

RL78/G22

Sample S/W for home appliance panel UI demo using MEC function

Introduction

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

The Multiple Electrode Connection (MEC) 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
(RTK0EG0042S01001BJ)

Contents

1.	Outline	5
1.1	Multiple Electrode Connection (MEC) function	5
1.1.1	Advantage 1 of MEC Function: Return from Standby Mode by Touching Any Electrode.....	5
1.1.2	Advantage 2 of MEC Function: Available as a Proximity Sensor	6
1.1.3	Advantage 3 of MEC Function: Low Power Consumption	6
1.2	How to Utilize the MEC Function in Appliance UI Demo.....	7
2.	Appliance UI Demo Hardware Overview	8
2.1	External View of the Enclosure	8
2.2	Appliance UI Demo Configuration	9
2.3	RL78/G22 CPU board Configuration (Connection of external trigger)	10
3.	Operation Confirmation Conditions	11
4.	Sample Programs	12
4.1	State Transition of Demonstration Screen	13
4.2	Overall Flowchart.....	14
4.3	Pins Used	15
4.4	Setting of Unused Pins	16
4.5	Sample Programs Structure	18
4.5.1	Peripheral Functions Used	18
4.5.2	Peripheral Function Settings	19
4.5.3	Capacitive Touch Settings.....	22
4.5.3.1	Touch Interface Configuration	22
4.5.3.2	Configuration (methods) Settings.....	22
4.5.3.3	Tuning results	23
4.5.4	Setting of Option Byte	23
4.5.5	File Structure	24
4.5.6	Variables.....	25
4.5.7	Constants	26
4.5.8	Functions	27
4.5.9	Function Specifications	29
4.5.10	Flowchart	37
4.5.10.1	Flowchart of Main Function	37
4.5.10.2	Flowchart of r_touch_init Function	38
4.5.10.3	Flowchart of r_sms_init Function	39
4.5.10.4	Flowchart of r_touch_main Function	40
4.5.10.5	Flowchart of r_snooze_mode_touch_prosses Function	41
4.5.10.6	Flowchart of r_change_eco_mode Function	41
4.5.10.7	Flowchart of r_prevent_long_presses Function	42
4.5.10.8	Flowchart of r_snooze_mode_init_cpu Function	42

4.5.10.9	Flowchart of r_snooze_mode_init_sms Function.....	43
4.5.10.10	Flowchart of r_snooze_mode Function.....	44
4.5.10.11	Flowchart of r_snooze_cpu Function.....	45
4.5.10.12	Flowchart of r_snooze_sms Function	46
4.5.10.13	Flowchart of r_touch_mec_scanstart_cpu Function	47
4.5.10.14	Flowchart of r_touch_mec_scanstop Function	47
4.5.10.15	Flowchart of r_touch_mec_dataget_cpu Function.....	48
4.5.10.16	Flowchart of r_sms_trigger_start Function	48
4.5.10.17	Flowchart of r_sms_trigger_stop Function.....	49
4.5.10.18	Flowchart of r_nomal_mode_init Function.....	50
4.5.10.19	Flowchart of r_nomal_mode Function	51
4.5.10.20	Flowchart of r_change_snooze_nomal Function	52
4.5.10.21	Flowchart of r_change_nomal_snooze Function	52
4.5.10.22	Flowchart of r_not_touched Function	53
4.5.10.23	Flowchart of r_ledport_input Function	53
4.5.10.24	Flowchart of r_ledport_output Function	54
4.5.10.25	Flowchart of r_led_init Function	54
4.5.10.26	Flowchart of r_led_turn_on_all_5s Function.....	55
4.5.10.27	Flowchart of r_change_led Function.....	56
4.5.10.28	Flowchart of r_change_led_position Function	57
4.5.10.29	Flowchart of r_led_turn_on Function	58
4.5.10.30	Flowchart of r_led_turn_off Function	59
4.5.10.31	Flowchart of r_ledmatrix_turn_on Function	60
4.5.10.32	Flowchart of r_ledmatrix_turn_off Function	61
4.5.10.33	Flowchart of r_ledmatrix_turn_on_a Function	62
4.5.10.34	Flowchart of r_Config_TAU0_0_interrupt Function	63
4.5.10.35	Flowchart of r_sec_count_timer_start Function.....	63
4.5.10.36	Flowchart of r_sec_count_timer_reset Function.....	64
4.5.10.37	Flowchart of r_Config_TAU0_1_interrupt Function	64
4.5.10.38	Flowchart of r_ledmatrix_timer_start Function.....	65
4.5.10.39	Flowchart of r_ledmatrix_timer_reset Function.....	65
5.	Importing a Project.....	66
5.1	Procedure in e ² studio	66
5.2	Procedure in CS+	67
6.	Starting a Demonstration	68
6.1	Powered on Appliance UI Demo and Menu Screen.....	69
6.2	Return from Standby Mode	69
6.3	Touch Operation	70
6.3.1	Set operation mode	70
6.3.2	Set freezing	70

6.3.3	Set refrigerator.....	70
6.3.4	Set ice making.....	71
6.3.5	Set cooling mode.....	71
6.3.6	Set chilled mode.....	71
6.3.7	eco mode (Proximity Sensor Mode).....	72
6.3.8	eco mode (Touch Sensor Mode).....	73
6.3.9	eco mode (Auto Judgment (using SMS) Mode).....	74
7.	How to Measure Current Consumption.....	75
7.1	Environment to Measure Current Consumption.....	75
7.2	Equipment and Software.....	75
7.3	How to Connect the Target Board and Each Equipment.....	76
7.4	RL78/G22 CPU Board Settings.....	77
7.5	Settings of Current Measuring Software.....	78
8.	Current Consumption Measurement Result.....	79
8.1	Current Consumption.....	79
8.2	Current Consumption Waveform.....	81
9.	Design Documents for Electrode Board.....	84
9.1	Schematic.....	84
9.2	Components Placement.....	85
9.3	Components List.....	86
10.	Reference Documents.....	87
	Revision History.....	88

1. Outline

This document describes the application example of the Multiple Electrode Connection (MEC) function, using the Appliance UI Demo with a refrigerator panel motif. In addition, this application note also shows the reference data of the current consumption when Appliance UI Demo is performed.

Appliance UI Demo 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 Multiple Electrode Connection (MEC) function

The Multiple Electrode Connection (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 Appliance UI Demo, six touch electrodes are used as one electrode as shown in Figure 1-1 by using the MEC function during standby mode. This allows the user to return from standby mode by touching any electrode within the white frame.

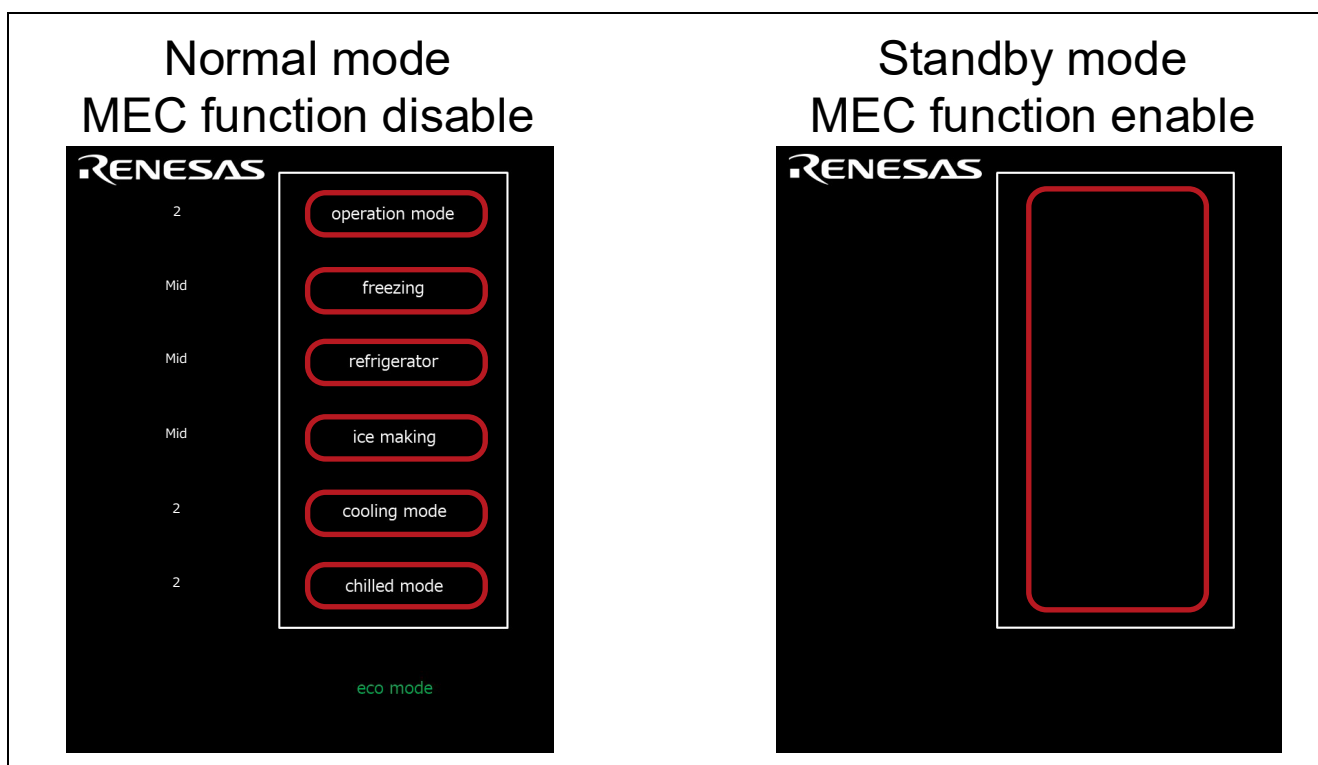


Figure 1-1 MEC function (one electrode)

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. The detection distance of the proximity sensor is approximately 20 mm.

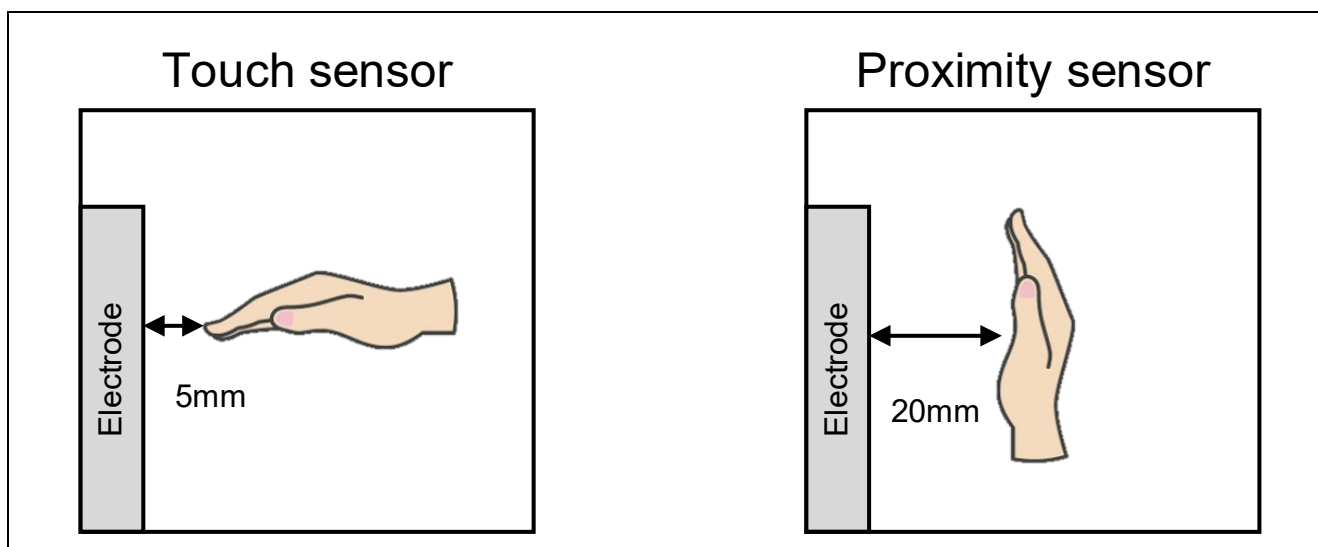


Figure 1-2 MEC function (proximity sensor)

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. Compared to when the MEC function is not used, the measurement time of the electrode can be reduced, enabling operation with low power consumption.

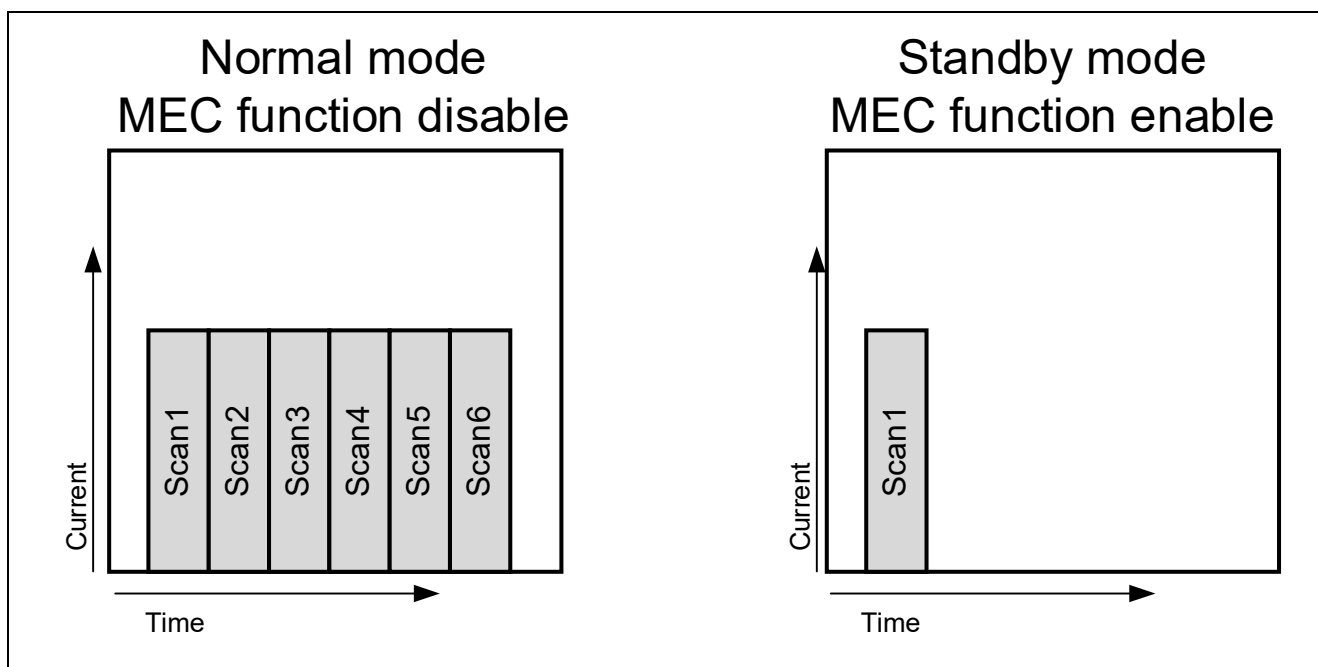


Figure 1-3 MEC function (low power consumption)

1.2 How to Utilize the MEC Function in Appliance UI Demo

Appliance UI Demo implements a variation of standby mode operation using the MEC function as operation mode.

Operation mode 1 operates as a proximity sensor during standby mode. In this mode, 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. CPU will make the decision to return from standby mode.

Operation mode 2 operates as a touch sensor during standby mode. In this mode, the touch threshold of the buttons is set to detect a direct hand touch. Touching any buttons returns from standby mode. CPU will make the decision to return from standby mode.

Operation mode 3 performs auto judgment measurements using SMS and operates as a touch sensor. The touch threshold of the touch button is set in the same way as in operation mode2. In this mode, uses SMS to determine whether to return from standby mode. This reduces power consumption during standby time.

2. Appliance UI Demo Hardware Overview

2.1 External View of the Enclosure

The following shows an external view of Appliance UI Demo.

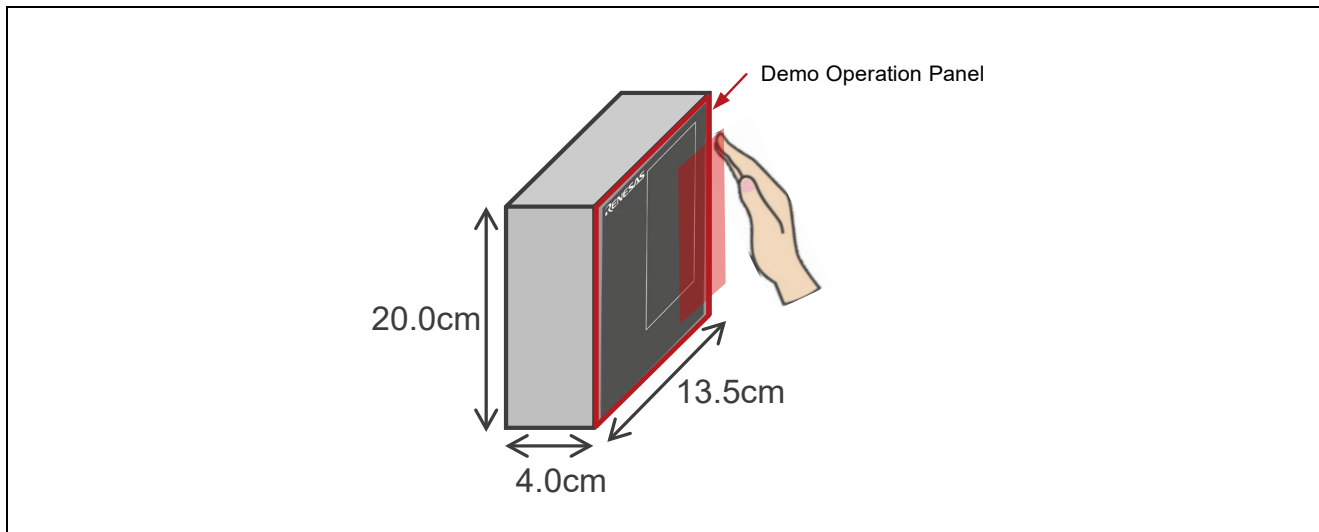


Figure 2-1 External view of Appliance UI Demo

The following figure shows the appearance of the Appliance UI Demo Operation Panel. Yellow areas in the figure indicate the position and size of the touch sensor electrodes. The size of each electrode is 50 × 15 mm. The electrodes are not visible from the actual appearance.

The front of the demo operation panel consists of a 135 × 200 × 2 mm acrylic plate. The surface of the operation panel is processed using blackout printing and silk printing.

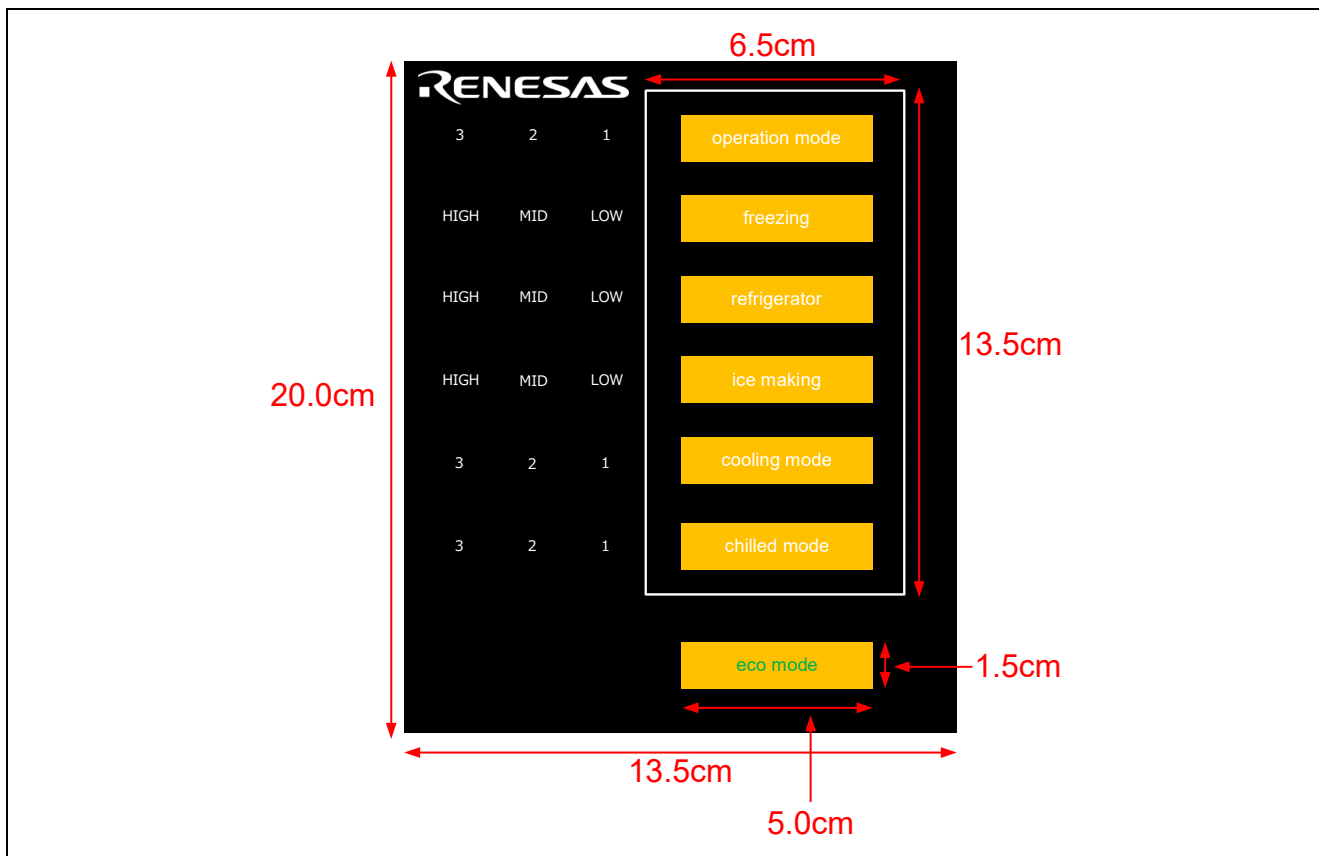


Figure 2-2 External view of Demo Operation Panel and electrode size

2.2 Appliance UI Demo Configuration

Appliance UI Demo consists of the CPU Board (RTK0EG0041C01001BJ) of RL78/G22 Capacitive Touch Evaluation System (RTK0EG0042S01001BJ) and an electrode board. For details of the electrode board, refer to Chapter 9 Design Documents for Electrode Board onward.

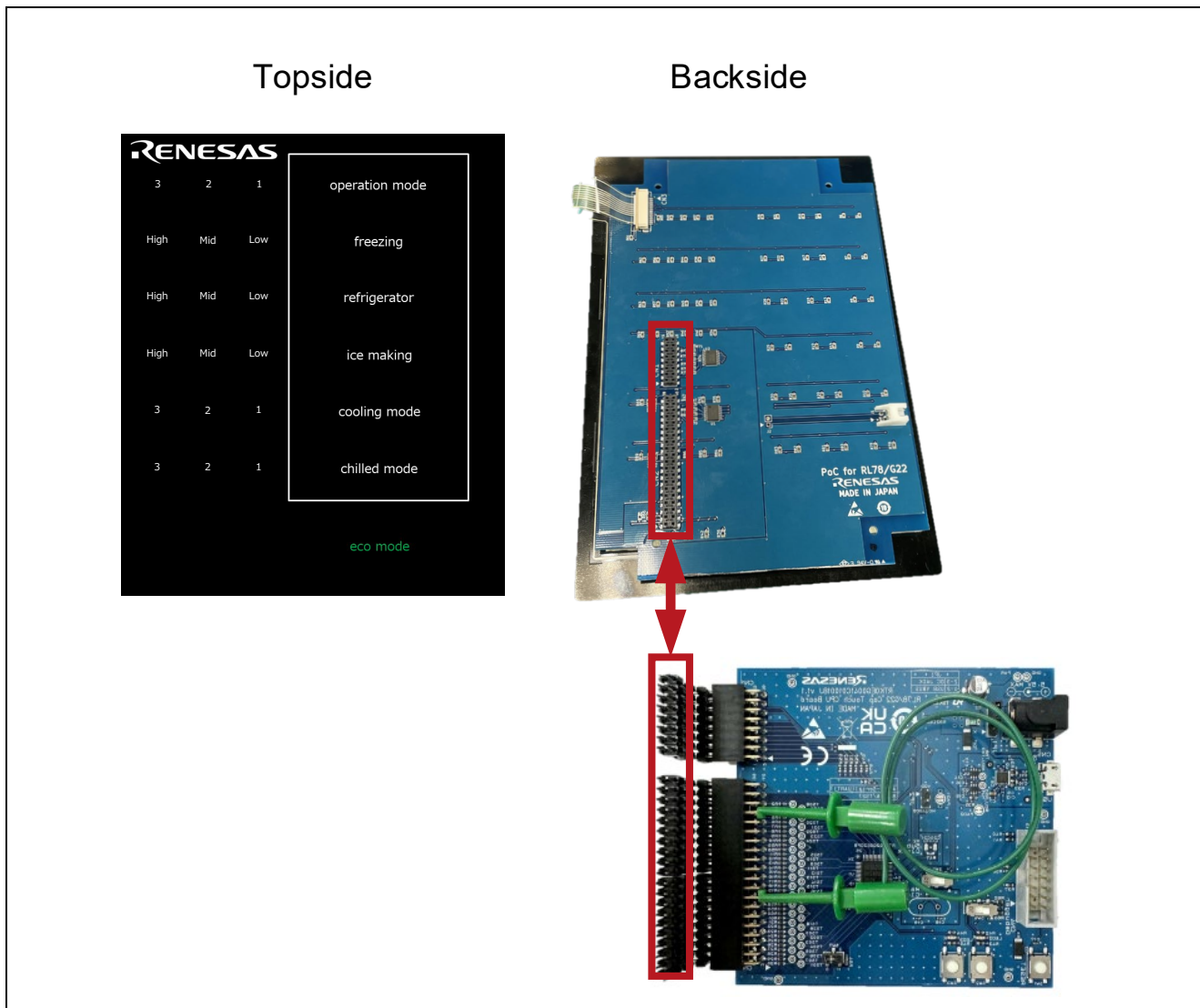


Figure 2-3 Appliance UI Demo board configuration diagram

2.3 RL78/G22 CPU board Configuration (Connection of external trigger)

To perform auto judgment measurement using SMS with RL78/G22, please make the following settings.
Connect the CN2 34th pin (P130/TS19) and 16th pin (P16/INTP5/TS17) on the MCU board side.

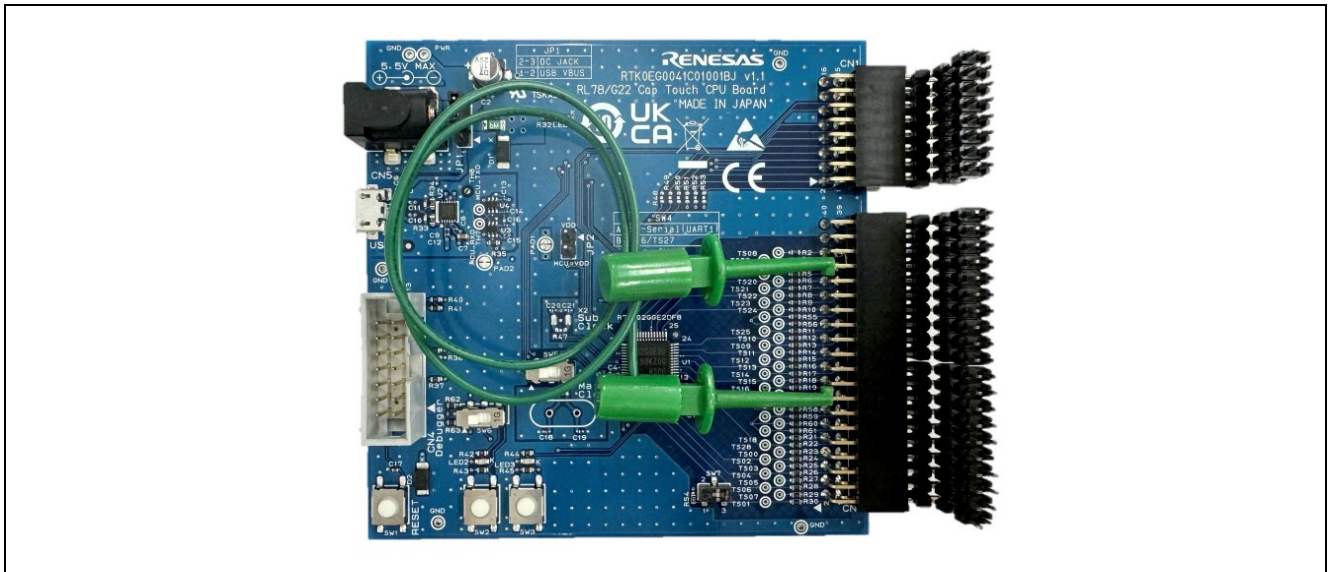


Figure 2-4 Connection of external trigger on RL78/G22 CPU board

3. Operation Confirmation Conditions

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

Table 3-1 Operation Confirmation Conditions

Item	Contents
CPU Board	RL78/G22 CPU Board (RTK0EG0041C01001BJ) (RL78/G22 Capacitive Touch Evaluation System (RSSK) (RTK0EG0042S01001BJ) accessory)
Electrode Board	- Appliance panel electrode board for Touch MEC function (with enclosure) - Self-capacitance method buttons: 7 - LED: 25
MCU used	RL78/G22 (R7F102GGE) (ROM: 64KB, RAM: 4KB)
Operating frequency	- Main system clock - High-speed on-chip oscillator clock (f_{IH}): 32 MHz - CPU/peripheral hardware clock (f_{CLK}): 32 MHz - Subsystem clock - Low-speed on-chip oscillator clock (f_{IL}): 32.768 kHz - Low-speed peripheral clock frequency (f_{SXP}): 32.768 kHz
Operating voltage	3.3 V (can operate from 2.7 V to 5.5 V) LVD0 detection voltage: reset mode At rising edge TYP, 2.67 V (2.59 V to 2.75 V) At falling edge TYP, 2.62 V (2.54 V to 2.70 V)
Integrated development environment (e ² studio)	Renesas Electronics e ² studio Version 2024-10 (24.10.0)
Smart Configurator (SC)	Renesas Electronics V1.11.0 (24.10.0) (Bundled with e ² studio)
C compiler (e ² studio)	Renesas Electronics CC-RL V1.14.00 Optimization level option: -Odefault
QE for Capacitive Touch	Renesas Electronics V4.0.0
Board support package (r_bsp)	V1.70
Emulator	Renesas E2 emulator Lite (RTE0T0002LKCE00000R)

The sample program uses the SIS driver/middleware and components shown in Figure 3-1.

Component	Version	Configuration
✓ Capacitive Sensing Unit driver. (r_ctsu)	2.00	r_ctsu(used)
✓ Event Link Controller	1.3.0	Config_ELC(ELC: used)
✓ Interrupt Controller	1.5.0	Config_INTC(INTC: used)
✓ Interval Timer	1.5.0	Config_ITL000(ITL000: used), Config_TAU0_0(TAU0_0: used), Config_ITL001(ITL001: u...
✓ Ports	1.5.0	Config_PORT(PORT: used)
✓ Touch middleware. (rm_touch)	2.00	rm_touch(used)
✓ Voltage Detector	1.4.0	Config_LVD0(LVD0: used)

Figure 3-1 Smart Configurator Components in Use

4. Sample Programs

An overview of the operation of each mode in the sample program is shown below.

Table 4-1 Operation Overview of Each Mode

Mode Name		Operation Overview	Use of MEC function	Use of SMS function	Touch measurement cycle
Normal mode (Config01 Note.)		Touch button mode Touching each touch button detects the touch and changes the LED lighting pattern.	-	-	20 ms
Standby mode	operation mode 1 (Config02 Note.)	Proximity sensor mode (Use the SNOOZE mode function of CTSU2La) Wake up by proximity sensor detection when hand is held over the white frame.	○	-	20 ms
	operation mode 2 (Config03 Note.)	Touch Sensor Mode (Use the SNOOZE mode function of CTSU2La) Touch any of the buttons in the white frame to wake up with touch detection.	○	-	20 ms
	operation mode 3 (Config04 Note.)	Auto judgment (using SMS) mode Touch any of the buttons to auto judgment measurement using SMS in the white frame to wake up with touch detection.	○	○	100 ms

○ : used

— : unused

Note. Config xx is the touch interface configuration name used in each mode. For details of each configuration, refer to Chapter 4.5.3 Capacitive Touch Settings onward.

4.1 State Transition of Demonstration Screen

The state transition of the demonstration screen for this sample program is shown below. Refer to Chapter 6 for details on the screen.

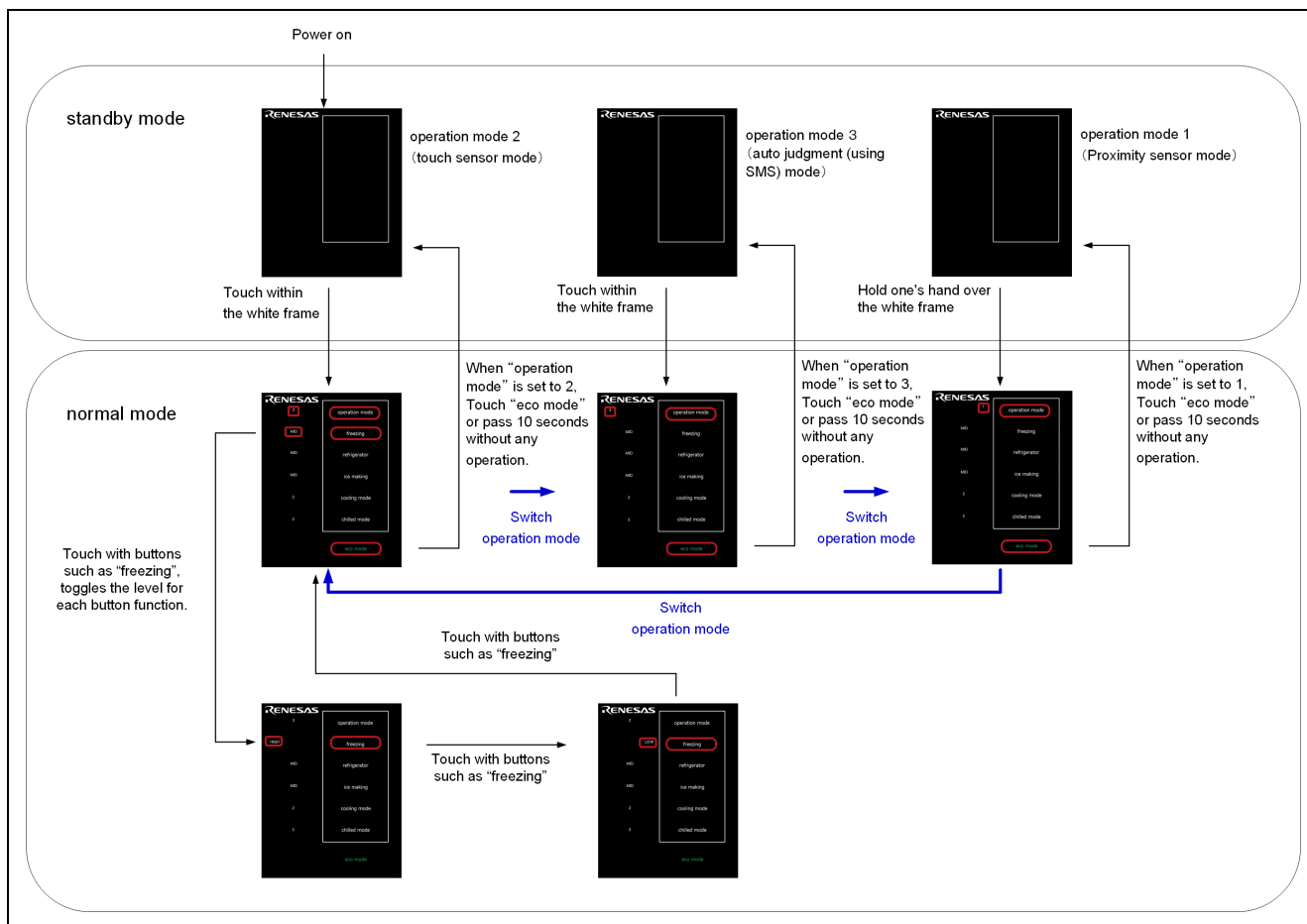


Figure 4-1 State Transition of Demonstration Screen

4.2 Overall Flowchart

The overall flowchart is shown below.

