RL78 ファミリ

RL78/G22 MEC 機能を用いた家電パネル UI デモ サンプルソフトウェア

要旨

本アプリケーションノートでは、RL78/G22 の CTSU2La を使用し、タッチボタンと MEC（Multiple Electrode Connection）機能を用いた家電パネル UI デモセットの PoC（以下、RL78/G22 PoC）を紹介します。

MEC 機能（複数電極接続機能）とは、複数のタッチ電極をまとめて一つの電極とみなす機能です。例えば 6 個のタッチボタンをもつシステムがあり、どのタッチボタンにタッチしてもスタンバイ・モードから復帰するようなシステムを組むのに最適です。MEC 機能がないデバイスでは、6 回のスキャンを行わないと、タッチの有無を判定できませんが、RL78/G22 では 1 回のスキャンでタッチの有無を判定できます。このように、スキャン回数が少なくて済むことから、低消費電力動作が可能です。

また、MEC 機能使用時のタッチ検出を高感度に設定することで、複数のタッチ電極を一つの大きな近接センサ電極としても使用可能です。

動作確認デバイス

RL78/G22

本アプリケーションノートを他のマイコンへ適用する場合、そのマイコンの仕様に合わせて変更し、十分評価してください。

動作確認ツール

RL78/G22 静電容量タッチ評価システム (RSSK) (RTK0EG0042S01001BJ)の
CPU ボード(RTK0EG0041C01001BJ)

目次

1. 概要

本アプリケーションノートでは、冷蔵庫パネルをモチーフとした RL78/G22 PoC の構成と操作方法を説明します。RL78/G22 PoC は、タッチボタンを 7 個搭載しています。これらのタッチボタンは、通常時には独立したタッチボタンとして働きます。一定時間タッチパネルを操作しなかった場合、パネルを非表示にして、スタンバイ・モードに遷移します。スタンバイ・モードでは、MEC 機能により 6 個のタッチボタンが一つのタッチボタンとして機能します。

1.1 MEC 機能

MEC 機能とは、複数チャネルのタッチ電極をまとめて一つの電極として測定する機能です。

1.1.1 MEC 機能のメリット 1（任意の電極へのタッチによるスタンバイ・モードからの復帰）

RL78/G22 PoC では、スタンバイ・モード時に MEC 機能により 6 つのタッチ電極を一つの電極として使用します。これにより、白枠内のどの電極にタッチしても、スタンバイ・モードから CPU の復帰が可能です。

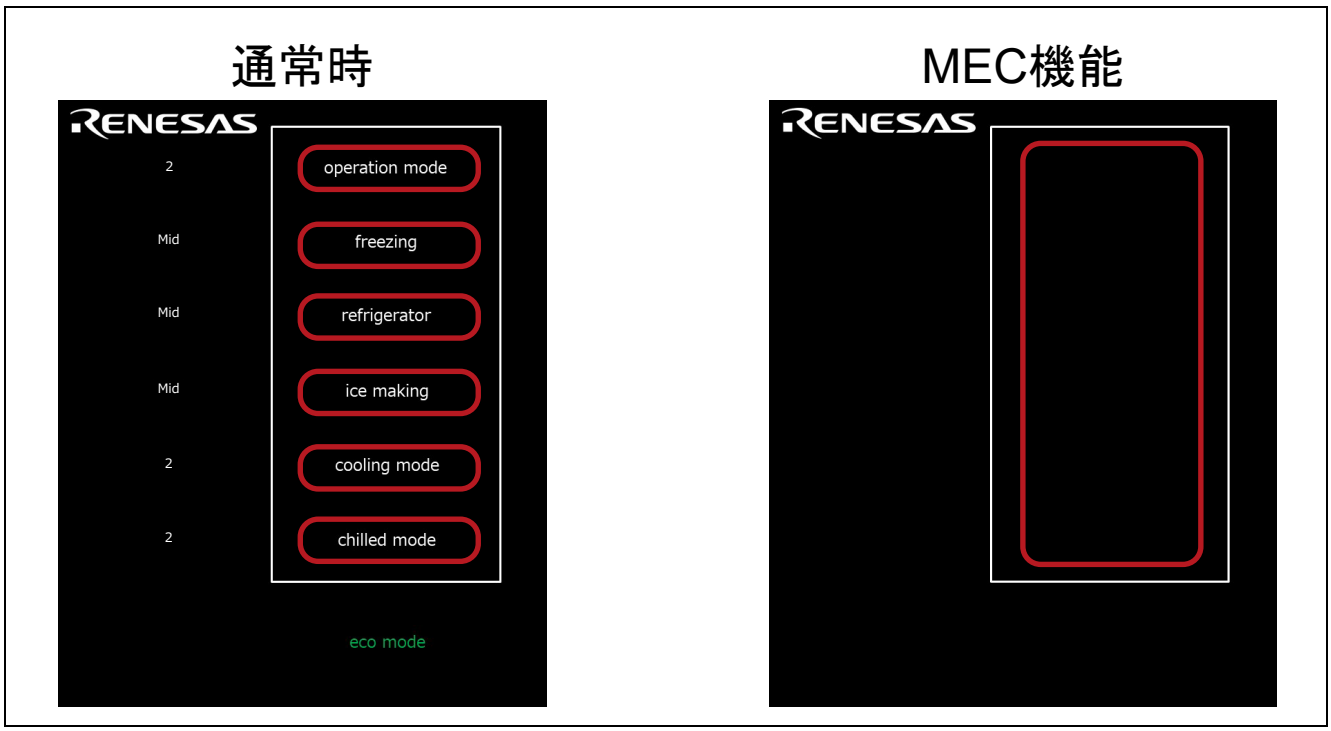

図 1-1　MEC 機能（一つの電極）

1.1.2　MEC 機能のメリット 2（近接センサとして使用可能）

タッチ電極を近接する配置構成にして MEC 機能を使用することで、複数のタッチ電極を一つの大きな電極とみなすことができます。そして、タッチ閾値の設定次第では近接センサとして使用可能です。

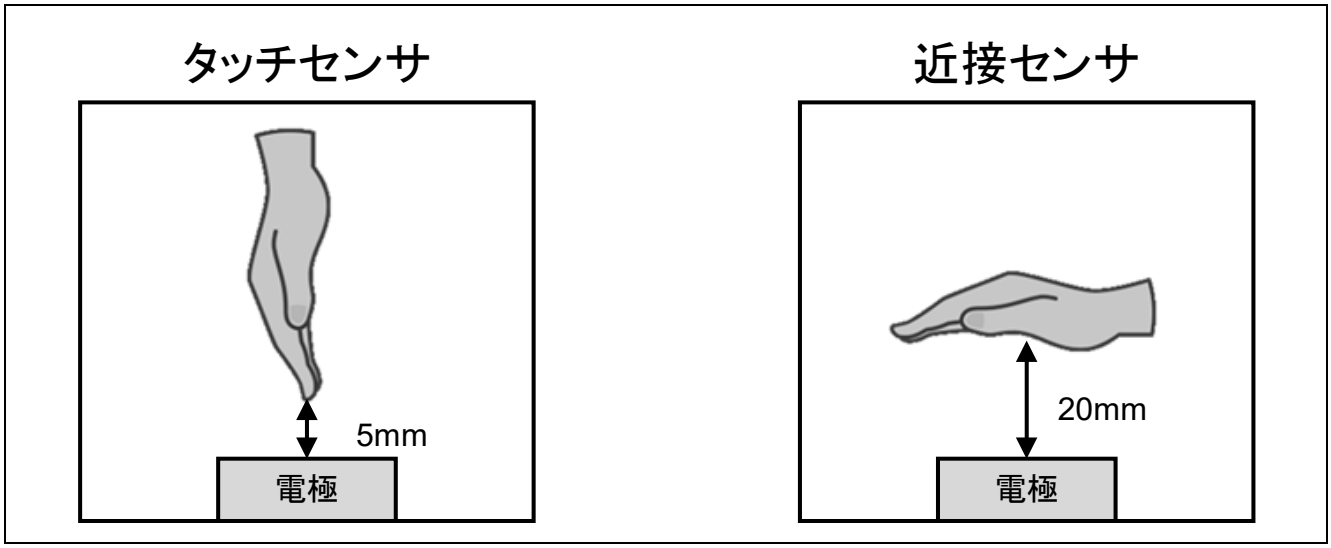

図 1-2　MEC 機能（近接センサ）

1.1.3　MEC 機能のメリット 3（低消費電力）

MEC 機能は、複数のタッチ電極を一つの電極として使用します。このため、電極のスキャンは 1 回で行えます。電極のスキャンが 1 回で済むため、低消費電力動作が可能です。

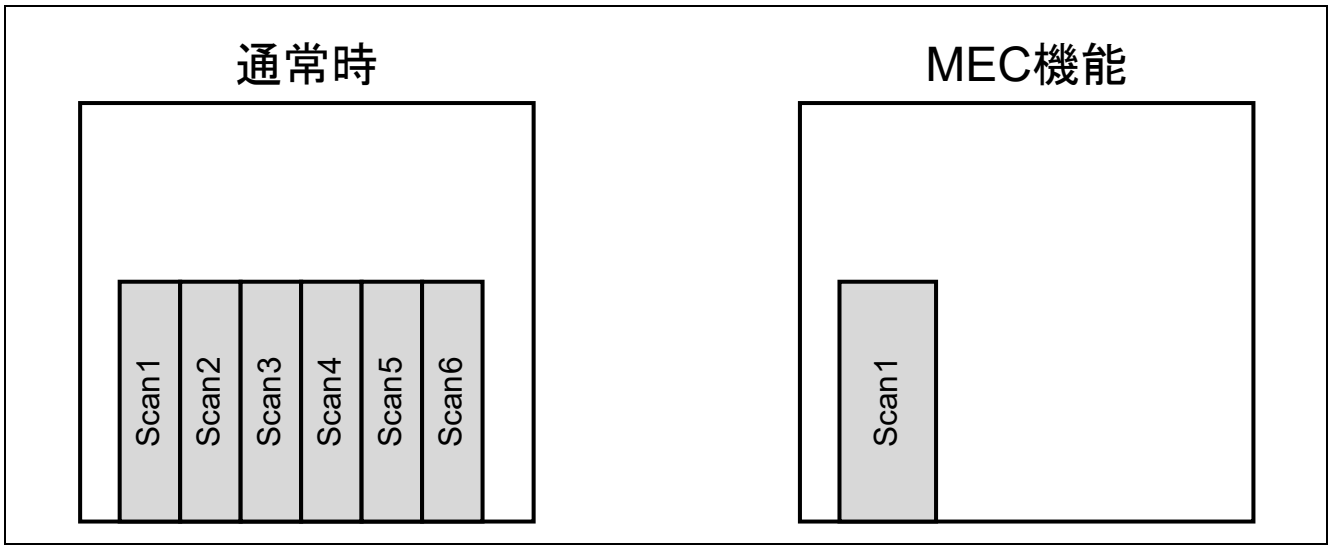

図 1-3　MEC 機能（低消費電力）

1.2　RL78/G22 PoC での MEC 機能活用方法

RL78/G22 PoC では、operation mode として、MEC 機能を使用した動作のバリエーションを実装しています。

operation mode1 では、タッチボタンのタッチ閾値を手の接近で検出できるように低く設定しています。白枠部分に手をかざすことで、スタンバイ・モードから復帰します。

operation mode2 では、タッチボタンのタッチ閾値を直接手で触れるタッチで検出できるように設定しています。いずれかのタッチボタンにタッチすると、スタンバイ・モードから復帰します。

operation mode3 は追加機能実装用のモードです。

※operation mode3 は実装していません。

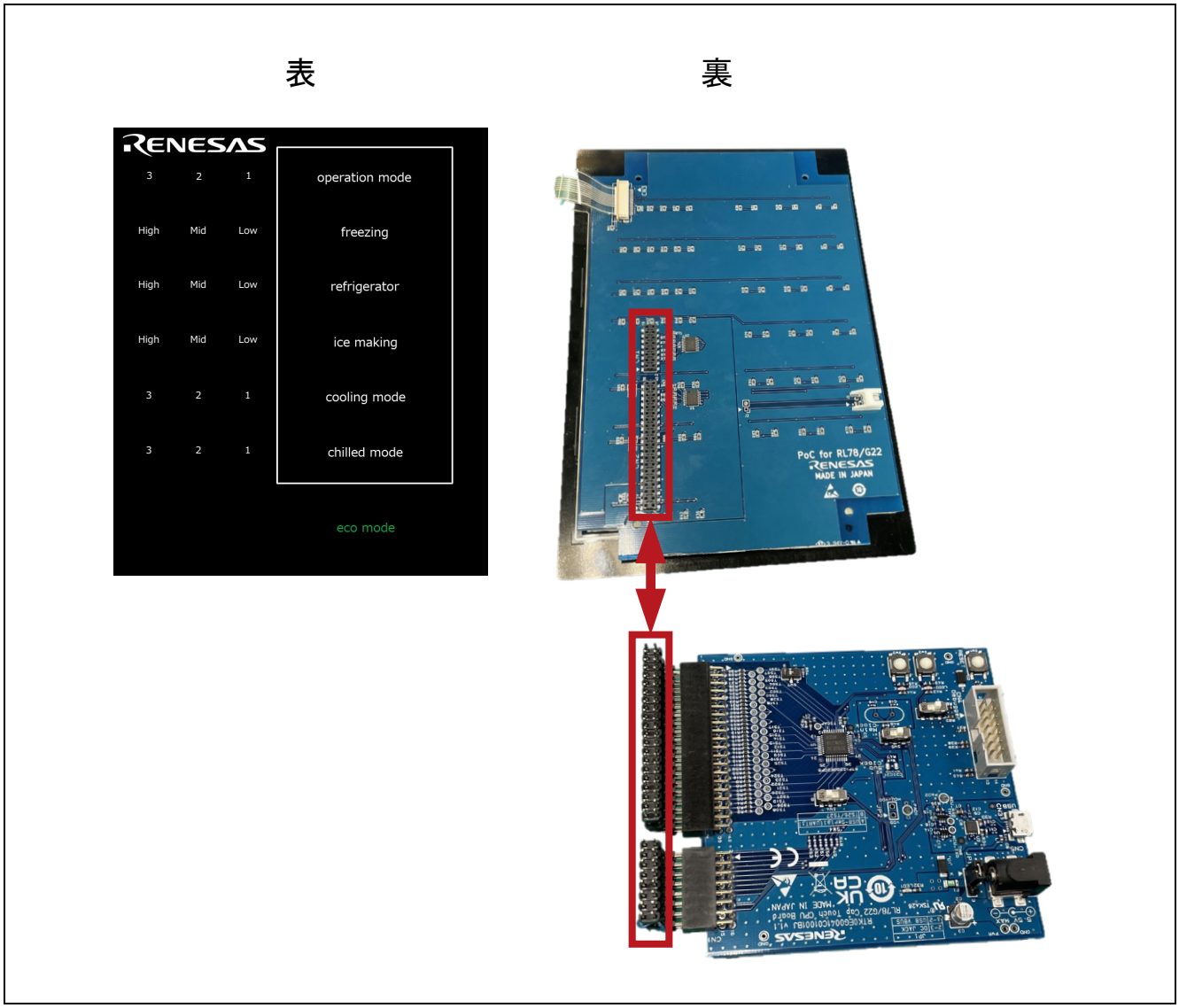

図 1-4　システム全体図

2. 動作確認条件

本サンプルプログラムは、下記の条件で動作を確認しています。

表 2-1　動作確認条件

項目	内容
CPU ボード	RL78/G22 RSSK CPU ボード (RTK0EG0041C01001BJ) (RL78/G22 静電容量タッチ評価システム (RSSK) (RTK0EG0042S01001BJ) 付属品)
電極ボード	• Touch MEC 機能向け家電パネル電極ボード（筐体あり） 自己容量方式ボタン: 7 個 LED: 25 個
使用マイコン	RL78/G22 (R7F102GGE) (ROM: 64KB, RAM: 4KB)
動作周波数（HOCO）	• メイン・システム・クロック 高速オンチップ・オシレータ・クロック (f_{IH}) : 32 MHz • CPU／周辺ハードウエア・クロック (f_{CLK}) : 32 MHz • サブシステム・クロック 低速オンチップ・オシレータ (f_{IL}) : 32.768 kHz • 低速周辺クロック周波数 (f_{SXP}) : 32.768 kHz
動作電圧	3.3V (1.8 V ～ 5.5 V で動作可能)
統合開発環境 (e^2 studio)	ルネサス エレクトロニクス製 e^2 studio Version 2023-01 (23.1.0)
Smart Configurator (SC)	ルネサス エレクトロニクス製 V1.5.0
C コンパイラ (e^2 studio)	ルネサス エレクトロニクス製 CC-RL V1.12.00
QE for Capacitive Touch	ルネサス エレクトロニクス製 V3.20
ボードサポートパッケージ（r_bsp）	V1.40
エミュレータ	Renesas E2 emulator Lite (RTE0T0002LKCE00000R)

サンプルコードでは図 2-1 に示す SIS ドライバ／ミドルウェアおよび Code Generator を使用しています。

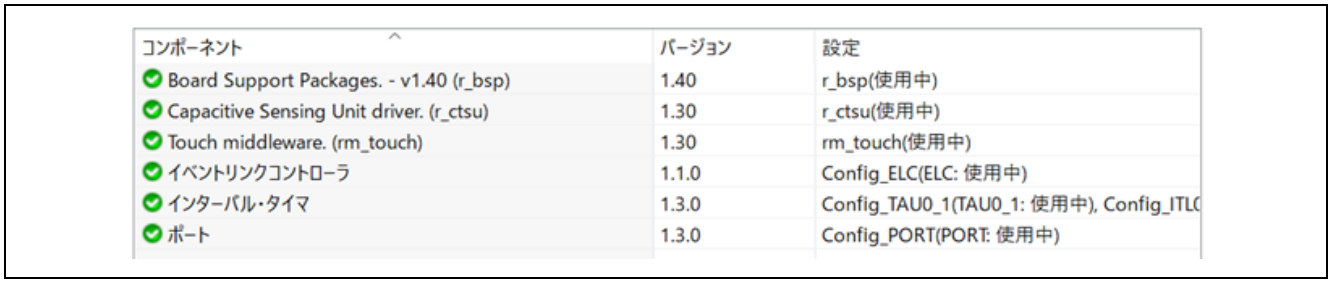

図 2-1 スマート・コンフィグレータの使用コンポーネント

3. サンプルプログラム

3.1 デモ画面の状態遷移

本サンプルプログラムのデモ画面状態遷移を以下に示します。画面の詳細は 5 章を参照してください。

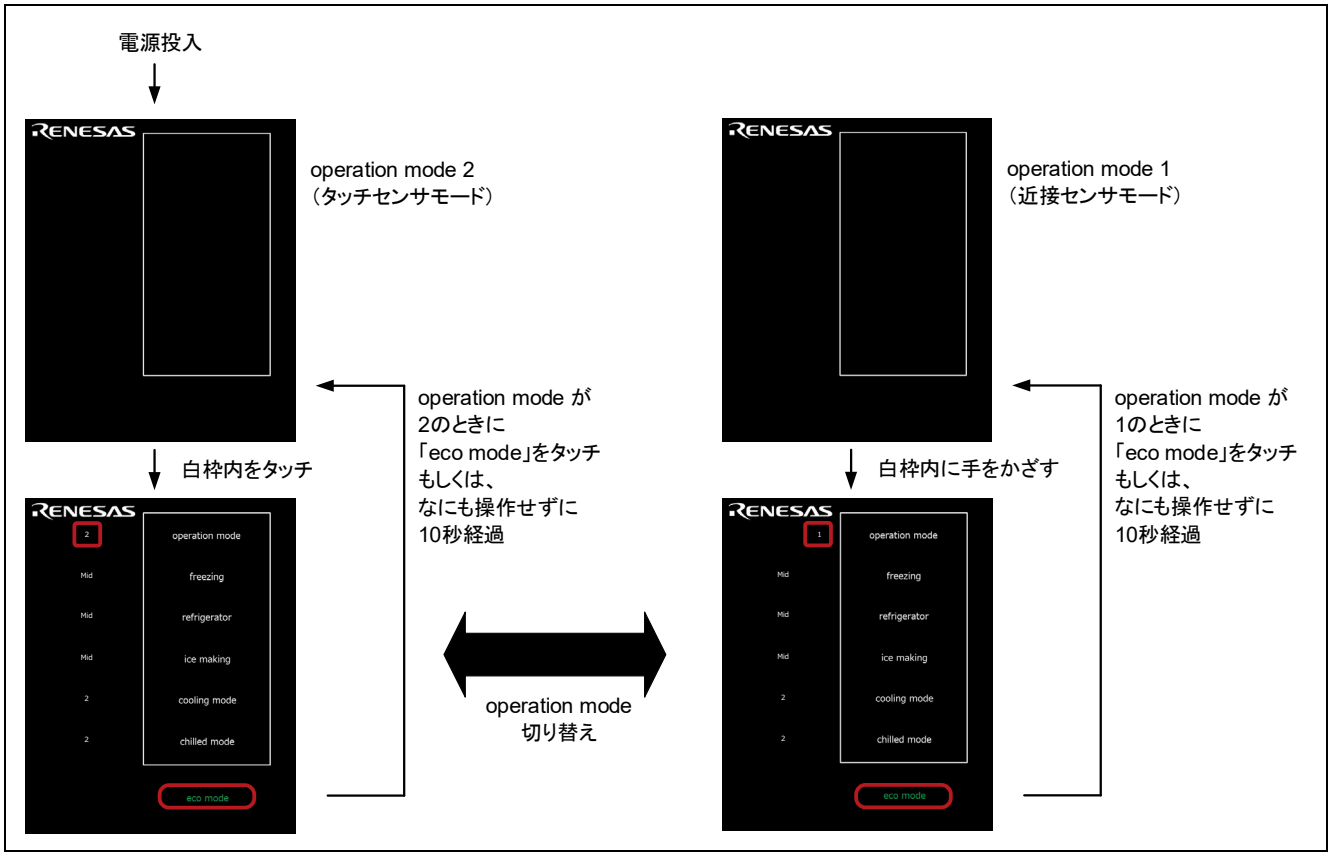

図 3-1 デモ画面の状態遷移

3.2　フローチャート

3.2.1　全体フローチャート

全体フローチャートを以下に示します。

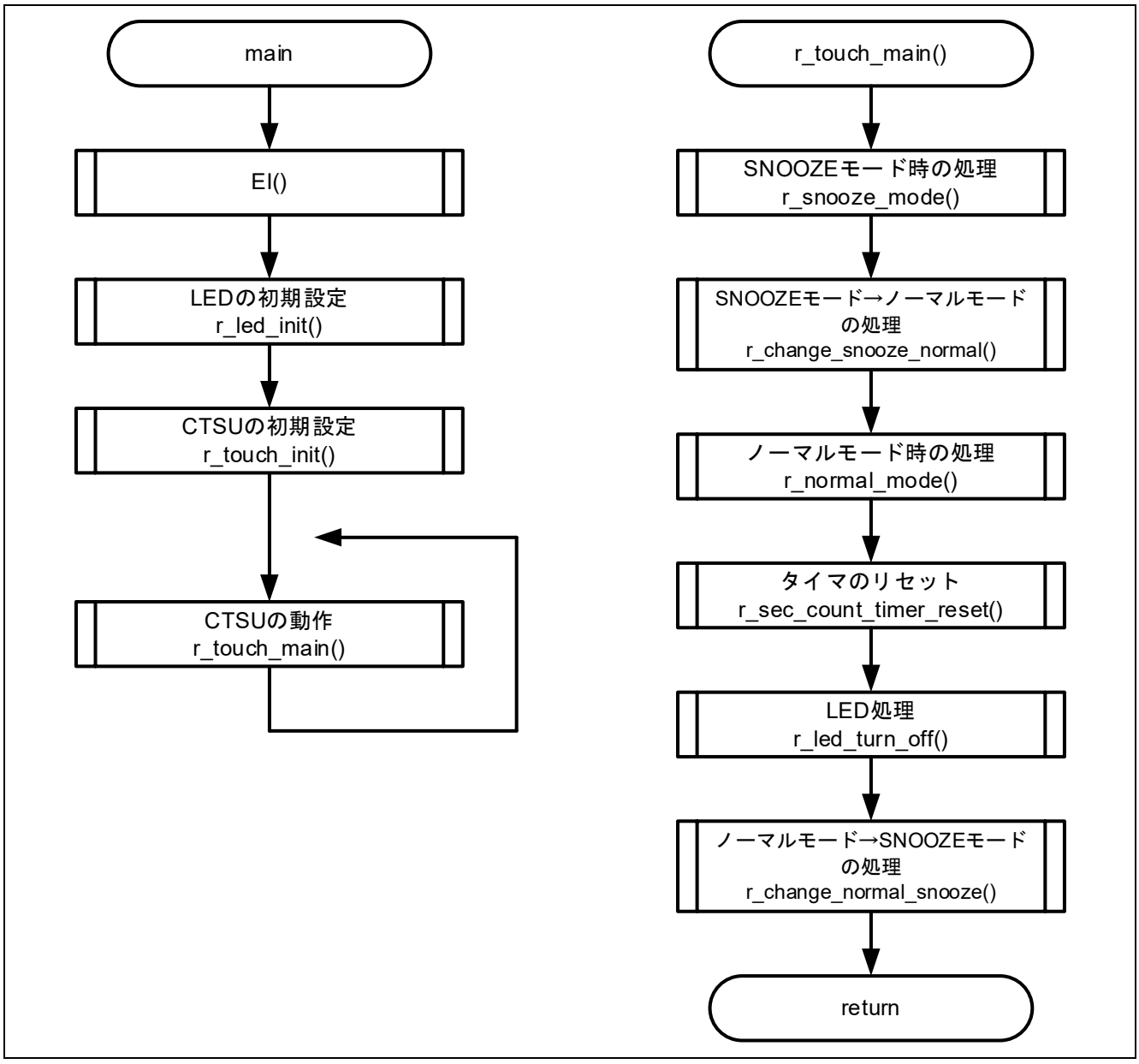

図 3-2 全体フローチャート

3.3　使用端子一覧

本サンプルプログラムの使用端子一覧を以下に示します。

表 3-1　使用端子一覧

端子名	入出力	用途
P26	入力注/出力	マトリクス LED アノード 0
P23	入力注/出力	マトリクス LED アノード 1
P21	入力注/出力	マトリクス LED アノード 2
P20	入力注/出力	マトリクス LED アノード 3
P120	入力注/出力	マトリクス LED カソード 0
P121	入力注/出力	マトリクス LED カソード 1
P122	入力注/出力	マトリクス LED カソード 2
P146	入力注/出力	マトリクス LED カソード 3
P41	入力注/出力	マトリクス LED カソード 4
P61	入力注/出力	マトリクス LED カソード 5
P62	出力	LED 単独制御
TS18	入力	operation mode
TS28	入力	freezing
TS00	入力	refrigerator
TS05	入力	ice making
TS06	入力	cooling mode
TS07	入力	chilled mode
TS01	入力	eco mode
TSCAP	-	TSCAP 端子

