RL78 Family

Using QE (standalone ver.) to Develop Touch Applications for FPB board

Introduction

This application note explains the steps to create an application example that uses capacitive touch sensing using the RL78/G22 FPB (Fast Prototyping Board) (product name: RTK7RLG220C00000BJ) with mounted touch electrodes.

This application note is capacitive touch application development guide using "CS+, standalone version Smart Configurator and standalone version QE for Capacitive Touch".

Using standalone version QE can develop application regardless of device or IDE.

If you are using the RL78/G22 Capacitive Touch Evaluation System (RTK0EG0042S01001BJ) with "CS+, standalone version Smart Configurator and standalone version QE for Capacitive Touch" as an alternative development environment, see the following application note.

- RL78 Family Using the standalone version of QE to Develop Capacitive Touch Applications (R01AN6574)

If you don't use standalone version QE but "e^2 studio, plug-in version Smart Configurator and plug-in version QE for Capacitive Touch" as development environment, see the following application note.

- RL78 Family Using QE and SIS to Develop Capacitive Touch Applications (R01AN5512)

Target Device

RL78/G22

RL78 family with Capacitive Sensing Unit (CTSU)

Contents

1. System Overview

QE for Capacitive Touch is a development tool that supports initial setup and adjusting sensitivity of the touch interface for a development of embedded system using capacitive touch sensors.

The main functions of QE for Capacitive Touch are as follows.

- Creating touch interface configurations
  It is possible to set visually assignments of touch sensor and positions of touch interface such as button.

- Tuning
  It is possible to tune automatically offset and sensitivity of touch interface.

- Monitoring and parameter adjustment
  It is possible to monitor the performance of touch interface and adjust details of parameters.

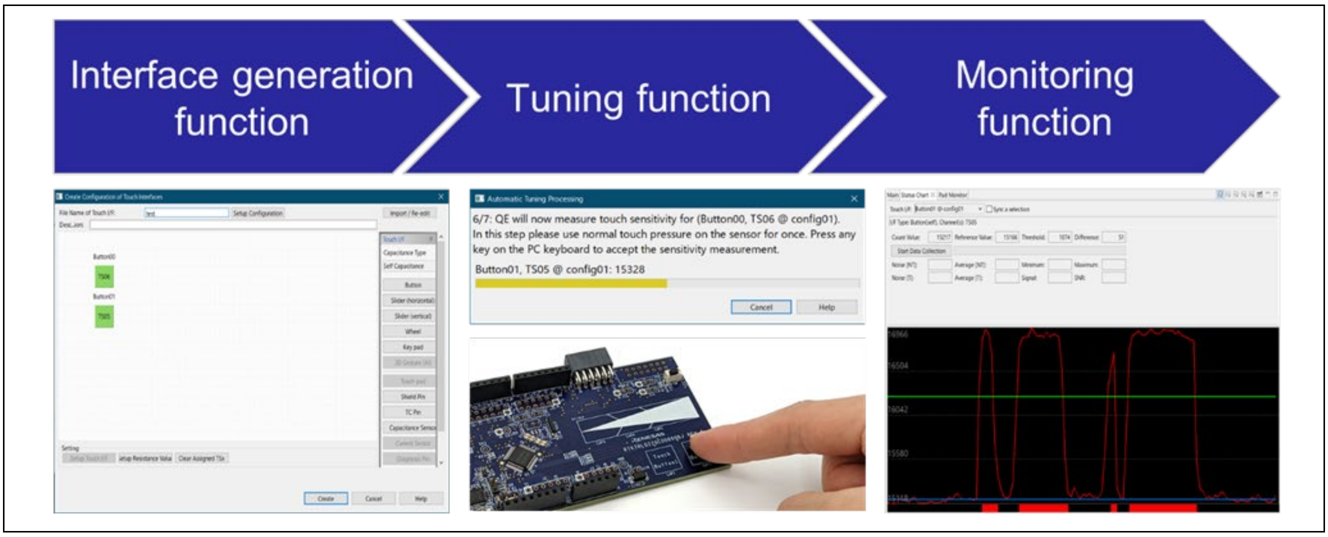


Figure 1-1. Main Functions of QE for Capacitive Touch

2. Operating Environment

Table 2-1 and Table 2-2 show the operating environment for this application note.

The program generated by the standalone version of QE is written to RL78/G22 by CS+, and then run on RL78/G22.

This application note can be utilized for other devices from the “RX/RA/RL78 family and Renesas Synergy ™ platform” with capacitive touch IP.

Table 2-1. Operating Environment (Software)

Items	Contents	Version
IDE	CS+ for CC	8.09.00 or later
Toolchains	CC-RL	1.12.00 or later
QE	Standalone Version QE for Capacitive Touch	3.2.0 or later
Smart Configurator	RL78 Smart Configurator	1.5.0 or later

Caution When using the CC-RL free evaluation edition V1.12.00 or later for tuning of touch sensors, select "debug precedence(-onothing)" as the optimization levels.

Table 2-2. Operating Environment (Hardware)

Items	Contents
Microcontroller used	RL78/G22 (R7F102GGE2DFB)
Target Board	RL78/G22 Fast Prototyping Board (RTK7RLG220C00000BJ)

3. Building the Development Environment

This chapter explains how to install tools and connect the board to PC.

This application example uses the following tools.

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

This chapter will not explain how to install CS+ and Smart Configurator. If you haven't installed them yet, install them according to their procedure.

3.1 Installation of the Standalone Version QE for Capacitive Touch

Install standalone version QE for Capacitive Touch by taking the following steps.

If you have already installed, this section is not necessary.

1. Download "QE for Capacitive Touch" from Renesas Electronics website.

2. The downloaded zip file has plugin version and standalone version.
   Extract the downloaded zip file.
   Then choose a folder for extraction which windows file path is not over the character limit (260 characters).
   For example, in the directory of "C:\Renesas".

3.2 Connection of the Target Board

Connect the target board to the PC.

Following Figure 3-1, connect the target board to PC. via USB.

In this application example, power is supplied to the target board via USB. Confirm the circuits on the target board, and then set switches or jumpers as necessary.

For the application example, set the jumpers of the target board as follows.

- JP16 : Open when performing the QE serial connection function
  : Closed when performing the COM PORT debug connection function
- JP17 : 1-2 short

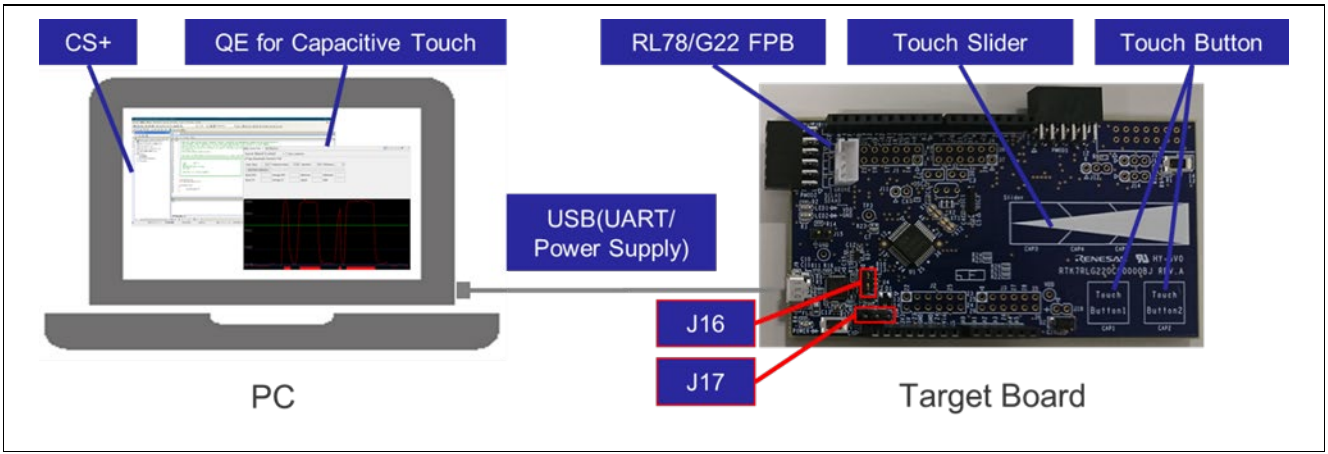


Figure 3-1. Target Board and PC Connection

4. Workflow for Developing an Application

This chapter explains how to create an application.

Follow the steps in the workflow of QE for Capacitive Touch to develop an application.

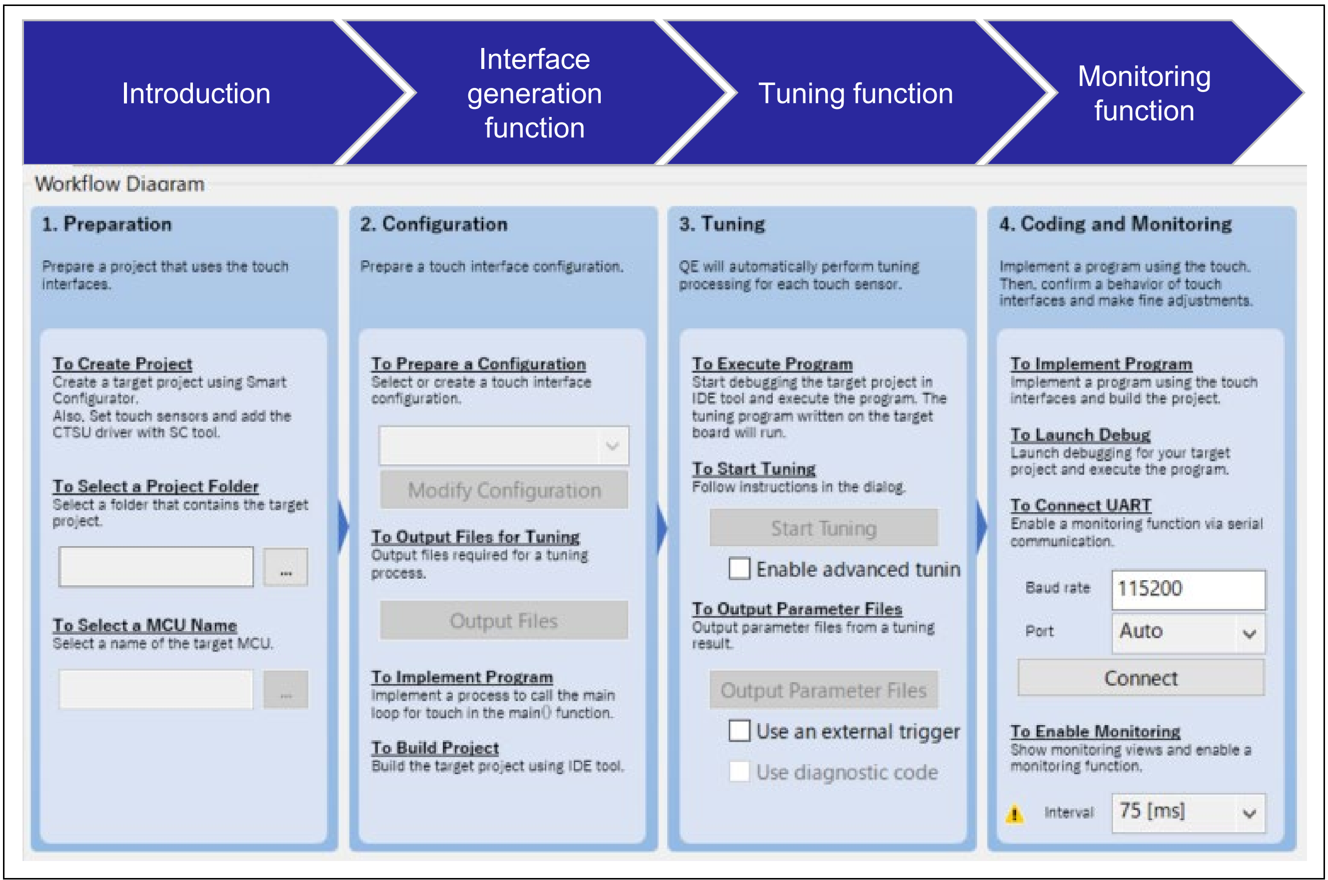


Figure 4-1. Workflow for Developing an Application

Table 4-1 shows each step within the workflow. Chapter numbers in the table are linked to the corresponding chapter page. Click each chapter number in the table to see how to use each function. IDE and Smart Configurator is used for project creation and coding, project build, and debug.

Table 4-1. Items of QE for Capacitive Touch

Items				Capture
Preparation	Project Creation	Creating Project Using IDE		6
		Setup of Smart Configurator		7
			Setup of Clock and System	7.2
			Setup of CTSU Driver	7.3.2
			Setup of Touch Middleware	7.3.3
			Setup of Serial Interface (UART)	7.4
			Setting Unused Pins to Low-level Output	7.5
	To Select a Project Folder			8.2
	To Select a 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. Application Example

5.1 Application Example Overview

This application note provides an example of an application which uses two buttons and one slider.

From chapter 6 onward, the application note explains how to create the application and monitor whether the either of the buttons or the slider is touched.

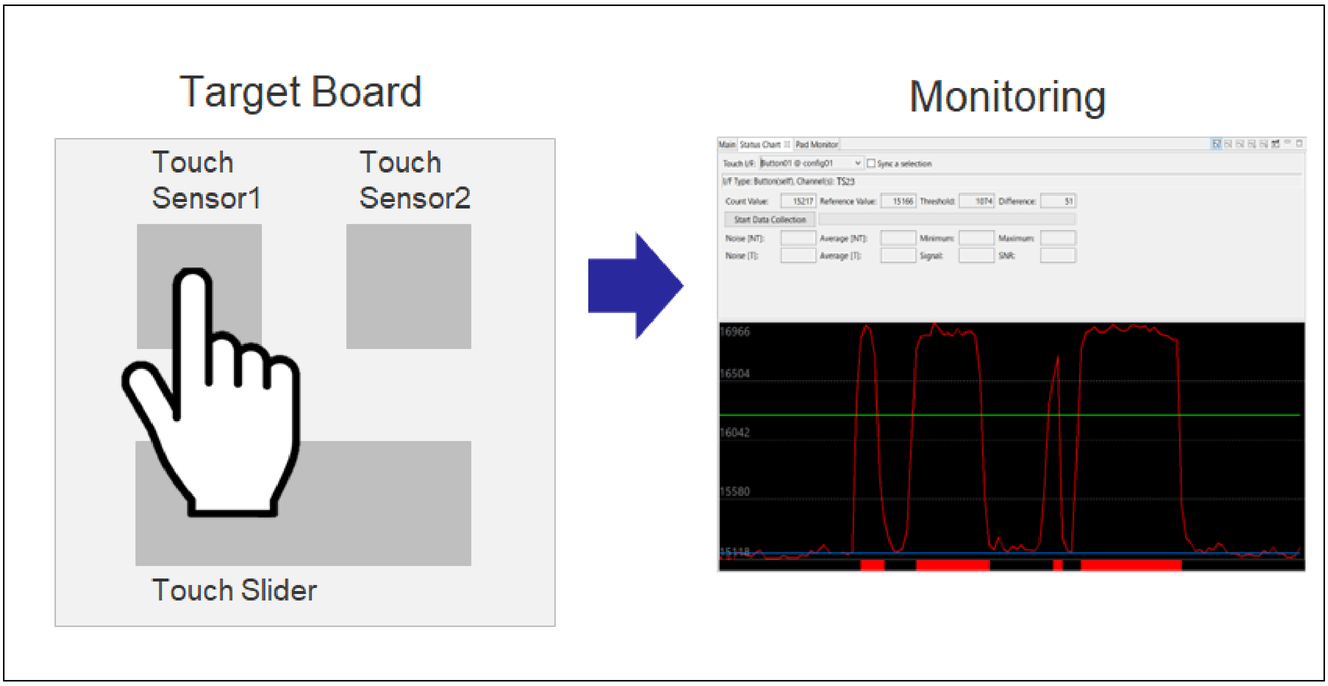


Figure 5-1. Application Example

5.2 List of Used Pins

Table 5-1 shows the pins used in this application example.

UART communication and touch sensors in the application depend on the target board you are using

Table 5-1. List of Used Pins for Application Example

Items	Pins	Uses
UART Communication	RxD0/P11	Tuning
	TxD0/P12	Monitoring
Touch Sensor 1	TS24/P26	Button (TS_B1)
Touch Sensor 2	TS23/P25	Button (TS_B2)
Touch Slider	TS20/P22	Slider (TS_S)
	TS21/P23	
	TS22/P24	

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

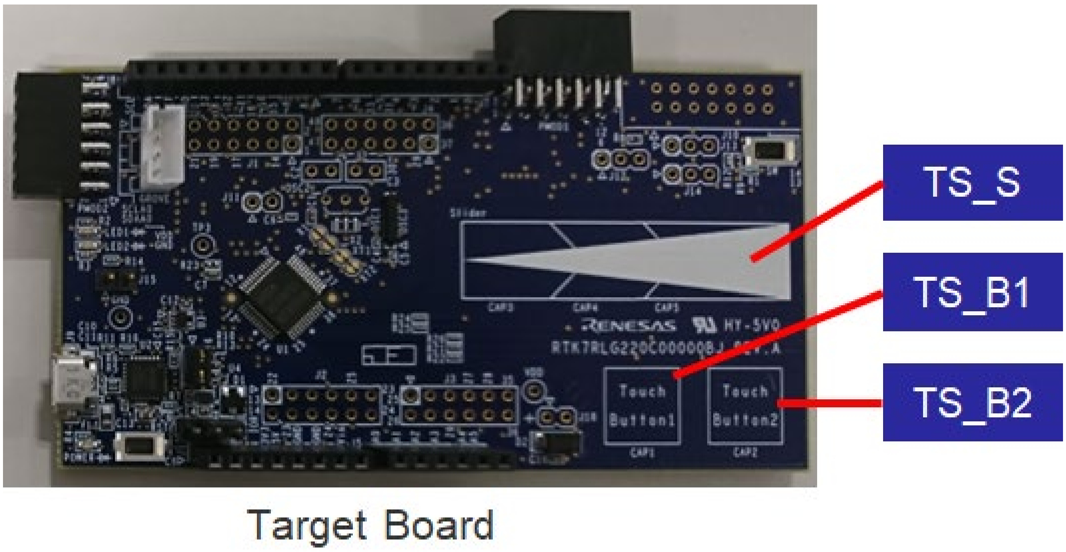


Figure 5-2. Position of Touch Sensors

6. Project Creation

Launch CS+ and create new project.

In "Create Project" dialog, select the following.

- Microcontroller : RL78
- Using microcontroller : R7F102GGExFB (48pin)
- Kind of project : Application (CC-RL)
- Project name : (Any project name)
- Place : (Any place)

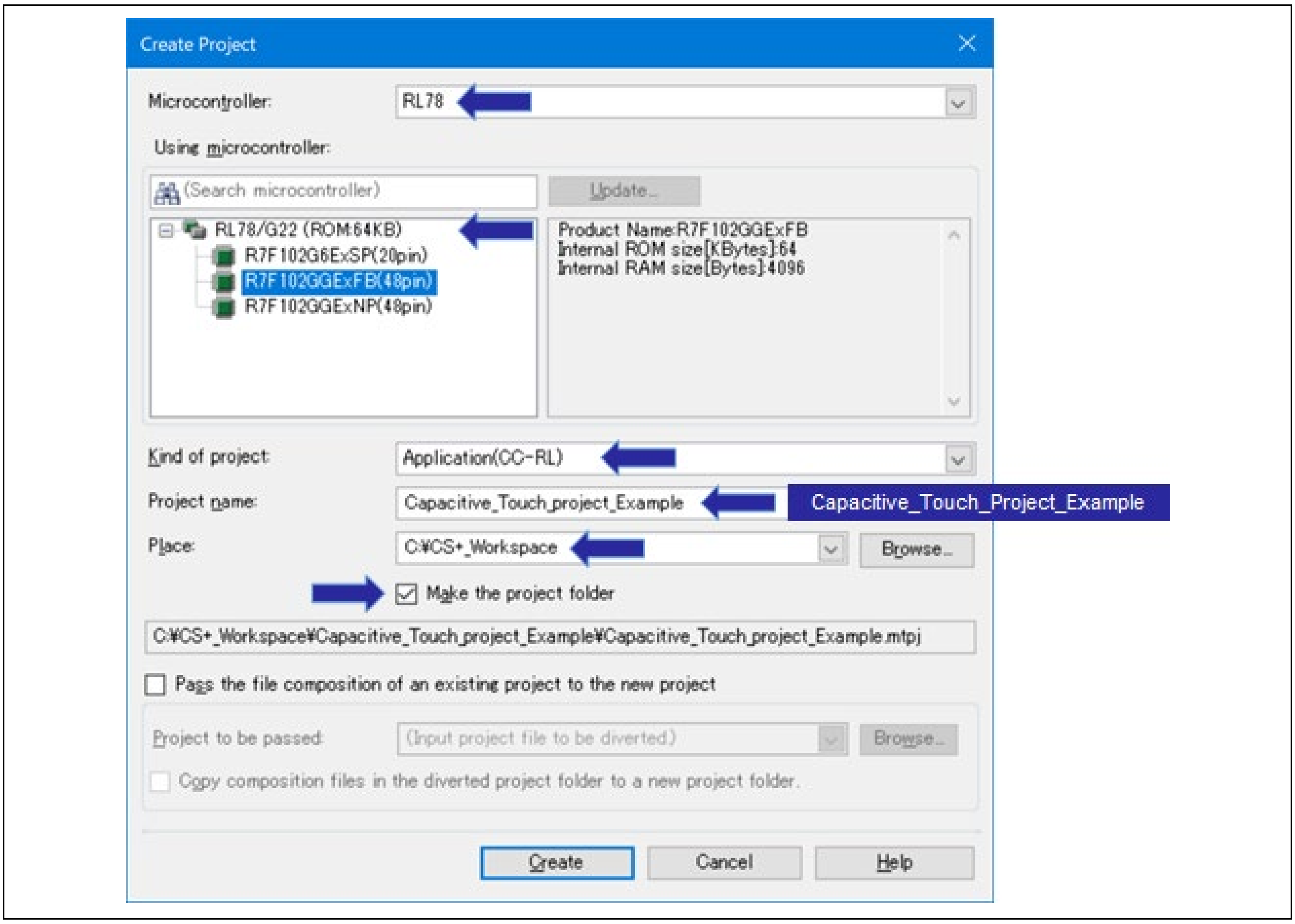


Figure 6-1. Creating New Project

7. Setup of Smart Configurator

This chapter explains how to set by Smart Configurator. Necessary setup for this application example is the following.

- Clock and system
- CTSU driver
- Touch middleware
- Serial interface (UART Communication)
- Unused pins to low-level output

7.1 Launching Smart Configurator

Double-click “Smart Configurator” in “Project Tree” of CS+, and launch Smart Configurator.

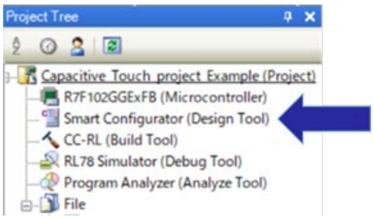


Figure 7-1. Launching Smart Configurator

If Smart Configurator cannot be launched, confirm the following.

- Whether file path in the property of Smart Configurator is correct.
- Whether “Smart Configurator for RL78 Communication Plug-in” is selected in “Tool” -> “Plug-in Manager” of Menu.

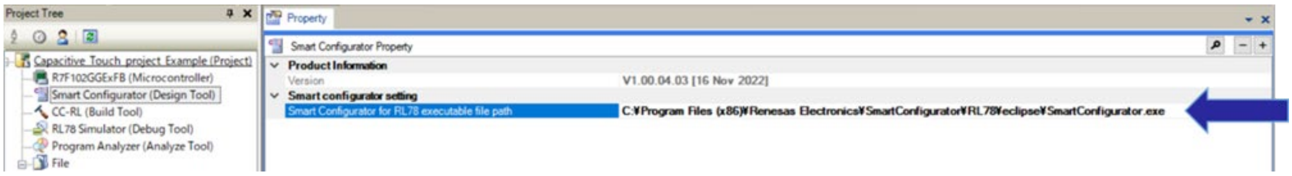


Figure 7-2. File Path of Smart Configurator

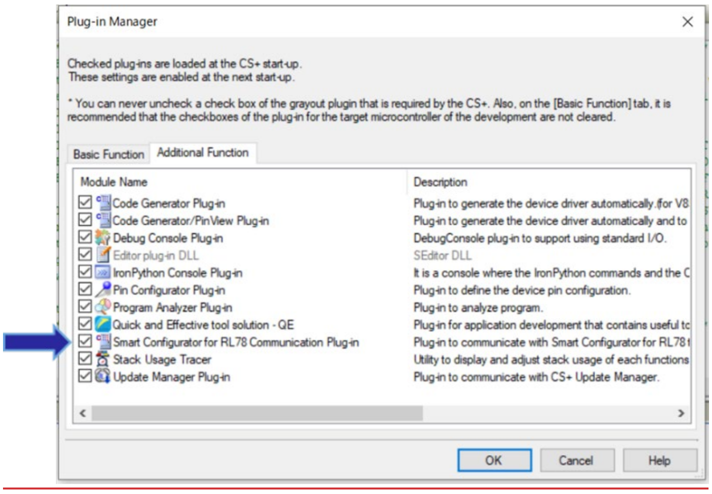


Figure 7-3. Plug-in Manager

7.2 Setup of Clock and System

This section explains how to set the clocks and system.

