Application Note

Description of RGB Lighting Control SDK Functions

AN-19120600-E1

Version 1.0.0

2019-12-06

Key Words:

RGB Lighting Control, SDK Function, RGB LED, CT LED

Brief:

This document provides the description for RGB lighting control SDK functions.

TELINK SEMICONDUCTOR

Published by

Telink Semiconductor

Bldg 3, 1500 Zuchongzhi Rd,

Zhangjiang Hi-Tech Park, Shanghai, China

Legal Disclaimer

Information:

For further information on the technology, product and business term, please contact Telink Semiconductor Company (www.telink-semi.com).

For sales or technical support, please send email to the address of:

telinkcnsales@telink-semi.com

telinkcnsupport@telink-semi.com

Revision History

Version 1.0.0 (2019-12-06)

This is the initial release.

Contents

1. Overview

RGB lighting control consists of five LEDs of different colors, red, green, blue, yellow, and white in two groups. Among the five LEDs, red, green and blue LEDs work as RGB LEDs, yellow and white LEDs work as color temperature (CT) LEDs. RGB LEDs and CT LEDs are mutually exclusive, when LEDs of one group are on, LEDs of the other group will be off, or LEDs of two groups are all off.

When the system is powered on, the system will enter pairing state. If no control information packet is received within 6s after power on, the system will exit the pairing state and enter normal state. If pairing packets are received, the system will save the pairing information and exit the pairing state, LEDs will flash three times. If memory clearance packets are received, the system will clear all the control information LEDs saved and exit the pairing state, LEDs will flash five times. If other control information packet is received, and the control information matches saved information, the system will exit the pairing state.

Once the system enters the normal state, the system will check if there are packets received. If yes, the system will check if the remote information in a packet matches the data saved by the lighting control. If matches, the system will check the serial number and control command of the packet, which, if are the same as those of a previous packet, the system will take the packet and the previous one as the same one and discard it; if either the serial number or control command is different from that of a previous packet, the system will take the control command as a new one and execute the control command.

When the system executes commands, if RGB LED(s) are on, CT LEDs will ignore other commands except key on command; if it is a key on command of CT LED(s), RGB LED(s) will be off, CT LED(s) will be on, and the state flag will switch to CT LED state; when in CT LED state, if RGB LED commands are received, the system will switch to RGB LED state, execute the commands and turn CT LED(s) off.

The state of LED(s) changes gradually. The states of LED(s) are updated every 3ms, when the state is updated to the target state, the update will be ceased and the system will clear the update flag.

RF of the lighting control adopts frequency hopping with four frequencies. The frequency updates every 20ms and cycles among the four frequencies.

2. Description of Data Structure

2.1 RF Packet Format

```
typedef struct{
    unsigned int dma_len;
    unsigned char rf_len;
    unsigned char rf_len1;
    unsigned short vid;
    unsigned int pid;
    unsigned char pkt_seq;
    unsigned char key_control;
    unsigned short value[3];
}LED_Package_t;
```

dma_len: RF is in DMA mode, dma_len represents the length of a packet, excluding dma_len.

rf_len: If the communication mode is private 2.4G, the length of data, rf_len = dma_len-1; if it's BLE mode, rf_len as header information can be defined by users.

rf_len1: If the communication mode is BLE mode, the length of data, rf_len1 = dma_len-2; if it's private 2.4G mode, rf_len1 as user data can be defined by users.

vid: ID of product types. IDs can be defined by users according to different products.

pid: Product ID. Every remote has its unique ID.

pkt_seq: Serial number of data packets. Once a command is sent by a remote, the serial number will increase by 1 automatically.

key_control: Control command value. The command values are as follows:

```
typedef enum{
    KEY_NONE_CMD=0, // Null command
    KEY_ON_CMD, // Key on command
    KEY_OFF_CMD, // Key off command
    KEY_LUMINANCE_INC_CMD, // Luminance increase command
    KEY_LUMINANCE_DEC_CMD, // Luminance decrease command
    KEY_CHROME_INC_CMD, // Chroma increase command
    KEY_CHROME_DEC_CMD, // Chroma decrease command
    KEY_SET_CHRO_LUMI_CMD, // Set chroma and luminance command
    KEY_NIGHT_CMD, // Nightlight command
    KEY_PAIRE_CODE_CMD, // Pairing command
    KEY_CLEAR_CODE_CMD, // Memory clearance command
    KEY_SET_RGB_CMD, // Set RGB LED command
    KEY_BREATH_RGB_MODE_CMD, // RGB breath mode command
    LED_LAST_CMD,
}LED_Control_CMD_e;
```

Value: When it's a command of setting luminance of RGB LEDs, the value indicates the luminance of RGB LEDs, the maximum value is 1000. Value[0], Value[1] and Value[2] indicate the luminance of the red, green and blue LEDs respectively. When it's a command of setting CT LEDs, Value[0] indicates the index of luminance, Value[1] indicates the index of chroma.