注：　制御対象の LED 以外はポート・モードを入力とすることで、LED の入力レベルをハイ・インピーダンスにし、発光させないようにしています。

3.4　サンプルプログラムの構成

3.4.1　使用する周辺機能

本サンプルプログラムで使用する周辺機能を以下に示します。

表 3-2　使用する周辺機能一覧

周辺機能	用途
CTSU2La	タッチボタンで使用
TAU00	秒数カウント用で使用
TAU01	LED マトリクス制御用に内部で使用
PORT	LED で使用
ITL000	CTSU2La のトリガとして使用
ELC	CTSU2La と 32bit インターバル・タイマを接続するために使用

3.4.2　周辺機能の設定

本サンプルプログラムで使用しているスマート・コンフィグレータの設定を以下に示します。スマート・コンフィグレータの設定における各表の項目、設定内容は設定画面の表記で記載しています。

表 3-3　スマート・コンフィグレータの設定

タグ名	コンポーネント	内容
クロック	-	動作モード：高速メインモード 1.8 (V) ~5.5 (V) 高速オンチップ・オシレータ：32MHz fHOCO 開始設定：通常 f_{IHP}：32MHz f_{MAIN}：32MHz f_{CLK}：32000kHz
システム	-	オンチップ・デバッグ動作設定：エミュレータを使う エミュレータ設定：E2 Lite 疑似 RRM/DMM 機能設定：使用する Start/Stop 関数機能設定：使用しない セキュリティ ID 設定：セキュリティ ID を設定する セキュリティ ID：0x00000000000000000000 セキュリティ ID 認証失敗時の設定：フラッシュ・メモリのデータを消去する
コンポーネント	r_bsp	Start up select : Enable (use BSP startup) Control of invalid memory access detection : Disable RAM guard space (GRAM0-1) : Disabled Guard of control registers of port function (GPORT) : Disabled Guard of registers of interrupt function (GINT) : Disabled Guard of control registers of clock control function, voltage detector, and RAM parity error detection function (GCSC) : Disabled Data flash access control (DFLEN) : Disables Initialization of peripheral functions by Code Generator/Smart Configurator : Enable API functions disable : Enable Parameter check enable : Enable Setting for starting the high-speed on-chip oscillator at the times of release from STOP mode and of transitions to SNOOZE mode : High-speed Enable user warm start callback (PRE) : Unused Enable user warm start callback (POST) : Unused Watchdog Timer refresh enable : Unused
	Config_ITL000	コンポーネント：インターバル・タイマ 動作モード：8 ビット・カウンタ・モード リソース：ITL000 動作クロック：f_{SXP} クロックソース：f_{ITL0}/16 インターバル時間；20ms 割り込み設定：使用しない

タグ名	コンポーネント	内容
コンポーネント	Config_TAU0_0	コンポーネント：インターバル・タイマ 動作モード：16 ビット・カウンタ・モード リソース：TAU0_0 動作クロック：CK00 クロックソース：f_{CLK}/2^10 インターバル時間：1000ms 割り込み設定：使用する 優先順位：レベル 3
	Config_TAU0_1	コンポーネント：インターバル・タイマ 動作モード：16 ビット・カウンタ・モード リソース：TAU0_1 動作クロック：CK01 クロックソース：f_{CLK}/2^8 インターバル時間：7ms 割り込み設定：使用する 優先順位：レベル 3
	Config_ELC	コンポーネント：イベントリンクコントローラ 出力先設定：CTSU2La 静電容量センシングユニット イベント発生元：32 -bit インターバルタイマー0 比較一致
	Config_PORT	コンポーネント：ポート ポート選択：PORT2、PORT4、PORT6、PORT12、PORT14 ポート・モード設定：Pmn レジスタ値を読み出す PORT2：P20、P21、P23、P26 を入力 PORT4：P41 を入力 PORT6：P61、P62 を出力　1 を出力 PORT12：P120～P122 を入力 PORT14：P146 を入力
	r_ctsu	コンポーネント：r_ctsu リソース：CTSU TS00 端子：使用する TS01 端子：使用する TS05 端子：使用する TS06 端子：使用する TS07 端子：使用する TS18 端子：使用する TS28 端子：使用する 以上以外の設定はデフォルトとする
	rm_touch	コンポーネント：rm_touch デフォルトの設定とする

3.4.3　ファイル構成

本サンプルプログラムのファイル構成を以下に示します。

表 3-4 ファイル構成

フォルダ名、ファイル名	説明
r01an6740_g22demo_touch_mec	プログラム格納用フォルダ
├ binary	メイン処理ソースファイル
├ includes	メイン処理ヘッダファイル
├ qe_gen	QE for capacitive touch 生成
├ src	LCD 関連ヘッダファイル
│　├ smc_gen	スマート・コンフィグレータ生成
│　│　├ Config_ELC	
│　│　├ Config_ITL000	
│　│　├ Config_PORT	
│　│　├ Config_TAU0_0	
│　│　├ Config_TAU0_1	
│　│　├ general	
│　│　├ r_bsp	
│　│　├ r_config	
│　│　├ r_ctsu	
│　│　├ r_pincfg	
│　│　└ rm_touch	
│　├ led.c	LED 制御ソースファイル
│　├ led.h	LED 制御ヘッダファイル
│　├ main.c	メイン処理ソースファイル
│　├ mode.c	mode 制御ソースファイル
│　├ mode.h	mode 制御ヘッダファイル
│　├ touch.c	touch 制御ソースファイル
│　└ touch.h	touch 制御ヘッダファイル
└QE-Touch	QE-Touch 生成

3.4.4　変数一覧

本サンプルプログラムで使用する変数一覧を以下に示します。

表 3-5　サンプルプログラムで使用する変数一覧

型	変数名	内容	ファイル
uint8_t	g_led_position[6]	マトリクス LED の点灯パタンの配列	led.c mode.c touch.c
uint8_t	g_pos_a	マトリクス LED の制御 LED のアノード側の指定変数	led.c
uint8_t	g_touch_button_flg	なにかボタンがタッチされたフラグ	led.c mode.c touch.c
uint8_t	g_eco_mode_flg	「eco mode」ボタンがタッチされたフラグ	led.c touch.c
uint8_t	g_sec_count_timer_count	秒数カウント変数	led.c Config_TAU0_0_user.c
uint8_t	g_sec_count_timer_stop_flg	秒数カウントの処理を終えたフラグ	Config_TAU0_0_user.c
uint8_t	g_mode	ノーマルモード/SNOOZE モードの切り替え変数	led.c mode.c touch.c
uint8_t	g_normal_end_flg	ノーマルモードの処理を終えたフラグ	led.c mode.c
uint64_t	g_button_status	どのボタンがタッチされたのかステータス	led.c mode.c touch.c
uint8_t	g_snooze_mode_init	SNOOZE モード初期化フラグ	mode.c
uint8_t	g_normal_mode_init	ノーマルモード初期化フラグ	mode.c
uint8_t	g_snooze_end_flg	SNOOZE モードの処理を終えたフラグ	mode.c

3.4.5　定数一覧

本サンプルプログラムで使用する定数一覧を以下に示します。

表 3-6　サンプルプログラムで使用する定数一覧

定数名	設定値	内容	ファイル
MEC	0x01	MEC のボタンタッチ時の g_button_status の値	touch.h
OPERATION_MODE	0x20	「operation mode」ボタンタッチ時の g_button_status の値	touch.h
FREEZING	0x40	「freezing」ボタンタッチ時の g_button_status の値	touch.h
REFRIGERATOR	0x01	「refrigerator」ボタンタッチ時の g_button_status の値	touch.h
ICE_MAKING	0x04	「ice making」ボタンタッチ時の g_button_status の値	touch.h
COOLING_MODE	0x08	「cooling mode」ボタンタッチ時の g_button_status の値	touch.h
CHILLED_MODE	0x10	「chilled mode」ボタンタッチ時の g_button_status の値	touch.h
ECO_MODE	0x02	「eco mode」ボタンタッチ時の g_button_status の値	touch.h
P_LED_A0	P2_bit.no6	マトリクス LED のアノード側の端子	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	マトリクス LED のカソード側の端子	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	マトリクス LED のアノード側の端子のポート・モード・レジスタ	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	マトリクス LED のカソード側の端子のポート・モード・レジスタ	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	エコモード用の LED 端子	led.c
LED_ON	0U	LED 点灯	led.c
LED_OFF	1U	LED 消灯	led.c
OUTPUT	0U	ポート・モード・レジスタを出力にする	led.c
INPUT	1U	ポート・モード・レジスタを入力にする	led.c
COUNT_10S	10U	10s カウント用定数	led.c
COUNT_5S	5U	5s カウント用定数	led.c

3.4.6　関数一覧

本サンプルプログラムで使用する関数一覧を以下に示します。

表 3-7　サンプルプログラムで使用する関数一覧

関数名	概要	ソースファイル
main	メイン処理	main.c
r_led_init	LED の初期化処理	led.c
r_ledport_input	LED ポートを入力モードにする	led.c
r_ledport_output	LED ポートを出力モードにする	led.c
r_led_turn_on_all_5s	すべての LED を 5s 間点灯する処理	led.c
r_led_turn_on	LED 点灯処理	led.c
r_led_turn_off	LED 消灯処理	led.c
r_ledmatrix_turn_on	マトリクス LED 点灯処理	led.c
r_ledmatrix_turn_off	マトリクス LED 消灯処理	led.c
r_ledmatrix_turn_on_a	マトリクス LED アノード側の点灯処理	led.c
r_change_led_position	マトリクス LED のポジション変更処理	led.c
r_change_led	マトリクス LED の点灯パタンの変更処理	led.c
r_snooze_mode_init	SNOOZE モードの初期化処理	mode.c
r_normal_mode_init	ノーマルモードの初期化処理	mode.c
r_snooze_mode	SNOOZE モードの動作処理	mode.c
r_normal_mode	ノーマルモードの動作処理	mode.c
r_change_snooze_normal	SNOOZE モードからノーマルモードへの変更処理	mode.c
r_change_normal_snooze	ノーマルモードから SNOOZE モードへの変更処理	mode.c
r_not_touched	タッチボタンに触れていない判定処理	mode.c
r_touch_init	CTSU2La の初期設定	touch.c
r_touch_main	ボタンをタッチしたときのメイン動作	touch.c
r_snooze_mode_touch_prosses	SNOOZE モードのタッチ処理	touch.c
r_change_eco_mode	エコモードの変更処理	touch.c
r_prevent_long_presses	長押し防止処理	touch.c
r_touch_mec_scanstart	MEC の ScanStart の切り替え関数	touch.c
r_touch_mec_scanstop	MEC の ScanStop の切り替え関数	touch.c
r_touch_mec_dataget	MEC の DataGet の切り替え関数	touch.c
r_Config_TAU0_0_interrupt	TAU0_0 の割り込み関数	Config_TAU0_0_user.c
r_sec_count_timer_start	秒数カウント用タイマのスタート処理	Config_TAU0_0_user.c
r_sec_count_timer_reset	秒数カウント用タイマのリセット処理	Config_TAU0_0_user.c
r_Config_TAU0_1_interrupt	TAU0_1 の割り込み関数	Config_TAU0_1_user.c
r_ledmatrix_timer_start	マトリクス LED 制御用タイマスタート処理	Config_TAU0_1_user.c
r_ledmatrix_timer_reset	マトリクス LED 制御用タイマリセット処理	Config_TAU0_1_user.c

3.4.7　関数仕様

本サンプルプログラムの関数仕様を以下に示します。

[関数名]main

概要	メイン処理
ヘッダ	r_smc_entry.h, touch.h, led.h
宣言	int main (void)
説明	LED と CTSU2La の初期化を行い、タッチ操作を繰り返します。
引数	なし
リターン値	なし
備考	なし

[関数名] r_led_init

概要	LED の初期化処理
ヘッダ	led.h
宣言	void r_led_init(void)
説明	電源を入れるとすべての LED を 5s 間点灯させ、g_led_position の初期化を行います。
引数	なし
リターン値	なし
備考	なし

[関数名] r_ledport_input

概要	LED ポートを入力モードにする
ヘッダ	led.h
宣言	void r_ledport_input(void)
説明	LED ポートを入力モードに変更します。
引数	なし
リターン値	なし
備考	なし

[関数名] r_ledport_output

概要	LED ポートを出力モードにする
ヘッダ	led.h
宣言	void r_ledport_output(void)
説明	LED ポートを出力モードに変更します。
引数	なし
リターン値	なし
備考	なし

[関数名] r_led_turn_on_all_5s

概要	すべての LED を 5s 間点灯する処理
ヘッダ	led.h
宣言	void r_led_turn_on_all_5s(void)
説明	すべての LED を 5s 間点灯します。
引数	なし
リターン値	なし
備考	なし

[関数名] r_led_turn_on

概要	LED 点灯処理
ヘッダ	led.h
宣言	void r_led_turn_on(void)
説明	LED を点灯します。
引数	なし
リターン値	なし
備考	なし

[関数名] r_led_turn_off

概要	LED 消灯処理
ヘッダ	led.h
宣言	void r_led_turn_off(void)
説明	LED を消灯して、タイマをリセットします。
引数	なし
リターン値	なし
備考	なし

[関数名] r_ledmatrix_turn_on

概要	マトリクス LED 点灯処理
ヘッダ	led.h
宣言	void r_ledmatrix_turn_on(void)
説明	マトリクス LED を点灯します。
引数	なし
リターン値	なし
備考	なし

[関数名] r_ledmatrix_turn_off

概要	マトリクス LED 消灯処理
ヘッダ	led.h
宣言	void r_ledmatrix_turn_off(void)
説明	マトリクス LED を消灯します。
引数	なし
リターン値	なし
備考	なし

[関数名] r_ledmatrix_turn_on_a

概要	マトリクス LED アノード側の点灯処理
ヘッダ	led.h
宣言	void r_ledmatrix_turn_on_a(void)
説明	マトリクス LED のアノード側の点灯制御をします。
引数	なし
リターン値	なし
備考	なし

[関数名] r_change_led_position

概要	マトリクス LED のポジション変更処理
ヘッダ	led.h
宣言	void r_change_led_position(uint8_t *pos,uint8_t status)
説明	マトリクス LED のポジション変更処理をします。
引数	* pos, status
リターン値	なし
備考	なし

[関数名] r_change_led

概要	マトリクス LED の点灯パタンの変更処理
ヘッダ	led.h
宣言	void r_change_led(void)
説明	マトリクス LED の点灯パタンの変更処理をします。
引数	なし
リターン値	なし
備考	なし

[関数名] r_snooze_mode_init

概要	SNOOZE モードの初期化処理
ヘッダ	mode.h
宣言	void r_snooze_mode_init(void)
説明	SNOOZE モードの初期化をします。
引数	なし
リターン値	なし
備考	なし

[関数名] r_normal_mode_init

概要	ノーマルモードの初期化処理
ヘッダ	mode.h
宣言	void r_normal_mode_init(void)
説明	ノーマルモードの初期化をします。
引数	なし
リターン値	なし
備考	なし

[関数名] r_snooze_mode

概要	SNOOZE モードの動作処理
ヘッダ	mode.h
宣言	void r_snooze_mode(void)
説明	SNOOZE モード時の処理をします。
引数	なし
リターン値	なし
備考	なし

[関数名] r_normal_mode

概要	ノーマルモードの動作処理
ヘッダ	mode.h
宣言	void r_normal_mode(void)
説明	ノーマルモード時の処理をします。
引数	なし
リターン値	なし
備考	なし

[関数名] r_change_snooze_normal

概要	SNOOZE モードからノーマルモードへの変更処理
ヘッダ	mode.h
宣言	void r_change_snooze_normal(void)
説明	SNOOZE モードからノーマルモードへの変更処理をします。
引数	なし
リターン値	なし
備考	なし

[関数名] r_change_normal_snooze

概要	ノーマルモードから SNOOZE モードへの変更処理
ヘッダ	mode.h
宣言	void r_change_normal_snooze(void)
説明	ノーマルモードから SNOOZE モードへの変更処理をします。
引数	なし
リターン値	なし
備考	なし

[関数名] r_not_touched

概要	タッチボタンに触れていない判定処理
ヘッダ	mode.h
宣言	void r_not_touched(void)
説明	タッチボタンがタッチされたかどうかを判断する処理をします。
引数	なし
リターン値	なし
備考	なし

[関数名] r_touch_init

概要	CTSU2La の初期設定
ヘッダ	touch.h
宣言	void r_touch_init(void)
説明	CTSU2La の初期設定をします。
引数	なし
リターン値	なし
備考	なし

[関数名] r_touch_main

概要	ボタンをタッチしたときのメイン動作
ヘッダ	touch.h
宣言	void touch_main(void)
説明	ボタンをタッチしたときのメイン処理をします。
引数	なし
リターン値	なし
備考	なし

[関数名] r_snooze_mode_touch_prosses

概要	SNOOZE モードのタッチ処理
ヘッダ	touch.h
宣言	void r_snooze_mode_touch_prosses(void)
説明	SNOOZE モードのタッチ処理をします。
引数	なし
リターン値	なし
備考	なし

[関数名] r_change_eco_mode

概要	エコモードの変更処理
ヘッダ	touch.h
宣言	void r_change_eco_mode(void)
説明	エコモードの変更処理をします。
引数	なし
リターン値	なし
備考	なし

[関数名] r_prevent_long_presses

概要	長押し防止処理
ヘッダ	touch.h
宣言	void r_prevent_long_presses(uint64_t p_button_status)
説明	ボタン長押し防止処理をします。
引数	p_button_status
リターン値	なし
備考	なし

[関数名] r_touch_mec_scanstart

概要	MEC の ScanStart の切り替え関数
ヘッダ	touch.h
宣言	fsp_err_t r_touch_mec_scanstart(void)
説明	MEC の ScanStart の切り替え処理をしています。
引数	なし
リターン値	err
備考	なし

[関数名] r_touch_mec_scanstop

概要	MEC の ScanStop の切り替え関数
ヘッダ	touch.h
宣言	fsp_err_t r_touch_mec_scanstop(void)
説明	MEC の ScanStop の切り替え処理をしています。
引数	なし
リターン値	err
備考	なし

[関数名] r_touch_mec_dataget

概要	MEC の DataGet の切り替え関数
ヘッダ	touch.h
宣言	fsp_err_t r_touch_mec_dataget(void)
説明	MEC の DataGet の切り替え処理をしています。
引数	なし
リターン値	err
備考	なし

[関数名] r_Config_TAU0_0_interrupt

概要	TAU0_0 の割り込み関数
ヘッダ	Config_TAU0_0.h
宣言	static void __near r_Config_TAU0_0_interrupt(void)
説明	秒数カウント用タイマのカウントをしています。
引数	なし
リターン値	なし
備考	なし

[関数名] r_sec_count_timer_start

概要	秒数カウント用タイマのスタート処理
ヘッダ	Config_TAU0_0.h
宣言	void r_sec_count_timer_start(void)
説明	秒数カウント用タイマのスタート処理をします。
引数	なし
リターン値	なし
備考	なし

[関数名] r_sec_count_timer_reset

概要	秒数カウント用タイマのリセット処理
ヘッダ	Config_TAU0_0.h
宣言	void r_sec_count_timer_reset(uint8_t flg)
説明	秒数カウント用タイマのリセット処理をします。
引数	flg
リターン値	なし
備考	なし

[関数名] r_Config_TAU0_1_interrupt

概要	TAU0_1 の割り込み関数
ヘッダ	Config_TAU0_1.h
宣言	static void __near r_Config_TAU0_1_interrupt(void)
説明	マトリクス LED の点灯処理をしています。
引数	なし
リターン値	なし
備考	なし

[関数名] r_ledmatrix_timer_start

概要	マトリクス LED 制御用タイマスタート処理
ヘッダ	Config_TAU0_1.h
宣言	void r_ledmatrix_timer_start(void)
説明	マトリクス LED 制御用タイマスタート処理をします。
引数	なし
リターン値	なし
備考	なし

[関数名] r_ledmatrix_timer_reset

概要	マトリクス LED 制御用タイマリセット処理
ヘッダ	Config_TAU0_1.h
宣言	void r_ledmatrix_timer_reset(void)
説明	マトリクス LED 制御用タイマリセット処理をします。
引数	なし
リターン値	なし
備考	なし

