

青风带你玩蓝牙 nRF52832 系列教程.....	2
-----作者: 青风.....	2
作者: 青风.....	3
出品论坛: www.qfv8.com	3
淘宝店: http://qfv5.taobao.com	3
QQ 技术群: 346518370.....	3
硬件平台: 青云 QY-nRF52832 开发板.....	3
2 蓝牙协议栈下按键的使用.....	3
1: 按键下控制触发 LED 灯.....	3
1.1 协议栈下按键初始化调用函数:	3
1.2 应用与调试.....	8
2: 按键下的长按和短按.....	8

青风带你玩蓝牙 nRF52832 系列教程

-----作者: 青风

出品论坛: www.qfv8.com 青风电子社区



作者: 青风

出品论坛: www.qfv8.com

淘宝店: <http://qfv5.taobao.com>

QQ 技术群: 346518370

硬件平台: 青云 QY-nRF52832 开发板

2 蓝牙协议栈下按键的使用

本节我们讲主要探讨一下蓝牙协议栈下按键模块的使用,在第一讲样例提到过按键初始化这个部分,这里面我详细的分析一下按键模块的注册和按键事件发生后如何回调自己注册的回调。

同时这里我们通过一个简单的例子: 蓝牙蓝牙样例,来进行一个简单的思路验证如何使用按键。注意本例在蓝牙样例的基础上进行修改。

1: 按键下控制触发 LED 灯

1.1 协议栈下按键初始化调用函数:

按键和 LED 的初始化函数如下所示,协议栈下实现了一个 BSP(板级支持包)的,主要是整合了按键模块的功能和一些“板级指令”以及与 ble 事件的交互。而实现这个功能的主要就是下面红色标注的两个函数 **bsp_init** 和 **bsp_btn_ble_init**:

```
static void buttons_leds_init(bool * p_erase_bonds)
{
    bsp_event_t startup_event;

    uint32_t err_code = bsp_init(BSP_INIT_LED | BSP_INIT_BUTTONS,
                                  APP_TIMER_TICKS(1000, APP_TIMER_PRESCALER),
                                  bsp_event_handler); //初始化外部设备
    APP_ERROR_CHECK(err_code);

    err_code = bsp_btn_ble_init(NULL, &startup_event); //初始化按键触
```

发模式

```

APP_ERROR_CHECK(err_code);

*p_erase_bonds = (startup_event == BSP_EVENT_CLEAR_BONDING_DATA);
}

```

bsp_init 函数就为按键和 LED 外设的初始化，同时触发对应的板级任务。函数可以直接查找对应函数解释：

```

uint32_t bsp_init ( uint32_t          type,
                    uint32_t          ticks_per_100ms,
                    bsp_event_callback_t callback
                    )

```

初始化板级设备 BSP.

初始化板级支持包，实现按键反应和状态指示。同时按键反应后分配给回调事件。

注意：在你使用这个功能前，你必须启用如下的模块：

```

Application Timer for LED support
GPIOE Handler   for button support
UART module     for UART support

```

参数：

[in] type	使用设备的类型（LED 和按键）
[in] ticks_per_100ms	Number of RTC ticks for 100 ms.
[in] callback	按键按下后发生对应事件后的回调函数

Return values

NRF_SUCCESS	板级设备初始化成功
NRF_ERROR_INVALID_STATE	时间未被初始化
NRF_ERROR_NO_MEM	如果达到最大时间限制
NRF_ERROR_INVALID_PARAM	如果 GPIOE 有多个使用者.
NRF_ERROR_INVALID_STATE	如果按键没有被初始化。

这个函数的功能就是初始化板级设备，包括 LED 和按键的初始化。同时定义按键按下后触发的事件的回调函数。回调函数就是在触发了事件后执行的操作。

事件指的是一些定义的特殊状态，程序中给了对应的结构体：

```
enum bsp_event_t {  
  
    BSP_EVENT_NOTHING    = 0,  
  
    BSP_EVENT_DEFAULT,  
  
    BSP_EVENT_CLEAR_BONDING_DATA,  
  
    BSP_EVENT_CLEAR_ALERT,  
  
    BSP_EVENT_DISCONNECT,  
  
    BSP_EVENT_ADVERTISING_START,  
  
    BSP_EVENT_ADVERTISING_STOP,  
  
    BSP_EVENT_WHITELIST_OFF,  
  
    BSP_EVENT_BOND,  
  
    BSP_EVENT_RESET,  
  
    BSP_EVENT_SLEEP,  
  
    BSP_EVENT_WAKEUP,  
  
    BSP_EVENT_DFU,  
  
    BSP_EVENT_KEY_0,  
  
    BSP_EVENT_KEY_1,  
  
    BSP_EVENT_KEY_2,  
  
    .....  
  
    .....  
  
    .....  
  
}
```

