RL78 Family

Using the standalone version of QE to Develop Capacitive Touch Applications

Introduction

This application note explains the steps to create an application example that uses capacitive touch sensing using Renesas RL78 Microcontrollers.

This application note is capacitive touch application development guide using standalone version QE for Capacitive Touch.

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

If you don't use standalone version QE but e^2 studio and the plugin 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)

If you use any target device other than RL78 family, please see the following application notes.

- RX Family Using QE and FIT to Develop Capacitive Touch Applications (R01AN4516)
- RA Family Using QE and FSP to Develop Capacitive Touch Applications (R01AN4934)

Target Device

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 target device in figures of this application note is RL78/G23, and the program is running on CS+ and E2 emulator Lite. Using standalone version QE can develop capacitive touch application regardless of device or IDE.

This application note is useful for other environments using a device of “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	8.07.00 or later
Toolchains	CC-RL	1.11.00 or later
QE	Standalone Version QE for Capacitive Touch	3.1.0 or later
Smart Configurator	RL78 Smart Configurator	1.3.0 or later

Table 2-2. Operating Environment (Hardware)

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

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 install 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 PC.

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

In this application example, Power supply for the target board uses USB. Then see the circuit of the target board, and set switches or jumpers as necessary.

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

- JP1 : 1-2 closed
- JP2 : closed
- JP3 : 1-2 closed
- JP4 : 1-2 closed

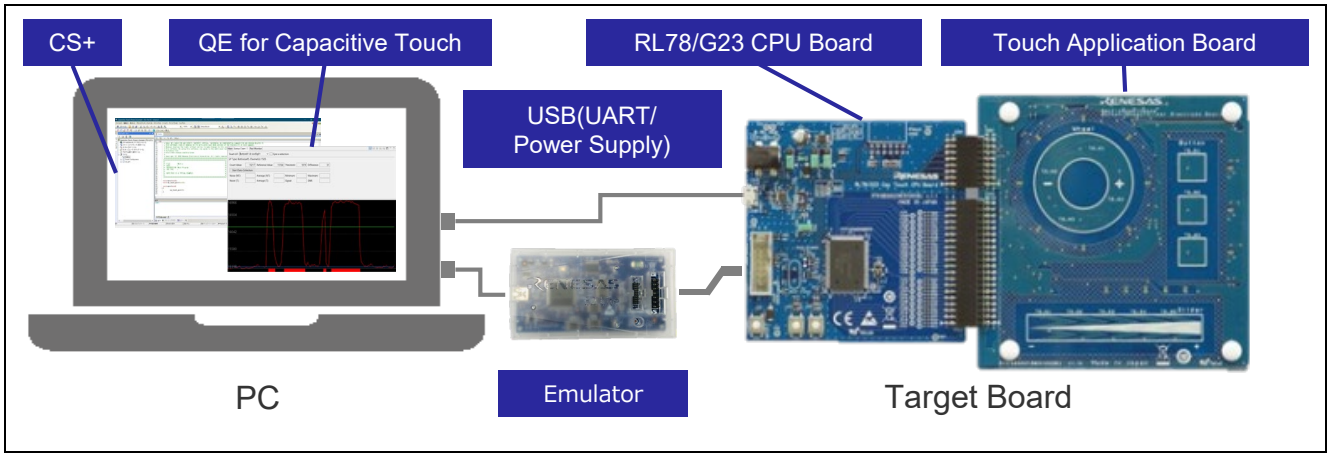


Figure 3-1. Connection of Target Board with PC

4. Workflow for Developing the Application

This chapter explains how to create application.

The steps for developing the application is following the workflow of QE for Capacitive Touch.

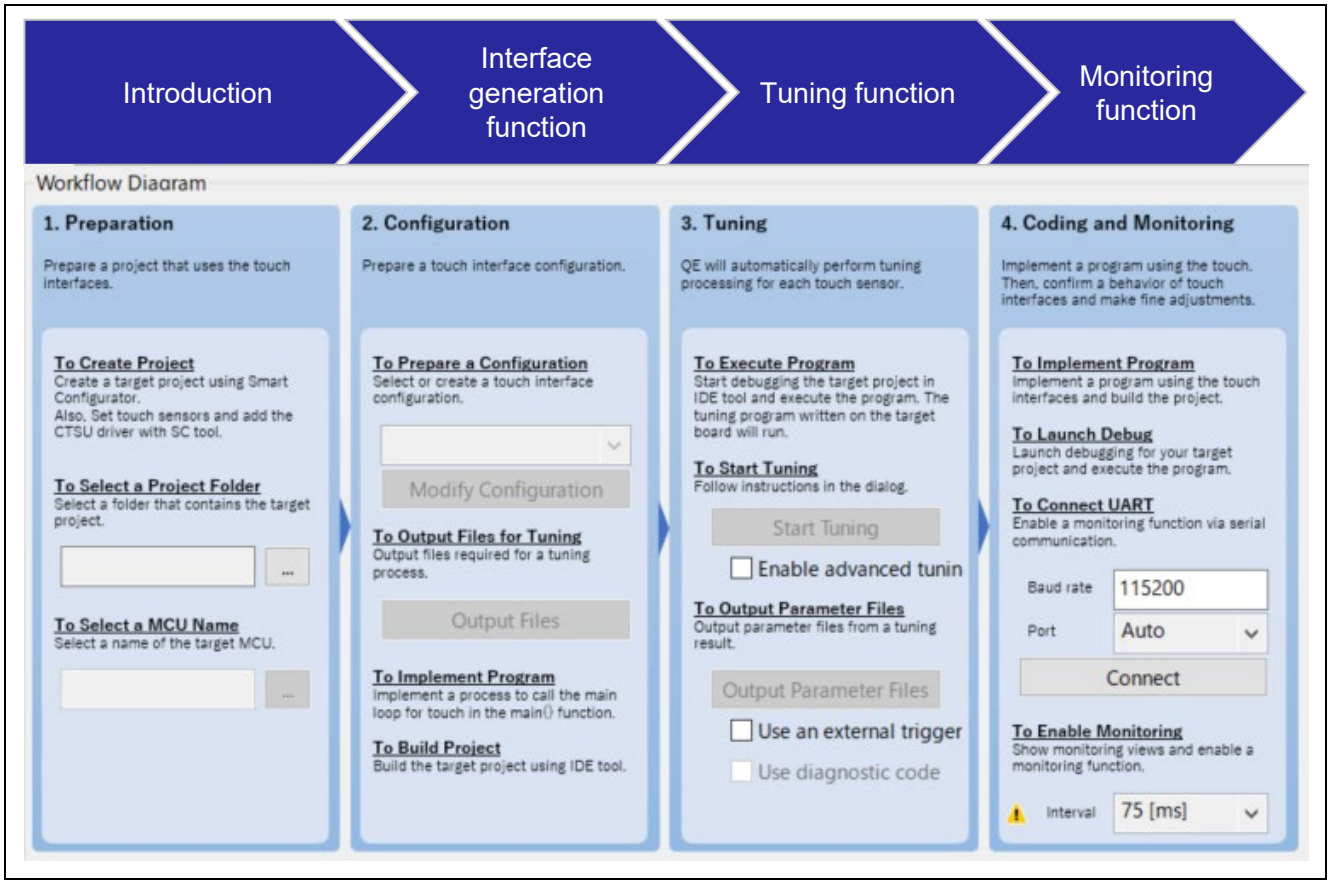


Figure 4-1. Workflow for Developing the Application

Table 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. 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. Example of Application

5.1 Application Example Overview

This application note takes an example of application which uses two buttons.

After chapter 6, this application note will explain how to create the application and monitor whether the each button is touched.

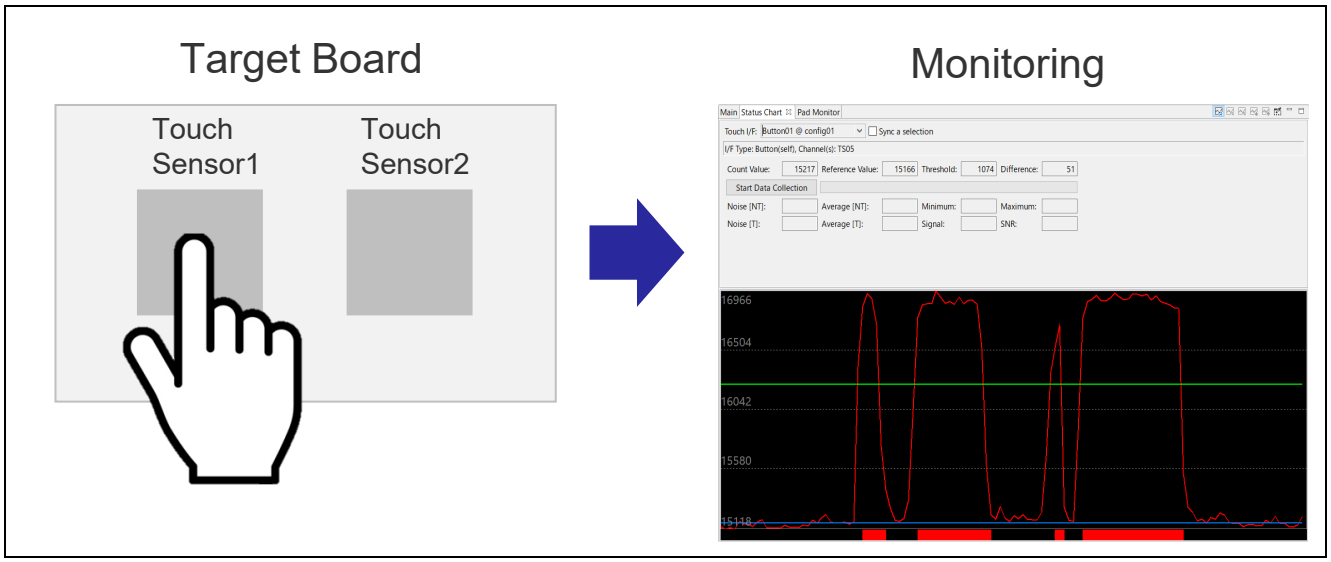


Figure 5-1. Application Example

5.2 List of Used Pins

Table 5-1 shows used pins for this application example.

UART communication and touch sensors for application depend on your using target board.

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

Items	Pins	uses
UART Communication	RxDA1/P33	Tuning Monitoring
	TxDA1/P34	
Touch Sensor 1	TS06/P74	Button (TS_B1)
Touch Sensor 2	TS05/P73	Button (TS_B2)

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

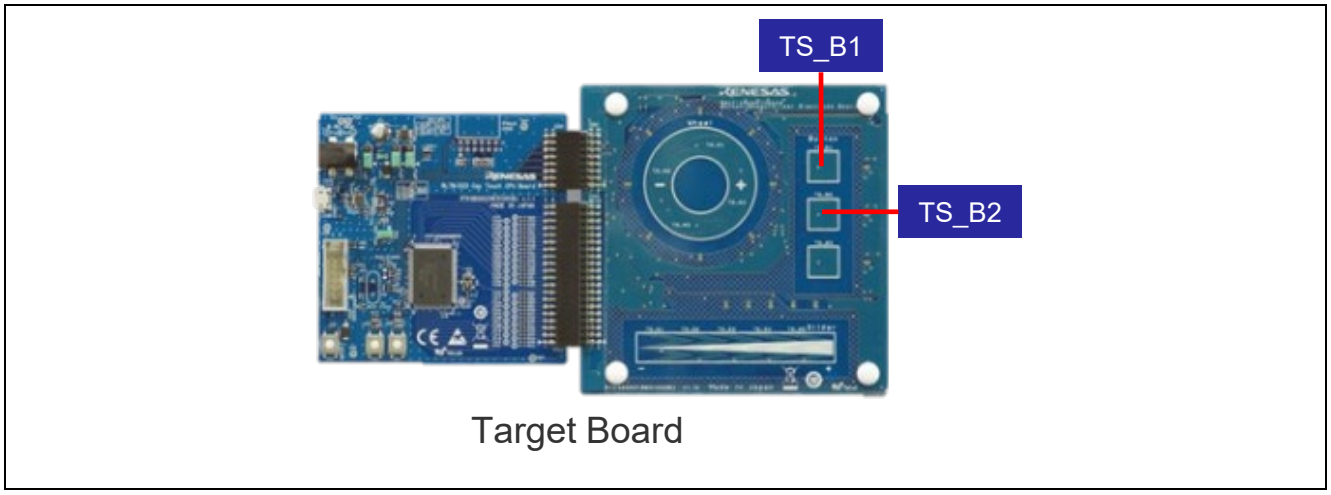


Figure 5-2. Position of Touch Sensors

6. Project Creation

Launch CS+ and create new project.

In “Create Project” dialog, select the following.

- Microcontroller : (Family name)
- Using microcontroller : (Product name)
- Kind of project : Application(CC-RL)
- Project name : (Any project name)
- Place : (Any place)

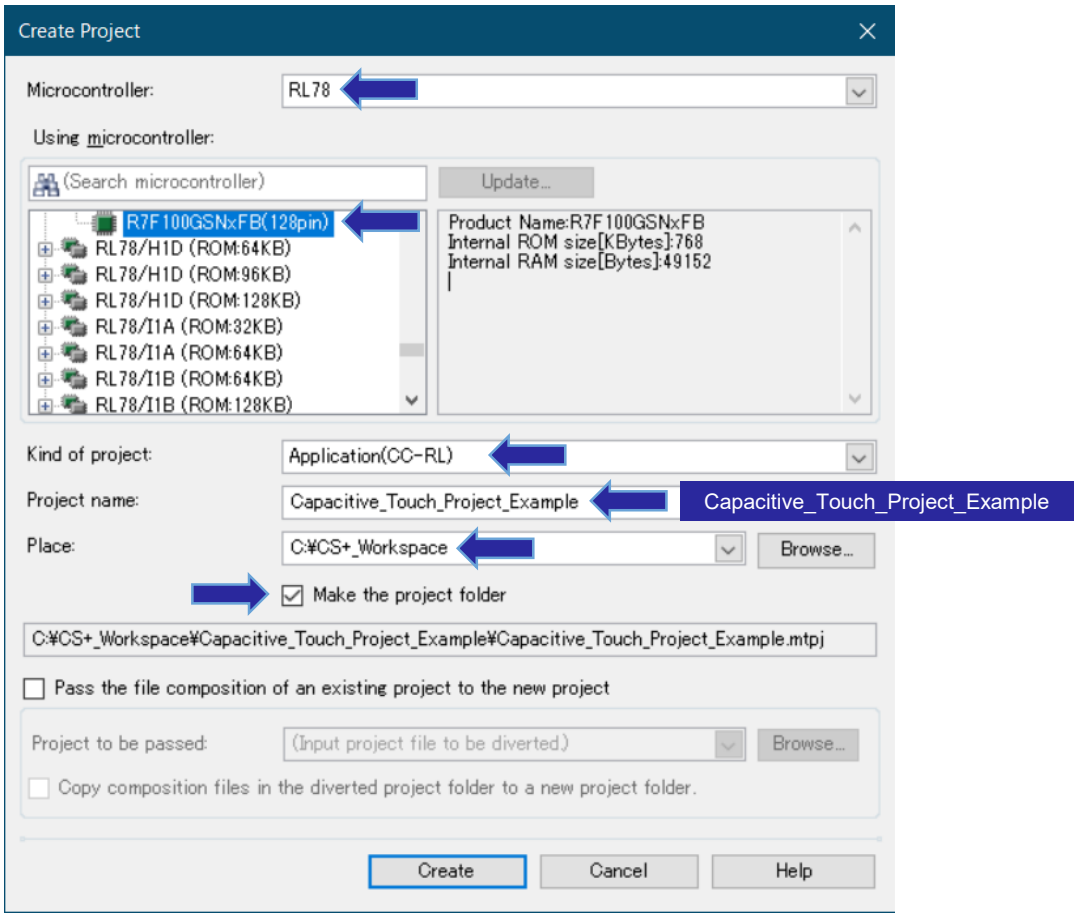


Figure 6-1. Creating New Project

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.

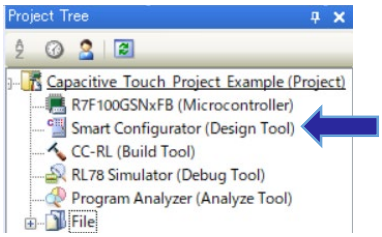


Figure 7-1. Launching Smart Configurator

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.

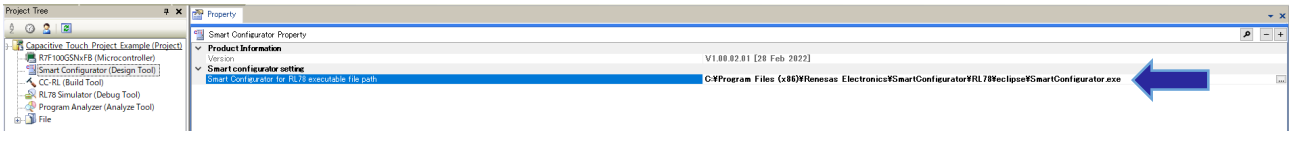


Figure 7-2. File Path of Smart Configurator

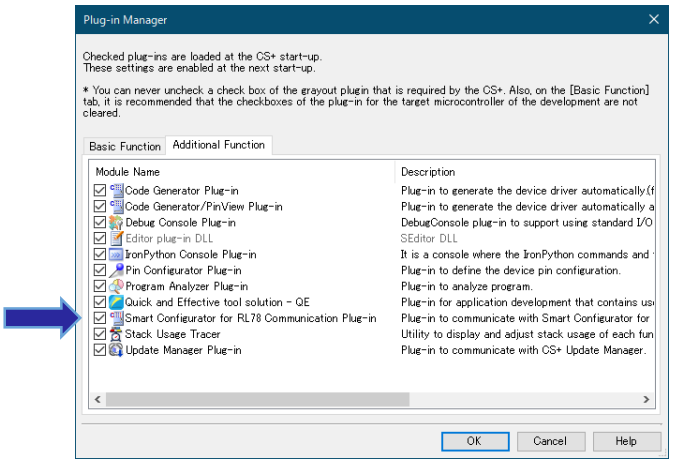


Figure 7-3. Plug-in Manager

7.2 Setup of Clock and System

This section explains how to set clock and system.

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

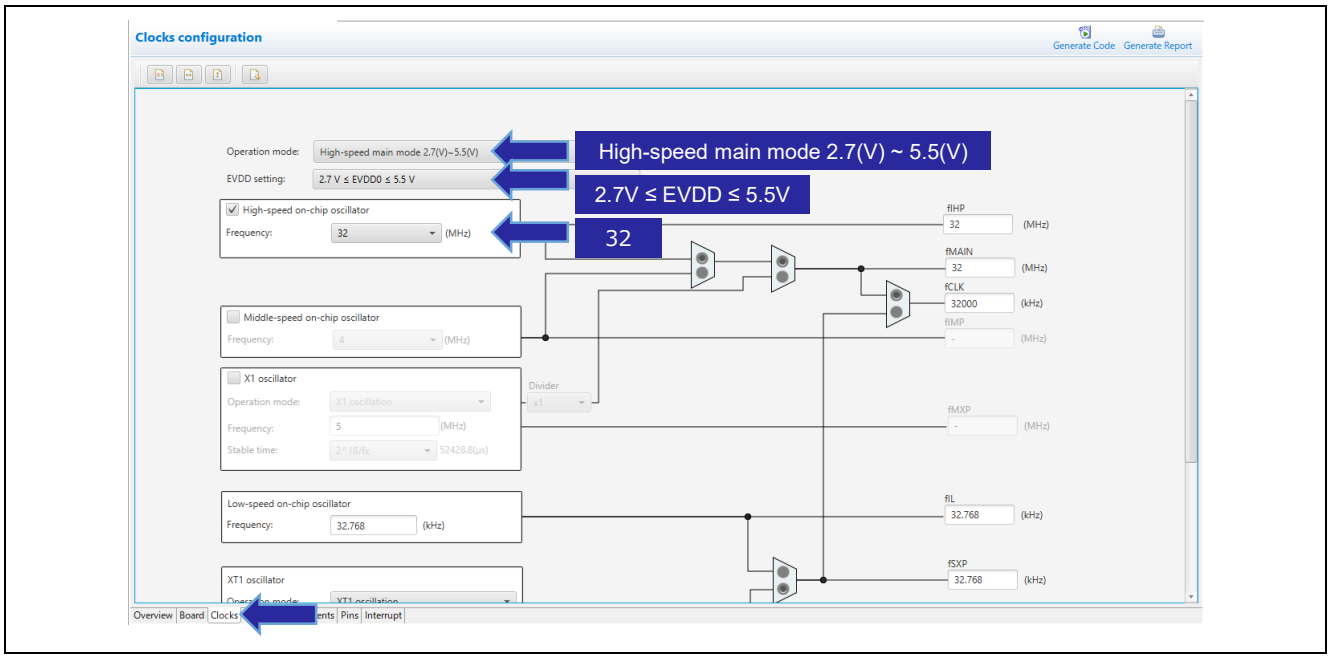


Figure 7-4. Setup of Clocks

2. Select “System” tab and set some items depending on your debug environment.

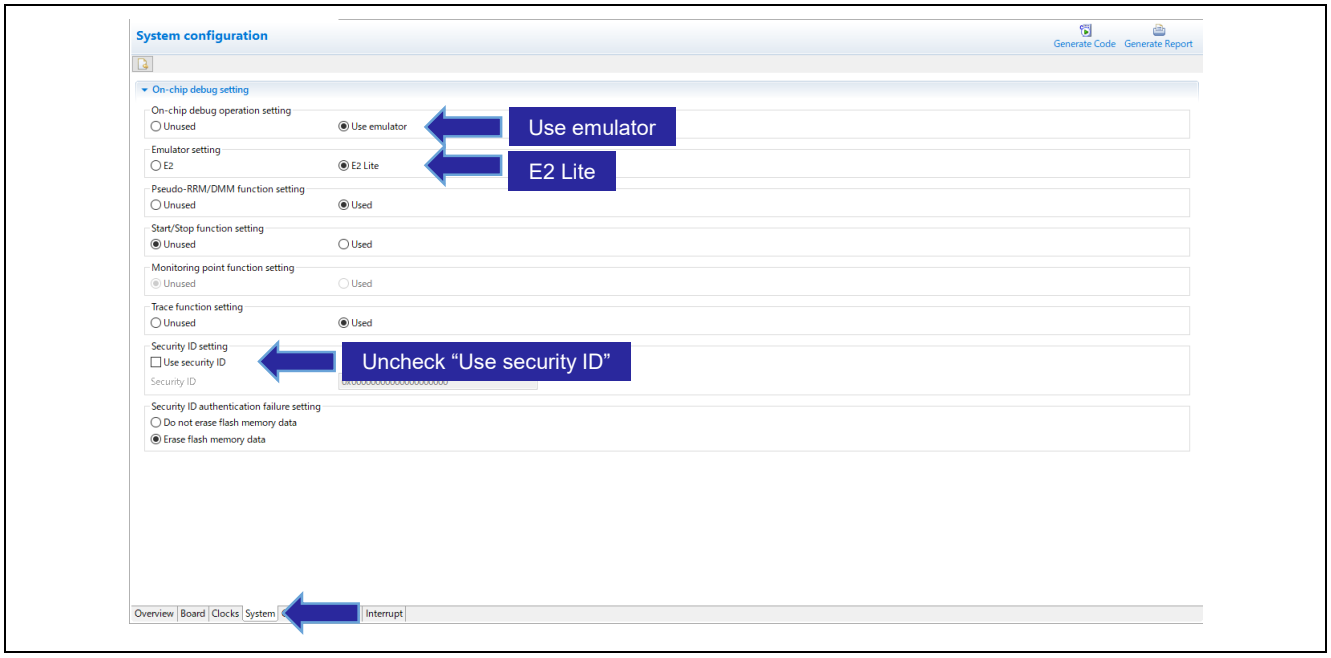


Figure 7-5. Setup of Debug

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  icon.

