RL78/G15

Temperature Control of a Heater Using PWM

Introduction

This application note explains a method for controlling the temperature of a small heater using PWM functionality, assuming a cooking mug (mug-shaped electric pot) as the target application.

PWM signal is applied to a MOSFET, and the heater's heat output is controlled by switching it ON/OFF. Feedback control is implemented based on the difference between the actual temperature, detected via a thermistor input, and the set temperature. This enables heating up to the target temperature and maintaining it for a specified duration.

Target Device

Evaluation Board	RL78/G15-20pin Fast Prototyping Board
Heater	AL-DC-A6-30 Aluminum Foil Flexible Heater
Thermistor	103AT-2 AT Thermistor

When applying this application note to other microcontrollers, heaters, or thermistors, make the necessary modifications according to their specifications and conduct thorough evaluations. Additionally, please be cautious of burns and fire hazards, as the heater can reach high temperatures.

Contents

1. Specification

1.1 Specification Summary

This application note describes temperature control of a heater by controlling a MOSFET using a PWM signal. The duty cycle of the PWM signal is adjusted based on temperature detection by a thermistor, regulating the heater's heat output. By controlling the PWM signal, precise heat output adjustment is achieved, enabling stable temperature management.

1.2 Operation Details

This application note provides an example of temperature control for a heater using the PWM function.

By pressing the button (SW2) on the Fast Prototyping Board (FPB), users can configure the operating mode, temperature, and time settings to control the heater's temperature. The details of each setting are shown in Table 1-1.

Table 1-1 List of Settings

Operating Mode	Temperature Setting	Time Setting
Boil Mode	40℃、55℃、70℃ (Three Levels)	-
Stew Mode	70℃ (Fixed)Note	10 minutes, 40 minutes, 70 minutes (Three Levels)

Note Due to the specifications of the heater used in this application note, the temperature is set to 70°C.

The operating mode can be switched by pressing and holding the button for more than one second while the operating mode LED is blinking. A short press of the button changes the temperature setting in Boil mode or the time setting in Stew mode.

The set temperature or time is indicated by a three-level LED display.

If no button operation is detected for more than three seconds after setting, the heater control starts in the configured operating mode. During operation, the corresponding operating mode LED remains lit. The LED lighting pattern changes according to the temperature and time settings. Pressing and holding the button for more than two seconds during operation stops the process and transitions the system to standby mode. If the temperature exceeds 80°C during operation, an emergency stop is triggered. To restart, a reset must be performed.

The state transition diagram for this application is shown in Figure 1-1, and the descriptions of each state are listed in Table 1-2.

Figure 1-1 Temperature Control State Transition Diagram

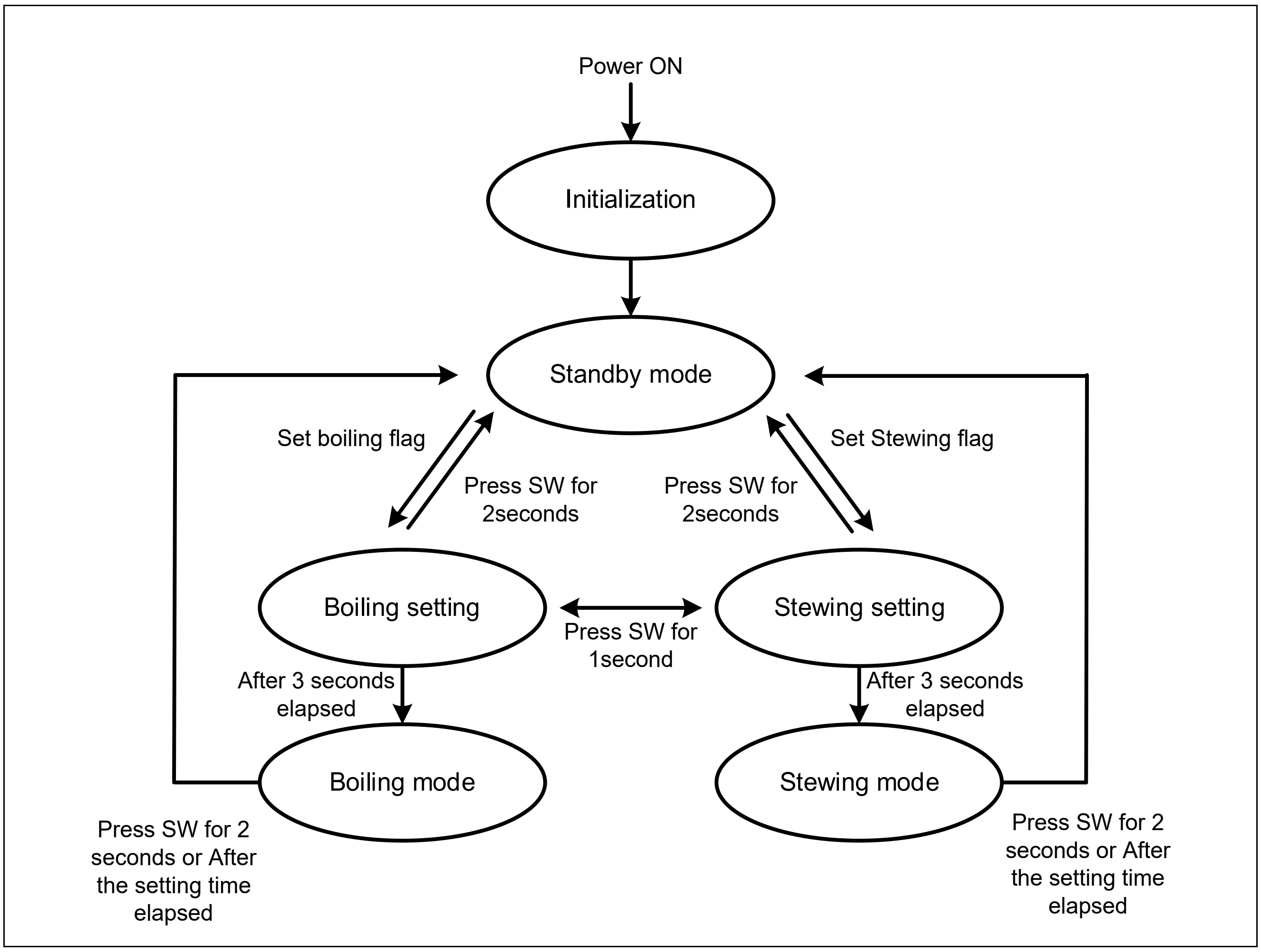
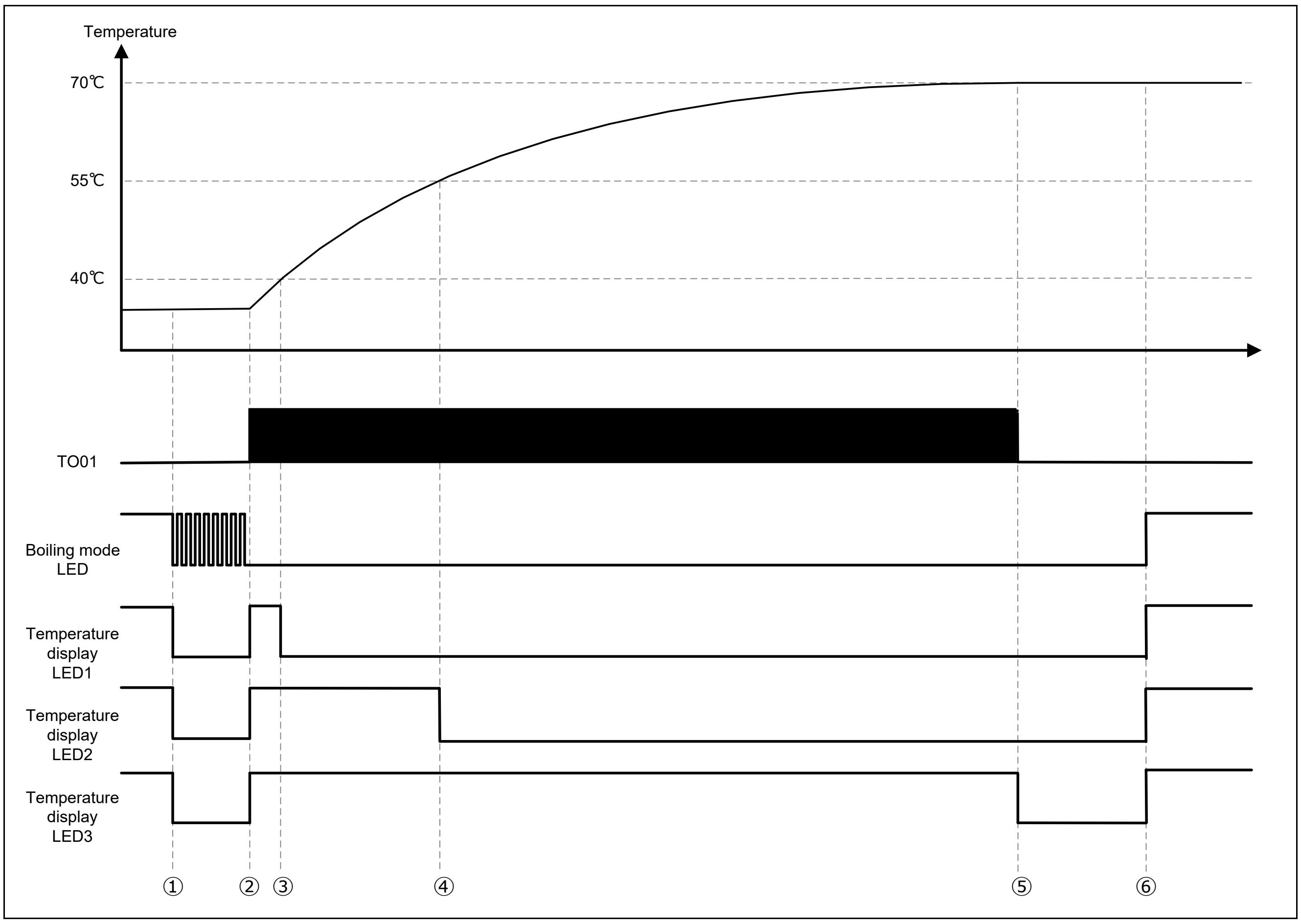


Table 1-2 Transition Factors and Processing Details for Each State

State	Transition Factors	Processing Details
Initial State	・When the power is turned on	・Initialization process ・All LEDs turn on
Standby	・Completion of initialization process ・Cancellation of Boil / Stew setting (SW long press for 2 seconds) ・Cancellation of Boil / Stew mode (SW long press for 2 seconds) ・Reaching the set temperature in Boil mode ・Elapsed set time in Stew mode	・Waiting for button operating (Boil / Stew setting) ・Set Boil flag (first time only) ・Stop interval time ・Turn off all LEDs
Boil Setting	・Button operating in Standby mode (when Boil flag is set) ・Mode change during Stew setting (SW long press for 1 sec)	・Set boiling temperature in 3 levels (Boil LED blinks) ・Accept mode change ・Accept Boil setting cancellation ・Set Boil flag
Stew Setting	・Button operation in Standby mode (when Stew flag is set) ・Mode change during Boil setting (SW long press for 1csec)	・Set Stewing time in 3 levels (Stew LED blinks) ・Accept mode change ・Accept Stew setting cancellation ・Set Stew flag
Boil Mode	・No Button operating for 3 seconds during Boil setting	・Output PWM signal until the set temperature is reached ・Display the current temperature with LED ・Accept Boil mode cancellation ・Transition to Standby mode upon reaching the set temperature
Stew Mode	・No Button operating for 3 seconds during Stew setting	・After reaching the specified temperature (70℃), maintain it using PWM control for the set time ・Display the remaining time with LED ・Accept Stew mode cancellation ・Transition to Standby mode after the set time has elapsed

The Boil mode automatically stops operation once the set temperature is reached. The operation details are shown in Figure 1-2. The temperature rise graph is for illustrative purpose only.

Figure 1-2 Detailed Operation of Boil Mode (at 70°C setting)



Note The black shading in the TO01 waveform indicates the continuous changes between High/Low of the PWM output. Please refer to Figure 1-4 for the waveform. Additionally, the LED lights up in an active-low configuration.

① The boil mode LED blinks, and the temperature setting is made.
(In this case, the temperature is set to 70°C, so all temperature display LEDs light up)

② After setting, if there is no button operation for more than 3 seconds, the heater operation will start. During operation, the duty cycle of the PWM signal is adjusted based on the difference between the temperature detected by the thermistor and the set temperature, and the heater's heat output is controlled via feedback. Additionally, the boil mode LED and the temperature display LEDs are lit according to the current temperature.
(In this case, since the temperature is below 40°C, all the temperature display LEDs are off.)

③ When the temperature reaches 40°C, temperature display LED1 will light up.

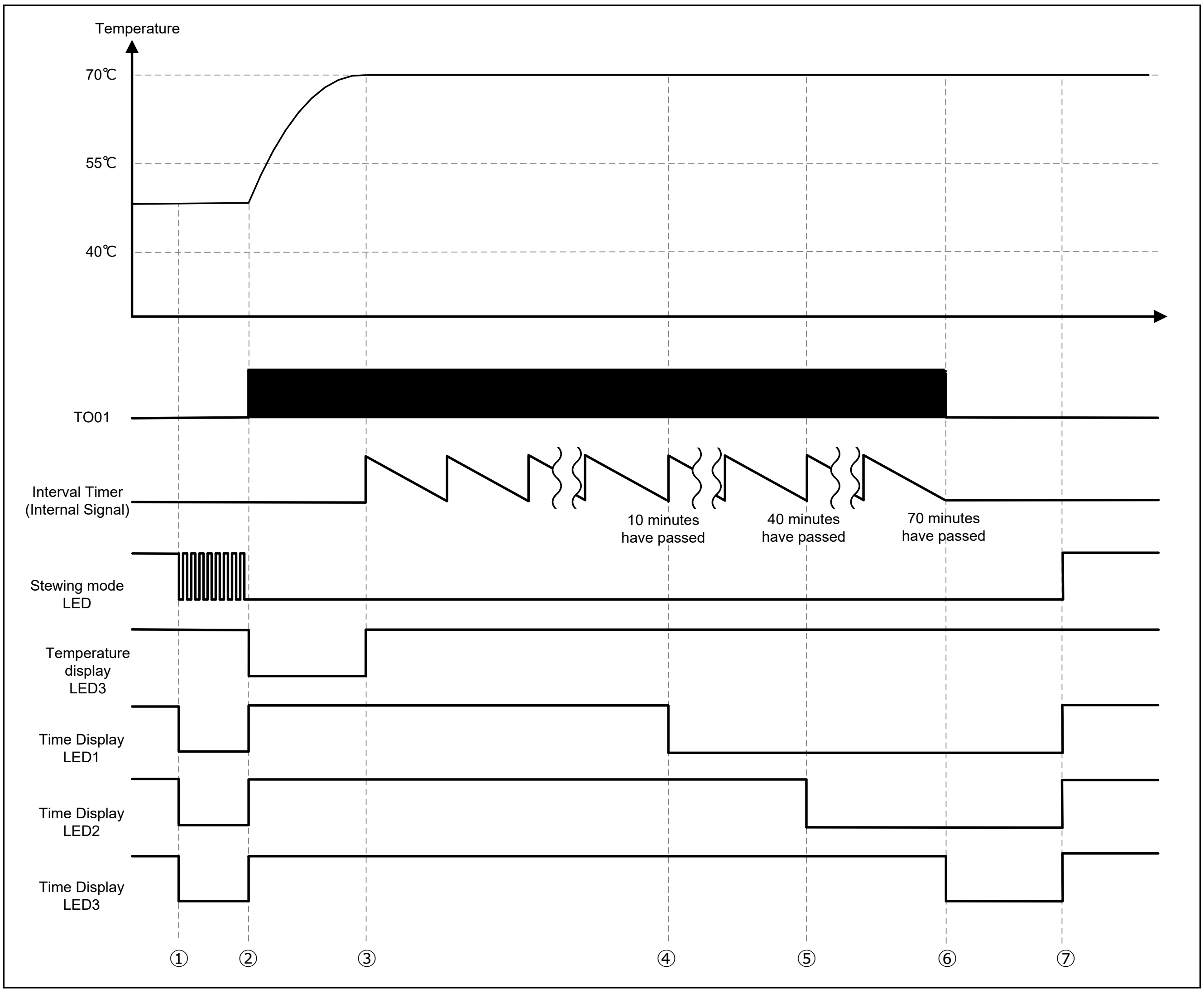
④ When the temperature reaches 55°C, temperature display LED2 will light up.

⑤ When the temperature reaches 70°C (the set temperature), temperature display LED3 will light up. The heater will also stop.

⑥ After transitioning to standby mode, all LEDs will turn off.

The Stew mode is a mode in which operation automatically stops after the specified time has elapsed once the designed temperature (70℃) is reached. The detailed operation is shown in Figure 1-3. The temperature rise graph is for illustrative purpose only.

Figure 1-3 Detailed Operation of Stew Mode (70 Minute Setting)



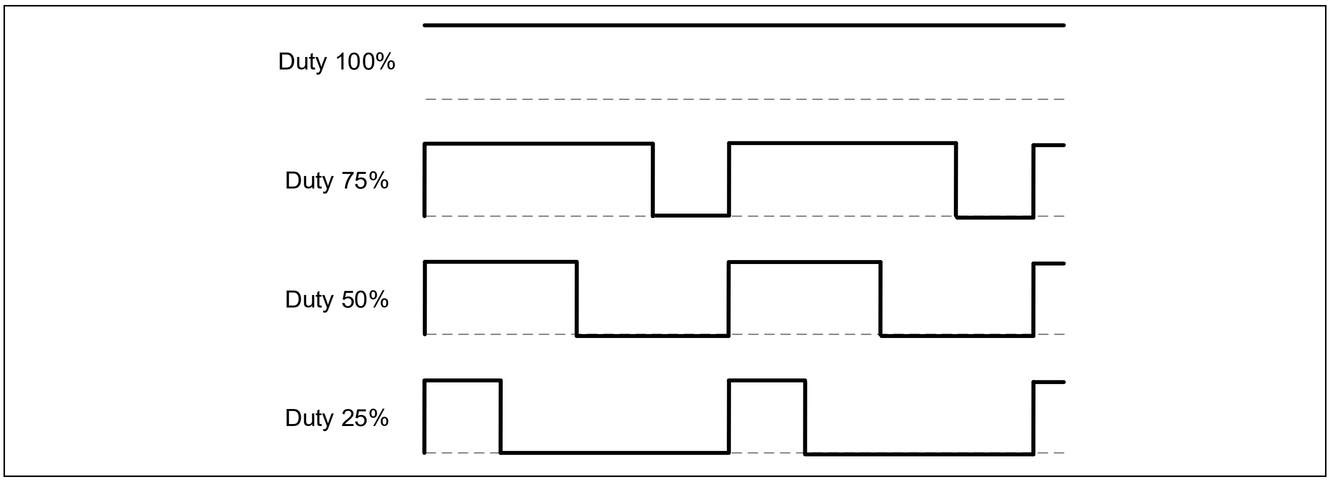
Note The black shading in the TO01 waveform indicates the continuous changes between High and Low of the PWM output.
Please refer to Figure 1-4 for the waveform.
Additionally, the LED lights up in an active low configuration.

① The stew mode LED blinks, and the time setting is adjusted.
(In this case, the time is set to 70 minutes, so all-time display LEDs light up.)
② After setting, if there is no button operation for more than 3 seconds, the heater operation starts.During operation, the duty cycle of the PWM signal is adjusted based on the difference between the temperature detected by the thermistor and the set temperature, and the heater's heat output is controlled via feedback. Additionally, only the boil mode LED and temperature display LED3 light up. All-time display LEDs turn off.
③ When the designed temperature (70℃) is reached, temperature LED3 turns off, and the interval timer starts operating.
④ When 10 minutes have elapsed after reaching the designed temperature (70℃), time display LED1 lights up.
⑤ When 40 minutes have elapsed after reaching the designed temperature (70℃), time display LED2 lights up.
⑥ When 40 minutes have elapsed after reaching the designed temperature (70℃), time display LED2 lights up.The heater operation and interval timer stop.
⑦ After transitioning to standby mode, all LEDs turn off.

The TO01 PWM waveform has four patterns: 100%, 75%, 50%, and 25% duty cycles. The duty cycles is set based on the difference between the current temperature and the set temperature.

