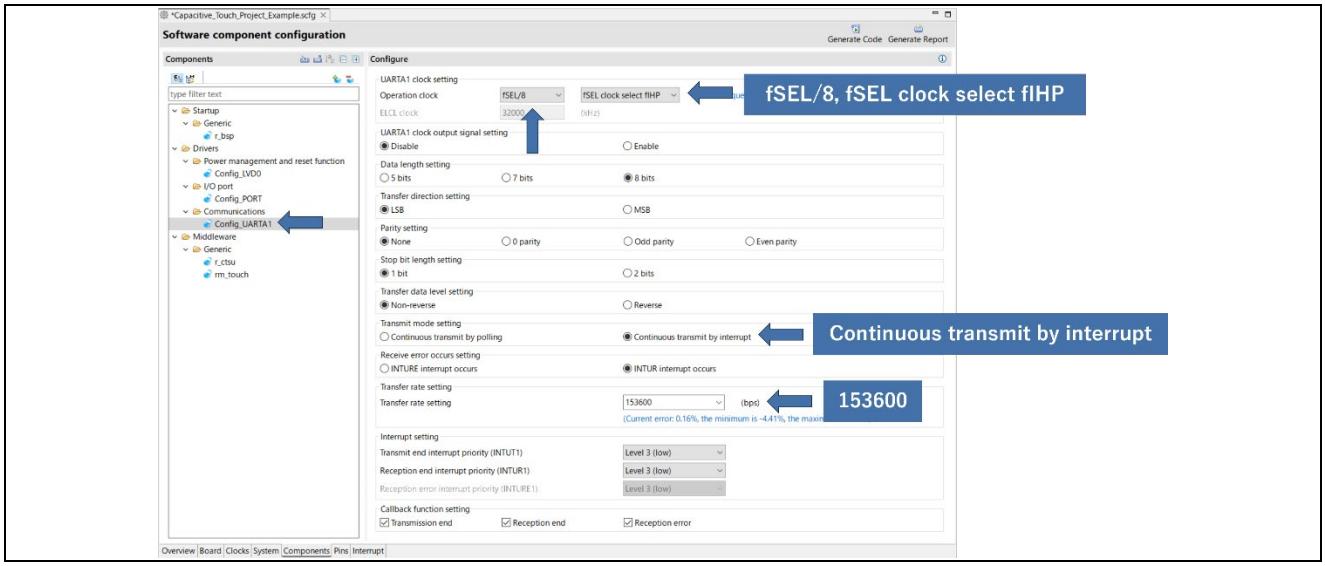


Figure 7-16 Setting the UART Communications Component (UARTA1)

After that, select the [Pins] tab and set up the pins to be used for UART communications.

Make the specifications shown in the figure below for this sample application.

Caution: These settings differ with the target device. For the pins that can be used for UART communications, refer to the circuit diagram of the FPB you are using. For the pin numbers to be specified, refer to section 1.3, Pin Configuration, in the User's Manual — Hardware for the target RL78 device.

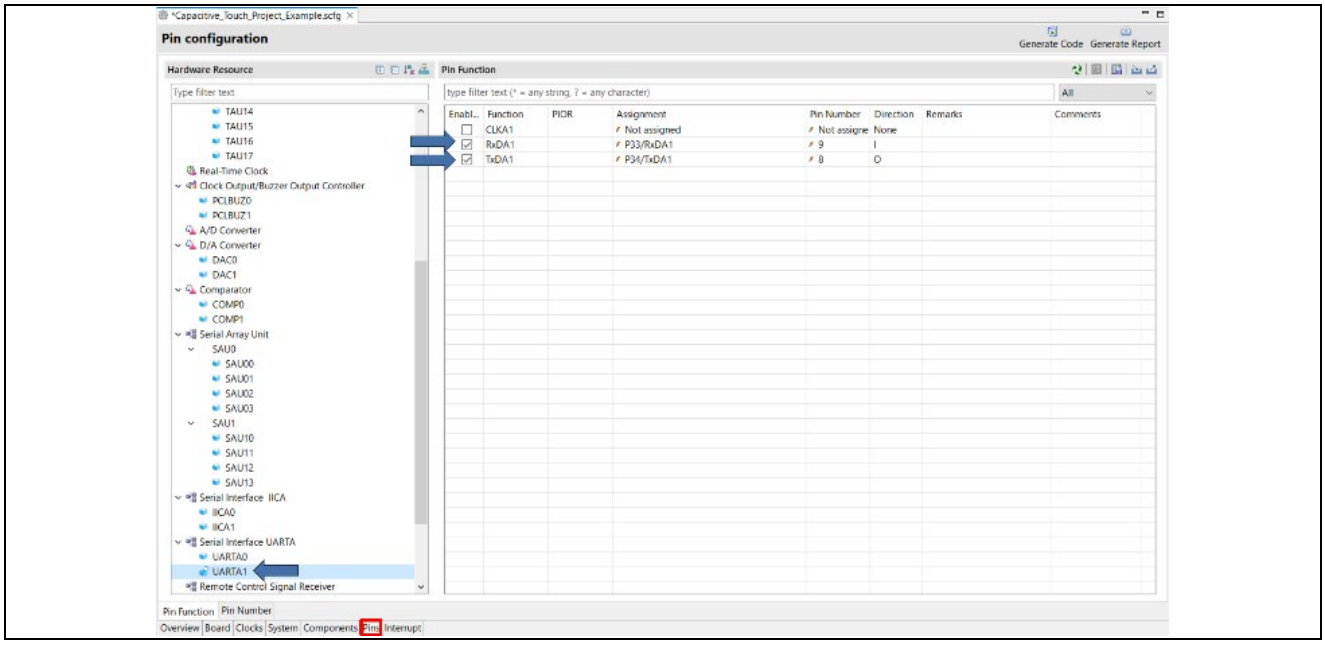


Figure 7-17 Assigning Pins to the UART Channel (UARTA1)

7.5.4 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.

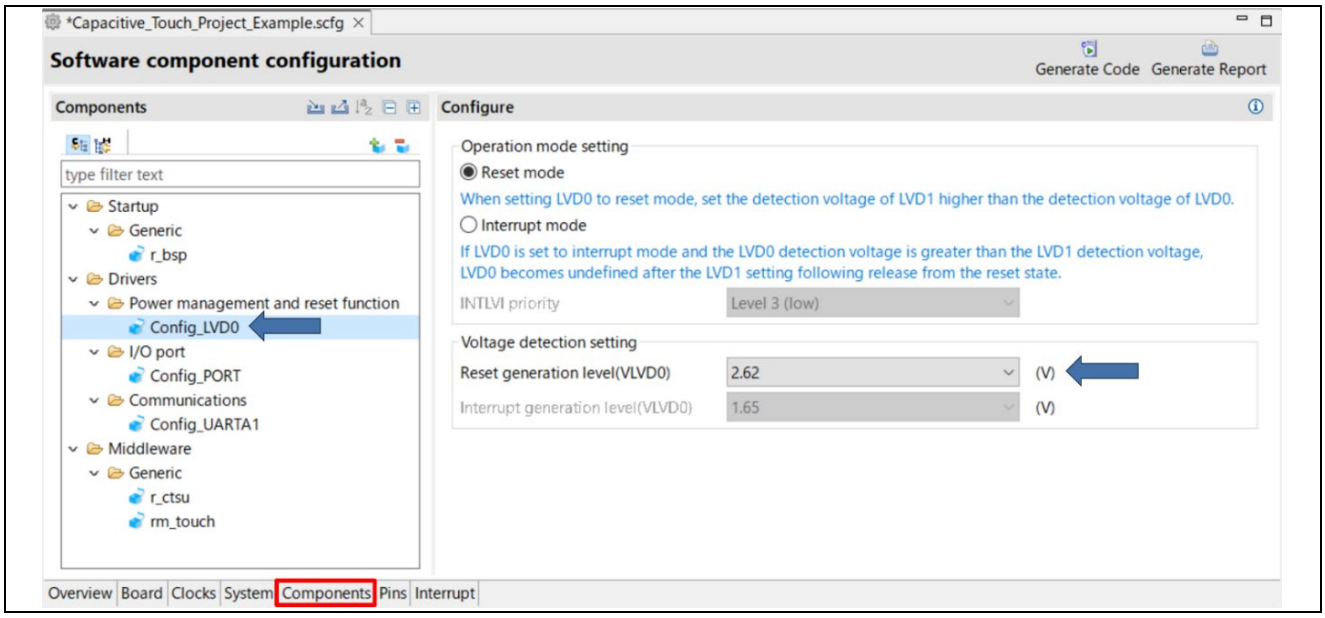


Figure 7-18 Setting the LVD Component (LVD0)

Caution: In the RL78/G16, the selectable power-on-reset circuit is used for the voltage detection function. For this function, specify the voltage for generating a reset on the [System] tabbed page as shown in Figure 7-19.

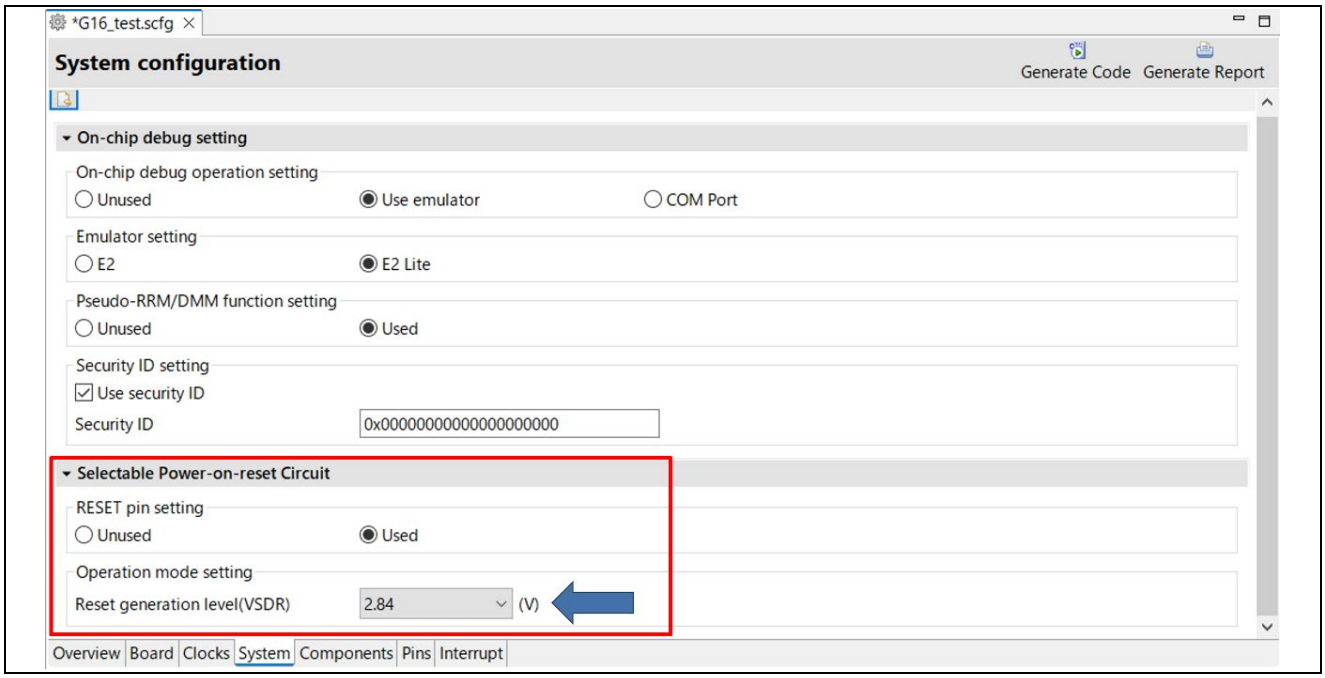


Figure 7-19 Setting the Voltage for Generating a Reset (RL78/G16)

7.5.5 Setting the PORT Component

Specify the port pins that are connected to the LEDs.

Set P60 and P61 as outputs initially at the high level in this sample application.

For details of the port pins used to control the individual LEDs, refer to the circuit diagram of the target board you are using.

1. Click on the "PORT" module and select "PORT6".

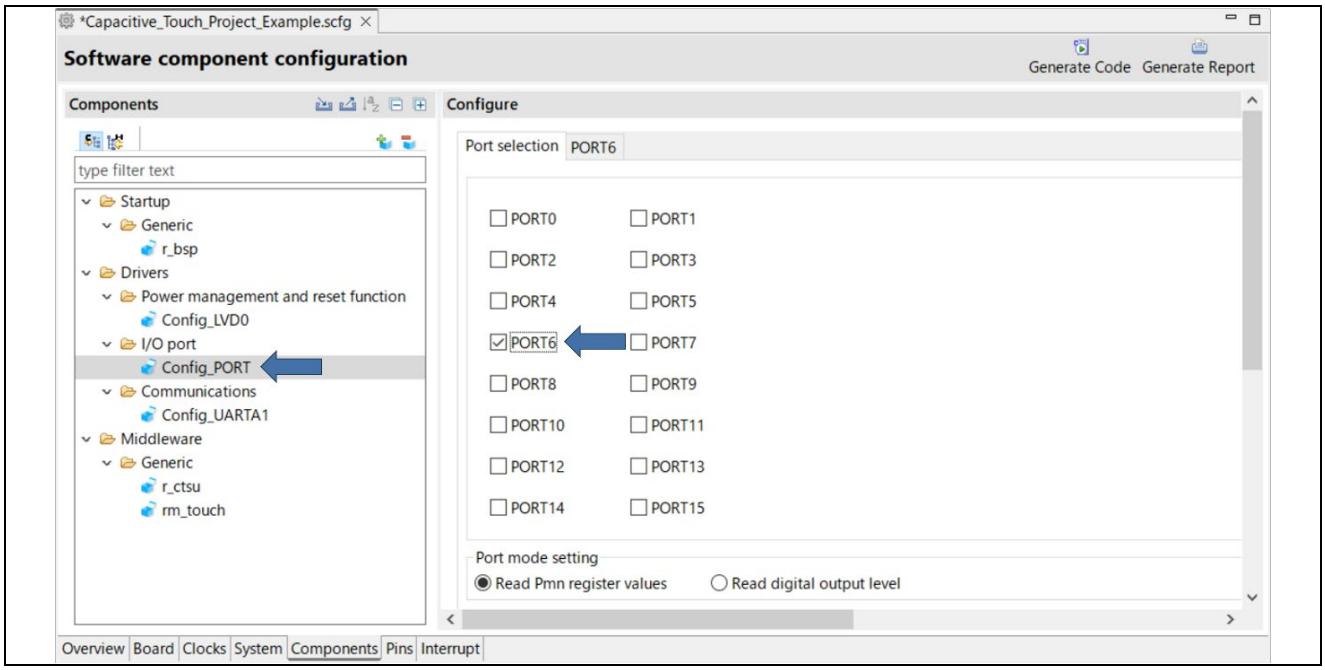


Figure 7-20 Setting the PORT Component

2. Set up the port pins so that LED1 and LED2 are not turned on at startup.
   Specifically, click on the [PORT6] tab and set "P60" and "P61" as outputs initially at the high level.

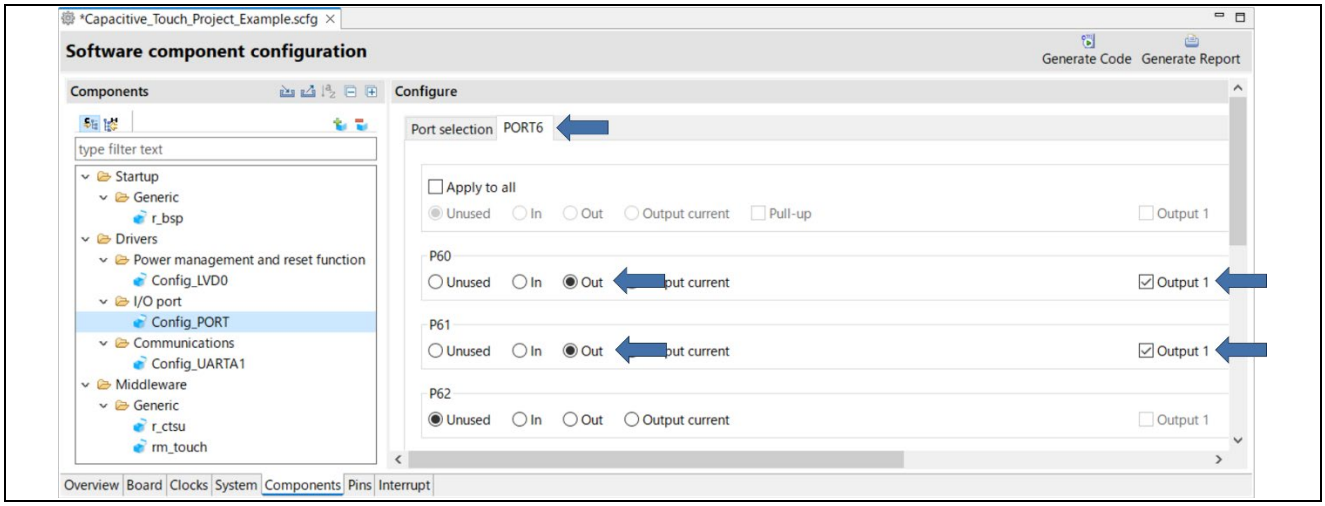


Figure 7-21 Setting P60 and P61 as Outputs at the High Level

7.5.6 Checking the Board Support Package (BSP) Settings

Select the "r_bsp" module and check that “Initialization of peripheral functions by Code Generator/Smart Configurator” is set to "Enable".

