Introduction
This document will demonstrate the necessary steps for creating an application example that integrates capacitive touch sensing using Renesas RA Microcontrollers.

Target Device
RA family with Capacitive Touch Sensing Unit (CTSU, CTSU2)
Contents

1. Application Example Overview .................................................................................................................. 3
2. Related Documents ..................................................................................................................................... 3
3. High Level Integration Steps ...................................................................................................................... 3
4. Required Development Tools and Software Components ........................................................................... 3
5. Application Example Overview .................................................................................................................. 4
6. Capacitance Touch Application Development Procedure ............................................................................ 4
   6.1 Project Creation ...................................................................................................................................... 4
   6.2 Using RA Configurator to Add Modules ................................................................................................. 8
   6.3 Creating the Capacitive Touch Interface ................................................................................................ 25
   6.4 Modifying Debug Session for Capacitive Touch Tuning ....................................................................... 31
   6.5 Tuning the Capacitive Touch Interface Using QE for Capacitive Touch Plug-in ..................................... 32
   6.6 Adding rm_touch middleware API Calls to Application Example .......................................................... 36
   6.7 Monitoring Touch Performance Using e²studio Expressions Window and QE for Capacitive Touch .......... 40
   6.8 Monitoring Touch Performance with QE for Capacitive Touch Using Serial Communication ............... 49
7. qe_touch_sample.c Listing After Modifications .......................................................................................... 52
   7.1 When Using a Software Trigger to Start the CTSU Measurement Process ............................................. 52
   7.2 When Using an External Trigger to Start the CTSU Measurement Process ............................................ 54

Website and Support ....................................................................................................................................... 56

Revision History .............................................................................................................................................. 57
1. **Application Example Overview**

This document will demonstrate how to implement capacitive touch sensing functions using Renesas RA MCUs based on the following methods:

- Create a project using the RA Smart Configurator with the RA6M2 MCU board or RA2L1 MCU board
- Create a cap touch interface with QE for Capacitive Touch for tuning and monitoring

2. **Related Documents**

This application example is intended to give the user a short introduction to creating a working RA capacitive touch sensing project. A thorough review of all the applicable documentation for the e2 studio/RA Smart Configurator, Flexible Software Package (FSP) drivers/middleware, and QE for Capacitive Touch plug-in help (contained within the e2 studio IDE help index) is strongly suggested to answer any questions or for more details on usage of any of the tools utilized in this application example.

3. **High Level Integration Steps**

The following high-level steps will give the reader an overview of the steps required to integrate touch detection into this project. These same steps should apply to any typical user development application.

1. Create a new project with the e2 studio project creation wizard.
2. Use the RA Smart Configurator to add the required modules to the created e2 studio project.
3. Use the QE for Capacitive Touch e2 studio plug-in to create the capacitive touch interface.
4. Use the QE for Capacitive Touch e2 studio plug-in to tune the application project.
5. Add the needed FSP module API calls to the user project to enable capacitive touch sensing operations in the application.
6. Monitor the application project using QE for Capacitive Touch e2 studio plug-in to demonstrate capacitive touch sensing detection.

4. **Required Development Tools and Software Components**

This project utilizes the following development environment:

<table>
<thead>
<tr>
<th>Table 1. Development Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Development Tool Software Components</strong></td>
</tr>
<tr>
<td>Board kit</td>
</tr>
<tr>
<td>Renesas e² studio Integrated Development Environment (IDE)</td>
</tr>
<tr>
<td>GCC ARM Embedded Compiler</td>
</tr>
<tr>
<td>Renesas QE for Capacitive Touch</td>
</tr>
<tr>
<td>Flexible Software Package (FSP)</td>
</tr>
</tbody>
</table>
5. Application Example Overview

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

A code listing of the completed application example after modifications is provided in “7. qe_touch_sample.c Listing After Modifications” for review.

![Diagram of application example overview]

6. Capacitance Touch Application Development Procedure

6.1 Project Creation

1. On your PC, start the e² studio IDE using the Windows -> Start menu or the icon on your desktop. When the dialog appears, create the Workspace wherever you like.

2. Start a new project by clicking File -> New -> Renesas C/C++ Project -> Renesas RA from the e² studio menu.

3. When the New C/C++ Project dialog box opens, select “Renesas RA CC++ Project”, then click Next.

4. In the Renesas RA C/C++ Project wizard, go to the Project Name and Location page and enter a Project Name (any name can be used). The example here uses Capacitive_Touch_Project_Example. When you have entered your project name, click Next.