The waveforms are shown below.

Figure 1-4 TO01 PWM Waveform



The following table shows the correlation between the temperature difference to the set temperature and the duty cycle of the PWM signal output in boiling mode and simmering mode.

Table 1-3 Correlation table between the temperature difference to the set temperature and the duty cycle of the PWM signal output.

Temperature Difference	Greater than 30°C	Greater than 20°C	Greater than 10°C	10°C or less
Duty Cycle	100%	75%	50%	25%

2. Operating Conditions

This sample program has been tested under the following conditions.

Table 2-1 Operating Verification Conditions

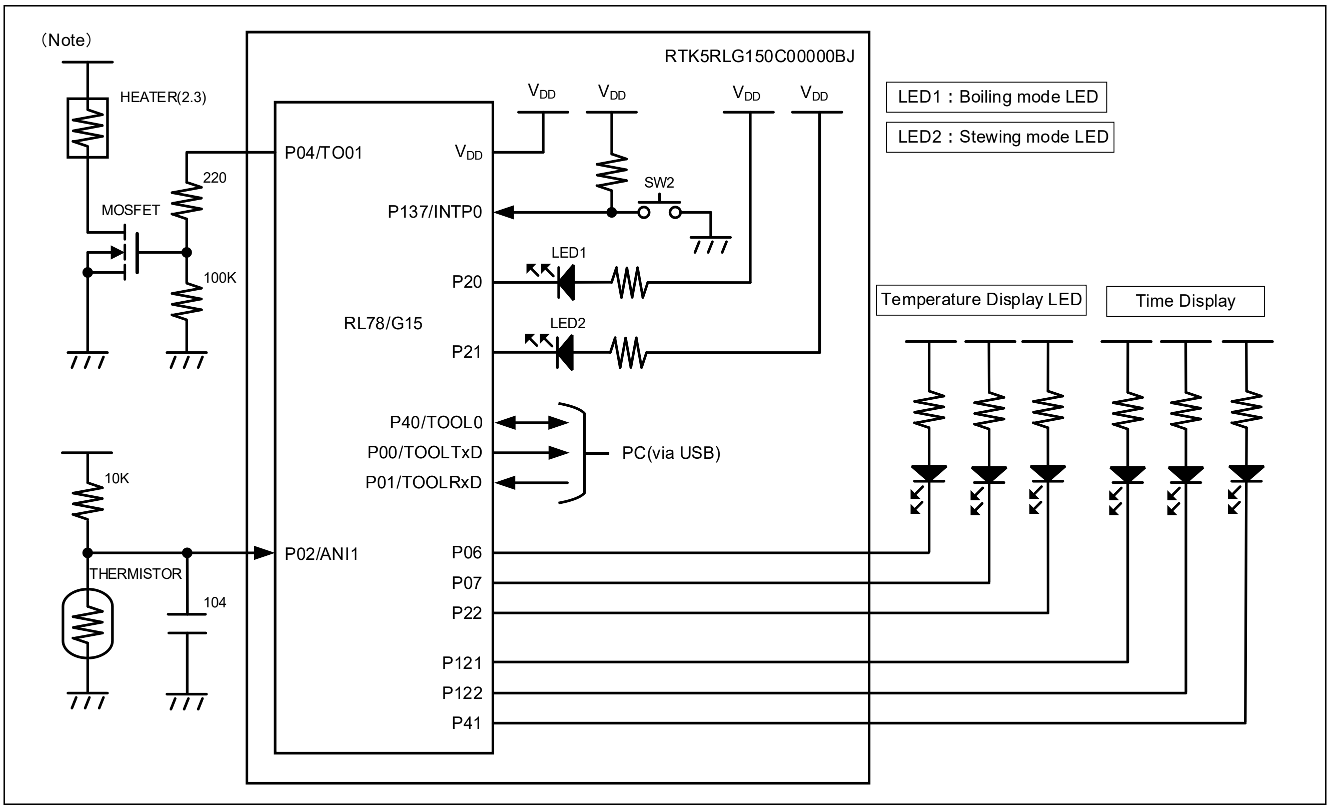
Item	Description
Microcontroller Used	RL78/G15 (R5F12068ASP)
Board Used	RL78/G15 Fast Prototyping Board (RTK5RLG150C00000BJ)
Operating Frequency	• High-speed on-chip oscillator clock：16MHz CPU / Peripheral hardware clock：16MHz
Operating Voltage	• 5V • SPOR detection voltage setting Rising TYP. 2.90V Falling TYP. 2.84V
Integrated Development Environment (CS+)	Renesas Electronics • CS+ for CC V8.12.0
C compiler (CS+)	Renesas Electronics CC-RL V1.14
Integrated Development Environment (e^2 studio)	Renesas Electronics e^2 studio 2024-10 (24.10.1)
C compiler (e^2 studio)	Renesas Electronics CC-RL V1.14
Smart Configurator	V.1.11.0
Board Support Package (r_bsp)	V.1.70
Heater	AL-DC-A6-30 Aluminum Foil Flexible Heater
Thermistor	103AT-2 AT Thermistor

3. Hardware Description

3.1 Example of Hardware Configuration

Figure 3-1 shows an example of the hardware configuration used in the sample code for this application.

Figure 3-1 Example of Hardware Configuration



Note When using the heater described in this application note, a power supply of 9 or higher is required to achieve temperature above 70℃. For details, please refer to the electrical characteristics of the heater.

Remark Keep the ground connection of the thermistor and MOSFET separate.

3.2 List of Used Pins

Table 3-1 shows the used pins and their functions。

Table 3-1 Used Pins and Functions

Pin name	I / O	Description
P02 / ANI1	Input	Connected to the thermistor, retrieves resistance values, and performs A/D conversion
P04 / TO01	Output	Connected to the relay (MOSFET), output PWM signal.
P20	Output	Boil mode selection LED
P21	Output	Stew mode selection LED
P06	Output	3-stage temperature display LED for boil mode.
P07	Output	
P22	Output	
P121	Output	3-stage temperature display LED for Stew mode.
P122	Output	
P41	Output	
P137/INTP0	Input	SW input

Caution This application note only process the used pins. This application note only process the used pins. When designing an actual circuit, ensure proper pin handling and meet electrical characteristics requirement.

4. Software Description

4.1 Peripheral Function Used

The peripheral functions used in this sample program are listed below.

Table 4-1 List of Used Peripheral Functions

Peripheral Function	Usage
Interrupt Controller (INTC)	Detects button press via external interruption (falling edge)
A/D converter (ADC)	Acquires the thermistor resistance value (10-bit resolution)
Timer Array Unit (TAU)	PWM master (period setting)
	PWM slave (duty cycle control), PWM output
	Determine long press (mode switching、standby transition) Confirm operating mode, preventing chattering
12-bit Interval Timer (IT)	Manages time for Stew mode
PORT	Controls LEDs

4.2 Option Byte Settings

The option byte setting for this sample program are shown below.

Table 4-2 Option Byte Setting

Address	Setting value	Description
000C0H / 020C0H	1110 1111B (0xEF)	Stop watchdog timer operation (Counting stop after reset release)
000C1H / 020C1H	1111 1011B (0xFB)	P125 pin control：$\overline{\text{RESET}}$ input SPOR detection voltage setting Rising TYP. 2.90V (2.76V ~ 3.02V) Falling TYP. 2.84V (2.70V ~ 2.96V)
000C2H / 020C2H	1111 1001B (0xF9)	High-speed on-chip oscillator clock : 16MHz Operating voltage range：2.4V ~ 5.5V
000C3H / 020C3H	1000 0101B (0x85)	On-chip debugging enabled

4.3 Folder Structure

The folder structure of the source files and header files in this sample program is shown below.

Note that files automatically generated by the integrated development environment and BSP environment files are excluded.

Table 4-3 Folder Structure

Folder / Folder Name	Description	Smart Configurator Used
¥r01an7611_rl78g15_heater_control<DIR>	Sample code folder	
¥src<DIR>	Program storage folder	
main.c	Sample code source file	
¥smc_gen<DIR>	Smart Configurator generated folder	√
¥Config_ADC<DIR>	ADC program storage folder	√
Config_ADC.c	ADC source file	√
Config_ADC.h	ADC header file	√
Config_ADC_user.c	ADC interrupt and temperature measurement source file	√
¥Config_INTC<DIR>	INTC program storage folder	√
Config_INTC.c	INTC source file	√
Config_INTC.h	INTC header file	√
Config_INTC_user.c	INTC interrupt source file	√
¥Config_IT<DIR>	IT program storage file	√
Config_IT.c	IT source file	√
Config_IT.h	IT header file	√
Config_IT_user.c	IT interrupt source file	√
¥Config_PORT<DIR>	PORT program storage folder	√
Config_PORT.c	PORT source file	√
Config_PORT.h	PORT header file	√
Config_PORT_user.c	LED control source file	√
¥Config_TAU0_0 <DIR>	TAU0_0 program storage folder	√
Config_ TAU0_0.c	TAU0_0 source file	√
Config_ TAU0_0.h	TAU0_0 header file	√
Config_ TAU0_0_user.c	Heater control source file	√
¥Config_ TAU0_2<DIR>	TAU0_2 program storage folder	√
Config_ TAU0_2.c	TAU0_2 source file	√
Config_ TAU0_2.h	TAU0_2 header file	√
Config_ TAU0_2_user.c	TAU0_2 interrupt source file	√

Note ” <DIR>” indicates a directory.

4.4 List of Variables

The variables used in this sample program are listed below.

Table 4-4 List of variables

Type	Variable Name	Description	File
e_system_status_t	g_system_status	Variable indicating the application state	main.c Config_TAU0_0_user.c
e_mode_select_t	g_mode_flag	Variable indicating the heater operation state	main.c
uint8_t	g_boil_mode_initialized	Initialization flag for boiling mode	main.c
int8_t	g_current_temp	Variable indicating the temperature	main.c Config_TAU0_0_user.c
uint8_t	g_adc_done	A/D interrupt flag	Config_ADC_user.c
e_stew_status_t	g_stew_status	Variable indicating the stew mode state	Config_IT_user.c main.c
uint16_t	g_stew_time_count	Variable for stew mode time count	Config_IT_user.c main.c
uint16_t	g_stew_time_set	Variable for setting stew mode time count	Config_IT_user.c Config_PORT_user.c
uint8_t	g_pwm_started	PWM output flag	Config_IT_user.c main.c Config_TAU0_0_user.c
uint8_t	g_led_blink_flag	LED blinking control 0: OFF　1: ON	Config_IT_user.c main.c
uint8_t	g_led_blink_count	LED blink counter	Config_IT_user.c
uint8_t	g_heater_status	Temperature status flag 0: Normal operation 1: Abnormal stop	Config_TAU0_0_user.c Config_IT_useer.c
uint16_t	g_button_press_time	Button press duration	Config_TAU0_2_user.c main.c Config_INTC_user.c
uint8_t	g_button_released	Button release detection	Config_TAU0_2_user.c main.c Config_INTC_user.c
uint16_t	g_no_operation_time	Time without button operation	Config_TAU0_2_user.c main.c Config_INTC_user.c
uint8_t	g_button_1sec_detected	Detection of button press for 1second or more	Config_TAU0_2_user.c main.c
uint8_t	g_button_2sec_detected	Detection of button press for 2 seconds or more	Config_TAU0_2_user.c main.c

4.5 List of Constants

The list of constants used in this sample is shown below.

Table 4-5 List of Constants (1/3)

Constant Name	Value	Description	File
SERIES_R	10000	Series register (10kΩ)	r_cg_userdefine.h Config_ADC_user.c
ADC_MAX	1023	Maximum value of 10-bit ADC	r_cg_userdefine.h Config_ADC_user.c
TEMP_ABNORMAL	80	Abnormal temperature	r_cg_userdefine.h Config_IT_user.c Config_TAU0_0_user.c
TARGET_TEMP_LOW	40	Target temperature (low) for boiling mode	r_cg_userdefine.h Config_PORT_user.c Config_TAU0_0_user.c
TARGET_TEMP_MID	55	Target temperature (mid) for boiling mode	r_cg_userdefine.h Config_PORT_user.c Config_TAU0_0_user.c
TARGET_TEMP_HIGH	70	Target temperature (high) for boiling mode	r_cg_userdefine.h Config_PORT_user.c Config_TAU0_0_user.c
TEMP_HYSTERESIS	2	Temperature hysteresis	r_cg_userdefine.h Config_TAU0_0_user.c
DUTY_100	0x3E80	Duty cycle 100%	r_cg_userdefine.h Config_TAU0_0_user.c
DUTY_75	0x2EDF	Duty cycle 75%	r_cg_userdefine.h Config_TAU0_0_user.c
DUTY_50	0x1F40	Duty cycle 50%	r_cg_userdefine.h Config_TAU0_0_user.c
DUTY_25	0x0F9F	Duty cycle 25%	r_cg_userdefine.h Config_TAU0_0_user.c

Table 4-6 List of Constants（2/3）

Constant Name	Value	Description	File
STEW_TIME_SHORT	10	Stew mode duration (short)	r_cg_userdefine.h Config_IT_user.c
STEW_TIME_MIDDLE	40	Stew mode duration (Medium)	r_cg_userdefine.h Config_IT_user.c
STEW_TIME_LONG	70	Stew mode duration (long)	r_cg_userdefine.h Config_IT_user.c
STEW_TEMP	TARGET_TEMP_HIGH	Stew mode target temperature	r_cg_userdefine.h Config_IT_user.c Config_TAU0_0_user.c
PRESS_TIME_1SEC	100	Button press duration for 1 second	r_cg_userdefine.h main.c Config_TAU0_2_user.c
PRESS_TIME_2SEC	200	Button press duration for 2 seconds	r_cg_userdefine.h Config_TAU0_2_user.c
NO_PRESS_TIME	300	No button operation for 3 seconds	r_cg_userdefine.h main.c
LED_DELAY_TIME	100	LED illumination time during initialization (ms)	r_cg_userdefine.h Config_PORT_user.c
LED_MIN_ON_TIME	100	Minimum illumination time for temperature display LED (10ms * 100 = 1s)	r_cg_userdefine.h Config_PORT_user.c
LED_BOIL_MODE_ON()	P2_bit.no0 = 0U	Turn on boiling mode LED	r_cg_userdefine.h main.c Config_PORT_user.c
LED_BOIL_MODE_OFF()	P2_bit.no0 = 1U	Turn off boiling mode LED	r_cg_userdefine.h main.c Config_PORT_user.c
LED_BOIL_TEMP1_ON()	P0_bit.no6 = 0U	Turn on temperature display LED1	r_cg_userdefine.h Config_PORT_user.c
LED_BOIL_TEMP1_OFF()	P0_bit.no6 = 1U	Turn off temperature display LED1	r_cg_userdefine.h main.c Config_PORT_user.c
LED_BOIL_TEMP2_ON()	P0_bit.no7 = 0U	Turn on temperature display LED2	r_cg_userdefine.h Config_PORT_user.c
LED_BOIL_TEMP2_OFF()	P0_bit.no7 = 1U	Turn off temperature display LED2	r_cg_userdefine.h main.c Config_PORT_user.c
LED_BOIL_TEMP3_ON()	P2_bit.no2 = 0U	Turn on temperature display LED3	r_cg_userdefine.h Config_IT_user.c Config_PORT_user.c
LED_BOIL_TEMP3_OFF()	P2_bit.no2 = 1U	Turn off temperature display LED3	r_cg_userdefine.h main.c Config_PORT_user.c

Table 4-7 List of Constants（3/3）

Constant Name	Value	Description	File
LED_STEW_MODE_ON()	P2_bit.no1 = 0U	Turn on the stew mode LED	r_cg_userdefine.h main.c Config_PORT_user.c
LED_STEW_MODE_OFF()	P2_bit.no1 = 1U	Turn off the stew mode LED	r_cg_userdefine.h main.c Config_PORT_user.c
LED_STEW_TIME1_ON()	P12_bit.no1 = 0U	Turn on time display LED1	r_cg_userdefine.h Config_PORT_user.c
LED_STEW_TIME1_OFF()	P12_bit.no1 = 1U	Turn off time display LED1	r_cg_userdefine.h main.c Config_PORT_user.c
LED_STEW_TIME2_ON()	P12_bit.no2 = 0U	Turn on time display LED2	r_cg_userdefine.h Config_PORT_user.c
LED_STEW_TIME2_OFF()	P12_bit.no2 = 1U	Turn off time display LED2	r_cg_userdefine.h main.c Config_PORT_user.c
LED_STEW_TIME3_ON()	P4_bit.no1 = 0U	Turn on time display LED3	r_cg_userdefine.h Config_PORT_user.c
LED_STEW_TIME3_OFF()	P4_bit.no1 = 1U	Turn off time display LED3	r_cg_userdefine.h main.c Config_PORT_user.c
LED_ALL_OFF()	do { ¥ LED_BOIL_MODE_OFF(); ¥ LED_STEW_MODE_OFF(); ¥ LED_BOIL_TEMP1_OFF(); ¥ LED_BOIL_TEMP2_OFF(); ¥ LED_BOIL_TEMP3_OFF(); ¥ LED_STEW_TIME1_OFF(); ¥ LED_STEW_TIME2_OFF(); ¥ LED_STEW_TIME3_OFF(); ¥ } while (0)	Turn off all LEDs	r_cg_userdefine.h main.c Config_PORT_user.c
thermistor_table[]	Stores values calculated using the B constant within the range of 50℃ to 110℃	Temperature characteristics of the thermistor	Config_ADC_user.c

4.6 List of Enumerations

This section is a list of the enumerations used in this sample program.

4.6.1 Enumerations : e_system_status_t

This is the enumeration that manages the application's state.

Table 4-8 Enumerations : e_system_status_t

Definition	Value	Description	File
SYSTEM_INIT	0	Initialization state	r_cg_userdefine.h main.c Config_TAU0_0_user.c
SYSTEM_STANDBY	1	Standby mode	
SYSTEM_BOIL_SETTING	2	Boiling mode setting	
SYSTEM_STEW_SETTING	3	Stew mode setting	
SYSTEM_BOIL_MODE	4	Boiling mode	
SYSTEM_STEW_MODE	5	Stewing mode	

4.6.2 Enumerations : e_mode_select_t

This enumeration manages the heater operation mode selection state.

Table 4-9 Enumerations : e_mode_select_t

Definition	Value	Description	File
MODE_BOIL	0	Boiling mode	r_cg_userdefine.h main.c Config_PORT_user.c
MODE_STEW	1	Stew mode	

4.6.3 Enumerations : e_stew_status_t

This enumerations manages the status of the stewing mode.

Table 4-10 Enumerations : e_stew_status_t

Definition	Value	Description	File
STEW_IDLE	0	Standby State in stewing mode	r_cg_userdefine.h main.c Config_IT_user.c
STEW_HEATING	1	Heating in stewing mode	
STEW_RUNNING	2	Maintaining temperature in stewing mode	
STEW_COMPLETE	3	Stewing mode complete	

4.6.4 Enumerations : e_button_status_t

This enumeration is used to manage the button press status.

表 4-11 Enumerations : e_button_status_t

Definition	Value	Description	Fle
BUTTON_IDLE	0	Button not pressed	r_cg_userdefine.h main.c Config_IT_user.c
BUTTON_DETECT	1	Button press started	
BUTTON_RELEASE	2	Waiting for button release	
BUTTON_STANDBY	3	Standby state after transition to standby mode	

4.7 List of Functions

The following table lists the functions used in this sample program.