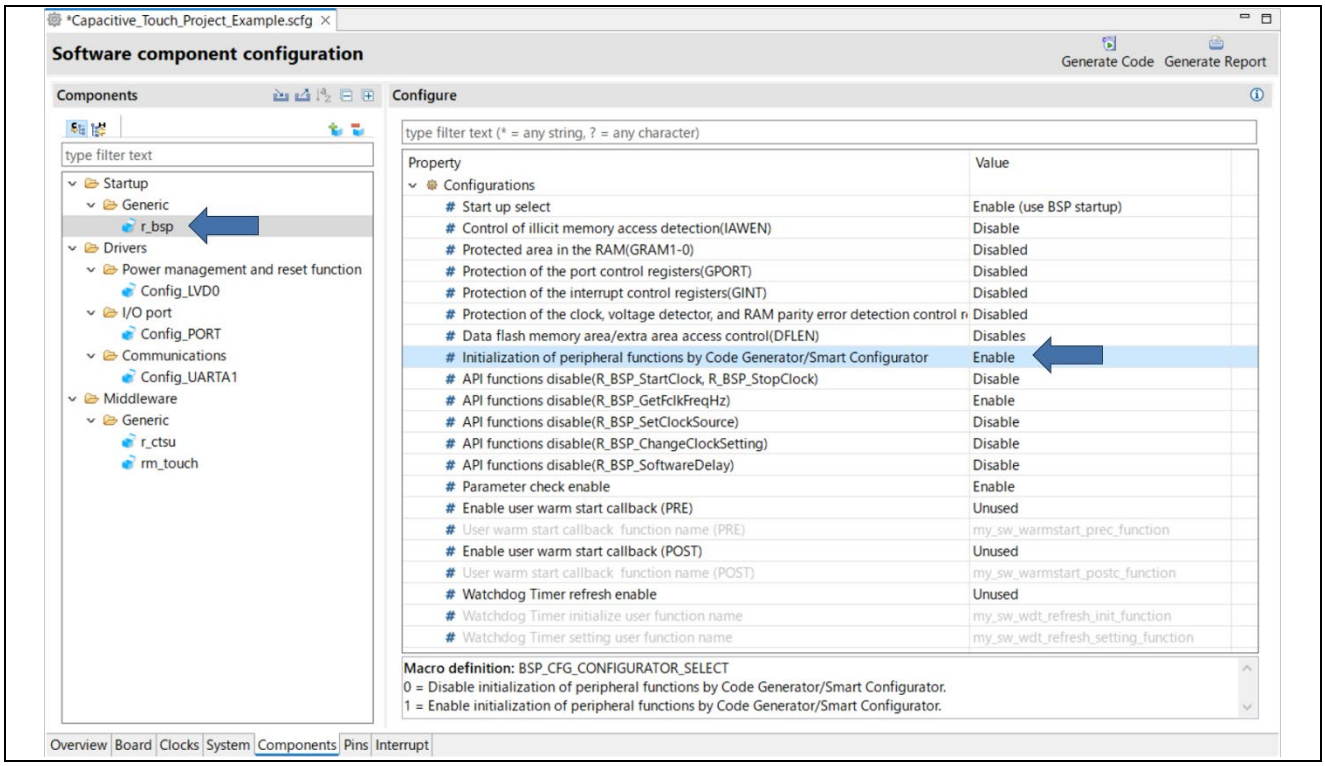


Figure 7-22 Setting r_bsp

7.6 Setting Unused Pins

Setting the unused pins as outputs at the low level is recommended.

As an example, this section describes the procedure for setting PORT07 as an output at the low level. Please see user's manual of your target board in order to confirm ports which you need to set to low-level output.

Caution: In designing circuits for a user board, appropriately handle the pins so that the electrical characteristics are satisfied.

1. Click on the "PORT" module and select "PORT0".

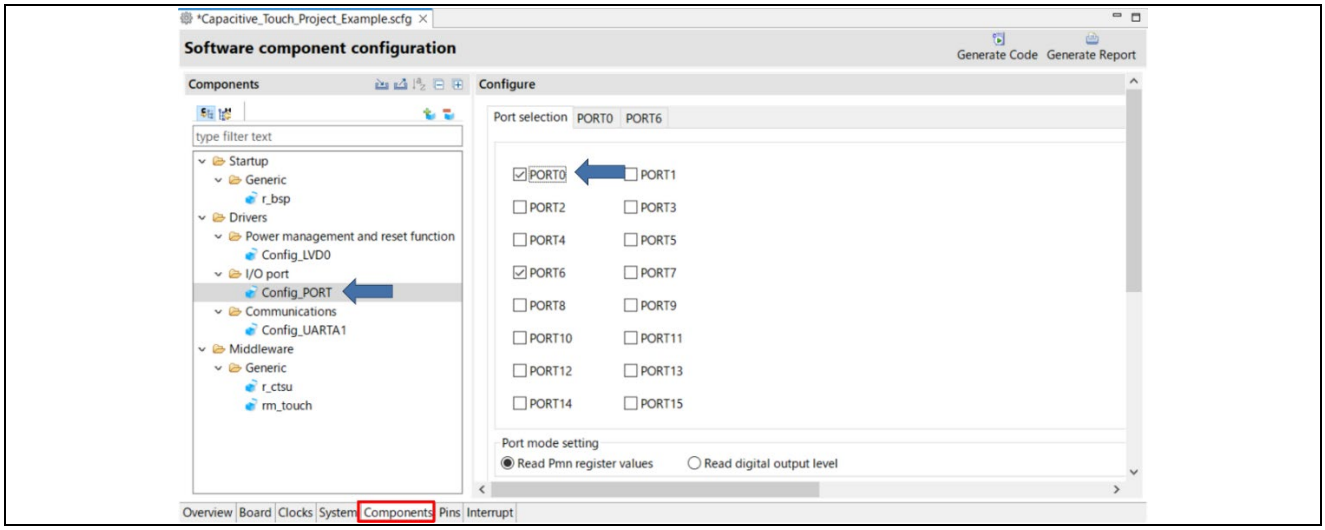


Figure 7-23 Setting the Port Module

2. Click on the [PORT0] tab and set "P07" as an output at the low level.

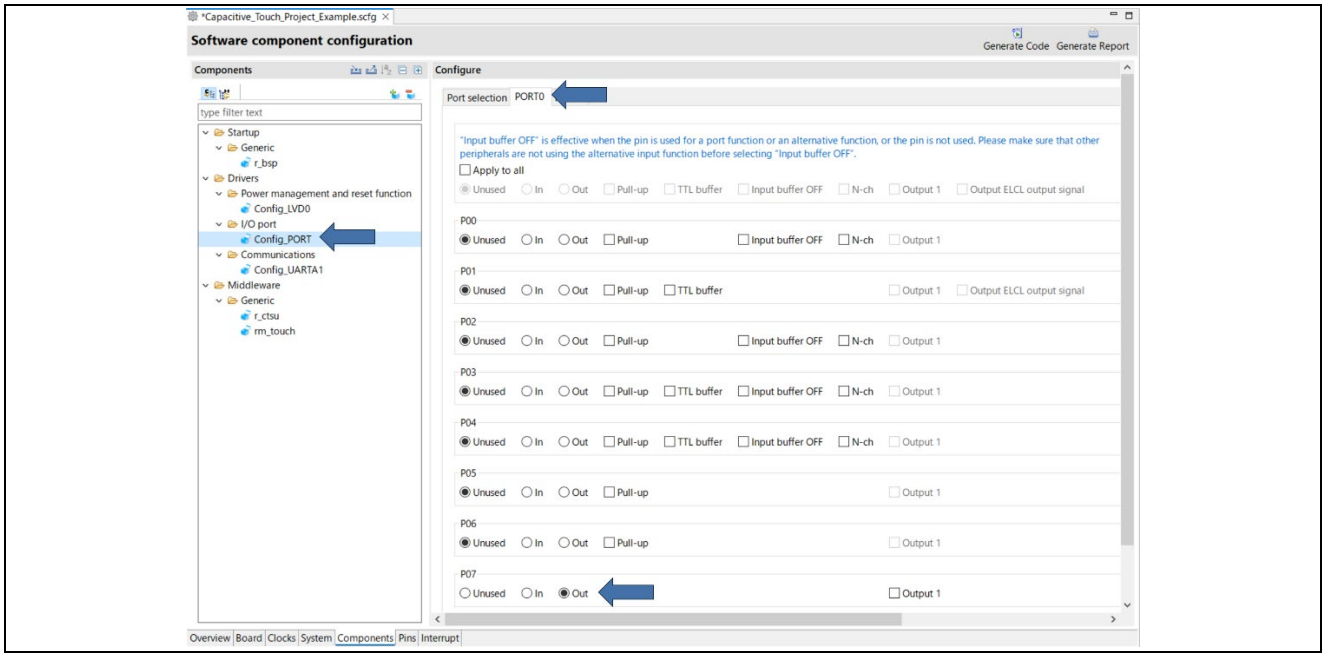


Figure 7-24 Setting P07 as an Output at the Low Level

7.7 Generating Code

Click on the icon of the Smart Configurator to generate code.

A cautionary message will appear before code generation. Ignore the message and proceed with generation.

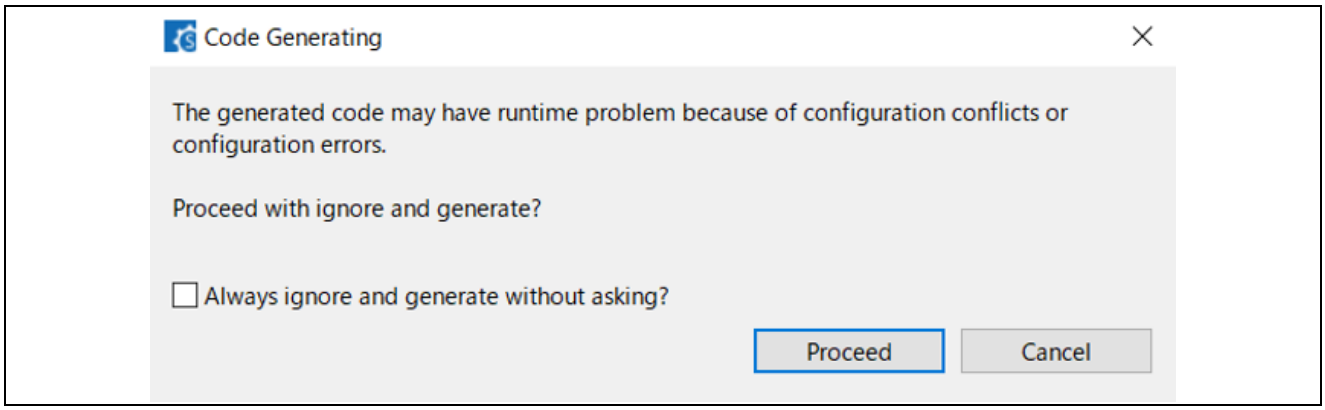


Figure 7-25 Cautionary Message before Code Generation

Also, 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.

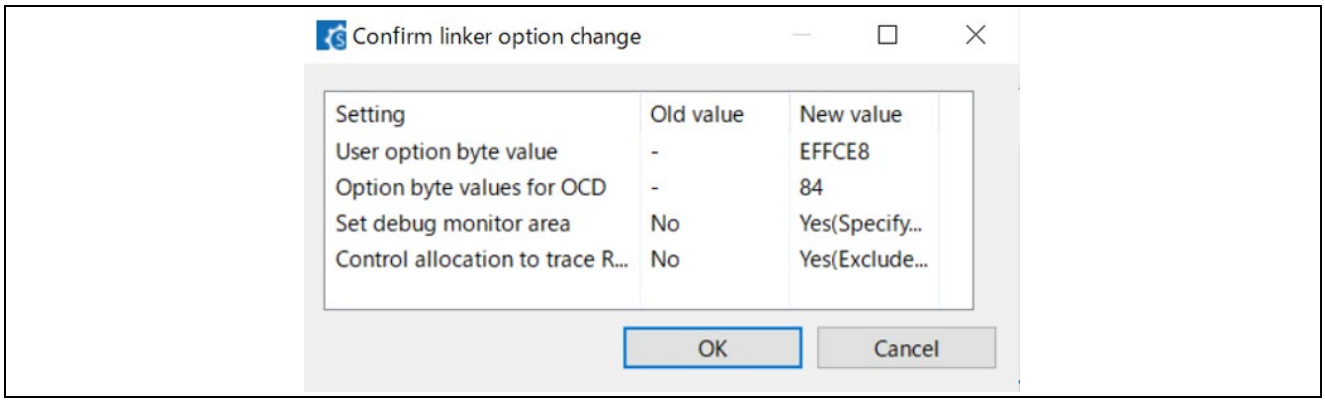


Figure 7-26 Message for Confirming Changes in Linker Options

8. Settings of QE for Capacitive Touch

8.1 Starting QE for Capacitive Touch

Start the standalone version of QE for Capacitive Touch (hereafter referred to as "the QE").

1. Start the QE from "QE-CapTouch (QE installation folder)/eclipse/qe-captouch.exe".
2. Figure 8-1 shows the window after the QE has started.

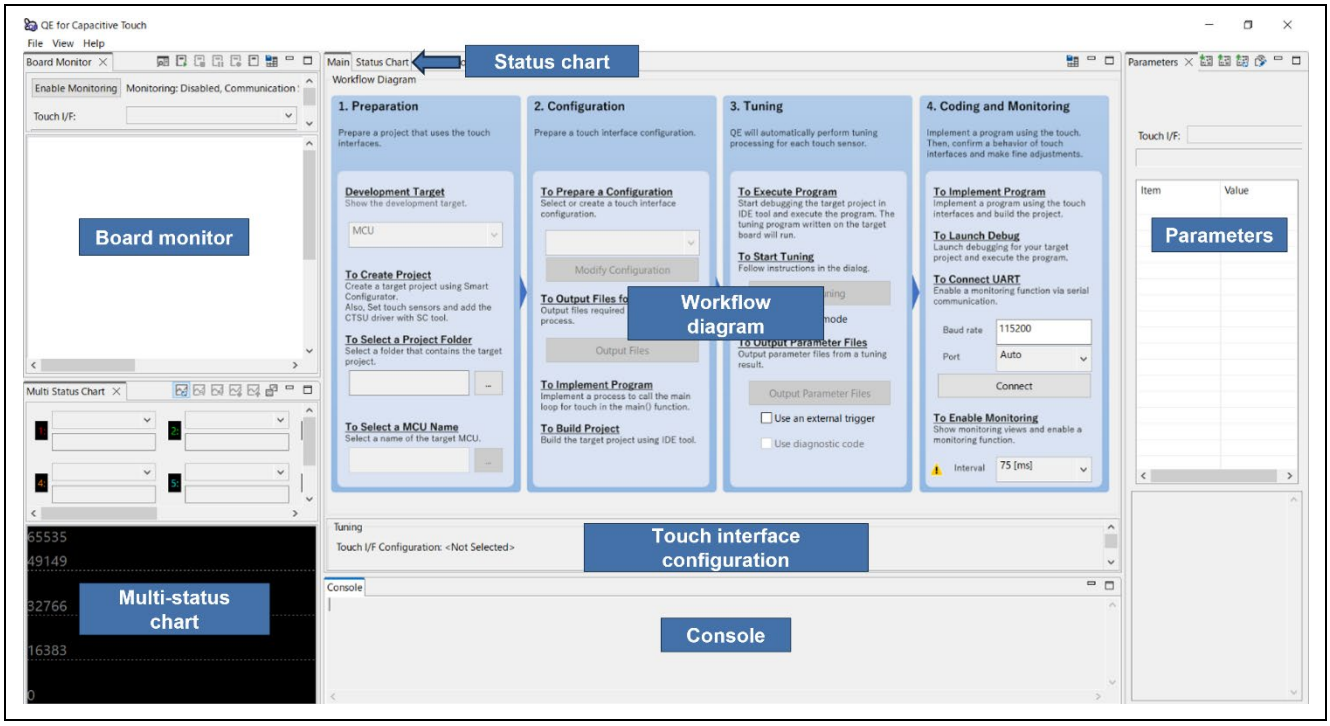


Figure 8-1 QE Window after Startup

If the layout is distorted in a full-screen display, change the "Scale and layout" for "Display" in the Windows settings to "100%".

8.2 Preparing a Project

Prepare a project that will use touch interfaces.

Set up the items under "Preparation" in the workflow diagram displayed across the middle of the QE window after startup.

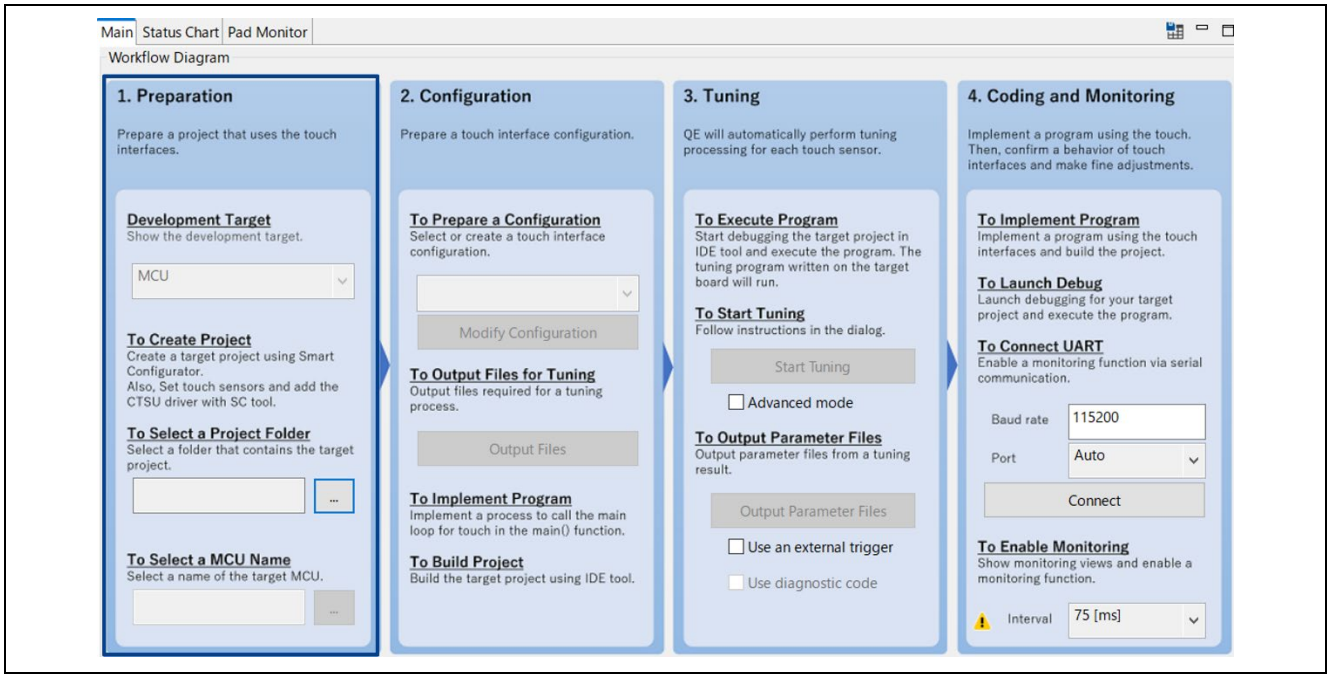


Figure 8-2 Workflow Diagram (Preparation)

1. Click on [...] under "To Select a Project Folder" and select the project folder that was created by CS+.
2. Click on [...] under "To Select an MCU Name" and select the target MCU to be used.

