RL78/I1A

LED Lighting System Equipped with PFC Control

R01AN2138EJ0100
Rev.1.00
Jul 15, 2014

Introduction

The purpose of this application note is to describe how to control LED lighting systems equipped with PFC control by using the RL78/I1A microcontroller features.

Target Devices

This document is intended for system engineers who design and develop LED lighting systems and power supply systems.

The target products are as follows:

- 20-pin: R5F1076C
- 30-pin: R5F107AE, R5F107AC
- 38-pin: R5F107DE

Table of Contents

1. Introduction

This application note describes a sample program that controls LED lighting by using the RL78/I1A microcontroller. This sample program controls three LEDs independently by a constant current control technique, individual dimming by using switches, and PFC output voltage. Feedback processing during LED constant current control or PFC output voltage control is based on the proportional-integral (PI) method.

This program can be evaluated using the RL78/I1A AC/DC LED control evaluation board. See the circuit diagram for the pin configuration on the RL78/I1A AC/DC LED control board.

2. Overview of LED Control Using RL78/I1A

2.1 RL78/I1A Features for Controlling LED Systems

The RL78/I1A microcontroller incorporates comprehensive features enabling efficient LED lighting system control:

— RL78/I1A MCU allows control of LED constant current and PWM dimming on up to six channels by using the outputs of 16-bit timers KB0, KB1, KB2 and 16-bit timer KC0. This eliminates the need for an external IC dedicated to LED constant current control, thus reducing the design cost. These 16-bit timers KBn include several powerful functions, one of them used in the sample program is Dithering function which contributes to increasing the average PWM resolution to 0.98 ns.

— RL78/I1A allows power factor correction (PFC) control in critical conduction mode (CRM) using the timer restart function, which is based on comparators and external interrupts that operate together with 16-bit timers KBn. This also eliminates the need for a dedicated PFC control IC, thus reducing design cost even further.

— RL78/I1A embeds protective functions that stop PWM output (without requiring control via the CPU) when an overcurrent or overvoltage is detected in an LED or PFC control circuit. This is realized by using the forced output stop function triggered by comparators and external interrupts that operate together with 16-bit timers KBn.
Furthermore, operation restart after the emergency stop can be controlled by software, thus realizing a flexible protective function according to the system requirements.

— The R78/I1A MCU incorporates a serial array unit (UART4/DALI) that supports the DALI communication function to enable transmission and reception of Manchester codes (8, 16, 17 or 24 bits), which is the communication method prescribed by the DALI communication standard.
This allows reducing the CPU load during data transmission and reception.

— RL78/I1A also supports DMX512 communication via its UART0 serial interface.
The input signal pulse width measurement function of Timer Array Unit channel 7 can be used to detect the falling edge of a break period on the RxD0 reception pin and to measure its length (low level for at least 88 µs), and the interval timer function of the 16-bit Timer Array Unit can be used to calculate and acknowledge the MARK AFTER BREAK signal width (high level for 8 µs to 1 s) and also to measure the MARK TIME BETWEEN SLOTS can be measured as well.

— The pulse interval measurement function of the 16-bit timer array unit can also be used to receive infrared (IR) remote control signals. This can reduce the CPU load during data reception.

Note PWM dimming using the 16-bit Timer KC0 gate control function is not used in the sample program described in this application note. Only DC dimming by adjusting the PWM values of timer KB0 and KB1 channels is used.

2.2 System Block Diagram

Figure 1 shows the system block diagram of the RL78/I1A AC/DC LED control evaluation board.

This LED lighting system controls PFC and three LEDs in response to the switch input.

The system requires no additional external ICs for controlling PFC and LEDs because they can be controlled by using the RL78/I1A microcontroller,.

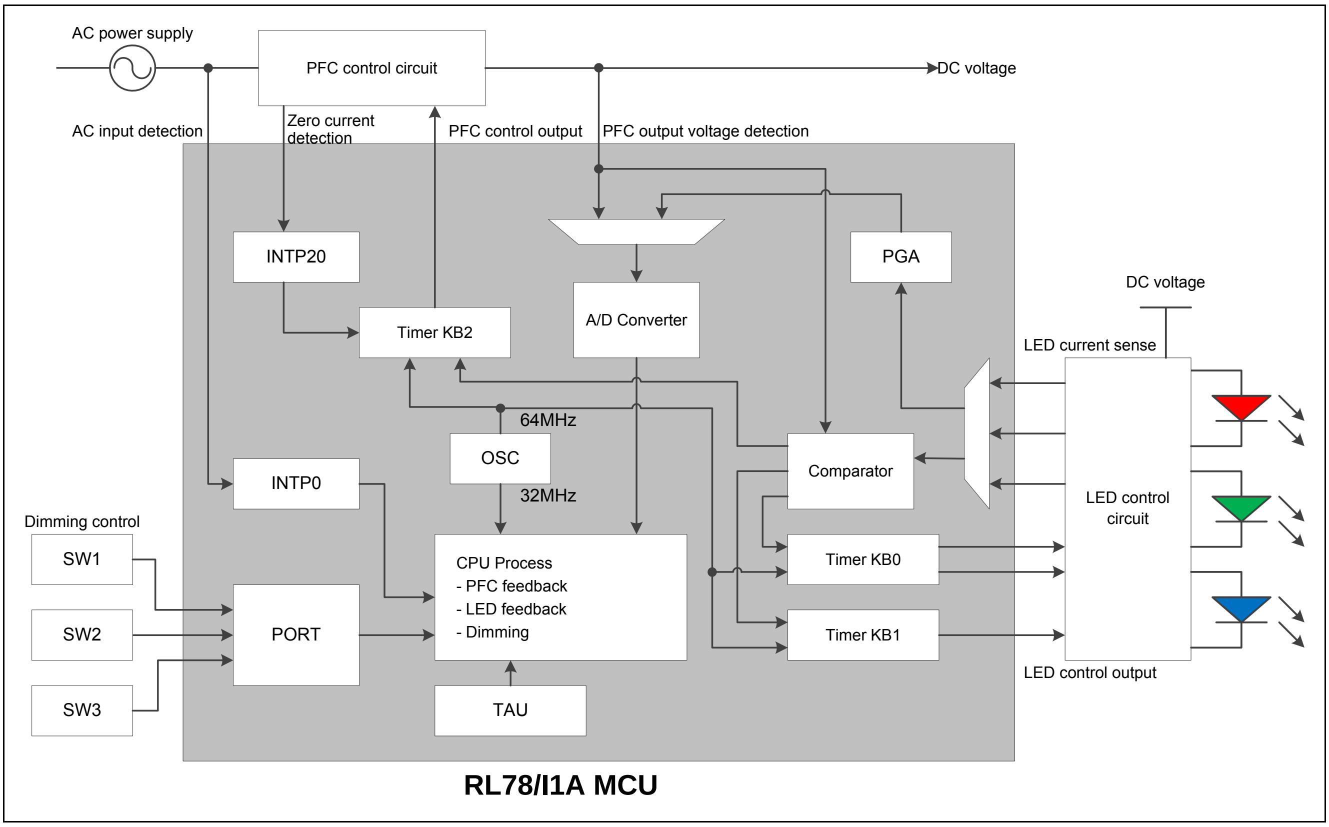


Figure 1 Block Diagram of RL78/I1A AC/DC LED Control Evaluation Board

2.3 Pin Functions of RL78/I1A

Table 1 presents the different pins which are used, and gives a brief description of their functions within the LED control system.

Table 1 Pin Functions

Function	Function Name	Pin Name	I/O	Description
LED control	TKBO00	P200	O	LED1 PWM output
	TKBO01	P201	O	LED2 PWM output
	TKBO10	P202	O	LED3 PWM output
	ANI4/CMP1P	P24	I	Analog input for monitoring LED1 current
	ANI5/CMP2P	P25	I	Analog input for monitoring LED2 current
	ANI6/CMP3P	P26	I	Analog input for monitoring LED3 current
PFC control	TKBO21	P205	O	PFC output
	INTP0	P137	I	Interrupt input for AC zero cross detection
	INTP20	P203	I	Interrupt input for zero current detection
	ANI7/CMP4P	P27	I	Analog input for monitoring PFC output voltage
Others	P75	P75	I	Switch 1 input
	P76	P76	I	Switch 2 input
	P77	P77	I	Switch 3 input

3. Control Software

This chapter describes the files of this sample program, the internal peripheral functions of the RL78/I1A to be used, and their initial settings. This chapter also gives an overview of the overall operation (constant current and dimming control) of this sample program, as well as an explanation about the PI method implemented for feedback control, and also includes flow charts of the different processing.

You can obtain this sample program from Renesas Electronics website by downloading Applilet EZ for HCD and selecting switch mode 1.

3.1 File Configuration

This sample program consists of the files listed below:

Table 2 Files

File Name	Contents
Init.asm	holds the option byte settings
r_systeminit.c	contains system initialization functions
r_userinit.c	contains top function for peripheral initialization settings
r_main.c	contains the main function
r_usermain.c	contains the LED dimming functions
r_swmode.c	contains initialization and implementation functions for SW input judging
r_swmode1.c	contains state transition judging function by SW input
r_ac.c	contains functions for judging AC power supply input
r_led_autotuning.c	contains auto-tuning functions
r_led_dimcontrol.c	contains functions for PFC control and LED dimming control
r_led_dimrequest.c	contains LED constant current control functions
r_led.c	contains initialization functions for LED control
r_timer.c	contains timer functions (interval mode) for AC/SW input sampling
r_wdt.c	contains watchdog initialization function
r_user.h	contains parameter definitions used for clock and ADC configurations
r_led_dimming.h	contains parameter definitions used for LED control
r_led_user.h	contains definitions for control target LED
r_system.h	contains definitions for r_systeminit.c
r_ac.h	contains definitions for r_ac.c
r_led.h	contains definitions for r_led_dimcontrol.c
r_swmode.h	contains definitions for r_swmode.c
r_wdt.h	contains definitions for r_wdt.c
r_timer.h	contains definitions for r_timer.c
r_macrodriver.h	contains macro definitions for drivers

3.2 Global Variables

Table 3 shows the global variables used in this sample program.

Table 3 Global Variables (1)

Data Type	Variable Name	Overview	Function That Uses This Variable
unsigned char	ucConnectLed	LED connection state b1: LED connection state (1: Connected, 0: Disconnected) b2: LED2 connection state (1: Connected, 0: Disconnected) b3: LED3 connection state (1: Connected, 0: Disconnected)	auto_tuning() at_check() at_finalize() dim_led1() dim_led2() dim_led3()
unsigned char	ucReqLed	LED operation request b1: LED1 operation request (1: ON, 0: OFF) b2: LED2 operation request (1: ON, 0: OFF) b3: LED3 operation request (1: ON, 0: OFF)	int_tm00() boost_pfc() stop_led1() stop_led2() stop_led3() dim_led1() dim_led2() dim_led3() dim_trig() dim_ini() at_check() at_run() at_finalize()
unsigned char	ucStateLight	LED operation state 0 = All OFF b0: PFC step-up (1: ON, 0: OFF) b1: LED1 operation state (1: ON, 0: OFF) b2: LED2 operation state (1: ON, 0: OFF) b3: LED3 operation state (1: ON, 0: OFF)	int_tm00() start_pfc() boost_pfc() feedback_led1() feedback_led2() feedback_led3() stop_led1() stop_led2() stop_led3() dim_trig() dim_ini()
unsigned short	usErrStatus	Error information (1: Exists, 0: None) 0 = No error b0: No LED detection b1: PFC output overvoltage (before step-up) b2: PFC output overvoltage (in step-up) b3: Step-up timeout b4: PFC output overvoltage (in lighting) b5: LED1 overcurrent b6: LED2 overcurrent b7: LED3 overcurrent b8: PFC output overvoltage (comparator detection)	main() start_pfc() boost_pfc() feedback_led1() feedback_led2() feedback_led3() feedback_pfc() at_check() at_run() at_finalize()

Table 4 Global Variables (2)

Data Type	Variable Name	Overview	Function That Uses This Variable
unsigned char	ucFeedbackCnt	Feedback processing phase 0x01: LED1 feedback processing 0x02: LED2 feedback processing 0x03: LED3 feedback processing 0x04: PFC feedback processing 0x05: Other processing	int_tm00() boost_pfc() at_finalize()
unsigned char	g_ucSwInputMode[3]	Operation mode 0x00: SWMODE_LED_OFF LED light-off state 0x01: SWMODE_LED_ON_MIN Minimum LED dimming value (SW ON) 0x02: SWMODE_LED_ON_MAX Maximum LED dimming value (SW ON) 0x03: SWMODE_LED_MAXFADE LED step dimming state (maximum target) 0x04: SWMODE_LED_MINFADE LED step dimming state (minimum target) 0x05: SWMODE_LED_ON_UP LED dimming stop state (minimum target) 0x06: SWMODE_LED_ON_DN LED dimming stop state (maximum target) 0x07: SWMODE_LED_ON_MAX_REL Maximum LED dimming value (SW OFF) 0x08: SWMODE_LED_ON_MIN_REL Minimum LED dimming value (SW OFF)	SwMode_init() SwMode1_get Value()
unsigned short	g_usACInputOn Counter	AC input detection counter Counts every 1 ms.	ACInput_init() AC_pulse_check() AC_on_check()
unsigned char	g_ucACInputOff Counter	AC input non-detection counter Counts every 1 ms.	ACInput_init() ACInput_Timer Interrupt() AC_pulse_check() AC_on_check()
unsigned short	usCntInterval	Interval counter for auto-tuning Counts every 64 μs.	at_check() at_run() at_finalize()
unsigned long	ulCntBoost	Measurement counter for the PFC step-up time Counts every 64 μs.	start_pfc() boost_pfc()

Table 5 Global Variables (3)

Data Type	Variable Name	Overview	Function That Uses This Variable
unsigned short	ushOffsetPhase	LED offset phase 1: LED1 offset processing 2: LED2 offset processing 3: LED3 offset processing	boost_pfc()
unsigned short	ushOffsetCount	LED offset counter Counts every time three LED offsets are completed.	boost_pfc()
signed short	shAdOffsetLed1	LED1 offset AD value	start_pfc() boost_pfc() feedback_led1() at_finalize()
signed short	shAdOffsetLed2	LED2 offset AD value	start_pfc() boost_pfc() feedback_led2() at_finalize()
signed short	shAdOffsetLed3	LED3 offset AD value	start_pfc() boost_pfc() feedback_led3() at_finalize()
unsigned char	g_ucSWInputCounter	Measurement counter for the SW input time Counts every 1 ms.	SwMode_init() SwMode_TimerInterrupt()
unsigned char	g_ucSwCheckTimePass[3]	SW judging elapsed time information 0: Not elapsed 1: Elapsed	SwMode_init() SwMode_TimerInterrupt() SWInput_check()
unsigned char	g_ucSwPushTime[3]	SW continuously pressed time Counts every 10 ms.	SwMode_init() SwMode_TimerInterrupt() SWInput_check()
unsigned char	g_ucSwPushFlag[3]	SW holding-down flag 0: Judgment of holding down 1: Confirmation of holding down	SwMode_init() SWInput_check()
unsigned char	g_ucSwFix_Data[3]	SW operation confirmation data 0x00: OFF confirmation 0x1F: ON confirmation	SwMode_init() SWInput_check()
unsigned char	g_ucSwIn_Data[3]	SW input data	SwMode_init() SWInput_check()
unsigned short	g_usSwValue[3]	Dimming request value by SW operation	SwMode_init() SwMode1_getValue()
signed short	shReqLed1	LED1 dimming target request value	feedback_led1() dim_led1() dim_ini() at_run()
signed short	shReqLed2	LED2 dimming target request value	feedback_led2() dim_led2() dim_ini() at_run()

Table 6 Global Variables (4)

Data Type	Variable Name	Overview	Function That Uses This Variable
signed short	shReqLed3	LED3 dimming target request value	feedback_led3() dim_led3() dim_ini() at_run()
signed short	shAdCled1Tgt	LED1 dimming target value	feedback_led1() stop_led1() dim_led1() dim_ini()
signed short	shAdCled2Tgt	LED2 dimming target value	feedback_led2() stop_led2() dim_led2() dim_ini()
signed short	shAdCled3Tgt	LED3 dimming target value	feedback_led3() stop_led3() dim_led3() dim_ini()
signed short	shAdLed1	LED1 feedback AD value	feedback_led1() boost_pfc() at_finalize()
signed short	shAdLed2	LED2 feedback AD value	feedback_led2() boost_pfc() at_finalize()
signed short	shAdLed3	LED3 feedback AD value	feedback_led3() boost_pfc() at_finalize()
unsigned short	ushPowLed1	LED1 load value	at_finalize()
unsigned short	ushPowLed2	LED2 load value	at_finalize()
unsigned short	ushPowLed3	LED3 load value	at_finalize()
unsigned short	ushPowTotal	LED total load value	at_finalize()
unsigned long	ulDpfcLed1	Ratio of LED1 load to total load	dim_led1() at_check() at_finalize()
unsigned long	ulDpfcLed2	Ratio of LED2 load to total load	dim_led2() at_check() at_finalize()
unsigned long	ulDpfcLed3	Ratio of LED3 load to total load	dim_led3() at_check() at_finalize()
signed long	slDpfcLed1	Correction value for PFC feed forward PFC PWM duty value ± = Correction value for PFC feed forward	feedback_led1() stop_led1() dim_led1()

Table 7 Global Variables (5)

Data Type	Variable Name	Overview	Function That Uses This Variable
signed long	slDpfcLed2	Correction value for PFC feed forward PFC PWM duty value ± = Correction value for PFC feed forward	feedback_led2() stop_led2() dim_led2()
signed long	slDpfcLed3	Correction value for PFC feed forward PFC PWM duty value ± = Correction value for PFC feed forward	feedback_led3() stop_led3() dim_led3()
unsigned long	ulSumDpfco	PFC PWM duty integration value	at_finalize()
unsigned long	ulSumCled1	LED1 current AD integration value	at_finalize()
unsigned long	ulSumCled2	LED2 current AD integration value	at_finalize()
unsigned long	ulSumCled3	LED3 current AD integration value	at_finalize()
unsigned long	ulSumDled1	LED1 PWM duty integration value	at_finalize()
unsigned long	ulSumDled2	LED2 PWM duty integration value	at_finalize()
unsigned long	ulSumDled3	LED3 PWM duty integration value	at_finalize()
unsigned long	ulSumCled1Count	LED1 PWM duty integration counter	at_finalize()
unsigned long	ulSumCled2Count	LED2 PWM duty integration counter	at_finalize()
unsigned long	ulSumCled3Count	LED3 PWM duty integration counter	at_finalize()
signed long	slLedA1	LED PI control coefficient A1	feedback_led1() feedback_led2() feedback_led3()
signed long	slLedA2	LED PI control coefficient A2	feedback_led1() feedback_led2() feedback_led3()
signed long	slPfcA1	PFC PI control coefficient A1	feedback_pfc()
signed long	slPfcA2	PFC PI control coefficient A2	feedback_pfc()
union long short	uData1 slErrLED1 sErrLED1[2]	LED1 PI control calculation result	boost_pfc() feedback_led1() stop_led1()
union long short	uData2 slErrLED2 sErrLED2[2]	LED2 PI control calculation result	boost_pfc() feedback_led2() stop_led2()
union long short	uData3 slErrLED3 sErrLED3[2]	LED3 PI control calculation result	boost_pfc() feedback_led3() stop_led3()

Table 8 Global Variables (6)

Data Type	Variable Name	Overview	Function That Uses This Variable
union long short	uData4 slErrPfc sErrPfc[2]	PFC PI control calculation result	start_pfc() feedback_led1() feedback_led2() feedback_led3() feedback_pfc() stop_led1() stop_led2() stop_led3()
signed short	shAdTempLed1	Variables for PI control calculation Holding the LED1 error value	feedback_led1() stop_led1() dim_ini()
signed short	shAdTempLed2	Variables for PI control calculation Holding the LED2 error value	feedback_led2() stop_led2() dim_ini()
signed short	shAdTempLed3	Variables for PI control calculation Holding the LED3 error value	feedback_led3() stop_led3() dim_ini()
signed long	slErrTemp	Variables for PI control calculation Holding (previous error value x coefficient A2)	feedback_led1() feedback_led2() feedback_led3()
signed long	slErrTemp1	Variables for PI control calculation Holding (latest error value x coefficient A1)	feedback_led1() feedback_led2() feedback_led3()
unsigned short	ushADtemp	PFC output AD value	start_pfc() boost_pfc() feedback_pfc()
signed short	shADtempPFC	Variables for PI control calculation Holding the latest error value of PFC	feedback_pfc()
signed short	shAdOldVout	Variables for PI control calculation Holding the previous error value of PFC	feedback_pfc()
unsigned char	ucIntP00	AC input interrupt flag (INTP0)	int_p00() AC_pulse_check() at_run() at_finalize()
unsigned char	ucIntTm00	TM00 timer interrupt flag (INTTM00)	int_tm00() at_check() at_run() at_finalize()

3.3 Functions

Table 9 shows the functions used in this sample program.