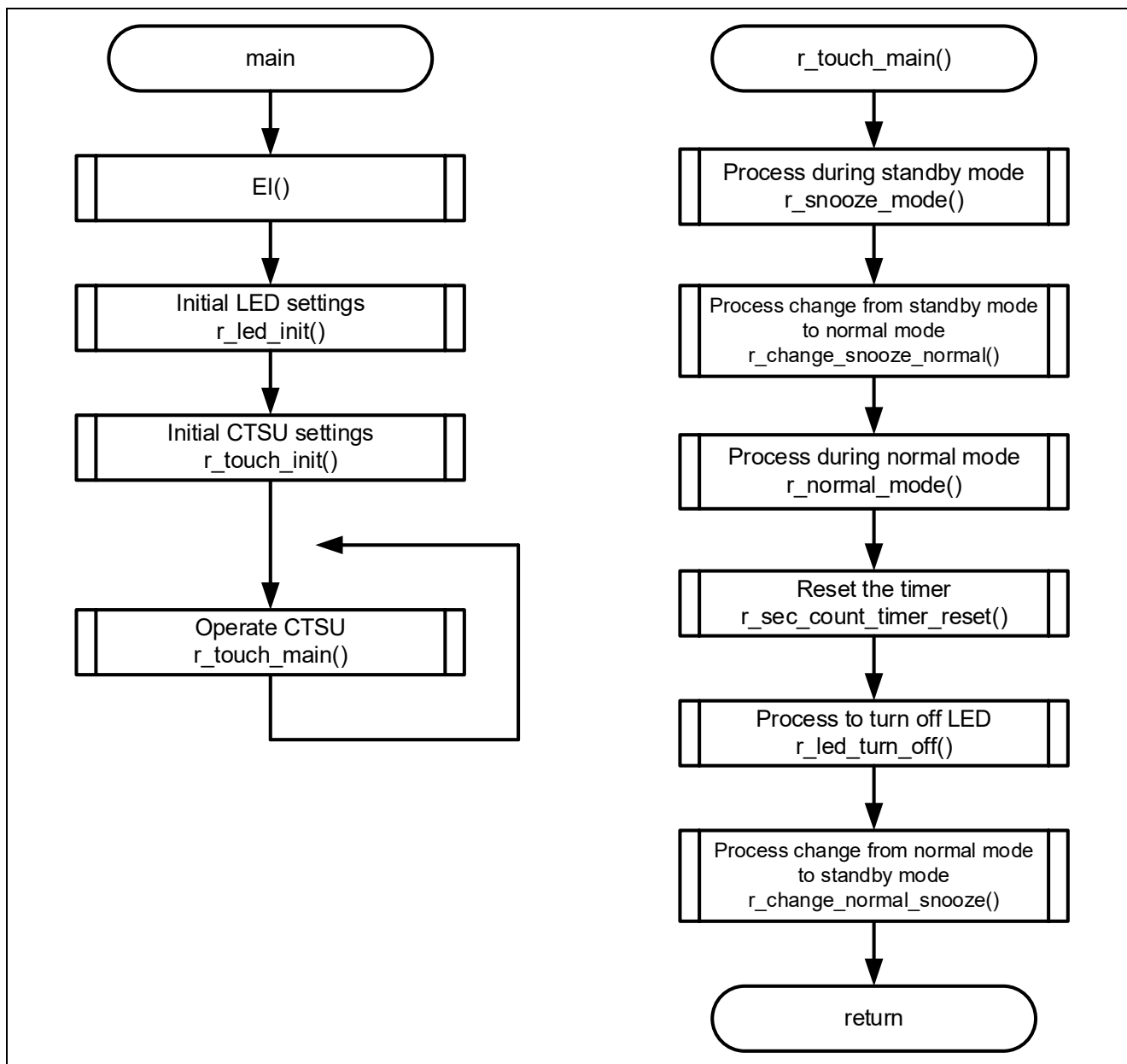


Figure 4-2 Overall Flowchart

4.3 Pins Used

The following shows list of pins used in this sample program.

Table 4-2 List of Pins and Functions

Pin name	Input/Output	Function
P17/TS18	Input/Output	Pins for electrostatic capacitance measurement Used for touch button (operation mode) and proximity sensor
P51/TS28	Input/Output	Pins for electrostatic capacitance measurement Used for touch button (freezing) and proximity sensor
P50/TS00	Input/Output	Pins for electrostatic capacitance measurement Used for touch button (refrigerator) and proximity sensor
P73/TS05	Input/Output	Pins for electrostatic capacitance measurement Used for touch button (ice making) and proximity sensor
P74/TS06	Input/Output	Pins for electrostatic capacitance measurement Used for touch button (cooling mode) and proximity sensor
P75/TS07	Input/Output	Pins for electrostatic capacitance measurement Used for touch button (chilled mode) and proximity sensor
P31/TS01	Input/Output	Pins for electrostatic capacitance measurement Used for touch button (eco mode)
P30/TSCAP	-	Connection pin to capacitor for measurement secondary power
P130	Output	Output port for SMS external trigger
P16 / INTP5	Input	Input port for SMS external trigger
P26	Input ^{Note} /Output	Matrix LED anode 0
P23	Input ^{Note} /Output	Matrix LED anode 1
P21	Input ^{Note} /Output	Matrix LED anode 2
P20	Input ^{Note} /Output	Matrix LED anode 3
P120	Input ^{Note} /Output	Matrix LED cathode 0
P121	Input ^{Note} /Output	Matrix LED cathode 1
P122	Input ^{Note} /Output	Matrix LED cathode 2
P146	Input ^{Note} /Output	Matrix LED cathode 3
P41	Input ^{Note} /Output	Matrix LED cathode 4
P61	Input ^{Note} /Output	Matrix LED cathode 5
P62	Output	LED independent control

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.

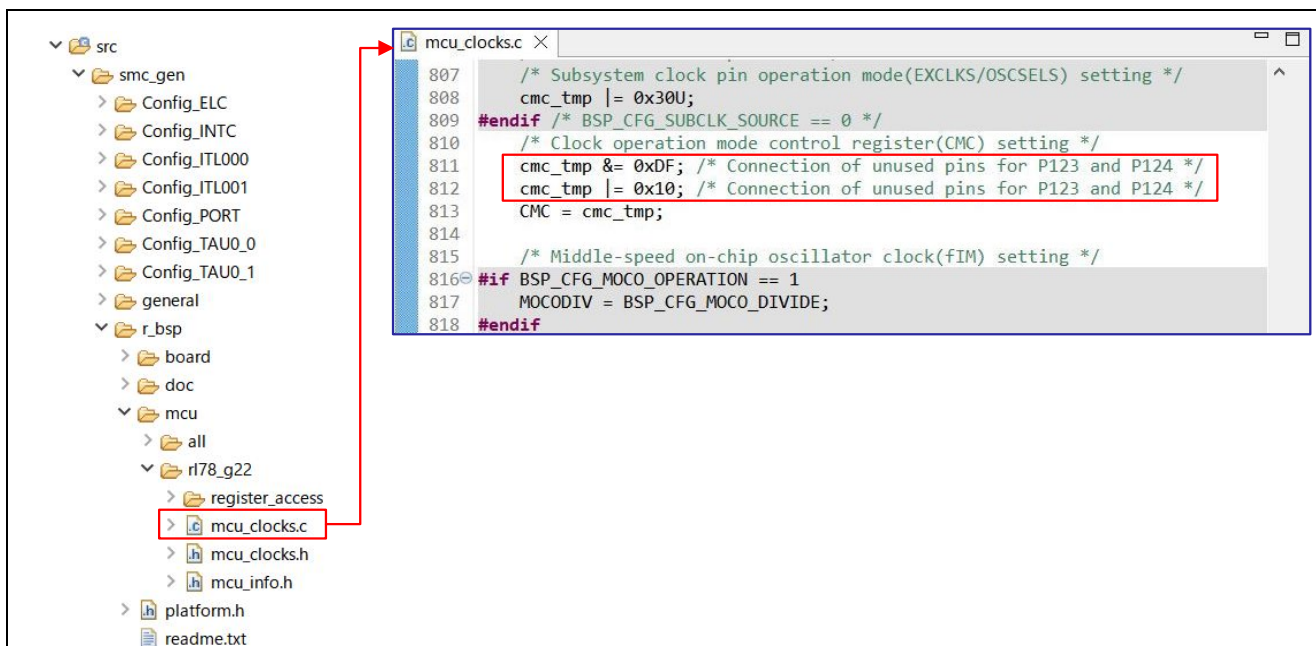
4.4 Setting of Unused Pins

shows the port pin settings when measuring current consumption. These are set in the “PORT module” component of the Smart Configurator. Port pins that also serve as TS pins and are not listed in the table below are fixed to the low output state by setting them to “Used” in the “CTSU module” component.

Table 4-3 Setting of Unused Pins

Pin No.	Port Name	Pin Connection	Parameters of Smart Configurator	Remarks
35	P00	CN2 (Pin Number:35)	Unused (Input buffer OFF)	The pins to V _{DD} via resistors.
34	P01	CN2 (Pin Number:33)	Unused (Internal Pull-up)	
22	P10	CN2 (Pin Number:22)	Unused (Input buffer OFF)	
21	P11	CN2 (Pin Number:21)	Unused (Input buffer OFF)	
20	P12	CN2 (Pin Number:20)	Unused (Input buffer OFF)	
19	P13	CN2 (Pin Number:19)	Unused (Input buffer OFF)	
18	P14	CN2 (Pin Number:18)	Unused (Input buffer OFF)	
17	P15	CN2 (Pin Number:17)	Unused (Input buffer OFF)	
30	P22	CN2 (Pin Number:32)	Out	Leave the pins open-circuit.
28	P24	CN2 (Pin Number:30)	Out	
27	P25	CN2 (Pin Number:29)	Out	
25	P27	CN2 (Pin Number:25)	Out	
39	P40	Pull up	Unused (Internal Pull-up)	The pins to V _{DD} via resistors.
1	P60	Pull up	In	
4	P63	Pull up	In	
11	P70	CN2 (Pin Number:7)	Unused (Internal Pull-up)	
10	P71	CN2 (Pin Number:6)	Unused (Internal Pull-up)	
9	P72	CN2 (Pin Number:5)	Unused (Internal Pull-up)	
42	P123	Open	Unused	Set the EXCLKS bit to 0 and the OSCSELS bit to 1 in the clock operation mode control register (CMC), set the XTSTOP bit in the clock operation status control register (CSC) to 1.
41	P124	Open	Unused	
43	P137	Pull down	Unused	The pins to GND via resistors.
36	P140	CN2(Pin Number:36)	Unused (Internal Pull-up)	The pins to V _{DD} via resistors.
24	P147	CN2(Pin Number:24)	Unused (Internal Pull-up)	

Note. The CSC register is set by the `r_sms_init` function in `touch.c`. The CMC register is set by the `mcu_clock_setup` function in `mcu_clocks.c` as shown in Figure 4-3.



```

807  /* Subsystem clock pin operation mode(EXCLKS/OSCSELS) setting */
808  cmc_tmp |= 0x30U;
809  #endif /* BSP_CFG_SUBCLK_SOURCE == 0 */
810  /* Clock operation mode control register(CMC) setting */
811  cmc_tmp &= 0xDF; /* Connection of unused pins for P123 and P124 */
812  cmc_tmp |= 0x10; /* Connection of unused pins for P123 and P124 */
813  CMC = cmc_tmp;
814
815  /* Middle-speed on-chip oscillator clock(fIM) setting */
816  #if BSP_CFG_MOCO_OPERATION == 1
817  MOCODIV = BSP_CFG_MOCO_DIVIDE;
818  #endif

```

Note: If you change the version of the "r_bsp" component in the Smart Configurator, mcu_clocks.c will be overwritten and any code added by the user will be erased. Therefore, whenever you change the version of the "r_bsp" component, you will need to add the code for the above settings.

Figure 4-3 Edit the mcu_clocks.c

4.5 Sample Programs Structure

4.5.1 Peripheral Functions Used

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

Table 4-4 List of Peripheral Functions Used and Functions

Peripheral Functions	Function
Capacitive Sensing Unit (CTS2La)	Measures electrostatic capacitance of the touch sensor.
32-bit Interval Timer channel 0 (TML32_000)	Timer to count the touch measurement cycles in normal mode and standby mode (operation mode 1, operation mode 2). Used for the measurement start trigger of CTS2La.
32-bit Interval Timer channel 1 (TML32_001)	Timer to count the touch measurement cycles in operation mode 3. Used for the measurement start trigger of CTS2La in auto judgment measurement using SMS.
Data Transfer Controller (DTC)	• Transfers the set value used for touch measurement from RAM to the touch-related register. • After touch measurement ends, the DTC transfers the measurement result (count value) from the touch-related register to RAM. • Used for auto judgment measurement using SMS
SNOOZE Mode Sequencer (SMS)	Used for auto judgment measurement using SMS
Event Link Controller (ELC)	• Used for connecting CTS2La and the event signal from 32-bit interval timer (ELCITL) • Used for SMS activating trigger in auto judgment measurement using SMS
Interrupt Controller (INTP)	Used for ELC activating trigger in auto judgment measurement using SMS
Timer Array Unit channel 0 (TAU_00)	Used for Standby mode transition timer
Timer Array Unit channel 1 (TAU_01)	Used for LED matrix control internally
PORT Functions (PORT)	• Used for LED control • used for auto judgment measurement using SMS (interrupt signals are generated using signals output from the port)

4.5.2 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 4-5 Parameters of Smart Configurator (1/3)

Tag name	Components	Contents
Clocks	-	Operation mode : High-speed main mode 2.7 (V) ~5.5 (V) High-speed on-chip oscillator : 32MHz f _{OCO} start setting : Normal f _{IHP} : 32MHz f _{MAIN} : 32MHz f _{CLK} : 32000kHz
System	-	On-chip debug operation setting : Use emulator Emulator setting : E2 emulator 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
Components	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

Table 4-6 Parameters of Smart Configurator (2/3)

Tag name	Components	Contents
Components	r_ctsu	Components : r_ctsu Resource : CTSU Data transfer of INTCTSUWR and INTCTSURD : DTC Auto judgment function in Snooze mode using SMS : Enable Data storage address setting for CTSURD : 0xFFC00 Data storage address setting for CTSUWR : 0xFFE00 Output port number for external trigger : PORT2 Bit number for external trigger output : BIT2 Interrupt port number for external trigger : INTP5 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.
	Config_ITL000	Components : Interval Timer Operation : 8 bit count mode Resource : ITL000 Operation clock : f _{SXP} Clock source : f _{ITLO} /16 Interval value : 20ms Interrupt setting : Unused
	Config_ITL001	Components : Interval Timer Operation : 8 bit count mode Resource : ITL001 Operation clock : f _{SXP} Clock source : f _{ITLO} /16 Interval value : 100ms Interrupt setting : Unused
	Config_INTC	Components : Interrupt Controller Resource : INTC INTP5 : Used Valid edge : Falling edge 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

Table 4-7 Parameters of Smart Configurator (3/3)

Tag name	Components	Contents
Components	Config_TAU0_0	Components : Interval Timer Operation : 16 bit count mode Resource : TAU0_0 Operation clock : CK00 Clock source : $f_{CLK}/2^{10}$ 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 : $f_{CLK}/2^8$ Interval value : 7ms Interrupt setting : Used Priority : Level 3
	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 Checked "Out" on P22 PORT4 : Checked "In" on P41 PORT6 : Checked "Out" and "Output 1" on P61 and P62 PORT12 : Checked "In" on P120~P122 PORT13 : Checked "Out" on P130 PORT14 : Checked "In" on P146

4.5.3 Capacitive Touch Settings

These are the touch interface configuration, configuration (method) settings and tuning results of this sample code. These use the tuning function of QE.

4.5.3.1 Touch Interface Configuration

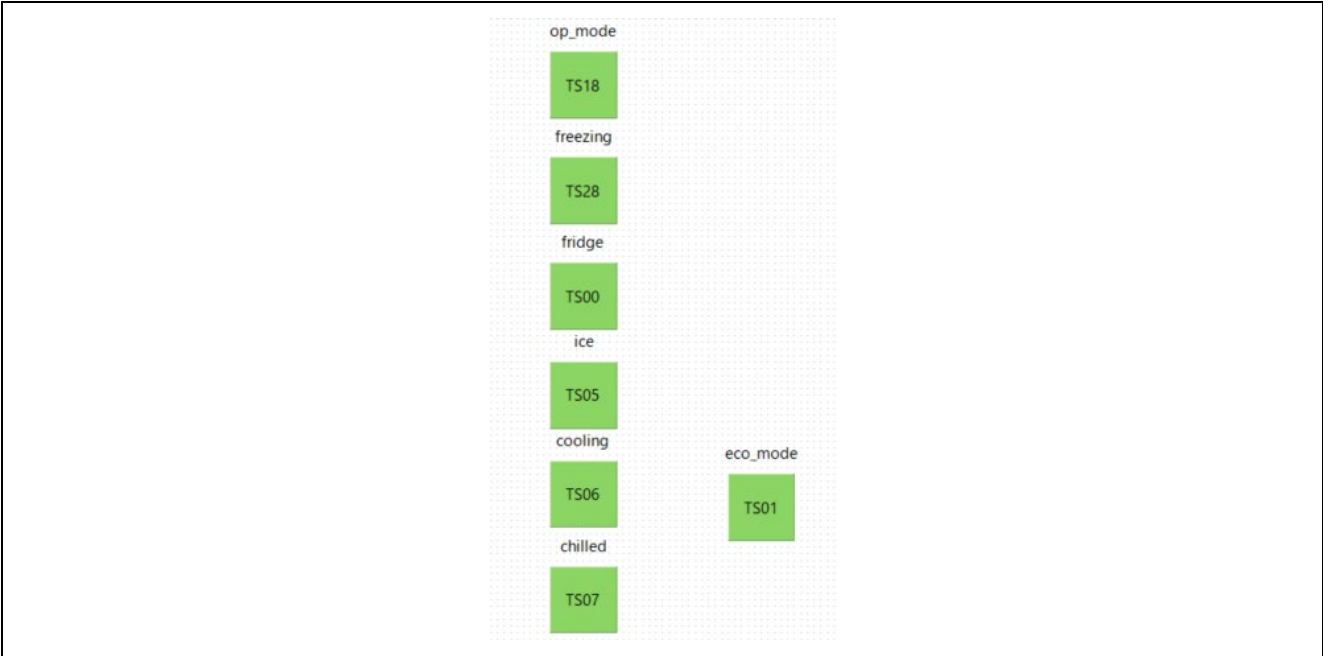


Figure 4-4 Touch Interface Configuration

4.5.3.2 Configuration (methods) Settings

In “config02” and “config03”, the MEC function is enable. In “config04”, the MEC function and the auto judgment measurements using SMS are enabled.

	☐ config01	☐ config02	☐ config03	☐ config04
eco_mode(self)	☒ Available	☐	☐	☐
chilled(self)	☒ Available	☒ Available	☒ Available	☒ Available
cooling(self)	☒ Available	☒ Available	☒ Available	☒ Available
ice(self)	☒ Available	☒ Available	☒ Available	☒ Available
fridge(self)	☒ Available	☒ Available	☒ Available	☒ Available
freezing(self)	☒ Available	☒ Available	☒ Available	☒ Available
op_mode(self)	☒ Available	☒ Available	☒ Available	☒ Available
Touch Judgment (SMS)	☐ Enable	☐ Enable	☐ Enable	☒ Enable
MEC	☐ Enable	☒ Enable	☒ Enable	☒ Enable

Figure 4-5 Configuration (methods) Settings

4.5.3.3 Tuning results

Figure 4-6 QE Tuning Results shows tuning results in QE tuning. Sample code operates with the following setting values. Since the values in QE tuning results depend on the operating environment at QE tuning, these values may change at QE tuning again.

Tuning

Gesture

Touch I/F Configuration: r01an6740_g22demo_touch_mec

Method	Kind	Name	Touch Sensor	Parasitic Capacitance[pF]	Sensor Drive Pulse Frequency[MHz]	Threshold	Scan Time[ms]	Overflow
config01	Button(self)	eco_mode	TS01	37.229	1.0	1693	0.576	None
config01	Button(self)	chilled	TS07	35.069	1.0	1479	0.576	None
config01	Button(self)	cooling	TS06	33.16	1.0	1700	0.576	None
config01	Button(self)	ice	TS05	33.59	1.0	1705	0.576	None
config01	Button(self)	fridge	TS00	31.493	1.0	1702	0.576	None
config01	Button(self)	freezing	TS28	33.806	1.0	1674	0.576	None
config01	Button(self)	op_mode	TS18	31.493	1.0	1672	0.576	None
config02	Button(self)	Mec00	TS00	128.646	0.5	1123	0.576	None
config03	Button(self)	Mec01	TS00	128.757	0.5	1106	0.576	None
config04	Button(self)	Mec02	TS00	128.653	0.5	514; 439; 589	0.576	None

Figure 4-6 QE Tuning Results

4.5.4 Setting of Option Byte

The following shows the option byte settings by sample program.

Table 4-8 option byte settings

Address	Setting Value	Contents
000C0H / 020C0H	1110 1111B (0xEF)	Disables the watchdog timer (Counting stopped after reset)
000C1H / 020C1H	1111 1100B (0xFC)	LVD0 detection voltage: reset mode At rising edge TYP. 2.670 V (2.59 V to 2.75 V) At falling edge TYP. 2.620 V (2.54 V to 2.70 V)
000C2H / 020C2H	1110 1000B (0xE8)	HS mode, High-speed on-chip oscillator clock (f_{IH}): 32 MHz
000C3H / 020C3H	1000 0100B (0x84)	Enables on-chip debugging

4.5.5 File Structure

The following shows file structure by sample program. The project configuration file and smart configurator generation file of the development environment are omitted.

Table 4-9 File Structure

Folder name, File name	Outline
r01an6740_g22demo_touch_mec	Project folder for program source
└─ qe_gen	QE for capacitive touch generation folder
└─ src	-
├─ smc_gen	Smart Configurator generation folder
├─ Config_ELC	
├─ Config_INTC	
├─ Config_ITL000	
├─ Config_ITL001	
├─ Config_PORT	
├─ Config_TAU0_0	
├─ Config_TAU0_1	
├─ general	
├─ r_bsp	
├─ r_config	
├─ r_ctsu	
├─ r_pincfg	
├─ rm_touch	
├─ led.c	Source file for LED control
├─ led.h	Header file for LED control
├─ main.c	Source file for main processing
├─ mode.c	Source file for mode control
├─ mode.h	Header file for mode control
├─ touch.c	Source file for touch control
├─ touch.h	Header file for touch control
└─ QE-Touch	QE for capacitive touch generation folder

4.5.6 Variables

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

Table 4-10 List of Variables Used in the Sample Code

Type	Variable name	Contents	Files
uint8_t	g_led_position[6]	Array of matrix LED lighting patterns	led.c mode.c touch.c
uint8_t	g_pos_a	Specified variable on the anode side of the control LED of the matrix LED	led.c
uint8_t	g_touch_button_flg	Flag that any button is touched	led.c mode.c touch.c
uint8_t	g_eco_mode_flg	Flag that "eco mode" button is touched	led.c touch.c
uint8_t	g_sec_count_timer_count	Seconds count variable	led.c Config_TAU0_0_us er.c
uint8_t	g_sec_count_timer_stop_flg	Flag that have completed the seconds count process	Config_TAU0_0_us er.c
uint8_t	g_mode	Normal mode / standby mode switching variable	led.c mode.c touch.c
uint8_t	g_normal_end_flg	Flag that ended processing in normal mode	led.c mode.c
uint64_t	g_button_status	Status of which button was touched	led.c mode.c touch.c
uint8_t	g_snooze_mode_init	Standby mode initialization flag	mode.c
uint8_t	g_normal_mode_init	Normal mode initialization flag	mode.c
uint8_t	g_snooze_end_flg	Flag that ended processing in standby mode	mode.c

4.5.7 Constants

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

Table 4-11 List of Constants Used in the Sample Code

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

4.5.8 Functions

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

Table 4-12 List of Functions Used in the Sample Code (1/2)

Function name	Outline	Source file
main	Main processing	main.c
r_led_init	Processing of LED initialization	led.c
r_ledport_input	Setting the LED port to input mode	led.c
r_ledport_output	Setting the LED port to output mode	led.c
r_led_turn_on_all_5s	Processing to turn on all LEDs for 5s	led.c
r_led_turn_on	Processing to turn on LEDs	led.c
r_led_turn_off	Processing to turn off LEDs	led.c
r_ledmatrix_turn_on	Processing to turn on the matrix LEDs	led.c
r_ledmatrix_turn_off	Processing to turn off the matrix LEDs	led.c
r_ledmatrix_turn_on_a	Processing to turn on the anode side of the matrix LED	led.c
r_change_led_position	Processing to change the position of the matrix LED	led.c
r_change_led	Processing to change the lighting pattern of the matrix LED	led.c
r_snooze_mode_init_cpu	Processing of initialization in standby mode for CPU operation	mode.c
r_snooze_mode_init_sms	Processing of initialization in standby mode for SMS operation	mode.c
r_normal_mode_init	Processing of initialization in normal mode	mode.c
r_snooze_cpu	Processing of operation in standby mode for CPU operation	mode.c
r_snooze_sms	Processing of operation in standby mode for SMS operation	mode.c
r_snooze_mode	Processing of standby mode operation	mode.c
r_normal_mode	Processing of normal mode operation	mode.c
r_change_snooze_normal	Processing to change from standby mode to normal mode	mode.c
r_change_normal_snooze	Processing to change from normal mode to SNOOZE mode	mode.c
r_not_touched	Processing to determine that touch buttons are not touched	mode.c
r_touch_init	Initial settings of CTSU2La	touch.c
r_sms_init	Initial settings of SMS	touch.c
r_touch_main	Main operation when buttons are touched	touch.c
r_snooze_mode_touch_presses	Processing touch operation in standby mode	touch.c
r_change_eco_mode	Processing to change eco mode	touch.c
r_prevent_long_presses	Processing to prevent long presses	touch.c
r_touch_mec_scanstart_cpu	Switching ScanStart of MEC	touch.c
r_touch_mec_scanstop	Switching ScanStop of MEC	touch.c
r_touch_mec_dataget_cpu	Switching DataGet of MEC	touch.c
r_sms_trigger_start	Processing to start SMS trigger	touch.c
r_sms_trigger_stop	Processing to stop SMS trigger	touch.c

Table 4-13 List of Functions Used in the Sample Code (2/2)

Function name	Outline	Source file
r_Config_TAU0_0_interrupt	Interrupt function of TAU0_0	Config_TAU0_0_us er.c
r_sec_count_timer_start	Start the timer for seconds count	Config_TAU0_0_us er.c
r_sec_count_timer_reset	Reset the timer for seconds count	Config_TAU0_0_us er.c
r_Config_TAU0_1_interrupt	Interrupt function of TAU0_1	Config_TAU0_1_us er.c
r_ledmatrix_timer_start	Start the timer for matrix LED control	Config_TAU0_1_us er.c
r_ledmatrix_timer_reset	Reset the timer for matrix LED control	Config_TAU0_1_us er.c

4.5.9 Function Specifications

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

[Function name] main

Outline	Main processing
Header	r_smc_entry.h, touch.h, led.h
Declaration	int main(void)
Description	Initializes LEDs and CTSU2La and repeats touch operation.
Arguments	None
Return value	None
Remarks	None

[Function name] r_led_init

Outline	Processing of LED initialization
Header	led.h
Declaration	void r_led_init(void)
Description	When power is turned on, all LEDs are turned on for 5s and g_led_position is initialized.
Arguments	None
Return value	None
Remarks	None

[Function name] r_ledport_input

Outline	Setting the LED port to input mode
Header	led.h
Declaration	void r_ledport_input(void)
Description	Changes the LED port to input mode.
Arguments	None
Return value	None
Remarks	None

[Function name] r_ledport_output

Outline	Setting the LED port to output mode
Header	led.h
Declaration	void r_ledport_output(void)
Description	Changes the LED port to output mode.
Arguments	None
Return value	None
Remarks	None

[Function name] r_led_turn_on_all_5s

Outline	Processing to turn on all LEDs for 5s
Header	led.h
Declaration	void r_led_turn_on_all_5s(void)
Description	Turns on all LEDs for 5s.
Arguments	None
Return value	None
Remarks	None

[Function name] 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

[Function name] 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

[Function name] 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

[Function name] 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

[Function name] 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 name] 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 name] 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 name] r_snooze_mode_init_cpu

Outline Processing of initialization in standby mode for CPU operation

Header mode.h

Declaration void r_snooze_mode_init_cpu (void)

Description Processes the initialization of standby mode for CPU operation.

Arguments None

Return value None

Remarks None

[Function name] r_snooze_mode_init_cpu

Outline Processing of initialization in standby mode for SMS operation

Header mode.h

Declaration void r_snooze_mode_init_sms (void)

Description Processes the initialization of standby mode for SMS operation.

Arguments None

Return value None

Remarks None

[Function name] 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_cpu

Outline	Processing of operation in standby mode for CPU operation
Header	mode.h
Declaration	void r_snooze_cpu(void)
Description	Processes during standby mode for CPU operation.
Arguments	None
Return value	None
Remarks	None

[Function name] r_snooze_sms

Outline	Processing of operation in standby mode for SMS operation
Header	mode.h
Declaration	void r_snooze_sms(void)
Description	Processes during standby mode for SMS operation.
Arguments	None
Return value	None
Remarks	None

[Function name] r_snooze_mode

Outline	Processing of standby mode operation
Header	mode.h
Declaration	void r_snooze_mode(void)
Description	Processes during standby 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 standby mode to normal mode
Header	mode.h
Declaration	void r_change_snooze_normal(void)
Description	Changes from standby 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 standby mode
Header	mode.h
Declaration	void r_change_normal_snooze(void)
Description	Changes from normal mode to standby 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

[Function name] r_touch_init

Outline	Initial settings of CTSU2La
Header	touch.h
Declaration	void r_touch_init(void)
Description	Sets the initial settings for CTSU2La.
Arguments	None
Return value	None
Remarks	None

[Function name] r_sms_init

Outline	Initial settings of SMS
Header	touch.h
Declaration	void r_sms_init(void)
Description	Sets the initial settings for SMS.
Arguments	None
Return value	None
Remarks	None

[Function name] r_touch_main

Outline	Main operation when buttons are touched
Header	touch.h
Declaration	void touch_main(void)
Description	Executes the main process when the button is touched.
Arguments	None
Return value	None
Remarks	None

[Function name] r_snooze_mode_touch_prosses

Outline Processing touch operation in standby mode
Header touch.h
Declaration void r_snooze_mode_touch_prosses(void)
Description Processes touch operation in standby mode.
Arguments None
Return value None
Remarks None

[Function name] r_change_eco_mode

Outline Processing to change eco mode
Header touch.h
Declaration void r_change_eco_mode(void)
Description Changes to eco-mode.
Arguments None
Return value None
Remarks None

[Function name] r_prevent_long_presses

Outline Processing to prevent long presses
Header touch.h
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 name] r_touch_mec_scanstart_cpu

Outline Switching ScanStart of MEC
Header touch.h
Declaration fsp_err_t r_touch_mec_scanstart_cpu(void)
Description Processes switching of ScanStart of MEC.
Arguments None
Return value err
Remarks None

[Function name] r_touch_mec_scanstop

Outline Switching ScanStop of MEC
Header touch.h
Declaration fsp_err_t r_touch_mec_scanstop(void)
Description Processes switching of ScanStop of MEC.
Arguments None
Return value err
Remarks None

[Function name] r_touch_mec_dataget_cpu

Outline Switching DataGet of MEC
Header touch.h
Declaration fsp_err_t r_touch_mec_dataget_cpu(void)
Description Processes switching of DataGet of MEC.
Arguments None
Return value err
Remarks None

[Function name] r_sms_trigger_start

Outline Processing to start SMS trigger
Header touch.h
Declaration void r_sms_trigger_start (void)
Description Starts SMS trigger.
Arguments None
Return value None
Remarks None

[Function name] r_sms_trigger_stop

Outline Processing to stop SMS trigger
Header touch.h
Declaration void r_sms_trigger_stop (void)
Description Stops SMS trigger.
Arguments None
Return value None
Remarks None

[Function name] r_Config_TAU0_0_interrupt

Outline Interrupt function of TAU0_0
Header Config_TAU0_0.h
Declaration static void __near r_Config_TAU0_0_interrupt(void)
Description Counts the timer for seconds count.
Arguments None
Return value None
Remarks None

[Function name] r_sec_count_timer_start

Outline Start the timer for seconds count
Header Config_TAU0_0.h
Declaration void r_sec_count_timer_start(void)
Description Starts the timer for seconds count.
Arguments None
Return value None
Remarks None

[Function name] 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

[Function name] 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

[Function name] 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

[Function name] 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

4.5.10 Flowchart

4.5.10.1 Flowchart of Main Function

The flowchart of main function is shown below.

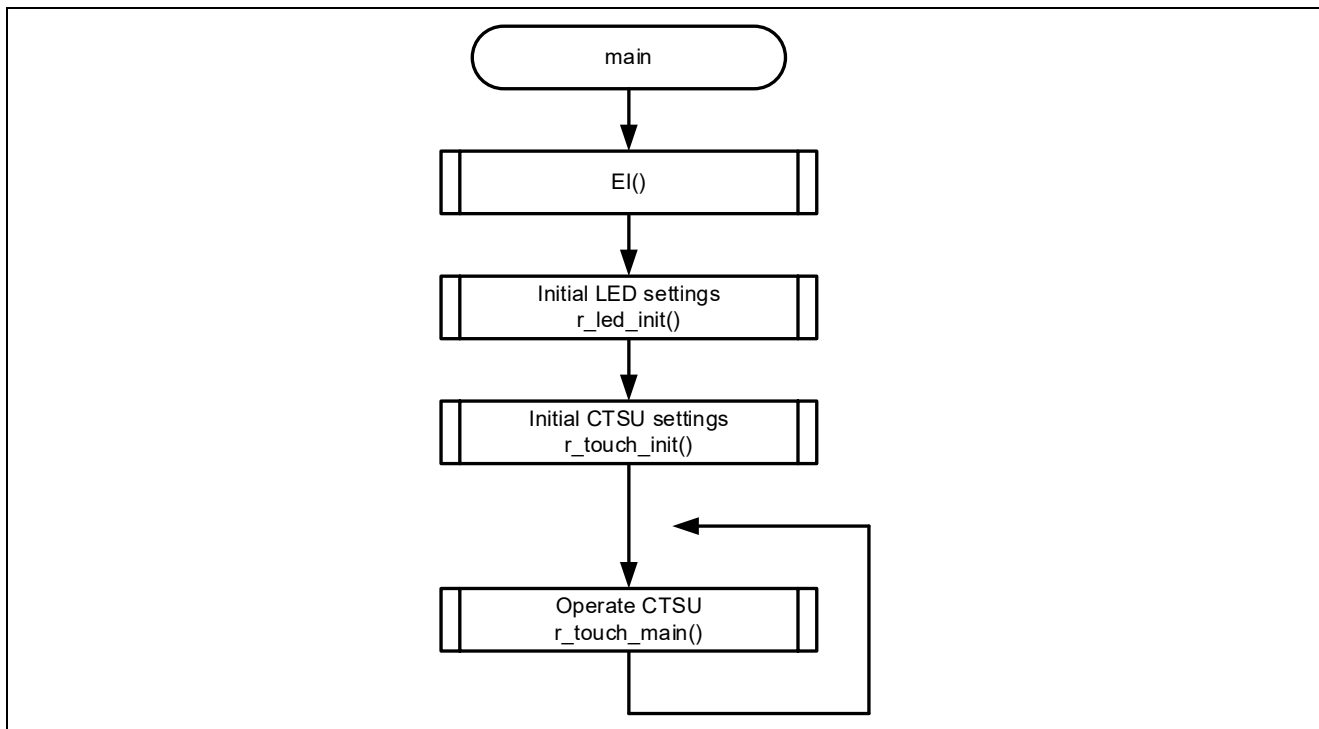


Figure 4-7 Flowchart of Main Function

4.5.10.2 Flowchart of r_touch_init Function

The flowchart of r_touch_init function is shown below.

