

RX Family

APPLICATION NOTE

R01AN4516EU0300 Rev.3.00 Sep.26.2023

Using QE and FIT to Develop Capacitive Touch Applications

Introduction

This document will demonstrate the needed steps to create an application example that integrates capacitive touch sensing using Renesas RX microcontrollers.

Target Device

RX671, RX140, RX231, RX230, RX130, RX113 with Capacitive Touch Sensing Unit (CTSU)



RX Family

Contents

1.	Application Example Overview3
2.	Related Documents3
3.	High Level Integration Steps3
4. 4.	Required Development Tools and Software Components 4 1 Application Example Overview 4
5.	Project Creation5
6.	Using Smart Configurator to Add Modules6
7.	[Additional function] Setting the serial communication monitor using UART (1/2) 10
8.	Creating the Capacitive Touch Interface12
9.	Modifying the e ² studio Project Debug Session for Capacitive Touch Tuning16
10.	Tuning the Capacitive Touch Interface Using QE for Capacitive Touch Plug-in 18
11.	Adding rm_touch_qe FIT Module Calls to Application Example21
12.	Monitoring Touch Performance using e ² studio Expressions Window and QE for Capacitive Touch
13.	[Additional function] Setting the serial communication monitor using UART (2/2) 31
14.	qe_touch_sample.c Listing35
15.	[Additional function] Setting The Automatic Judgement and MEC functions37
We	bsite and Support41
Rev	vision History42



1. Application Example Overview

This document will demonstrate how to implement capacitive touch sensing using Renesas RX microcontrollers. Using these steps, the user will be shown the process of connecting the RX140 MCU board, using Smart Configurator for project creation, and QE for Capacitive Touch to create, tune and monitor a Capacitive Touch project.

2. Related Documents

- Firmware Integration Technology User's Manual (R01AN1833EU)
- Board Support Package Firmware Integration Technology Module (R01AN1685EU)
- Adding Firmware Integration Technology Modules to Projects (R01AN1723EU)
- QE CTSU Module Using Firmware Integration Technology (R01AN4469EU0100)
- QE Touch Module Using Firmware Integration Technology (R01AN4470EU0100)

This application example is intended to give a user a short introduction to creating a working example. A thorough review of all the applicable documentation for the e² studio IDE/Smart Configurator, Firmware Integration Technology (FIT) drivers/middleware, Renesas Code Generator, 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 needed to integrate touch detection into this project. These same steps should apply to any typical user development application.

- Create the initial project using the e² studio project creation wizard
- Use Smart Configurator to add the needed modules to the created e² studio project
- Use the QE for Capacitive Touch e² studio plug-in to create the capacitive touch interface
- Use the QE for Capacitive Touch e² studio plug-in to tune the application project
- Add the needed FIT application code function calls to the user project to enable capacitive touch operations in the application project
- Monitor the application project using QE for Capacitive Touch e² studio plug-in to demonstrate capacitive touch detection



4. Required Development Tools and Software Components

The project used for this example is a simplistic example application that performs touch detection. The project uses API calls from the FIT modules described above in the c.

The project utilizes the following development environment:

- RX140 Capacitive Touch Evaluation System (RTK0EG0039S01001BJ)
- Renesas e² studio IDE, V2023-07 or later
- Renesas CC-RX compiler (v3.05.00 or later) installed
- Renesas E2 emulator Lite (RTE0T0002LKCE00000R)
- Renesas QE for Capacitive Touch V3.3.0 or later
- Renesas Firmware Integration Technology (FIT) and Renesas Code Generator
- QE CTSU FIT module (r_ctsu_qe) v2.20
- QE Touch FIT module (rm_touch_qe) v2.20
- Renesas Smart Configurator for RX
- Renesas Code Generator

4.1 Application Example Overview

In the main loop of the application example created:

- The global flag used to determine if the rm_touch_qe module is ready to be processed is checked
 - If the flag is TRUE (ready to process)
 - The global flag is reset to FALSE
 - A call to the rm_touch_qe FIT module processes data from the previous completed scan, updates needed data, then starts the next touch scan process
 - A call to the rm_touch_qe FIT module populates a user created global variable with the binary determination of a touch on the sensor board

A code listing of the completed application example is in "14. qe_touch_sample.c Listing" for review.



5. Project Creation

- 1. On the PC start the IDE using the Windows->Start menu or the icon on the desktop. When the dialog appears, create the Workspace at: C:\Workspace\Capacitive_Touch_Project_Example.
- 2. Start a new project by clicking File->New->C/C++ Project
- 3. At the dialog box that opens, select Renesas RX and highlight with a single-click **Renesas CC-RX C/C++ Executable Project**, then click Next
- 4. In the next dialog box, enter a Project name-this can be any name desired. The example here uses **Capacitive_Touch_Project_Example**. When complete, click Next
- 5. In the next dialog box, ensure the following is selected:
 - Language: C
 - Toolchain: Renesas CC-RX
 - Toolchain Version: v3.05.00
 - Endian: Little
 - Create Hardware Debug Configuration is CHECKED
 - E2 Lite (RX) is selected in the pull-down
 - Target Device: R5F51406AxFN

Note: Use the '...' to select the proper device using the menu that appears

							×
New Renesas CC-RX Execut Select toolchain, device & deb	-						\$
Toolchain Settings Language: C C<th></th><th>Manage Toolch</th><th> ✓ ✓ ✓ ✓ ✓ ✓ </th><th></th><th></th><th></th><th></th>		Manage Toolch	 ✓ ✓ ✓ ✓ ✓ ✓ 				
Device Settings				Configurations			
Target Board: Custom			~	Create Hardwar	e Debug C	onfigura	tion
Target Device: R5F51406AxF		oad additional boo		E2 Lite (RX)	onfiguratio	on	~
Endian: Little			~	RX Simulator			\sim
Project Type: Default			\sim	Create Release C	Configurati	ion	
?	[< Back	Ne	xt > Finish		Cance	4



earch Device						
Device	RAM	ROM	Pin	RTOS	Smart C	Peripher
✓ RX100	NAIVI	NOIVI	PIN	RIUS	Sindri C	Fenbriet
> RX100						
> RX110						
> RX113						
> RX130						
> RX13T						
✓ RX140						
> RX140 - 32pin						
> RX140 - 48pin						
> RX140 - 64pin						
✓ RX140 - 80pin						
R5F51405AxFN	32 KB	128 KB	80		~	×
R5F51405BxFN	32 KB	128 KB	80		~	×
R5F51406AxFN	64 KB	256 KB	80		~	×
R5F51406BxFN	64 KB	256 KB	80		~	×
> RX200						
> RX600						
> RX700						

- 6. Once complete, click Next.
- 7. In the next dialog box that appears, check the box for "Use Smart Configurator", then click Finish

Once complete, a default window will open for the IDE with the Smart Configurator perspective open and ready for project configuration. This completes the project creation.

6. Using Smart Configurator to Add Modules

1. Using the tabs in the lower-middle pane of the IDE, select the Clocks tab to display the clock tree for the RX140 MCU.

