Introduction

This application note introduces PoC for the home appliance panel UI demo set using CTSU2La on RL78/G22 with touch buttons and MEC (Multiple Electrode Connection) function (hereinafter RL78/G22 PoC).

The MEC function (Multiple Electrode Connection function) is that regards multiple touch electrodes as a one electrode. For example, there is a system with six touch buttons and this function is ideal for a system that returns from standby mode by touching any of the touch buttons. Devices without the MEC function require six scans to determine whether a touch is detected, whereas RL78/G22 can determine whether a touch is detected with one scan. Thus, fewer scans are required, which enables low power consumption operation.

Also, setting the touch detection with high sensitivity when using the MEC function allows multiple touch electrodes can be used as one large proximity sensor electrode.

Target Device

RL78/G22

When using this application note with other Renesas MCUs, careful evaluation is recommended after making modifications to comply with the alternate MCU.

Target Tool

CPU Board (RTK0EG0041C01001BJ) of RL78/G22 Capacitive Touch Evaluation System (RSSK) (RTK0EG0042S01001BJ)
Contents

1. Outline .................................................................................................................................................. 4
   1.1 MEC function .................................................................................................................................. 4
      1.1.1 Advantage 1 of MEC function (return from standby mode by touching any electrode) .......... 4
      1.1.2 Advantage 2 of MEC function (available as a proximity sensor) ............................................ 5
      1.1.3 Advantage 3 of MEC function (low power consumption) .......................................................... 5
   1.2 How to utilize the MEC function in RL78/G22 PoC ................................................................. 6

2. Operation Confirmation Conditions .................................................................................................... 7

3. Sample Programs .................................................................................................................................. 8
   3.1 State Transition of Demonstration Screen .................................................................................. 8
   3.2 Flowchart ....................................................................................................................................... 9
      3.2.1 Overall Flowchart .................................................................................................................... 9
   3.3 Pins Used ...................................................................................................................................... 10
   3.4 Sample Programs Structure .......................................................................................................... 10
      3.4.1 Peripheral Functions Used ..................................................................................................... 10
      3.4.2 Peripheral Function Settings .................................................................................................. 11
      3.4.3 File Structure .......................................................................................................................... 13
      3.4.4 Variables .................................................................................................................................. 14
      3.4.5 Constants ................................................................................................................................ 15
      3.4.6 Functions .................................................................................................................................. 16
      3.4.7 Function Specifications ............................................................................................................ 17
      3.4.8 Flowchart ............................................................................................................................... 24
         3.4.8.1 Flowchart of main function ................................................................................................. 24
         3.4.8.2 Flowchart of r_touch_init function ..................................................................................... 24
         3.4.8.3 Flowchart of r_touch_main function .................................................................................... 25
         3.4.8.4 Flowchart of r_snooze_mode_touch_proses function ............................................................ 26
         3.4.8.5 Flowchart of r_change_eco_mode function ............................................................................ 26
         3.4.8.6 Flowchart of r_prevent_long_presses function .................................................................... 27
         3.4.8.7 Flowchart of r_snooze_mode_init function ........................................................................... 27
         3.4.8.8 Flowchart of r_snooze_mode function .................................................................................. 28
         3.4.8.9 Flowchart of r_touch_mec_scanstart function ..................................................................... 29
         3.4.8.10 Flowchart of r_touch_mec_scansstop function .................................................................. 29
         3.4.8.11 Flowchart of r_touch_mec_dataget function ...................................................................... 30
         3.4.8.12 Flowchart of r_nomal_mode_init function .......................................................................... 31
         3.4.8.13 Flowchart of r_nomal_mode function .................................................................................. 32
         3.4.8.14 Flowchart of r_change_snooze_nomal function .................................................................. 33
         3.4.8.15 Flowchart of r_change_nomal_snooze function ................................................................. 33
         3.4.8.16 Flowchart of r_not_touched function ................................................................................ 34
         3.4.8.17 Flowchart of r_ledport_input function .............................................................................. 34
3.4.8.18 Flowchart of r_ledport_output function ................................................................. 35
3.4.8.19 Flowchart of r_led_init function ............................................................................. 35
3.4.8.20 Flowchart of r_led_turn_on_all_5s function ............................................................. 36
3.4.8.21 Flowchart of r_change_led function ......................................................................... 37
3.4.8.22 Flowchart of r_change_led_position function ........................................................... 38
3.4.8.23 Flowchart of r_led_turn_on function ......................................................................... 39
3.4.8.24 Flowchart of r_led_turn_off function ....................................................................... 40
3.4.8.25 Flowchart of r_ledmatrix_turn_on function ............................................................... 41
3.4.8.26 Flowchart of r_ledmatrix_turn_off function ............................................................... 42
3.4.8.27 Flowchart of r_ledmatrix_turn_on_a function ............................................................ 43
3.4.8.28 Flowchart of r_Config_TAU0_0_interrupt function .................................................... 44
3.4.8.29 Flowchart of r_sec_count_timer_start function ........................................................ 45
3.4.8.30 Flowchart of r_sec_count_timer_reset function ........................................................ 46
3.4.8.31 Flowchart of r_Config_TAU0_1_interrupt function .................................................... 46
3.4.8.32 Flowchart of r_ledmatrix_timer_start function ........................................................ 47
3.4.8.33 Flowchart of r_ledmatrix_timer_reset function ........................................................ 47

4. Importing a Project .............................................................................................................. 48
   4.1 Procedure in e² studio ..................................................................................................... 48
   4.2 Procedure in CS+ ........................................................................................................... 49

5. Starting a Demonstration .................................................................................................... 50
   5.1 Powered on RL78/G22 PoC and menu screen ............................................................... 51
   5.2 Return from standby mode ............................................................................................ 51
   5.3 Touch operation ............................................................................................................. 52
       5.3.1 Set operation mode ................................................................................................. 52
       5.3.2 Set freezing ............................................................................................................ 52
       5.3.3 Set refrigerator ....................................................................................................... 52
       5.3.4 Set ice making ...................................................................................................... 53
       5.3.5 Set cooling mode ................................................................................................... 53
       5.3.6 Set chilled mode ................................................................................................... 53
       5.3.7 eco mode (proximity sensor mode) ........................................................................ 54
       5.3.8 eco mode (touch sensor mode) ............................................................................. 55

6. Reference Documents ......................................................................................................... 56

Revision History ...................................................................................................................... 57
1. Outline

This application note describes the configuration and operation of RL78/G22 PoC with a refrigerator panel motif. RL78/G22 PoC is equipped with seven touch buttons. These touch buttons work as independent touch buttons during normal mode. If the touch panel is not operated for a certain time, the panel is hidden and the device transitions to standby mode. In standby mode, the six touch buttons function as one touch button by MEC function.

1.1 MEC function

The MEC function is a function to measure multiple channels of touch electrodes as one electrode.

1.1.1 Advantage 1 of MEC function (return from standby mode by touching any electrode)

In RL78/G22 PoC, six touch electrodes are used as one electrode by the MEC function during standby mode. Thus, the CPU can return from standby mode by touching any of the electrodes within the white frame.

![Figure 1-1 MEC function (one electrode)](image)
1.1.2 Advantage 2 of MEC function (available as a proximity sensor)

By using the MEC function in an arrangement configuration that places touch electrodes in proximity, multiple touch electrodes can be regarded as one large electrode. Also depending on the touch threshold setting, it is available to be used as a proximity sensor.

