# Application Note
# Description of 2.4G Lighting Control SDK Functions

**AN-19121100-E2**

Version 1.1.0

2019-12-11

**Key Words:**

2.4G Lighting Control, SDK Function, Luminance, Chroma

**Brief:**

This document provides the specification for 2.4G Lighting Control SDK Functions.

TELINK SEMICONDUCTOR

**Published by**

**Telink Semiconductor**

**Bldg 3, 1500 Zuchongzhi Rd,**

**Zhangjiang Hi-Tech Park, Shanghai, China**

## Legal Disclaimer

This document is provided as-is. Telink Semiconductor reserves the right to make improvements without further notice to this document or any products herein. This document may contain technical inaccuracies or typographical errors. Telink Semiconductor disclaims any and all liability for any errors, inaccuracies or incompleteness contained herein.

## 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 1.1.0 (2019-12-11)

| Section | Change Description |
|---|---|
| 3 | Change function void fm24c02_init (void) to function void fm24c02_init_func (unsigned int scl, unsigned int sda). |

## Version 1.0.0 (2019-11-13)

This is the Initial release.

# Contents

Revision History ........ 2

1. Overview ........ 4

2. Description of Data Structure ........ 5

  2.1 RF Packet Format ........ 5

  2.2 Data Format of LED Control Information ........ 5

3. Definition of Functions ........ 7

  void set_pwm_init_func (void) ........ 7

  void check_sys_state_func (void) ........ 7

  void rf_init_func (void) ........ 7

  void led_init_func (void) ........ 7

  void fm24c02_init_func (unsigned int scl, unsigned int sda) ........ 7

  void rf_packget_pro_func (void) ........ 8

  void led_task_process_func (void) ........ 8

  void time_event_process_func (void) ........ 8

  void send_relay_pkt (void) ........ 8

  void led_event_proc_func (unsigned char Cmd) ........ 9

  void led_set_lumi_chrome_func (unsigned short Lumi, unsigned short Chroma) ........ 9

  unsigned short lumina_one_step_updata (unsigned short target, unsigned short cur) ........ 9

  unsigned short chroma_one_step_updata (unsigned short target, unsigned short cur) ........ 9

  void led_para_save_func (void) ........ 9

  void led_power_control_func (int Lumina, int Chroma) ........ 10

  void led_pwm_control_func (int Lumina, int Chroma) ........ 10

  void rf_change_channel_func (void) ........ 10

  unsigned char check_pkt_info (rf_packet_led_remote_t *pkt) ........ 10

  unsigned char paired_ID_match (unsigned int pid, unsigned char grp) ........ 10

  void PWM_MaxFqeSet (PWM_TypeDef pwmNumber, unsigned short MaxValue, unsigned short Fqe) ........ 11

  void PWM_DutyValueSet (PWM_TypeDef pwmNumber, unsigned short value) ........ 11

  void user_PWMInit (PWM_TypeDef pwmNumber, unsigned short MaxValue, unsigned short Fqe) ........ 11

# 1. Overview

The luminance and chroma of the lighting are shown according to the remote's control information. The lighting supports eeprom, LEDs are initialized after power on, based on the info stored in eeprom. During LEDs' initialization, system will check the state flag, If it is equal to 0xff, the system will set the state according to indexes. There are four modes for lighting, all white, all yellow, half white half yellow, and nightlight. If the state flag is not 0xff, the system will show according to the mode before power off.

After initialization, the system enters pairing state. In the pairing state, if no data packet is received within 6s after power-on, the system will exit the pairing state and enter the normal state. If a turn on command is received, the system will enter memory clearance state. In the memory clearance state, if five on commands are continuously received at an interval of 500ms, the system will clear information stored before, the LEDs will flash five times. If no key on command is received at an interval of 500ms after entering the memory clearance state, the system will take it as pairing and save the information of the remote. LEDs will flash three times. After the pairing is completed, the system will enter the normal state.

After the system enters the normal state, if no packets is received, the state will keep unchanged. If a data packet is received, the system will check VID, PID, and GRP. If the check is passed, the system will process related commands, or it will discard the data packet.