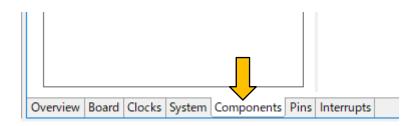
	OCO clo	ck						
	Roard	Clocks	Suctom	Components	Dinc	Inter	nunte	
Overview	DOard		system	components	PIIIS	inten	rupts	

2. The setting of the clock configuration that is used for this application note is shown below.

Clocks configuration	Generate Code Generate Report
VCC: 3.3 (V) (Actual value: 3.3)	
Cacillation source: Prequency: S (MHz)	
Wait time: 8192	FlashIF clock (FCLK) 48 (MHz) System clock (ICLK) 48 (MHz)
Sub-clock Frequency: 32.768 (kHz)	Peripheral module clock (PCI-
Oscillator drive capacity: Standard CL	24 (MHz) Peripheral module clock (PCI 48 (MHz)
W HOCO clock Frequency: 48 (MHz)	Low power timer clock (LPTC
Enable HOCO oscillation after reset	- (kHz)



3. Next, move to the Components tab by selecting it at the bottom of the pane



4. Modules now need to be added to the project for application use. Add a module by clicking on the '+' sign in the Components tab

Software comp	onent configuration
Components	≧ ⊿ 👌 🗆 🕀 🖨 ▼
type filter text ✓ ➢ Startup ✓ ➢ Generic ✓ r_bsp	

5. A new dialog window will open showing all the available modules that can be added into the project. It will appear similar to the below picture.

ategory	All				\sim
unction	All				~
ilter					_
_		Short Name	-		^
Compor		Short Name	Туре	Version	î
🖶 8-Bit			Code Generator	1.10.0	
	M compress/de-compress codec s		Firmware Integr	3.04	
	YZ012 Module control functions fo		Firmware Integr	1.01	
	d Support Packages.	r_bsp	Firmware Integr	7.41	
Buse:			Code Generator	1.11.0	
	based circular buffer library.	r_byteq	Firmware Integr	2.10	
	lar Module control functions for Re	r_cellular	Firmware Integr	1.04	
	Frequency Accuracy Measurement		Code Generator	1.11.0	
	Synchronous Control Module for	r_eeprom_spi	Firmware Integr	3.10	
🗄 Clock	Synchronous Control Module for	r_flash_spi	Firmware Integr	3.20	
🗄 CMT		r_cmt_rx	Firmware Integr	5.50	
🖶 Com	parator		Code Generator	1.9.0	
🖶 Com	pare Match Timer		Code Generator	2.3.0	
	plementary PWM Mode Timer		Code Generator	1.11.0	
🖶 Conti	nuous Scan Mode S12AD		Code Generator	1.13.0	
🗄 Conti	rol Low Power States.	r_lpc_rx	Firmware Integr	2.20	
CRC (Calculator		Code Generator	1.11.0	¥
Show	only latest version				
- Hide it	ems that have duplicated functionality	,			
escriptic		'			
· · ·	ware component generates two units	((TMD)	and a late	~
	prise two 8-bit counter channels, tota		on-chip o-bit timer (TWR)	module	
	•				
					\vee
ownload	the latest FIT drivers and middleware				
onfigure	general settings				



6. Scroll down the list until rm_touch_qe is shown. Single-click this entry and click Finish in the lower part of that open dialog box.

	HTCP/IP protocol stack [M3S-T4-Tiny] for	r_t4_rx	Firmware Integr	2.10				
	🖶 Touch QE API	rm_touch_qe	Firmware Integr	2.20				
/	🖶 Unsigned 32-bit circular buffer library.	r_longq	Firmware Integr	2.00				
	H Voltage Detection Circuit		Code Generator	1.11.0	\mathbf{v}			
 Show only latest version Hide items that have duplicated functionality Description 								
Dependency : r_bsp version(s) 6.10 Dependency : r_ctsu_qe version(s) 2.20 This module allows for touch detection of buttons, sliders, and wheels using CTSU sensor data.								

7. The module will appear in the project tree as shown in the picture below:

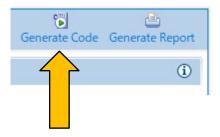
Components	2	i ∣a _z	+	≩ ▼
			t	•
🗸 🗁 Startup				^
🗸 🗁 Generic				
💣 r_bsp				
🗸 🗁 Drivers				
🗸 🦢 Capacitive	Touch			
💣 r_ctsu_	qe			
🗸 🧁 Middleware				
🗸 🦢 Capacitive	Touch			
💣 rm_tou	ch_qe			

- 8. Next, assignment between the sensor pads on the Buttons/Wheels/Slider board needs to be made to connect them to the MCU pins. On the Buttons/Wheels/Slider board, the specific MCU sensor pins are labeled for easy identification on the board silkscreen. To do this assignment, select the r_ctsu_qe module and in the Configure pane a list of pins associated with the module will appear. This application example will only assign the two buttons on the upper right-hand side of the Buttons/Wheels/Slider board (TS8 and TS9). Click the following pins in the Configure pane so they can be used in the project:
 - TSCAP Pin
 - TS8 Pin
 - TS9 Pin

Components 🚵 🛃 🎘 🖃 🕀 🐳 🔻	Configure	0
10 T	Property	Value ^
type filter text	v 🏶 Configurations	
V 🗁 Startup	# Parameter check	Use system default
V 🦻 Generic	# Data transfer of INTCTSUWR and INTCTSURD	Interrupt handler
r_bsp	# Select automatic judgement code	Disable
V > Drivers	# Interrupt level for INTCTSUWR	Level 2
V 🗁 Capacitive Touch	# Interrupt level for INTCTSURD	Level 2
🔐 r_ctsu_qe	# Interrupt level for INTCTSUFN	Level 2
V > Middleware	✓ I Resources	
✓ → Capacitive Touch	V 🖪 CTSU	
rm_touch_qe	🛰 TSCAP Pin	Vsed
	🛰 TSO Pin	Used
	🛰 TS1 Pin	Used
	🛰 TS2 Pin	Used
	🛰 TS3 Pin	Used
	🛰 TS4 Pin	Used
	🛰 TS5 Pin	Used
	🛰 TS6 Pin	Used
	🛰 TS7 Pin	Used
	🛰 TS8 Pin	Vsed
	🛰 TS9 Pin	Used
	🛰 TS10 Pin	Used
	🛰 TS11 Pin	🔲 Used 🗸 🗸



9. At this point, all needed application modules for capacitive touch operations have been added. The final step is to generate the needed application code modules for the project. Do this by clicking the Generate Code icon in the upper-right of the Smart Configure pane as shown in the picture below





7. [Additional function] Setting the serial communication monitor using UART (1/2)

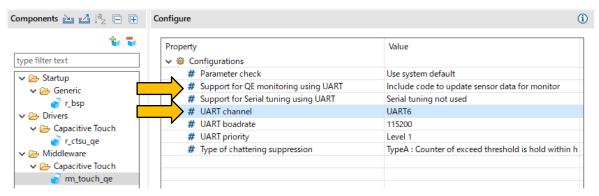
Note: Monitoring touch performance for touch applications can be confirmed by communication via the emulator.

On the other hand, monitoring touch performance can also be achieved via serial communication. Therefore, if you want to monitor smoothly, please add the monitoring function via serial communication.

Chapters 7 and 13 (including this chapter) shown below describe setting the serial communication

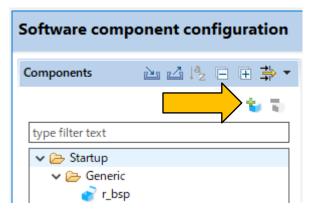
monitor using UART.

- "7. [Additional function] Setting the serial communication monitor using UART (1/2)"
- "13. [Additional function] Setting the serial communication monitor using UART (2/2)"
- 1. In the Components tab, select the rm_touch module, set **Support QE monitor using UART** to **Include code to update sensor date for monitor** and **UART channel** to **UARTA6** as shown below



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

2. Add a module by clicking on the '+' sign in the Components tab.



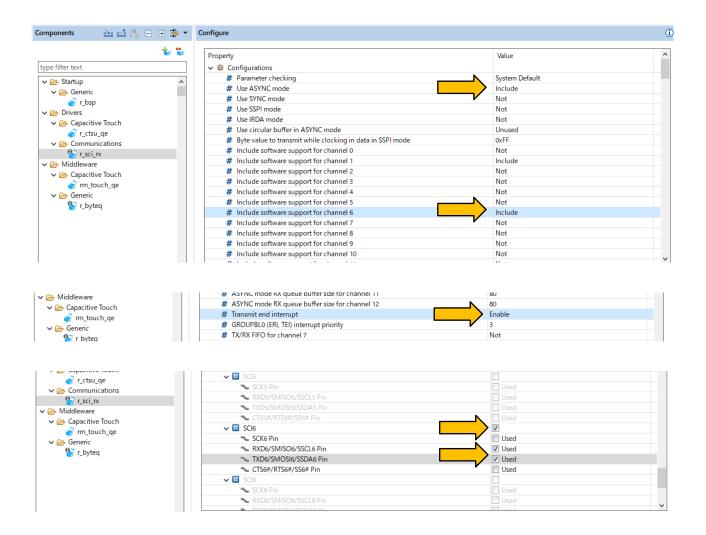


3. Add the code generation component **r_sci_rx** in the same way as the previous **rm_touch_qe**. Single-click this entry and click Finish in the lower part of that open dialog box.