Table 4-12 List of Functions

Function Name	Description	Source File
main	Main Processing	main.c
system_init	Initialization setting	main.c
r_Config_ADC_interrupt	A/D conversion completion interrupt processing	Config_ADC_user.c
calculate_resistance	Calculates thermistor resistance from A/D conversion results	Config_ADC_user.c
calculate_temperature	Calculates temperature from thermistor resistance	Config_ADC_user.c
r_get_temperature	Retrieves temperature from thermistor resistance	main.c Config_ADC_user.c
r_Config_INTC_intp0_interrupt	External interruption processing for SW2	Config_INTC_user.c
r_Config_IT_interrupt	Interval timer interrupt processing	Config_IT_user.c
r_start_stew_mode	Converts stew mode set time into interval timer interrupt count	main.c Config_IT_user.c
r_control_stew_mode	Overall control of stew mode	main.c Config_IT_user.c
r_led_sequence	Processing that makes the timing of LED lighting and extinguishing constant	main.c Config_PORT_user.c
display_setting_led	Displays setting via LED	main.c Config_PORT_user.c
r_display_temp_led	Displays current temperature via LED	main.c Config_PORT_user.c
r_display_time_led	Displays elapsed time via LED	main.c Config_PORT_user.c
r_control_heater_pwm	Adjust PWM output duty cycle based on current temperature	main.c Config_TAU0_0_user.c
r_start_boil_mode	Starts boiling mode	main.c Config_TAU0_0_user.c
r_stew_heater_pwm	Controls PWM output duty cycle in stew mode	Config_TAU0_0_user.c Config_IT_user.c
r_get_target_temp	Retrieves target temperature for boiling mode	main.c Config_TAU0_0_user.c Config_PORT_user.c
r_stop_pwm_output	Stops PWM output and fixes TO output terminal to Loe	main.c Config_TAU0_0_user.c Config_IT_user.c
r_Config_TAU0_2_interrupt	TAU0_2 interrupt processing	Config_TAU0_2_user.c

4.8 Function Specifications

Function Specifications for This Sample Program

[Function name] main

Overview	Main processing
Header	None
Declaration	void main(void)
Description	Executes the main processing for heater control
Arguments	None
Return	None
Note	None

[Function name] system_init

Overview	Initialization setting
Header	None
Declaration	void system_init(void)
Description	Performs initialization setting before transitioning to standby mode
Arguments	None
Return	None
Note	None

[Function name] r_Config_ADC_interrupt

Overview	A/D conversion completion interrupt processing
Header	Config_ADC.h
Declaration	static void __near r_Config_ADC_interrupt(void)
Description	Sets the A/D operation completion flag
Arguments	None
Return	None
Note	None

[Function name] calculate_resistance

Overview	Calculation of thermistor resistance from A/D conversion result
Header	Config_ADC.h
Declaration	uint32_t calculate_resistance(uint16_t adc_value)
Description	Calculates the thermistor resistance from the A/D conversion result
Argument	adc_value : A/D conversion result
Return	r : Thermistor resistance
Note	None

[Function name] calculate_temperature

Overview	Temperature calculation from thermistor resistance
Header	Config_ADC.h
Declaration	int8_t calculate_temperature(uint16_t adc_value)
Description	Calculates the temperature based on the resistance of the thermistor
Arguments	adc_value : A/D conversion result
Return	temp : Temperature
Note	None

[Function name] r_get_temperature

Overview	Temperature calculation from A/D conversion result
Header	Config_ADC.h
Declaration	int8_t r_get_temperature(void)
Description	Calculates the temperature from the A/D conversion result
Arguments	None
Return	temperature : temperature
Note	None

[Function name] r_Config_INTC_intp0_interrupt

Overview	External interruption processing for SW2
Header	Config_INTC.h
Declaration	static void __near r_Config_INTC_intp0_interrupt(void)
Description	Initializes the processing when SW2 is pressed
Arguments	None
Return	None
Note	None

[Function name] r_Config_IT_interrupt

Overview	Interval timer interrupt processing
Header	Config_IT.h
Declaration	static void __near r_Config_IT_interrupt(void)
Description	Counts the number of interrupt occurrences
Arguments	None
Return	None
Note	None

[Function name] r_start_stew_mode

Overview	Conversion of stew mode setting time to interval timer interrupt counts
Header	Config_IT.h
Declaration	void r_start_stew_mode(uint8_t stew_level)
Description	Converts the stew mode setting time into interval timer interrupt counts
Arguments	stew_level : Setting time
Return	None
Note	None

[Function name] r_control_stew_mode

Overview	Overall control processing for stew mode
Header	Config_IT.h
Declaration	void r_control_stew_mode(int8_t current_temp)
Description	Controls the overall stew mode operation
Arguments	current_temp : Current temperature
Return	None
Note	None

[Function name] r_led_sequence

Overview	Ensure consistent LED on/off timing
Header	Config_PORT.h
Declaration	void r_led_sequence(void)
Description	Controls the timing of LED turning on and off to be consistent
Arguments	None
Return	None
Note	None

[Function name] display_setting_led

Overview	Displays setting information using LED
Header	Config_PORT.h
Declaration	void display_setting_led(uint8_t level, e_mode_select_t mode)
Description	Determines which LEDs should light up based on the selected mode
Arguments	level : 3 level setting value mode : selected operation mode（boiling mode or stew mode）
Return	None
Note	None

[Function name] r_display_temp_led

Overview	Displays setting information using LEDs
Header	Config_PORT.h
Declaration	void r_display_temp_led(int8_t current_temp, uint8_t setting_level)
Description	Controls the LEDs for temperature display to indicate the current temperature
Arguments	current_temp : Current temperature setting_level : 3 level setting value
Return	None
Note	None

[Function name] r_display_time_led

Overview	Displays elapsed time using LEDs
Header	Config_PORT.h
Declaration	void r_display_time_led(uint16_t time_count)
Description	Controls the LEDs for time display to indicate elapsed time
Arguments	time_count : Interval timer interrupt count
Return	None
Note	None

[Function name] r_control_heater_pwm

Overview	Adjusts the PWM duty cycle based on the current temperature
Header	Config_TAU0_0.h
Declaration	void r_control_heater_pwm(int8_t current_temp, int8_t target_temp)
Description	Adjusts the PWM output duty cycle according to the current temperature
Arguments	current_temp : Current temperature obtained from A/D conversion target_temp : Target temperature set in the selected mode
Return	None
Note	If g_heater_status is set to 1, the PWM signal will not be output. A reset is required to resume PWM signal output

[Function name] r_start_boil_mode

Overview	Starts boil mode operation
Header	Config_TAU0_0.h
Declaration	void r_start_boil_mode(uint8_t boil_level)
Description	Initiates the operation of boil mode
Arguments	boil_level : Set temperature for boil mode
Return	None
Note	None

[Function name] r_stew_heater_pwm

Overview	Controls the PWM duty cycle in stew mode
Header	Config_TAU0_0.h
Declaration	void r_stew_heater_pwm(int8_t current_temp)
Description	Controls the PWM output duty cycle for stew mode
Arguments	current_temp : Current temperature obtained from A/D conversion
Return	None
Note	None

[Function name] r_get_target_temp

Overview	Retrieves the set temperature for boil mode
Header	Config_TAU0_0.h
Declaration	int8_t r_get_target_temp(uint8_t level)
Description	Obtains the set temperature for boil mode
Arguments	level : Temperature setting for boil mode
Return	Target temperature for boil mode
Note	None

[Function name] r_stop_pwm_output

Overview	Stops PWM output and sets the TO pin to Low
Header	Config_TAU0_0.h
Declaration	void r_stop_pwm_output(void)
Description	Stops the PWM output and fixes the TO output pin to Low
Arguments	None
Return	None
Note	None

[Function name] r_Config_TAU0_2_interrupt

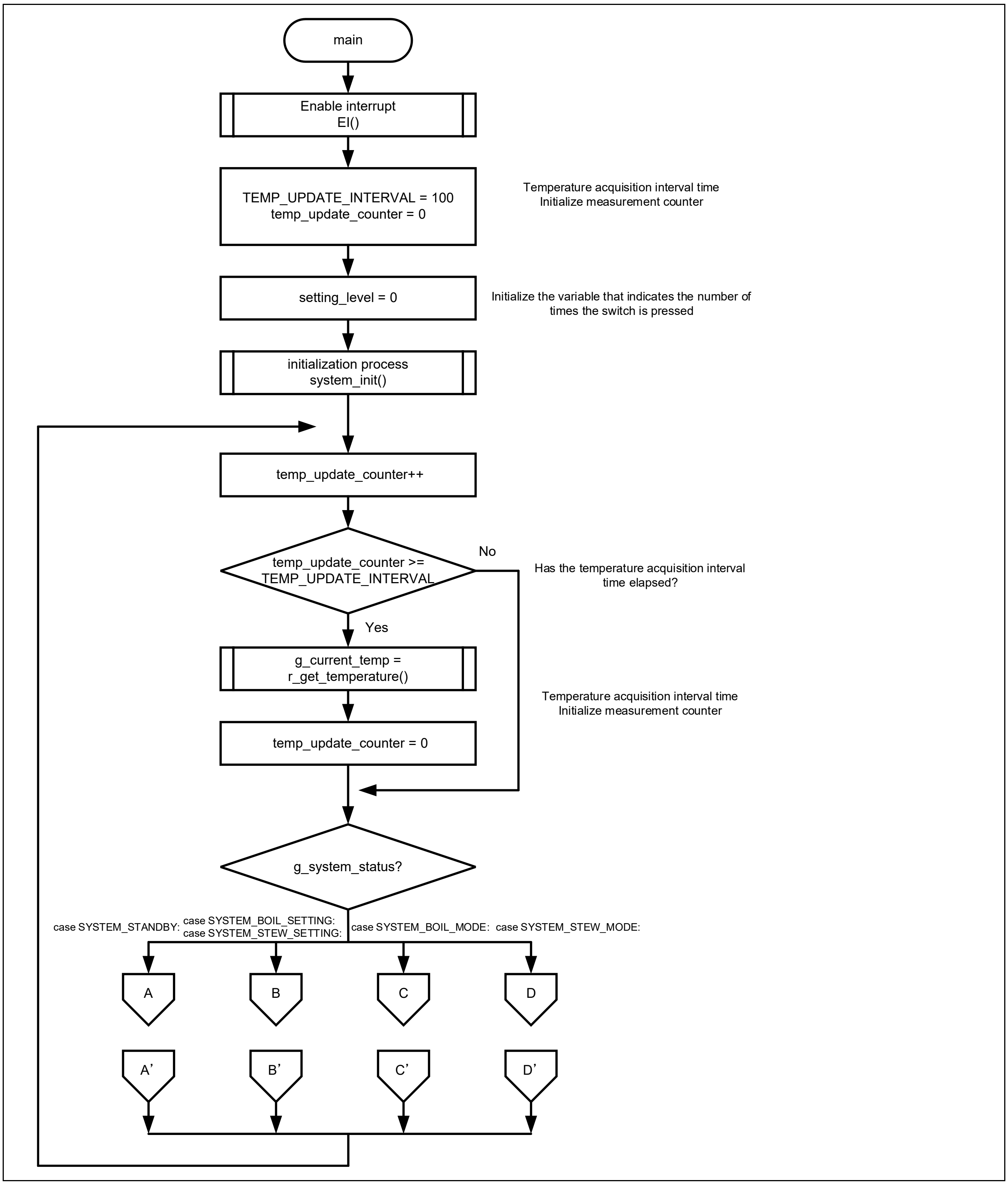
Overview	Interrupt processing for TAU0_2
Header	Config_TAU0_2.h
Declaration	static void __near r_Config_TAU0_2_interrupt(void)
Description	Counts button long-press duration
Arguments	None
Return	None
Note	None

4.9 Flowcharts

4.9.1 Flowchart of main Functions

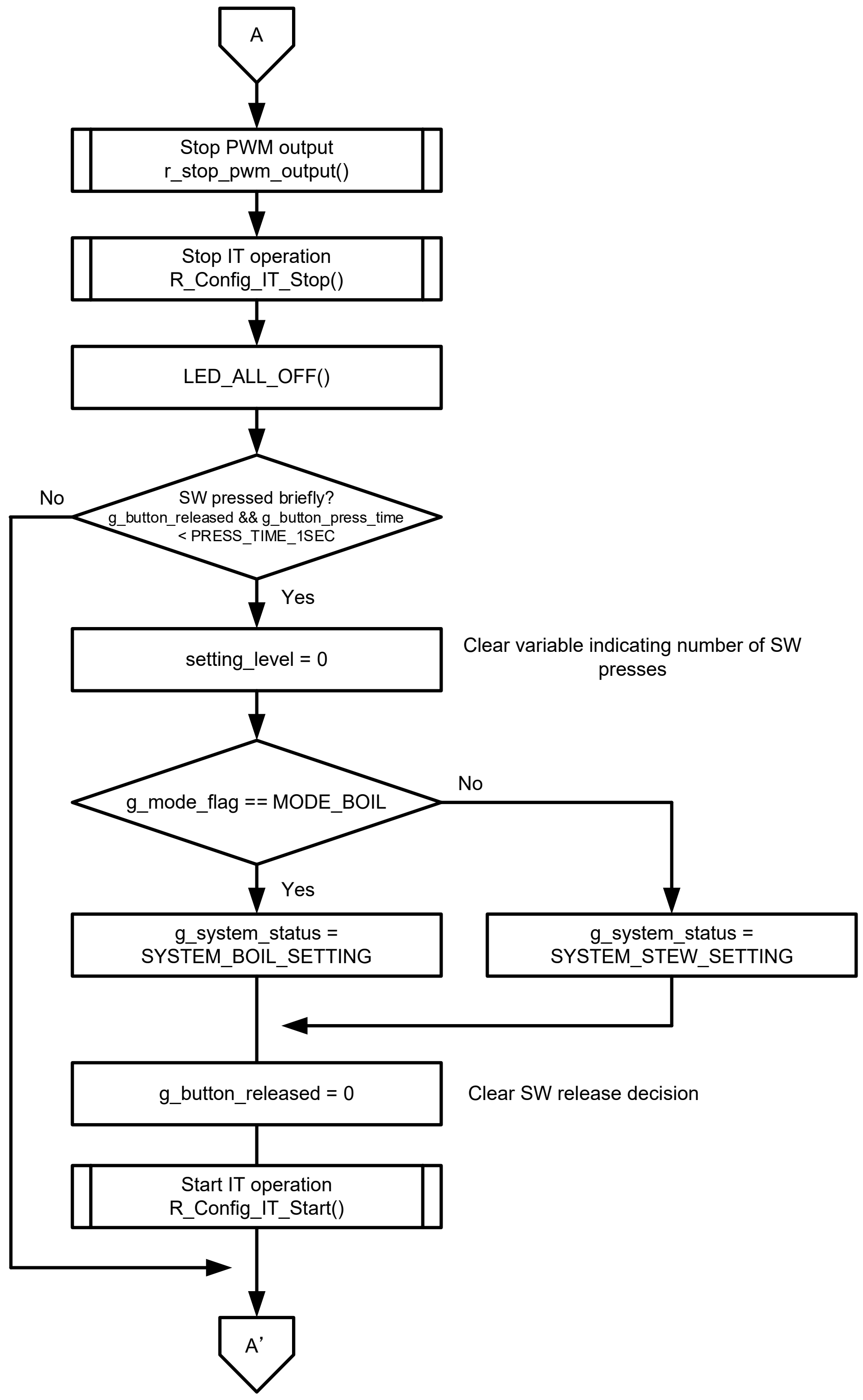
The flowchart of the main function is shown below.

Figure 4-1 Flowchart of main Functions (1/6)


main
Enable interrupt
EI()
TEMP_UPDATE_INTERVAL = 100
temp_update_counter = 0
Temperature acquisition interval time
Initialize measurement counter
setting_level = 0
Initialize the variable that indicates the number of times the switch is pressed
initialization process
system_init()
temp_update_counter++
temp_update_counter >= TEMP_UPDATE_INTERVAL
No
Has the temperature acquisition interval time elapsed?
Yes
g_current_temp = r_get_temperature()
Temperature acquisition interval time
Initialize measurement counter
temp_update_counter = 0
g_system_status?
case SYSTEM_STANDBY:
case SYSTEM_BOIL_SETTING:
case SYSTEM_STEW_SETTING:
case SYSTEM_BOIL_MODE:
case SYSTEM_STEW_MODE:
A
B
C
D
A'
B'
C'
D'

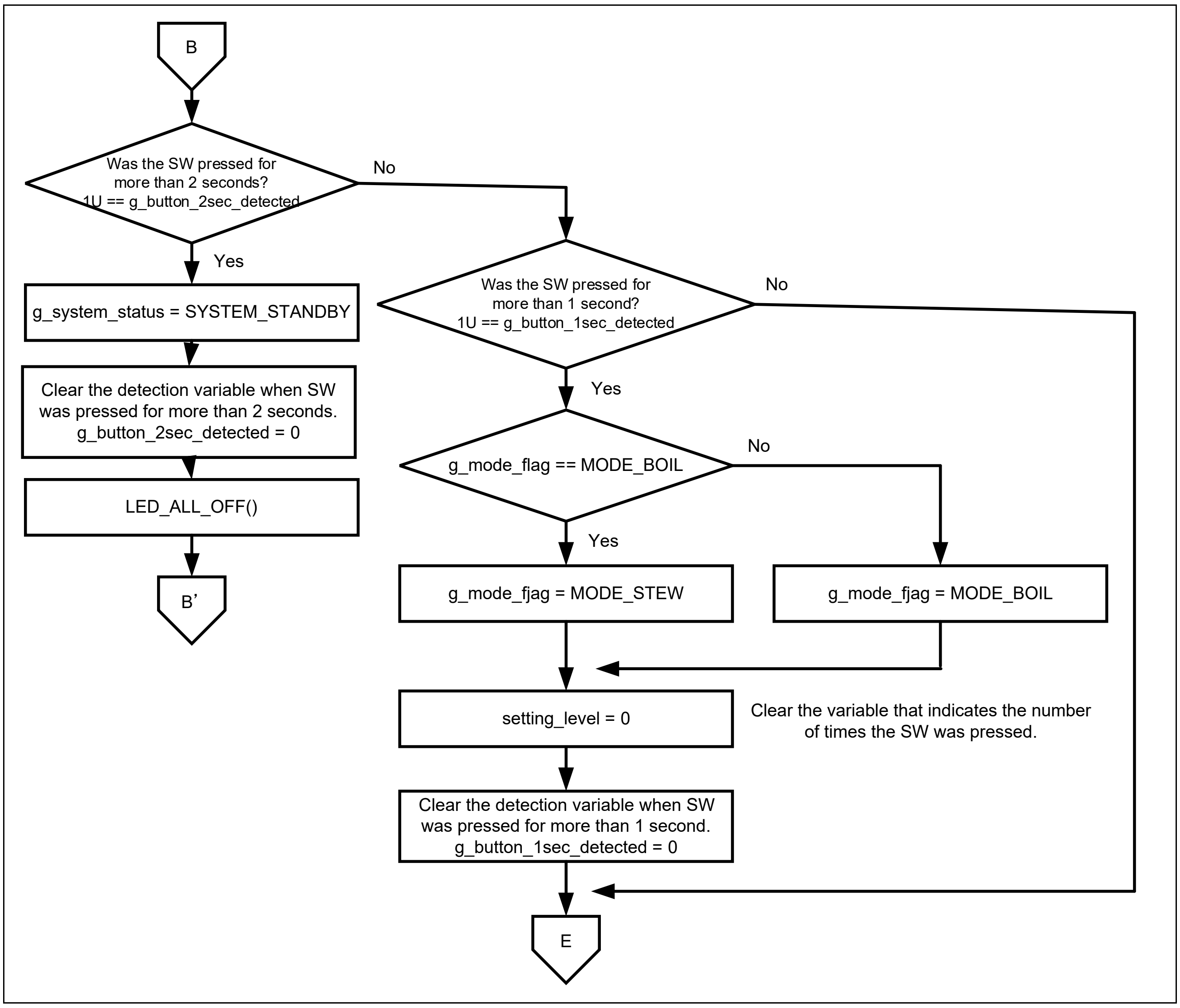
The process of standby mode is shown below.

