XRADIO Button Developer Guide

Revision 1.1

Oct 11, 2019

Copyright @2019 Xradio Technology Co., Ltd. All Rights Reserved

Declaration

THIS DOCUMENTATION IS THE ORIGINAL WORK AND COPYRIGHTED PROPERTY OF XRADIO TECHNOLOGY ("XRADIO"). REPRODUCTION IN WHOLE OR IN PART MUST OBTAIN THE WRITTEN APPROVAL OF XRADIO AND GIVE CLEAR ACKNOWLEDGEMENT TO THE COPYRIGHT OWNER.

THE PURCHASED PRODUCTS, SERVICES AND FEATURES ARE STIPULATED BY THE CONTRACT MADE BETWEEN XRADIO AND THE CUSTOMER. PLEASE READ THE TERMS AND CONDITIONS OF THE CONTRACT AND RELEVANT INSTRUCTIONS CAREFULLY BEFORE USING, AND FOLLOW THE INSTRUCTIONS IN THIS DOCUMENTATION STRICTLY. XRADIO ASSUMES NO RESPONSIBILITY FOR THE CONSEQUENCES OF IMPROPER USE (INCLUDING BUT NOT LIMITED TO OVERVOLTAGE, OVERCLOCK, OR EXCESSIVE TEMPERATURE).

THE INFORMATION FURNISHED BY XRADIO IS PROVIDED JUST AS A REFERENCE OR TYPICAL APPLICATIONS, ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS DOCUMENT DO NOT CONSTITUTE A WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. XRADIO RESERVES THE RIGHT TO MAKE CHANGES IN CIRCUIT DESIGN AND/OR SPECIFICATIONS AT ANY TIME WITHOUT NOTICE.

NOR FOR ANY INFRINGEMENTS OF PATENTS OR OTHER RIGHTS OF THE THIRD PARTIES WHICH MAY RESULT FROM ITS USE. NO LICENSE IS GRANTED BY IMPLICATION OR OTHERWISE UNDER ANY PATENT OR PATENT RIGHTS OF XRADIO.THIRD PARTY LICENCES MAY BE REQUIRED TO IMPLEMENT THE SOLUTION/PRODUCT. CUSTOMERS SHALL BE SOLELY RESPONSIBLE TO OBTAIN ALL APPROPRIATELY REQUIRED THIRD PARTY LICENCES. XRADIO SHALL NOT BE LIABLE FOR ANY LICENCE FEE OR ROYALTY DUE IN RESPECT OF ANY REQUIRED THIRD PARTY LICENCE. XRADIO SHALL HAVE NO WARRANTY, INDEMNITY OR OTHER OBLIGATIONS WITH RESPECT TO MATTERS COVERED UNDER ANY REQUIRED THIRD PARTY LICENCE.

Copyright @2019 Xradio Technology Co., Ltd. All Rights Reserved

Revision History

Version	Date	Summary of Changes
1.0	2019-6-19	Initial Version
1.1	2019-10-11	版面调整

Table 1- 1 Revision History

Copyright @2019 Xradio Technology Co., Ltd. All Rights Reserved

Contents

Declaration....2
Revision History....3
Contents....4
Tables....6
Figures....7
1 概述....8
1.1 功能介绍....8
1.2 软件按键类型说明....8
1.2.1 短按按键....8
1.2.2 长按按键....8
1.2.3 长短按键....9
1.2.4 重复按键....9
1.2.5 组合按键....10
1.3 代码位置....10
1.3.1 模块源代码位置....10
1.3.2 模块示例代码位置....11
2 模块结构体....12
2.1 按键对象操作结构体....12
2.2 按键状态枚举....12
2.3 按键类型枚举....12
2.4 按键掩码....13
2.5 底层按键注册接口....13
3 模块接口....14
3.1 按键模块初始化....14
3.2 按键模块反初始化....14
3.3 设置按键模块栈大小....14
3.4 设置按键模块线程优先级....14

Copyright @2019 Xradio Technology Co., Ltd. All Rights Reserved

3.5 创建短按按键 15
3.6 创建长按按键 15
3.7 创建长短按键 16
3.8 创建重复按键 16
3.9 创建组合按键 17
4 模块使用示例 18
4.1 模块使用流程 18
4.1.1 添加按键的配置信息 18
4.1.2 模块初始化 19
4.1.3 创建按键对象 20
4.1.4 设置按键回调函数 21
4.1.5 启动按键 21
4.1.6 停止按键 22
4.1.7 销毁按键 22
5 模块测试命令 23
6 模块常见问题 24