Table 9 Functions

Function Name	Overview
int_p00	INTP0 interrupt processing
int_tm00	INTTM00 interrupt processing
start_pfc	PFC operation start processing
boost_pfc	PFC step-up processing
feedback_led1	LED1 feedback processing
feedback_led2	LED2 feedback processing
feedback_led3	LED3 feedback processing
feedback_pfc	PFC feedback processing
stop_led1	LED1 stop processing
stop_led2	LED2 stop processing
stop_led3	LED3 stop processing
stop_pfcled	PFC/LED stop processing
dim_led1	LED1 control request processing
dim_led2	LED2 control request processing
dim_led3	LED3 control request processing
dim_trig	Dimming control timer operation start
SWInput_check	SW input judging
SwMode1_getValue	Judgment of dimming state transition by SW
SwValue_StepUp	Dimming value step-up
SwValue_StepDown	Dimming value step-down
SwMode_TimerInterrupt	Timer cycle processing for SW judgment
ACInput_check	AC power supply input check
AC_pulse_check	AC pulse input judgment processing
AC_on_check	AC power supply input state judgment
ACInput_TimerInterrupt	Timer cycle processing for AC input judgment
auto_tuning	Auto tuning processing
at_check	Tuning judgment processing
at_run	Tuning execution processing
at_finalize	Tuning completion processing
user_init	User-defined initialization processing
hdwinit	Register initialization processing
dim_ini	Dimming initialization processing
ACInput_init	AC input judgment initialization processing
Timer_init	Timer initialization processing
LED_init	LED control initialization processing
SwMode_init	SW judgment initialization processing
user_main	User-defined main processing
main	Main processing
Timer_Interrupt	1-ms timer cycle processing
WDT_Reset	Watchdog timer reset processing

3.4 Function Specifications

This section describes the detailed specifications of the software functions.

int_p00

Overview	INTP0 interrupt processing
Declaration	__interrupt void int_p00(void)
Description	Interrupt of AC power supply zero cross detection
Argument	None
Return value	None
Remarks	Interrupt cycle = ½ fz

int_tm00

Overview	INTTM00 interrupt processing
Declaration	__interrupt void int_tm00(void)
Description	- LEDn feedback processing - PFC feedback processing - PFC control PWM timer operation start - PFC step-up processing - Other processing
Argument	None
Return value	None
Remarks	Interval time (64 μs) Other processing is provided in the LED/PFC feedback cycle for adding user processing. Add processing as needed. In this case, do not exceed the 32 μs of processing time because adding user processing might interfere with the LED/PFC feedback cycle.

start_pfc

Overview	PFC operation start processing
Declaration	void start_pfc(void)
Description	Starts the PFC control PWM timer operation.
Argument	None
Return value	None
Remarks	None

boost_pfc

Overview	PFC step-up processing
Declaration	void boost_pfc(void)
Description	Executes PFC step-up processing. - LEDn control PWM timer operation start - LEDn offset value acquisition Judges the PFC output step-up timeout. Judges the PFC output overvoltage.
Argument	None
Return value	None
Remarks	None

feedback_led1

Overview	LED1 feedback processing
Declaration	void feedback_led1(void)
Description	Reflects the calculation results of the PI control formula in the LED1 control PWM duty cycle. Judges the LED1 overcurrent. Controls PFC feed forward by LED1 control.
Argument	None
Return value	None
Remarks	None

feedback_led2

Overview	LED2 feedback processing
Declaration	void feedback_led2(void)
Description	Reflects the calculation results of the PI control formula in the LED2 control PWM duty cycle. Judges the LED2 overcurrent. Controls PFC feed forward by LED2 control.
Argument	None
Return value	None
Remarks	None

feedback_led3

Overview	LED3 feedback processing
Declaration	void feedback_led3(void)
Description	Reflects the calculation results of the PI control formula in the LED3 control PWM duty cycle. Judges the LED3 overcurrent. Controls PFC feed forward by LED3 control.
Argument	None
Return value	None
Remarks	None

feedback_pfc

Overview	PFC feedback processing
Declaration	void feedback_pfc(void)
Description	Reflects the calculation results of the PI control formula in the PFC control PWM duty cycle. Judges the PFC output overvoltage.
Argument	None
Return value	None
Remarks	None

stop_led1

Overview	LED1 stop processing
Declaration	void stop_led1(void)
Description	Executes the stop processing of the LED1 control PWM timer. Controls PFC feed forward by LED1 control.
Argument	None
Return value	None
Remarks	None

stop_led2

Overview	LED2 stop processing
Declaration	void stop_led2(void)
Description	Executes the stop processing of the LED2 control PWM timer. Controls PFC feed forward by LED2 control.
Argument	None
Return value	None
Remarks	None

stop_led3

Overview	LED3 stop processing
Declaration	void stop_led3(void)
Description	Executes the stop processing of the LED3 control PWM timer. Controls PFC feed forward by LED3 control.
Argument	None
Return value	None
Remarks	None

stop_pfcled

Overview	PFC & LED stop processing
Declaration	void stop_pfcled(void)
Description	Stops all the PFC/LED control timers.
Argument	None
Return value	None
Remarks	None

dim_led1

Overview	LED1 control request processing
Declaration	void dim_led1(unsigned short ushInLed)
Description	Sets the LED1 dimming target request value and an LED operation request.
Argument	• unsigned short Dimming target value
Return value	None
Remarks	None

dim_led2

Overview	LED2 control request processing
Declaration	void dim_led2(unsigned short ushInLed)
Description	Sets the LED2 dimming target request value and an LED operation request.
Argument	• unsigned short Dimming target value
Return value	None
Remarks	None

dim_led3

Overview	LED3 control request processing
Declaration	void dim_led3(unsigned short ushInLed)
Description	Sets the LED3 dimming target request value and an LED operation request.
Argument	• unsigned short Dimming target value
Return value	None
Remarks	None

dim_trig

Overview	Dimming control timer operation start
Declaration	void dim_trig(void)
Description	Starts the timer TM00 operation synchronized with PFC/LED control.
Argument	None
Return value	None
Remarks	None

SWInput_check

Overview	SW input judging
Declaration	unsigned char SWInput_check(unsigned char ucSwNum)
Description	Judges the ON/OFF status of SWs. - Judgment of pressing - Judgment of holding down
Argument	• unsigned char SW No. (0 to 2)
Return value	• 0: SW: N/A • 1: SW: Pressing • 2: SW: Holding down • 3: SW: OFF • 4: SW: ON (rising edge)
Remarks	None

SwMode1_getValue

Overview	Judgment of dimming state transition by SW
Declaration	unsigned short SwMode1_getValue(unsigned char ucChannel)
Description	Determines the dimming target value based on the SW state and operation mode. For definition of the operation mode, refer to the following. - SWMODE_LED_OFF LED light-off state - SWMODE_LED_ON_MIN Minimum LED dimming value (SW ON) - SWMODE_LED_ON_MIN_REL Minimum LED dimming value (SW OFF) - SWMODE_LED_MINFADE LED step dimming state (minimum target) - SWMODE_LED_ON_UP LED dimming stop state (minimum target) - SWMODE_LED_ON_MAX Maximum LED dimming value (SW ON) - SWMODE_LED_ON_MAX_REL Maximum LED dimming value (SW OFF) - SWMODE_LED_MAXFADE LED step dimming state (maximum target) - SWMODE_LED_ON_DN LED dimming stop state (maximum target)
Argument	• unsigned char LED channel No. (1 to 3)
Return value	• Dimming target value
Remarks	None

SwValue_StepUp

Overview	Dimming value step-up
Declaration	unsigned short SwValue_StepUp(unsigned short usStep)
Description	Increases the LED dimming request value by one step. In this program, one step equals 1.
Argument	• unsigned short Current dimming request value
Return value	Post-processing dimming request value
Remarks	None

SwValue_StepDown

Overview	Dimming value step-down
Declaration	unsigned short SwValue_StepDown(unsigned short usStep)
Description	Reduces the LED dimming request value by one step. In this program, one step equals 1.
Argument	• unsigned short Current dimming request value
Return value	Post-processing dimming request value
Remarks	None

SwMode_TimerInterrupt

Overview	Timer cycle processing for SW judgment
Declaration	void SwMode_TimerInterrupt(void)
Description	Executes the following processing in a 1-ms timer cycle. - SW input time judgment - SW continuously pressed time judgment
Argument	None
Return value	None
Remarks	None

ACInput_check

Overview	AC power supply input check
Declaration	void ACInput_check(void)
Description	Judges AC pulse input. If the AC pulse cannot be detected in 23 ms, PFC/LED control is stopped to wait for AC pulse input.
Argument	None
Return value	None
Remarks	None

AC_pulse_check

Overview	AC pulse input judgment processing
Declaration	void AC_pulse_check(void)
Description	Judges the AC pulse input interrupt (INTP0) state, counts the number of the AC input detection counter, and clears the number of the AC input non-detection counter.
Argument	None
Return value	None
Remarks	None

AC_on_check

Overview	AC power supply input state judgment
Declaration	void AC_on_check(void)
Description	Waits for the AC power supply input interrupt (INTP0).
Argument	None
Return value	None
Remarks	None

ACInput_TimerInterrupt

Overview	Timer cycle processing for AC input judgment
Declaration	void ACInput_TimerInterrupt(void)
Description	Executes the following processing in a 1-ms timer cycle. - Counts the number of the AC input non-detection counter.
Argument	None
Return value	None
Remarks	None

auto_tuning

Overview	Auto tuning processing
Declaration	void auto_tuning(void)
Description	Executes auto-tuning.
Argument	None
Return value	None
Remarks	None

at_check

Overview	Tuning judgment processing
Declaration	void at_check(void)
Description	Waits for SW1 input. When SW1 input is continuously detected in 51 ms, LED1, LED2, and LED3 start to light up.
Argument	None
Return value	None
Remarks	None

at_run

Overview	Tuning execution processing
Declaration	void at_run(void)
Description	Increases the dimming request values up to the maximum dimming values of the LEDs every 64 µs in the order of LED1, LED2, and LED3. Executes tuning completion processing when all the three LEDs are increased to the maximum dimming request value after 16 ms.
Argument	None
Return value	None
Remarks	None

at_finalize

Overview	Tuning completion processing
Declaration	void at_finalize(void)
Description	Calculates the total load ratio of each LED to use it for PFC feed forward control. Judges the non-detected LED.
Argument	None
Return value	None
Remarks	None

user_init

Overview	User-defined initialization processing
Declaration	void user_init(void)
Description	Executes the initialization processing required for each processing.
Argument	None
Return value	None
Remarks	Add initialization processing as needed.

hdwinit

Overview	Register initialization processing
Declaration	void user_init(void)
Description	Executes the initialization processing of each register. For details about initialization, see Section 3.5.
Argument	None
Return value	None
Remarks	Add initialization processing as needed.

dim_ini

Overview	Dimming initialization processing
Declaration	void dim_ini(void)
Description	Executes the initialization processing of the variables required for LED dimming.
Argument	None
Return value	None
Remarks	Add initialization processing as needed.

ACInput_init

Overview	AC input judgment initialization processing
Declaration	void ACInput_init(void)
Description	Executes the initialization processing of the variables required for AC input judgment.
Argument	None
Return value	None
Remarks	Add initialization processing as needed.

Timer_init

Overview	Timer initialization processing
Declaration	void Timer_init(void)
Description	Executes the initialization processing of the variables required for the timer processing.
Argument	None
Return value	None
Remarks	Add initialization processing as needed.

LED_init

Overview	LED control initialization processing
Declaration	void LED_init(void)
Description	Executes the initialization processing of the variables required for LED control.
Argument	None
Return value	None
Remarks	Add initialization processing as needed.

SwMode_init

Overview	SW judgment initialization processing
Declaration	void SwMode_init(void)
Description	Executes the initialization processing of the variables required for SW judgment.
Argument	None
Return value	None
Remarks	Add initialization processing as needed.

user_main

Overview	User-defined main processing
Declaration	void user_main(void)
Description	TM01 time (1 ms) cycle monitoring AC power supply input state judgment Reading SW1, SW2, and SW3 inputs - Dimming value change - LED ON/OFF Dimming processing of LED1, LED2, and LED3
Argument	None
Return value	None
Remarks	None

main

Overview	Main processing
Declaration	void main(void)
Description	Initialization processing in starting up - AC power supply input state judgment - User-defined initialization processing - LED offset value acquisition User-defined main processing
Argument	None
Return value	None
Remarks	None

Timer_Interrupt

Overview	1-ms timer cycle processing
Declaration	void Timer_Interrupt (void)
Description	Waits for TM01 timer (1 ms) interrupt. Executes the timer cycle processing as listed below after detecting the interrupt. - Timer cycle processing for AC input judgment - Timer cycle processing for SW judgment
Argument	None
Return value	None
Remarks	None

WDT_Reset

Overview	Watchdog timer reset processing
Declaration	void WDT_Reset(void)
Description	Resets the watchdog timer enable (WDTE) register, and restarts the watchdog timer.
Argument	None
Return value	None
Remarks	None

3.5 Initialization of the Internal Peripheral Functions

The following internal peripheral functions of the RL78/I1A microcontroller are used in this sample program:

- Dimming processing interval control: 16-bit TAU Channel 0
- Feedback processing interval control: 16-bit TAU Channel 1
- PWM output with dithering function: 16-bit timers KB0, KB1, and KB2
- Feedback input amplification: PGA channel n (n = 4, 5, 6)
- PFC voltage feedback input: 10-bit A/D converter channel n (n = 7)
- LED current feedback input (from PGA output): 10-bit A/D converter channel n (n = 3)

Table 10 A/D Converter Channel Allocation

Analog Input Channel	Function
ANI0	AVREFP
ANI1	AVREFM
ANI4	Feedback current input of LED1
ANI5	Feedback current input of LED2
ANI6	Feedback current input of LED3
ANI7	PFC output voltage input

The hardware initialization includes the following settings:

1. Option byte setting
— Stopping watchdog timer operation
— Setting of LVD (Low-voltage detector) operation mode and detection level
— Setting VLVI (low-voltage detection voltage) to 4.06 V
— Setting LVD in reset mode (Generating an internal reset when VDD is less than VLVI)
— Selecting the high-speed internal oscillator (4 MHz) as the system clock source
— Enabling on-chip debugging

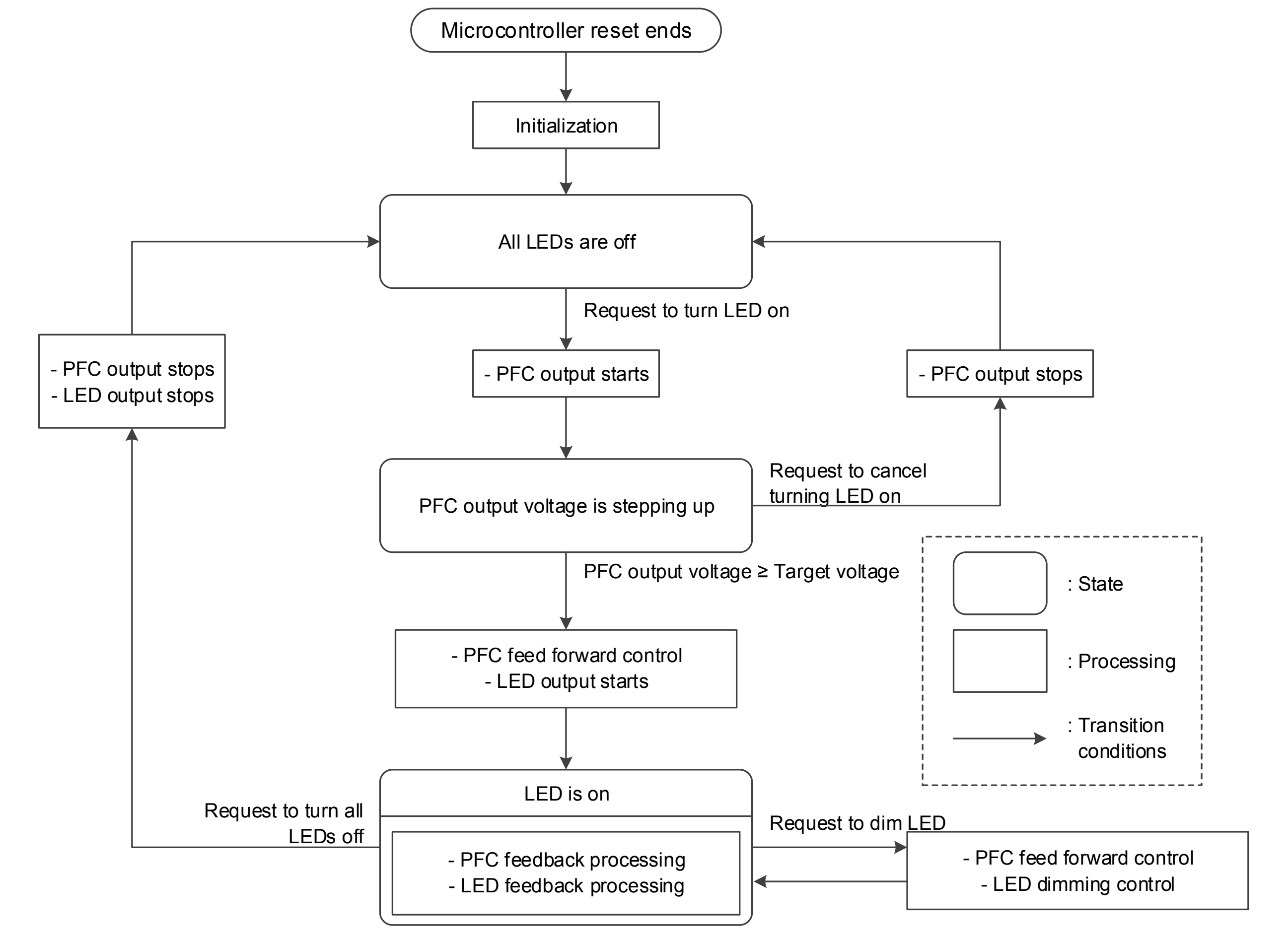
2. Peripheral setting
— Setting CPU clock frequency to 32 MHz using PLL (16 times the internal high-speed oscillation clock $f_{IH} \times 1/2$)
— Peripheral function clock supply setting
— I/O port setting
— 16-bit TAU Channel 0 setting
— Setting count clock to f_{CLK} (32 MHz)
— Setting the interval time to 1 ms ($(TDR00 + 1)/f_{CLK}$)
— Unmasking the interrupt INTTM00
— 16-bit TAU Channel 1 setting
— Setting count clock to f_{CLK} (32 MHz)
— Setting the interval time to 64 µs ($(TDR01 + 1)/f_{CLK}$)
— Unmasking the interrupt INTTM01
— A/D converter setting
— Setting A/D converting time to 2.125 µs
— Masking the interrupt INTAD
— Programmable gain amplifier (PGA) setting
— Setting PGA amplification factor to × 8
— Setting input channel to ANI2

- Comparator setting
  - Setting comparator 1, comparator 2, and comparator 3 to LED overvoltage detection
  - Setting comparator 4 and comparator 5 to PFC overvoltage detection
  - Allowing output of comparators 1 to 4
- 16-bit Timer KB setting
  - Setting count clock to f_{CLK} = 64 MHz
  - Enabling TKBO00, TKBO01 and TKBO10 PWM output dithering function
  - Setting operation mode of TKBO and TKB1 to standalone mode
  - Setting output default level of TKBO00, TKBO01 and TKBO10 to low level and active level to active high
  - Setting frequency of PWM output to 250 kHz (f/(TKBCRn0 + 1), n = 0, 1)
    - 64 MHz counting clock source with a resolution of 8 bits (64 MHz / 28)
  - Masking the interrupts INTTMKB0, INTTMKB1
  - Setting comparator 1 as a trigger of the TKBO00 forced output stop function
  - Setting comparator 2 as a trigger of the TKBO01 forced output stop function
  - Setting comparator 3 as a trigger of the TKBO10 forced output stop function
  - Setting comparators 4 and 5 as triggers of the TKBO21 forced output stop function

After initialization, the 250 kHz PWM signal is output from 16-bit timers KB0 and KB1 to drive the LEDs. Basically, the A/D converter detects the sense voltages from the feedback input pins, compares them with the ADC target levels and adjusts the duty of the PWM outputs to maintain a constant current.

3.6 System Operation Overview

Figure 2 shows a transition of state in LED lighting control by using the RL78/I1A microcontroller. The state is roughly divided into three states: "All LEDs are off", "PFC output voltage is stepping up", and "LED is on".



Definition of transition condition

Request to turn LED on:	Request to turn on one or more LEDs among LED1 to LED3
Request to cancel turning LED on:	Request to turn off all LEDs to be turned on, or request to stop stepping up the PFC output voltage when an error occurs (when the PFC output voltage does not reach the target voltage within a certain time after stepping up the voltage starts or when the PFC output overvoltage is detected)
PFC output voltage ≥ Target voltage:	The PFC output voltage reaches the target voltage (100 V for this evaluation board).
Request to dim an LED:	Request to dim or turn off an LED or request to turn the LED on
Request to turn all LEDs off:	Request to turn all the LEDs off

Figure 2 Transition of State in LED Lighting Control Equipped with PFC Control by Using RL78/I1A Microcontroller

3.7 LED Control

3.7.1 Dimming control based on SW input

In this program, SWn[Note 1] input (pressing or holding down) enables you to adjust the brightness of LEDs. SW1 corresponds to LED1, SW2 to LED2, and SW3 to LED3.

SW input is sampled every 10 ms in synchronized with the main processing cycle. The LED on/off status is determined by high or low level detected five times in a row.

In this program, SW operation is judged according to the following manner.

- LED ON: Low level detected five times in a row
- LED OFF: High level detected five times in a row
- SW pressed: Edge of changing ON to OFF detected
- SW held down: ON detected continuously for 500 ms. After that, ON detected continuously every 50 ms

A state transitions based on the determined SW operation, and the dimming level (target level: 'shAdClednTgt'[Note 2]) that indicates the LED brightness is determined. Figure 3 shows a transition of the dimming level based on SW operation.