1. Select “Clocks” tab in lower-middle menu, and set clocks.

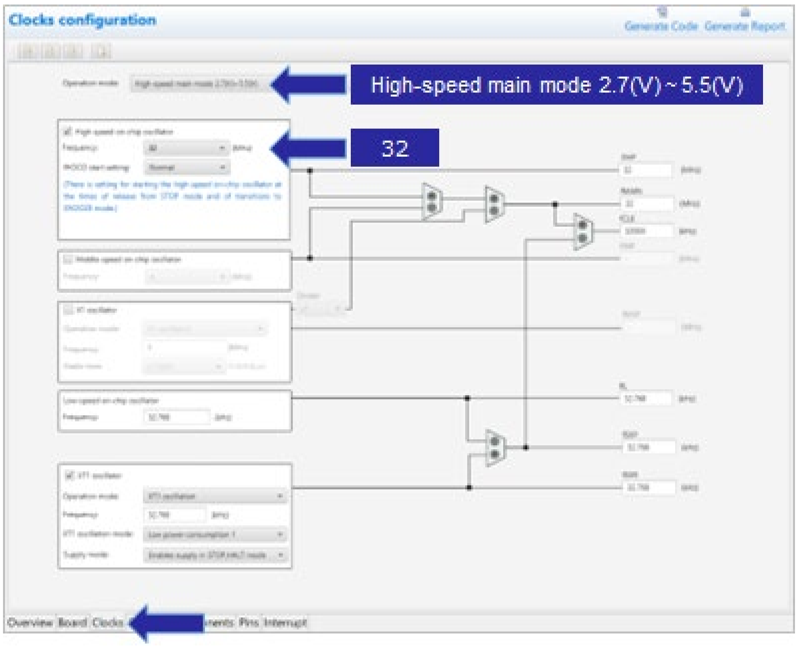


Figure 7-4. Setup of Clocks

2. Select “System” tab and set the debug environment.

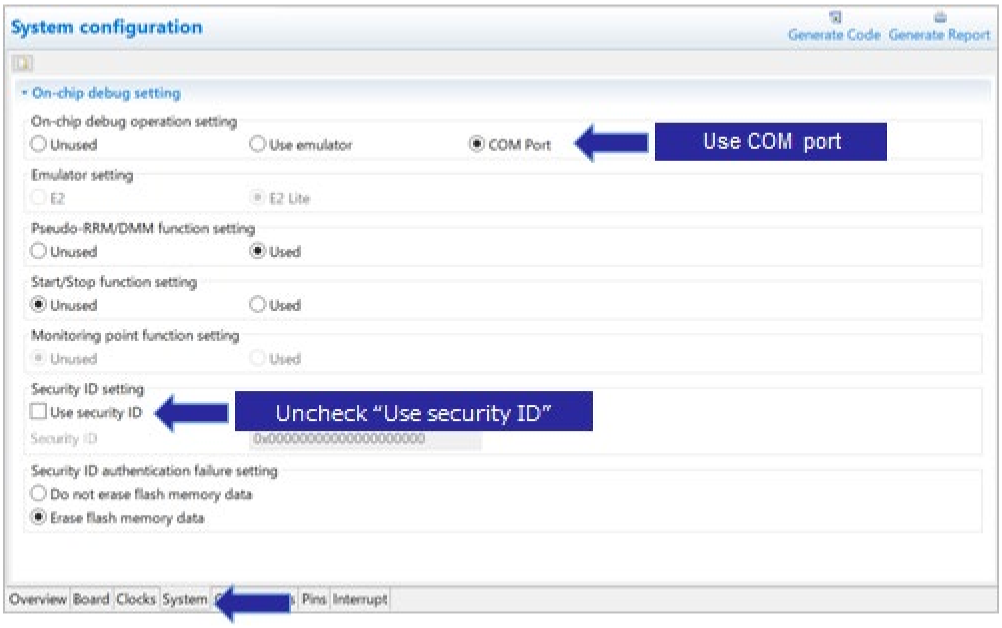


Figure 7-5. Setup of Debug

7.3 Setup of SIS (software Integration System) Modules

This section explains how to add two SIS modules which are “CTSU Driver” and “Touch Middleware” used for QE for Capacitive Touch and set them.

7.3.1 Download of SIS Modules

Download “CTSU Driver” and “Touch Middleware” by Smart Configurator.

If you have already installed, this section is not necessary.

1. Select “Components” tab and click the  icon.

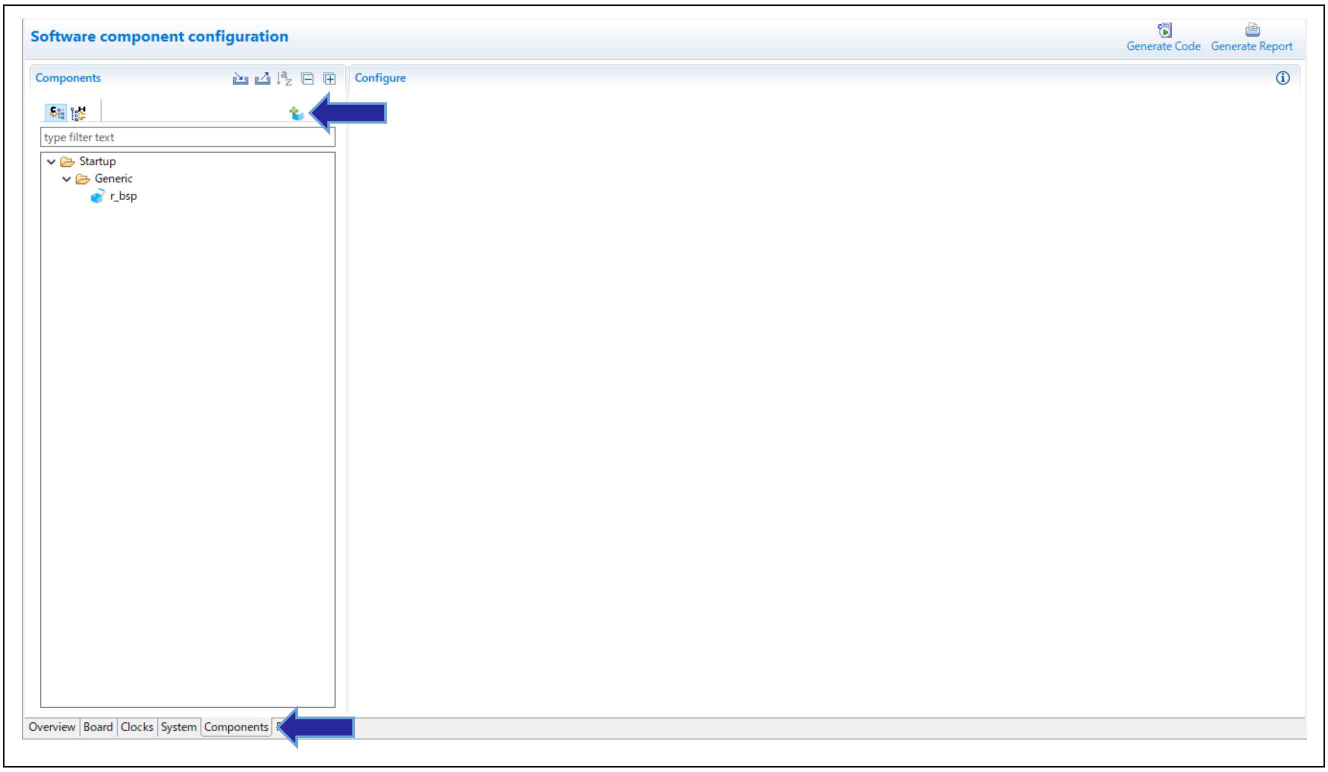


Figure 7-6. Software Component Configuration

2. Click “Download RL78 Software Integration System modules” at lower of “New Component” dialog.

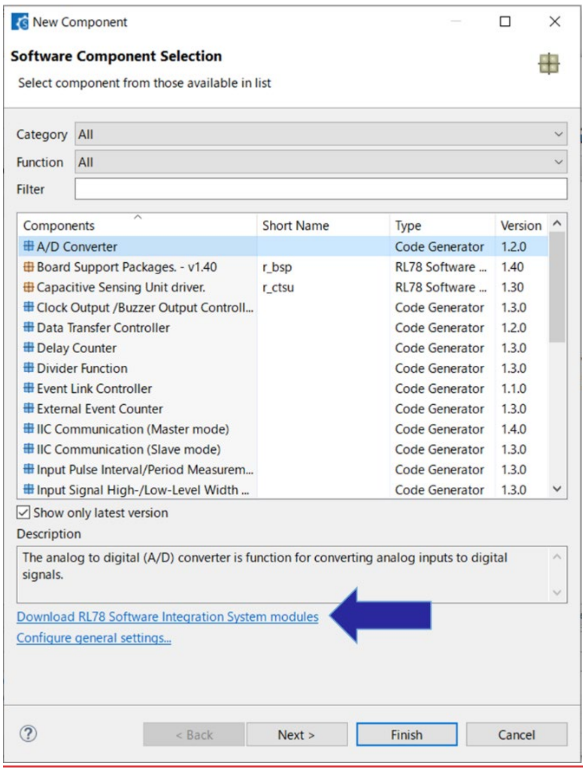


Figure 7-7. Software Component Selection Dialog Box

3. Select the following, and click “Download”.
   — RL78 Family CTSU Module Software Integration System
   — RL78 Family TOUCH Module Software Integration System

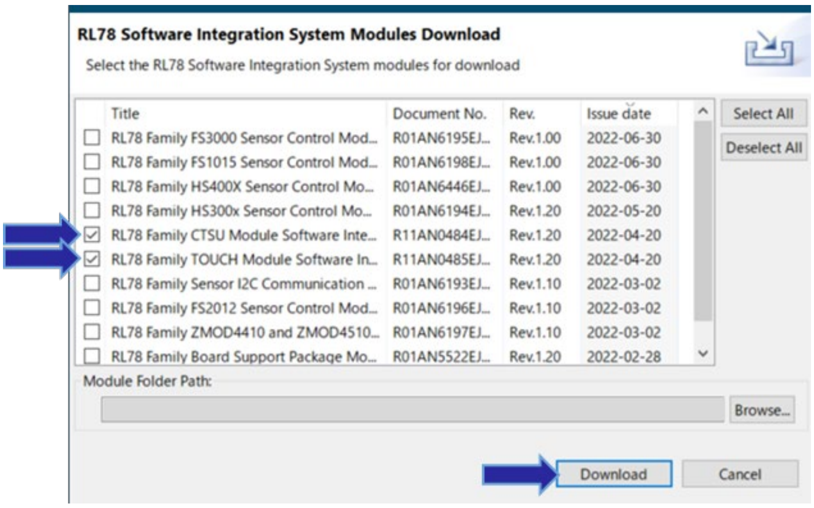


Figure 7-8. Download SIS Modules

7.3.2 Setup of CTSU Driver

This subsection explains how to set “CTSU Driver”.

1. Select “Components” tab and click icon. In the displayed dialog, select “r_ctsu” module and click “Finish”.

Figure 7-9. “r_ctsu” Module

2. Click “r_ctsu” module and enable TS pins used for this application example.
   In this application example, five TS pins are used.
   Please check user’s manual of your target board in order to confirm assignment between TS pins and touch sensor.

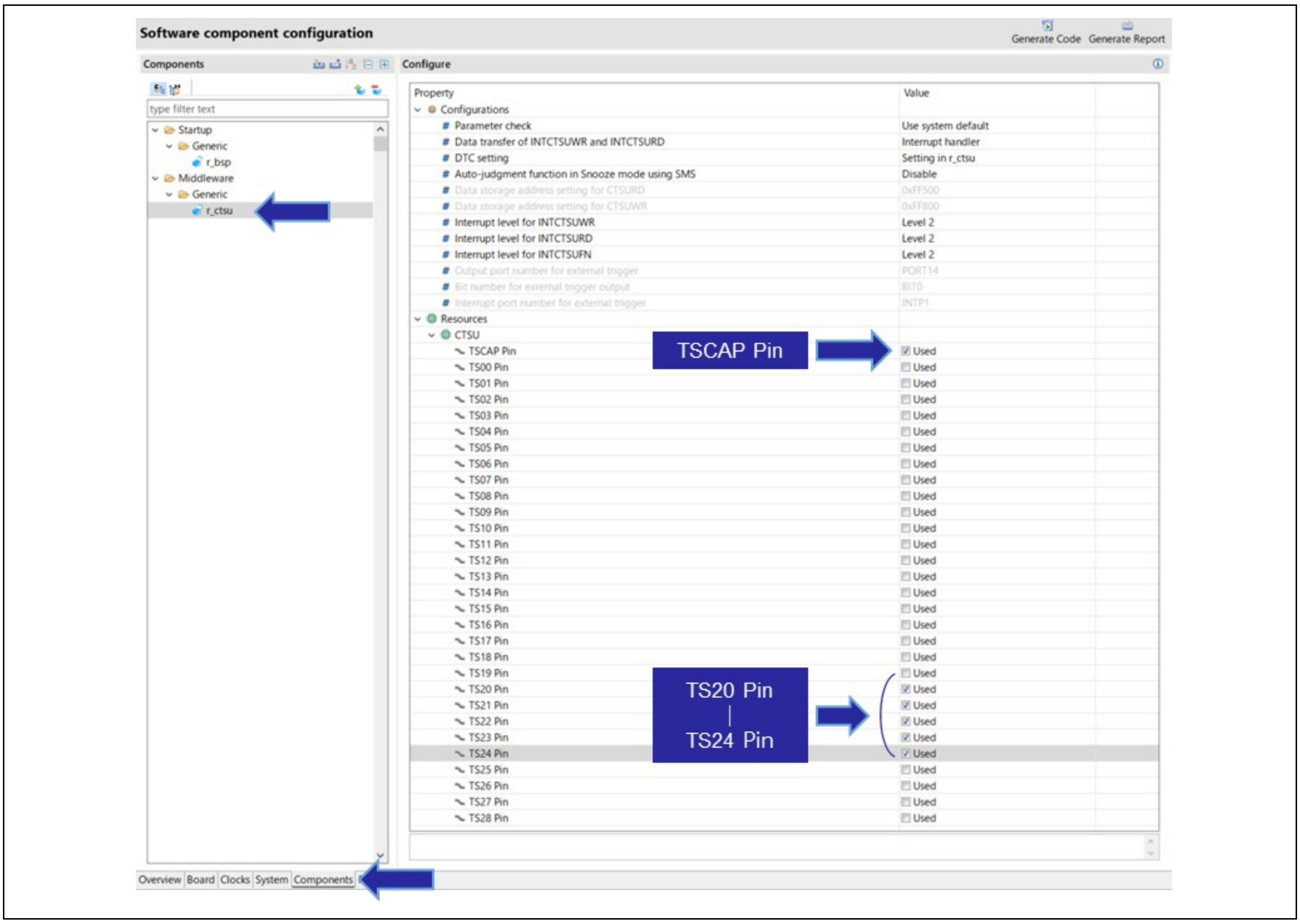


Figure 7-10. Enable Used TS Pins

3. It is recommended to set unused TS pins to low-level output. In CTSU2, when TS pins not used in the application are enabled, the TS pins are set to low-level output as non-measurement pins.

   In this application example, enable all TS pins, including unused pins. Note that pins TS12/TS13 pins are excluded as their dual functions are used in the application.

   In designing your circuit, make sure to perform sufficient pin processing and satisfy electrical characteristic requirements.

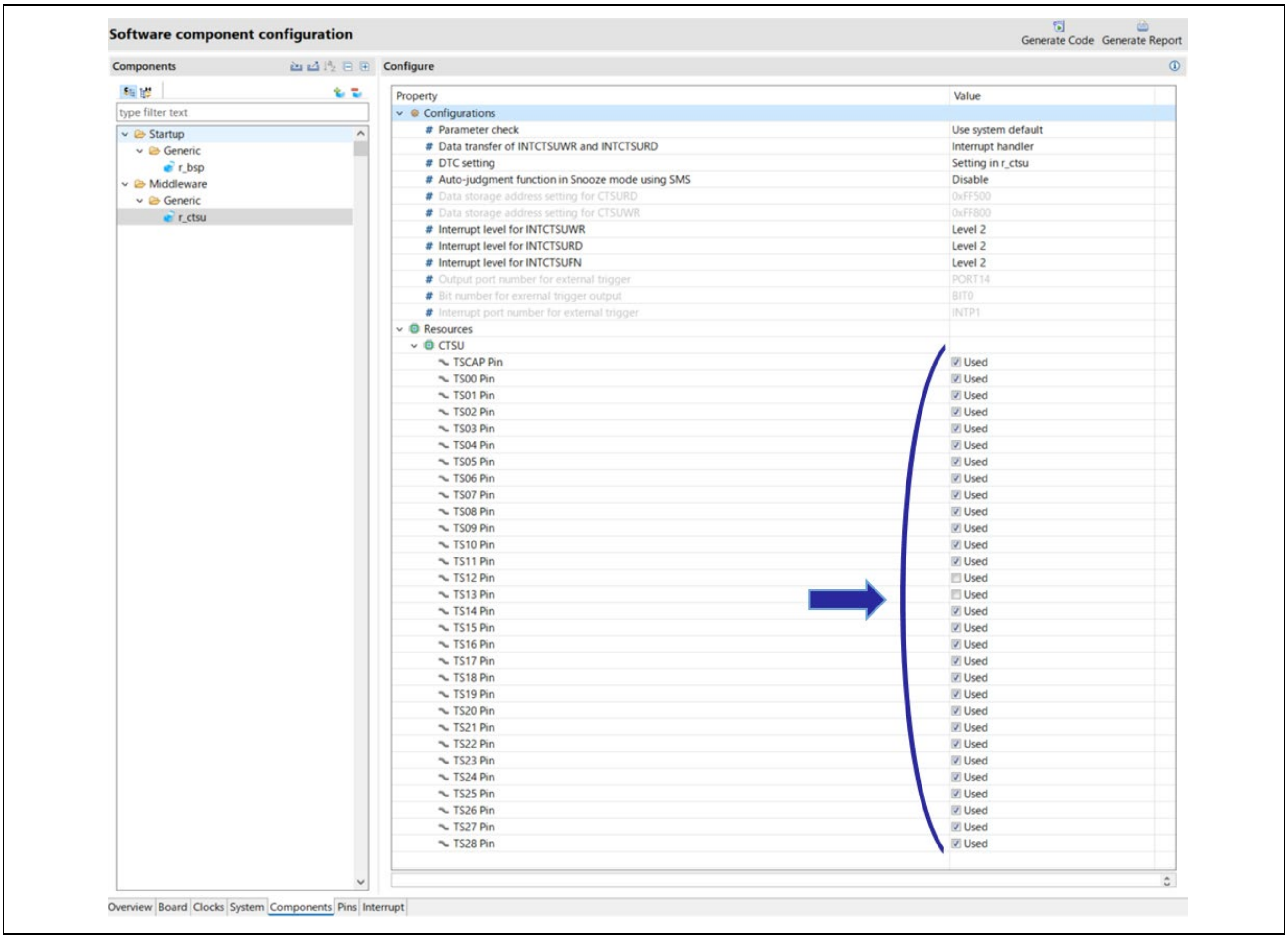


Figure 7-11. Enable TS Pins Unused in the Application

7.3.3 Setup of Touch Middleware

This subsection explains how to set “Touch Middleware”.

Monitoring touch performance for touch applications is possible by communication via the OCD (On-Chip Debugging) emulator. However, in RL78 family case, monitoring performance is limited by the OCD function of the RL78 family.

Monitoring touch performance using serial communication enable smooth monitoring. Also it is possible to tune using serial communication.

1. Click icon and select “rm_touch” module in the displayed dialog, and click “Finish”.

Figure 7-12. “rm_touch” Module

2. Click “rm_touch” module and set the following.
   - Enable to support QE monitor using UART
   - Enable to support QE tuning using UART
   - UART channel UART0

UART channel to set depends on your target board.

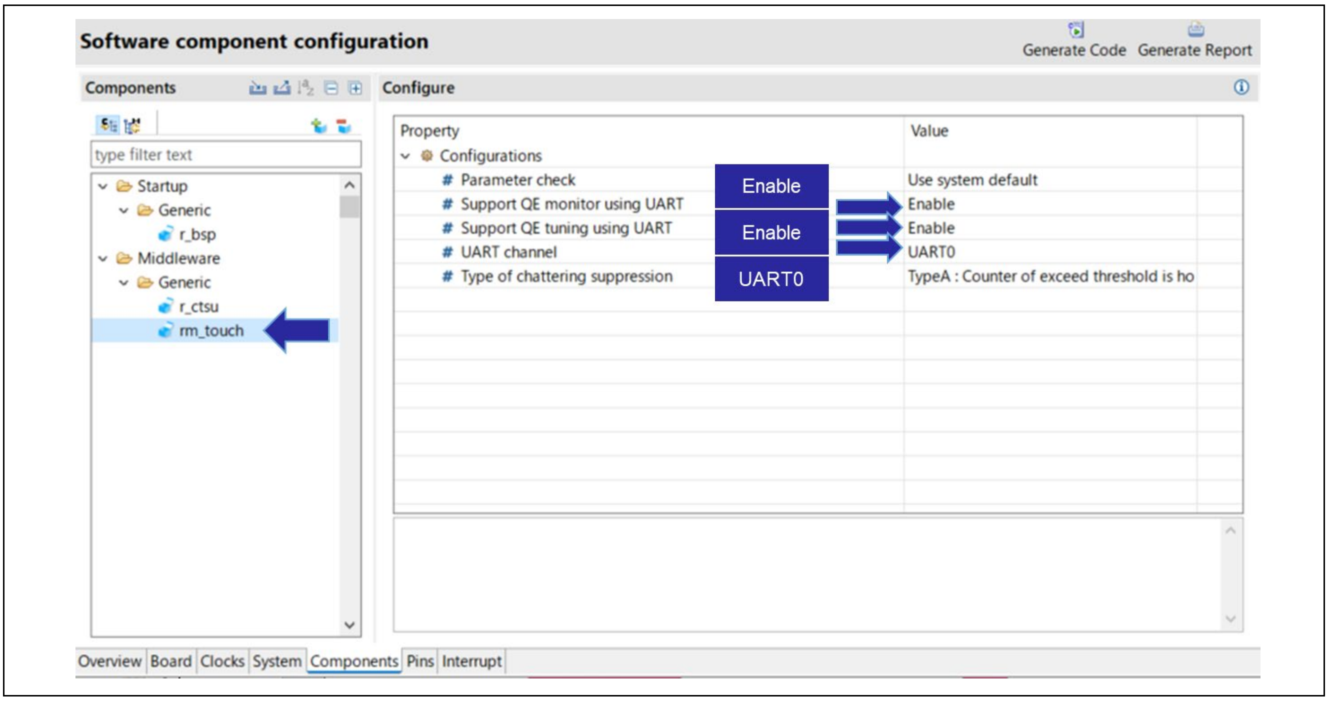


Figure 7-13. Setup of “rm_touch” Module

7.4 Setup of Serial Interface (UART)

This section explains how to set UART for tuning and monitoring of touch sensors.

The UART channel and port to be set depend on your target board.