5. Next, in the Device and Tools Selection dialog box, select the following:

<table>
<thead>
<tr>
<th>Item</th>
<th>RA6M2 (CTSU)</th>
<th>RA2L1 (CTSU2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FSP Version</td>
<td>3.6.0</td>
<td></td>
</tr>
<tr>
<td>Board</td>
<td>EK-RA6M2</td>
<td>RSSK-RA2L1</td>
</tr>
<tr>
<td>Device</td>
<td>R7FA6M2AF3CFB</td>
<td>R7FA2L1AB2DFP</td>
</tr>
<tr>
<td>Toolchains</td>
<td>GNU ARM Embedded 10.3.1.20210824</td>
<td></td>
</tr>
<tr>
<td>Debugger</td>
<td>J-Link ARM</td>
<td>J-Link ARM, E2 (ARM) OR E2 Lite (ARM)</td>
</tr>
</tbody>
</table>
RA Family Using QE and FSP to Develop Capacitive Touch Applications

Figure 2. Device and Toolchain (RA6M2)

6. Once complete, click Next.
7. In the **Build Artifact and RTOS Selection** dialog box, select the following:

<table>
<thead>
<tr>
<th>Item</th>
<th>RA6M2 (CTSU)</th>
<th>RA2L1 (CTSU2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build Artifact Selection</td>
<td>Executable</td>
<td></td>
</tr>
<tr>
<td>RTOS Selection</td>
<td>No RTOS</td>
<td></td>
</tr>
</tbody>
</table>

**Table 3. Build and RTOS Selection**

![Figure 4. Build and RTOS Selection](image)

8. Once complete, click **Next**.
9. In the next dialog box, **Project Template Selection**, select “Bare Metal – Minimal”, then click **Finish**.

![Project Template Selection](image)

**Figure 5. Project Template Selection**

Once complete, the RA Smart Configurator perspective will appear in the e² studio default window, ready for project configuration. This completes the new project creation process.
6.2 Using RA Configurator to Add Modules

1. Using the tabs in the lower-middle pane of the e² studio, select the BSP tab to display the Board Support Package (BSP) configuration.

![BSP Tab](image)

Figure 6. BSP Tab

2. To set the power supply, select the following from the lower-left of the e² studio screen: Properties -> Settings -> RA Common -> MCU Vcc (mV). For this example, the MCU Vcc (mV) will be set to 3300mV.

![Power Supply Voltage Setting](image)

Figure 7. Power Supply Voltage Setting
3. Select the **Clocks** tab to configure the clocks.

![Figure 8. Clocks Tab](image)

4. For this example, the following settings are used.

**Table 4. Clocks Configuration**

<table>
<thead>
<tr>
<th>Clock</th>
<th>RA6M2 (CTSU)</th>
<th>RA2L1 (CTSU2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLL Src</td>
<td>XTAL</td>
<td>(no setting)</td>
</tr>
<tr>
<td>Clock Src</td>
<td>PLL</td>
<td>HOCO</td>
</tr>
<tr>
<td>PCLKB</td>
<td>PCLKB Div /4</td>
<td>PCLKB Div /2</td>
</tr>
</tbody>
</table>

![Figure 9. Clocks Configuration (RA6M2)](image)
5. Next, select the **Pins** tab. Assign the pins in order to connect the sensor port of the MCU to the Capacitive Touch Application Board.

   Note: Depending on the selections made in Device Selection -> Board when creating the project, the sensor port may be assigned to the TS pin by default.

6. Under **Pin Selection**, expand **Peripherals**. Open **Input:CTSU** and select **CTSU0**.

7. In **Pin Configuration**, change the **Operation Mode** from “Disabled” to “Enabled”.

---

**Figure 10** Clocks Configuration (RA2L1)

**Figure 11** Pins Tab

**Figure 12** Select Peripherals (CTSU0)
8. Enable the TS pins that you will be using. Since RA2L1 (CTSU2) can control the output of non-measurement pins at once, all TS pins should be enabled. In this application example, the following TS pins are assigned for use in capacitive touch interface, etc.