Figure 4-2 Flowchart of main Functions (2/6)



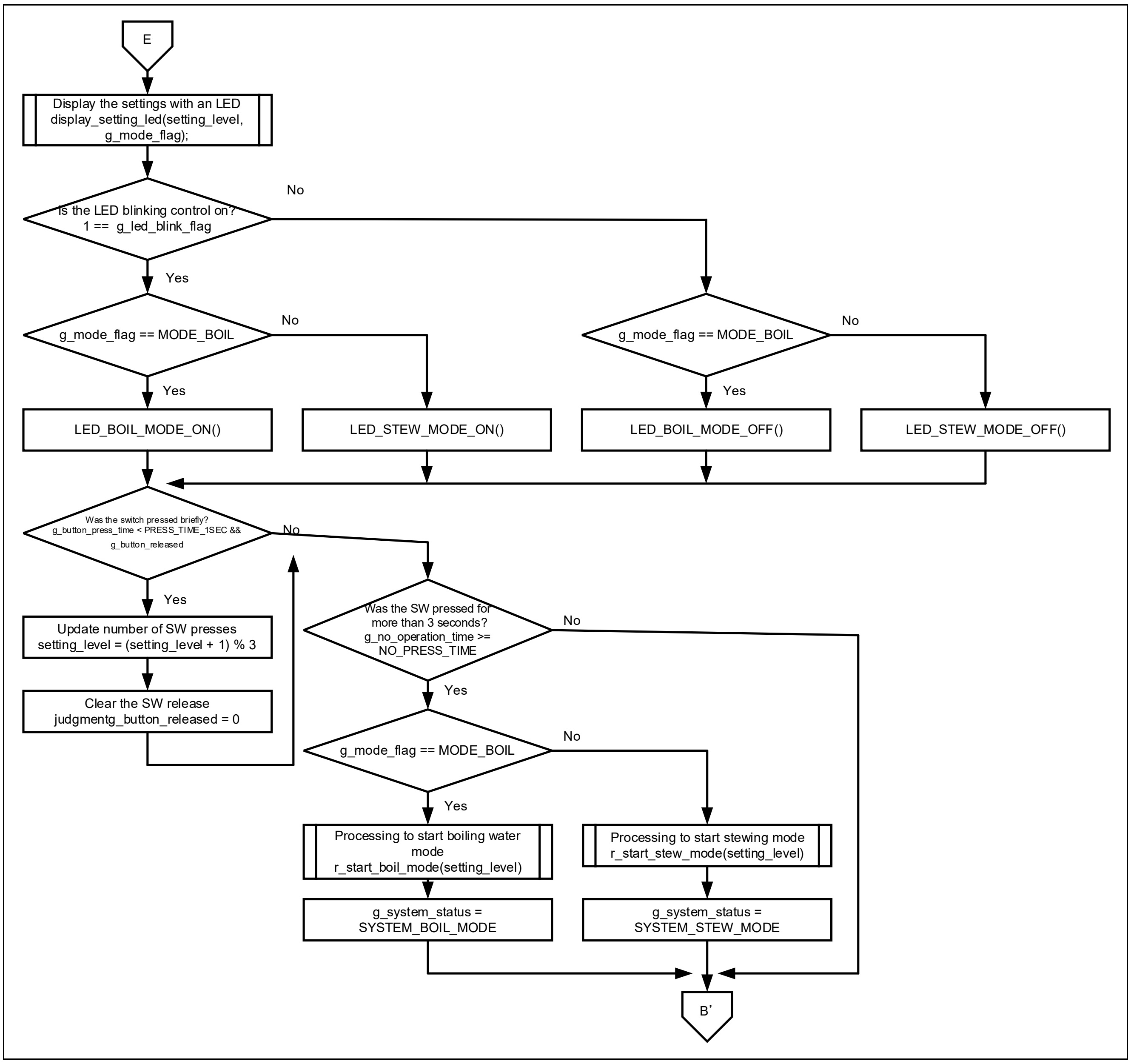
The first half of the boil setting and stew setting process is shown below.

Figure 4-3 Flowchart of main Functions (3/6)


B
Was the SW pressed for more than 2 seconds?
1U == g_button_2sec_detected
No
Yes
g_system_status = SYSTEM_STANDBY
Clear the detection variable when SW was pressed for more than 2 seconds.
g_button_2sec_detected = 0
LED_ALL_OFF()
B'
Was the SW pressed for more than 1 second?
1U == g_button_1sec_detected
No
Yes
g_mode_flag == MODE_BOIL
No
Yes
g_mode_fjag = MODE_STEW
g_mode_fjag = MODE_BOIL
setting_level = 0
Clear the variable that indicates the number of times the SW was pressed.
Clear the detection variable when SW was pressed for more than 1 second.
g_button_1sec_detected = 0
E

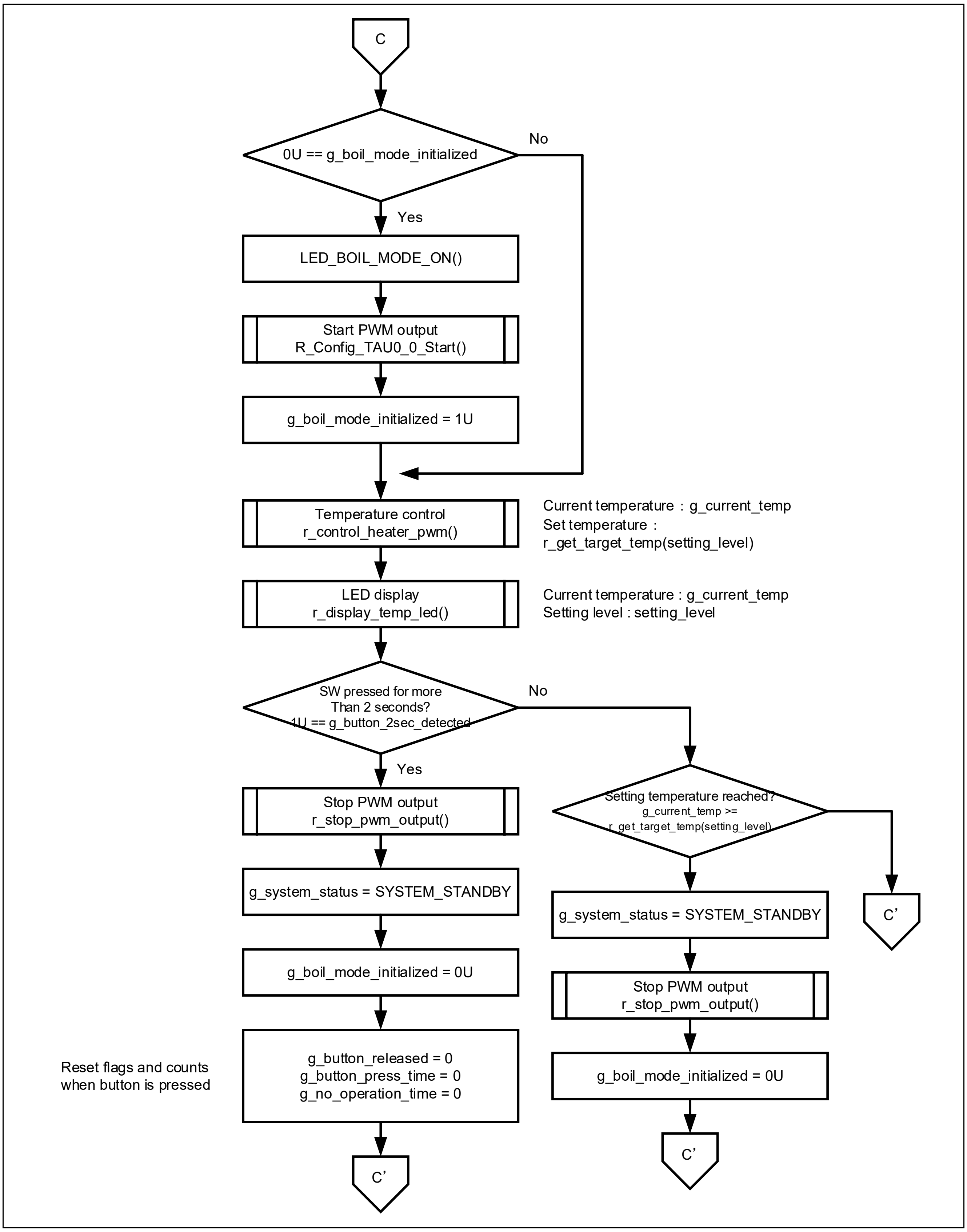
The second half of the boil setting and stew setting processes is shown below.

Figure 4-4 Flowchart of main Functions (4/6)


E
Display the settings with an LED
display_setting_led(setting_level, g_mode_flag);
Is the LED blinking control on?
1 == g_led_blink_flag
No
Yes
g_mode_flag == MODE_BOIL
No
Yes
LED_BOIL_MODE_ON()
LED_STEW_MODE_ON()
g_mode_flag == MODE_BOIL
No
Yes
LED_BOIL_MODE_OFF()
LED_STEW_MODE_OFF()
Was the switch pressed briefly?
g_button_press_time < PRESS_TIME_1SEC &&
g_button_released
No
Yes
Update number of SW presses
setting_level = (setting_level + 1) % 3
Clear the SW release
judgmentg_button_released = 0
Was the SW pressed for more than 3 seconds?
g_no_operation_time >= NO_PRESS_TIME
No
Yes
g_mode_flag == MODE_BOIL
No
Yes
Processing to start boiling water mode
r_start_boil_mode(setting_level)
Processing to start stewing mode
r_start_stew_mode(setting_level)
g_system_status = SYSTEM_BOIL_MODE
g_system_status = SYSTEM_STEW_MODE
B'

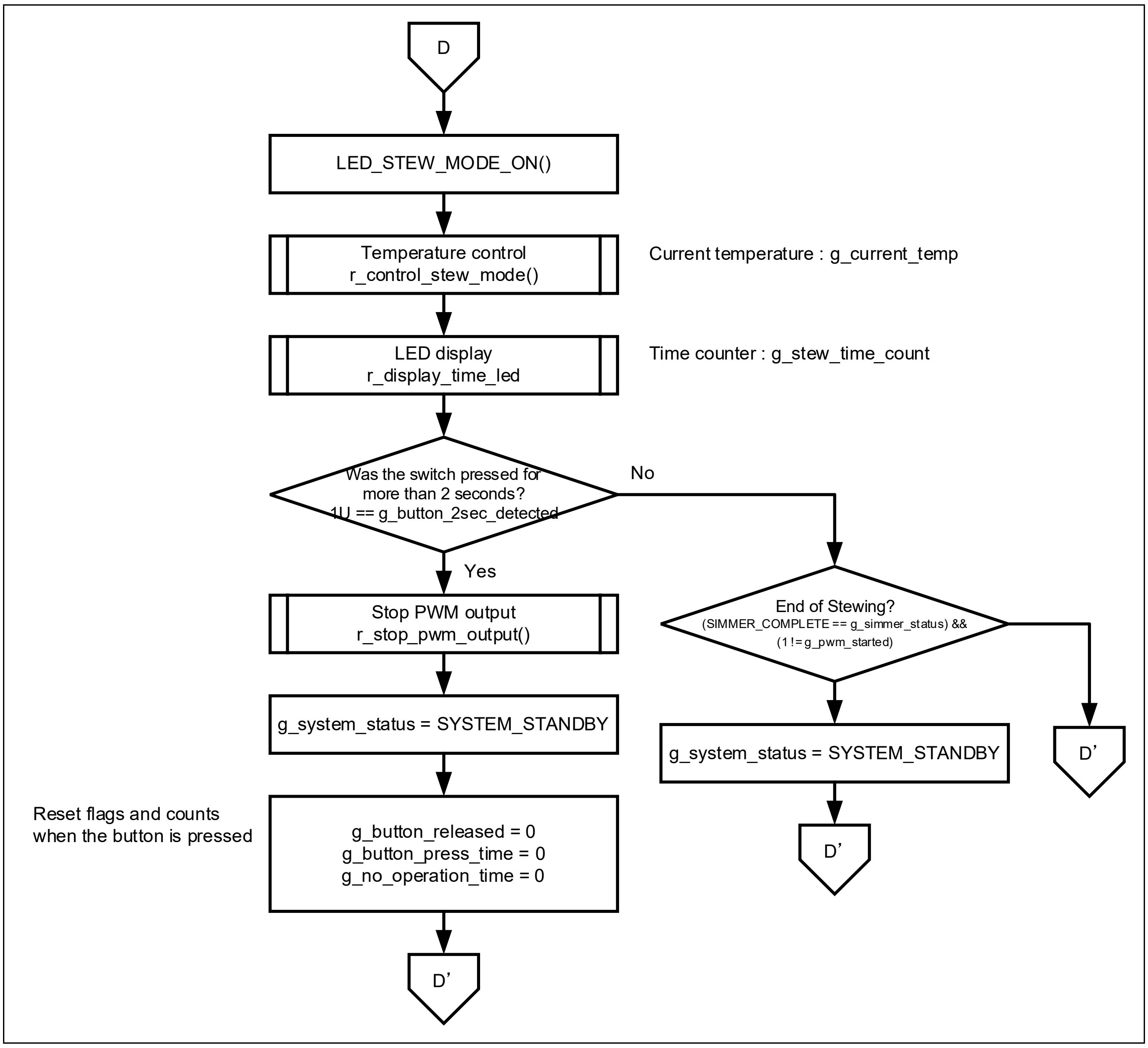
The process of boil mode is shown below.

Figure 4-5 Flowchart of main Functions (5/6)


C
0U == g_boil_mode_initialized
No
Yes
LED_BOIL_MODE_ON()
Start PWM output
R_Config_TAU0_0_Start()
g_boil_mode_initialized = 1U
Temperature control
r_control_heater_pwm()
Current temperature : g_current_temp
Set temperature :
r_get_target_temp(setting_level)
LED display
r_display_temp_led()
Current temperature : g_current_temp
Setting level : setting_level
SW pressed for more
Than 2 seconds?
1U == g_button_2sec_detected
No
Yes
Stop PWM output
r_stop_pwm_output()
g_system_status = SYSTEM_STANDBY
g_boil_mode_initialized = 0U
Reset flags and counts
when button is pressed
g_button_released = 0
g_button_press_time = 0
g_no_operation_time = 0
C'
Setting temperature reached?
g_current_temp >=
r_get_target_temp(setting_level)
g_system_status = SYSTEM_STANDBY
Stop PWM output
r_stop_pwm_output()
g_boil_mode_initialized = 0U
C'
C'

The process of stew mode is shown below.

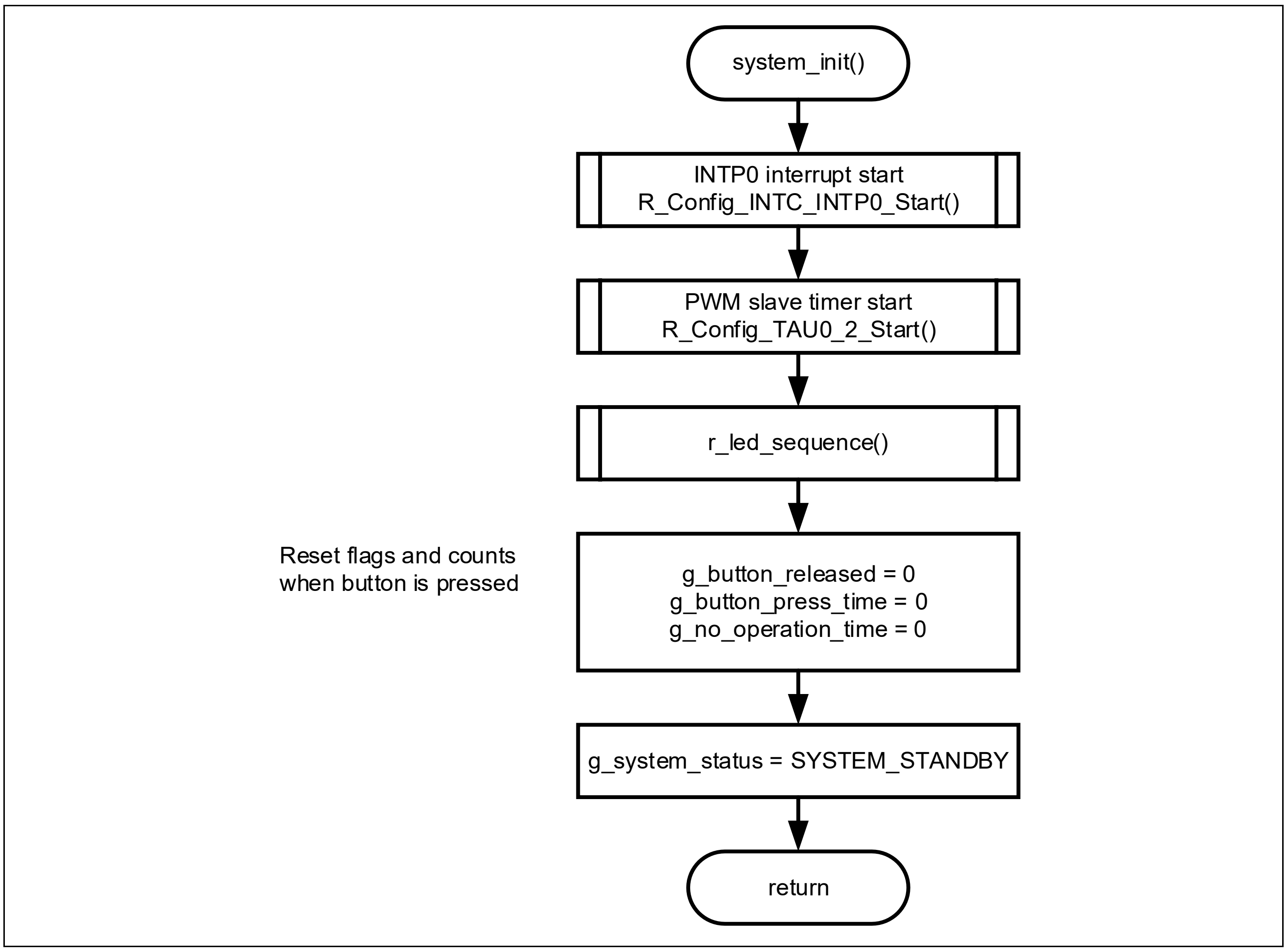
Figure 4-6 Flowchart of main Functions (6/6)


D
LED_STEW_MODE_ON()
Temperature control
r_control_stew_mode()
Current temperature : g_current_temp
LED display
r_display_time_led
Time counter : g_stew_time_count
Was the switch pressed for more than 2 seconds?
1U == g_button_2sec_detected
No
Yes
Stop PWM output
r_stop_pwm_output()
g_system_status = SYSTEM_STANDBY
Reset flags and counts when the button is pressed
g_button_released = 0
g_button_press_time = 0
g_no_operation_time = 0
D’
End of Stewing?
(SIMMER_COMPLETE == g_simmer_status) &&
(1 != g_pwm_started)
g_system_status = SYSTEM_STANDBY
D’
D’

4.9.2 Flowchart of system_init Function

The flowchart of the system_ini function is shown below.

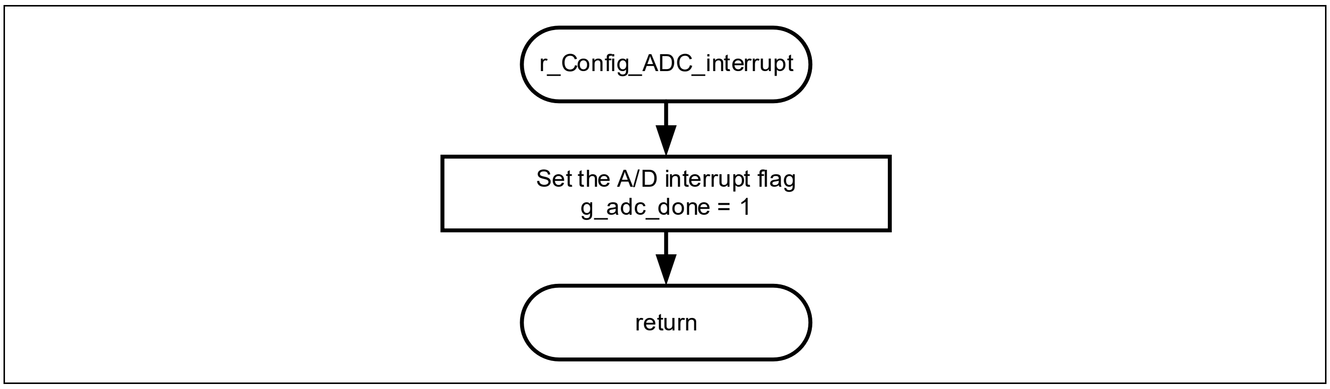
Figure 4-7 Flowchart of system_init Function



4.9.3 Flowchart of r_Config_ADC_interrupt Function

The flowchart of the r_Config_ADC_interrupt function is shown below.