那么回调函数内执行的内容,大家可以根据自己的要求进行设置,比如样例里,当方式了 BSP_EVEBNT_SLEEP 事件后,系统会进入休眠模式,执行 sleep_mode_enter().当然我们也可以设置自己的事件,比如我们在最后添加一个 BSP_EVENT_KEY3 事件后,执行开灯操作:

```
651 static void bsp_event_handler(bsp_event_t event)
652 {
653     ret_code_t err_code;
654     switch (event)
655     {
656     case BSP_EVENT_SLEEP:
657         sleep_mode_enter();
658         break; // BSP_EVENT_SLEEP
659
660     case BSP_EVENT_DISCONNECT:
661         err_code = sd_ble_gap_disconnect(m_conn_handle,
662                                         BLE_HCI_REMOTE_USER_TERMINATED_CONNECTION);
663         if (err_code != NRF_ERROR_INVALID_STATE)
664         {
665             APP_ERROR_CHECK(err_code);
666         }
667         break; // BSP_EVENT_DISCONNECT
668
669     case BSP_EVENT_WHITELIST_OFF:
670         if (m_conn_handle == BLE_CONN_HANDLE_INVALID)
671         {
672             err_code = ble_advertising_restart_without_whitelist(&m_advertising);
673             if (err_code != NRF_ERROR_INVALID_STATE)
674             {
675                 APP_ERROR_CHECK(err_code);
676             }
677         }
678         break; // BSP_EVENT_KEY_0
679
680     case BSP_EVENT_KEY_3:
681         nrf_gpio_pin_toggle(LED_2);
682         break;
683     default:
684         break;
685     }
686 }
```

那么如何触发这些定义的事件了?这里面有一个关键函数:

`uint32_t bsp_event_to_button_action_assign`

`(uint32_t button, bsp_button_action_t action, bsp_event_t event)`

该函数有 3 个参数:

第一个参数是 button id。通常与板子板子相关。比如我用的官方板 pca10028 有四个按键那么使用的 id 就是 0-3,

第二个参数是触发动作如 APP_BUTTON_PUSH, APP_BUTTON_RELEASE 分别表示注册的按键事件是按键按下和按键释放。

第三个参数是第二个参数指明的触发条件被触发后产生的 BSP 事件。第三个参数可能有点不容易理解。BSP 模块中实现的按键功能在注册的按键事件触发后,不管是哪个按键,执行的处理函数都是一样的,处理过程就是根据按键按下或释放将上面第三个参数,也就是 bsp 事件,传递给最终的 bsp 事件处理函数。而这个最终的 bsp 事件处理函数就是 bsp_init 函数中的第三个参数注册的回调函数的事件。

`bsp_event_to_button_action_assign` 函数在使能按键模式，是按键和 BLE 事件交互的函数 `bsp_btn_ble_init` 中被调用,首先看下 `bsp_btn_ble_init` 函数介绍:

```
uint32_t bsp_btn_ble_init ( bsp_btn_ble_error_handler_t error_handler,  
                           bsp_event_t *                  p_startup_bsp_evt  
                           )
```

初始化蓝牙按键模式，在使用这个功能之前，板级必须要初始化按键。

参数:

- | | |
|--------------------------------|--|
| [out] error_handler | 按键模式出错下返回错误码. |
| [out] p_startup_bsp_evt | 启动时，当按键被按下，这个值为触发 BLE 事件(or <code>BSP_EVENT_NOTHING</code>)。比如,如果绑定接触唤醒模式按键被按下，用于唤醒设备，指针 <code>*p_startup_bsp_evt</code> 被设置为 <code>BSP_EVENT_CLEAR_BONDING_DATA</code> . |

Return values

NRF_SUCCESS 初始化成功

这个函数中调用了一个配置函数,广播按键配置函数，如下图所示:

```
uint32_t bsp_btn_ble_init(bsp_btn_ble_error_handler_t error_handler, bsp_evt_t * p_startup_bsp_evt)  
{  
    uint32_t err_code = NRF_SUCCESS;  
    m_error_handler = error_handler;  
    if (p_startup_bsp_evt != NULL)  
    {  
        err_code = startup_event_extract(p_startup_bsp_evt);  
        RETURN_ON_ERROR(err_code);  
    }  
    if (m_num_connections == 0)  
    {  
        err_code = advertising_buttons_configure();  
    }  
    return err_code;  
}
```