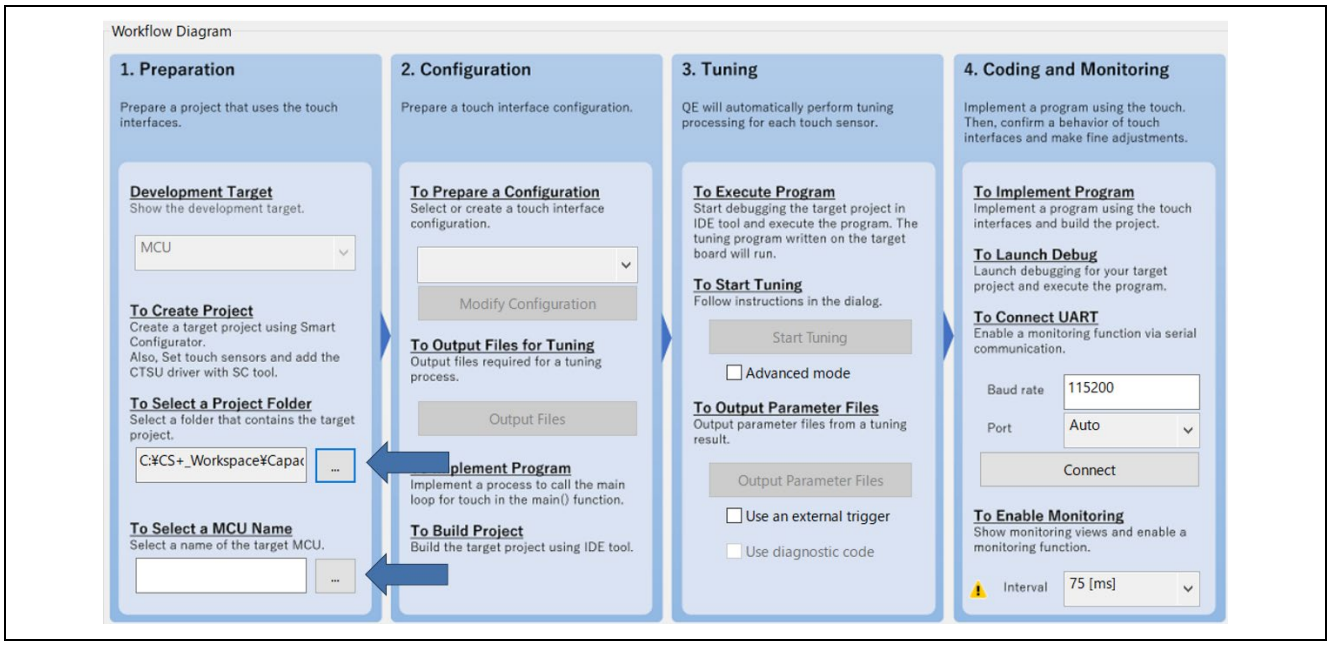


Figure 8-3 Preparing a Project

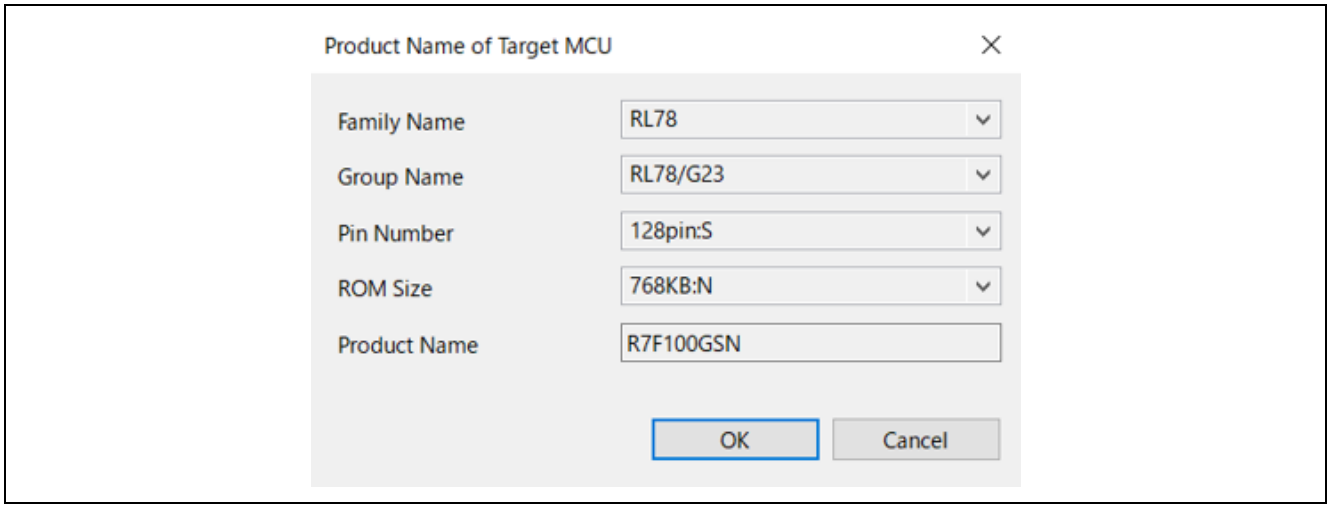


Figure 8-4 Selecting the Target MCU

If attempting to select an MCU causes the following error, it may indicate a problem with the location or pathname of the folder where the QE has been installed. Terminate the QE, move the installation folder to another location such as under "C:\Renesas", and then restart the QE.

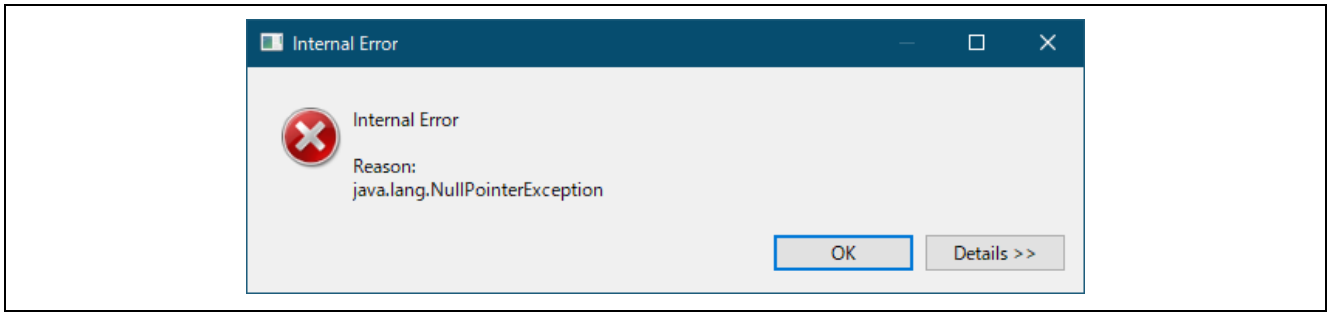


Figure 8-5 Error on Attempting to Select an MCU

8.3 Configuring the Touch Interface

Set up the items under "Configuration" in the workflow diagram.

Figure 8-6 Workflow Diagram (Configurarion)

1. Click on ⌄ under "To Prepare a Configuration" and select "Create a new configuration".

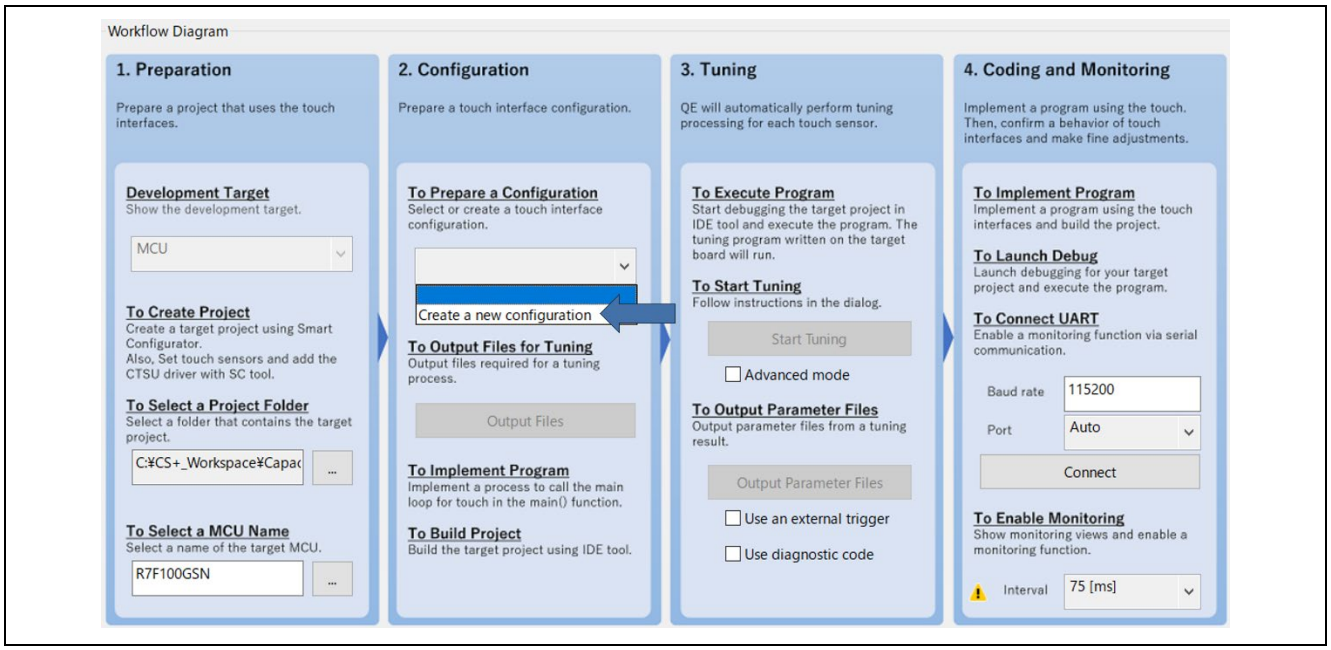


Figure 8-7 Creating a New Touch Interface Configuration

2. The [Create Configuration of Touch Interfaces] window will open and display the area for placing the elements of the touch interface.
   Click on [Button] in the [Touch I/F] panel on the right to change the cursor to the one for use in button placement. A button can then be placed by clicking on the area where it is to be placed.
   Place two buttons (Button00 and Button01) as shown below and press the [Esc] key to finish the button placement.
   Figure 8-8 shows the window after placement of the elements of the touch interface.

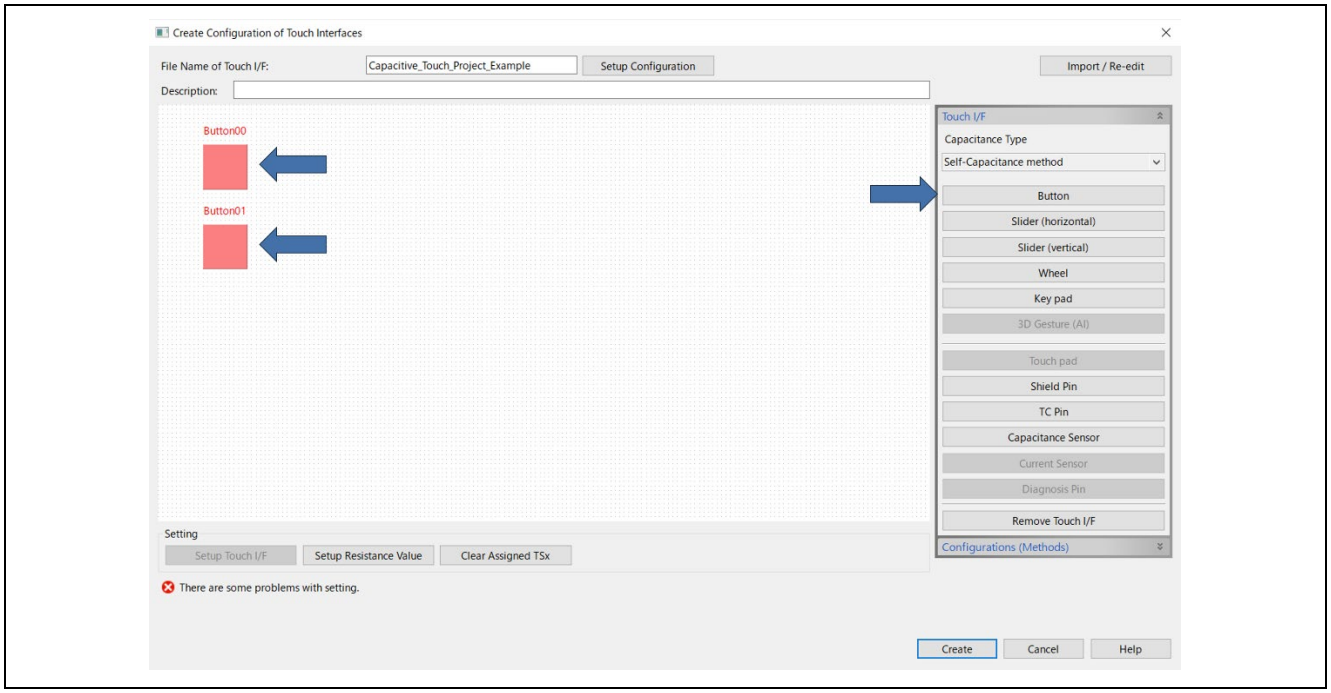


Figure 8-8 Placing Buttons

3. Assign names and touch sensors to the buttons.
   Double-click on [Button00] placed in the previous step and the [Setup Touch Interface] dialog box will open. Make the following settings in the dialog box.
   — Touch Sensor: TS06
   — Resistance [ohm]: 560
   For the resistance, refer to the user's manual or circuit diagram of the target board.

Caution: If TS pin numbers do not appear correctly in the dialog box for assigning a touch sensor to a button, the Windows settings require changing. Set the "Scale and layout" for "Display" to "100%" in the Windows system settings and then restart the QE for Capacitive Touch.

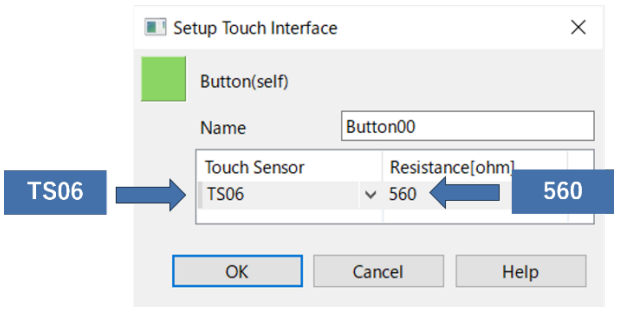


Figure 8-9 Setting a Touch-Interface Element

4. In the same way, make the following settings for [Button01].
   — Touch Sensor: TS05
   — Resistance [ohm]: 560

5. After the settings for the elements of the touch interface are complete, the color of the electrodes changes to green as shown in Figure 8-10. Click on [Create] at this point.

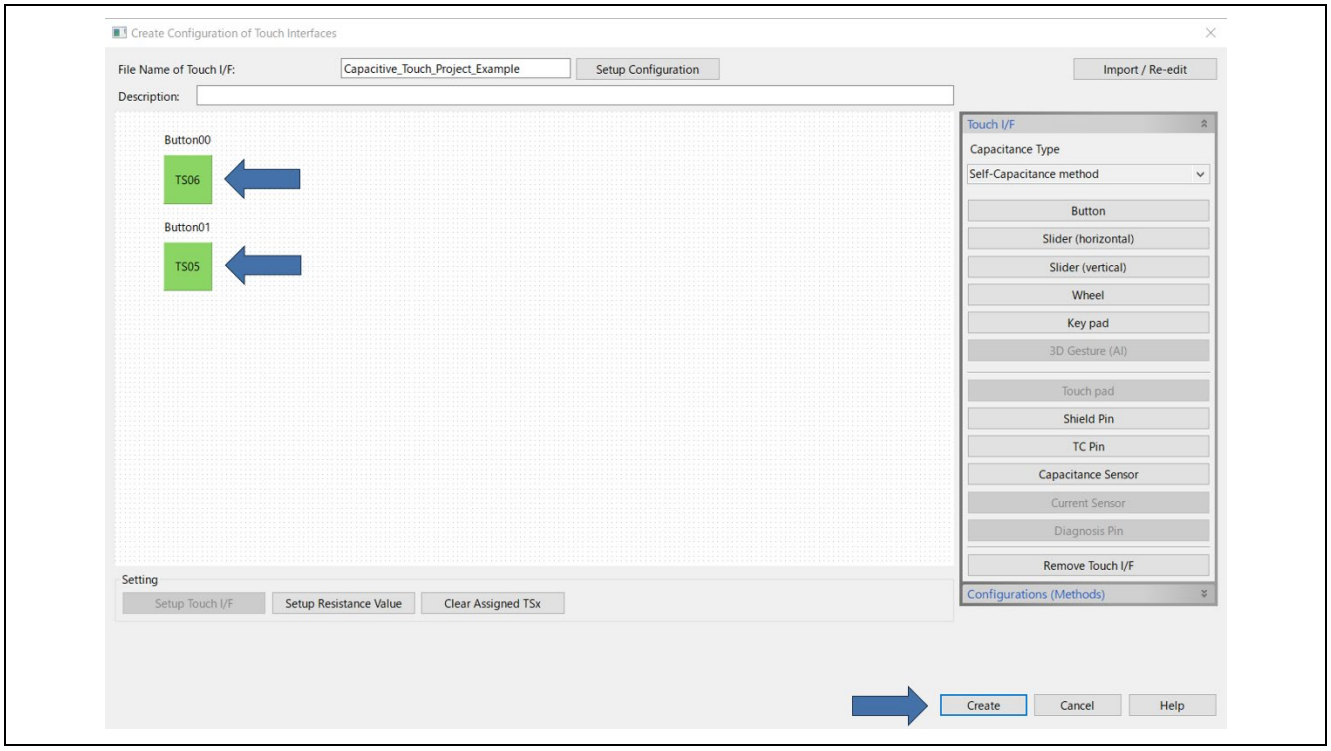
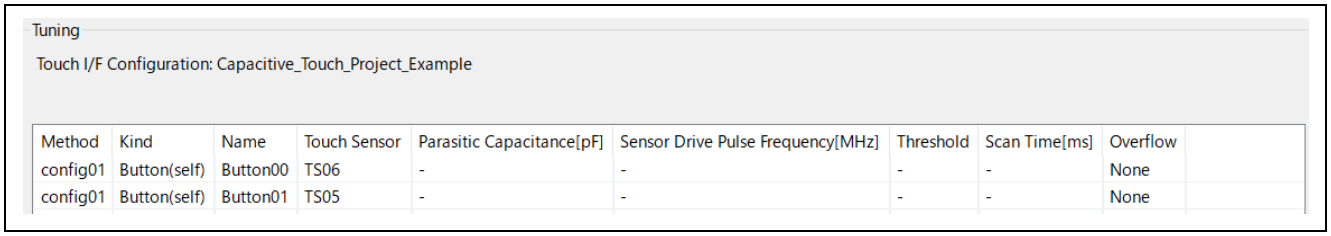


Figure 8-10 Touch Interface Configuration after the Settings are Complete

6. The touch interface configuration will be displayed in the [Tuning] panel.



Method	Kind	Name	Touch Sensor	Parasitic Capacitance[pF]	Sensor Drive Pulse Frequency[MHz]	Threshold	Scan Time[ms]	Overflow
config01	Button(self)	Button00	TS06	-	-	-	-	None
config01	Button(self)	Button01	TS05	-	-	-	-	None

Figure 8-11 Touch Interface Configuration Displayed in the [Tuning] Panel

7. Create a folder for storing the output files required for tuning. For this sample application, create a new folder "qe_gen" under "Capacitive_Touch_Project_Example/src". Click on [Output Files] in the QE workflow diagram and select the folder for storing the output files.