![Diagram of proximity sensor and touch sensor](image)

**Figure 1-2 MEC function (proximity sensor)**

1.1.3 Advantage 3 of MEC function (low power consumption)

The MEC function uses multiple touch electrodes as one electrode. Thus, the electrodes can be scanned only once. Since only one electrode scan is required, low power consumption operation is possible.

![Diagram of normal mode and MEC function](image)

**Figure 1-3 MEC function (low power consumption)**
1.2 How to utilize the MEC function in RL78/G22 PoC

RL78/G22 PoC implements a variation of operation using the MEC function as operation mode.

In operation mode 1, the touch threshold of the buttons is set low enough to detect at hand proximity. Holding the hand over the white frame area returns from standby mode.

In operation mode 2, the touch threshold of the buttons is set to detect a direct hand touch. Touching any buttons returns from standby mode.

Operation mode 3 is a mode for implementing additional functions.

Note: The operation mode 3 is not implemented.

Figure 1-4 Overall system image
2. Operation Confirmation Conditions

The operation of the sample program has been confirmed under the following conditions.

Table 2-1 Operation Confirmation Conditions

<table>
<thead>
<tr>
<th>Item</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU Board</td>
<td>RL78/G22 RSSK CPU Board (RTK0EG0041C01001BJ)</td>
</tr>
<tr>
<td></td>
<td>(RL78/G22 Capacitive Touch Evaluation System (RSSK) (RTK0EG0042S01001BJ) accessory)</td>
</tr>
<tr>
<td>Electrode Board</td>
<td>• Appliance panel electrode board for Touch MEC function</td>
</tr>
<tr>
<td></td>
<td>(with enclosure)</td>
</tr>
<tr>
<td></td>
<td>Self-capacitance method buttons: 7</td>
</tr>
<tr>
<td></td>
<td>LED: 25</td>
</tr>
<tr>
<td>MCU used</td>
<td>RL78/G22 (R7F102GGE)</td>
</tr>
<tr>
<td></td>
<td>(ROM: 64KB, RAM: 4KB)</td>
</tr>
<tr>
<td>Operating frequency (HOCO)</td>
<td>• Main system clock</td>
</tr>
<tr>
<td></td>
<td>High-speed on-chip oscillator clock (fIH) : 32 MHz</td>
</tr>
<tr>
<td></td>
<td>• CPU/peripheral hardware clock (fCLK) : 32 MHz</td>
</tr>
<tr>
<td></td>
<td>• Subsystem clock</td>
</tr>
<tr>
<td></td>
<td>Low-speed on-chip oscillator clock (fIL) : 32.768 kHz</td>
</tr>
<tr>
<td></td>
<td>• Low-speed peripheral clock frequency (fSXP) : 32.768 kHz</td>
</tr>
<tr>
<td>Operating voltage</td>
<td>3.3 V (can operate from 1.8 V to 5.5 V)</td>
</tr>
<tr>
<td>Integrated development environment (e² studio)</td>
<td>Renesas Electronics</td>
</tr>
<tr>
<td></td>
<td>e² studio Version 2023-01 (23.1.0)</td>
</tr>
<tr>
<td>Smart Configurator (SC)</td>
<td>Renesas Electronics</td>
</tr>
<tr>
<td></td>
<td>V1.5.0</td>
</tr>
<tr>
<td>C compiler (e² studio)</td>
<td>Renesas Electronics</td>
</tr>
<tr>
<td></td>
<td>CC-RL V1.12.00</td>
</tr>
<tr>
<td>QE for Capacitive Touch</td>
<td>Renesas Electronics</td>
</tr>
<tr>
<td></td>
<td>V3.2.0</td>
</tr>
<tr>
<td>Board support package (r_bsp)</td>
<td>V1.40</td>
</tr>
<tr>
<td>Emulator</td>
<td>Renesas E2 emulator Lite (RTE0T0002LKCE00000R)</td>
</tr>
</tbody>
</table>

The sample code uses the SIS driver/middleware and Code Generator shown in Figure 2-1.
3. Sample Programs

3.1 State Transition of Demonstration Screen

The state transition of demonstration screen for this sample program is shown below. Refer to Chapter 5 for details on the screen.

![Diagram of state transition]

**Figure 3-1 State transition of demonstration screen**
3.2 Flowchart

3.2.1 Overall Flowchart
The overall flowchart is shown below.

```
main

EI()

Initial LED settings
r_led_init()

Initial CTSU settings
r_touch_init()

Operate CTSU
r_touch_main()

r_touch_main()

Process during SNOOZE mode
r_snooze_mode()

Process change from SNOOZE mode to normal mode
r_change_snooze_normal()

Process during normal mode
r_normal_mode()

Reset the timer
r_sec_count_timer_reset()

Process to turn off LED
r_led_turn_off()

Process change from normal mode to SNOOZE mode
r_change_normal_snooze()

return
```

Figure 3-2 Overall flowchart
3.3 Pins Used

The following shows lists pins used in this sample program.

### Table 3-1 List of Pins and Functions

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Input/Output</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>P26</td>
<td>Input/Output</td>
<td>Matrix LED anode 0</td>
</tr>
<tr>
<td>P23</td>
<td>Input/Output</td>
<td>Matrix LED anode 1</td>
</tr>
<tr>
<td>P21</td>
<td>Input/Output</td>
<td>Matrix LED anode 2</td>
</tr>
<tr>
<td>P20</td>
<td>Input/Output</td>
<td>Matrix LED anode 3</td>
</tr>
<tr>
<td>P120</td>
<td>Input/Output</td>
<td>Matrix LED cathode 0</td>
</tr>
<tr>
<td>P121</td>
<td>Input/Output</td>
<td>Matrix LED cathode 1</td>
</tr>
<tr>
<td>P122</td>
<td>Input/Output</td>
<td>Matrix LED cathode 2</td>
</tr>
<tr>
<td>P146</td>
<td>Input/Output</td>
<td>Matrix LED cathode 3</td>
</tr>
<tr>
<td>P41</td>
<td>Input/Output</td>
<td>Matrix LED cathode 4</td>
</tr>
<tr>
<td>P61</td>
<td>Input/Output</td>
<td>Matrix LED cathode 5</td>
</tr>
<tr>
<td>P62</td>
<td>Output</td>
<td>LED independent control</td>
</tr>
<tr>
<td>TS18</td>
<td>Input</td>
<td>operation mode</td>
</tr>
<tr>
<td>TS28</td>
<td>Input</td>
<td>Freezing</td>
</tr>
<tr>
<td>TS00</td>
<td>Input</td>
<td>refrigerator</td>
</tr>
<tr>
<td>TS05</td>
<td>Input</td>
<td>ice making</td>
</tr>
<tr>
<td>TS06</td>
<td>Input</td>
<td>cooling mode</td>
</tr>
<tr>
<td>TS07</td>
<td>Input</td>
<td>chilled mode</td>
</tr>
<tr>
<td>TS01</td>
<td>Input</td>
<td>eco mode</td>
</tr>
<tr>
<td>TSCAP</td>
<td>-</td>
<td>TSCAP pin</td>
</tr>
</tbody>
</table>

Note: Except for the controlled LEDs, the port mode is set to input so that the input level of the LEDs is set to high impedance and they are not emitted.

3.4 Sample Programs Structure
3.4.1 Peripheral Functions Used

The following shows lists peripheral functions used in this sample program.