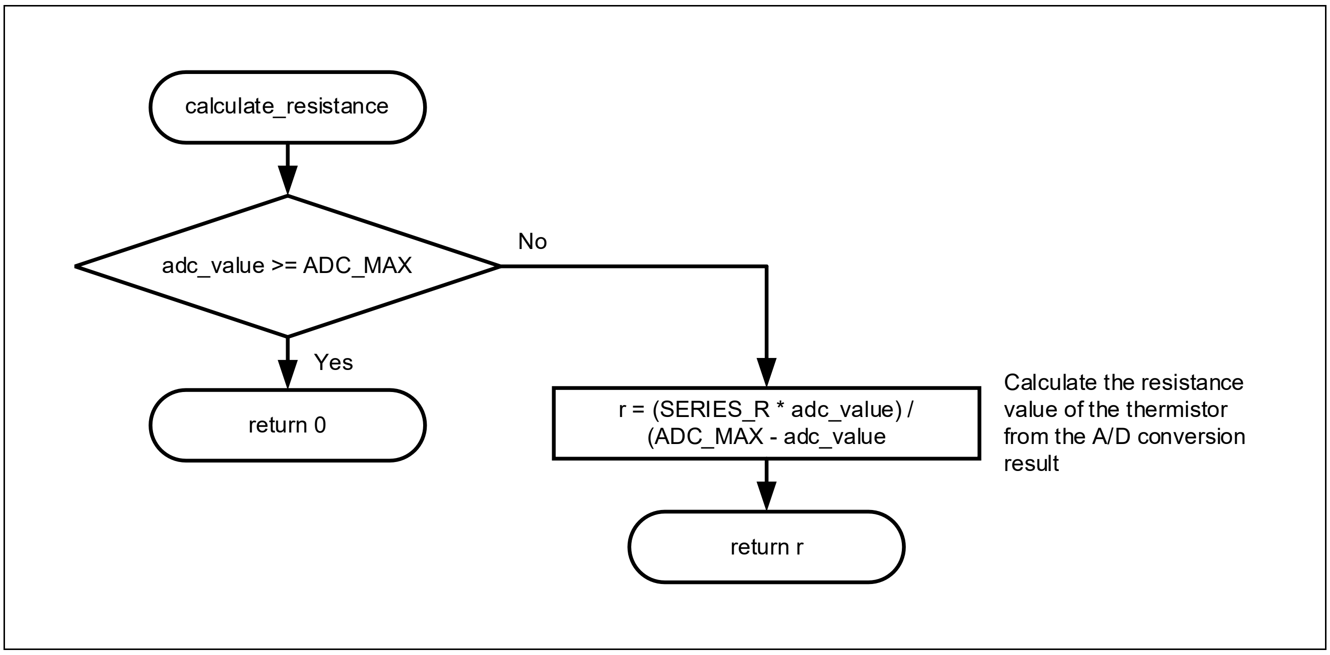
Figure 4-8 Flowchart of r_Config_ADC_interrupt Function



4.9.4 Flowchart of calculate_resistance Function

The flowchart of calculate_resistance function is shown below.

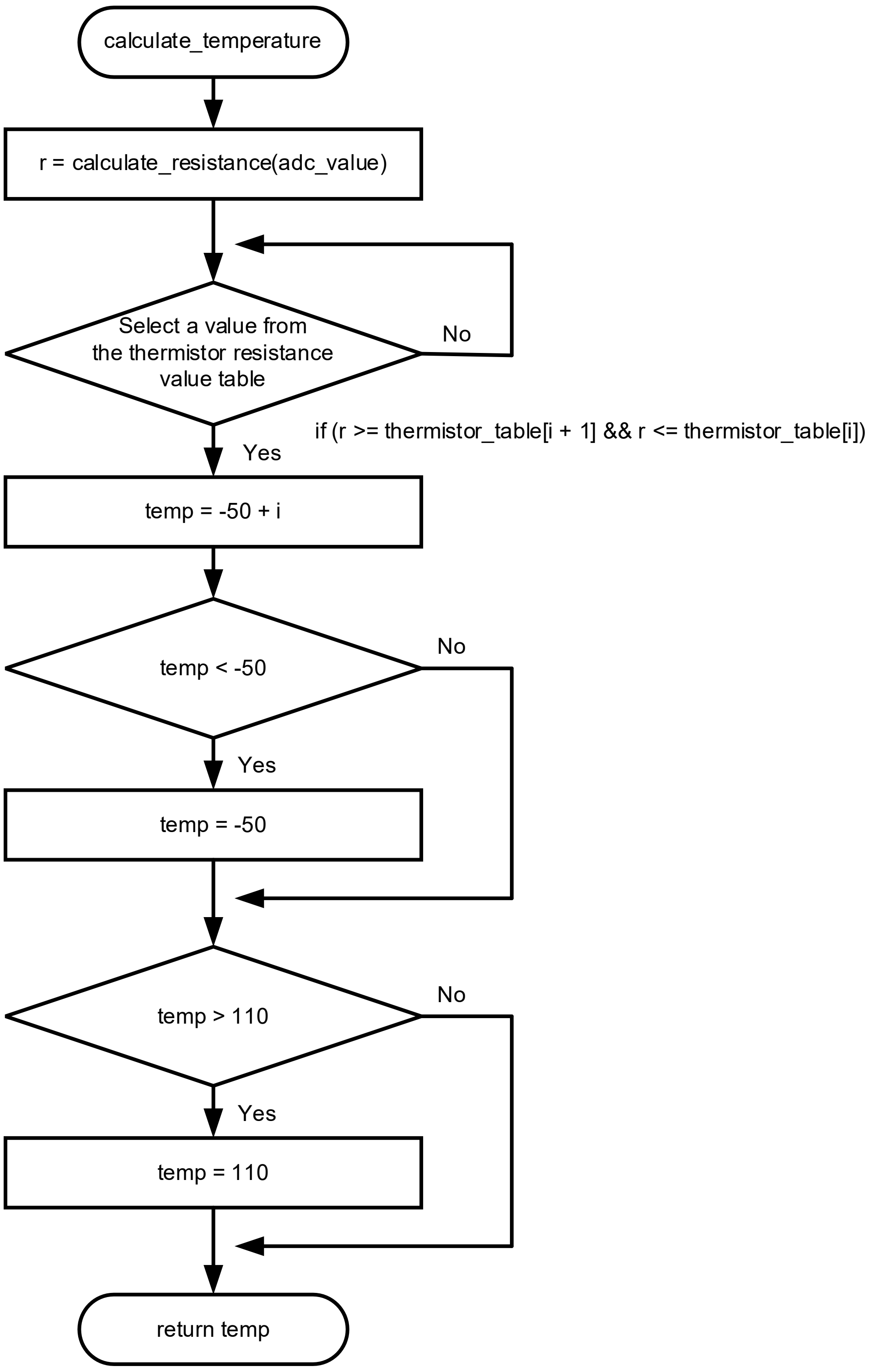
Figure 4-9 Flowchart of calculate_resistance Function


calculate_resistance
adc_value >= ADC_MAX
No
Yes
return 0
r = (SERIES_R * adc_value) / (ADC_MAX - adc_value
Calculate the resistance value of the thermistor from the A/D conversion result
return r

4.9.5 Flowchart of calculate_temperature Function

The flowchart of calculate_temperature function is shown below.

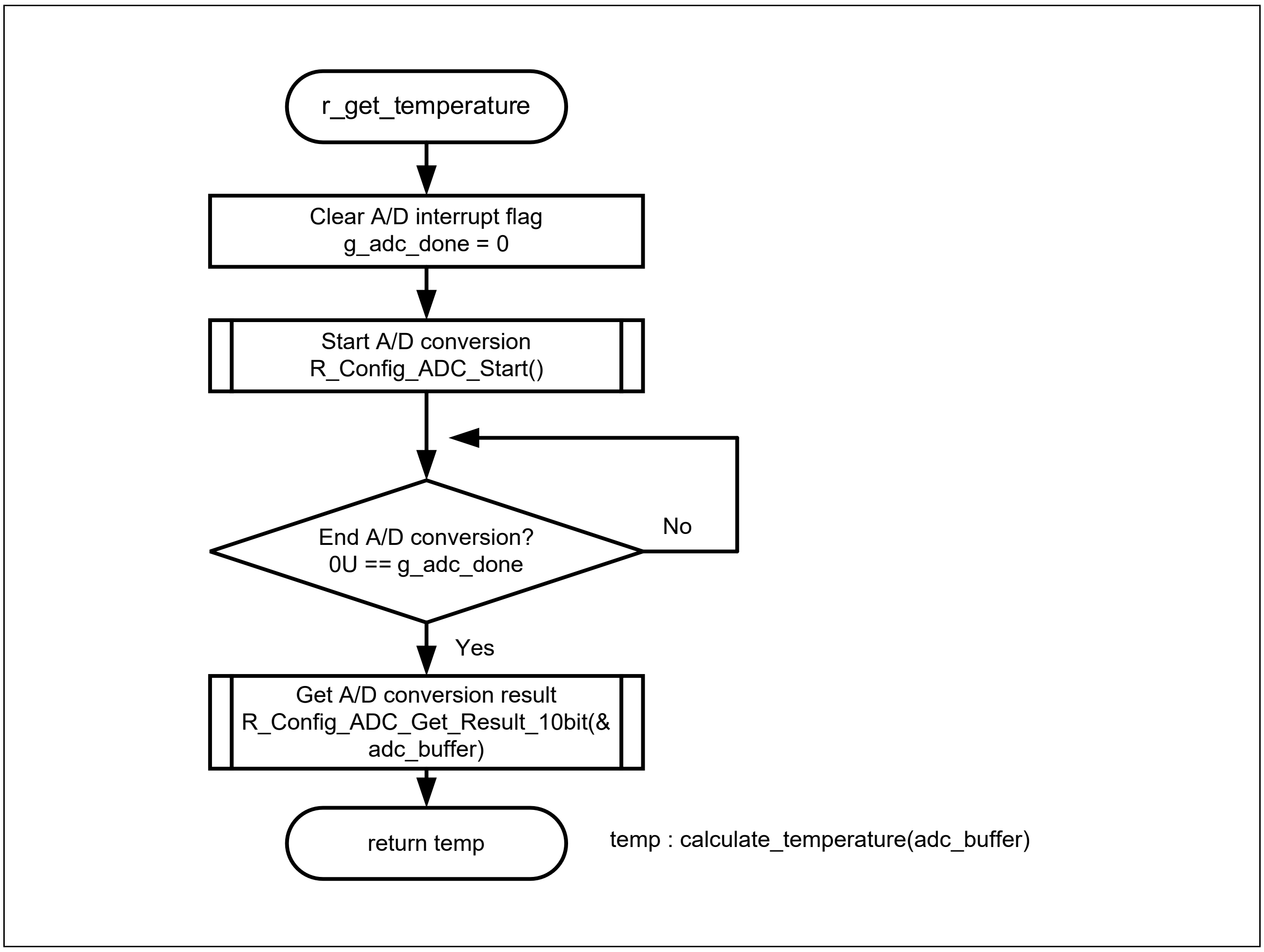
Figure 4-10 Flowchart of calculate_temperature Function


calculate_temperature
r = calculate_resistance(adc_value)
Select a value from the thermistor resistance value table
No
if (r >= thermistor_table[i + 1] && r <= thermistor_table[i])
Yes
temp = -50 + i
temp < -50
No
Yes
temp = -50
temp > 110
No
Yes
temp = 110
return temp

4.9.6 Flowchart of r_get_temperature Function

The flowchart of r_get_temperature function is shown below.

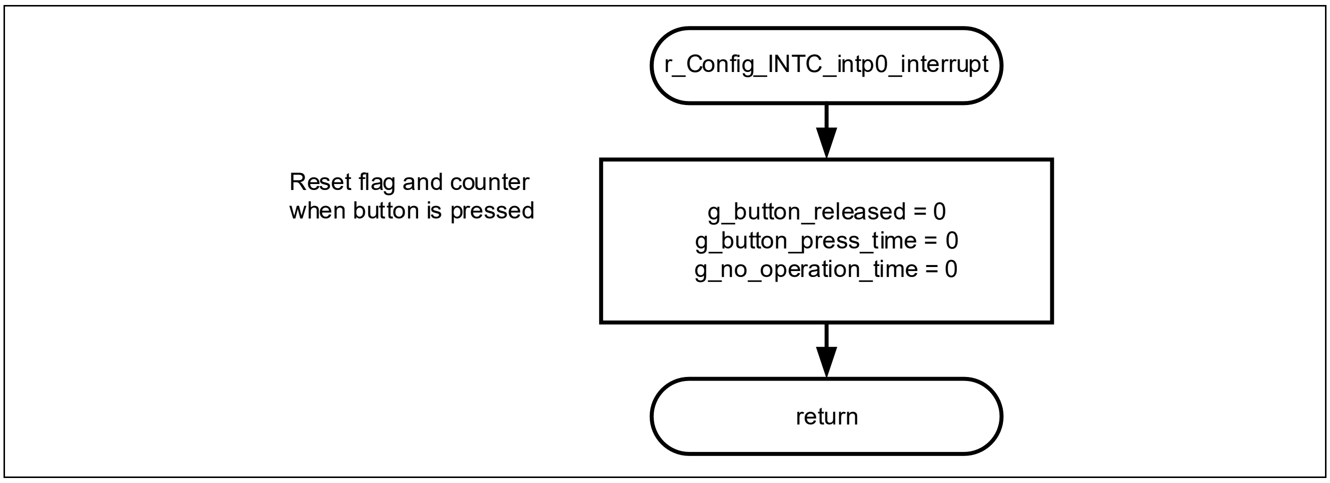
Figure 4-11 Flowchart of r_get_temperature Function



4.9.7 Flowcharts of r_Config_INTC_intp0_interrupt Function

The flowchart of r_Config_INTC_intp0_interrupt function is shown below.

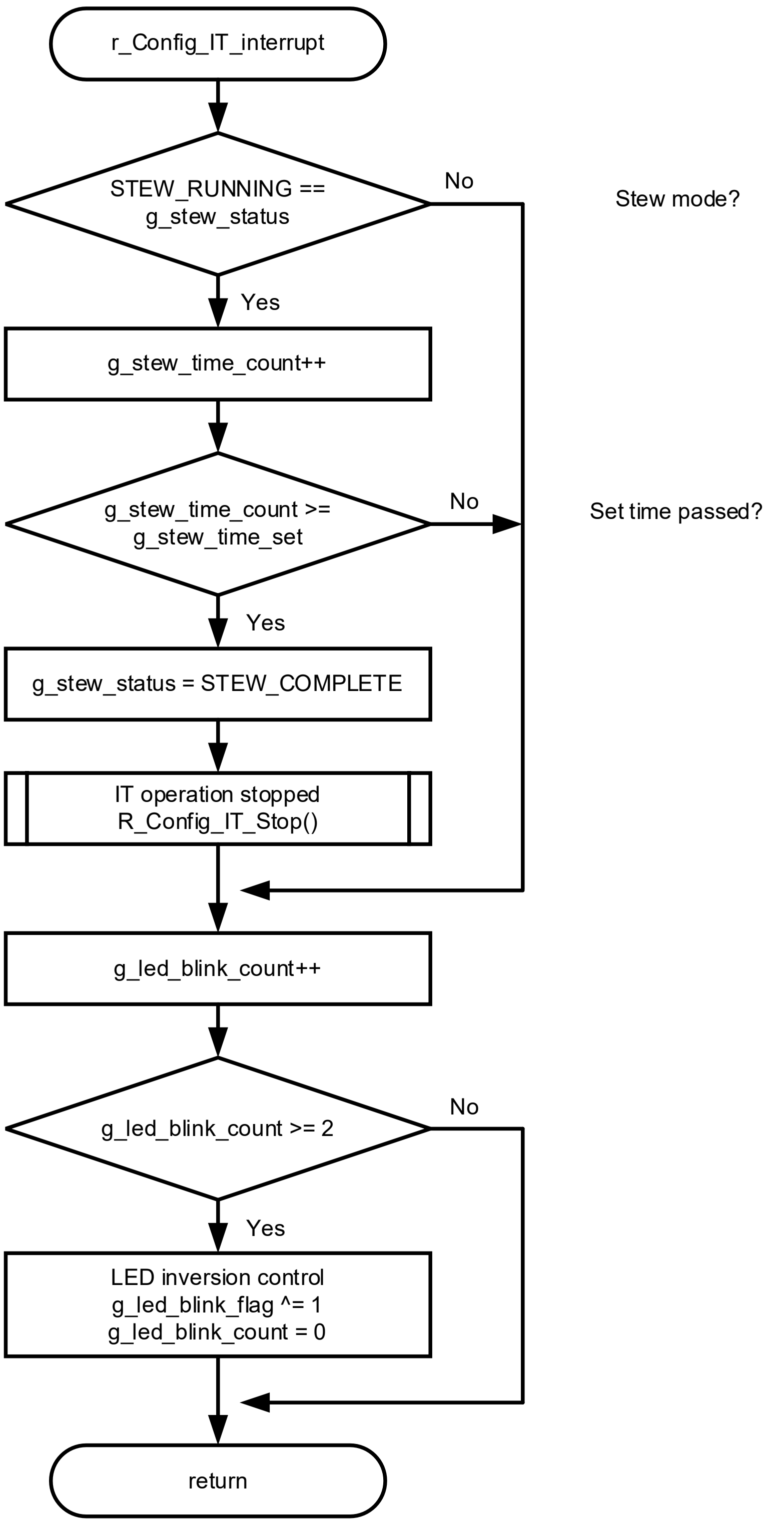
Figure 4-12 Flowchart of r_Config_INTC_intp0_interrupt Function


r_Config_INTC_intp0_interrupt
Reset flag and counter when button is pressed
g_button_released = 0
g_button_press_time = 0
g_no_operation_time = 0
return

4.9.8 Flowchart of r_Config_IT_interrupt Function

The flowchart of r_Config_IT_interrupt function is shown below.

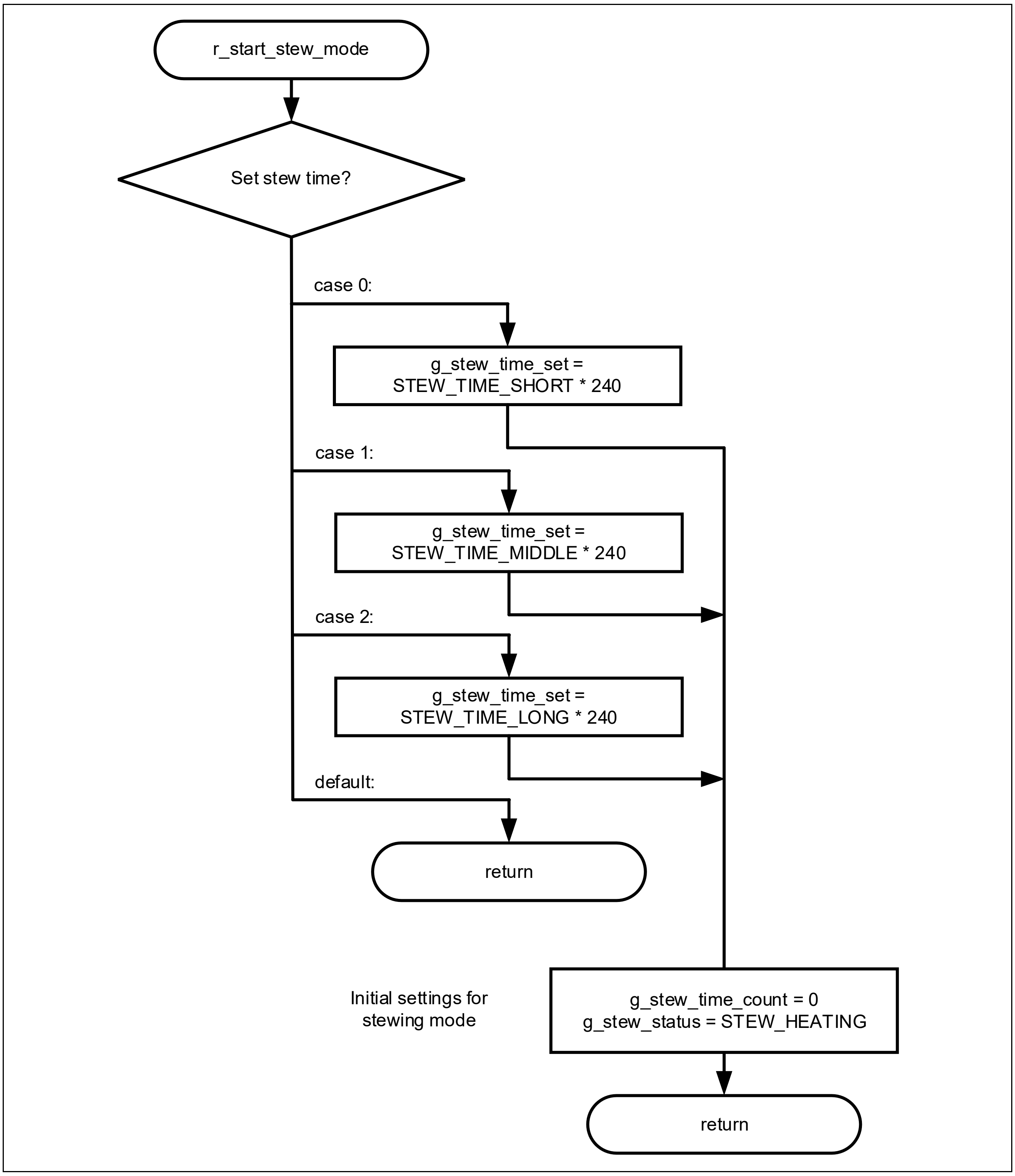
Figure 4-13 Flowchart of r_Config_IT_interrupt Function


r_Config_IT_interrupt
STEW_RUNNING == g_stew_status
No
Stew mode?
Yes
g_stew_time_count++
g_stew_time_count >= g_stew_time_set
No
Set time passed?
Yes
g_stew_status = STEW_COMPLETE
IT operation stopped R_Config_IT_Stop()
g_led_blink_count++
g_led_blink_count >= 2
No
Yes
LED inversion control g_led_blink_flag ^= 1 g_led_blink_count = 0
return

4.9.9 Flowchart of r_start_stew_mode Function

The flowchart of r_start_stew_mode function is shown below.