Table 5. TS Pins Used in Application Example

<table>
<thead>
<tr>
<th>Item</th>
<th>RA6M2 (CTSU)</th>
<th>RA2L1 (CTSU2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TS pins used in the application</td>
<td>TSCAP pin</td>
<td>TSCAP pin</td>
</tr>
<tr>
<td>example</td>
<td>TS2 pin</td>
<td>TS0 pin</td>
</tr>
<tr>
<td></td>
<td>TS11-CFC pin</td>
<td></td>
</tr>
</tbody>
</table>

Figure 13. Operation Mode (CTSU0)

Figure 14. TS Pin Configurations (RA6M2)
9. For RA6M2 (CTSU), we recommend that the touch sensor pins that are NOT being used by the application be setup so that they are driven to ‘low-level output’ at startup. To do so, expand the **Ports** tree node in **Pin Selection**. Expand the **P2** sub-node and select “P206”. Change the **Mode** setting of the pin configuration from “Disable” to “Output” mode (Initial Low)

Note: Only one port is setup here as a usage example. Set all unused sensor ports in the same manner.
10. Select **Pin Selection -> Peripherals**, then open the **Connectivity:SCI** tree and select “SCI9.”

Note: Steps 10 and 11 should only be set when using serial communication for monitoring. Otherwise, skip to Step 12.

![Figure 17. Select Peripherals (SCI9)](image)

11. Select **Pin Configuration -> Operation Mode**, then change “Disabled” to “Asynchronous UART”. Also make sure **TXD** is set to “P203” and **RXD** is set to “P202”.

![Figure 18. Operation Mode (SCI9)](image)
12. Next, select the **Stacks** tab.

![Figure 19. Stacks Tab](image)

13. Select **New Stack -> CapTouch -> Touch (rm_touch)** to add the CTSU driver and middleware.

![Figure 20. Add CTSU Driver and Middleware](image)

14. **HAL/Common Stacks** will appear as shown in the figure below:

![Figure 21. HAL/Common Stacks (after CTSU is added)](image)
15. Next, click the **CTSU Driver on r_ctsu** module to display **Properties**.

   Note: Steps 15 to 18 should only be set when using the Data Transfer Controller (DTC). Use the DTC to transfer data between memory and registers without going through the CPU. When not using the DTC, skip to Step 19.

![CTSU Driver on r_ctsu Module](image1)

**Figure 22. CTSU Driver on r_ctsu Module**

16. Set the DTC from **Properties** at the bottom-left of the screen. Select **Settings -> Common -> Support for using DTC** and change the setting from “Disabled” to “Enabled”.

![Support for Using DTC](image2)

**Figure 23. Support for Using DTC**
17. Click the **Add DTC Driver for Transmission** module and select the **New -> Transfer (r_dtc)** to add the DTC driver for transmission.

![Figure 24. Add DTC Driver for Transmission](image)

18. In the same manner, click the **Add DTC Driver for Reception** module and select the **New -> Transfer (r_dtc)** to add the DTC driver for reception.

![Figure 25. Add DTC Driver for Reception](image)
19. Click the **TOUCH (rm_touch)** module to display **Properties**.
Note: Steps 19 to 26 should only be set when using serial communication for monitoring. Otherwise, skip to Step 27.

**Figure 26. TOUCH Driver on rm_touch Module**

20. At the bottom left of the screen, select **Properties -> Settings -> Common -> Support for QE monitoring using UART** and change “Disabled” to “Enabled”.

**Figure 27. Support for QE Monitoring using UART**
21. Click the **Add SCI UART Driver** module and select **New -> UART (r_sci_uart)** to add the UART driver.

Figure 28. **Add UART Driver**

22. Click the **g_uart_qe UART Driver on r_sci_uart** module to display **Properties**.

Figure 29. **g_uart_qe UART Driver on r_sci_uart Module**
23. Set the SCI channel. At the bottom-left of the screen, select **Properties -> Settings -> Module g_uart_qe UART Driver on r_sci_uart** -> **General -> Channel** and change the channel setting from “0” to “9”.

![Figure 30. Channel](image)

24. From **Properties** set the DTC as well. Select **Common -> DTC Support** and change “Disable” to “Enable”.

Note: Steps 24 to 26 should only be set when using the Data Transfer Controller (DTC). Use the DTC to transfer data between memory and registers without going through the CPU. When not using the DTC, skip to Step 27.

![Figure 31. DTC Support](image)
25. Click the **Add DTC Driver for Transmission** module and select **New -> Transfer (r_dtc)** to add the DTC driver for transmission.

![Figure 32. Add DTC Driver for Transmission (SC19)](image)

26. In the same manner, click the **Add DTC Driver for Reception** module and select **New -> Transfer (r_dtc)** to add the DTC driver for reception.

