

# Telink

## Telink B80 24G Lighting

## SDK Development Handbook

AN-21113000-E1

---

Ver1.0.0

2021.11.30

### Keyword

B80, remote, control, 24GHz

### Brief

This document is the development guide for Telink B80 24G lighting SDK.

**Published by**  
**Telink Semiconductor**

**Bldg 3, 1500 Zuchongzhi Rd,**  
**Zhangjiang Hi-Tech Park, Shanghai, China**

**© Telink Semiconductor**  
**All Rights Reserved**

### **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.

Copyright © 2021 Telink Semiconductor (Shanghai) Co., Ltd.

### **Information**

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

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

[telinksales@telink-semi.com](mailto:telinksales@telink-semi.com)

[telinksupport@telink-semi.com](mailto:telinksupport@telink-semi.com)

## Revision History

---

Version	Change description
V1.0.0	Initial release

---

Telink Semiconductor

# Contents

Revision History	3
1 SDK overview	5
1.1 SDK file structure	5
1.1.1 chip	5
1.1.2 common	6
1.1.3 demo	6
1.1.4 project	7
1.1.5 Other folders	7
1.2 Demo project	8
1.3 SDK sending and receiving packets and handling of pairing	8
1.3.1 SDK pairing process introduction	8
1.3.2 Introduction to the flow of SDK sending and receiving packets	9
1.4 remote matrix key scan	12
1.5 EEPROM	13
1.6 LED	13
1.7 SDK debugging instructions	14
1.7.1 Tdebug	14
1.7.2 Serial print debugging instructions	14
2 SDK code description	16
2.1 REMOTE	16
2.1.1 Introduction of initialization function interface for REMOTE	16
2.1.2 Loop interface introduction for REMOTE	16
2.2 LIGHT	16
2.2.1 Introduction of initialization function interface for LIGHT	16
2.2.2 Loop interface introduction for LIGHT	17
2.3 LIGHT_BEACON	18
2.4 REMOTE_BEACON	18
2.5 LIGHT_RGB	18
2.6 REMOTE_RGB	18
3 SDK configuration file description	19
3.1 Configuration of RF parameters and frequencies	19
3.2 Configuration of IO ports	19
3.3 Modify main frequency	19
3.4 Configure the product VID	19
3.5 Configure the OTP address of the REMOTE PID	19
3.6 Configure the number of times the command sending	20
4 Operation introduction	21
4.1 Key functions of REMOTE& LIGHT and REMOTE_BEACON& LIGHT_BEACON	21
4.2 Key functions of REMOTE_RGB&LIGHT_RGB	22

## 1 SDK overview

This SDK provides users with remote control lights based on B80 series chips, including remote-light, remote-RGB light, and remote-beacon. Users can develop application layer code and adapt pcb boards based on these demo codes.

### 1.1 SDK file structure

The current SDK file structure is shown as below.

chip	2021/11/26 16:58	文件夹
common	2021/11/26 16:58	文件夹
demo	2021/11/26 16:58	文件夹
doc	2021/11/26 16:58	文件夹
project	2021/11/26 16:58	文件夹
tools	2021/11/26 16:58	文件夹