1. Click icon. In the displayed dialog, select “UART Communication” module and click “Next”. Then set as follows and click “Finish”.
   — Operation : Transmission/reception
   — Resource : UART0

Figure 7-14. “UART Communication” Module

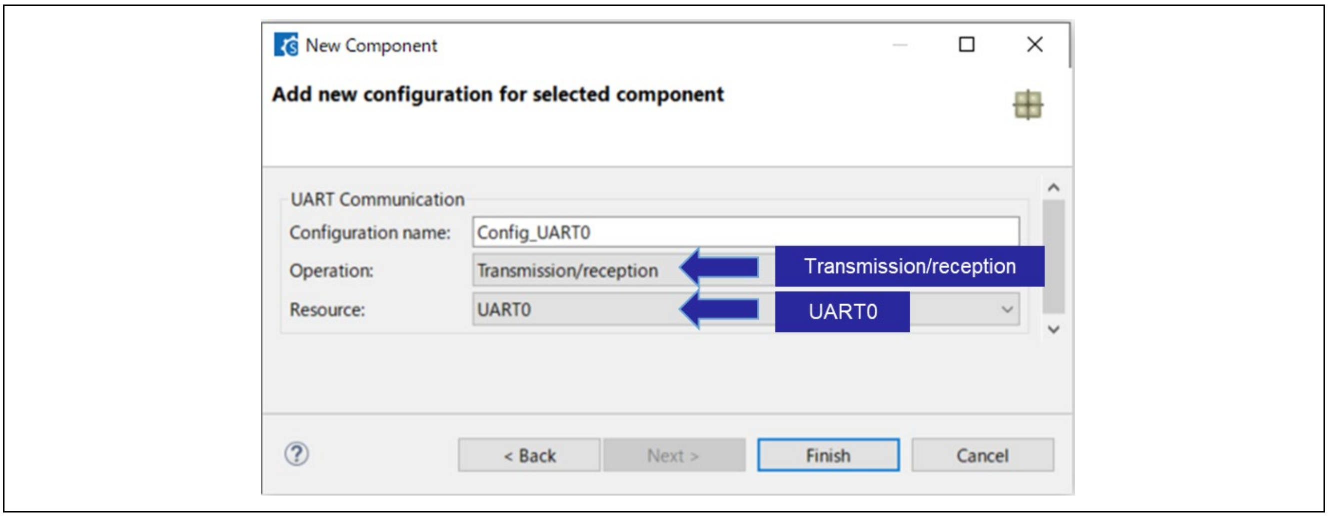


Figure 7-15. Select UART Channel

2. Click the added “UART Communication” module and set the operation clock and transfer rate (baud rate) in the transmit and receive sections.

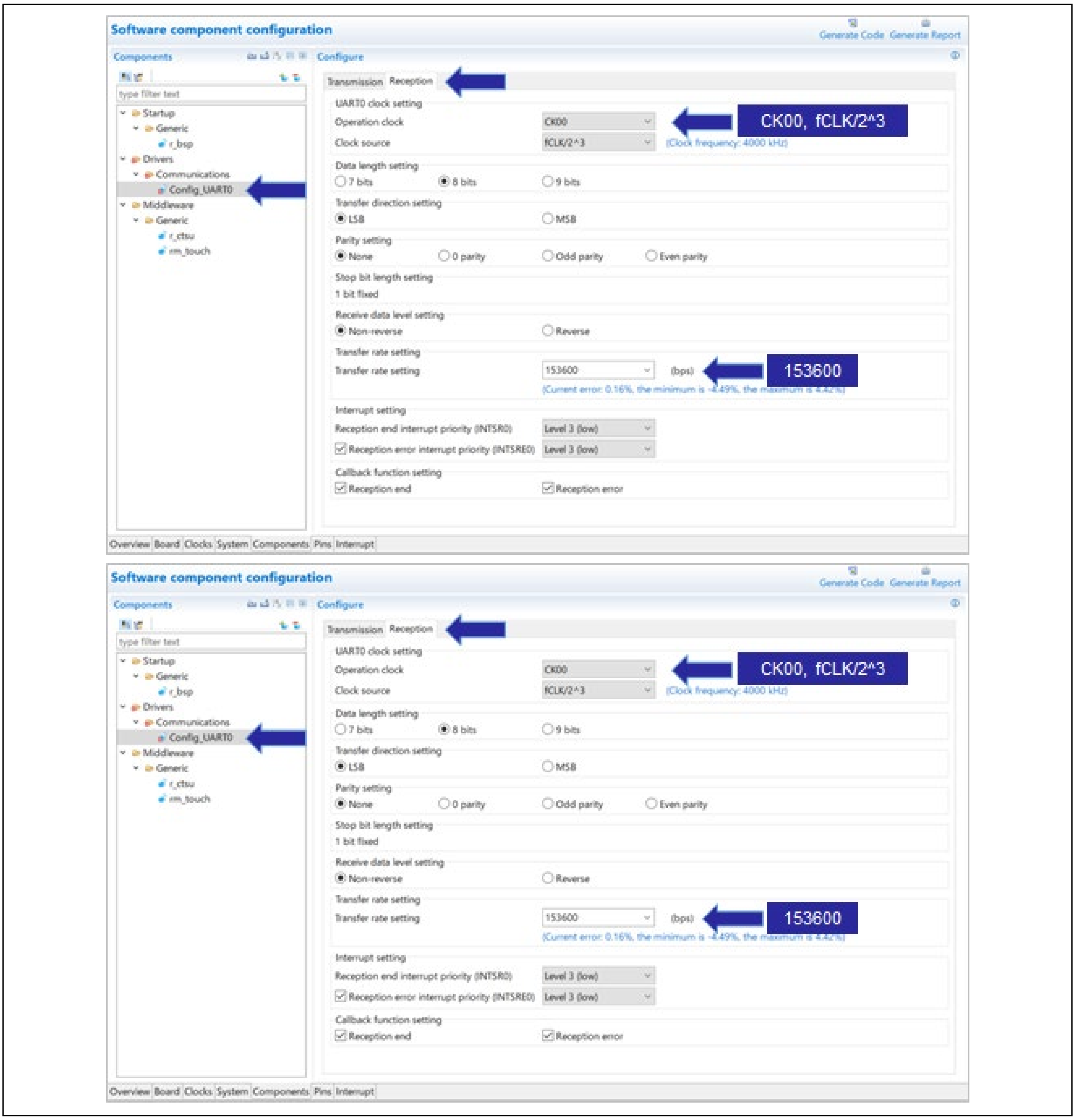


Figure 7-16. Setup of “UART Communication” Module (UART0)

3. Select “Pins” tab and assign the following pins to the UART (SAU00) channel.
   - RxD0 : 21
   - TxD0 : 20

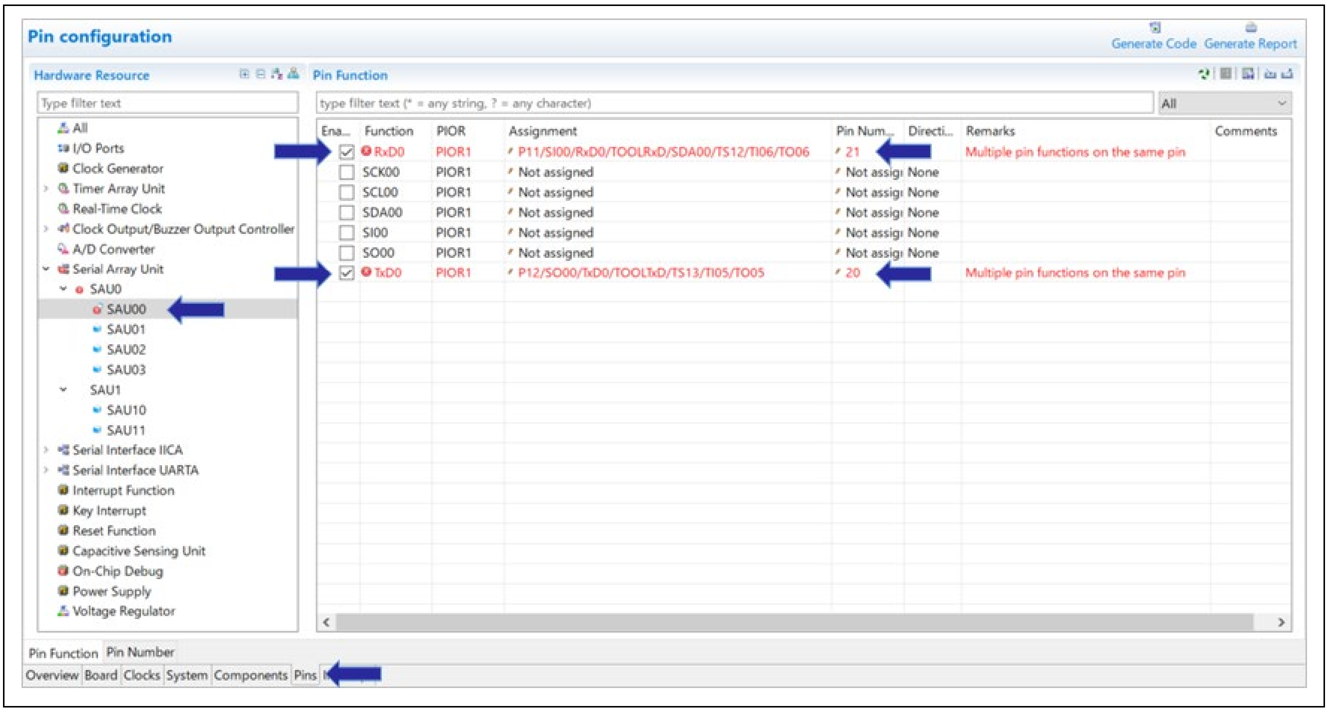


Figure 7-17. Assignment of Pins for UART Channel (UART0)

RxD0/TxD0 UART0 pin assignment error may occur depending on the tool version used, but the error should be ignored.

In this application, the program generated by the COM port debug function is written to RL78/G22 using CS+. The pins used to write the program (TOOLRxD/TOOLTxD) also function as UART0 RxD0/TxD0 pins, which may cause pin conflicts in the Smart Configurator. However, conflicts will not occur in actual use since the CS+ and standalone version of QE are not used at the same time.

When using CS+ (writing a program): operates as TOOLRxD/TOOLTxD pins

When using standalone version of QE: operates as RxD0/TxD0 pins

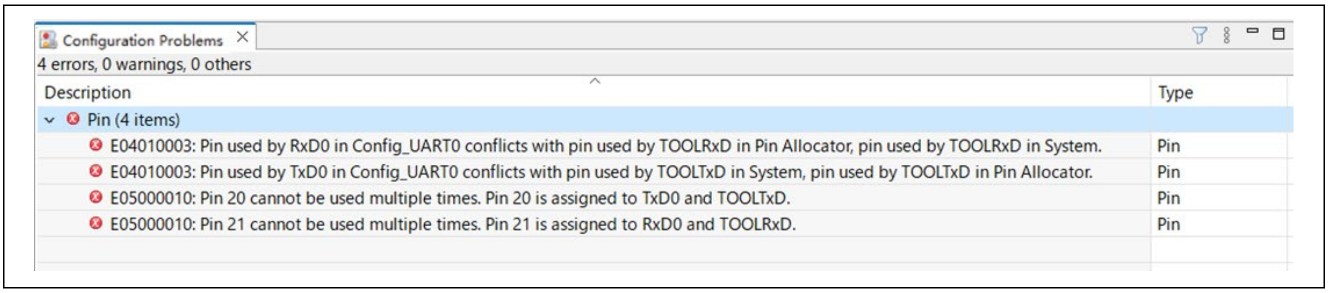


Figure 7-18. Pin Assignment Error for UART0

7.5 Setting Unused Pins to Low-level Output

It is recommended to set ports unused in the application to low-level output.

In designing your circuit, make sure to perform sufficient pin processing and satisfy electrical characteristic requirements.

Please see user's manual of your target board in order to confirm ports which you need to set to low-level output.

As example, this section explains how to set "PORT63" to low-level.

1. Select "Components" tab and click icon. In the displayed dialog, select "Port" module and click "Finish".

Figure 7-19. "Ports" Module

2. Select "Port" module and check "PORT6".

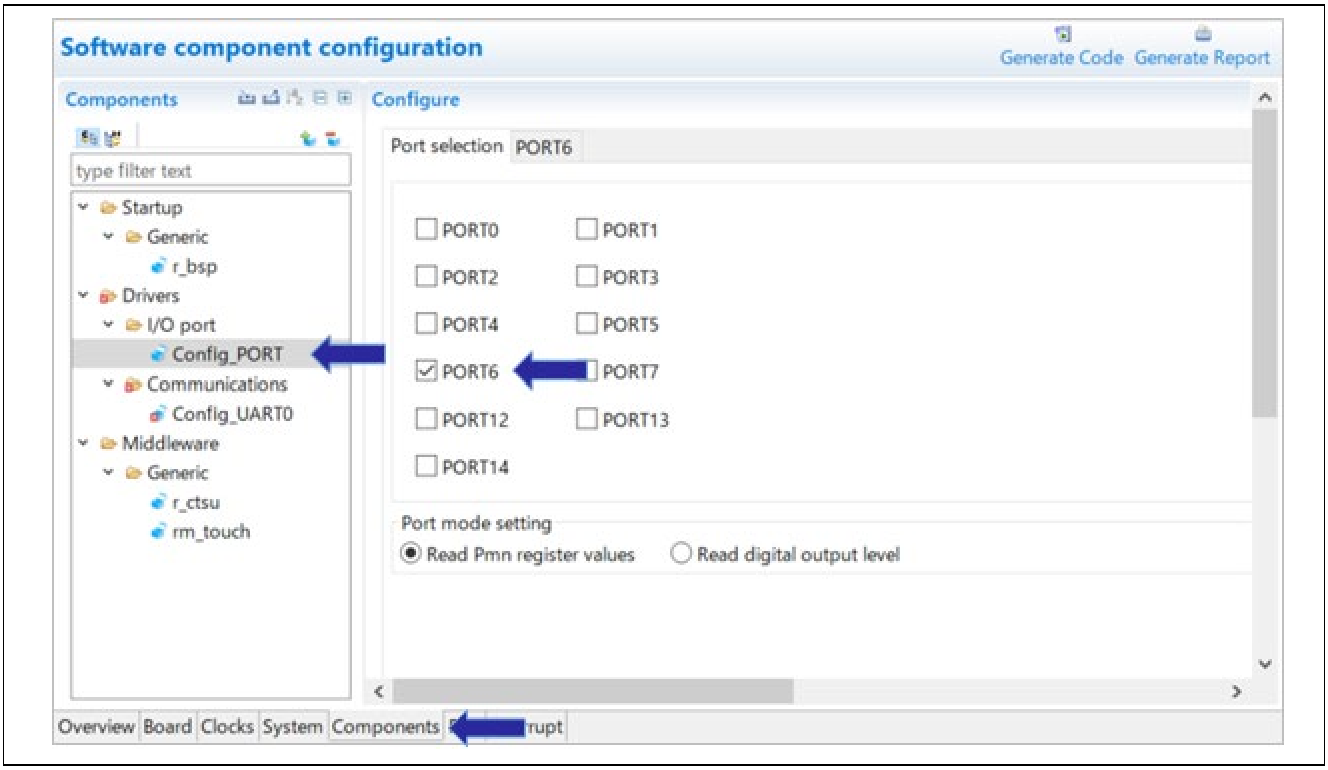


Figure 7-20. Setup of "Ports" Module

3. Click “PORT6” tab and set “P63” to output.

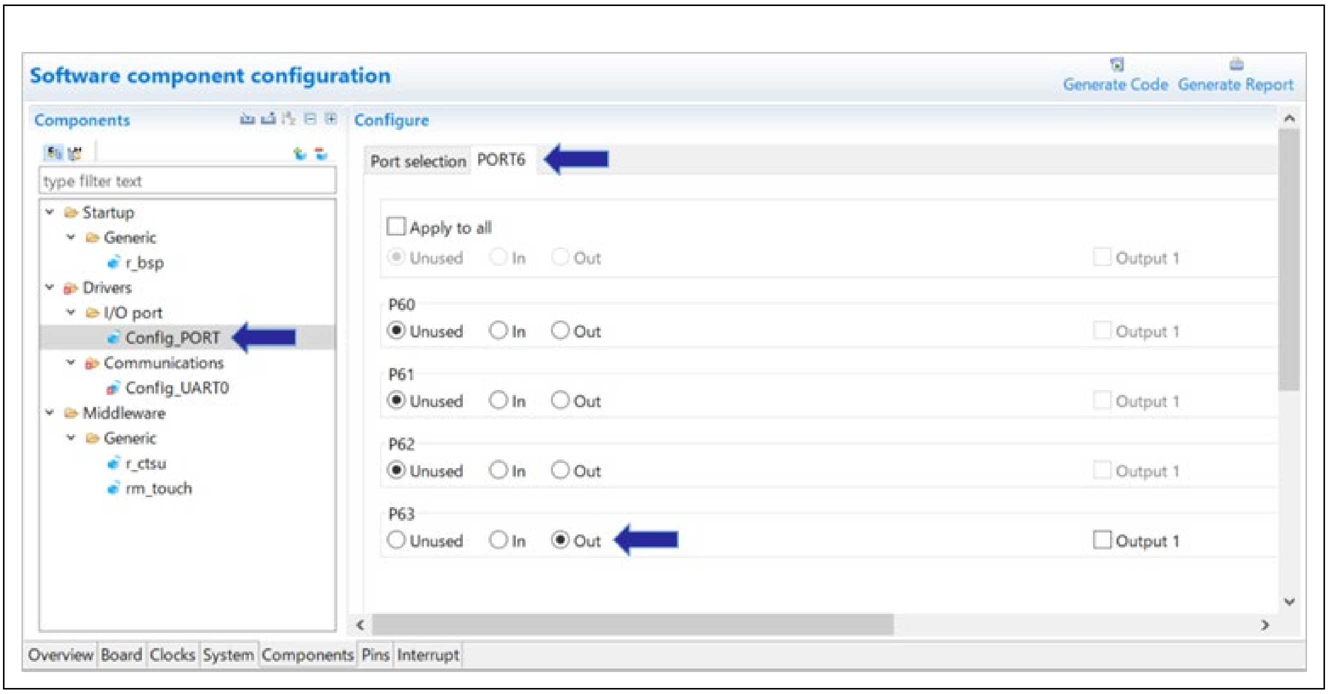


Figure 7-21. Setting “P63” to Output

7.6 Generating Code

Perform generating code.

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

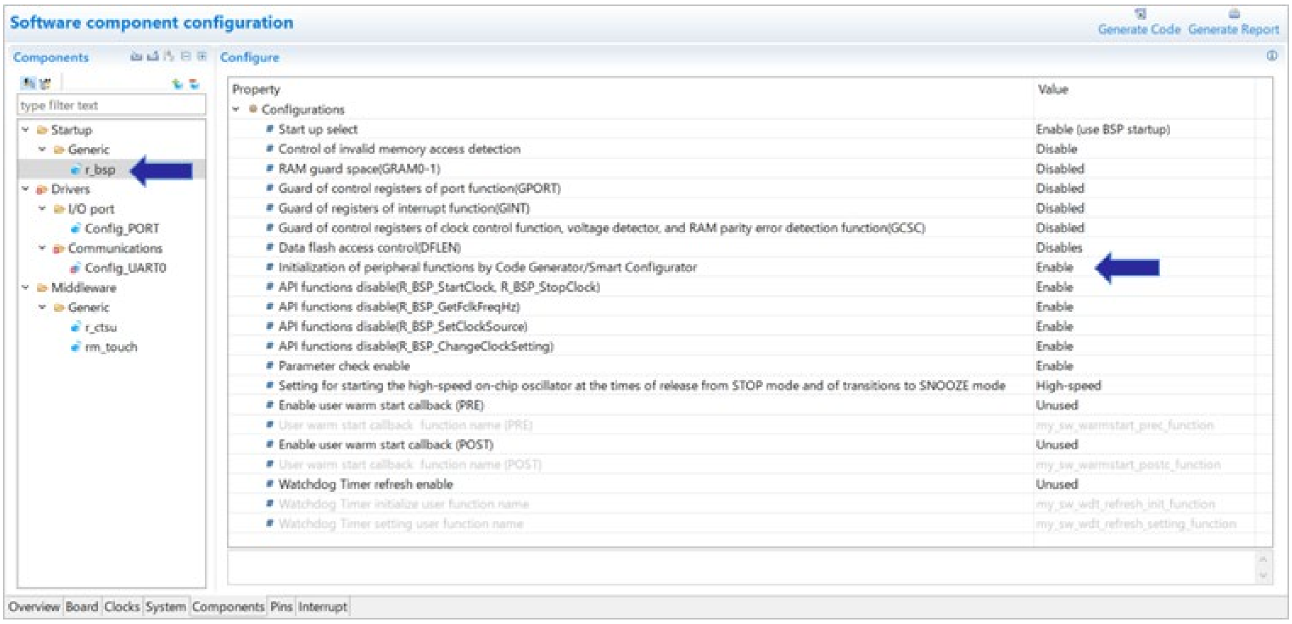


Figure 7-22. Setup of “r_bsp”

2. Click  icon on Smart Configurator to perform generating code.

   When setting of on-chip debugging or option byte is changed, “Confirm linker option change” dialog may be displayed. Confirm the changes and click “OK”.

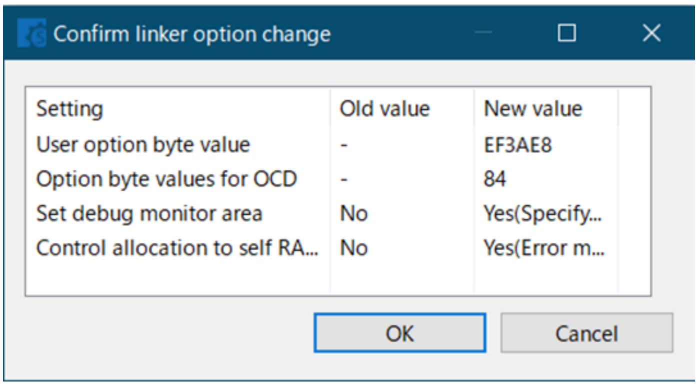


Figure 7-23. Confirm Linker Option Change

8. Setup of QE for Capacitive Touch

8.1 Launching QE for Capacitive Touch

Launch standalone version QE for Capacitive Touch (QE).

1. Launch QE by “QE-CapTouch (install folder of QE) / eclipse / qe-captouch.exe”.

2. Figure 8-1 shows the window of QE after launching.

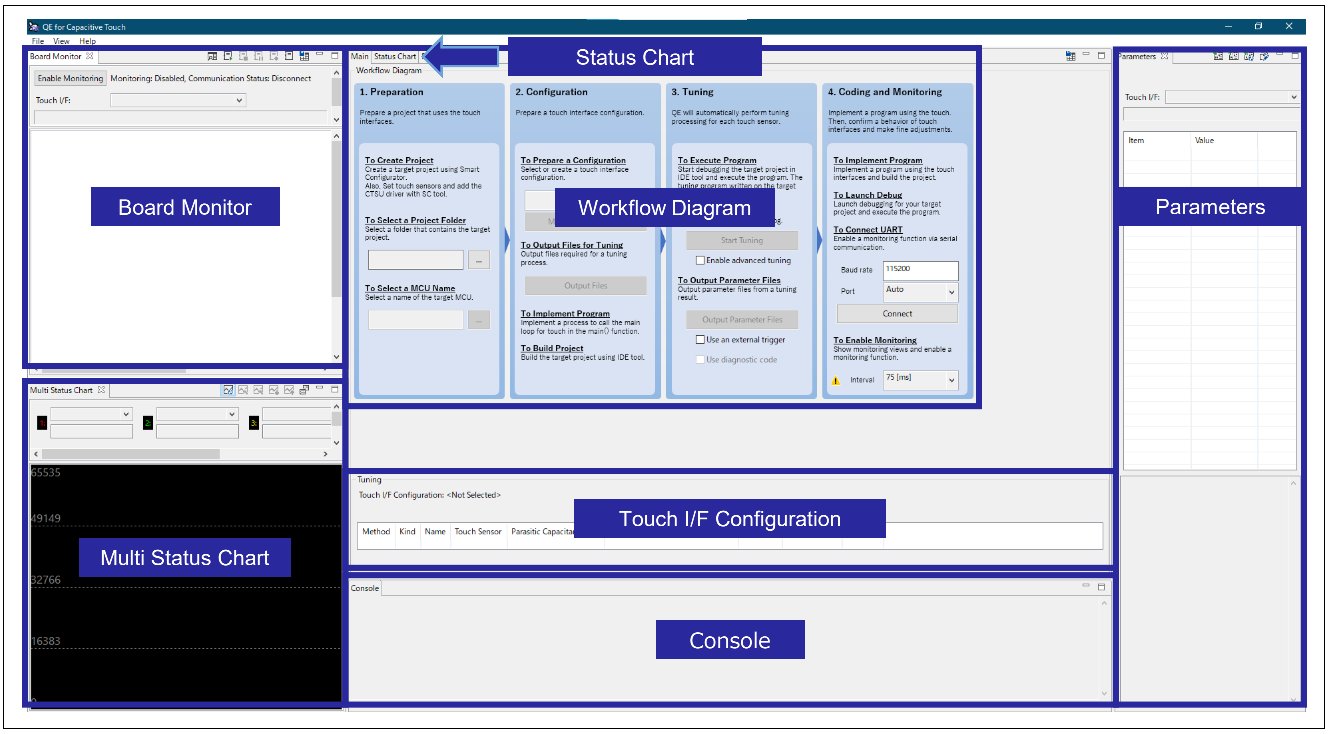


Figure 8-1. QE Window after Launching

If the layout in full window collapses, set layout of Windows to 100% by Windows setting.

8.2 Preparation

Set items according to “Preparation” of Workflow Diagram at middle of QE window.

