# RK3308 Key 接口介绍

文件标识：RK-KF-YF-318

发布版本：V1.0.1

日期：2020-03-02

文件密级：□绝密 □秘密 □内部资料 ■公开

**免责声明**

本文档按“现状”提供，瑞芯微电子股份有限公司（“本公司”，下同）不对本文档的任何陈述、信息和内容的准确性、可靠性、完整性、适销性、特定目的性和非侵权性提供任何明示或暗示的声明或保证。本文档仅作为使用指导的参考。

由于产品版本升级或其他原因，本文档将可能在未经任何通知的情况下，不定期进行更新或修改。

**商标声明**

“Rockchip”、“瑞芯微”、“瑞芯”均为本公司的注册商标，归本公司所有。

本文档可能提及的其他所有注册商标或商标，由其各自拥有者所有。

**版权所有© 2020 瑞芯微电子股份有限公司**

超越合理使用范畴，非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

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input_event输入子系统，对按键的常用需求，包括短按、长按、组合按键等需求做了封装处理，方便开发。

# 2. 接口说明

- `Callback函数定义`

最基础的Callback回调，会回调每一次按键的up和down事件：

```
typedef int (*RK_input_callback)(const int key_code, const int key_value);
```

经过处理的Callback回调，一次短按事件只会回调一次：

```
typedef int (*RK_input_press_callback)(const int key_code);
```

长按事件回调接口：

```
typedef int (*RK_input_long_press_callback)(const int key_code, const uint32_t
time);
```

心跳长按事件回调接口（即满足长按条件后，若保持按下则定时回调接口）：

```
typedef int (*RK_input_long_press_hb_callback)(const int key_code, const int
times);
```

组合按键回调接口：

```
typedef int (*RK_input_compose_press_callback)(const char* compose, const
uint32_t time);
```

事务按键回调接口：

```
typedef int (*RK_input_transaction_press_callback)(const char* trans, const
uint32_t time);
```

多次点击回调接口：

```
typedef int (*RK_input_multiple_press_callback)(const int key_code, const int
times);
```

按键模块初始化接口，需要传入一个基础的RK_input_callback 回调函数：

```
int RK_input_init(RK_input_callback input_callback_cb)
```

注册按键单击事件回调，按键单击触发：

```
- `int RK_input_register_press_callback(RK_input_press_callback cb)
```

为key_code按键注册时长为time ms的长按事件：

```
RK_input_register_long_press_callback(RK_input_long_press_callback cb, const
uint32_t time, const int key_code)
```

为key_code按键注册hb长按事件，每time ms触发一次：

```
int RK_input_register_long_press_hb_callback(RK_input_long_press_hb_callback cb,
const uint32_t time, const int key_code)
```

为key_code按键注册times次多击事件（即单击key_code times次，两两相差不超过500ms）：

```
int RK_input_register_multiple_press_callback(RK_input_multiple_press_callback
cb, const int key_code, const int times)
```

为key_code按键集注册组合事件，key_code按键集同时按下达到time ms触发：

```
int RK_input_register_compose_press_callback(RK_input_compose_press_callback cb,
const uint32_t time, const int key_code, ...)
```

为key_code按键集注册事务事件，按顺序依次按下key_code集后触发：

```
int
RK_input_register_transaction_press_callback(RK_input_transaction_press_callback
cb, const uint32_t time, int key_code, ...)
```

按键模块退出，并释放相关资源：

```
int RK_input_exit(void)
```

# 3. 使用示例

```
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <linux/input.h>
#include <DeviceIo/Rk_key.h>

static int _RK_input_callback(const int key_code, const int key_value)
{
    printf("_RK_input_callback key_code:%d; key_value:%d\n", key_code,
key_value);
```

```
    return 0;
}

static int _RK_input_press_callback(const int key_code)
{
    printf("_RK_input_press_callback key_code:%d;\n", key_code);
    return 0;
}

static int _RK_input_long_press_callback(const int key_code, const uint32_t time)
{
    printf("_RK_input_long_press_callback key_code:%d; time:%lu\n", key_code,
time);
    return 0;
}

static int _RK_input_long_press_hb_callback(const int key_code, const int times)
{
    printf("_RK_input_long_press_hb_callback key_code:%d; times:%d\n", key_code,
times);
    return 0;
}

static int _RK_input_multiple_press_callback(const int key_code, const int times)
{
    printf("_RK_input_multiple_press_callback key_code:%d; times:%d\n", key_code,
times);
    return 0;
}

static int _RK_input_transaction_press_callback(const char* trans, const uint32_t
time)
{
    printf("_RK_input_transaction_press_callback trans:%s; time:%lu\n", trans,
time);
    return 0;
}

static int _RK_input_compose_press_callback(const char* compose, const uint32_t
time)
{
    printf("_RK_input_compose_press_callback compose:%s; time:%lu\n", compose,
time);
    return 0;
}

int main(int argc, char **argv)
{
    // 初始化input模块
    RK_input_init(_RK_input_callback);
    // 注册单击回调
    RK_input_register_press_callback(_RK_input_press_callback);
    // 注册KEY_VOLUMEUP按键的5000ms长按事件
    RK_input_register_long_press_callback(_RK_input_long_press_callback, 5000,
KEY_VOLUMEUP);
    // 注册KEY_VOLUMEDOWN按键的hb长按事件, 每500ms触发一次hb
    RK_input_register_long_press_hb_callback(_RK_input_long_press_hb_callback,
500, KEY_VOLUMEDOWN);
```

```
    // 注册KEY_POWER的双击事件
    RK_input_register_multiple_press_callback(_RK_input_multiple_press_callback,
KEY_POWER, 2);
    // 注册KEY_VOLUMEUP->KEY_VOLUMEUP->KEY_VOLUMEDOWN->KEY_VOLUMEDOWN的事务事件

RK_input_register_transaction_press_callback(_RK_input_transaction_press_callback
, 2000, 4, KEY_VOLUMEUP, KEY_VOLUMEUP, KEY_VOLUMEDOWN, KEY_VOLUMEDOWN);
    // 注册KEY_VOLUMEUP + KEY_VOLUMEDOWN 5000ms的组合按键
    RK_input_register_compose_press_callback(_RK_input_compose_press_callback,
5000, 2, KEY_VOLUMEUP, KEY_VOLUMEDOWN);


    for (;;);

    RK_input_exit();
    return 0;
}
```