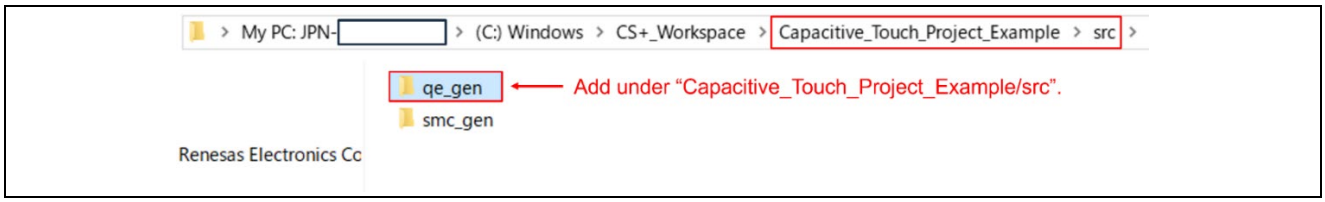


Figure 8-12 Creating a New Folder "qe_gen"

The following shows the place of the created folder in the directory structure and the files to be output.

```
Capacitive_Touch_Project_Example        ← CS+ project folder (Project name specified in
 |- src                                    chapter 6, Creating a new Project)
  |- smc_gen
  |- qe_gen                              ← New folder created
   |- qe_touch_config.c                  ← Output file
   |- qe_touch_config.h                  ← Output file
   |- qe_touch_define.h                  ← Output file
   |- qe_touch_sample.c                  ← Output file
```

8. Specify the frequency of the clock to be supplied to the CPU and peripheral hardware. After the folder for storing the output files has been selected, the following dialog box will open. Set the frequency (f_{CLK}) of the clock for the CPU and peripheral hardware and click on [OK].

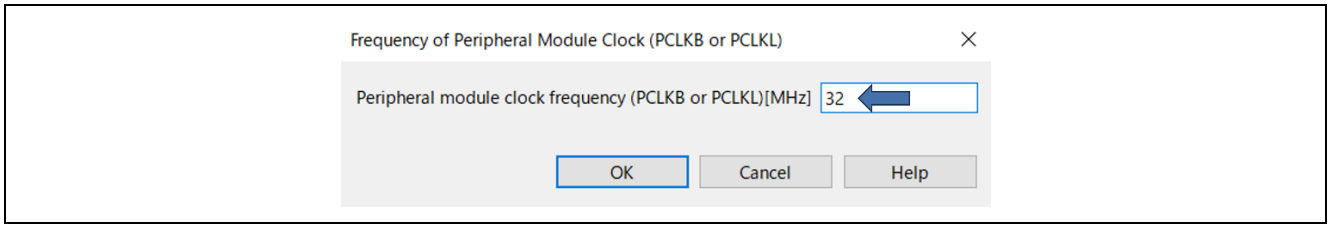


Figure 8-13 Setting the Frequency of the Peripheral Module Clock

9. Specify the voltage of the power supply for the MCU. After the [Power Supply Voltage of MCU] dialog box appears, specify the voltage and click on [OK].
   For the voltage to be specified, refer to the electrical characteristics of the target MCU.
   If the MCU uses EV_{DD}, enter the EV_{DD} value in the VDD field.

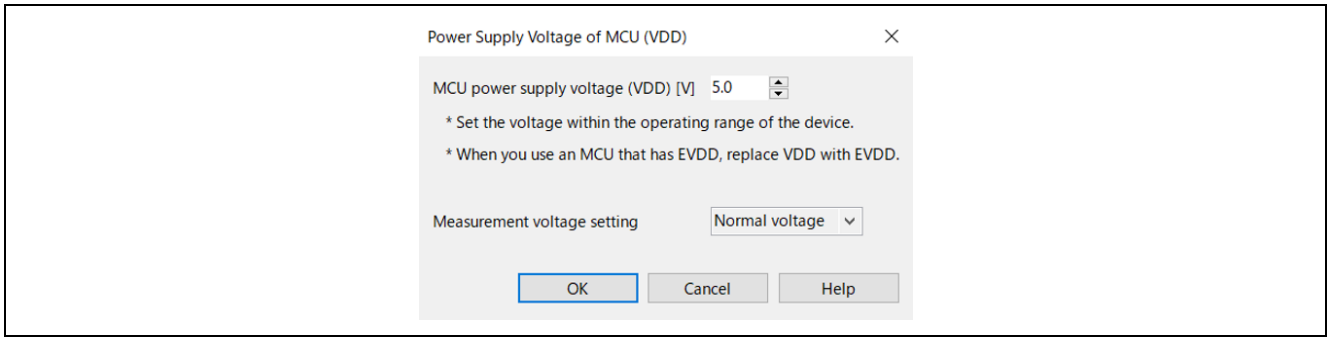


Figure 8-14 Setting the Power-Supply Voltage for the MCU

10. The [QE for Capacitive Touch] dialog box will open.
    The contents of this box are also displayed in the [Console] panel at the bottom of the QE window.

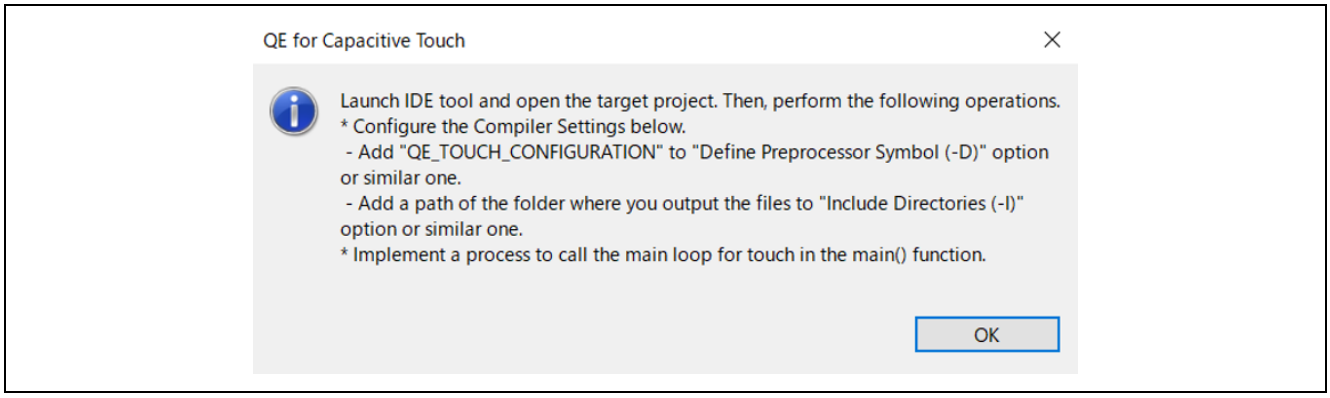


Figure 8-15 [QE for Capacitive Touch] Dialog Box

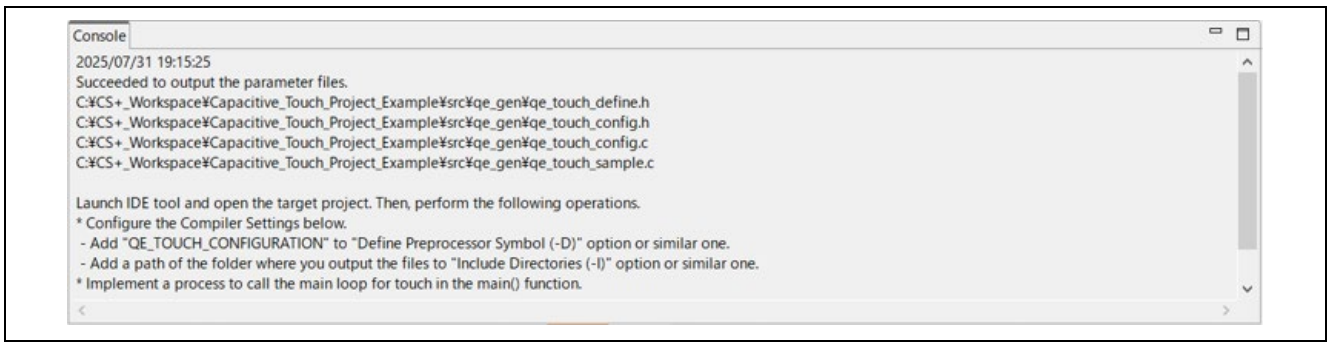


Figure 8-16 [Console] Panel

11. Set the compiler options. Open the CS+ window and select "CC-RL (Build Tool)" in [Project Tree]. In the [Common Options] tabbed page of [Property], select "Macro definition" under "Frequently Used Options (for Compile)" and click on [...] on the right side of the page.

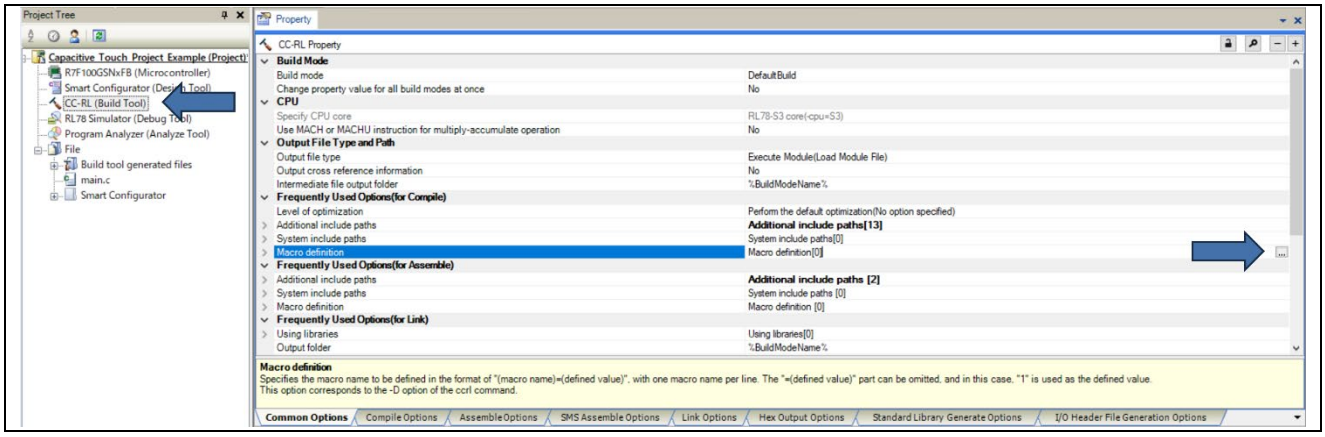


Figure 8-17 Selecting Macro Definition

12. The [Text Edit] dialog box will open. Enter "QE_TOUCH_CONFIGURATION" in the [Text] field in the dialog box and click on [OK].

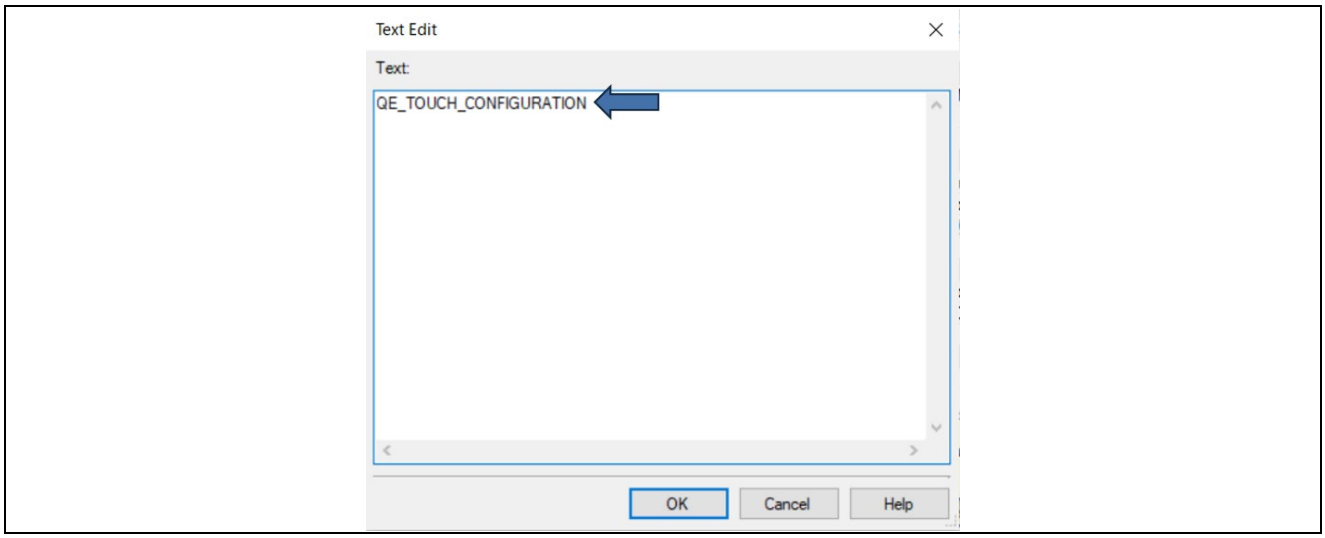


Figure 8-18 Editing a Macro Definition

13. Add the "qe_gen" folder to the sample project. Drag and drop the "qe_gen" folder from Explorer to the [Project Tree] of CS+.

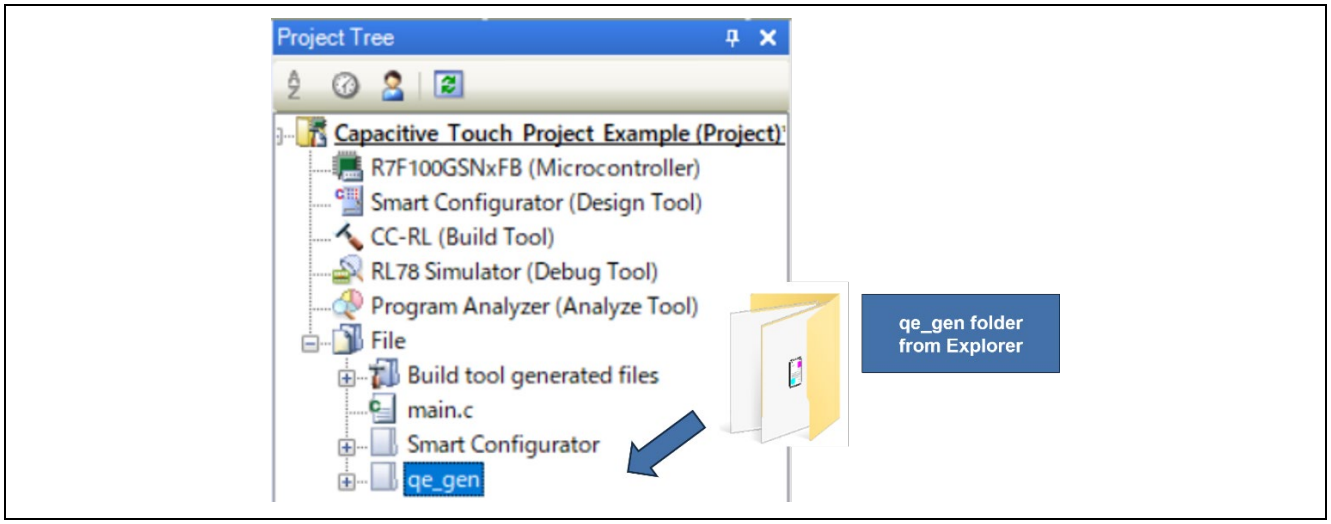


Figure 8-19 Adding the qe_gen Folder

14. Add the path to the "qe_gen" folder as an include path. On the [Common Options] tabbed page of [Property], select "Additional include paths" under "Frequently Used Options (for Compile)" and click on [...] on the right side of the page.
The [Path Edit] dialog box will open. Check that "src\qe_gen" has been added to the [Path] field in the dialog box and click on [OK].

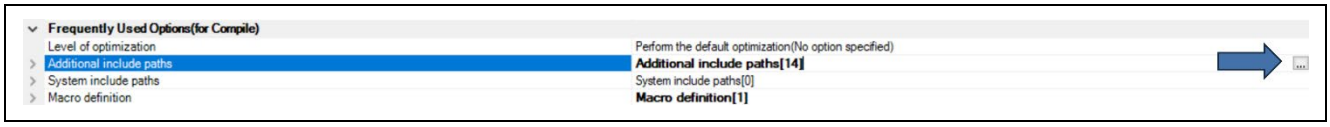


Figure 8-20 Adding an Include Path for the Compiler

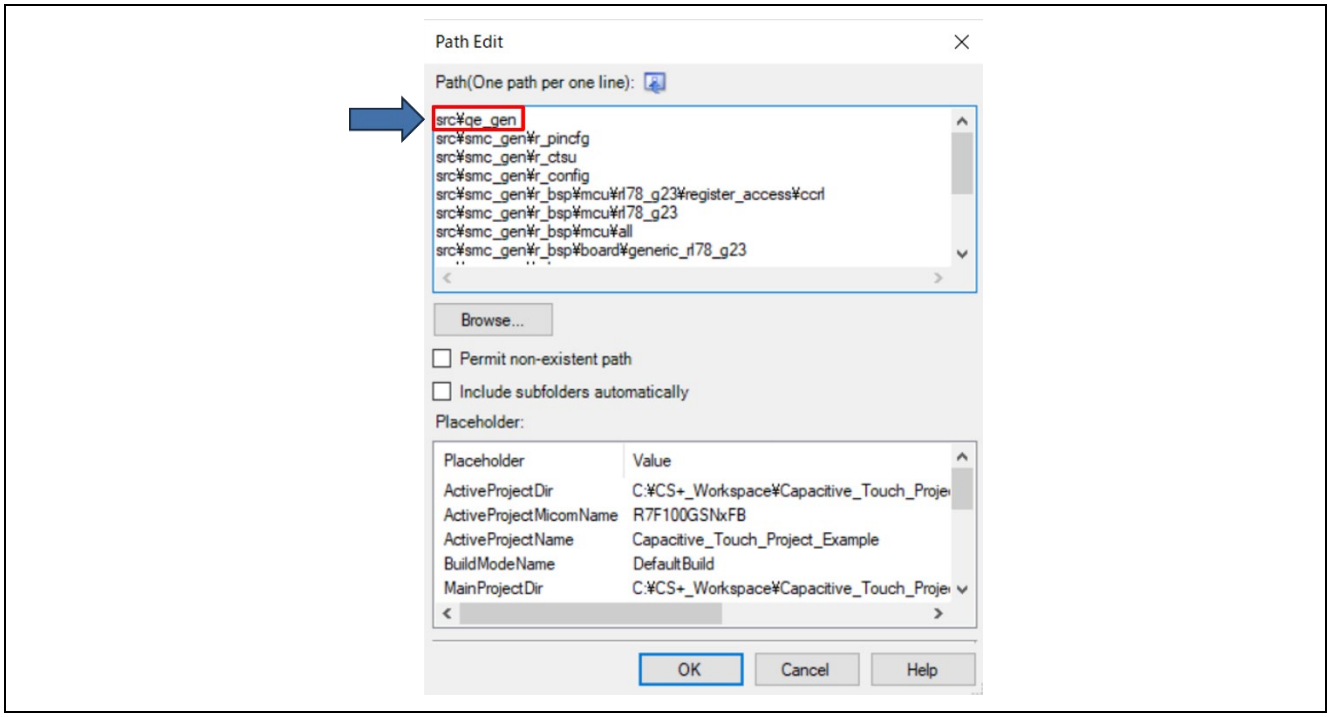


Figure 8-21 Checking Addition of the Include Path

15. Specify the language variant of the C source files.
Click on "Source" on the [Compile Options] tabbed page and click on "Language of the C source file".
Click on ⌄ on the right side of the page and select "C99 (-lang=c99)".