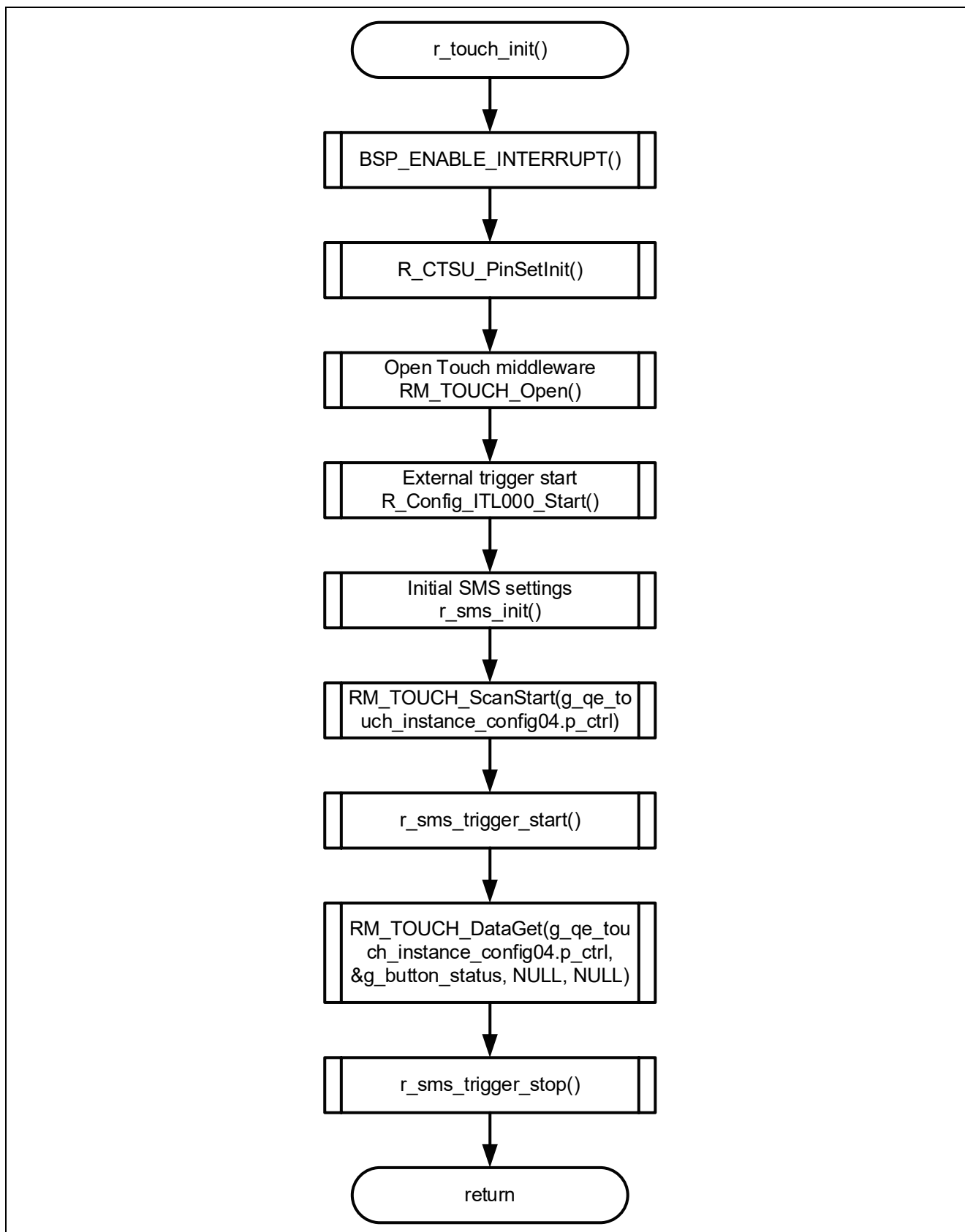


Figure 4-8 Flowchart of r_touch_init Function

4.5.10.3 Flowchart of r_sms_init Function

The flowchart of r_sms_init function is shown below.

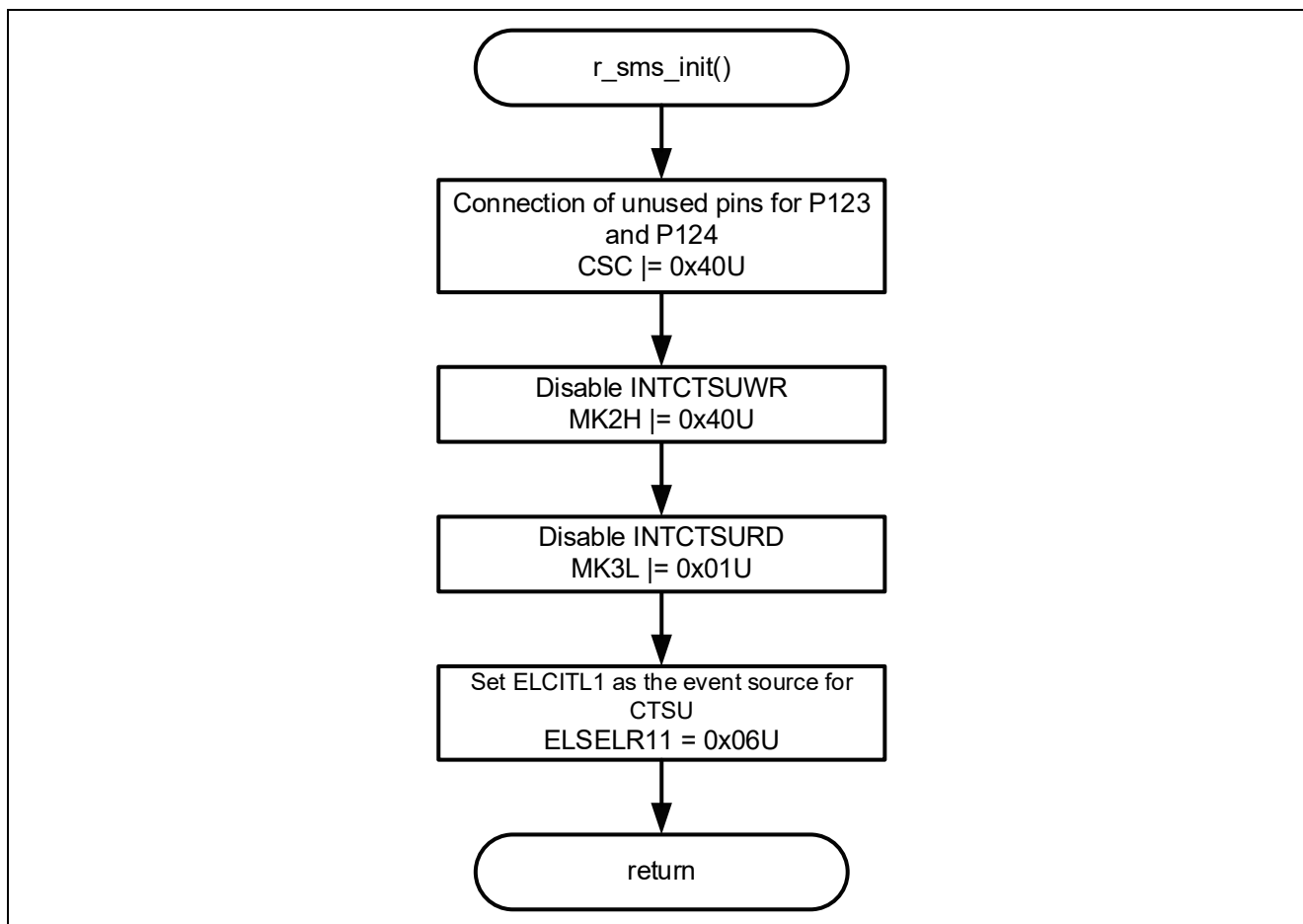


Figure 4-9 Flowchart of r_sms_init Function

4.5.10.4 Flowchart of r_touch_main Function

The flowchart of r_touch_main function is shown below.

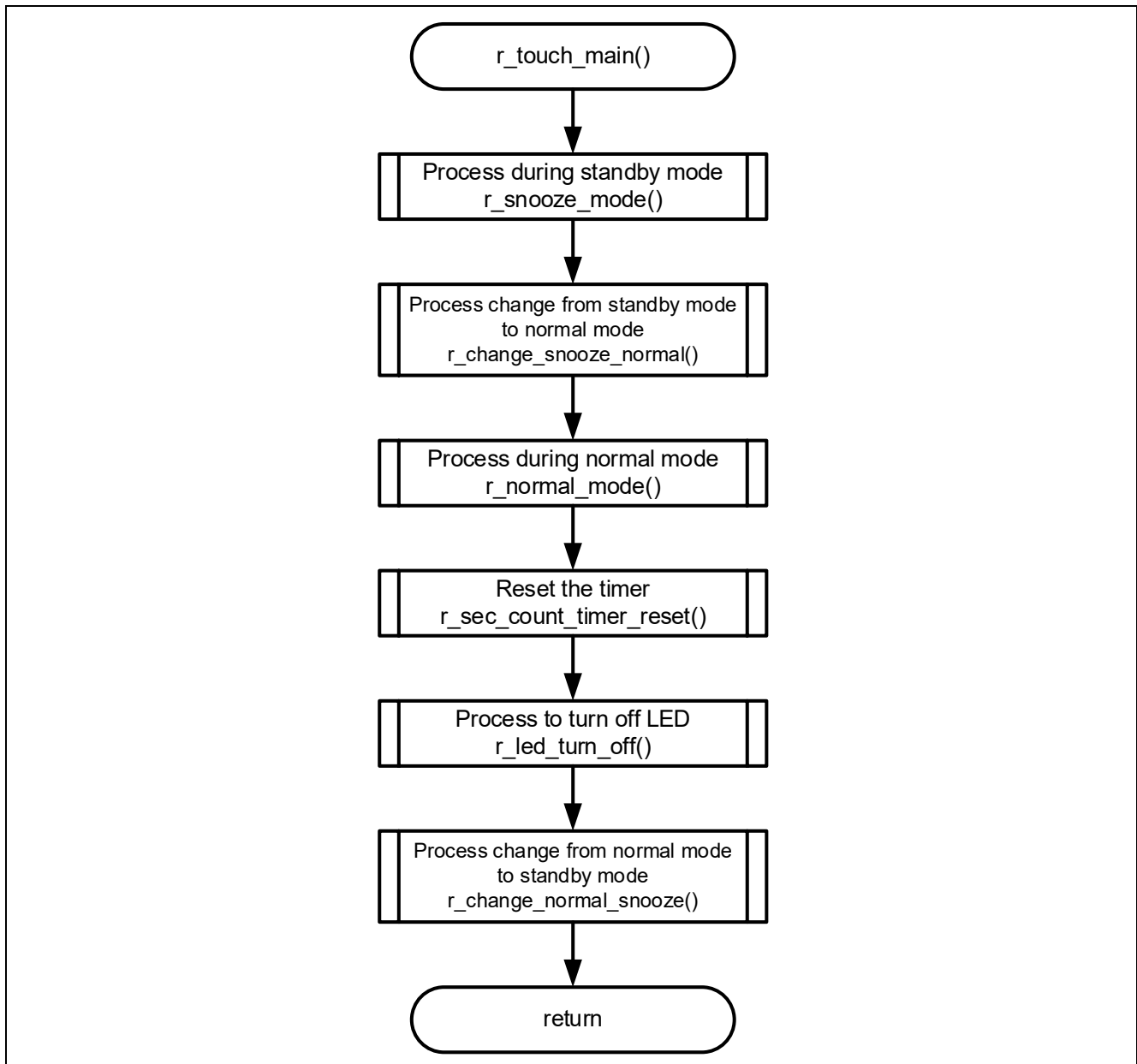


Figure 4-10 Flowchart of r_touch_main Function

4.5.10.5 Flowchart of r_snooze_mode_touch_prosses Function

The flowchart of r_snooze_mode_touch_prosses function is shown below.

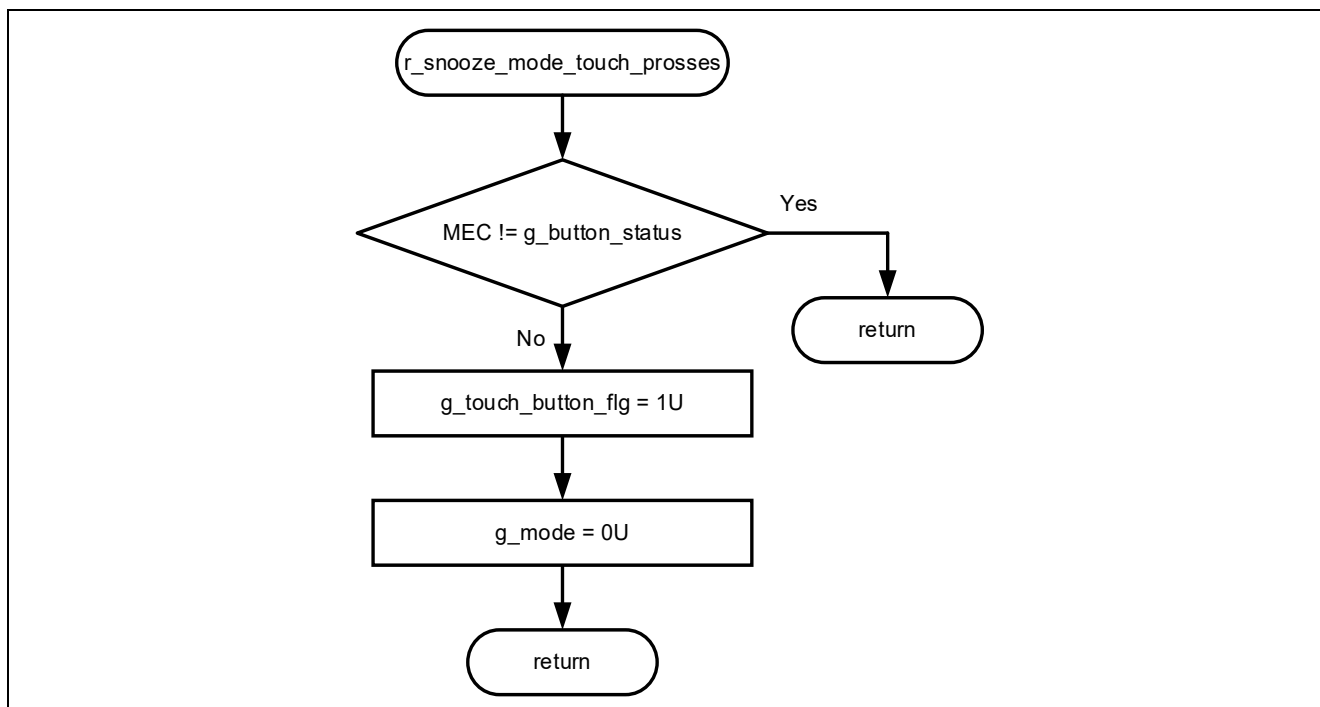


Figure 4-11 Flowchart of r_snooze_mode_touch_prosses Function

4.5.10.6 Flowchart of r_change_eco_mode Function

The flowchart of r_change_eco_mode function is shown below.

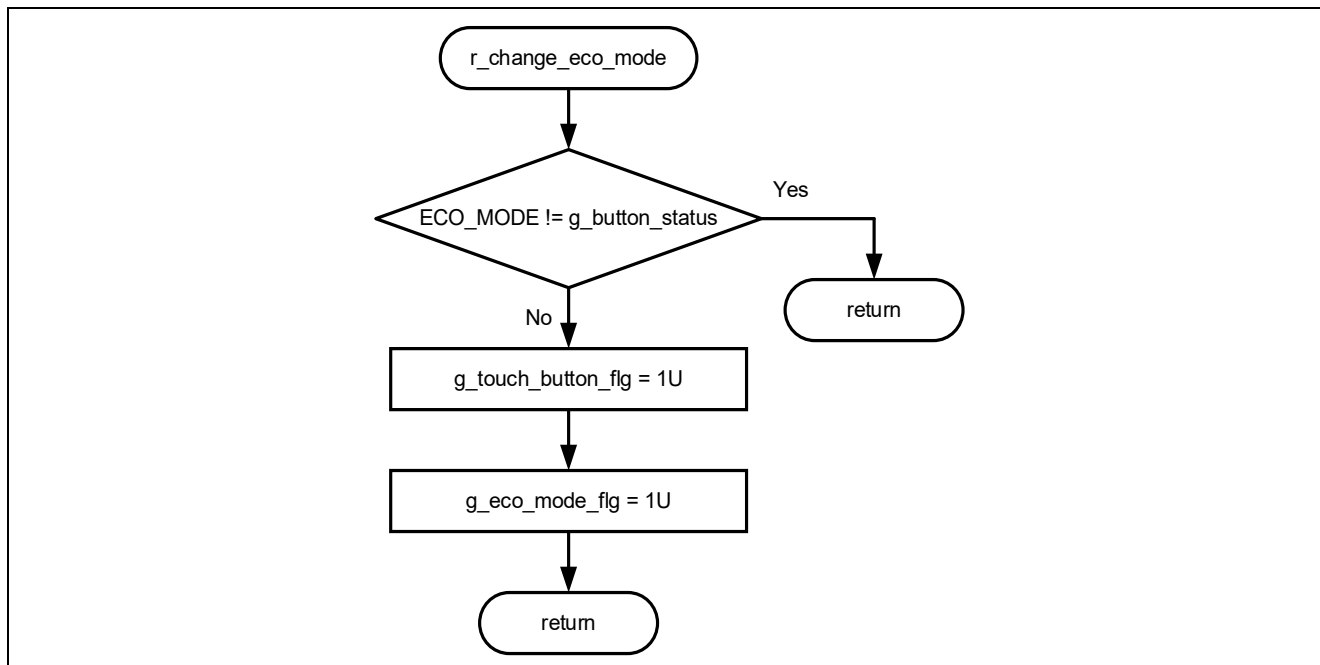


Figure 4-12 Flowchart of r_change_eco_mode Function

4.5.10.7 Flowchart of r_prevent_long_presses Function

The flowchart of r_prevent_long_presses function is shown below.

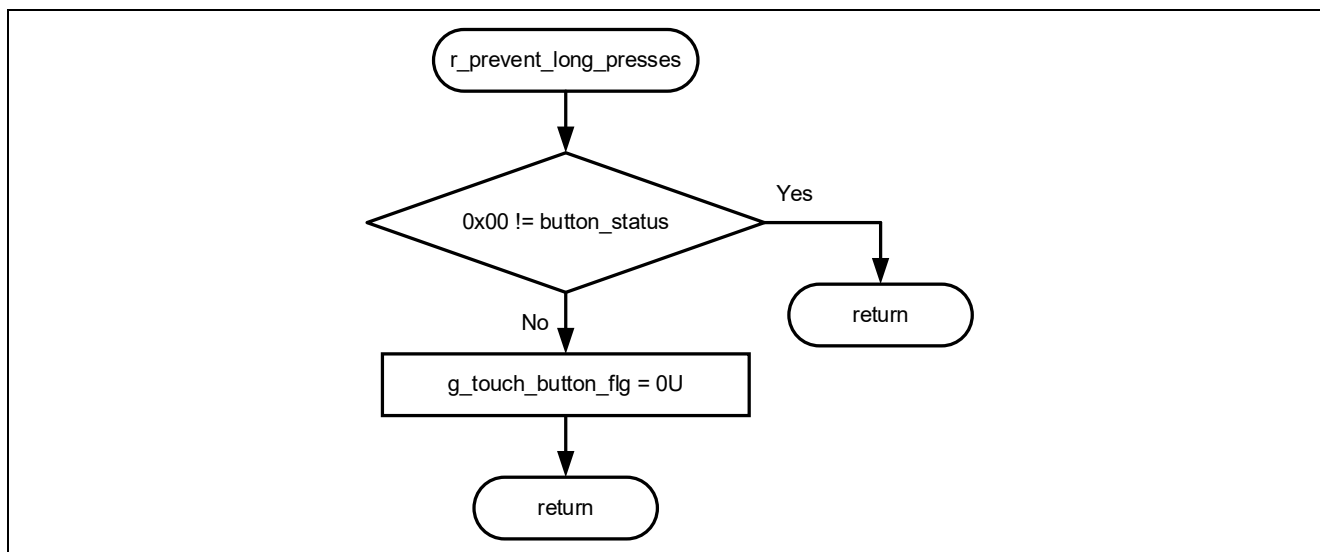


Figure 4-13 Flowchart of r_prevent_long_presses Function

4.5.10.8 Flowchart of r_snooze_mode_init_cpu Function

The flowchart of r_snooze_mode_init_cpu function is shown below.

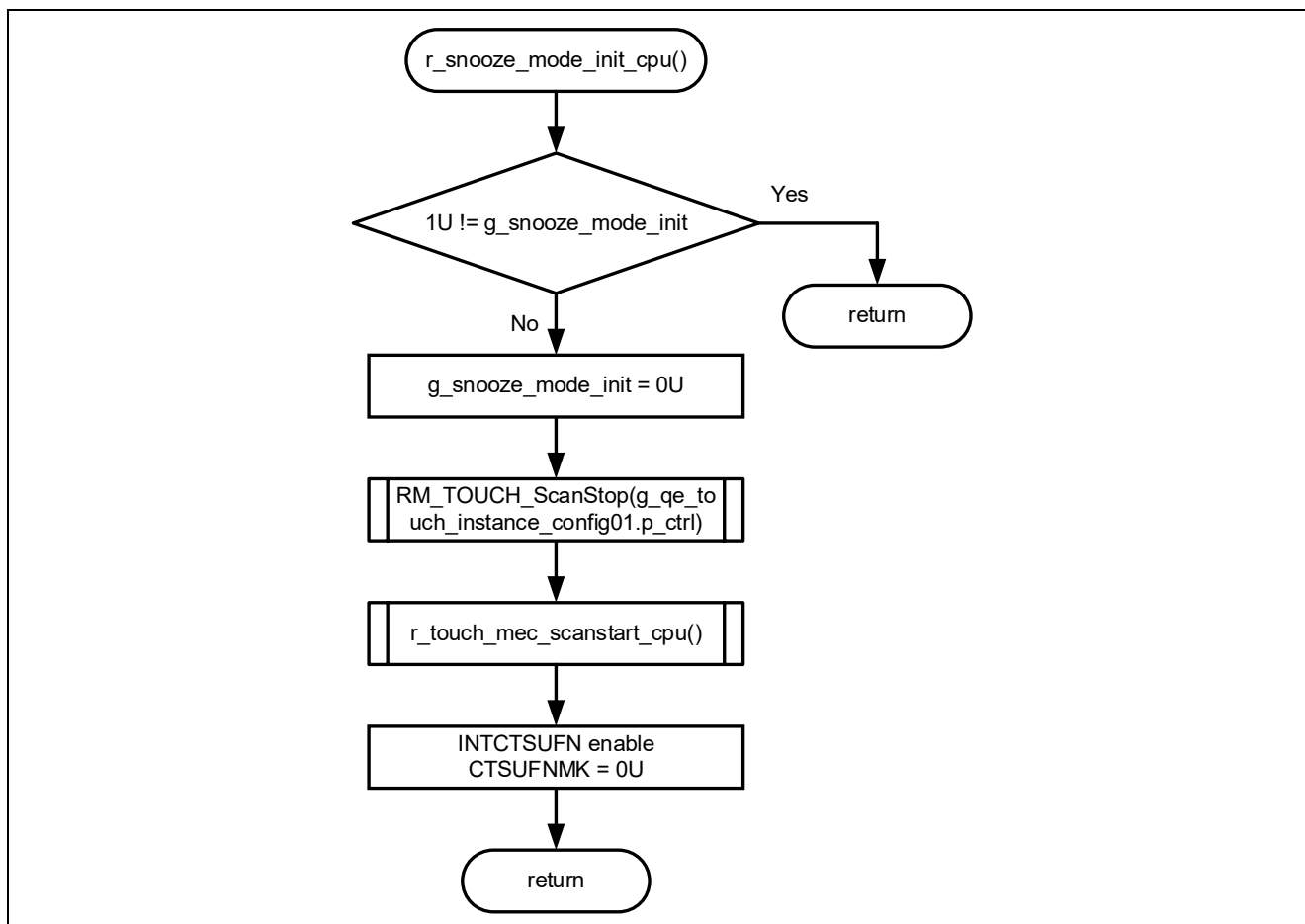


Figure 4-14 Flowchart of r_snooze_mode_init_cpu Function

4.5.10.9 Flowchart of r_snooze_mode_init_sms Function

The flowchart of r_snooze_mode_init_sms function is shown below.

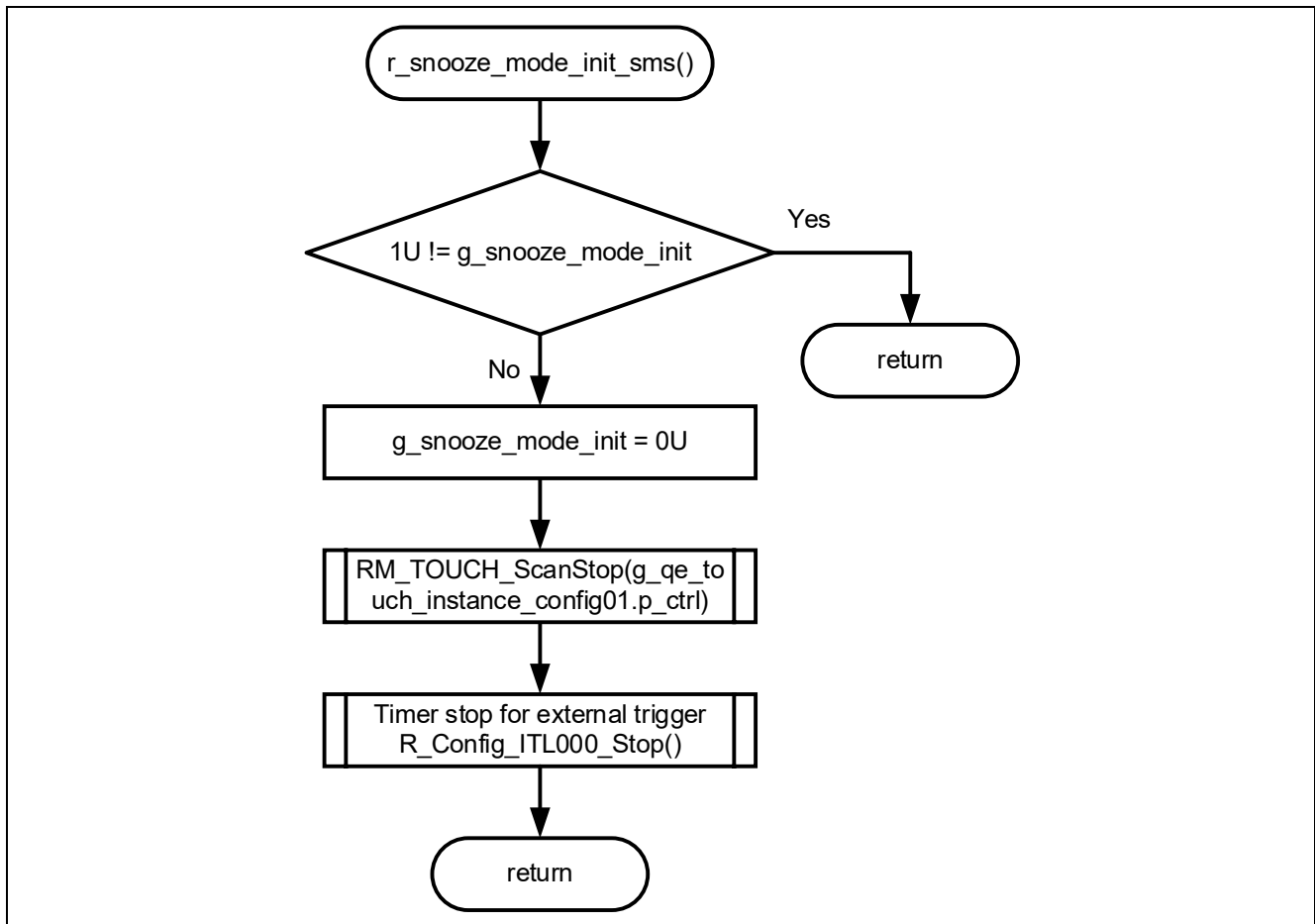


Figure 4-15 Flowchart of r_snooze_mode_init_sms Function

4.5.10.10 Flowchart of r_snooze_mode Function

The flowchart of r_snooze_mode function is shown below.

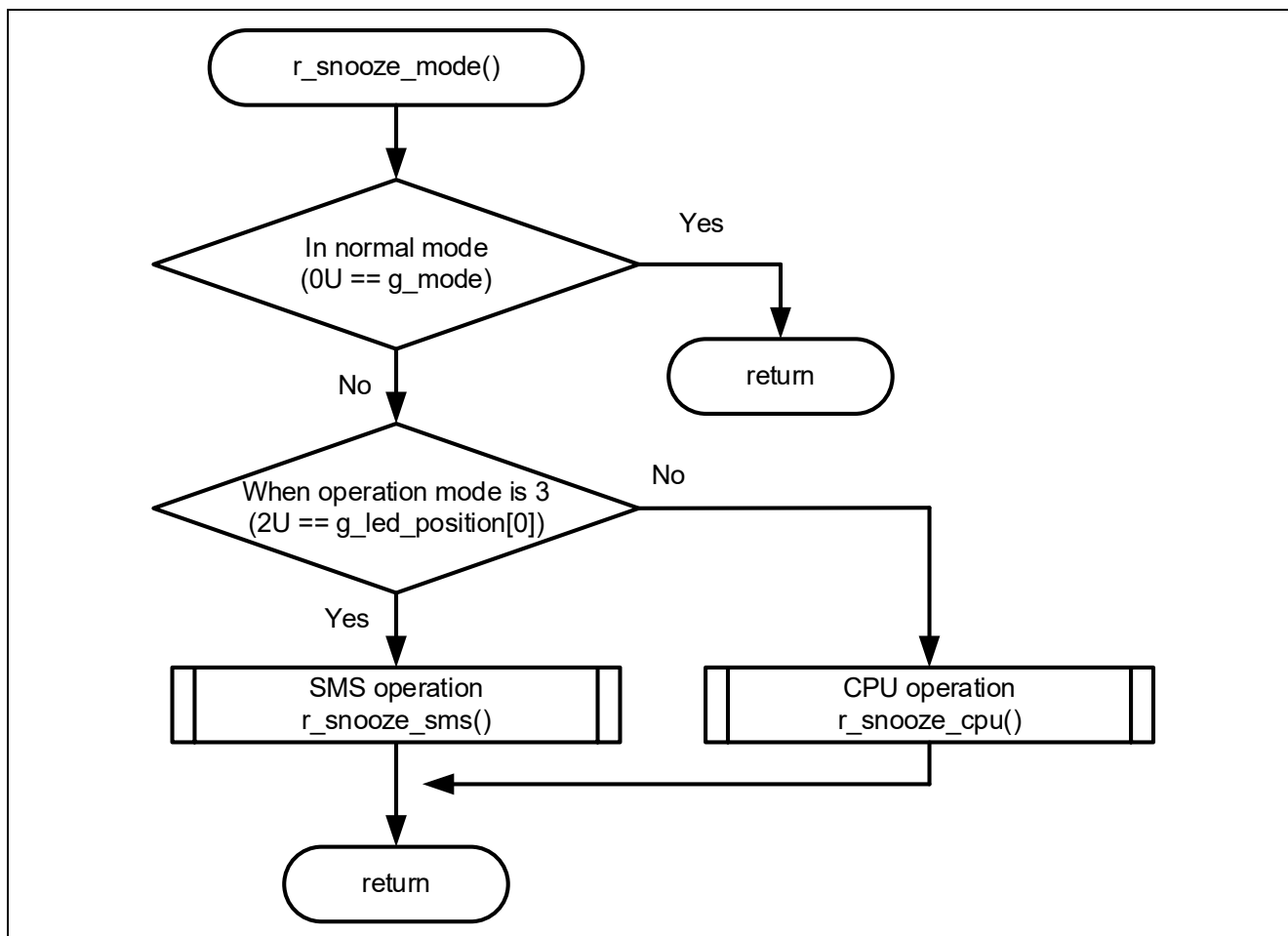


Figure 4-16 Flowchart of r_snooze_mode Function

4.5.10.11 Flowchart of r_snooze_cpu Function

The flowchart of r_snooze_cpu function is shown below.

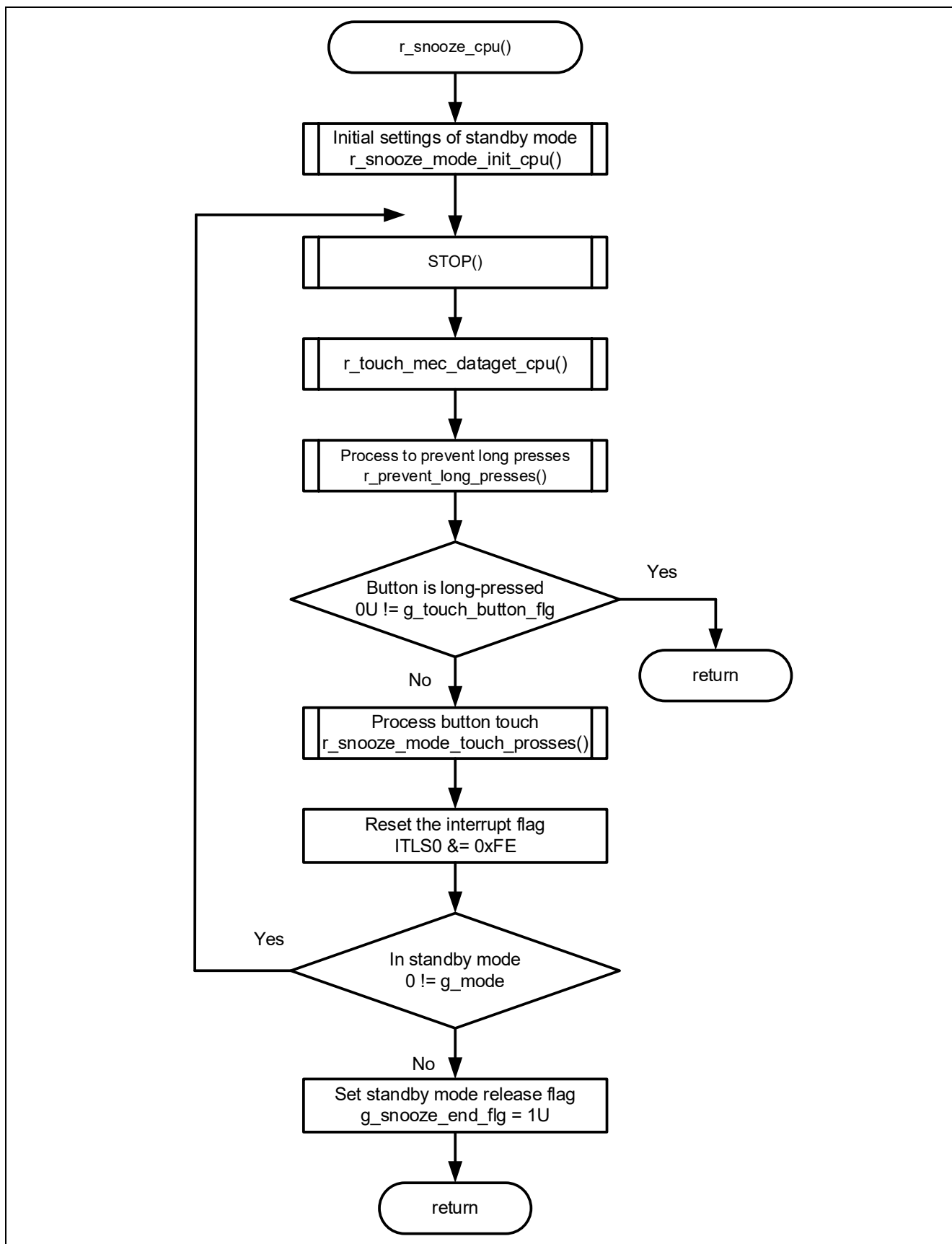


Figure 4-17 Flowchart of r_snooze_cpu Function

4.5.10.12 Flowchart of r_snooze_sms Function

The flowchart of r_snooze_sms function is shown below.

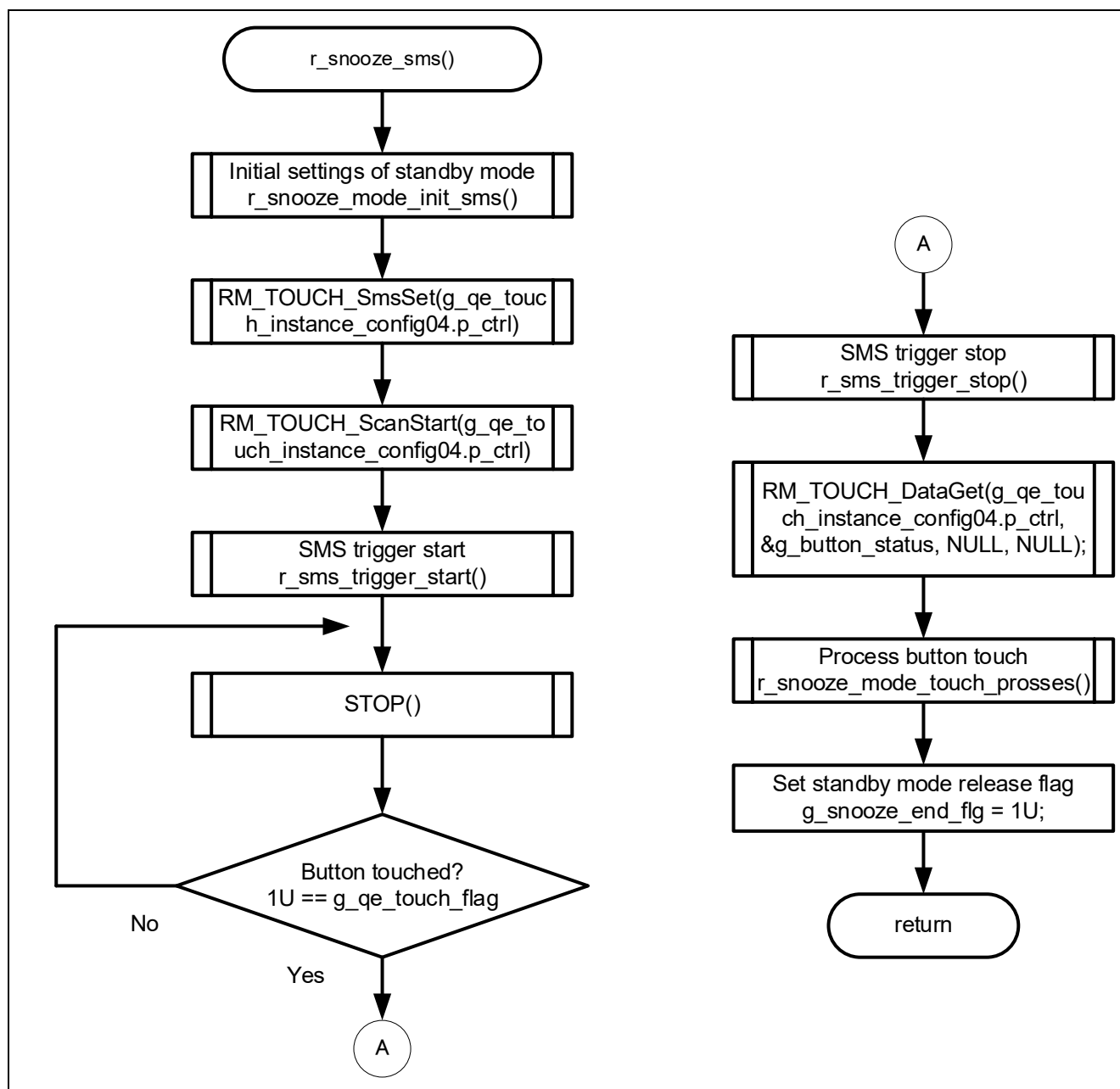


Figure 4-18 Flowchart of r_snooze_sms Function

4.5.10.13 Flowchart of r_touch_mec_scanstart_cpu Function

The flowchart of r_touch_mec_scanstart_cpu function is shown below.