![Figure 33. Add DTC Driver for Reception (SC19)](image)
27. Select **New Stack -> Timers -> Timer, Low-Power (r_agt)** to add the AGT driver. 
   Note: Steps 27 to 33 should only be set when using an external trigger as the CTSU measurement start trigger. Otherwise (i.e. when using a software trigger), skip to Step 34.

![Image showing the process of adding AGT Driver]
28. Click the **g_timer0 Timer, Low-Power (r_agt)** module to display Properties.

![Figure 35. g_timer0 Timer Driver on r_agt Module](image)

29. Set the AGT. At the bottom left of the screen, select **Properties -> Settings -> Module g_timer0 Timer Driver on r_agt -> General -> Period**. Enter “100” as the **Period** setting and select milliseconds as the unit.

![Figure 36. Period / Period Unit](image)
30. Now set the AGT interrupts by selecting **Module g_timer0 Timer Driver on r_agt** -> **Interrupts** -> **Underflow Interrupt Priority** and setting the priority to anything other than “Disabled”. In the example, Priority 2 is selected, which generates an AGT0 interrupt every 100ms.

![Figure 37. Underflow Interrupt Priority](image)

31. Select **New Stack** -> **System** -> **Event Link Controller (r_elc)** to add the ELC Driver.

![Figure 38. Add ELC Driver](image)
32. Click the CTSU Driver on r_ctsu module to display Properties.

![Figure 39. CTSU Driver on r_ctsu Module](image)

33. Set the CTSU measurement start trigger. At the bottom-left of the screen, select Properties -> Settings -> Module CTSU Driver on r_ctsu -> Scan Start Trigger and change the trigger from “Software” to “AGT0 INT (AGT interrupt)”. This setting makes the CTSU start the measurement with an AGT0 interrupt.

![Figure 40. Scan Start Trigger](image)

34. At this point, all application modules necessary for capacitive touch operations have been added. The final step is to generate the application code modules necessary for the project by clicking Generate Project Content icon at the upper-right of the screen.

![Figure 41. Generate Project Content](image)
6.3 Creating the Capacitive Touch Interface

1. From the e² studio IDE, use **Renesas Views -> Renesas QE -> CapTouch Main / Sensor Tuner** to open the main perspective for configuring capacitive touch to the project.

   ![CapTouch Main / Sensor Tuner Menu](image)

   *Figure 42. CapTouch Main / Sensor Tuner Menu*

2. In the **CapTouch Main / Sensor Tuner** pane, select the project you want to configure for touch interface by using the pull-down tab and selecting “Capacitive_Touch_Project_Example” as shown below.

   ![Select a Project](image)

   *Figure 43. Select a Project*
3. Next, create a new touch interface configuration by using the lower pull-down menu (To Prepare a Configuration) and selecting Create a new configuration.

![Select a Configuration](image)

**Figure 44. Select a Configuration**

4. The Create Configuration of Touch Interfaces dialog box will open, showing the area for positioning the touch interface (default blank canvas).

![Create Configuration of Touch Interfaces Dialog Box](image)

**Figure 45. Create Configuration of Touch Interfaces Dialog Box**
5. Add a button to the touch interface area by selecting **Button** on the right side of the **Create Configuration of Touch Interfaces** dialog box, and then clicking on the blank canvas. Press the ESC key on your keyboard or select **Button** again to complete the step. The added button will remain red, as shown in the figure below, indicating it has not been assigned a touch sensor yet.

![Create Configuration of Touch Interfaces](image)

**Figure 46. Add a Button**

6. Double click “Button00” in the touch interface canvas to display the **Setup Touch Interface** dialog box. From the pull-down menu, select the MCU sensor port you want to assign to this button. When a sensor port that has been enabled by the RA Smart Configurator is correctly assigned to the button, the setting error will disappear and the button will turn green, indicating it has been set.

<table>
<thead>
<tr>
<th>Item</th>
<th>RA6M2 (CTSU)</th>
<th>RA2L1 (CTSU2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Touch sensor to be selected</td>
<td>TS02</td>
<td>TS11</td>
</tr>
</tbody>
</table>
7. Next, on the right side of the **Create Configuration of Touch Interfaces** dialog box, select **Shield Pin** and then click on the canvas to add a shield pin. Press the ESC key on your keyboard or select **Shield Pin** again to complete the step. The added shield pin will remain red, as shown in the figure below, indicating it has not been assigned a touch sensor yet.