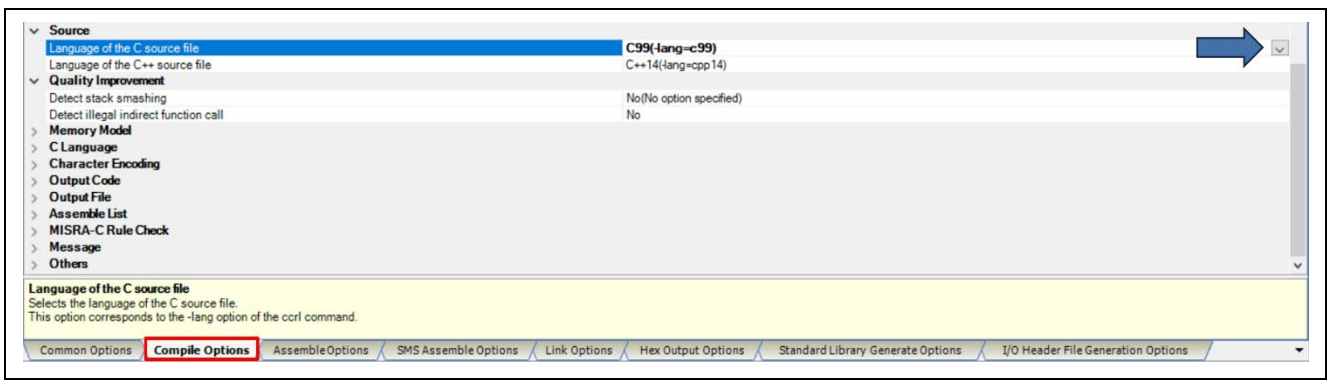


Figure 8-22 Selecting the C-Language Standard

16. Set the on-chip debugging (OCD) option byte and user option bytes.
Click on "Device" on the [Link Options] tabbed page and make the following specifications. For meanings of the values of the option bytes, refer to the user's manual of the target MCU.

— Option byte value for OCD: 84
— Set debug monitor area: Yes (Specify address range) (-DEBUG_MONITOR=<Address range>)
— User option byte value: EFFCE8

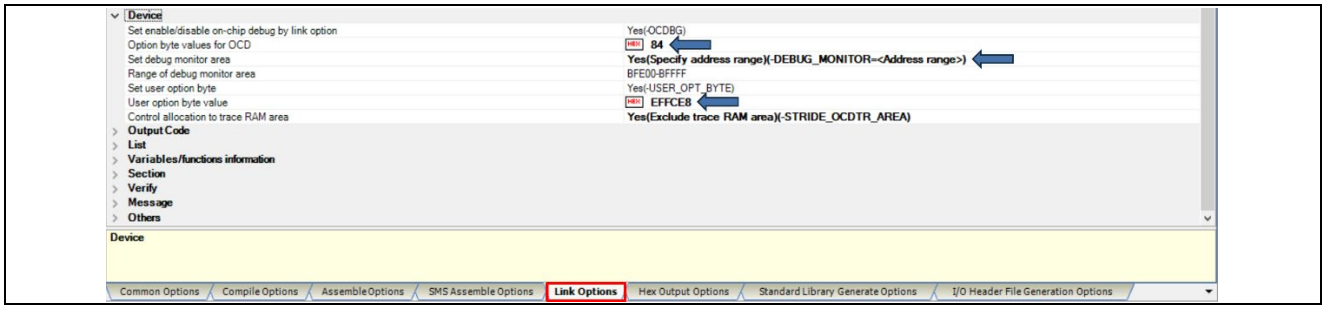


Figure 8-23 Setting the Option Bytes

17. If the free evaluation edition of CC-RL V1.12.00 or a later version is to be used for compilation, "Debug precedence (-Onothing)" should be selected as the level of optimization by the compiler before building.
Double-click on “Optimization” on the [Compile Options] tabbed page and select "Debug precedence (-Onothing)" for "Level of optimization”.

Remark: This setting is only necessary for tuning. After tuning is complete, any optimization level can be specified.

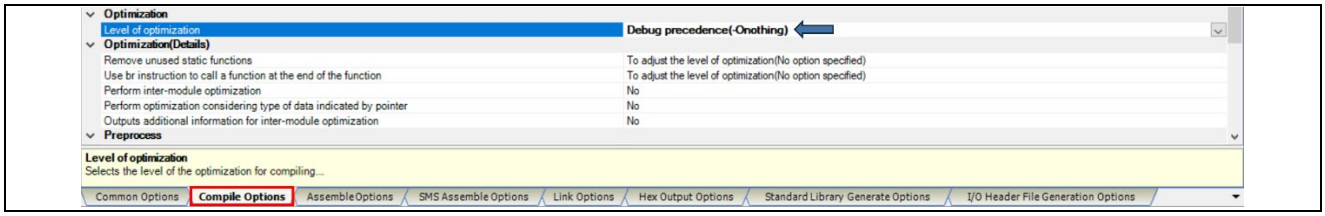


Figure 8-24 Setting the Level of Optimization by the Compiler

18. Implement the processing for calling the main function of the touch measurement processing in the main() function.
This requires a call of the qe_touch_main() function from the main() function.
Add the following statements to the "main.c" file at the points indicated in the listing below.
- extern void qe_touch_main(void);
- qe_touch_main();