Auto-tuning must be performed by pressing SW1 before dimming of LEDs starts. For details about auto-tuning, see *3.8.3*.

Notes 1 n = 1, 2, 3 (LED1 lights red, LED2 lights green, and LED3 lights blue on the board.)

2 The string enclosed by single quotation marks is a global variable used in the sample program.

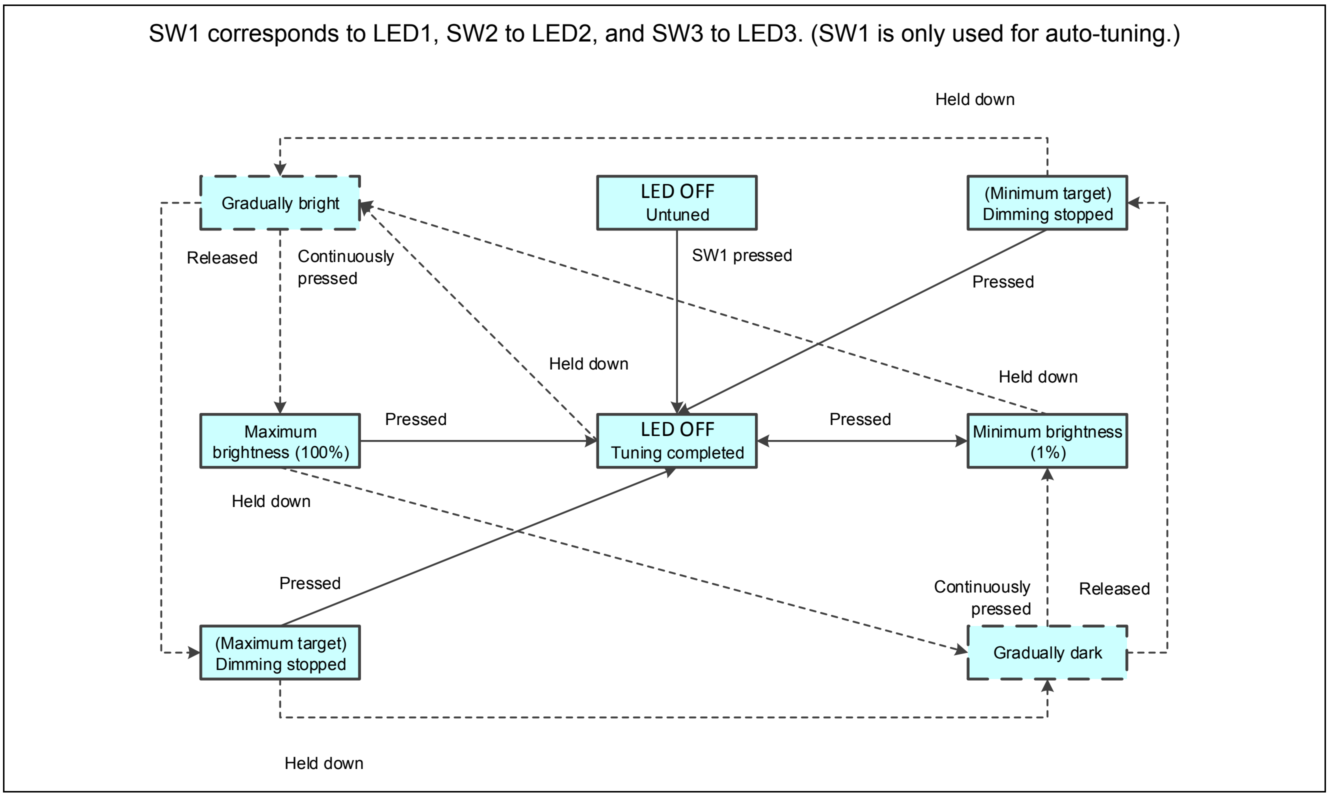


Figure 3 Transition of LED Dimming Level Based on SW Operation

3.7.2 Constant current control

TAU channels 0 and 1, timers KB0 and KB1, and A/D converter start operating after initialization. When the operation starts, the main loop program calculates the dimming target level (‘shAdClednTgt’ Note 2) of LED channels, which is determined based on the SW operation.

The feedback processing controls the constant current in LEDn Note 1 by adjusting the duty of the PWM outputs in an interrupt service routine that handles an interrupt by TAU channel 0 (INTTM00) that occurs every 64 µs.

The result of the A/D conversion stored in ‘shAdLedn’ Note 2 is compared with the last result stored in ‘shAdTempLedn’ Note 2 and the target level ‘shAdClednTgt’ Note 2 in a feedback process of PI (Proportional-Integral) control. See *3.9* for details about PI control.

By using this feedback process, the sense voltage can be made closer to the target level voltage. If the target level radically changed, the feedback process is executed multiple times before the sense voltage reaches the target.

The analog input channel shifts to one of the feedback channels when the TM00 interrupt occurs (every 64 µs). The program then updates the ‘ucFeedbackCnt’ variable to shift the input channel to the next feedback channel at the next interrupt service routine iteration.

Notes 1 n = 1, 2, 3 (LED1 lights red, LED2 lights green, and LED3 lights blue on the board.)
2 The string enclosed by single quotation marks is a global variable used in the sample program.

Figure 4 shows the circuit used for controlling the constant current in the LED. The RL78/I1A PWM output is used to switch the buck converter MOSFET by using a pre-driver, then the A/D converter input is used to measure the LED feedback current. The RL78/I1A CPU implements PI control to maintain a constant current in the LED. The target A/D conversion value is determined based on the SW operation described in *3.7.1*.

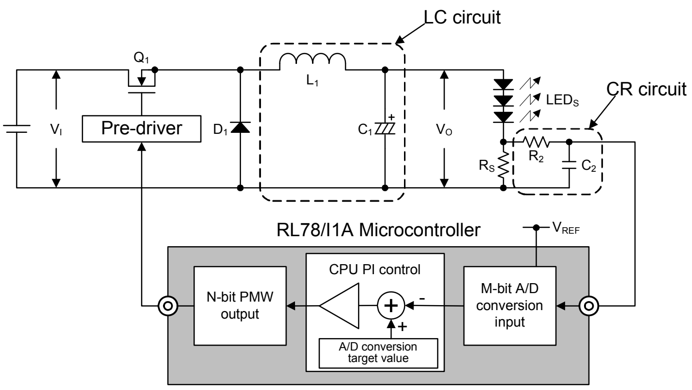


Figure 4 Buck Converter Circuit for Controlling LED Constant Current

3.8 PFC Control

3.8.1 Overview

(1) PFC control

(a) in Figure 5 shows the ideal waveform of the power supply voltage and current of the AC power supply. At this time, the voltage and current are in the same phase and show a sinusoidal waveform, and the power factor equals 1. *(b)* in Figure 5 shows the actual waveform of the power supply voltage and current of the power supply when PFC control is not used. At this time, the current continuity time is short and its peak current value is large. Accordingly, the head of the waveform of the voltage is flat. The following problems might occur if the power factor is low.

- The product might not comply with the regulations and might not be shipped due to harmonic generation.
- A thicker electric wire than usually necessary is required due to the higher peak current.
- The breaker is easily blown.

PFC needs to be controlled to improve the power factor. To control PFC, critical conduction mode (CRM) is used considering the relatively small number of parts required and less switching noise in general LED lighting systems. *(c)* in Figure 5 shows the waveform of the power supply voltage and the AC power supply current when PFC control is used. Turning on and off the power supply current repeatedly spreads the distribution of the current value, and controls the voltage and current so that the mean value is the same as that of the power supply voltage and show a sinusoidal waveform. The RL78/I1A microcontroller enables PFC control by using critical conduction.

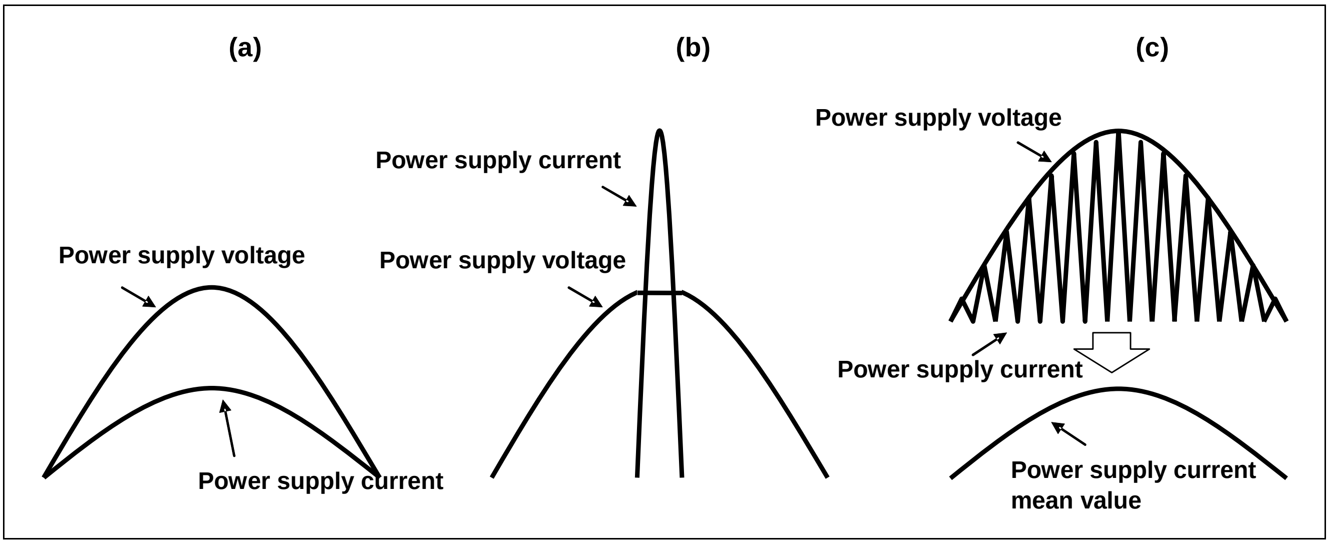


Figure 5 Waveform of Power Supply Voltage and Current, and PFC Control

(2) PFC Control using RL78/I1A

By combining the timer restart function in single operation mode and the A/D converter, PFC can be controlled in critical conduction mode by using the RL78/I1A microcontroller. Accordingly, an additional analog IC dedicated to PFC control is not required. In addition, the RL78/I1A microcontroller not only controls PFC but also LEDs, i.e., the PFC output load. Load variation can therefore be foreseen and controlled in advance. Compared with the feedback executed after the load varies, the voltage variation can be kept smaller when the load changes.

Figure 6 shows an example of flyback-converter-type PFC circuit configuration using the RL78/I1A microcontroller. In this figure, three pins are required for PFC control: PFC output (TKBO21 pin), zero current detection input (INTP20 pin), and PFC output voltage monitoring input (ANI7/CMP4P pin).

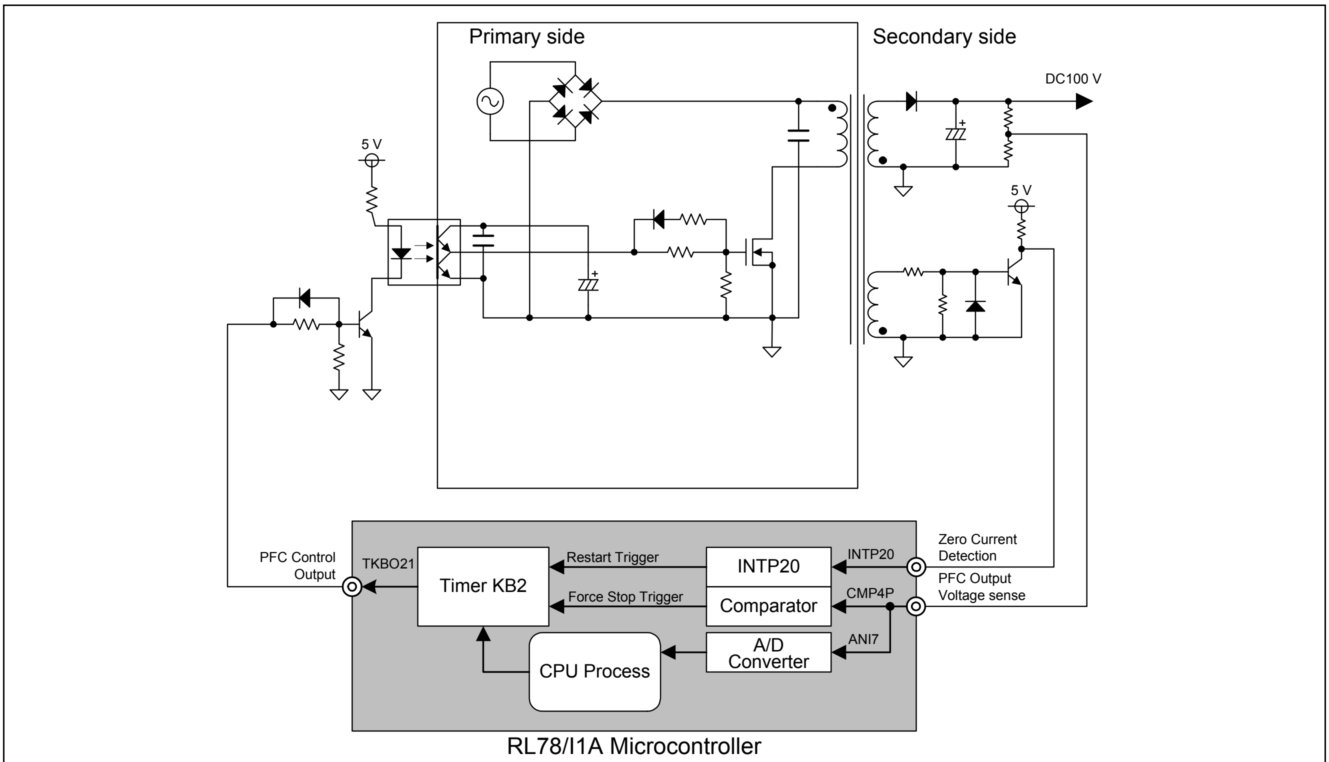


Figure 6 Example of Flyback-converter-type PFC Circuit Configuration Using the RL78/I1A Microcontroller

Figure 7 shows a waveform when PFC is controlled in this configuration. Here, PFC output (TKBO21 output) is on, and I_{ON} equals (V_{IN} / L) × t_{ON}. Accordingly, I_{ON} is in proportion to V_{IN} by keeping the t_{ON} output time constant. The waveform of the I_{ON} peak current I_{PEAK} is sinusoidal in the same phase as V_{IN}. Additionally, $I_{AVERAGE}$ equals I_{PEAK} / 2 and is also in proportion to V_{IN} because the current waveform is triangular. The average current waveform is therefore sinusoidal in the same phase as the power supply voltage waveform, and a waveform with a power factor close to 1 can be obtained.

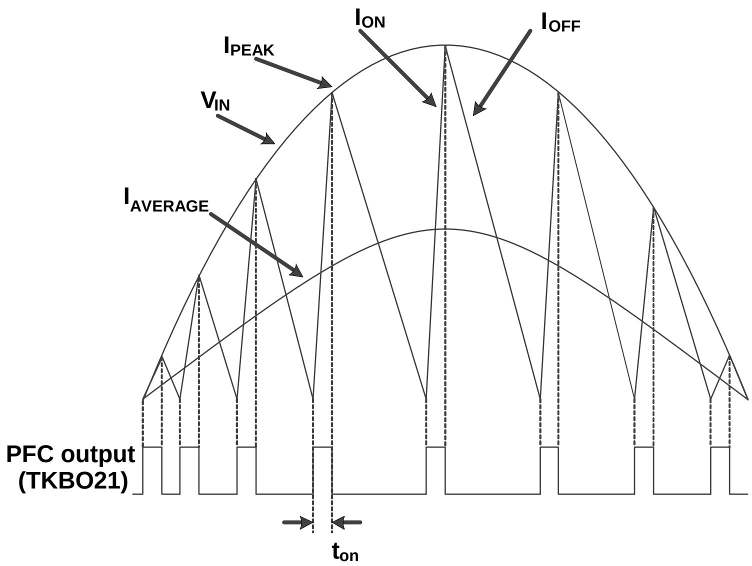


Figure 7 Waveform of PFC Control Using the RL78/I1A Microcontroller

The following peripheral hardware and function of the RL78/I1A microcontroller are used for PFC control:

- 16-bit timer KB2: PFC output
- A/D converter: PFC output voltage monitoring

The features of PFC control using the peripheral hardware are as follows:

- By using 16-bit timer KB2 in single operation mode, PFC output can be turned on automatically (without software processing) when a zero current is detected.
- 64 MHz can be selected for the count clock of 16-bit timer KB2, and the ON time of PFC output can be controlled in 15-ns units. The PFC output restart cycle while no zero current has been detected can flexibly be set up to approx. 1.02 ms in 15-ns units.
- 16-bit timer KB2 can change the ON time of PFC output without stopping timer operation.
- The A/D converter with a resolution of up to 10 bits can detect the PFC output voltage.

3.8.2 PFC control when the load changes (feed forward control)

Figure 8 shows a PFC control waveform when the load changes such as when an LED is dimming, turns on, or turns off. Because LED lighting is controlled by the RL78/I1A microcontroller, you can foresee the timing and degree of load change. Accordingly, PFC output at the same time as dimming i.e., foreseen control, can be performed to reduce the variation in PFC output voltage even when the load is greatly changed by dimming.

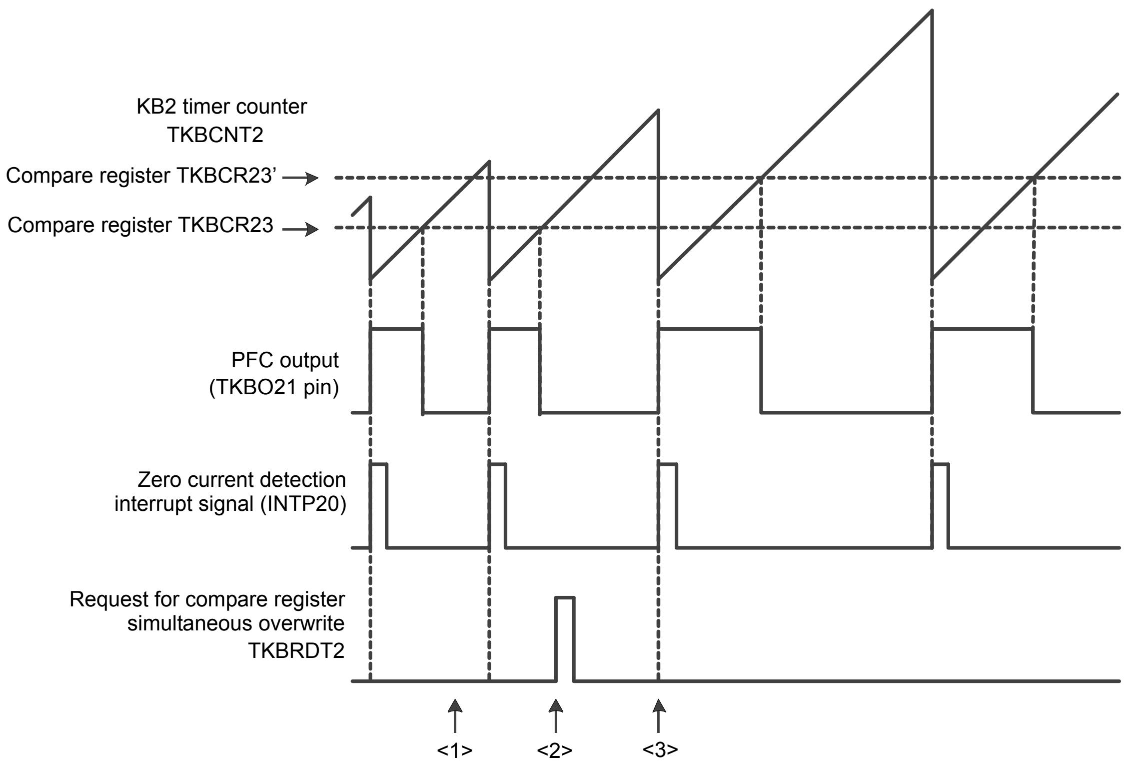


Figure 8 Waveform of PFC Control When the Load Changes

An operation overview is provided below:

<1> Increase or reduce the ON time (TKBCR23 in Figure 8) of the PFC output to the value (TKBCR23’ in Figure 8) corresponding to the load.

<2> To reflect the update of <1> in the actual output, write 1 to bit 0 (TKBRDT2) of trigger register 2 (TKBTRG2) and request a compare register simultaneous overwrite.

<3> The ON time of the PFC output is updated when a zero current detection interrupt occurs next time.

3.8.3 Auto-tuning

Auto-tuning must be performed before dimming of LEDs and PFC control start. This processing starts PFC control, sets the load on LEDs on the evaluation board to the maximum, and then calculates the load ratio of the LEDs. The calculated load ratio for PFC feed forward control is used during normal dimming. All LEDs are turned off after auto-tuning completes.

Calculate the LED1 load ratio as described below. In fact, the load ratio for each LED is to be calculated.