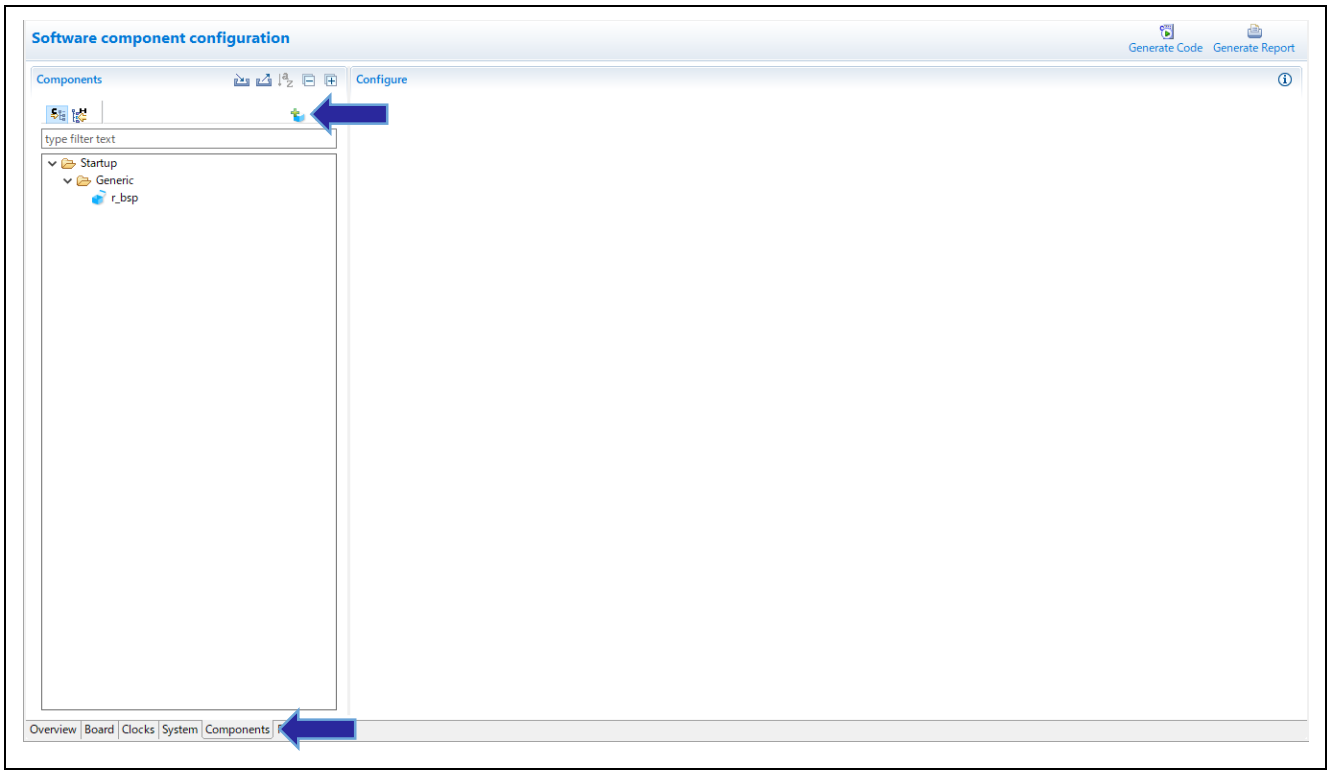


Figure 7-6. Software Component Configuration

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

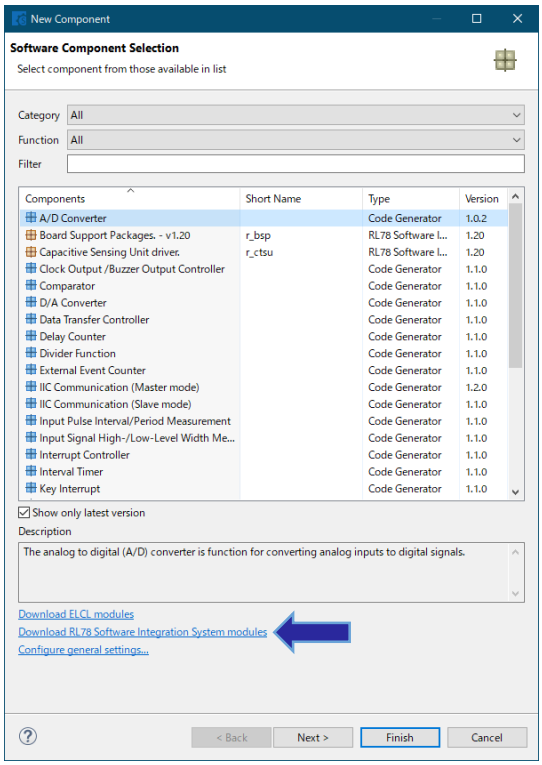


Figure 7-7. Software Component Selection Dialog Box

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

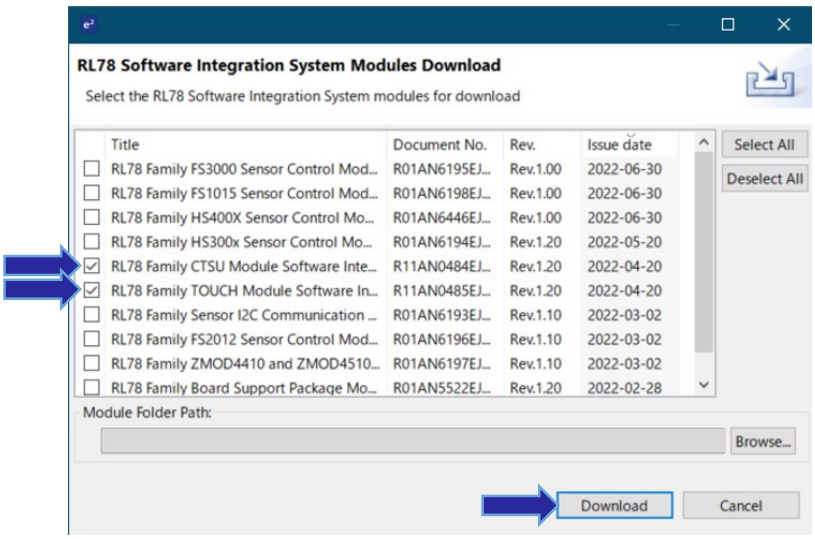


Figure 7-8. Download SIS Modules

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”.

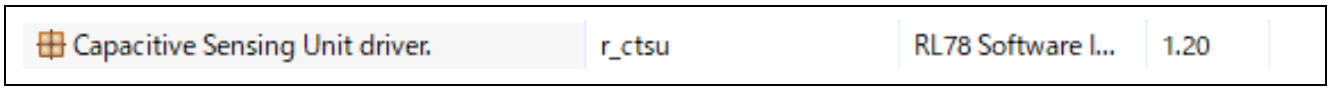


Figure 7-9. “r_ctsu” Module

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

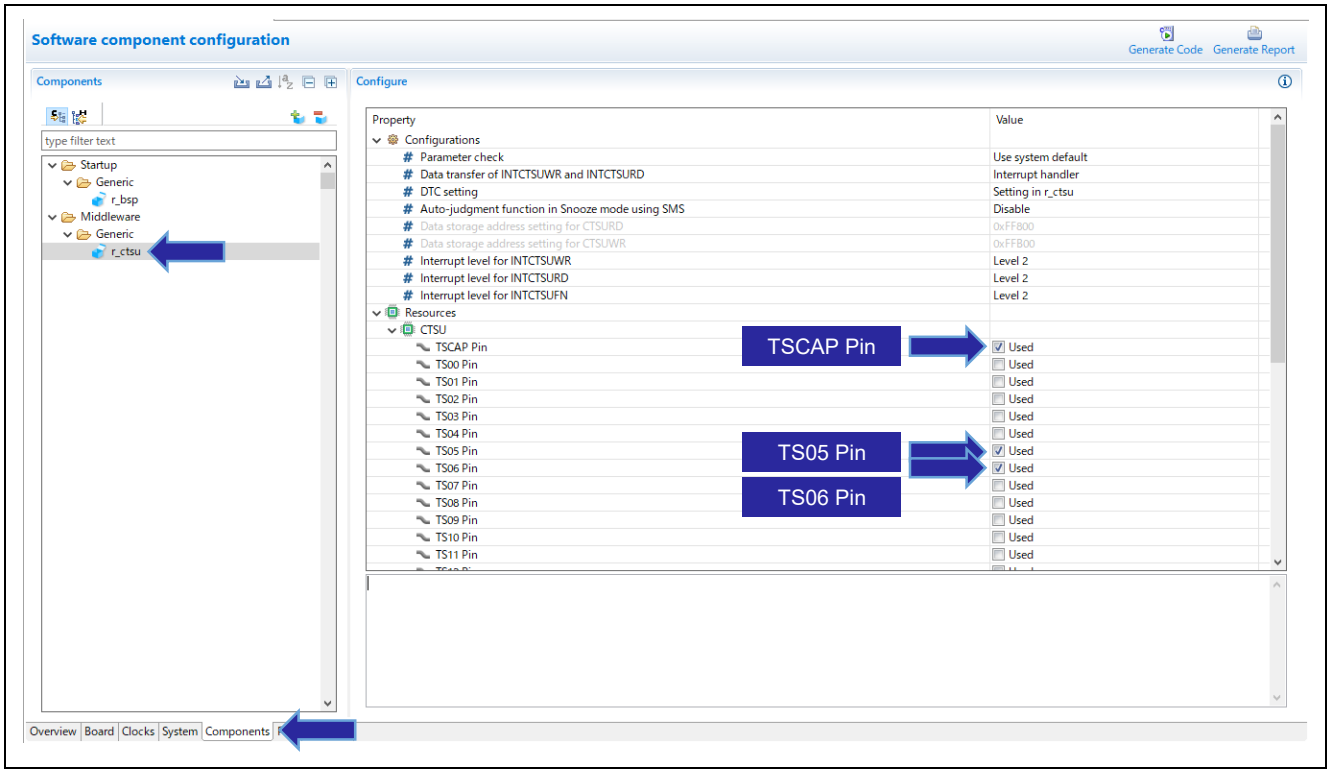


Figure 7-10. Enable Used TS Pins

3. It is recommended to set TS pins unused in the application to low-level output. In the case of CTSU2, when TS pins unused in the application is 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.

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

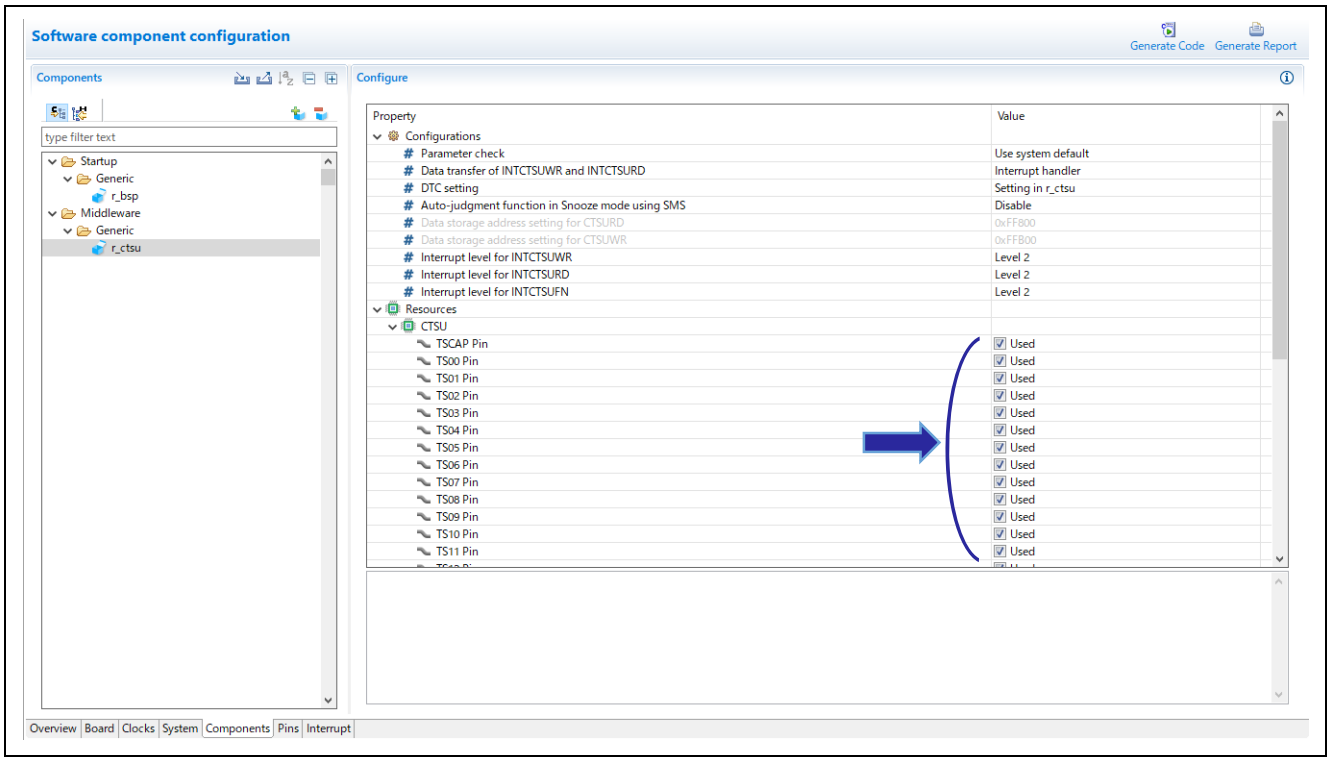


Figure 7-11. Enable TS Pins Unused in the Application

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”.

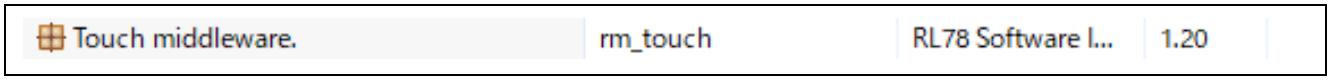


Figure 7-12. “rm_touch” Module

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

UART channel to set depends on your target board.

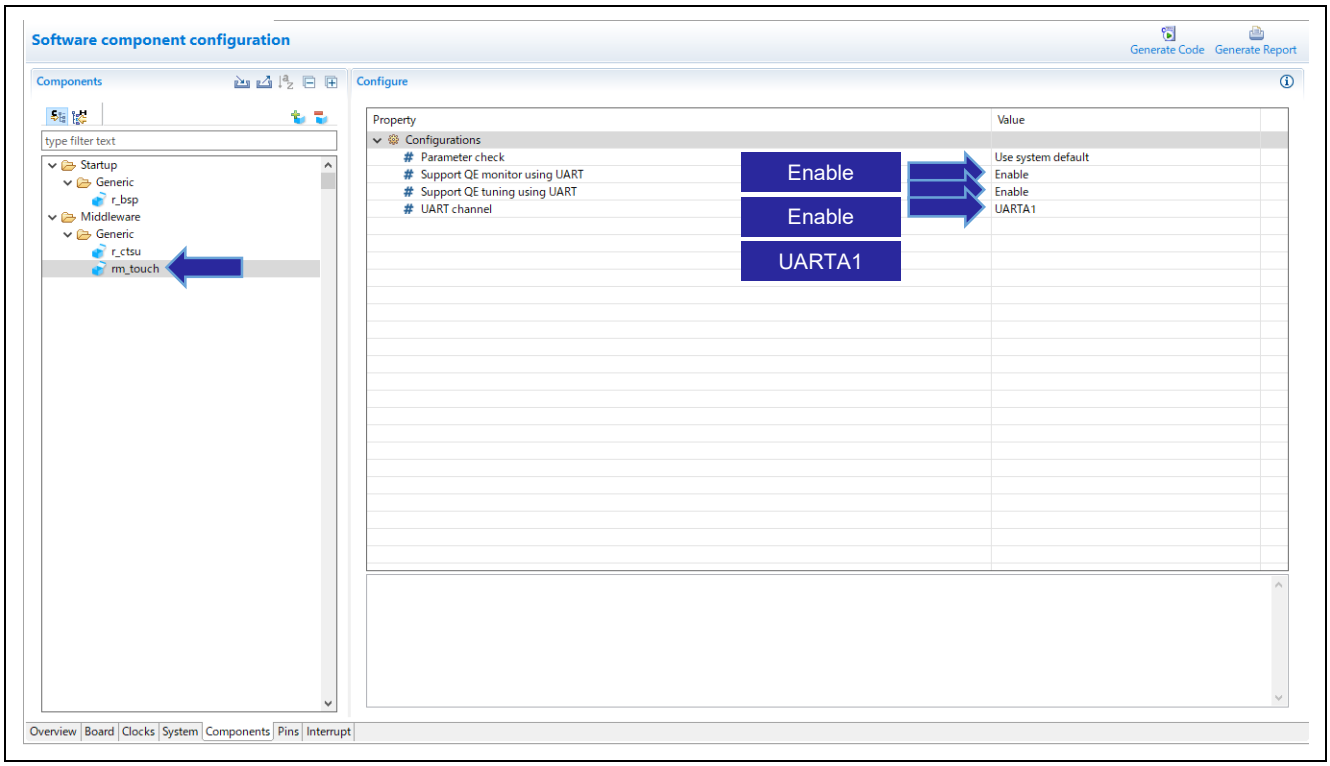


Figure 7-13. Setup of “rm_touch” Module

7.4 Setup of Serial Interface (UART)

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

UART channel and port to 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 : (Your using UART channel)