### Table 3-2 List of Peripheral Functions Used and Functions

<table>
<thead>
<tr>
<th>Peripheral Functions</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTSU2La</td>
<td>Used for touch buttons</td>
</tr>
<tr>
<td>TAU00</td>
<td>Used for seconds count</td>
</tr>
<tr>
<td>TAU01</td>
<td>Used for LED matrix control internally</td>
</tr>
<tr>
<td>PORT</td>
<td>Used for LED</td>
</tr>
<tr>
<td>ITL000</td>
<td>Used for CTSU2La trigger</td>
</tr>
<tr>
<td>ELC</td>
<td>Used for connecting CTSU2La and 32-bit interval timer</td>
</tr>
</tbody>
</table>
3.4.2 Peripheral Function Settings

The Smart Configurator settings used in this sample program are shown below. The items and settings in each table in the Smart Configurator settings are described in the notation on the configuration screen.

### Table 3-3 Parameters of Smart Configurator

<table>
<thead>
<tr>
<th>Tag name</th>
<th>Components</th>
<th>Contents</th>
</tr>
</thead>
</table>
| **Clocks** | - | **Operation mode**: High-speed main mode 1.8 (V) - 5.5 (V)  
High-speed on-chip oscillator : 32MHz  
fHOCO start setting : Normal  
fHISP : 32MHz  
fMAIN : 32MHz  
fCLK : 32000kHz |
| **System** | - | **On-chip debug operation setting**: Use emulator  
Emulator setting : E2 Lite  
Pseudo-RRM/DMM function setting : Used  
Start/Stop function setting : Unused  
Security ID setting : Use security ID  
Security ID : 0x00000000000000000000  
Security ID authentication failure setting : Erase flash memory data |
| **Component** | r_bsp | **Start up select**: Enable (use BSP startup)  
Control of invalid memory access detection : Disable  
RAM guard space (GRAM0-1) : Disabled  
Guard of control registers of port function (GPORT) : Disabled  
Guard of registers of interrupt function (GINT) : Disabled  
Guard of control registers of clock control function, voltage detector, and RAM parity error detection function (GCSC) : Disabled  
Data flash access control (DFLEN) : Disables  
Initialization of peripheral functions by Code Generator/Smart Configurator : Enable  
API functions disable : Enable  
Parameter check enable : Enable  
Setting for starting the high-speed on-chip oscillator at the times of release from STOP mode and of transitions to SNOOZE mode : High-speed  
Enable user warm start callback (PRE) : Unused  
Enable user warm start callback (POST) : Unused  
Watchdog Timer refresh enable : Unused |
| **Config_ITL000** | Components : Interval Timer  
Operation : 8 bit count mode  
Resource : ITL000  
Operation clock : fHISP  
Clock source : fITL/16  
Interval value : 20ms  
Interrupt setting : Unused |
<table>
<thead>
<tr>
<th>Tag name</th>
<th>Components</th>
<th>Contents</th>
</tr>
</thead>
</table>
| Compone nts | Config_TAU0_0 | Components : Interval Timer  
Operation : 16 bit count mode  
Resource : TAU0_0  
Operation clock : CK00  
Clock source : fCLK/2^10  
Interval value : 1000ms  
Interrupt setting : Used  
Priority : Level 3 |
|             | Config_TAU0_1 | Components : Interval Timer  
Operation : 16 bit count mode  
Resource : TAU0_1  
Operation clock : CK01  
Clock source : fCLK/2^8  
Interval value : 7ms  
Interrupt setting : Used  
Priority : Level 3 |
|             | Config_ELC | Components : Event Link Controller  
Output destination setting : CTSU2La Capacitive sensing unit  
Event generation source : 32-bit interval timer 0 compare match |
|             | Config_PORT | Components : Ports  
Port selection : PORT2, PORT4, PORT6, PORT12, PORT14  
Port mode setting : Read Pmn resister values  
PORT2 : Checked “In” on P20, P21, P23 and P26  
PORT4 : Checked “In” on P41  
PORT6 : Checked “Out” and “Output 1” on P61 and P62  
PORT12 : Checked “In” on P120–P122  
PORT14 : Checked “In” on P146 |
|             | r_ctsu | Components : r_ctsu  
Resource : CTSU  
TS00 Pin : Used  
TS01 Pin : Used  
TS05 Pin : Used  
TS06 Pin : Used  
TS07 Pin : Used  
TS18 Pin : Used  
TS28 Pin : Used  
Settings other than the above are defaults. |
|             | rm_touch | Components : rm_touch  
Default setting is used. |
### 3.4.3 File Structure

The following shows file structure by sample program.

#### Table 3-4 File Structure

<table>
<thead>
<tr>
<th>Folder name, File name</th>
<th>Outline</th>
</tr>
</thead>
<tbody>
<tr>
<td>r01an6740_g22demotouch_mec</td>
<td>Folder for program source</td>
</tr>
<tr>
<td>binary</td>
<td>Source file for main processing</td>
</tr>
<tr>
<td>includes</td>
<td>Header file for main processing</td>
</tr>
<tr>
<td>qe_gen</td>
<td>QE for capacitive touch generation</td>
</tr>
<tr>
<td>src</td>
<td>Header file for LCD related</td>
</tr>
<tr>
<td>smc_gen</td>
<td>Smart Configurator generation</td>
</tr>
<tr>
<td>└ Config_ELC</td>
<td></td>
</tr>
<tr>
<td>└ Config_ITL000</td>
<td></td>
</tr>
<tr>
<td>└ Config_PORT</td>
<td></td>
</tr>
<tr>
<td>└ Config_TAU0_0</td>
<td></td>
</tr>
<tr>
<td>└ Config_TAU0_1</td>
<td></td>
</tr>
<tr>
<td>└ general</td>
<td></td>
</tr>
<tr>
<td>└ r_bsp</td>
<td></td>
</tr>
<tr>
<td>└ r_config</td>
<td></td>
</tr>
<tr>
<td>└ r_ctsu</td>
<td></td>
</tr>
<tr>
<td>└ r_pincfg</td>
<td></td>
</tr>
<tr>
<td>└ rm_touch</td>
<td></td>
</tr>
<tr>
<td>└ led.c</td>
<td>Source file for LED control</td>
</tr>
<tr>
<td>└ led.h</td>
<td>Header file for LED control</td>
</tr>
<tr>
<td>└ main.c</td>
<td>Source file for main processing</td>
</tr>
<tr>
<td>└ mode.c</td>
<td>Source file for mode control</td>
</tr>
<tr>
<td>└ mode.h</td>
<td>Header file for mode control</td>
</tr>
<tr>
<td>└ touch.c</td>
<td>Source file for touch control</td>
</tr>
<tr>
<td>└ touch.h</td>
<td>Header file for touch control</td>
</tr>
<tr>
<td>└ QE-Touch</td>
<td>QE-Touch generation</td>
</tr>
</tbody>
</table>
### Variables

The following shows the variables that are used in this sample program.