这个函数的内部就定义了 `bsp_event_to_button_action_assign` 函数，也就是按键触发分配函数，分配定义的触发任务:


```
static uint32_t advertising_buttons_configure()
{
    uint32_t err_code;

    err_code = bsp_event_to_button_action_assign(BTN_ID_DISCONNECT,
                                                  BTN_ACTION_DISCONNECT,
                                                  BSP_EVENT_DEFAULT);
    RETURN_ON_ERROR_NOT_INVALID_PARAM(err_code);

    err_code = bsp_event_to_button_action_assign(BTN_ID_WHITELIST_OFF,
                                                  BTN_ACTION_WHITELIST_OFF,
                                                  BSP_EVENT_WHITELIST_OFF);
    RETURN_ON_ERROR_NOT_INVALID_PARAM(err_code);

    err_code = bsp_event_to_button_action_assign(BTN_ID_SLEEP,
                                                  BTN_ACTION_SLEEP,
                                                  BSP_EVENT_SLEEP);
    RETURN_ON_ERROR_NOT_INVALID_PARAM(err_code);

    return NRF_SUCCESS;
}
```

我们可以在里面添加我们自己的触发事件，由于按键 4 没有使用，按键标号对应 3，同时事件 BSP_EVENT_KEY_3 也没用到，可以加入代码如下：

```
err_code=bsp_event_to_button_action_assign(3, BSP_BUTTON_ACTION_PUSH, BSP_EV
ENT_KEY_3);
```

```
RETURN_ON_ERROR_NOT_INVALID_PARAM(err_code);
```

这样就可以触发 BSP_EVENT_KEY_3 事件，在按键按下按键 4 的时候，执行 bsp_event_handler 中点灯的操作。

1.2 应用与调试

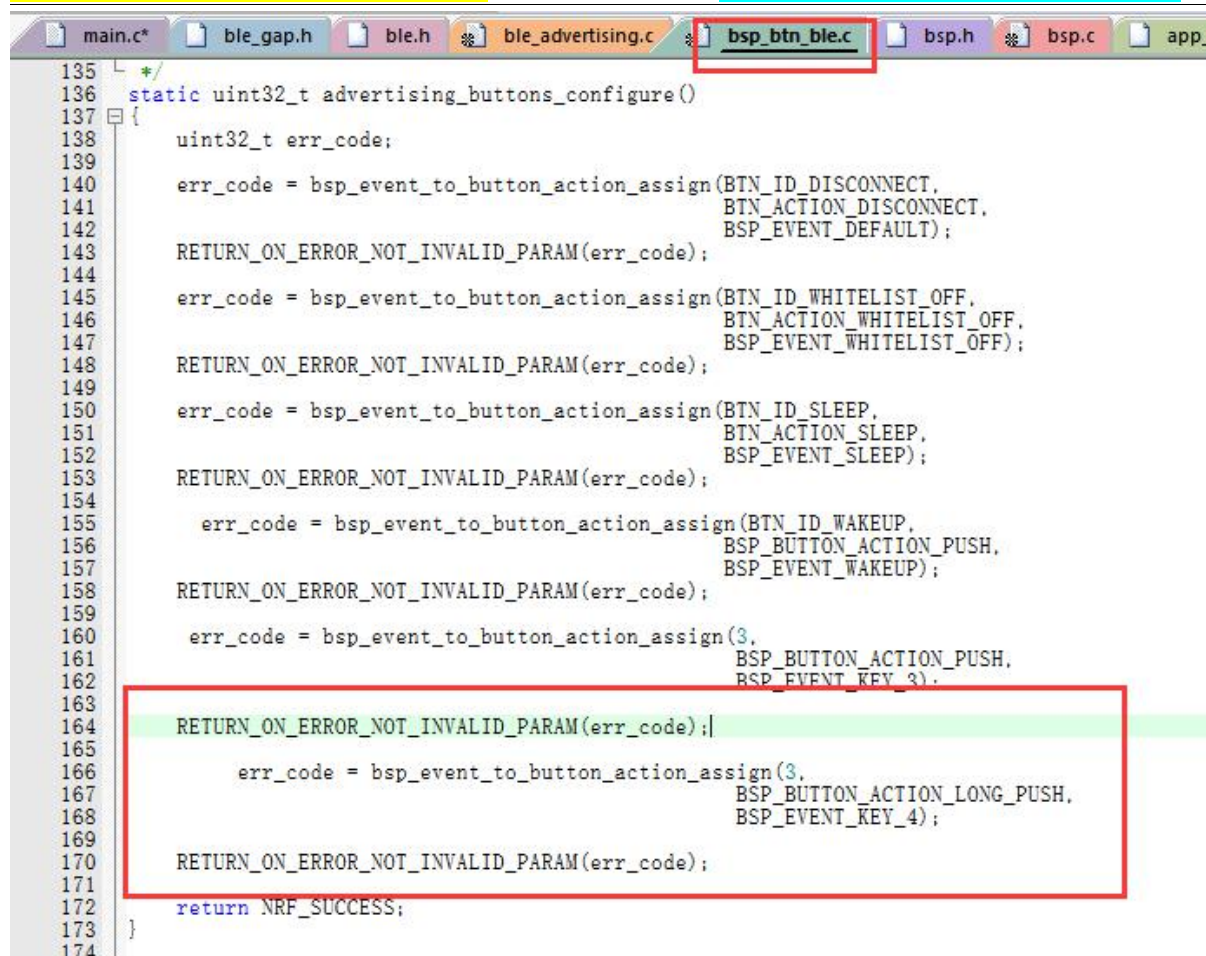
本例使用的协议栈为 S132 版本，NRFgo 进行下载，可以参考软件篇介绍。首先整片擦除，后下载协议栈。然后使用 KEIL 下载程序