# 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     control_key;
        unsigned char     rf_seq_no;
        unsigned short   button_keep_counter;
        unsigned short   control_key_value[3];
        unsigned char     reserved;
}rf_packet_led_remote_t;
```

dma_len: RF is in DMA mode, dma_len represents the length of the packet, does not include 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 the 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 is the user data and 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.

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

control_key: control command value. The command values are as follows:

```
typedef enum{
        LED_NONE_CMD=0, //release the button, send null value
        LED_ON_CMD, //key on command
        LED_OFF_CMD, //key off command
        LED_LUMINANCE_INC_CMD, //luminance increase command
        LED_LUMINANCE_DEC_CMD, //luminance decrease command
        LED_CHROME_INC_CMD, //chroma increase command
        LED_CHROME_DEC_CMD, //chroma decrease command
        LED_SET_CHRO_LUMI_CMD, //set chroma/luminance command
        LED_NIGHT_CMD, //nightlight command
        LED_LAST_CMD,
}Led_control_cmd;
```

button_keep_counter, reserved: Reserved

control_key_value: When the value of key_control is *LED_SET_CHRO_LUMI_CMD*, control_key_value[0] is the luminance value, and control_key_value[1] the chroma value.

## 2.2 Data Format of LED Control Information

```
typedef struct{
        Pairing_info_t pared_remote[MAX_PAIRED_REMOTER];
        unsigned char   paire_index;
        unsigned char   luminance_index;
        unsigned char   chroma_index;
        unsigned char   led_on;
        unsigned char    power_on_recover;
        unsigned char    seg_index;
}LED_control_info_t;
```

pared_remote: A structure array. The size of the array, MAX_PAIRED_REMOTER represents the maximum number of information of paired remotes saved by a lighting. The structure contains IDs of remotes and groups of remotes.

paire_index: This parameter is used to save the array indexes of remotes. When the next pairing signal comes, it saves the information of remotes to the index array.

luminance_index: The index of LEDs’ luminance. The luminance value is the array value of led_luminance_value which corresponds to its index.

chroma_index: The index of LEDs’ chroma. The chroma value is the array value of led_chroma_value which corresponds to its index.

led_on: A LED’ s state. 1 represents the LED is on, 0 represents the LED is off.

power_on_recover: This parameter shows if the power on switches between the four modes: all white, all yellow, half white half yellow and nightlight. 0xff represents switch enabled, 0 represents switch disabled.

seg_index: Indexes of switches between the four modes: all white, all yellow, half white half yellow and nightlight.

# 3. Definition of Functions

### void set_pwm_init_func (void)

**Function:** Initialization of hardware PWM

**Parameter:**

**Return Value:**

**Note:** Please set PWM series to 1000, frequency to 16K for Hardware PWM initialization.

*Note: PWM series * Frequency <= CLOCK_SYS_CLOCK_HZ*

### void check_sys_state_func (void)

**Function:** Check system state

**Parameter:**

**Return Value:**

**Note:** After the system is powered on and enters pairing state, if no data is received within 6s, the system will enter the normal state. After the system enters the memory clearance state, if no data packet is received within 500ms, the system will take it as pairing, save the information and exit the pairing state. The LED will flash three times.

### void rf_init_func (void)

**Function:** RF initialization

**Parameter:**

**Return Value:**

**Note:** Please set buffer address and size for data received by RF, enabled RF receive interrupt.

### void led_init_func (void)

**Function:** Initialization of power on of LEDs

**Parameter:**

**Return Value:**

**Note:** Initialization of LEDs. If the power on switch sign led_control.power_on_recover is 0xff, the system will set one of the four modes: all white, all yellow, half white half yellow and nightlight according to the index led_control.seg_index in saving state. If the power on switch sign led_control.power_on_recover is not 0xff, the system will show the state of LEDs before power off.

### 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:** Actually IIC initialization.

### void rf_packget_pro_func (void)

**Function:** Process packets received by RF

**Parameter:**

**Return Value:**

**Note:** g_packget_new is the sign of receiving packets. 1 represents the packet is received.

When the system is in pairing state (*PAIRRING_STATE*), if it receives key on command, the system will enter memory clearance state; if it receives non-key on command, the system will check with saved information, and will exit pairing state and enter the normal state (*NORMAL_STATE*) after the check passed. If the data do not match the saved information, the system will not process them.