N	BSPI Driver	r_rspi_rx	Firmware Integr	3.10	
	🖶 SCI Driver	r_sci_rx	Firmware Integr	4.80	
	H SCI/SCIF Asynchronous Mode		Code Generator	1.12.0	~
	Show only latest version Hide items that have duplicated functionality	y			

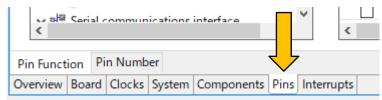
4. Next, select **r_sci_rx** and display the ports associated with this module in the configuration panel and configure as follows.

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





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



6. Assign RXD6 function to PD1 and TXD1 function to PD0.

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

Hardware Resource	i 🗆 🞝 📇	Pin Function	n				🤣 🔛 🔛 🔛
Type filter text		type filter	text (* = any str	ing, ? = any character)			All
 TMR2 TMR3 	^	Enabled	Function CTS6#	Assignment Not assigned 	Pin Number Not assigned	Direction None	Remarks
 Serial communications interface SCI1 			RTS6# RXD6	 Not assigned PD1/MTIOC4B/RXD6/SMISO6/SSCL6/IRQ1/AN025 	 Not assigned 65 		
SCI5 SCI6			SCK6 SMISO6	 Not assigned Not assigned 	 Not assigned Not assigned 	None None	
Cl8			SMOSI6 SS6#	 Not assigned Not assigned 	 Not assigned Not assigned 	None	
 I2C bus interface 			SSCL6 SSDA6	 Not assigned Not assigned 	 Not assigned Not assigned 	None	
 ♥ RIIC0 ♥ #[™] Serial peripheral interface 			TXD6	PD0/TXD6/SMOSI6/SSDA6/IRQ0/AN024	/ 66	0	

7. Generate the needed application code modules for the project. Do this by clicking the Generate Code icon in the upper-right of the Smart Configurator pane as shown in the picture below.



- 8. Creating the Capacitive Touch Interface
 - 1. From the e² studio IDE, use Renesas Views->Renesas QE->CapTouch Workflow (QE) to open the main perspecitve for configuring capacitive touch to the project

Project	Ren	esas Views	Run	Renesas /	AI N	Vindo	ow Help		
		C/C++			>	1			
		Code Gene	erator		>	F		- 8	h
		Debug			>				
		Partner OS	5		>				
		Renesas A	I.		>				
		Renesas Q	E		>	\mathfrak{S}	CapTouch Gesture Monitor (QE)		
		Smart Con	figura	tor	>	\mathfrak{D}	CapTouch Board Monitor (QE)		
		Solution Te	oolkit		>	\mathfrak{S}	CapTouch Pad Monitor (QE)		
	2	Renesas So	oftwar	e Installer		\mathfrak{D}	CapTouch Multi Status Chart (QE)		
	_					\mathfrak{L}	CapTouch Parameters (QE)		
						\mathfrak{D}	CapTouch Status Chart (QE)		
						\mathfrak{S}	CapTouch Tuning Result (QE)		
						\mathfrak{D}	CapTouch Workflow (QE)	<	
						₩.	Measuring Current Consumption (QE)		



2. In the CapTouch Workflow (QE) pane, select the project to configure the touch interface for by using the pulldown tab and selecting the **Capacitive_Touch_Project_Example** project as shown below.

CapTouch Workflow (QE) ×				🔊 i 🗖 🗖
Preparation	Tuning	Coding	Monitoring	
1.Preparation -	$\hat{}$	Select a I	Project	^
Select a Project				
Prepare a Configuration	Prepare a proje	ect that uses the touch i	nterraces.	
2.Tuning Touch Sensors 🔹	Select the targe	et project.		
Start Tuning (Serial)				<u> </u>
Output Parameter Files				<u> </u>
3.Coding -	Select all e-	Capacitive_Touch_F		
Implement Program		· · · · · · · · · · · · · · · · · · ·	roject and select the Sm	
4.Monitoring		0	tool. After creating the the settings listed below	1 2 1
Start Monitoring (Emulator)		-	-	
Start Monitoring (Serial)	 Set up the 	PCLKB, PCLKL or FCLK	Colock.	
	• Add the ca middlewar	apacitive touch sensing re.	unit (CTSU) driver &	•

3. Next, select [Prepare a Configuration] in the menu on the left-hand side of "CapTouch Workflow (QE)" to display the items for setting. Select [Create a new configuration] from the pull-down menu to generate a new configuration for a touch interface.

Preparation	Tuning	Coding	Monitoring	
1.Preparation	 → 	To Prepare a Conf	iguration	Â
⊘ Select a Project	Select or croate	e a touch interface configur	ation	
Prepare a Configuration	Select of create	e a touch interface configur		
2.Tuning Touch Sensors	- L			
Start Tuning (Emulator)		Create a new config	uration	
Start Tuning (Serial)				

4. A new menu window will open with shows the default blank canvas for creating the touch interface.

G Create Configuration of Touch Inte	rfaces		×
File Name of Touch I/F:	Capacitive_Touch_Project_Example	Setup Configuration	Import / Re-edit
Description:			
			Touch I/F 🔶
			Capacitance Type
			Self Capacitance 🗸 🗸
			Button
			Slider (horizontal)
			Slider (vertical)
			Wheel
			Key pad
			3D Gesture (AI)
			Touch pad
			Shield Pin
			TC Pin
			Capacitance Sensor
			Current Sensor
			Diagnosis Pin
Setting			Remove Touch I/F
Setup Touch I/F Setu	p Resistance Value Clear Assigned TSx		Configuratoins (Methods) 🛛 🗧
			Create Cancel Help



5. Add 2 buttons to the canvas by selecting the **Button** menu item from the right-hand side and adding two (2) buttons to the canvas. Press the **ESC** key to exit once two buttons are added. The canvas will look similar to below.

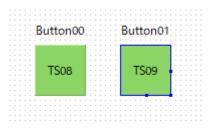
Name of Touch I/F:	Capacitive_Touch_Project_Example	Setup Configuration	Import / Re-edit
scription:			
			Touch I/F
			Capacitance Type
			Self Capacitance
			Jen capacitance
			Button
			Slider (horizontal)
			Slider (vertical)
	Button00 Button01		Wheel
			Key pad
			3D Gesture (Al)
			Touch pad
			Shield Pin
			TC Pin
			Capacitance Sensor
			Current Sensor
			Diagnosis Pin
tting			Remove Touch I/F
Setup Touch I/F	Setup Resistance Value Clear Assigned TSx		Configuratoins (Methods)
There are some problems with	a setting		
mere are some problems with	, second,		

6. To make this connection, double-click on **Button00** and a dialog box will appear. In this case, using the pulldown, select **TS08** as the MCU sensor to assign to this button.

