

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, 福州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, 由其各自所有者所有。

版权所有 © 2019 福州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。

福州瑞芯微电子股份有限公司

Fuzhou 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

RK3308 Key 接口介绍

前言

目录

1、概述

2、接口说明

3、使用示例

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：

```
1 | 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：

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

长按事件回调接口：

```
1 | 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）：

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

组合按键回调接口：

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

事务按键回调接口：

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

多次点击回调接口：

```
1 | 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：

```
1 | 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：

```
1 | - `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：

```
1 | 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：

```
1 | 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）：

```
1 | 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：

```
1 | 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：

```
1 | int  
   | RK_input_register_transaction_press_callback(RK_input_transaction_press_callb  
   | ack cb, const uint32_t time, int key_code, ...)
```

按键模块退出，并释放相关资源：

```
1 | int RK_input_exit(void)
```

3、使用示例

```
1  #include <stdio.h>
2  #include <string.h>
3  #include <unistd.h>
4  #include <linux/input.h>
5  #include <DeviceIo/Rk_key.h>
6
7  static int _RK_input_callback(const int key_code, const int key_value)
8  {
9      printf("_RK_input_callback key_code:%d; key_value:%d\n", key_code,
10 key_value);
11      return 0;
12  }
13
14  static int _RK_input_press_callback(const int key_code)
15  {
16      printf("_RK_input_press_callback key_code:%d;\n", key_code);
17      return 0;
18  }
19
20  static int _RK_input_long_press_callback(const int key_code, const uint32_t
21 time)
22  {
23      printf("_RK_input_long_press_callback key_code:%d; time:%lu\n",
24 key_code, time);
25      return 0;
26  }
27
28  static int _RK_input_long_press_hb_callback(const int key_code, const int
29 times)
30  {
31      printf("_RK_input_long_press_hb_callback key_code:%d; times:%d\n",
32 key_code, times);
33      return 0;
34  }
35
36  static int _RK_input_multiple_press_callback(const int key_code, const int
37 times)
38  {
39      printf("_RK_input_multiple_press_callback key_code:%d; times:%d\n",
40 key_code, times);
41      return 0;
42  }
43
44  static int _RK_input_transaction_press_callback(const char* trans, const
45 uint32_t time)
46  {
47      printf("_RK_input_transaction_press_callback trans:%s; time:%lu\n",
48 trans, time);
49      return 0;
50  }
51
52  static int _RK_input_compose_press_callback(const char* compose, const
53 uint32_t time)
54  {
55      printf("_RK_input_compose_press_callback compose:%s; time:%lu\n",
56 compose, time);
57      return 0;
58  }
59
60  static int _RK_input_key_callback(const int key_code, const int key_value)
61  {
62      printf("_RK_input_key_callback key_code:%d; key_value:%d\n", key_code,
63 key_value);
64      return 0;
65  }
66
67  static int _RK_input_key_press_callback(const int key_code)
68  {
69      printf("_RK_input_key_press_callback key_code:%d;\n", key_code);
70      return 0;
71  }
72
73  static int _RK_input_key_long_press_callback(const int key_code, const
74 uint32_t time)
75  {
76      printf("_RK_input_key_long_press_callback key_code:%d; time:%lu\n",
77 key_code, time);
78      return 0;
79  }
80
81  static int _RK_input_key_long_press_hb_callback(const int key_code, const
82 int times)
83  {
84      printf("_RK_input_key_long_press_hb_callback key_code:%d; times:%d\n",
85 key_code, times);
86      return 0;
87  }
88
89  static int _RK_input_key_multiple_press_callback(const int key_code, const
90 int times)
91  {
92      printf("_RK_input_key_multiple_press_callback key_code:%d; times:%d\n",
93 key_code, times);
94      return 0;
95  }
96
97  static int _RK_input_key_transaction_press_callback(const char* trans, const
98 uint32_t time)
99  {
100     printf("_RK_input_key_transaction_press_callback trans:%s; time:%lu\n",
101 trans, time);
102     return 0;
103 }
104
105 static int _RK_input_key_compose_press_callback(const char* compose, const
106 uint32_t time)
107 {
108     printf("_RK_input_key_compose_press_callback compose:%s; time:%lu\n",
109 compose, time);
110     return 0;
111 }
```

```
45     printf("_RK_input_compose_press_callback compose:%s; time:%lu\n",
compose, time);
46     return 0;
47 }
48
49 int main(int argc, char **argv)
50 {
51     // 初始化input模块
52     RK_input_init(_RK_input_callback);
53     // 注册单击回调
54     RK_input_register_press_callback(_RK_input_press_callback);
55     // 注册KEY_VOLUMEUP按键的5000ms长按事件
56     RK_input_register_long_press_callback(_RK_input_long_press_callback,
5000, KEY_VOLUMEUP);
57     // 注册KEY_VOLUMEDOWN按键的hb长按事件，每500ms触发一次hb
58
59     RK_input_register_long_press_hb_callback(_RK_input_long_press_hb_callback,
500, KEY_VOLUMEDOWN);
60     // 注册KEY_POWER的双击事件
61
62     RK_input_register_multiple_press_callback(_RK_input_multiple_press_callback,
KEY_POWER, 2);
63     // 注册KEY_VOLUMEUP->KEY_VOLUMEUP->KEY_VOLUMEDOWN->KEY_VOLUMEDOWN的事务事
件
64
65     RK_input_register_transaction_press_callback(_RK_input_transaction_press_cal
lback, 2000, 4, KEY_VOLUMEUP, KEY_VOLUMEUP, KEY_VOLUMEDOWN, KEY_VOLUMEDOWN);
66     // 注册KEY_VOLUMEUP + KEY_VOLUMEDOWN 5000ms的组合按键
67
68     RK_input_register_compose_press_callback(_RK_input_compose_press_callback,
5000, 2, KEY_VOLUMEUP, KEY_VOLUMEDOWN);
69
70     for (;;)
71     {
72         RK_input_exit();
73         return 0;
74     }
75 }
```