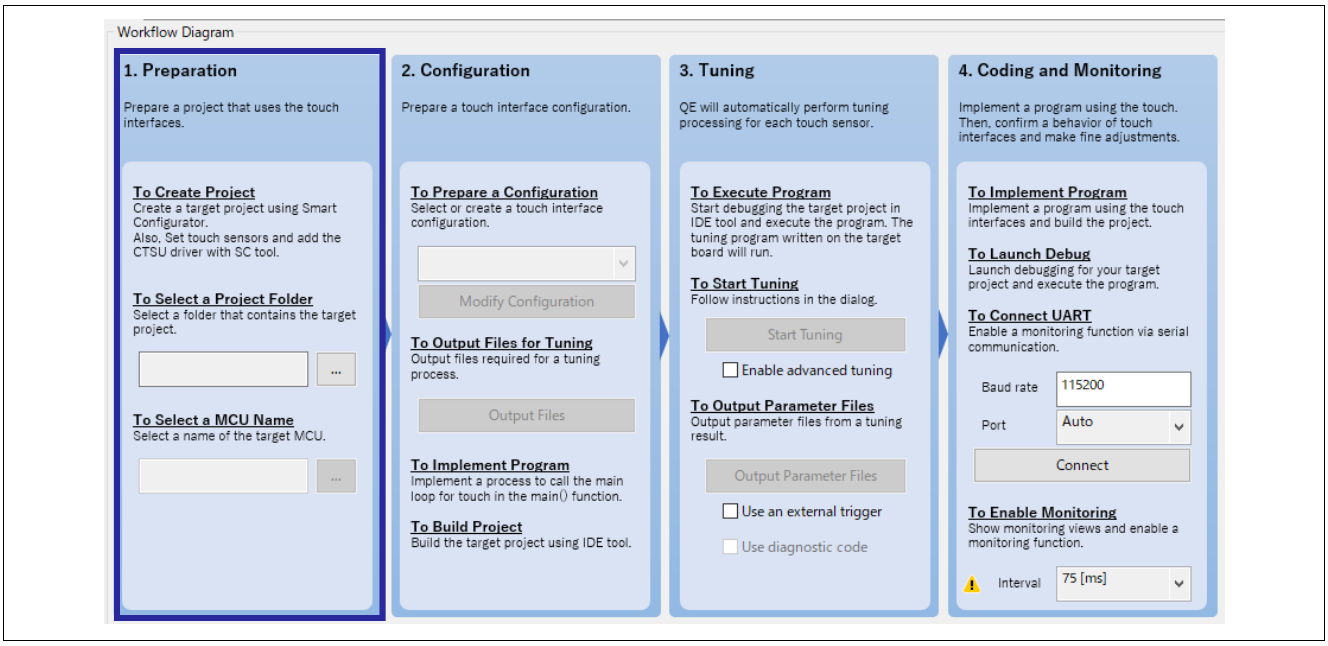


Figure 8-2. Workflow Diagram (Preparation)

1. Click “…” under “To Select a Project Folder” and select your project folder created by CS+.

2. Click “…” under “To Select a MCU Name” and select your using microcontroller.

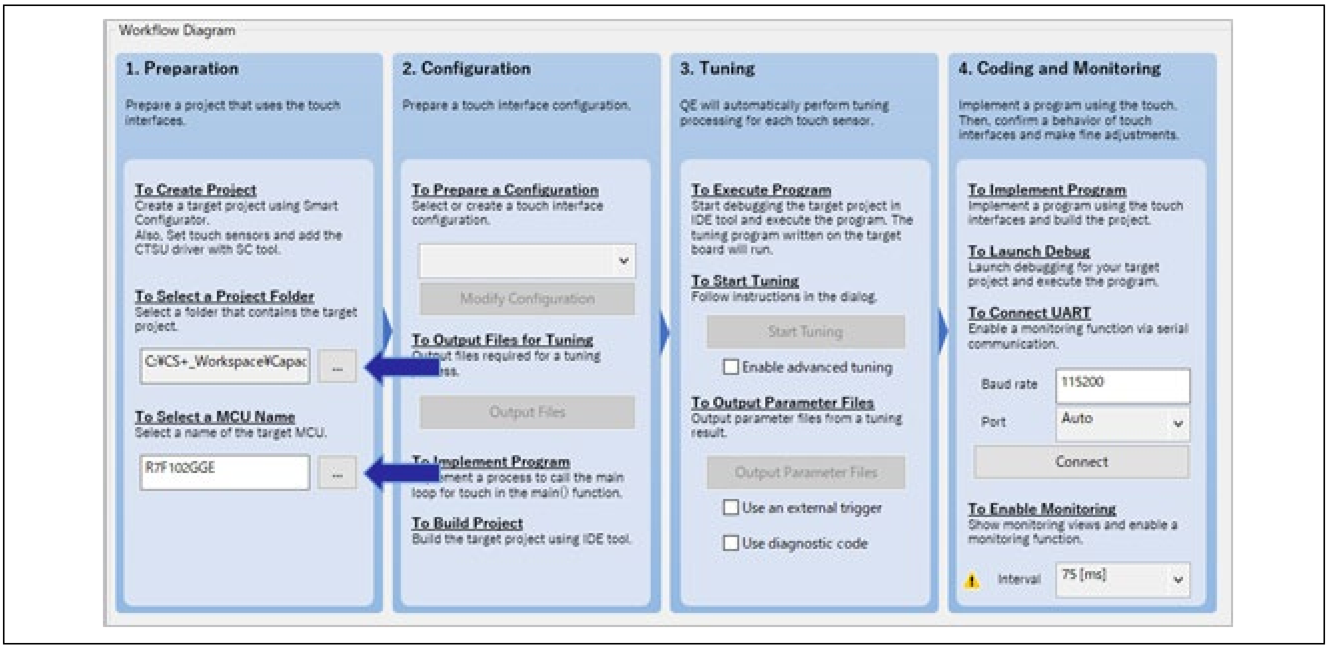


Figure 8-3. Preparation

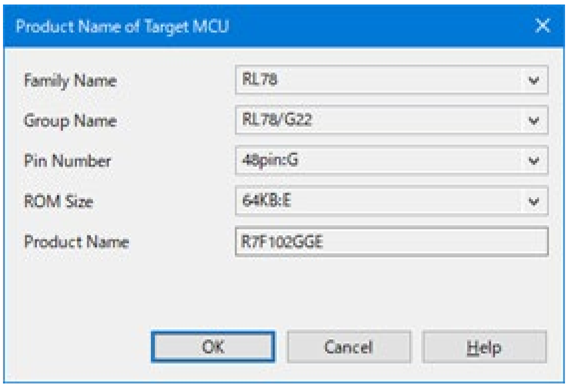


Figure 8-4. To Select a MCU Name

If the following error is occurred by "To Select a MCU Name", the place of QE install folder may be incorrect. Stop QE, move the install folder to other place such as in the directory of "C:\Renesas" and launch QE.

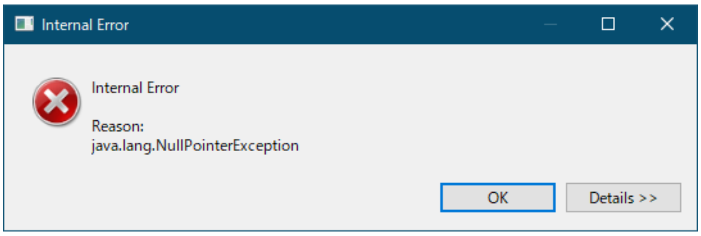


Figure 8-5. Internal Error

8.3 Configuration

Set items according to “configuration” of Workflow Diagram.

Figure 8-6. Workflow Diagram (Configuration)

1. Click icon under “To Prepare a Configuration” and select “Create a new configuration”.

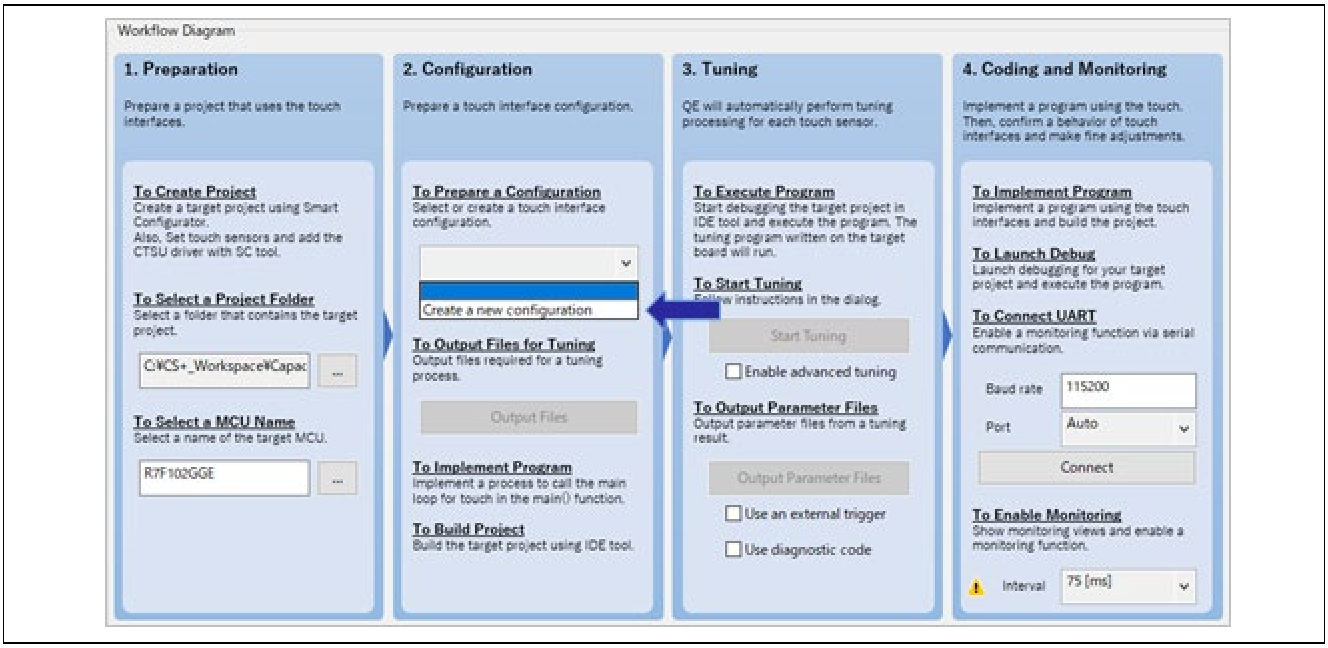


Figure 8-7. Create a New Configuration

2. “Create Configuration of Touch Interfaces” window appears and displays the area for setting touch interface.
   Click “Button” in the “Touch I/F” panel on the right to enable the cursor for button placement, then click wherever you wish to place a button.
   Set two buttons as shown below, and then press the “Esc” key to cancel the button positioning function.
   In the same manner, click “Slider (horizontal)” in the “Touch I/F" panel to use the cursor to place a slider; click anywhere within the area you wish to place the slider. To cancel the slider positioning function, press the “ESC” key.

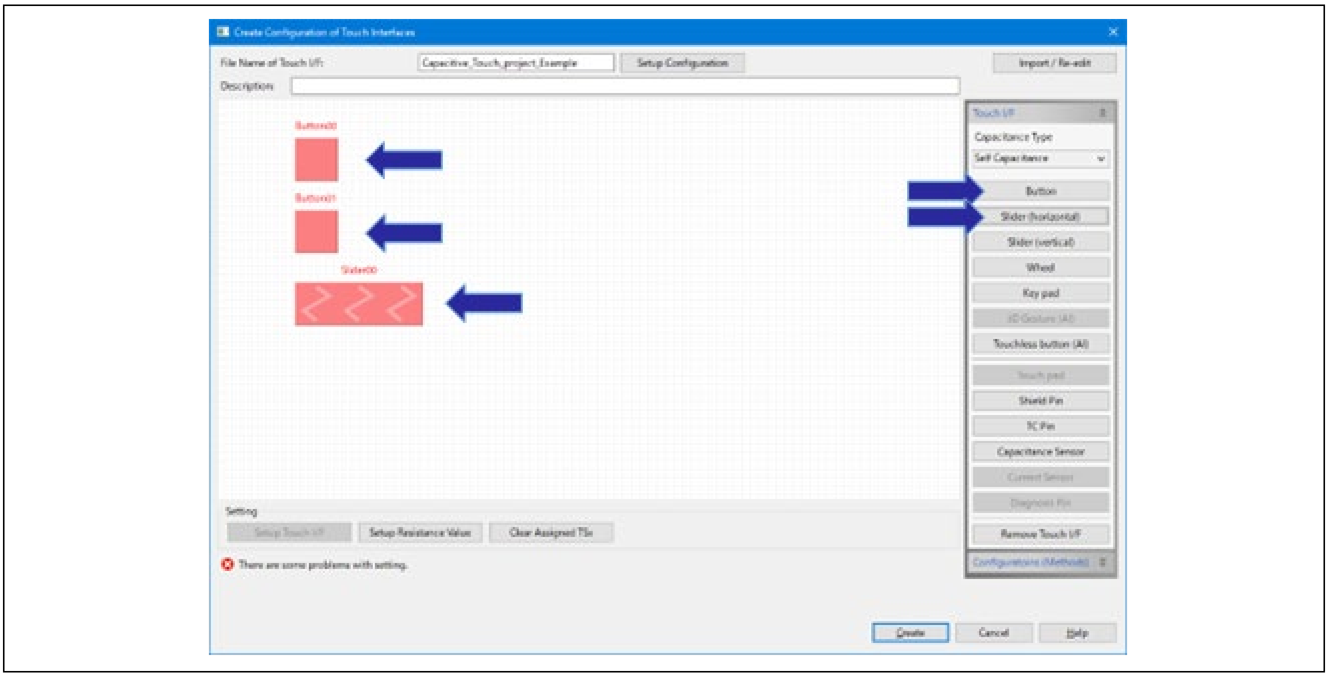


Figure 8-8. Adding Buttons and Slider

3. Double click the “Button00” created in the previous step and set as follows in “Setup Touch Interface” dialog.
   — Touch Sensor : TS24
   — Resistance[Ω] : 560

For the resistance value, please see user's manual or circuit diagram of the target board.

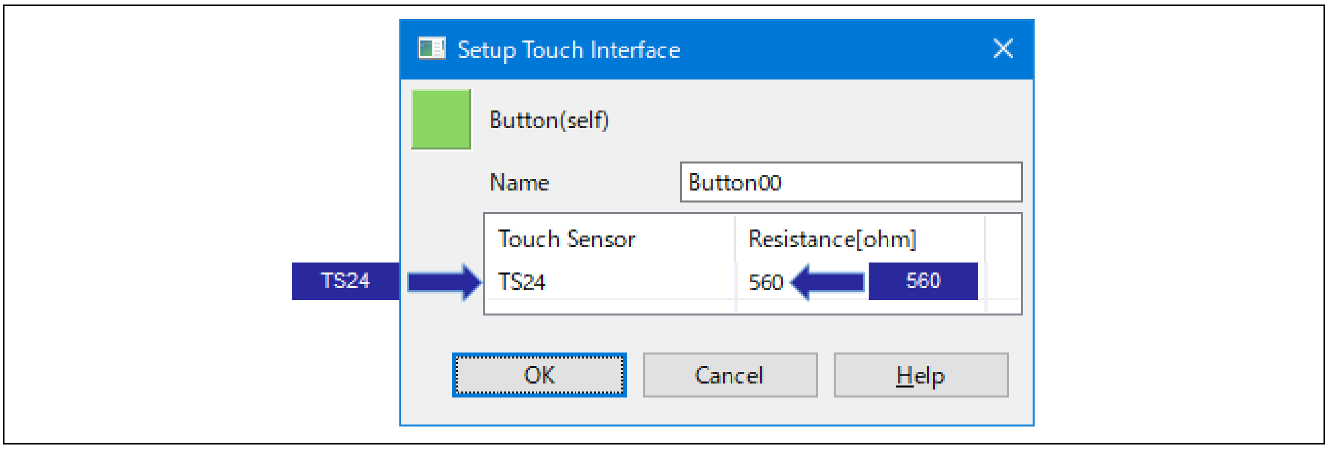


Figure 8-9. Setup of Touch Interface (Button)

4. Set “Button01” as follows.
   - Touch Sensor : TS23
   - Resistance[Ω] : 560

5. Set “Slider00” as follows.
   - Touch Sensor : TS20
     : TS21
     : TS22
   - Resistance[Ω] : 560

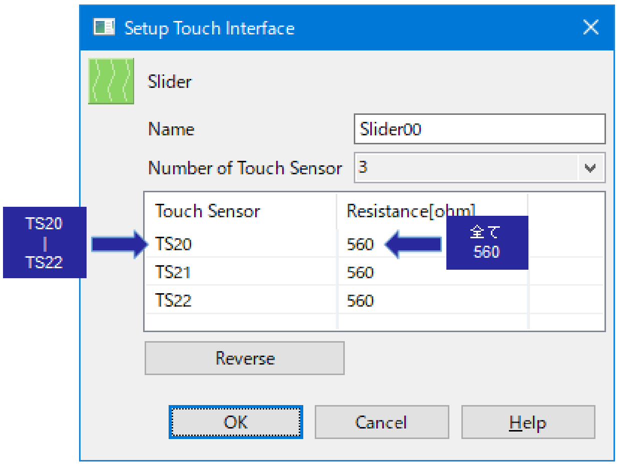


Figure 8-10. Setup of Touch Interface (Slider)

6. After setting touch interface, the area should look as follows. To complete the settings, click “Create”.

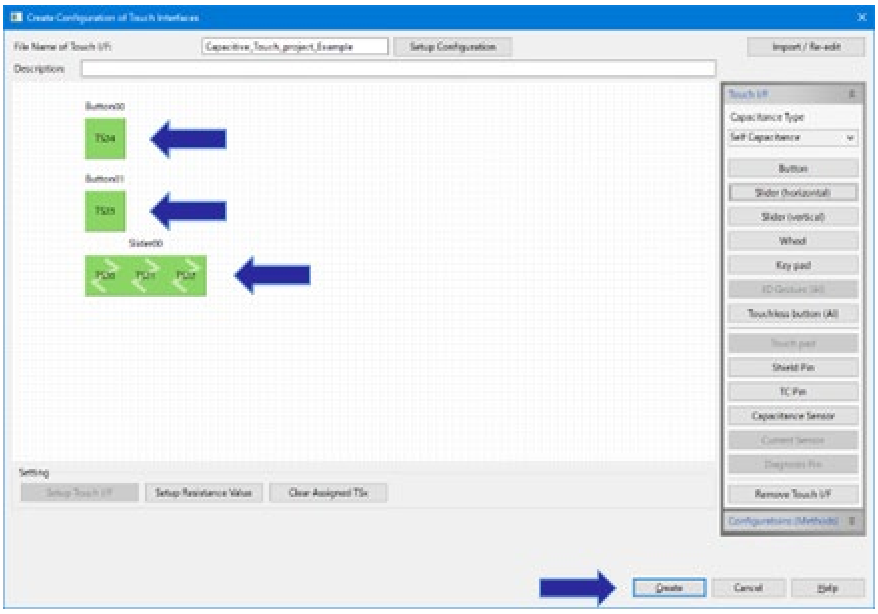
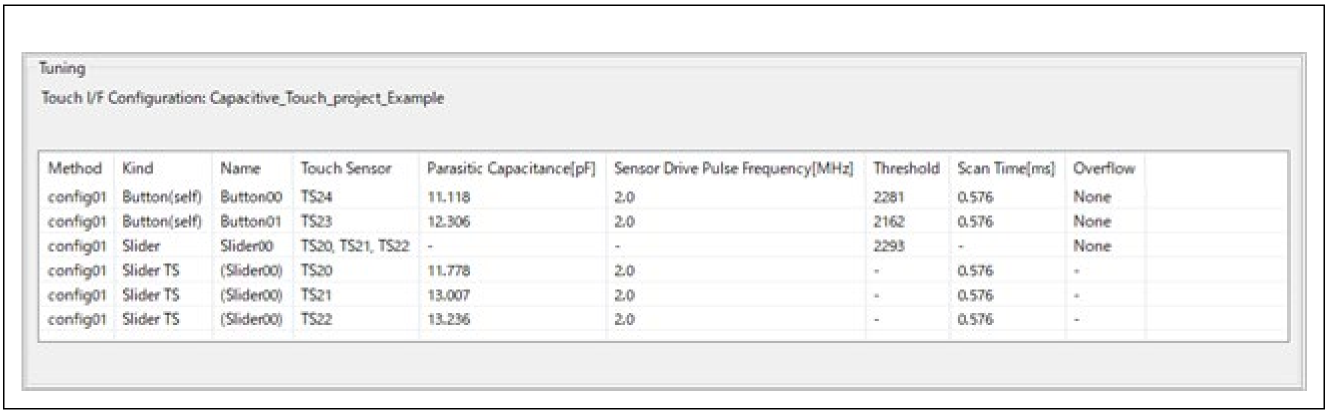


Figure 8-11. Touch Interface Configuration after Setting

7. “Touch I/F Configuration” is displayed in “Tuning” panel.



Tuning

Touch I/F Configuration: Capacitive_Touch_project_Example

Method	Kind	Name	Touch Sensor	Parasitic Capacitance[pF]	Sensor Drive Pulse Frequency[MHz]	Threshold	Scan Time[ms]	Overflow
config01	Button(self)	Button00	TS24	11.118	2.0	2281	0.576	None
config01	Button(self)	Button01	TS23	12.306	2.0	2162	0.576	None
config01	Slider	Slider00	TS20, TS21, TS22	-	-	2293	-	None
config01	Slider TS	(Slider00)	TS20	11.778	2.0	-	0.576	-
config01	Slider TS	(Slider00)	TS21	13.007	2.0	-	0.576	-
config01	Slider TS	(Slider00)	TS22	13.236	2.0	-	0.576	-

Figure 8-12. **Touch Interface configuration**

8. Click “Output Files” and select folder for the output files. Create new folder “qe_gen” under “Capacitive_Touch_Project_Example/src” and output them to the folder.
   The following is the configuration of the folder including output files.
   Capacitive_Touch_Project_Example ← CS+ Project Folder
   |- src
   |- src_gen
   |- qe_gen ← New Folder
   |- qe_touch_config.c ← Output File
   |- qe_touch_config.h ← Output File
   |- qe_touch_define.h ← Output File
   |- qe_touch_sample.c ← Output File

9. After selecting folder for output files, the following dialog appears. Set clock and click “OK”.

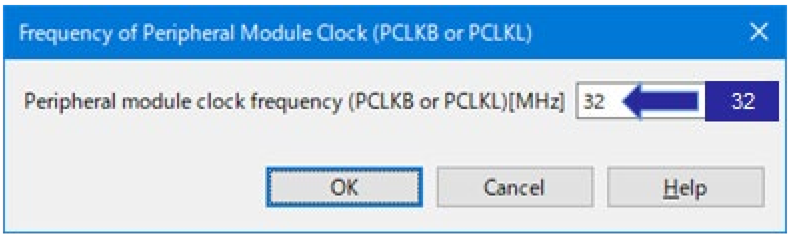


Figure 8-13. Setting Frequency of Peripheral Module Clock

10. In the following dialog, set power supply voltage and click “OK”.
 Please confirm the electric characteristics of the microcontroller you are using.
 When using the RL78/G22, set the power supply voltage of VDD.

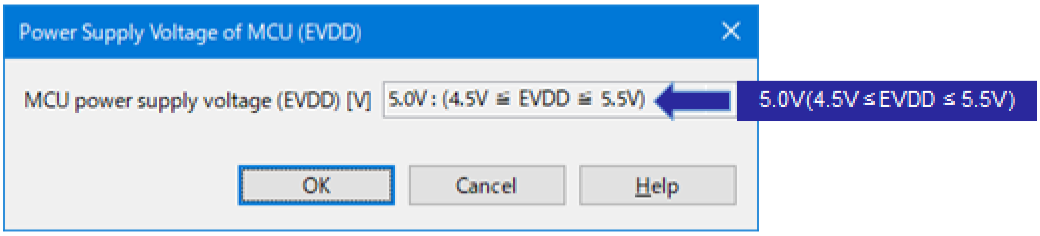


Figure 8-14. Setting Power Supply Voltage of MCU

11. Next, “QE for Capacitive Touch” dialog appears. Follow the instructions of the dialog.
 Also the contents of the dialog is displayed in “Console” panel at lower of QE window.