1. In LED1 feedback control cycles, add the feedback A/D value of LED1 to 'ulSumCled1'[Note], and add the PWM duty value of LED1 to 'ulSumDled1'[Note 1].

```
; ulSumCled1 += (unsigned long)(shAdLed1 - shAdOffsetLed1);
; ulSumDled1 += TKBCR01;
```

2. In the INTP0 interrupt cycle (zero cross), add the PWM duty value of PFC to 'ulSumDpfco' [Note].

```
; ulSumDpfco += TKBCR23;
```

3. When the INTP0 interrupt counter reaches 128, check that the maximum LED1 A/D value to be fed back and divided by 2 is smaller than the accumulated LED1 A/D value to be fed back and divided by the number of addings, and roughly judge whether or not the LED is connected.

```
; if ((AD_CLED1_TGT >> 1) < (ulSumCled1 / ulSumCled1Count))
```

4. After the LED is judged to be connected, calculate the value in proportion to the LED1 load by multiplying the integrated LED1 PWM value by the maximum LED1 A/D value to be fed back.

```
; ushPowLed1 = (unsigned short)((ulSumDled1 * AD_CLED1_TGT) >> 24);
```

   Considering an overflow that occurs due to the variable data type, shift to the right by 24 bits, and obtain the effective value only.

5. Add the LED1 load value to the value that holds the total load of the three LEDs.

```
; ushPowTotal += ushPowLed1;
```

6. To calculate the mean value from the integrated PFC PWM duty value added in 128 cycles, divide the value by 128.

```
; ulSumDpfco = ulSumDpfco >> 7;
```

7. Calculate the ratio of the LED1 load against the total load by dividing the maximum LED1 A/D value by the average PFC PWM duty value in 128 cycles.

```
; ulDpfcLed1 = ((unsigned long)(ushPowLed1) * ulSumDpfco) / ushPowTotal;
; ulDpfcLed1 = (ulDpfcLed1 << 16) / AD_CLED1_TGT;
```

Note The string enclosed by single quotation marks is a global variable used in the sample program.

3.8.4 PFC output voltage feedback

The PFC output voltage must be monitored by using the A/D converter and the feedback must be controlled while PFC output is on so that the PFC output voltage value is constant. Basically, the load greatly changes only by intentional dimming or turning on and off the LED. Variation of the PFC output voltage in this case can be reduced by using feed forward control as described in *3.8.2*. Accordingly, feedback is controlled when brightness of the LED is constant, i.e., when the LED load is approximately constant. Feedback is controlled based on PI control similarly to LED control. See *3.9* for details about PI control.

3.9 Feedback Control by PI Control

3.9.1 PI control

The constant current in LEDs, dimming (including turning on and off), and PFC output voltage can be controlled by using feedback processing based on the PI control.

The general equation for PI feedback is as follows.

For how to calculate the coefficients A1 and A2, see *3.9.2 Calculation of the PI control formula's coefficients*.

$$D(n) = D(n-1) + A_1 \cdot E(n) + A_2 \cdot E(n-1)$$

D (n): Latest PWM output duty
D (n-1): Previous PWM output duty
E (n): Latest error value = (target A/D conversion value) – (Latest measured A/D conversion value)
E (n-1): Previous error value = (target A/D conversion value) – (Previous measured A/D conversion value)
A1, A2: Coefficients

1) Control of the constant current in LEDs

The target value of LED current I_{LED} is determined based on the target A/D conversion value. When the target A/D conversion value is X_{TARGET} Note, it is obtained as follows.

$$X_{TARGET} = \frac{(I_{LED} \times 8) \times R_S}{V_{REF}} \times 2^M$$

For example, when the constant current is controlled based on the LED current I_{LED} = 350 mA, set target A/D conversion value X_{TARGET} to 744 assuming that the sense resistor R_S = 1.3 Ω, A/D converter reference voltage V_{REF} = 5 V, and A/D conversion resolution M = 10 bits.

The LED current feedback measurement value is amplified through the programmable gain amplifier with a gain of 8. Therefore, the target A/D conversion value X_{TARGET} needs to be multiplied by 8.

2) Dimming control of the LEDs

Current dimming can be controlled by changing the target value of the LED constant current. In other words, the target A/D conversion value X_{TARGET} can be changed for dimming. The target value of PI control is changed as a result, and the RL78/I1A microcontroller controls feedback to obtain an ideal X_{TARGET} value. To change the LED current from 350 mA to 100 mA, for example, change the X_{TARGET} value from 744 to 216.

3) PGA input offset voltage compensation

When using the programmable gain amplifier (PGA), introducing an input offset voltage might cause the input voltage to be amplified to increase or decrease by 5 mV to 10 mV. In this case, the PGA will not supply an accurate voltage to the feedback loop.

The sample program takes a positive offset voltage into account and compensates for it.

During the first LED feedback iteration on each channel, that is, when LEDs are still off, the sample program calculates the LED feedback voltage values and saves these values in 'shAdOffsetLedn'. These values represent the offset voltages induced by the PGA when no current is flowing through the LEDs. In the LED feedback processing, we subtract these values from the next LED feedback voltage values, when LEDs light, to cancel the offset voltage.

Furthermore, the RL78/I1A AC/DC LED control evaluation board has been designed to counteract the effect of positive offset voltages by means of pull-up resistors on each channel feedback circuit (R110, R210 and R310).

The overview of the operation is as follows.

<1> Start PGA amplification.
<2> Start A/D conversion of the LED feedback voltage (measured through the sense resistor).
<3> Read the target A/D conversion value.
<4> Calculate "A2 × E(n−1)" of PI control taking the offset voltage into account.
<5> If an over-current occurs in the LED as a result of A/D conversion, stop processing of LED output. (In this case, do not perform processing for PI control from <6> to <8> below.)
<6> If an over-current does not occur in the LED as a result of A/D conversion, calculate "A1 × E(n) + A2 × E(n−1)" of PI control still taking the offset voltage into account.
<7> Compare the last PWM output duty "D(n−1)" with the result of "A1 × E(n) + A2 × E(n−1)" obtained in <6>.
<8> If the result of "D(n−1) + A1 × E(n) + A2 × E(n-1)" is within the range between the maximum and minimum PWM duty values, set the duty D(n) according to the error calculation result, or set the duty D(n) to the maximum or minimum value otherwise.
<9> Update the PWM output duty setting.
<10> Save the A/D conversion value of the LED feedback voltage as the last one.

4) PFC output voltage control

The target A/D conversion value XTARGET for the PFC output voltage control is obtained as follows:

$$X_{TARGET} = \frac{\left(\frac{V_{PFCO}}{33}\right) \times 2^{M}}{\mathrm{V_{REF}}}$$

PFC output voltage V_{PFCO} = 100 V, and the PFC feedback voltage is divided by 33. Because A/D converter standard voltage V_{REF} equals 5 V and A/D conversion resolution *M* equals 10 bits, set target A/D conversion value X_{TARGET} to 620. (However, adjust the target A/D conversion value according to the actual environment because the value might vary due to individual differences.)

3.9.2 Calculation of the PI control formula's coefficients

This section describes how to calculate the coefficients in the PI control formula shown in *3.9.1*. The coefficients A1 and A2 can be obtained from the following equations.

$$A_1 = (\pi \times f_Z \times T + 1) \cdot K_P$$
$$A_2 = (\pi \times f_Z \times T - 1) \cdot K_P$$

π: Pi
f_Z: Zero point frequency
T: Feedback cycle
K_P: Proportional constant

In other words, the coefficients A1 and A2 can be calculated by determining the three parameters: f_Z, T and K_P. These parameters can be obtained from the gain of the LED control circuit and PFC control circuit.

1. Calculation of the PI control formula's coefficients in LED control circuit

1) Calculation of the zero point frequency (f_Z) from the pole point frequencies of the control circuit

As shown in Figure 4, the LED control circuit has two pole points, one for the LC circuit and one for the CR circuit. These pole point frequencies can be regarded as being equal to the respective cut-off frequencies. Assuming that the former is f_{C1} and the latter is f_{C2}, the following values can be obtained with: L_1 = 2.2 mH, C_1 = 33 µF, C_2 = 0.1 µF and R_2 = 220 Ω.

$$f_{C1} = \frac{1}{2\pi\sqrt{L_1 \cdot C_1}} = 0.6[kHz]$$
$$f_{C2} = \frac{1}{2\pi \cdot C_2 \cdot R_2} = 7.2[kHz]$$

We then choose a zero point frequency which is lower than these 2 frequencies, as shown below.

$$f_Z = 500Hz$$

2) Calculation of the feedback cycle (*T*) from the zero point frequency (f_Z)

Because of the sampling theorem, the sampling frequency (which is equal to the inverse of the feedback cycle *T*) must be twice or more than the zero point frequency f_Z. In other words, the relation between the feedback cycle T and the zero point frequency f_Z can be described with the following equation.

$$T < \frac{1}{2 f_Z}$$

Therefore, as f_Z = 500 Hz, the feedback cycle *T* must be less than 1 ms.

We also need to consider the CPU load dedicated to the feedback processing. A total of three LED channels require constant current feedback control. In this sample program, the CPU load is thus shared in cycles of 64 µs to perform feedback control for each LED channel as shown in Figure 9 below. The feedback cycle T is then set as follows.

$$T = 320\mu s$$

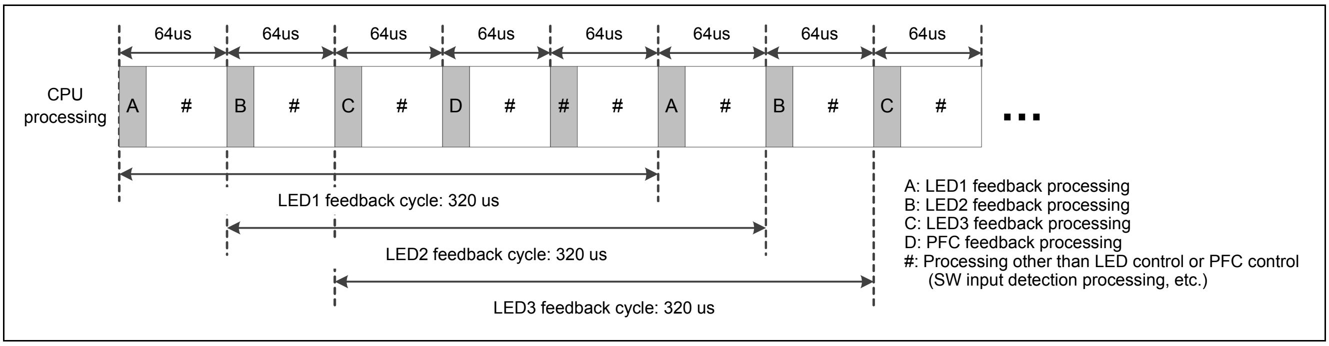


Figure 9 Image of the CPU Load Distribution for Feedback Processing

3) Calculation of the proportional constant K_P from the gain (ADC input / PWM output) of the microcontroller

The gain (A/D converter input / PWM output) of the microcontroller can be obtained by looking at the change in LED current for a particular A/D conversion resolution and PWM resolution.

First, we need to determine the change in LED current for a specific A/D conversion resolution. When the LED current is I_{LED}, the A/D conversion result of the feedback voltage (through the sense resistor R_S) is *X*, the A/D conversion resolution is *M* bits, and the ADC reference voltage is V_{REF}, we can establish the following equation.

$$I_{LED} \cdot R_S = \frac{V_{REF} \cdot X}{2^M}$$

Here, we call i_{AD} the change in LED current for an A/D conversion value equal to 1 (X = 1), and we get the below result.

$$i_{AD} = \frac{V_{REF}}{R_S \cdot 2^M}$$

Next, we need to determine the change in LED current for a specific PWM resolution. When the LED current is I_{LED}, total LED forward voltage is V_{FT}, the input voltage is V_I, the (PWM output duty register value +1) is *Y*, and the PWM output resolution is *N* bits, we can establish the following equation.

$$I_{LED} \cdot R_S + V_{FT} = \frac{V_I \cdot Y}{2^N}$$

Here, we call i_{PWM} the change in LED current for a PWM duty value equal to 1 (Y = 1), furthermore the LED forward voltage remains constant, so we get the below result.

$$i_{PWM} = \frac{V_I}{R_S \cdot 2^N}$$

Therefore, the gain i_{PWM} / i_{AD} can be deducted from the above equations.

$$\frac{i_{PWM}}{i_{AD}} = \frac{V_I}{V_{REF}} \cdot 2^{(M-N)}$$

Considering that the A/D conversion resolution M is 13 bits (10 bits of the ADC + 3 bits of the PGA because of the 2^3 = 8 amplification gain), the PWM output resolution N is 12 bits (8 bits of the PWM + 4 more bits because of the dithering function), the input voltage V_I is 5 V, and the A/D converter reference voltage V_{REF} is 5 V, we then get the following gain result (A/D converter input / PWM output).

$$\frac{i_{PWM}}{i_{AD}} = 2$$

The proportional constant K_P must be set to a value smaller than the inverse of this gain.

$$K_P < \frac{1}{\left(\frac{i_{PWM}}{i_{AD}}\right)}$$

We choose K_P as follows.

$$K_P = 0.05$$

From the above results, we can then calculate the PI control coefficients A1 and A2 in the LED control.

$$A_1 = 0.075132$$
$$A_2 = -0.024868$$

In the sample program, both coefficients are multiplied by 2^{16} (= 65,536) as well as the PWM duty and error values in order to get integer variables and make computations easier.

$$A_1 = 4923$$
$$A_2 = -1629$$

2. Calculation of coefficients of the PI control formula in the PFC control circuit

Similarly, also calculate the PI control coefficients in PFC control.
In the PFC control circuit, f_Z, T, and K_P are the following values:

$$f_Z = 1Hz$$
$$T = 320\mu s$$
$$K_P = 1.0$$

From the above results, we can then calculate PI control coefficients A1 and A2 in PFC control.

$$A_1 = 1.001$$
$$A_2 = -0.999$$

In the sample program, both coefficients are multiplied by 2^{16} (= 65,536) as well as the PWM duty and error values in order to get integer variables and make computations easier.