2.2 Data Format of LED Control Information

```
typedef struct{
    unsigned int remote_id[MAX_PAIRED_REMOTER];
    unsigned char paire_index;
    unsigned char luminance_index; // Index of luminance
    unsigned char chroma_index; // Index of chroma
    unsigned char led_state;
    unsigned short rgb_value[3];
}LED_Control_Info_t;
```

remote_id: Arrays of paired remote IDs. MAX_PAIRED_REMOTER indicates the number of saved remotes.

paire_index: Indexes of IDs of newly paired remotes. When the number of saved remote IDs exceed MAX_PAIRED_REMOTER, paire_index will be reset, saved remote IDs will cover the remote IDs saved before.

luminance_index: Luminance Index of CT LEDs

chroma_index: Chroma Index of CT LEDs

led_state: States of LEDs, values are as follows:

```
typedef enum{
    LED_OFF_STATE=0,
    LED_YL_ON_STATE,
    LED_RGB_ON_STATE,
    LED_RGB_BREATH_STATE,
    LED_LAST_STATE,
}LED_State_e;
    LED_OFF_STATE // LED off, including CT LEDs and RGB LEDs
    LED_YL_ON_STATE // CT LED on
    LED_RGB_ON_STATE // RGB LED on
    LED_RGB_BREATH_STATE // RGB breath mode
rgb_value // Arrays of luminance of RGB LEDs
```

3. Description of Functions

void rf_init_func(void)

Function: RF initialization

Parameter:

Return Value:

Note: Set RF address, buffer address for data received by RF, RF interrupt, etc.

void fm24c02_init_func(unsigned int scl,unsigned int sda)

Function: fm24c02 initialization

Parameter: scl, clock pins of IIC

sda, data pins of IIC

Return Value:

Note: IIC initialization actually.

void set_pwm_init_func(void)

Function: PWM initialization

Parameter:

Return Value:

Note: Set output pins, output frequencies, and the maximum series of PWM.

*Note: Output frequency * PWM series <= System clock*

void led_init_func(void)

Function: LED initialization

Parameter:

Return Value:

Note: Read data saved in EEPROM and recover the state before last power off.

void time_event_process_func(void)

Function: Inquire timestamps

Parameter:

Return Value:

Note: RF receives frequency hopping and switches frequencies every 20ms.

void rf_packget_pro_func(void)

Function: Process RF packets

Parameter:

Return Value:

Note: Inquire if RF receives packets, and check if the command in the received packet is identical to that of a previous packet. If it's a different command, the system will execute it.

void led_pask_process_func(void)

Function: Process LED processing tasks

Parameter:

Return Value:

Note: Process flash, gradients of CT LEDs and RGB LEDs, and breath mode switch of RGB LEDs.

void led_para_init_func(void)

Function: Parameter initialization

Parameter:

Return Value:

Note: Read LED control parameters from EEPROM

void led_on_rgb_func(void)

Function: Turn RGB LEDs on

Parameter:

Return Value:

Note:

void led_rgb_off_func(void)

Function: Turn RGB LEDs off

Parameter:

Return Value:

Note:

void led_rgb_set_func(unsigned short Red_v,unsigned short Green_v,unsigned short Blue_v)

Function: Set the luminance of RGB LEDs

Parameter: Red_v, PWM of the red LED

Green_v, PWM of the green LED

Blue_v, PWM of the blue LED

Return Value:

Note: The target luminance of RGB LEDs which is not reached immediately but gradually.

void led_set_rgb_power_func(unsigned short Red_v,unsigned short Green_v,unsigned short Blue_v)

Function: Set the luminance of RGB LEDs

Parameter: Red_v, PWM of the red LED

Green_v, PWM of the green LED

Blue_v, PWM of the blue LED

Return Value:

Note: The luminance of RGB LEDs which is reached immediately after setting.

void led_updata_lumi_chrome_func(unsigned short Lumi,unsigned char Chroma)