3.4.8　フローチャート

3.4.8.1　main 関数のフローチャート

main 関数のフローチャートを以下に示します。

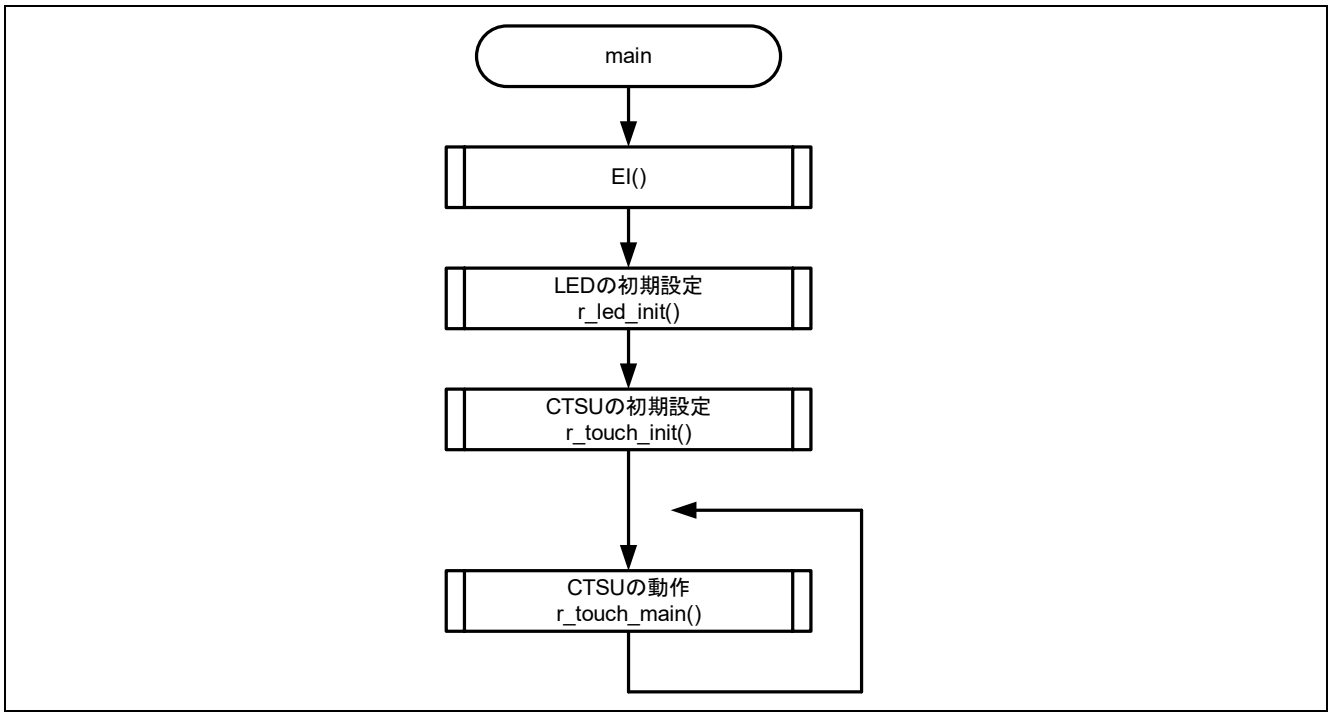

図 3-3 main 関数フローチャート

3.4.8.2　r_touch_init 関数のフローチャート

r_touch_init 関数のフローチャートを以下に示します。

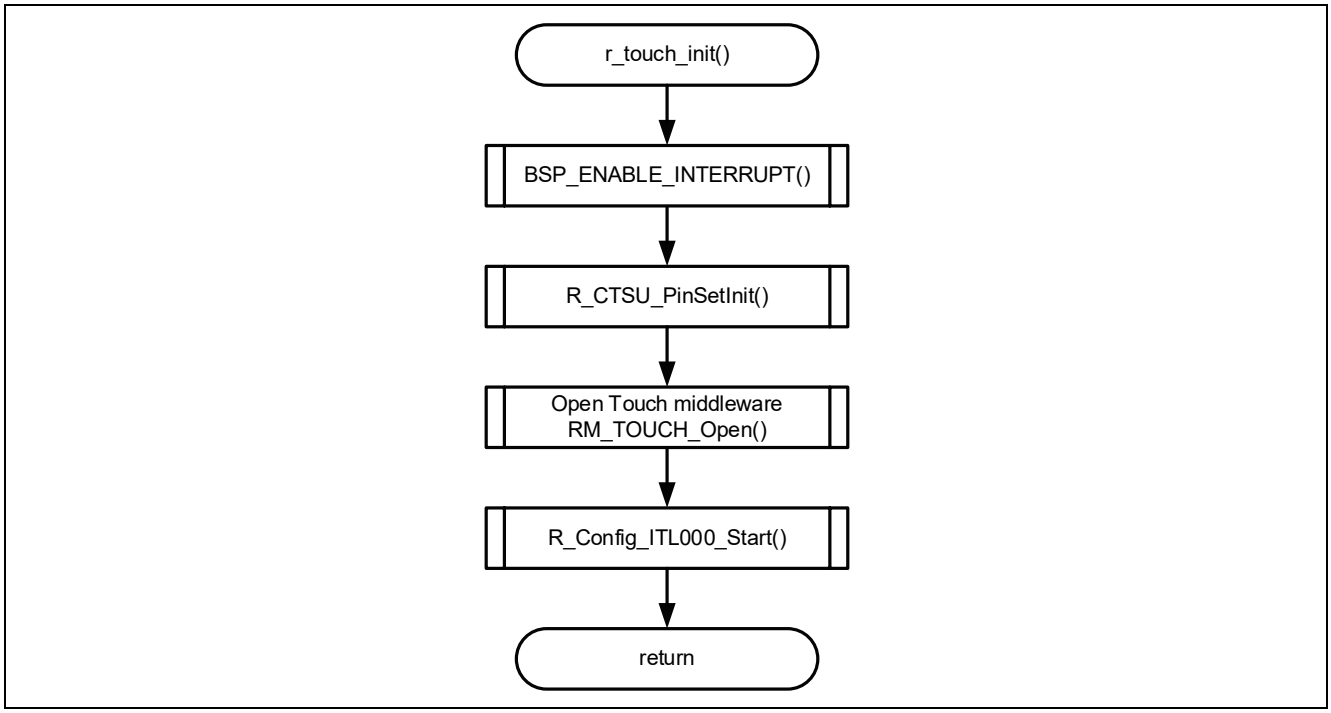

図 3-4 r_touch_init 関数フローチャート

3.4.8.3　r_touch_main 関数のフローチャート

r_touch_main 関数のフローチャートを以下に示します。

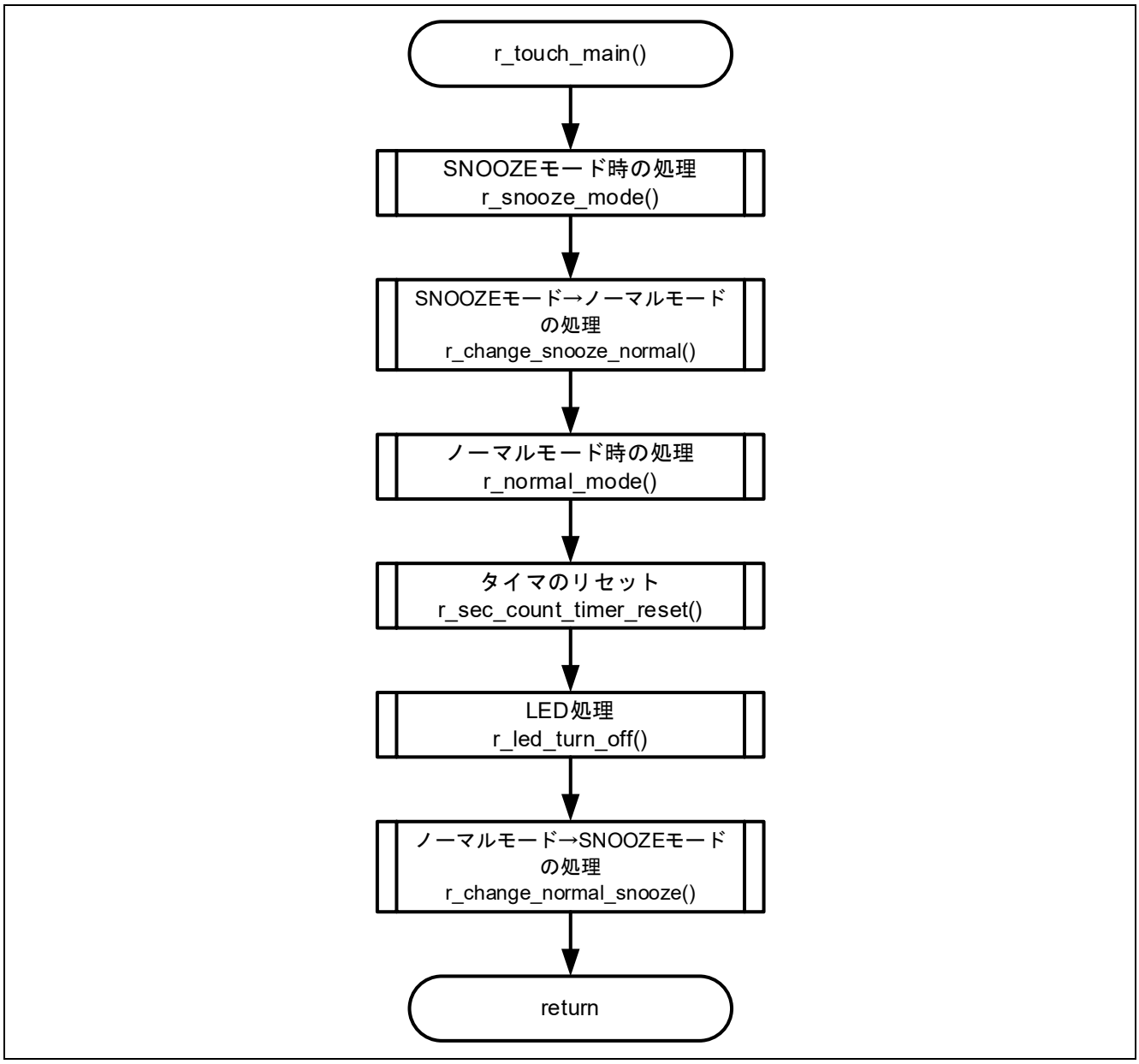

図 3-5 r_touch_main 関数フローチャート

3.4.8.4　r_snooze_mode_touch_prosses 関数のフローチャート

r_snooze_mode_touch_prosses 関数のフローチャートを以下に示します。

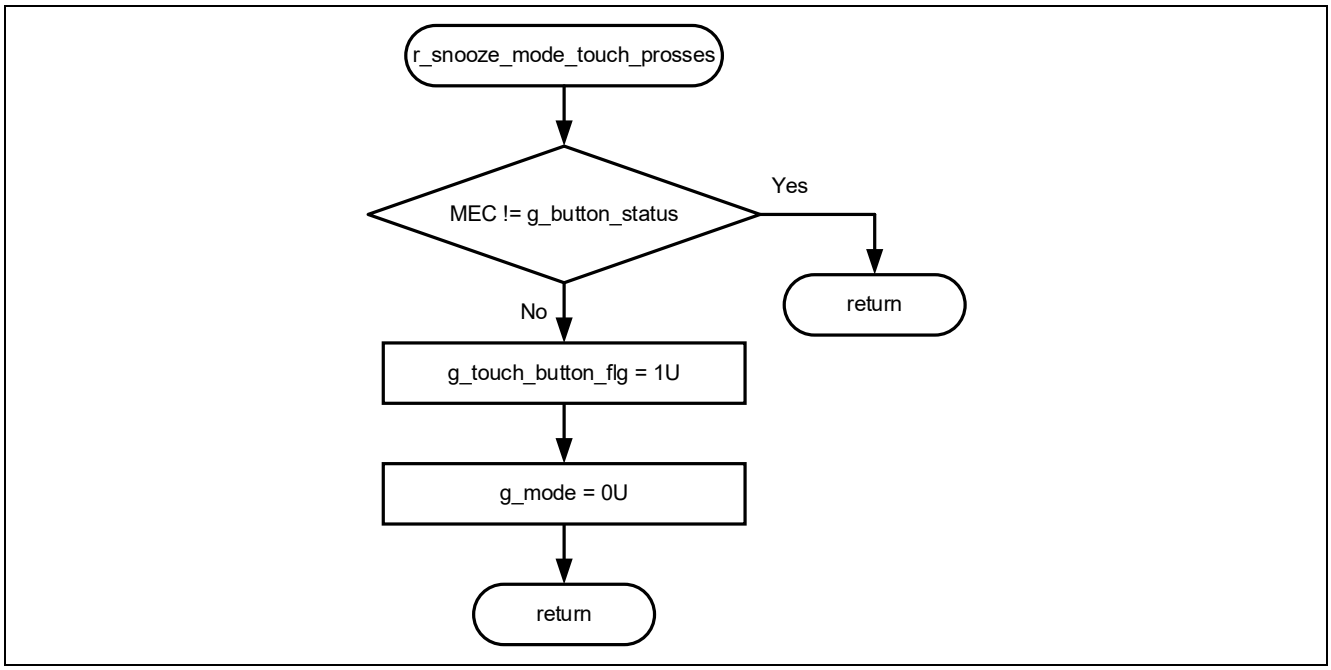

図 3-6 r_snooze_mode_touch_prosses 関数フローチャート

3.4.8.5　r_change_eco_mode 関数のフローチャート

r_change_eco_mode 関数のフローチャートを以下に示します。

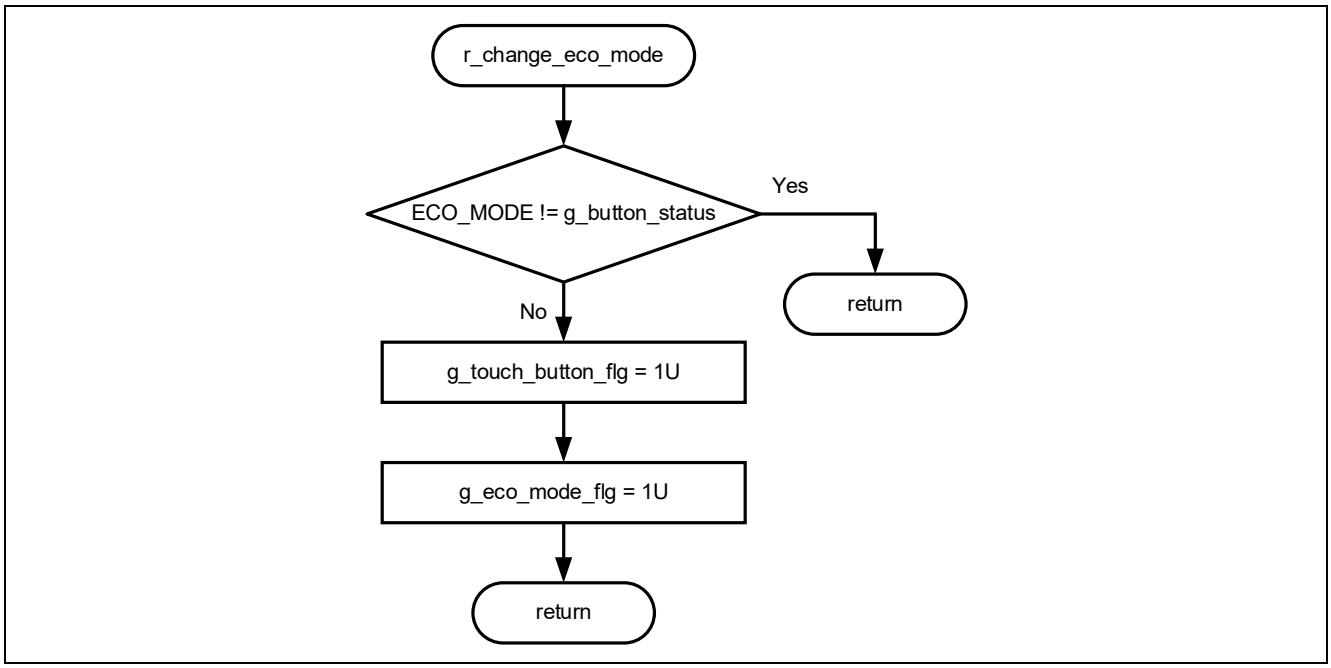

図 3-7 r_change_eco_mode 関数フローチャート

3.4.8.6　r_prevent_long_presses 関数のフローチャート

r_prevent_long_presses 関数のフローチャートを以下に示します。

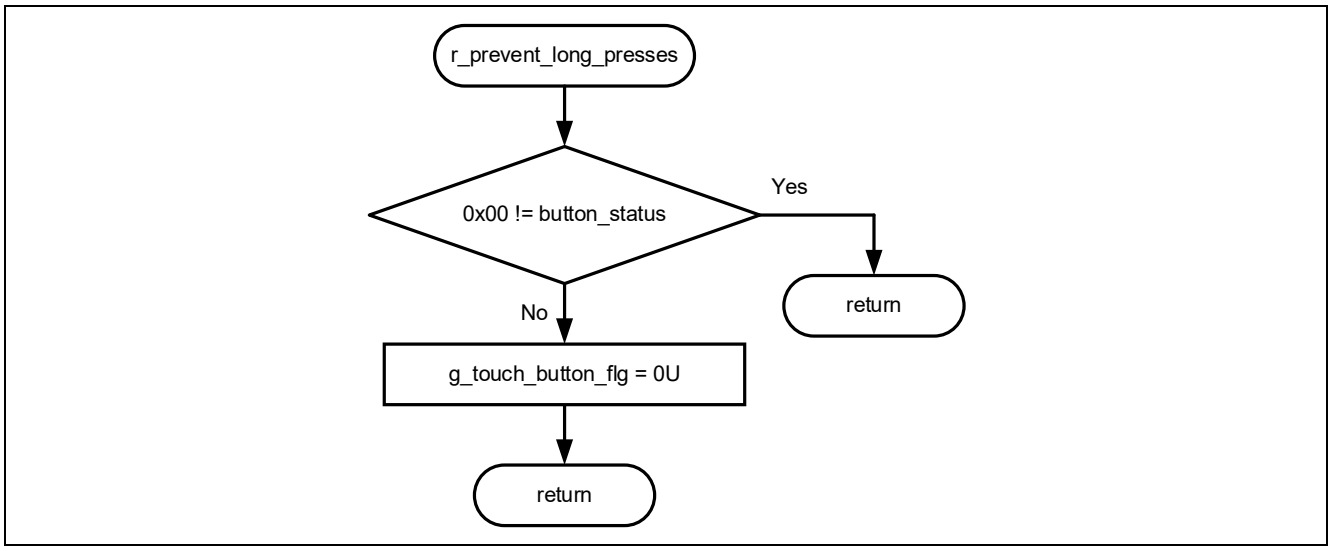

図 3-8 r_prevent_long_presses 関数フローチャート

3.4.8.7　r_snooze_mode_init 関数のフローチャート

r_snooze_mode_init 関数のフローチャートを以下に示します。

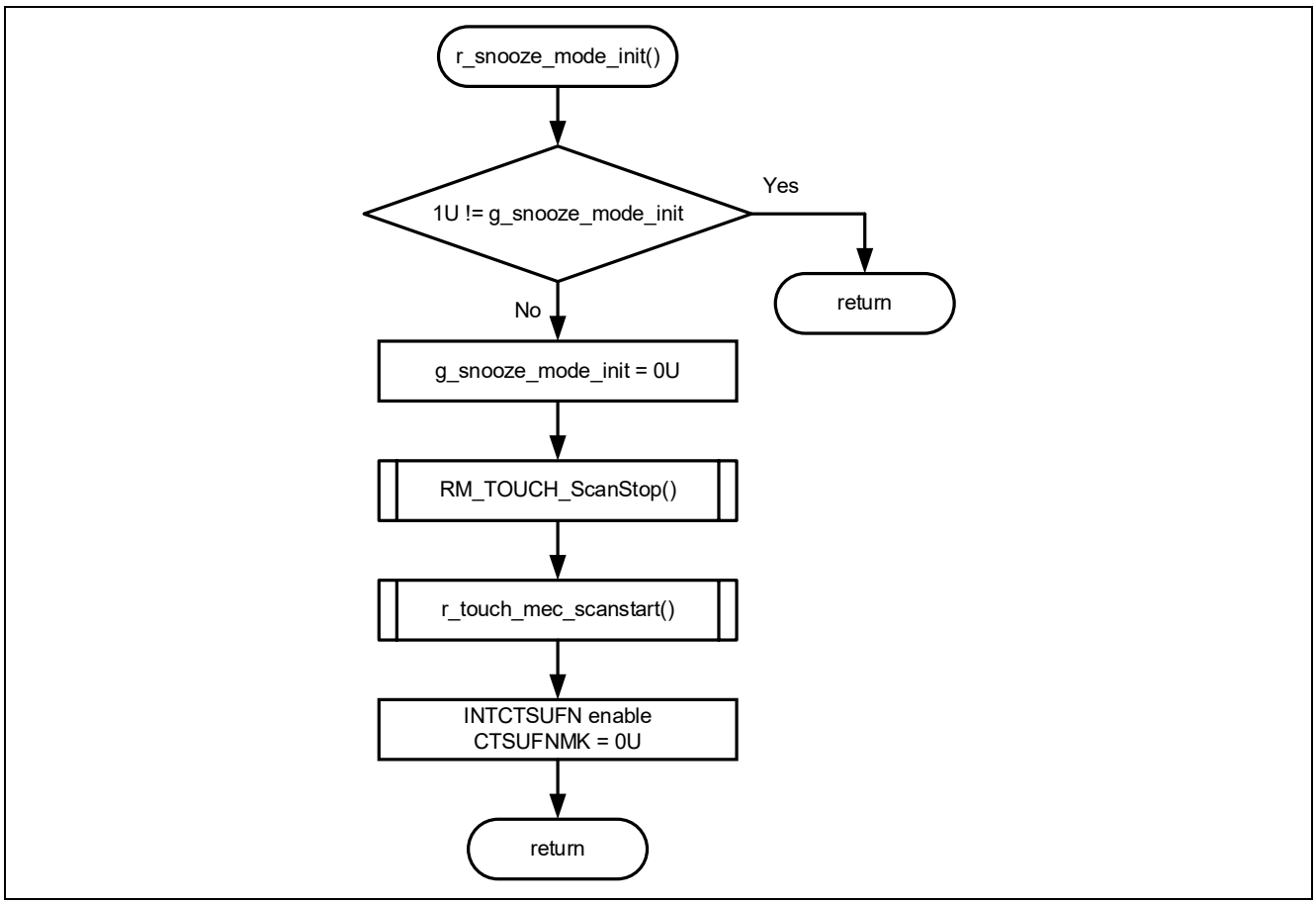

図 3-9 r_snooze_mode_init 関数フローチャート

3.4.8.8　r_snooze_mode 関数のフローチャート

r_snooze_mode 関数のフローチャートを以下に示します。

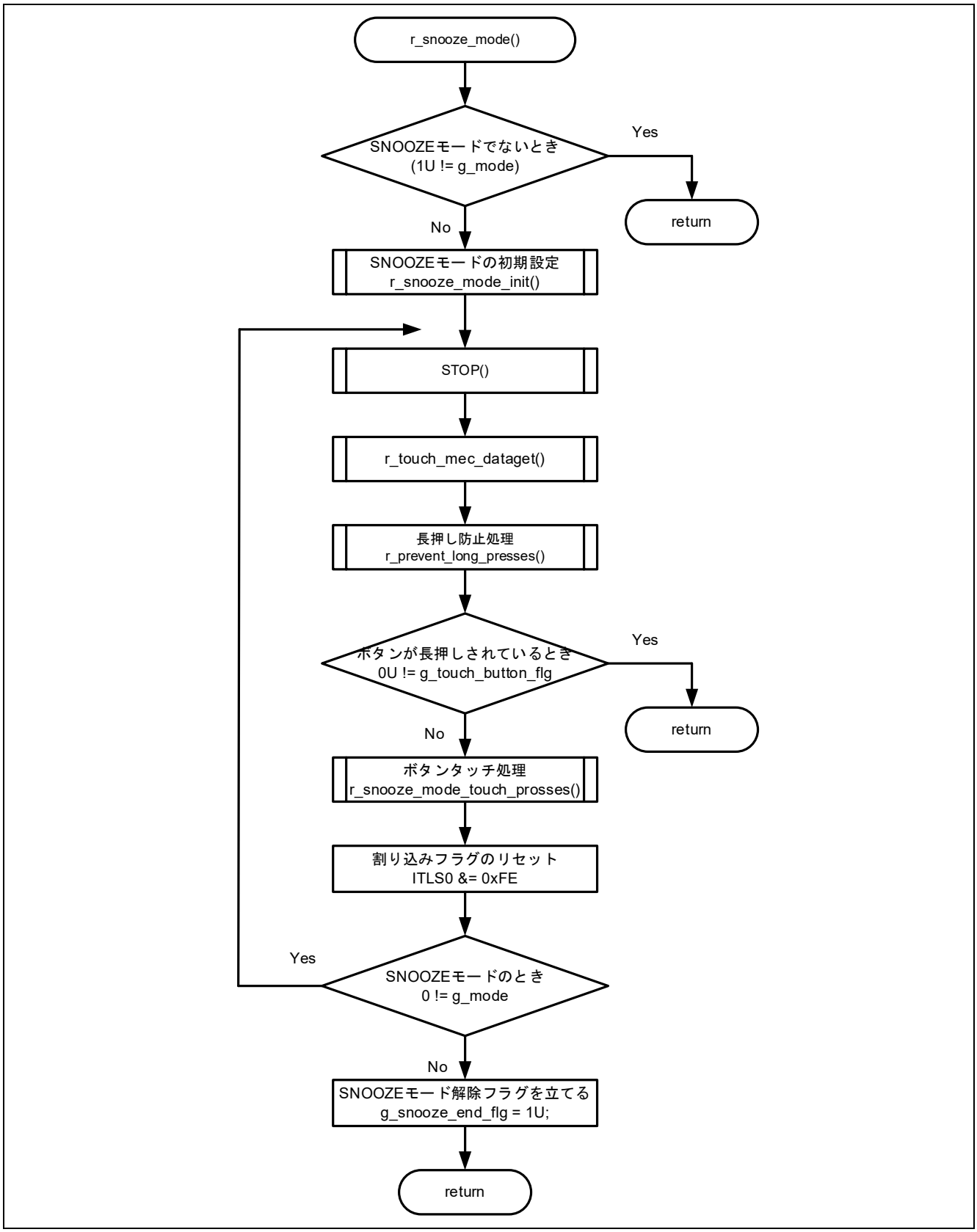

図 3-10 r_snooze_mode 関数フローチャート

3.4.8.9 r_touch_mec_scanstart 関数のフローチャート

r_touch_mec_scanstart 関数のフローチャートを以下に示します。

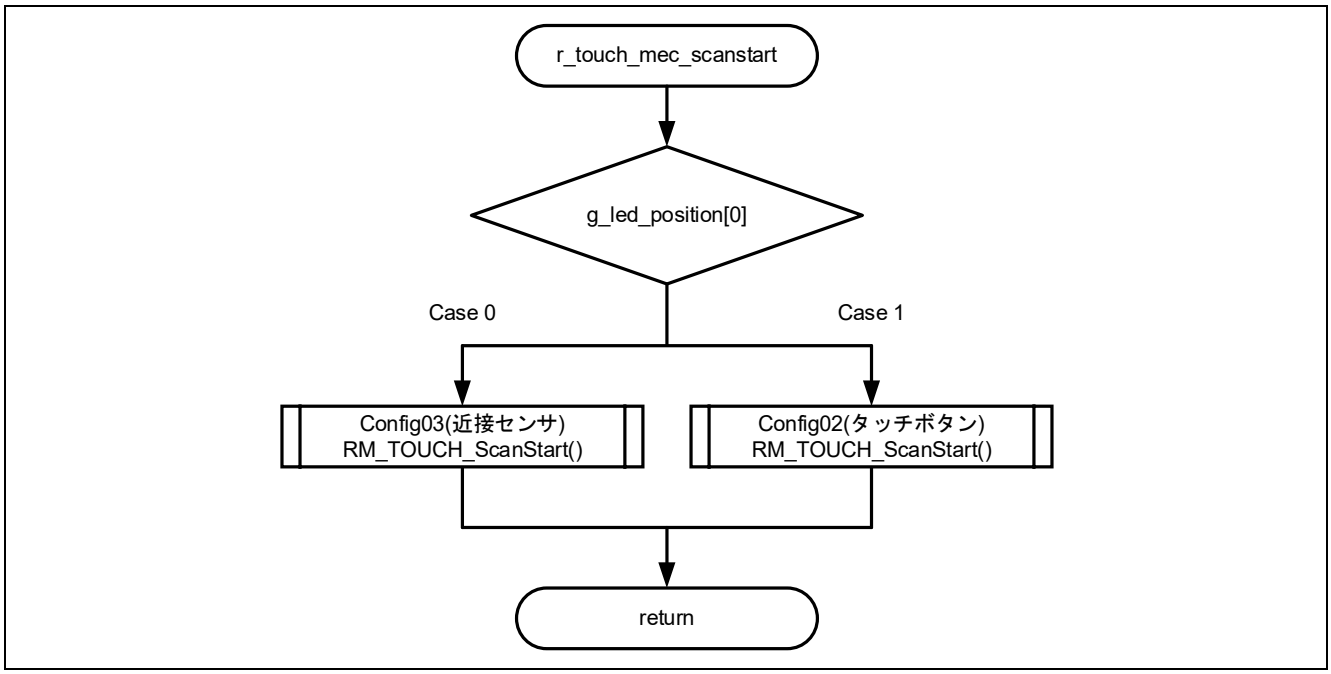