$$A_1 = 65601$$
$$A_2 = -65470$$

4. Flowchart

4.1 Main Processing

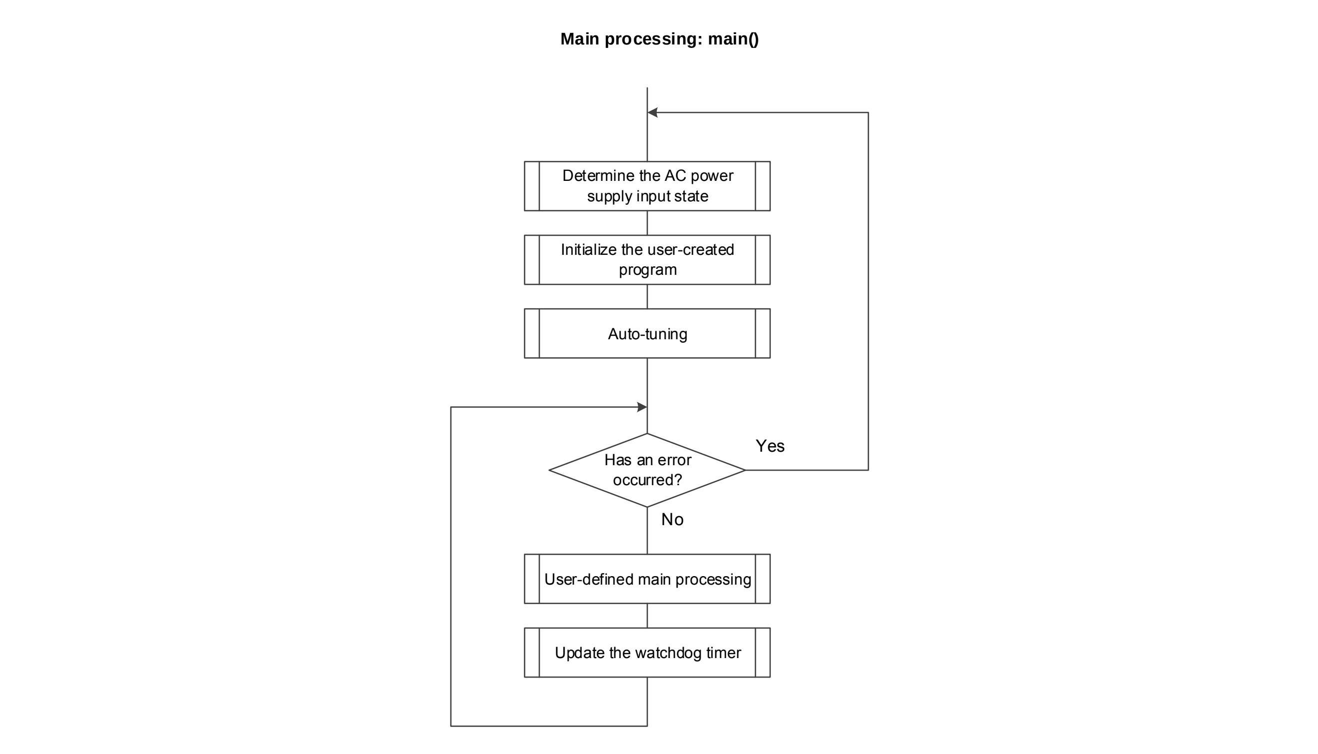


Figure 10 Main Processing Flowchart

4.2 User-Defined Main Processing

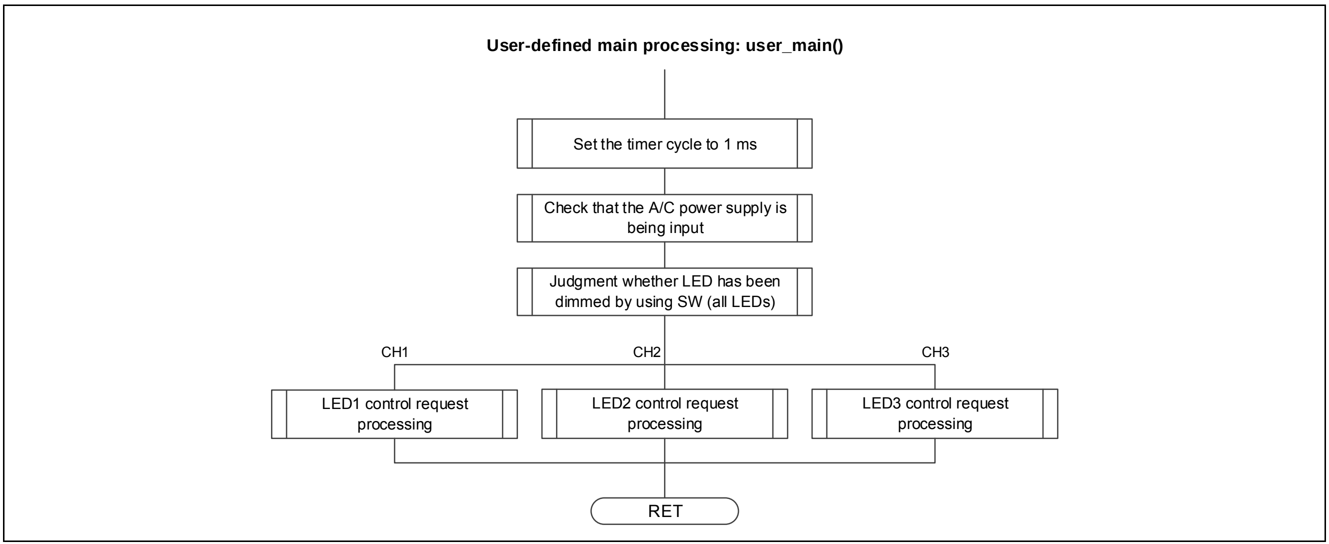


Figure 11 User-Defined Main Processing Flowchart

4.3 User-Defined Initialization Processing

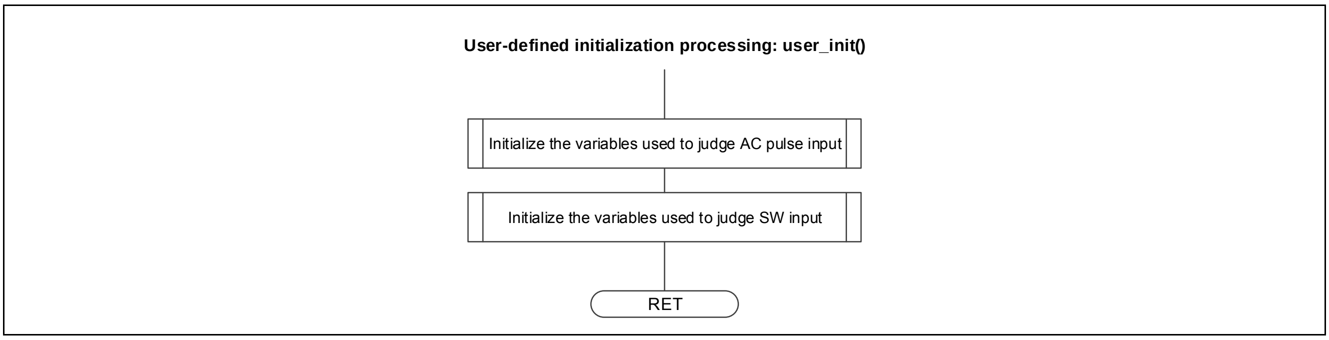


Figure 12 User-Defined Initialization Processing Flowchart

4.4 INTP00 Interrupt

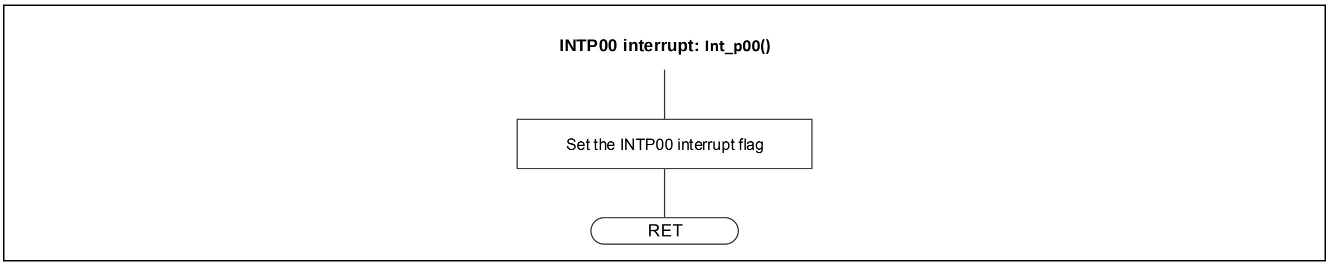


Figure 13 INTP00 Interrupt Flowchart

4.5 INTTM00 Interrupt

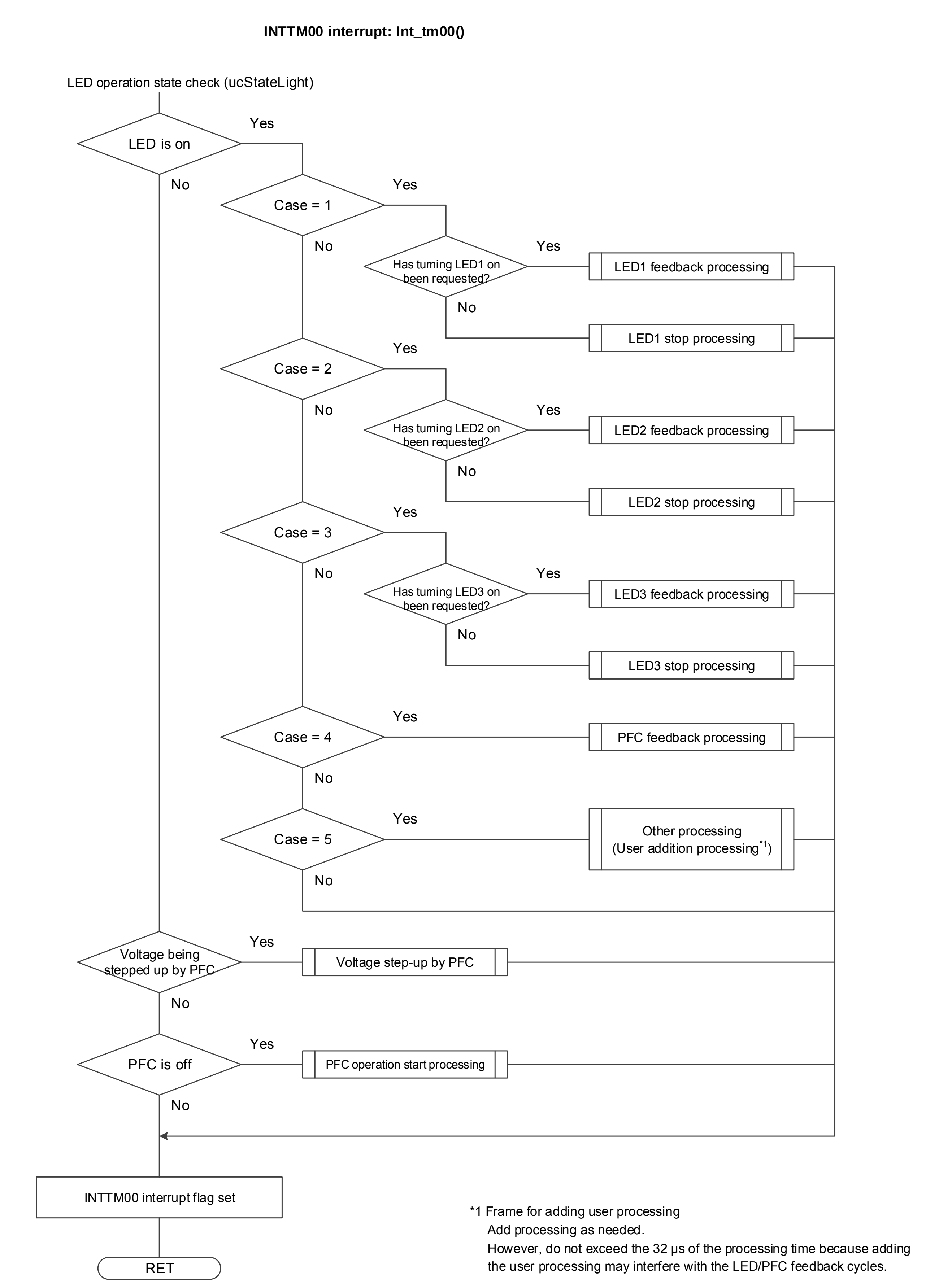


Figure 14 INTTM00 Interrupt Flowchart

4.6 AC Power Supply Input State Judgment

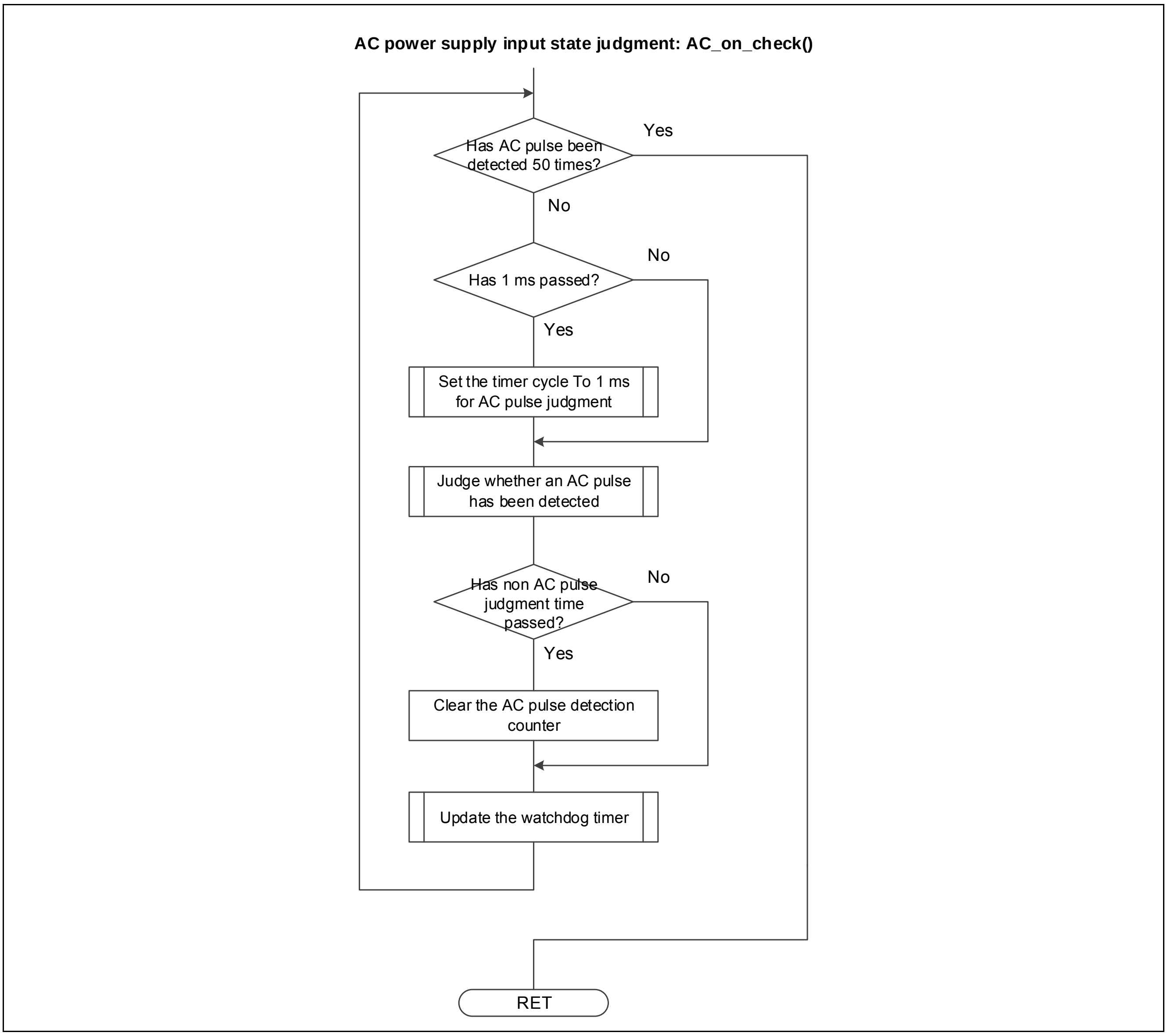


Figure 15 AC Power Supply Input State Judgment Flowchart

4.7 Auto-tuning

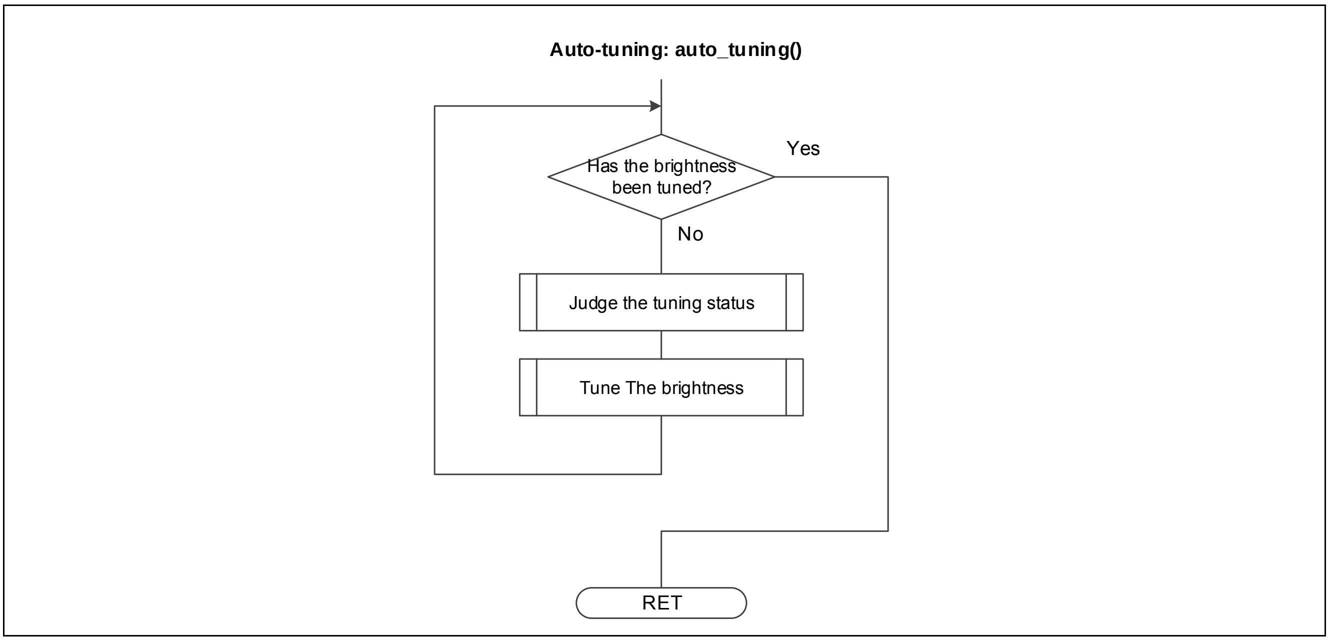


Figure 16 Auto-tuning Processing Flowchart

4.8 PFC Operation Start

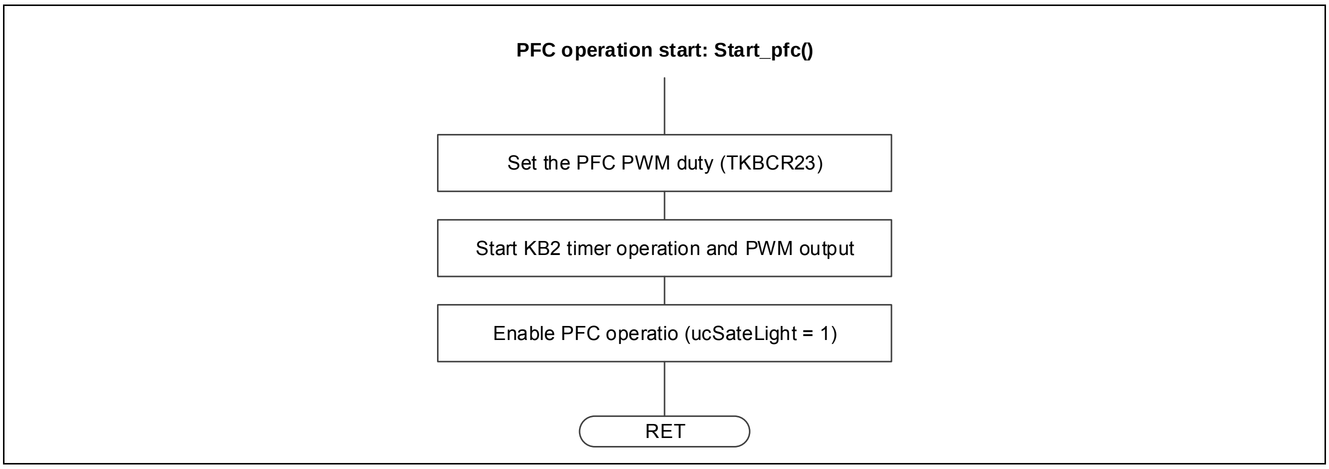