**Figure 1:** "SDK file structure"

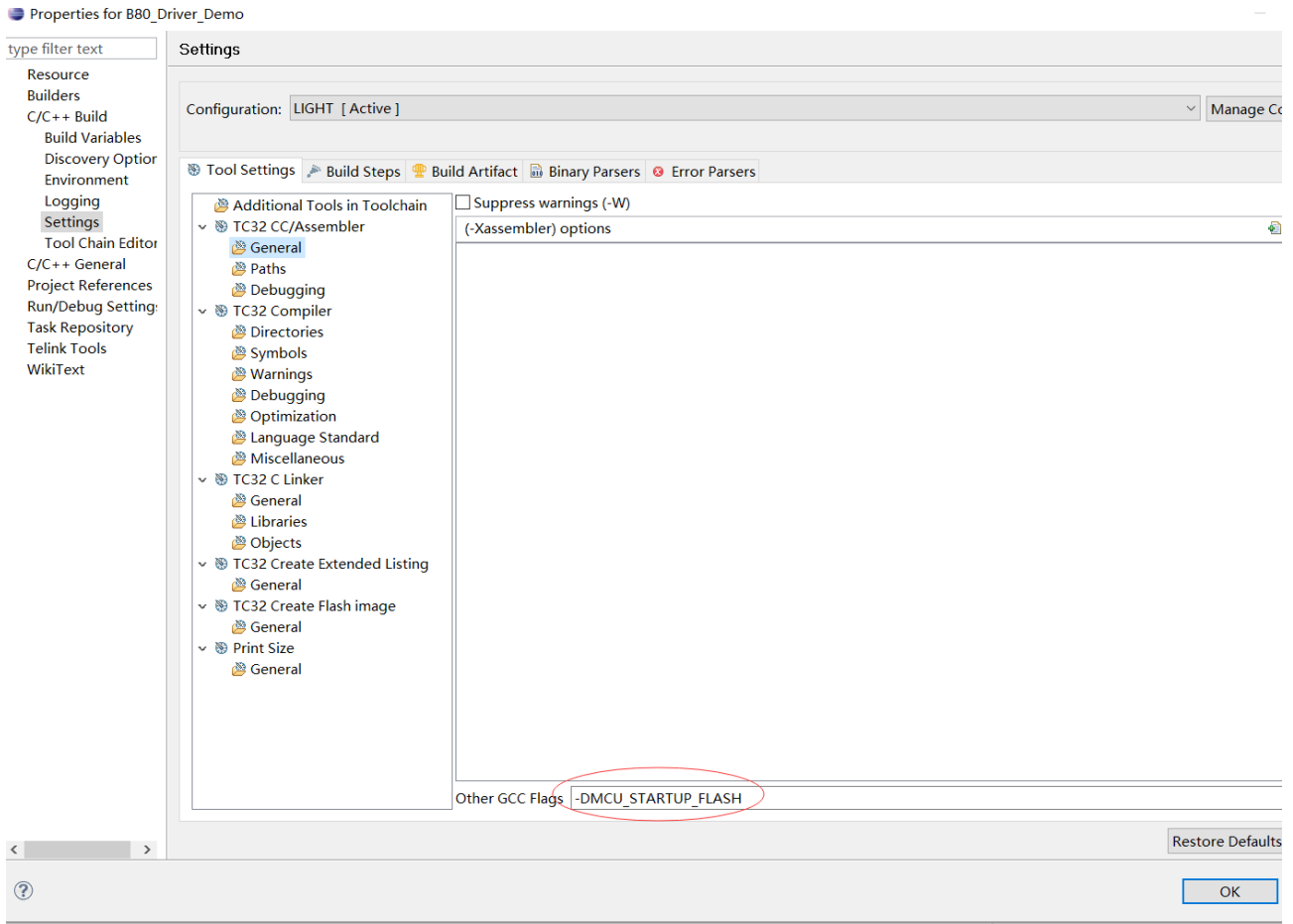
#### 1.1.1 chip

The chip mainly contains the driver and boot code of the B80 chip, where the boot code contains three modes, the boot folder contains cstartup\_flash.S, cstartup\_otp.S, cstartup\_sram.S, which correspond to the macros MCU\_STARTUP\_FLASH, MCU\_STARTUP\_OTP, MCU\_STARTUP\_SRAM. You can control different modes by configuring the corresponding macros in the IDE.

cstartup\_flash.S: flash mode, the burning program runs in flash, you can choose this mode when the development board equips with external flash.

cstartup\_otp.S: otp mode, the burning program runs in otp, as B80 eventually uses built-in otp, the otp option will be used for the final mass production products.

cstartup\_sram.S: sram mode, you can use this mode for debugging when there is no external flash, and you need to do simple debugging to avoid the otp can't be changed after it is written.



**Figure 2:** "Compile mode options"




Inside the Driver folder corresponds to the driver file for B80 and the library for rf and pm.

### 1.1.2 common

The div\_mod.s defines some generic interfaces, such as assembly interfaces for multiplying large numbers. The rest of the functions are generic type definitions and interfaces like memcpy, memcmp and so on.

### 1.1.3 demo

Inside the demo folder is the defined code for the user layer.

 common	2021/11/26 16:58	文件夹
 light	2021/11/29 17:18	文件夹
 light_beacon	2021/11/26 16:58	文件夹
 light_rgb	2021/11/29 13:42	文件夹
 remote	2021/11/26 16:58	文件夹
 remote_beacon	2021/11/26 16:58	文件夹
 remote_rgb	2021/11/26 16:58	文件夹

**Figure 3:** "demo folder structure"

The common folder contains the folder options that are common to all compilation options. The rest of the folders correspond to different compilation branches.

#### 1.1.4 project

The project folder is the folder for this project, where the .cproject and .project file are the corresponding project files, .boot and .link are the link files, which are not recommended to be modified by the customer, the proj\_lib is the path where the library is placed, and the folders in it are the files generated by the compilation, for example, LIGHT.bin and LIGHT.lst will be generated inside the LIGHT folder.

名称	修改日期	类型	大小
 .settings	2021/11/26 17:08	文件夹	
 LIGHT	2021/11/29 17:14	文件夹	
 LIGHT_BEACON	2021/11/26 17:11	文件夹	
 LIGHT_RGB	2021/11/26 17:09	文件夹	
 It_8208	2021/11/26 17:09	文件夹	
 proj_lib	2021/11/26 16:58	文件夹	
 REMOTE	2021/11/26 17:10	文件夹	
 REMOTE_BEACON	2021/11/26 17:10	文件夹	
 REMOTE_RGB	2021/11/26 17:12	文件夹	
 RF_DEMO	2021/11/26 17:09	文件夹	
 .cproject	2021/11/29 17:19	CPROJECT 文件	785 KB
 .project	2021/11/26 17:05	PROJECT 文件	5 KB
 boot.link	2021/11/26 16:58	LINK 文件	6 KB

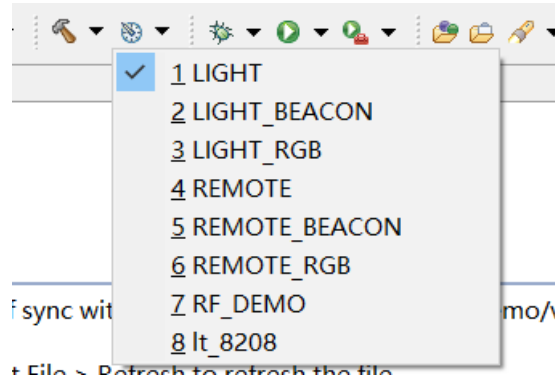
**Figure 4:** "project folder structure"

#### 1.1.5 Other folders

The doc folder corresponds to the release version information, and the tool folder corresponds to some packing tools.

## 1.2 Demo project

The B80 remote light SDK is a compilation branch for customer development, which allows users to observe the visual effects through demo operations and also to modify on the demo code to complete the development of their own applications.



**Figure 5:** "List of compilation options"

## 1.3 SDK sending and receiving packets and handling of pairing

The following description of the pairing and sending/receiving packet process is currently based on the remote-light mode.

### 1.3.1 SDK pairing process introduction

Parameters of Tx transmission.

Transmit channels: 2401MHz, 2424MHz, 2451MHz, 2476MHz, 4 signals.

Transmit accesscode: {0x71,0x76,0x51,0x39,0x95}

Transmit power: 11.46dbm

SDK pairing principle:

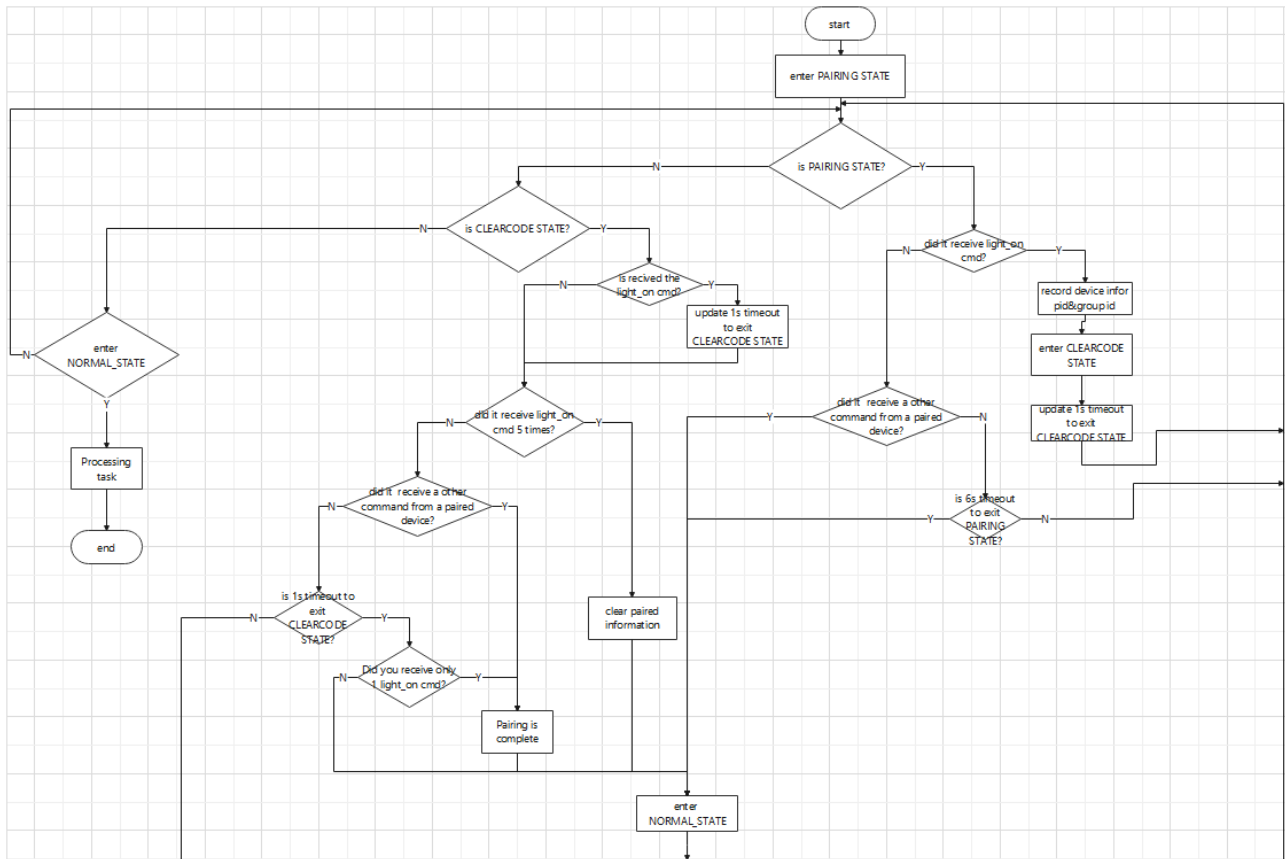
Within 5s of power up, short press the on light key of a group once, the pairing will be completed. The LED flashes 3 times after pairing success.

Within 5s of power up, short press the on light key of a group for consecutive 5 times (interval less than 500ms), it will finish clearing the pairing information. The LED flashes 5 times after clear pairing information successfully.

Within 5s of power up, if a non-light-on key command is received, it will exit pairing mode and enter the normal state.

Flowchart of SDK pairing is as below.





**Figure 6:** "light&remote pairing process"

### 1.3.2 Introduction to the flow of SDK sending and receiving packets

Take the remote project as an example. The description of the function for sending packet is as below.

```
typedef struct{
    unsigned int dma_len;           // 0~3 DMA length
    unsigned char rf_len;           // 4 rf data length = 0x10
    unsigned char rf_len1;
    unsigned short vid;             // 5~6 vendor ID
    unsigned int pid;               // 7~10 product ID

    unsigned char control_key;       // 11 function control key
    unsigned char rf_seq_no;         // 12 rf sequence total number, save this value in 3.3v analog register.

    unsigned short button_keep_counter; // 13~14 sequence number in one certain channel.
    unsigned short control_key_value[3]; // 15, 16, 17, 18
    unsigned char ttl;

}rf_packet_led_remote_t; //rf data packet from remoter end.
```

**Figure 7:** "Command packet data structure"

```
void package_data_init_func(void)
```

Sending packet data initialization. Configure VID information, reassign a specific otp address or PID of the remote, and get the saved information from the analog register.

```
void package_data_set_newcmd(unsigned char key_value,unsigned char* para)
```

Configure the new command. Update the rf\_seq\_no, control\_code and control\_code parameters.

```
void package_data_send_func(void)
```

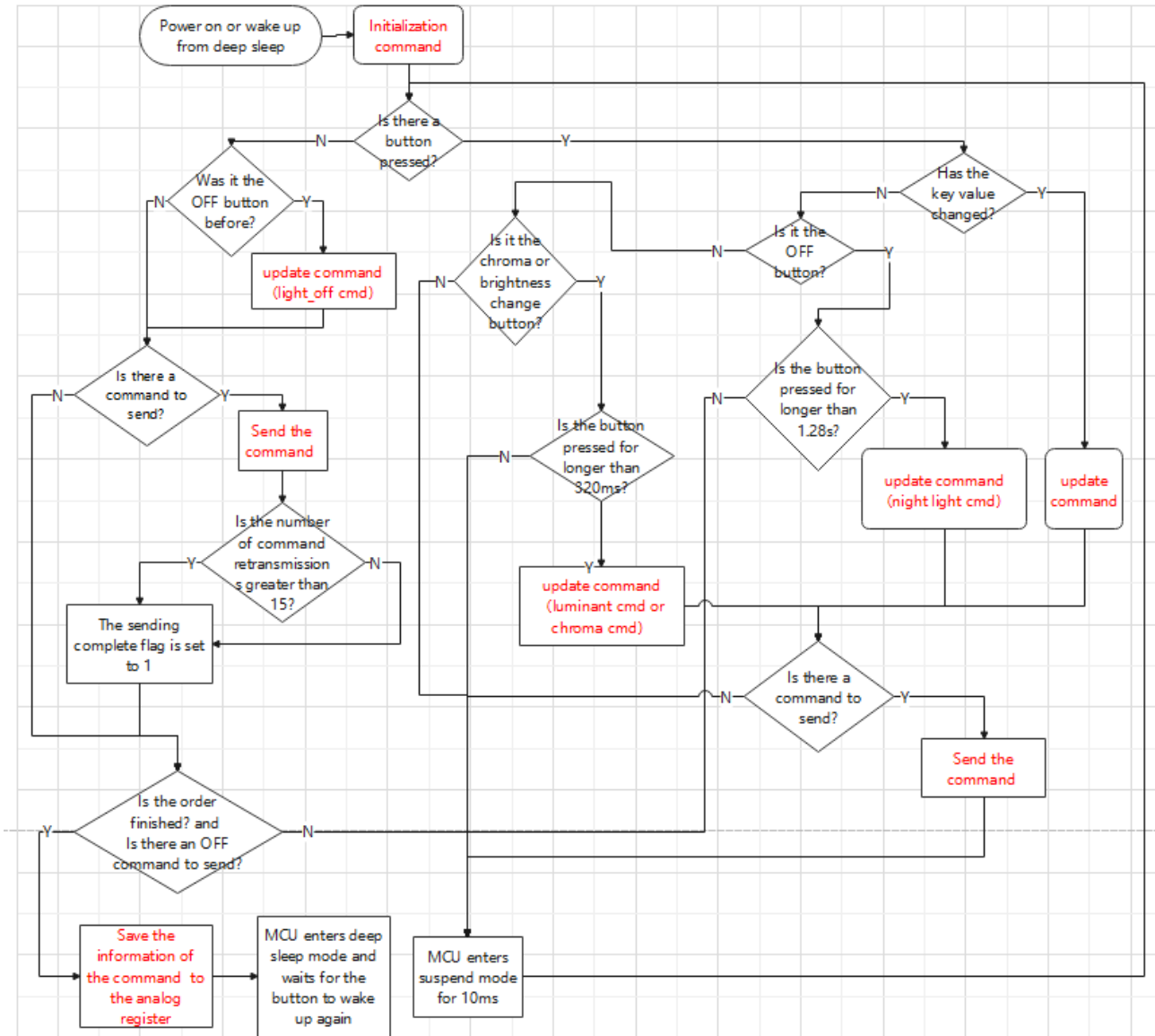
Call the rf module to send the command out. (remote send TTL=5) (The command is normally repeated 15 times unless it is updated to a new command halfway through.)

```
void package_data_store_func(void)
```

Store the group&rf\_seq\_no information of the command code to the analog register. Prevents loss of information at a wake-up after deep sleep.

The logic flow chart for the sending packet is as below.

Telink Semiconductor

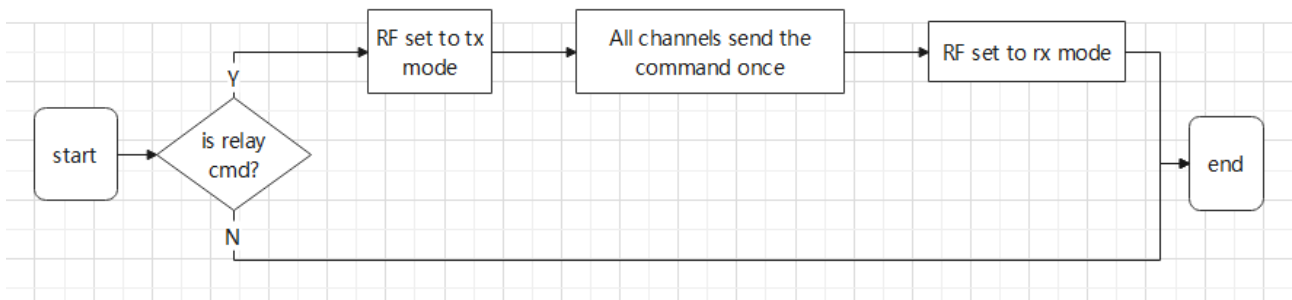


**Figure 8:** “remote command sending packet process”

Take the light project as an example. The function that forwards the command is described as below.  
Light forwards the received remote control commands to other light devices.

```
void rfc_send_relay_pkt(void)
```

Forward incoming remote commands, ttl minus 1, to prevent infinite forwarding.  
The logic flow chart for the sending packet is as below.



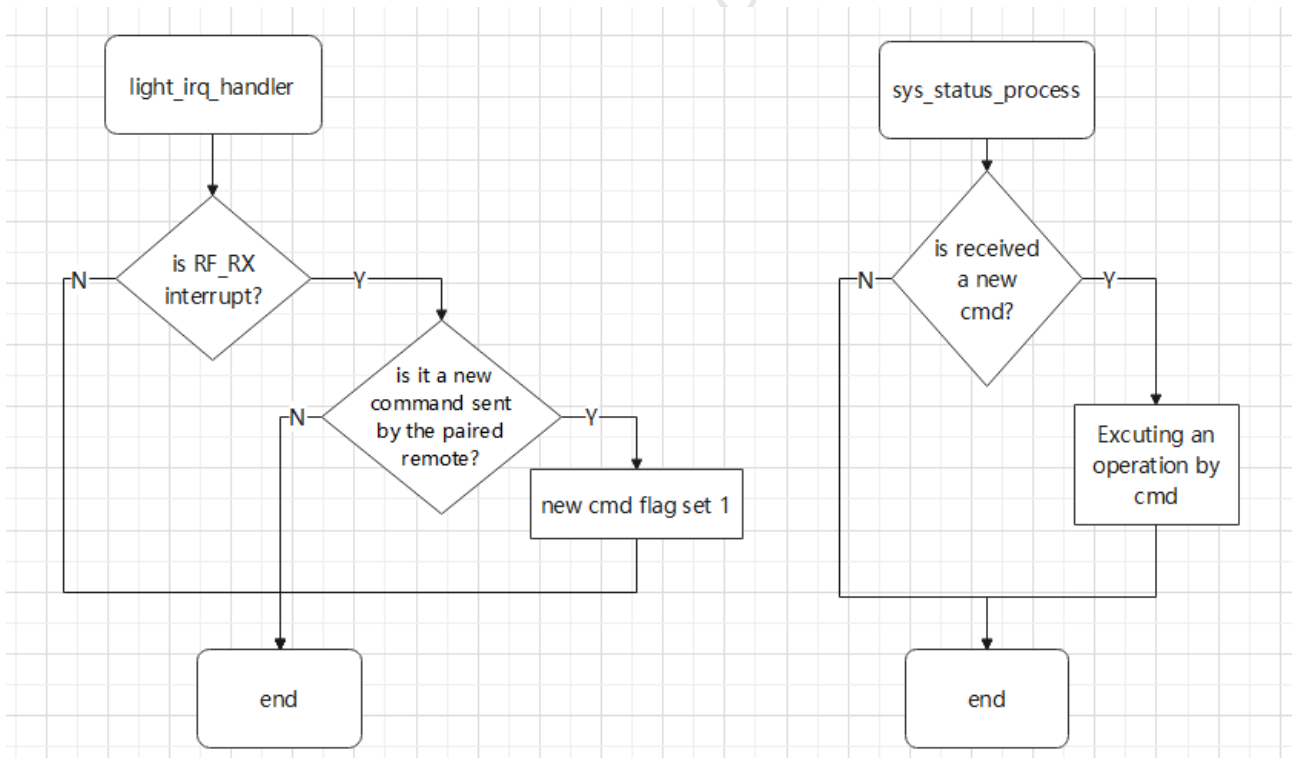
**Figure 9:** "light command forwarding flow"

Interface description for packet receiving.

```
void sys_status_process(void)
```

The interrupt function "void light\_irq\_handler(void)" determines that the data received is a new command from the paired remote. The corresponding control command is then processed in the function "void sys\_status\_process(void)".

The logic flow chart for packet receiving is as below.



**Figure 10:** "light command response flow"

## 14 remote matrix key scan

Configuration of the number of rows and columns of the key matrix:

```
#define KB_COL_NUM    5
#define KB_ROW_NUM    3
```

Pin configuration of key rows and columns.

```
gpio_column[KB_COL_NUM]={GPIO_PB0,GPIO_PB1,GPIO_PB4,GPIO_PB5,GPIO_PB6};//Column IO for the
↪ matrix
gpio_row[KB_ROW_NUM]    ={GPIO_PD3,GPIO_PD6,GPIO_PA0};//Row IO for the matrix
```

Key configuration:

```
key_table[KB_ROW_NUM][KB_COL_NUM] //Key table
```

Key scan principle: When a key is connected to an IO port at both ends, one IO port is set high and the other is set low, when the key is pressed the high IO port level is pulled low, the other end is still low, at this time the detection IO port value is two low levels.

## 1.5 EEPROM

The EEPROM is currently built into the B80 chip internally to store information about paired pids and groups, and is 256bytes in size.

```
void eeprom_init();
```

Configure the initialization of the EEPROM, and the initialization of the internal EEPROM's pins.

```
void eeprom_write (u8 adr, u8 *p, int len);
```

The adr is the internal address of the EEPROM, p is the header pointer to the write buffer, and len is the length of the write to the EEPROM.

```
void eeprom_read (u8 adr, u8 *p, int len);
```

The adr is the internal address of the EEPROM, p is the header pointer to the read buffer, and len is the length of the read to the EEPROM.

## 1.6 LED

The principle of luminance increase and decrease keys and chroma increase and decrease keys tuning the LED change is as below.

The light is currently preset to 10 levels of variation, each parameter is stored in the arrays led\_luminance\_value and led\_chroma\_value. After receiving the luminance or chroma change command, the next parameter

is jumped to as the target value of the LED. After the LED gradually change to the target value, it will no longer change.

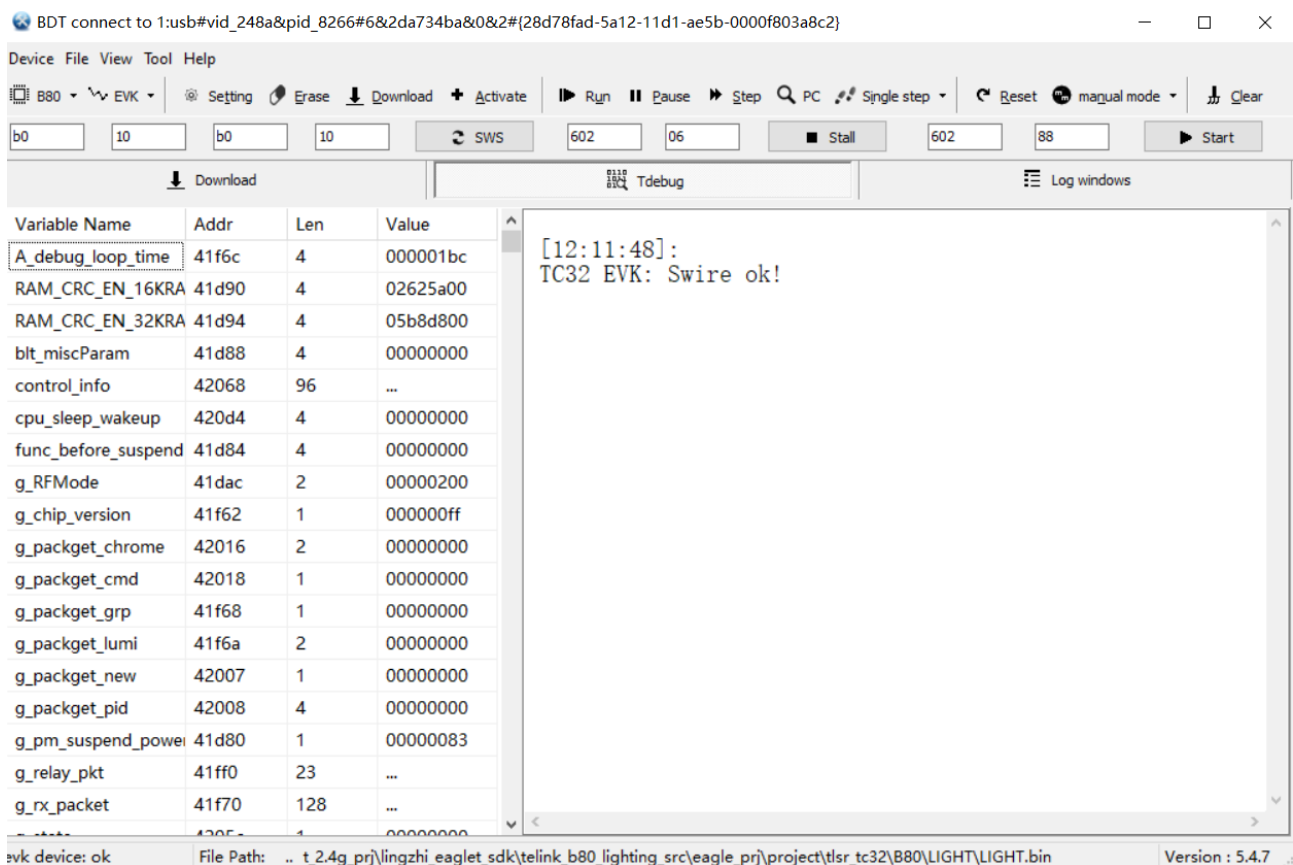
LED color gradual change process is as below.

Check every 5ms, when there is a difference between the set target luminance or chroma and the current value, adjust the current parameter in a single step (luminance single step change to 10, chroma single step change to 1, not enough to align the target value), until the adjustment to the target value. The process of color gradient is completed.

## 1.7 SDK debugging instructions

### 1.7.1 Tdebug

Debug by looking at the corresponding variables through the wtcdB tool.



**Figure 11:** “wtcdB debug”

### 1.7.2 Serial print debugging instructions

```
#define DEBUG_MODE 1
```

The print function is turned on and off via DEBUG\_MODE.

```
#define PRINT_BAUD_RATE      115200 //1M baud rate,should NOT be bigger than 1Mb/s
#define DEBUG_INFO_TX_PIN    GPIO_PD3
```

Configure the baud rate of the serial port via PRINT\_BAUD\_RATE and the interface for printing via DEBUG\_INFO\_TX\_PIN. The printf is used as the interface for standard printing and printhex prints arrays.

DEBUG\_MODE =0 turns off the print mode, and during the compilation process, no related functions are programmed and printed. The customer can turn it on during the debugging phase, and it is recommended to turn it off to save ram and code in the mass production program.

Telink Semiconductor

## 2 SDK code description

The three modes, remote-light, remote-rgb light, and remote-beacon, are introduced in detail in this chapter. The initialization interface and the processing inside the main loop are introduced using the light project as an example.

### 2.1 REMOTE

#### 2.1.1 Introduction of initialization function interface for REMOTE

```
led_gpio_init(LED1)
```

Initialize the LED pins. Only 1 pin can be initialized at a time.

```
keyscan_gpio_init();
```

Initialize key matrix pins.

```
package_data_init_func()
```

Initialize the sending packet data, includes the solidified VID, the PID read from the OTP, and the group&rf\_seq\_no recovered from the analog register.

```
rfc_init_func()
```

Initialize the RF module. Configure the RF module as tx\_mode for sending commands.

#### 2.1.2 Loop interface introduction for REMOTE

```
unsigned char keyscan_scan_func(void)
```

It scans the key matrix and returns key values. Send control commands by key press. (Lift OFF command key to start sending, long press OFF to send night light mode command, color temperature and luminance change support long press more than 320ms to send new command continuously, press other command keys to send commands.)

### 2.2 LIGHT

#### 2.2.1 Introduction of initialization function interface for LIGHT



```
rfc_init_func()
```

Initialize the RF module. Configure the RF module to Rx\_mode for receiving remote commands.

```
led_pwm_init_func()
```

Configure the LED pin to PWM function. Subsequently, change the LED luminance and chromaticity as well as the on and off state by changing the PWM waveform.

```
led_init_func()
```

Restore the lighted state of light according to the previously saved parameters.

```
sys_status_init()
```

Initializes the system state. By default network state is set.

#### 2.2.2 Loop interface introduction for LIGHT

```
void sys_status_process(void)
```

Process the control commands received by the RF module and trigger the corresponding operations.

```
led_task_process_func()
```

Process LED blinking. Every 500ms, the luminance changes from 500 to 0 and then from 0 to 500. It flashes cyclically.

Process LED luminance and chromaticity gradients. Every 5ms, single step changes the current luminance and chromaticity parameters of the LED, gradually approaching the target luminance and chromaticity and reaching the target to save the LED parameters.

Change the LED local mode information. If the non-local mode is powered up for more than 500ms, the status "Next power up does not require switching to local mode" is saved. It will not enter local mode in the next power-up.

```
time_event_process_func()
```

Switch RF's receiving channel timely. (By default every 20ms switch once, 4-channel cyclic switching)

```
sys_status_check_func()
```

Check system status. If the pairing mode exceeds 6s, switch to normal mode. If the interval between CMD\_ON commands exceeds 500ms, exit the clear code state.

## 2.3 LIGHT\_BEACON

Same as 2.2.

## 2.4 REMOTE\_BEACON

Same as 2.3.

## 2.5 LIGHT\_RGB

```
rf_packget_pro_func()
```

Process the control commands received by the RF module and trigger the corresponding operations.

```
void led_task_process_func(void)
```

Process LED blinking. Every 500ms, the luminance changes from 500 to 0 and then from 0 to 500. It flashes cyclically.

Process LED luminance and chromaticity gradients. Every 5ms, single step changes the current luminance and chromaticity parameters of the LED, gradually approaching the target luminance and chromaticity and reaching the target to save the LED parameters.

RGB color gradient for LEDs, includes flowing light function and RGB display.

## 2.6 REMOTE\_RGB

Same as 2.3.

### 3 SDK configuration file description

#### 3.1 Configuration of RF parameters and frequencies

Accesscode: by modifying RF\_ACCESS\_CODE\_USE

Frequency: LIGHT&REMOTE and LIGHT RGB/REMOTE RGB use frequency {1,24,51,76} by default, change the variable rf\_channel[4] to change the frequency. LIGHT BEACON&REMOTE BEACON use fixed frequency: {37,38,39}

Transmit Power:

```
#define RF_POWER          RF_POWER_P11p46dBm
```

Mode selection for transmitting:

```
#define RF_MODE            RF_PRIVATE_2M
```

#### 3.2 Configuration of IO ports

```
#define LED_R              GPIO_PB3//red
#define LED_G              GPIO_PB4//green
#define LED_B              GPIO_PB5//blue
#define LED_Y              GPIO_PB6//yellow
#define LED_W              GPIO_PD4//white
```

Modify the configuration of different LEDs by modifying the definition of their GPIO.

#### 3.3 Modify main frequency

You can modify the main frequency by modifying the macro definition CLOCK\_SYS\_CLOCK\_HZ in the app\_config.h file, the macro definition options are 12M, 16M, 24M, 32M, and 48M.

#### 3.4 Configure the product VID

```
#define REMOTE_VID         0x5453
```

#### 3.5 Configure the OTP address of the REMOTE PID

```
#define PID_ADDR          0x3fe0
```

### 3.6 Configure the number of times the command sending

```
#define TTL_MAX           5 //forwardable times of remote command
#define NUM_SENDING_CMD_CTR 15//minimum number of times a single control command is sent,
↪ unless a new command is switched midway
#define NUM_SENDING_CMD_NONE 5//null command retransmission count
```

Telink Semiconductor

## 4 Operation introduction

### 4.1 Key functions of REMOTE& LIGHT and REMOTE\_BEACON& LIGHT\_BEACON



**Figure 12:** "REMOTE&LIGHT and REMOTE\_BEACON&LIGHT\_BEACON keys introduction"

#### All On:

Short or long press to send out an All On command to turn on all nearby lights paired with the remote.

#### All Off:

Press "<" for 1.5s, the key lifts and sends the All Off command to turn off all nearby lights paired with the remote.

Press the key ">" for 1.5s to send the all into night light mode command to put all nearby lights paired with the remote into night light mode.

#### Group 1 On:

Short or long press to send out a Group 1 On command to turn on all nearby lights that are paired with Group 1.

#### Group 1 Off:

Press "<" for 1.5s, the key lifts and sends the Group 1 Off command, turning off all nearby lights paired with Group 1.

Press the key ">" for 1.5s to send the group 1 into night light mode command to put all nearby lights paired with group 1 into night light mode.

**Group 2 on / group 3 on / group 4 on:** Ditto for "Group 1 On".

**Group 2 off / group 3 off / group 4 off:** Ditto for "Group 1 Off".

**Luminance increase / Luminance decrease:**

Select the group (press the group on/off key, e.g. "Group 1 On" or "Group 1 Off") and press the Lumi\_inc / Lumi\_dec key.

Short press of the key sends a single command in a single pass. The luminance changes once.

Long press of the key for about 320ms each, to send a continuous command and the luminance changes continuously.

**Chromaticity increase / chromaticity decrease:**

Select the group (press the group on/off key, e.g. "Group 1 On" or "Group 1 Off") and press the Chroma\_inc / Chroma\_dec key.

Short press of the key sends a single command in a single pass. The chroma changes once.

Long press of the key for about 320ms each, and the command is sent continuously and the chroma changes continuously.

## 4.2 Key functions of REMOTE\_RGB&LIGHT\_RGB



**Figure 13:** "REMOTE\_RGB&LIGHT\_RGB keys introduction"

**Color temperature light on:**

Short or long press to send out a light on command to turn on all the nearby color temperature lights paired with the remote.

**Color temperature light off:**

Short or long press to send out a light off command to turn off all nearby color temperature lights paired with the remote.

**Luminance increase / Luminance decrease / chromaticity increase /chromaticity decrease:**

Short press the key to send a single command, light led changes once.

Long press the key, 320ms timed update command, light led changes continuously.

RGB breathing light mode does not respond to color temperature and luminance adjustment commands.

**Night light mode:**

Short or long press to send out a night light mode command to put all nearby color temperature lights paired with the remote into night light mode.

**RGB breathing light mode:**

Short or long press to send out a night light mode command to put all the nearby color temperature lights paired with the remote into RGB breathing light mode.

**Pair code:**

light powers up < 6s, short press or long press the remote to send out pairing command, light receives command, retains pairing information and flashes light 3 times.

light power on > 6s, short press or long press the remote to send out pairing command, no action on the light.

**Clear Code:**

light powers up < 6s, short press or long press the remote to send out a clear code command, light receives the command, clears the pairing information and flashes 5 times.

light power on > 6s, short or long press the remote to send clear code command, no action on the light.

**RGB settings:**

Short or long press to send out the RGB setting command to turn on all RGB lights of nearby color temperature lights paired with the remote by 50%.