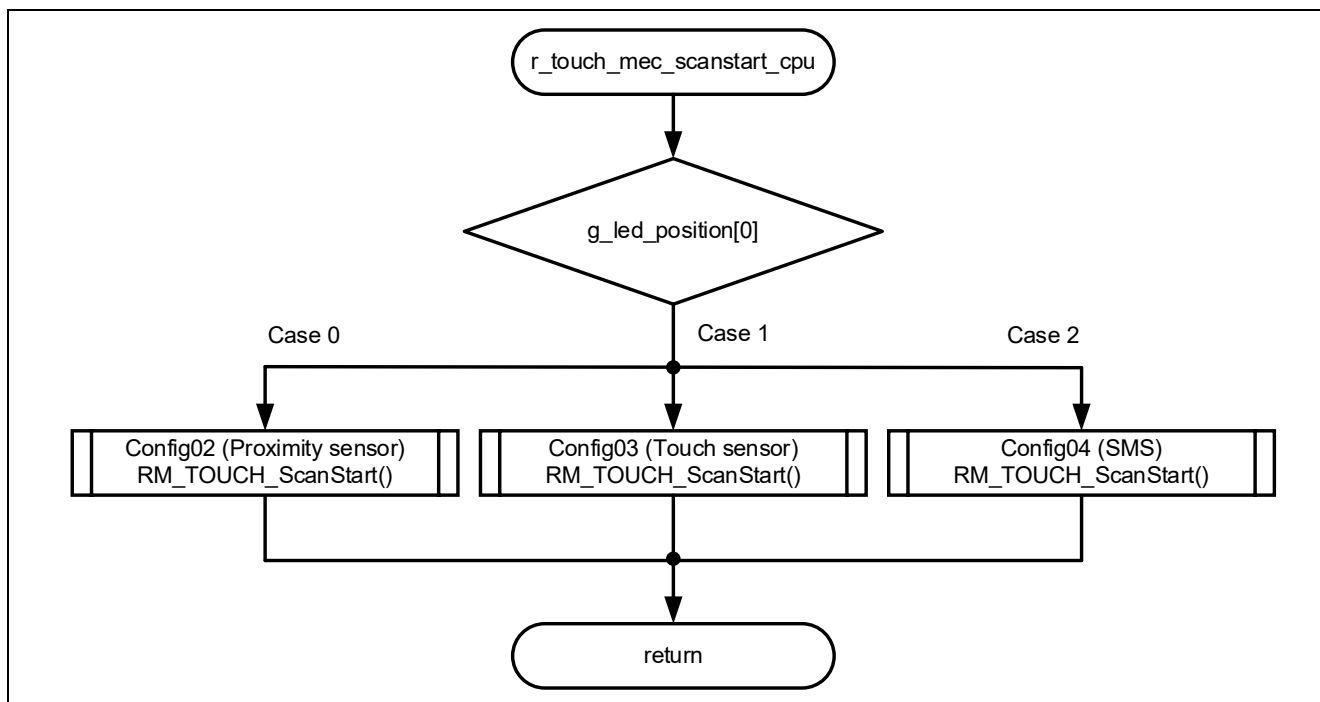


Figure 4-19 Flowchart of r_touch_mec_scanstart_cpu Function

4.5.10.14 Flowchart of r_touch_mec_scanstop Function

The flowchart of r_touch_mec_scanstop function is shown below.

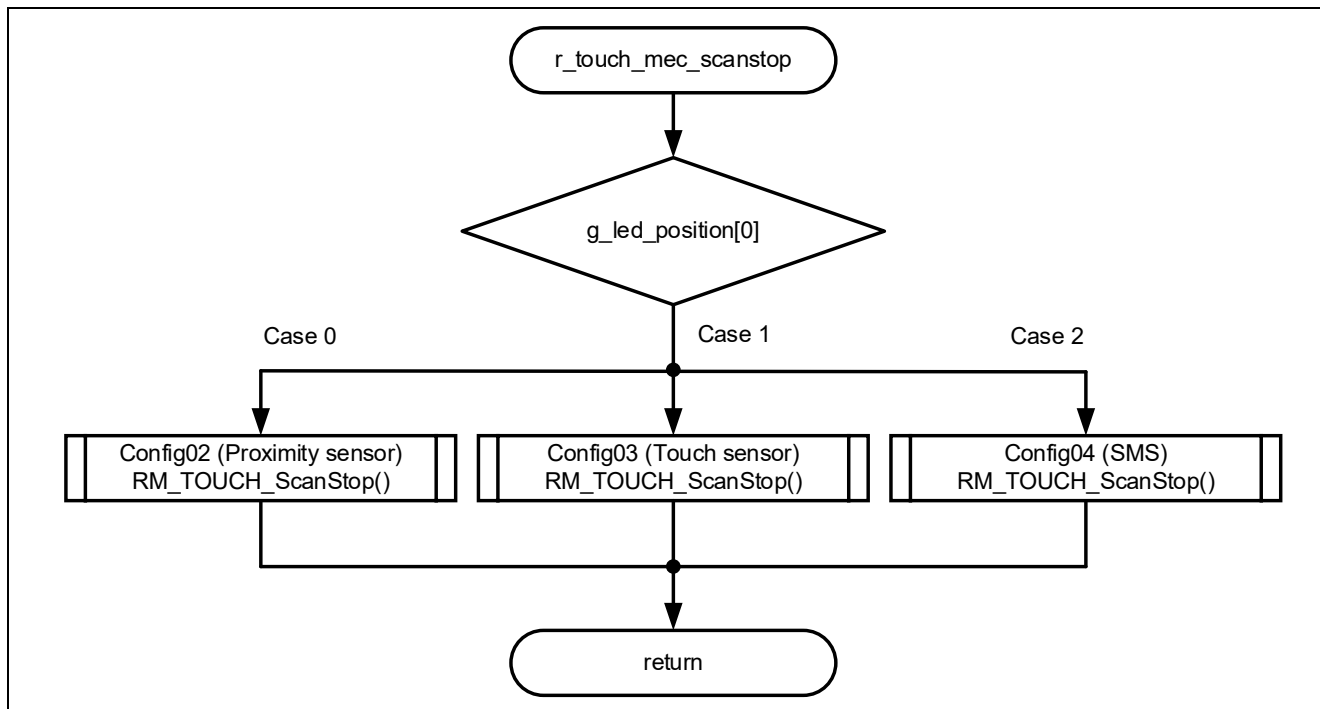


Figure 4-20 Flowchart of r_touch_mec_scanstop Function

4.5.10.15 Flowchart of r_touch_mec_dataget_cpu Function

The flowchart of r_touch_mec_dataget_cpu function is shown below.

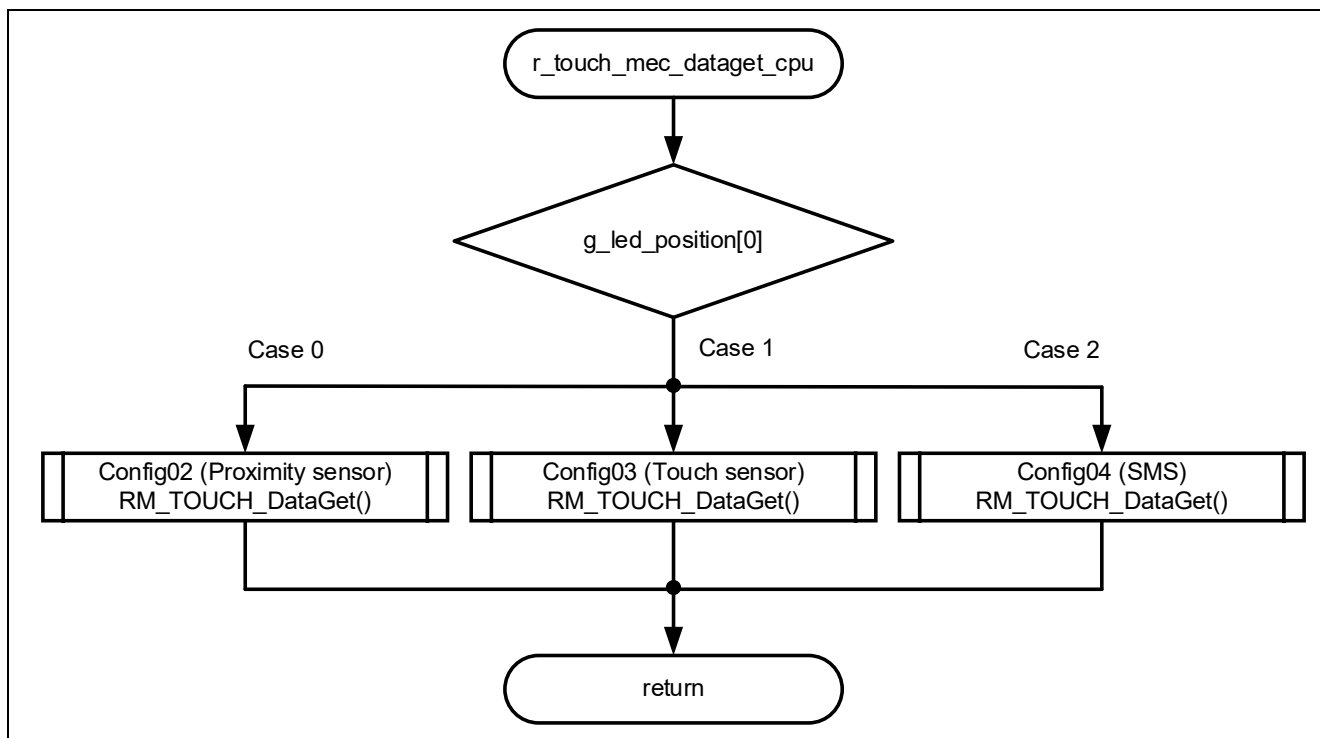


Figure 4-21 Flowchart of r_touch_mec_dataget_cpu Function

4.5.10.16 Flowchart of r_sms_trigger_start Function

The flowchart of r_sms_trigger_start function is shown below.

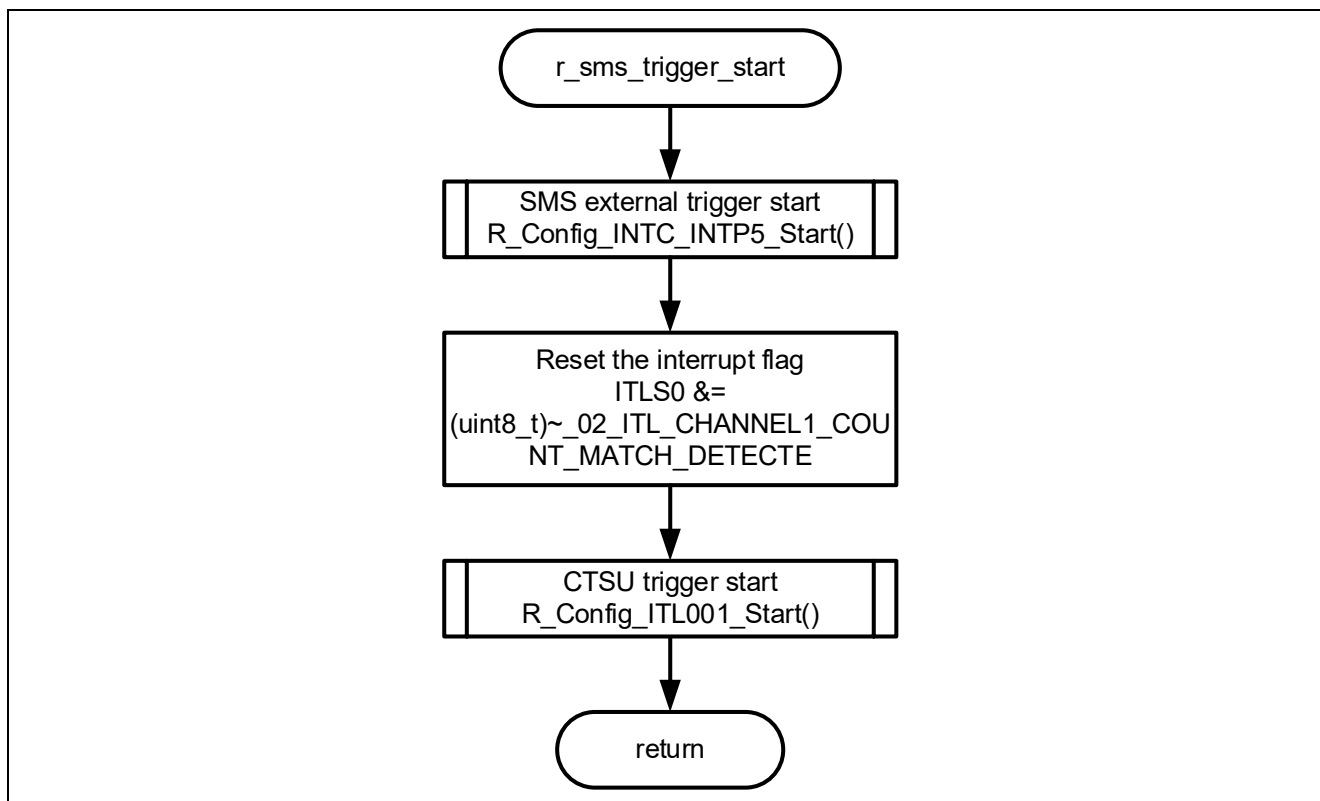


Figure 4-22 Flowchart of r_sms_trigger_start Function

4.5.10.17 Flowchart of r_sms_trigger_stop Function

The flowchart of r_sms_trigger_stop function is shown below.

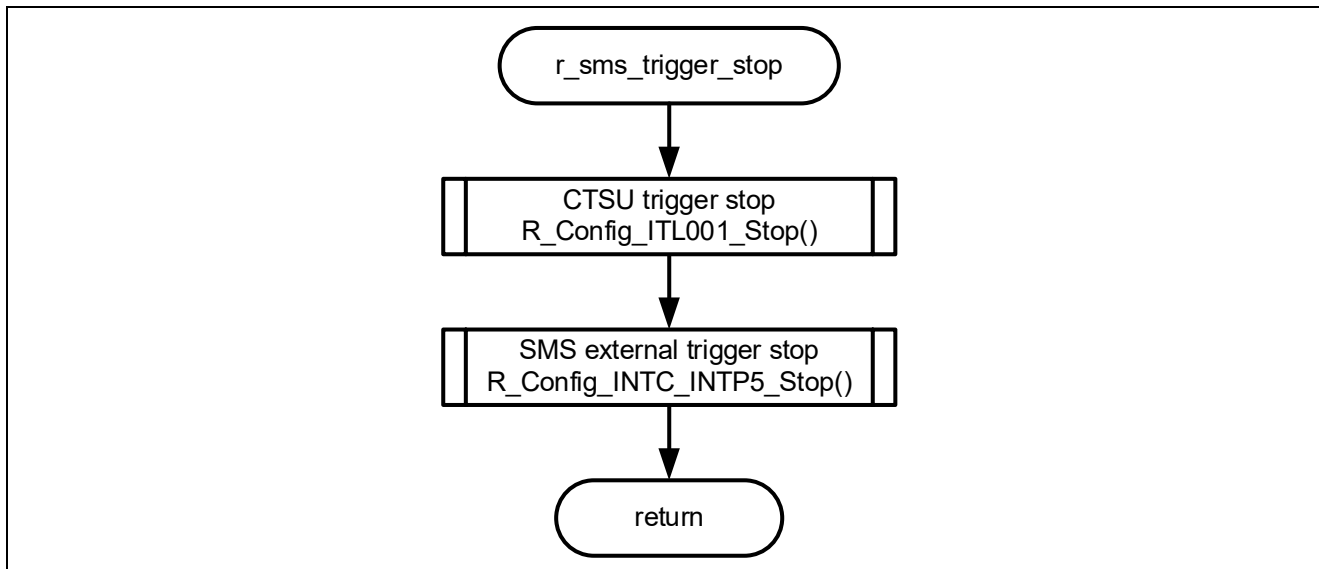


Figure 4-23 Flowchart of r_sms_trigger_stop Function

4.5.10.18 Flowchart of r_normal_mode_init Function

The flowchart of r_normal_mode_init function is shown below.

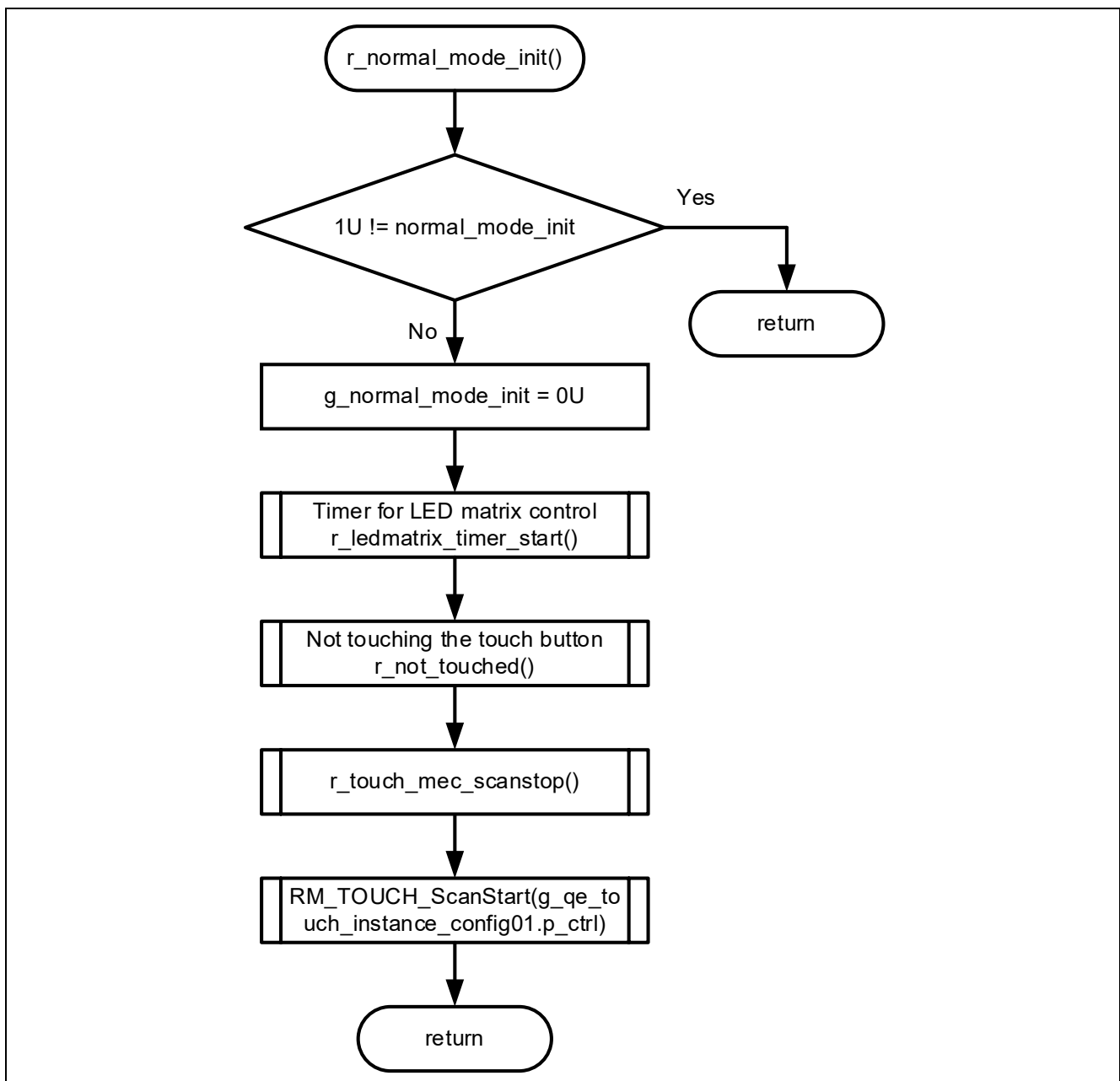


Figure 4-24 Flowchart of r_normal_mode_init Function

4.5.10.19 Flowchart of r_nomal_mode Function

The flowchart of r_nomal_mode function is shown below.

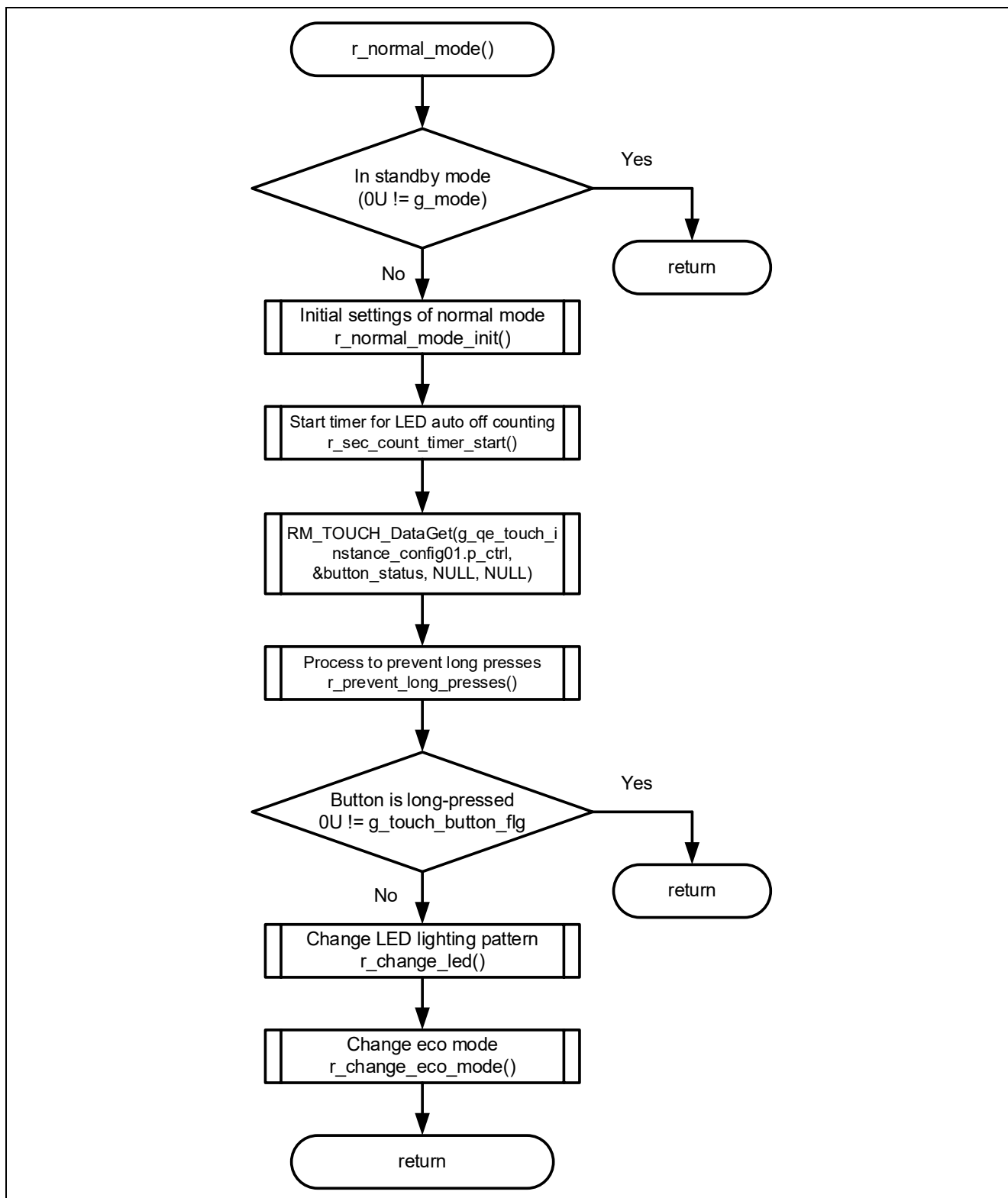


Figure 4-25 Flowchart of r_nomal_mode Function

4.5.10.20 Flowchart of r_change_snooze_nomal Function

The flowchart of r_change_snooze_nomal function is shown below.

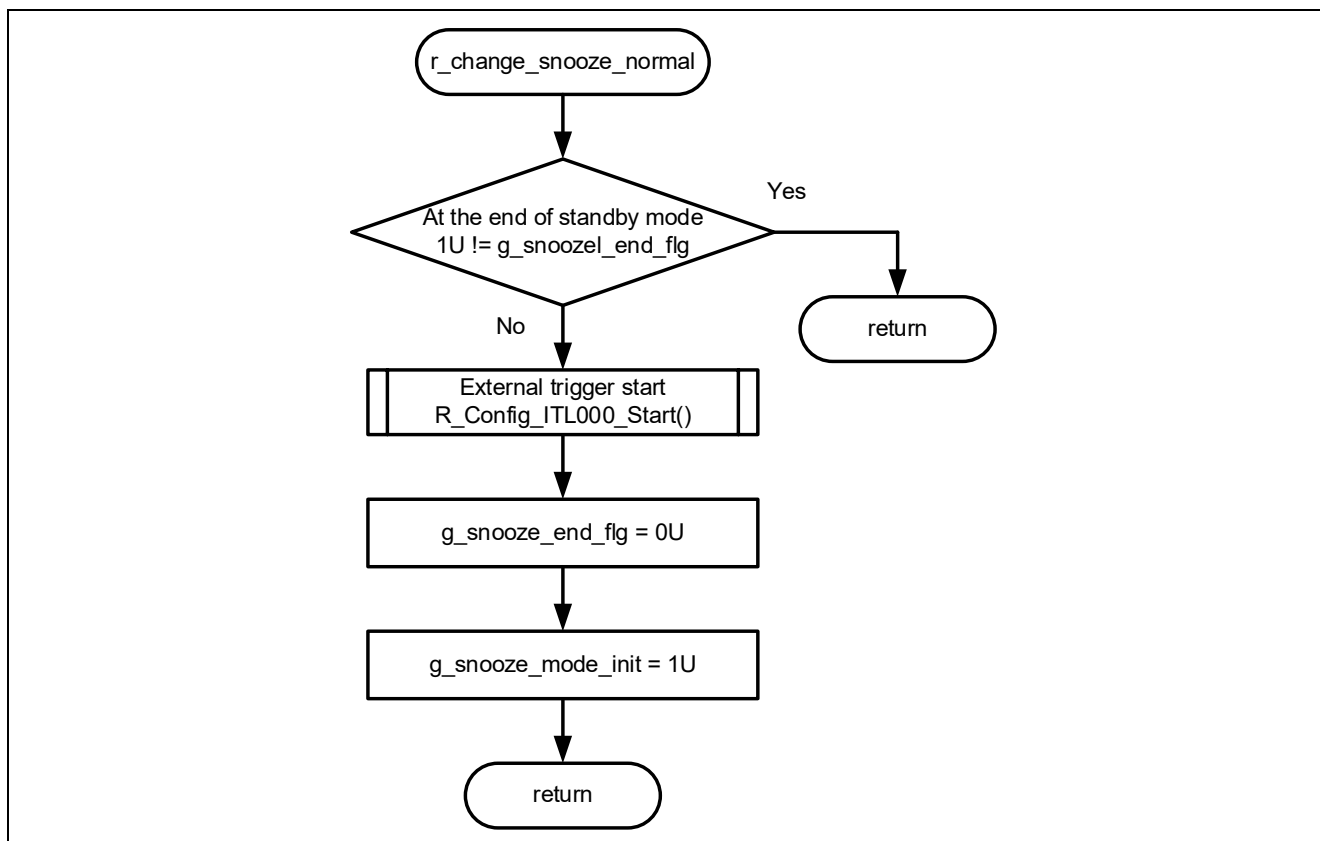


Figure 4-26 Flowchart of r_change_snooze_nomal Function

4.5.10.21 Flowchart of r_change_nomal_snooze Function

The flowchart of r_change_nomal_snooze function is shown below.

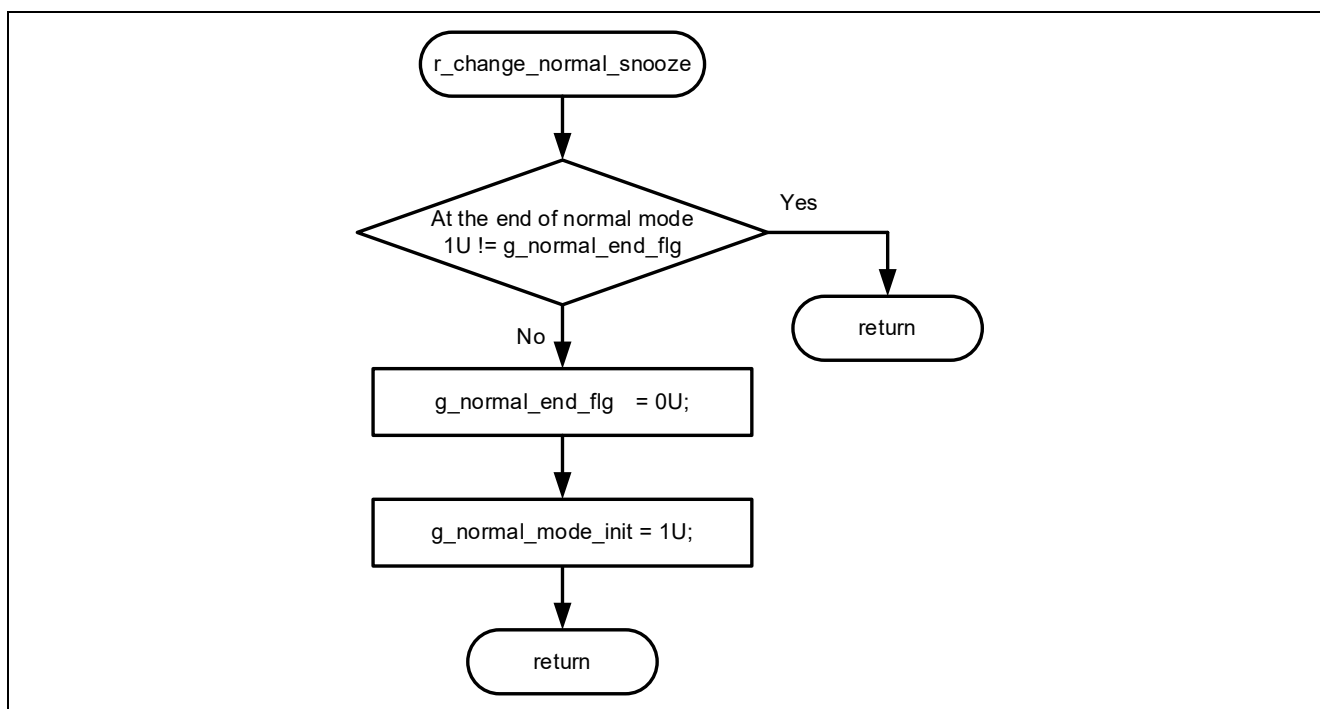


Figure 4-27 Flowchart of r_change_nomal_snooze Function

4.5.10.22 Flowchart of r_not_touched Function

The flowchart of r_not_touched function is shown below.

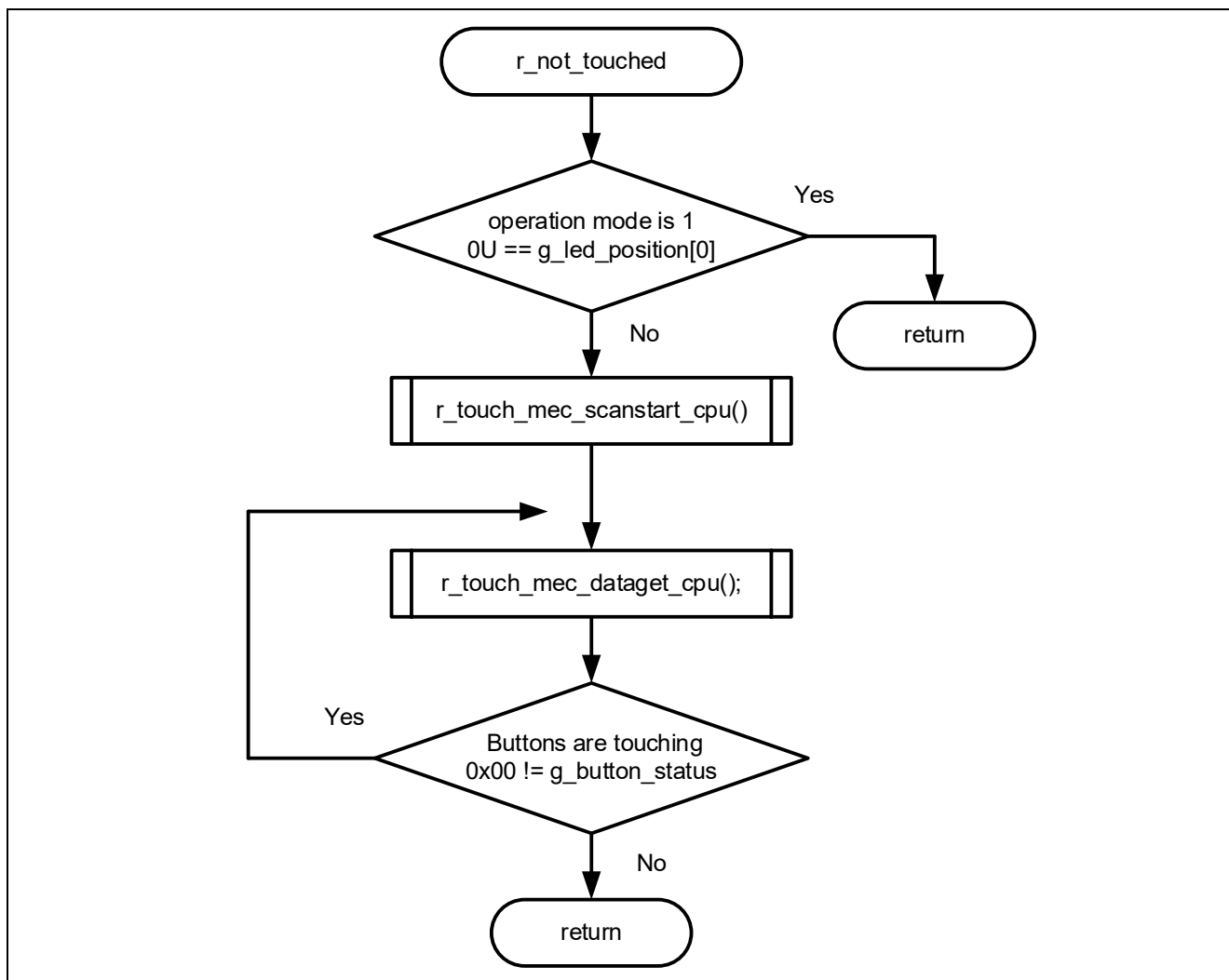


Figure 4-28 Flowchart of r_not_touched Function

4.5.10.23 Flowchart of r_ledport_input Function

The flowchart of r_ledport_input function is shown below.