Figure 17 PFC Operation Start Flowchart

4.9 PFC Step-up Processing

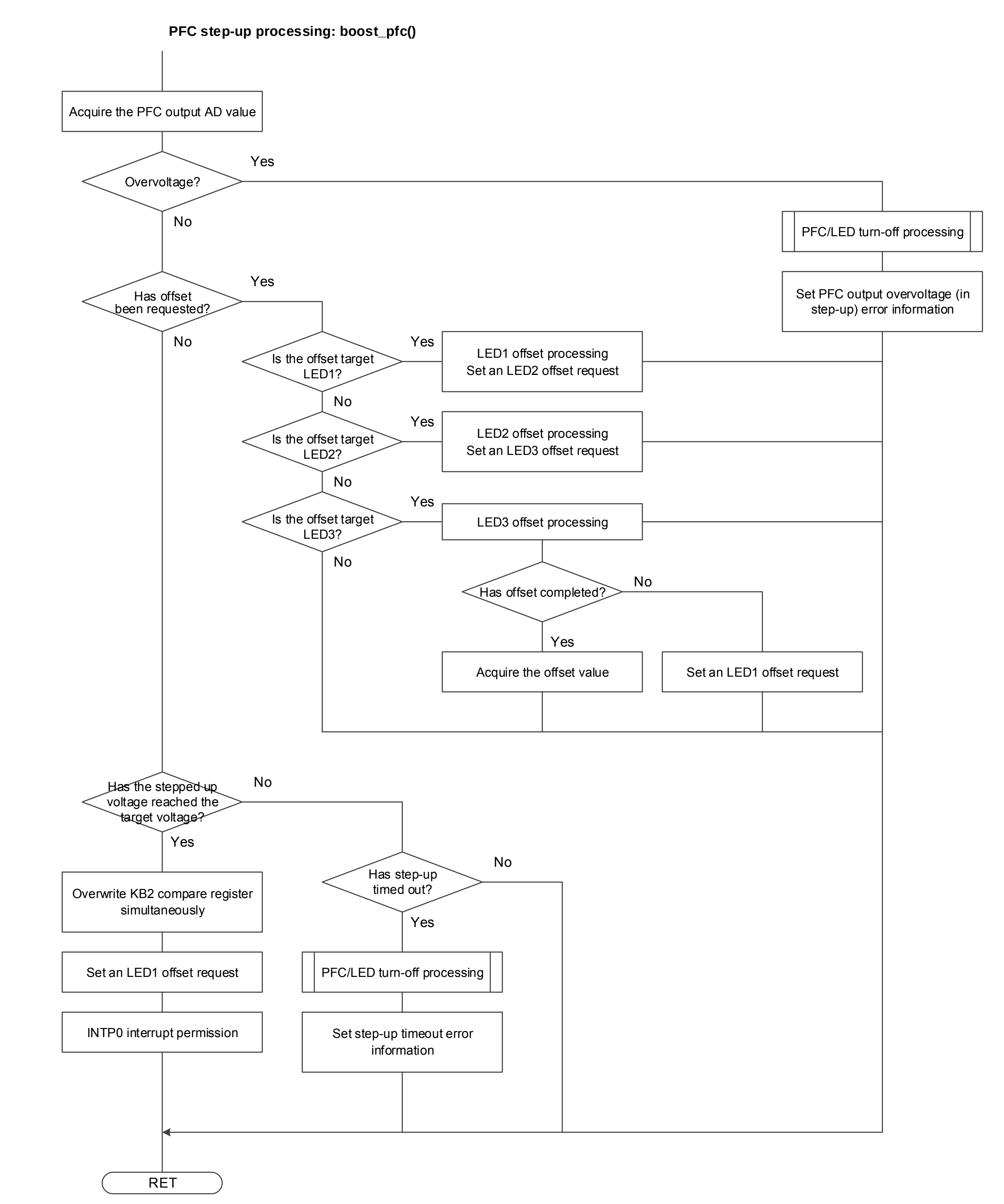


Figure 18 PFC Step-up Flowchart

4.10 LED1 Feedback Processing

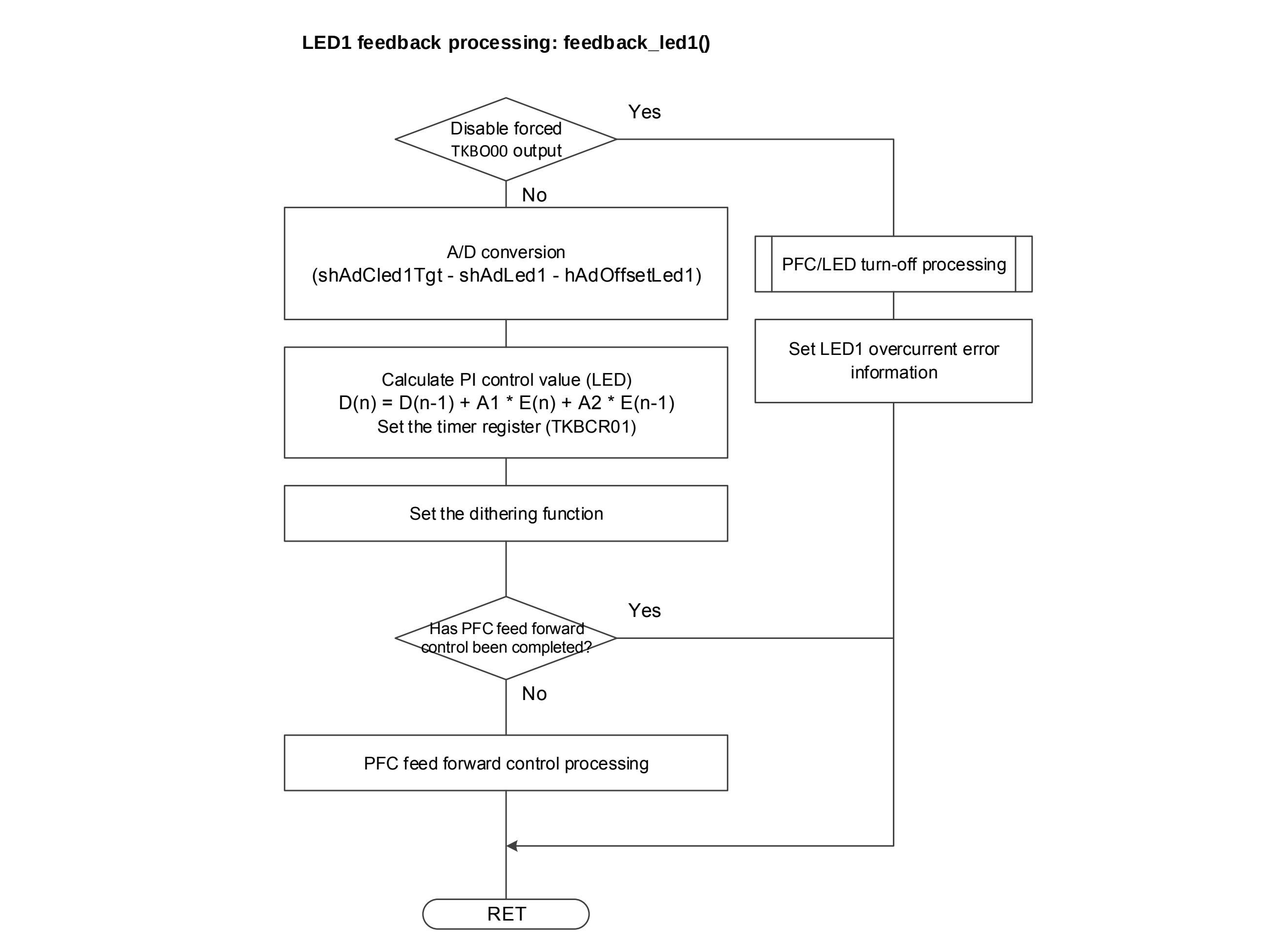


Figure 19 LED1 Feedback Processing Flowchart

4.11 LED2 Feedback Processing

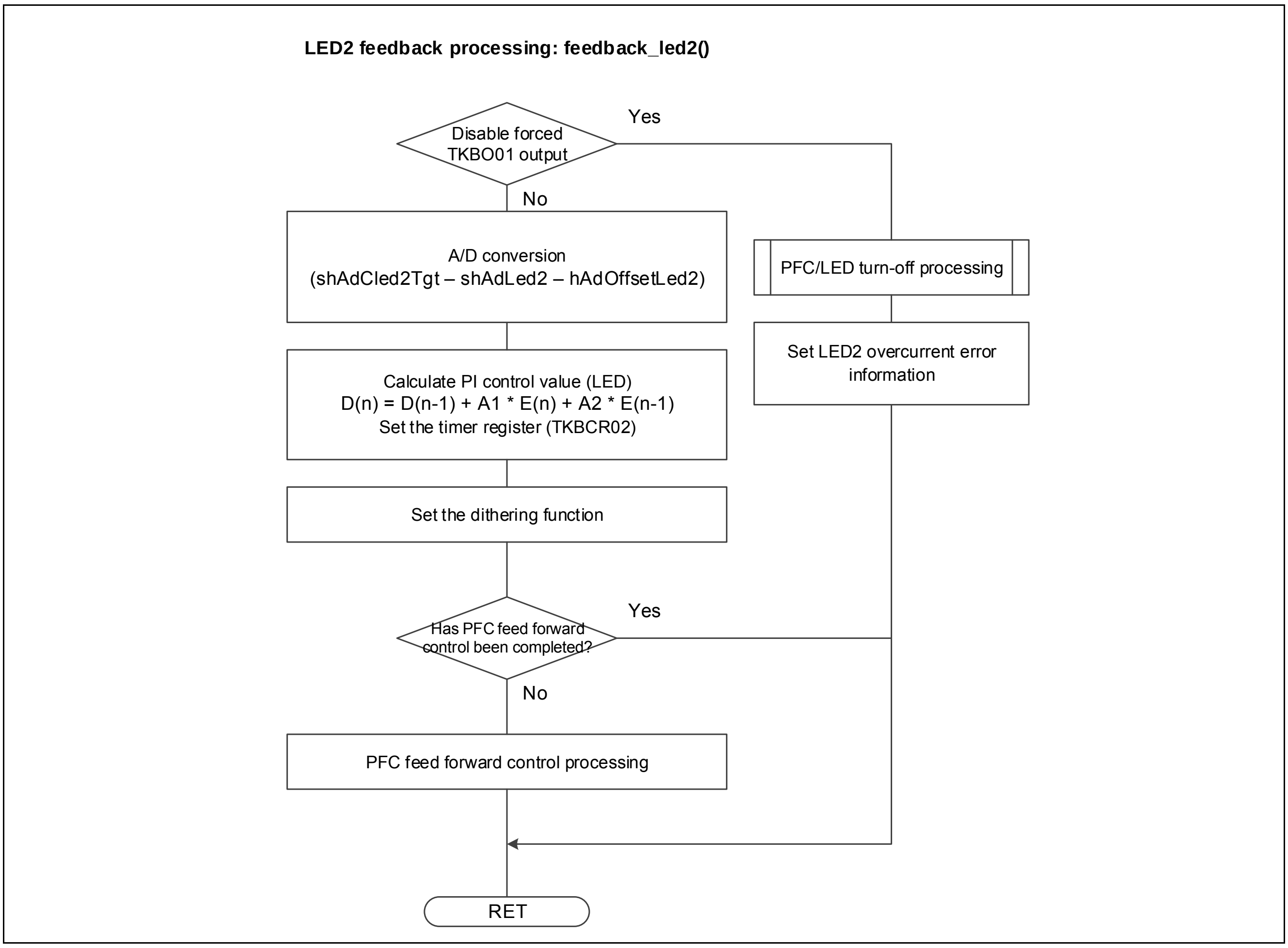


Figure 20 LED2 Feedback Processing Flowchart

4.12 LED3 Feedback Processing

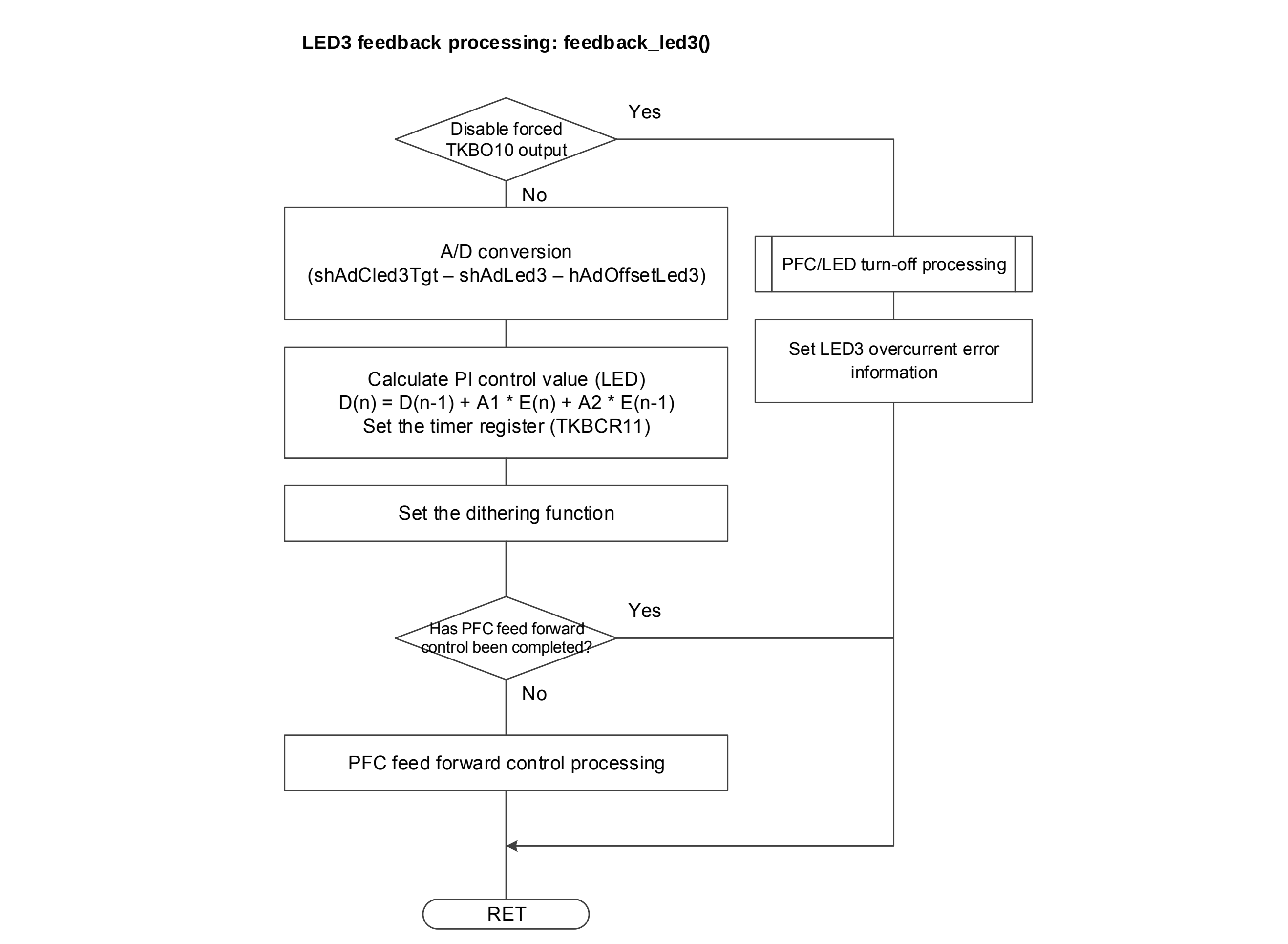


Figure 21 LED3 Feedback Processing Flowchart

4.13 PFC Feedback Processing

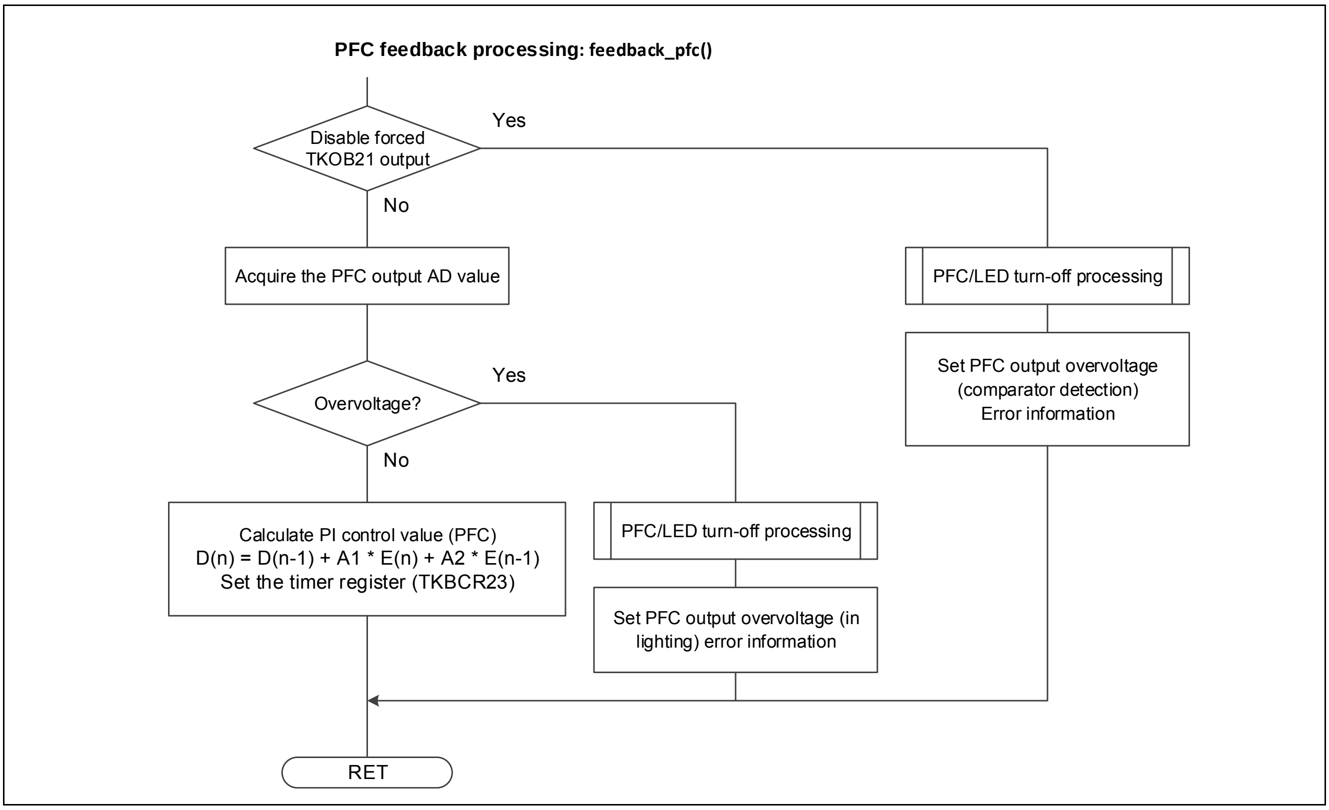


Figure 22 PFC Feedback Processing Flowchart

4.14 LED1 Stop Processing

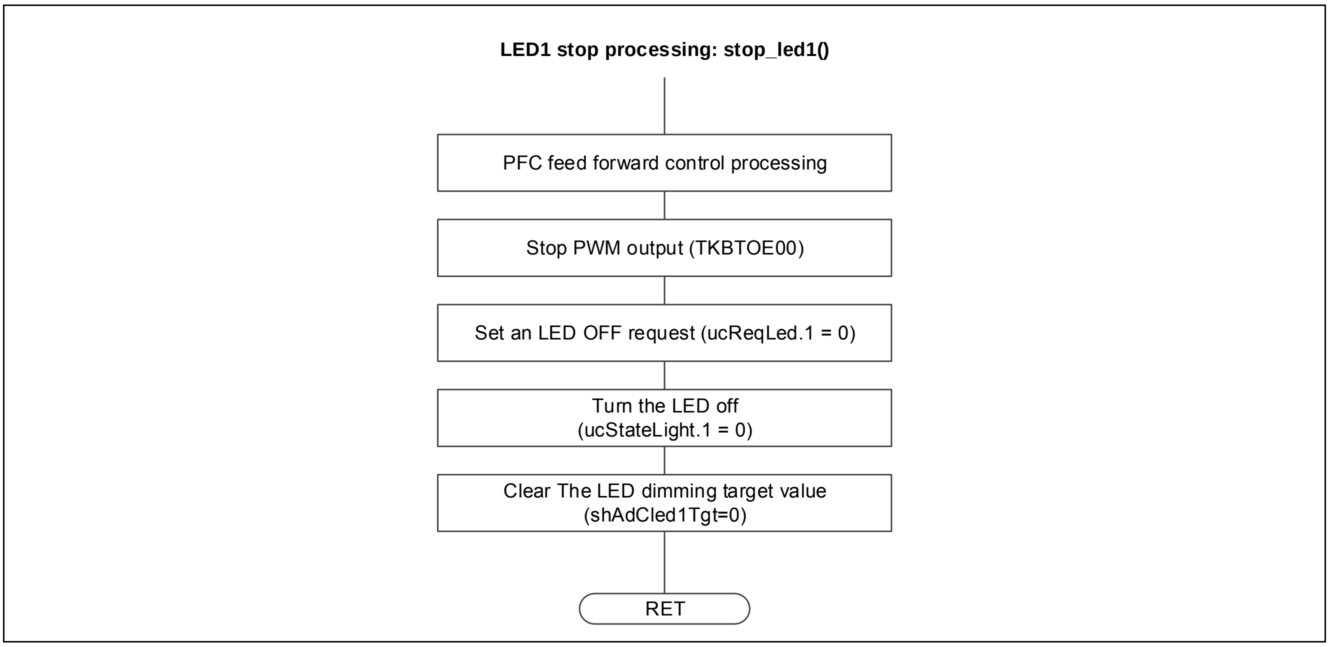


Figure 23 LED1 Stop Processing Flowchart

4.15 LED2 Stop Processing

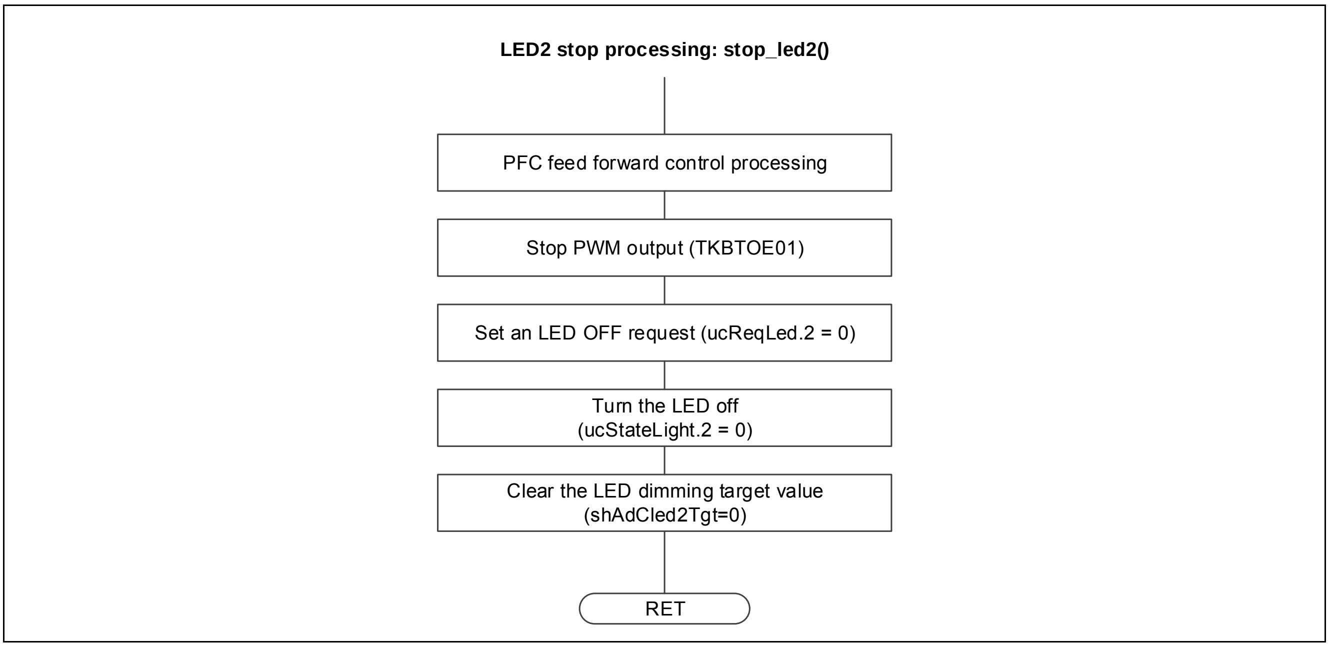


Figure 24 LED2 Stop Processing Flowchart

4.16 LED3 Stop Processing

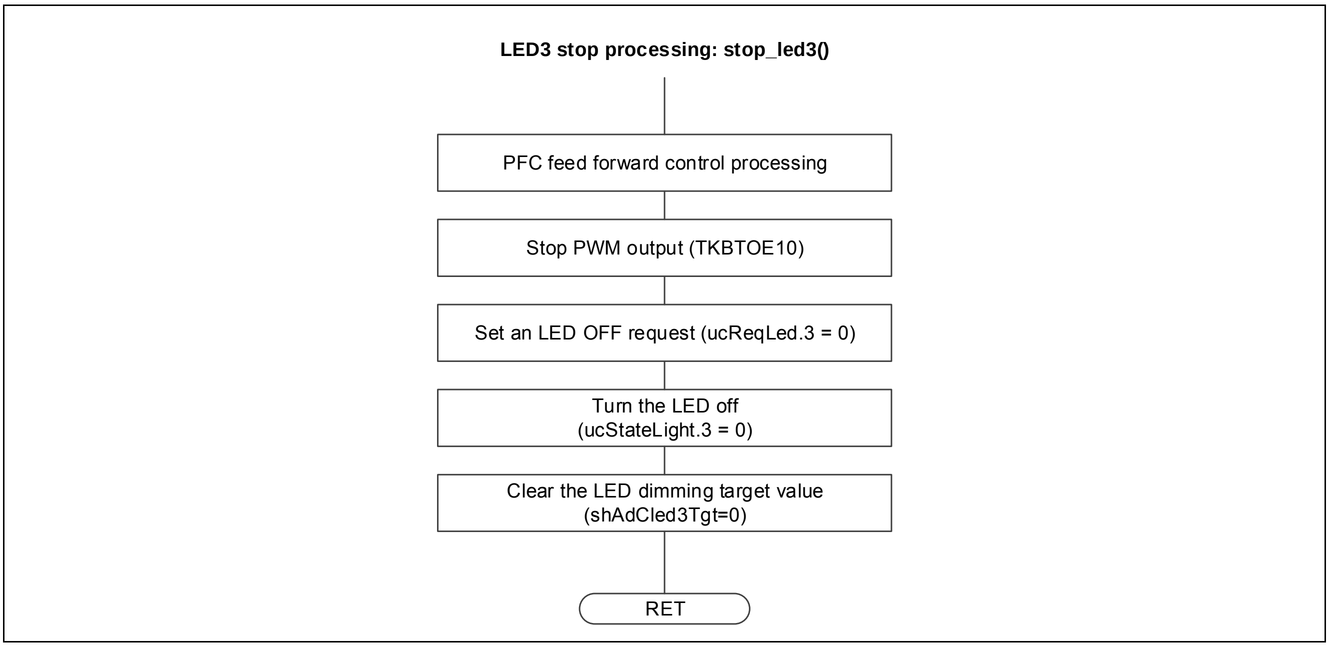


Figure 25 LED3 Stop Processing Flowchart