#### Table 3-5 List of variables used in the sample code

<table>
<thead>
<tr>
<th>Type</th>
<th>Variable name</th>
<th>Contents</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>uint8_t</td>
<td>g_led_position[6]</td>
<td>Array of matrix LED lighting patterns</td>
<td>led.c mode.c touch.c</td>
</tr>
<tr>
<td></td>
<td>g_pos_a</td>
<td>Specified variable on the anode side of the control LED of the matrix LED</td>
<td></td>
</tr>
<tr>
<td></td>
<td>g_touch_button_flg</td>
<td>Flag that any button is touched</td>
<td>led.c mode.c touch.c</td>
</tr>
<tr>
<td></td>
<td>g_eco_mode_flg</td>
<td>Flag that &quot;eco mode&quot; button is touched</td>
<td>led.c touch.c</td>
</tr>
<tr>
<td></td>
<td>g_sec_count_timer_count</td>
<td>Seconds count variable</td>
<td>led.c Config_TAU0_0_user.c</td>
</tr>
<tr>
<td></td>
<td>g_sec_count_timer_stop_flg</td>
<td>Flag that have completed the seconds count process</td>
<td>Config_TAU0_0_user.c</td>
</tr>
<tr>
<td></td>
<td>g_mode</td>
<td>Normal mode/SNOOZE mode switching variable</td>
<td>led.c mode.c touch.c</td>
</tr>
<tr>
<td></td>
<td>g_normal_end_flg</td>
<td>Flag that ended processing in normal mode</td>
<td>led.c mode.c</td>
</tr>
<tr>
<td></td>
<td>g_button_status</td>
<td>Status of which button was touched</td>
<td>led.c mode.c touch.c</td>
</tr>
<tr>
<td></td>
<td>g_snooze_mode_init</td>
<td>SNOOZE mode initialization flag</td>
<td>mode.c</td>
</tr>
<tr>
<td></td>
<td>g_normal_mode_init</td>
<td>Normal mode initialization flag</td>
<td>mode.c</td>
</tr>
<tr>
<td></td>
<td>g_snooze_end_flg</td>
<td>Flag that ended processing in SNOOZE mode</td>
<td>mode.c</td>
</tr>
</tbody>
</table>
### 3.4.5 Constants

The following shows the constants that are used in this sample program.

#### Table 3-6 List of constants used in the sample code

<table>
<thead>
<tr>
<th>Constant name</th>
<th>Setting value</th>
<th>Contents</th>
<th>File</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEC</td>
<td>0x01</td>
<td>Value of g_button_status at MEC button touch</td>
<td>touch.h</td>
</tr>
<tr>
<td>OPERATION_MODE</td>
<td>0x20</td>
<td>Value of g_button_status at &quot;operation mode&quot; button touch</td>
<td>touch.h</td>
</tr>
<tr>
<td>FREEZING</td>
<td>0x40</td>
<td>Value of g_button_status at &quot;freezing&quot; button touch</td>
<td>touch.h</td>
</tr>
<tr>
<td>REFRIGERATOR</td>
<td>0x01</td>
<td>Value of g_button_status at &quot;refrigerator&quot; button touch</td>
<td>touch.h</td>
</tr>
<tr>
<td>ICE_MAKING</td>
<td>0x04</td>
<td>Value of g_button_status at &quot;ice making&quot; button touch</td>
<td>touch.h</td>
</tr>
<tr>
<td>COOLING_MODE</td>
<td>0x08</td>
<td>Value of g_button_status at &quot;cooling mode&quot; button touch</td>
<td>touch.h</td>
</tr>
<tr>
<td>CHILLED_MODE</td>
<td>0x10</td>
<td>Value of g_button_status at &quot;chilled mode&quot; button touch</td>
<td>touch.h</td>
</tr>
<tr>
<td>ECO_MODE</td>
<td>0x02</td>
<td>Value of g_button_status at &quot;eco mode&quot; button touch</td>
<td>touch.h</td>
</tr>
<tr>
<td>P_LED_A0</td>
<td>P2_bit.no6</td>
<td>Pins on the anode side of the matrix LED</td>
<td>led.c</td>
</tr>
<tr>
<td>P_LED_A1</td>
<td>P2_bit.no3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P_LED_A2</td>
<td>P2_bit.no1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P_LED_A3</td>
<td>P2_bit.no0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P_LED_C0</td>
<td>P12_bit.no0</td>
<td>Pins on the cathode side of the matrix LED</td>
<td>led.c</td>
</tr>
<tr>
<td>P_LED_C1</td>
<td>P12_bit.no1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P_LED_C2</td>
<td>P12_bit.no2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P_LED_C3</td>
<td>P14_bit.no6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P_LED_C4</td>
<td>P4_bit.no1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P_LED_C5</td>
<td>P6_bit.no1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PM_LED_A0</td>
<td>PM2_bit.no6</td>
<td>Port mode register on the anode side pins of the matrix LED</td>
<td>led.c</td>
</tr>
<tr>
<td>PM_LED_A1</td>
<td>PM2_bit.no3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PM_LED_A2</td>
<td>PM2_bit.no1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PM_LED_A3</td>
<td>PM2_bit.no0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PM_LED_C0</td>
<td>PM12_bit.no0</td>
<td>Port mode register on the cathode side pins of the matrix LED</td>
<td>led.c</td>
</tr>
<tr>
<td>PM_LED_C1</td>
<td>PM12_bit.no1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PM_LED_C2</td>
<td>PM12_bit.no2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PM_LED_C3</td>
<td>PM14_bit.no6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PM_LED_C4</td>
<td>PM4_bit.no1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PM_LED_C5</td>
<td>PM6_bit.no1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ECO_MODE_LED</td>
<td>P6_bit.no2</td>
<td>LED pin for eco mode</td>
<td>led.c</td>
</tr>
<tr>
<td>LED_ON</td>
<td>0U</td>
<td>LED turns on</td>
<td>led.c</td>
</tr>
<tr>
<td>LED_OFF</td>
<td>1U</td>
<td>LED turns off</td>
<td>led.c</td>
</tr>
<tr>
<td>OUTPUT</td>
<td>0U</td>
<td>Port mode register is set to output</td>
<td>led.c</td>
</tr>
<tr>
<td>INPUT</td>
<td>1U</td>
<td>Port mode register is set to input</td>
<td>led.c</td>
</tr>
<tr>
<td>COUNT_10S</td>
<td>10U</td>
<td>Constant for 10s count</td>
<td>led.c</td>
</tr>
<tr>
<td>COUNT_5S</td>
<td>5U</td>
<td>Constant for 5s count</td>
<td>led.c</td>
</tr>
</tbody>
</table>
### 3.4.6 Functions

The following shows the functions that are used in this sample program.