Figure 7-14. “UART Communication” Module

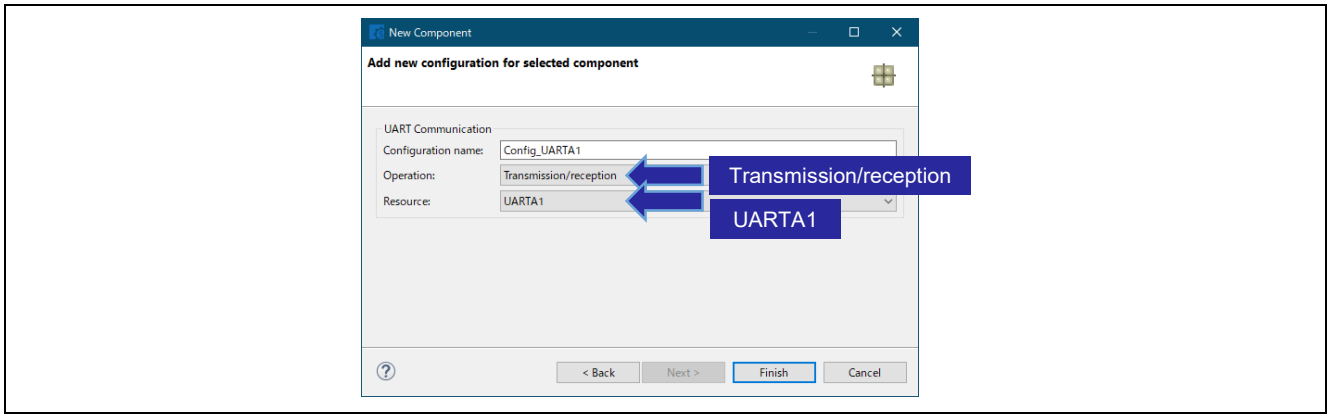


Figure 7-15. Select UART Channel

2. Click “UART Communication” module and set operating clock and transfer rare (band rate).

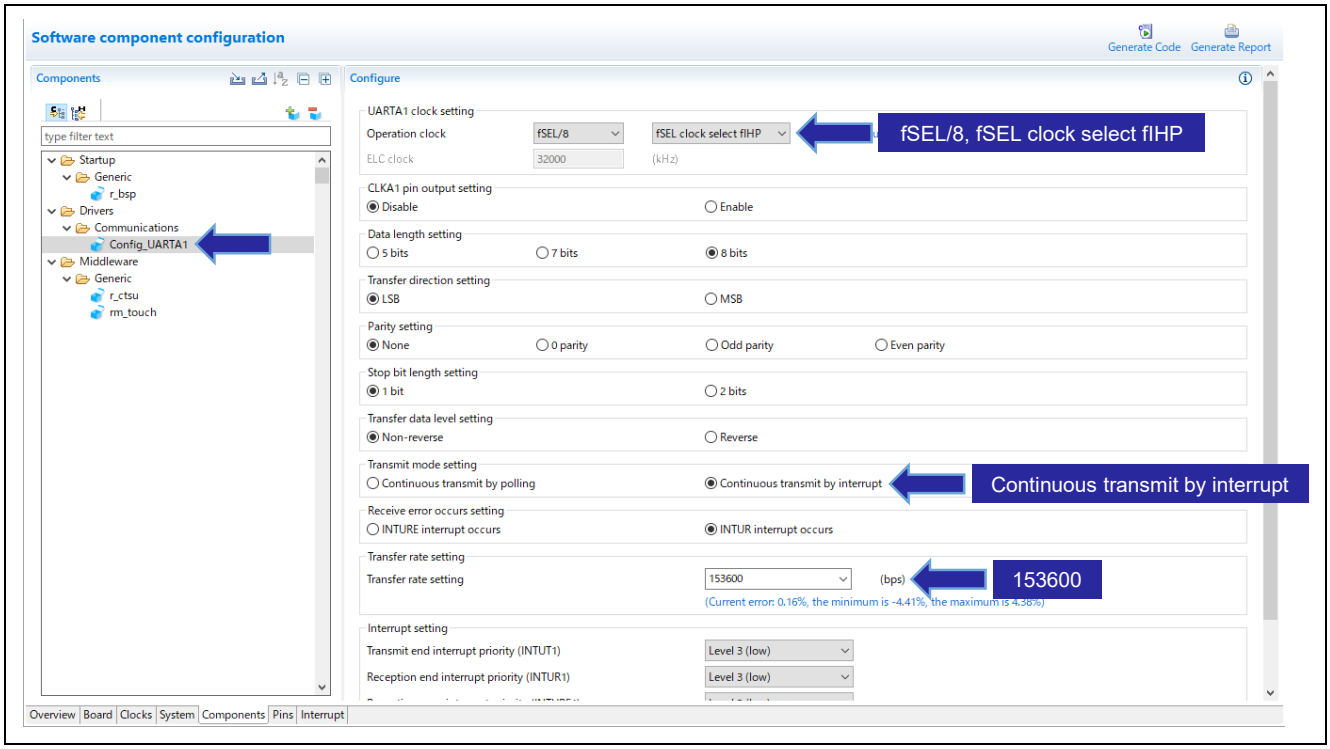


Figure 7-16. Setup of “UART Communication” Module (UARTA1)

3. Select “Pins” tab and assign Pins to UART channel.

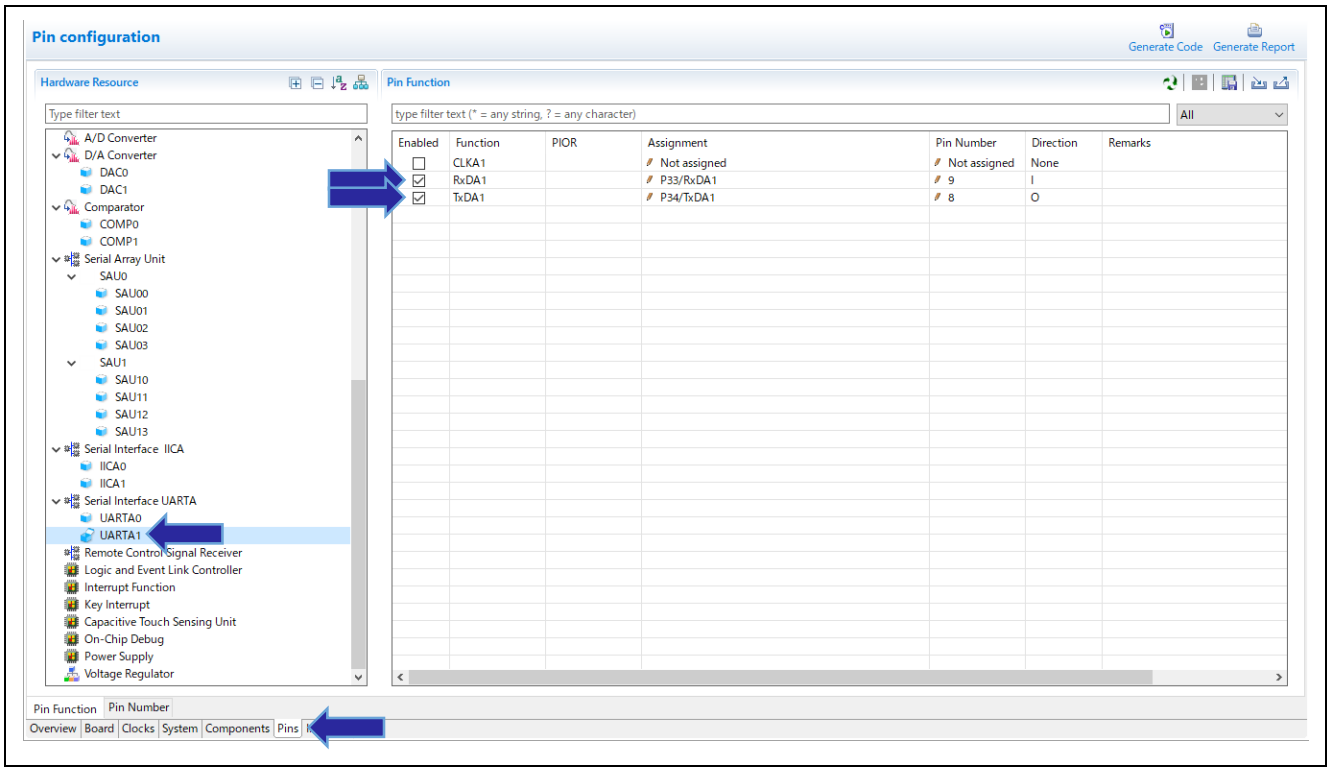


Figure 7-17. Assignment of Pins for UART Channel

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 "PORT07" to low-level.

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

Figure 7-18. "Ports" Module

2. Select "Port" module and check "PORT0".

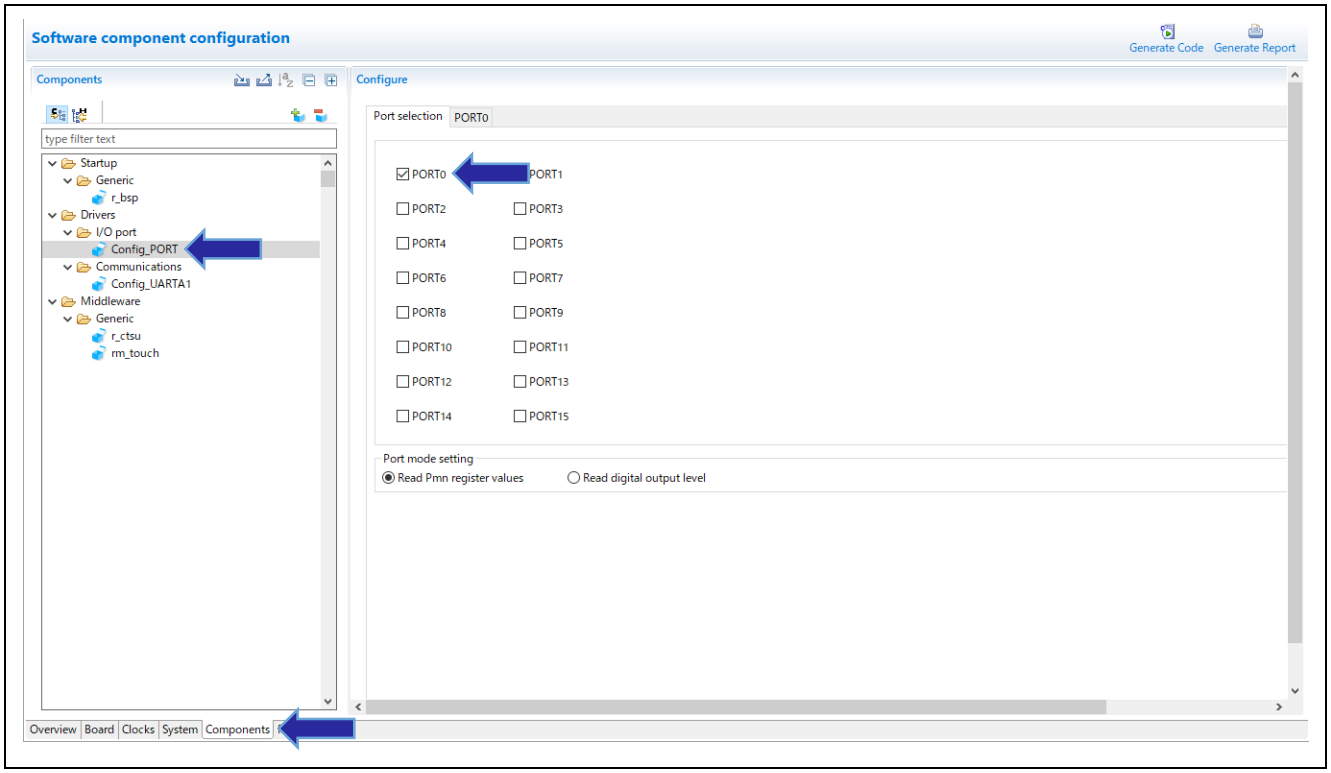


Figure 7-19. Setup of "Ports" Module

3. Click “PORT0” tab and set “P07” to output.

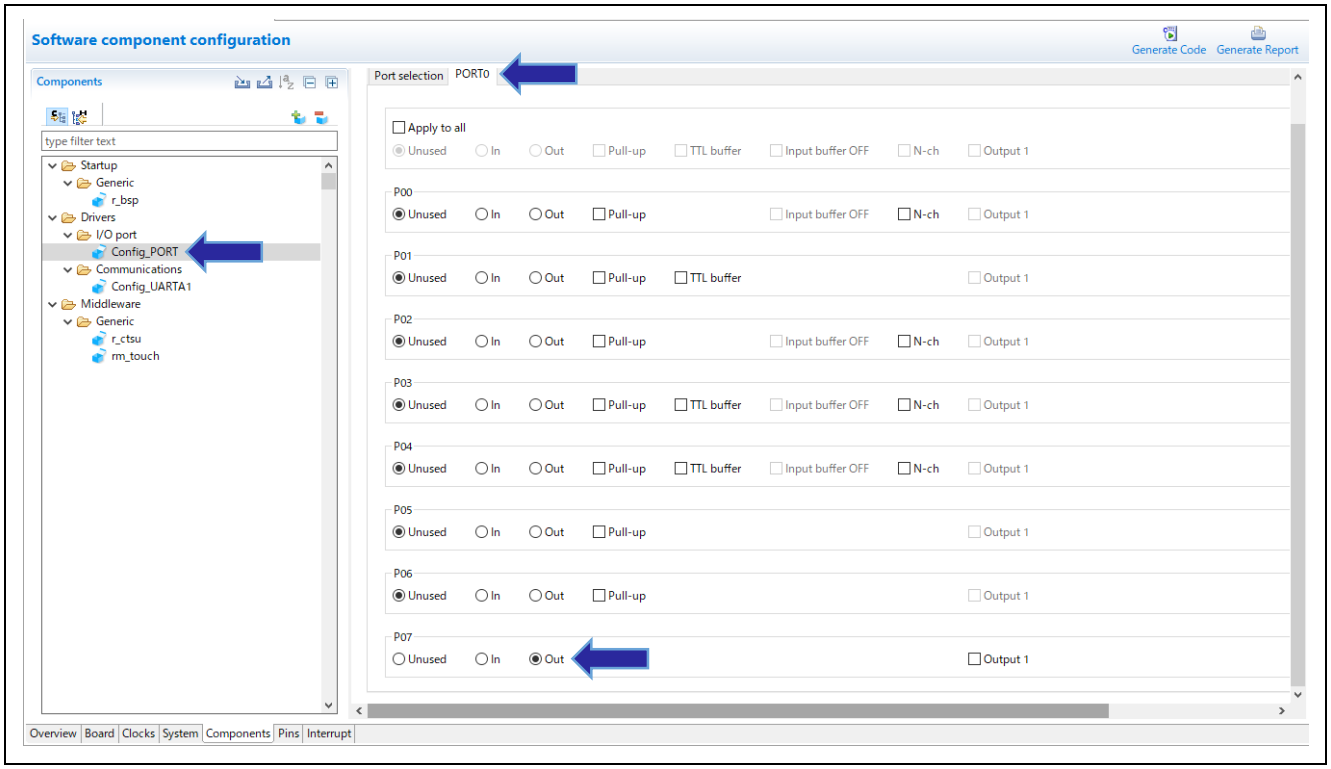


Figure 7-20. Setting “P07” to Output

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”.

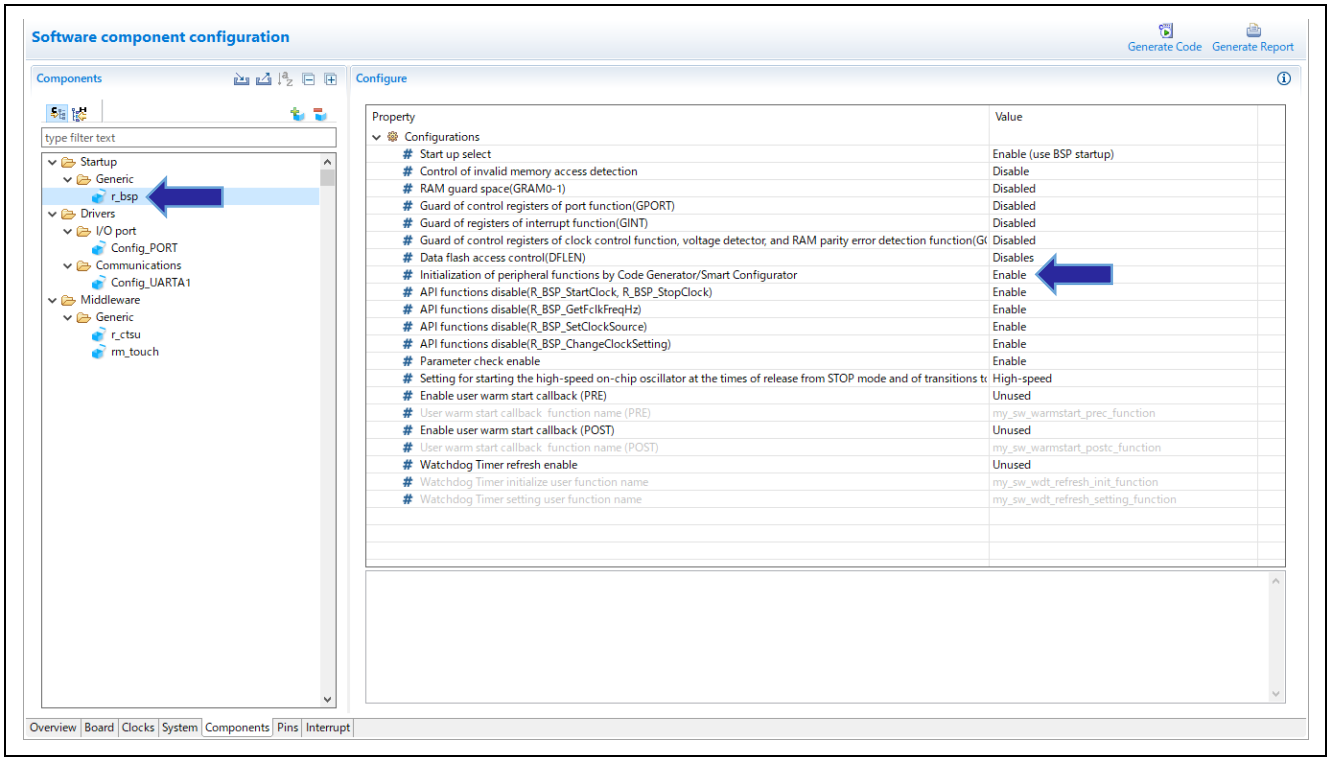


Figure 7-21. Setup of “r_bsp”

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 is displayed. Confirm the changes and click “OK”.

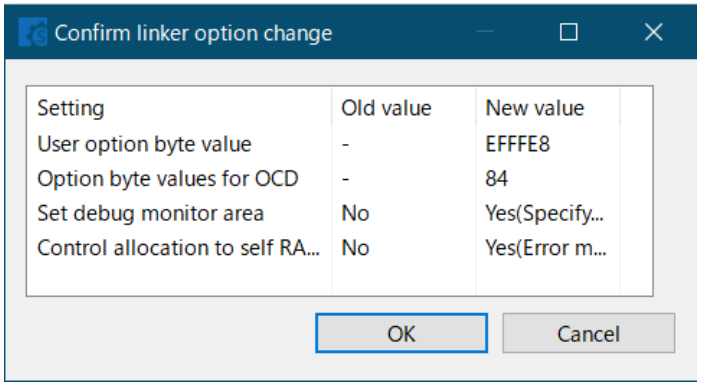


Figure 7-22. Confirm Linker Option Change

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.

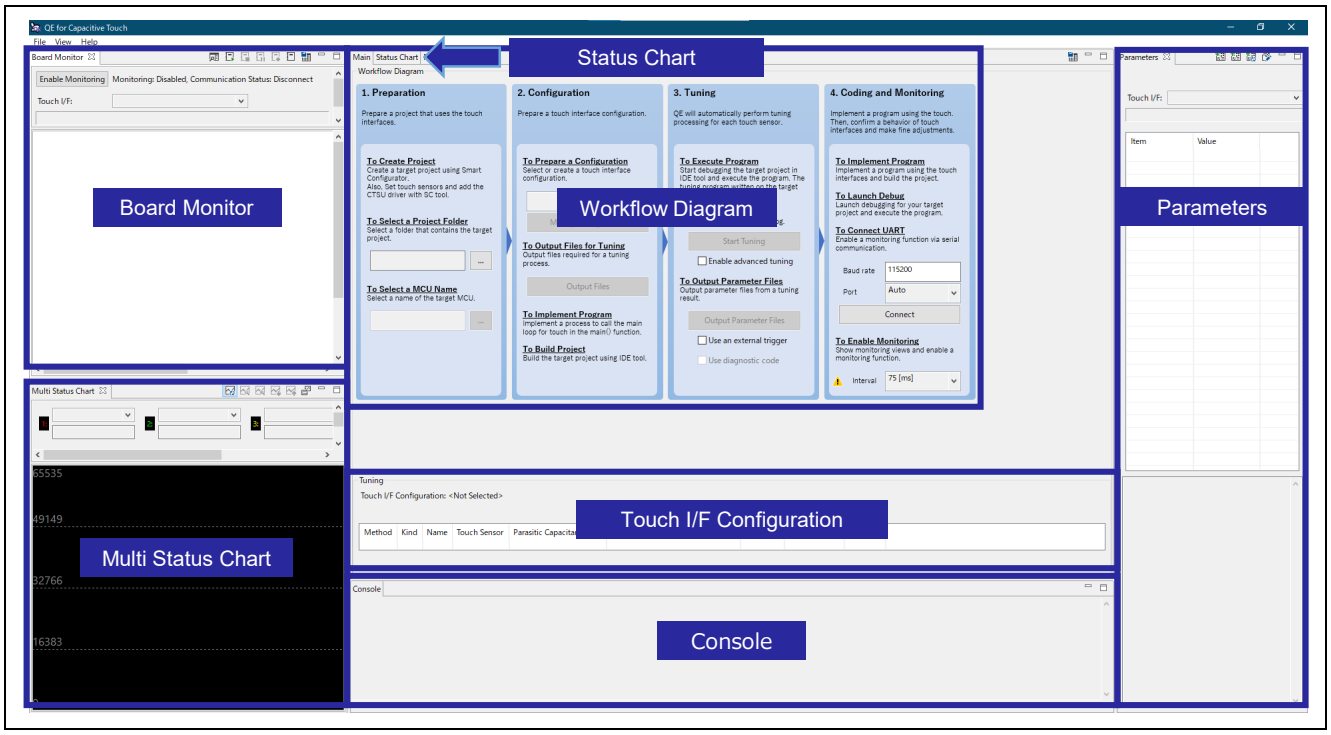


Figure 8-1. QE Window after Launching

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.