Copyright @2019 Xradio Technology Co., Ltd. All Rights Reserved

Tables

表 1-1 模块源代码位置..... 10

表 1-2 模块示例代码位置..... 11

Copyright @2019 Xradio Technology Co., Ltd. All Rights Reserved

Figures

图 1-1 短按按键.....8
图 1-2 长按按键.....9
图 1-3 长短按键.....9
图 1-4 重复按键.....10
图 1-5 组合按键.....10

Copyright @2019 Xradio Technology Co., Ltd. All Rights Reserved

1 概述

1.1 功能介绍

button 模块用于需要用到按键且按键逻辑较复杂的场景中，button 模块能有效降低不同模块对按键操作逻辑的耦合性，能够独立开不同模块对按键操作的代码。

button 模块支持的硬件按键类型有：AD 按键、GPIO 按键；支持的软件按键类型有：短按按键、长按按键、长短按键、重复按键、组合按键。button 模块最多支持 32 个实体按键，不支持矩阵按键。

1.2 软件按键类型说明

按照按键行为进行分类，可分为五类：短按按键、长按按键、长短按键、重复按键、组合按键。

1.2.1 短按按键

短按按键没有按下的状态，只有释放的状态（RELEASE）。如果一个硬件按键被设置为短按，当该按键被按下时，短按按键对象不会触发按下的状态，只有当该按键释放时，短按按键对象才会触发释放的状态（RELEASE）。该类型按键可用于控制音量加/减、上/下一首、暂停/播放等操作，即只有在按键释放时执行相应的操作。

图 1-1 短按按键

1.2.2 长按按键

长按按键有按下（PRESS）和释放（RELEASE）两种状态。如果一个硬件按键被设置为长按，当该按键被按下并且按下的时间超过设定的时间时（假设 500ms），长按按键对象会触发按下的状态（PRESS），当按键释放时（假设在 1000ms 处释放），长按按键对象会触发释放状态（RELEASE）。如果按键提前释放了（假设在 300ms 处释放了），则按键对象不会触发任何状态。该类型按键几乎可以应用于任何场景，其他类型的按键

Copyright @2019 Xradio Technology Co., Ltd. All Rights Reserved

都可通过该类型按键和定时器来实现。

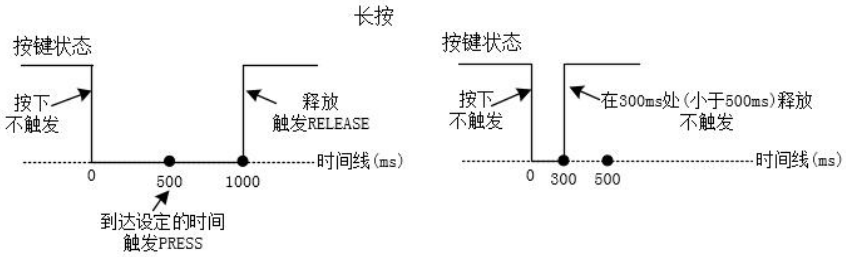

图 1-2 长按按键

1.2.3 长短按键

长短按键有按下（PRESS）、释放（RELEASE）、重复释放（REPEAT_RELEASE）三种状态，该类型按键将长按按键和短按按键相结合，同时具有长按和短按的特性。如果一个硬件按键被设置为长短按键，当该按键被按下并且按下的时间超过设定的时间时（假设 700ms），长短按键对象会触发按下的状态（PRESS），即长按的按下特性，当按键释放时（假设在 1000ms 处释放），长短按键对象会触发重复释放的状态（REPEAT_RELEASE），即长按的释放特性。如果按键提前释放了（假设在 300ms 处释放），则按键对象会触发释放状态（RELEASE），即短按的释放特性。该类型按键可用于长按录音场景，长按时间低于 500ms 时，播放“长按一会”提示音，长按时间超过 500ms 时，开始录音。