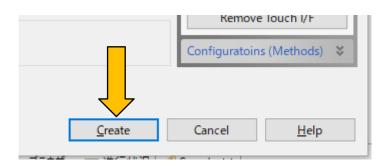
	Setup Touch Inter	face	×
Button00 Bi	Button(self)		
	Name	Button00	
	Touch Sensor	Resista	nce[ohm]
	TS08	∀ 560	~
	TS00	^	
	TS01	cel	Help
	TS02		
	TS03 TS04		
	TS05		
	TS06 TS07		
· · · · · · · · · · · · · · · · · · ·	TS08		
	TS09		



7. Perform the same operation as the previous step for **Button01** and assign it to **TS09**. The canvas should look similar to below. Note also, the indication of a configuration error will go away once all assignment are made properly and correspond to the enabled channels in the Smart Configurator.



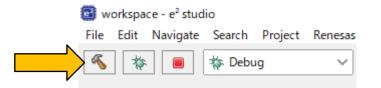
8. Click **Create** in the dialog box. This will setup the Touch Interface.



 Select [Renesas Views] – [Renesas QE] – [CapTouch Tuning Result (QE)] from the menu of the e² studio and open the [CapTouch Tuning Result (QE)] view to display the configuration of the touch interface in the [Tuning] panel.

CapTouch Tuning Result (QE) X							80	- [
ning Ge	esture								
ouch I/F (Configuration:	Capacitive_1	Touch_Project_E	kample					
Method	Kind	Name	Touch Sensor	Parasitic Capacitance[pF]	Sensor Drive Pulse Frequency[MHz]	Threshold	Scan Time[ms]	Overflow	
Method			Touch Sensor TS08	Parasitic Capacitance[pF] -	Sensor Drive Pulse Frequency[MHz]	Threshold	Scan Time[ms]	Overflow None	
	Button(self)		TS08	Parasitic Capacitance[pF] - -			Scan Time[ms] - -		

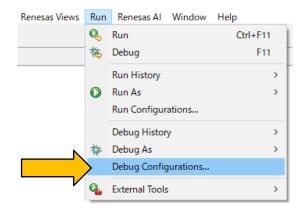
10. Build the project using the hammer icon in the upper left-hand side of the IDE. The project should build without any errors or warnings.





9. Modifying the e² studio Project Debug Session for Capacitive Touch Tuning

 The debug session 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 Run->Debug Configurations...



2. A dialog box will now open. In the list in the left-hand pane, open the **Renesas GDB Hardware Debugging** entry and select the named project *HardwareDebug* configuration, in this example: **Capacitive_Touch_Project_Example HardwareDebug**

type filter text ×	
C C/C++ Application	
C C/C++ Remote Application	
EASE Script	
C GDB Hardware Debugging	
💽 GDB Simulator Debugging (RH850)	
🜌 Java Applet	
🗾 Java Application	
🚭 Launch Group	
🖳 Remote Java Application	
🗸 💽 Renesas GDB Hardware Debugging	
Capacitive_Touch_Project_Example HardwareDebug	
c ≥ Renesas Simulator Debugging (RX, RL78)	

3. In the pane that opens, select the **Debugger** tab.

Name: Capacitive_1	Fouch_Project_Exa	mple Hardwa	reDebug		
📄 Main 🕸 Debu	ıgger 🐌 Startup	🦆 Source	Common		
Project:	4				
Capacitive_Tou	Project_Example				
C/C++ Applicati					
HardwareDebug/C	HardwareDebug/Capacitive_Touch_Project_Example.x				



4. Select the **Connection Settings** tab. For this application example, the power for the target board is supplied from the emulator power supply. **Main Clock Source**, Ensure Power Target From The Emulator (MAX 200mA) and Supply Voltage [V] are set as shown below.

) Main 😵 Debugger 🕟 Startup 🧤 Source 🔲 C Debug hart E2 Lite (RX) 🗸 Target Dev	ice: R5F51406	
GDB Setting Connection Settings Debug Tool Set	tings	
✓ Clock		~
Main Clock Source	носо	
Extal Frequency[MHz]	22.0	
Operating Frequency [MHz]	48.000	
Permit Clock Source Change On Writing Intern	al Flash Memory Yes 🗸 🗸	
 Connection with Target Board 		
Emulator	(Auto)	
Connection Type	Fine 🗸	
JTag Clock Frequency[MHz]	6.00 🗸	
Fine Baud Rate[Mbps]	1.50 🗸	
Hot Plug	No	
V Power		
Power Target From The Emulator (MAX 200mA)	Yes 🗸 🗸	
Supply Voltage (V)	3.3	
 CPU Operating Mode 		
Register Setting	Single Chip 🗸 🗸	
Mode pin	Single-chip mode 🗸 🗸	
Change startup bank	No	
A	0.1.0	~

5. Select the Startup tab.

Name: Capacitive_Touch_Project_Example HardwareDebug					
📄 Main 🕏	🌣 Debugger	▶ Startup	🧤 Source	Common	
	on Commands				
Reset	and Delay (sec	ond: 0			
Halt					

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

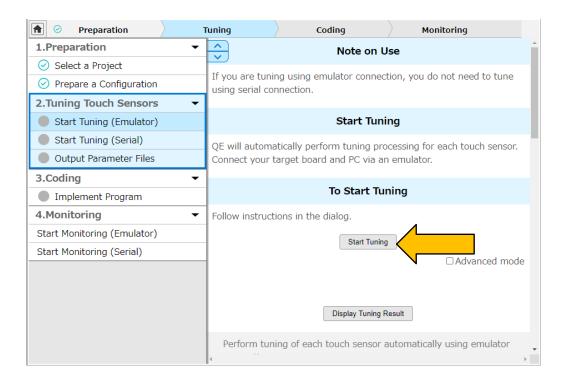
Runtime Options		
Set program counter at (hex):		
☑ Set breakpoint at: ☑ Resume	main]
Run Commands		

7. Click **Apply** then **Close** to use these modified settings. This completes the project configuration and debug setup for tuning.

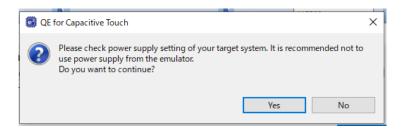


10. Tuning the Capacitive Touch Interface Using QE for Capacitive Touch Plug-in

1. Select [Start Tuning (Emulator)] in the menu on the left-hand side of "CapTouch Workflow (QE)" to open the settings of "Tuning Touch Sensors". Click on [Start Tuning] to start automatic tuning.



2. If the E1 debugger is supplying power to the target board, the following message will be displayed. Click **YES** to continue the tuning process.



Note: For sake of simplicity, in this application example, the power for the target board is supplied from the emulator power supply. Renesas recommends that tuning be done with the end application power supply to alleviate any voltage deviations that might occur from using the power supplied thru the E2 emulator Lite/PC USB port.

3. At the start of the first debug session, e² studio **MAY** display a message indicating the debug perspective will be switched to. Click the **Remember my decision** check box and **Yes** to continue the debug process and the QE for Capacitive Touch automatic tuning.

e ² Con	firm Perspective Switch	<
?	This kind of launch is configured to open the Debug perspective when it suspends. This Debug perspective is designed to support application debugging. It incorporates views for displaying the debug stack, variables and breakpoint management.	
	Do you want to open this perspective now?	
⊠ <u>R</u> en	nember my decision]
	<u>Y</u> es <u>N</u> o	



4. QE for Capacitive Touch automatic tuning will now begin. Please carefully read the tuning 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	×
1/7: QE is beginning the tuning process. During the tuning process, please do not touch the sensors on the target board until instructed the QE Tuning Program.	by
Cancel Help	>

5. After a number of 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 using **normal touch pressure** on the sensor being indicated in the dialog box (Button00/TS08). When pressing, the bar graph will increase to the right and the touch counts go numerically **UP.** While holding that pressure, **press any key on the PC keyboard** to accept the measurement.

\times

6. Repeat the previous steps for Button01/TS09.

Automatic Tuning Processing	×
6/7: QE will now measure touch sensitivity for (Button01, TS09 @ config01). In this step please use normal touch pressure on the sensor for once. Press any key on the PC keyboard to accept the sensitivity measurement.	
Button01, TS09 @ config01: 15326	
Cancel Help	



7. Once both buttons are complete, you will see a screen similar to what is shown below. This is the detection threshold that is used by the middleware to determine if a touch event has occurred.