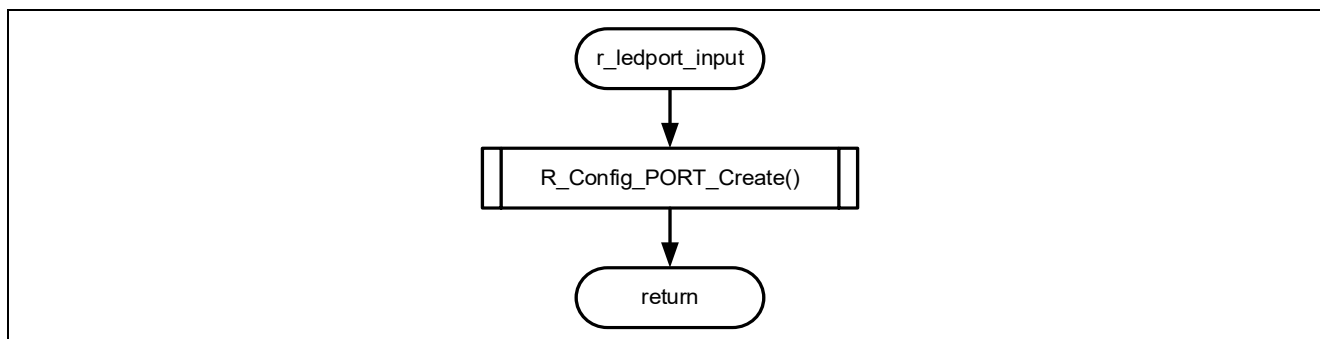


Figure 4-29 Flowchart of r_ledport_input Function

4.5.10.24 Flowchart of r_ledport_output Function

The flowchart of r_ledport_output function is shown below.

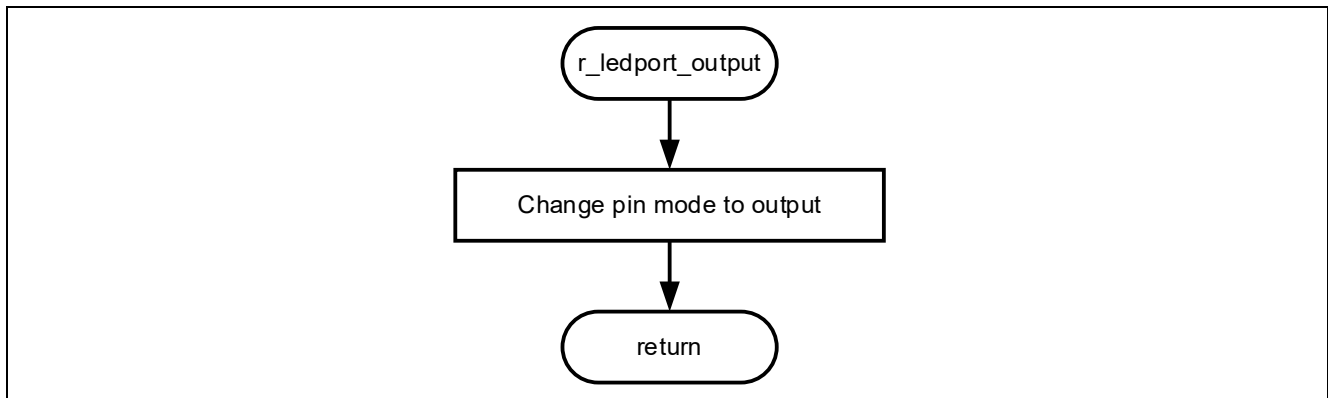


Figure 4-30 Flowchart of r_ledport_output Function

4.5.10.25 Flowchart of r_led_init Function

The flowchart of r_led_init function is shown below.

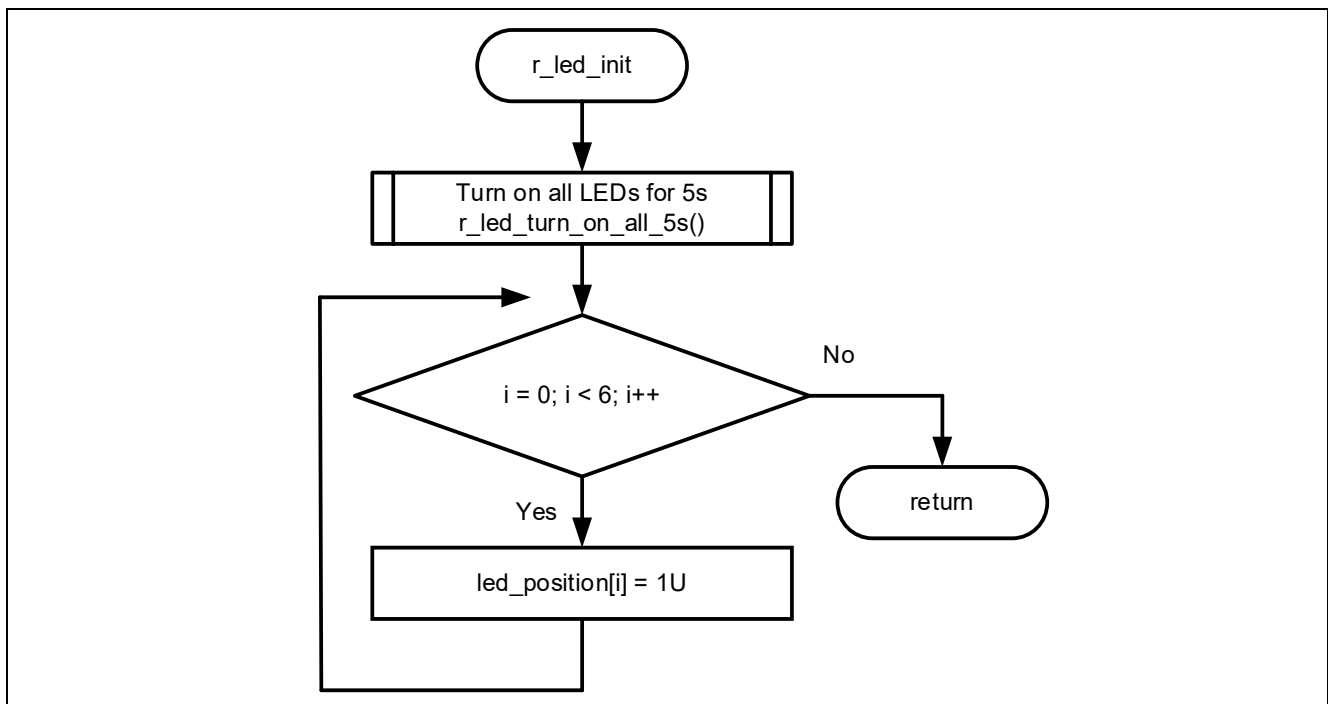


Figure 4-31 Flowchart of r_led_init Function

4.5.10.26 Flowchart of r_led_turn_on_all_5s Function

The flowchart of r_led_turn_on_all_5s function is shown below.

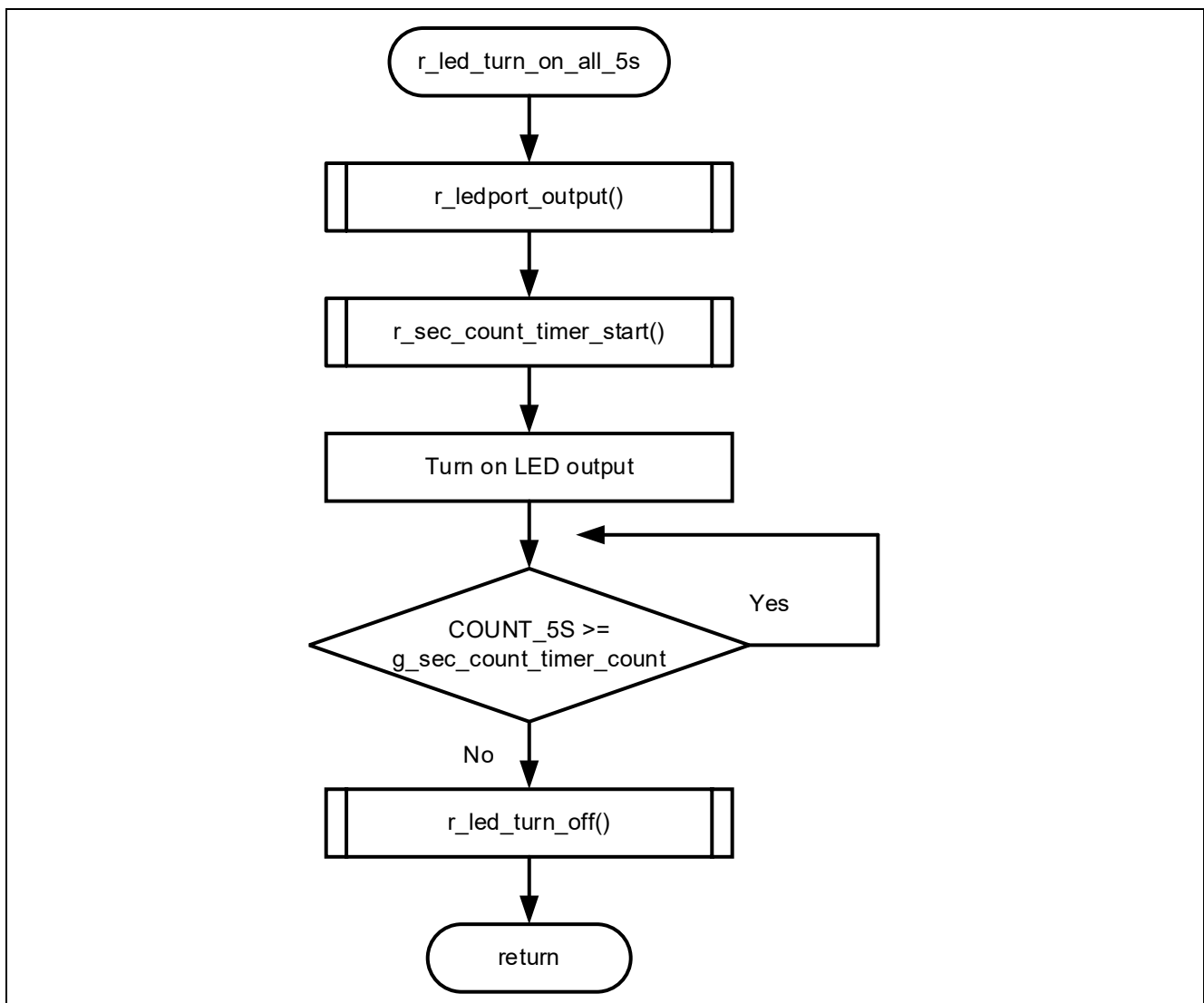


Figure 4-32 Flowchart of r_led_turn_on_all_5s Function

4.5.10.27 Flowchart of r_change_led Function

The flowchart of r_change_led function is shown below.

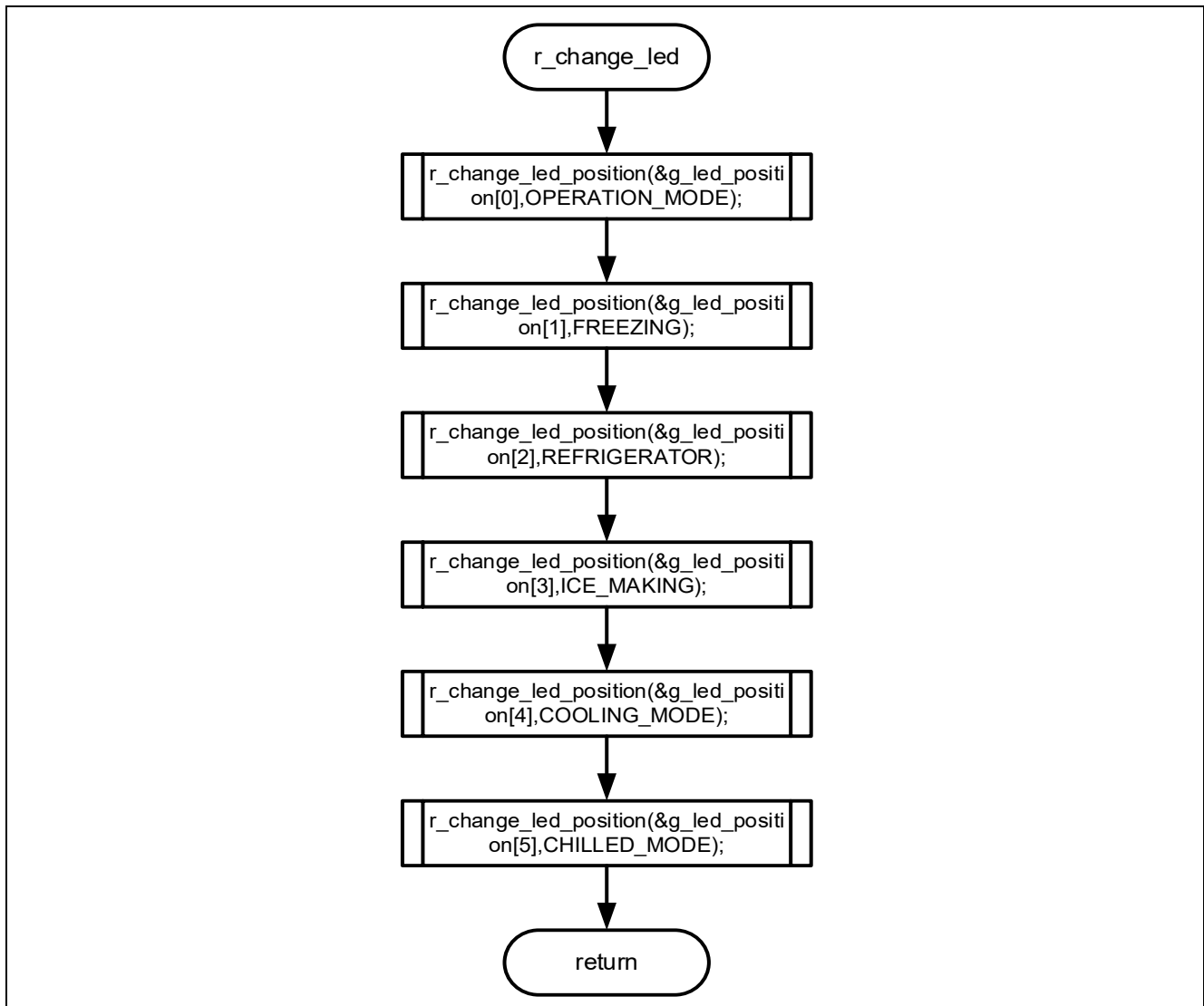


Figure 4-33 Flowchart of r_change_led Function

4.5.10.28 Flowchart of r_change_led_position Function

The flowchart of r_change_led_position function is shown below.

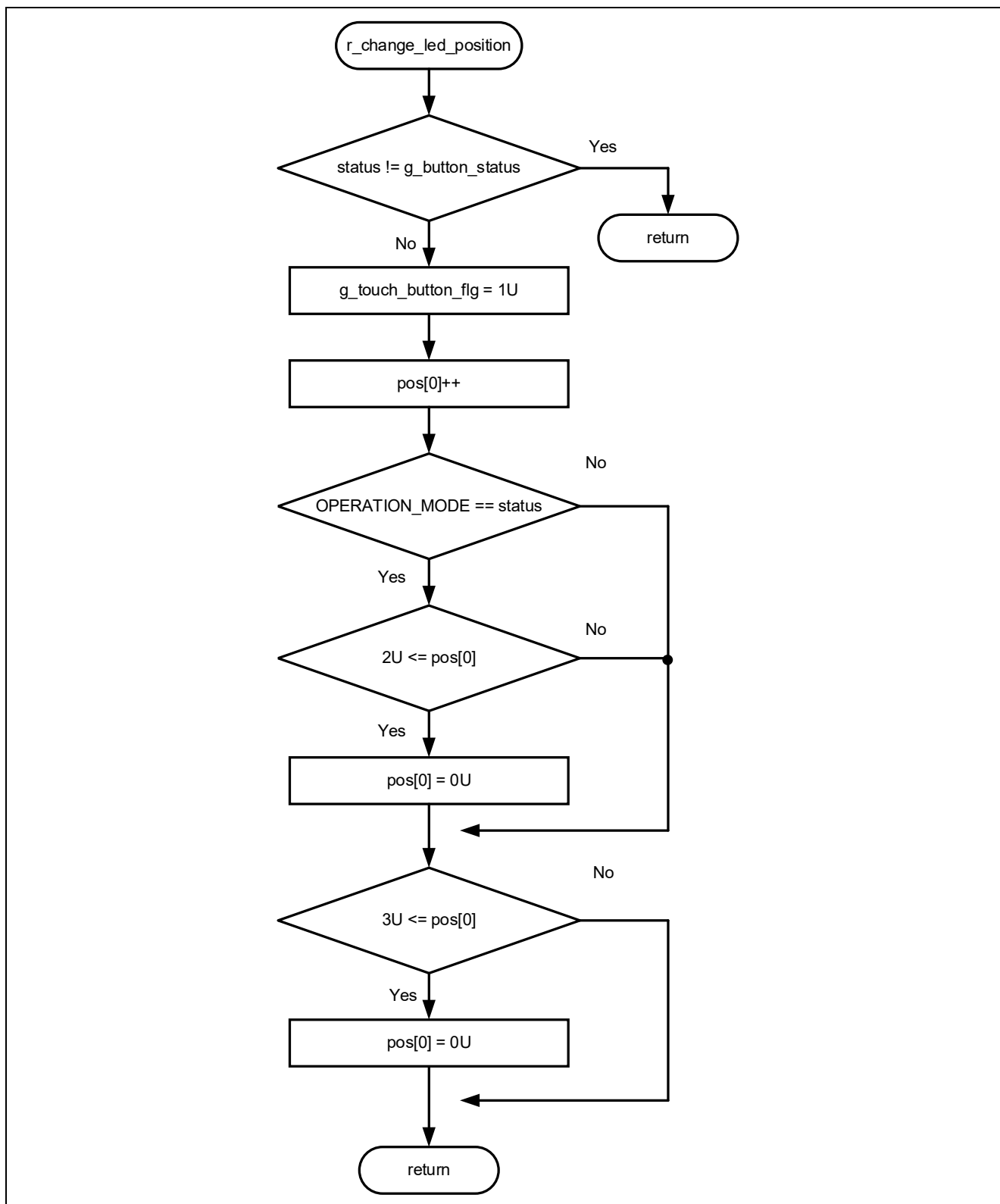


Figure 4-34 Flowchart of r_change_led_position Function

4.5.10.29 Flowchart of r_led_turn_on Function

The flowchart of r_led_turn_on function is shown below.

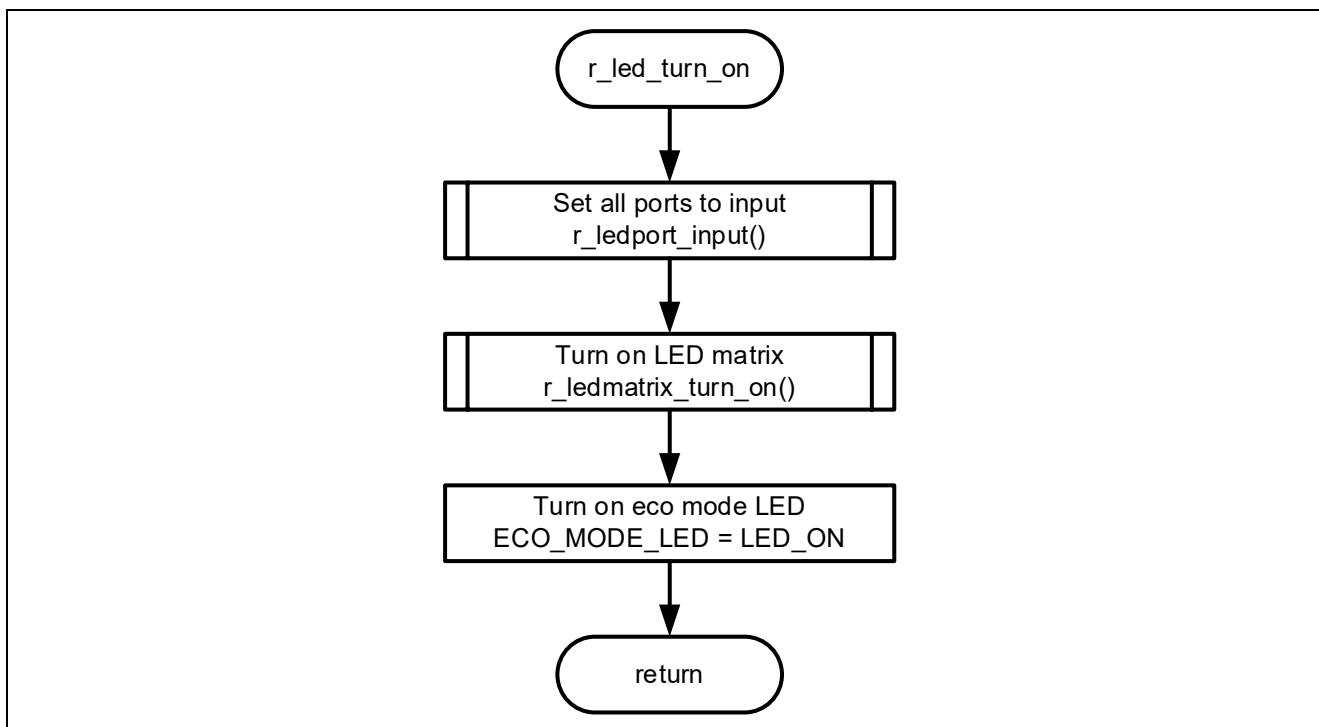


Figure 4-35 Flowchart of r_led_turn_on Function

4.5.10.30 Flowchart of r_led_turn_off Function

The flowchart of r_led_turn_off function is shown below.

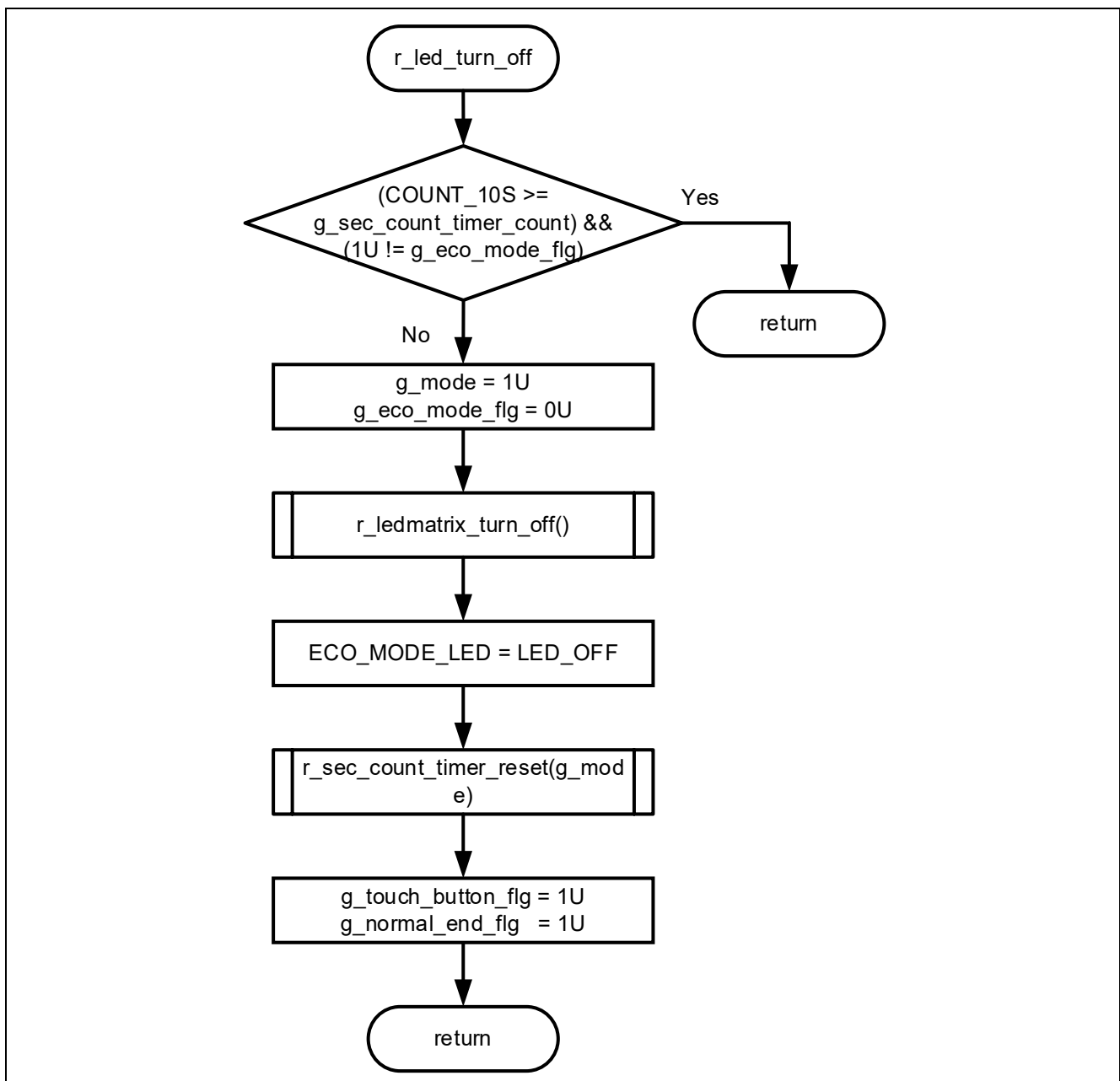


Figure 4-36 Flowchart of r_led_turn_off Function

4.5.10.31 Flowchart of r_ledmatrix_turn_on Function

The flowchart of r_ledmatrix_turn_on function is shown below.

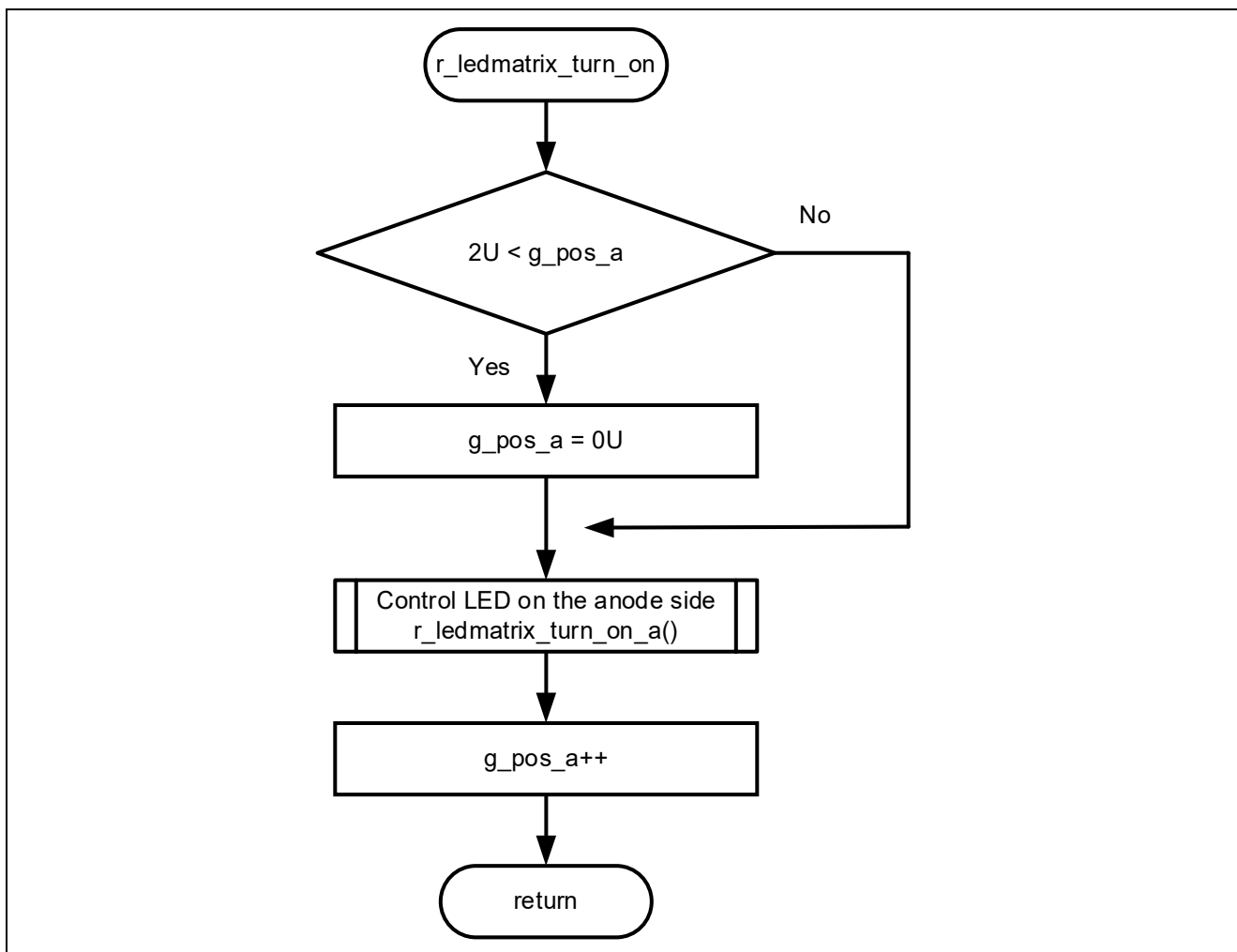


Figure 4-37 Flowchart of r_ledmatrix_turn_on Function

4.5.10.32 Flowchart of r_ledmatrix_turn_off Function

The flowchart of r_ledmatrix_turn_off function is shown below.

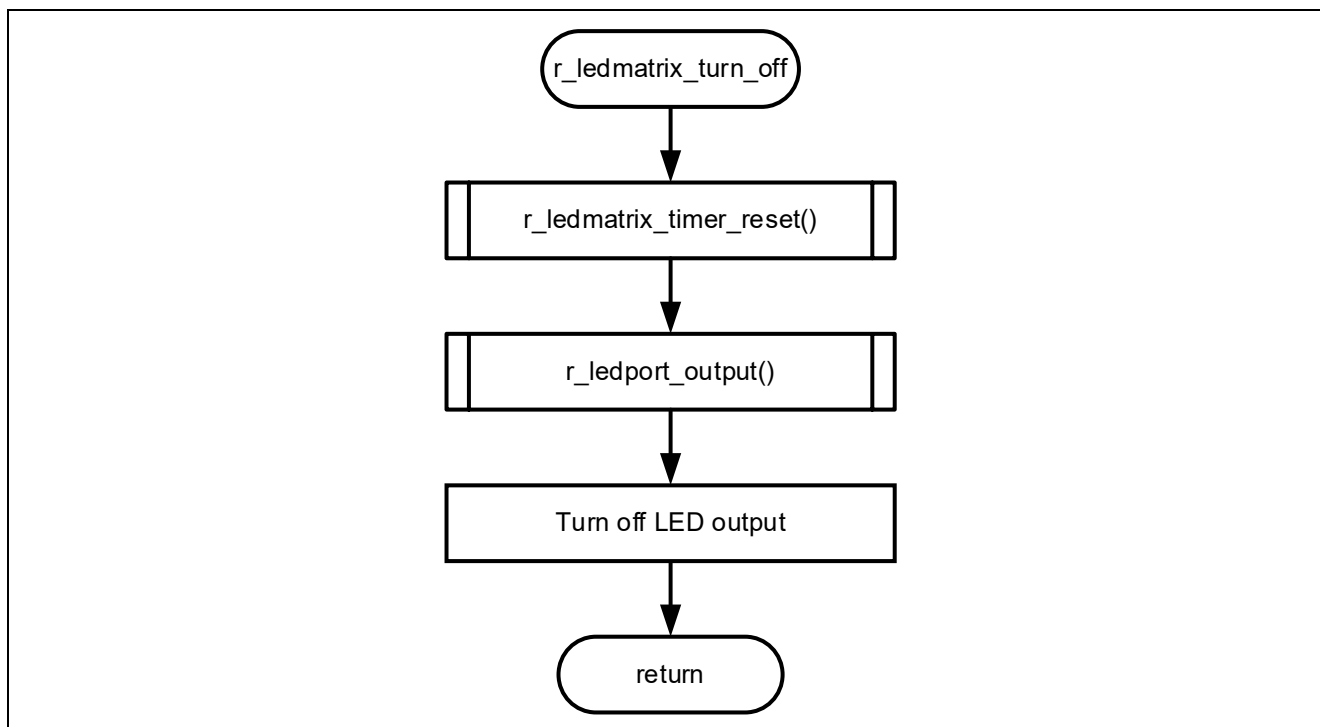


Figure 4-38 Flowchart of r_ledmatrix_turn_off Function

4.5.10.33 Flowchart of r_ledmatrix_turn_on_a Function

The flowchart of r_ledmatrix_turn_on_a function is shown below.

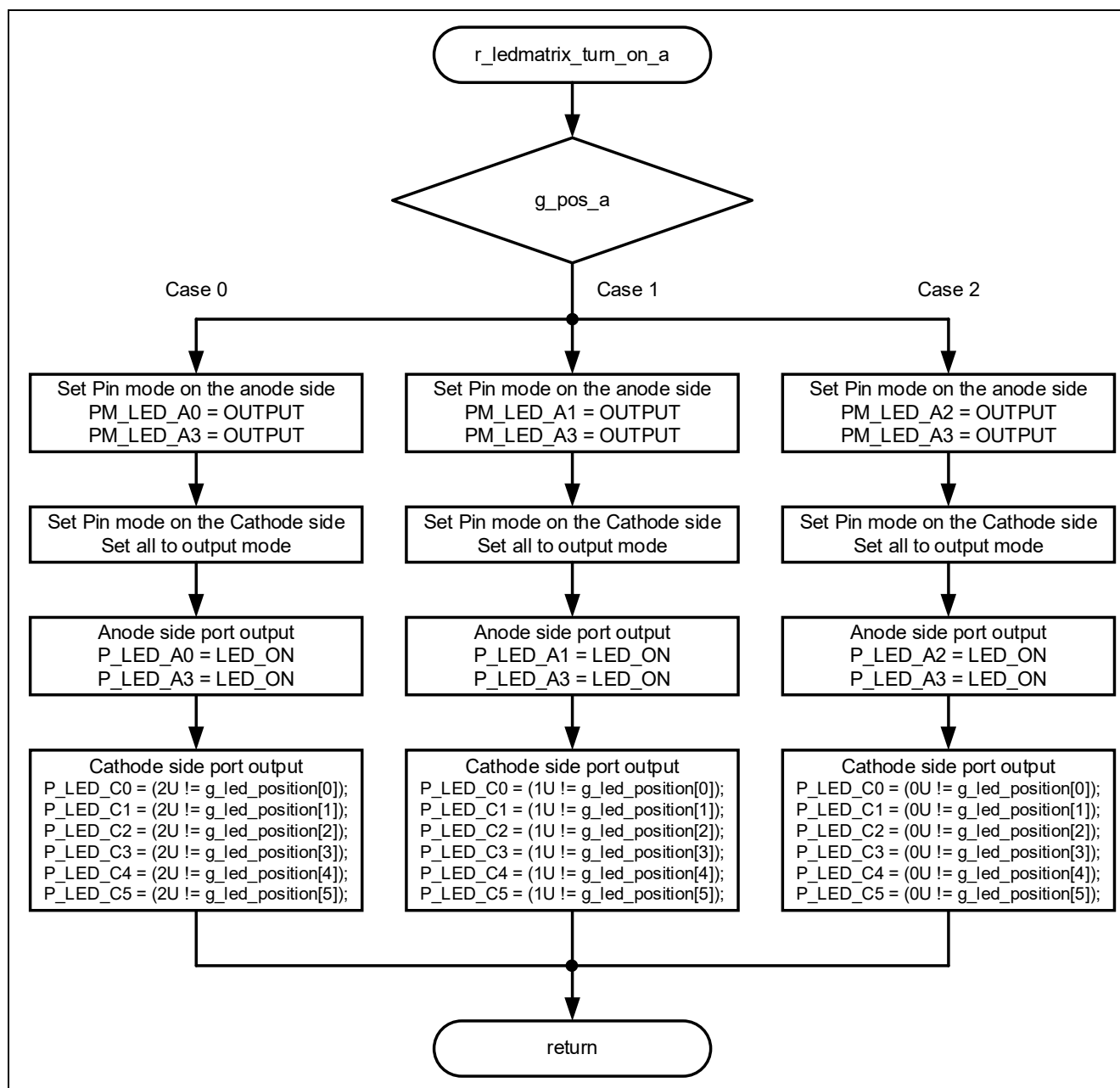


Figure 4-39 Flowchart of r_ledmatrix_turn_on_a Function

4.5.10.34 Flowchart of r_Config_TAU0_0_interrupt Function

The flowchart of r_Config_TAU0_0_interrupt function is shown below.

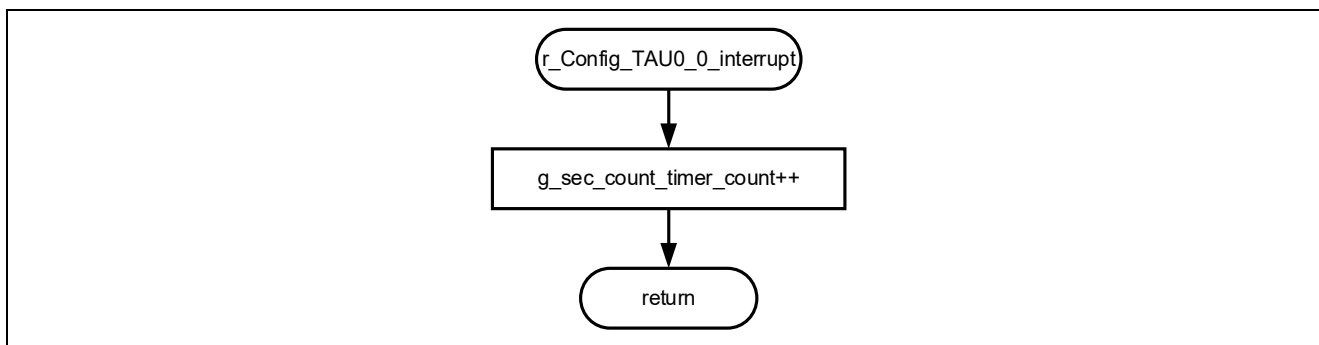


Figure 4-40 Flowchart of r_Config_TAU0_0_interrupt Function

4.5.10.35 Flowchart of r_sec_count_timer_start Function

The flowchart of `r_sec_count_timer_start` function is shown below.