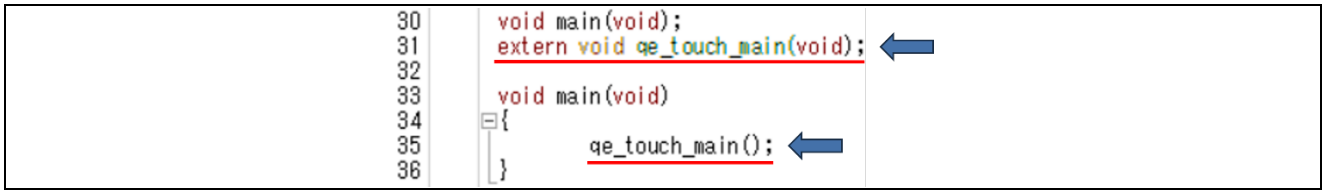


Figure 8-25 main.c

19. Add the function for serial communications to the "Config_<UART channel>_user.c" file.
    Add the following statements to the "main.c" file at the points indicated in the listing below.
    - extern void touch_uart_callback(uint16_t event);
    - touch_uart_callback(0);
    - touch_uart_callback(1);

```
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 */


73    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    }

86    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 8-26 Config_UARTA1_user.c

20. Build the project in CS+. Click on the icon under the menu bar of CS+ to start the process of building. Check that the build process has been completed without any errors or warnings.

    If the following warning (W0511187) is generated during building, change the level of optimization by the compiler to "Debug precedence (-Onothing)" as shown in Figure 8-24 and rebuild the project.

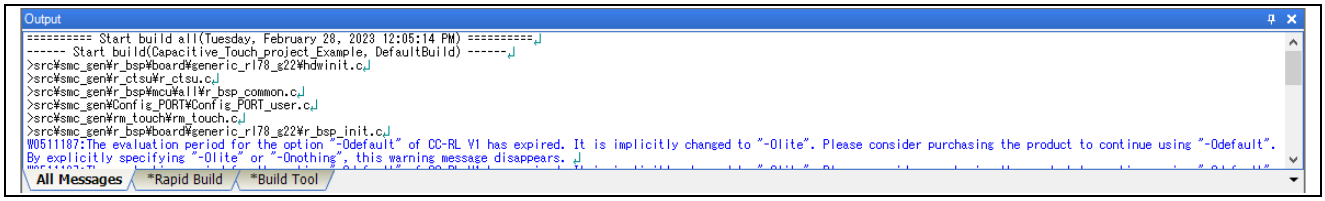


Figure 8-27 Warning during Building (W0511187)

8.4 Tuning

Set up the items under "Tuning" in the workflow diagram.

Figure 8-28 Workflow Diagram (Tuning)

1. Select the debugging tool to be used. Right-click on "Debug Tool" in [Project Tree] of CS+ and select "RL78 E2 Lite(E)" from [Using Debug Tool].

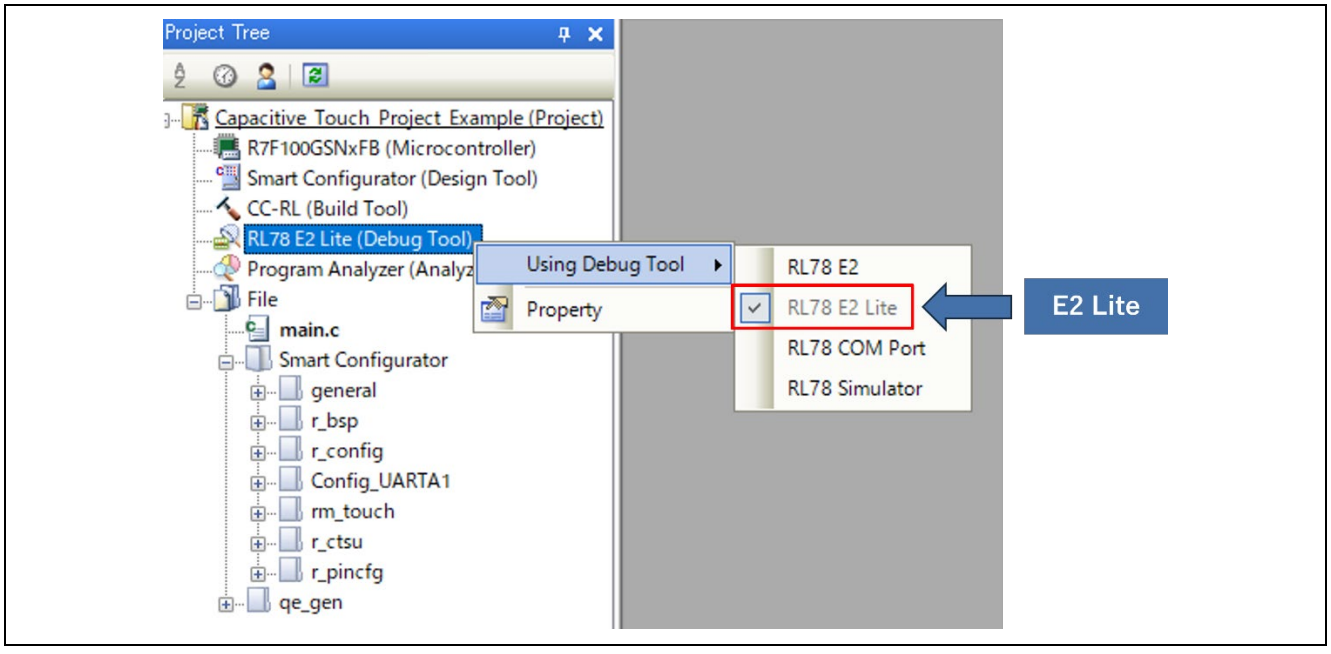


Figure 8-29 Selecting the Debugging Tool

2. Set “Power target from the emulator” to “No” in property of “Debug Tool”.

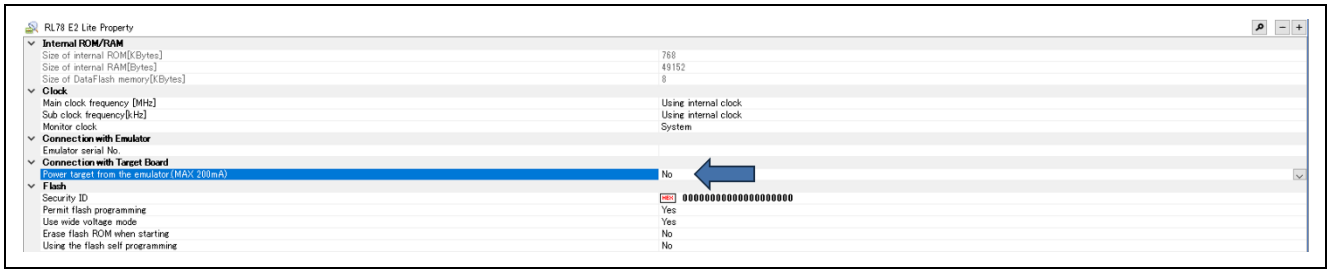


Figure 8-30 Property of the Debugging Tool

3. Execute the serial connection function of the QE. Confirm connection between PC and target board and build the project and write the program by clicking on the CS+ icon. After finishing downloading the program, click icon to execute the program.

4. On QE, set “Baud rate” of “To Connect UART” to the value which is set in chapter 7.5.3

Figure 8-31 Setting "Baud rate"

5. Click on [Start Tuning] to start automatic tuning.

Figure 8-32 Automatic Tuning

6. A dialog box will open. Specify "Baud rate" in the dialog box and click on [Connect].

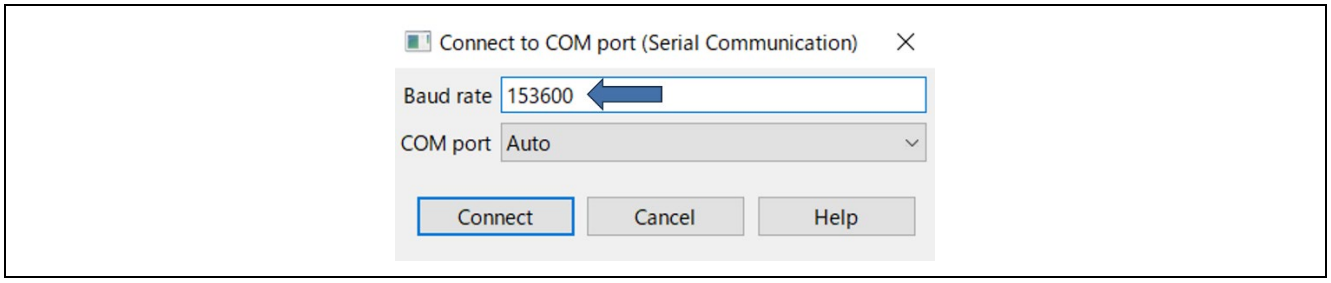


Figure 8-33 Setting "Baud rate"

7. In the dialog box opened in response to the previous step, specify the frequency of the clock for the CPU and peripheral hardware and click on [OK].

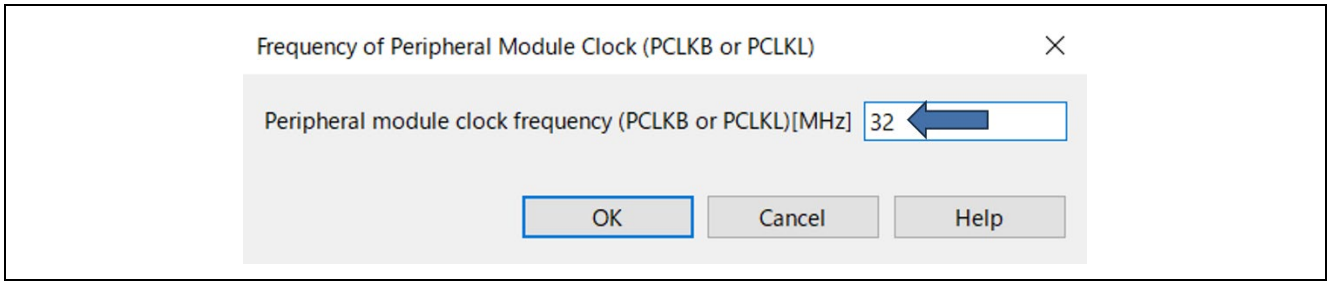


Figure 8-34 Setting the Frequency of the Peripheral Module Clock

8. Automatic tuning will begin. Check the messages in the [Automatic Tuning Processing] dialog box, which guides the user through the steps of tuning, and follow the instructions for proceeding through the steps.

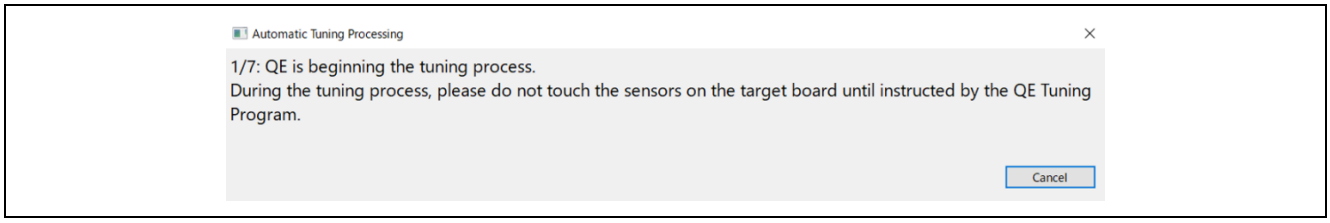


Figure 8-35 [Automatic Tuning Processing] Dialog Box

After several steps of the tuning process are complete, the dialog box shown in the figure below will open. Measure the touch sensitivity at this point. Use normal pressure in touching the Button01 touch sensor as instructed by the dialog box. While the touch sensor is being touched, the bar in the dialog box will be extended to the right and the numerical value of touch counting will increase. While continuing to touch the sensor with your finger, press any key on the keyboard of the PC to confirm the measured value.

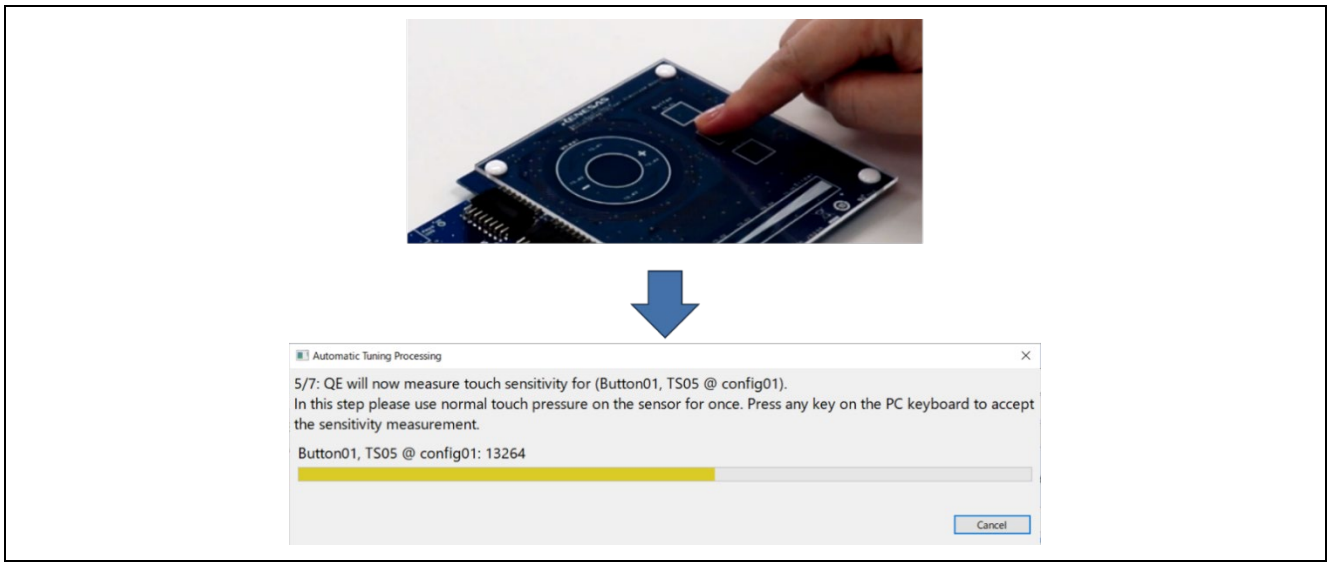


Figure 8-36 Measuring the Touch Sensitivity

9. Proceed with measurement for the other button touch sensor in the same way.

10. The following dialog box will open after tuning is complete and you can check the threshold values. These threshold values are used by the middleware to determine whether touch events have occurred. Click on [Continue the Tuning Process]. This is the end of the steps for automatic tuning.

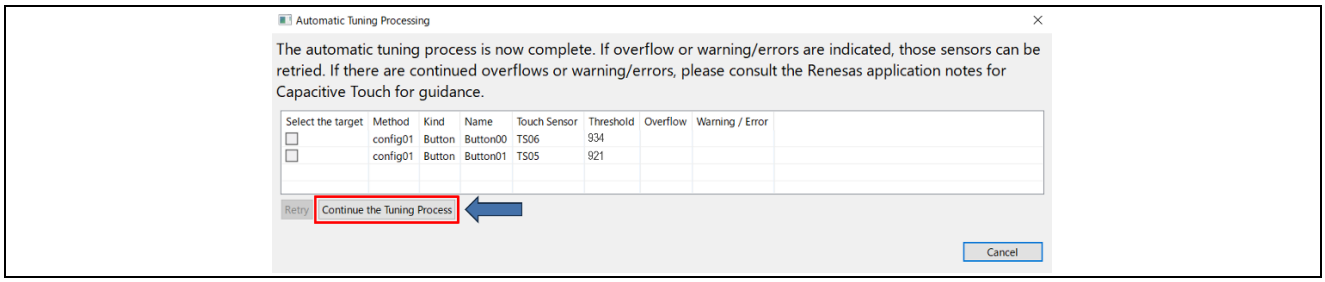


Figure 8-37 Threshold Values for the Touch Sensors

11. Click “Output Parameter Files” and output parameter files including result of tuning. Choose “qe_gen” folder created at chapter 8.3 as the folder for output files and overwrite the files.
    The output files are same as the following files that is outputted at “Output files” of chapter 8.3.
    |- qe_touch_config.c ← Output File
    |- qe_touch_config.h ← Output File
    |- qe_touch_define.h ← Output File
    |- qe_touch_sample.c ← Output File

Figure 8-38 Output of the Parameter Files

13. On CS+, click on the icon to stop the program and then click on the icon for disconnection from the debugging tool.

8.5 Coding and Monitoring

8.5.1 Monitoring

Monitor the touch interface operation by proceeding through the steps under "Coding and Monitoring" in the workflow diagram.

Figure 8-39 Workflow Diagram (Coding and Monitoring)

1. Build the project and write the program by clicking on the CS+ icon. After finishing downloading the program, click icon to execute the program.

2. Click on [Connect] for serial connection with the target board. [Connect] shown in the red frame in Figure 8-40 will change to [Disconnect].

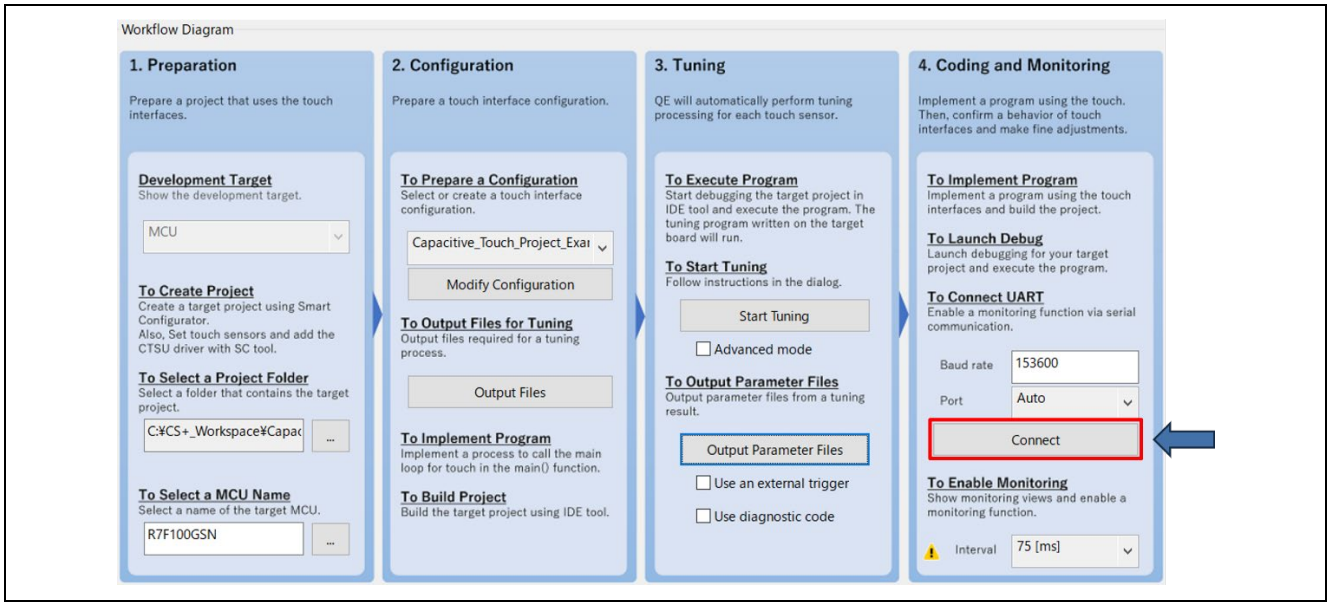


Figure 8-40 Serial Connection

3. Click on [Enable Monitoring] in the [Board Monitor] panel in the top left part of the QE window. The indication "Monitoring: Disabled" will change to "Monitoring: Enabled".