**Table 3-7 List of functions used in the sample code**

<table>
<thead>
<tr>
<th>Function name</th>
<th>Outline</th>
<th>Source file</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main</td>
<td>Main processing</td>
<td>main.c</td>
</tr>
<tr>
<td>r_led_init</td>
<td>Processing of LED initialization</td>
<td>led.c</td>
</tr>
<tr>
<td>r_ledport_input</td>
<td>Setting the LED port to input mode</td>
<td>led.c</td>
</tr>
<tr>
<td>r_ledport_output</td>
<td>Setting the LED port to output mode</td>
<td>led.c</td>
</tr>
<tr>
<td>r_led_turn_on_all_5s</td>
<td>Processing to turn on all LEDs for 5s</td>
<td>led.c</td>
</tr>
<tr>
<td>r_led_turn_on</td>
<td>Processing to turn on LEDs</td>
<td>led.c</td>
</tr>
<tr>
<td>r_led_turn_off</td>
<td>Processing to turn off LEDs</td>
<td>led.c</td>
</tr>
<tr>
<td>r_ledmatrix_turn_on</td>
<td>Processing to turn on the matrix LEDs</td>
<td>led.c</td>
</tr>
<tr>
<td>r_ledmatrix_turn_off</td>
<td>Processing to turn off the matrix LEDs</td>
<td>led.c</td>
</tr>
<tr>
<td>r_ledmatrix_turn_on_a</td>
<td>Processing to turn on the anode side of the matrix LED</td>
<td>led.c</td>
</tr>
<tr>
<td>r_change_led_position</td>
<td>Processing to change the position of the matrix LED</td>
<td>led.c</td>
</tr>
<tr>
<td>r_change_led</td>
<td>Processing to change the lighting pattern of the matrix LED</td>
<td>led.c</td>
</tr>
<tr>
<td>r_snooze_mode_init</td>
<td>Processing of initialization in SNOOZE mode</td>
<td>mode.c</td>
</tr>
<tr>
<td>r_normal_mode_init</td>
<td>Processing of initialization in normal mode</td>
<td>mode.c</td>
</tr>
<tr>
<td>r_snooze_mode</td>
<td>Processing of SNOOZE mode operation</td>
<td>mode.c</td>
</tr>
<tr>
<td>r_normal_mode</td>
<td>Processing of normal mode operation</td>
<td>mode.c</td>
</tr>
<tr>
<td>r_change_snooze_normal</td>
<td>Processing to change from SNOOZE mode to normal mode</td>
<td>mode.c</td>
</tr>
<tr>
<td>r_change_normal_snooze</td>
<td>Processing to change from normal mode to SNOOZE mode</td>
<td>mode.c</td>
</tr>
<tr>
<td>r_not_touched</td>
<td>Processing to determine that touch buttons are not touched</td>
<td>mode.c</td>
</tr>
<tr>
<td>r_touch_init</td>
<td>Initial settings of CTSU2La</td>
<td>touch.c</td>
</tr>
<tr>
<td>r_touch_main</td>
<td>Main operation when buttons are touched</td>
<td>touch.c</td>
</tr>
<tr>
<td>r_snooze_mode_touch_presses</td>
<td>Processing touch operation in SNOOZE mode</td>
<td>touch.c</td>
</tr>
<tr>
<td>r_change_eco_mode</td>
<td>Processing to change eco mode</td>
<td>touch.c</td>
</tr>
<tr>
<td>r_prevent_long_presses</td>
<td>Processing to prevent long presses</td>
<td>touch.c</td>
</tr>
<tr>
<td>r_touch_mec_scanstart</td>
<td>Switching ScanStart of MEC</td>
<td>touch.c</td>
</tr>
<tr>
<td>r_touch_mec_scantop</td>
<td>Switching ScanStop of MEC</td>
<td>touch.c</td>
</tr>
<tr>
<td>r_touch_mec_dataget</td>
<td>Switching DataGet of MEC</td>
<td>touch.c</td>
</tr>
<tr>
<td>r_Config_TAU0_0_interrupt</td>
<td>Interrupt function of TAU0_0</td>
<td>Config_TAU0_0_user.c</td>
</tr>
<tr>
<td>r_sec_count_timer_start</td>
<td>Start the timer for seconds count</td>
<td>Config_TAU0_0_user.c</td>
</tr>
<tr>
<td>r_sec_count_timer_reset</td>
<td>Reset the timer for seconds count</td>
<td>Config_TAU0_0_user.c</td>
</tr>
<tr>
<td>r_Config_TAU0_1_interrupt</td>
<td>Interrupt function of TAU0_1</td>
<td>Config_TAU0_1_user.c</td>
</tr>
<tr>
<td>r_ledmatrix_timer_start</td>
<td>Start the timer for matrix LED control</td>
<td>Config_TAU0_1_user.c</td>
</tr>
<tr>
<td>r_ledmatrix_timer_reset</td>
<td>Reset the timer for matrix LED control</td>
<td>Config_TAU0_1_user.c</td>
</tr>
</tbody>
</table>
### 3.4.7 Function Specifications

The following shows function specifications that are used in this sample program.

<table>
<thead>
<tr>
<th>Function name</th>
<th>main</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Outline</strong></td>
<td>Main processing</td>
</tr>
<tr>
<td><strong>Header</strong></td>
<td>r_smc_entry.h, touch.h, led.h</td>
</tr>
<tr>
<td><strong>Declaration</strong></td>
<td>int main (void)</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Initializes LEDs and CTSU2La and repeats touch operation.</td>
</tr>
<tr>
<td><strong>Arguments</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>Return value</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>Remarks</strong></td>
<td>None</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function name</th>
<th>r_led_init</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Outline</strong></td>
<td>Processing of LED initialization</td>
</tr>
<tr>
<td><strong>Header</strong></td>
<td>led.h</td>
</tr>
<tr>
<td><strong>Declaration</strong></td>
<td>void r_led_init(void)</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>When power is turned on, all LEDs are turned on for 5s and g_led_position is initialized.</td>
</tr>
<tr>
<td><strong>Arguments</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>Return value</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>Remarks</strong></td>
<td>None</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function name</th>
<th>r_ledport_input</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Outline</strong></td>
<td>Setting the LED port to input mode</td>
</tr>
<tr>
<td><strong>Header</strong></td>
<td>led.h</td>
</tr>
<tr>
<td><strong>Declaration</strong></td>
<td>void r_ledport_input(void)</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Changes the LED port to input mode.</td>
</tr>
<tr>
<td><strong>Arguments</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>Return value</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>Remarks</strong></td>
<td>None</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function name</th>
<th>r_ledport_output</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Outline</strong></td>
<td>Setting the LED port to output mode</td>
</tr>
<tr>
<td><strong>Header</strong></td>
<td>led.h</td>
</tr>
<tr>
<td><strong>Declaration</strong></td>
<td>void r_ledport_output(void)</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Changes the LED port to output mode.</td>
</tr>
<tr>
<td><strong>Arguments</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>Return value</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>Remarks</strong></td>
<td>None</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function name</th>
<th>r_led_turn_on_all_5s</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Outline</strong></td>
<td>Processing to turn on all LEDs for 5s</td>
</tr>
<tr>
<td><strong>Header</strong></td>
<td>led.h</td>
</tr>
<tr>
<td><strong>Declaration</strong></td>
<td>void r_led_turn_on_all_5s(void)</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Turns on all LEDs for 5s.</td>
</tr>
<tr>
<td><strong>Arguments</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>Return value</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>Remarks</strong></td>
<td>None</td>
</tr>
</tbody>
</table>
### r_led_turn_on

**Outline**  
Processing to turn on LEDs

**Header**  
led.h

**Declaration**  
void r_led_turn_on(void)

**Description**  
Turns on LEDs.

**Arguments**  
None

**Return value**  
None

**Remarks**  
None

### r_led_turn_off

**Outline**  
Processing to turn off LEDs

**Header**  
led.h

**Declaration**  
void r_led_turn_off(void)

**Description**  
Turns off LEDs and resets the timer.

**Arguments**  
None

**Return value**  
None

**Remarks**  
None

### r_ledmatrix_turn_on

**Outline**  
Processing to turn on the matrix LEDs

**Header**  
led.h

**Declaration**  
void r_ledmatrix_turn_on(void)

**Description**  
Turns on the matrix LEDs.

**Arguments**  
None

**Return value**  
None

**Remarks**  
None

### r_ledmatrix_turn_off

**Outline**  
Processing to turn off the matrix LEDs

**Header**  
led.h

**Declaration**  
void r_ledmatrix_turn_off(void)

**Description**  
Turns off the matrix LEDs.

**Arguments**  
None

**Return value**  
None

**Remarks**  
None

### r_ledmatrix_turn_on_a

**Outline**  
Processing to turn on the anode side of the matrix LED

**Header**  
led.h

**Declaration**  
void r_ledmatrix_turn_on_a(void)

**Description**  
Controls the lighting of the anode side of the matrix LED.

**Arguments**  
None

**Return value**  
None

**Remarks**  
None
### Function: r_change_led_position

**Outline:** Processing to change the position of the matrix LED  
**Header:** led.h  
**Declaration:** void r_change_led_position(uint8_t *pos, uint8_t status)  
**Description:** Changes matrix LED position.  
**Arguments:** * pos, status  
**Return value:** None  
**Remarks:** None