4.17 PFC/LED Light-off Processing

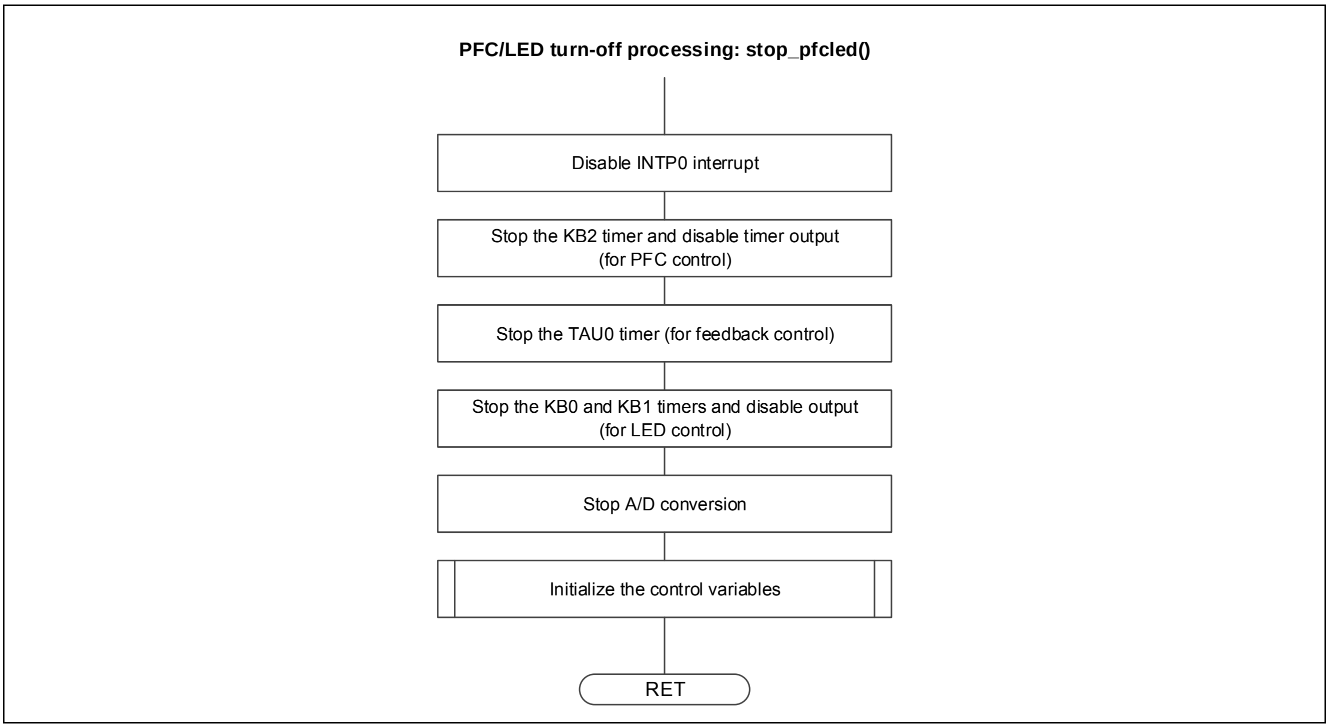


Figure 26 PFC/LED Light-off Processing Flowchart

4.18 LED1 Control Request Processing

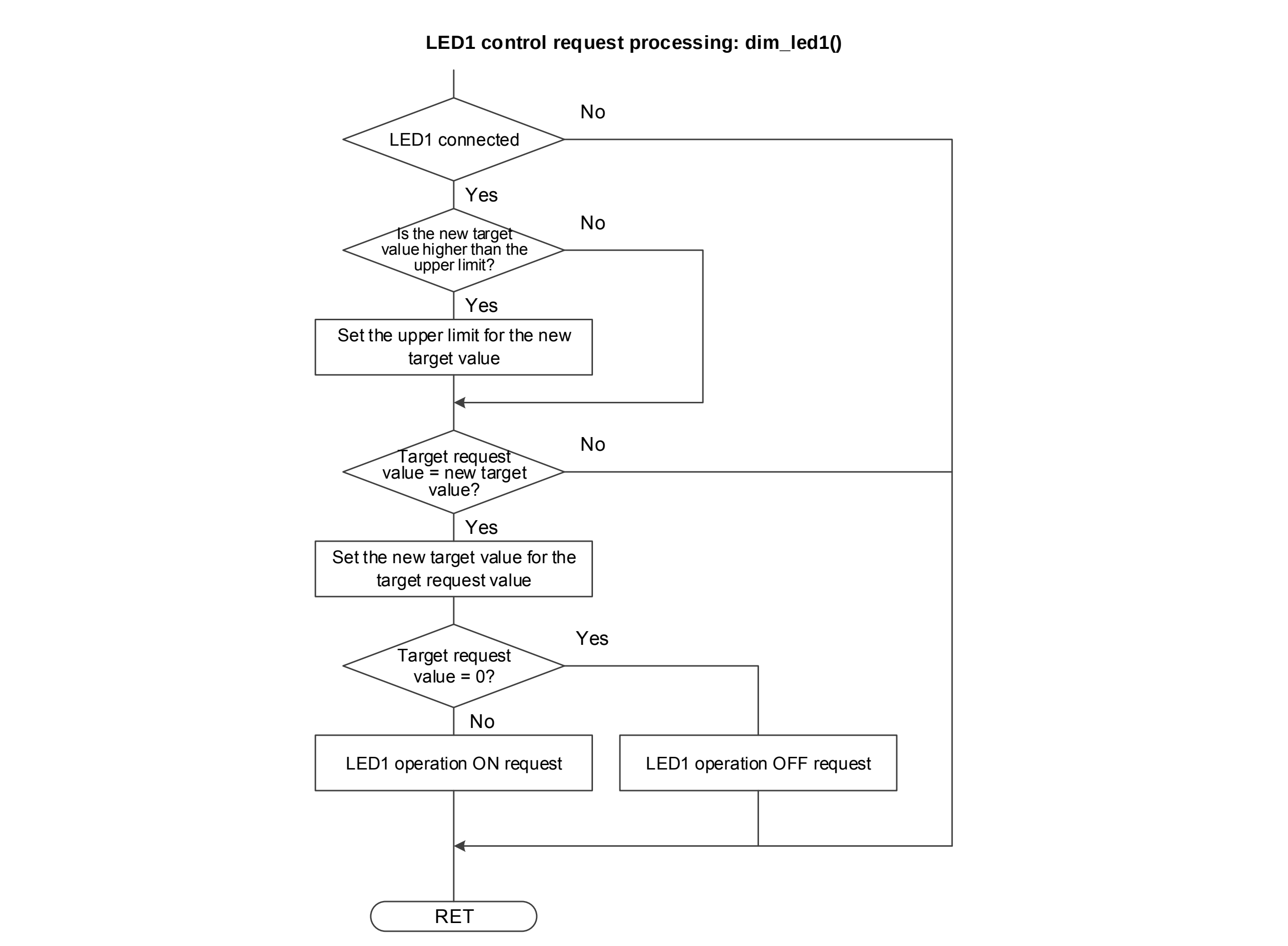


Figure 27 LED1 Control Request Processing Flowchart

4.19 LED2 Control Request Processing

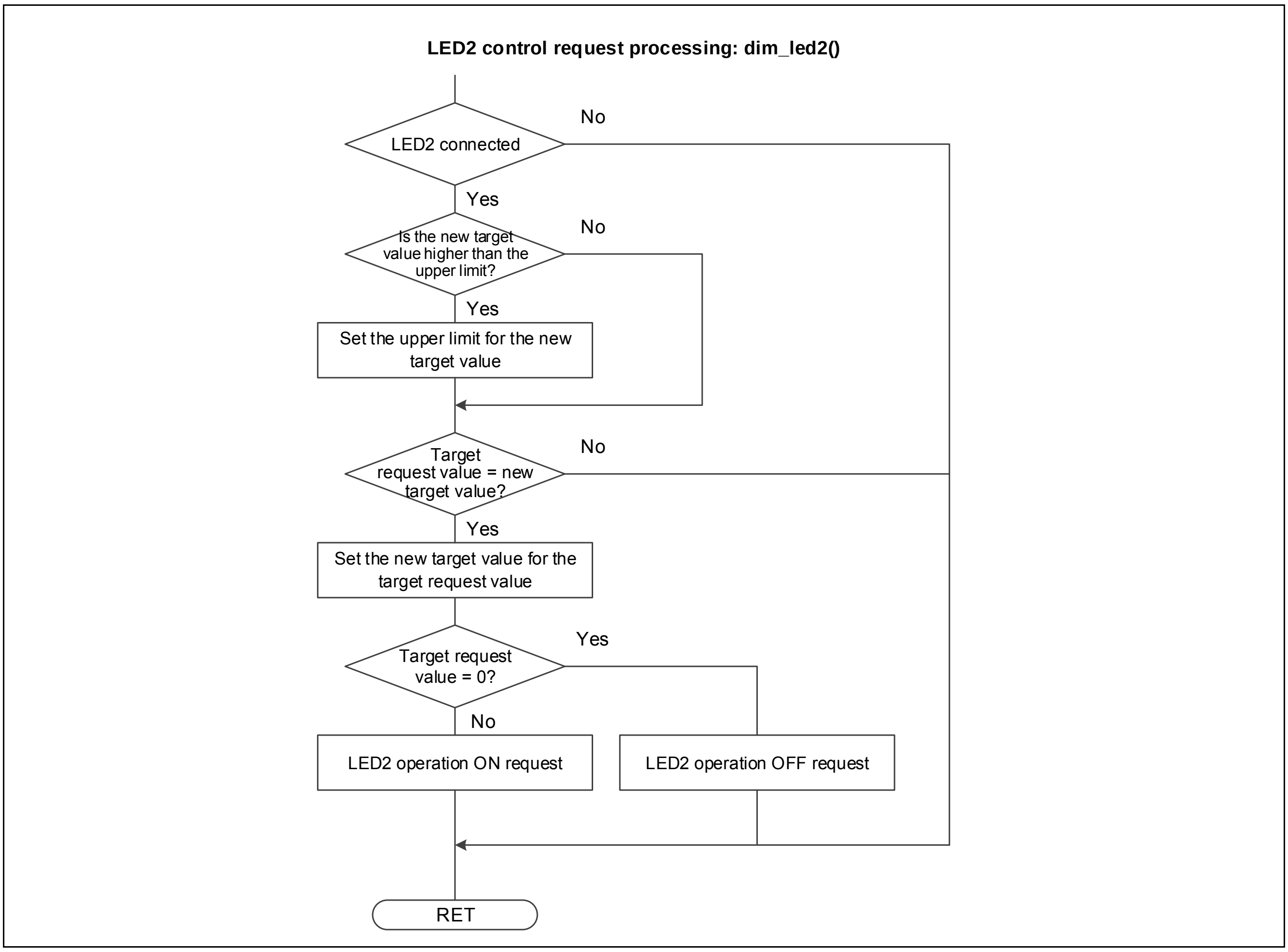


Figure 28 LED2 Control Request Processing Flowchart

4.20 LED3 Control Request Processing

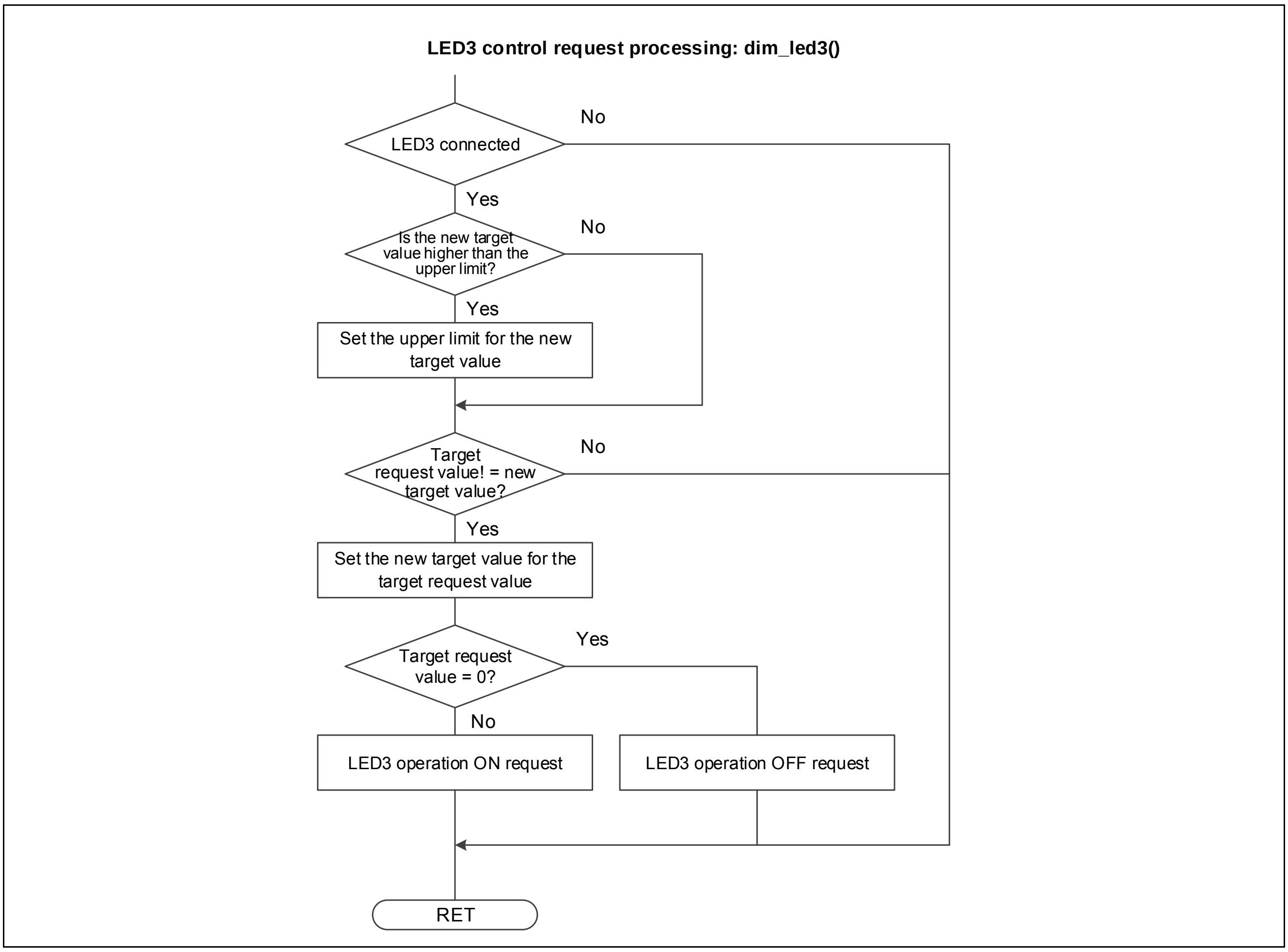


Figure 29 LED3 Control Request Processing Flowchart

4.21 Start of Dimming Control Timer Operation

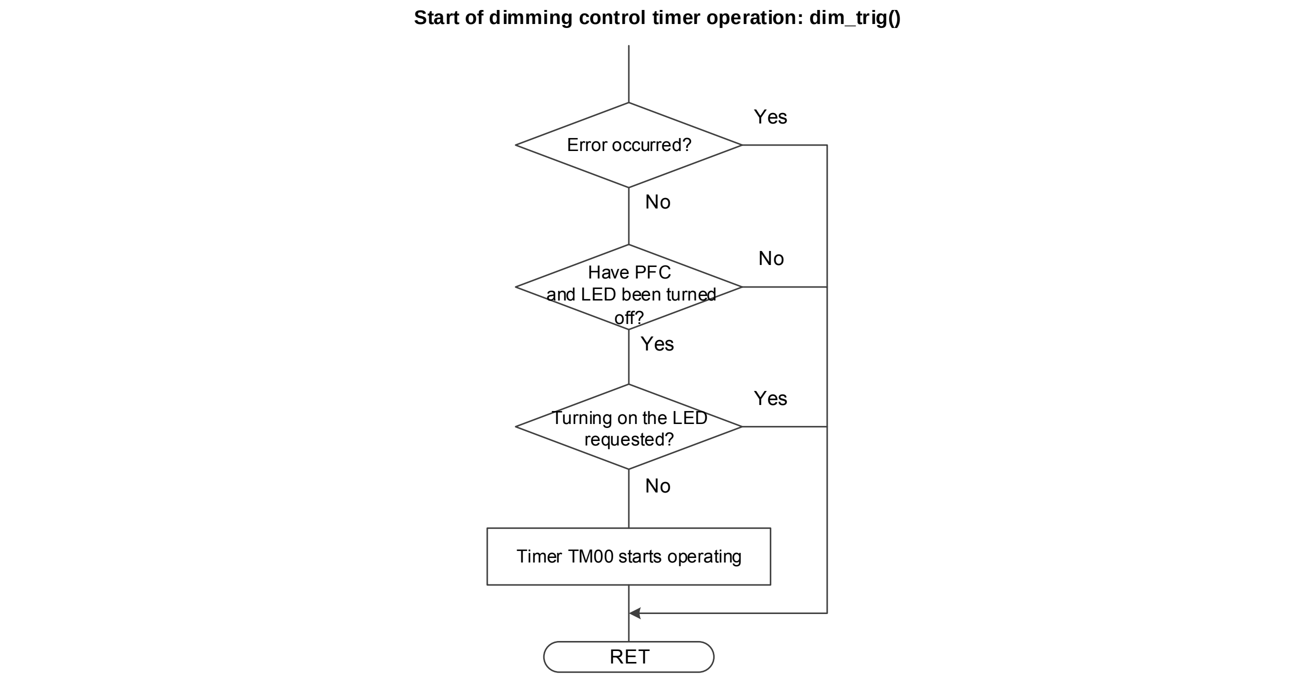


Figure 30 Start of Dimming Control Timer Operation Flowchart

4.22 SW Input Judging

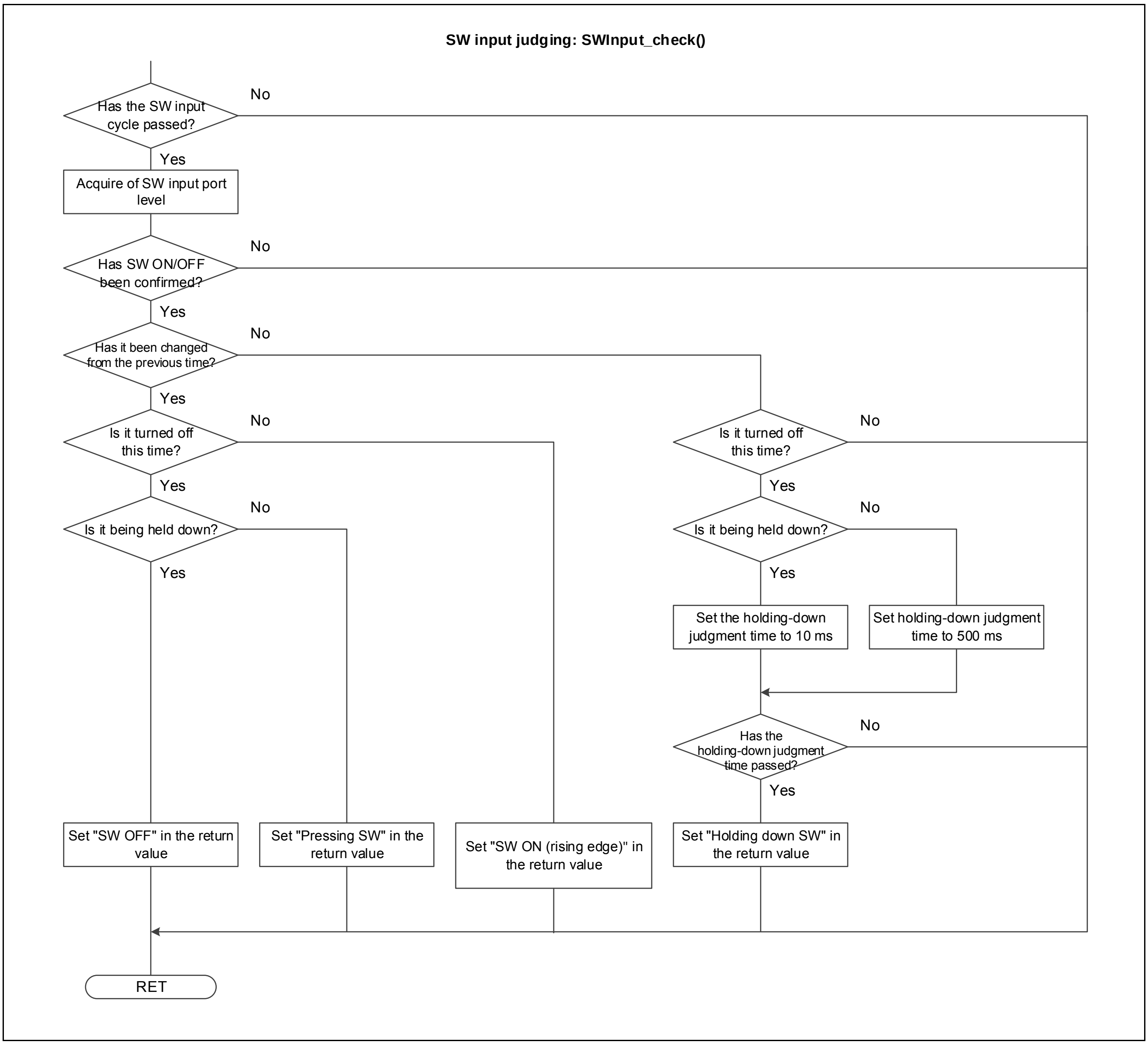


Figure 31 SW Input Judging Flowchart

4.23 Judgment of Dimming State Transition by SW

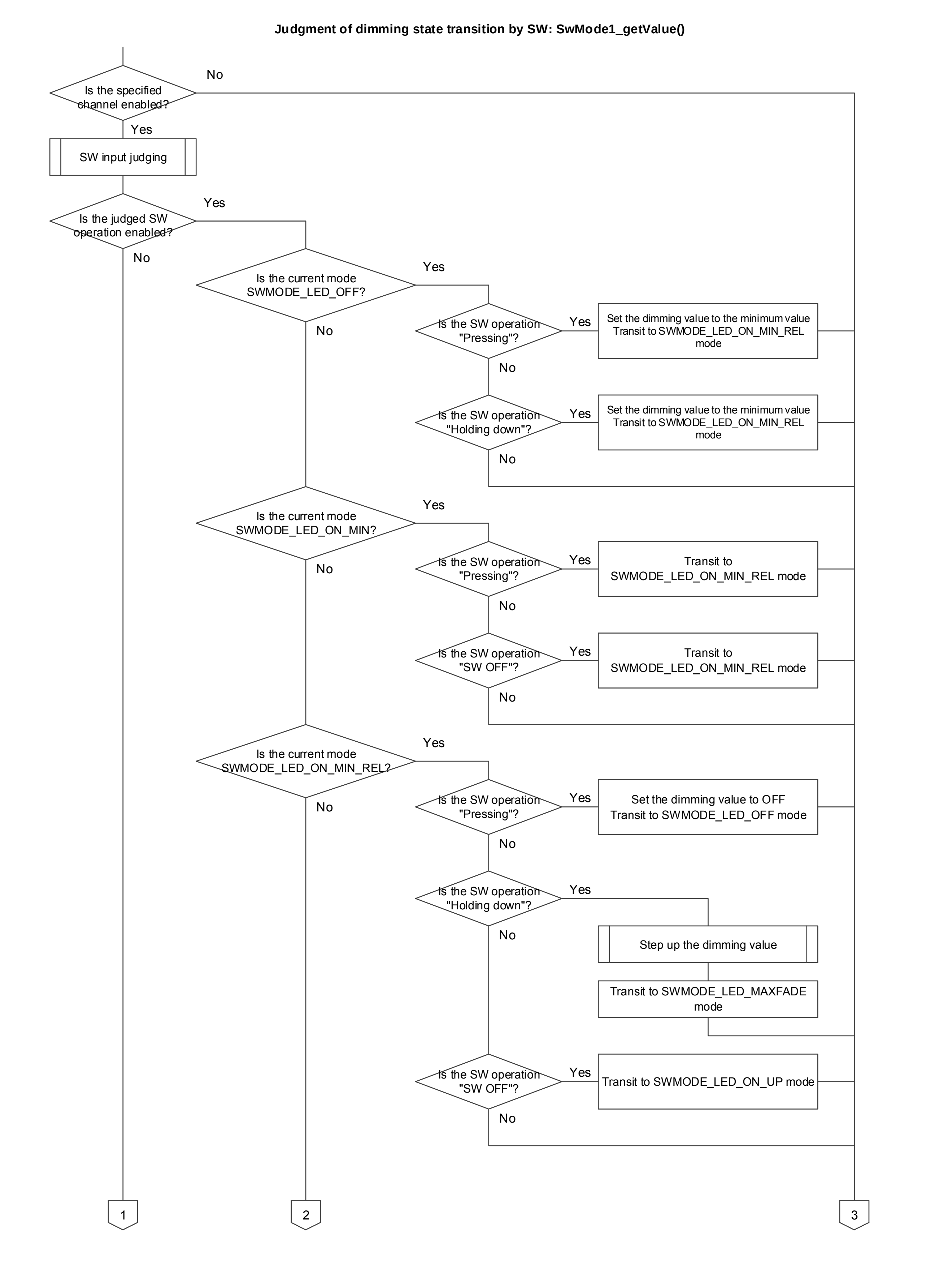


Figure 32 Flowchart of Judgment of Dimming State Transition by SW (1)