図 3-11 r_touch_mec_scanstart 関数フローチャート

3.4.8.10 r_touch_mec_scanstop 関数のフローチャート

r_touch_mec_scanstop 関数のフローチャートを以下に示します。

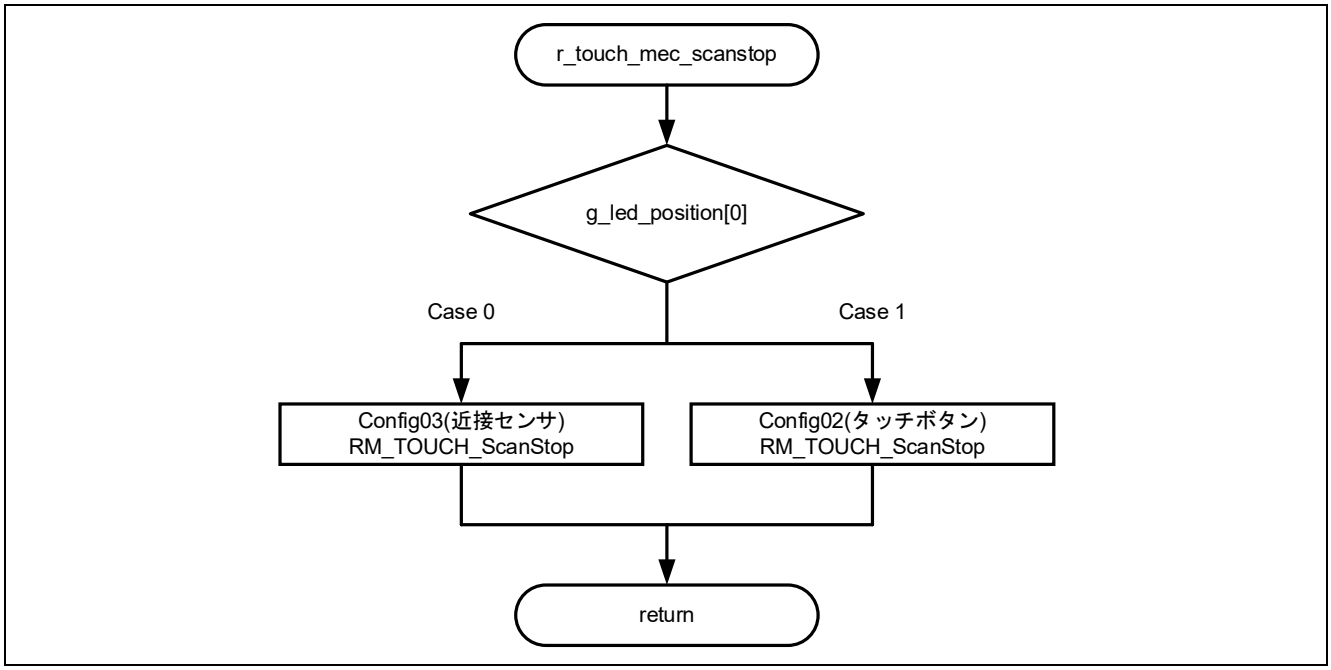

図 3-12 r_touch_mec_scanstop 関数のフローチャート

3.4.8.11 r_touch_mec_dataget 関数のフローチャート

r_touch_mec_dataget 関数のフローチャートを以下に示します。

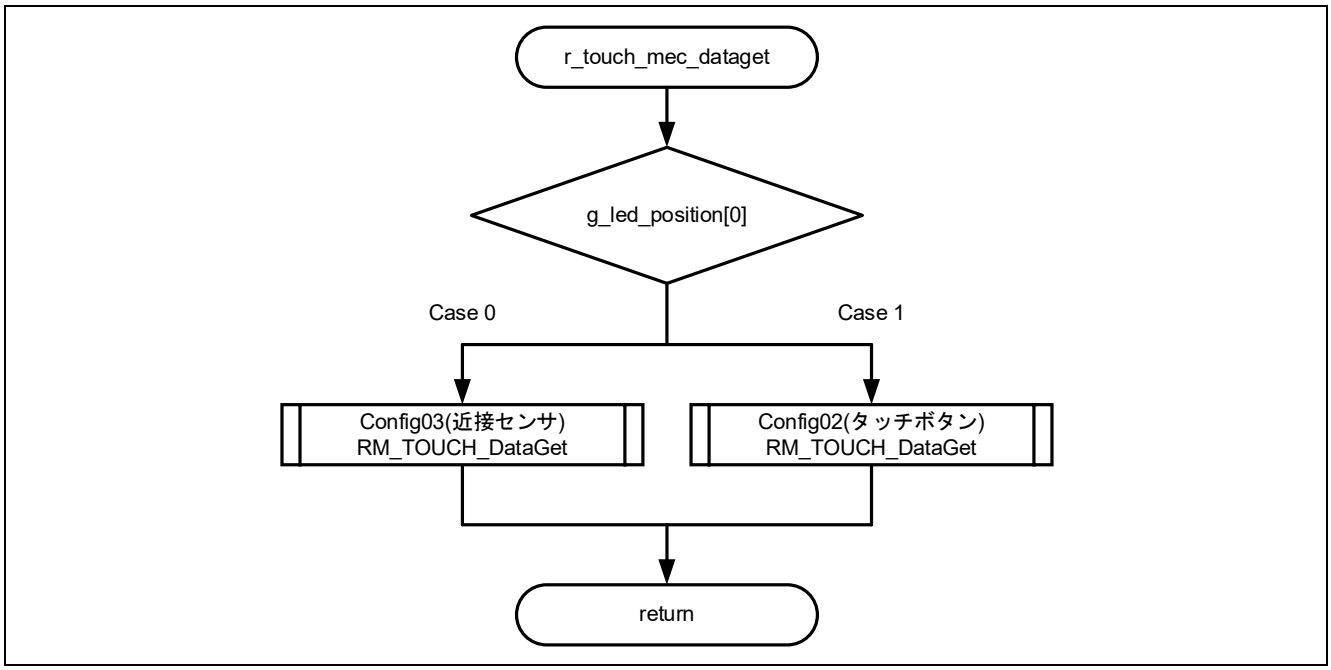

図 3-13 r_touch_mec_dataget 関数フローチャート

3.4.8.12 r_nomal_mode_init 関数のフローチャート

r_nomal_mode_init 関数のフローチャートを以下に示します。

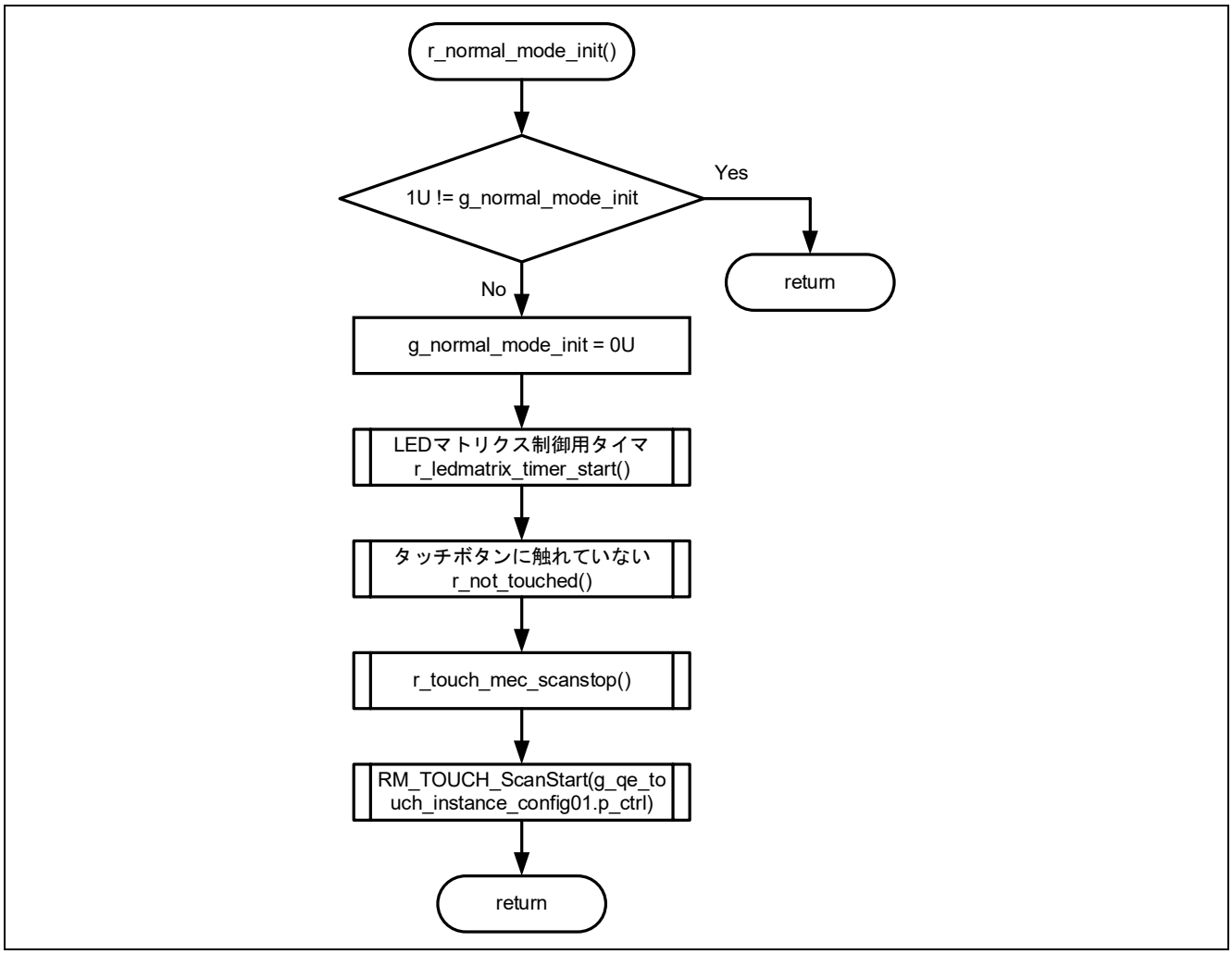

図 3-14 r_nomal_mode_init 関数フローチャート

3.4.8.13 r_nomal_mode 関数のフローチャート

r_nomal_mode 関数のフローチャートを以下に示します。

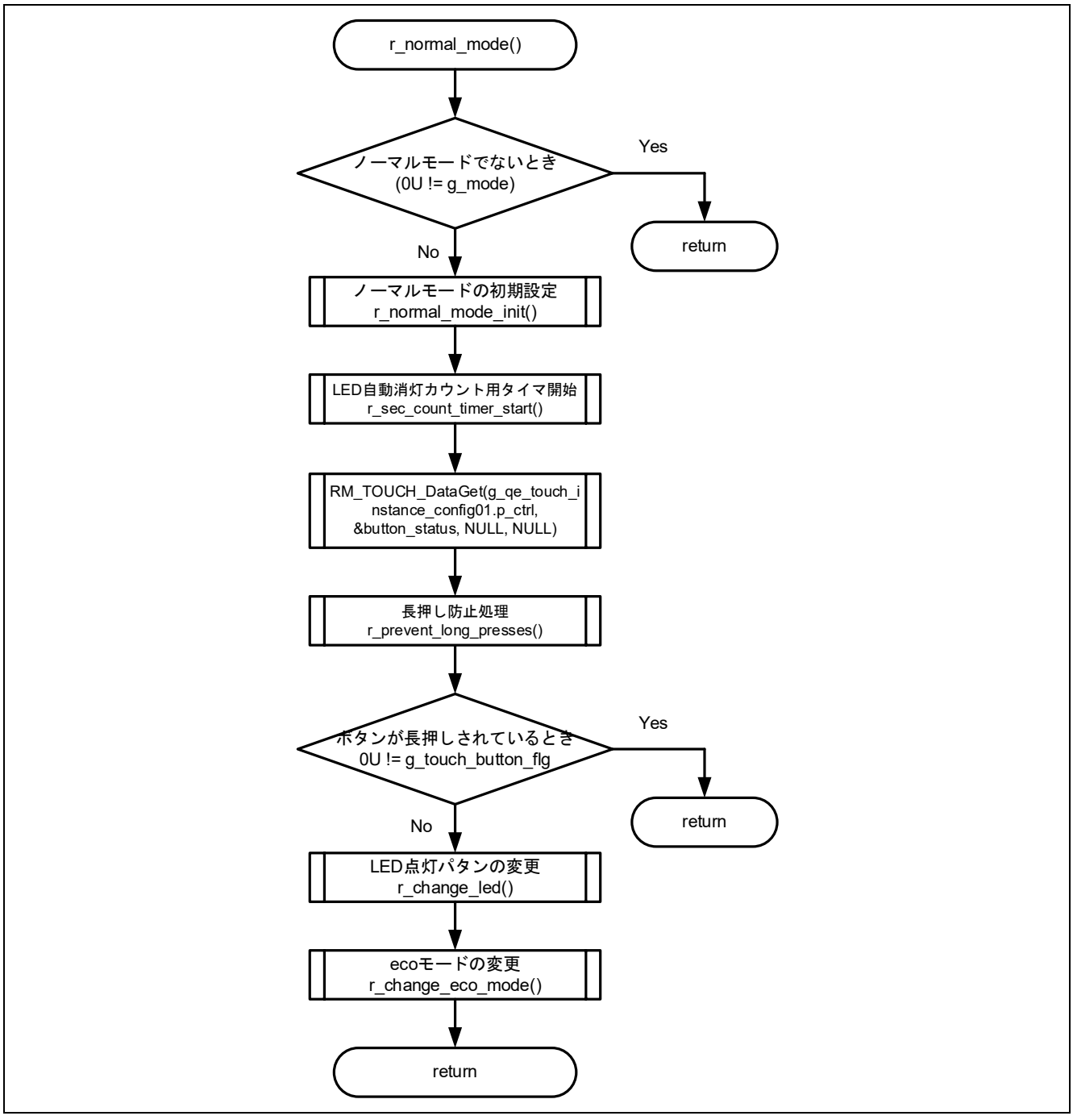

図 3-15 r_nomal_mode 関数フローチャート

3.4.8.14 r_change_snooze_nomal 関数のフローチャート

r_change_snooze_nomal 関数のフローチャートを以下に示します。

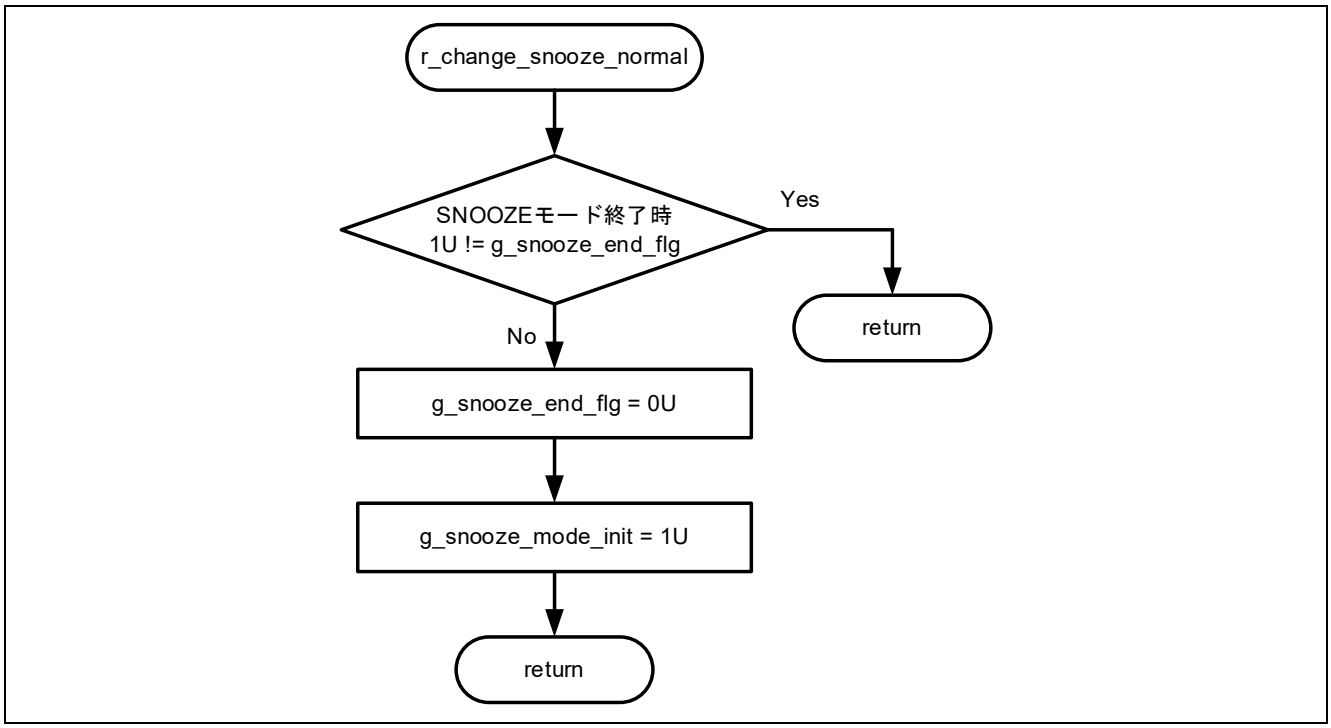

図 3-16 r_change_snooze_nomal 関数フローチャート

3.4.8.15 r_change_nomal_snooze 関数のフローチャート

r_change_nomal_snooze 関数のフローチャートを以下に示します。

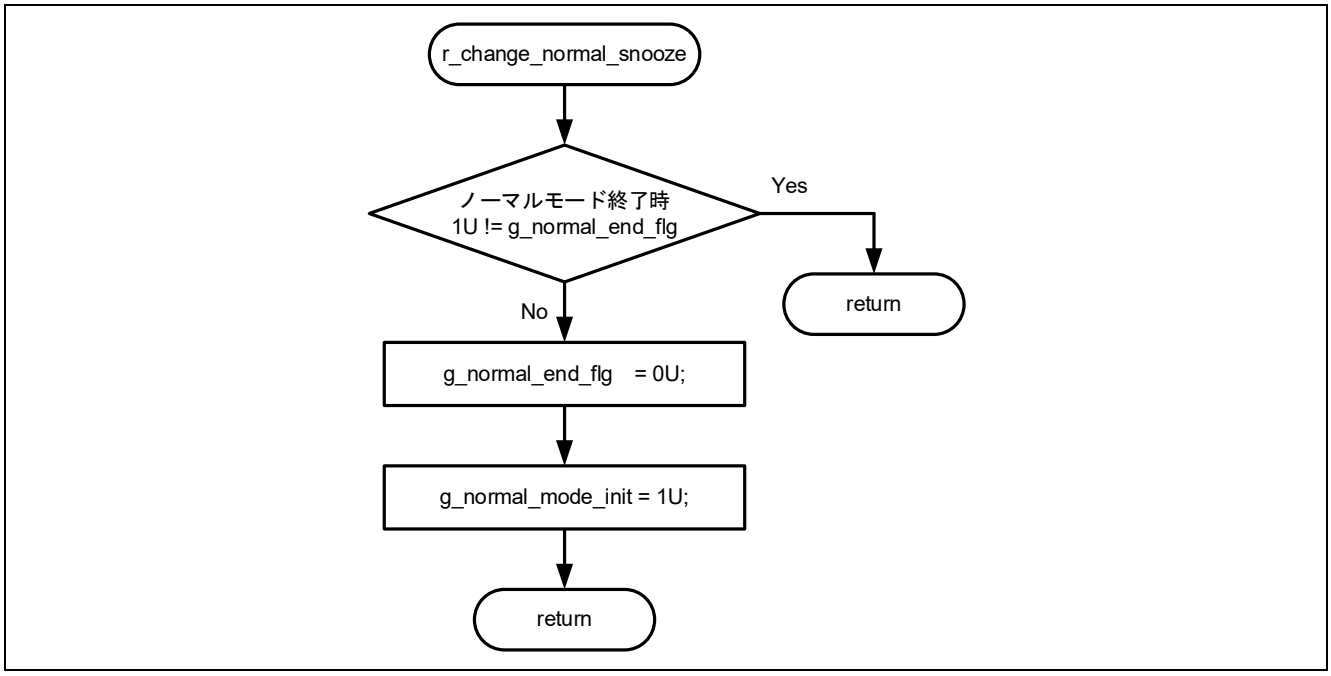

図 3-17 r_change_nomal_snooze 関数フローチャート

3.4.8.16 r_not_touched 関数のフローチャート

r_not_touched 関数のフローチャートを以下に示します。

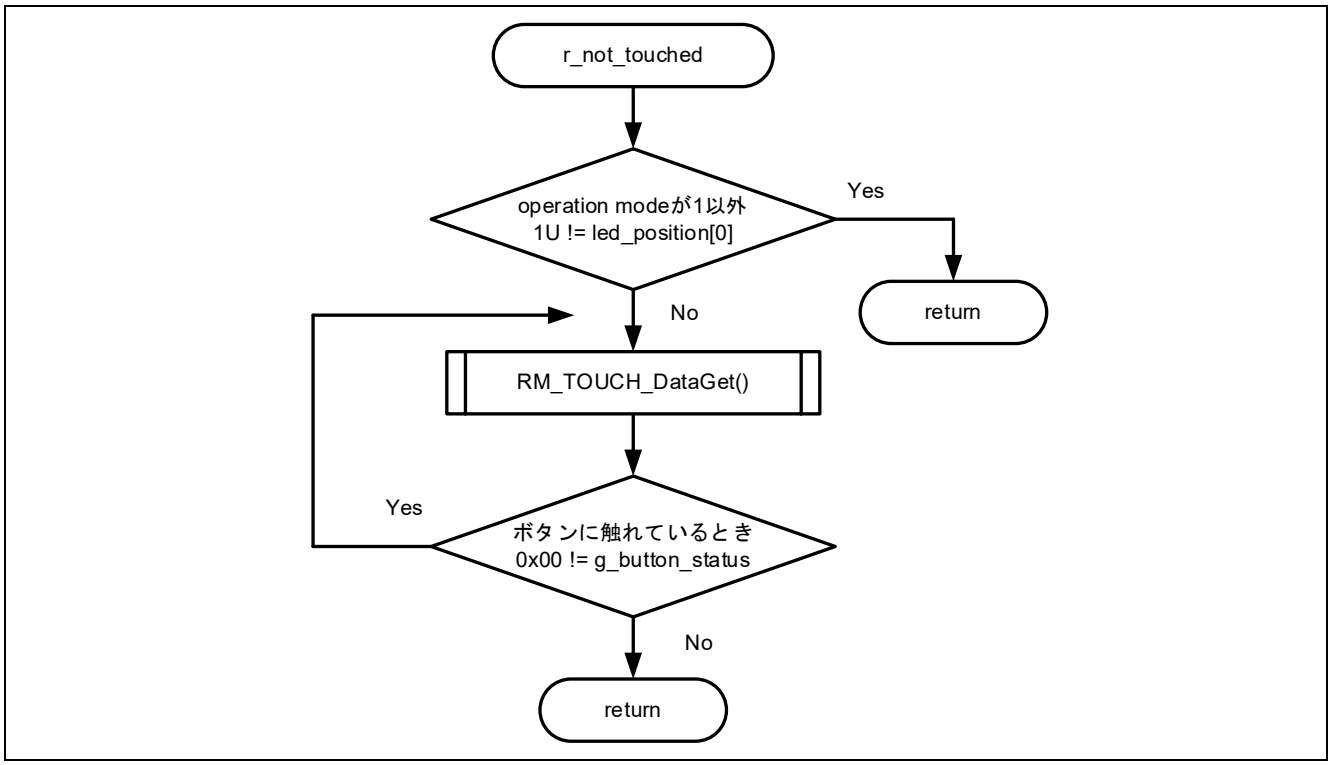

図 3-18 r_not_touched 関数フローチャート

3.4.8.17 r_ledport_input 関数のフローチャート

r_ledport_input 関数のフローチャートを以下に示します。

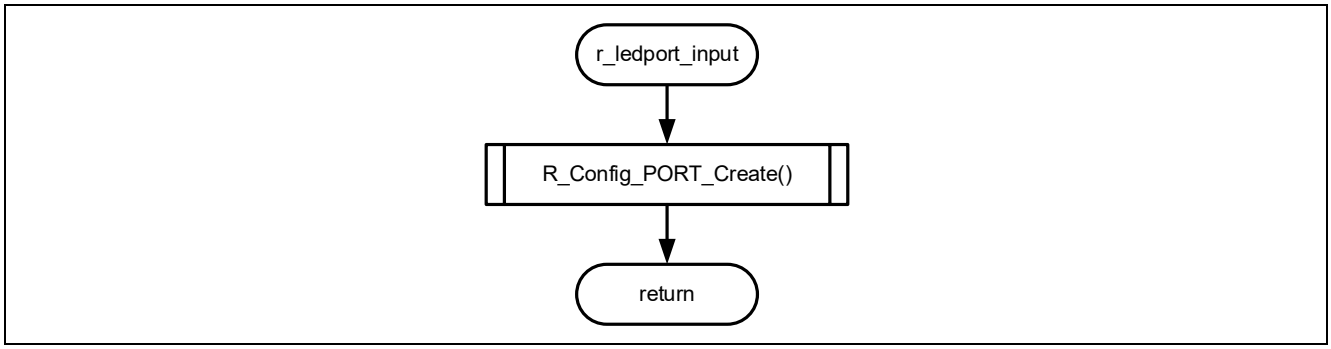

図 3-19 r_ledport_input 関数フローチャート

3.4.8.18 r_ledport_output 関数のフローチャート