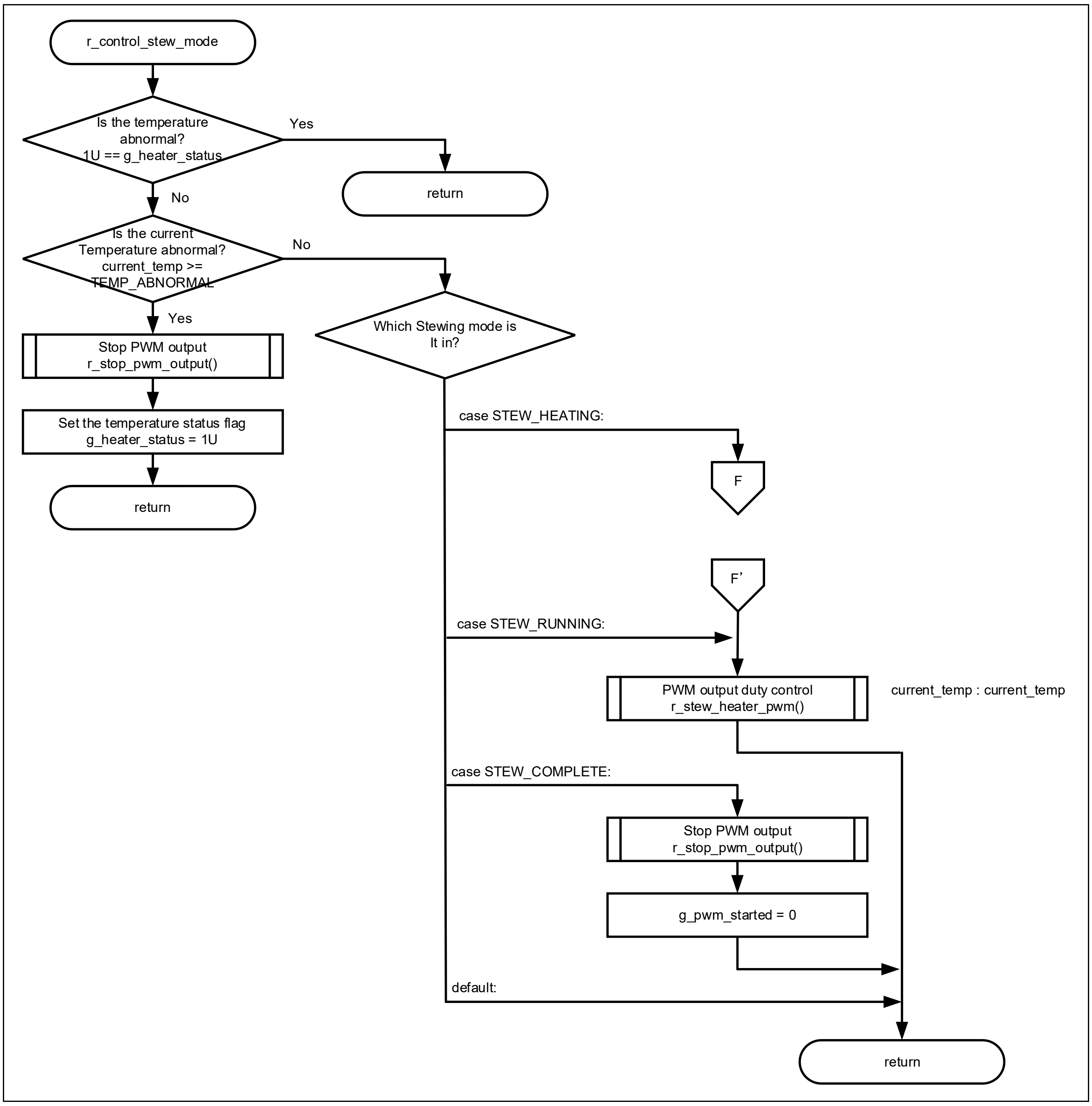
Figure 4-14 Flowchart of r_start_stew_mode Function


r_start_stew_mode
Set stew time?
case 0:
g_stew_time_set = STEW_TIME_SHORT * 240
case 1:
g_stew_time_set = STEW_TIME_MIDDLE * 240
case 2:
g_stew_time_set = STEW_TIME_LONG * 240
default:
return
Initial settings for stewing mode
g_stew_time_count = 0
g_stew_status = STEW_HEATING
return

4.9.10 Flowcharts of r_control_stew_mode Function

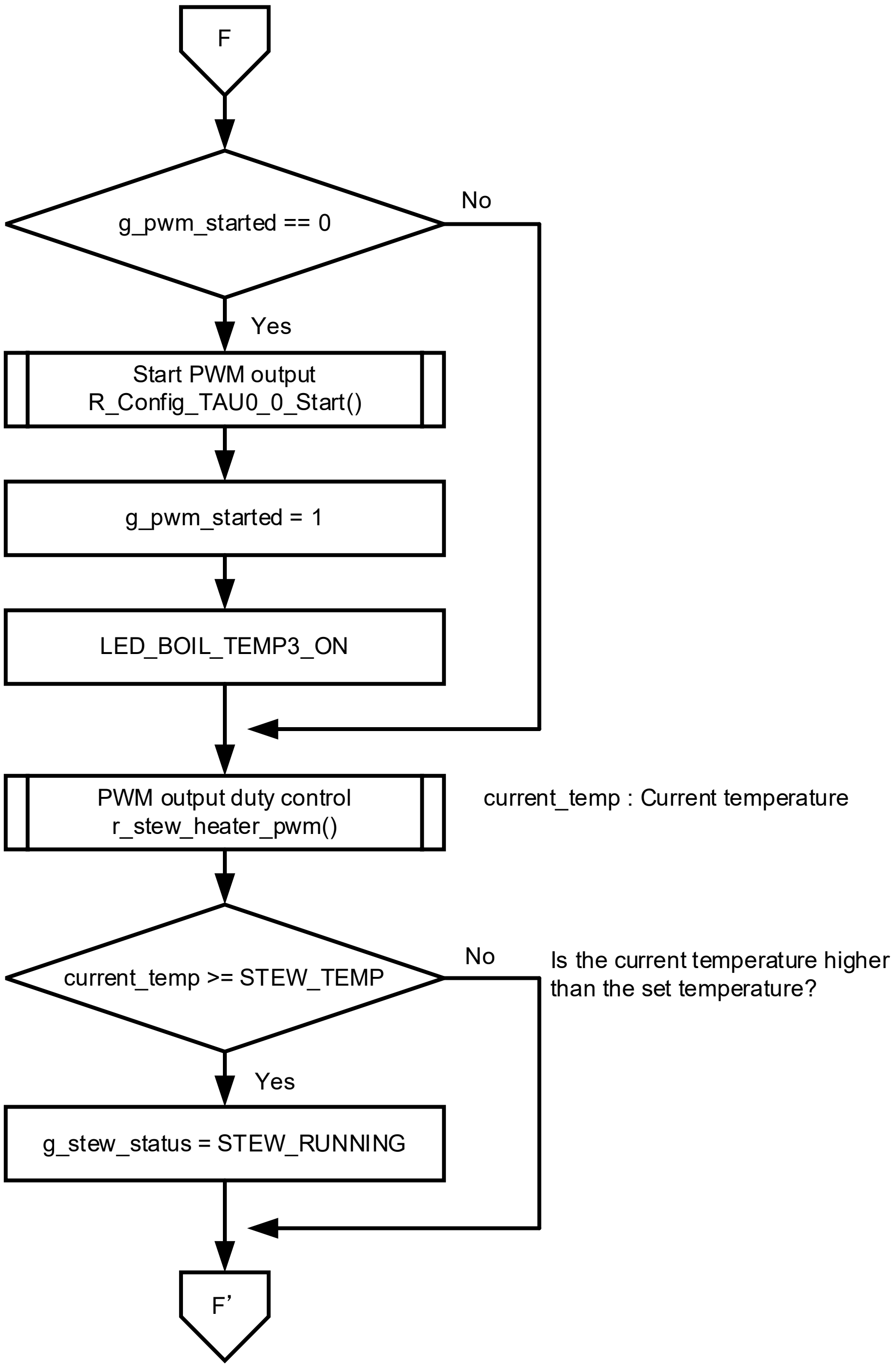
The flowchart of r_control_stew_mode function is shown below.

Figure 4-15 Flowcharts of r_control_stew_mode Function (1/2)


r_control_stew_mode
Is the temperature abnormal? 1U == g_heater_status
Yes
return
No
Is the current Temperature abnormal? current_temp >= TEMP_ABNORMAL
No
Yes
Stop PWM output r_stop_pwm_output()
Set the temperature status flag g_heater_status = 1U
return
Which Stewing mode is It in?
case STEW_HEATING:
F
F'
case STEW_RUNNING:
PWM output duty control r_stew_heater_pwm()
current_temp : current_temp
case STEW_COMPLETE:
Stop PWM output r_stop_pwm_output()
g_pwm_started = 0
default:
return

The processing of stew mode during heating is shown below.

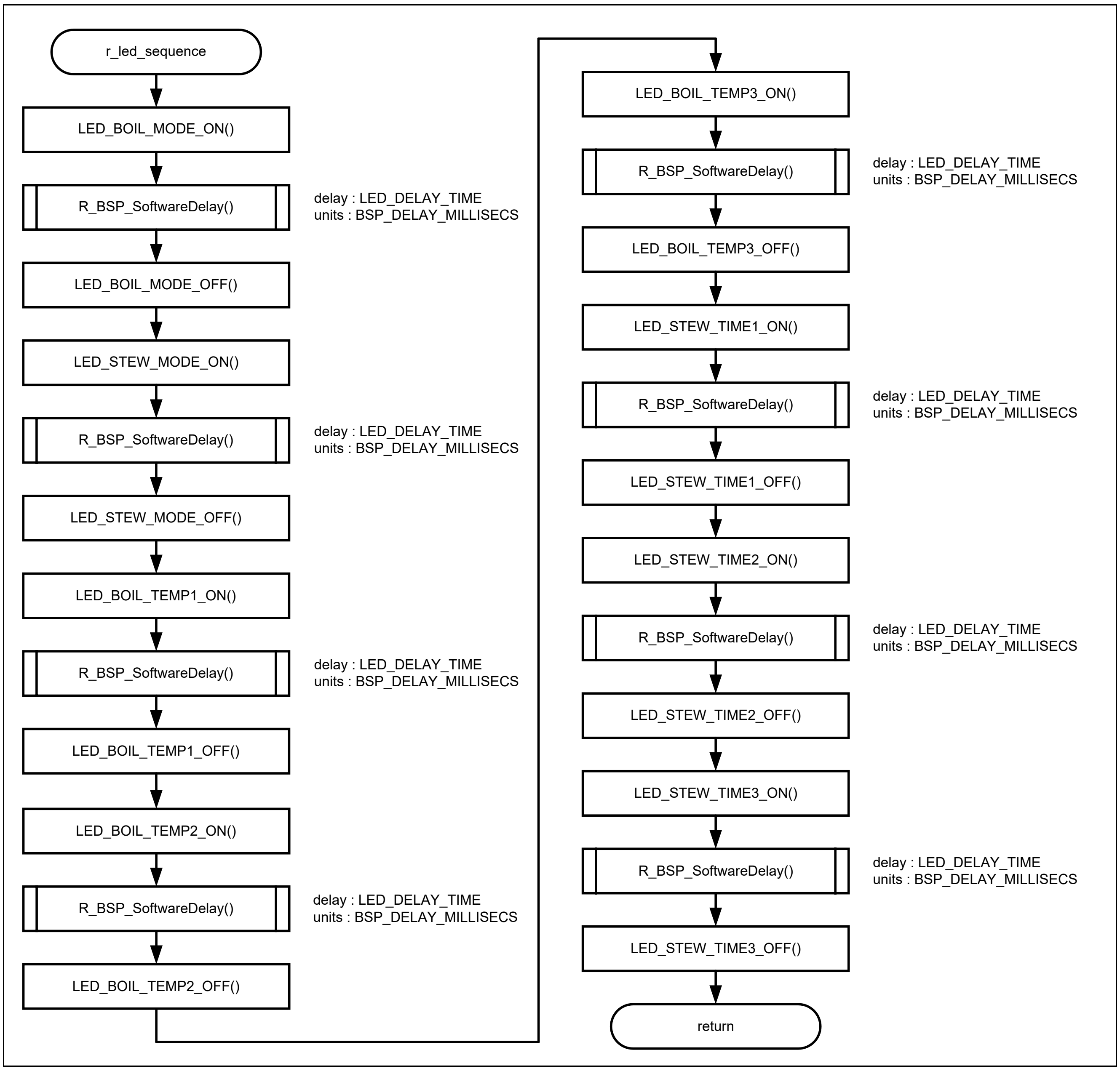
Figure 4-16 Flowchart of r_control_stew_mode Function (2/2)


F
g_pwm_started == 0
No
Yes
Start PWM output
R_Config_TAU0_0_Start()
g_pwm_started = 1
LED_BOIL_TEMP3_ON
PWM output duty control
r_stew_heater_pwm()
current_temp : Current temperature
current_temp >= STEW_TEMP
No
Is the current temperature higher than the set temperature?
Yes
g_stew_status = STEW_RUNNING
F'

4.9.11 Flowchart of r_led_sequence Function

The flowchart of r_led_sequence function is shown below.

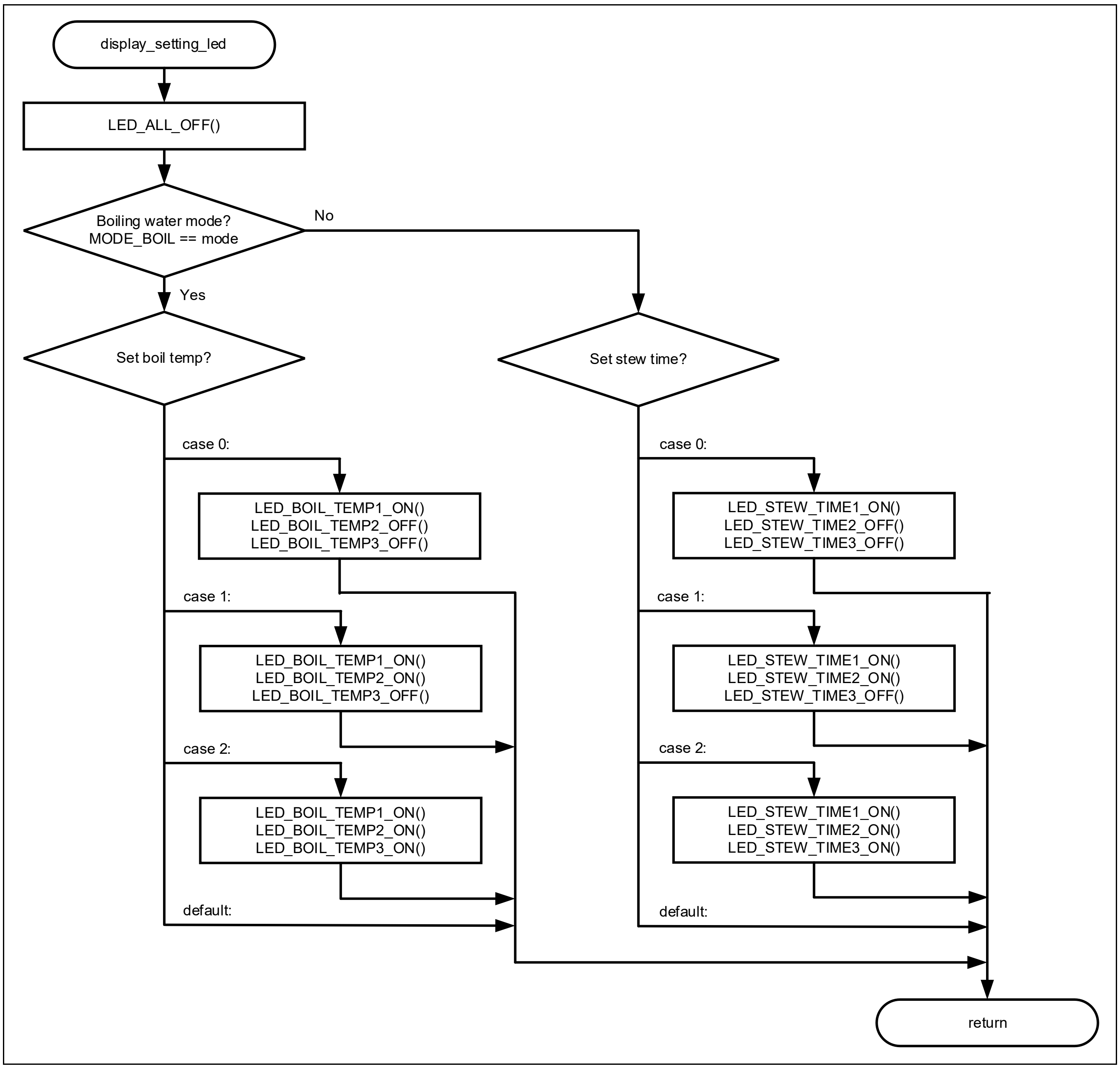
Figure 4-17 Flowchart of r_led_sequence Function



4.9.12 Flowchart of display_setting_led Function

The flowchart of display_setting_led function is shown below.