### Function: r_change_led

**Outline:** Processing to change the lighting pattern of the matrix LED  
**Header:** led.h  
**Declaration:** void r_change_led(void)  
**Description:** Changes the lighting pattern of the matrix LED.  
**Arguments:** None  
**Return value:** None  
**Remarks:** None

### Function: r_snooze_mode_init

**Outline:** Processing of initialization in SNOOZE mode  
**Header:** mode.h  
**Declaration:** void r_snooze_mode_init(void)  
**Description:** Processes the initialization of SNOOZE mode.  
**Arguments:** None  
**Return value:** None  
**Remarks:** None

### Function: r_normal_mode_init

**Outline:** Processing of initialization in normal mode  
**Header:** mode.h  
**Declaration:** void r_normal_mode_init(void)  
**Description:** Processes the initialization of normal mode.  
**Arguments:** None  
**Return value:** None  
**Remarks:** None
[Function name] r_snooze_mode
Outline   Processing of SNOOZE mode operation
Header    mode.h
Declaration void r_snooze_mode(void)
Description Processes during SNOOZE mode.
Arguments None
Return value None
Remarks None

[Function name] r_normal_mode
Outline   Processing of normal mode operation
Header    mode.h
Declaration void r_normal_mode(void)
Description Processes during normal mode.
Arguments None
Return value None
Remarks None

[Function name] r_change_snooze_normal
Outline   Processing to change from SNOOZE mode to normal mode
Header    mode.h
Declaration void r_change_snooze_normal(void)
Description Changes from SNOOZE mode to normal mode.
Arguments None
Return value None
Remarks None

[Function name] r_change_normal_snooze
Outline   Processing to change from normal mode to SNOOZE mode
Header    mode.h
Declaration void r_change_normal_snooze(void)
Description Changes from normal mode to SNOOZE mode.
Arguments None
Return value None
Remarks None

[Function name] r_not_touched
Outline   Processing to determine that touch buttons are not touched
Header    mode.h
Declaration void r_not_touched(void)
Description The process of determining that touch buttons are touched or not.
Arguments None
Return value None
Remarks None
### r_touch_init

**Outline**
Initial settings of CTSU2La

**Declaration**
void r_touch_init(void)

**Description**
Sets the initial settings for CTSU2La.

**Arguments**
None

**Return value**
None

**Remarks**
None

### r_touch_main

**Outline**
Main operation when buttons are touched

**Declaration**
void touch_main(void)

**Description**
Executes the main process when the button is touched.

**Arguments**
None

**Return value**
None

**Remarks**
None

### r_snooze_mode_touch_processes

**Outline**
Processing touch operation in SNOOZE mode

**Declaration**
void r_snooze_mode_touch_processes(void)

**Description**
Processes touch operation in SNOOZE mode.

**Arguments**
None

**Return value**
None

**Remarks**
None

### r_change_eco_mode

**Outline**
Processing to change eco mode

**Declaration**
void r_change_eco_mode(void)

**Description**
Changes to eco-mode.

**Arguments**
None

**Return value**
None

**Remarks**
None

### r_prevent_long_presses

**Outline**
Processing to prevent long presses

**Declaration**
void r_prevent_long_presses(uint64_t p_button_status)

**Description**
Processes to prevent button long pressed.

**Arguments**

p_button_status

**Return value**
None

**Remarks**
None
### Function: r_touch_mec_scanstart

<table>
<thead>
<tr>
<th>Outline</th>
<th>Switching ScanStart of MEC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>touch.h</td>
</tr>
<tr>
<td>Declaration</td>
<td>fsp_err_t r_touch_mec_scanstart(void)</td>
</tr>
<tr>
<td>Description</td>
<td>Processes switching of ScanStart of MEC.</td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
<tr>
<td>Return value</td>
<td>Err</td>
</tr>
<tr>
<td>Remarks</td>
<td>None</td>
</tr>
</tbody>
</table>

### Function: r_touch_mec_scanstop

<table>
<thead>
<tr>
<th>Outline</th>
<th>Switching ScanStop of MEC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>touch.h</td>
</tr>
<tr>
<td>Declaration</td>
<td>fsp_err_t r_touch_mec_scanstop(void)</td>
</tr>
<tr>
<td>Description</td>
<td>Processes switching of ScanStop of MEC.</td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
<tr>
<td>Return value</td>
<td>Err</td>
</tr>
<tr>
<td>Remarks</td>
<td>None</td>
</tr>
</tbody>
</table>

### Function: r_touch_mec_dataget

<table>
<thead>
<tr>
<th>Outline</th>
<th>Switching DataGet of MEC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>touch.h</td>
</tr>
<tr>
<td>Declaration</td>
<td>fsp_err_t r_touch_mec_dataget(void)</td>
</tr>
<tr>
<td>Description</td>
<td>Processes switching of DataGet of MEC.</td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
<tr>
<td>Return value</td>
<td>Err</td>
</tr>
<tr>
<td>Remarks</td>
<td>None</td>
</tr>
</tbody>
</table>

### Function: Config_TAU0_0_interrupt

<table>
<thead>
<tr>
<th>Outline</th>
<th>Interrupt function of TAU0_0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>Config_TAU0_0.h</td>
</tr>
<tr>
<td>Declaration</td>
<td>static void __near r_Config_TAU0_0__interrupt(void)</td>
</tr>
<tr>
<td>Description</td>
<td>Counts the timer for seconds count.</td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
<tr>
<td>Return value</td>
<td>None</td>
</tr>
<tr>
<td>Remarks</td>
<td>None</td>
</tr>
</tbody>
</table>

### Function: r_sec_count_timer_start

<table>
<thead>
<tr>
<th>Outline</th>
<th>Start the timer for seconds count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>Config_TAU0_0.h</td>
</tr>
<tr>
<td>Declaration</td>
<td>void r_sec_count_timer_start(void)</td>
</tr>
<tr>
<td>Description</td>
<td>Starts the timer for seconds count.</td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
<tr>
<td>Return value</td>
<td>None</td>
</tr>
<tr>
<td>Remarks</td>
<td>None</td>
</tr>
</tbody>
</table>
### r_sec_count_timer_reset

**Outline**

Reset the timer for seconds count

**Header**

Config_TAU0_0.h

**Declaration**

void r_sec_count_timer_reset(uint8_t flg)

**Description**

Resets the timer for seconds count.

**Arguments**

flg

**Return value**

None

**Remarks**

None

---

### r_Config_TAU0_1_interrupt

**Outline**

Interrupt function of TAU0_1

**Header**

Config_TAU0_1.h

**Declaration**

static void __near r_Config_TAU0_1_interrupt(void)

**Description**

Turns on the matrix LEDs.

**Arguments**

None

**Return value**

None

**Remarks**

None

---

### r_ledmatrix_timer_start

**Outline**

Start processing of the timer for matrix LED control

**Header**

Config_TAU0_1.h

**Declaration**

void r_ledmatrix_timer_start(void)

**Description**

Starts the timer for matrix LED control.

**Arguments**

None

**Return value**

None

**Remarks**

None

---

### r_ledmatrix_timer_reset

**Outline**

Reset processing of the timer for matrix LED control

**Header**

Config_TAU0_1.h

**Declaration**

void r_ledmatrix_timer_reset(void)

**Description**

Resets the timer for matrix LED control.