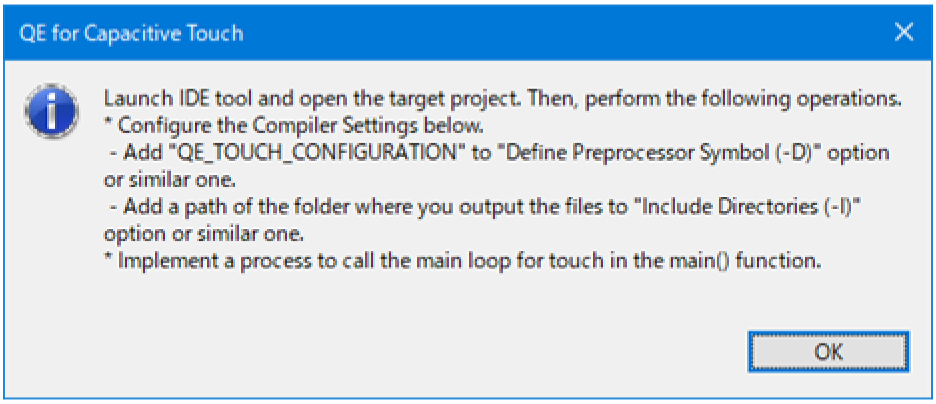


Figure 8-15. QE for Capacitive Touch Dialog

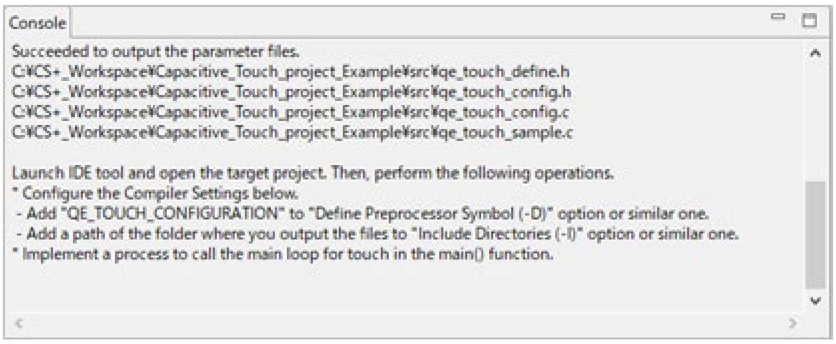


Figure 8-16. Console

A. Set compiler option.
   Select “CC-RL (Build Tool)” in Project Tree of CS+.
   Select “Macro definition” of “Frequency Used Options(for Compile)” in property and click “…” at right side.

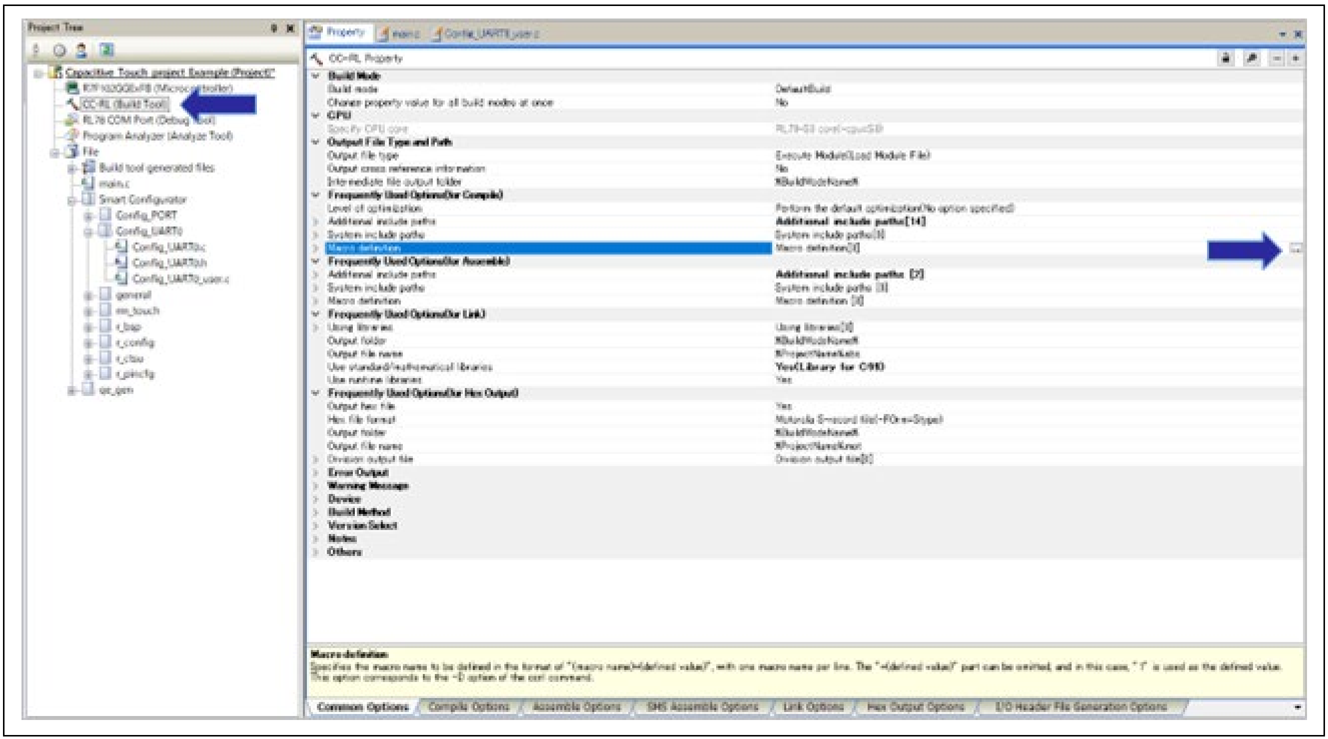


Figure 8-17. Selecting Macro Definition

Add “QE_TOUCH_CONFIGURATION” to text field in “Text Edit” dialog and click “OK”.

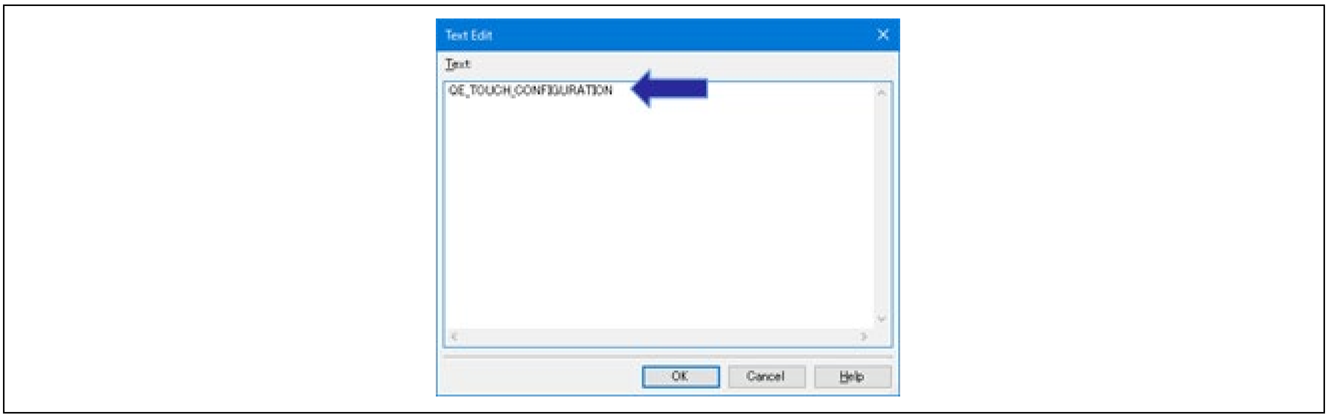


Figure 8-18. Edit Macro Definition

Next, select “Additional include path” of “Frequency Used Options (for Compile)” and click “···” at right side.
Add “src¥qe_gen” to path field in “Path Edit” dialog and click “OK”.

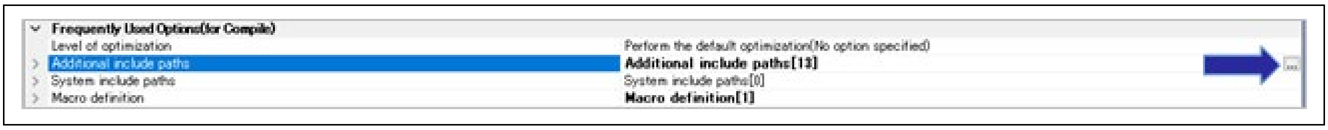


Figure 8-19. Additional Include Paths

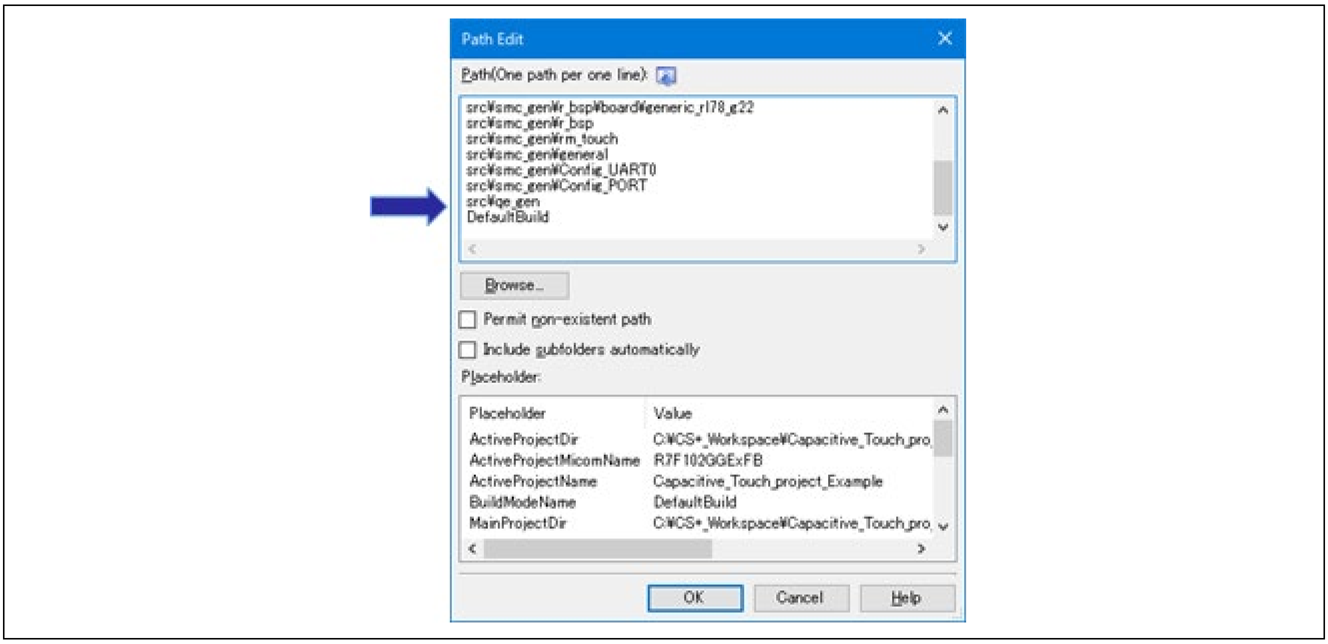


Figure 8-20. Path Edit

Next, click “Use Standard / Math Library” under “Frequently Used Options (links)”.
Click (drop-down arrow) on the right side, and select “Yes (Library for C99)”.

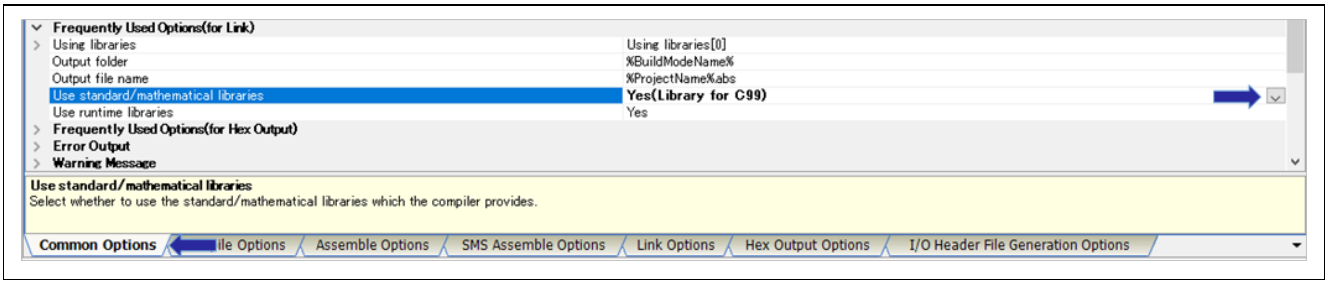


Figure 8-21. Selecting / Standard Math Library

Next, double-click “Source” in “Compile Options”, and then click “Language of the C source file”. Click (drop-down arrow) on the right and select “C99 (-lang=c99)”.

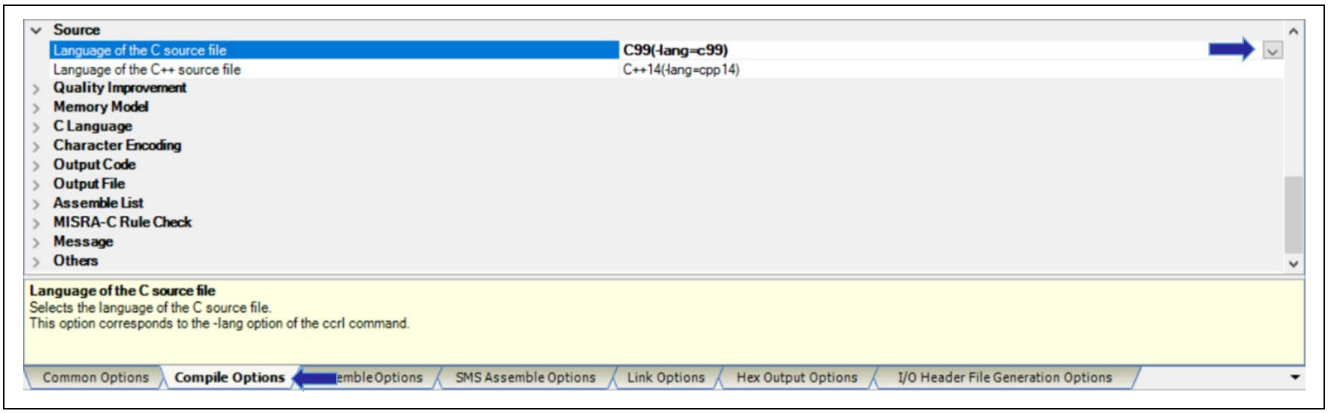


Figure 8-22. Selecting C Language Standard

Next, double-click "Device" in the "Link Options" tab and enter "84" as the "On-Chip Debug Option Byte Control Value”
Similarly, for the "Set debug monitor area”, select "Yes (range setting) (-DEBUG_MONITOR=<address range>)".
For the "User Option Byte Value, "enter "EFFFE8”.
For the option byte value setting, refer to the user's manual of the microcontroller you are using.

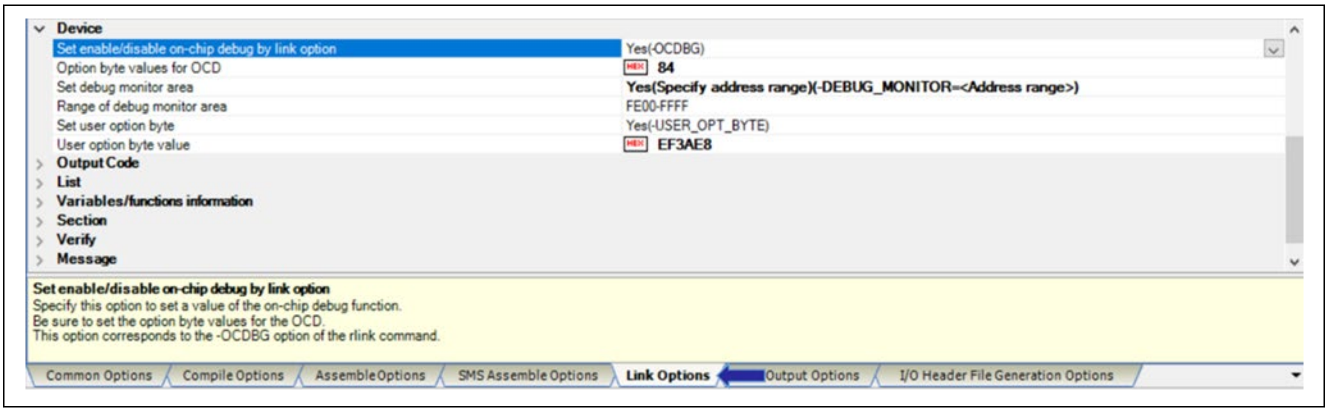


Figure 8-23. Selecting Option Byte

If you use the CC-RL free evaluation edition V1.12.00 or later, select "debug precedence(-onothing)" as the optimization levels.

— Double-click “Optimization” of “Compile Options”, and then click “Level of optimization”.

— Click (drop-down arrow) on the right, and select “debug precedence(-onothing)”.

Remark This optimization setting need only for tuning of touch sensors. After tuning is complete, you can use this application with any optimization settings.

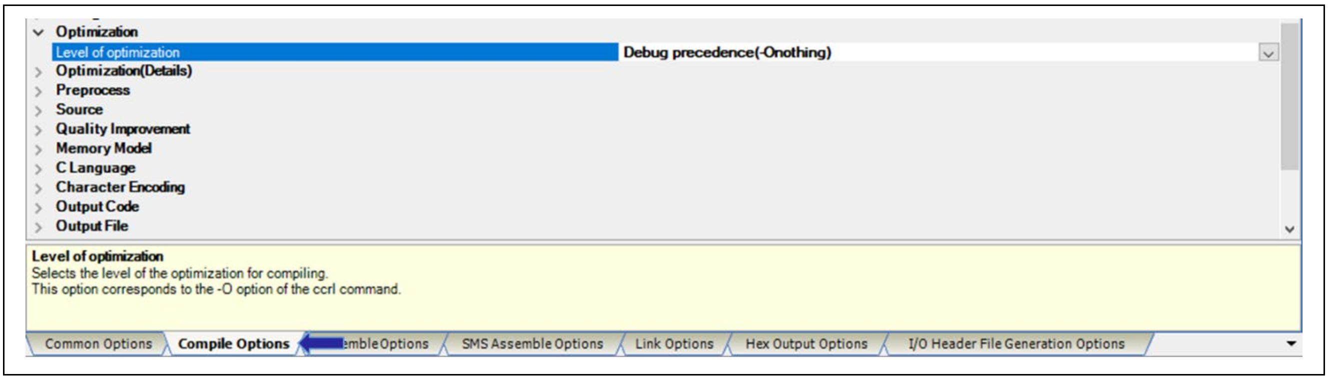


Figure 8-24. Selecting level of optimization

B. Perform coding of touch main function in main() function.
If “qe_gen” folder is not in project tree of CS+, add “qe_gen” folder to project tree from Windows Drag and drop from Explorer.

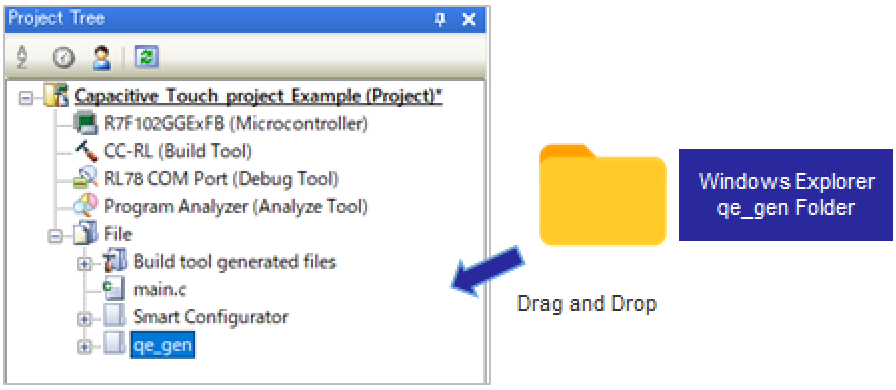


Figure 8-25. Adding “qe_gen” Folder to Project Tree

Call “qe_touch_main()” function in main() function. Add the following code to main.c.

- extern void qe_touch_main(void);
- qe_touch_main();

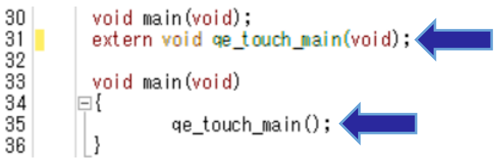


Figure 8-26. main.c

12. Add function for serial communication to “Config_UART0 user.c”.
    Add the following code.
    — extern void touch_uart_callback(uint16_t event);
    — touch_uart_callback(0);
    — touch_uart_callback(1);

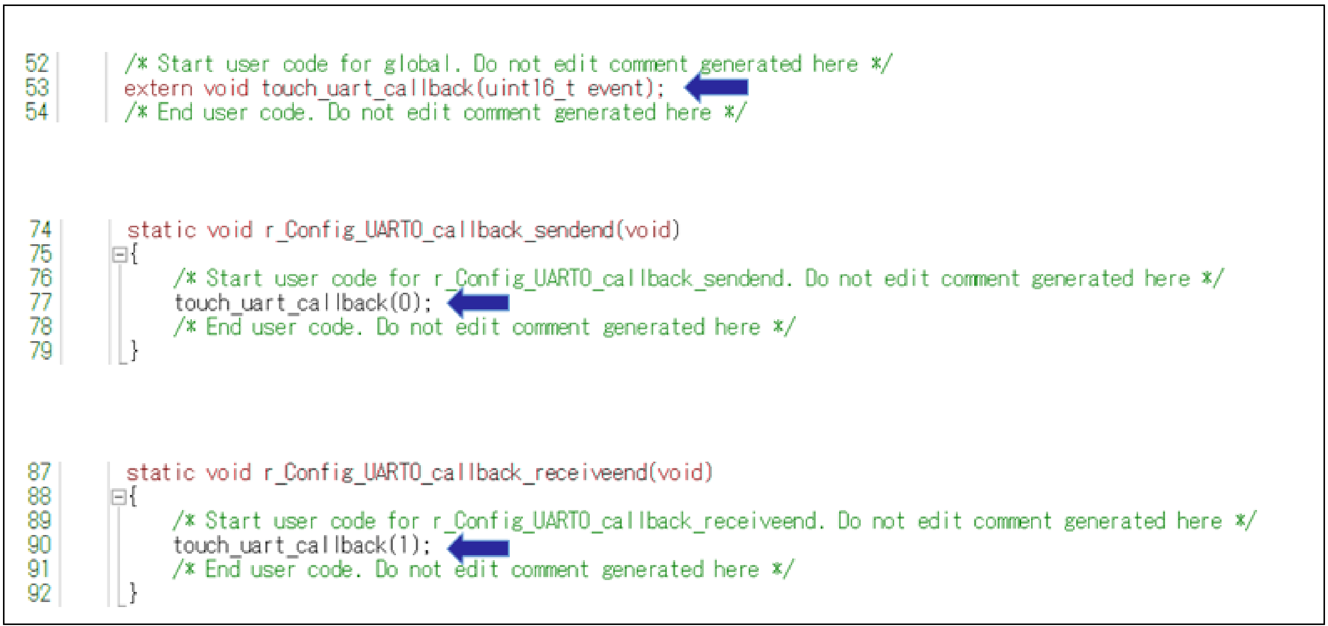


Figure 8-27. Config_URAT0_user.c

13. Build the project by CS+. Click  icon on CS+ and start build. Confirm that build finished without any errors or warning.

    If the following warning (W0511187) is occurred when build the project, change the optimization levels to “debug precedence(-onothing)” as shown in Figure 8-24 on page36. And then, rebuild the project.

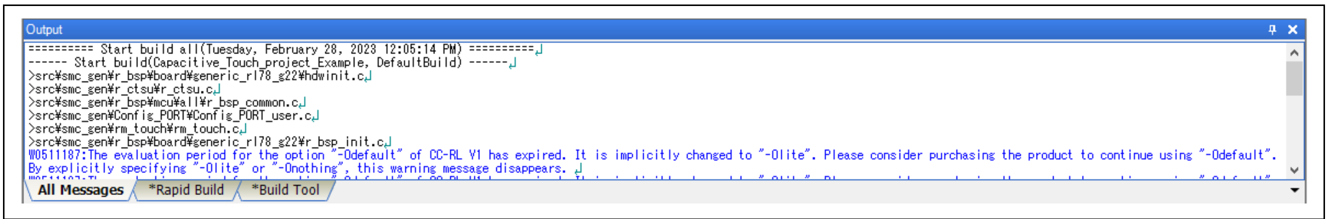


Figure 8-28. Warning (W0511187)

8.4 Tuning

Set according to “Tuning” of Workflow Diagram.