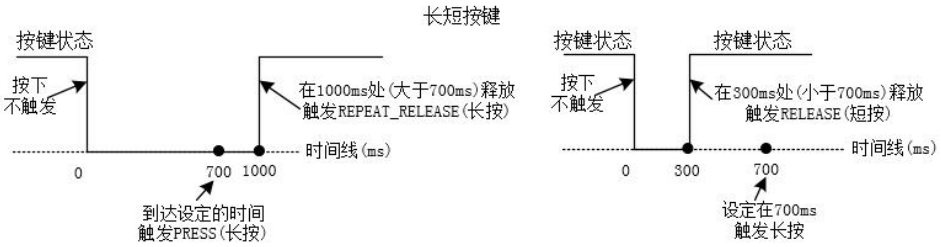

图 1-3 长短按键

1.2.4 重复按键

重复按键有按下（PRESS）、重复按下（REPEAT_PRESS）、释放（RELEASE）三种状态。如果一个硬件按键被设置为重复按键，当该按键被按下并且按下的时间超过设定的时间时（假设 500ms），重复按键对象会触发按下的状态（PRESS），后面每隔一定的时间（假设 300ms），按键对象会触发一次重复按下的状态（REPEAT_PRESS），当按键释放时，按键对象会触发释放状态（RELEASE）。如果按键提前释放了（假设在 300ms 处释放），则按键对象不会触发任何状态。该类型按键可用于，长按时需要重复执行某项操作的场景，比如长按音量键，音量持续增加或减小。

Copyright @2019 Xradio Technology Co., Ltd. All Rights Reserved

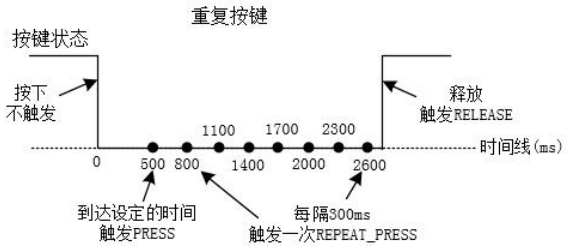

图 1-4 重复按键

1.2.5 组合按键

组合长按是长按按键的一种扩展，支持多个硬件按键，即多个硬件按键对应一个按键对象，其他类型的按键只支持单个硬件按键，即一个硬件按键对应一个按键对象，其有按下（PRESS）和释放（RELEASE）两种状态。如果 K1、K2 被设置为组合长按，当 K1、K2 同时按下并且按下的时间超过设定的时间时（假设 500ms），组合按键对象会触发按下的状态（PRESS），当任何一个按键释放了（假设在 1000ms 处释放），组合按键对象会触发释放状态（RELEASE）。如果按键提前释放了（假设在 300ms 处释放），则按键对象不会触发任何状态。该按键类型一般需要多个按键同时触发的场景，比如同时按上一首/下一首，触发配网按键对象，进入配网场景。

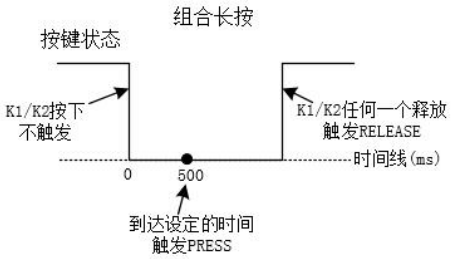

图 1-5 组合按键

1.3 代码位置

1.3.1 模块源代码位置

表 1-1 模块源代码位置

文件名	位置
button.c	sdk/project/common/apps/buttons/buttons.c
	sdk/project/common/apps/buttons/buttons.h
low_level_button.c	sdk/project/common/apps/buttons/buttons_low_level.c
	sdk/project/common/apps/buttons/buttons_low_level.h

Copyright @2019 Xradio Technology Co., Ltd. All Rights Reserved

1.3.2 模块示例代码位置

表 1-2 模块示例代码位置

文件名	位置
main.c	sdk/project/example/button/main.c
board_config.c	sdk/project/common/board/xradio_storybot/board_config.c

Copyright @2019 Xradio Technology Co., Ltd. All Rights Reserved

2 模块结构体

2.1 按键对象操作结构体

```
typedef struct button_handle {
    void (*start)(struct button_handle *handle); //使能按键对象
    void (*stop)(struct button_handle *handle); //失能按键对象
    int (*destroy)(struct button_handle *handle); //销毁按键对象
    int (*get_state)(struct button_handle *handle); //获取按键对象状态
    void (*cb)(BUTTON_STATE sta, void *arg); //按键对象被触发时的回调函数
    void *arg; //回调函数传入的参数
} button_handle;
```