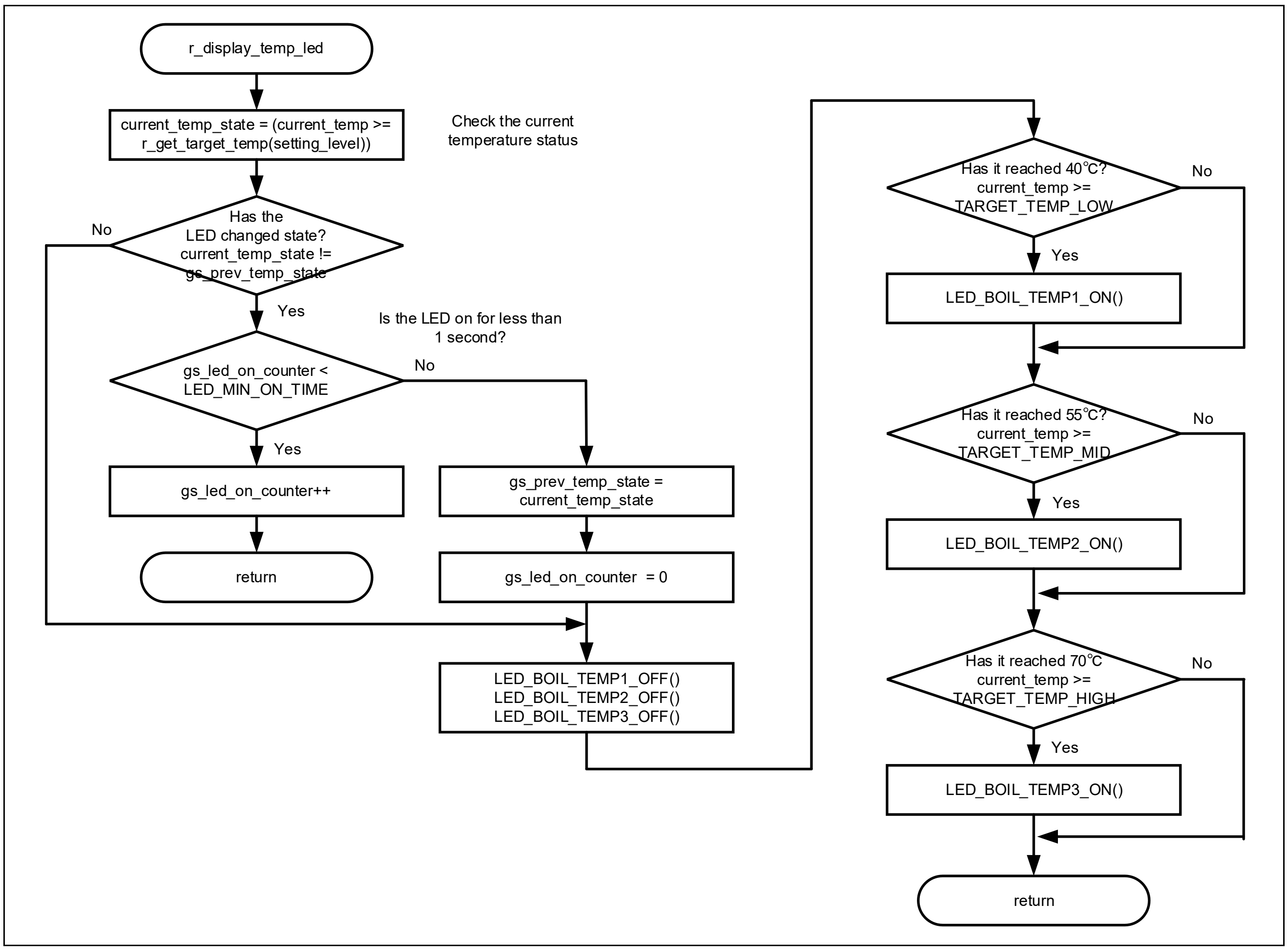
Figure 4-18 Flowchart of display_setting_led Function



4.9.13 Flowchart of r_display_temp_led Function

The flowchart of r_display_temp_led function is shown below.

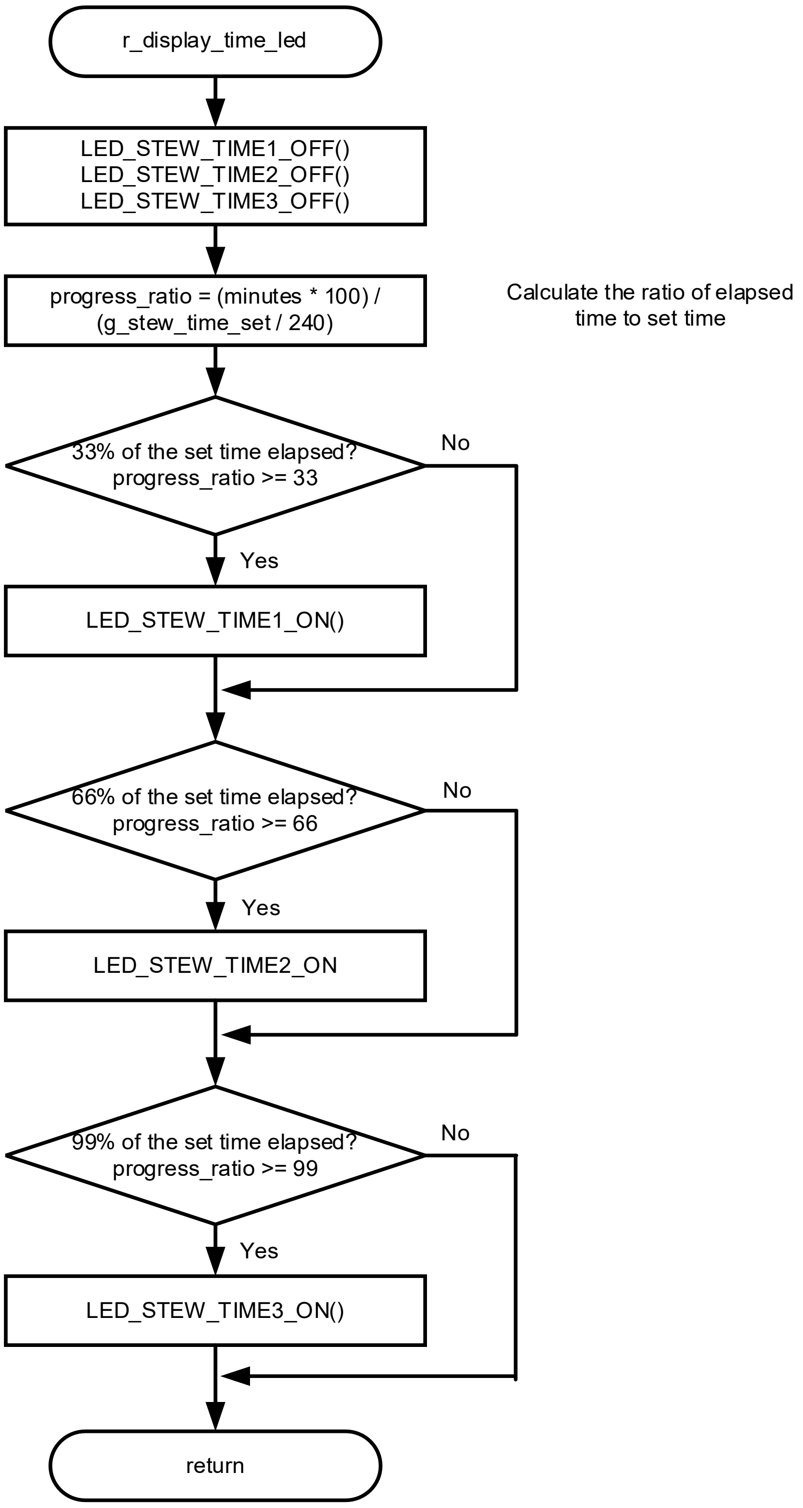
Figure 4-19 Flowchart of r_display_temp_led Function


r_display_temp_led
current_temp_state = (current_temp >= r_get_target_temp(setting_level))
Check the current temperature status
Has the LED changed state? current_temp_state != gs_prev_temp_state
No
Yes
Is the LED on for less than 1 second?
gs_led_on_counter < LED_MIN_ON_TIME
No
Yes
gs_led_on_counter++
return
gs_prev_temp_state = current_temp_state
gs_led_on_counter = 0
LED_BOIL_TEMP1_OFF()
LED_BOIL_TEMP2_OFF()
LED_BOIL_TEMP3_OFF()
Has it reached 40℃? current_temp >= TARGET_TEMP_LOW
No
Yes
LED_BOIL_TEMP1_ON()
Has it reached 55℃? current_temp >= TARGET_TEMP_MID
No
Yes
LED_BOIL_TEMP2_ON()
Has it reached 70℃ current_temp >= TARGET_TEMP_HIGH
No
Yes
LED_BOIL_TEMP3_ON()
return

4.9.14 Flowchart of r_display_time_led Function

The flowchart of r_display_time_led function is shown below.

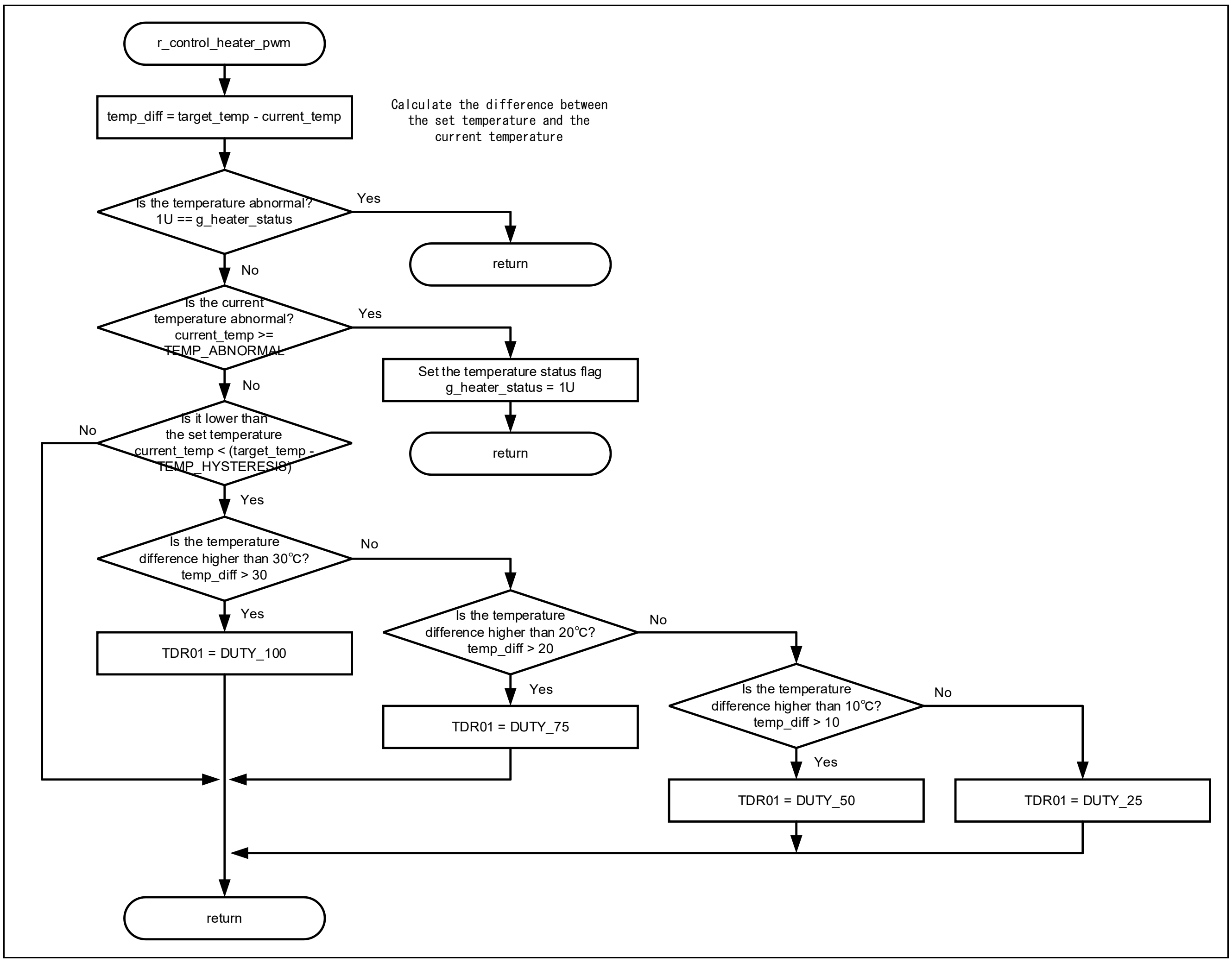
Figure 4-20 Flowchart of r_display_time_led Function


r_display_time_led
LED_STEW_TIME1_OFF()
LED_STEW_TIME2_OFF()
LED_STEW_TIME3_OFF()
progress_ratio = (minutes * 100) / (g_stew_time_set / 240)
Calculate the ratio of elapsed time to set time
33% of the set time elapsed? progress_ratio >= 33
No
Yes
LED_STEW_TIME1_ON()
66% of the set time elapsed? progress_ratio >= 66
No
Yes
LED_STEW_TIME2_ON
99% of the set time elapsed? progress_ratio >= 99
No
Yes
LED_STEW_TIME3_ON()
return

4.9.15 Flowchart of r_control_heater_pwm Function

The flowchart of r_control_heater_pwm function is shown below.

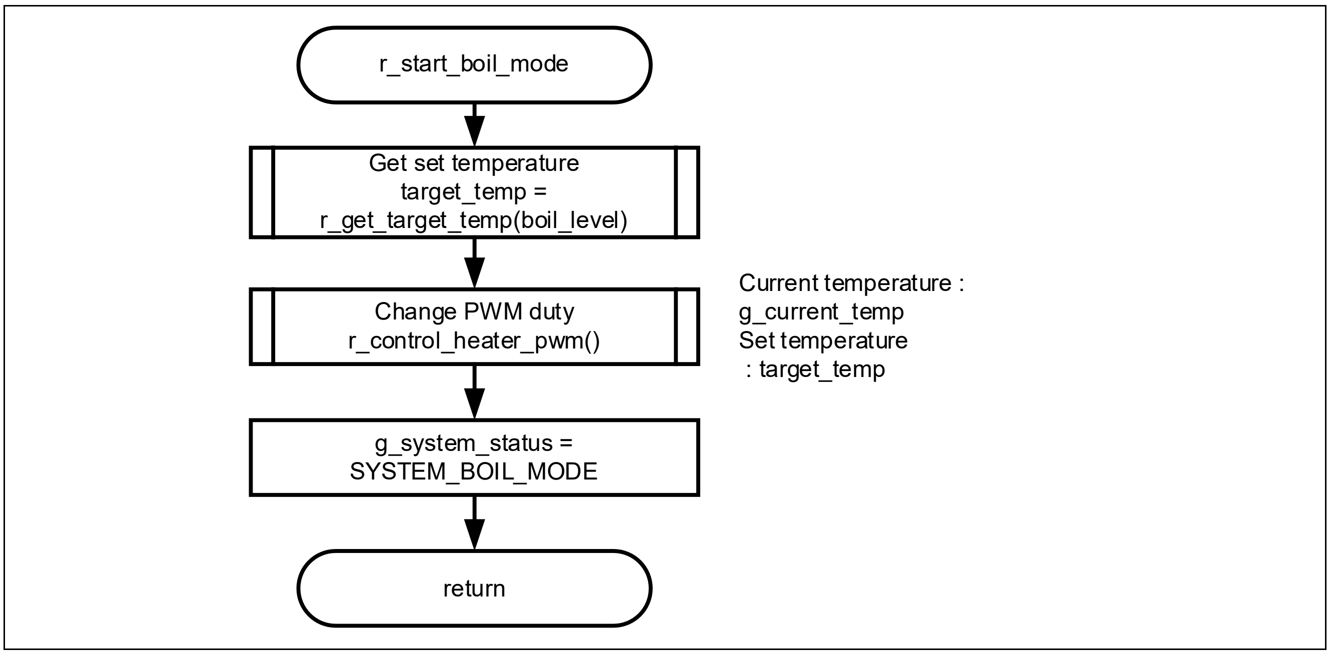
Figure 4-21 Flowchart of r_control_heater_pwm Function


r_control_heater_pwm
temp_diff = target_temp - current_temp
Calculate the difference between the set temperature and the current temperature
Is the temperature abnormal? 1U == g_heater_status
Yes
return
No
Is the current temperature abnormal? current_temp >= TEMP_ABNORMAL
Yes
Set the temperature status flag g_heater_status = 1U
return
No
Is it lower than the set temperature current_temp < (target_temp - TEMP_HYSTERESIS)
No
Yes
Is the temperature difference higher than 30℃? temp_diff > 30
No
Yes
TDR01 = DUTY_100
Is the temperature difference higher than 20℃? temp_diff > 20
No
Yes
TDR01 = DUTY_75
Is the temperature difference higher than 10℃? temp_diff > 10
No
Yes
TDR01 = DUTY_50
TDR01 = DUTY_25
return

4.9.16 Flowchart of r_start_boil_mode Function

The flowchart of r_start_boil_mode function is shown below.

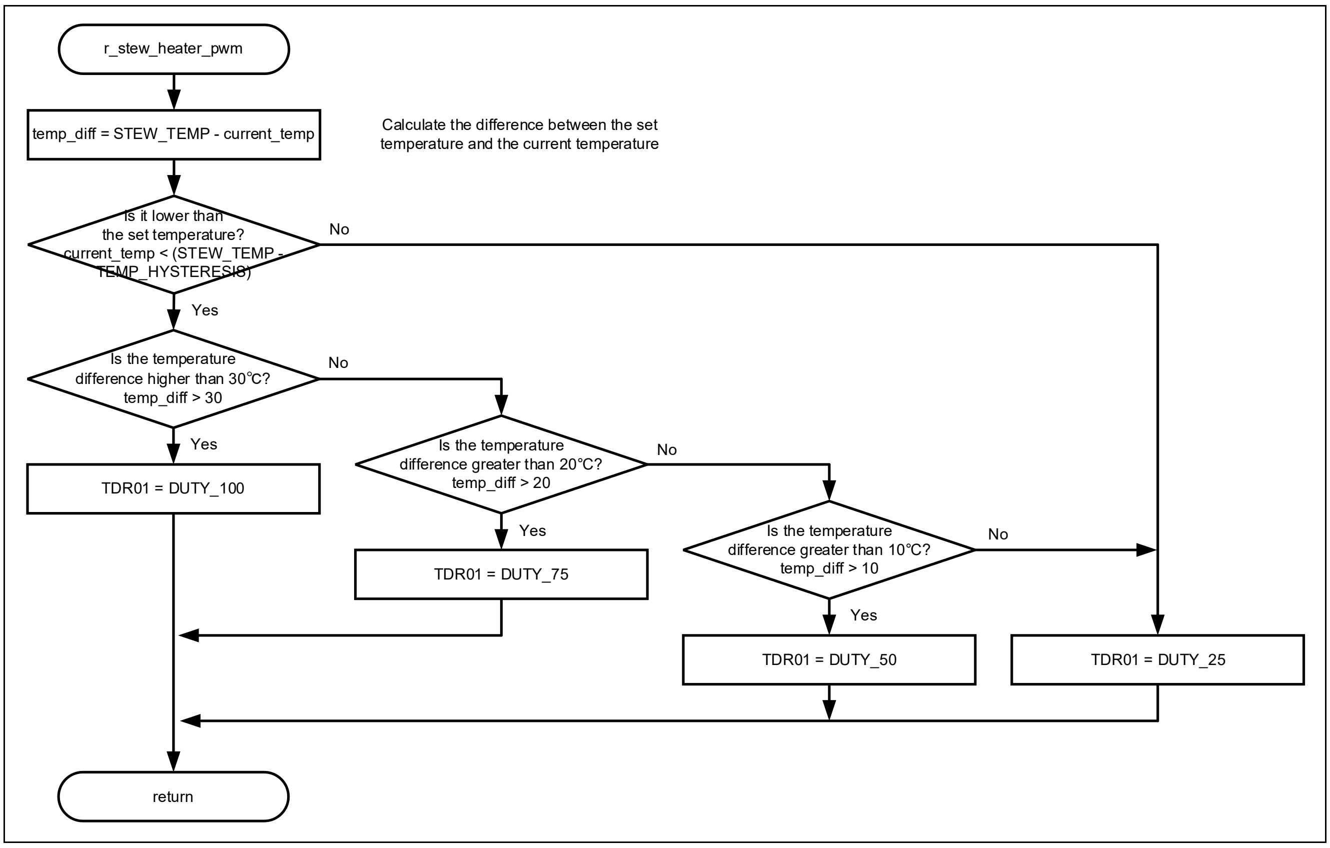
Figure 4-22 Flowchart of r_start_boil_mode Function



4.9.17 Flowchart of r_stew_heater_pwm Function

The flowchart of r_stew_heater_pwm function is shown below.