Automatic Tuni The automatic sensors can b application no	c tuning e retriec	proce I. If the	ere are co	ontinued ov	verflows				× ed, those t the Renesas
Select the target	Method	Kind	Name	Touch Sensor	Threshold	Overflow	Warning / Error		
	config01	Button	Button00	TS08	1500				
	config01	Button	Button01	TS09	1321				
Retry Continue	the Tuning	Process					[Cancel	Help

8. 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. You should return to the default CapTouch Workflow (QE) screen in the e² studio IDE.

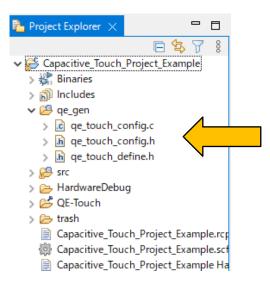
		config01	Button	Butto	
Retry	Continue	the Tuning	Process <		

9. The only remaining step is the output of the tuned parameter files. Select [Output Parameter Files] from the menu on the left-hand side of "CapTouch Workflow (QE)", and click on [Output Parameter Files].

Preparation	Tuning Coding Monitoring
1.Preparation -	To Output Parameter Files
📀 Select a Project	
Prepare a Configuration	Output parameter files from a tuning result.
2.Tuning Touch Sensors -	Output Parameter Files
Start Tuning (Emulator)	□ Specify an output folder
📀 Start Tuning (Serial)	□Use an external trigger
Output Parameter Files	□Use diagnostic code
3.Coding -	□Use API compatilibity mode
Implement Program	The result of tuning is output as a parameter file and can be read by



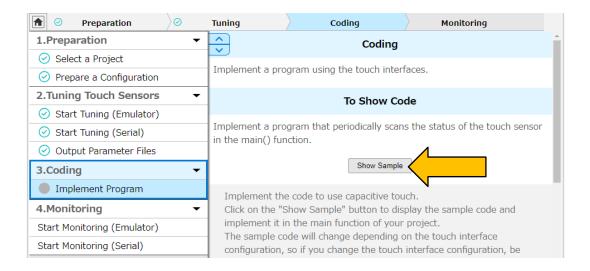
10. In the **Project Explorer** window and you will see that a **qe_touch_config.c**, **qe_touch_config.h** and **qe_touch_define.h** file pair have been added. These contain the needed tuning information to enable touch detection using the FIT API middleware.



11. Build the project using the hammer icon in the upper left-hand side of the IDE. The Console output showing build results should show no errors.

11. Adding rm_touch_qe FIT Module Calls to Application Example

1. To implement a program to scan the states of the touch sensors, select [Implement Program] in the menu on the left-hand side of "CapTouch Workflow (QE)", and then click on [Show Sample].

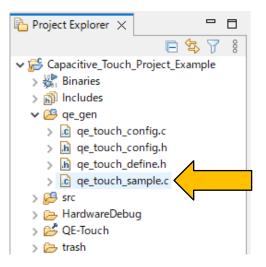




2. A new menu window will open showing the sample code in a text. Click the button Output to a File.

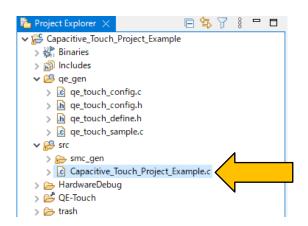
Show Sample Code		×	<			
Sample code of main() function:						
V*************************************	*****	1	•			
* FILE : qe_sample_main.c * DATE : 2022-03-09 * DESCRIPTION : Main Program for F *	X					
* NOTE:THIS IS A TYPICAL EXAMPLE	-					
<pre>#include "qe_touch_config.h" #include "qe_touch_config.h" #if ((TOUCH_CFG_UART_MONITOR_SUPPORT == 1) (TOUCH_CFG_UART_TUNING_SUPPORT == 1)) #include "r_sci_rx_pinset.h" #endif #endif #define TOUCH_SCAN_INTERVAL_EXAMPLE (20) /* milliseconds */</pre>						
void R_CTSU_PinSetInit(void); void qe_touch_main(void);						
#if ((TOUCH_CFG_UART_MONITOR_ #if (TOUCH_CFG_UART_NUMBER == #define QE_SCI_PIN_SET R_SCI_PinSe #elif (TOUCH_CFG_UART_NUMBER = #define QE_SCI_PIN_SET R_SCI_PinSe	et_SCI0 == 1)	T_TUNING_SUPPORT == 1))	,			
Copy to the Clipboard	Output to a File	Show the Application Note				
		OK Help]			

3. Confirm that the qe_touch_sample.c file has been generated in the [Project Explorer] window.

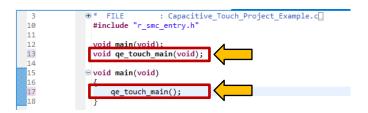




4. Open the Capacitive_Touch_Project_Example.c file.



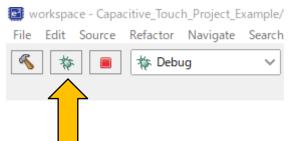
5. In the main() function, call the qe_touch_main(). Add the code ("void qe_touch_main(void);" and "qe_touch_main();") in the red frame to the Capacitive_Touch_Project_Example.c file as shown in the image below.



6. This completes all the needed code modifications required for this simple application example. Building the code should result in no errors or warnings for this simplified application example.

12. Monitoring Touch Performance using e² studio Expressions Window and QE for Capacitive Touch

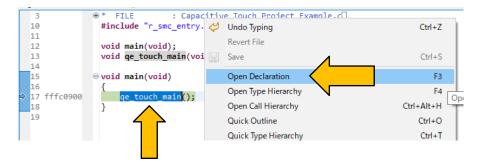
1. Start a debug session by clicking the **Bug** icon in the upper left hand corner of the e² studio. A debug session will start.



2. The debugger will stop at the qe_touch_main() function call. This is the first code point in the main() function.



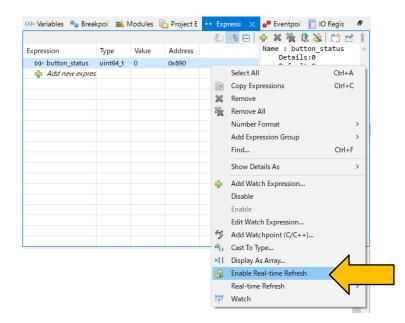
3. Open the declaration of the **qe_touch_main**() function.



4. Scroll down in the **qe_touch_sample.c** file to the **RM_TOUCH_DataGet**() function in the while (true) loop. Add the variable **button_status** to the expressions window.

99 fffc08fc	<pre>while (true) {}</pre>		Keractor	,
100 101 fffc08d0	<pre>} while (0 == g ge touch flag) {}</pre>		Declarations	>
102 fffc08e4	<pre>g_qe_touch_flag = 0;</pre>		References	>
103 104 fffc08de	err = RM TOUCH DataGet (g qe touch instance config01.p ctrl, &button status, NULL, NULL);		Search Text	>
105 😑	if (FSP_SUCCESS == err)		Build Selected File(s)	Ctrl+Alt+Shift+B
106	{ /* TOD0: Add your own code here. */		Clean Selected File(s)	Ctrl+Alt+Shift+C
108	}			
109 110			Resource Configurations	>
110			Step Into Selection	Ctrl+F5
2 112	/* FIXME: Since this is a temporary process, so re-create a waiting process yourself. */	⇒[Run to Line	Ctrl+R
113 fffc08f6 114 }	<pre>R_BSP_SoftwareDelay (TOUCH_SCAN_INTERVAL_EXAMPLE, BSP_DELAY_MILLISECS);</pre>	8	Move to Line	•
115 }		ъ.	Resume at Line	
116 117		×+y ≡?	Add Watch Expression	

5. Enable Real-time Refresh on the variable in the Expressions window.





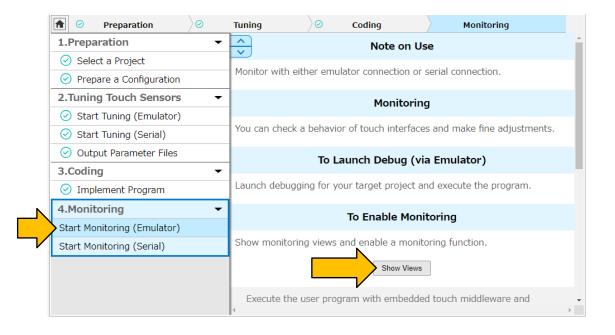
6. Click the '**Resume**' button located approximately in the middle of the e² studio menu bar to continue code execution.