r_ledport_output 関数のフローチャートを以下に示します。

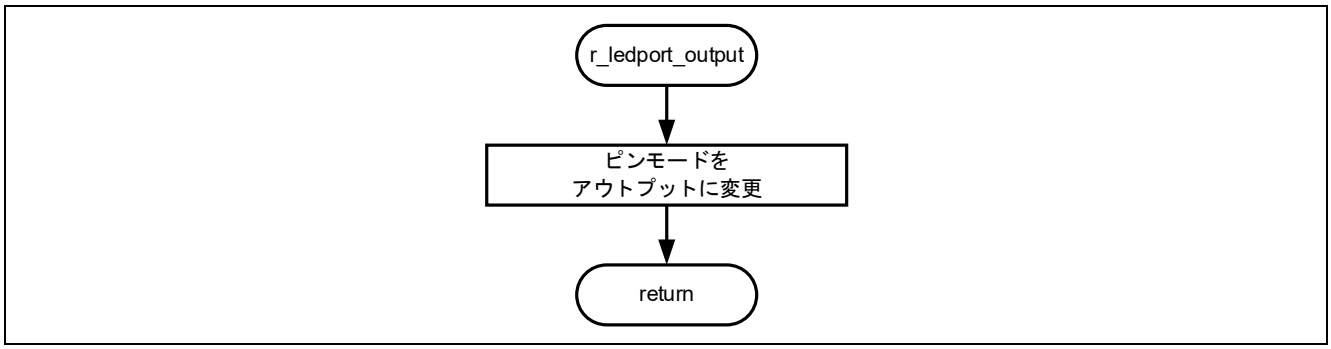

図 3-20 r_ledport_output 関数フローチャート

3.4.8.19 r_led_init 関数のフローチャート

r_led_init 関数のフローチャートを以下に示します。

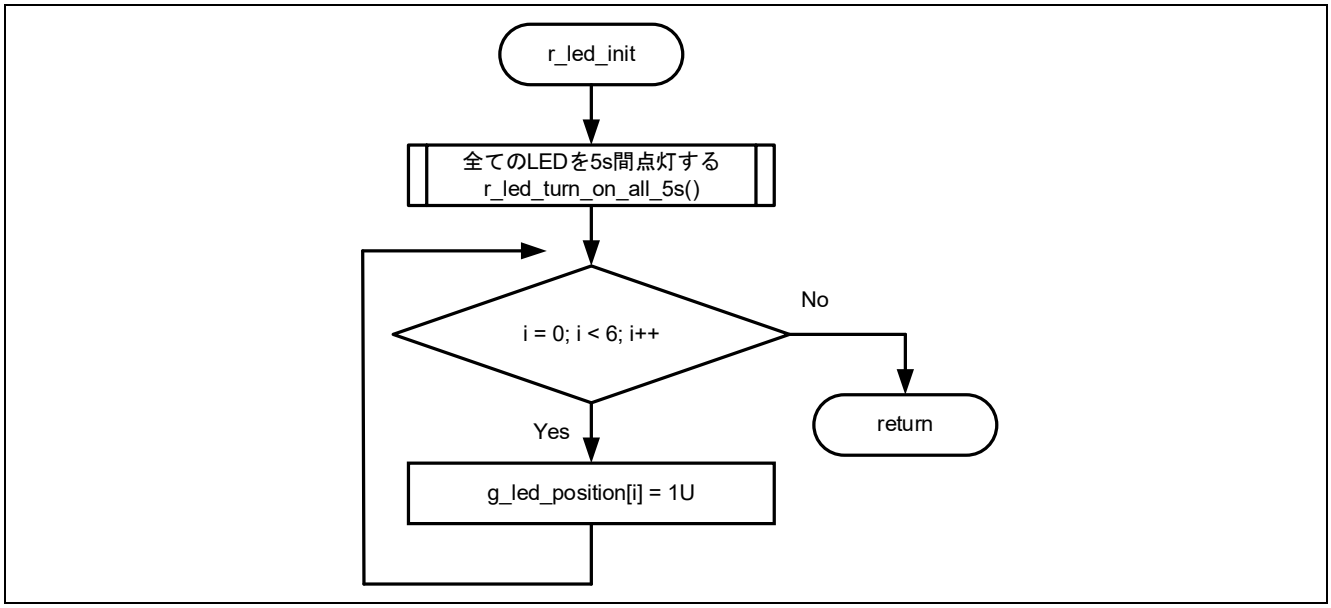

図 3-21 r_led_init 関数フローチャート

3.4.8.20 r_led_turn_on_all_5s 関数のフローチャート

r_led_turn_on_all_5s 関数のフローチャートを以下に示します。

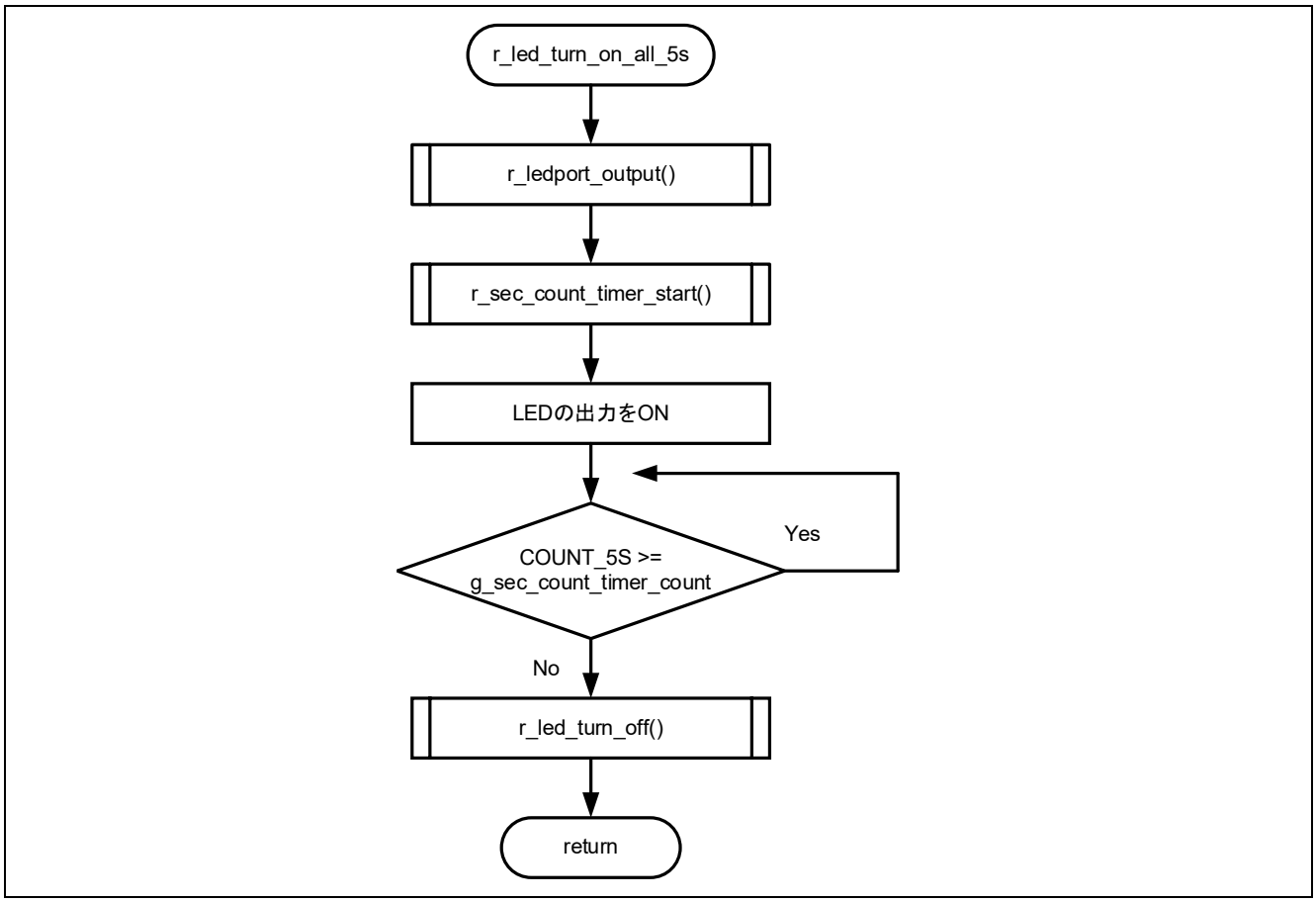

図 3-22 r_led_turn_on_all_5s 関数フローチャート

3.4.8.21 r_change_led 関数のフローチャート

r_change_led 関数のフローチャートを以下に示します。

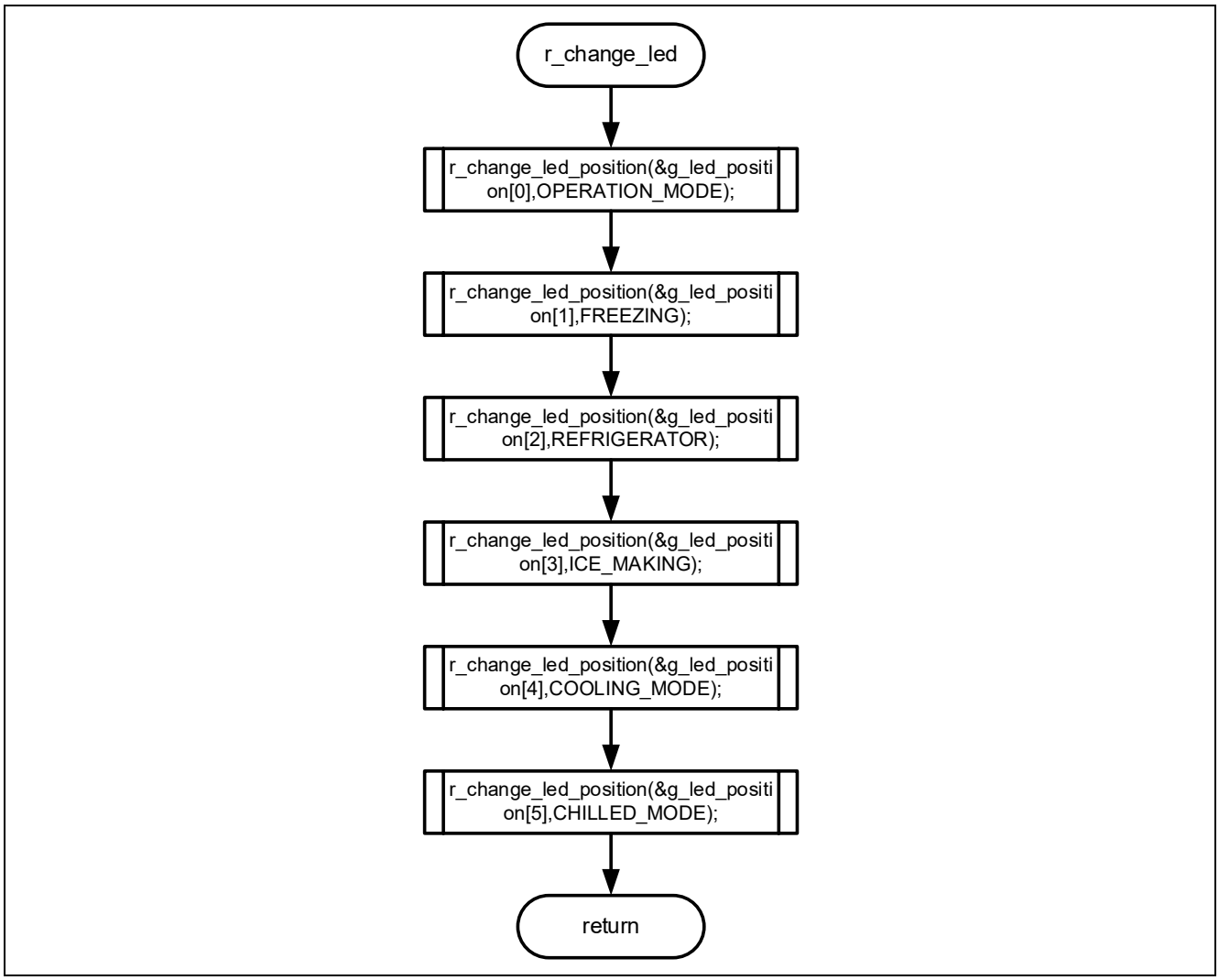

図 3-23 r_change_led 関数フローチャート

3.4.8.22 r_change_led_position 関数のフローチャート

r_change_led_position 関数のフローチャートを以下に示します。

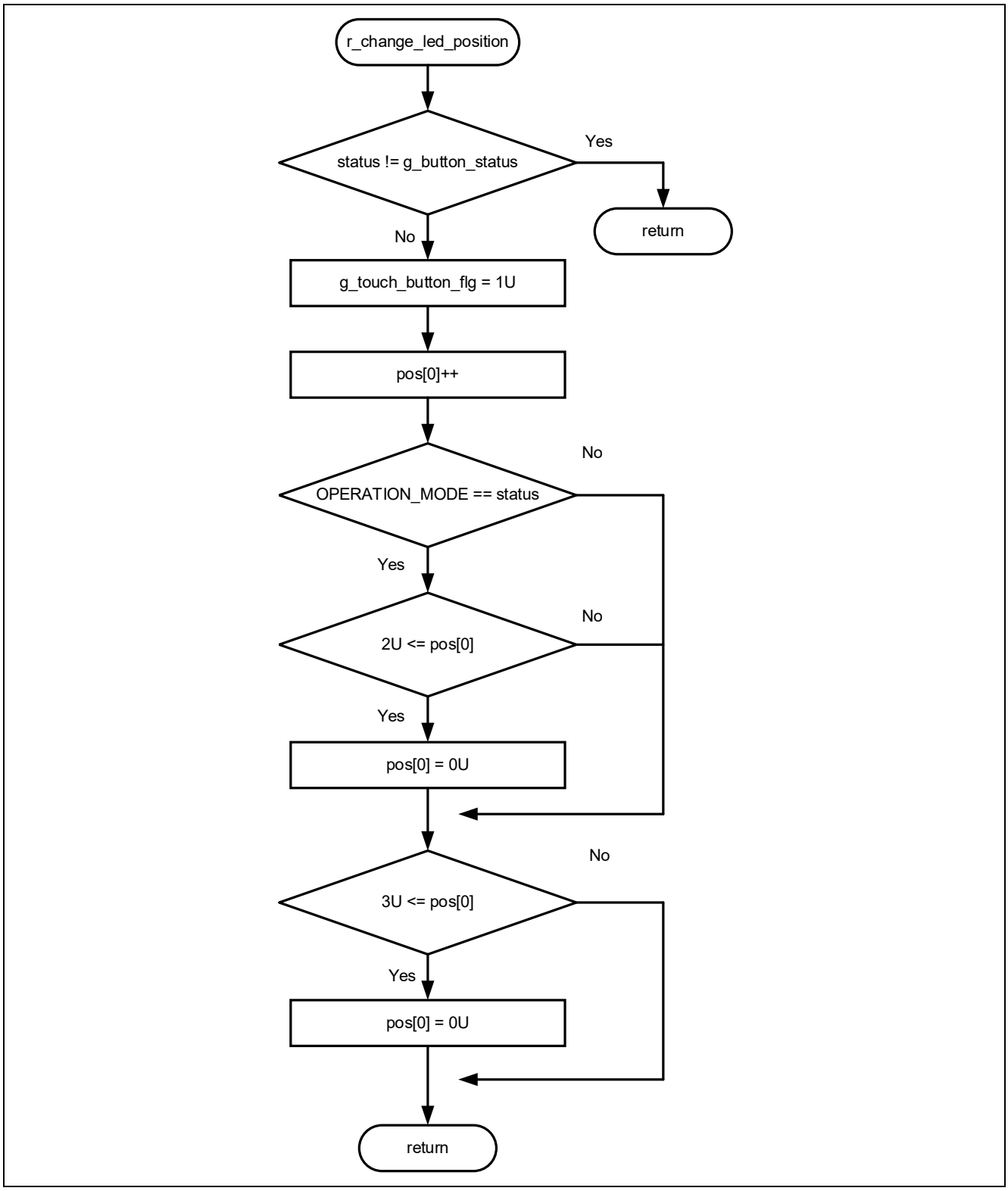

図 3-24 r_change_led_position 関数フローチャート

3.4.8.23 r_led_turn_on 関数のフローチャート

r_led_turn_on 関数のフローチャートを以下に示します。

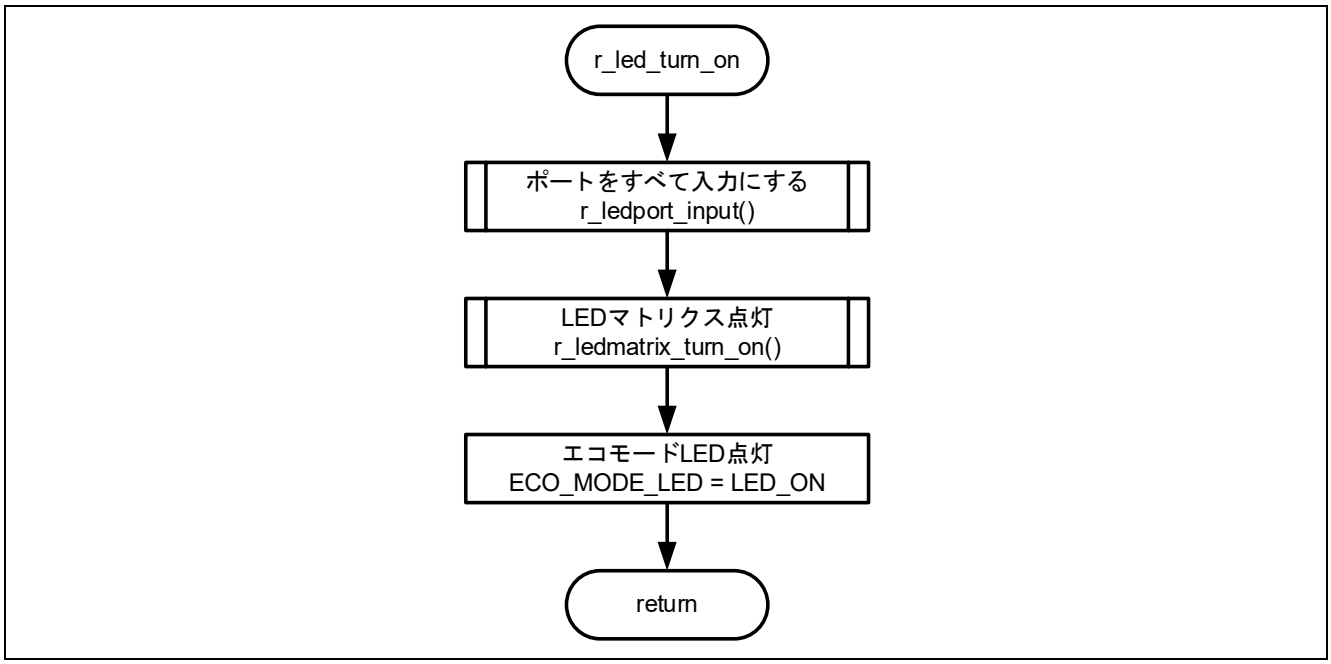

図 3-25 r_led_turn_on 関数フローチャート

3.4.8.24 r_led_turn_off 関数のフローチャート

r_led_turn_off 関数のフローチャートを以下に示します。

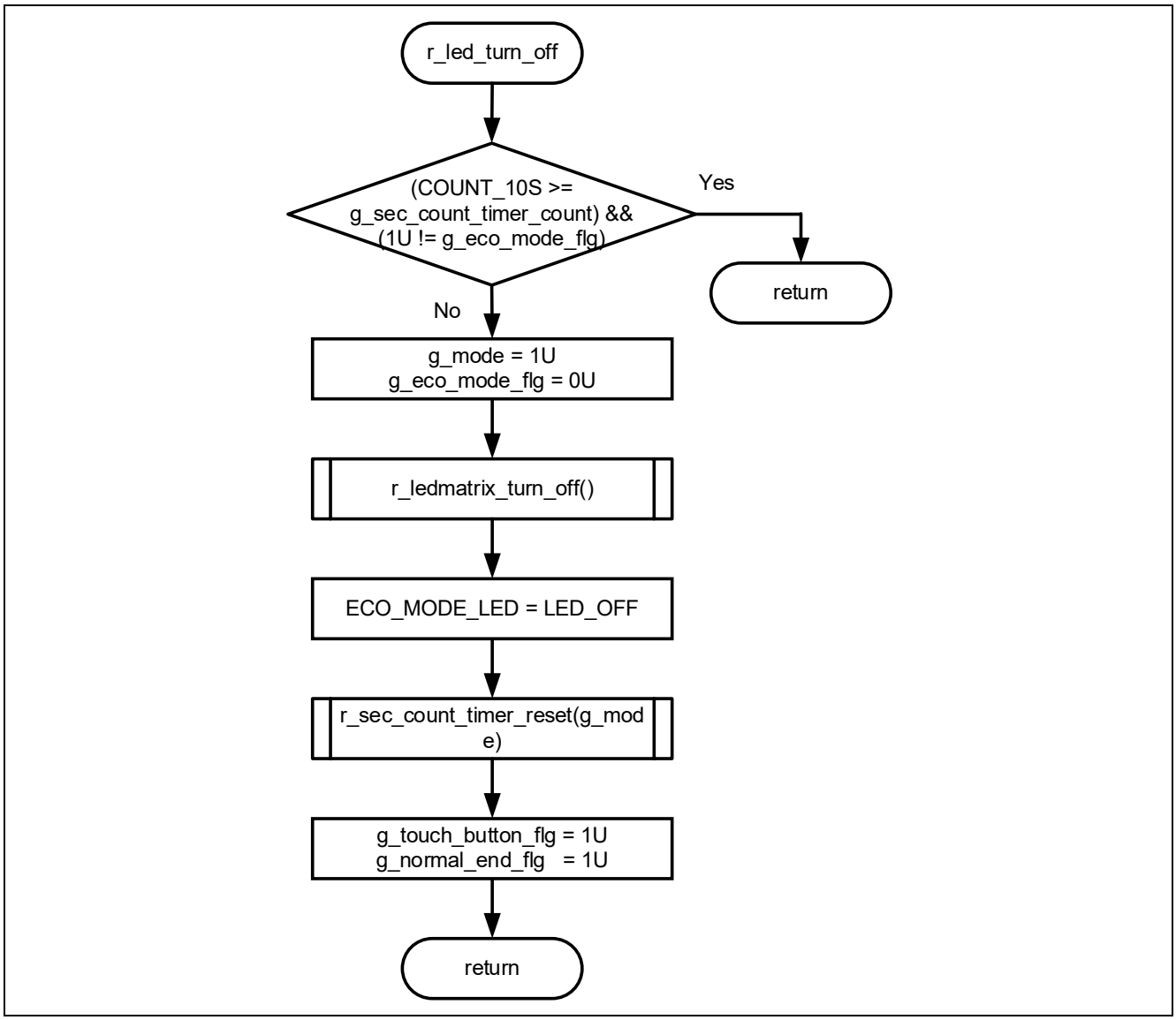

図 3-26 r_led_turn_off 関数フローチャート

3.4.8.25 r_ledmatrix_turn_on 関数のフローチャート

r_ledmatrix_turn_on 関数のフローチャートを以下に示します。

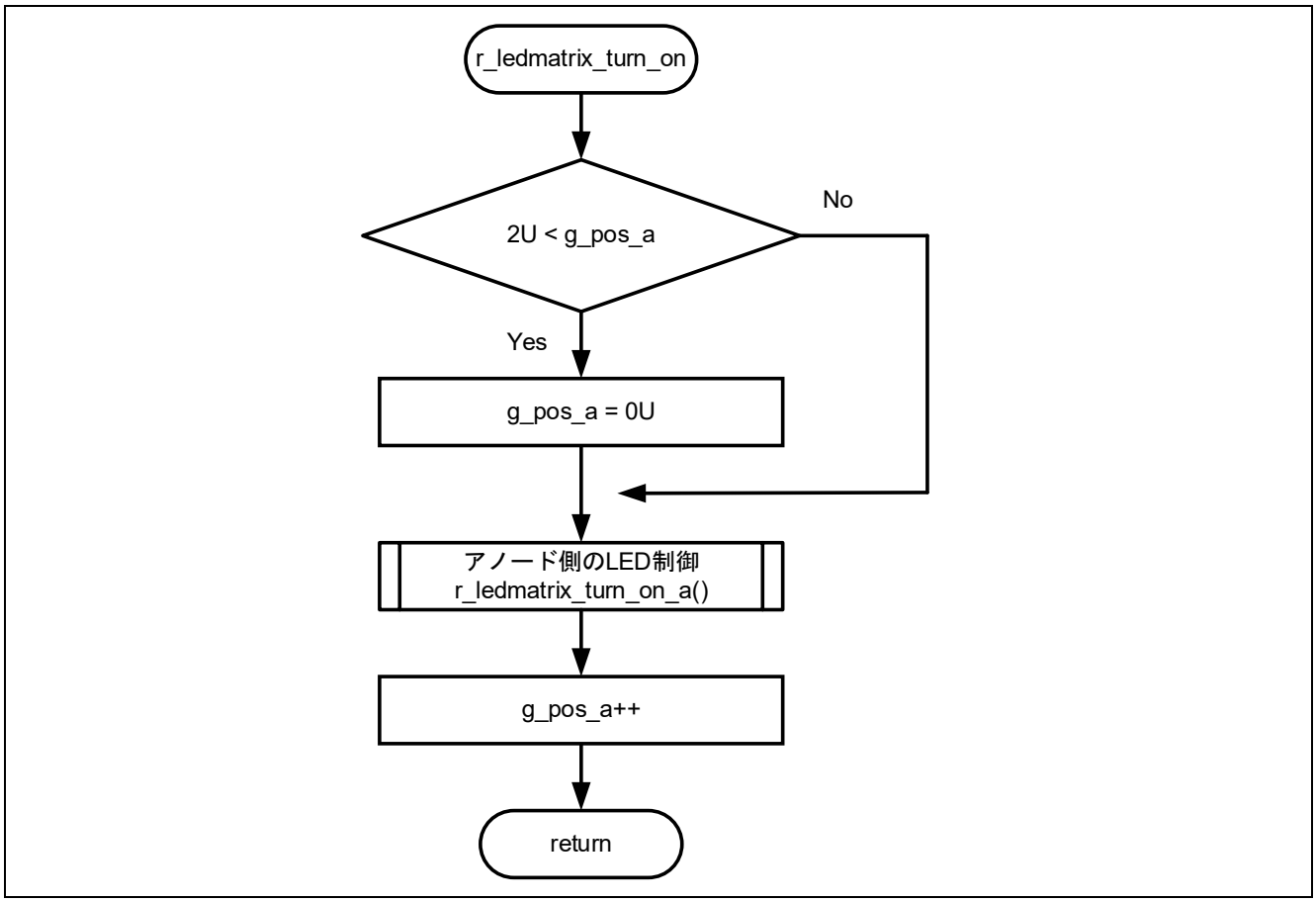

図 3-27 r_ledmatrix_turn_on 関数フローチャート

3.4.8.26 r_ledmatrix_turn_off 関数のフローチャート

r_ledmatrix_turn_off 関数のフローチャートを以下に示します。

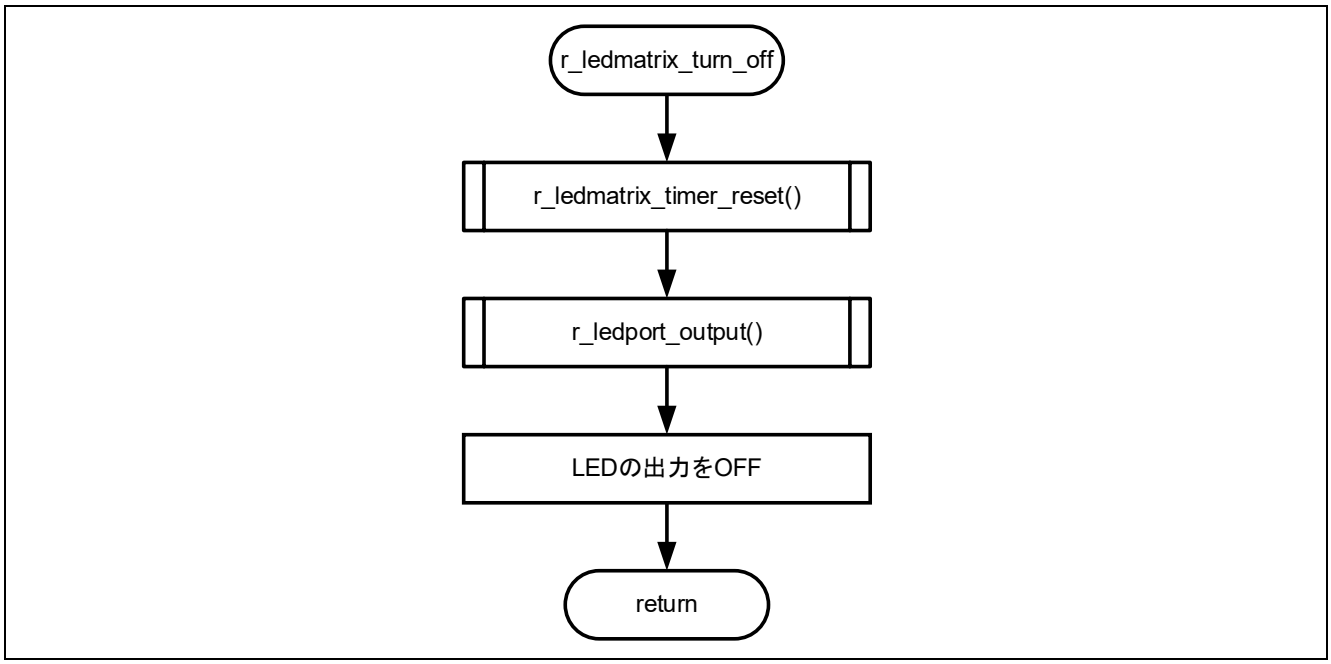

図 3-28 r_ledmatrix_turn_off 関数フローチャート