当一个按键对象被创建后，即可调用按键对象的接口来操作按键。如：short_button->start(short_button)、short_button->stop(short_button)、short_button->destroy(short_button)

2.2 按键状态枚举

```
typedef enum {
    PRESS, //用于长按按键、长短按键、组合按键、重复按键
    RELEASE, //用于短按按键、长按按键、长短按键、组合按键、重复按键
    REPEAT_PRESS, //用于重复按键
    REPEAT_RELEASE, //用于长短按键
    INVALID_STA, //无效按键状态
} BUTTON_STATE;
```

2.3 按键类型枚举

```
typedef enum {
    SHORT_BUTTON, //短按按键
    LONG_BUTTON, //长按按键
    SHORT_LONG_BUTTON, //长短按键
```

Copyright @2019 Xradio Technology Co., Ltd. All Rights Reserved

```
    COMBINED_LONG_BUTTON, //组合按键
    REPEAT_LONG_BUTTON, //重复按键
} BUTTON_TYPE;
```

2.4 按键掩码

宏 KEY0~宏 KEY31 每个硬件按键对应一个掩码（对应 1bit），一一对应，不能重复。

2.5 底层按键注册接口

```
typedef struct {
    int (*low_level_init)(void); //底层按键初始化
    int (*low_level_get_state)(void); //底层按键获取所有按键状态
    int (*low_level_wait_semaphore)(uint32_t waitms); //底层按键等待一个信号量，即等待硬件按键的触发信号
    int (*low_level_release_semaphore)(void); //底层释放一个信号量，即释放硬件按键的触发信号
    int (*low_level_deinit)(void); //底层按键反初始化
} button_impl_t;
```

该接口无特殊需求，可无需修改，直接注册 buttons_low_level.c 中的接口。

Copyright @2019 Xradio Technology Co., Ltd. All Rights Reserved

3 模块接口

3.1 按键模块初始化

int buttons_init(button_impl_t* impl)		备注
功能	注册底层按键接口，初始化按键模块。	若无特殊需求，参数可参考示例代码
参数	impl：底层按键接口。	
返回值	0：成功，-1：失败	

3.2 按键模块反初始化

int buttons_deinit(void)		备注
功能	反初始化按键模块。	在需要是释放按键模块资源时，可调用，但是一般情况下很少调用
参数	void	
返回值	0：成功，-1：失败	

3.3 设置按键模块栈大小

void buttons_set_thread_stack_size(uint32_t size)		备注
功能	设置按键模块内部线程栈大小，由于按键对象的回调函数都是在按键模块内部线程中被调用的，故线程栈大小需要根据应用代码来设置，默认栈大小为 512 字节。	需在按键模块初始化函数前调用
参数	size：栈大小	
返回值	void	

3.4 设置按键模块线程优先级

void buttons_set_thread_priority(uint32_t priority)	备注

Copyright @2019 Xradio Technology Co., Ltd. All Rights Reserved

功能	设置按键模块内部线程优先级，默认优先级为 3	需在按键模块初始化函数前调用，优先级不宜设置太高，太高优先级虽然能加快按键响应速度，但是可能会影响系统其它线程的正常运行。
参数	priority：优先级	
返回值	void	

3.5 创建短按按键

button_handle* create_short_button(uint32_t id_mask)		备注
功能	创建短按按键对象	
参数	id_mask：按键掩码 KEY0~KEY31	
返回值	按键对象的操作接口	

使用示例：

```
button_handle *short_button;

short_button = create_short_button(KEY0);
```

3.6 创建长按按键

button_handle* create_long_button(uint32_t id_mask, uint32_t timeout_ms)		备注
功能	创建长按按键对象	
参数	id_mask：按键掩码 KEY0~KEY31 timeout_ms：触发长按按键的时间，按键在小于 timeout_ms 时释放，操作无效；在 timeout_ms 时，按键对象的回调函数会被调用，并传递 PRESS 状态；在大于 timeout_ms 时释放，按键对象回调函数会被调用，并传递 RELEASE 状态。	
返回值	按键对象的操作接口	