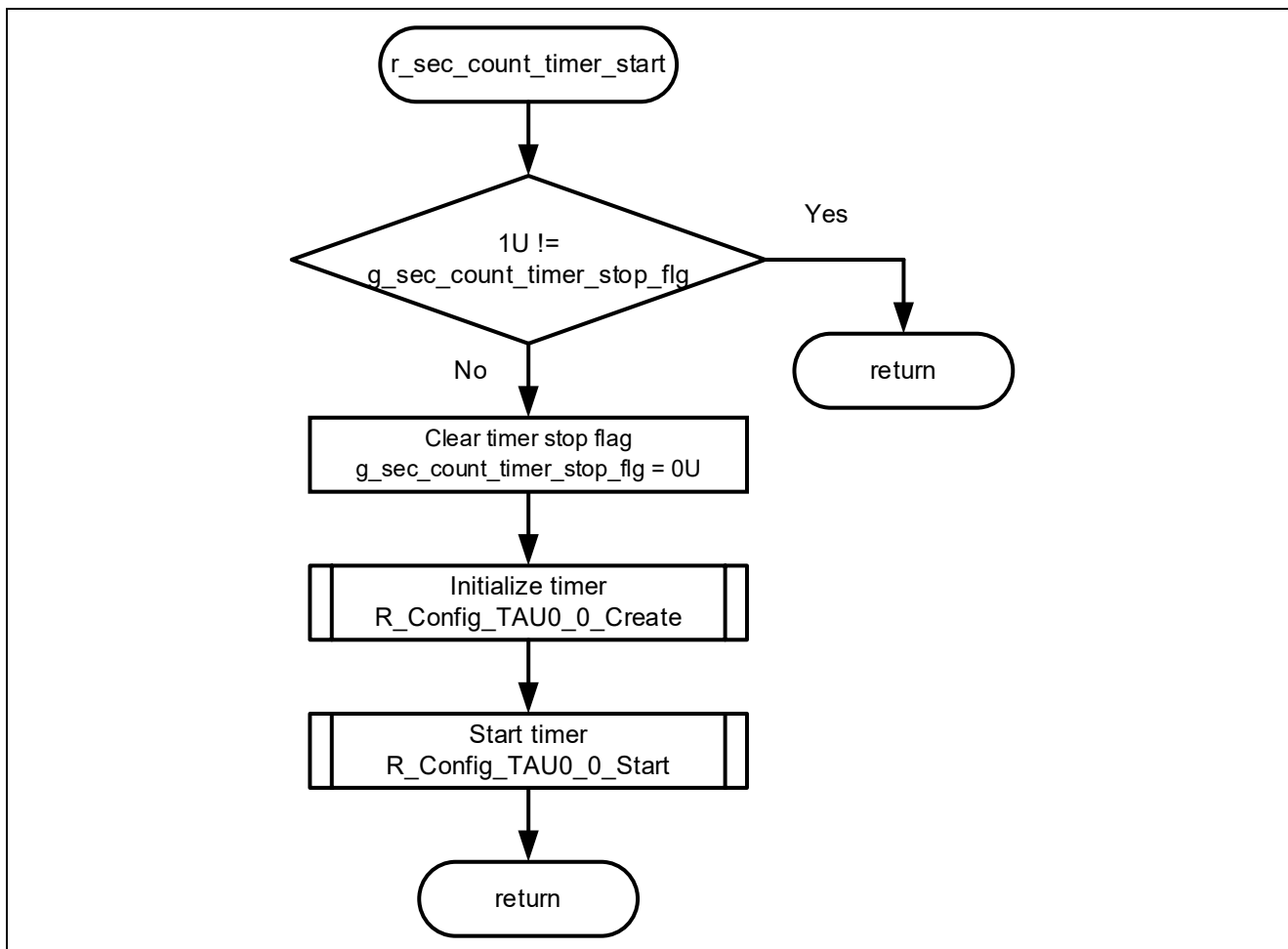


Figure 4-41 Flowchart of r_sec_count_timer_start Function

4.5.10.36 Flowchart of r_sec_count_timer_reset Function

The flowchart of r_sec_count_timer_reset function is shown below.

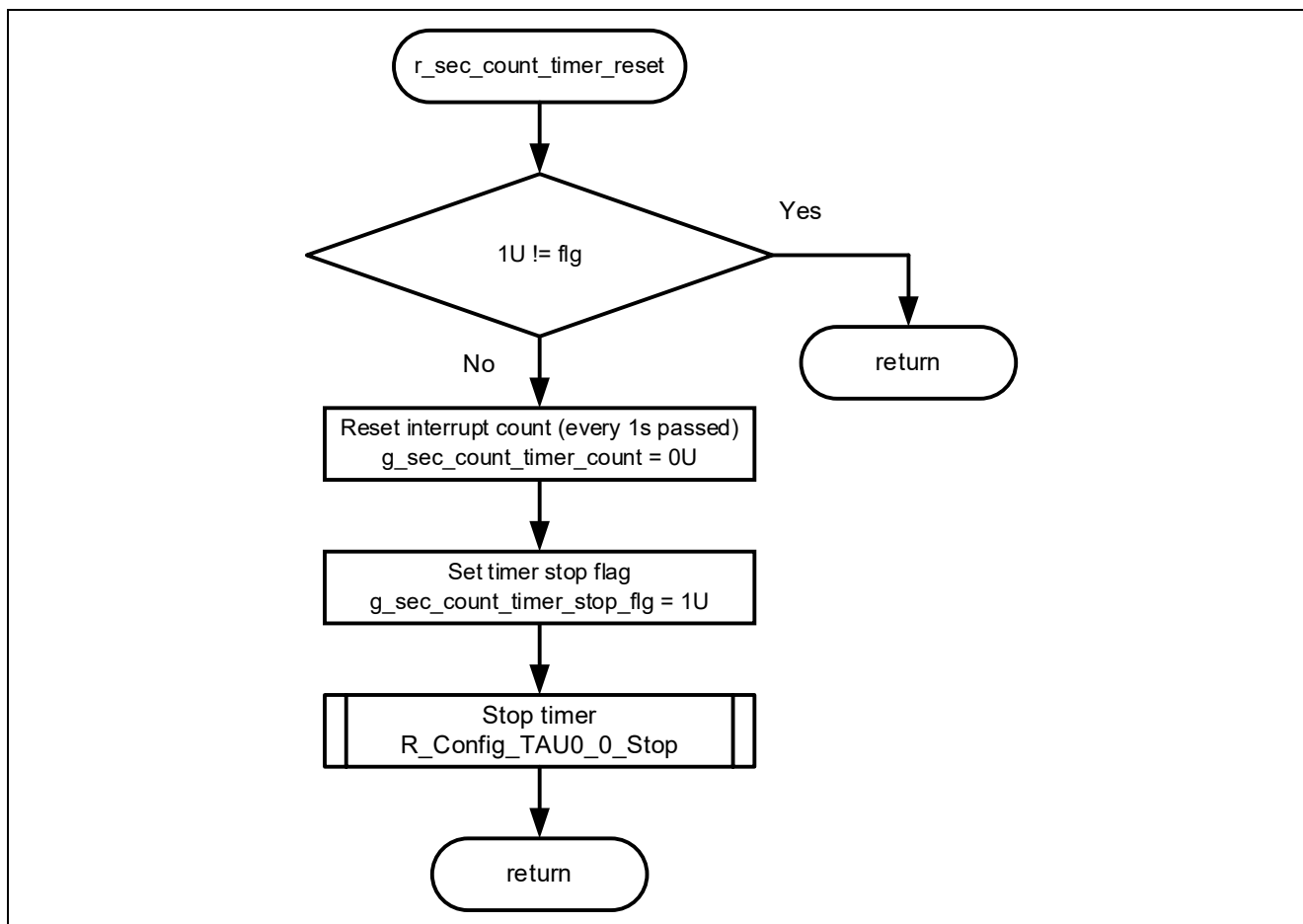


Figure 4-42 Flowchart of r_sec_count_timer_reset Function

4.5.10.37 Flowchart of r_Config_TAU0_1_interrupt Function

The flowchart of r_Config_TAU0_1_interrupt function is shown below.

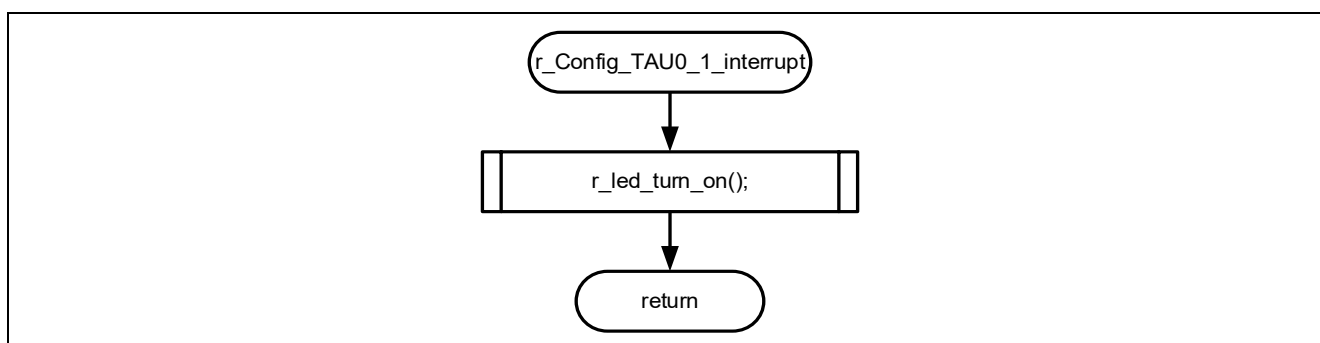


Figure 4-43 Flowchart of r_Config_TAU0_1_interrupt Function

4.5.10.38 Flowchart of r_ledmatrix_timer_start Function

The flowchart of r_ledmatrix_timer_start function is shown below.

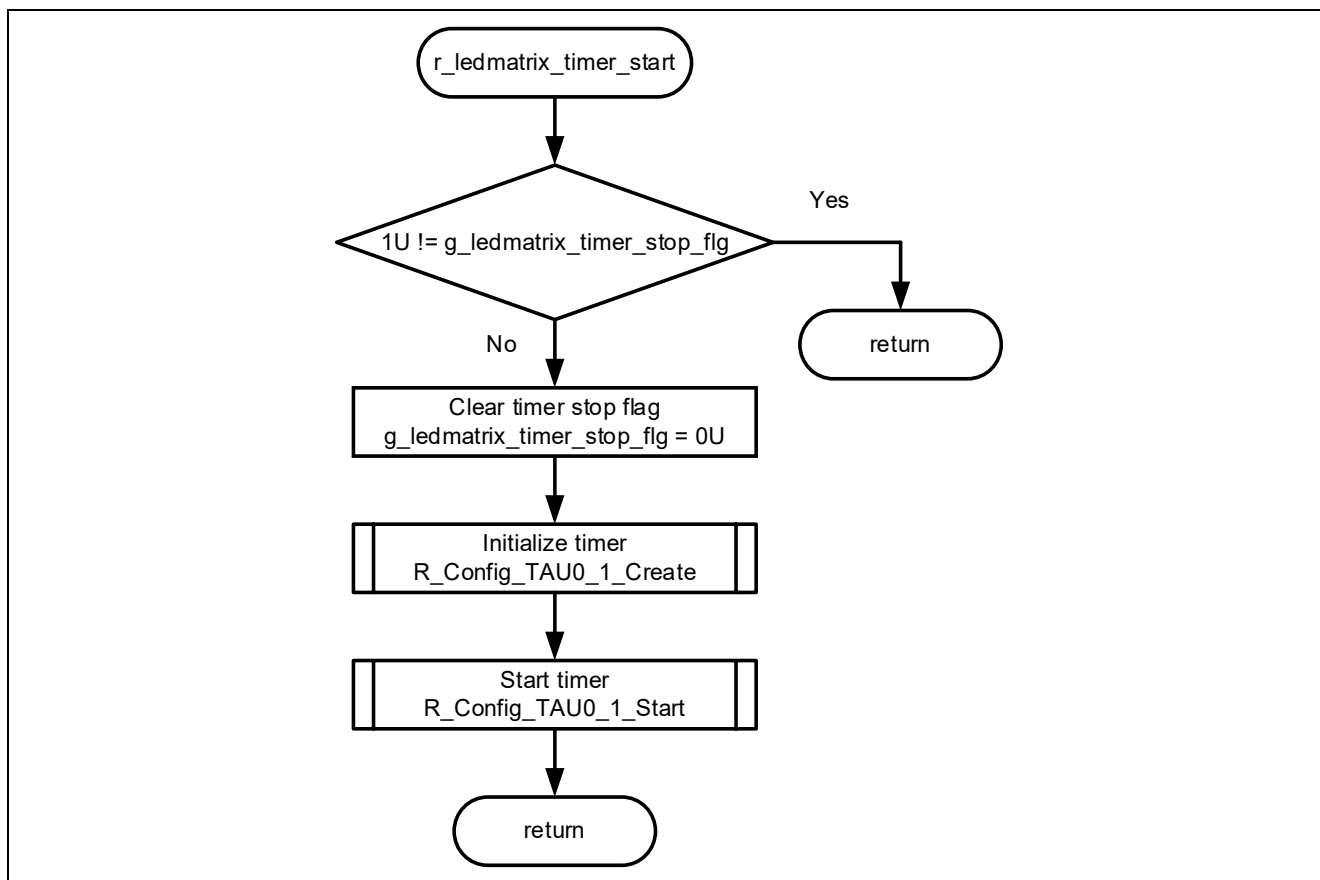


Figure 4-44 Flowchart of r_ledmatrix_timer_start Function

4.5.10.39 Flowchart of r_ledmatrix_timer_reset Function

The flowchart of r_ledmatrix_timer_reset function is shown below.

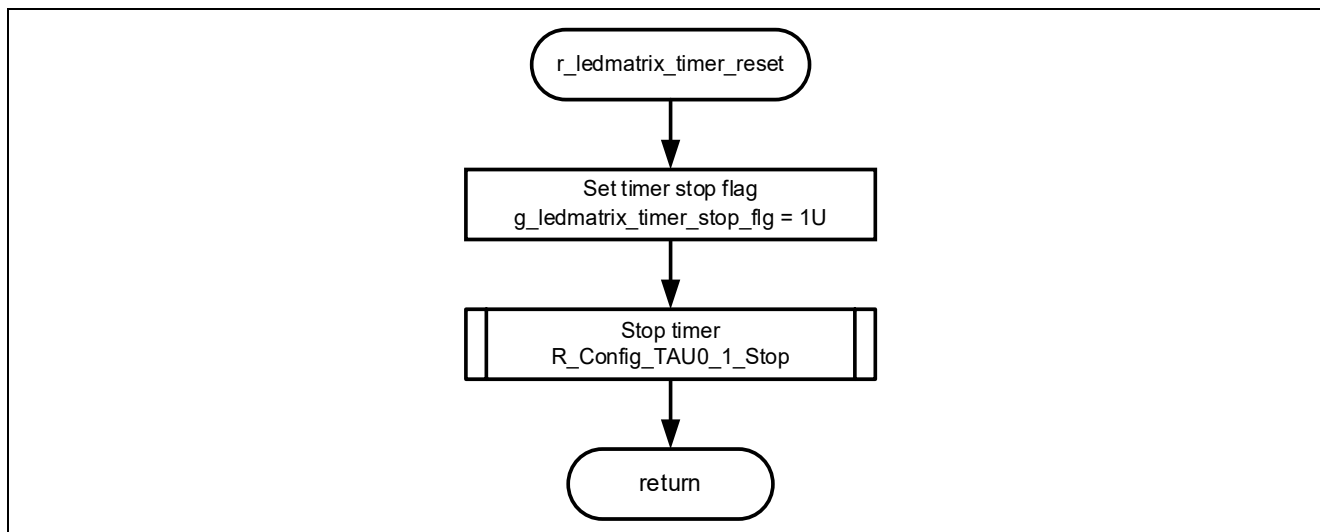


Figure 4-45 Flowchart of r_ledmatrix_timer_reset Function

5. 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.

5.1 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.)

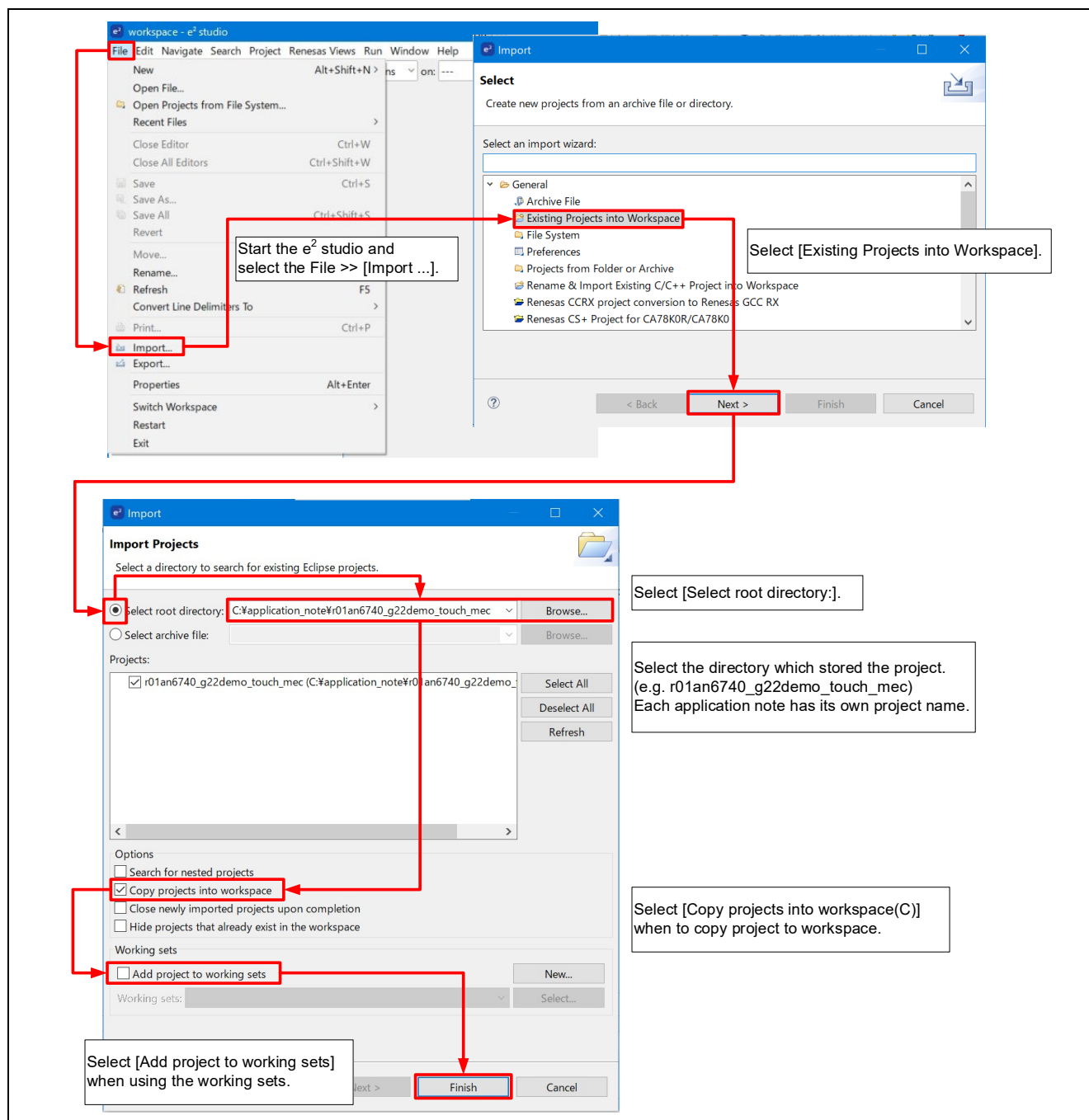


Figure 5-1 Import a Project into e² studio

5.2 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.)

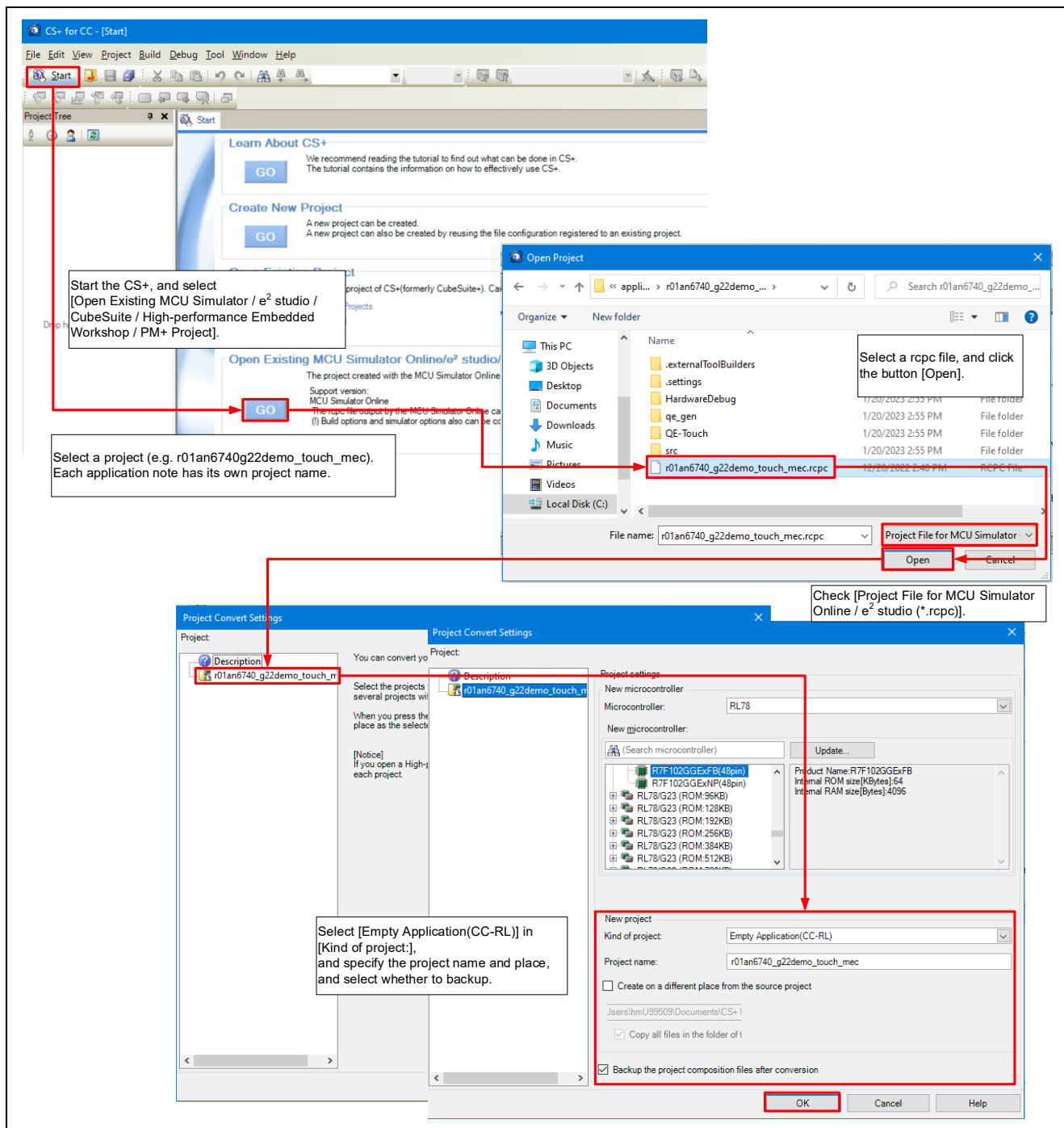


Figure 5-2 Import a Project into CS+

6. Starting a Demonstration

When the E2 emulator Lite connector is disconnected and Appliance UI Demo 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.

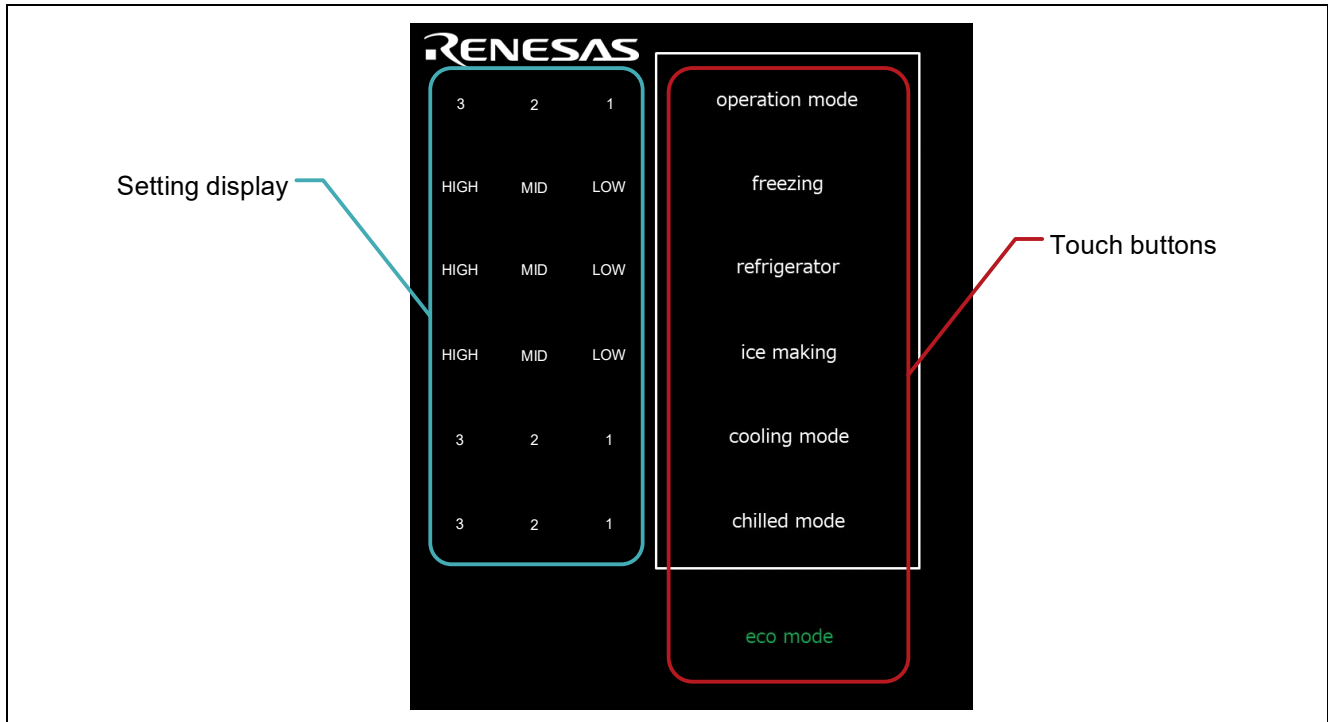


Figure 6-1 Demonstration Operation Panel

6.1 Powered on Appliance UI Demo and Menu Screen

When Appliance UI Demo is powered on, all characters on the touch panel are displayed for approximately 5 seconds. After the display finishes, the demonstration program starts and Appliance UI Demo transits the standby mode (operation mode1).

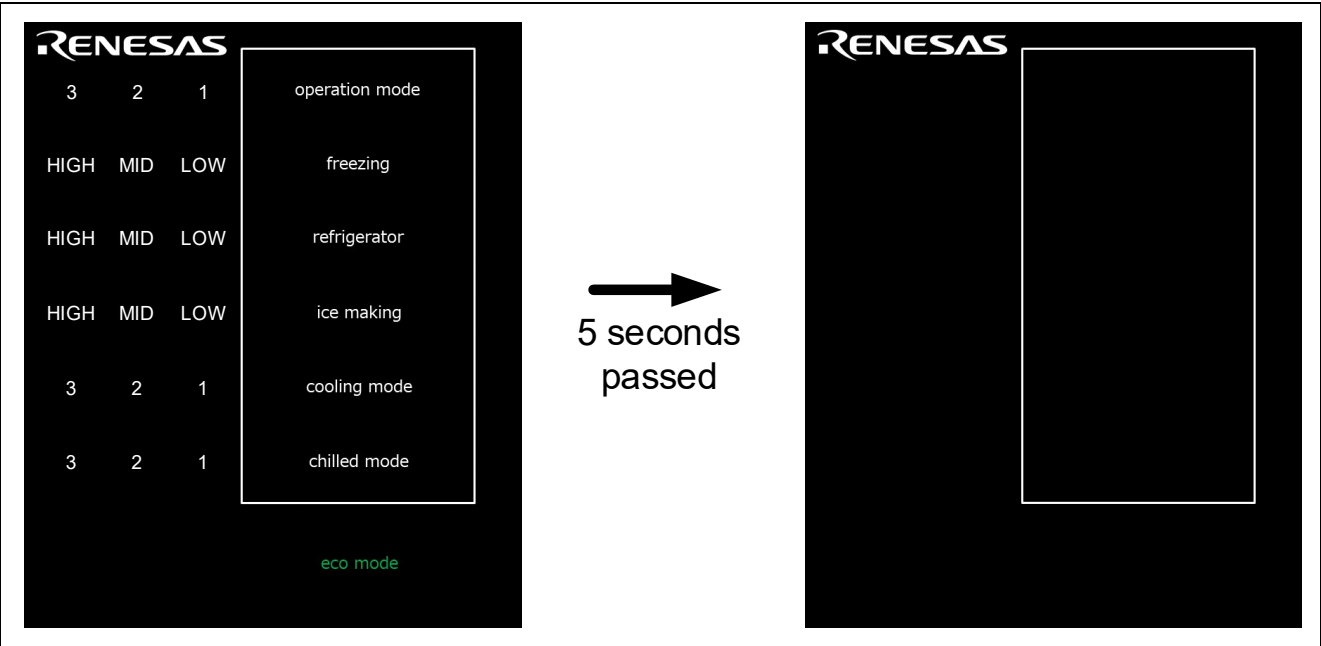


Figure 6-2 Start of the Demonstration

6.2 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.

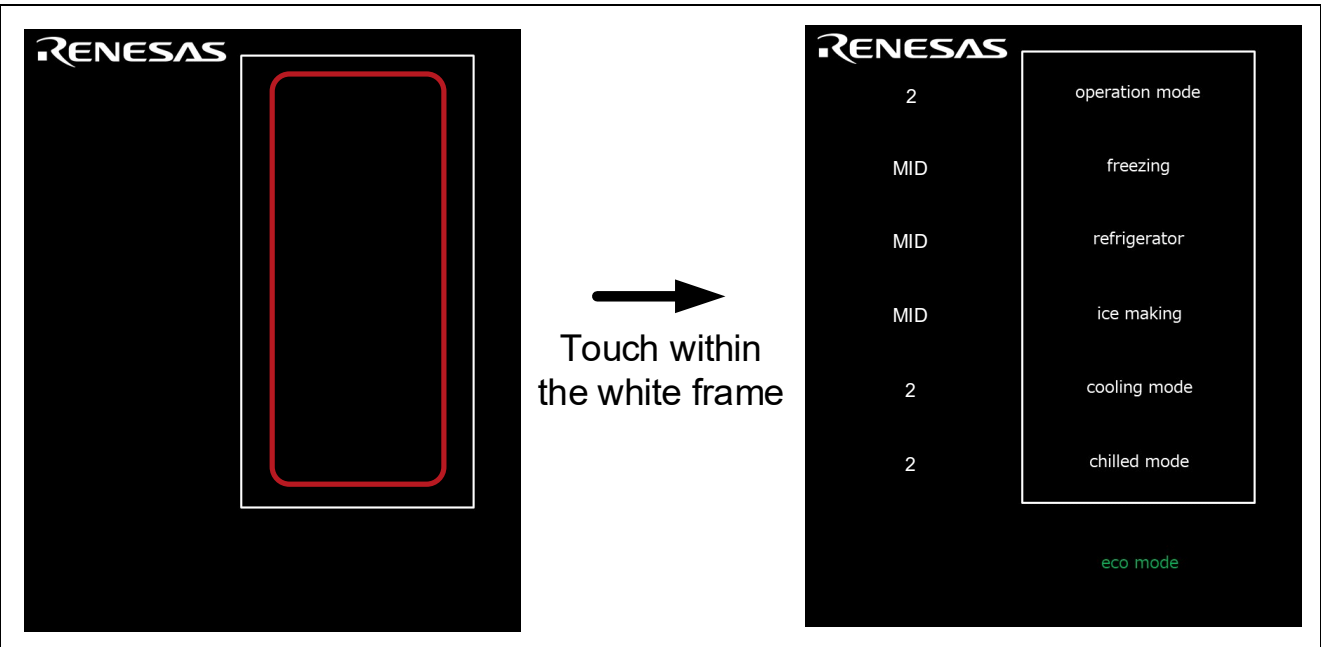


Figure 6-3 How to Operate the Menu Screen

6.3 Touch Operation

6.3.1 Set operation mode

By touching the operation mode button, the setting values can be changed in the order shown in Figure 6-4.

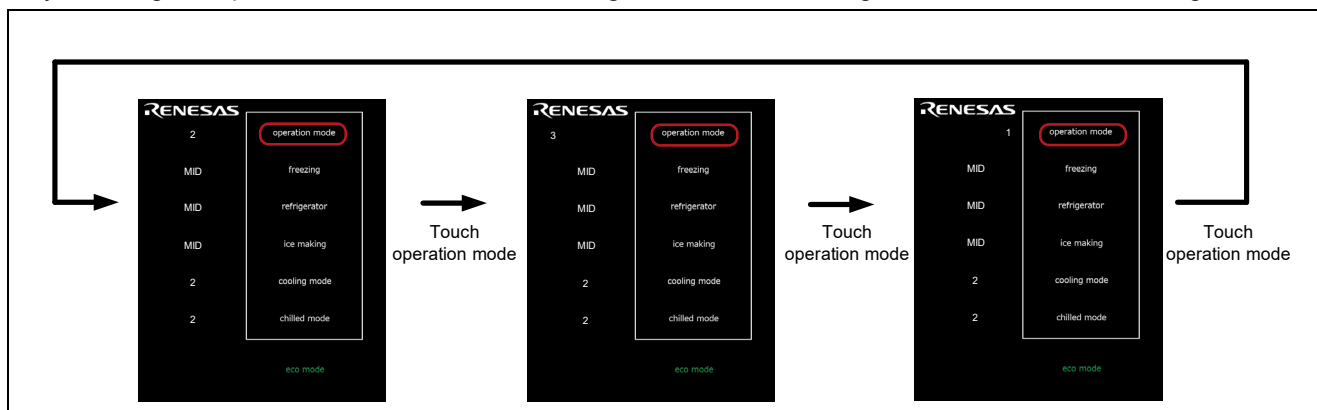


Figure 6-4 Set operation mode

6.3.2 Set freezing

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

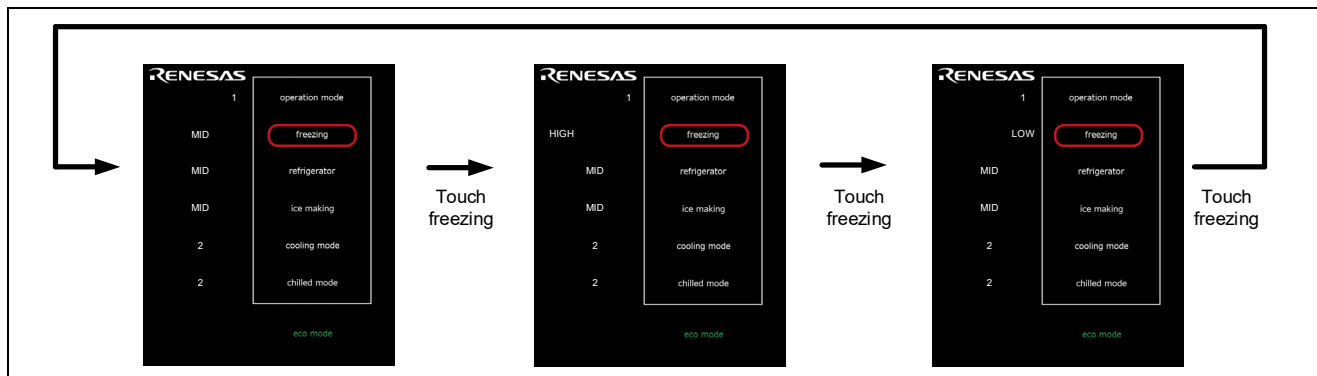


Figure 6-5 Set freezing

6.3.3 Set refrigerator

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

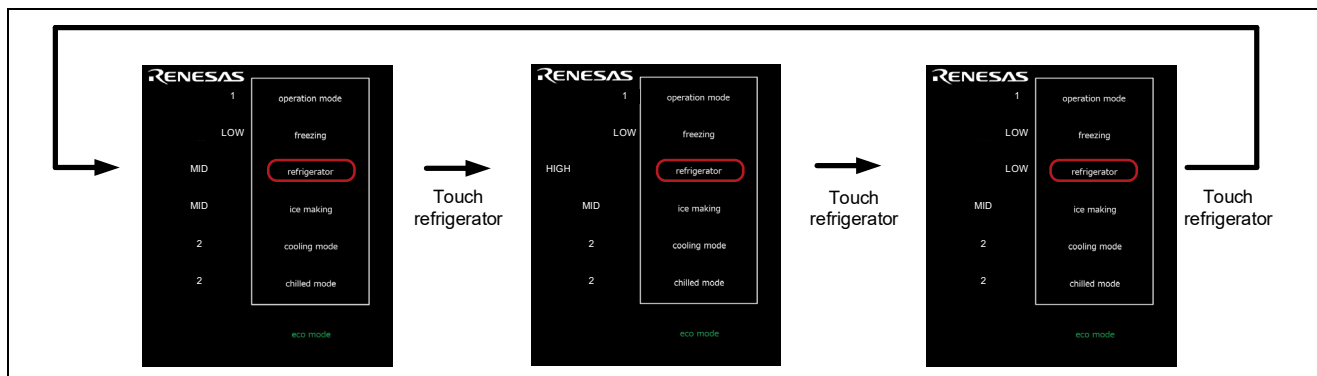


Figure 6-6 Set refrigerator

6.3.4 Set ice making

By touching the ice making button, the setting values can be changed in the order shown in Figure 6-7.