3.4.8.27 r_ledmatrix_turn_on_a 関数のフローチャート

r_ledmatrix_turn_on_a 関数のフローチャートを以下に示します。

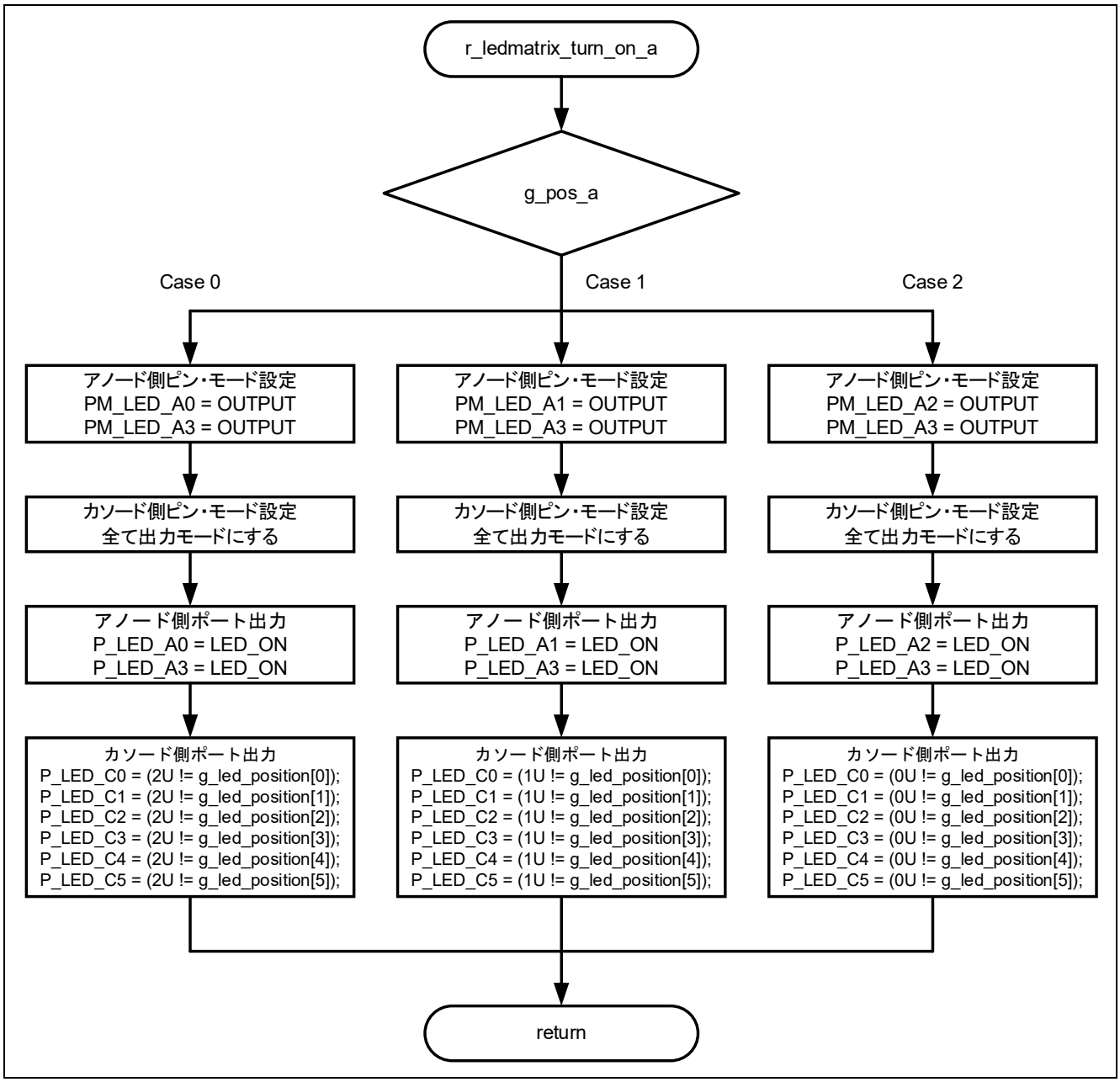

図 3-29 r_ledmatrix_turn_on_a 関数フローチャート

3.4.8.28 r_Config_TAU0_0_interrupt 関数のフローチャート

r_Config_TAU0_0_interrupt 関数のフローチャートを以下に示します。

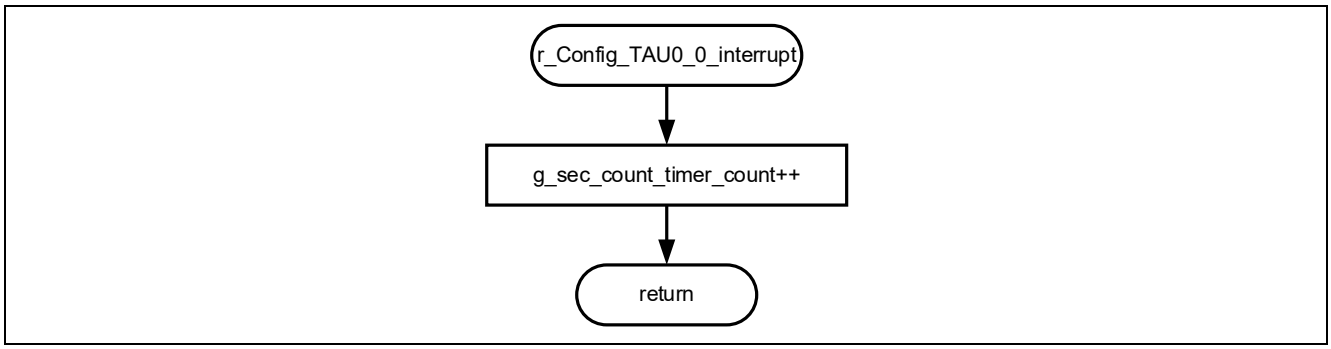

図 3-30 r_Config_TAU0_0_interrupt 関数フローチャート

3.4.8.29 r_sec_count_timer_start 関数のフローチャート

r_sec_count_timer_start 関数のフローチャートを以下に示します。

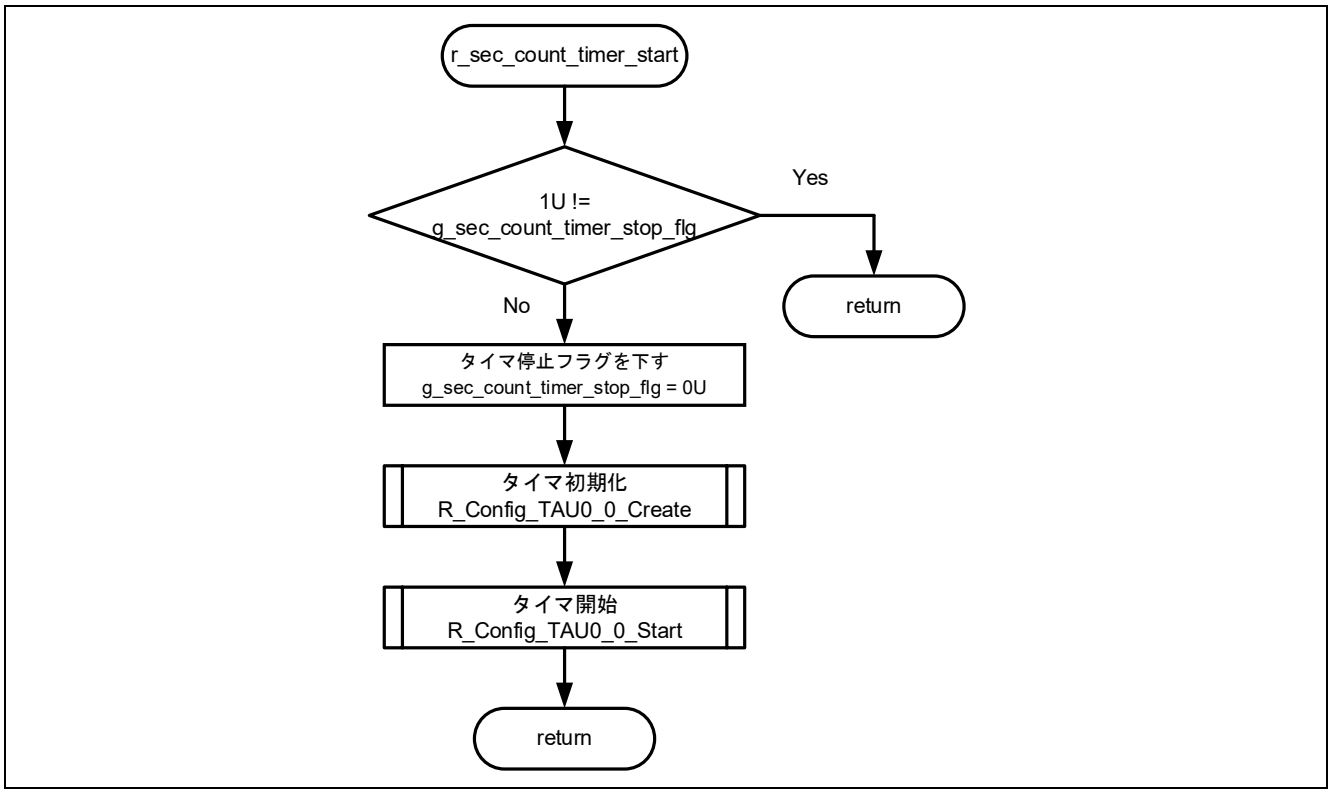

図 3-31 r_sec_count_timer_start 関数フローチャート

3.4.8.30 r_sec_count_timer_reset 関数のフローチャート

r_sec_count_timer_reset 関数のフローチャートを以下に示します。

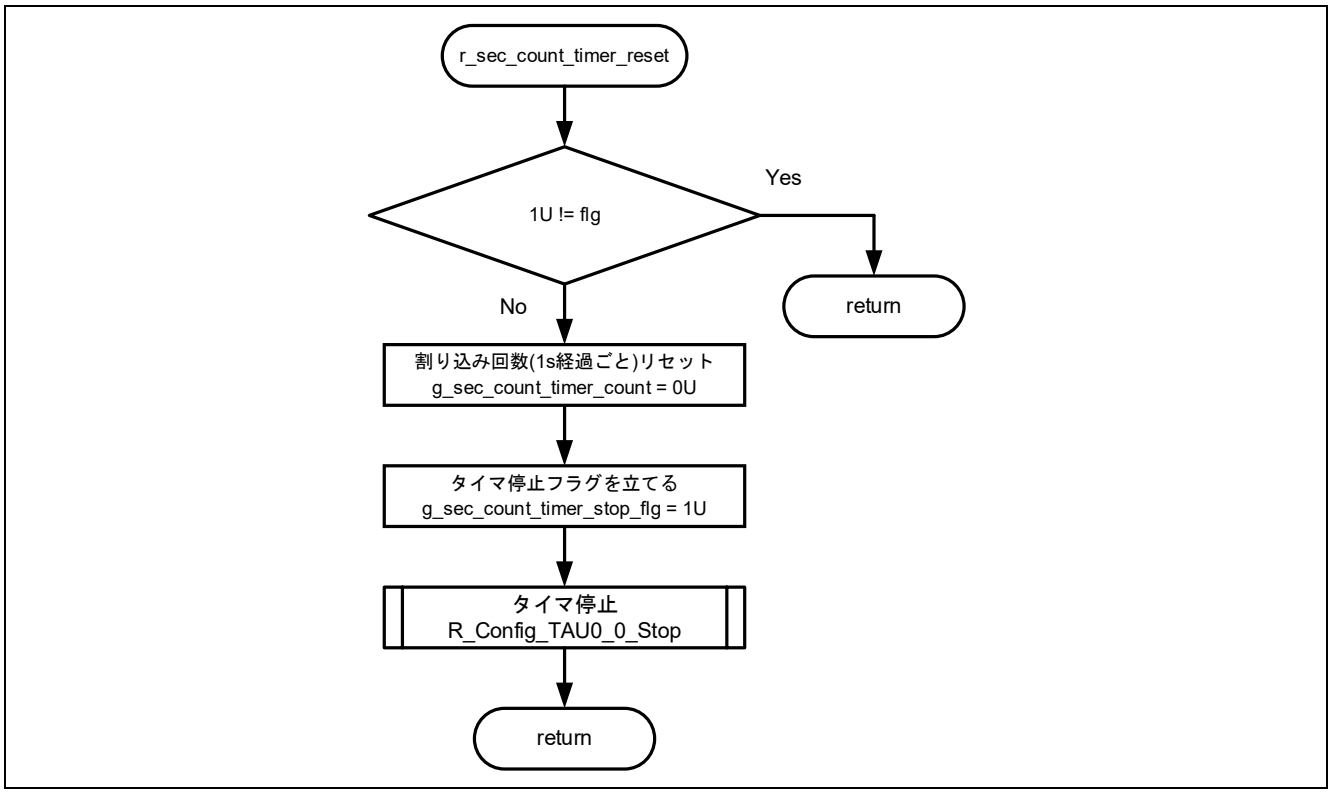

図 3-32 r_sec_count_timer_reset 関数フローチャート

3.4.8.31 r_Config_TAU0_1_interrupt 関数のフローチャート

r_Config_TAU0_1_interrupt 関数のフローチャートを以下に示します。

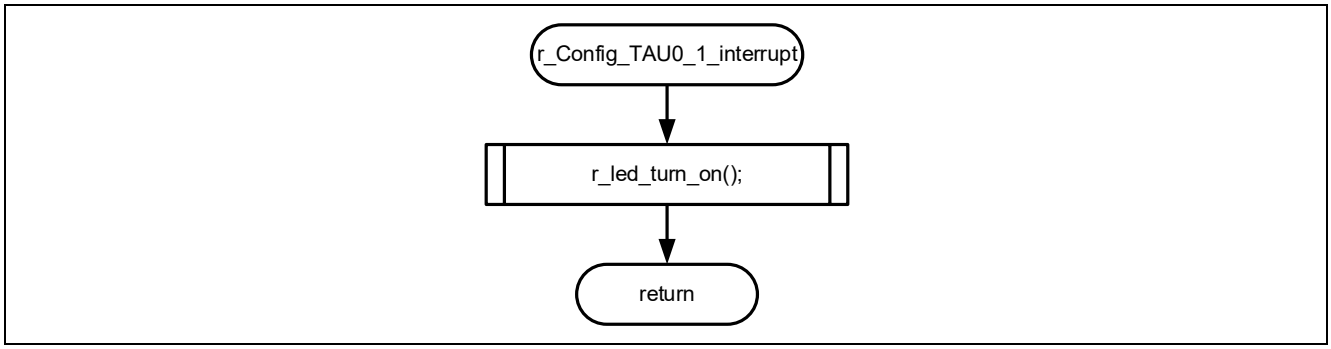

図 3-33 r_Config_TAU0_1_interrupt 関数フローチャート

3.4.8.32 r_ledmatrix_timer_start 関数のフローチャート

r_ledmatrix_timer_start 関数のフローチャートを以下に示します。

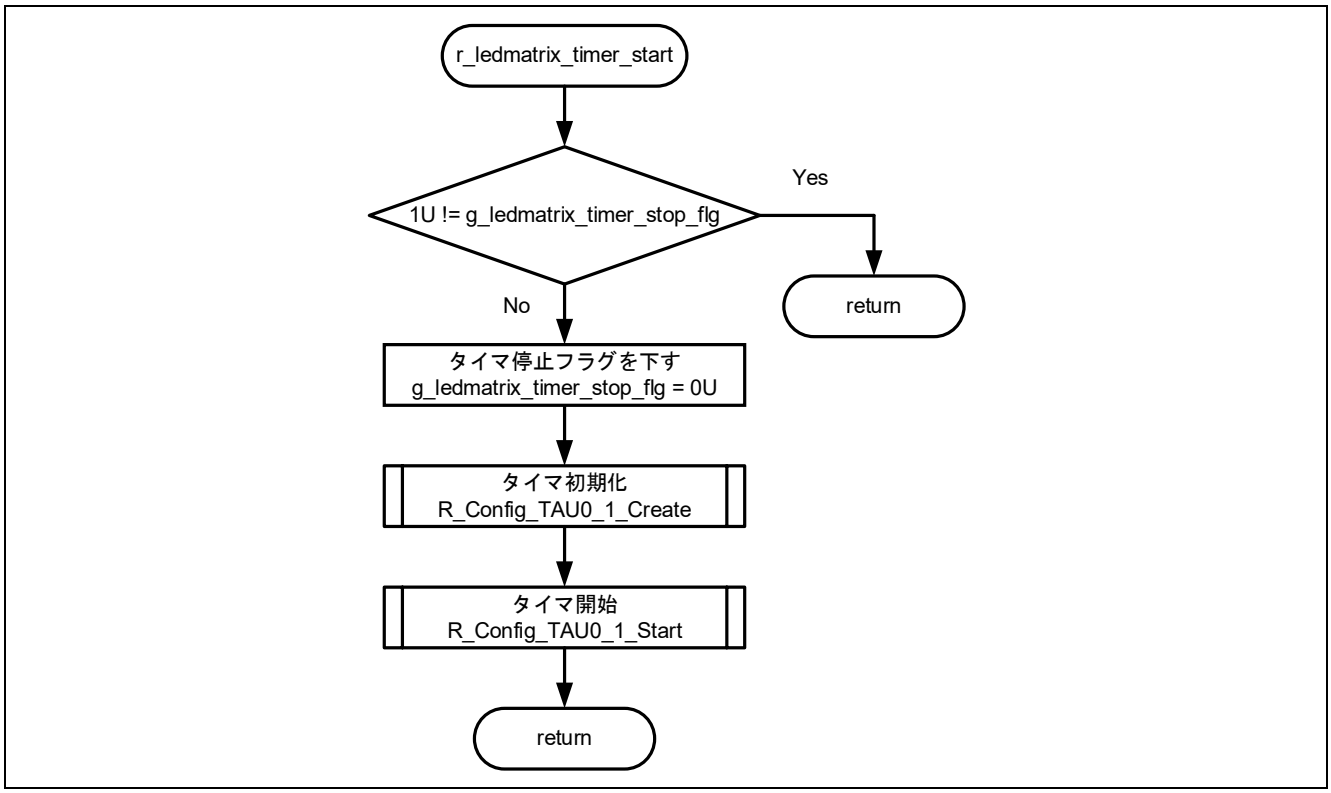

図 3-34 r_ledmatrix_timer_start 関数フローチャート

3.4.8.33 r_ledmatrix_timer_reset 関数のフローチャート

r_ledmatrix_timer_reset 関数のフローチャートを以下に示します。

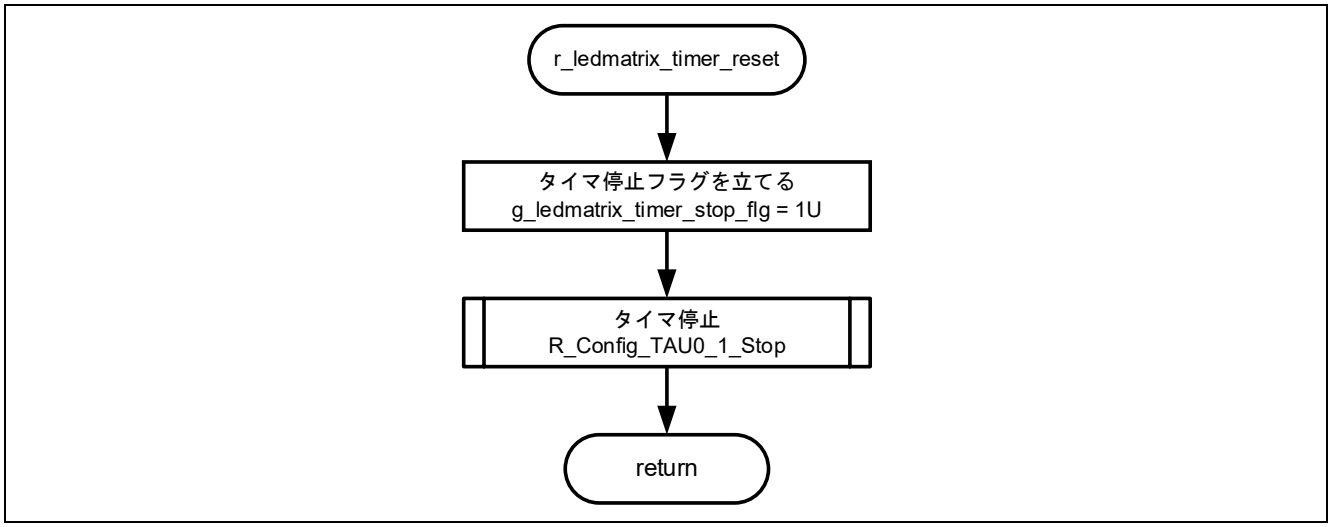

図 3-35 r_ledmatrix_timer_reset 関数フローチャート

4.　プロジェクトのインポート方法

サンプルプログラムは e² studio のプロジェクト形式で提供しています。本章では、e² studio および CS+ へプロジェクトをインポートする方法を示します。インポート完了後、ビルドおよびデバッガの設定を確認してください。