7. Press **TS-B1** on the board, which was configured as **Button00** in Chapter ***8. Creating the Capacitive Touch Interface**" of this application guide. When pressed, a '1' will appear for **button_status** in the Expression window, indicating a binary indication of touch.

(x)= Variables 🛛 🔍 💩 Break	xpoi 🛛 🛋 N	Aodules	陷 Project E	$\cdot \cdot$ Expressi \times
				1 🛋 🔁
Expression	Туре	Value	Address	
🎄 button_status	uint64_t	1	0x890	
🐈 Add new expres		\wedge		

8. Select [Start Monitoring (Emulator)] in the menu on the left-hand side of "CapTouch Workflow (QE)". Click on [Show Views] to start [CapTouch Board Monitor (QE)].





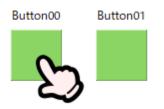
9. It may be necessary to drag the pane up for better viewing, however you should see the **CapTouch Board Monitor (QE)** pane appear similar to the image below.

🏷 CapTouch Board M	Monitor (QE) $ imes$		Ē.	ji D	1	G	₽.	000		
Enable Monitoring	Monitoring: Disa	abled, Communication	Status: (Connec	ting v	ia OC	D en	nulat	or	^
Touch I/F:		~								
										~
										^
		Button00	Butt	on01						

10. Click the Enable Monitoring button. The dialog text will change to Monitoring: Enabled.

ఏ CapTouch Board N	Monitor (QE) 🗙	5 🗖 C
Enable Monitoring	Monitoring: Disabled, Communication	on Status: Connecti
Touch I/F:		v
1		

11. T ouch the button **TS-B1** on the target board. The **CapTouch Board Monitor (QE)** will show a touch with a finger image on the button like the image below.





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

😂 CapTouch Status Chart (QE) 🛛 🗙		50 6	3 64 8	3 🖪	≥ ₁ 8	-	
Touch I/F:	Sync a selection						^
Select the touch I/F to monitor							
Touch Position: Reference Value:	Threshold: Difference:						
Start Data Collection							
							~
65535							
49149							
32766							
16383							
0							

13. Using the pulldown, select Button00 @ config01

Touch I/F:		×	Sync a selection
Select the te	Button00 @ config Button01 @ config	01	
Touch Pos		Reference Value:	Thresho



14. The graph will begin to display running values. Touch **TS-B1** on the board and you should see the 'Count Value' show as a step change on the running graph. The **GREEN** line is the touch '**Threshold**', which the middleware uses to determine whether a button is actuated/touched. The **RED BELT** at the bottom of the graph is a visual indication to the user that the 'Count Value' have crossed above the threshold and a touch is detected.

∑ CapTouch Status Chart (QE) ×					°° – 🗖
Touch I/F: Button00 @ config01 🗸		Syr	nc a selection		^
I/F Type: Button(self), Channel(s): TS08					
Count Value: 15414 Reference Value:	15425 Threshold:	1038 Difference:	-11		
Start Data Collection					
					~
17780				\sim	
\frown					
17170					
17179					
16579					
15979					
15379			_ ~		

Note: Sections 15 to 18 should only be set when displaying and measuring standard deviation.

15. Clicking on [Start Data Collection] displays the [Data Collection Settings] dialog box. Select the number of data to be collected from the pull-down menu in this dialog box. Select [Non Touch + Touch] for [Data collection target] and click on [Start Data Collection]. Do not touch the electrodes while collecting the touch-off state.

Touch I/F: Button00	@ config01 V			Sync a selection	^
I/F Type: Button(self),	Channel(s): TS08				
Count Value:	15379 Reference Value: 1	5411 Threshold:	1038 Difference:	-32	
Start Data Collection					
					~
	Data Collection Setting	s		×	
	Number of data collected:	1000		~	
	Data collection target:	O Non Touch	Non Touch + To	uch	

Start Data Collection



Cancel

16. When the green bar reaches the right edge, collection of the touch-off data is completed and a dialog box is displayed. Click on [OK] to close the dialog box.

Touch I/F: Button0	0 @ config01 🗸 🗸	Sync a selection	^
I/F Type: Button(self)	, Channel(s): TS08		
Count Value:	15404 Reference Value:	15407 Threshold: 1038 Difference: -3	
Stop Data Collectio	n		
CapTouch Status C	hart (QE) ×	<mark></mark>	• •
Touch I/F: Button0) @ config01 v	Sync a selection	^
I/F Type: Button(self)	, Channel(s): TS08		
Count Value:	15358 Reference Value:	15419 Threshold: 1038 Difference: -61	
Start Data Collectio	n		
			×
16457			
16180 🖾 QE fo	or Capacitive Touch	×	
	Next, collect data during Touc Start data Collection button.	h. When you are ready to take measurements, click the	
15906		ОК	

17. After that, touch the electrodes to collect touch-on data. Click on [Start Data Collection] while touching the electrodes. Continue to touch the electrodes until the data collection is complete.

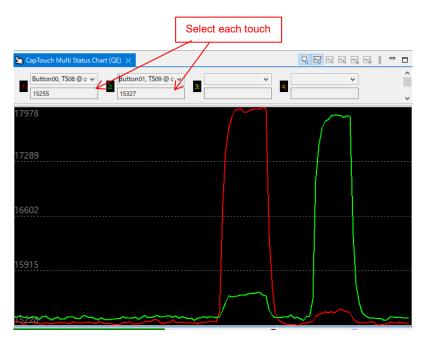
∑ CapTouch Status Chart (QE) ×			
Touch I/F: Button00 @ config01	¥	Sync a selection	
I/F Type: Button(self), Channel(s): TS08			
Count Value: 17529 Reference	e Value: 15405 Threshold: 1038	Difference: 2124	
Start Data Collection			
17531			~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
1.000			
16989			
16449			
15909			
15369			



18. When the green bar reaches the right edge to indicate the completion of data collection, the [Standard Deviation Measurement Result] dialog box appears. This dialog box shows the standard deviations of noise and the SNRs.

🔀 CapTouch Status Chart (QE) 🗙		- 0
Touch I/F: Button00 @ config01 v		^
VF Type: Button(self), Channel(s): TS08		
Count Value: 16960 Reference Value: 15381 Threshold: 1038 Difference: 1579		
		*
17034	~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~	~
Standard Deviation Measurement Result	×	
166 Noise Standard Deviation [NT]: 21.6 Average [NT]: 15389 Minimum: 15327 Maximum: 11	5503	
Noise Standard Deviation [T]: 21.9 Average [T]: 16980 Signal: 1591 SNR: 1	12.71	
SNR is calculated with a probability of 2.88.		
Measurement results are saved in the QE-Touch folder.		
162 ок		

19. To see a graphical representation of the 'touch counts' for multiple touch sensors, use the **CapTouch Multi Status Chart (QE)**.