Note: When creating a project, if “EK-RA6M2” (RA6M2 MC Group Evaluation Kit) is selected in Device Selection -> Board, the Shield Pin will be disabled and cannot be selected.
8. Double click “Shield00” in the touch interface canvas to open the **Setup Touch Interface** dialog box. From the pull-down menu, select the MCU sensor port to assign to this shield pin. When a sensor port that has been enabled by the RA Smart Configurator is correctly assign to the pin, the setting error will disappear and the button will turn green, indicating it has been set.

<table>
<thead>
<tr>
<th>Item</th>
<th>RA6M2 (CTSU)</th>
<th>RA2L1 (CTSU2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Touch sensor to be selected</td>
<td>- (Cannot be set for EK-RA6M2)</td>
<td>TS00</td>
</tr>
</tbody>
</table>

![Figure 49. Setup Touch Sensor (Shield Pin)](image-url)

9. Click **Create** in the **Create Configuration of Touch Interfaces** dialog box to complete the touch interface settings.

![Figure 49. Create Button](image-url)

10. The **CapTouch Main / Sensor Tuner** window will now display the configuration of the touch interface in the **Tuning** pane.

![Figure 50. Tuning Pane](image-url)
11. Build the project using the hammer icon in the upper left-hand side of the e² studio screen. The project should build without any errors or warnings.

Figure 51. Build Button
### 6.4 Modifying Debug Session for Capacitive Touch Tuning

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

![Debug Configuration Editing Button](image)

*Figure 52. Debug Configuration Editing Button*

2. Select the **Startup** tab.

![Startup Tab](image)

*Figure 53 Startup Tab*

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

![Runtime Option](image)

*Figure 54. Runtime Option*

4. Click **OK** to use these modified settings. This completes the project configuration and debug setup for tuning.
6.5 Tuning the Capacitive Touch Interface Using QE for Capacitive Touch Plug-in

1. Make sure the emulator is connected correctly to the board and PC.

2. To start the automatic tuning process, click the **Start Tuning** button in the **CapTouch Main / Sensor Tuner** e² studio IDE.

   ![Start Tuning](image)
   
   **Figure 55. Start Tuning**

3. At the start of the first debug session, e² studio may display a message regarding a switch to the Debug perspective. Click the **Remember my decision** check box and **YES** to continue the Debug session and the QE for Capacitive Touch automatic tuning.

   ![Confirm Perspective Setting](image)
   
   **Figure 56. Confirm Perspective Setting**
4. QE for Capacitive Touch automatic tuning will now begin. Please carefully read the Automatic Tuning Processing dialog windows as they will guide you through the tuning process. An example screen is shown below. Typically, no interaction is required during the initial tuning process steps.

![Automatic Tuning Processing Dialog (now preparing)](image)

**Figure 57. Automatic Tuning Processing Dialog (now preparing)**

5. After several automated steps, a dialog box with information similar to what is shown below will appear. This is the touch sensitivity measurement step of the tuning process. As the first interactive step of the tuning process, press “Button00” (sensor) using normal touch pressure as indicated in the dialog box (RA6M2: TS02, RA2L1: TS11-CFC). When pressing the sensor, the bar graph will increase to the right and the touch counts will increase numerically. While continuing to apply pressure to the sensor, press any key on the PC keyboard to accept the measurement.

![Automatic Tuning Processing Dialog (now measuring)](image)

**Figure 58. Automatic Tuning Processing Dialog (now measuring)**
6. Once sensitivity measurement for the button is complete, a dialog like the following will appear allowing threshold confirmation. This is the detection threshold that is used by the middleware to determine if a touch event has occurred.

![Automatic Tuning Processing (tuning complete)](image)

Figure 59  Automatic Tuning Processing (tuning complete)

7. Click the **Continue the Tuning Process** button in the dialog box shown. This will exit the tuning process and disconnect from the Debug session on the target board. This should return you to the default **Cap Touch Main / Sensor Tuner** screen in the e² studio IDE.

![Continue the Tuning Process Button](image)

Figure 60.  Continue the Tuning Process Button

8. In this example, the final step is to output the tuning parameter files. Click the **Output Parameter Files** button.

   Note: Carry out this step if you are NOT using an external trigger (i.e. when you are using a software trigger) to start the CTSU measurement.

![Output Parameter Files](image)

Figure 61.  Output Parameter Files
9. When using an external trigger to start the CTSU measurement, check the **Use an external trigger** box, and then click **OK**.

   Note: Carry out this step if you are using an external trigger to start the CTSU measurement process.