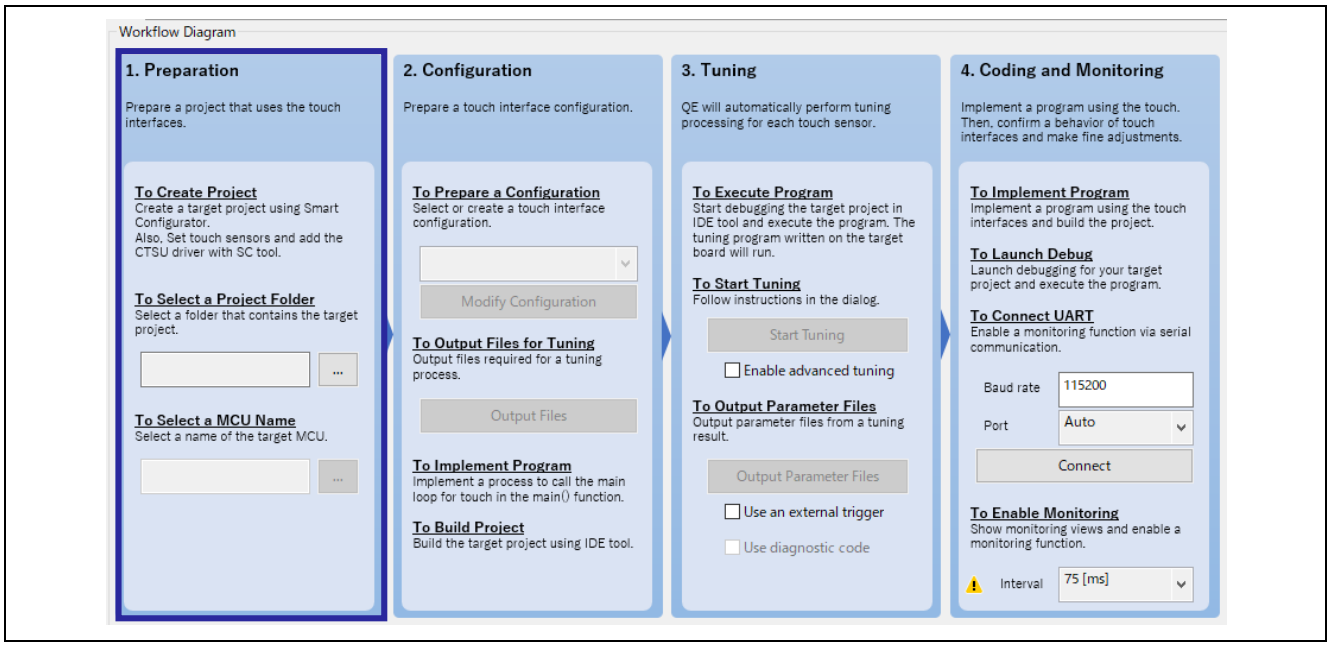


Figure 8-2. Workflow Diagram (Preparation)

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.

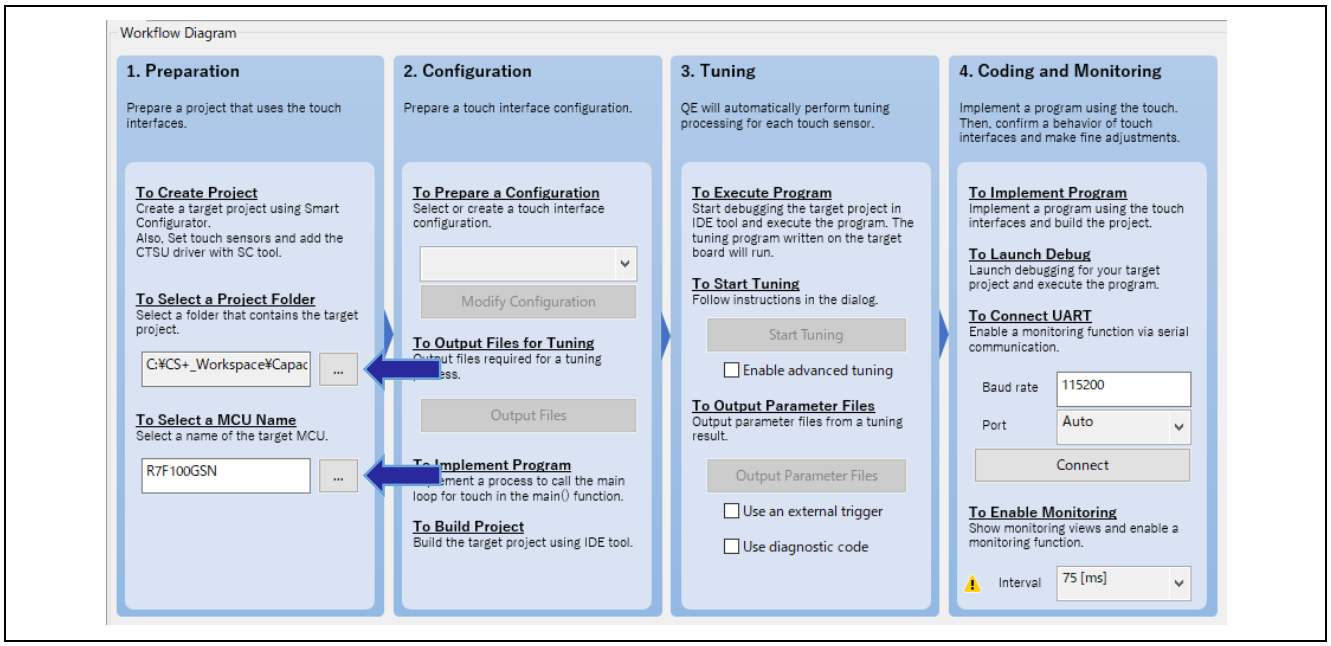


Figure 8-3. Preparation

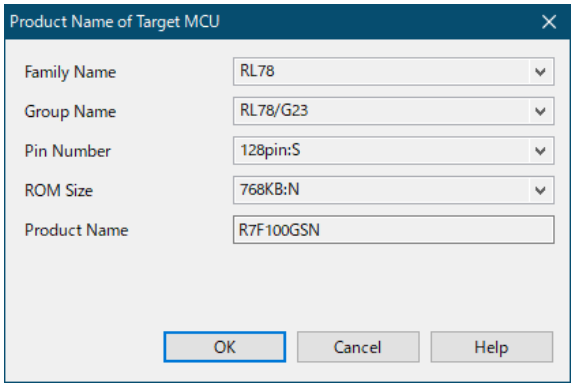


Figure 8-4. To Select a MCU Name

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.

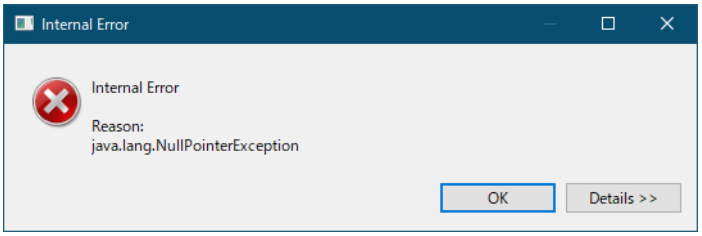


Figure 8-5. Internal Error

8.3 Configuration

Set items according to “configuration” of Workflow Diagram.

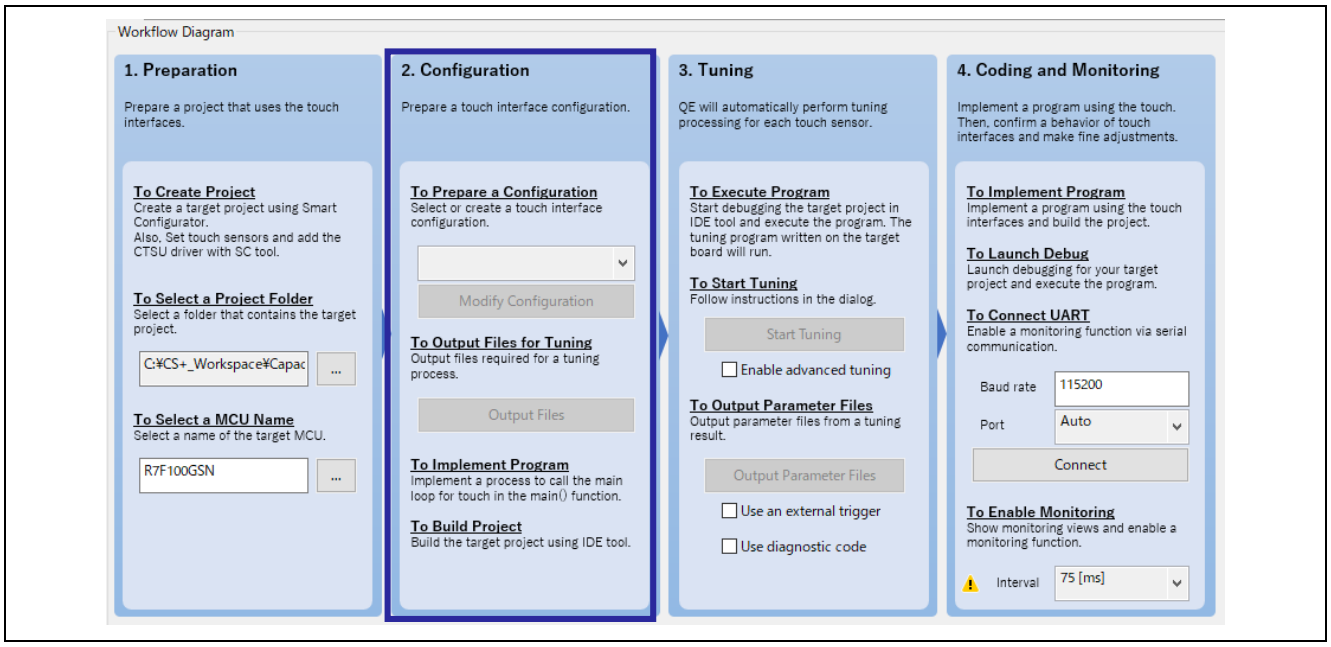


Figure 8-6. Workflow Diagram (Configurarion)

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

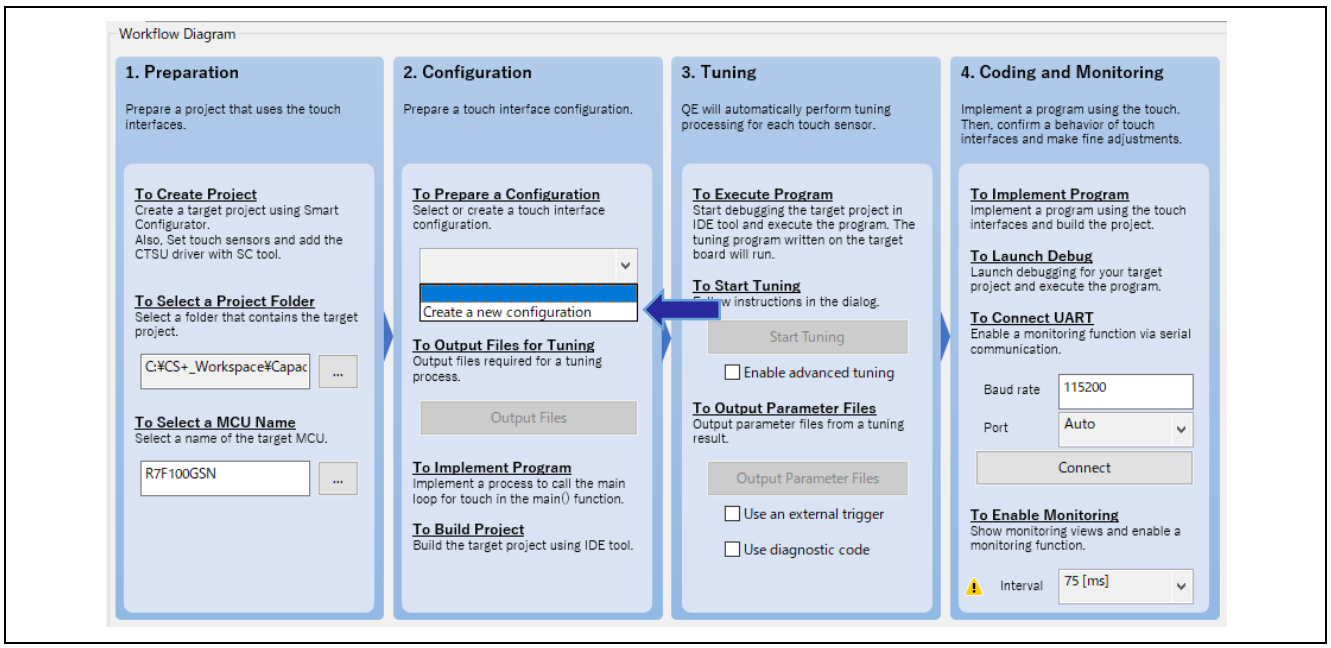


Figure 8-7. Create a New Configuration

2. “Create Configuration of Touch Interfaces” window appears and displays the area for setting touch interface.
   Click “Button” in “Touch I/F” panel at right side, and then cursor will be condition to add button at the area by click.
   Set two buttons as follows and click “esc” key to stop addition of touch interface.

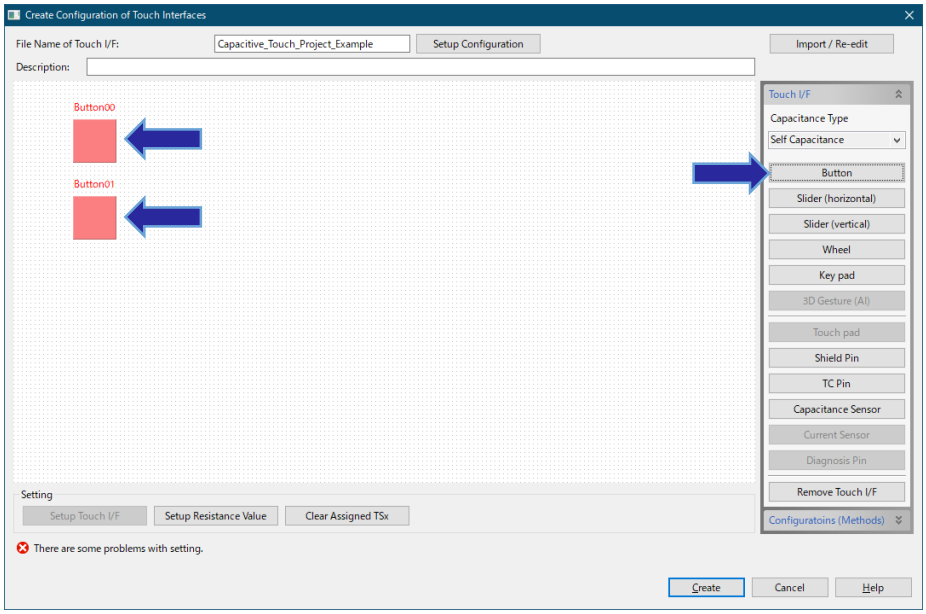


Figure 8-8. Adding Button

3. Double click “Button00” of the area, and set as follows in “Setup Touch Interface” dialog.
   — Touch Sensor : (A TS pin set in chapter 7.3.2)
   — Resistance : (Resistance connected to the TS pin)

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

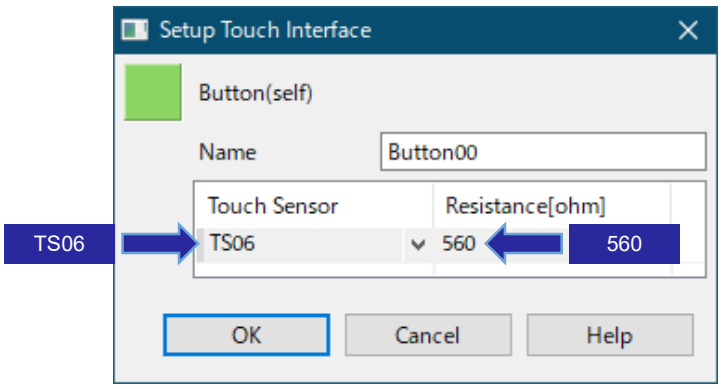


Figure 8-9. Setup of Touch Interface

4. Also set “Button01” as follows.
   — Touch Sensor : (The other pin set in chapter 7.3.2)
   — Resistance : (Resistance connected to the TS pin)