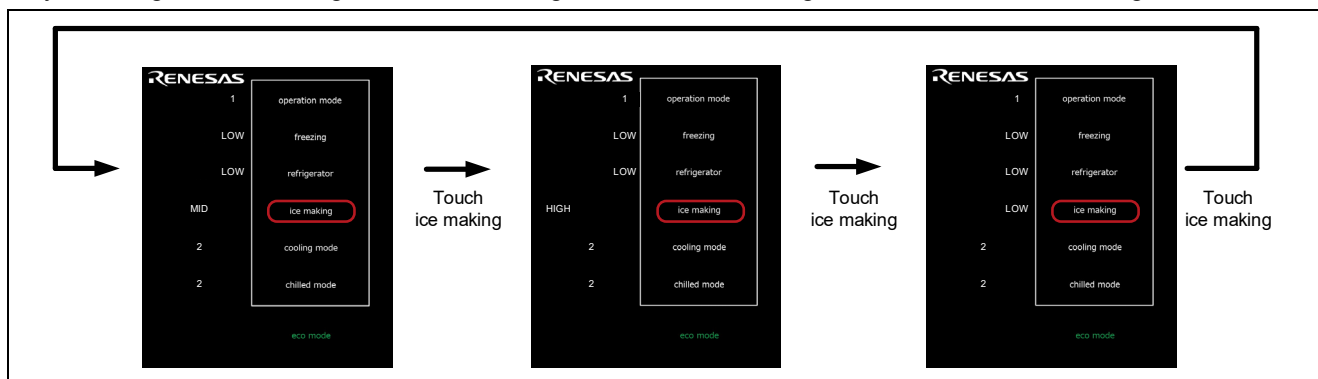


Figure 6-7 Set ice making

6.3.5 Set cooling mode

By touching the cooling mode button, the setting values can be changed in the order shown in Figure 6-8.

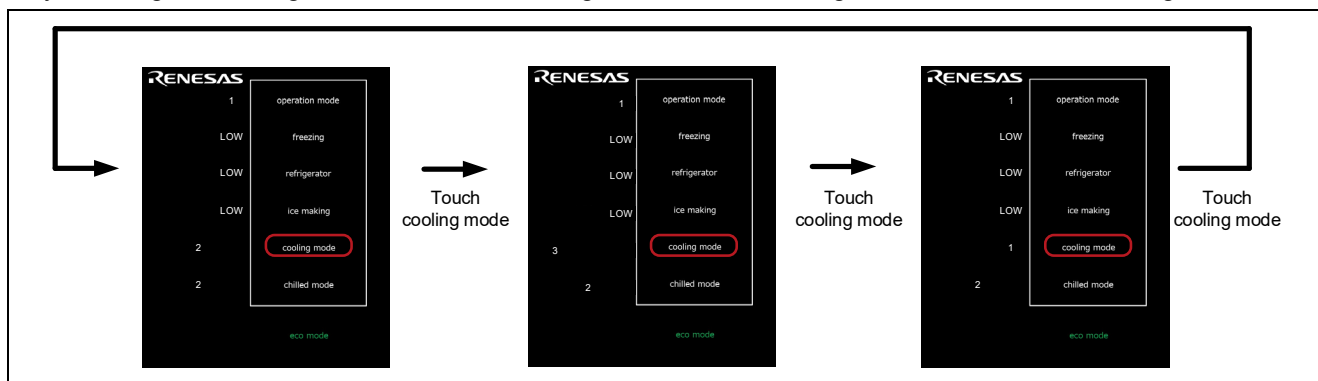


Figure 6-8 Set cooling mode

6.3.6 Set chilled mode

By touching the chilled mode button, the setting values can be changed in the order shown in Figure 6-9.

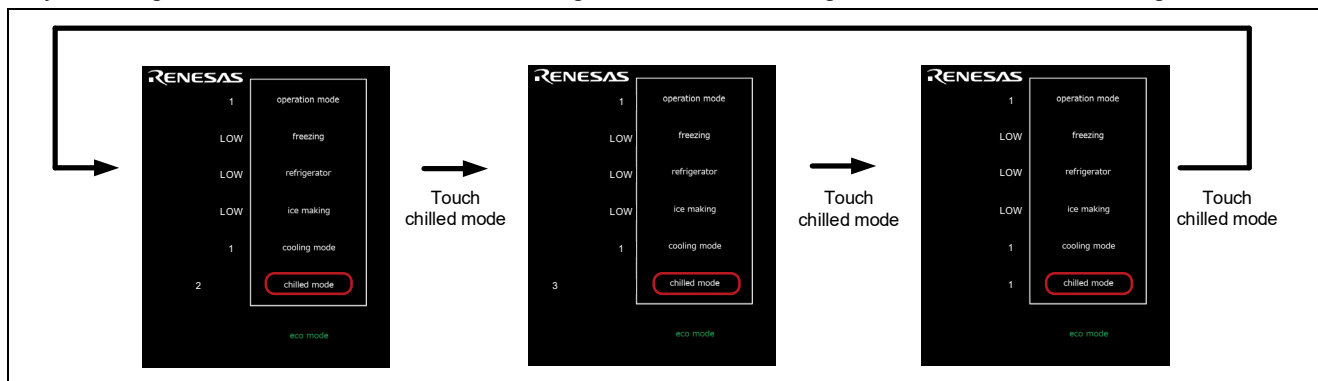


Figure 6-9 Set chilled mode

6.3.7 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. CPU will make the decision to return from standby mode.

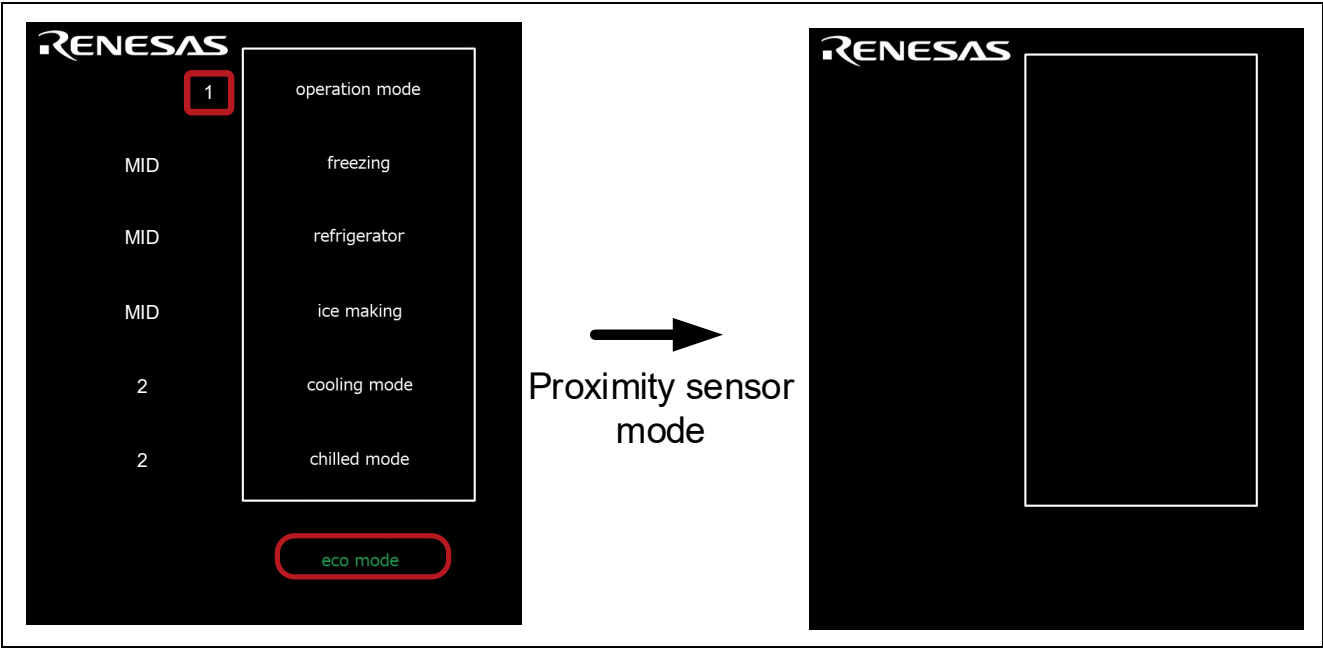


Figure 6-10 When operating mode 1 is Set, Touch eco mode

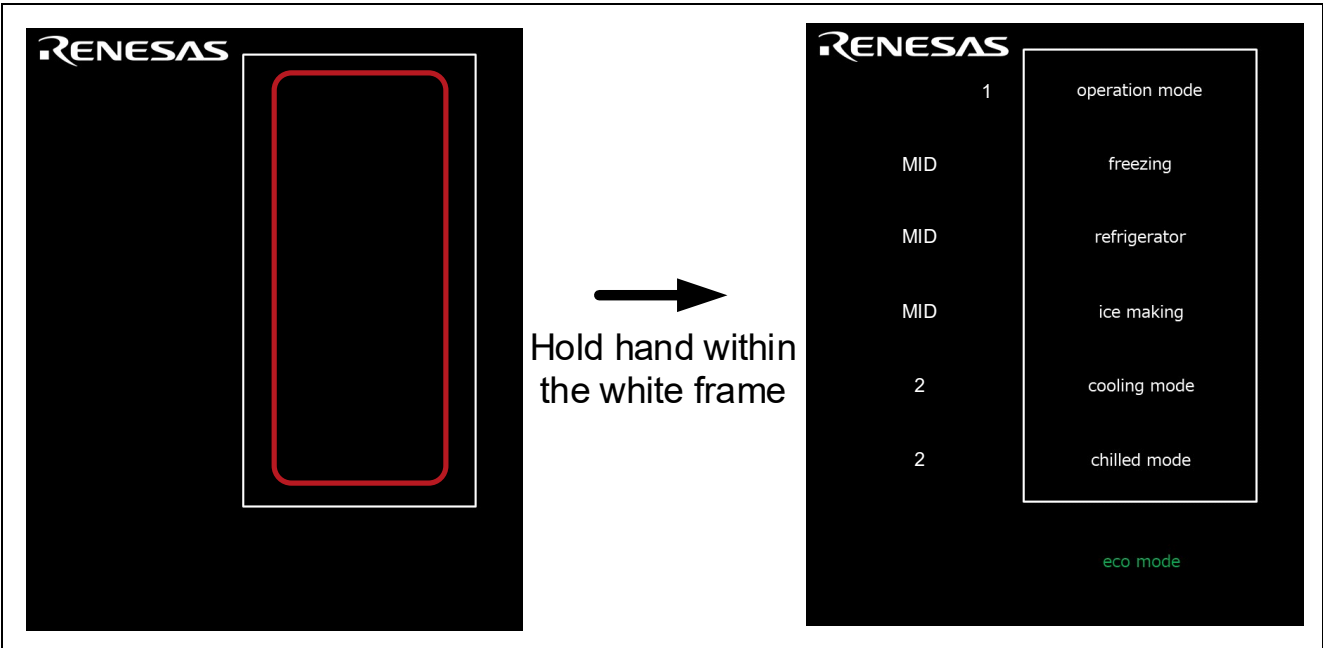


Figure 6-11 Return from Standby Mode in Proximity Sensor Mode

6.3.8 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. CPU will make the decision to return from standby mode.

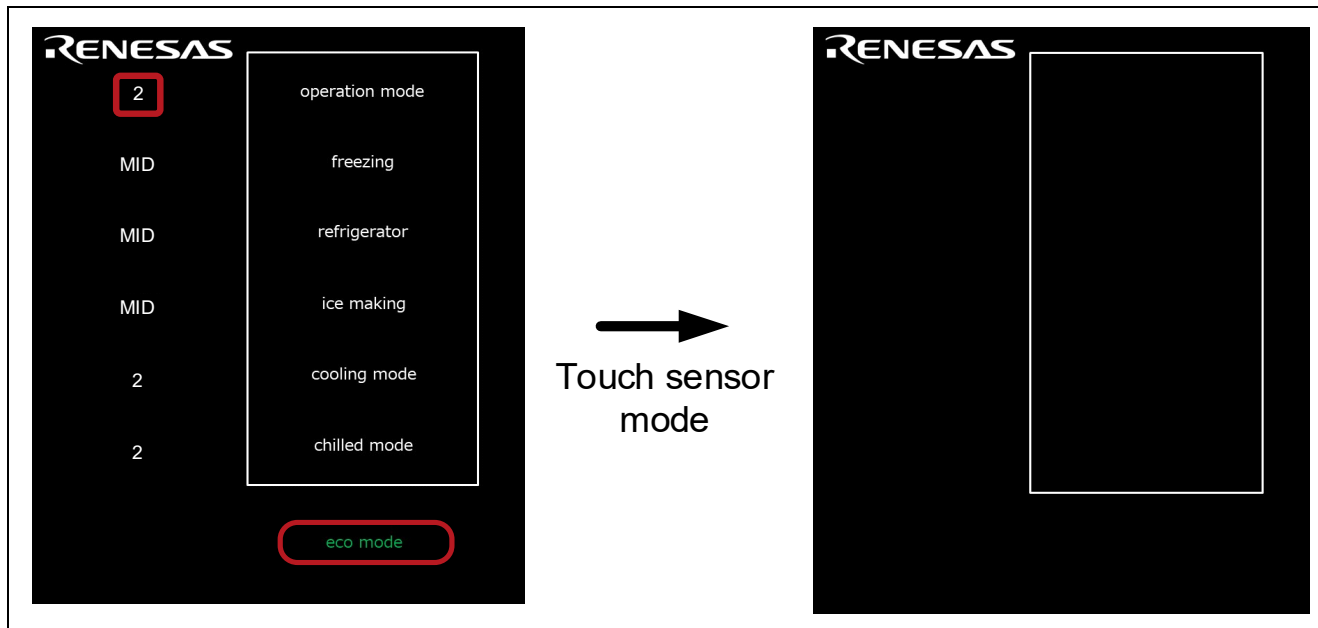


Figure 6-12 When operating mode 2 is Set, Touch eco mode

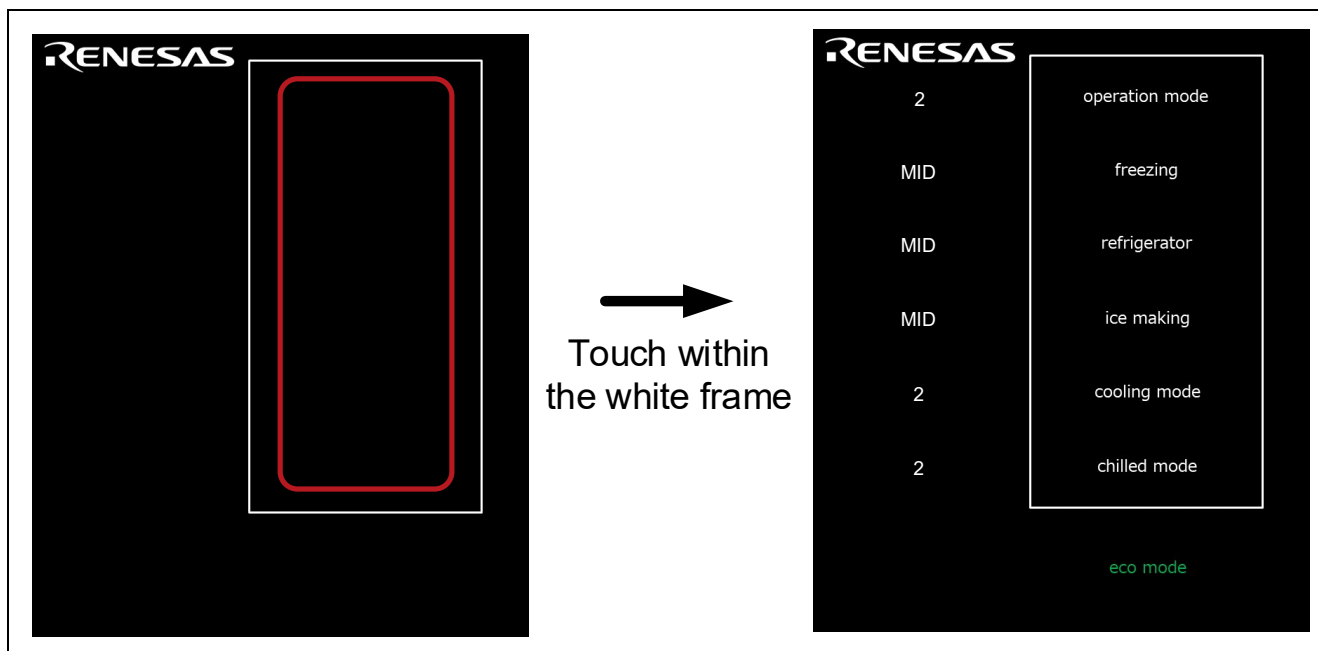


Figure 6-13 Return from Standby Mode in Touch Sensor Mode

6.3.9 eco mode (Auto Judgment (using SMS) Mode)

Touching the eco mode button when the operating mode is set to 3, the device transits the standby mode in the Auto judgment (using SMS) mode. In Auto judgment (using SMS) mode, touching the button in the white frame returns to normal mode.

SMS will make the decision to return from standby mode.

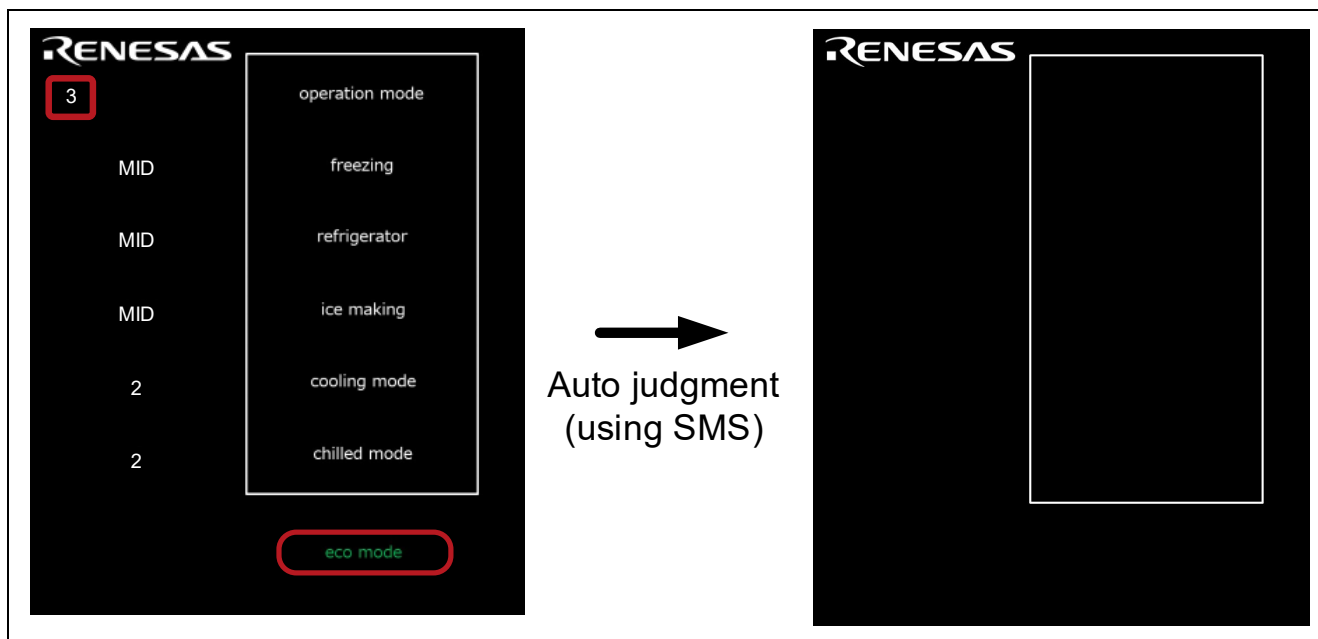


Figure 6-14 When operating mode 3 is Set, Touch eco mode

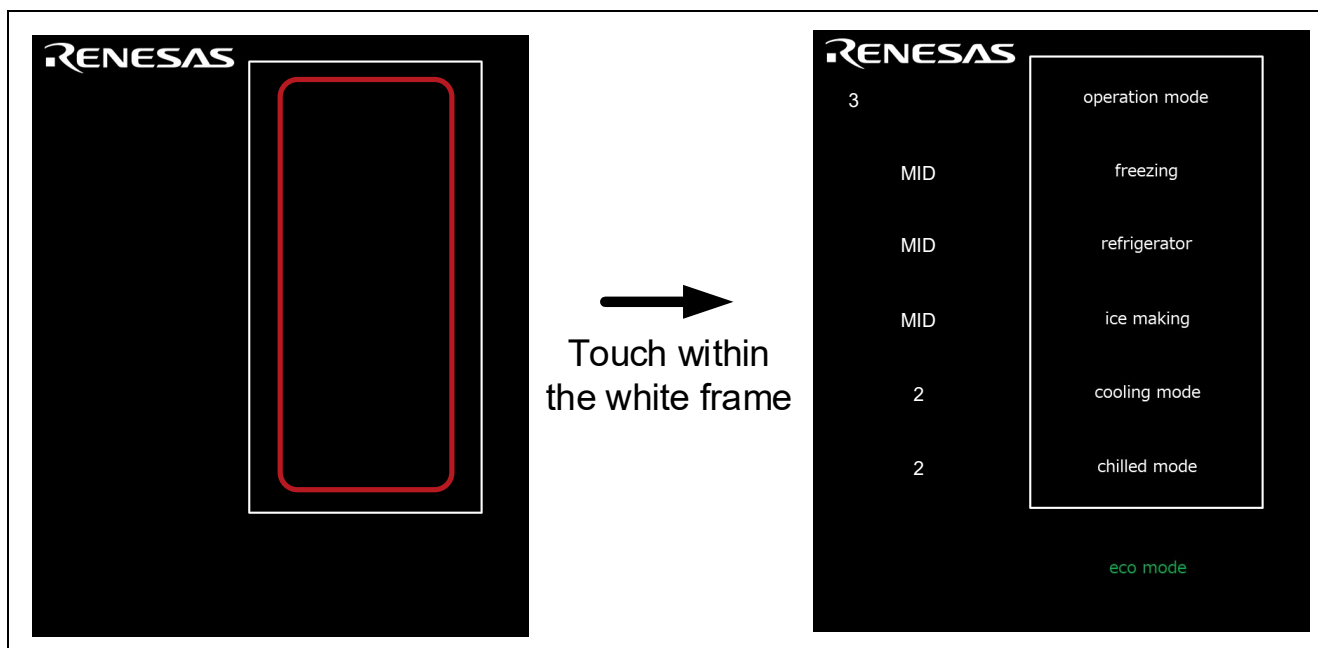


Figure 6-15 Return from Standby Mode in Auto Judgment (using SMS) Mode

7. How to Measure Current Consumption

7.1 Environment to Measure Current Consumption

Figure 7-1 shows the environment to measure current consumption.

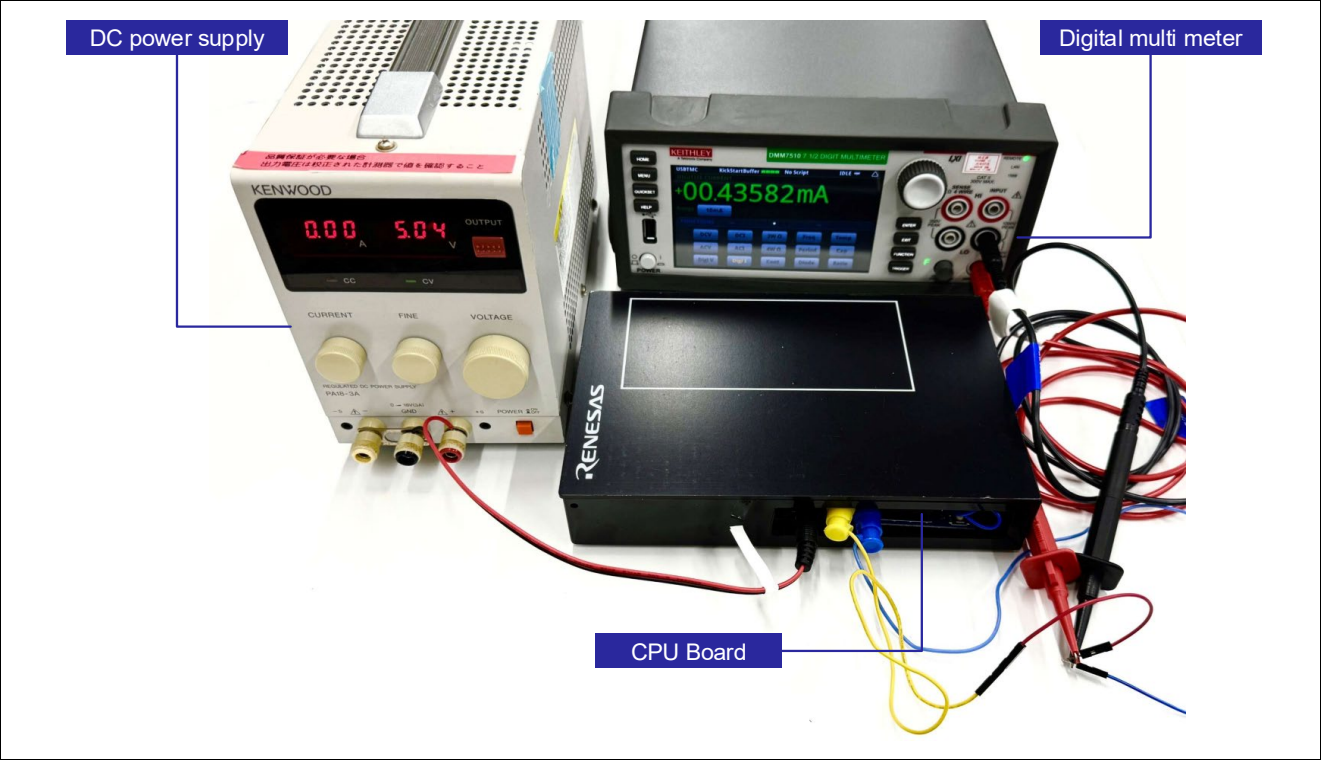


Figure 7-1 Environment to Measure Current Consumption

7.2 Equipment and Software

Table 7-1 shows equipment and software used in current consumption measurement.

Table 7-1 Current Measuring Equipment and Software

Type	Name	Use
Digital multi meter	KEITHLEY/DMM7510	Measure current consumption.
DC power supply	KENWOOD/PA18-3A	Supply power to RL78/G22 Capacitive Touch Evaluation System CPU board (RTK0EG0042S01001BJ).
Software	KEITHLEY/KickStart Software	Get result of current consumption measurement from Keithley DMM7510 and output the result to log-file.

7.3 How to Connect the Target Board and Each Equipment

Figure 7-2 shows how to connect the RL78/G22 CPU board and each equipment, and Figure 7-3 shows the power supply system diagram for the RL78/G22 CPU board.

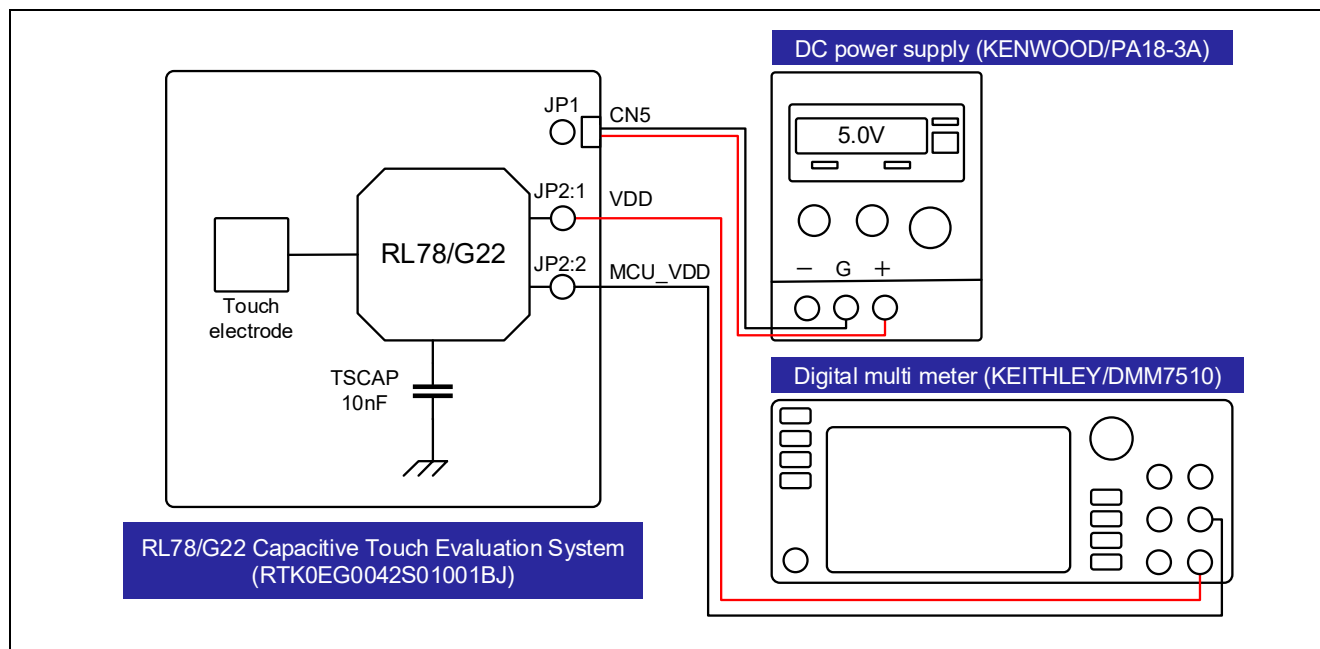


Figure 7-2 Connect the RL78/G22 CPU Board and Each Equipment

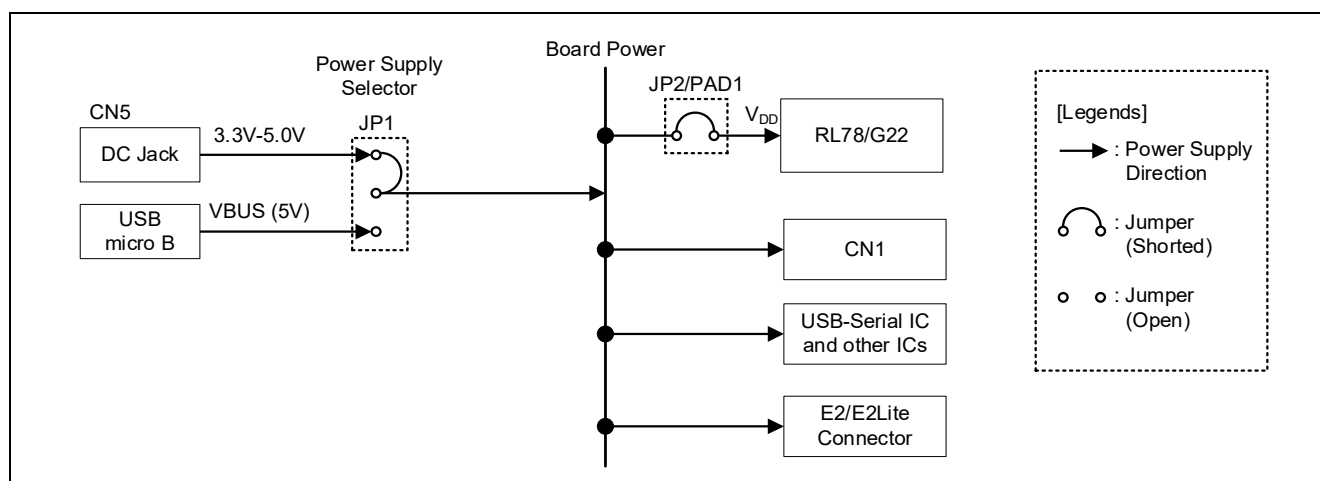


Figure 7-3 Power Supply System Diagram for the RL78/G22 CPU Board

7.4 RL78/G22 CPU Board Settings

Figure 7-4 shows the settings of wiring the RL78/G22 CPU board for current consumption measurement. The 16-pin and 34-pin wires of the Application Header (CN2) for auto judgment measurement using SMS. Table 7-2 CPU Board Jumper and SW Settings shows the jumper settings on RL78/G22 CPU Board.

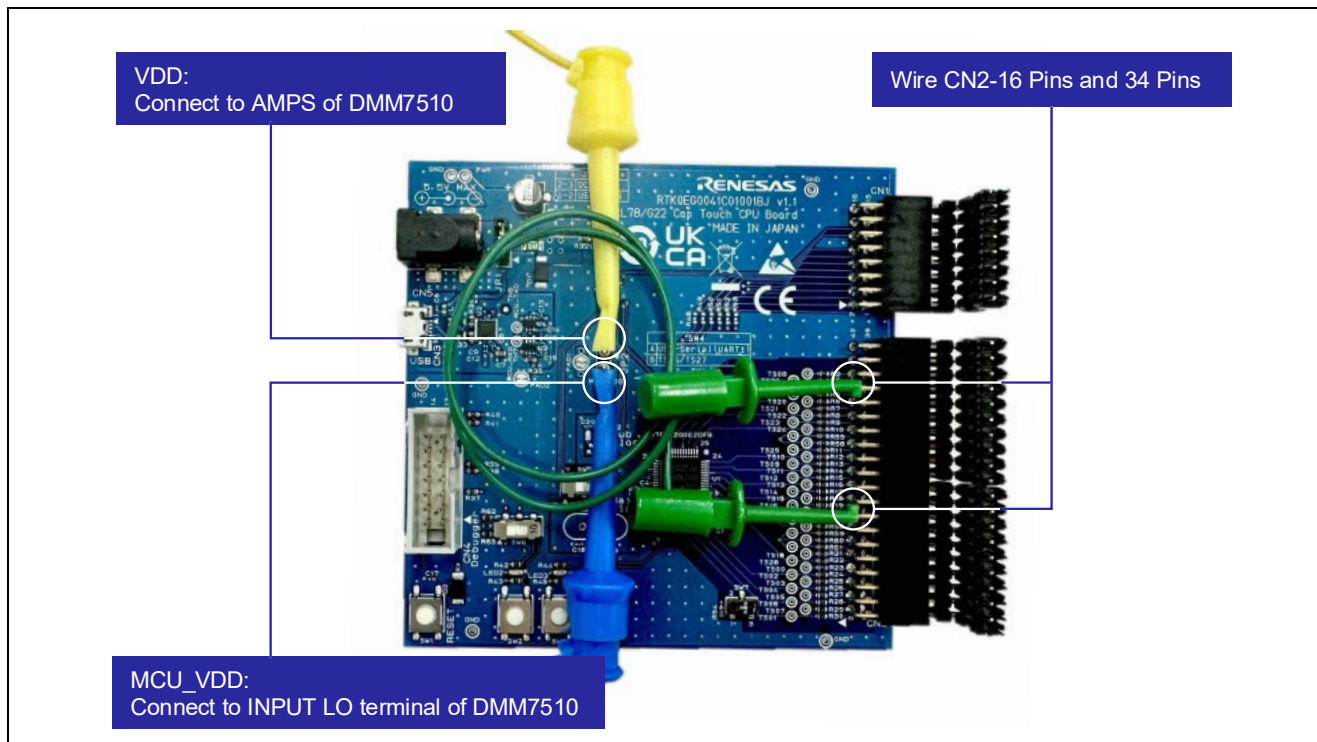


Figure 7-4 Settings of Wiring the RL78/G22 CPU Board for Current Consumption Measurement.