![Parameter File Output Settings](image)

**Figure 62. Parameter File Output Settings**

10. In the **Project Explorer** window, confirm that `qe_touch_config.c`, `qe_touch_config.h` and `qe_touch_define.h` files have been added. These contain the tuning information necessary for enabling touch detection with drivers.

![Parameter Files](image)

**Figure 63. Parameter Files**

11. Build the project using the hammer icon in the upper left-hand side of the e² studio IDE. Confirm that the build results shown in the **Console** window do not show any errors.

   Note: A build error will be generated if monitoring via serial communication is enabled in Flexible Software Package (FSP) V3.0.0. See [https://github.com/renesas/fsp/issues/78](https://github.com/renesas/fsp/issues/78) for more details.
6.6 Adding rm_touch middleware API Calls to Application Example

1. To implement a program (code) that will scan and report the state of the touch sensor, click the **Show Sample** button in the **CapTouch Main / Sensor Tuner e² studio IDE**.

![Show Sample Code](image)

**Figure 64. Show Sample Code**

2. The **Show Sample Code** window will open which shows the sample code in text. Click the **Output to a File** button to output the sample code. Finally, click **OK** to close the window.

![Show Sample Code Window](image)

**Figure 65. Show Sample Code Window**
3. You can confirm that the new `qe_touch_sample.c` file has been created in **Project Explorer**.

![Project Explorer with `qe_touch_sample.c` highlighted](image)

**Figure 66. qe_touch_sample.c**

4. In **Project Explorer**, select `ra_gen` and double click on main.c to open the file. Make sure `hal_entry` is selected in the `main()` function and right-click on **Open Declaration**. This will open the `hal_entry()` function declaration.

![Project Explorer with `hal_entry` highlighted](image)

**Figure 67. Open Declaration (hal_entry function)**
5. In the `hal_entry()` function, add the code to call the `qe_touch_main()` function and the prototype declaration as shown in the image below.

![Figure 68. Function Call (qe_touch_main function)](image1)

6. In the `hal_entry()` function, select the `qe_touch_main()` function, then click on Open Declaration to open the `qe_touch_main()` declaration.

Note: Carry out Steps 6 and 7 only if you are using an external trigger to start the CTSU measurement process. If you are NOT using an external trigger (i.e. when you are using a software trigger), skip to Step 8.

![Figure 69. Open Declaration (qe_touch_main function)](image2)
7. When using an external trigger to start the CTSU measurement process, enable the code after commenting out the API that controls the AGT driver of the `qe_touch_main()` function.

8. This completes all the necessary code modifications required for this application example. Build the code to confirm that there are no errors or warnings.
6.7 Monitoring Touch Performance Using e²studio Expressions Window and QE for Capacitive Touch

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

![Figure 71. Open Declaration (hal_entry function)](image1)

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

![Figure 72. Open Declaration (qe_touch_main function)](image2)
5. Scroll down in the `qe_touch_main.c` file to the `RM_TOUCH_DataGet()` function in the `while (true)` loop.
With the argument “button_status” selected, right click and select Add Watch Expression... This opens the Add Watch Expression dialog box, where you can add the variable “button_status” to the expressions window.

![Figure 73. Add Watch Expression... Menu / Dialog](image)

6. Right click in the Expressions window to select Enable Real-time Refresh and enable real-time refresh in the added variable.

![Figure 74. Enable Real-time Refresh Menu](image)
7. Click the **Resume** (green arrow) button, located near the middle of the e² studio tool bar, to continue code execution.

8. Press the sensor on the board (RA6M2: TS02, RA2L1: TS11-CFC) which was configured as “Button00” in Section 6.3 **Creating Capacitive Touch Interface**. When pressed, the “button_status” value in the Expressions window will become ‘1’, confirming the touch with a binary indication.

![Figure 75. Touch Status Confirmation in Expressions Window](image)

9. Click the **Show Views** button in the Monitoring pane of CapTouch Main / Sensor Tuner to startup the CapTouch Board Monitor pane.

![Figure 76. Start Monitoring](image)
10. The **CapTouch Board Monitor** pane should appear like the image below.

**Figure 77.** CapTouch Board Monitor Window (RA6M2)

**Figure 78.** CapTouch Board Monitor Window (RA2L1)
11. Click the **Enable Monitoring** button. The dialog text will change from “Monitoring: Disabled” to “Monitoring: Enabled.”