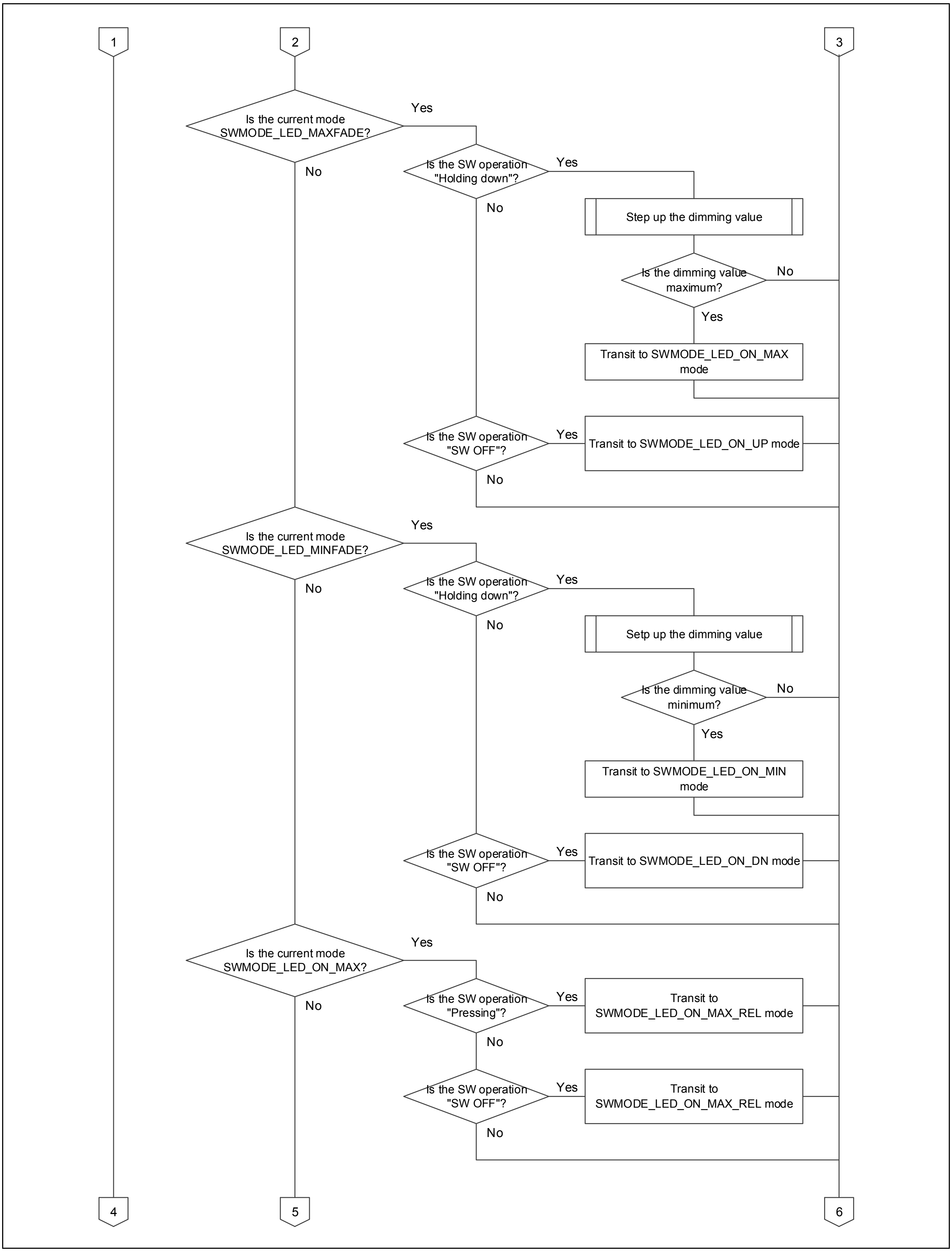


Figure 33 Flowchart of Judgment of Dimming State Transition by SW (2)

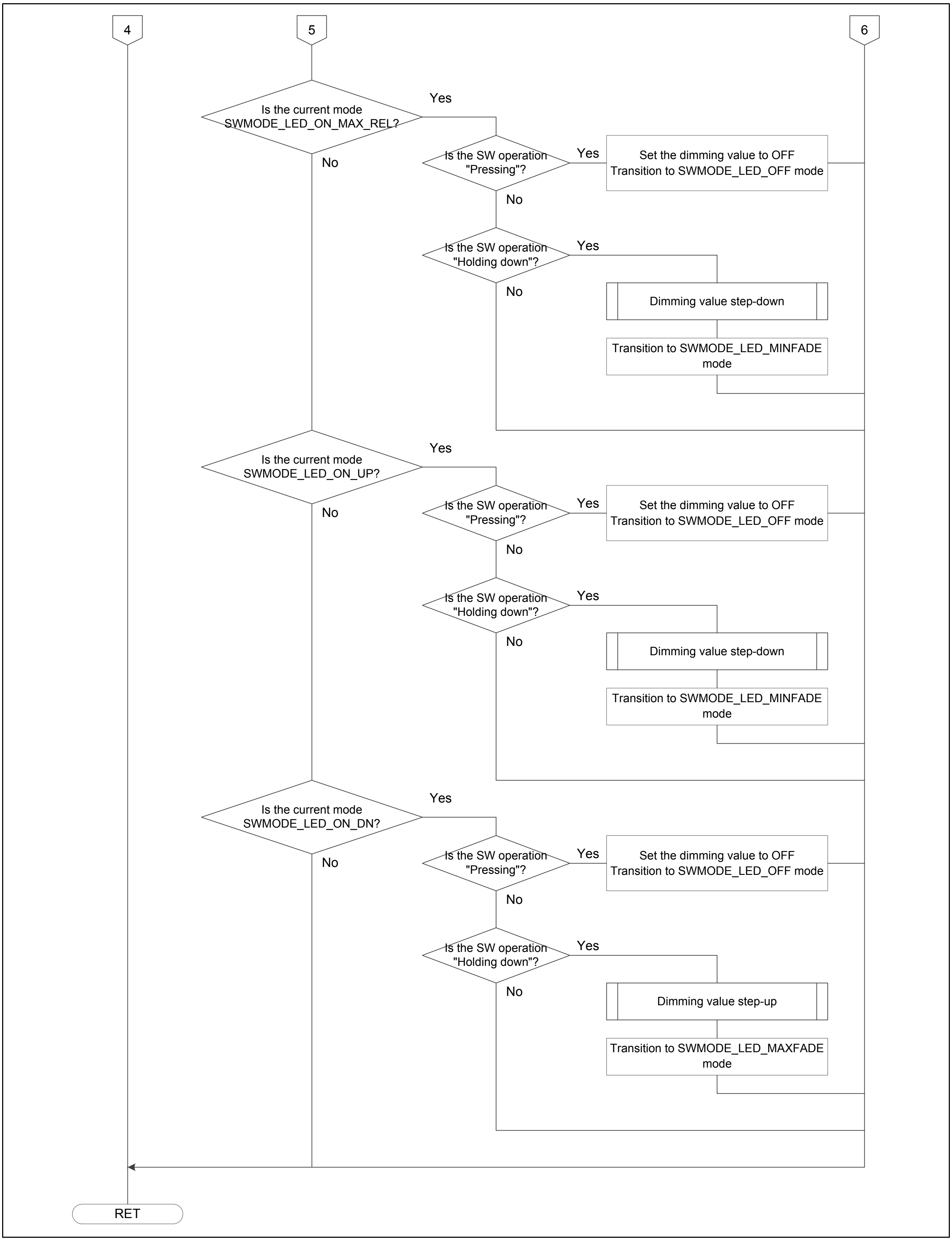


Figure 34 Flowchart of Judgment of Dimming State Transition by SW (3)

5. Characteristics

Frequency characteristics

The frequency characteristics in PFC control and LED control are shown below:

Frequency characteristics R: Control target, G: PI controller, B: Open loop characteristics, C: PWM gain
[Phase margin] 122.5 degrees [Gain cross frequency] 0.001 kHz

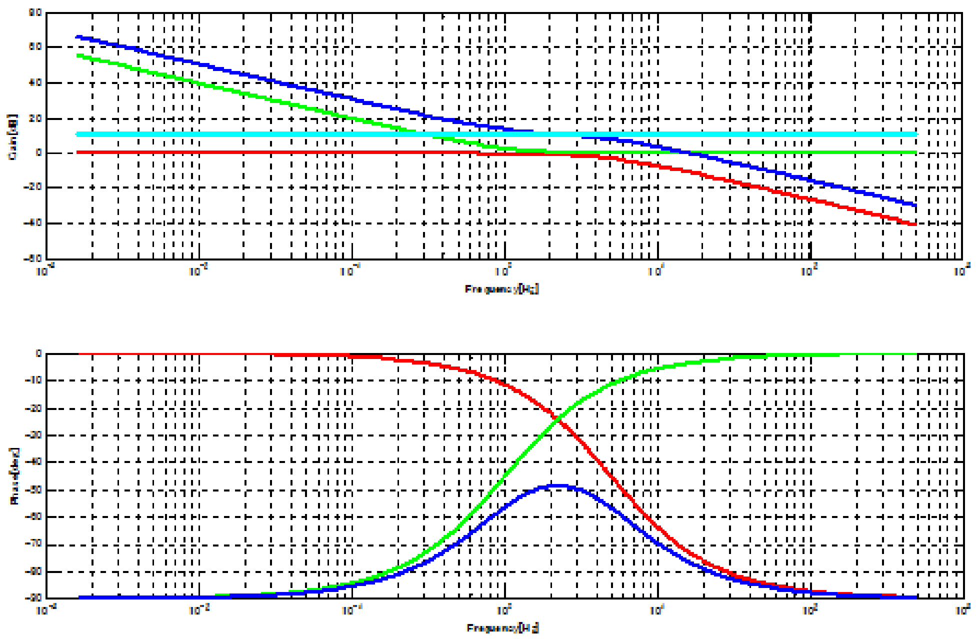


Figure 35 Frequency Characteristics (PFC Control)

Frequency characteristics R: Control target, G: PI controller, B: Open loop characteristics, C: PWM gain
[Phase margin] 94.8 degrees [Gain cross frequency] 0.497kHz

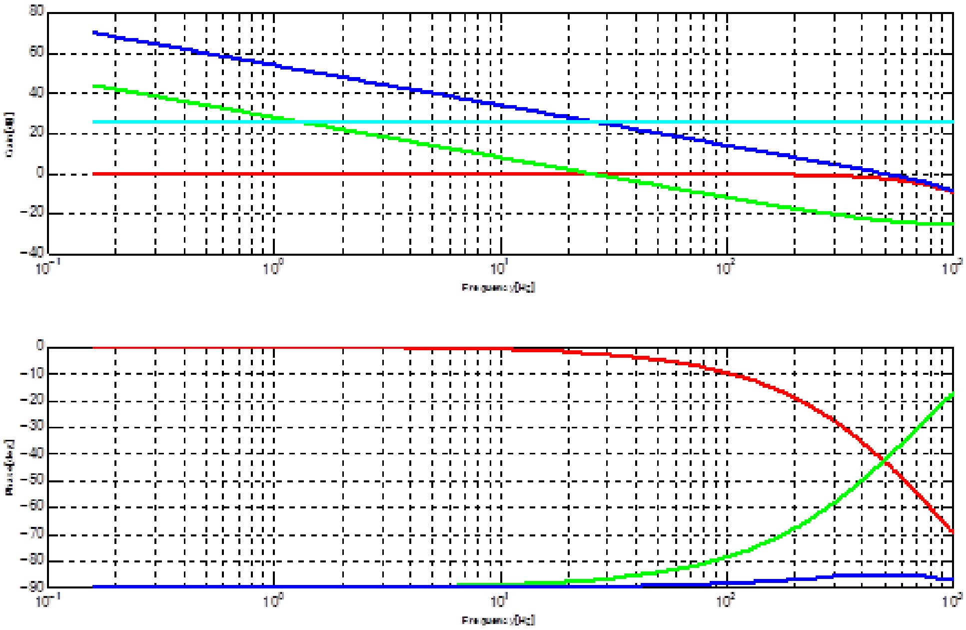


Figure 36 Frequency Characteristics (LED Control)

Harmonic content rate

Figure 37 and Figure 38 show the harmonic content rate when performing input rated 90-W output at 100 and 240 VAC by using this sample program.

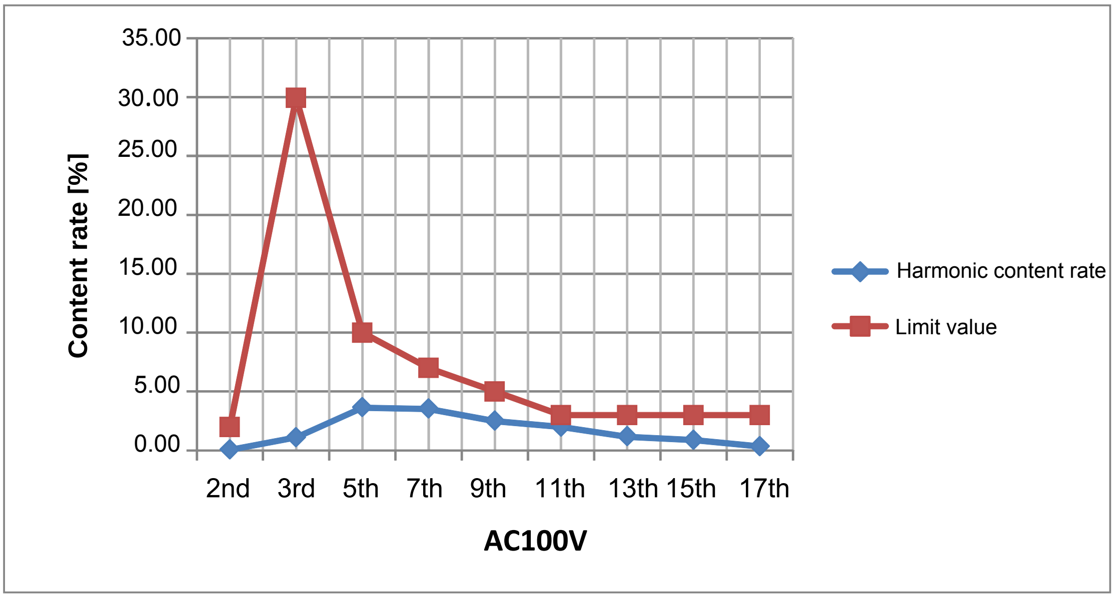


Figure 37 Harmonic Content Rate (100 VAC Input, 90 W)

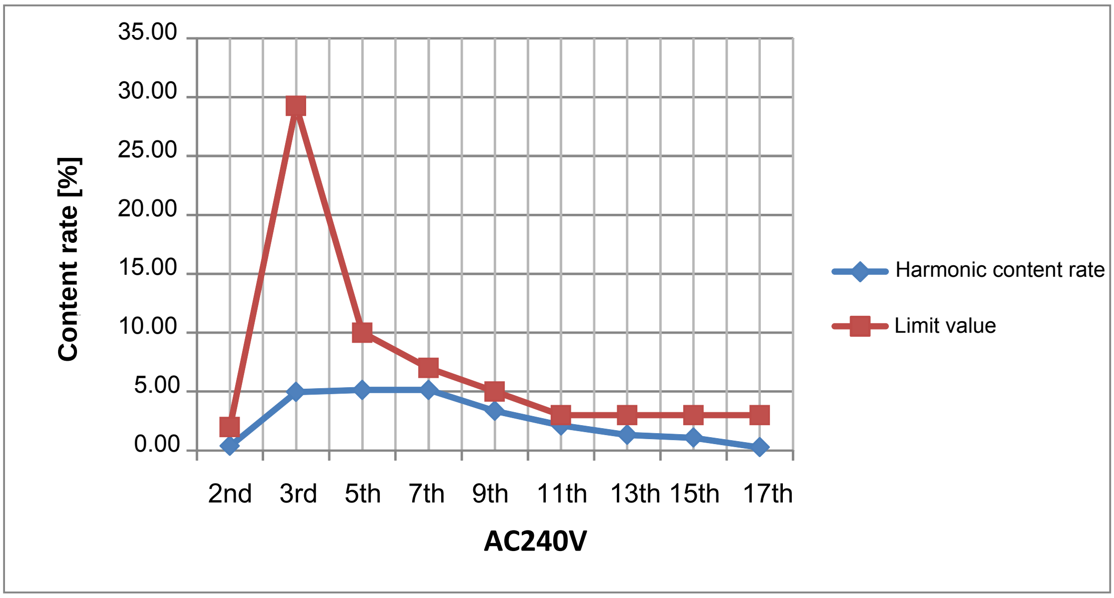


Figure 38 Harmonic Content Rate (240 VAC Input, 90 W)

- Standby energy
  Measurement conditions
  LED control board: Turning SW5-2, 5-4, and 5-6, and SW7-1, 7-3, 7-5, and 7-6 ON
  RL78/I1A: Port processing completed and transition to STOP mode
  Input voltage: 100 VAC

  Standby energy
  0.394W

- Noise terminal
  CISPR15 noise terminal control value is met.

Website and Support

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

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

REVISION HISTORY

Rev.	Date	Description	
		Page	Summary
1.00	Jul 15, 2014	-	First edition issued

General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU products from Renesas. For detailed usage notes on the products covered by this manual, refer to this document and Technical Update.

1. Handling of Unused Pins

   Caution Handle unused pins in accord with the directions given under Handling of Unused Pins in the manual.

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

2. Processing at Power-on

   Caution The state of the product is undefined at the moment when power is supplied.

   The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.

   In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed.

   In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.

3. Prohibition of Access to Reserved Addresses (Reserved Area)

   Caution Access to reserved addresses (reserved area) is prohibited.

   The reserved addresses (reserved area) are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.

4. Clock Signals

   Caution After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has stabilized.

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

5. Differences between Products

   Caution When changing to products of different part numbers, implement a system-evaluation test for each of the products.

   The characteristic value, operation margin, noise capacity, and noise emission amount within the range of electric characteristics of MPU/MCU in the same group but having different part numbers may differ because of the differences in internal ROM and layout pattern. When changing to products of different part numbers, implement a system-evaluation test for each of the products.

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
2801 Scott Boulevard Santa Clara, CA 95050-2549, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
1101 Nicholson Road, Newmarket, Ontario L3Y 9C3, Canada
Tel: +1-905-898-5441, Fax: +1-905-898-3220

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
Room 1709, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100191, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, P. R. China 200333
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

Renesas Electronics Hong Kong Limited
Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022/9044

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre, Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

Renesas Electronics Malaysia Sdn.Bhd.
Unit 906, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics Korea Co., Ltd.
12F., 234 Teheran-ro, Gangnam-Ku, Seoul, 135-920, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 4.0