使用示例：

```
button_handle *long_button;

long_button = create_long_button(KEY1, 50); //KEY1 按下时间超过 50ms，才算有效操作
```

Copyright @2019 Xradio Technology Co., Ltd. All Rights Reserved

3.7 创建长短按键

button_handle* create_short_long_button(uint32_t id_mask, uint32_t timeout_ms)		备注
功能	创建长短按键对象	
参数	id_mask：按键掩码 KEY0~KEY31 timeout_ms：长按和短按的时间分界点，按键在小于 timeout_ms 时释放，触发短按特性，按键对象回调函数会被调用，并传递 RELEASE 状态；在 timeout_ms 时，触发长按特性，按键对象回调函数会被调用，并传递 PRESS 状态；在大于 timeout_ms 时释放，按键回调函数会被调用，并传递 REPEAT_RELEASE 状态。	
返回值	按键对象的操作接口	

使用示例：

```
button_handle *short_long_button;

short_long_button = create_short_long_button(KEY2, 500); //500ms 为长按和短按的分界点
```

3.8 创建重复按键

button_handle* create_repeat_long_button(uint32_t id_mask, uint32_t timeout_ms, uint32_t repeat_timeout_ms)		备注
功能	创建重复按键对象	
参数	id_mask：按键掩码 KEY0~KEY31 timeout_ms：触发重复按键的时间，按键在小于 timeout_ms 时释放，操作无效；在 timeout_ms 时，按键对象回调函数会被调用，并传递 PRESS 状态；在大于 timeout_ms 时释放，按键回调函数会被调用，并传递 RELEASE 状态。 repeat_timeout_ms：重复触发的时间间隔，当按键按下的时间超过 timeout_ms 且一直按着时，每隔 repeat_timeout_ms 时间，按键对象会被调用，并触发 REPEAT_PRESS 状态。	
返回值	按键对象的操作接口	

使用示例：

```
button_handle *repeat_button;

repeat_button = create_repeat_long_button(KEY3, 500, 300); //在 500ms 时触发按键，之后每隔 300ms 触发一次
```

Copyright @2019 Xradio Technology Co., Ltd. All Rights Reserved

3.9 创建组合按键

button_handle* create_combined_long_button(uint32_t id_mask, uint32_t timeout_ms)		备注
功能	创建组合按键对象	组合按键的种类有同通道的AD+AD、不同通道的AD+AD、AD+GPIO、GPIO+GPIO四种方式，只有同通道的AD+AD组合方式需要事先测量组合下的AD值并配置到board_config.c中（详见”模块使用事例“部分），其他三种方式都可以在创建对象时随意组合。
参数	id_mask：按键掩码 KEY0~KEY31 timeout_ms：触发组合按键的时间，组合按键在小于timeout_ms时释放，操作无效；组合按键同时按下的时间达到timeout_ms时，按键对象回调函数会被调用，并传递PRESS状态；在大于timeout_ms时，任何一个按键释放了导致按键组合被打破，按键回调函数会被调用，并传递RELEASE状态。	
返回值	按键对象的操作接口	

使用示例：

```
button_handle *combined_button;

combined_button = create_combined_long_button(KEY1 | KEY2, 50);
```

Copyright @2019 Xradio Technology Co., Ltd. All Rights Reserved

4 模块使用示例

4.1 模块使用流程

具体代码可参考 sdk/project/example/button/main.c ， 该工程代码用的 board_config 为 sdk/board/xradio_storybot/board_config.c