**Arguments**

None

**Return value**

None

**Remarks**

None
3.4.8 Flowchart
3.4.8.1 Flowchart of main function
The flowchart of main function is shown below.

![Flowchart of main function]

Figure 3-3 Flowchart of main function

3.4.8.2 Flowchart of r_touch_init function
The flowchart of r_touch_init function is shown below.

![Flowchart of r_touch_init function]

Figure 3-4 Flowchart of r_touch_init function
3.4.8.3 Flowchart of r_touch_main function

The flowchart of r_touch_main function is shown below.

```
Figure 3-5 Flowchart of r_touch_main function
```

```
Process during SNOOZE mode
r_snooze_mode()

Process change from SNOOZE mode to normal mode
r_change_snooze_normal()

Process during normal mode
r_normal_mode()

Reset the timer
r_sec_count_timer_reset()

Process to turn off LED
r_led_turn_off()

Process change from normal mode to SNOOZE mode
r_change_normal_snooze()

return
```
3.4.8.4 Flowchart of `r_snooze_mode_touch_processes` function

The flowchart of `r_snooze_mode_touch_processes` function is shown below.

![Flowchart of `r_snooze_mode_touch_processes` function](image)

3.4.8.5 Flowchart of `r_change_eco_mode` function

The flowchart of `r_change_eco_mode` function is shown below.

![Flowchart of `r_change_eco_mode` function](image)
3.4.8.6 Flowchart of r_prevent_long_presses function

The flowchart of r_prevent_long_presses function is shown below.

```
return
r_prevent_long_presses

0x00 != button_status
```

Yes

No

```
g_touch_button_flg = 0U
```

return

---

Figure 3-8 flowchart of r_prevent_long_presses function

3.4.8.7 Flowchart of r_snooze_mode_init function

The flowchart of r_snooze_mode_init function is shown below.

```
return
r_snooze_mode_init()

1U != g_snooze_mode_init
```

Yes

No

```
g_snooze_mode_init = 0U
```

```
RM_TOUCH_ScanStop()
```

```
r_touch_mec_scanstart()
```

```
INTCTSUFN enable
CTSUFNMK = 0U
```

return

---

Figure 3-9 Flowchart of r_snooze_mode_init function
3.4.8.8 Flowchart of r_snooze_mode function

The flowchart of r_snooze_mode function is shown below.

![Flowchart of r_snooze_mode function](image)

**Figure 3-10 Flowchart of r_snooze_mode function**
3.4.8.9 Flowchart of r_touch_mec_scanstart function

The flowchart of r_touch_mec_scanstart function is shown below.

```
return
```

```
r_touch_mec_scanstart
```

```
g_led_position[0]
```

```
Case 0  Case 1
```

```
Config03(Proximity sensor)  Config02(Touch button)
RM_TOUCH_ScanStart()  RM TOUCH_ScanStart()
```

```
return
```

Figure 3-11 Flowchart of r_touch_mec_scanstart function

3.4.8.10 Flowchart of r_touch_mec_scanstop function

The flowchart of r_touch_mec_scanstop function is shown below.

```
return
```

```
r_touch_mec_scanstop
```

```
g_led_position[0]
```

```
Case 0  Case 1
```

```
Config03(Proximity sensor)  Config02(Touch button)
RM_TOUCH_ScanStop()  RM TOUCH_ScanStop()
```

```
return
```

Figure 3-12 Flowchart of r_touch_mec_scanstop function
3.4.8.11 Flowchart of r_touch_mec_dataget function

The flowchart of r_touch_mec_dataget function is shown below.

```
return r_touch_mec_dataget

if g_led_position[0] == 0:
    Case 0
    Config03 (Proximity sensor)
    RM_TOUCH_DataGet()

else:
    Case 1
    Config02 (Touch button)
    RM_TOUCH_DataGet()

return
```

Figure 3-13 Flowchart of r_touch_mec_dataget function
3.4.8.12 Flowchart of r_nomal_mode_init function

The flowchart of r_nomal_mode_init function is shown below.

![Flowchart of r_nomal_mode_init function](image)

Figure 3-14 Flowchart of r_nomal_mode_init function
### 3.4.8.13 Flowchart of r_normal_mode function

The flowchart of r_normal_mode function is shown below.

![Flowchart of r_normal_mode function](image)

**Figure 3-15 Flowchart of r_normal_mode function**
3.4.8.14 Flowchart of r_change_snooze_nomal function

The flowchart of r_change_snooze_nomal function is shown below.

![Flowchart of r_change_snooze_nomal function](image)

3.4.8.15 Flowchart of r_change_nomal_snooze function

The flowchart of r_change_nomal_snooze function is shown below.

![Flowchart of r_change_nomal_snooze function](image)
3.4.8.16 Flowchart of r_not_touched function

The flowchart of r_not_touched function is shown below.

![Flowchart of r_not_touched function](image)

3.4.8.17 Flowchart of r_ledport_input function

The flowchart of r_ledport_input function is shown below.

![Flowchart of r_ledport_input function](image)
3.4.8.18 Flowchart of r_ledport_output function
The flowchart of r_ledport_output function is shown below.

```
return r_ledport_output

Change pin mode to output

return
```

Figure 3-20 Flowchart of r_ledport_output function

3.4.8.19 Flowchart of r_led_init function
The flowchart of r_led_init function is shown below.

```
i = 0; i < 6; i++

Yes

led_position[i] = 1U

No

Turn on all LEDs for 5s
r_led_turn_on_all_5s()

return
```

Figure 3-21 Flowchart of r_led_init function
### 3.4.8.20 Flowchart of r_led_turn_on_all_5s function

The flowchart of `r_led_turn_on_all_5s` function is shown below.

![Flowchart](image)

**Figure 3-22 Flowchart of r_led_turn_on_all_5s function**
3.4.8.21 Flowchart of \texttt{r\_change\_led} function

The flowchart of \texttt{r\_change\_led} function is shown below.

\begin{center}
\includegraphics[width=0.8\textwidth]{flowchart.png}
\end{center}

\textit{Figure 3-23 Flowchart of \texttt{r\_change\_led} function}
3.4.8.22 Flowchart of r_change_led_position function

The flowchart of r_change_led_position function is shown below.

![Flowchart of r_change_led_position function](image-url)
3.4.8.23 Flowchart of r_led_turn_on function

The flowchart of r_led_turn_on function is shown below.

![Flowchart of r_led_turn_on function](image)

Figure 3-25 Flowchart of r_led_turn_on function
3.4.8.24 Flowchart of r_led_turn_off function
The flowchart of r_led_turn_off function is shown below.

![Flowchart of r_led_turn_off function]

Figure 3-26 Flowchart of r_led_turn_off function
3.4.8.25 Flowchart of r_ledmatrix_turn_on function

The flowchart of r_ledmatrix_turn_on function is shown below.

![Flowchart of r_ledmatrix_turn_on function](image-url)

**Figure 3-27 Flowchart of r_ledmatrix_turn_on function**
3.4.8.26 Flowchart of r_ledmatrix_turn_off function

The flowchart of r_ledmatrix_turn_off function is shown below.

```
r_ledmatrix_turn_off

r_ledmatrix_timer_reset()

r_ledport_output()

Turn off LED output

return
```

Figure 3-28 Flowchart of r_ledmatrix_turn_off function
3.4.8.27 Flowchart of r_ledmatrix_turn_on_a function

The flowchart of r_ledmatrix_turn_on_a function is shown below.