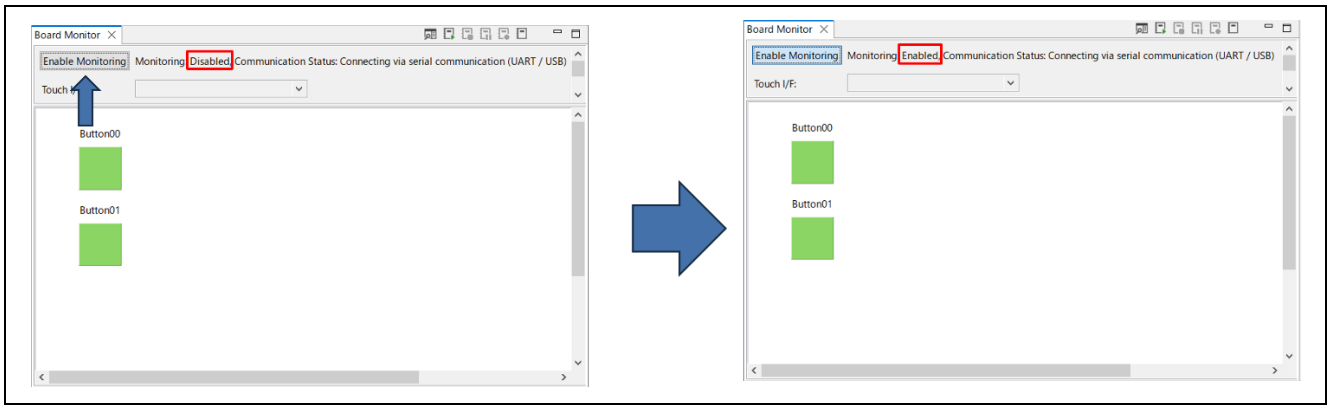


Figure 8-41 Enabling Monitoring

4. When touching of a touch sensor is detected, the state of touching is indicated by a finger icon.

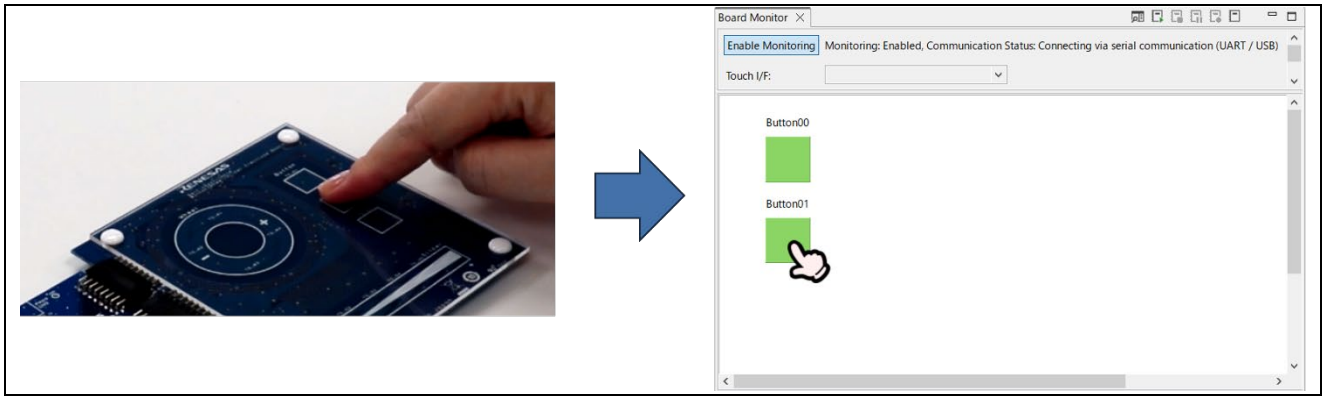


Figure 8-42 Display of the State of a Touch Sensor being Touched

5. Display a graph of the touch counting value in the status chart.
   A. Click on the [Status Chart] tab.

   B. Click on for "Touch I/F" on the opened [Status Chart] page and select a touch sensor of the touch interface configuration.

   The chart shows the real-time value of touch counting as it is being measured. The change in the touch counting value while the touch sensor is being touched can be confirmed on the chart.
   The green line shows the threshold value, which is used by the "rm_touch" middleware to judge whether operation of the touch sensor is in progress; that is, it is being touched.
   The red strip at the bottom of the chart shows the duration over which the touch counting value exceeds the threshold value; that is, touching is being detected.

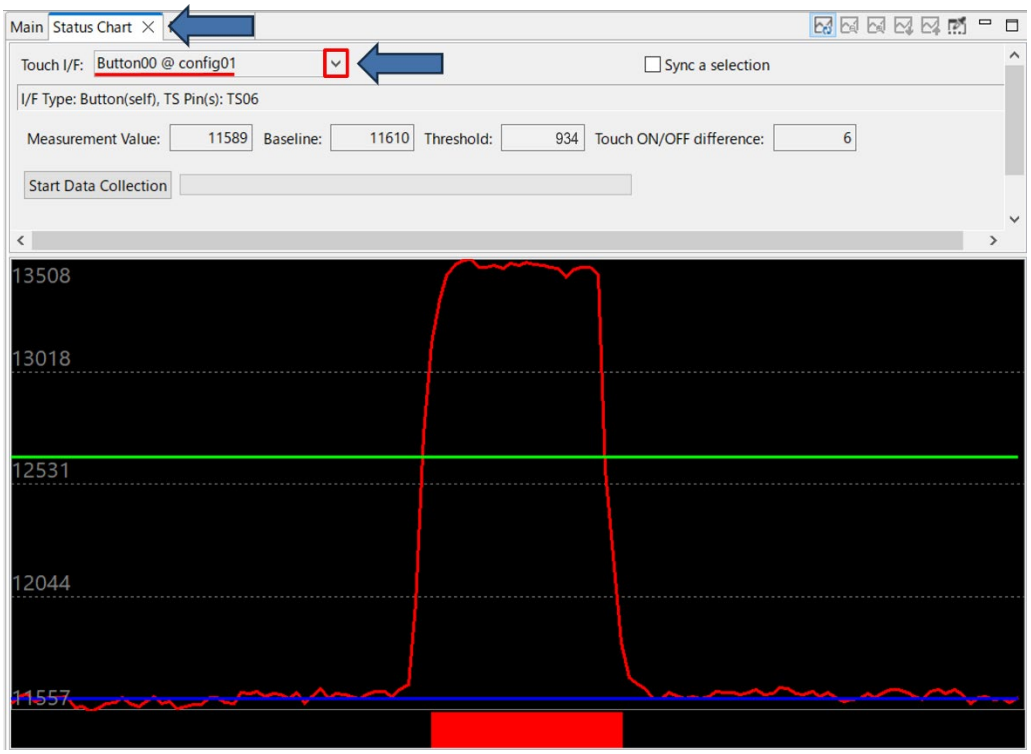


Figure 8-43 Graph of the Touch Counting Value

6. Measure the signal-to-noise ratio (SNR) values as required.
   A. Click on [Start Data Collection] on the [Status Chart] page.

Figure 8-44 Collecting Data in the Touch-off State

   B. 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.

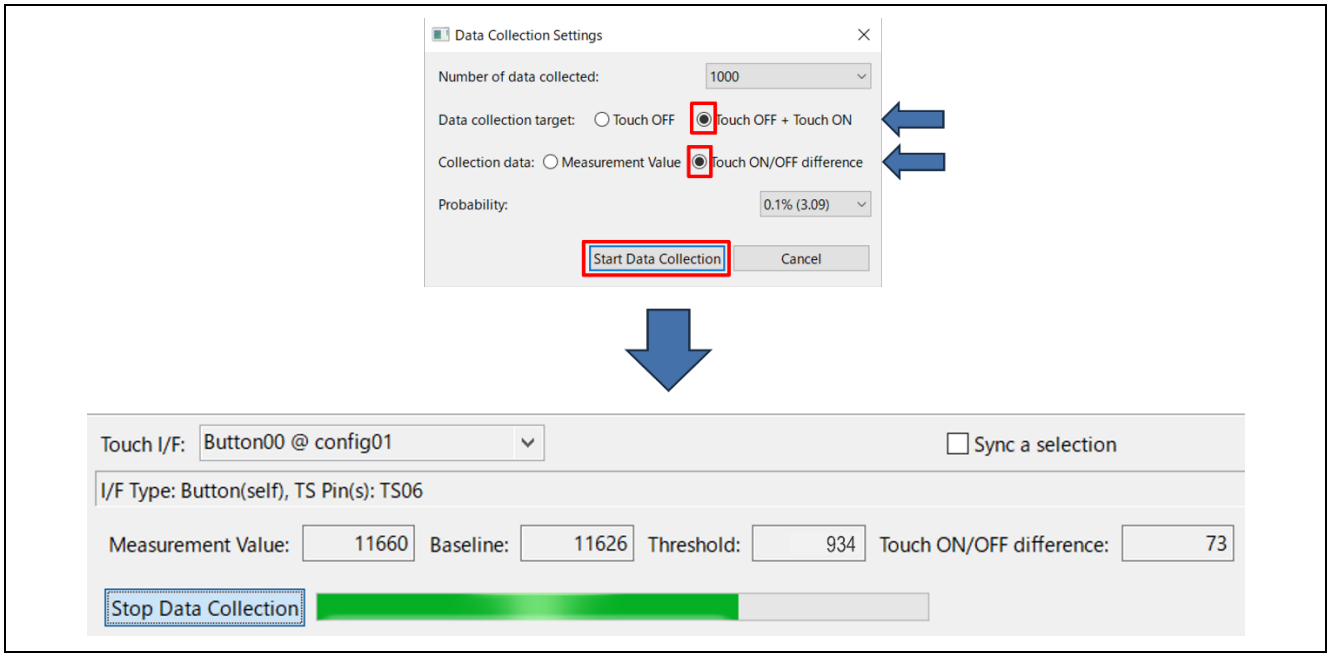


Figure 8-45 Starting Data Collection

C. 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.

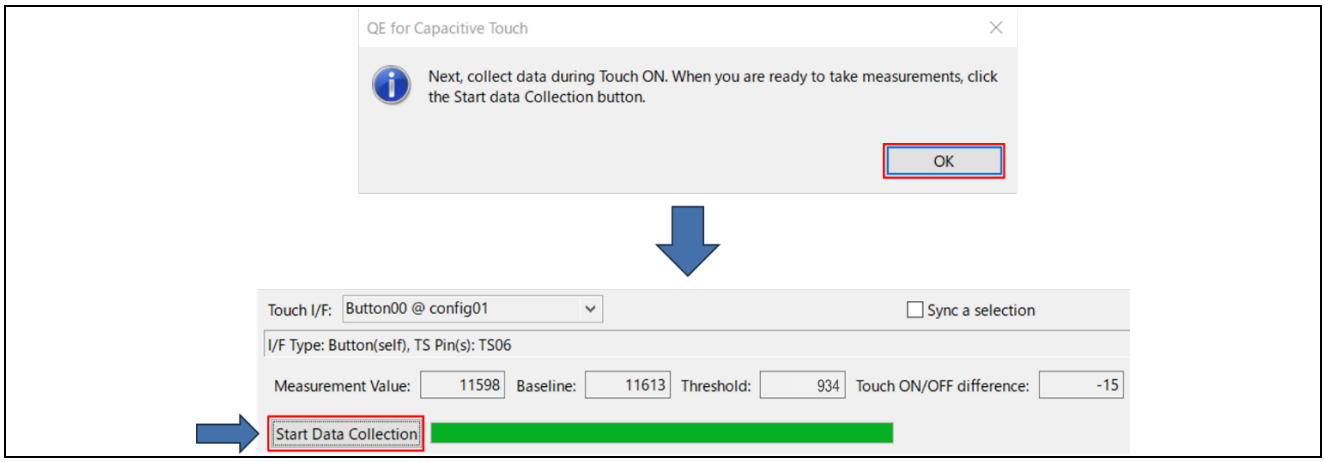


Figure 8-46 Starting Data Collection in the Touch-on State

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

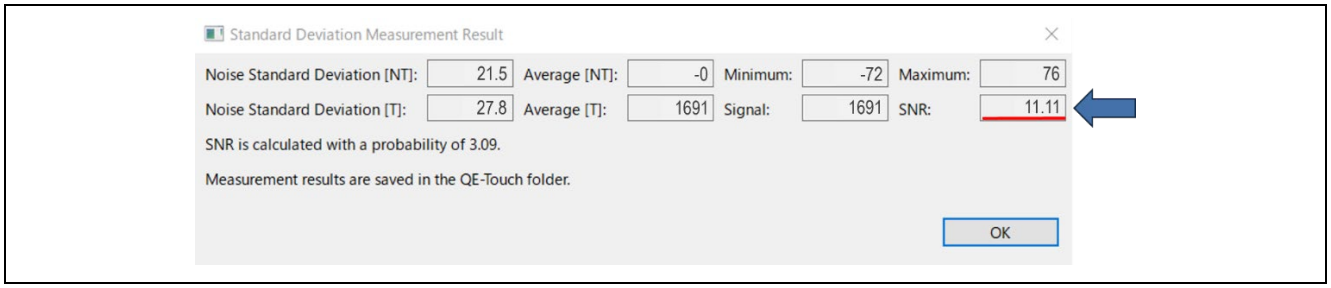


Figure 8-47 SNR Value

7. 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 [Multi Status Chart] tabbed page in the lower left part of the QE window.

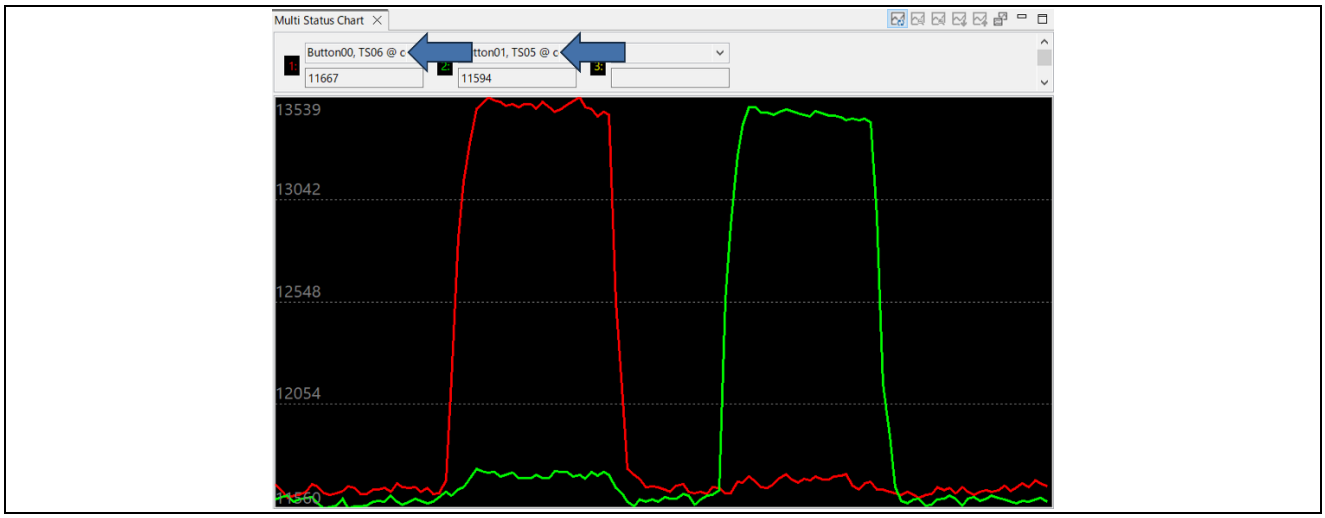


Figure 8-48 Multi-Status Chart

8. Manually adjust parameters as required.
   Use the [Parameters] panel in the right part of the QE window.

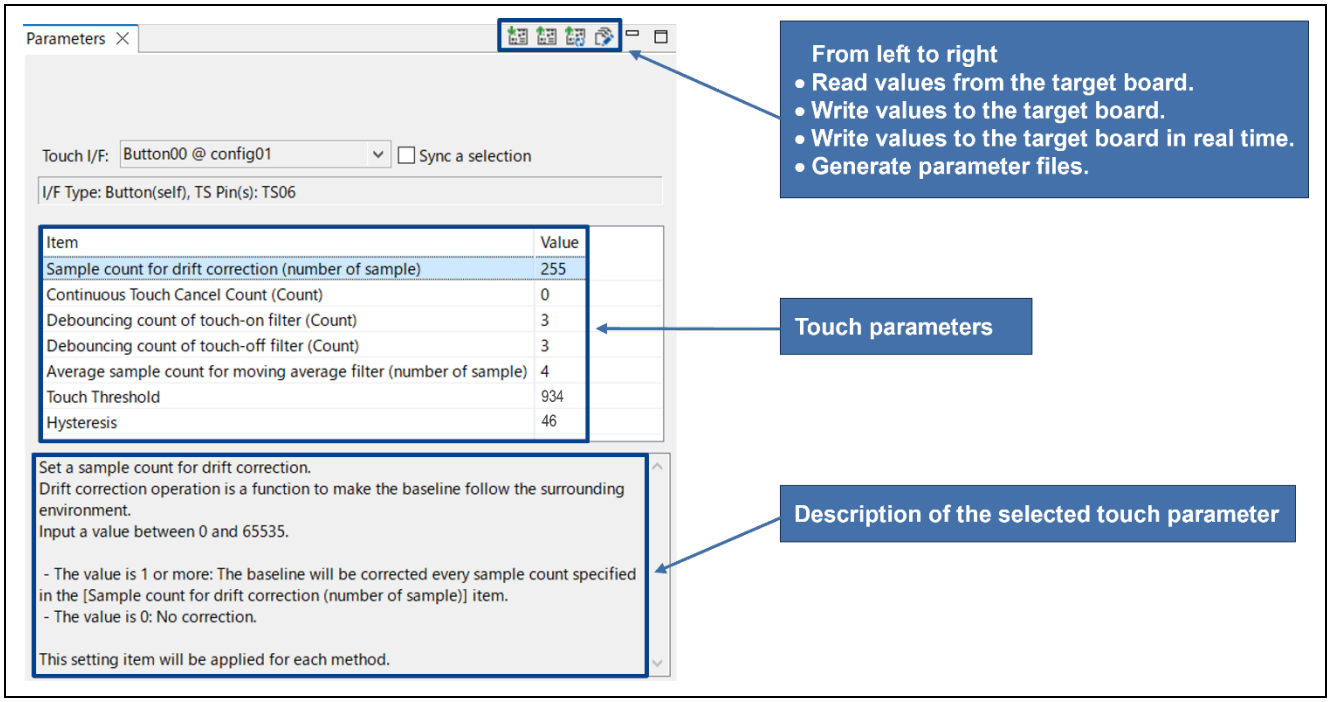


Figure 8-49 Adjusting Parameters

9. While "Monitoring: Enabled" is being displayed, click on [Enable Monitoring] to stop monitoring.

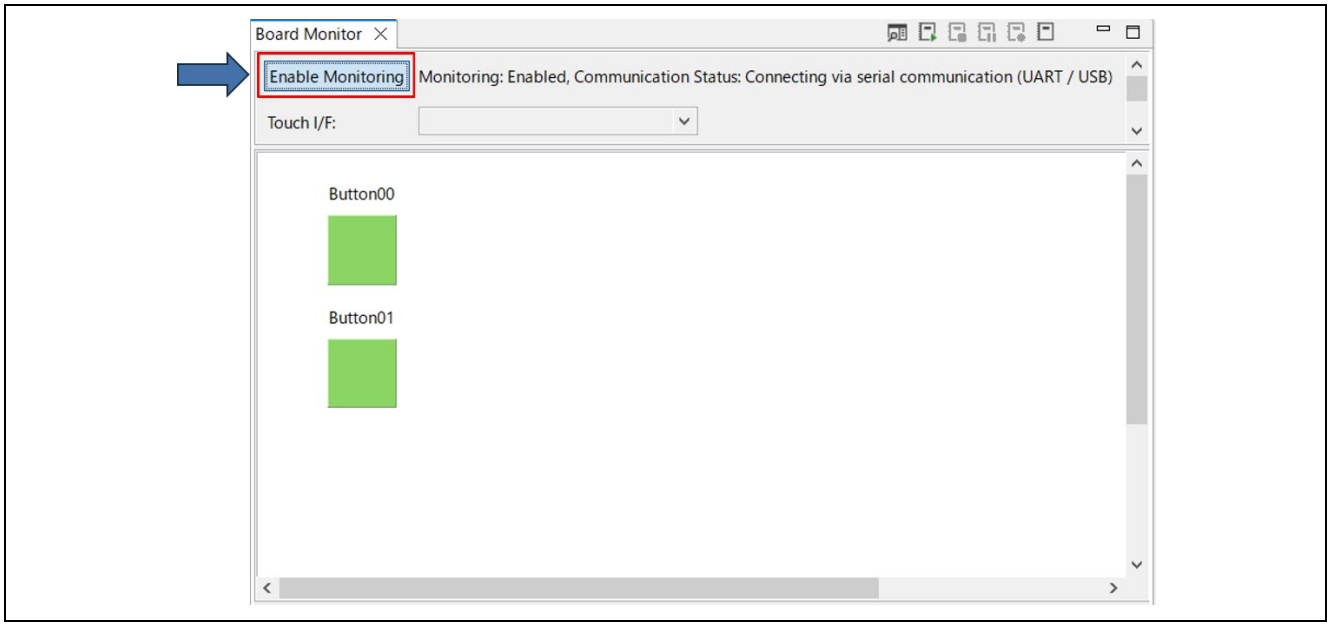


Figure 8-50 Termination of Monitoring

10. Click on [Disconnect] to disconnect the serial connection.

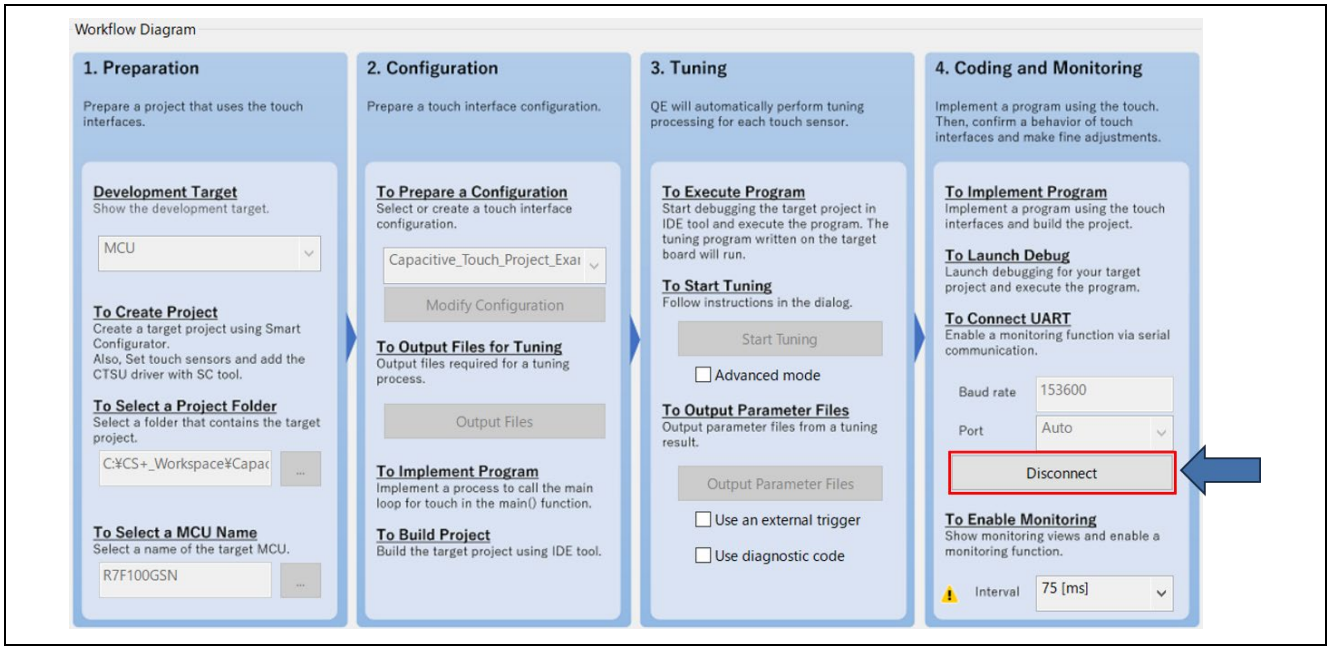


Figure 8-51 Disconnecting the Serial Connection

11. On CS+, click on the icon to stop the program and then click on the icon for disconnection from the debugging tool.

8.6 Flowchart (Software Timer)

Figure 8-52 is a flowchart of the touch measurement control processing with the use of a software timer.

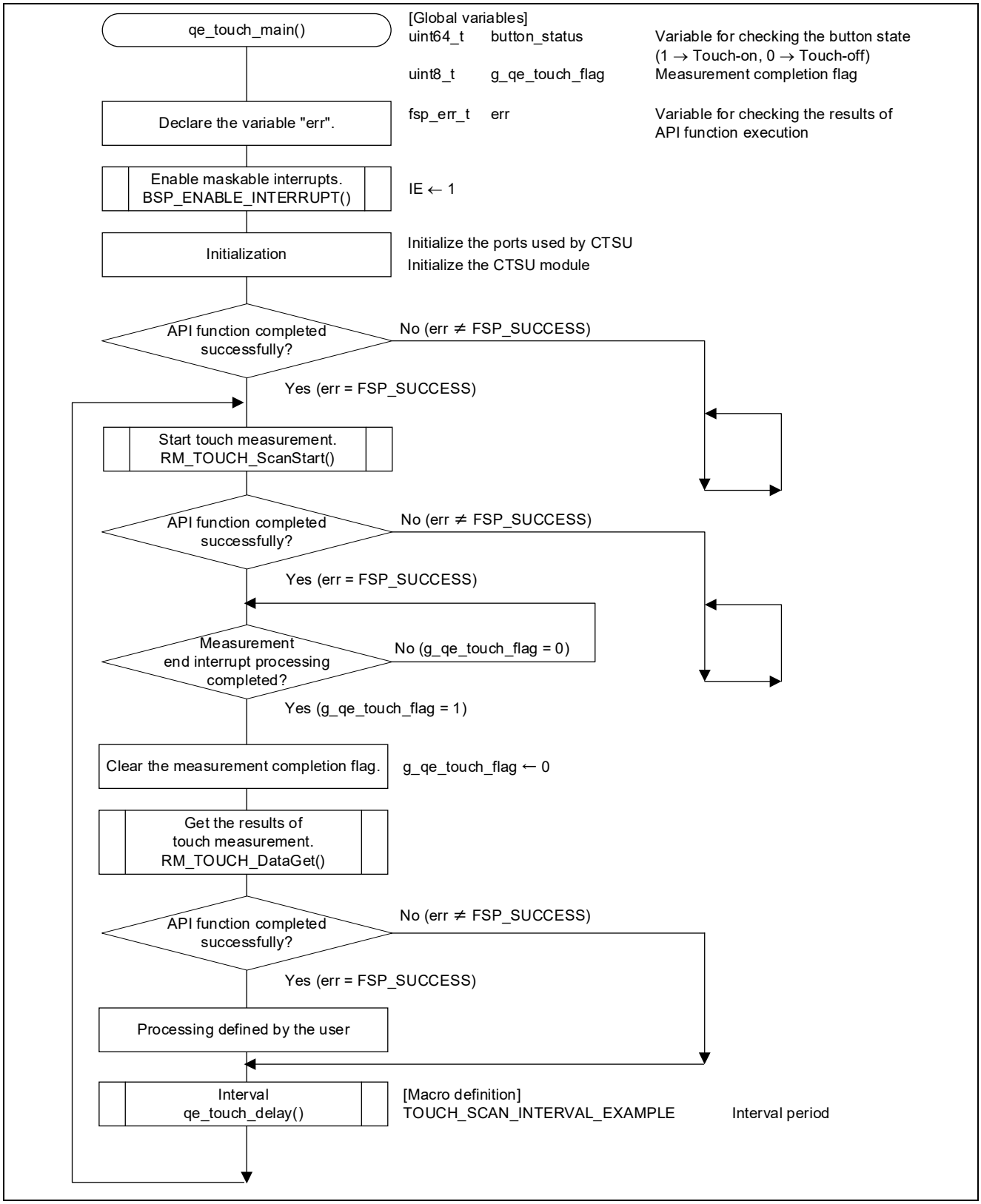


Figure 8-52 Touch Measurement Control Processing with the Use of a Software Timer

9. Another Implementation of the Sample Application

9.1 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) or touch sensor 2 (TS_B2) 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 chapter 7, Settings of Smart Configurator.

Remark: The timer array unit or 12-bit interval timer can also be used instead of the 32-bit interval timer.

9.1.1 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 the interval timer. The low-speed peripheral clock (fSXP) is used in this example. In addition, deselect the XT1 oscillator.

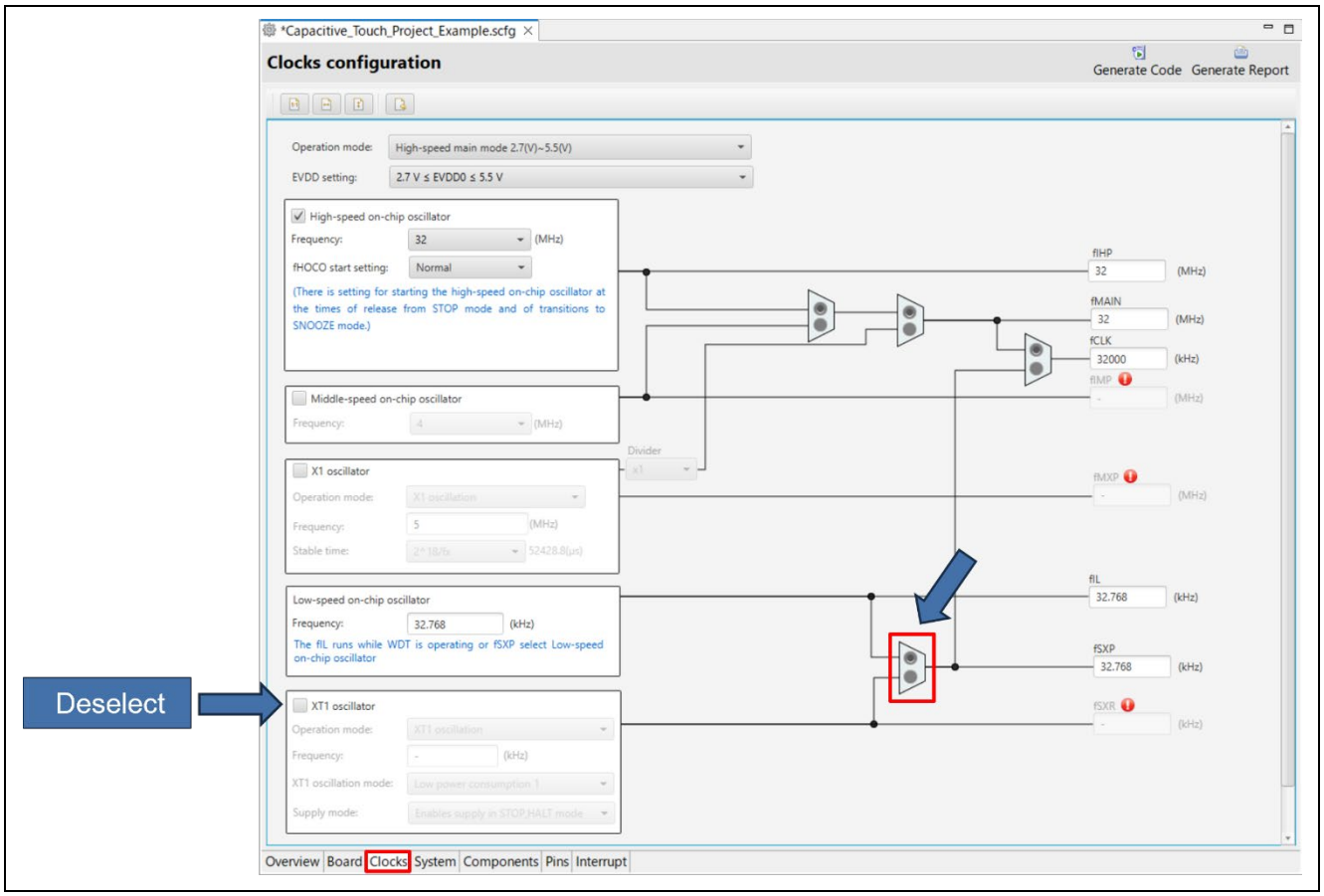


Figure 9-1 Setting the Clock

2. Add the peripheral functions that are required for touch measurement and LED control with the use of a hardware timer.
   Select the [Components] tab and click on to open the [New Component] dialog box. Select the "Interval Timer" and "Ports" modules and click on [Next].
   After that, assign resources for the selected components. The following settings are used in this example.

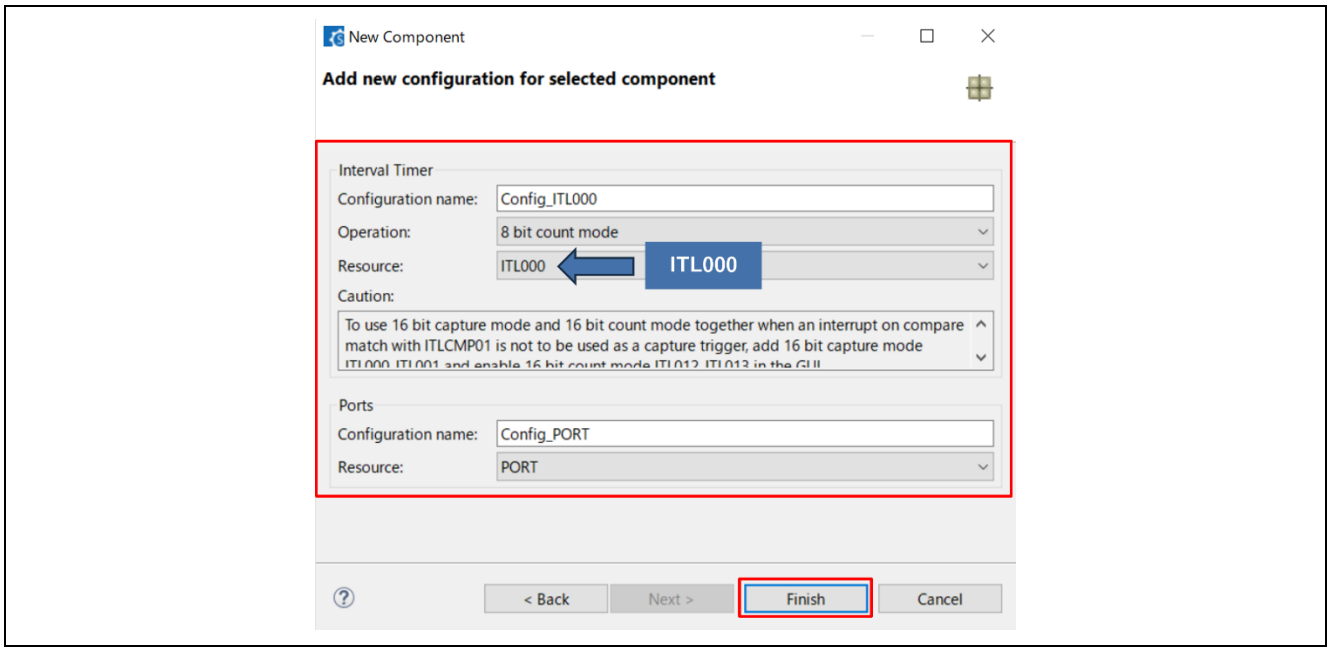


Figure 9-2 Assigning Resources for the Interval Timer and Ports

3. Set up the interval timer. Select the "Config_ITL000" component and make the following specifications.

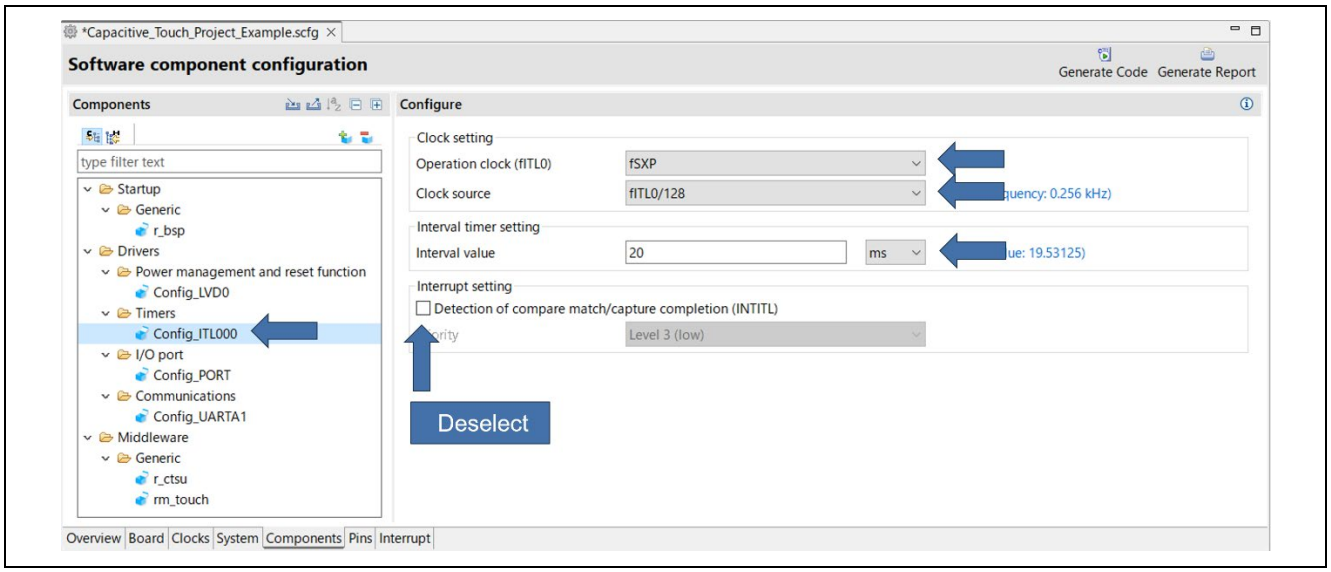


Figure 9-3 Setting the Config_ITL000 Component

4. Set up the pin to be used for LED control. Select the "PORT" module and set the P60 pin as an output initially at the high level.

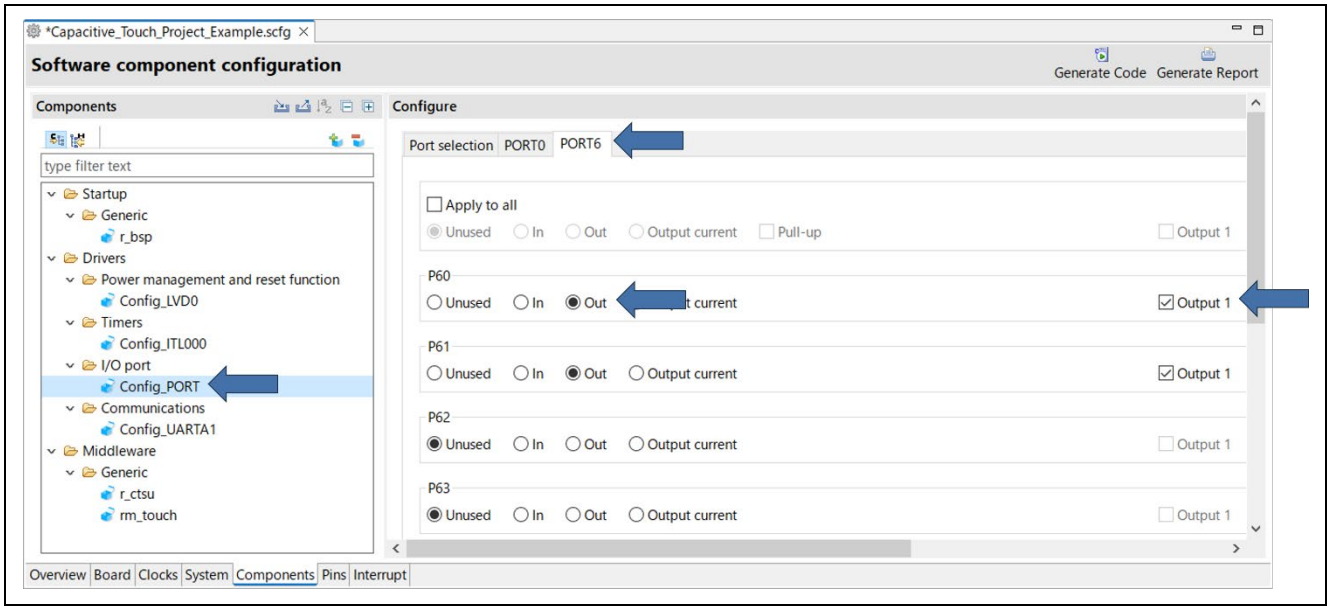


Figure 9-4 Setting the P60 Pin

5. Click on the icon in the top right part of the Smart Configurator view to generate code.

After this, follow the steps described in chapter 8, Settings of QE for Capacitive Touch, to complete the settings.

9.1.2 Flowchart (Hardware Timer)

Figure 9-5 is a flowchart of the touch measurement control processing with the use of a hardware timer.

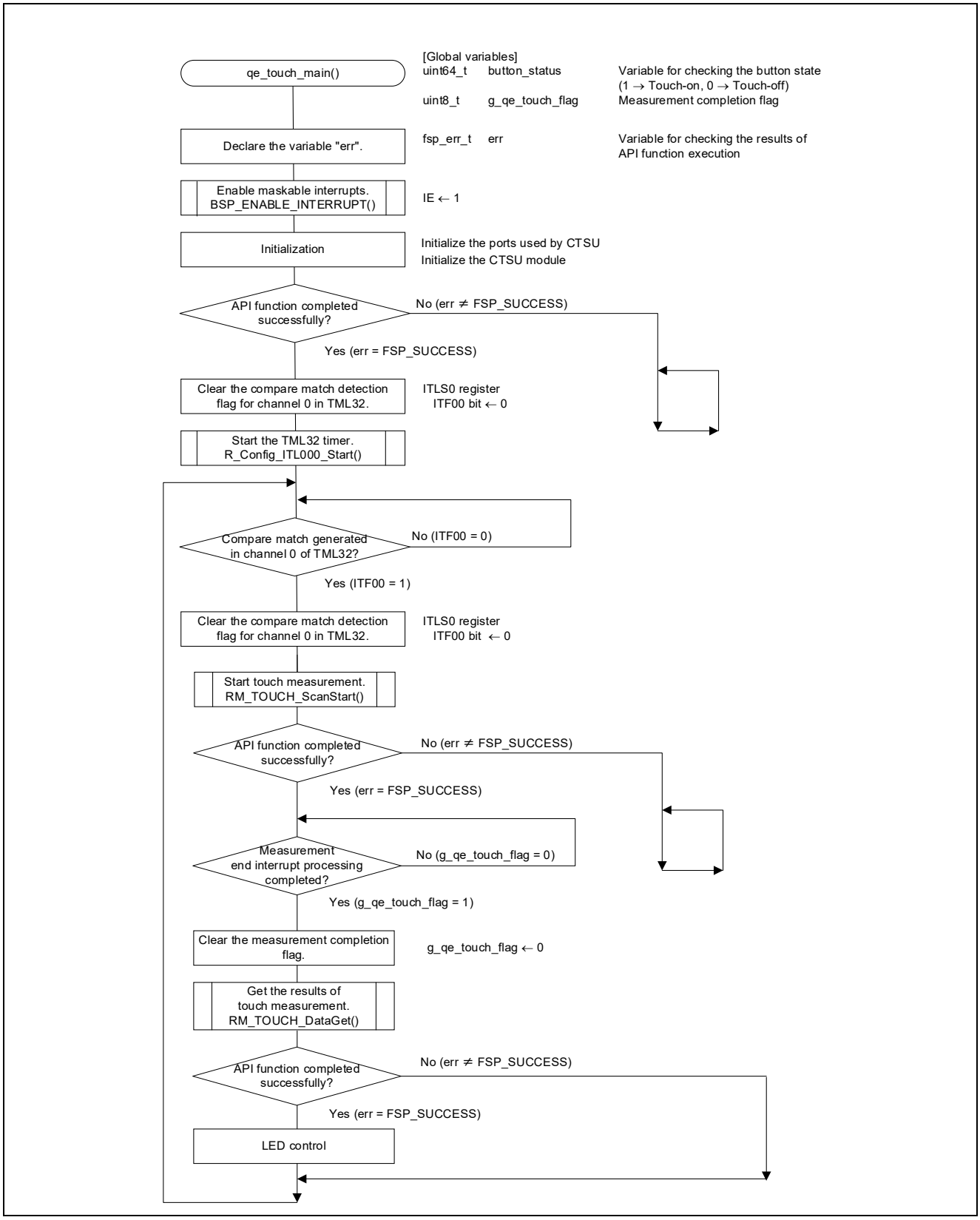


Figure 9-5 Touch Measurement Control Processing with the Use of a Hardware Timer

9.1.3 Sample Code (Hardware Timer)

The following is a listing of the sample code (qe_touch_sample.c) with the use of a hardware timer in implementing touch measurement.

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