5. After setting touch interface, the area is as follows. Then click “Create”.

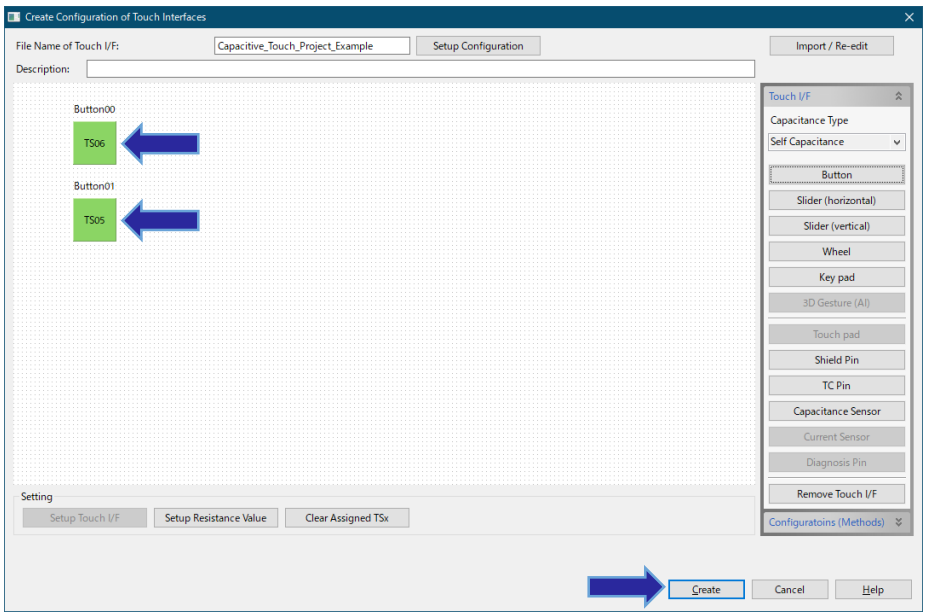


Figure 8-10. Touch Interface Configuration after Setting

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

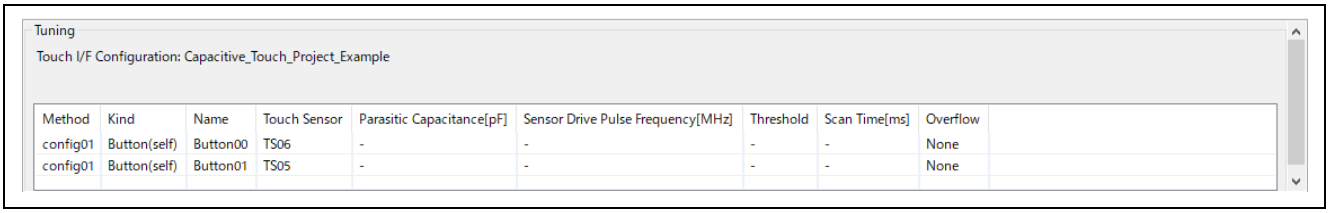


Figure 8-11. Touch Interface configuration

7. 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
```

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

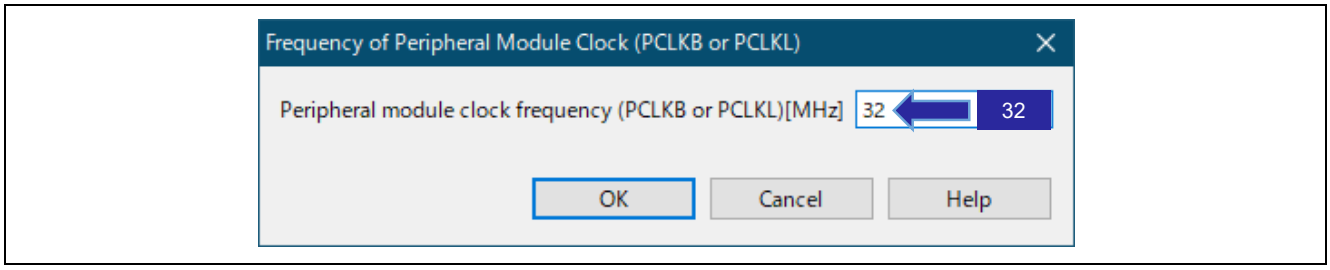


Figure 8-12. Setting Frequency of Peripheral Module Clock

9. In the following dialog, set power supply voltage and click “OK”.
   Please confirm electric characteristic of your using microcontroller.

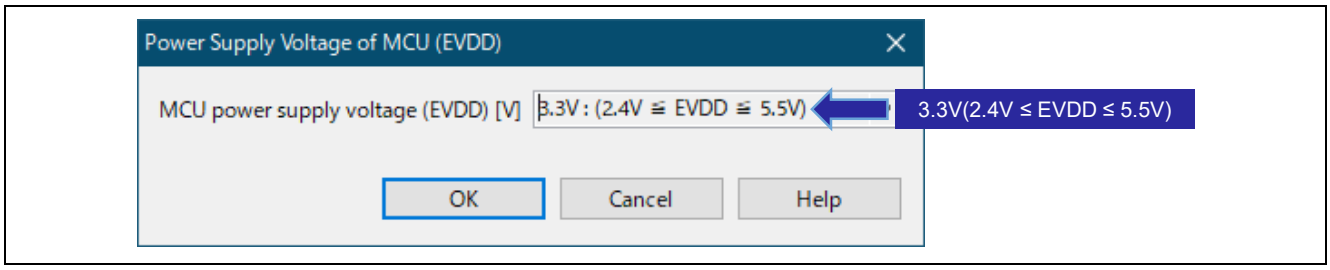


Figure 8-13. Setting Power Supply Voltage of MCU

10. 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.

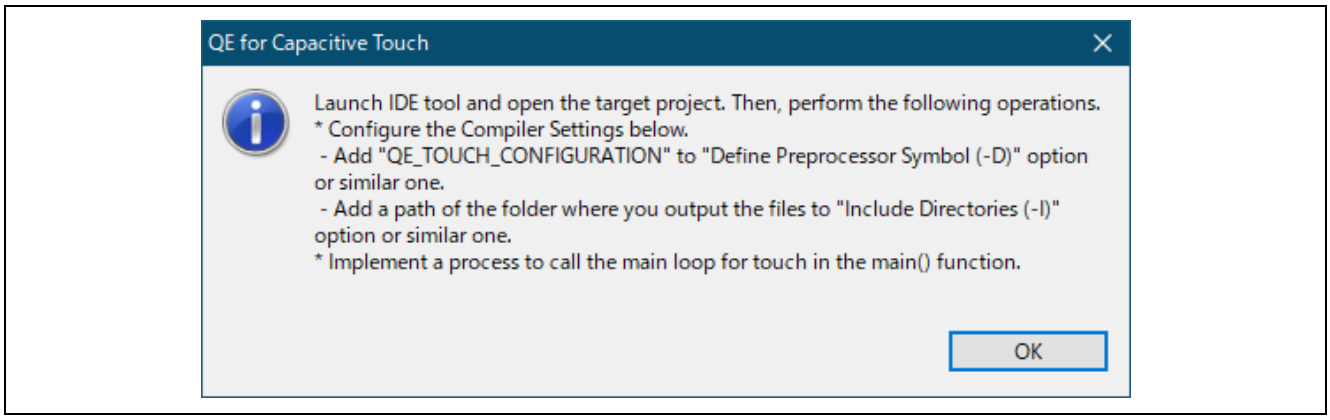


Figure 8-14. QE for Capacitive Touch Dialog

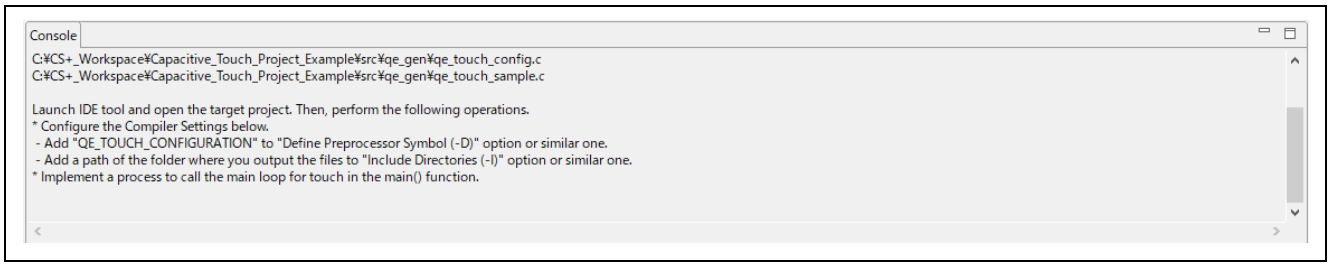


Figure 8-15. Console

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.

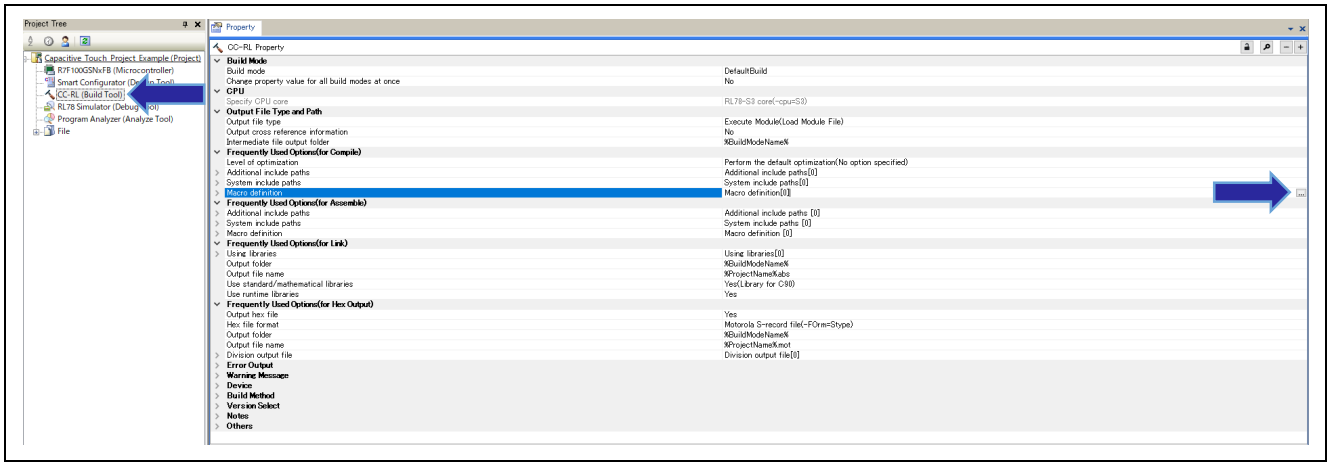


Figure 8-16. Selecting Macro Definition

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

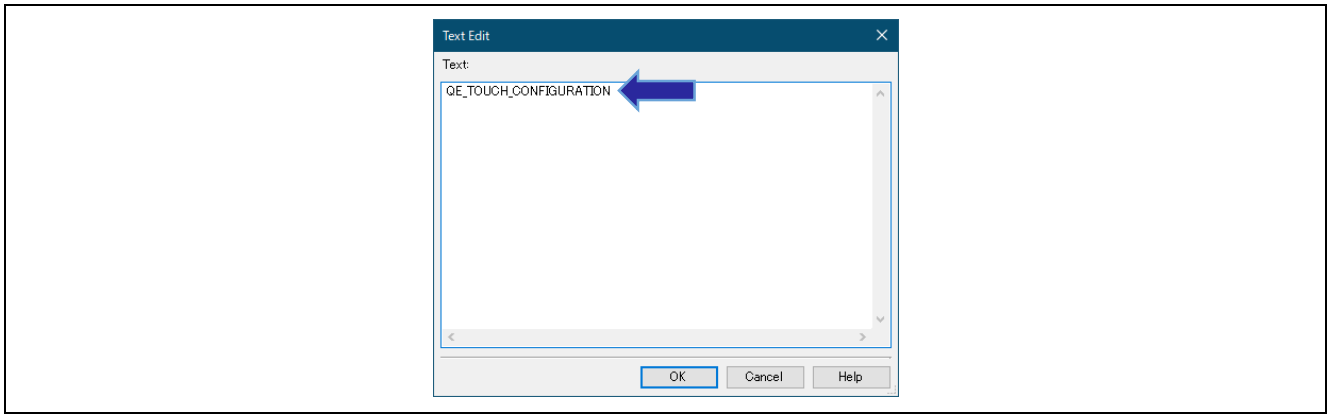


Figure 8-17. Edit Macro Definition

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”.

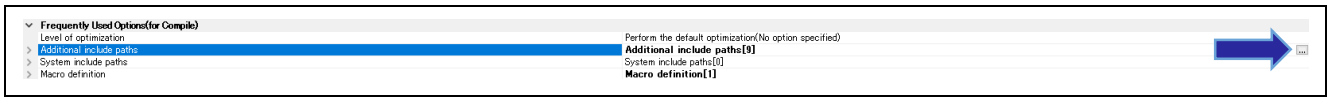


Figure 8-18. Additional Include Paths

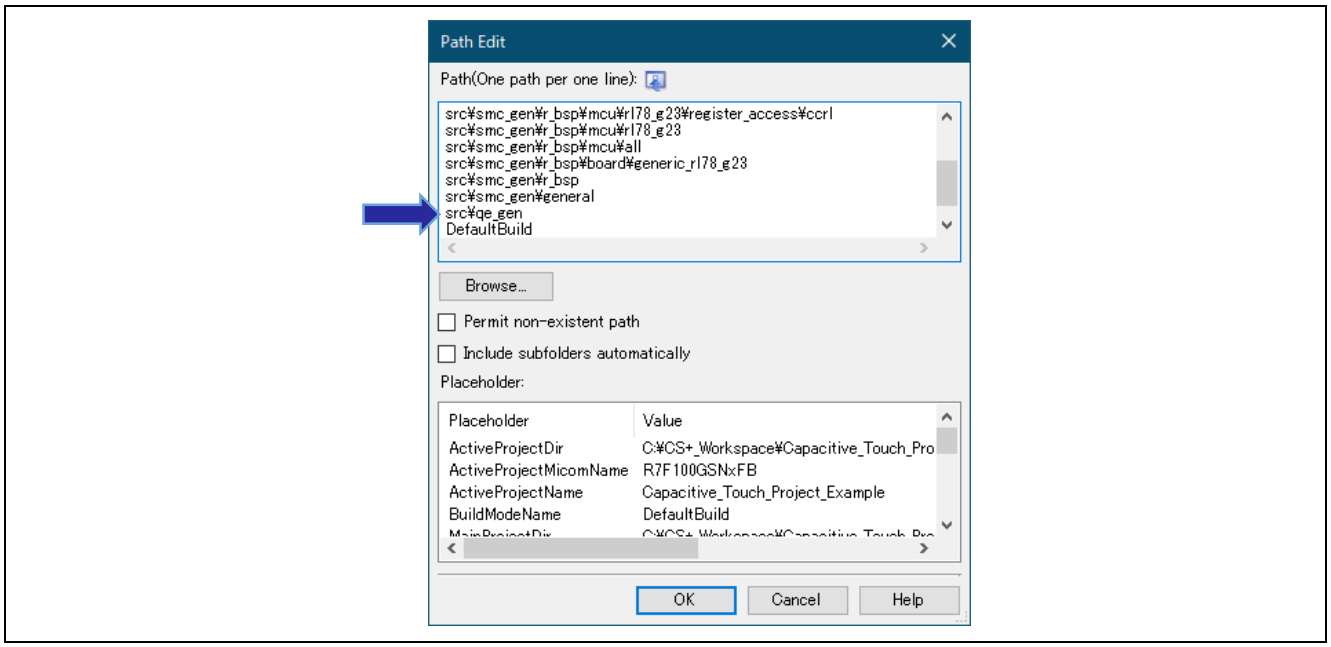


Figure 8-19. Path Edit

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 Explorer by drag and drop.

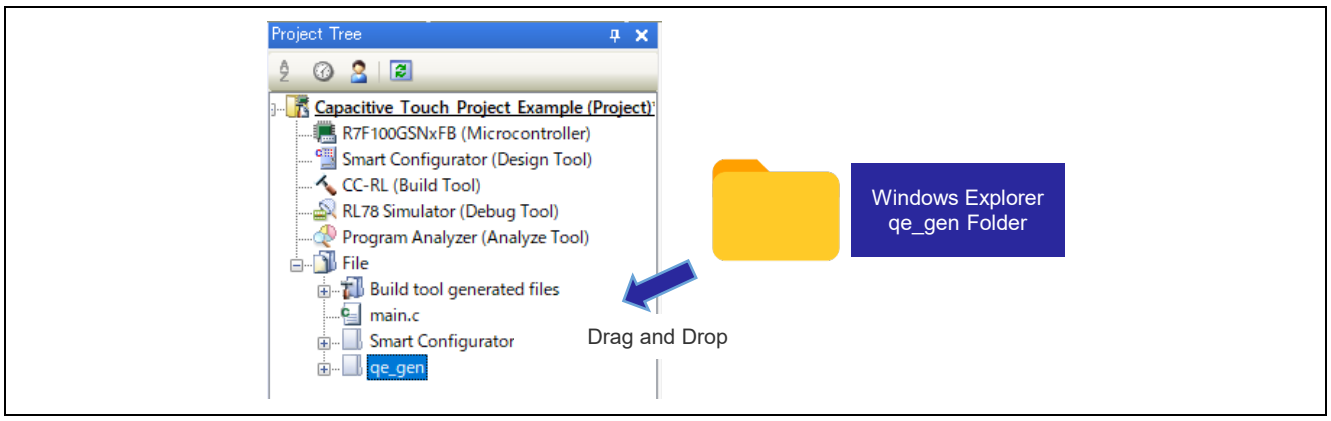


Figure 8-20. Adding “qe_gen” Folder to Project Tree

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

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

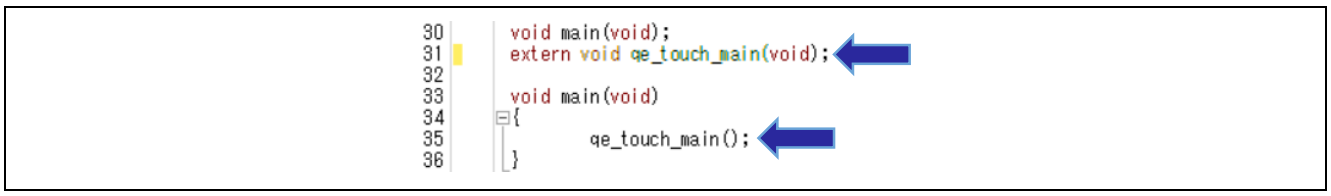


Figure 8-21. main.c

11. Add function for serial communication to “Config_<UART Channel>_user.c”.
    Add the following code.
    — extern void touch_uart_callback(uint16_t event);
    — touch_uart_callback(0);
    — touch_uart_callback(1);

```
52    /* Start user code for global. Do not edit comment generated here */
53    extern void touch_uart_callback(uint16_t event);
54    /* End user code. Do not edit comment generated here */



74    static void r_Config_UARTA1_callback_sendend(void)
75    {
76        /* Start user code for r_Config_UARTA1_callback_sendend. Do not edit comment generated here */
77        touch_uart_callback(0);
78        /* End user code. Do not edit comment generated here */
79    }



87    static void r_Config_UARTA1_callback_receiveend(void)
88    {
89        /* Start user code for r_Config_UARTA1_callback_receiveend. Do not edit comment generated here */
90        touch_uart_callback(1);
91        /* End user code. Do not edit comment generated here */
92    }
```

Figure 8-22. Config_URATA1_user.c

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

8.4 Tuning

Set according to “Tuning” of Workflow Diagram.

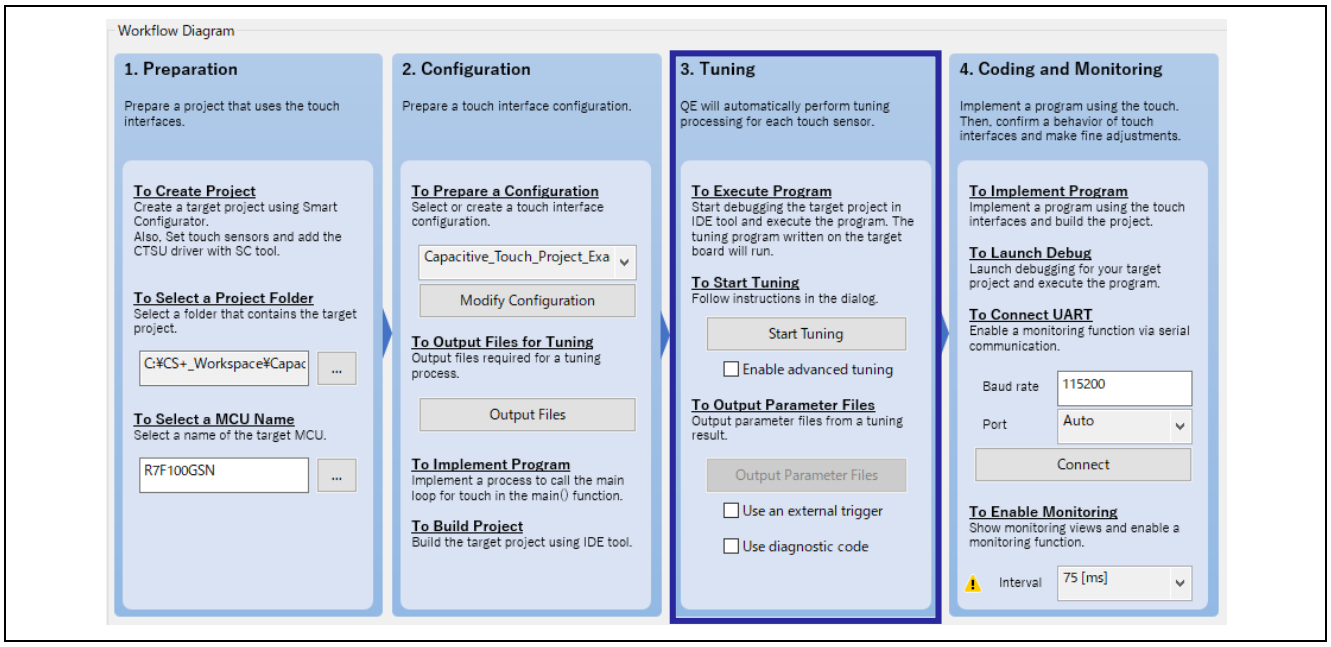


Figure 8-23. Workflow Diagram (Tuning)

1. Right-click “Debug Tool” of project tree of CS+, and click “Using Debug Tool”. Select debug tool depending on your debug environmental.

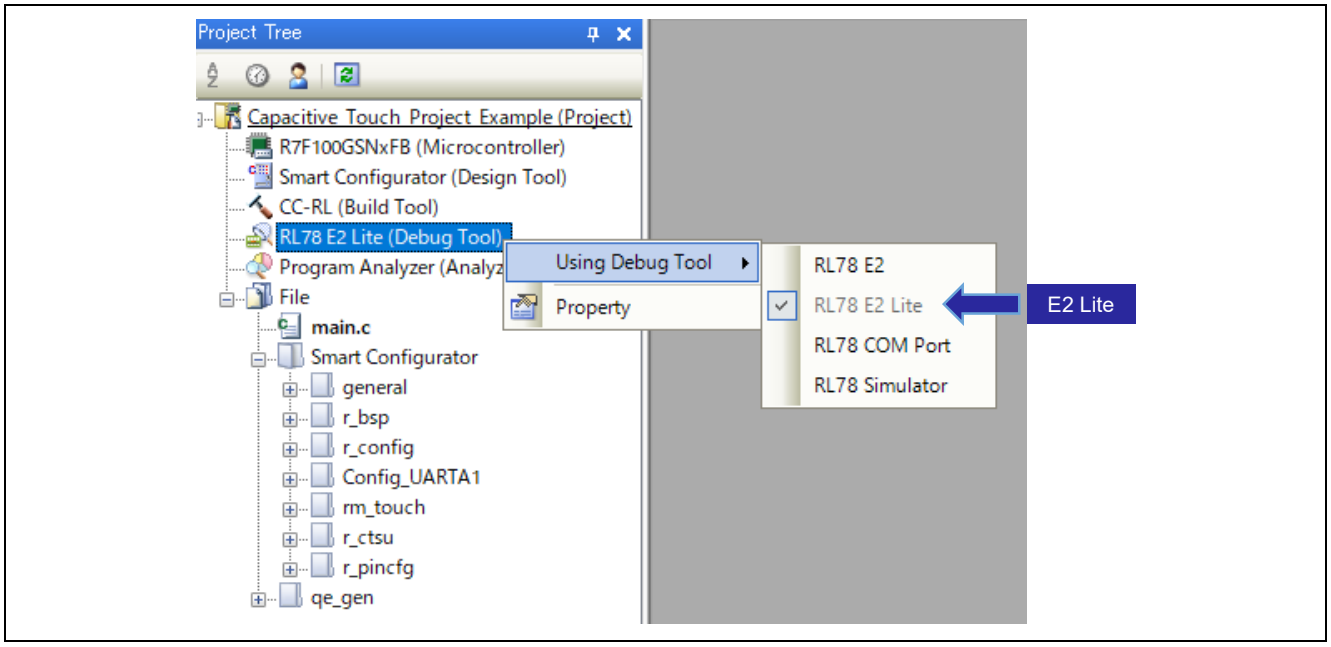


Figure 8-24. Selecting Debug Tool

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

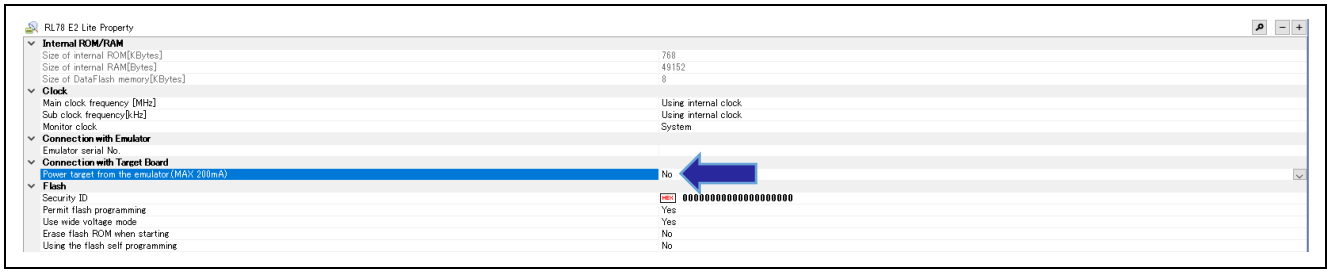


Figure 8-25. Property of Debug Tool

3. Confirm connection between PC and target board, and click  icon to download the program to debug tool after build. 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 7.4.

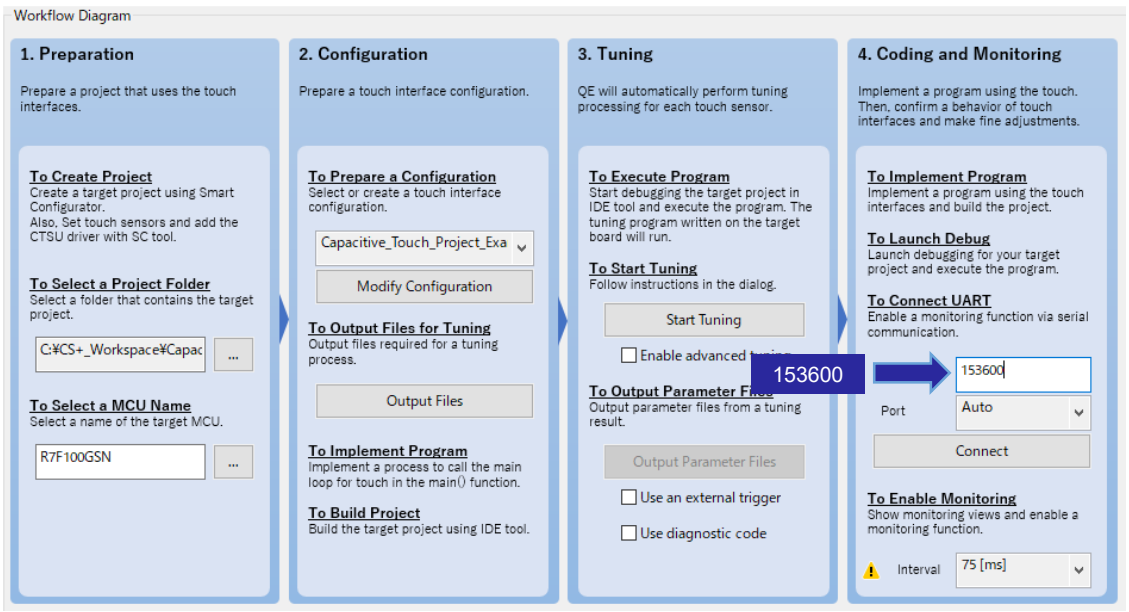


Figure 8-26. Setting Baud Rate

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

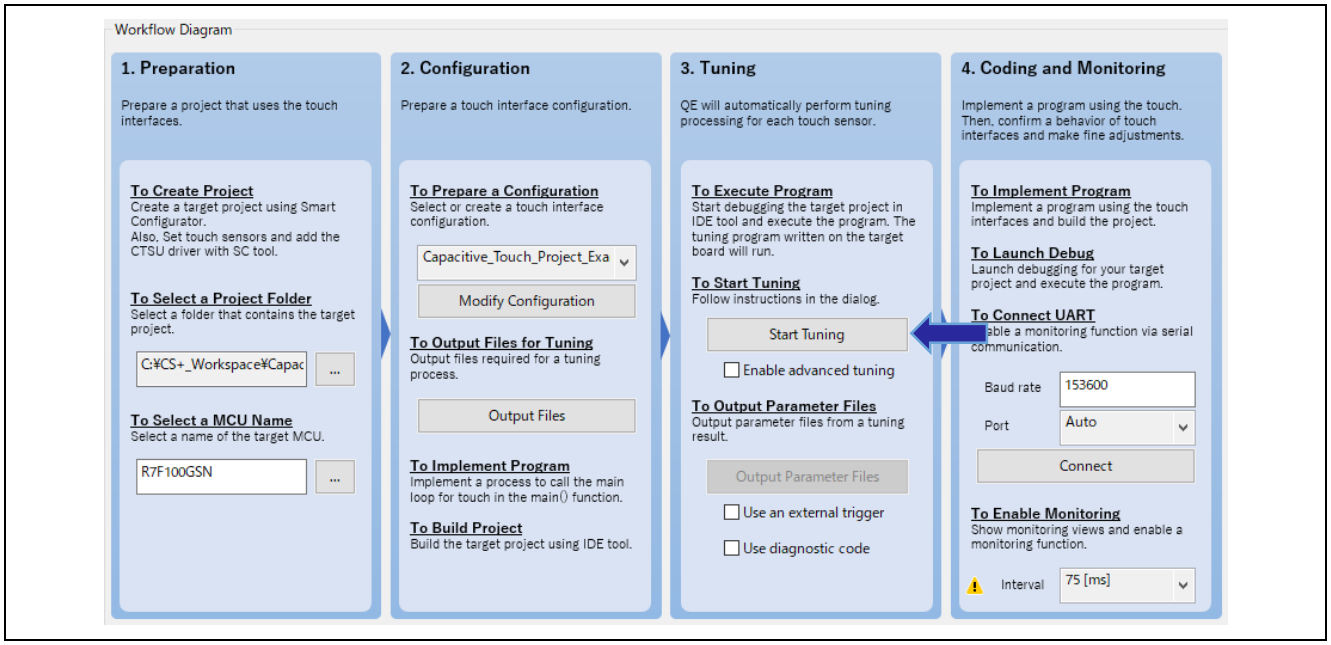


Figure 8-27. Tuning

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

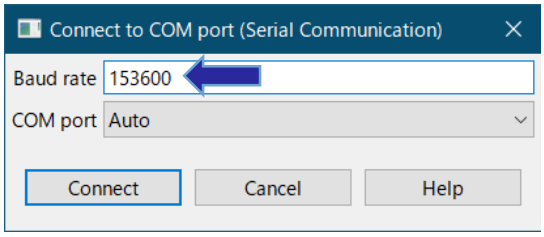


Figure 8-28. Setting Baud Rate

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

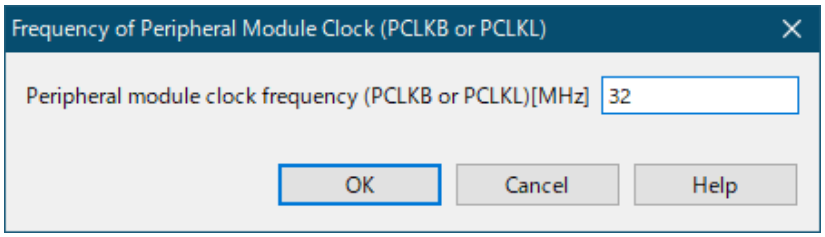


Figure 8-29. Setting Frequency of Peripheral Module Clock

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

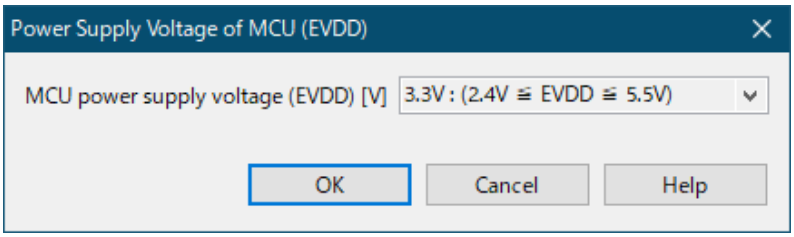


Figure 8-30. Setting Power Supply Voltage of MCU

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

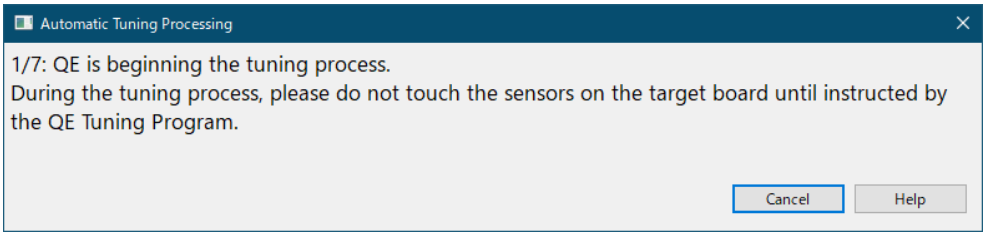


Figure 8-31. Automatic Tuning Processing Dialog

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 extend to the right, and also touch counts increase.
Under touching, press any key on the PC keyboard to accept the sensitivity measurement.

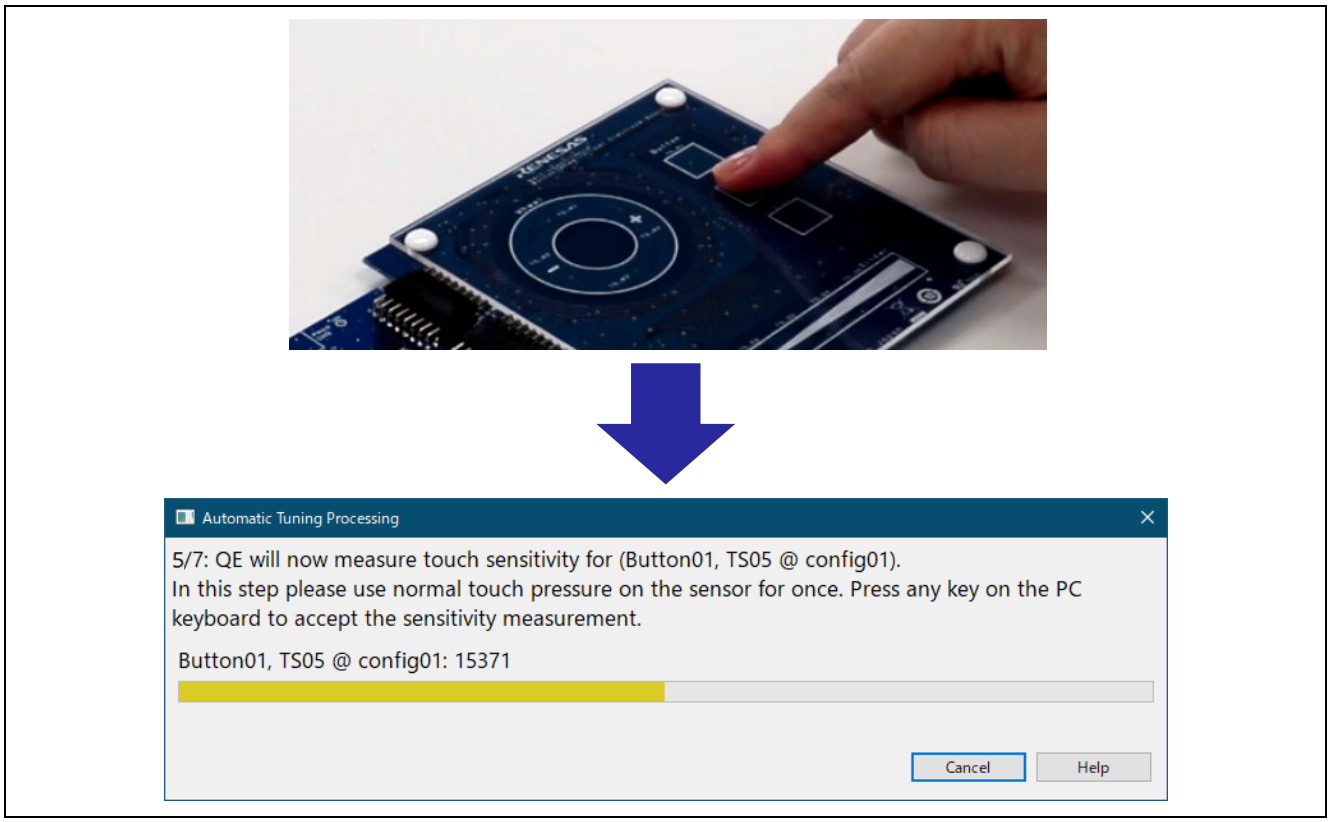


Figure 8-32. Measuring Touch Sensitivity

10. Likewise, measure the touch sensitivity of the other touch sensor.

11. After tuning, it is possible to check the threshold in the following dialog. This threshold is used for judgement of touch event on the middleware.
Click “Continue the Tuning Process”. Then Tuning is done.

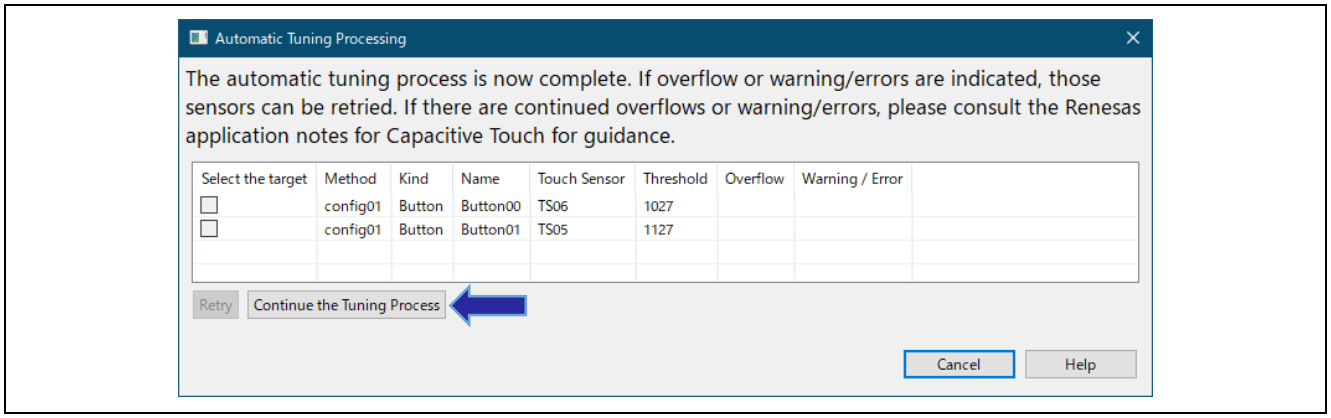


Figure 8-33. Threshold of Touch Sensor

12. 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

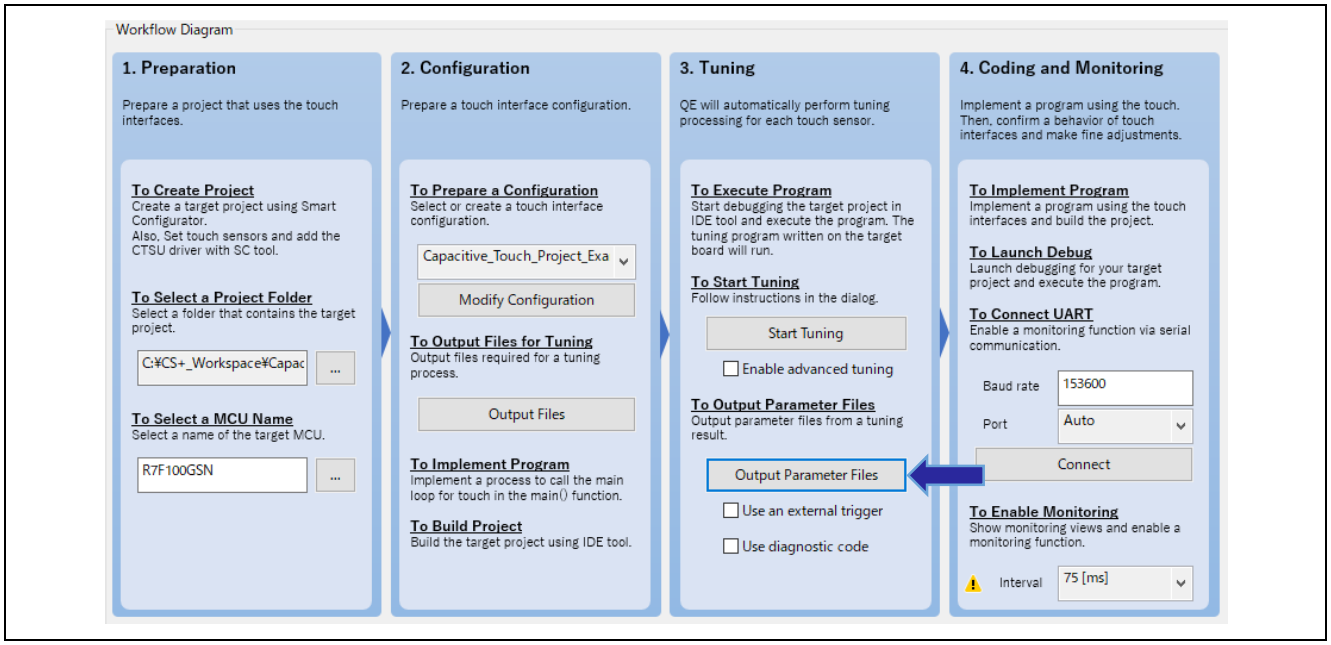


Figure 8-34 To Output Parameter Files

13. On CS+, click  icon to stop the program, and click  icon to disconnect from debug tool.

8.5 Coding and Monitoring

8.5.1 Monitoring

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

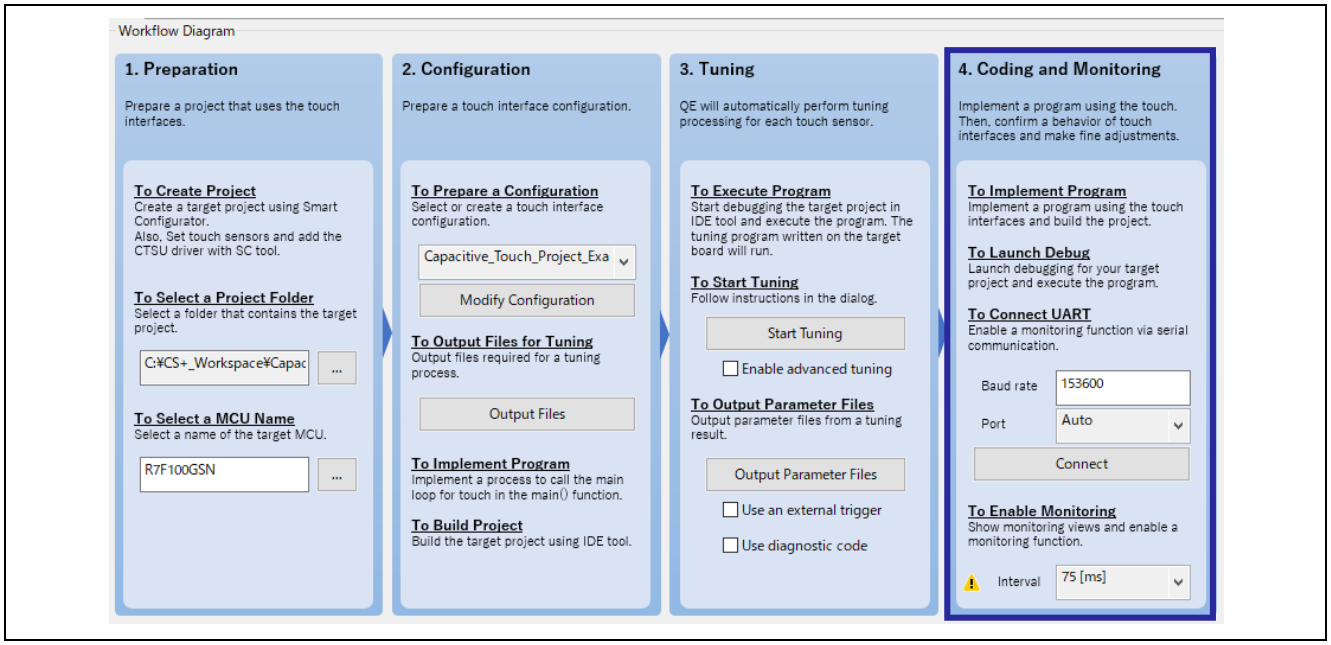


Figure 8-35. Workflow Diagram (Coding and Monitoring)

1. Confirm connection between PC and target board, and click  icon to download the program to debug tool after build. After finishing downloading the program, click  icon to execute the program.

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

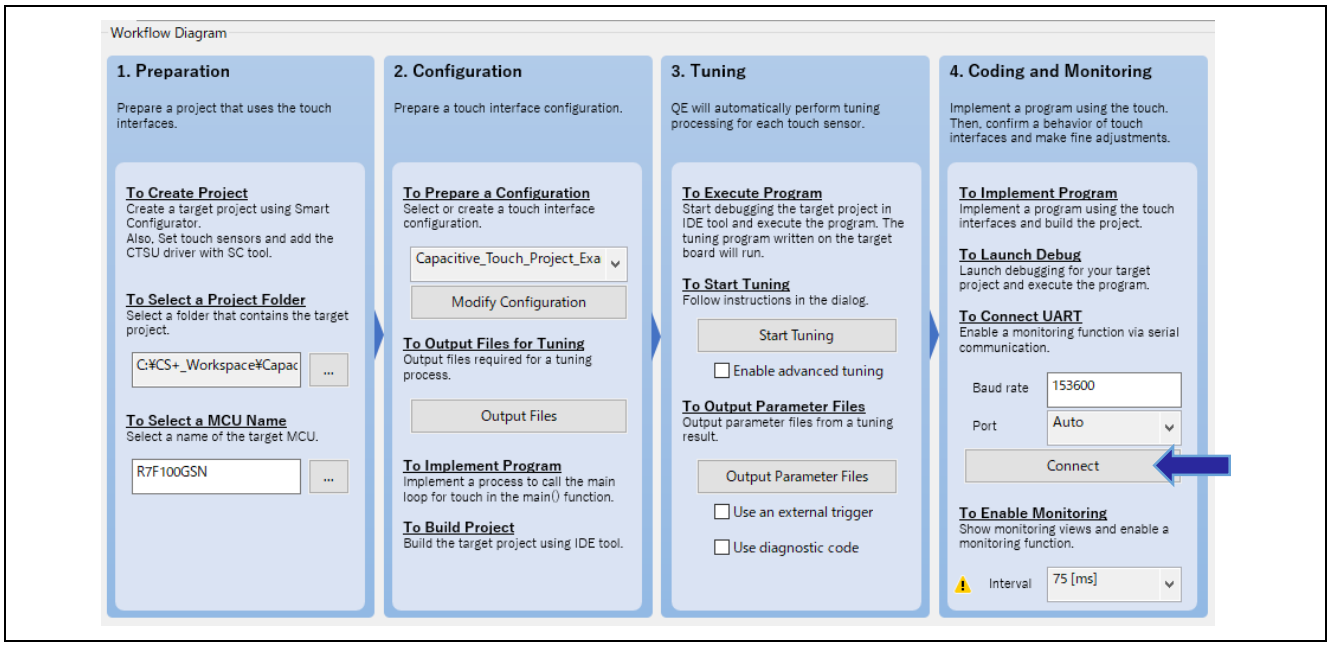


Figure 8-36. To Connect UART

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

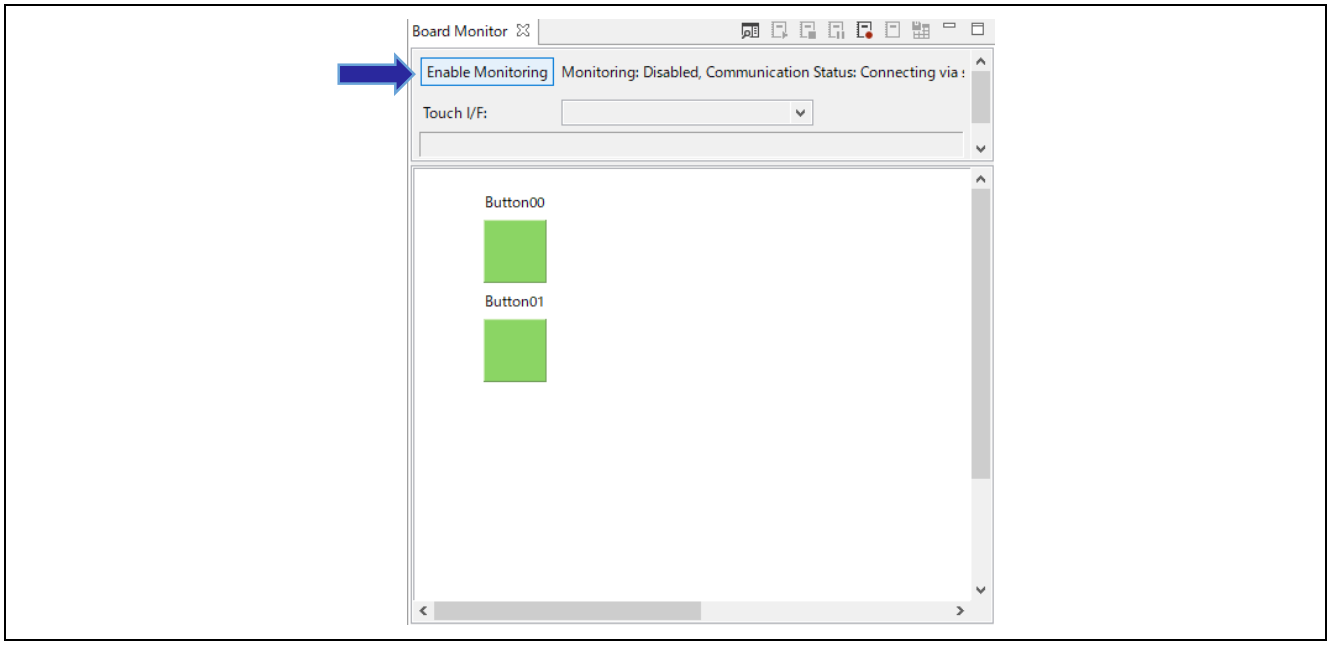


Figure 8-37. Enable Monitoring

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

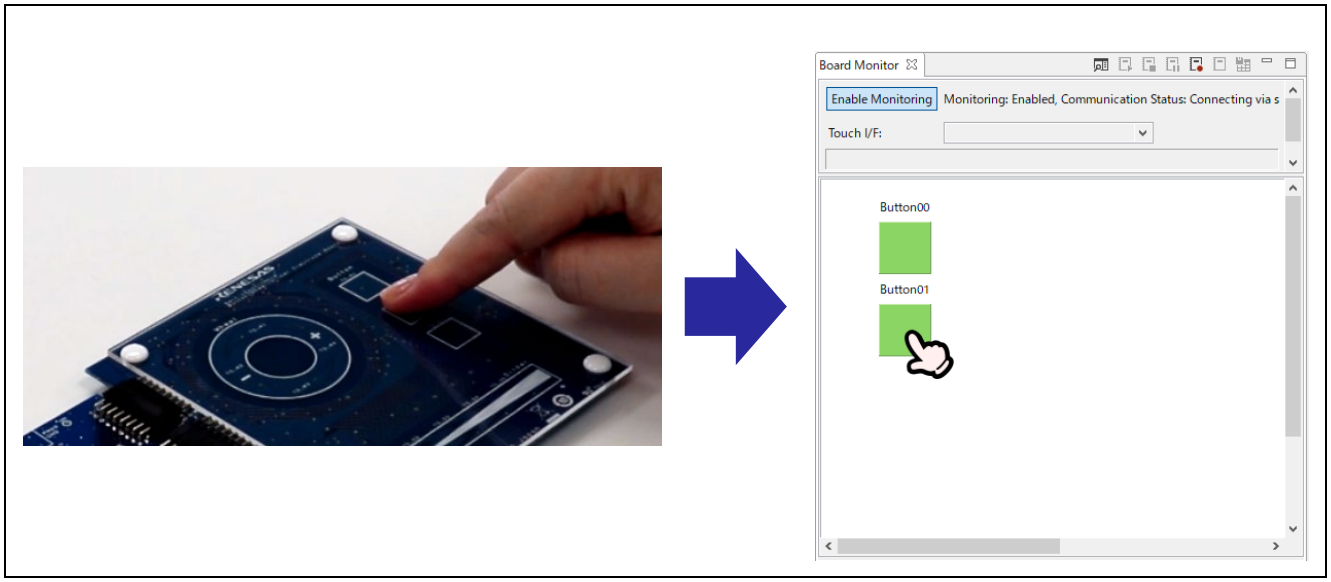


Figure 8-38. Display of the Condition while Touching

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 use 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.

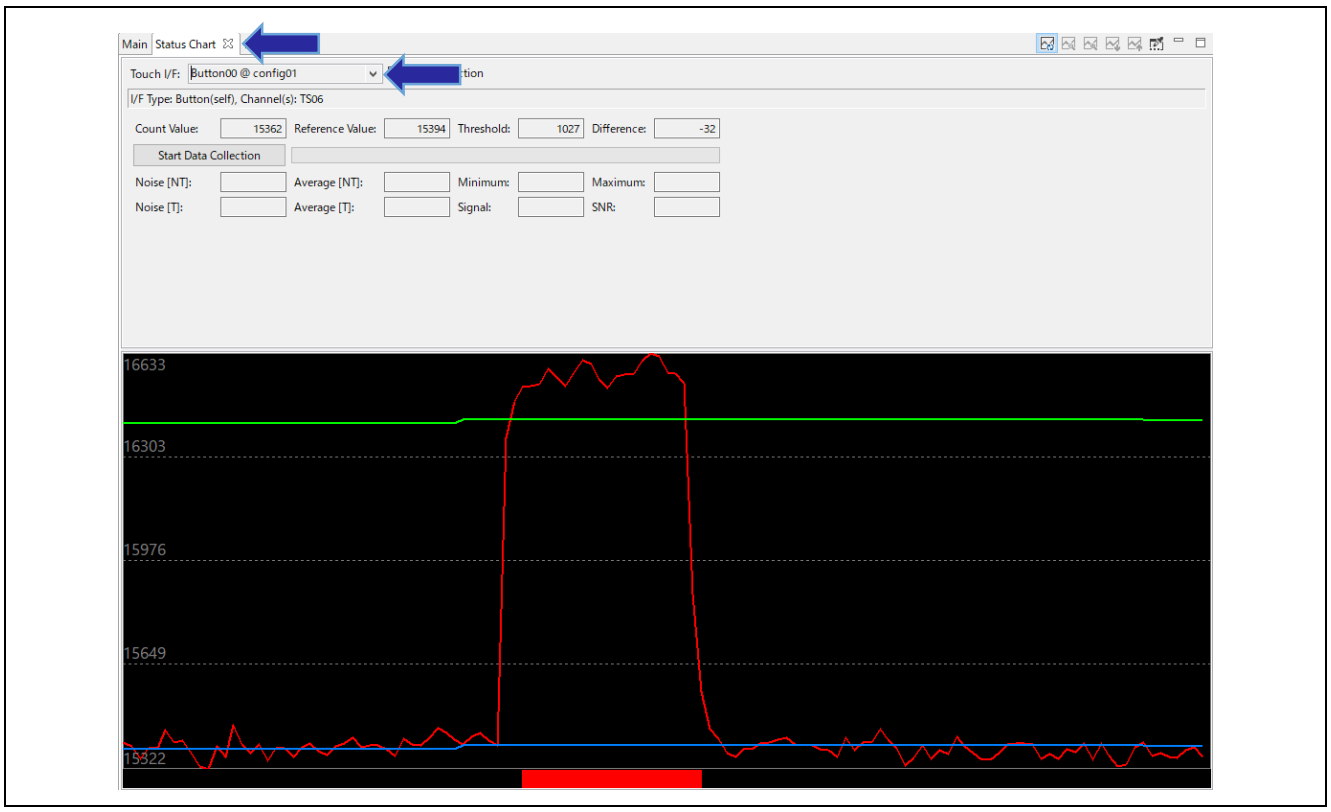


Figure 8-39. Graphical Representation of Touch Counts

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.

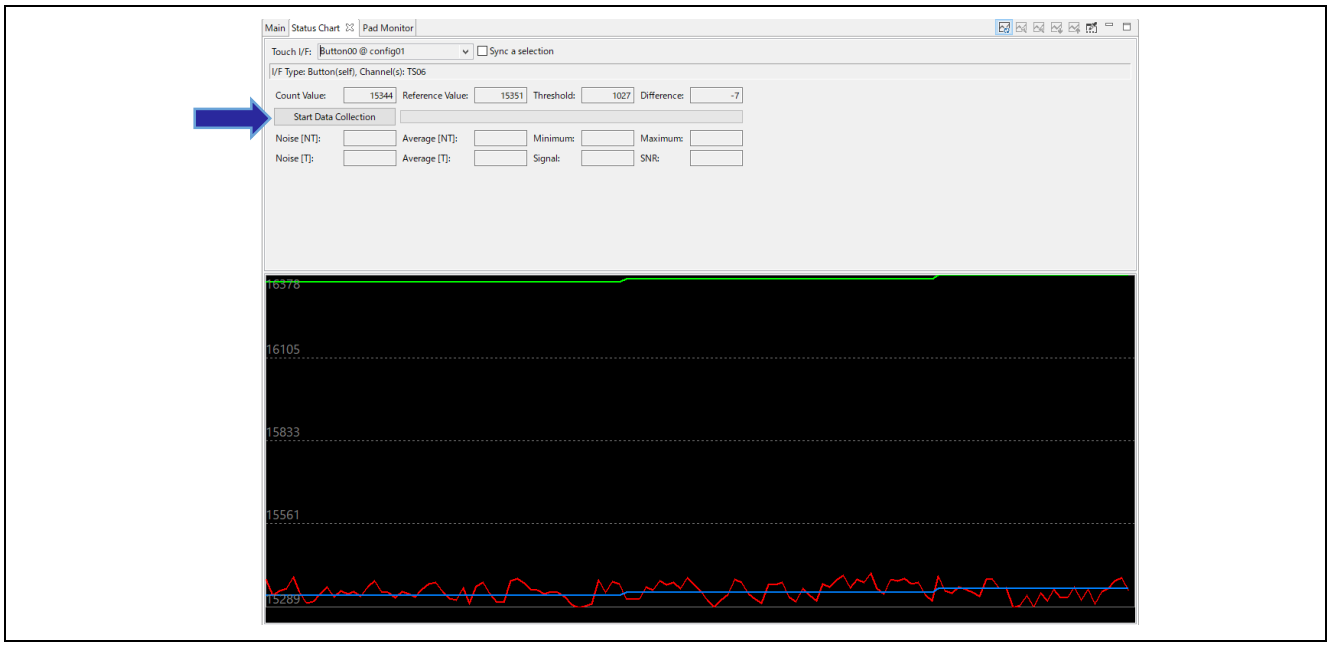


Figure 8-40. Data Collection of Touch-off State

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

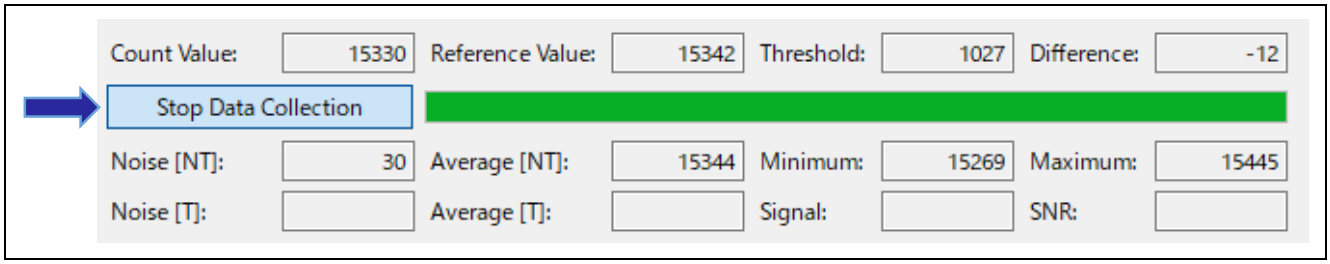


Figure 8-41. Stop Data Collection

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

D. After finishing data collection, SNR value appears.

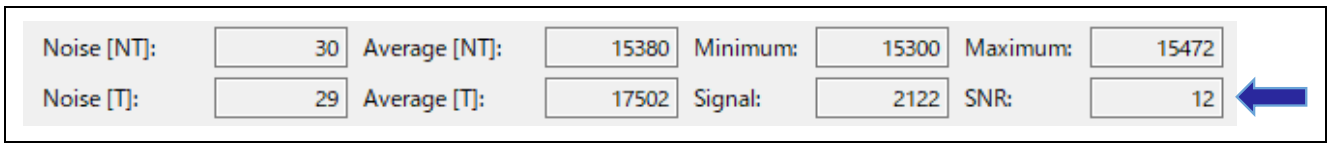


Figure 8-42. SNR value

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 8-43. Multi Status Chart

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

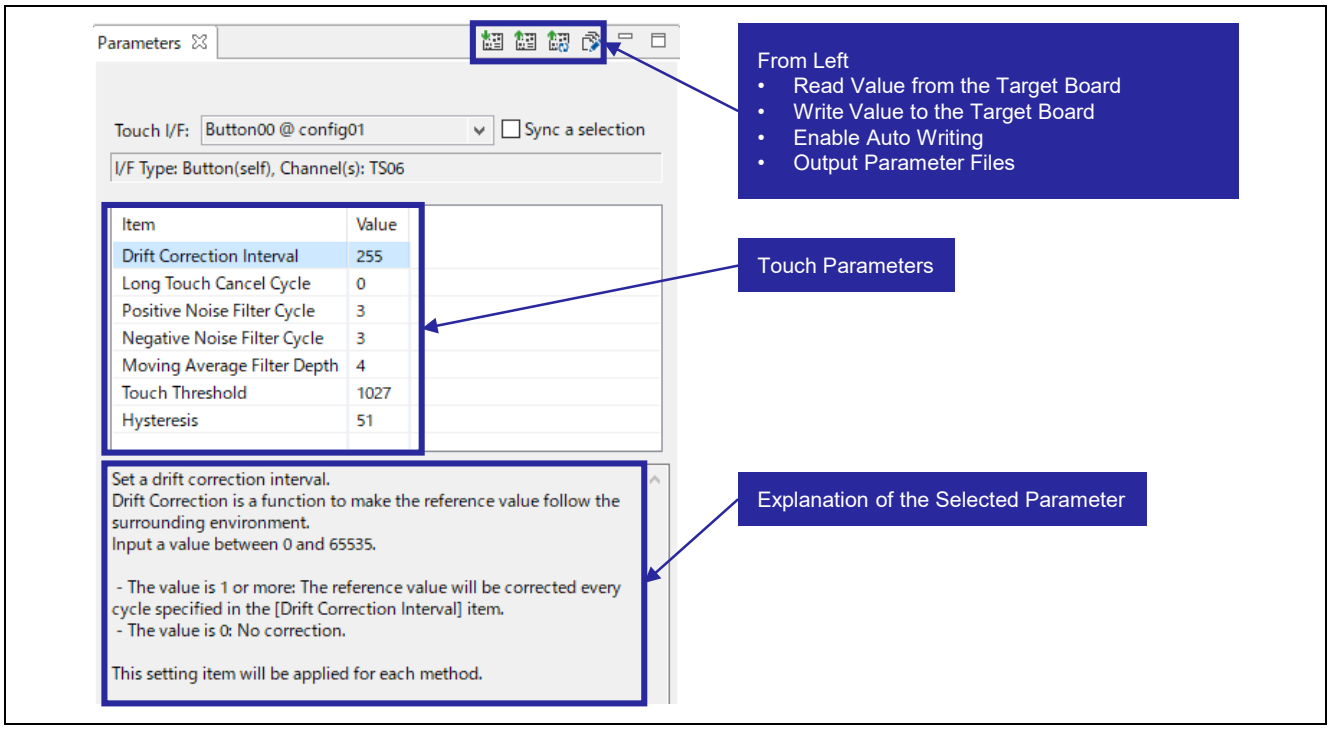


Figure 8-44. Adjustment of Parameters

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

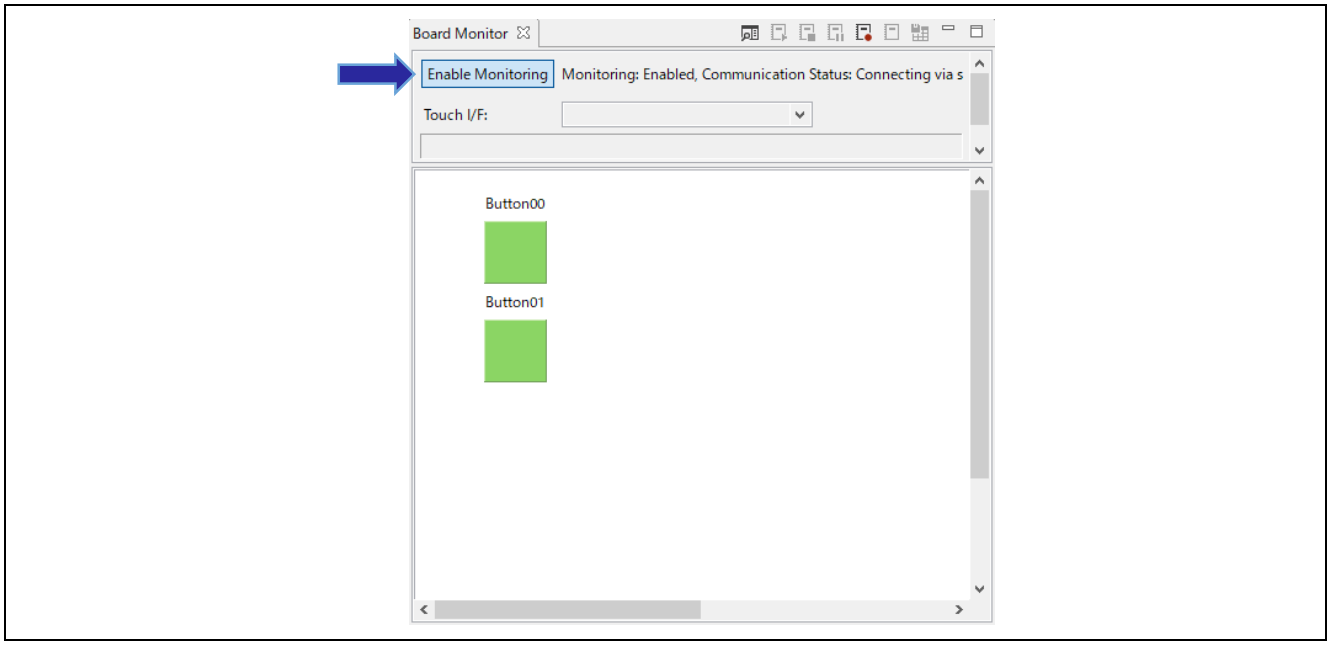


Figure 8-45. Stop Monitoring

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

Figure 8-46. Disconnect UART

11. On the CS+, click  icon to stop the program, and click  icon to disconnect from debug tool.

8.6 Sample Code

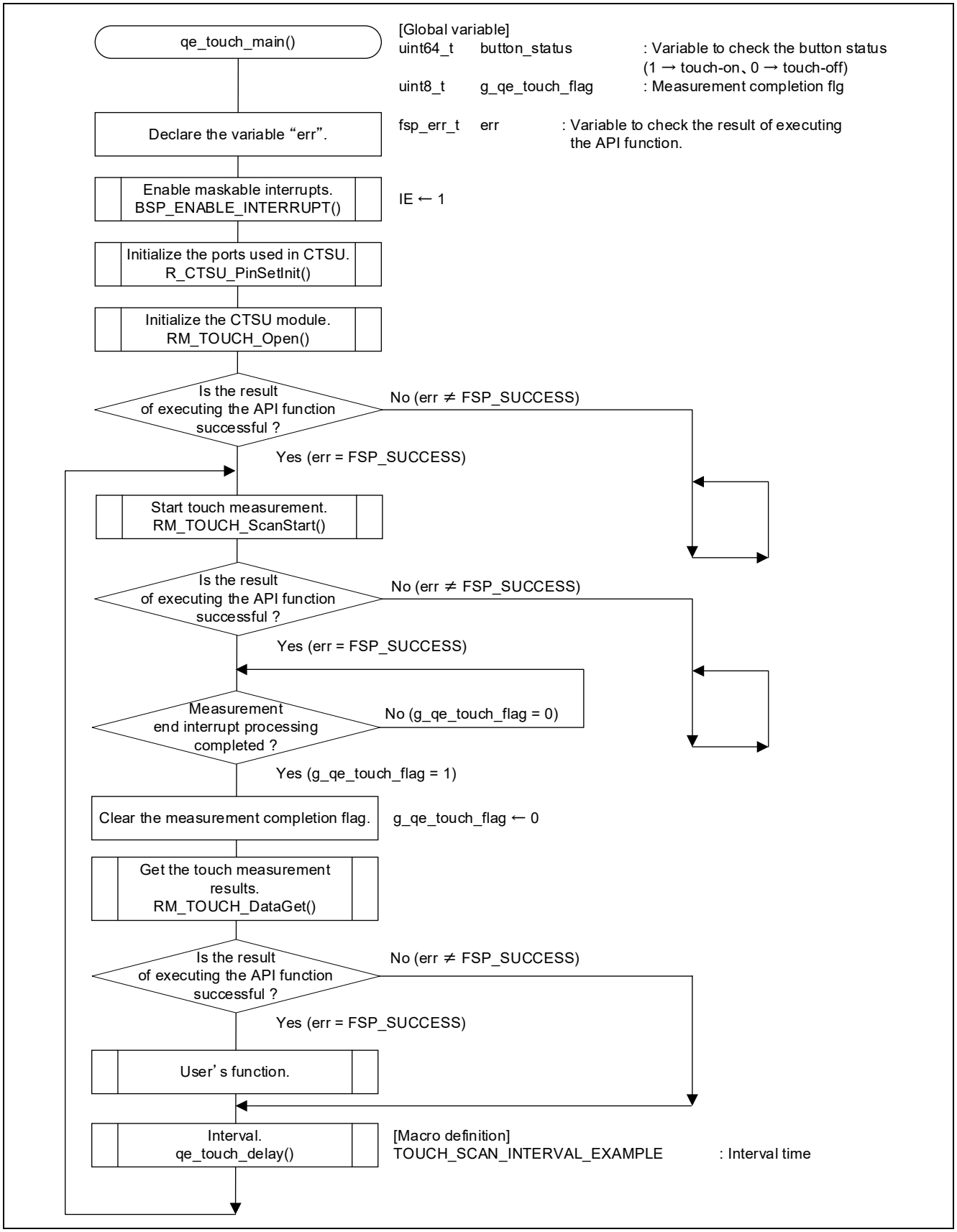
The sample code (qe_touch_sample.c) outputted by QE for Capacitive Touch is as follows.

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