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

CapTouch Parameters (X	Smart Browser	- Enable Monitoring - Display in Advanced Mode
V		- Read Value from the Target Board
ouch I/F: Button00 @ config	01 🗸 🗸 V 🖓 V 🗸 V V	- Write Value to the Target Board
/F Type: Button(self), Channel(s): TS08	- Enable Auto Writing
ltem	Value	- Output Parameter Files
Drift Correction Interval	255	
Long Touch Cancel Cycle	0	
Positive Noise Filter Cycle	3	
Negative Noise Filter Cycle	3	Touch Parameters
Moving Average Filter Depth	4	
Touch Threshold	1500	
Hysteresis	75	
CTSUSO	248	
CTSUSNUM	7	
et a drift correction interval. rift Correction is a function to ollow the surrounding environ put a value between 0 and 65 . The value is 1 or more: The re orrected every cycle specified terval] item. . The value is 0: No correction. his setting item will be applied	ment. 535. ference value will be in the [Drift Correction	A description is displayed for the selected touch

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

Note: Monitoring touch performance for touch applications can be confirmed by communication via the emulator.

On the other hand, monitoring touch performance can also be achieved via serial communication. Therefore, if you want to monitor smoothly, please add the monitoring function via serial communication.

Chapters 7 and 13 (including this chapter) shown below describe setting the serial communication monitor using UART.

- "7. [Additional function] Setting the serial communication monitor using UART (1/2)"
- "13. [Additional function] Setting the serial communication monitor using UART (2/2)"
- 1. Connect the target board to the PC via serial connection (UART / USB).



- 2. Run the touch application program on the target board.
- 3. Open the "CapTouch Workflow (QE)" panel. Make sure that Capacitive_Touch_Project_Example and Capacitive_Touch_Project_Example.tifcfg have been respectively set in [To Select a Project] and [To Prepare a Configuration]. Then select [Start Monitoring (Serial)] in the menu on the left-hand side of "CapTouch Workflow (QE)". Set the baud rate which was set in Chapter 7 for [Baud rate], and click on the [Connect] button.

▲ Oreparation	Tuning Octing Monitoring
1.Preparation -	Note on Use
⊘ Select a Project	Monitor with either emulator connection or serial connection.
Prepare a Configuration	Monitor with either emulator connection or serial connection.
2.Tuning Touch Sensors -	Monitoring
🕑 Start Tuning (Emulator)	-
🕑 Start Tuning (Serial)	You can check a behavior of touch interfaces and make fine adjustments.
⊘ Output Parameter Files	To Connect UART
3.Coding -	
 Implement Program 	Enable a monitoring function via serial communication, if you do not use an emulator.
4.Monitoring -	
Start Monitoring (Emulator)	Baud rate
Start Monitoring (Serial)	Port
	Auto
	To Enable Monitoring
	Show monitoring views and enable a monitoring function.
	Show Views
	Execute the user program with embedded touch middleware and

4. When serial connection is executed, the following display will be changed. Confirm a displaying message.

🕗 Output Parameter Files	To Connect UART	
3.Coding -	lo connect orien	
⊘ Implement Program	Enable a monitoring function via serial communication, if you do not us emulator.	e an
4.Monitoring -	Baud rate	
Start Monitoring (Emulator)	115200	
Start Monitoring (Serial)	Port	
	Auto	~
	To Enable Monitoring	
	Show monitoring views and enable a monitoring function.	
	Show Views	
	Execute the user program with embedded touch middleware and	▼
🗐 Console 🗙 🚟 Registers 🔝 Problems 🛞 Smart Brow	ser 🙀 Debugger Console	🖹 🛃 📝
QE for Capacitive Touch Connected to \\.\COM4.		



5. Open the Monitoring view from Show Views of CapTouch Main (QE).

🕑 Output Parameter Files	To Connect UART
3.Coding -	
Implement Program	Enable a monitoring function via serial communication, if you do not use an emulator.
4.Monitoring -	Baud rate
Start Monitoring (Emulator)	115200
Start Monitoring (Serial)	Port
	Auto
	Disconnect
	To Enable Monitoring
	Show monitoring views and enable a monitoring function.
	Show Views
	Execute the user program with embedded touch middleware and \checkmark

6. CapTouch Board Monitor (QE) pane will appear as shown below.

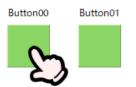
ScapTouch Board Monitor (QE) X	' 🗆
Enable Monitoring Monitoring: Disabled, Communication Status: Connecting via serial communication (UART / USB)	^
Touch I/F:	~
	^
Button00 Button01	
٢	>

7. Click the Enable Monitoring button. The dialog text will change to Monitoring: Enabled.

🏷 CapTouch Board Monitor (QE) 🔀	Ę,				[]	000		
Enable Monitoring Monitoring: Enabled, Communication Status: Connecting via ser	ial con	nmuni	catior	n (UA	RT / U	USB)		^
Touch								-
								^
Button00 Button01								
								~
<							>	•



8. Touch the button **TS-B1** on the target board. **The CapTouch Board Monitor (QE)** will show a touch with a finger image on the button like the below image.



- 9. The same procedure as in Chapter 12, paragraphs 11-20, can be used to verify that the touch count value changes.
- 10. If you will finish monitoring, click the **Enable Monitoring** button. The dialog text will change to **Monitoring: Disabled**.

🏷 CapTouch Board Monitor (QE) 🗙			5. pi []		Γ.	8	- 0
Enable Monitoring Monitoring: Enabled, Con	nmunication Sta	tus: Connecting via seri	al communica	tion (UA	RT / L	JSB)	^
	*						_ ~
							^
	Button00	Button01					
,			_				<u>ب</u> ۲
<							1

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

	Tuning	Coding	Monitoring	
1.Preparation -	\sim	Note on Us	e	Â
📀 Select a Project				1
Prepare a Configuration	Monitor with eit	her emulator connection or s	serial connection.	
2.Tuning Touch Sensors -		Monitoring	1	
Start Tuning (Emulator)			-	1
🔗 Start Tuning (Serial)	You can check a	behavior of touch interface	s and make fine adjustments.	
⊘ Output Parameter Files		To Connect U	ART	
3.Coding -				
 Implement Program 	Enable a monito emulator.	ring function via serial comr	munication, if you do not use an	
4.Monitoring -	Baud rate			
Start Monitoring (Emulator)	115200			1
Start Monitoring (Serial)	Port			J
	Auto	•	~	1
		Disconnect]	
		To Enable Moni	toring	
	Show monitoring	g views and enable a monito	oring function.	
		Show Views		
	Execute the u	ser program with embedded	d touch middleware and	+