![Enable Monitoring Function](image)

**Figure 79. Enable Monitoring Function**

12. Touch “**Button00**” (RA6M2: TS02, RA2L1: TS11-CFC) on the Capacitive Touch Application board. The **CapTouch Board Monitor** will show a touch with a finger image on the button like the below image.

![Touch Status Confirmation in CapTouch Board Monitor Window](image)

**Figure 80. Touch Status Confirmation in CapTouch Board Monitor Window**
13. To see a graphical representation of the ‘touch counts’ from the board, use the **CapTouch Status Chart**

![Figure 81. CapTouch Status Chart Window](image)

14. Using the **Touch I/F** pull-down menu, select “**Button00 @ config01**”.

![Figure 82. Select Touch I/F](image)
15. The graph will begin to display running values. Touch “Button00” (RA6M2: TS02, RA2L1: TS11-CFC) on the board to view the touch counts shown as a step change on the running graph. The green line is the touch threshold, which the rm_touch middleware uses to determine whether a button is actuated/touched. The red blocks at the bottom of the graph are a visual indication to the user that the touch counts have exceeded the threshold and a touch is detected.

Figure 83. Confirm Touch Status in CapTouch Status Chart Window
Note: Steps 16 to 19 must be set when displaying and measuring standard deviation.

16. Next, measure the standard deviation. Click the **Start Data Collection** button. While collecting data in the **touch-off state**, don’t touch the electrode. The green bar is the data collection rate. When the green bar goes all the way to the right, the data collection in the touch-off state is complete.

![Figure 84. Start Data Collection Button (touch-off state)](image1)

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

![Figure 85. Stop Data Collection Button (touch-off state)](image2)
18. Next, touch the electrode in order to collect data in the touch-on state. Click the **Start Data Collection** button while touching the electrode.

![Figure 86. Start Data Collection Button (touch-on state)](image)

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

![Figure 87. Stop Data Collection Button (touch-on state)](image)
6.8 Monitoring Touch Performance with QE for Capacitive Touch Using Serial Communication

1. When monitoring is in operation, click the **Enable Monitoring** button. The dialog text will change from “Monitoring: Enabled” to “Monitoring: Disabled”.

2. To finish the debug session, click the **Stop** icon at the top-right of the e2 studio window.

3. Disconnect the emulator from the PC and target board. Confirm that the USB cable is correctly connected to the target board and PC.

   Note: Although operations can be carried out with the emulator connected, the emulator is removed in this step to confirm successful monitoring without the emulator.

4. Reset the board by pressing the **RESET** switch.
5. Open the **CapTouch Main / Sensor Tuner** pane and make sure the following folder/file are selected:
   - **To Select a Project**: “Capacitive_Touch_Project_Example”
   - **To Prepare a Configuration**: “Capacitive_Touch_Project_Example.tifcfg”

![Figure 89. Select Project / Configuration](image)

6. In the **CapTouch Main / Sensor Tuner** pane under **Monitoring**, select **Connect** to enable monitoring using serial communication.

![Figure 90. Connect Button](image)
7. “Connected to COMn” should appear at the bottom of Console window. Confirm the message to make sure the connection is successful.

Note: The COM port number for connection varies depending on the PC environment.

![Console Window Output](image)

Figure 91. Console Window Output

8. The rest of the process is the same as Step 9 on in Section 6.7 Monitoring Touch Performance Using e²studio Expressions Window and QE for Capacitive Touch.
7. **qe_touch_sample.c Listing After Modifications**

7.1 **When Using a Software Trigger to Start the CTSU Measurement Process**

```c
/**
 * FILE : qe_sample_main.c
 * DATE : 2020-09-10
 * DESCRIPTION : Main Program
 *
 * NOTE: THIS IS A TYPICAL EXAMPLE.
 *
 */
#include "qe_touch_config.h"
#define TOUCH_SCAN_INTERVAL_EXAMPLE (20)    /* milliseconds */

void qe_touch_main(void);

uint64_t button_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;

    /* 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. */
    R_BSP_SoftwareDelay(TOUCH_SCAN_INTERVAL_EXAMPLE, BSP_DELAY_UNITS_MILLISECONDS);
}
}
7.2 When Using an External Trigger to Start the CTSU Measurement Process

```c
/* ***********************************************************************
 * FILE : qe_sample_main.c
 * DATE : 2020-09-10
 * DESCRIPTION : Main Program
 * *
 * NOTE:THIS IS A TYPICAL EXAMPLE.
 * *
 * **************************************************************************/