In memory clearance state (*CLEARCODE_STATE)*, if the system receives the key on command more than four times (five times if count the key on command in pairing state), it will clear LED's information that saved before and exit the code clearance state, the LED will flash five times. In code clearance state, if other commands which match the saved information are received or the key on commands received are less than five times, the system will exit the code clearance state and not do any processing.

In normal state (*NORMAL_STATE*), if the information of data packets match the saved information, the system will execute the commands or it will not do any processing.

### void led_task_process_func (void)

**Function:** Process LEDs' flash and gradient

**Parameter:**

**Return Value:**

**Note:** When led_flash_cnt is not 0, it means LED flashes and switches a state every 500ms. When led_state_change_flag is not 0, it means there are gradient tasks and the system will perform gradient. When led_lumina_cur equals to led_lumina_target, led_chroma_cur equals to led_chroma_target, it means gradient is finished, the system will reset led_state_change_flag.

### void time_event_process_func (void)

**Function:** Process time events

**Parameter:**

**Return Value:**

**Note:** The system switches receiving frequency every 50ms.

### void send_relay_pkt (void)

**Function:** Relay packets

**Parameter:**

**Return Value:**

**Note:** Upon receipt of the packets, the system relays the packets and starts receiving.

**void led_event_proc_func (unsigned char Cmd)**

**Function:** Execute remote commands

**Parameter:** Cmd, command value of Cmd

**Return Value:**

**Note:** If the remote information matches the information saved by the LED, the system will execute commands, such as key on, key off, chroma increase or decrease, luminance increase or decrease, nightlight mode. In key off state, the system will not execute chroma increase or decrease, and luminance increase or decrease.

**void led_set_lumi_chrome_func (unsigned short Lumi, unsigned short Chroma)**

**Function:** Set specified chroma value and luminance value

**Parameter:** Lumi, the setting value of luminance

Chroma, the setting value of chroma

**Return Value:**

**Note:**

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

**Function:** Update luminance value

**Parameter:** target, target value of luminance

cur, current value of luminance

**Return Value:** Updated luminance value

**Note:** Updated luminance value is the current value of luminance

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

**Function:** Update chroma value

**Parameter:** target, target value of chroma

cur, current value of chroma

**Return Value:** Updated chroma value

**Note:** Updated chroma value is the current value of chroma

**void led_para_save_func (void)**

**Function:** Save parameters of LED control

**Parameter:**

**Return Value:**

**Note:** Save parameter values of LED control to eeprom

### void led_power_control_func (int Lumina, int Chroma)

**Function:** Set PWM values for lighting

**Parameter:** Lumina, luminance value

Chroma, chroma value, maximum 100

**Return Value:**

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

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

### void led_pwm_control_func (int Lumina, int Chroma)

**Function:** Set LED' luminance value and chroma value

**Parameter:** Lumina, luminance value

Chroma, chroma value

**Return Value:**

**Note:** Set PWM value, set duty cycle of two PWM values according to luminance value and chroma value

*Note: The difference between this function and void led_power_control_func (int Lumina, int Chroma) is that the latter includes calculation of indexes of luminance value and chroma value.*

### void rf_change_channel_func (void)

**Function:** Receive frequency hopping

**Parameter:**

**Return Value:**

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

### unsigned char check_pkt_info (rf_packet_led_remote_t *pkt)

**Function:** Check information of packets

**Parameter:** pkt, receive address pointer of remotes

**Return Value:** If the remote ID, key and serial number of current packet are the same as those of a previous packet, the system will take it as a duplicate packet and return 0; if any one of the remote ID, key and serial number is different from that of a previous packet, the system will take it as a different packet and return 1.

**Note:** Only after the check is passed, the system will process the packet.

### unsigned char paired_ID_match (unsigned int pid, unsigned char grp)

**Function:** Check information of received packets

**Parameter:** pid, ID of remotes

grp, group of remotes

**Return Value:** When ID and group of the remote both matches the saved information, the function return 1; if not match, return 0.

**Note:**

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

**Function:** Set maximum value and frequency of PWM

**Parameter:** pwmNumber, PWM index

MaxValue, maximum value of PWM

Fqe, PWM frequency

**Return Value:**

**Note:** The default maximum value 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:**