4.1.1 添加按键的配置信息

修改对应工程的 board_config.c，添加或修改按键的配置信息。

```
#include "common/apps/buttons/buttons.h"
#include "common/apps/buttons/buttons_low_level.h"

/* do not set const */
static ad_button g_ad_buttons[] = {
    {.name = "mode", .mask = KEY0, .channel = ADC_CHANNEL_0, .value = 815, .debounce_time = 50},
    {.name = "wechat", .mask = KEY1, .channel = ADC_CHANNEL_0, .value = 1515, .debounce_time = 50},
    {.name = "voice", .mask = KEY2, .channel = ADC_CHANNEL_0, .value = 2250, .debounce_time = 50},
    {.name = "next", .mask = KEY3, .channel = ADC_CHANNEL_0, .value = 3560, .debounce_time = 50},
    {.name = "prev", .mask = KEY4, .channel = ADC_CHANNEL_0, .value = 3755, .debounce_time = 50},
    {.name = "prev+next", .mask = KEY3 | KEY4, .channel = ADC_CHANNEL_0, .value = 2800, .debounce_time = 50},
};

__xip_rodata
static const gpio_button g_gpio_buttons[] = {
    {.name = "play", .mask = KEY5, .active_low = 1, { GPIO_PORT_A, GPIO_PIN_7,    { GPIOA_P7_F6_EINTA7, GPIO_DRIVING_LEVEL_1, GPIO_PULL_UP } }, .debounce_time = 50},
};

static HAL_Status board_get_cfg(uint32_t major, uint32_t minor, uint32_t param)
```

Copyright @2019 Xradio Technology Co., Ltd. All Rights Reserved

```
{
    HAL_Status ret = HAL_OK;
    switch (major) {
     •••  //其他代码
    case HAL_DEV_MAJOR_AD_BUTTON:
        ((ad_button_info *)param)->ad_buttons_p = g_ad_buttons;
        ((ad_button_info *)param)->count = sizeof(g_ad_buttons) / sizeof(g_ad_buttons[0]);
        break;
    case HAL_DEV_MAJOR_GPIO_BUTTON:
        ((gpio_button_info *)param)->gpio_buttons_p = g_gpio_buttons;
        ((gpio_button_info *)param)->count = sizeof(g_gpio_buttons) / sizeof(g_gpio_buttons[0]);
        break;
     •••  //其他代码
    }
    return ret;
}
```

g_ad_buttons 为 AD 按键的配置信息。name 为按键的名称；mask 为按键的掩码，每个硬件按键对应一个掩码；channel 为 AD 按键的通道号，不同通道号的 AD 按键也可放于此数组；value 为 AD 按键按下后的 AD 值，需要通过实际测量并取平均；debounce_time 为按键消抖时间。

如果同一个通道下的 AD 按键需要组合按键的话，需要将组合按键的 AD 值添加到 g_ad_buttons 数组中，如组合按键 prev+next，由 KEY3 和 KEY4 组合，都在通道 0，故需要采集 KEY3 和 KEY4 同时按下时的 AD 值。不同 AD 通道的组合按键、GPIO 与 AD 的组合按键或者 GPIO 与 GPIO 的组合按键不需要添加配置信息，可在创建按键对象时直接组合。

g_gpio_buttons 为 GPIO 按键的配置信息。name 为按键名称；mask 为按键掩码；active_low 为低电平有效，如果为 1，则表示按键按下后，状态为低电平；{ GPIO_PORT_A, GPIO_PIN_7, { GPIOA_P7_F6_EINTA7, GPIO_DRIVING_LEVEL_1, GPIO_PULL_UP } }为 gpio 引脚参数；debounce_time 为消抖时间。

board_get_cfg 函数是外部模块获取 g_ad_buttons 和 g_gpio_buttons 的接口，需要添加 g_ad_buttons 和 g_gpio_buttons。

4.1.2 模块初始化

注册底层按键接口，设置优先级，栈大小，初始化按键模块。

```
#include "common/apps/buttons/buttons.h"
```

Copyright @2019 Xradio Technology Co., Ltd. All Rights Reserved

```
#include "common/apps/buttons/buttons_low_level.h"


int example_buttons_init(void)
{
	int ret;
	/* register the low level button interface */
	button_impl_t impl = {
		buttons_low_level_init,
		buttons_low_level_get_state,
		buttons_low_level_wait_semaphore,
		buttons_low_level_release_semaphore,
		buttons_low_level_deinit
	};
	/* set buttons thread priority and stack size */
	buttons_set_thread_priority(4);
	buttons_set_thread_stack_size(1024);
	/* init buttons, will init the low level buttons, ad buttons and gpio buttons */
	ret = buttons_init(&impl);
	return ret;
}
```

4.1.3 创建按键对象

```
#define MODE	KEY0
#define WECHAT KEY1
#define VOICE	KEY2
#define NEXT	KEY3
#define PREV	KEY4
//全局变量，其他场景需要用到按键
button_handle *short_button;
```