```
/***********************************************************************
* FILE : qe_sample_sample.c
* DESCRIPTION : CTSU2L 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;
             }
         }
    }
}
```

10. Documents for Reference

- User's Manual
  - RL78/G23 User's Manual: Hardware [R01UH0896]
  - RL78 Family User's Manual: Software [R01US0015]

  The latest versions are available on the Renesas Electronics Web site.

- Technical Update and Technical News
  The latest information is available on the Renesas Electronics Web site.

- User's Manual: Development Environment
  - RL78/G23 Capacitive Touch Evaluation System User's Manual [R12UZ0095]

  The latest version is available on the Renesas Electronics Web site.

- Application Note
  - Capacitive Sensor Microcontrollers CTSU Capacitive Touch Introduction Guide [R30AN0424]
  - RL78 Family Using the Standalone Version of QE to Develop Touch Applications for a Fast Prototyping Board [R01AN6741]
  - 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 QE and SIS to Develop Capacitive Touch Applications [R01AN5512]

  The latest versions are available on the Renesas Electronics Web site.

Website

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

- QE for Capacitive Touch related page
  https://www.renesas.com/qe-capacitive-touch

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

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Sep.02.22	-	First edition
2.00	Aug.08.25	1	"Introduction" was modified.
		4	Chapter 1, Overview, was added.
		4	The target devices of this document were corrected.
		5	The descriptions in section 2 were updated.
		5	Table 2-1 was updated.
		5	SIS modules were added to Table 2-1.
		5	Table 2-2 was updated.
		5	Table 2-3, Conditions for Verifying Operation, was added.
		6	Section 2.1, Functions of QE for Capacitive Touch, was added. The contents are equivalent to those of chapter 1, System Overview, in Rev. 1.00.
		6	Figure 2-1 was updated.
		7	The procedures for installing CS+ and the Smart Configurator were added to section 3.1. The title of section 3.1 was changed to "Installing Development Tools".
		8	Table 3-1, Jumper Settings on the Board, was added.
		8	Figure 3-1 was updated.
		9	Figure 4-1 was updated.
		10	Table 4-1 was updated.
		11	A statement on the method of communications for tuning and monitoring in this application note was added.
		11	The restrictions on monitoring described in section 7.3.3 in Rev. 1.00 were moved to section 5.1 as a remark.
		11	Figure 5-1 was updated.
		11	An overview of the attached sample code was added to section 5.1.
		11	Table 5-1, Overview of the Attached Sample Code, was added.
		12	Table 5-2 was updated.
		14	"Voltage detector (LVD)" and "Port functions (PORT)" were added as required settings to chapter 7.
		15	Figure 7-4 was updated.
		16	The descriptions in section 7.3 were updated.
		16	Figure 7-7 was updated.
		17	Figure 7-7 was updated.
		17	A cautionary note on the procedure for downloading of SIS modules was added.
		18	A note on the method of setting the voltage detection function of the RL78/G16 was added.
		18-19	Section 7.4, Adding Components, was added to describe the procedures for adding the components to be used and assigning resources to them.
		18-19	Figure 7-9 to Figure 7-11 were added.
		20-26	Section 7.5, Modifying the Component Settings in the Smart Configurator, was added as an overview of the settings of the components.
		24	Section 7.5.4, Setting the LVD Component, was added.

		25	Section 7.5.5, Setting the PORT Component, was added.
		26	Section 7.5.6, Board Support Package, was added. The contents are equivalent to those of step 1 in section 7.6, Generating Code, in Rev. 1.00.
		28	Figure 7-24, Cautionary Message before Code Generation, was added.
		28	The value of the user option byte was modified in Figure 7-25.
		33	The description of step 2 in section 8.3 was updated.
		33	A description and a cautionary note were added to step 3 in section 8.3.
		34	A description was added to step 5 in section 8.3.
		35	The description of step 7 in section 8.3 was updated.
		35	Figure 8-13, Creating a New Folder "qe_gen", was added.
		35	Figure 8-14 was updated.
		37-39	The description of step 13,14 in section 8.3 was updated.
		38	Step 15 in Section 8.3 was added.
		38	Figure 8-22 was added.
		39	Step 16,17 in Section 8.3 was added.
		39	Figure 8-23 and Figure 8-24 were added.
		40	A statement describing how to handle warnings was added.
		40	Figure 8-27 was added.
		43	Figure 8-30 (p. 39 in Rev. 1.00) were deleted.
		44	Figure 8-37 was updated.
		47	Figure 8-41 was updated.
		49	Figure 8-44 and Figure 8-45 were updated.
		49-50	The description of step C was updated in step 6 of section 8.5.1.
		50	Figure 8-46 was added.
		50	Figure 8-47 and Figure 8-48 were updated.
		51	Figure 8-49 was updated.
		53	Software Timer Sample Code (described on p. 48 and p. 49 of Rev. 1.00) was deleted.
		53	A title for Figure 8-52 was added.
		53	CTSU port initialization and CTSU module initialization in Figure 8-52 changed to initialization process
		54	The descriptions in section 9.1 were updated.
		54	A point to note on the other implementation of the sample application was added to section 9.1.
		54	The description of step 1 in section 9.1.1 was updated.
		54	Figure 9-1 was updated.
		55	The description of step 2 in section 9.1.1 was updated.
		55	Figure 9-2 was updated.
		56	Supplementary information regarding the procedure for settings following step 5 in section 9.1.1 was added.
		57	Changed Figure 9-5. CTSU port initialization and CTSU module initialization were changed to initialization process. Changed LED lighting processing section to LED control
		57	A title for Figure 9-5 was added.

		58-59	The descriptions in section 9.1.3 were updated, and the newly added code sections were clearly indicated.
		60	Added the following reference document to Chapter 10. ・Added reference document [R01AN5512] (moved from Rev. 1.00 abstract) ・Added reference document [R30AN0424] ・Added reference document [R01AN6741]
		60	The reference document [R01AN5886] from Chapter 10 was deleted.

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

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/.

© 2022 Renesas Electronics Corporation. All rights reserved.