Table 7-2 CPU Board Jumper and SW Settings

Position	Circuit Group	Setting	Use
JP1	VDD power	Shorted pin 2-3	Power supply from DC jack (CN5)
JP2	MCU_VDD_power	Open	Measure current consumption
SW4	USB-Serial Conversion / Application Header (CN2)	OFF (Shorted pin 2-3, 4-5)	Using P00/TS26/TxD1, P01/TS27/RxD1 as a serial communication terminal
SW5	Clock Circuit / Application Header (CN1)	OFF (Shorted pin 2-3, 4-5)	Use P121/X1, P122/X2 as GPIO (CN1)
SW6	Push-Switch & LED/ Application Headers (CN1)	OFF (Shorted pin 2-3, 4-5)	Use P61, P62 as GPIO (CN1)
SW7	Capacitive touch	OFF (Shorted pin 2-3)	Using TS01 as Normal CTSU Pin

7.5 Settings of Current Measuring Software

Figure 7-5 shows settings of KEITHLEY/KickStart software to measure current consumption.

Measurement Settings

Function

Digitize Current

Range

10mA

Aperture (s)

0.000001

Auto Aperture

☒

Display Digits

6.5

☐ Rel

Trigger

Trigger Mode

Immediate

Acquisition

Sample Rate

100000

Sample Count

100000

Start at HH:MM

2024/08/27 13:11:54

☐

Timestamp Format

Relative

Figure 7-5 Settings of KEITHLEY/KickStart Software

8. Current Consumption Measurement Result

The sample code operates the main system clock (this means the clock source for the CPU clock) at 32 MHz for RL78/G22. As reference data, the current consumption when the main system clock runs at 32MHz, 12MHz and 6MHz is shown below. This chapter shows the current consumption for Appliance UI Demo. Therefore, it means the current consumption including the RL78/G22 RSSK CPU board and the electrode board of operation panel.

8.1 Current Consumption

Table 8-1 to Table 8-3 show the average current consumption under each condition.

Table 8-1 Current Consumption (main system clock: 32 MHz)

Mode	Main System Clock [MHz]	Touch Measurement Cycle [ms]	Average Current Consumption per 100ms [μ A]	Remarks
operation mode 1	32	20	71.6	Assuming that one touch measurement is performed during 100ms period, the average current consumption is 14.3 μ A (this is a calculated value).
operation mode 2		20	71.8	Assuming that one touch measurement is performed during 100ms period, the average current consumption is 14.4 μ A (this is a calculated value).
operation mode 3 ^{note}		100	13.3	—
Normal		20	3100.0	Assuming that one touch measurement is performed during 100ms period, the average current consumption is 620.0 μ A (this is a calculated value).

Table 8-2 Current Consumption (main system clock: 12 MHz)

Mode	Main System Clock [MHz]	Touch Measurement Cycle [ms]	Average Current Consumption per 100ms [μ A]	Remarks
operation mode 1	12	20	65.0	Assuming that one touch measurement is performed during 100ms period, the average current consumption is 13.0 μ A (this is a calculated value).
operation mode 2		20	65.2	Assuming that one touch measurement is performed during 100ms period, the average current consumption is 13.0 μ A (this is a calculated value).
operation mode 3 ^{note}		100	10.8	—
Normal		20	1600.0	Assuming that one touch measurement is performed during 100ms period, the average current consumption is 320.0 μ A (this is a calculated value).

Table 8-3 Current Consumption (main system clock: 6 MHz)

Mode	Main System Clock [MHz]	Touch Measurement Cycle [ms]	Average Current Consumption per 100ms [μ A]	Remarks
operation mode 1	6	20	74.1	Assuming that one touch measurement is performed during 100ms period, the average current consumption is 14.8 μ A (this is a calculated value).
operation mode 2		20	73.5	Assuming that one touch measurement is performed during 100ms period, the average current consumption is 14.7 μ A (this is a calculated value).
operation mode 3 note		100	10.4	—
Normal		20	1200.0	Assuming that one touch measurement is performed during 100ms period, the average current consumption is 240.0 μ A (this is a calculated value).

Note. Operation mode 3 (touch sensor) achieves low power operation with auto judgment function using SMS. The auto judgment function using SMS cannot be performed when f_{CLK} is set to 4 MHz or lower.

The current consumption for Appliance UI Demo is described below.

In operation mode 1 (proximity sensor) and operation mode 2 (touch sensor), low power operation is realized using the SNOOZE mode function by CTSU2La. The yellow highlighted indications in Table 8-1 to Table 8-3 show the results of the lowest current consumption when the main system clock is running at 12 MHz for low power operation using the SNOOZE mode function.

In operation mode 3 (touch sensor), low power operation is realized using the auto judgment function using SMS. The blue highlighted indication in Table 8-1 to Table 8-3 shows the results of the lowest current consumption when the main system clock is running at 6 MHz for low power operation using the auto judgment function using SMS.

In normal mode, current consumption varies depending on the user program processing.

In the case of the Appliance UI Demo, the current consumption in normal mode is due to the process of changing the LED lighting pattern for each touch button detection.

8.2 Current Consumption Waveform

The current consumption waveforms in each mode of low power operation are shown below.

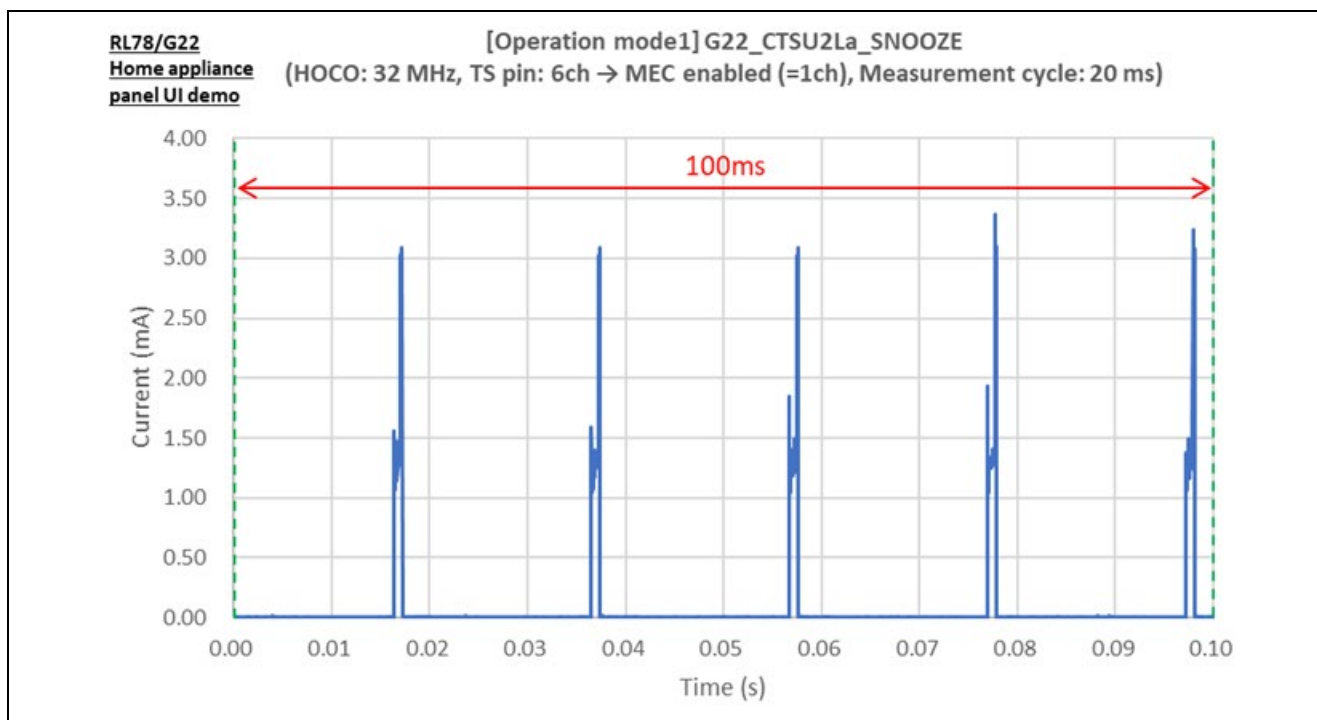


Figure 8-1 Current Consumption Waveforms in 100ms period (operation mode1)

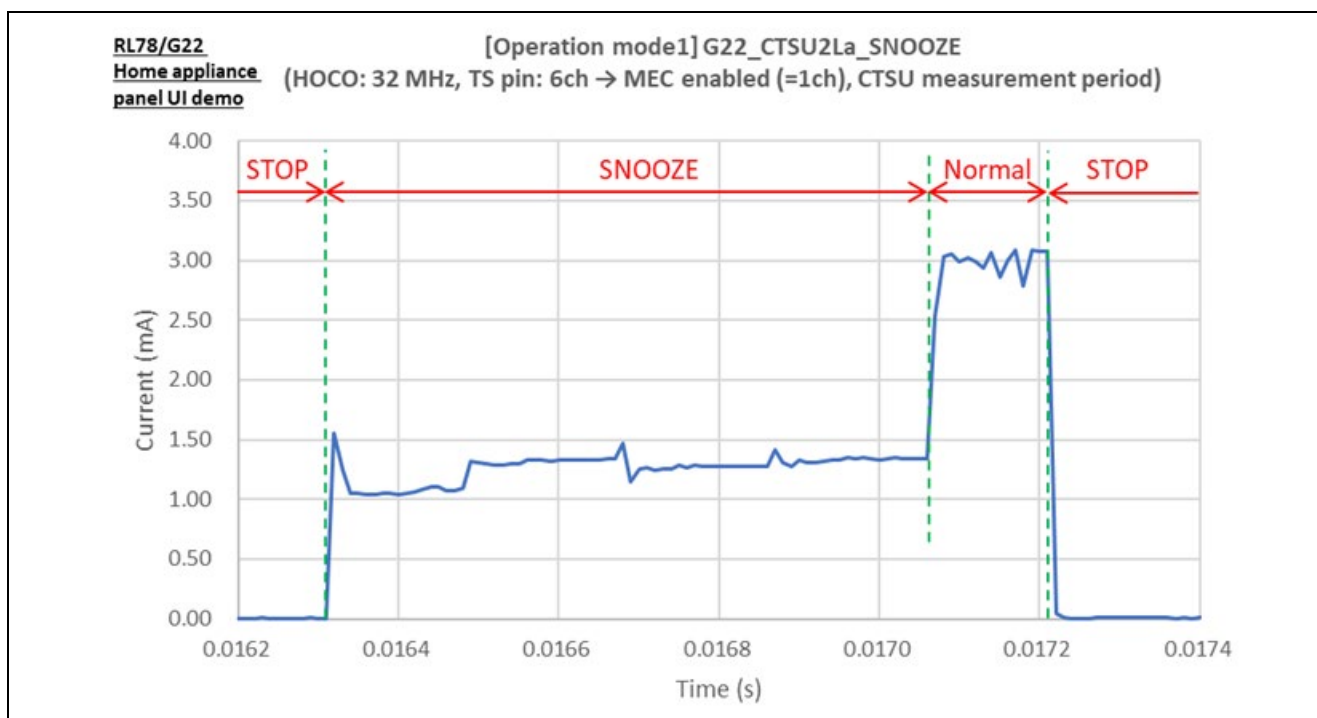


Figure 8-2 Current Consumption Waveforms during touch measurement processing (operation mode1)

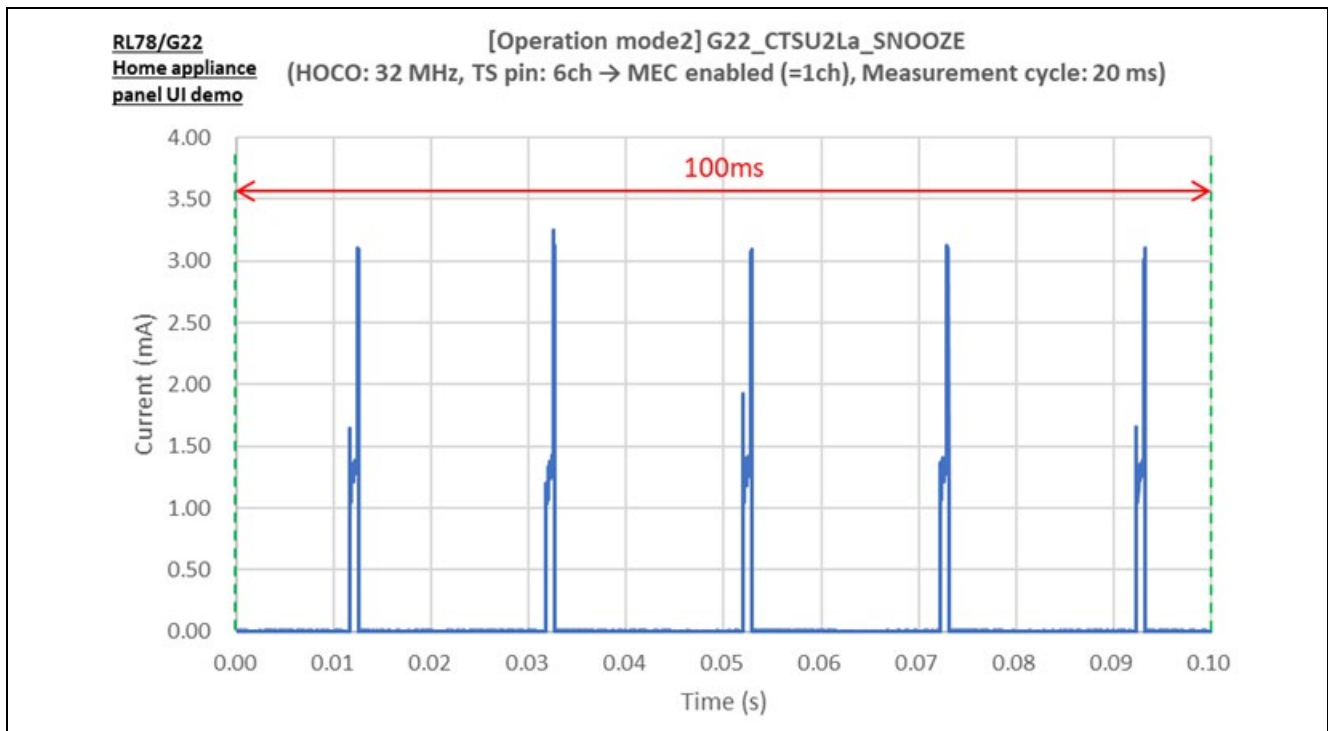


Figure 8-3 Current Consumption Waveforms in 100ms period (operation mode2)

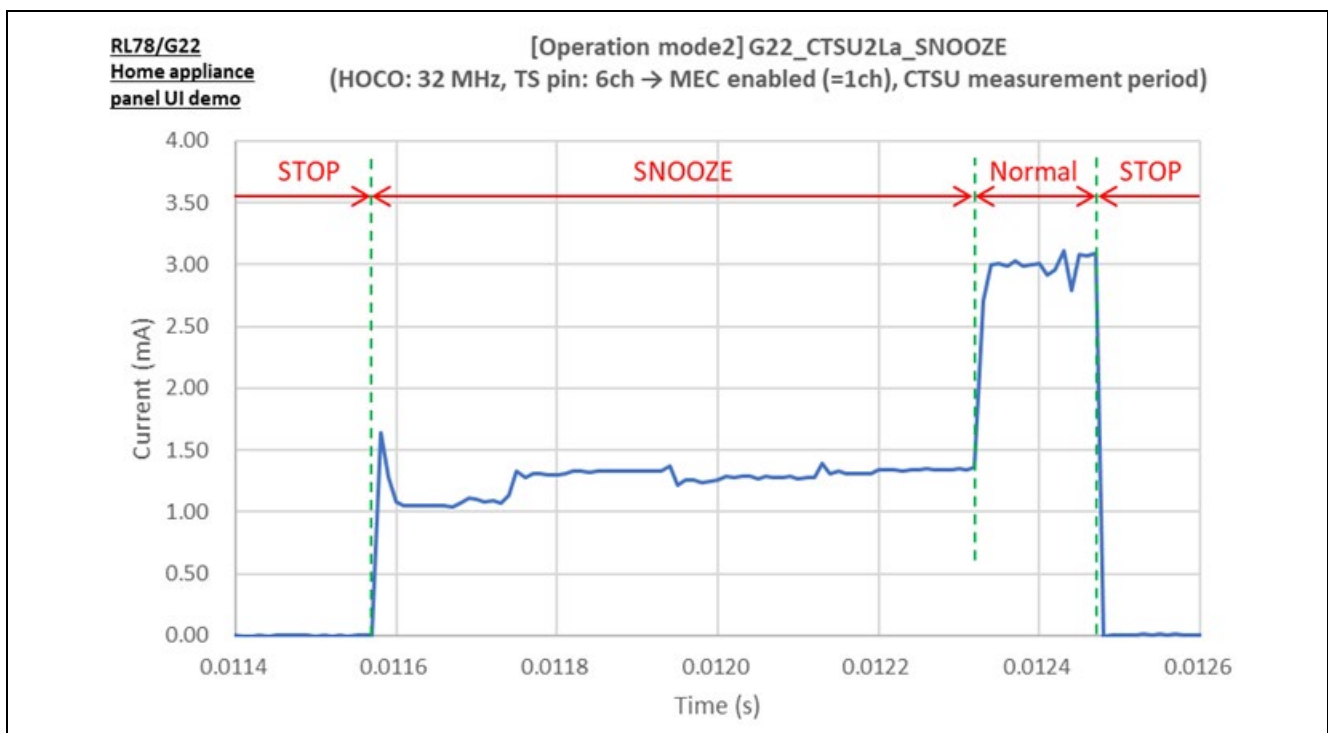


Figure 8-4 Current Consumption Waveforms during touch measurement processing (operation mode2)

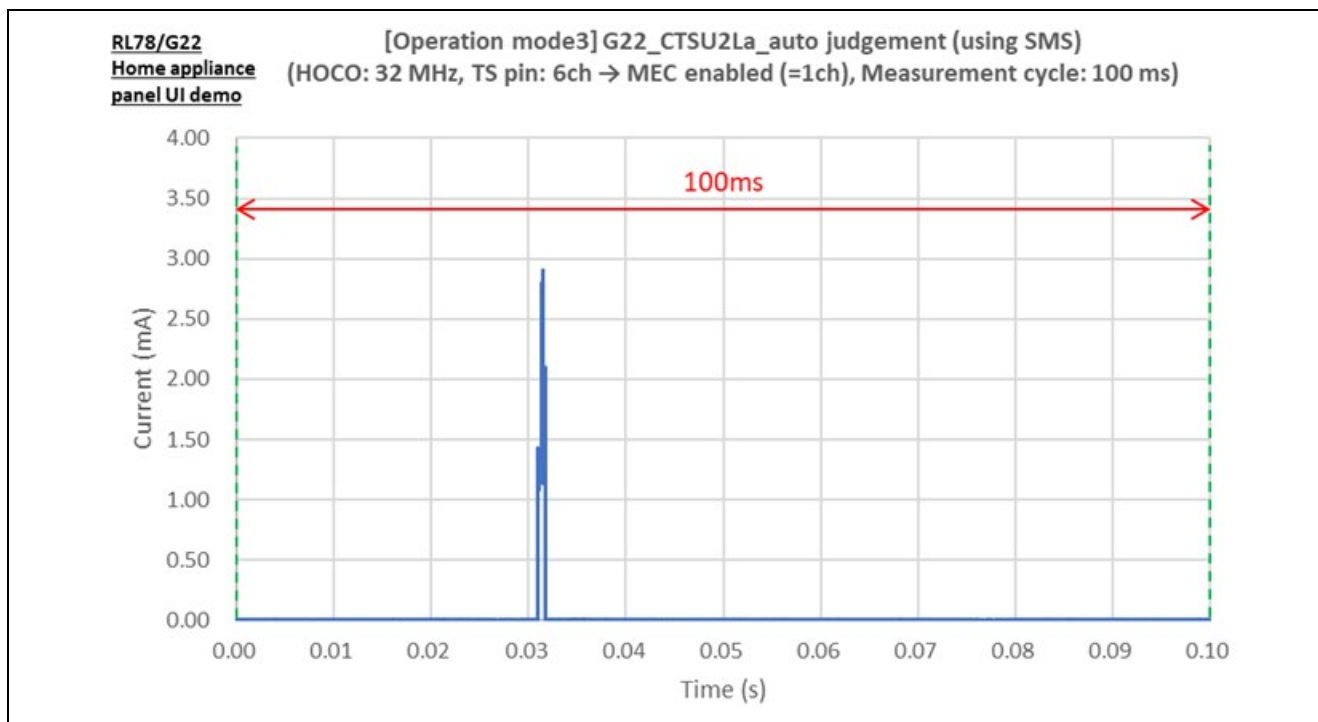


Figure 8-5 Current Consumption Waveforms in 100ms period (operation mode3)

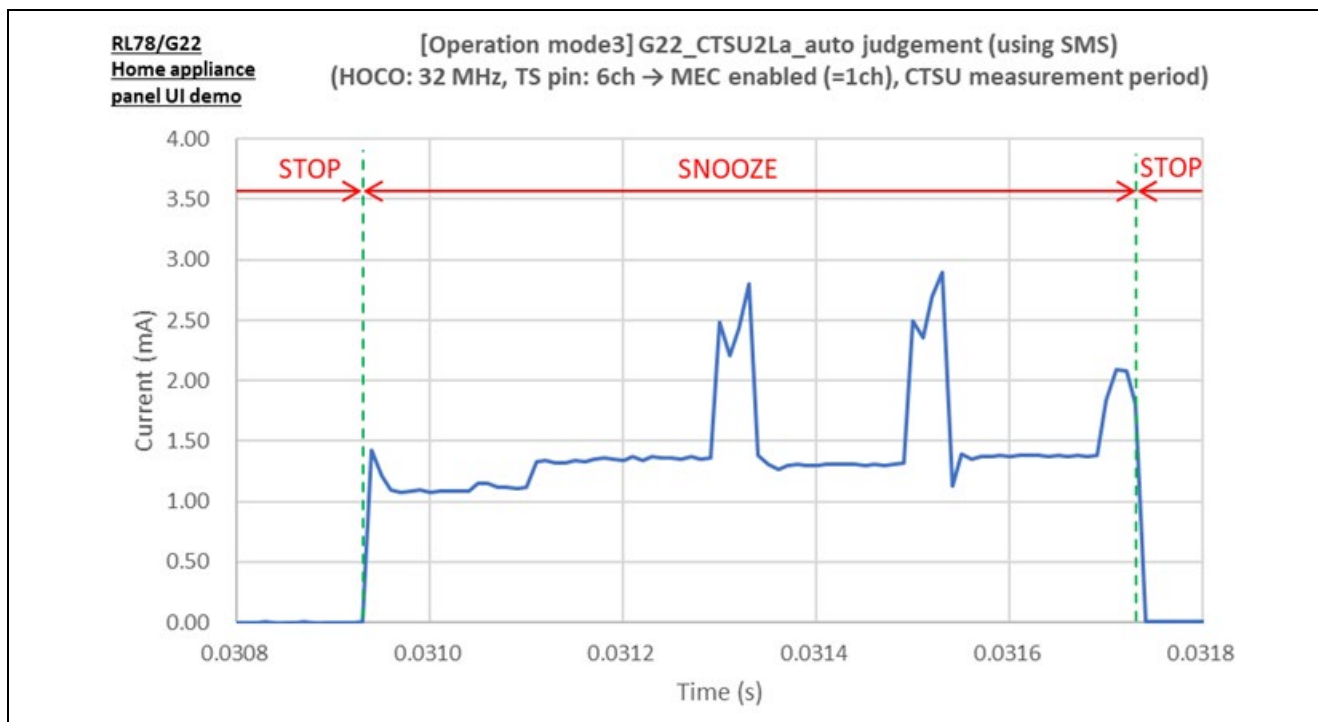


Figure 8-6 Current Consumption Waveforms during touch measurement processing (operation mode3)

The following shows a schematic of Electrode Board.

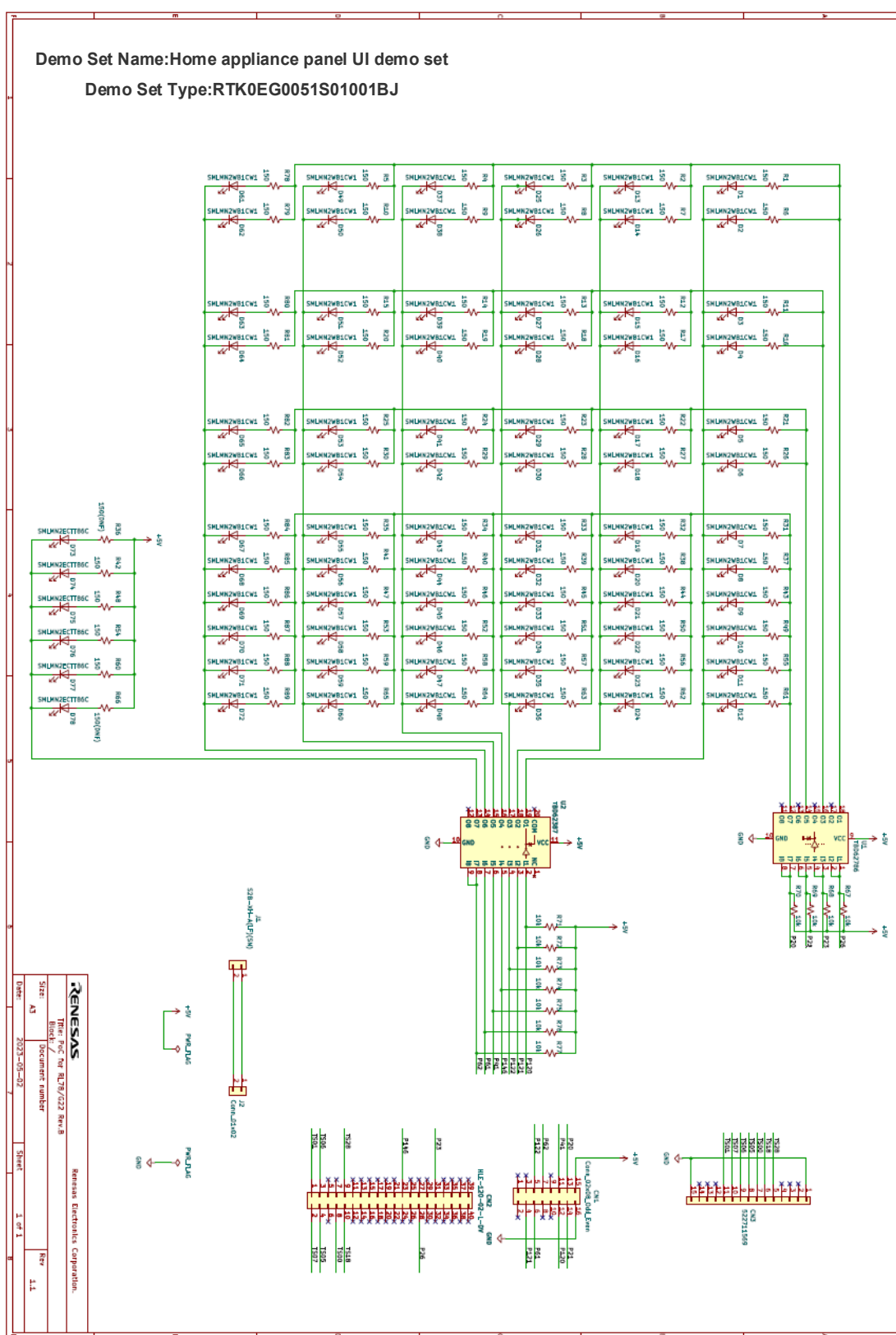


Figure 9-1 Electrode Board Schematic

9.2 Components Placement

The following shows a components placement of the Electrode Board.

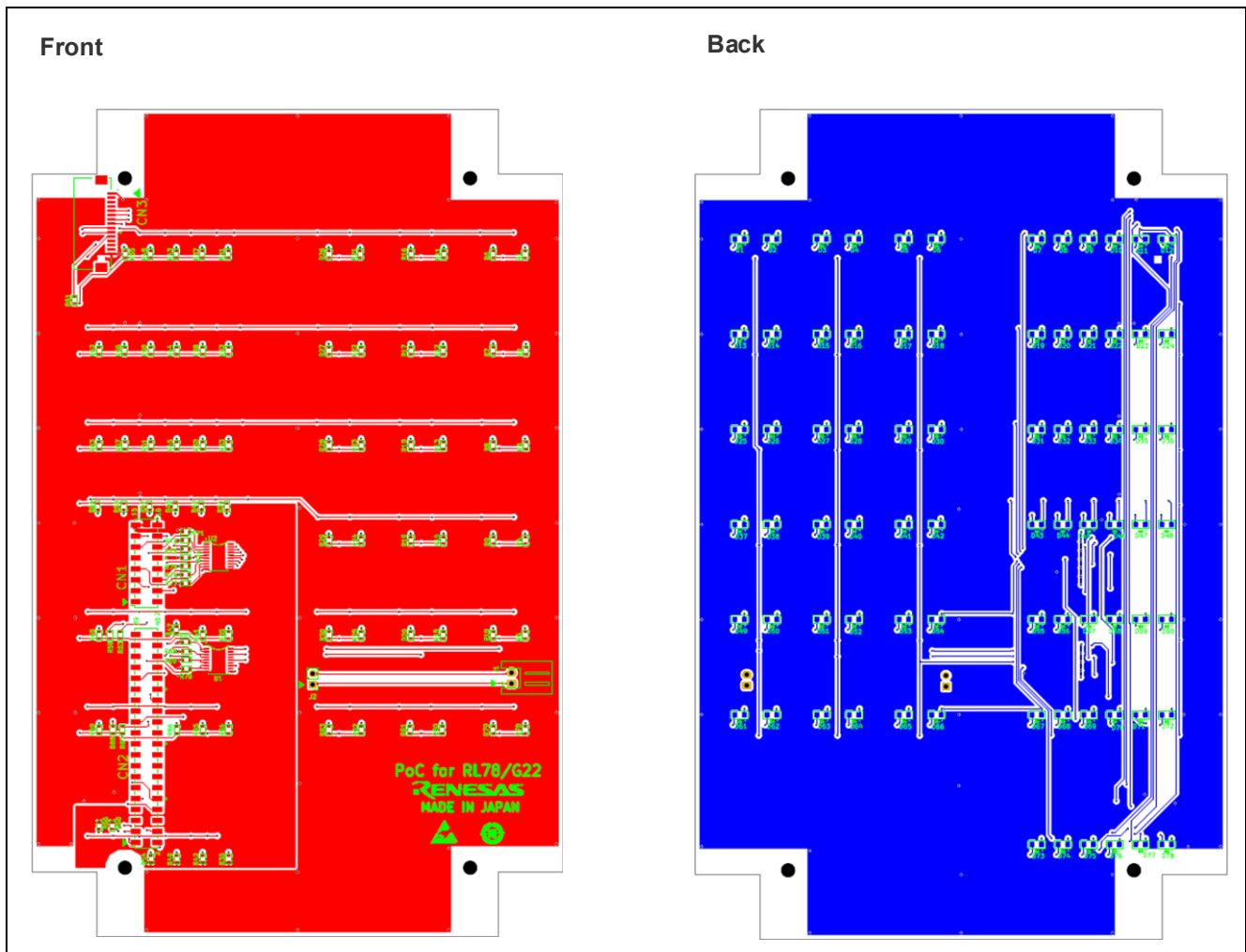


Figure 9-2 Electrode Board Components Placement

9.3 Components List

The following shows a components list of the Electrode Board.

Table 9-1 Electrode Board Components List

Comment	Description	Designator	Manufacturer	Quantity	notes
TBD62786	Transistor Array	U1	TOSHIBA	1	
TBD62387	Transistor Array	U1	TOSHIBA	1	
HLE-108-02-L-DV	Connector	CN1	Samtec	1	
HLE-120-02-L-DV	Connector	CN2	Samtec	1	
522711569	Connector	CN3	Molex	1	
SMLMN2WB1CW1	LED	D1,D2,D3,D4,D5,D6,D7, D8,D9,D10,D11,D12, D13,D14,D15,D16,D17, D18,D19,D20,D21,D22, D23,D24,D25,D26,D27, D28,D29,D30,D31,D32, D33,D34,D35,D36,D37, D38,D39,D40,D41,D42, D43,D44,D45,D46,D47, D48,D49,D50,D51,D52, D53,D54,D55,D56,D57, D58,D59,D60,D61,D62, D63,D64,D65,D66,D67, D68,D69,D70,D71,D72	ROHM	72	White
SMLMN2ECTT86C	LED	D73,D74,D75,D76,D77, D78	ROHM	6	Green
S2B-XH-A(LF)(SN)	Connector	J1	JST	1	
FFC-2AEMP	Connector	J2	HTK	1	
MCR03EZPJ151	Resistor	R1,R2,R3,R4,R5,R6,R7, R8,R9,R10,R11,R12, R13,R14,R15,R16,R17, R18	ROHM	76	1608m 150
RMC1/16K151FTP			KAMAYA		
MCR03EZPJ151	Resistor	R36,R66	ROHM	0	150 1608m
MCR03EZPJ103	Resistor	R67,R68,R69,R70,R71, R72,R73,R74,R75,R76, R77	ROHM	11	10k 1608m
RMC1/16K103FTP			KAMAYA		10k 1608m
PSR-420257-8	Connector	Mounted on CN1	Hirosugi-Keiki	1	
PSR-420257-10	Connector	Two mounted on CN2	Hirosugi-Keiki	2	

10. Reference Documents

○User's manual

- 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

RL78/G22 Capacitive Touch Evaluation System (RTK0EG0042S01001BJ) User's manual (R12UZ0110)

(The latest version can be downloaded from the Renesas Electronics website.)

○Application Note

- Capacitive Sensor Microcontrollers CTSU Capacitive Touch Introduction Guide (R30AN0424)
- Capacitive Sensor Microcontrollers CTSU Capacitive Touch Electrode Design Guide (R30AN0389)

- RL78 Family Capacitive Touch Low Power Application Development using SMS (R01AN7261)
- RL78 Family CTSU Module Software Integration System (R11AN0484)
- RL78 Family TOUCH Module Software Integration System (R11AN0485)
-

(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

Rev.	Date	Description	
		Page	Summary
1.00	Jun.23.23	—	First edition
2.00	Apr.7.25	—	For operation mode 3 (auto judgment using SMS)
		5, 6	Corrected text in 1.1. Corrected Figure 1-1, Figure 1-3
		7	Added the contents of operation mode 3 (auto judgment using SMS) to 1.2
		8 - 10	Added the chapter 2 Appliance UI Demo Hardware Overview.
		11	Renumbered Section 2 to 3. Version update of the development environment in Table 3-1 and Figure 3-1.
		12	Renumbered Section 3 to 4. Added Table 4-1 Operation Overview of Each Mode.
		13	Added the operation mode 3 (auto judgment using SMS) to Figure 4-1.
		15 - 22	Reflect the changes in Smart Configurator settings for operation mode 3 (auto judgment using SMS) in Table 4-2, Table 4-3, Table 4-4, Table 4-5, Table 4-6, Table 4-7, and Table 4-9.
		15	Corrected Table 4-2
		16 - 17	Added 4.4 Setting of Unused Pins.
		19	Added Table 4-8 option byte settings.
		22	Corrected Table 4-9
		23	Corrected Table 4-10
		25 - 34	Added and corrected the function for operation mode 3 (auto judgment using SMS) in 4.5.8 Functions, 4.5.9 Function Specifications
		36 - 50	Added and corrected the function for operation mode 3 (auto judgment using SMS) in 4.5.10 Flowchart
		68	Added the operation mode 3 (auto judgment using SMS) to Figure 6-4
		72	Added 6.3.9 eco mode (Auto Judgment (using SMS) Mode) to describe the contents of operation mode 3 (auto judgment using SMS).
		73 - 76	Added the chapter 7 How to Measure Current Consumption.
		77, 78	Added the chapter 8 Current Consumption Measurement Result.
		79 - 81	Added the chapter 9 Design Documents for Electrode Board.
		82	Added the following items in reference documents • RL78 Family Capacitive Touch Low Power Application Development using SMS (R01AN7261) • Capacitive Sensor Microcontrollers CTSU Capacitive Touch Electrode Design Guide (R30AN0389)

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

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

1. Precaution against Electrostatic Discharge (ESD)

A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity.

Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.

2. Processing at power-on

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

3. Input of signal during power-off state

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

4. Handling of unused pins

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

5. Clock signals

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

6. Voltage application waveform at input pin

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

7. Prohibition of access to reserved addresses

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

8. Differences between products

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

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
6. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The intended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.

"Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.

"High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user's manual or other Renesas Electronics document.

7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION ("Vulnerability Issues"). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. When using Renesas Electronics products, refer to the latest product information (data sheets, user's manuals, application notes, "General Notes for Handling and Using Semiconductor Devices" in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1 October 2020)

Corporate Headquarters

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

Trademarks

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

Contact information

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