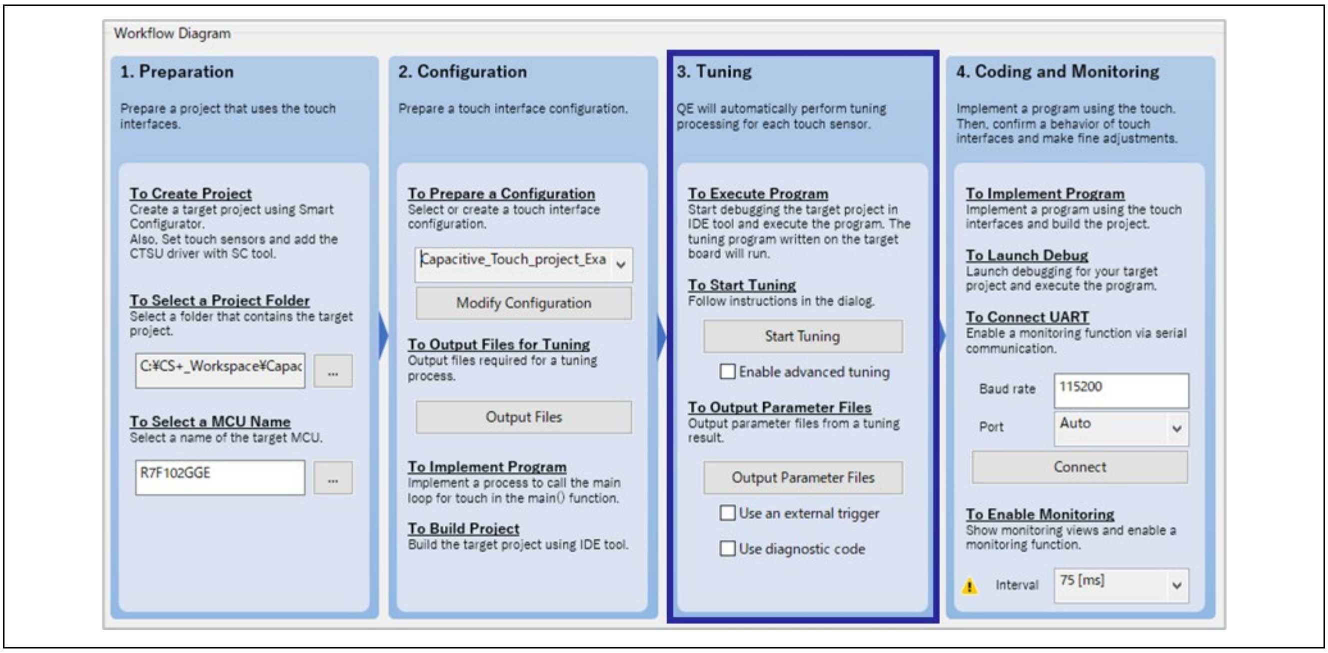


Figure 8-29. Workflow Diagram (Tuning)

1. Right-click “Debug Tool” in “Project Tree” of CS+, and click “Using Debug Tool”. Select RL78 COM Port” as the debug tool you intend to use.

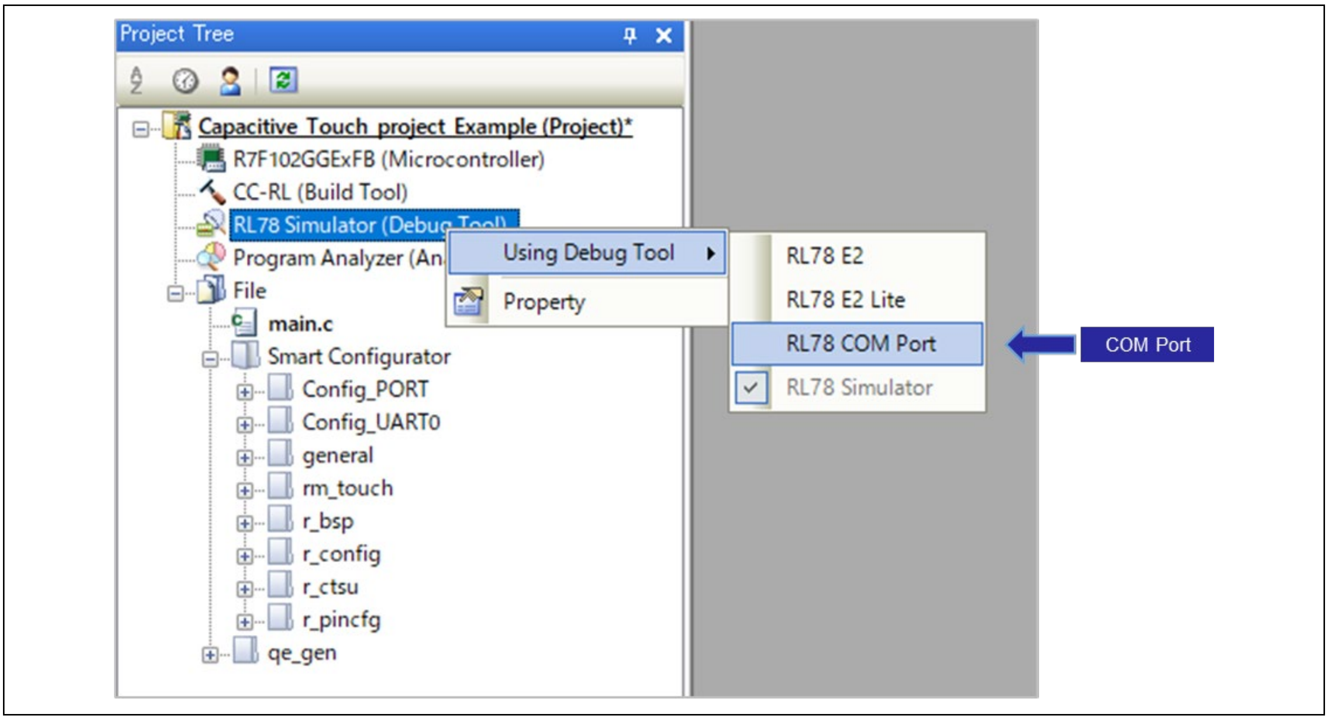


Figure 8-30. Selecting Debug Tool

2. Set the "Communication port" in the "Debug Tool" properties.
   This application example uses COM3 as the communication port.

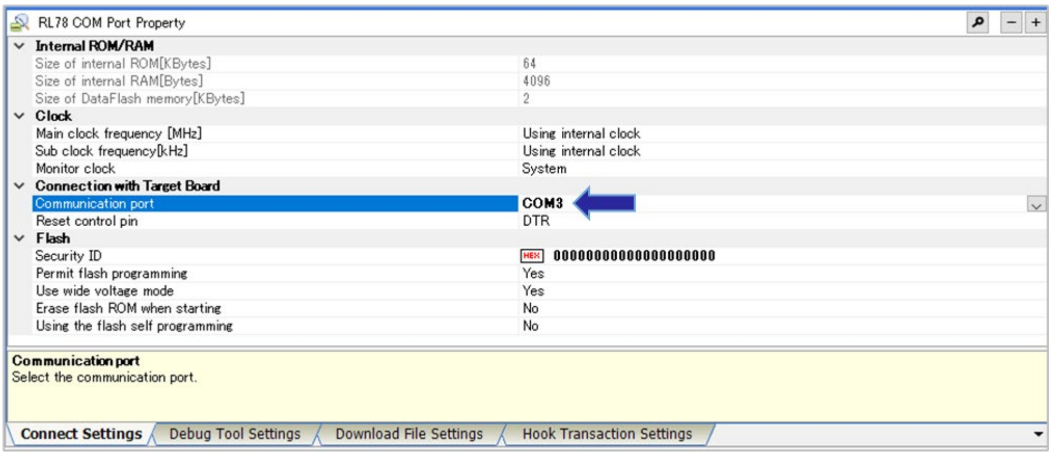


Figure 8-31. Property of Debug Tool

Confirm the communication port setting in the Device Manager.

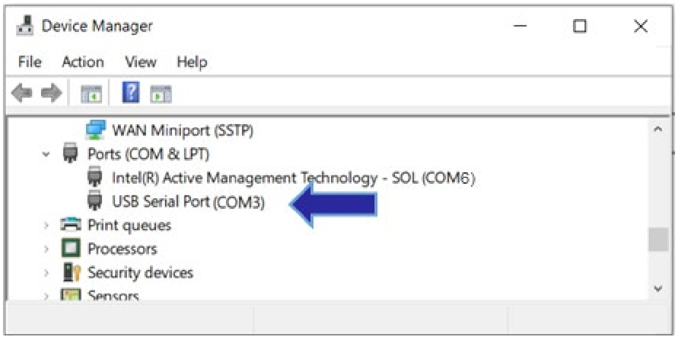


Figure 8-32. Device Manager

3. Confirm that the QE serial connection switching jumper (J16) on the target board is shorted and that the PC and target board are connected with a USB cable, then click the CS+ icon to build and write the program. When the download is complete after writing the program, click the icon to stop the program, and, finally, click the icon to disconnect.

   After disconnection, remove the USB cable connecting the PC and target board, and open the QE serial connection switching jumper (J16).
   Next, reconnect the USB cable between the PC and target board so that you can connect QE. At this time, the target board will be in standby state for connection with QE, while it runs the written program.

   For details regarding the QE serial connection switching jumper (J16), refer to the target board user's manual. Always use a USB cable that supports data transfer.

.

4. On QE, set “Baud rate” of “To Connect UART” to the value which is set in chapter 7.4.

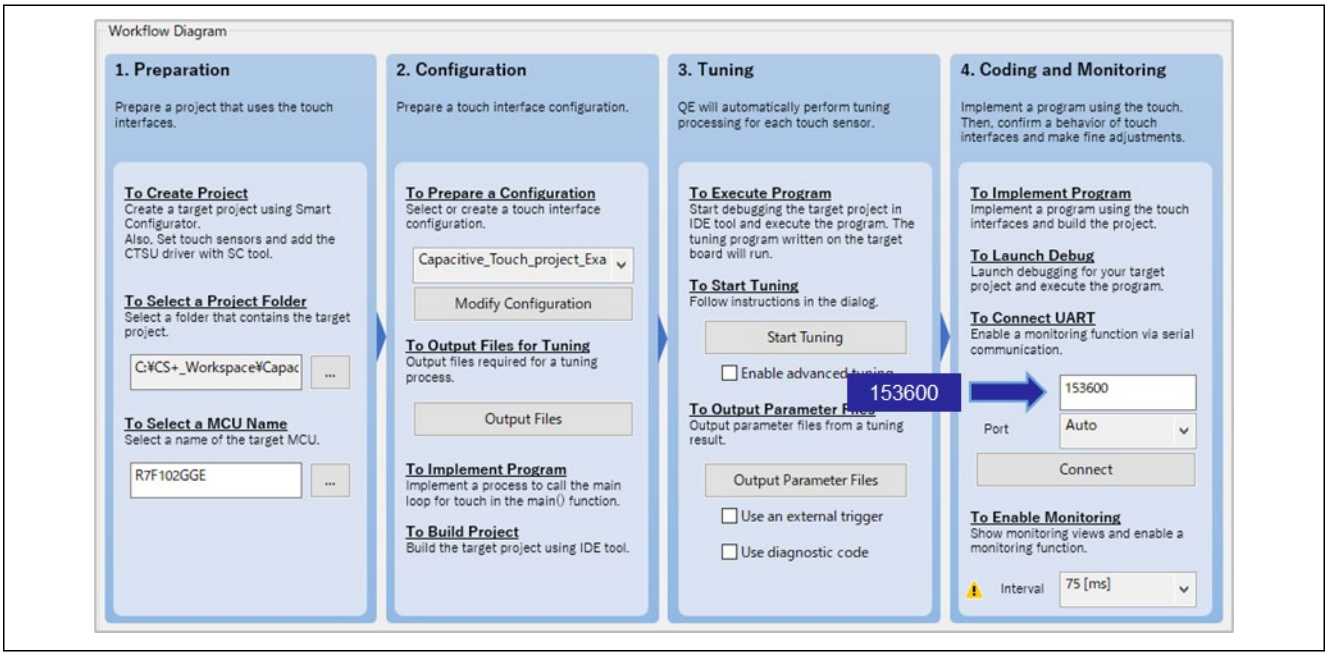


Figure 8-33. Setting Baud Rate

5. Click “Start Tuning”, and start tuning.

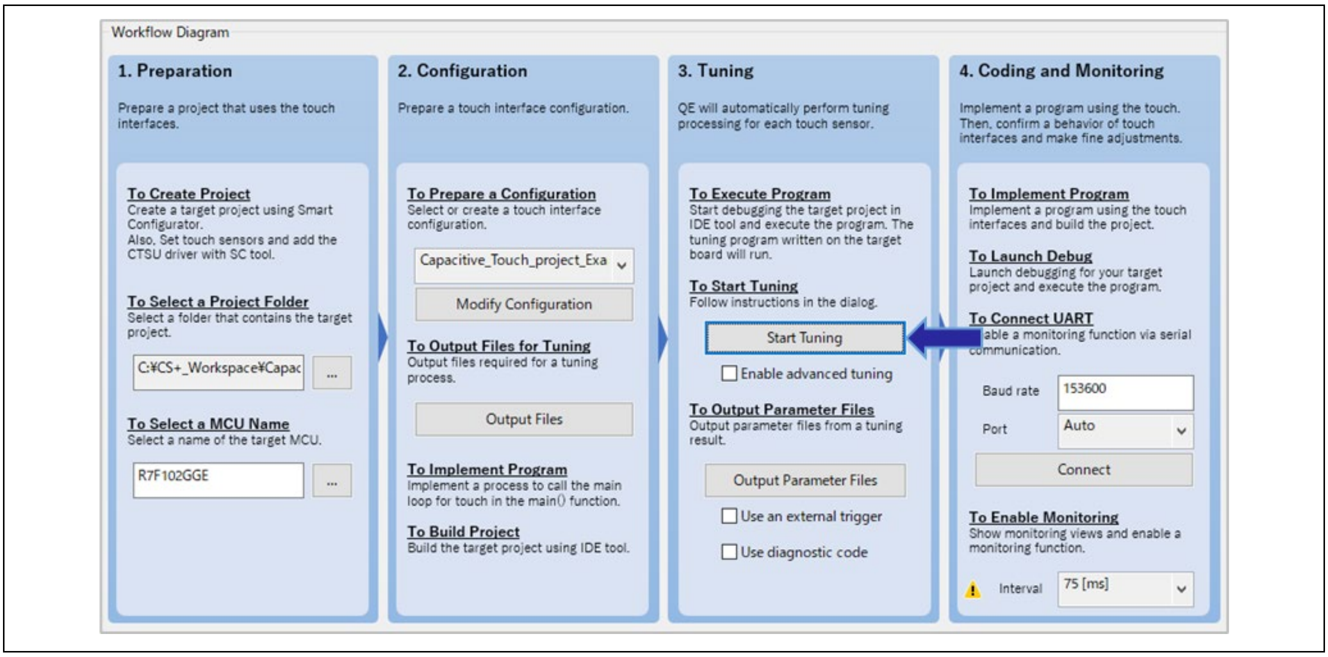


Figure 8-34. Tuning

6. Set baud rate and click “Connect” on the displayed dialog.

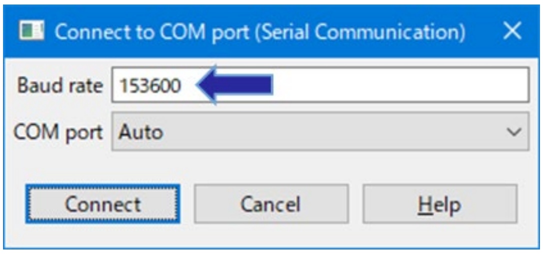


Figure 8-35. Setting Baud Rate

7. In the next dialog, set clock and click “OK”.

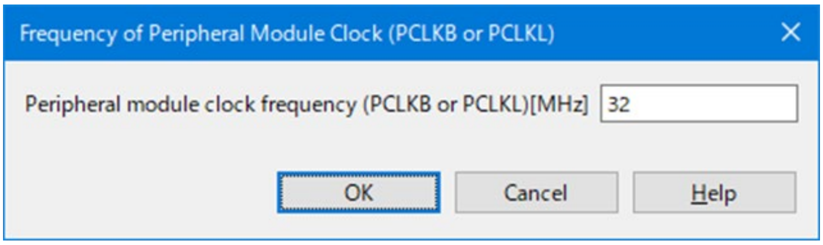


Figure 8-36. Setting Frequency of Peripheral Module Clock

8. In the next dialog, set power supply voltage and click “OK”.
   Please confirm electric characteristic of your using microcontroller.
   When using the RL78/G22, set the power supply voltage of VDD.

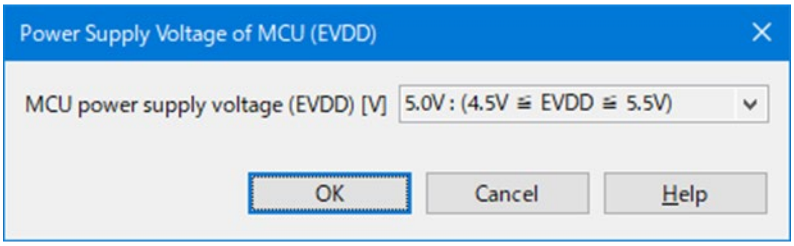


Figure 8-37. Setting Power Supply Voltage of MCU

9. Tuning start. Confirm the contents of “Automatic Tuning Processing” dialog which shows guidelines for tuning process and follow the instructions of the dialog.

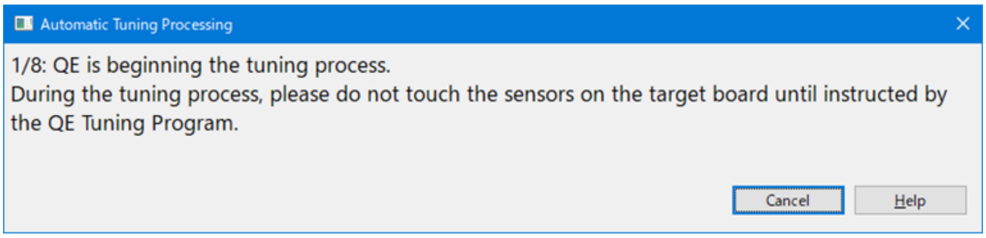


Figure 8-38. Automatic Tuning Processing Dialog

After some steps, the following dialog appears.
This step is for measuring touch sensitivity. Touch with normal pressure the touch sensor indicated in the dialog. While touching the touch sensor, the bar graph will extend to the right and touch counts will increase.
While touching, press any key on the PC keyboard to confirm the sensitivity measurement.

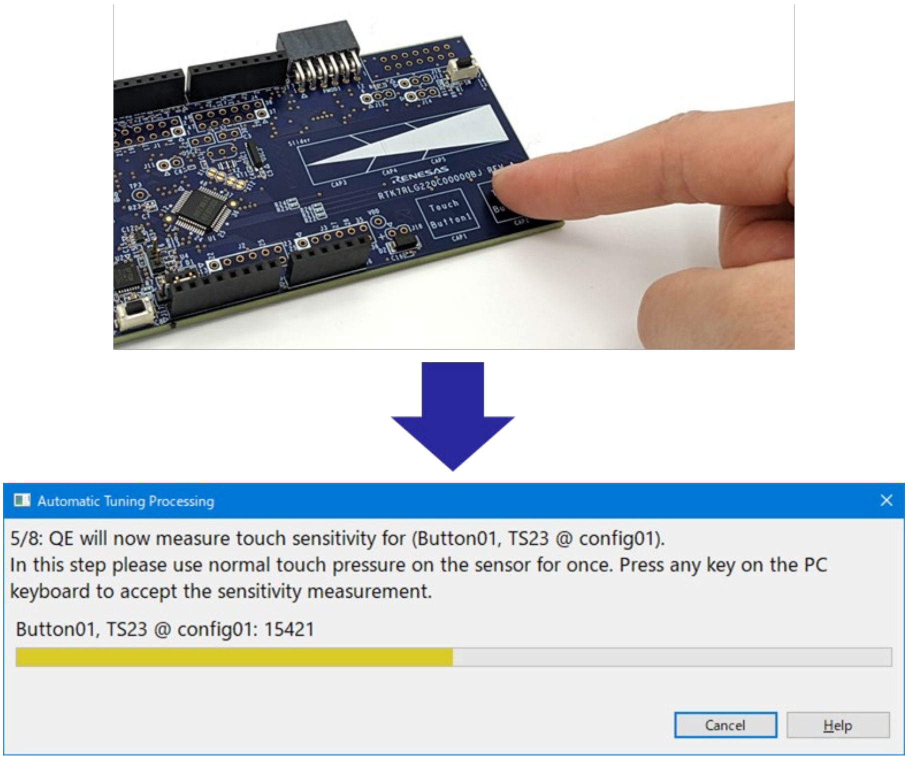


Figure 8-39. Measuring Touch Sensitivity (Button)

10. The touch sensitivity of the other touch sensor can be measured in the same manner

11. Touch sensitivity can also measured for the slider touch sensor. After tracing the slider on the target board 3 or 4 times up and down or left and right with normal pressure, keep your finger on the slider and press any key on the PC keyboard to confirm the measurement.

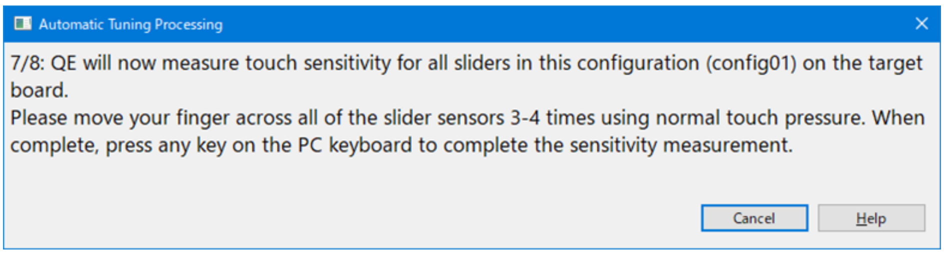


Figure 8-40. Measuring Touch Sensitivity (Slider)

12. The threshold can be confirmed in the following dialog, which appears when tuning is completed. This threshold is used to determine touch events in the middleware.
After confirming the threshold, click “Continue the Tuning Process”. This completes automatic tuning.

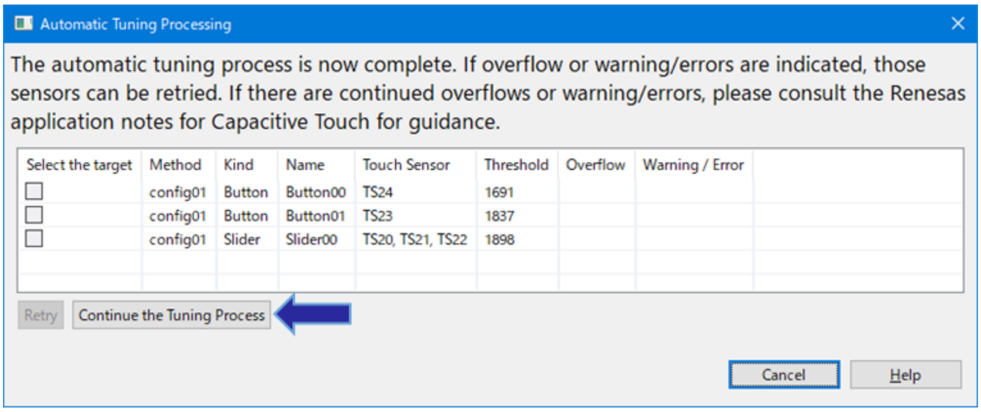


Figure 8-41. Threshold of Touch Sensor

13. 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

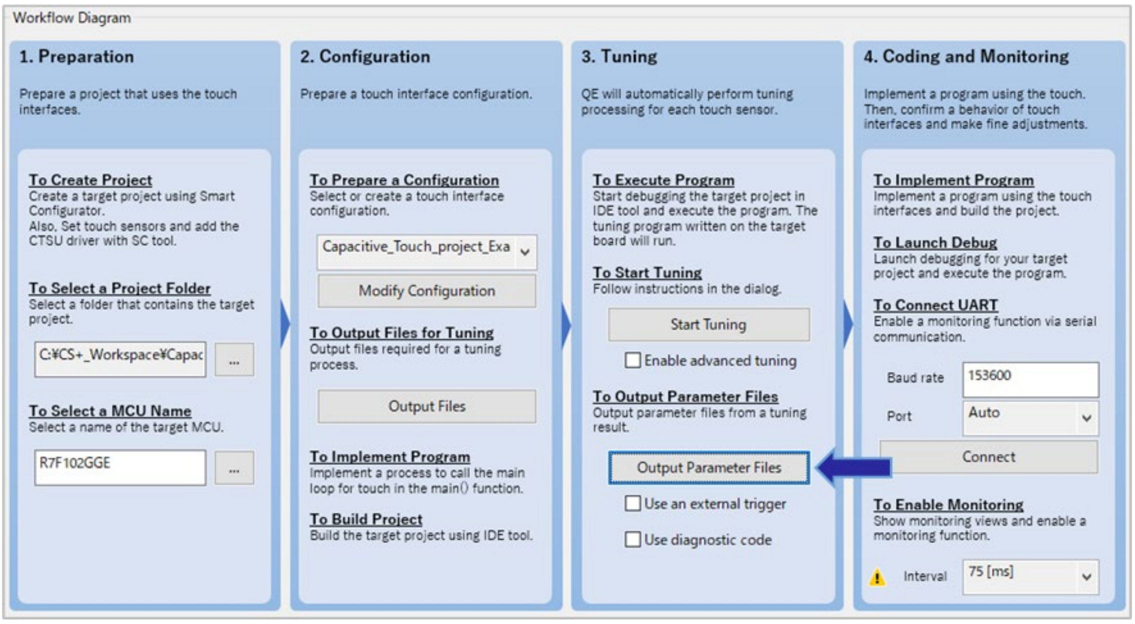


Figure 8-42 To Output Parameter Files

8.5 Coding and Monitoring

8.5.1 Monitoring

Set according to “Coding and Monitoring” of Workflow Diagram.