14. qe_touch_sample.c Listing * FILE : qe_sample_main.c * DATE : 2022-03-09 * DESCRIPTION : Main Program for RX * NOTE: THIS IS A TYPICAL EXAMPLE. #include "ge touch config.h" #if ((TOUCH CFG UART MONITOR SUPPORT == 1) || (TOUCH CFG UART TUNING SUPPORT == 1)) #include "r sci rx pinset.h" #endif #define TOUCH_SCAN_INTERVAL_EXAMPLE (20) /* milliseconds */ void R CTSU PinSetInit(void); void qe_touch_main(void); #if ((TOUCH_CFG_UART_MONITOR_SUPPORT == 1) || (TOUCH_CFG_UART_TUNING_SUPPORT == 1)) #if (TOUCH_CFG_UART_NUMBER == 0) #define QE_SCI_PIN_SET R_SCI_PinSet_SCI0 #elif (TOUCH CFG UART NUMBER == 1) #define QE SCI PIN SET R SCI PinSet SCI1 #elif (TOUCH_CFG_UART_NUMBER == 2) #define QE_SCI_PIN_SET R_SCI_PinSet_SCI2 #elif (TOUCH_CFG_UART_NUMBER == 3) #define QE SCI PIN SET R SCI PinSet SCI3 #elif (TOUCH CFG UART NUMBER == $\overline{4}$) #define QE_SCI_PIN_SET R_SCI PinSet SCI4 #elif (TOUCH_CFG_UART_NUMBER == 5) #define QE_SCI_PIN_SET R_SCI_PinSet_SCI5 #elif (TOUCH_CFG_UART_NUMBER == 6) #define QE_SCI_PIN_SET R_SCI_PinSet_SCI6 #elif (TOUCH CFG UART NUMBER == 7) #define QE_SCI_PIN_SET R_SCI_PinSet_SCI7 #elif (TOUCH_CFG_UART_NUMBER == 8) #define QE_SCI_PIN_SET R_SCI_PinSet_SCI8 #elif (TOUCH_CFG_UART_NUMBER == 9) #define QE_SCI_PIN_SET R_SCI_PinSet_SCI9 #elif (TOUCH_CFG_UART_NUMBER == 10) #define QE SCI PIN SET R SCI PinSet SCI10 #elif (TOUCH_CFG_UART_NUMBER == 11) #define QE_SCI_PIN_SET R_SCI_PinSet_SCI11 #elif (TOUCH CFG UART NUMBER == 12) #define QE SCI PIN SET R SCI PinSet SCI12 #endif #endif 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;
  /* Initialize pins (function created by Smart Configurator) */
  R_CTSU_PinSetInit();
#if ((TOUCH_CFG_UART_MONITOR_SUPPORT == 1) || (TOUCH_CFG_UART_TUNING_SUPPORT == 1))
  /* Setup pins for SCI (function created by Smart Configurator) */
  QE_SCI_PIN_SET();
#endif
  /* 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_MILLISECS);
  }
```

}



15. [Additional function] Setting The Automatic Judgement and MEC functions

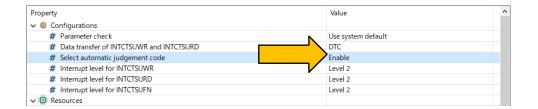
This Chapters shown below describe setting The Automatic Judgement and MEC (Multi Electrode Connection) functions.

Note: The Automatic Judgement and MEC functions are supported by products equipped with CTSU2SL.

- 1. Create a project and add modules following the same procedure as described in section 6.11.
- Add the code generation component r_dtc_rx in the same way as the rm_touch_qe as described in section 6.6. Select r_dtc_rx and click Finish.

Components	Short Name	Туре	Version	^
🖶 Dead-time Compensation Counter		Code Generator	1.11.0	
🖶 DTC driver	r_dtc_rx	Firmware Integr	4.30	
🖶 Event Link Controller		Code Generator	1.9.1	
🖶 Flash API for RX100, RX200, RX600. and R	r_flash_rx	Firmware Integr	5.00	

3. Then select **r_ctsu_qe** and display the ports associated with this module in the configuration panel. Set **Data transfer of INTCTSUWR and INTCTSURD**"to **DTC** and **Select automatic judgement code** to **Enable**.



4. Next, select **r_dtc_rx** in the **Components** tab and display the ports associated with this module in the configuration panel. Set **DMAC FIT check** to **DMAC FIT module is not used with DTC FIT module**.

Property	Value
🗸 🏶 Configurations	
# Parameter check	System Default
# DTCER control	Clear all DTCER registers in R_DTC_Open()
# Address mode	Full address mode
# Transfer Data Read Skip	Enable transfer data read skip
# DMAC FIT check	DMAC FIT module is not used with DTC FIT modu 🗸
# Sequence transfer	DMAC FIT module is not used with DTC FIT module.
	DMAC FIT module is used with DTC FIT module.

5. Next, select **r_bsp** in the **Components** tab and set **Help size** to **0x1000**.

Property	Value	
🗸 🏶 Configurations		
# User stack setting	2 stacks	
# User stack size	0x400	
# Interrupt stack size	0x100	
# Heap size	0x1000	
# Initializes C input and output library functions	Enable	
# Enable user stdio charget function	Use BSP charget() function	
# User stdio charget function name	my_sw_charget_function	



6. Generate the needed application code modules for the project. Do this by clicking the Generate Code icon in the upper-right of the Smart Configurator pane as shown in the picture below.



- 7. Follow the same procedure as in Chapter 8.1 through Chapter 8.7 to set up the touch interface..
- 8. Click on Setup Configuration.

Create Configuration of Touch Interfa	ices		×
File Name of Touch I/F:	Capacitive_Touch_Project_Example	Setup Configuration	Import / Re-edit
Description:			
			Touch I/F 🖈
			Capacitance Type
			Self Capacitance 🗸
			Button
			Slider (horizontal)
	Button00 Button01		Slider (vertical)
	Buttonuu		Wheel
	TS08 TS09		Key pad
			3D Gesture (AI)
			Touch pad
			Shield Pin
			TC Pin
			Capacitance Sensor
			Current Sensor
			Diagnosis Pin
Setting			Remove Touch I/F
Setup Touch I/F Setup	Resistance Value Clear Assigned TSx		Configuratoins (Methods) 🛛 🗧
			Create Cancel Help

9. The Setup Configuration (Method) will appear. Select Add Configuration and add a method.

Setup Configuratio	ns (Methods) X
Add Configuration	Remove Configuration
	Config01
Button00(self)	✓ Available
Button01(self)	✓ Available
Auto Sensing by Hard	vare Enable
Multiple Electrode Con	nection Enable
c	K Cancel Help



10. The method **config02** is added.

Setup Configurations (Method Add Configuration Remove Co	s) nfiguration	×
	config01	config02
Button00(self) Button01(self)	✓ Available✓ Available	 ✓ Available ✓ Available
Auto Sensing by Hardware Multiple Electrode Connection	Enable	Enable Enable Enable
	OK	Cancel Help

11. Check Enable for Auto Sensing by Hardware and Multiple electrode connection in config02 and click OK.

Setup Configurations (Methods))	×
Add Configuration Remove Con	figuration	
	Config01	config02
Button00(self)	✓ Available	✓ Available
Button01(self)	✓ Available	✓ Available
Auto Sensing by Hardware	Enable	
Multiple Electrode Connection	Enable	🗹 Enable
	OK	Cancel Help

12. Click Create in the Create Configuration of Touch Interface window. This will setup the Touch Interface.

		Remove	Touch I/F	
		Configuratoin	s (Methods) 💦	\$
 	_			
Create		Cancel	Help	



 You can check the configuration of the touch interface by selecting [Renesas Views] – [Renesas QE] – [CapTouch Tuning Result (QE)] from the menu of the e² studio. Open the [CapTouch Tuning Result (QE)] view to display the [Tuning] panel.

Tuning Ge	esture							
Touch I/F C	Configuration:	Capacitive_1	Touch_Project_Ex	ample				
Method	Kind	Name	Tauch Canada	Demoitie Comositer estad	Come Drive Dates Frances (MUL-1			
		INDITIC	louch sensor	Parasitic Capacitance[pr]	Sensor Drive Pulse Frequency[MHz]	Ihreshold	Scan Time[ms]	Overflow
	Button(self)			-	-	I hreshold	Scan Time[ms]	Overflow None
config01		Button00	TS08	-	- -		Scan Time[ms] - -	

14. Build the project using the hammer icon in the upper left-hand side of the IDE. The project should build without any errors or warnings.

🖾 workspace - e² studio							
File	Edit	Navigate	Search	Project	Renesas		
	*		🎄 Debu	ig	~		

15. Thereafter, capacitive touch sensor tuning and monitoring can be performed using the same procedure as in Chapter 9 and beyond.



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/ge-capacitive-touch

Technical Support Contact Form

https://www.renesas.com/support

Inquiries

http://www.renesas.com/contact/

All trademarks and registered trademarks are the property of their respective owners.



Revision History

		Description	
Rev.	Date	Page	Summary
1.00	Oct.17.2018	-	Initial Revision
2.00	Aug.26.2022	-	Updated the development environment
3.00	Sep.26.2023	-	Updated the development environment



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 1Discharge (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 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 systemevaluation test for the given product.

Notice

- 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.
- This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
 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/.