```
/***********************************************************************
*
* FILE : qe_sample_main.c
* DATE : 2021-07-29
* 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, NULL, NULL);
        if (FSP_SUCCESS == err)
        {
            /* TODO: Add your own code here. */
        }

        /* FIXME: Since this is a temporary process, so re-create a waiting process yourself. */
        qe_touch_delay(TOUCH_SCAN_INTERVAL_EXAMPLE);
    }
}

void qe_touch_delay(uint16_t delay_us)
{
    uint32_t i;
    uint32_t loops_required;
    uint16_t clock_mhz;
    clock_mhz = (uint16_t)(R_BSP_GetFclkFreqHz() / 1000000);
    if (0 == clock_mhz)
    {
        clock_mhz = 1;
    }
    loops_required = ((uint32_t)delay_us * (uint32_t)clock_mhz);
    loops_required /= 20;
    for (i = 0; i < loops_required; i++)
    {
        BSP_NOP();
    }
}
```

8.7 Flowcharts



9. Appendix

9.1 Touch Measurement by Hardware Timer

This section explains the program using hardware timer (Interval Timer) to create a touch measurement cycle.

To test operation, turn on/off LED depending on the state of touch sensor.

9.1.1 Setup of Smart Configurator

1. Select “Clocks” tab on Smart Configurator and set clock (fSXP) for interval timer.

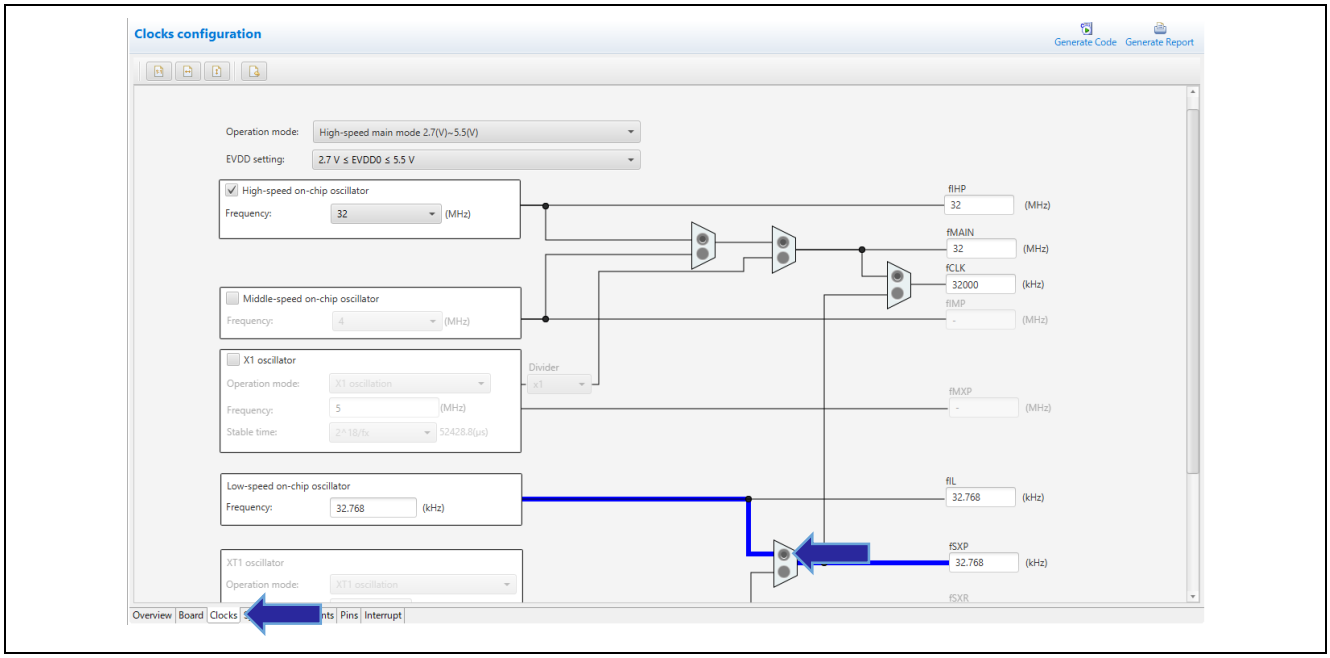


Figure 9-1. Setting Clock

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”.

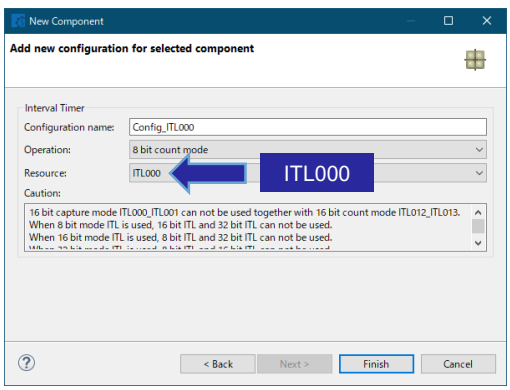


Figure 9-2. Configuration of Interval Timer

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

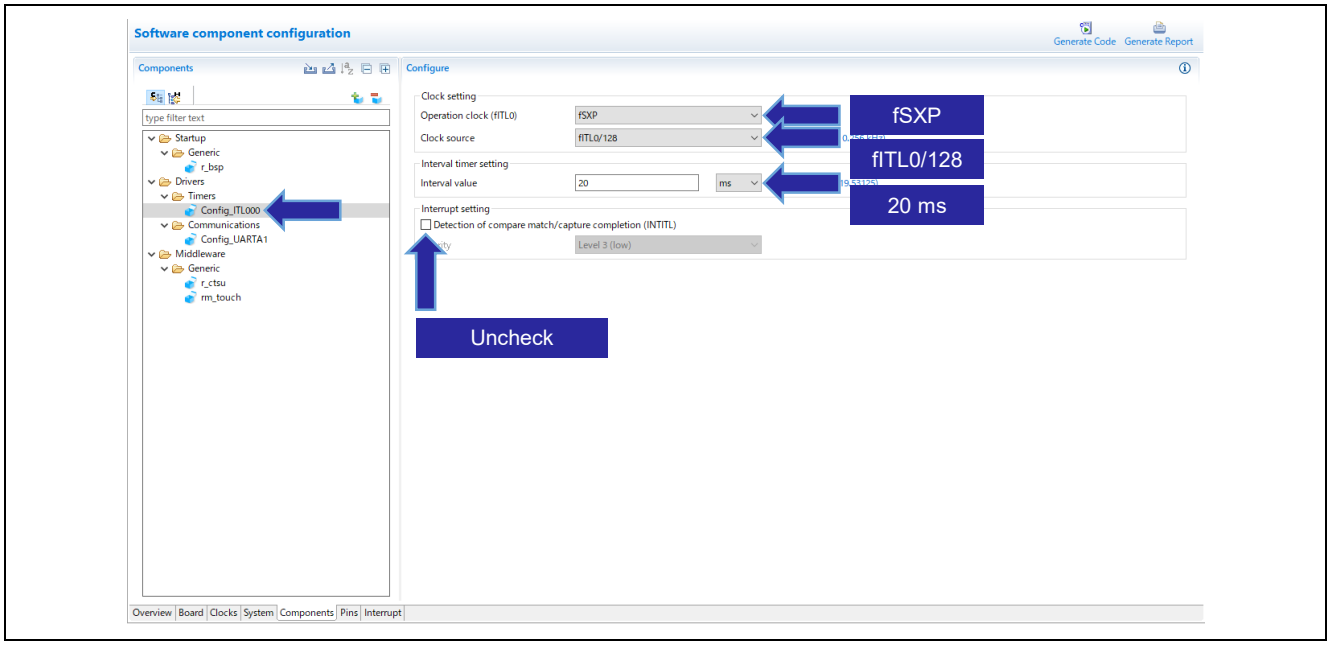


Figure 9-3. Setting “Config_ITL000”

4. Set the Pin for LED. Set “P60” to high-level output in “Ports” module.

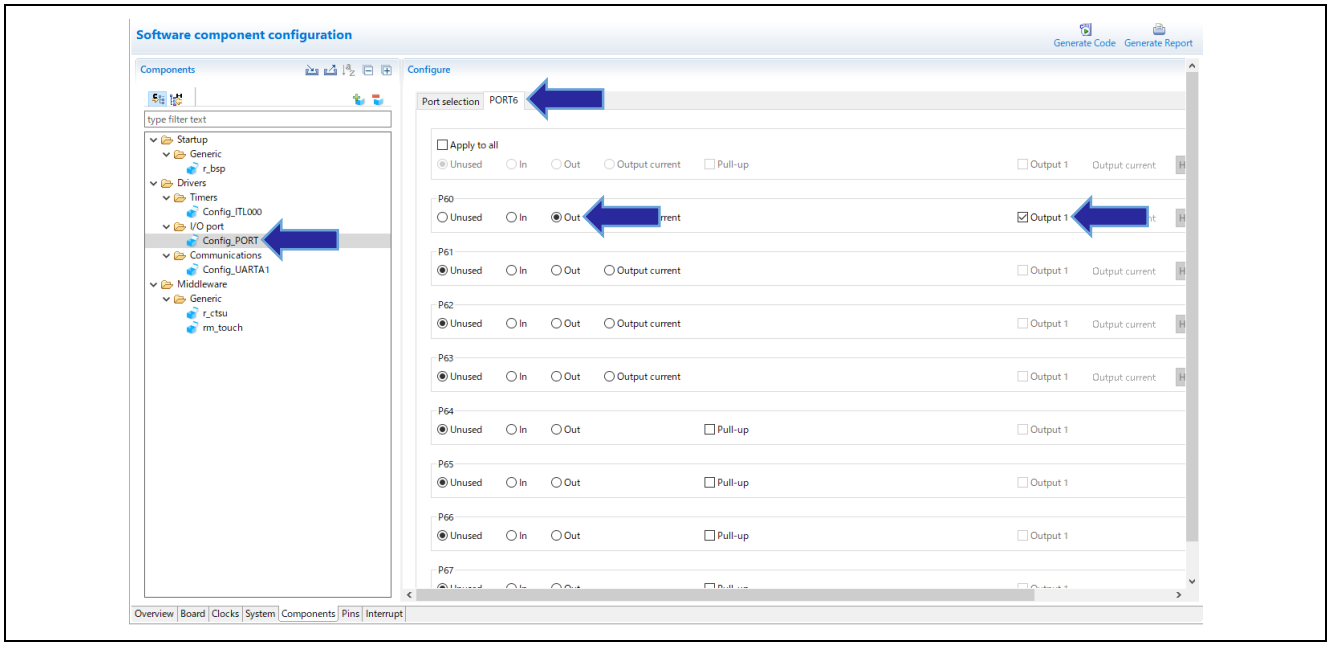


Figure 9-4. Setting “P60”

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

9.1.2 Sample Code

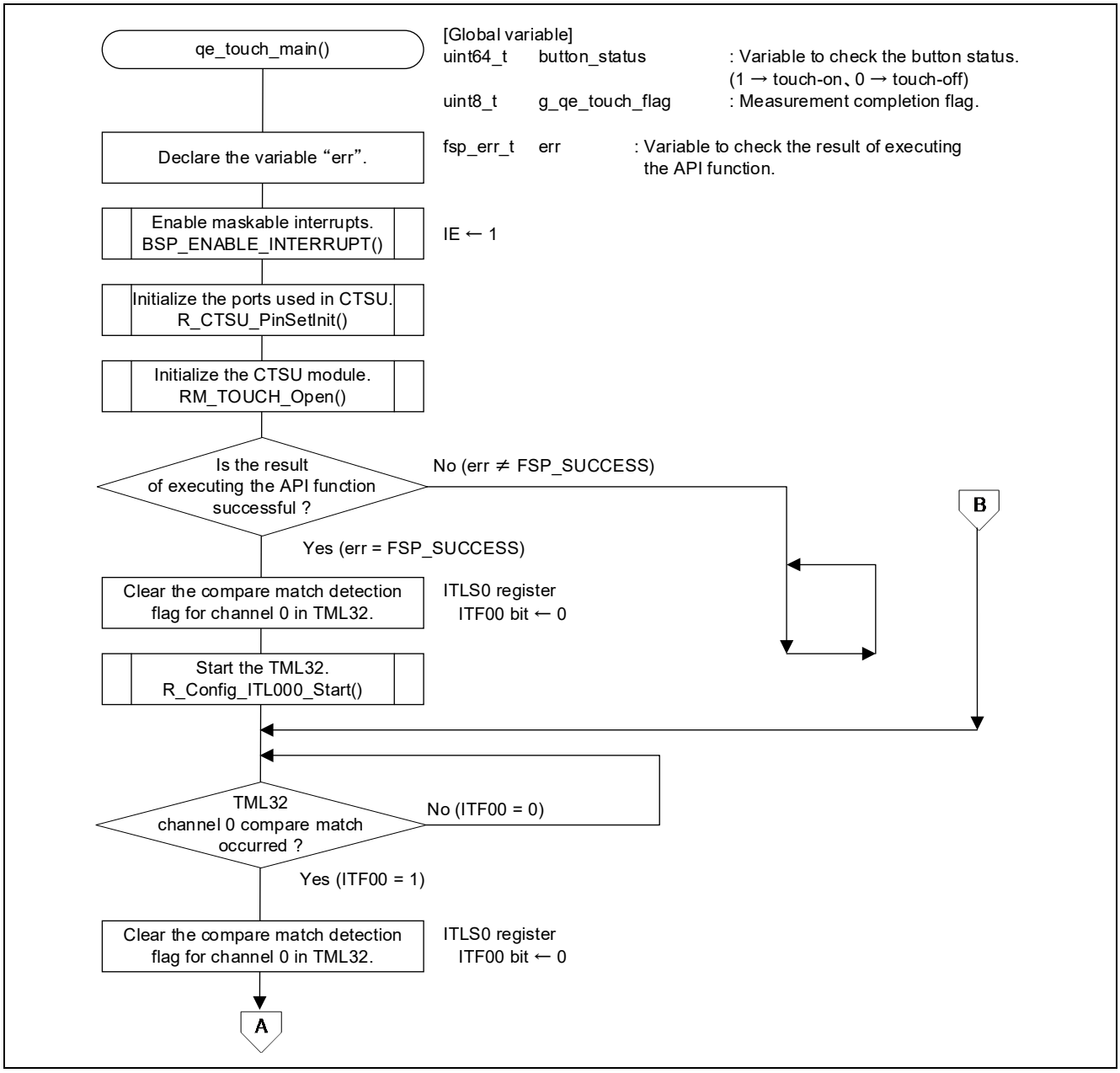
The sample code (qe_touch_sample.c) outputted by QE for Capacitive Touch is as follows.

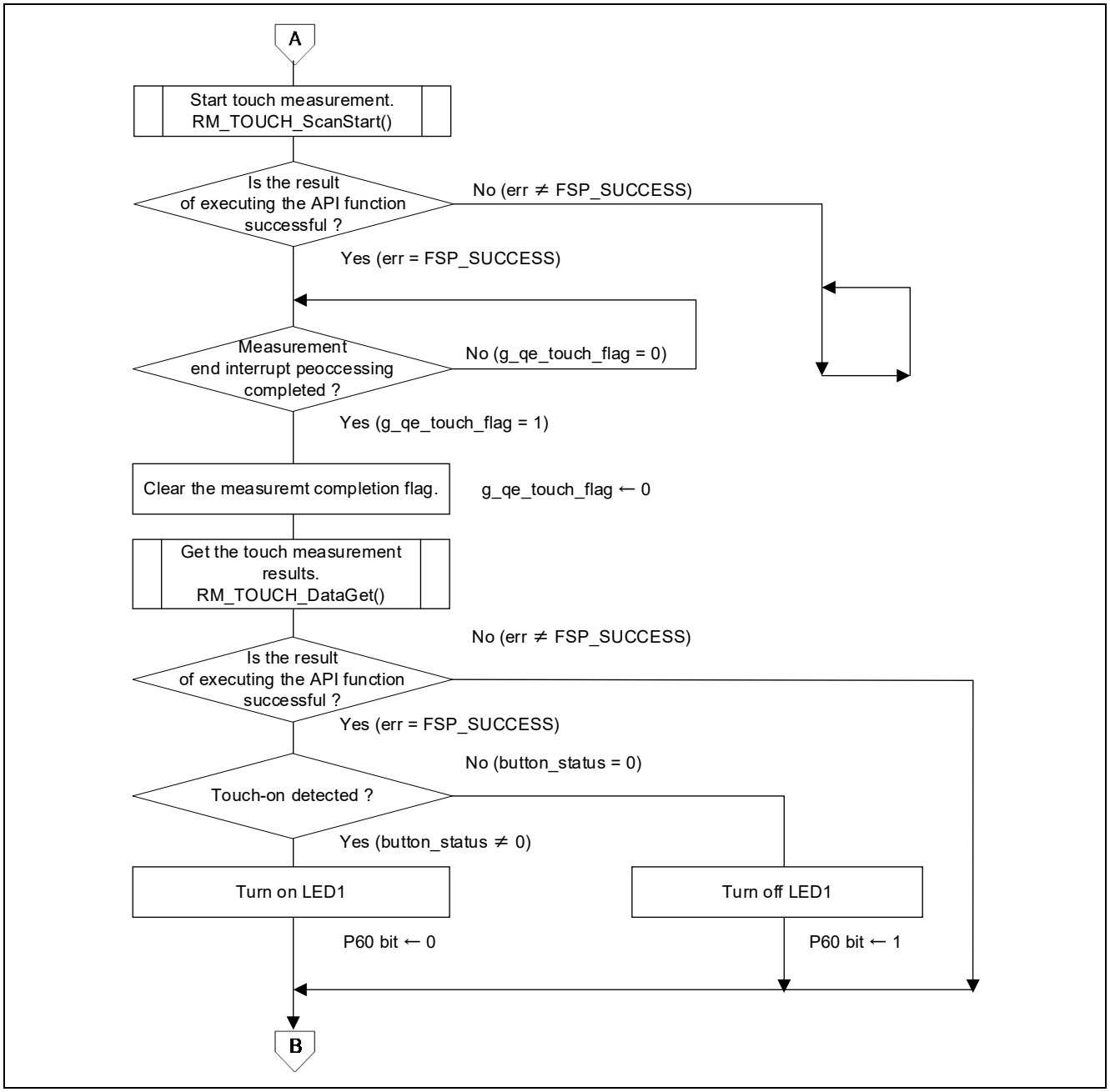
In this sample code, a touch measurement cycle is created using a hardware timer.

```
/***********************************************************************
*
* FILE : qe_sample_main.c
* DATE : 2022-05-09
* 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, 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;
            }
        }
    }
}
```

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
P60 bit ← 0
P60 bit ← 1
B

10. Documents for Reference

- RL78/G23 User's Manual: Hardware (R01UH0896)
- RL78 Family User's Manual: Software (R01US0015)
- 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 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)
  (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	Sep.02.22	-	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.

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/.