#include "qe_touch_config.h"
#define TOUCH_SCAN_INTERVAL_EXAMPLE (20)    /* milliseconds */

void qe_touch_main(void);

uint64_t button_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;

    /* Initializes the software and sets the links defined in the control structure. */
    R_ELC_Open(g_elc.p_ctrl, g_elc.p_cfg);
    /* Globally enable event linking in the ELC. */
    R_ELC_Enable(g_elc.p_ctrl);

    /* 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) {} 
    }

    /* Open AGT driver */
    R_AGT_Open(g_timer0.p_ctrl, g_timer0.p_cfg);
    /* Start AGT. */
    R_AGT_Start(g_timer0.p_ctrl);
}
```
/* 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. */
    R_BSP_SoftwareDelay(TOUCH_SCAN_INTERVAL_EXAMPLE, BSP_DELAY_UNITS_MILLISECONDS);
}
Website and Support

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

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

Inquiries
http://www.renesas.com/contact/
# Revision History

<table>
<thead>
<tr>
<th>Rev.</th>
<th>Date</th>
<th>Description</th>
<th>Page</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>2020.2.28</td>
<td>-</td>
<td>-</td>
<td>First edition issued</td>
</tr>
<tr>
<td>1.10</td>
<td>2020.4.6</td>
<td>2 Overview of a simplified example application</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 Change file generation location</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 Addition of power supply voltage setting method</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>5 Change clock frequency setting method</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>8,9 Added DTC setup instructions</td>
<td>8,9</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>13 Change in debugging method</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>14 Omission of tuning process</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>18 Change the method of adding code to the application example</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>19,20 Changed the way the monitoring view opens.</td>
<td>19,20</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>23 Addition of standard deviation measurement method</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>25,26 Sample Code Update</td>
<td>25,26</td>
<td></td>
</tr>
<tr>
<td>2.00</td>
<td>2022.6.7</td>
<td>Entire document</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Added RA2L1 (CTSU2) environment and process steps</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated IDE version and name</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Added numbers to figures and table</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Changed black arrows in figures to red outlines</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 Chapter 2/4: Removed Renesas Code Generator</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Chapter 3: Changed title</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 Chapter 5: Changed implementation description to figure</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Chapter 6: Changed title, moved remaining development steps from chapters to</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>sections</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>5 Added Fig. 3 Device and Toolchain (RA2L1)</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>6 Section 6.1: Added Step 7 for building an artifact and RTOS selection</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>7 Added Fig. 5 Project Template Selection</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>10 Added Fig. 10 Clocks Configuration (RA2L1)</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>12 Added Fig. 15 TS Pin Configurations (RA2L1)</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>13 Section 6.2: Added Steps 10 and 11 for when using serial communication</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>for monitoring</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>17~21 Section 6.2: Added Steps 19 to 26 for when using serial</td>
<td>17~21</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>communication for monitoring</td>
<td>17~21</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>21~24 Section 6.2: Added Steps 27 to 33 for when using an external</td>
<td>21~24</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>trigger to start CTSU measurement</td>
<td>21~24</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>28 Section 6.3: Added Steps 7 and 8 for adding a shield pin.</td>
<td>28</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>31 Section 6.4: Added Step 1 for confirming emulator connection.</td>
<td>31</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>35 Section 6.5: Added steps for when using an external trigger to start</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>CTSU measurement</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Section 6.5: Added note regarding build error to Step 11</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>38, 39 Section 6.6: Added Steps 6 and 7 for when using a software</td>
<td>38, 39</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>trigger to start the CTSU measurement</td>
<td>38, 39</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>41 Added Fig. 75 Enable Real-time Refresh Menu</td>
<td>41</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>43 Added Fig. 79 CapTouch Board Monitor RA, RL78, Synergy (QE) Window (RA2L1)</td>
<td>43</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>49~51 Added Section 6.8 Monitoring Touch Performance with QE for Capacitive</td>
<td>49~51</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Touch [RA, RL78, Synergy] Using Serial Communication</td>
<td>49~51</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>52, 53 Section 7.1: Updated source Code</td>
<td>52, 53</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>54, 55 Added Section 7.2 When Using an External Trigger to Start the</td>
<td>54, 55</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CTSU Measurement Process</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------------</td>
<td>--------------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>56</td>
<td>Changed URL</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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 VIL (Max.) and VIH (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 VIL (Max.) and VIH (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 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 system, in 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 product 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 PRODUCTS DO 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 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 public safety 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

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/

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

© 2021 Renesas Electronics Corporation. All rights reserved.