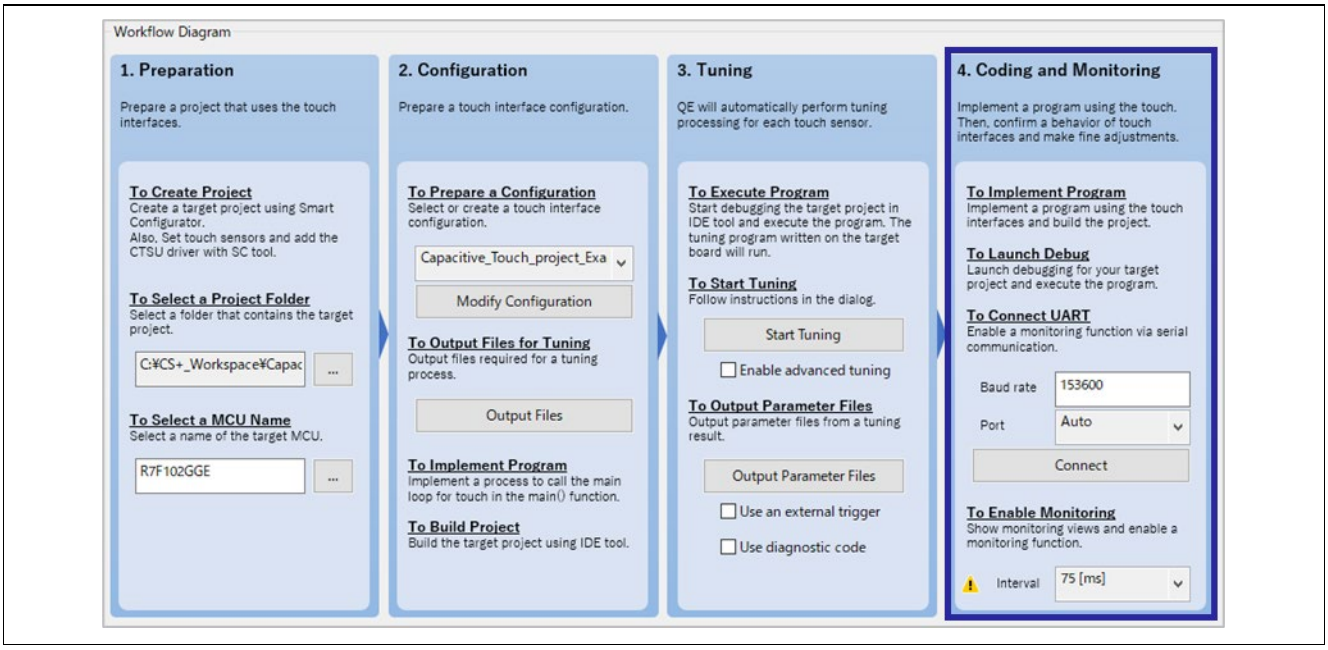


Figure 8-43. Workflow Diagram (Coding and Monitoring)

1. Remove the USB cable connecting the PC and target board, and short the QE serial switching jumper (J16). Next, reconnect the USB cable between the PC and target board so that you can connect CS+.

   Click the CS+ icon to build and write the program. When the download is complete after writing the program, click the icon to stop the program, and, finally, click the icon to disconnect.

   After disconnection, remove the USB cable connecting the PC and target board, and open the QE serial connection switching jumper (J16).
   Next, reconnect the USB cable between the PC and target board so that you can connect QE. At this time, the target board will be in standby state for connection with QE, while it runs the written program.

2. Click “Connect”. “Connect” changes to “Disconnect”.

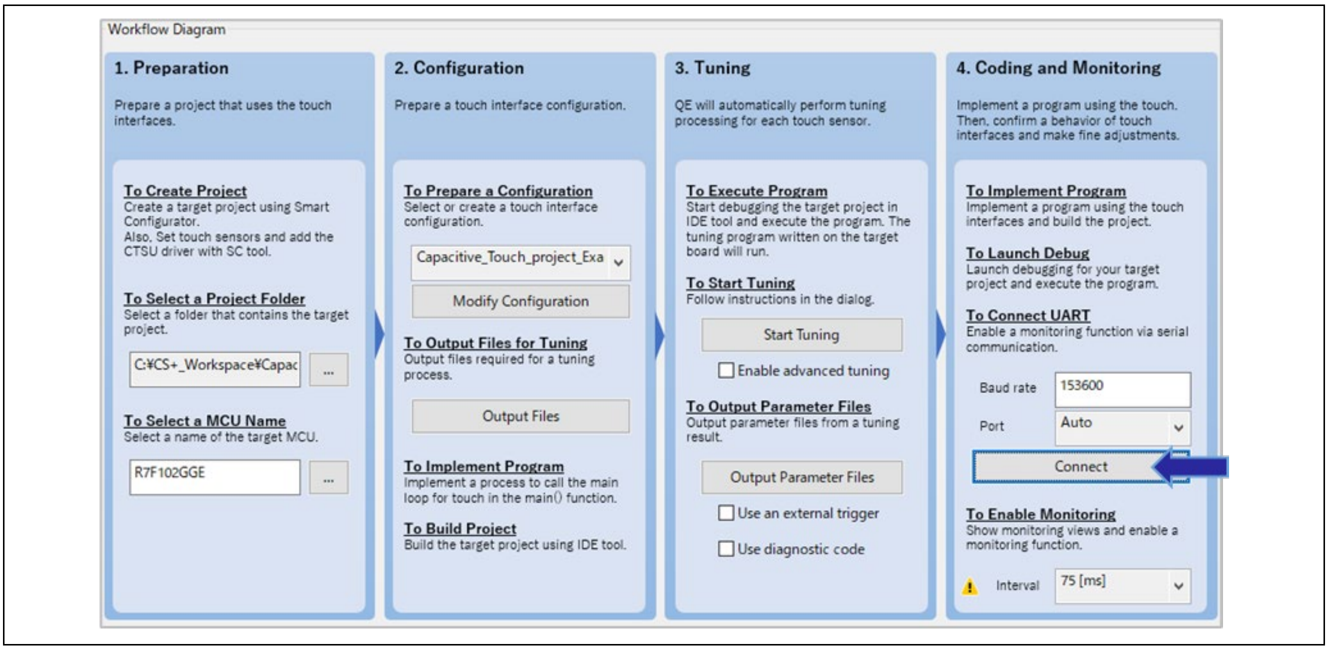


Figure 8-44. To Connect UART

3. Click "Enable Monitoring" of "Board Monitor" panel at top left of QE window. "Monitoring: Disabled" changes to "Monitoring: Enabled".

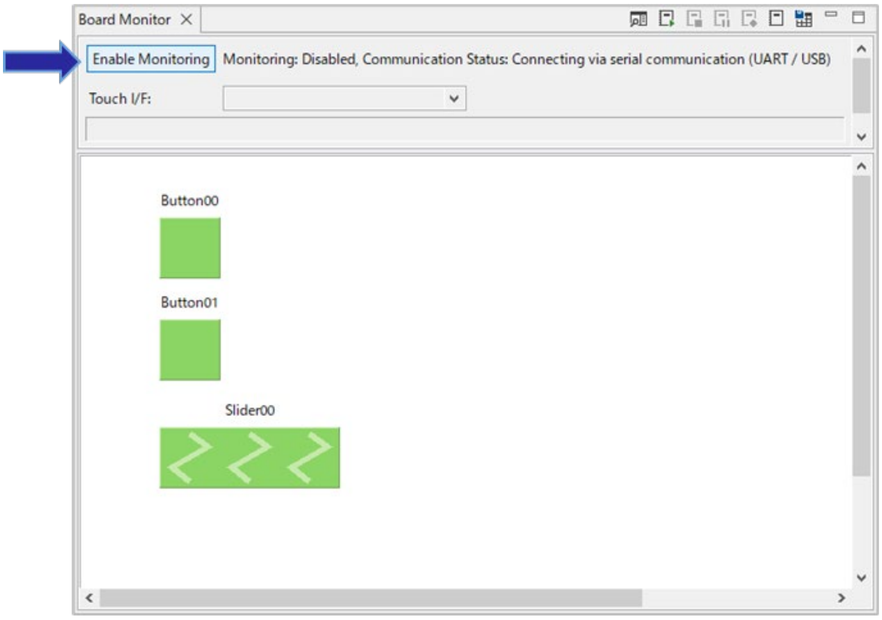


Figure 8-45. Enable Monitoring

4. While touching the touch sensor, the finger icon shows the state of touch sensor.

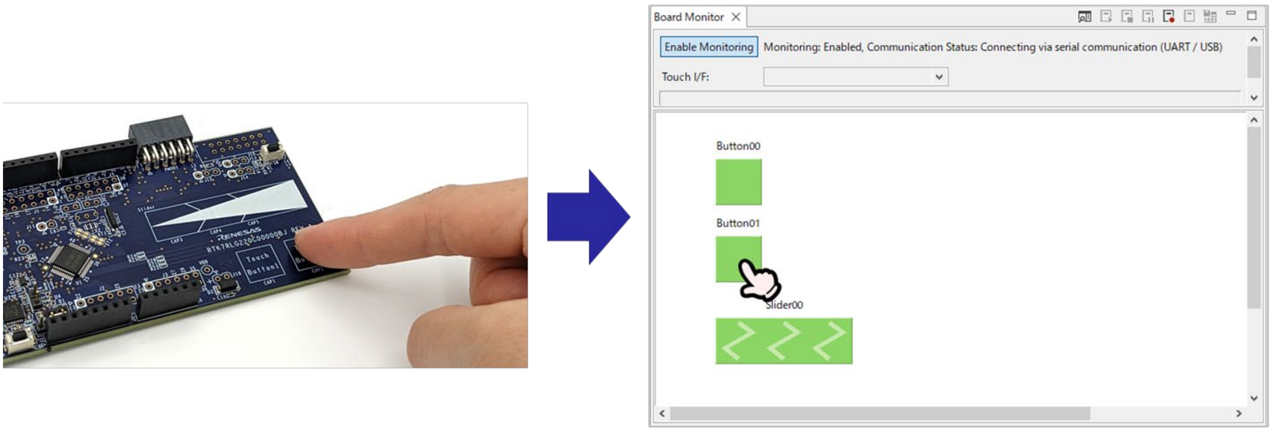


Figure 8-46. Display of the Condition while Touching

5. Represent a graph of the touch counts
   A. Click “Status Chart” tab at the panel including “Workflow Diagram”.

   B. Click icon of “Touch I/F” at “Status Chart” window and select touch interface.

   C. The Graph shows real-time value of the touch sensor. When touching the touch sensor, touch counts change on the graph.
   The green line shows the threshold, which “rm_touch” middleware uses to determine whether the touch sensor is actuated/touched.
   The red belt at the bottom of the graph shows that touch counts is over the threshold and the touch sensor is being touched.

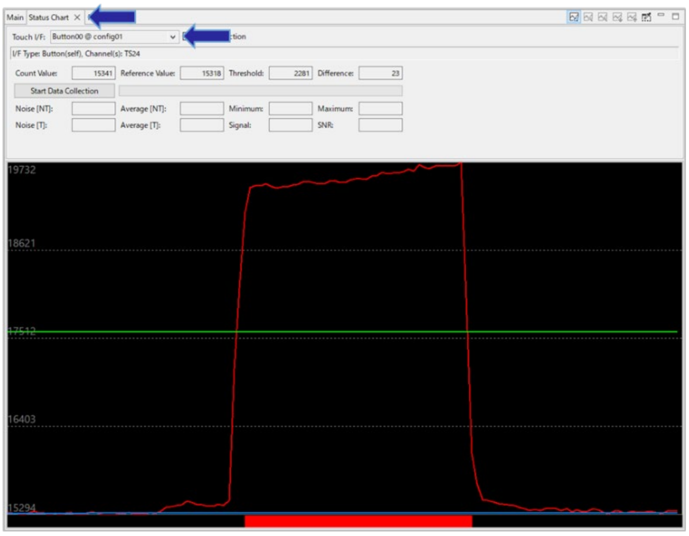


Figure 8-47. Graphical Representation of Touch Counts (Button)

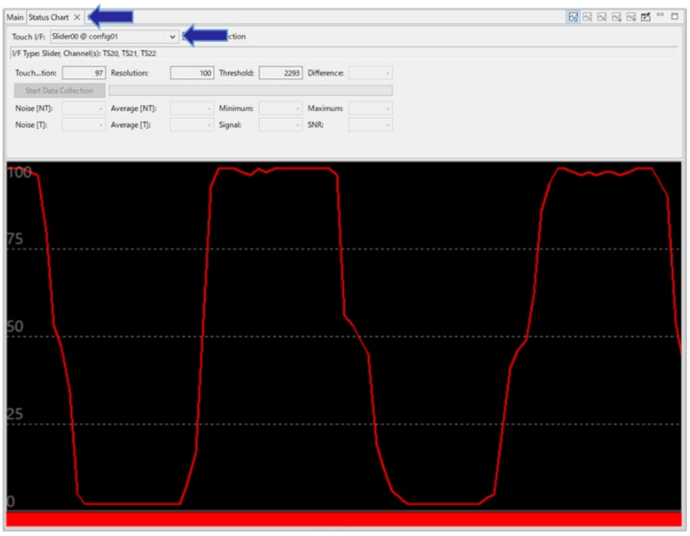


Figure 8-48. Graphical Representation of Touch Counts (Slider)

6. As necessary, measure standard deviation.
   A. Click “Start Data Collection” without touching. Don’t touch the touch sensor while measuring the value in the state of touch-off.
      The green bar shows the rate of the data collection. When the green bar goes all the way to the right, the data collection for touch-off state is done.

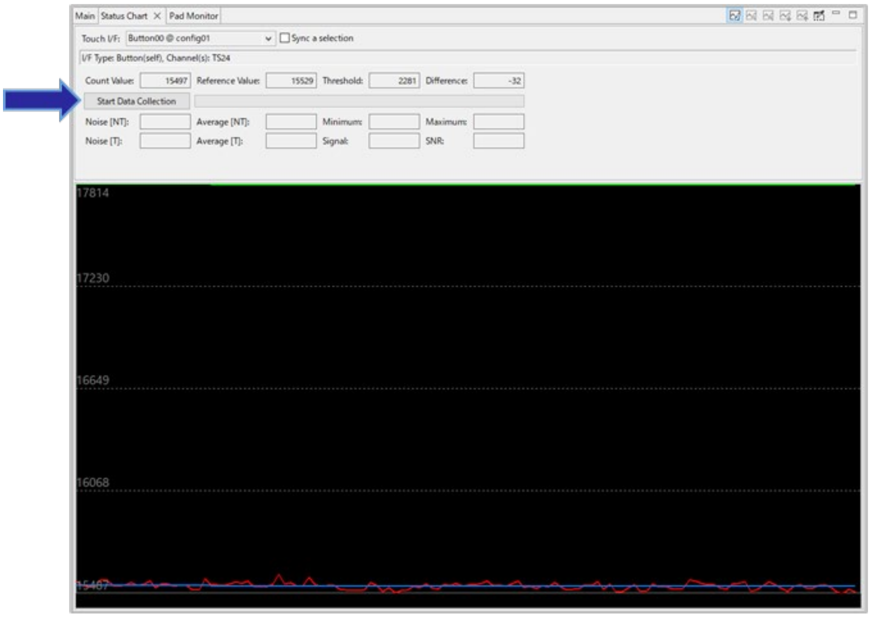


Figure 8-49. Data Collection of Touch-off State

   B. Click “Stop Data Collection”, when the green bar goes all the way to the right.

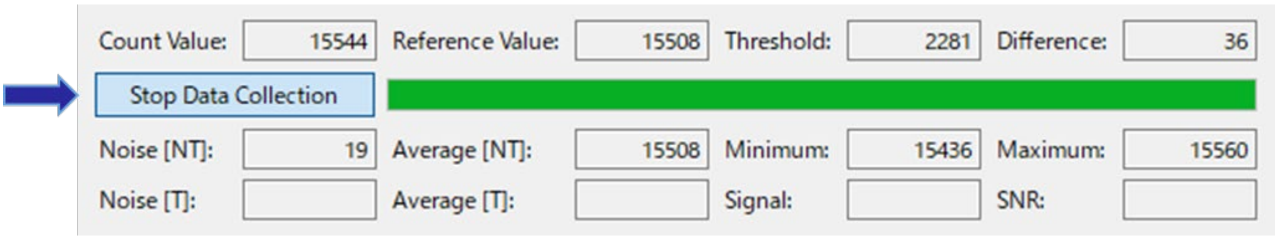


Figure 8-50. Stop Data Collection

   C. Next, in the same way, start data collection in the state of touch-on.

   D. After finishing data collection, SNR value appears.

Figure 8-51. SNR value

7. Represent a graph of the touch counts for multiple touch sensors.
   Select the touch sensors in “Multi Status Chart” panel at the lower-left of QE window.

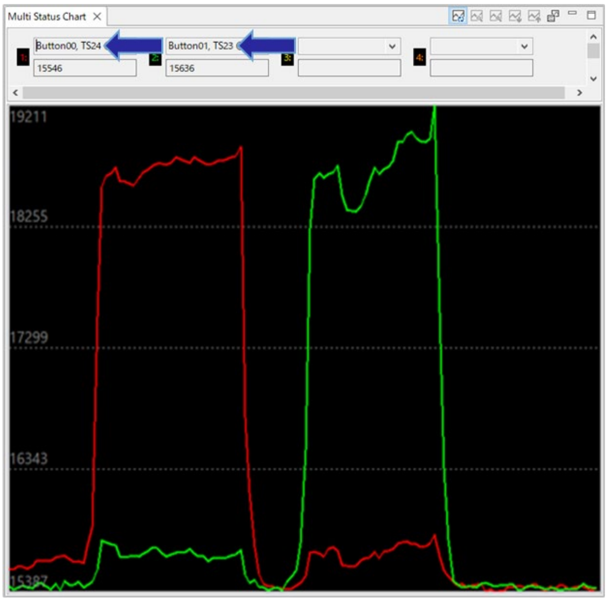


Figure 8-52. Multi Status Chart

8. As necessary, adjust parameters manually.
   Adjust parameters in “Parameters” panel at the right side of QE.

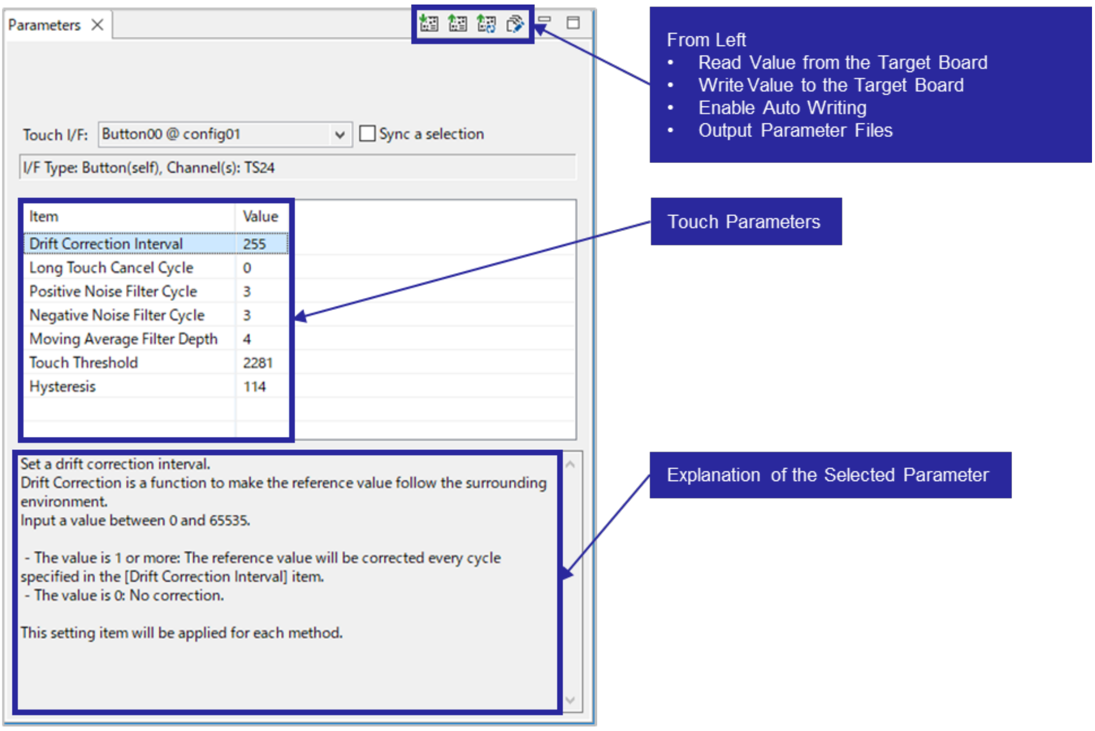


Figure 8-53. Adjustment of Parameters

9. Click “Enable Monitoring” in the state of “Monitoring: Enabled” to stop monitoring.

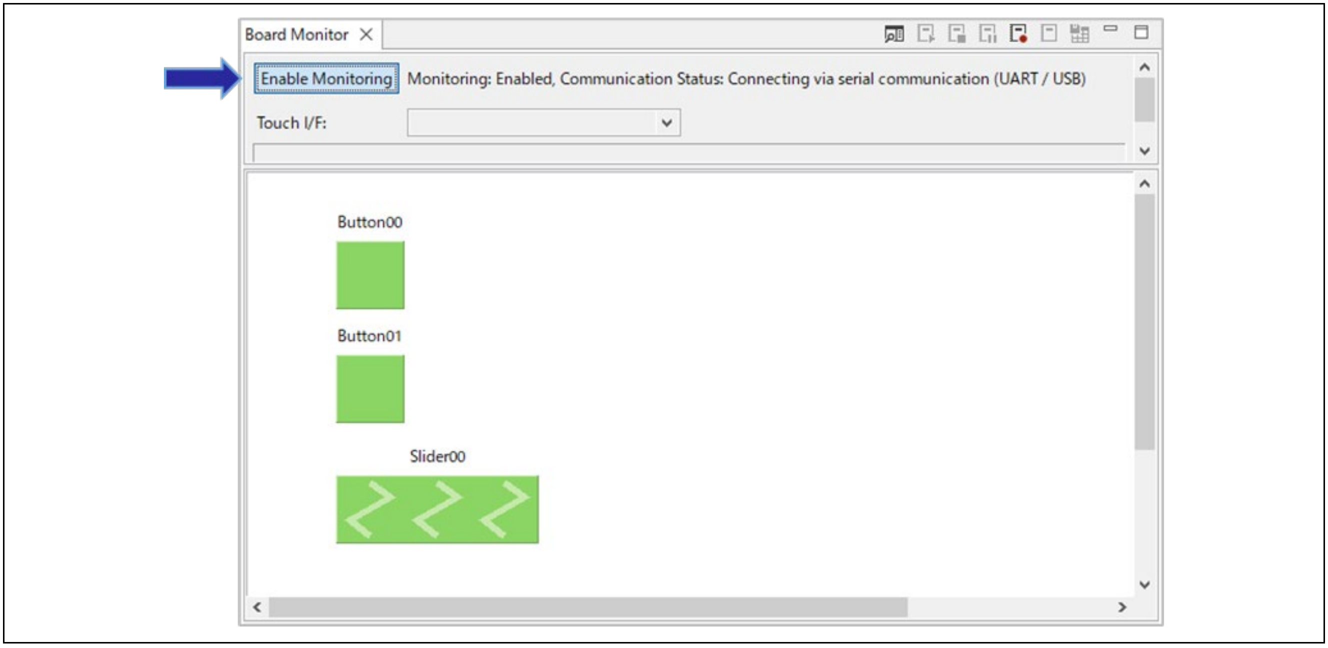


Figure 8-54. Stop Monitoring

10. Click “Disconnect” to disconnect the connection of UART.

Figure 8-55. Disconnect UART

8.6 Sample Code

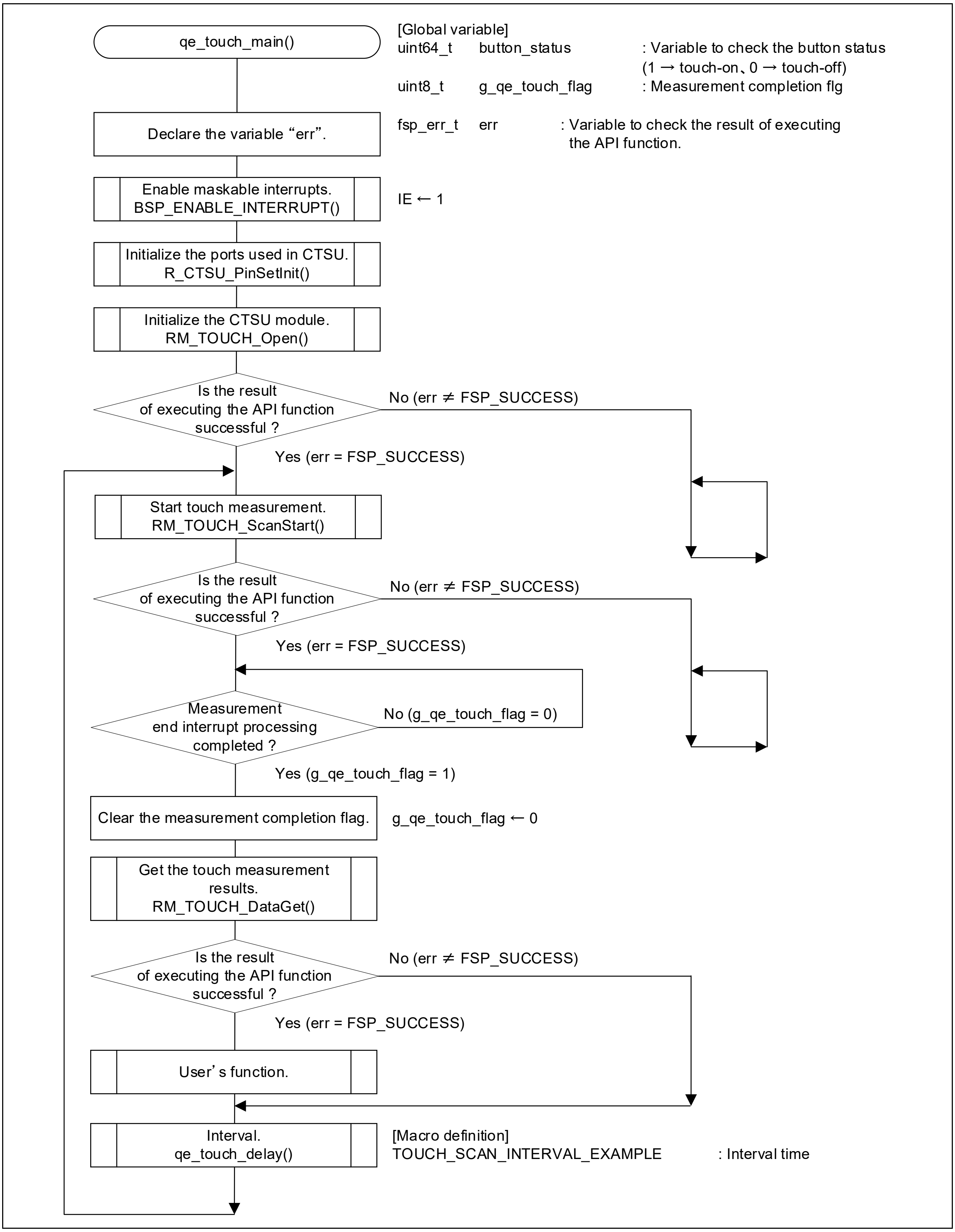
The sample code (qe_touch_sample.c) outputted by QE for Capacitive Touch is as follows.