Function: Update the target state of CT LEDs

Parameter: Lumi, luminance

Chroma, chroma

Return Value:

Note:

void led_pwm_control_func(int Lumina, int Chroma)

Function: Set PWM of CT LEDs

Parameter: Lumina, luminance

Chroma, chroma, maximum value 100

Return Value:

Note: PWM of White LED = Lumina * Chroma / 100；

PWM of Yellow LED = Lumina * (100-Chroma) / 100

unsigned short lumina_one_step_updata(unsigned short target,unsigned short cur)

Function: Update luminance

Parameter: target, target luminance

cur, current luminance

Return Value: Updated luminance

Note: The updated luminance is the current luminance.

unsigned short chroma_one_step_updata(unsigned short target,unsigned short cur)

Function: Update chroma

Parameter: target, target chroma

cur, current chroma

Return Value: Updated chroma

Note: The updated chroma is the current chroma.

void led_on_func(void)

Function: CT LED on

Parameter:

Return Value:

Note:

void led_off_func(void)

Function: CT LED off

Parameter:

Return Value:

Note:

void led_updata_luminance_func(unsigned char Type)

Function: Update luminance of CT LEDs

Parameter: Type, type of luminance, 1 represents 1 level up, 0 represents 1 level down.

Return Value:

Note: When the luminance reaches the maximum value, level up will lead to no change; when the luminance reaches the minimum value, level down will lead to no change.

void led_updata_chroma_func(unsigned char Type)

Function: Update chroma of CT LEDs

Parameter: Type, type of chroma, 1 represents 1 level up, 0 represents 1 level down.

Return Value:

Note: When the chroma reaches the maximum value, level up will lead to no change; when the chroma reaches the minimum value, level down will lead to no change.

void led_set_lumi_chrome_func(unsigned short Lumi,unsigned short Chroma)

Function: Set luminance and chroma

Parameter: Lumi, luminance

Chroma, chroma

Return Value:

Note: The value set by this function is a specified value, not a list value.

void led_event_proc_func(unsigned char Cmd)

Function: Execute the command of receiving packets

Parameter: Cmd, command

Return Value:

Note:

void write_id_direct(unsigned char Position,unsigned int Id)

Function: Save remote IDs

Parameter: Position, position of ID indexes

Id, remote ID

Return Value:

Note:

void write_data_direct_func(unsigned char Addr,unsigned char Data)

Function: Save data to EEPROM

Parameter: Addr, EEPROM address

Data, saved data

Return Value:

Note:

unsigned char save_remote_ID_func(unsigned int Id)

Function: Save remote IDs

Parameter: Id, remote ID

Return Value:

Note: The function is used in pairing.

void clear_remote_ID_func(void)

Function: Clear all saved remote IDs

Parameter:

Return Value:

Note: This function is used in memory clearance.

unsigned char paired_ID_match(unsigned int Id)

Function: Check if the remote ID of a packet matches saved information

Parameter: Id, remote ID

Return Value: If ID matches return 1, if not return 0.

Note: Check the remote ID of each received packet, execute related commands only if the ID matches.

void save_led_control_info_func(void)

Function: Save LED control information

Parameter:

Return Value:

Note: This function saves all remote information including remote IDs.

void save_led_state_info_func(void)

Function: Save LED state information

Parameter:

Return Value:

Note: This function does not save remote IDs.

void rf_change_channel_func(void)

Function: Receive frequency hopping

Parameter:

Return Value:

Note: Frequency hops every 20ms and cycles between 2401MHz, 2424MHz, 2451MHz, and 2476MHz.

void PWM_MaxFqeSet(PWM_TypeDef pwmNumber,unsigned short MaxValue,unsigned short Fqe)

Function: Set the maximum value and frequencies of PWM

Parameter: pwmNumber, PWM Index

MaxValue, maximum value of PWM

Fqe, PWM frequency

Return Value:

Note: The default maximum value of PWM is 0xffff.

void PWM_DutyValueSet(PWM_TypeDef pwmNumber,unsigned short value)

Function: Set duty cycle of PWM

Parameter: pwmNumber, PWM Index

value, PWM value

Return Value:

Note: Duty cycle of PWM = value / maxvalue

void user_PWMInit(PWM_TypeDef pwmNumber,unsigned short MaxValue,unsigned short Fqe)

Function: PWM initialization

Parameter: pwmNumber, PWM index

MaxValue, maximum value of PWM

Fqe, PWM frequency

Return Value:

Note: