RK3308 Led 接口介绍

文件标识：RK-KF-YF-319

发布版本：V1.0.1

作者：Jacky Ge

日期：2020-03-02

文件密级：□绝密 □秘密 □内部资料 ■公开

瑞芯微电子股份有限公司

Rockchip Electronics Co., Ltd.

地址：　福建省福州市铜盘路软件园A区18号

网址：　www.rock-chips.com

客户服务电话：+86-4007-700-590

客户服务传真：+86-591-83951833

客户服务邮箱：fae@rock-chips.com

前言

概述

该文档旨在介绍RK3308 DeviceIo库中接口。

芯片名称

RK3308

读者对象

本文档（本指南）主要适用于以下工程师：

技术支持工程师

软件开发工程师

修订记录

日期	版本	作者	修改说明
2019-3-29	V1.0.0	Jacky Ge	初始版本
2020-03-02	V1.0.1	Ruby Zhang	调整文档格式，更新文档名称

目录

1. 概述

该代码模块集成在libDeviceIo.so动态库里面，基于PWM驱动的单个RGB Led灯，封装了包括Led等的亮灭、闪烁灯效、呼吸灯效等接口。采用分层设计以适应不同的业务场景需求，支持灯效的优先级设定，可根据现有接口构建复杂的灯效需求。

整个框架分为三层：TEMP、REALTIME、STABLE。

TEMP ：只包含单个灯效，优先级最高。可用于处理类似于按键提示灯等时间较短的灯效。

REALTIME：只包含单个灯效，优先级次于TEMP。可用于处理一整套事务流程下LED的状态切换，如智能音响的Recording、Recognize和Response的状态切换。

STABLE ：包含一个支持优先级设定的灯效栈，始终取栈顶灯效，优先级次于REALTIME。可用于处理设备的状态，如低电量、静麦模式、配网模式等。

综上，若TEMP层有元素，始终显示TEMP层元素；否则检查REALTIME层是否有元素，有则显示REALTIME层元素，反之显示STABLE层栈顶元素。若STABLE层栈空则等待。

2. 接口说明

- `RK_Led_Effect_layer_e`

effect layer枚举类型，包含TEMP、REALTIME和STABLE层。在设定灯效的时候需要被指定。

```
typedef enum RK_Led_Effect_layer {
    Led_Effect_layer_TEMP = 0,
    Led_Effect_layer_STABLE,
    Led_Effect_layer_REALTIME
} RK_Led_Effect_layer_e;
```

- `RK_Led_Effect_type`

effect type结构体类型，包含NONE、BLINK和BREATH灯效效果。在设定灯效的时候需要被指定。

```
typedef enum RK_Led_Effect_type {
    Led_Effect_type_NONE = 0,
    Led_Effect_type_BLINK,
    Led_Effect_type_BREATH
} RK_Led_Effect_type_e;
```

- `RK_Led_Effect`

effect 灯效结构体类型。设置灯效的时候需要传入的结构体参数

```
  typedef struct RK_Led_Effect {
      int period;                     // 灯效周期，例如呼吸一次为3000ms. <=0 表示周
期无限大
      int timeout;                    // 超时时间，<=0 表示无限大
      int colors;                     // 灯效需要显示的RGB值，如0xFFFFFF
      int colors_blink;               // 闪烁灯效，其他灯效不需要设置
      int priority;                   // 灯效优先级
      char name[64];                  // 灯效名称
      RK_Led_Effect_type_e type;      // 灯效类型
      RK_Led_Effect_layer_e layer;    // 灯效层级
  } RK_Led_Effect_type_e;
```

- `int RK_led_init(void)`

Led模块初始化，初始化相关参数。

- `int RK_set_all_led_status(const int Rval, const int Gval, const int Bval)`

设置Led灯的基础接口，传入参数为对应的RGB值（0x00-0xFF）

- `int RK_set_all_led_off(void)`

关闭Led灯基础接口

- `int RK_set_led_effect(RK_Led_Effect *effect)`

设置Led灯效，参数为effect结构体

- `int RK_set_led_effect_off(const RK_Led_Effect_layer_e layer, const char *name)`

关闭指定层级指定名称的灯效。（如果关闭的是当前显示的灯效，会自动显示上一个灯效）

- `int RK_set_all_led_effect_off(void)`

清除所有设置的effect，并关闭Led灯

- `int RK_led_exit(void)`

Led模块反初始化，释放资源

3. 使用示例

```
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <DeviceIo/Rk_led.h>

static void rk_led_effect_default(RK_Led_Effect_t *effect)
{
    effect->period = -1;
    effect->timeout = -1;
    memset(effect->name, 0, sizeof(effect->name));
    effect->layer = Led_Effect_layer_TEMP;
    effect->colors = 0;
    effect->colors_blink = 0;
    effect->priority = 0;
}
```

```c
static int remove_layer(const RK_Led_Effect_layer_e layer, const char *name)
{
    if (!name || strlen(name) == 0) {
        if (Led_Effect_layer_STABLE == layer) {
            return -1;
        } else {
            RK_set_led_effect_off(layer, "");
            return 0;
        }
    }

    RK_set_led_effect_off(layer, name);
    return 0;
}

// STABLE层级的Red Led呼吸灯，周期为1000ms
int stable_breath_red(const char *name)
{
    if (name == NULL)
        return -1;

    RK_Led_Effect_t effect;
    rk_led_effect_default(&effect);

    effect.colors = 0xFF0000;
    effect.period = 1000;
    effect.type = Led_Effect_type_BREATH;
    effect.layer = Led_Effect_layer_STABLE;
    strncpy(effect.name, name, sizeof(effect.name));

    RK_set_led_effect(&effect);
    return 0;
}

// STABLE层级的Red Led闪烁灯，周期为1000ms
int stable_blink_red(const char *name)
{
    if (name == NULL)
        return -1;

    RK_Led_Effect_t effect;
    rk_led_effect_default(&effect);

    effect.colors = 0xFF0000;
    effect.period = 1000;
    effect.type = Led_Effect_type_BLINK;
    effect.layer = Led_Effect_layer_STABLE;
    strncpy(effect.name, name, sizeof(effect.name));

    RK_set_led_effect(&effect);
    return 0;
}

// REALTIME层级的Green Led闪烁灯，周期1000ms
int realtime_blink_green(void)
{
    RK_Led_Effect_t effect;
    rk_led_effect_default(&effect);

    effect.colors = 0x00FF00;
    effect.period = 1000;
    effect.type = Led_Effect_type_BLINK;
    effect.layer = Led_Effect_layer_REALTIME;

    RK_set_led_effect(&effect);
    return 0;
}

// TEMP层级的While Led灯
int temp_none_white(void)
{
    RK_Led_Effect_t effect;
    rk_led_effect_default(&effect);

    effect.colors = 0xFFFFFF;
    effect.type = Led_Effect_type_NONE;
    effect.layer = Led_Effect_layer_TEMP;

    RK_set_led_effect(&effect);
    return 0;
}

int main(int argc, char **argv)
{
    RK_led_init();
    // 重置Led灯状态
    RK_set_all_led_effect_off();

    // 显示红色Led呼吸灯效
    stable_breath_red("stable_breath_red");
    sleep(10);

    // 显示红色闪烁灯效
    stable_blink_red("stable_blink_red");
    sleep(10);

    // 移除红色闪烁灯效，自动显示上一次灯效，即红色呼吸灯效
    remove_layer(Led_Effect_layer_STABLE, "stable_blink_red");
    sleep(10);

    // 显示REALTIME层绿色闪烁灯效
    realtime_blink_green();
    sleep(10);

    // 显示TEMP层白色常亮
    temp_none_white();
    sleep(10);

    // 由于TEMP层有元素，还是显示TEMP层白色常亮
    realtime_blink_green();
    sleep(10);

    // 移除TEMP层白色灯效，自动显示REALTIME层绿色闪烁灯
    remove_layer(Led_Effect_layer_TEMP, "");
    sleep(10);
    // 移除REALTIME层灯效，自动显示STABLE红色呼吸灯效
    remove_layer(Led_Effect_layer_REALTIME, "");
    sleep(10);

    // 清除所有灯效，并关闭LED灯
    RK_set_all_led_effect_off();

    for (;;);
    RK_led_exit();

    return 0;
}
```