![Flowchart of r_ledmatrix_turn_on_a function](image-url)

Figure 3-29 Flowchart of r_ledmatrix_turn_on_a function
3.4.8.28 Flowchart of r_Config_TAU0_0_interrupt function

The flowchart of r_Config_TAU0_0_interrupt function is shown below.

![Flowchart of r_Config_TAU0_0_interrupt function](image)

Figure 3-30 Flowchart of r_Config_TAU0_0_interrupt function
3.4.8.29 Flowchart of r_sec_count_timer_start function

The flowchart of r_sec_count_timer_start function is shown below.

![Flowchart of r_sec_count_timer_start function](image)

Figure 3-31 Flowchart of r_sec_count_timer_start function
3.4.8.30 Flowchart of r_sec_count_timer_reset function
The flowchart of r_sec_count_timer_reset function is shown below.

![Flowchart of r_sec_count_timer_reset function](image)

Figure 3-32 Flowchart of r_sec_count_timer_reset function

3.4.8.31 Flowchart of r_Config_TAU0_1_interrupt function
The flowchart of r_Config_TAU0_1_interrupt function is shown below.

![Flowchart of r_Config_TAU0_1_interrupt function](image)

Figure 3-33 Flowchart of r_Config_TAU0_1_interrupt function
3.4.8.32 Flowchart of r_ledmatrix_timer_start function

The flowchart of r_ledmatrix_timer_start function is shown below.

![Flowchart of r_ledmatrix_timer_start function](image)

3.4.8.33 Flowchart of r_ledmatrix_timer_reset function

The flowchart of r_ledmatrix_timer_reset function is shown below.

![Flowchart of r_ledmatrix_timer_reset function](image)
4. Importing a Project

The sample programs are distributed in e² studio project format. This section shows how to import a project into e² studio or CS+. After importing a project, check the build and debug settings.

4.1 Procedure in e² studio

To use sample programs in e² studio, follow the steps below to import them into e² studio. In projects managed by e² studio, do not use space codes, multibyte characters, and symbols such as "$", "#", "%" in folder names or paths to them.

(Note that depending on the version of e² studio you are using, the interface may appear somewhat different from the screenshots below.)

![Image of import process]

- Start the e² studio and select the File >> [Import ...].
- Select [Existing Projects into Workspace].
- Select [Select root directory:].
- Select the directory which stored the project. (e.g. r01an6740_g22demo_touch_mec)
- Each application note has its own project name.
- Select [Copy projects into workspace(C)] when to copy project to workspace.
- Select [Add project to working sets] when using the working sets.

Figure 4-1 Import a Project into e² studio
4.2 Procedure in CS+

To use sample programs in CS+, follow the steps below to import them into CS+. In projects managed by CS+, do not use space codes, multibyte characters, and symbols such as "$", "#", "%" in folder names or paths to them.

(Note that depending on the version of CS+ you are using, the interface may appear somewhat different from the screenshots below.)

Figure 4-2 Import a Project into CS+
5. Starting a Demonstration

When the E2 emulator Lite connector is disconnected and RL78/G22 PoC power is turned on, the demonstration program will start. This demonstration program assumes control of the display and settings of the refrigerator panel. The display settings of the refrigerator panel are configured with the touch buttons checking the setting display.

Hereafter, touch buttons are referred to as buttons.

![Demonstration operation panel](image)

**Figure 5-1 Demonstration operation panel**
5.1 Powered on RL78/G22 PoC and menu screen
When RL78/G22 PoC is powered on, all characters on the touch panel are displayed for approximately 5 seconds. After the display finishes, the demonstration program starts and RL78/G22 PoC transits the standby mode (operation mode1).

5.2 Return from standby mode
Touching within the white frame returns from standby mode. Each setting value indicates the center value such as 2 or Mid.
5.3  Touch operation

5.3.1  Set operation mode
By touching the operation mode button, the setting values can be changed in the order shown in Figure 5-4.

![Figure 5-4 Set operation mode](image)

5.3.2  Set freezing
By touching the freezing button, the setting values can be changed in the order shown in Figure 5-5.

![Figure 5-5 Set freezing](image)

5.3.3  Set refrigerator
By touching the refrigerator button, the setting values can be changed in the order shown in Figure 5-6.

![Figure 5-6 Set refrigerator](image)
5.3.4 Set ice making
By touching the ice making button, the setting values can be changed in the order shown in Figure 5-7.

![Figure 5-7 Set ice making](image)

5.3.5 Set cooling mode
By touching the cooling mode button, the setting values can be changed in the order shown in Figure 5-8.

![Figure 5-8 Set cooling ice making](image)

5.3.6 Set chilled mode
By touching the chilled mode button, the setting values can be changed in the order shown in Figure 5-9.

![Figure 5-9 Set chilled mode](image)
5.3.7 eco mode (proximity sensor mode)
Touching the eco mode button when the operating mode is set to 1, the device transits the standby mode in the proximity sensor mode. In proximity sensor mode, holding the hand within the white frame returns to normal mode.

![Diagram](image1)

Figure 5-10 When operating mode 1 is set, touch eco mode

![Diagram](image2)

Figure 5-11 Return from standby mode in proximity sensor mode
5.3.8 eco mode (touch sensor mode)

Touching the eco mode button when the operating mode is set to 2, the device transits the standby mode in the touch sensor mode. In touch sensor mode, touching the button in the white frame returns to normal mode.

![Diagram showing eco mode button and modes](image)

Figure 5-12 When operating mode 2 is set, touch eco mode

![Diagram showing touch within white frame](image)

Figure 5-13 Return from standby mode in touch sensor mode
6. Reference Documents

- RL78/G22 User’s Manual: Hardware (R01UH0978)
- RL78 Family User’s Manual: Software (R01US0015)
  (The latest version can be downloaded from the Renesas Electronics website.)
- Technical Update / Technical News
  (The latest version can be downloaded from the Renesas Electronics website.)
- User’s Manual : Development Environment
  (The latest version can be downloaded from the Renesas Electronics website.)
- User’s Manual : RL78/G22 Capacitive Touch Evaluation System (RTK0EG0042S01001BJ)
  (The latest version can be downloaded from the Renesas Electronics website.)
- Application Note RL78 Family
  Capacitive Touch Sensing Unit (CTSU2L) Operation Explanation (R01AN5744)
- Application Note RL78 Family
  Using QE and SIS to Develop Capacitive Touch Applications (R01AN5512)
- Application Note RL78 Family CTSU Module Software Integration System (R11AN0484)
- Application Note RL78 Family TOUCH Module Software Integration System (R11AN0485)
- Application Note CTSU Capacitive Touch Electrode Design Guide (R30AN0389)
  (The latest version can be downloaded from the Renesas Electronics website.)

Website and Support

Renesas Electronics website
  http://www.renesas.com/

Capacitive Sensor Unit related Pages
  https://www.renesas.com/solutions/touch-key
  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>Feb.13.23</td>
<td></td>
<td></td>
<td>First edition</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 $V_{IL}$ (Max.) and $V_{IH}$ (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between $V_{IL}$ (Max.) and $V_{IH}$ (Min.).

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

8. Differences between products
   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.
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 the 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. In case a 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, 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 systems; underwater repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user's manual or other Renesas Electronics document.

7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION ("Vulnerability Issues"). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.

8. When using Renesas Electronics products, refer to the latest product information (data sheets, user's manuals, application notes, "General Notes for Handling and Using Semiconductor Devices" in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.

9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.

10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.

11. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.

12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.

13. This document shall not be reproduced, reprinted 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)