In this sample code, a touch measurement cycle is created by a software timer.

```
/***********************************************************************
*
* FILE : qe_sample_main.c
* DATE : 2022-02-14
* DESCRIPTION : Main Program for RL78
*
* NOTE:THIS IS A TYPICAL EXAMPLE.
*
***********************************************************************/
#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, slider_position,
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();
    }
}
```

8.7 Flowcharts



9. Appendix

9.1 Touch Measurement by Hardware Timer

This section explains the program using hardware timer (32-bit interval timer channels in 8-bit counter mode) to create a touch measurement cycle.

In addition, operations can be confirmed by turning on/off the LED on the target board depending on the sensor (button) touch state.

9.1.1 Setup of Smart Configurator

1. Select the “Clocks” tab on the Smart Configurator and set the fSXP clock to be used as the interval timer. Also, uncheck the XT1 oscillator.

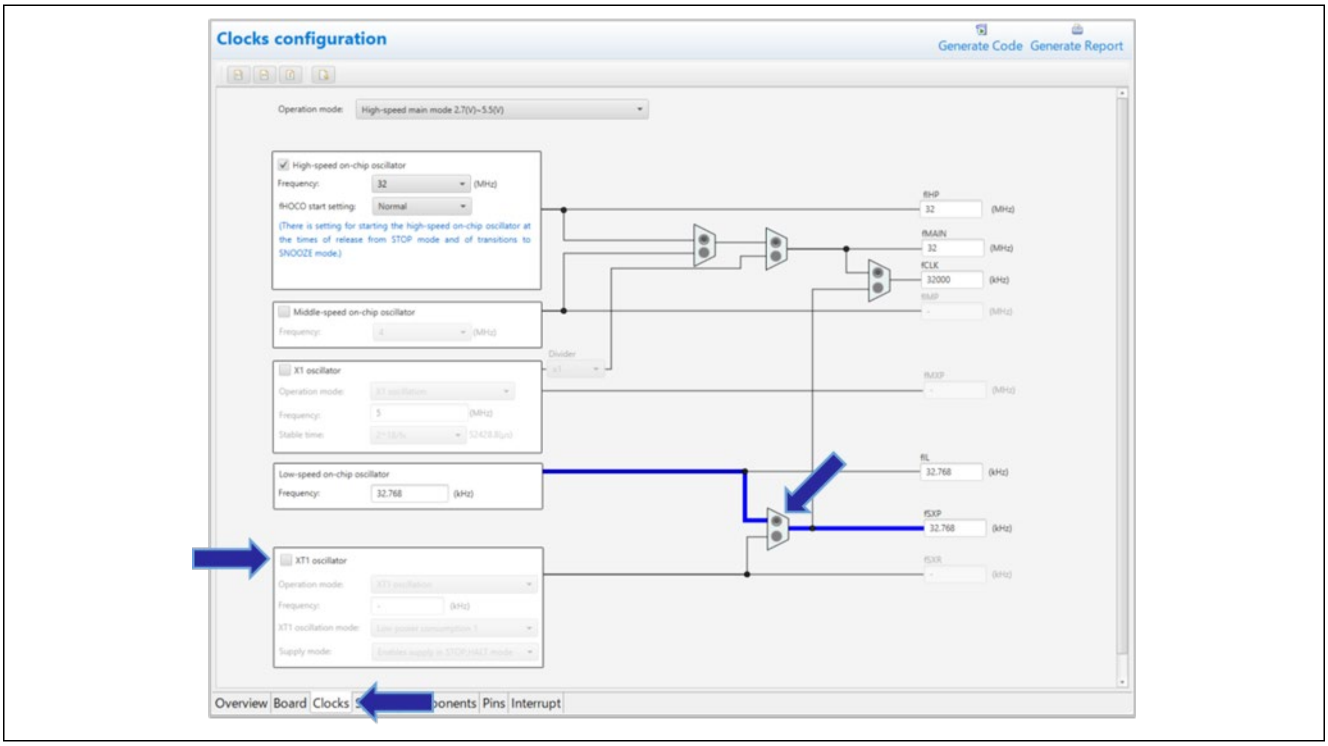


Figure 9-1. Setting Clock

2. Select “Components” tab and click icon to open “New Component” dialog. Select “Interval Timer” module and click “Next”.
   Set configuration of “Interval Timer” to the follow and click “Finish”.

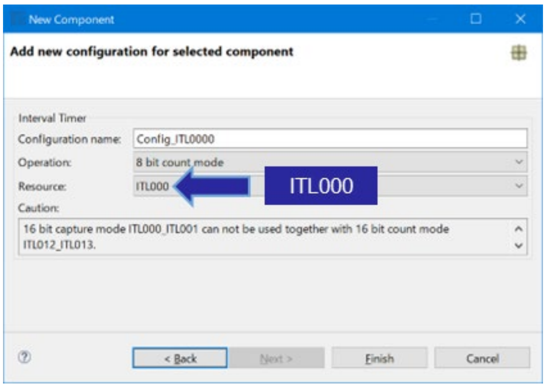


Figure 9-2. Configuration of Interval Timer

3. Select “Interval Timer” module and set such as clocks.

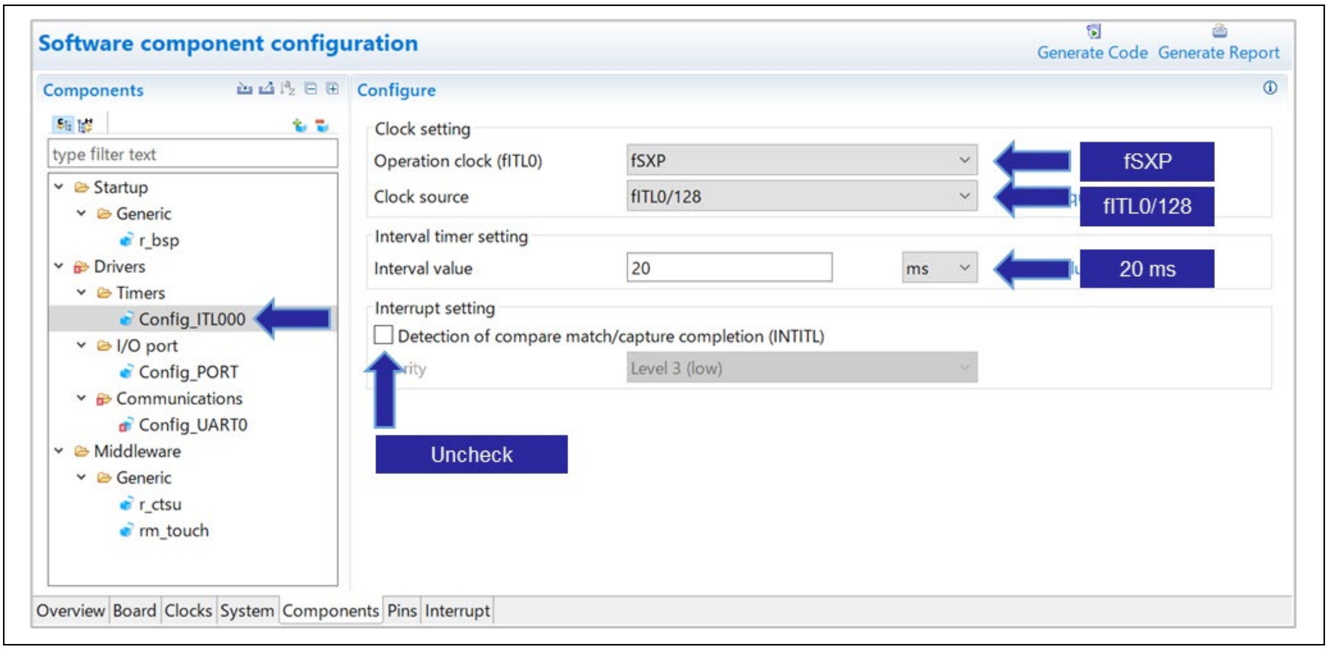


Figure 9-3. Setting “Config_ITL000”

4. Set the Pin for LED. Set “P62” to high-level output in “Ports” module.

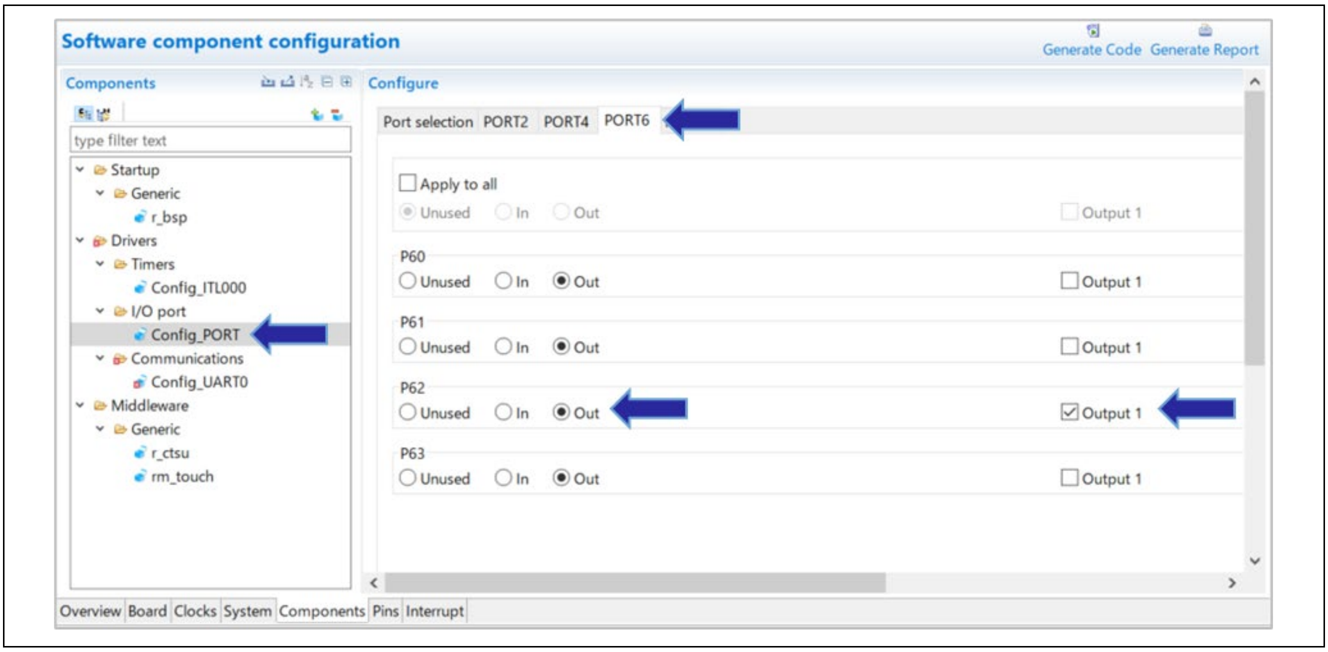


Figure 9-4. Setting “P62”

5. Click  icon on Smart Configurator to perform generating code.

9.1.2 Sample Code

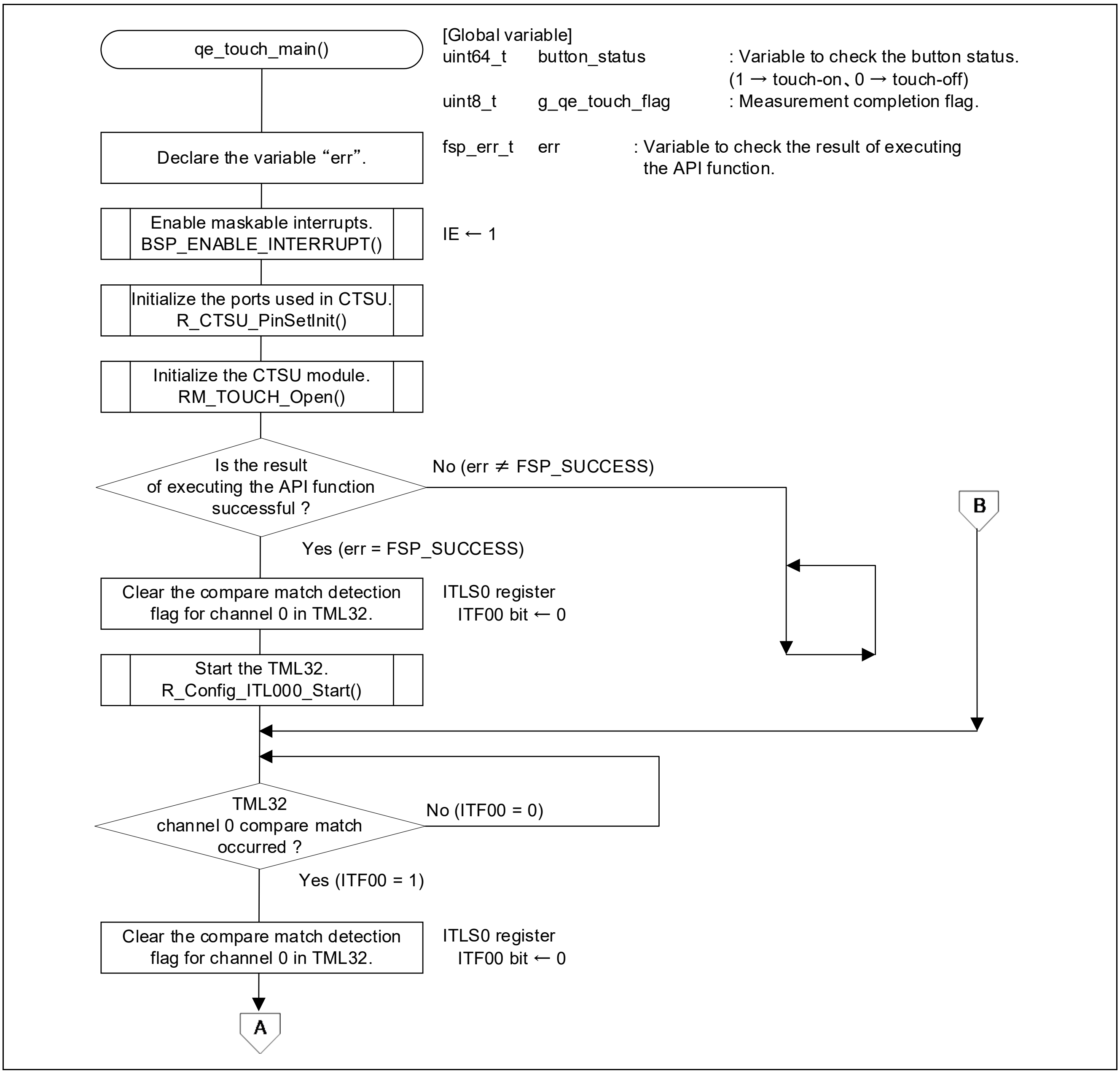
The sample code (qe_touch_sample.c) outputted by QE for Capacitive Touch is as follows.

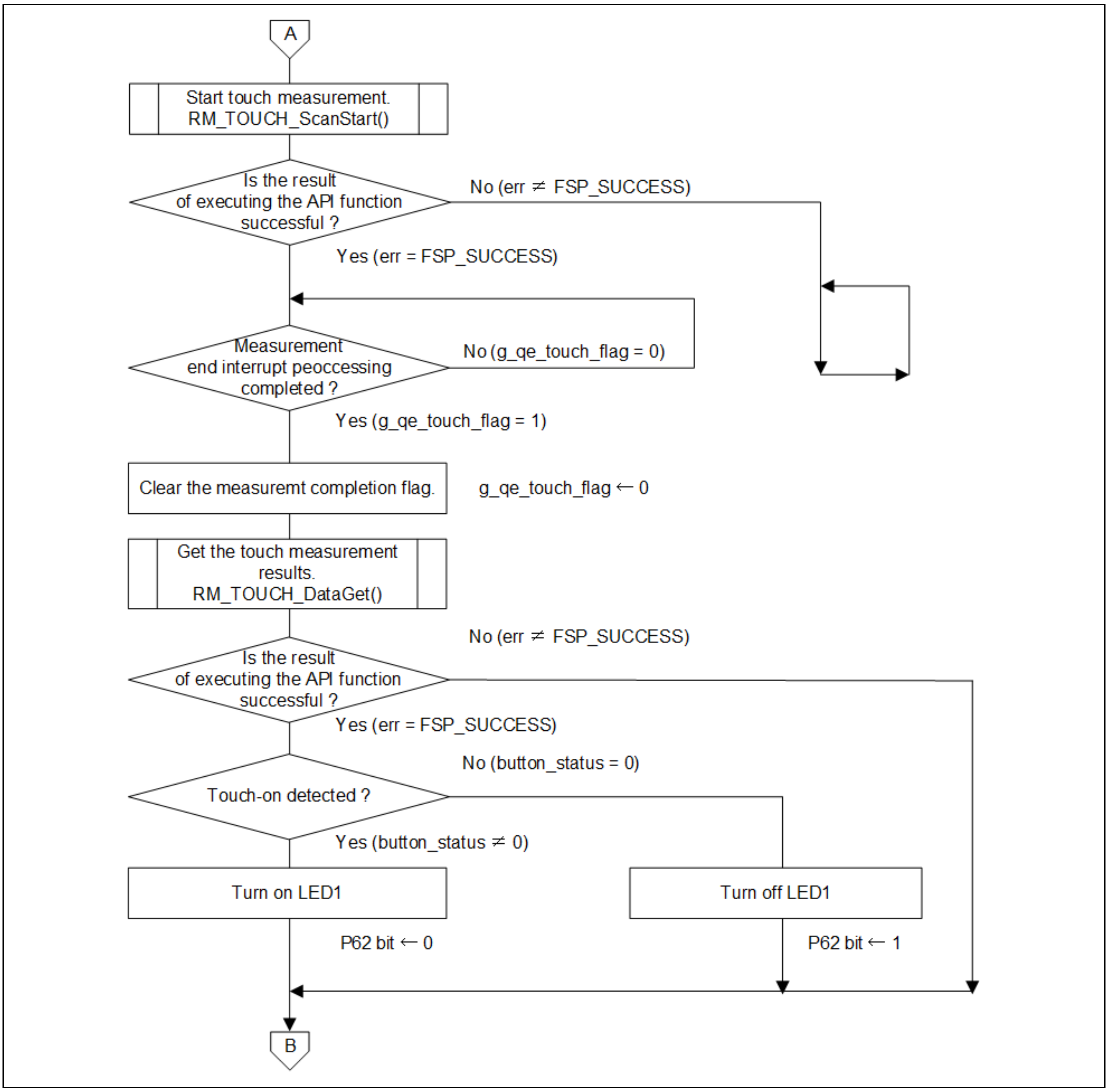
In this sample code, a touch measurement cycle is created using a hardware timer.

```
/***********************************************************************
*
* FILE : qe_sample_main.c
* DATE : 2022-12-15
* DESCRIPTION : CTSU2L Program for RL78
*
* NOTE:THIS IS A TYPICAL EXAMPLE.
*
***********************************************************************/
#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, slider_position,
NULL);
        if (FSP_SUCCESS == err)
        {

            /* TODO: Add your own code here. */
            if (0 != button_status)
            {
                P6_bit.no2 = 0;
            }
            else
            {
                P6_bit.no2 = 1;
            }
        }
    }
}
```

9.1.3 Flowcharts




A
Start touch measurement.
RM_TOUCH_ScanStart()
Is the result of executing the API function successful ?
No (err ≠ FSP_SUCCESS)
Yes (err = FSP_SUCCESS)
Measurement end interrupt peoccessing completed ?
No (g_qe_touch_flag = 0)
Yes (g_qe_touch_flag = 1)
Clear the measuremt completion flag.
g_qe_touch_flag ← 0
Get the touch measurement results.
RM_TOUCH_DataGet()
No (err ≠ FSP_SUCCESS)
Is the result of executing the API function successful ?
Yes (err = FSP_SUCCESS)
No (button_status = 0)
Touch-on detected ?
Yes (button_status ≠ 0)
Turn on LED1
Turn off LED1
P62 bit ← 0
P62 bit ← 1
B

10. Documents for Reference

- RL78/G22 User's Manual: Hardware (R01UH0978)
- RL78 Family User's Manual: Software (R01US0015)
- RL78/G22 Fast Prototyping Board User’s Manual (R20UT5121)
- 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 RL78 Family
  Using the standalone version of QE to Develop Capacitive Touch Applications (R01AN6574)
- Application Note RL78 Debugging Functions Using the Serial Port (R20AN0632)
- Application Note RL78 Family
  Using QE and SIS to Develop Capacitive Touch Applications (R01AN5512)
- Application Note RL78 Family Capacitive Touch Sensing Unit (CTSU2L) Operation Explanation (R01AN5744)
- Application Note RL78 Family CTSU Module Software Integration System (R11AN0484)
- Application Note RL78 Family TOUCH Module Software Integration System (R11AN0485)
- Application Note Capacitive Sensor Microcontrollers CTSU Capacitive Touch Electrode Design Guide (R30AN0389)
- Application Note RL78 Family RL78/G23 Capacitive Touch Low Power Guide (SNOOZE function) (R01AN5886)
- RL78/G23 Capacitive Touch Low Power Guide (SMS function) (R01AN6670)
  (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.)

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	Mar.20.23	-	First edition

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)

   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on

   The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state

   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins

   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.
5. Clock signals

   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.
6. Voltage application waveform at input pin

   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).
7. Prohibition of access to reserved addresses

   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products

   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Corporate Headquarters

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

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit: www.renesas.com/contact/.