4.1　e² studio での手順

e² studio でご使用になる際は、以下の手順で e² studio にインポートしてください。

なお、e² studio で管理するプロジェクトのフォルダ名、およびそのフォルダに至るファイルパスには、空白文字の他、半角カナ文字、全角文字、半角記号 (特に'$','#','%') が混じらないようにしてください。

（使用する e² studio のバージョンによっては画面が異なる場合があります。）

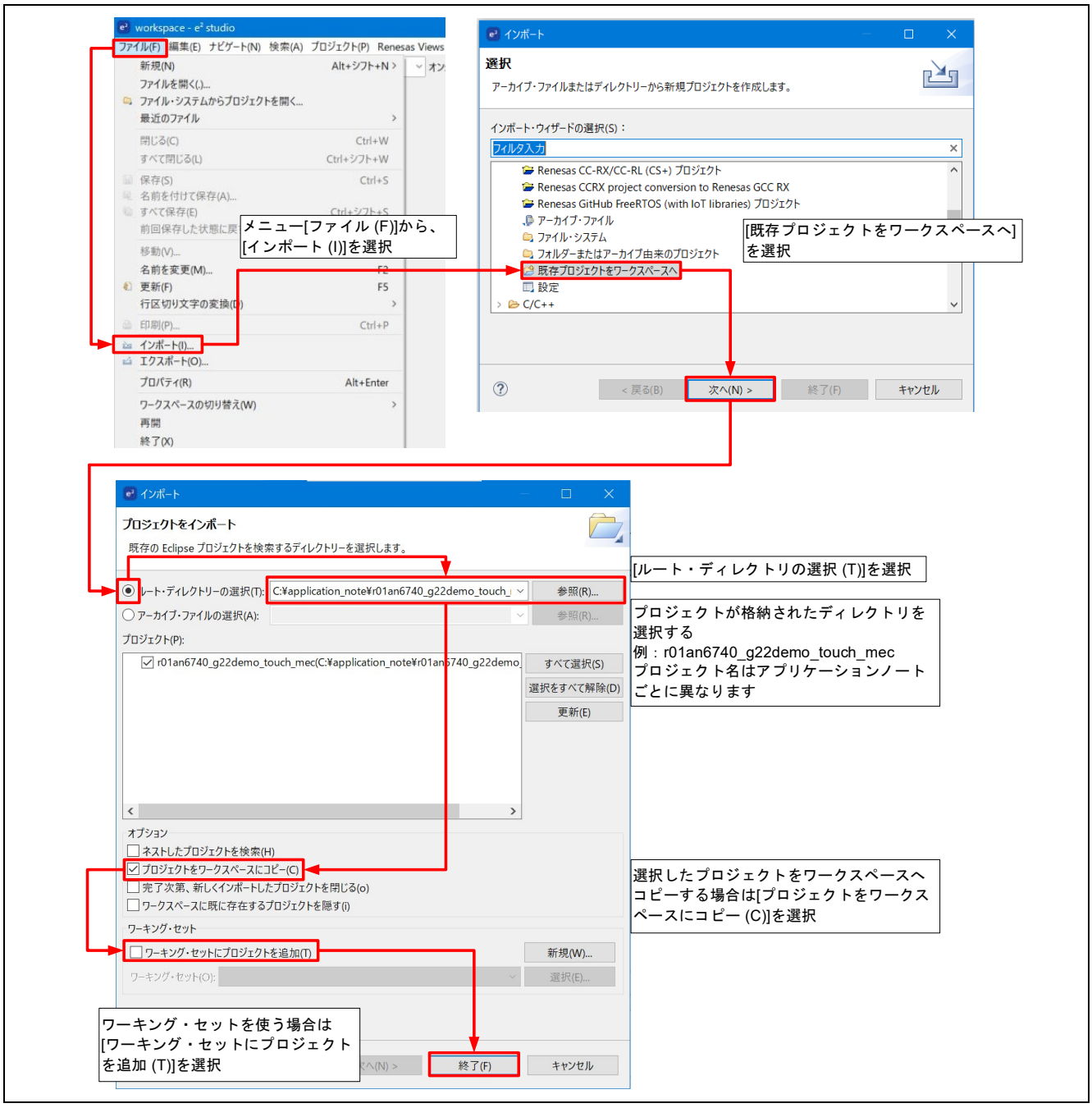

図 4-1　プロジェクトを e² studio にインポートする方法

4.2　CS+ での手順

CS+ でご使用になる際は、以下の手順で CS+ にインポートしてください。

なお、CS+で管理するプロジェクトのフォルダ名、およびそのフォルダに至るファイルパスには、 空白文字の他、半角カナ文字、全角文字、半角記号 (特に'$','#','%') が混じらないようにしてください。

（使用する CS+ のバージョンによっては画面が異なる場合があります。）

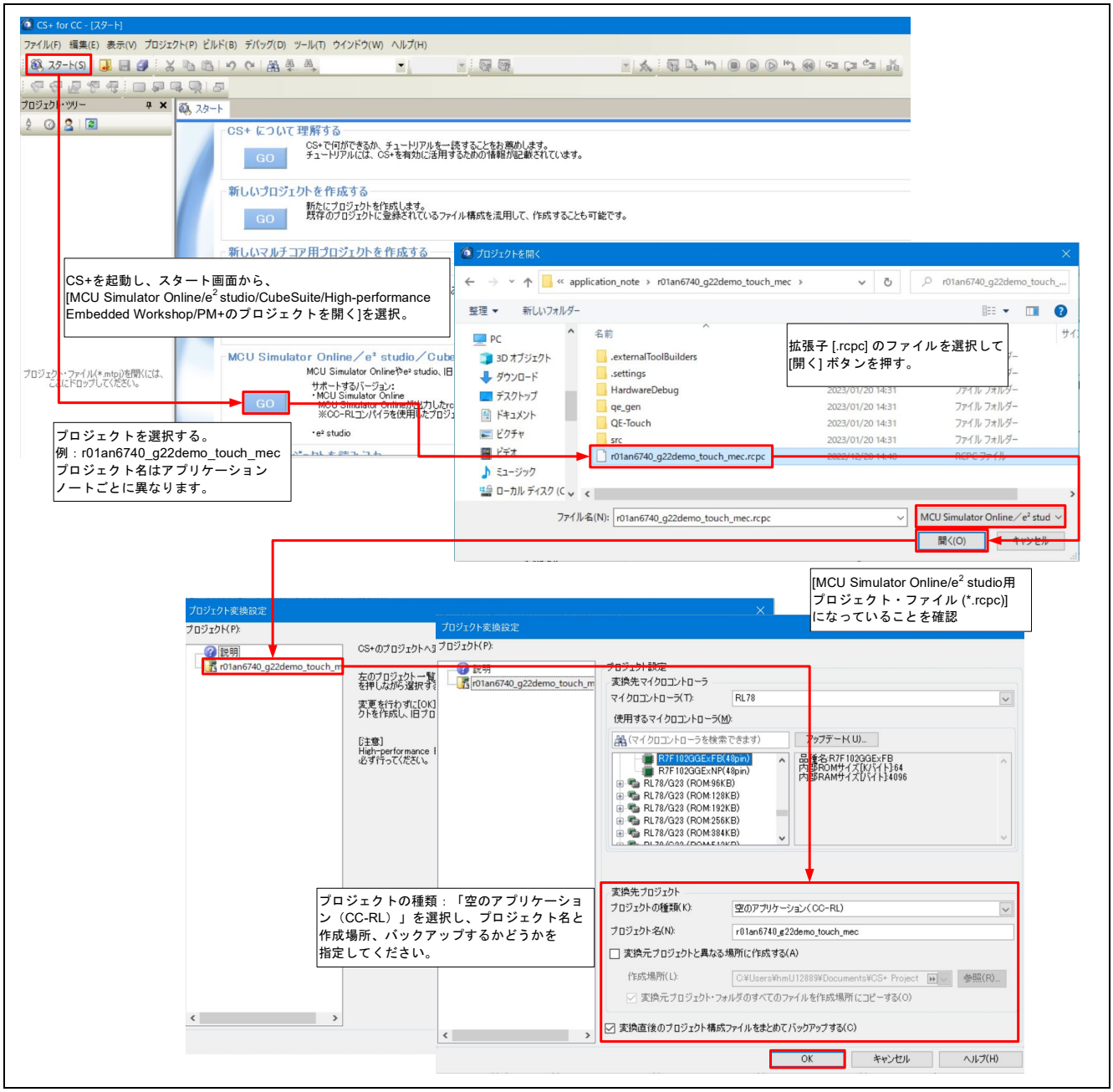

図 4-2　プロジェクトを CS+ にインポートする方法

5. デモの起動

E2 エミュレータ Lite コネクタを外して RL78/G22 PoC の電源を投入すると、デモプログラムがスタートします。本デモプログラムは、冷蔵庫パネルの表示と設定の制御を想定しています。冷蔵庫パネルの表示設定を、設定表示部で確認しながら、タッチボタンで設定します。

以降はタッチボタンをボタンと記載します。

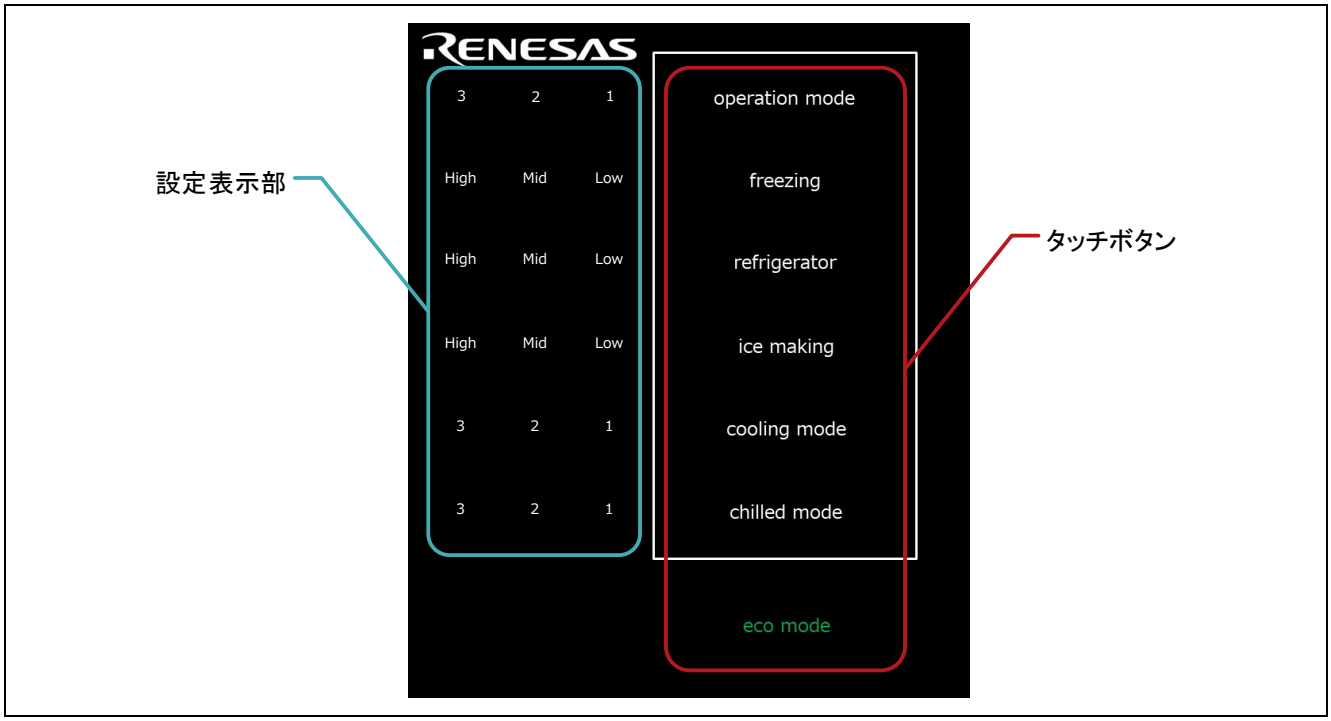

図 5-1 デモ操作パネル

5.1　RL78/G22 PoC の電源オン～メニュー画面

RL78/G22 PoC の電源を入れると、タッチパネルの全ての文字を約 5 秒間表示します。表示が終わるとデモプログラムがスタートし、RL78/G22 PoC は、スタンバイ・モード（operation mode1）に遷移します。