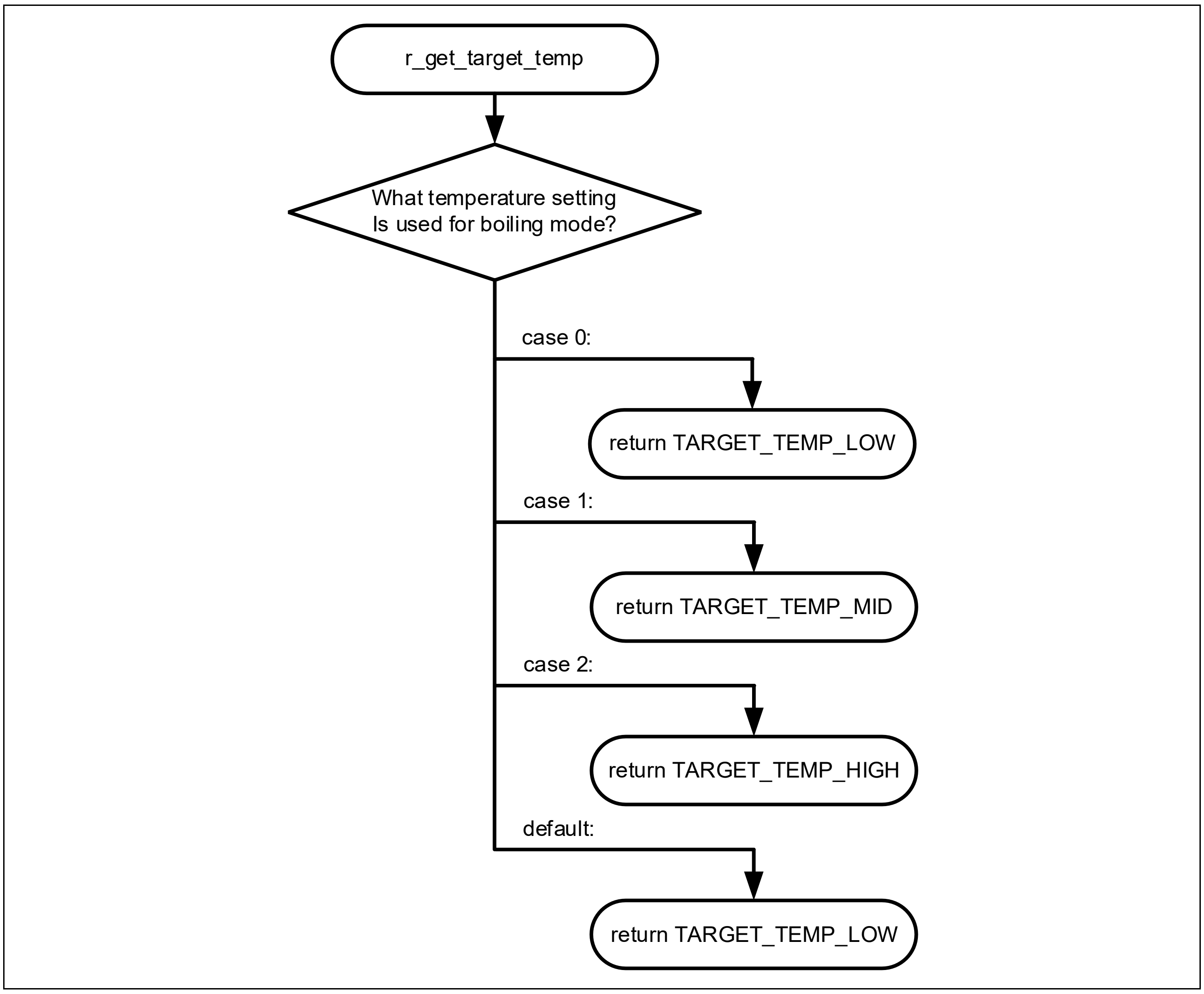
Figure 4-23 Flowchart of r_stew_heater_pwm Function


r_stew_heater_pwm
temp_diff = STEW_TEMP - current_temp
Calculate the difference between the set temperature and the current temperature
Is it lower than the set temperature? current_temp < (STEW_TEMP - TEMP_HYSTERESIS)
No
Yes
Is the temperature difference higher than 30°C? temp_diff > 30
No
Yes
TDR01 = DUTY_100
Is the temperature difference greater than 20°C? temp_diff > 20
No
Yes
TDR01 = DUTY_75
Is the temperature difference greater than 10°C? temp_diff > 10
No
Yes
TDR01 = DUTY_50
TDR01 = DUTY_25
return

4.9.18 Flowchart of r_get_target_temp Function

The flowchart of r_get_target_temp function is shown below.

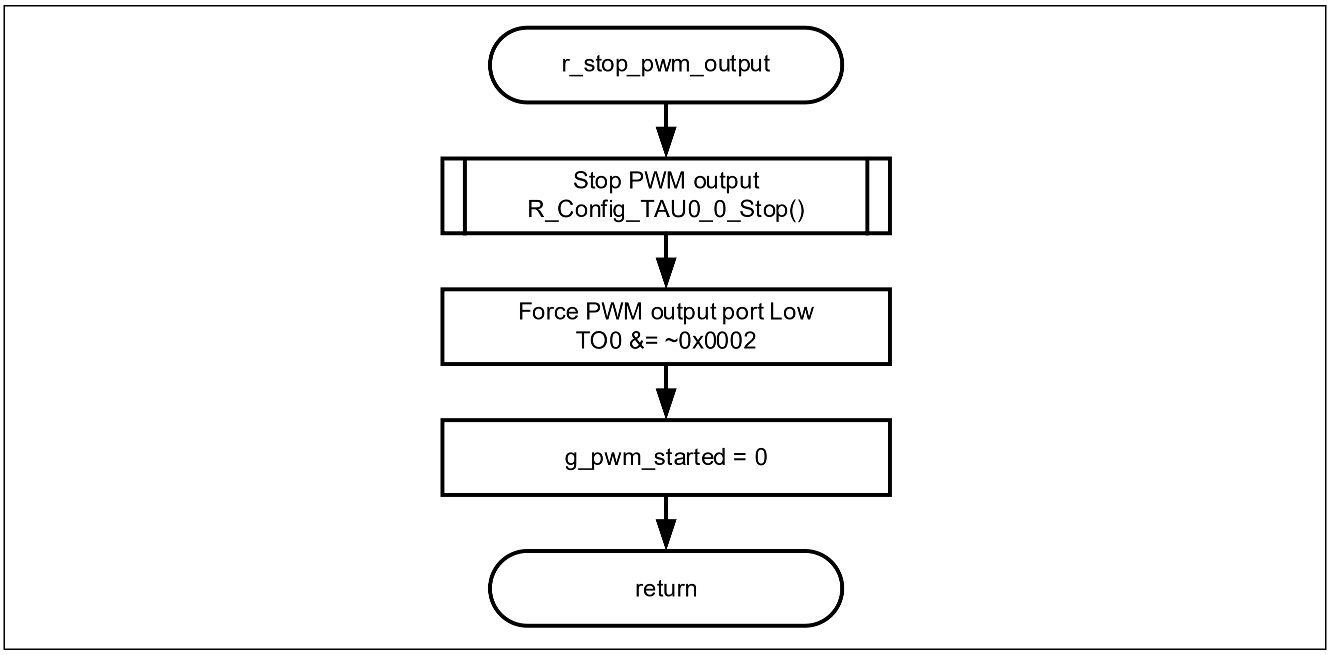
Figure 4-24 Flowchart of r_get_target_temp Function



4.9.19 Flowchart of r_stop_pwm_output Function

The flowchart of r_stop_pwm_output function is shown below.

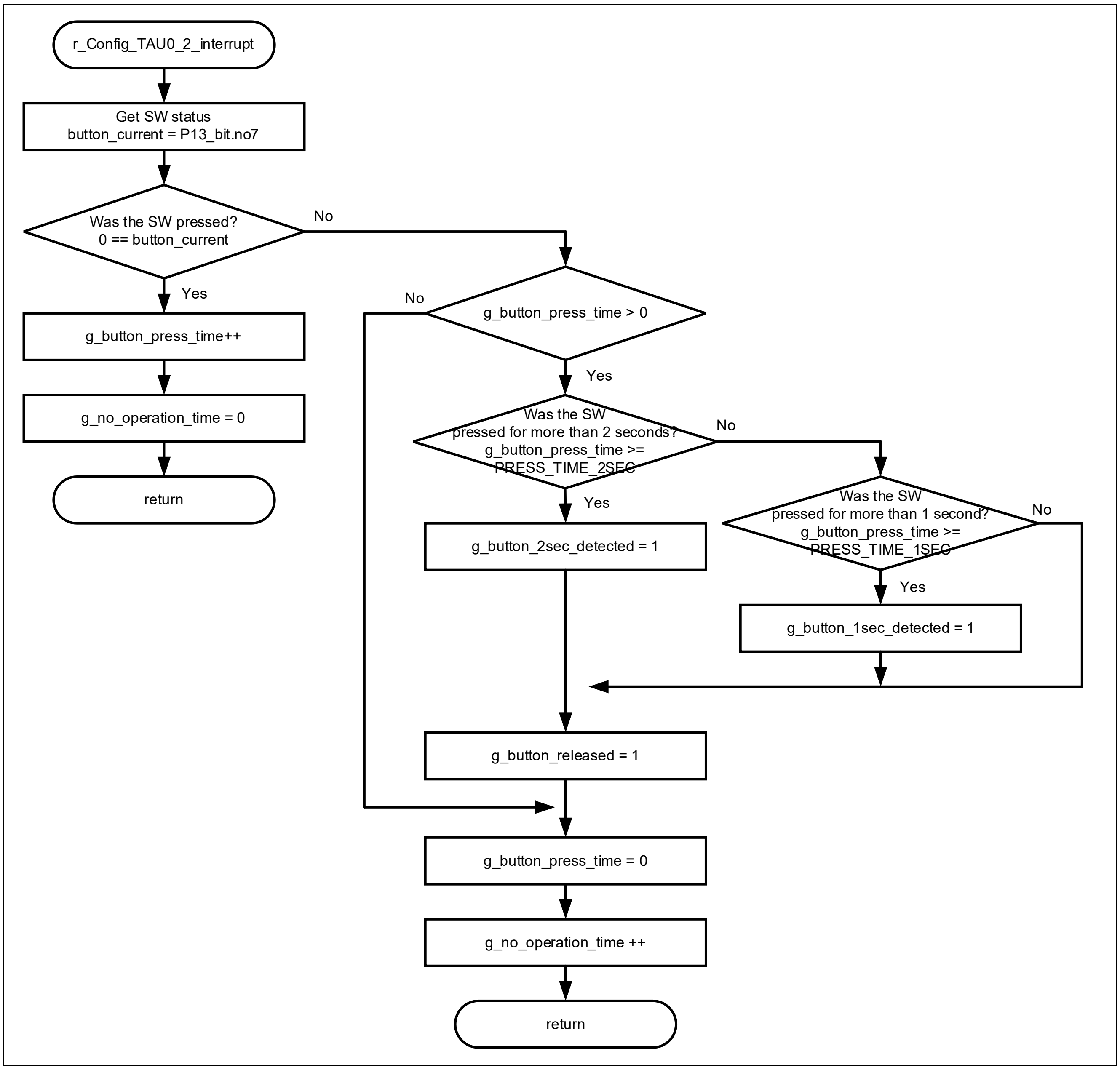
Figure 4-25 Flowchart of r_stop_pwm_output Function



4.9.20 Flowchart of r_Config_TAU0_2_interrupt Function

The flowchart of r_Config_TAU0_2_interrupt function is shown below.

Figure 4-26 Flowchart ofr_Config_TAU0_2_interrupt Function


r_Config_TAU0_2_interrupt
Get SW status
button_current = P13_bit.no7
Was the SW pressed?
0 == button_current
No
Yes
g_button_press_time++
g_no_operation_time = 0
return
g_button_press_time > 0
No
Yes
Was the SW pressed for more than 2 seconds? g_button_press_time >= PRESS_TIME_2SEC
No
Yes
g_button_2sec_detected = 1
Was the SW pressed for more than 1 second? g_button_press_time >= PRESS_TIME_1SEC
No
Yes
g_button_1sec_detected = 1
g_button_released = 1
g_button_press_time = 0
g_no_operation_time ++
return

4.10 Application Examples

This application note contains the following Smart Configurator configuration file in addition to the sample code :

r01an7611_rl78g15_heater_control.scfg

The description, configuration examples, and precautions for using this file are provided below.

4.10.1 r01an7611_rl78g15_heater_control.scfg

This is the Smart Configurator configuration file used in sample code. It includes all functionalities configurated in the Smart Configurator. The sample code setting is as follows :

Table 4-13 Smart Configurator Setting

Tag Name	Component	Description
Clock	-	Operation mode：High-speed main mode 2.7 (V) ~ 5.5 (V) EVDD setting：2.7V≦EVDD0＜5.5V High-Speed On-Chip Oscillator : 16MHz Low-Speed On-Chip Oscillator : 15kHz Used for supplying clock to the 12-bit interval timer fMAIN : 16MHz fCLK : 16000kHz (High-Speed On-Chip Oscillator) fIL : 15kHz (Low-Speed On-Chip Oscillator）
System	-	On-Chip Debug Operation Setting : COM Port Pseudo RRM/DMM Function Setting : Enabled Security ID Setting : Configurated Security ID : 0x00000000000000000000 $\overline{\text{RESET}}$ Pin Setting : Enabled Reset Voltage : 2.52V
Component	r_bsp	Start up select : Enable (use BSP startup) Initialization of peripheral functions by Code Generator/Smart Configurator : Enable API functions disable (R_BSP_StartClock,R_BSP_StopCloc) : Disable API functions disable (R_BSP_GetFclkFreqHz) : Enable API functions disable (R_BSP_SetClockSource) : Disable API functions disable (R_BSP_ChangrClockSetting) : Disable API functions disable (R_BSP_SoftwareDelay) : Enable Parameter check enable : Enable Enable user warm start callback (PRE) : Unused Enable user warm start callback (POST) : Unused Watchdog Timer refresh enable : Unused
	Config_INTC	Component : Interrupt Controller INTP0 Setting : INTP0 Enabled Edge : Falling Edge Priority Levels : Level 3 (Low Priority）

Table 4-14 Smart Configurator Setting

Tag Name	Component	Description
	Config_IT	Component : Interval Timer Operating Mode : 12-bit Counter Mode Resource : IT Interval Time : 250ms Interrupt Setting : Enabled Priority : Level 3 (Low Priority)
	Config_TAU0_0	Component : PWM Output Resource : TAU0_0 Operating Clock : CK00 Clock Source : f_{CLK} Period Setting : 1ms Interrupt Setting : Enabled Priority : Level 3 (Low Priority) PWM Slave Selection : Channel 1 Slave (Slave 1) PWM Duty Cycle : 50% Initial Output Value : 0 Output Level : Active High Interrupt Setting : Enabled Priority : Level 3 (Low Priority)
	Config_TAU0_2	Component : Interval timer Operating Mode : 16-bit Counter Mode Resource : TAU0_2 Operating Clock : CK01 Clock Source : f_{CLK} / 2^8 Interval Time : 10ms No INTTM02 Interrupt Trigger on Start Interrupt Setting : Enabled Priority : Level 3 (Low Priority)
	Config_ADC	Component : A/D Converter Resource : ADC Comparator Operation : Enabled Resolution Setting : 10-bit A/D Chanel Selection : AN1 Conversion Time Mode : Standard 1 Conversion Time : 184 / f_{CLK} Interrupt Setting : Enabled Priority : Level 3 (Low Priority)

Table 4-15 Smart Configurator Setting

Tag Name	Component	Description
	Config_PORT	Component : Port Port Selection : PORT0, PORT2, PORT4, PORT12、 P06 : Output (Outputs 1) P07 : Output (Outputs 1) P20 : Output (Outputs 1) P21 : Output (Outputs 1) P22 : Output (Outputs 1) P41 : Output (Outputs 1) P121 : Output (Outputs 1) P122 : Output (Outputs 1)

4.10.1.1 Clock

This selection configurates the clock setting used in the sample code.

4.10.1.2 System

This selection configurates the on-chip debug settings for the sample code.

The "On-Chip Debug Operation Setting" and "Setting When Security ID Authentication Fails" affect the "On-chip Debug Operation Setting" in "4.2 Operation Byte Setting List". Be cautious when modifying these settings.

4.10.1.3 r_bsp

This selection configurates the startup setting for the sample code.

4.10.1.4 Config_INTC

This selection configurates INTP0 in the sample code.

The sample code detects the falling edge when SW2 is pressed. During interrupting vector processing, an initialization process is executed when SW2 is pressed.

4.10.1.5 Config_IT

This selection configurates the IT settings for the sample code.

The sample code uses it for time management in stew mode.

4.10.1.6 Config_TAU0_0

This selection configurates TAU00 in the sample code.

In the sample code, it is used for PWM output to control heater overheating.

4.10.1.7 Config_TAU0_2

This selection configurates TAU02 in the sample code.

In the sample code, it is used for long-press button detection, operation mode confirmation, and debounce processing.

4.10.1.8 Config_ADC

This selection configurates ADC in the sample code.

In the sample code, it is used to obtain the resistance value of the thermistor.

4.10.2 Method for Changing Temperature and Time Settings

This sample code has predefined temperature and time setting. To modify these settings, change the values of the constants listed below. When making changes, ensure proper evaluation according to the specifications of the heater and thermistor. For temperatures above a certain threshold, voltage adjustments for the heater are necessary.

Table 4-16 Various Setting

Setting	Constant Name	Value
Boiling Mode Temperature Settings	TARGET_TEMP_LOW	40
	TARGET_TEMP_MID	55
	TARGET_TEMP_HIGH	70
Stewing Mode Temperature Setting	STEW_TEMP	TARGET_TEMP_HIGH
Stewing Mode Time Settings	STEW_TIME_SHORT	10
	STEW_TIME_MIDDLE	40
	STEW_TIME_LONG	70
Emergency Stop Temperature Setting	TEMP_ABNORMAL	80

5. References Documents

- RL78/G15 User's Manual Hardware (R01UH0959J)
- RL78 Family User's Manual Software (R01US0015)
- RL78 Smart Configurator User's Guide : CS+ Edition (R20AN0580J)
- RL78 Smart Configurator User's Guide : e^2 studio Edition (R20AN0579J)
  (The latest versions can be obtained from the Renesas Electronics website.)

Revision History

Rev.	Publication Date	Revision Details	
		Pages	Point
1.00	Mar.6.2025	—	First edition released

Precautions for Product Use

This section explains general precautions applicable to all microcontroller products. For specific precautions, please refer to this document and technical updates.

1. Electrostatic Discharge Protection

   When handling CMOS products, take precautions against electrostatic discharge. CMOS devices may suffer gate oxide breakdown due to strong static electricity. During transportation and storage, use conductive trays, magazine cases, conductive cushioning materials, or metal cases as used in our shipping packaging. Ensure proper grounding in the assembly process. Avoid placing on plastic boards or touching their materials. The same precautions should be applied to board with mounted CMOS products.
2. Handling During Power-On

   At power-on, the state of the device is undefined. The internal circuits, register settings, and terminal state of the LSI are indeterminate during this phase. For products that require an external reset terminal, the state of the terminals is not guaranteed from power-on until reset signal becomes active. Similarly, for products that use an internal power-on reset function, the terminal states are not guaranteed until the supply voltage reaches the threshold required to trigger the reset.
3. Input Signals When Power is Off

   When the product's power is off, do not apply input signals or enable pull-up power for input/output pins. Injecting current from input signals or pull-up power may cause malfunction or lead to abnormal current flow, potentially degrading internal components. If the product documentation includes specific instructions regarding "input signals when power is off" be sure to follow them.
4. Handling of Unused Pins

   Unused pins must be handled according to the "Handling of Unused Pins" guidelines. CMOS product input pins generally have high impedances. Leaving unused pins open may result in induced noise from the surrounding environment, leading to excessive current flow inside the LSI or unintentional input recognition, which may cause malfunction.
5. Clock Considerations

   During reset, ensure that the clock is stable before releasing the reset. When switching clocks during program execution, the transition should only occur once the new clock source has established. For systems using an external oscillator (or external oscillation circuit) as the clock source at startup, release the reset only after the clock has fully stabilized. Similarly, when switching to an external oscillator (or external oscillation circuit) during program execution, ensure that the target clock has stabilized before making the transition
6. Input Signal Waveform

   Noise or waveform distortion due to signal reflections can cause malfunctions and must be considered. If CMOS inputs are affected by noise and remain within the transition range between VIL (Max) and VIH (Min), unexpected behavior may occur. Ensure that input levels remain stable and that no chattering noise occurs during the transition period between VIL (Max) and VIH (Min).
7. Prohibited Access to Reserved Addresses

   Access to reserved addresses is prohibited. Certain address spaces are reserved for future functionality and should not be accessed. The behavior resulting from accessing these reserved addresses is not guaranteed, so such access must be strictly avoided.
8. Differences Between Products

   When switching to a different product model, conduct system evaluation tests for each product. Even within the same microcontroller group, variations in flash memory, layout patterns, and electrical characteristics may cause differences in parameters such as operating margins, noise immunity, and emitted noise levels. Always perform system evaluation tests for each product to ensure proper operation.