Copyright @2019 Xradio Technology Co., Ltd. All Rights Reserved

```
button_handle *long_button0;
button_handle *long_button1;
button_handle *short_long_button;
button_handle *combined_long_button0;
button_handle *repeat_long_button;

short_button = create_short_button(MODE);
long_button0 = create_long_button(WECHAT, 50);
long_button1 = create_long_button(VOICE, 500);
short_long_button = create_short_long_button(NEXT, 500);
combined_long_button0 = create_combined_long_button(NEXT | PREV, 50);
repeat_long_button = create_repeat_long_button(PREV, 500, 300);
```

4.1.4 设置按键回调函数

```
short_button->cb = short_button_cb;
long_button0->cb = long_button0_cb;
long_button1->cb = long_button1_cb;
short_long_button->cb = short_long_button_cb;
combined_long_button0->cb = combined_long_button0_cb;
repeat_long_button->cb = repeat_long_button_cb;
```

按键的回调函数不是固定的，在不同的场景下，可以设置不同的回调函数，回调函数的处理逻辑或代码在不同的场景中实现，从而实现不同场景下按键操作的解耦。当按键对象被触发后，会调用回调函数并传递相应的按键状态。

4.1.5 启动按键

只有按键对象被启动后，才会相应按键的操作。

```
short_button->start(short_button);
long_button0->start(long_button0);
long_button1->start(long_button1);
short_long_button->start(short_long_button);
```

Copyright @2019 Xradio Technology Co., Ltd. All Rights Reserved

```
combined_long_button0->start(combined_long_button0);
repeat_long_button->start(repeat_long_button);
```

4.1.6 停止按键

如果在某些场景下需要停止某些按键的操作，可停止按键，比如在 ota 升级场景下，要停止所有按键的操作。

```
short_button->stop(short_button);
long_button0->stop(long_button0);
long_button1->stop(long_button1);
short_long_button->stop(short_long_button);
combined_long_button0->stop(combined_long_button0);
repeat_long_button->stop(repeat_long_button);
```

4.1.7 销毁按键

在某些场景下，可能需要临时创建并销毁按键，比如在开机场景下，可临时创建一个厂测按键，多个特定的按键同时按下即进入自动测试场景，自动测试完成或者退出开机场景时，可销毁临时创建的按键。

```
short_button->destroy(short_button);
long_button0->destroy(long_button0);
long_button1->destroy(long_button1);
short_long_button->destroy(short_long_button);
combined_long_button0->destroy(combined_long_button0);
repeat_long_button->destroy(repeat_long_button);
```

Copyright @2019 Xradio Technology Co., Ltd. All Rights Reserved

5 模块测试命令

按键模块不支持命令操作

Copyright @2019 Xradio Technology Co., Ltd. All Rights Reserved

6 模块常见问题

（1）在按键初始化后，按键无响应

可能原因：

1. 硬件问题

2. 按键配置信息未配置对

3. 其他代码占用了或配置了按键的引脚

解决方法：

1. 测量 AD 按键按下后，AD 引脚的电压变化；测量 GPIO 按键按下后，GPIO 引脚的电平变化。可排除硬件问题。

2. 检查按键配置信息

3. 检查其他代码或其他模块是否占用了或重新配置了按键的引脚。

（2）同一通道的 AD 按键有些能用，有些不能用；或者有时能用，有时不能用；或者会出现错误检测按键的现象

可能原因：

1. 硬件中，AD 按键分压电阻精确度不高，或温飘太大。

2. 硬件中，各 AD 按键的 AD 值比价接近，导致容易出现按键误判现象。

解决方法：

1. 不同环境下，多次测量 AD 按键按下后的 AD 值，计算平均值。

2. 调整 project\common\apps\buttons\buttons_low_level.h 中的宏 AD_FLUCTUATION 到合适的值，该宏为按键 AD 值的波动区间，若某按键 AD 值的平均值为 1000，则该按键 AD 值的波动区为 1000±AD_FLUCTUATION。当出现有时能用，有时不能用时，说明区间太小，可增加该值；当出现错误检测按键现象时，说明区间太大，导致不同按键区间有重合，可减小该值。

Copyright @2019 Xradio Technology Co., Ltd. All Rights Reserved