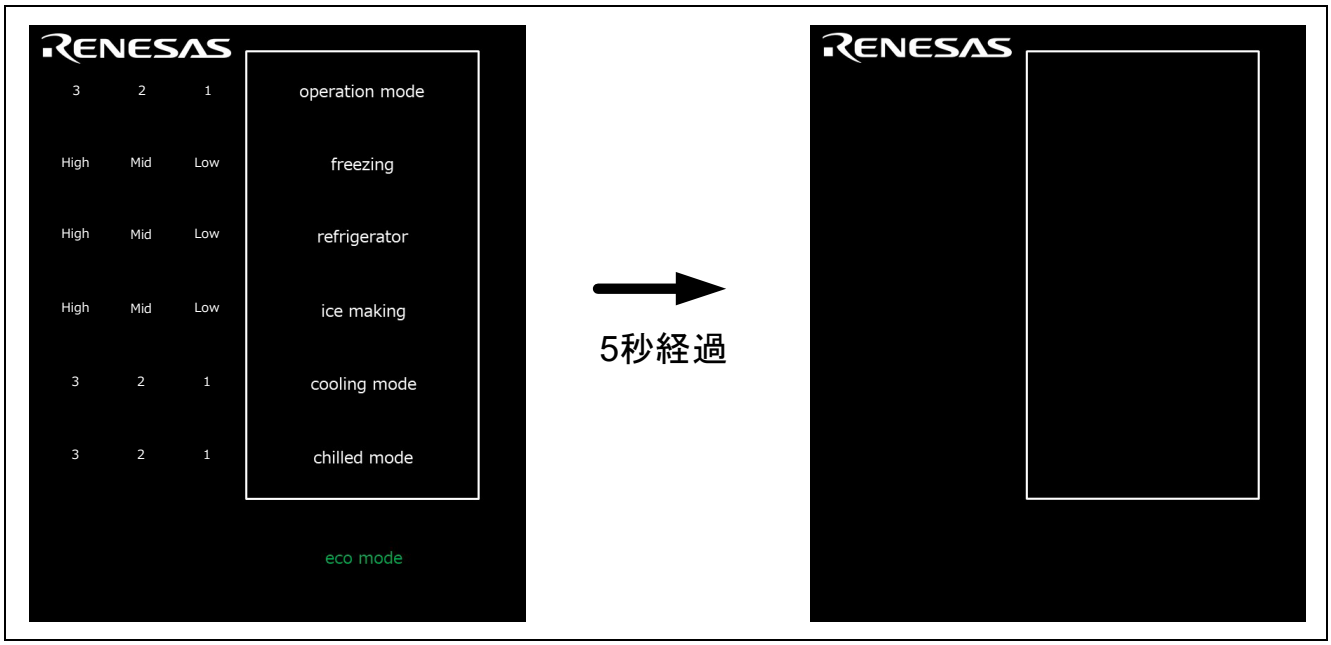

図 5-2 デモ開始時

5.2　スタンバイ・モードからの復帰

白枠内をタッチすると、スタンバイ・モードから復帰します。各設定値は、2 や Mid など、センター値を示します。

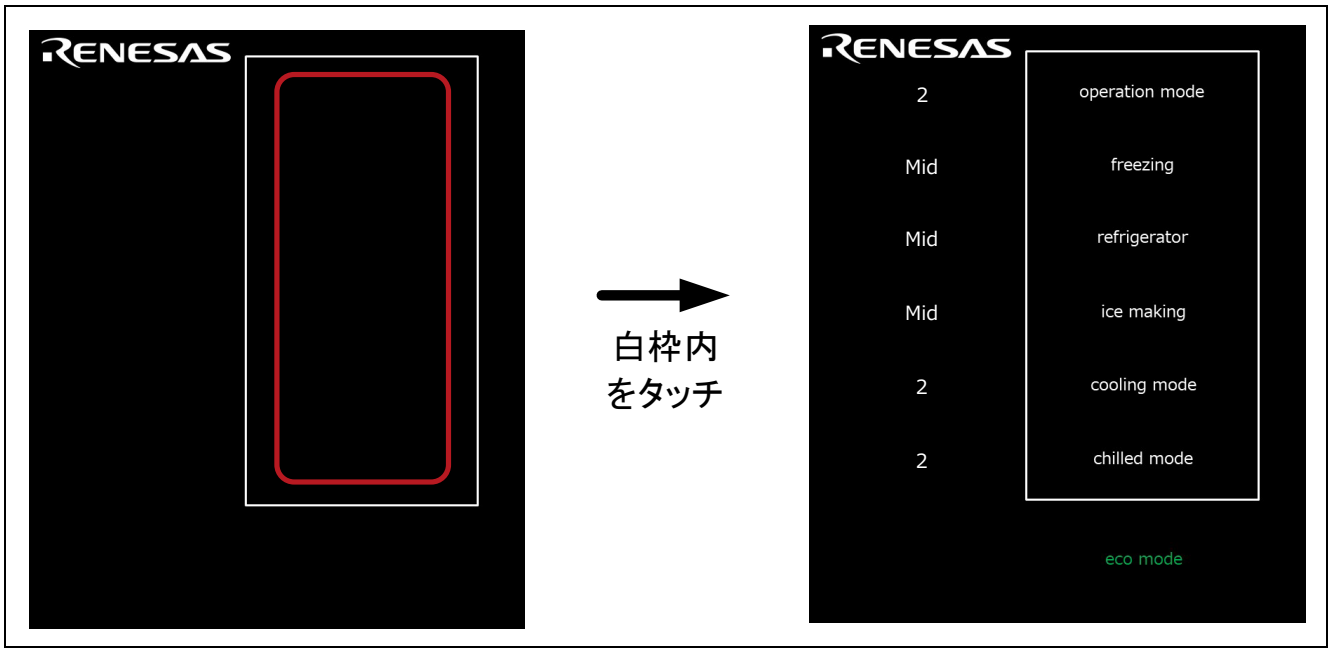

図 5-3 メニュー画面の操作方法

5.3　タッチ操作

5.3.1　operation mode を設定する

operation mode ボタンをタッチすると、図 5-4 の順に設定値を変更できます。

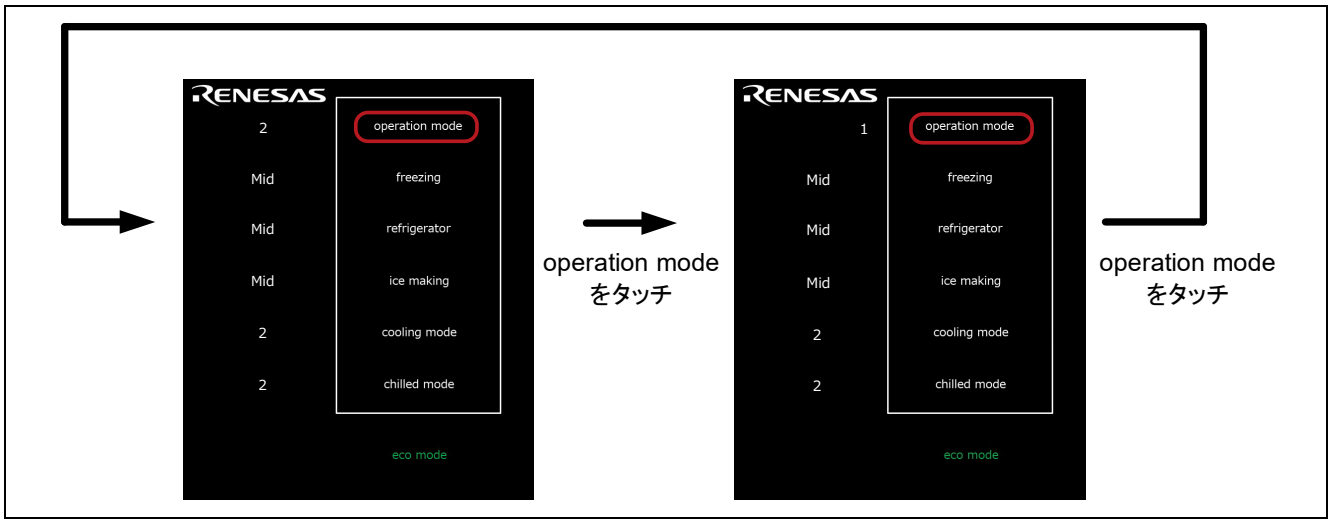

図 5-4 operation mode の設定

5.3.2　freezing を設定する

freezing ボタンをタッチすると、図 5-5 の順に設定値を変更できます。

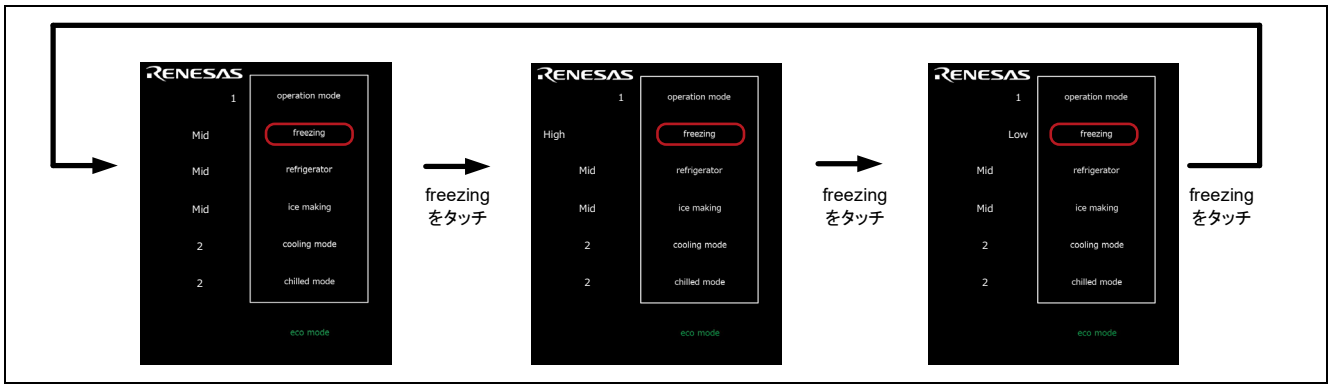

図 5-5 freezing の設定

5.3.3　refrigerator を設定する

refrigerator ボタンをタッチすると、図 5-6 の順に設定値を変更できます。

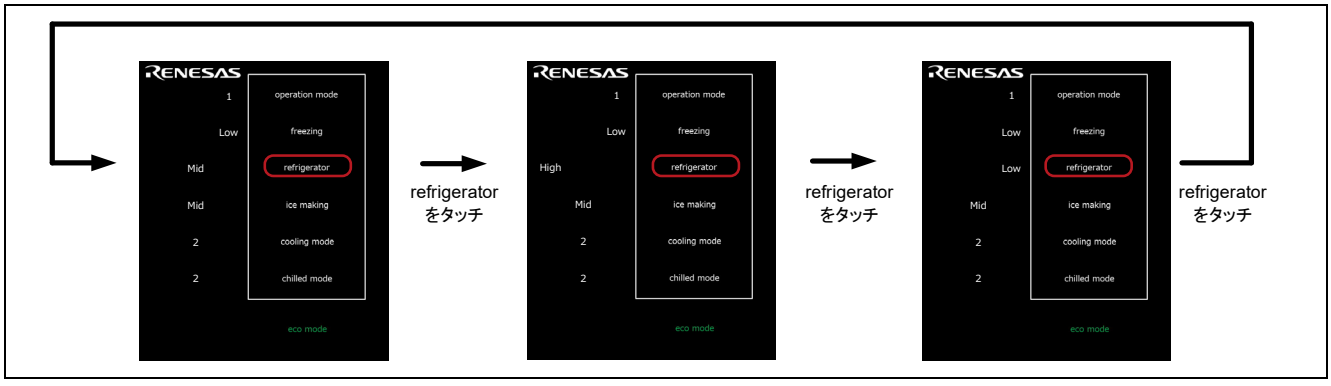

図 5-6 refrigerator の設定

5.3.4　ice making を設定する

ice making ボタンをタッチすると、図 5-7 の順に設定値を変更できます。

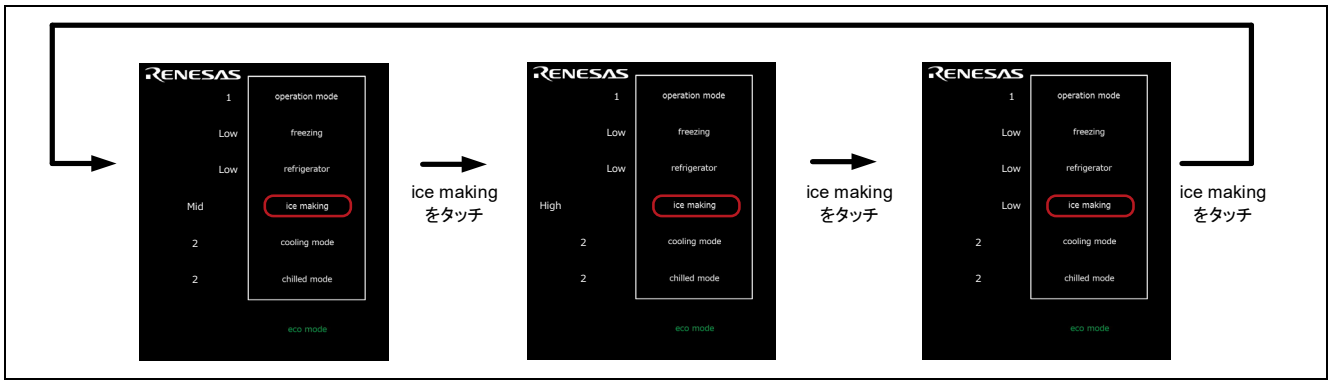

図 5-7 ice making の設定

5.3.5　cooling mode を設定する

cooling mode ボタンをタッチすると、図 5-8 の順に設定値を変更できます。

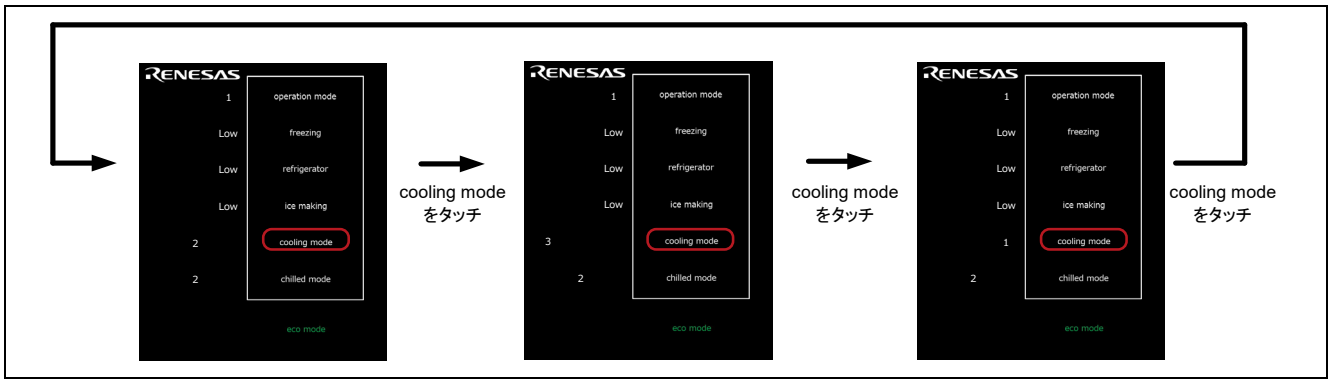

図 5-8 cooling mode の設定

5.3.6　chilled mode を設定する

chilled mode ボタンをタッチすると、図 5-9 の順に設定値を変更できます。

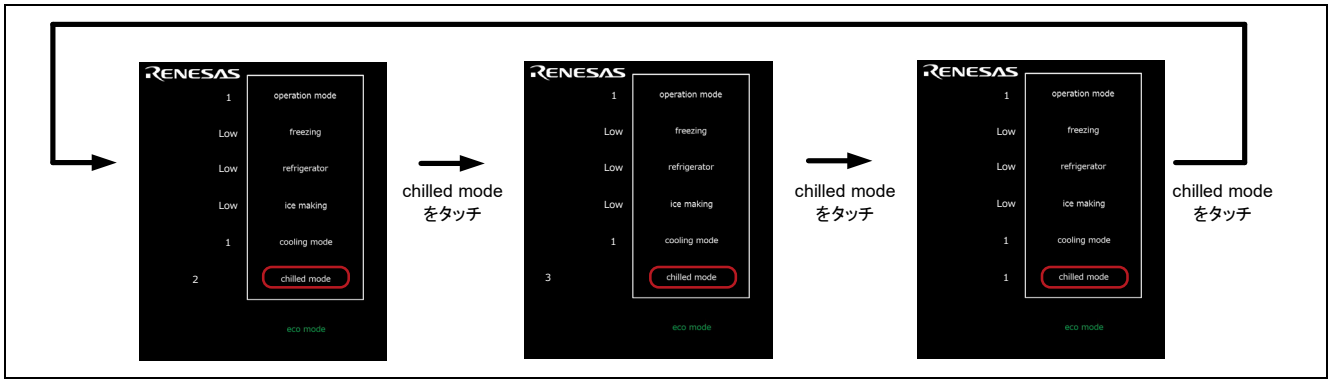

図 5-9 chilled mode の設定

5.3.7　eco mode（近接センサモード）

operation mode が 1 の時に eco mode ボタンをタッチすると、近接センサモードのスタンバイ・モードに遷移します。近接センサモードでは、白枠内に手をかざすと、通常モードに戻ります。

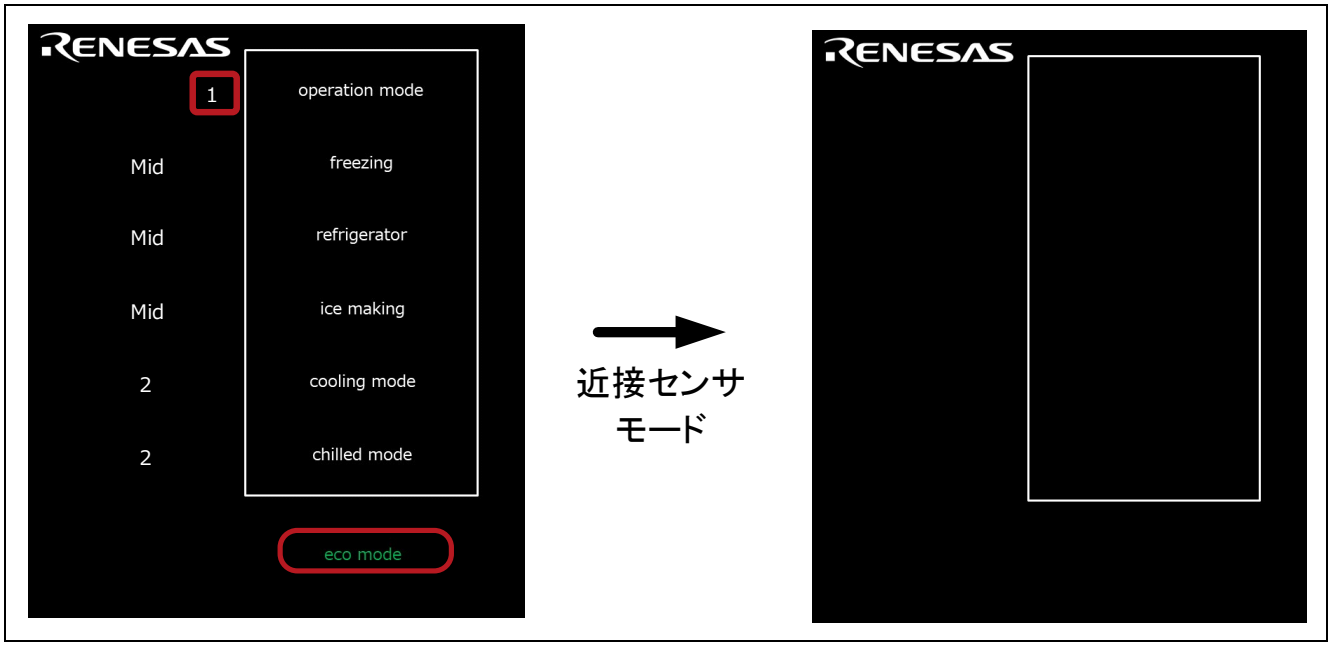

図 5-10 operation mode 1 の時に、eco mode をタッチ

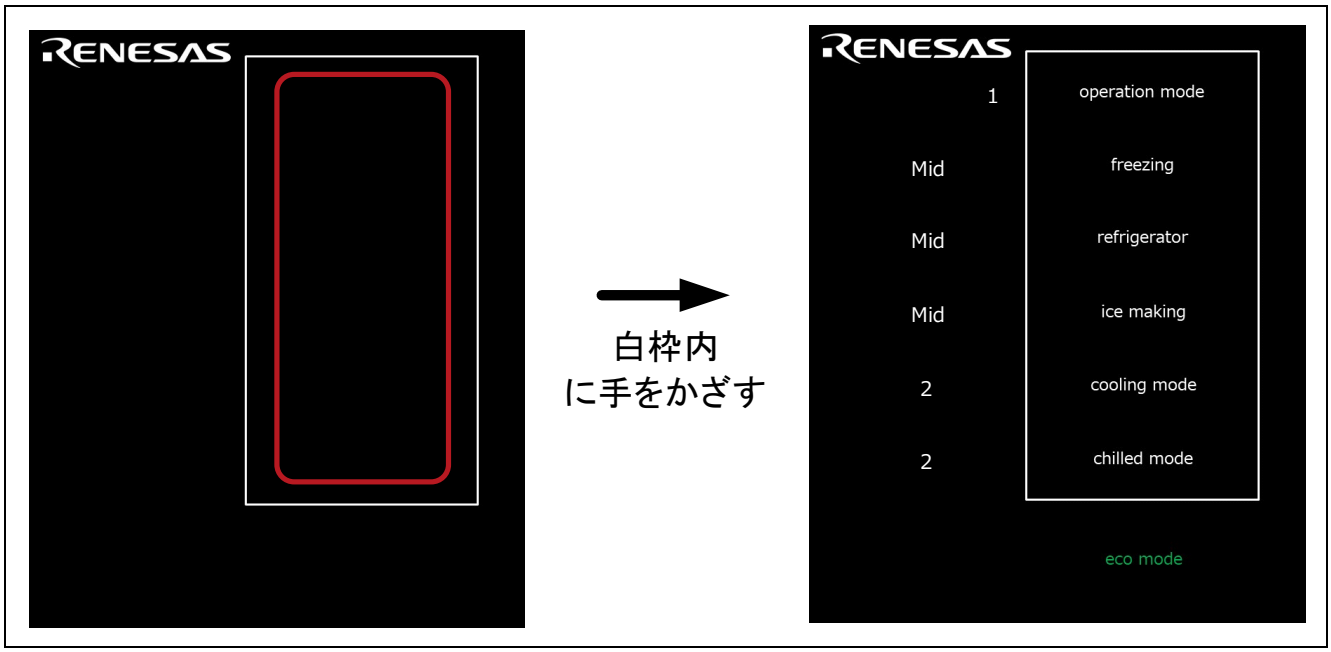

図 5-11 近接センサモードのスタンバイ・モードからの復帰

5.3.8　eco mode（タッチセンサモード）

operation mode が 2 の時に eco mode ボタンをタッチすると、タッチセンサモードのスタンバイ・モードに遷移します。タッチセンサモードでは、白枠内をタッチすると、通常モードに戻ります。

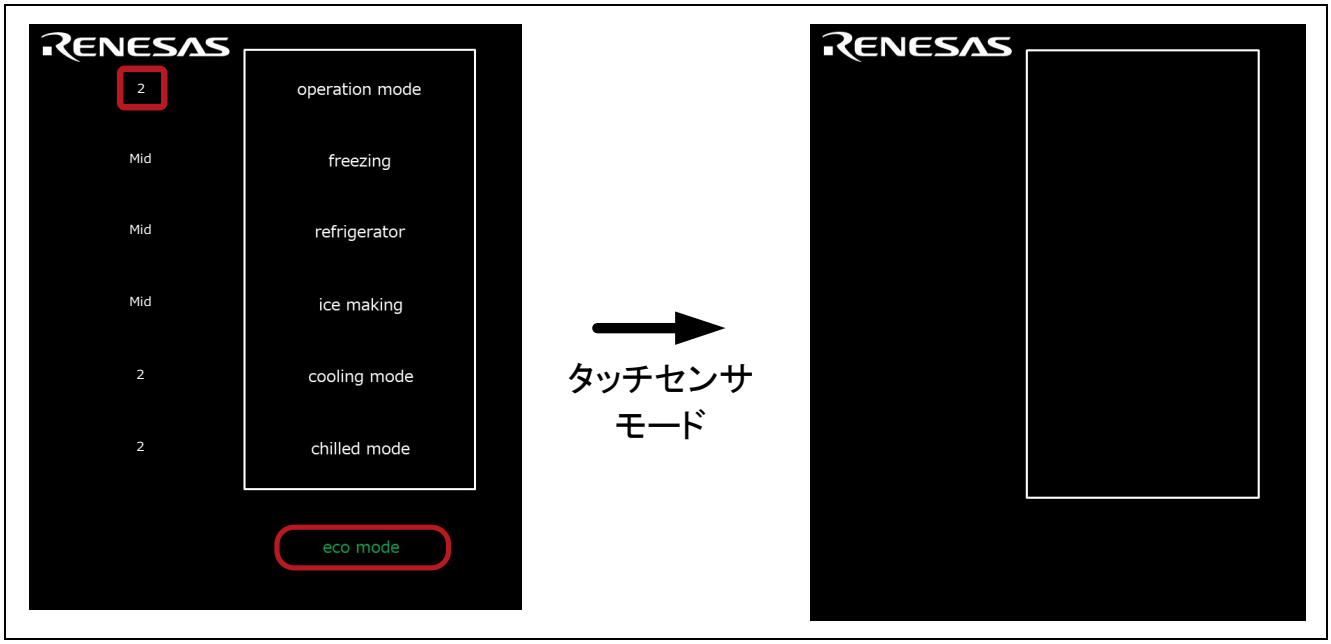

図 5-12 operation mode 2 の時に、eco mode をタッチ

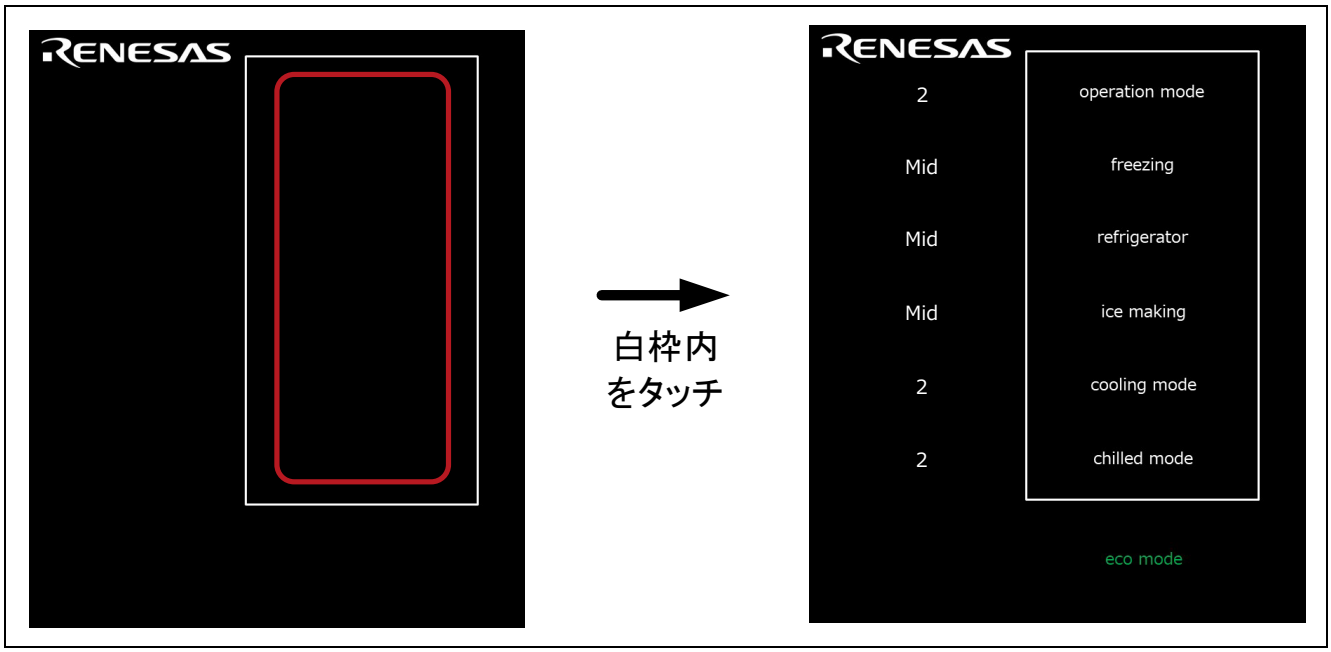

図 5-13 タッチセンサモードのスタンバイ・モードからの復帰

6. 参考資料

- RL78/G22 ユーザーズマニュアル ハードウェア編 (R01UH0978)
- RL78 ファミリ ユーザーズマニュアル ソフトウェア編 (R01US0015)
  (最新版をルネサス エレクトロニクスホームページから入手してください。)
- テクニカルアップデート／テクニカルニュース
  (最新の情報をルネサス エレクトロニクスホームページから入手してください。)
- ユーザーズマニュアル：開発環境
  (最新版をルネサス エレクトロニクスホームページから入手してください。)
- ユーザーズマニュアル：RL78/G22 静電容量タッチ評価システム（RTK0EG0042S01001BJ)
  (最新版をルネサス エレクトロニクスホームページから入手してください。)
- アプリケーションノート RL78 ファミリ 静電容量センサユニット（CTSU2L）動作説明 (R01AN5744)
- アプリケーションノート RL78 ファミリ
  QE と SIS を使用した静電容量タッチアプリケーションの開発 (R01AN5512)
- アプリケーションノート RL78 ファミリ CTSU モジュール Software Integration System (R11AN0484)
- アプリケーションノート RL78 ファミリ TOUCH モジュール Software Integration System (R11AN0485)
- アプリケーションノート 静電容量センサマイコン 静電容量タッチ電極デザインガイド (R30AN0389)
  (最新版をルネサス エレクトロニクスホームページから入手してください。)

ホームページとサポート窓口

ルネサス エレクトロニクスホームページ

http://www.renesas.com/

静電容量センサユニット関連ページ

https://www.renesas.com/solutions/touch-key
https://www.renesas.com/qe-capacitive-touch

お問い合わせ

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

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	Feb.13.23	ー	初版発行

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本ドキュメントおよびテクニカルアップデートを参照してください。

1. 静電気対策
   CMOS 製品の取り扱いの際は静電気防止を心がけてください。CMOS 製品は強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には、当社が出荷梱包に使用している導電性のトレーやマガジンケース、導電性の緩衝材、金属ケースなどを利用し、組み立て工程にはアースを施してください。プラスチック板上に放置したり、端子を触ったりしないでください。また、CMOS 製品を実装したボードについても同様の扱いをしてください。
2. 電源投入時の処置
   電源投入時は、製品の状態は不定です。電源投入時には、LSI の内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。
3. 電源オフ時における入力信号
   当該製品の電源がオフ状態のときに、入力信号や入出力プルアップ電源を入れないでください。入力信号や入出力プルアップ電源からの電流注入により、誤動作を引き起こしたり、異常電流が流れ内部素子を劣化させたりする場合があります。資料中に「電源オフ時における入力信号」についての記載のある製品は、その内容を守ってください。
4. 未使用端子の処理
   未使用端子は、「未使用端子の処理」に従って処理してください。CMOS 製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI 周辺のノイズが印加され、LSI 内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。
5. クロックについて
   リセット時は、クロックが安定した後、リセットを解除してください。プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。
6. 入力端子の印加波形
   入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。CMOS 製品の入力がノイズなどに起因して、V_{IL}（Max.）から V_{IH}（Min.）までの領域にとどまるような場合は、誤動作を引き起こす恐れがあります。入力レベルが固定の場合はもちろん、V_{IL}（Max.）から V_{IH}（Min.）までの領域を通過する遷移期間中にチャタリングノイズなどが入らないように使用してください。
7. リザーブアドレス（予約領域）のアクセス禁止
   リザーブアドレス（予約領域）のアクセスを禁止します。アドレス領域には、将来の拡張機能用に割り付けられている リザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。
8. 製品間の相違について
   型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。同じグループのマイコンでも型名が違うと、フラッシュメモリ、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ幅射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。回路、ソフトウェアおよびこれらに関連する情報を使用する場合、お客様の責任において、お客様の機器・システムを設計ください。これらの使用に起因して生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
2. 当社製品または本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を組み込んだ製品の輸出入、製造、販売、利用、配布その他の行為を行うにあたり、第三者保有の技術の利用に関するライセンスが必要となる場合、当該ライセンス取得の判断および取得はお客様の責任において行ってください。
5. 当社製品を、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング等により生じた損害に関し、当社は、一切その責任を負いません。
6. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。

   標準水準：　コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット等

   高品質水準：輸送機器（自動車、電車、船舶等）、交通制御（信号）、大規模通信機器、金融端末基幹システム、各種安全制御装置等

   当社製品は、データシート等により高信頼性、Harsh environment 向け製品と定義しているものを除き、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
7. あらゆる半導体製品は、外部攻撃からの安全性を 100%保証されているわけではありません。当社ハードウェア／ソフトウェア製品にはセキュリティ対策が組み込まれているものもありますが、これによって、当社は、セキュリティ脆弱性または侵害（当社製品または当社製品が使用されているシステムに対する不正アクセス・不正使用を含みますが、これに限りません。）から生じる責任を負うものではありません。当社は、当社製品または当社製品が使用されたあらゆるシステムが、不正な改変、攻撃、ウイルス、干渉、ハッキング、データの破壊または窃盗その他の不正な侵入行為（「脆弱性問題」といいます。）によって影響を受けないことを保証しません。当社は、脆弱性問題に起因しまたはこれに関連して生じた損害について、一切責任を負いません。また、法令において認められる限りにおいて、本資料および当社ハードウェア／ソフトウェア製品について、商品性および特定目的との合致に関する保証ならびに第三者の権利を侵害しないことの保証を含め、明示または黙示のいかなる保証も行いません。
8. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は、データシート等において高信頼性、Harsh environment 向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
10. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。かかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
12. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。
13. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。
14. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

注 1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社が直接的、間接的に支配する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

(Rev.5.0-1　2020.10)

本社所在地

〒135-0061 東京都江東区豊洲 3-2-24（豊洲フォレシア）

www.renesas.com

お問合せ窓口

弊社の製品や技術、ドキュメントの最新情報、最寄の営業お問合せ窓口に関する情報などは、弊社ウェブサイトをご覧ください。

www.renesas.com/contact/

商標について

ルネサスおよびルネサスロゴはルネサス エレクトロニクス株式会社の商標です。すべての商標および登録商標は、それぞれの所有者に帰属します。