RK3308 Key Interface Introduction

Document ID: RK-KF-YF-318

Release Version: V1.0.1

Date: 2020-03-02

Security Level: □Top-Secret □Secret □Internal ■Public

Rockchip Electronics Co., Ltd.

No.18 Building, A District, No.89, software Boulevard Fuzhou, Fujian,PRC

Website: www.rock-chips.com

Customer service Tel: +86-4007-700-590

Customer service Fax: +86-591-83951833

Customer service e-Mail: fae@rock-chips.com

Preface

Overview

This document describes the interfaces in RK3308 DeviceIo library.

Chipset

RK3308

Intended Audience

This document (this guide) is intended primarily for the following engineers:

Field Application Engineer

Software Development Engineer

Revision History

Date	Revision No.	Author	Revision History
2019-3-29	V1.0.0	Jacky Ge	Initial version
2020-03-02	V1.0.1	Ruby Zhang	Update the format and the name of the document

Contents

1. Overview

This code module is integrated in the libDeviceIo.so dynamic library, based on the input_event input subsystem, including short press, long press, and combination key common requirements which are packaged to facilitate development.

2. Interface Introduction

- `Callback function definition`

It is the most basic callback, which will callback up and down events for each key press:

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

It is a processed callback, and only be called back once by a short press event:

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

Long press event callback interface:

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

Heartbeat long press event callback interface(That is, if the button is kept pressed, the callback interface will be callback regularly after meeting the long-press condition.)

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

Combination key callback interface:

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

Transaction key callback interface:

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

Multiple press callback interface:

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

Key module initialization interface, which need to be assigned a basic RK_input_callback function:

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

Register a key press event callback and triggered by single click:

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

Register a long press event with the time of "time ms" for key_code:

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

Register a "hb" long press event for key_code key, triggered every "time ms":

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

Register times multi-press events for key_code key (that is, times of pressing key_code, the time difference between each two press should not exceed 500ms):

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

Register combination events for key_code key set, triggered when key_code key set is pressed at the same time for time ms:

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

Register transaction events for the collection of key_code, which is triggered when the collection of key_code pressed in order:

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

Key module exits and releases related resources:

```
int RK_input_exit(void)`
```

3. Application Example

```
#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)
{
    // Initialize the input module
    RK_input_init(_RK_input_callback);
    // Register single press callback
    RK_input_register_press_callback(_RK_input_press_callback);
    // Register 5000ms long press event of KEY_VOLUMEUP key
    RK_input_register_long_press_callback(_RK_input_long_press_callback, 5000,
KEY_VOLUMEUP);
```

```
    // Register a hb long-press event of the KEY_VOLUMEDOWN button, triggering hb
every 500ms
    RK_input_register_long_press_hb_callback(_RK_input_long_press_hb_callback,
500, KEY_VOLUMEDOWN);
    // Register KEY_POWER double-press event
    RK_input_register_multiple_press_callback(_RK_input_multiple_press_callback,
KEY_POWER, 2);
    // Register KEY_VOLUMEUP->KEY_VOLUMEUP->KEY_VOLUMEDOWN->KEY_VOLUMEDOWN
transaction event

RK_input_register_transaction_press_callback(_RK_input_transaction_press_callback
, 2000, 4, KEY_VOLUMEUP, KEY_VOLUMEUP, KEY_VOLUMEDOWN, KEY_VOLUMEDOWN);
    // Register KEY_VOLUMEUP + KEY_VOLUMEDOWN 5000ms key combination
    RK_input_register_compose_press_callback(_RK_input_compose_press_callback,
5000, 2, KEY_VOLUMEUP, KEY_VOLUMEDOWN);


    for (;;);

    RK_input_exit();
    return 0;
}
```