LED1 广播灯闪烁，按下按键 4 后，LED2 会翻转，不管是在蓝牙连接状态下还是广播状态下，按键都可以触发 LED2 灯的翻转。

2：按键下的长按和短按

我们可以在里面添加我们自己的长接触发事件，还是使用按键 4，按键标号还是对应 3，同时事件 BSP_EVENT_KEY_4 触发事件也没用到，可以加入代码如下：

```
err_code =
bsp_event_to_button_action_assign(3, BSP_BUTTON_ACTION_LONG_PUSH,
                                  BSP_EVENT_KEY_4);
RETURN_ON_ERROR_NOT_INVALID_PARAM(err_code);
```

```
135 /*
136 static uint32_t advertising_buttons_configure()
137 {
138     uint32_t err_code;
139
140     err_code = bsp_event_to_button_action_assign(BTN_ID_DISCONNECT,
141     BTN_ACTION_DISCONNECT,
142     BSP_EVENT_DEFAULT);
143     RETURN_ON_ERROR_NOT_INVALID_PARAM(err_code);
144
145     err_code = bsp_event_to_button_action_assign(BTN_ID_WHITELIST_OFF,
146     BTN_ACTION_WHITELIST_OFF,
147     BSP_EVENT_WHITELIST_OFF);
148     RETURN_ON_ERROR_NOT_INVALID_PARAM(err_code);
149
150     err_code = bsp_event_to_button_action_assign(BTN_ID_SLEEP,
151     BTN_ACTION_SLEEP,
152     BSP_EVENT_SLEEP);
153     RETURN_ON_ERROR_NOT_INVALID_PARAM(err_code);
154
155     err_code = bsp_event_to_button_action_assign(BTN_ID_WAKEUP,
156     BSP_BUTTON_ACTION_PUSH,
157     BSP_EVENT_WAKEUP);
158     RETURN_ON_ERROR_NOT_INVALID_PARAM(err_code);
159
160     err_code = bsp_event_to_button_action_assign(3,
161     BSP_BUTTON_ACTION_PUSH,
162     BSP_EVENT_KEY_3);
163
164     RETURN_ON_ERROR_NOT_INVALID_PARAM(err_code);
165
166     err_code = bsp_event_to_button_action_assign(3,
167     BSP_BUTTON_ACTION_LONG_PUSH,
168     BSP_EVENT_KEY_4);
169
170     RETURN_ON_ERROR_NOT_INVALID_PARAM(err_code);
171
172     return NRF_SUCCESS;
173 }
174
```

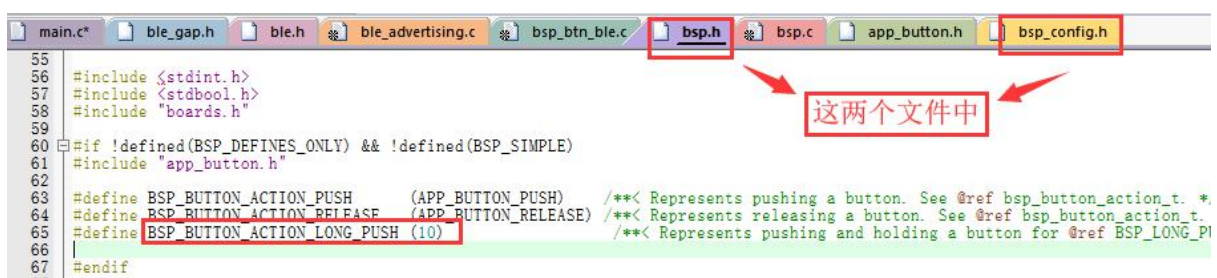
BSP_EVENT_KEY_4 触发事件在按键的回调函数内执行的对应内容，我们设置长按的时候 LED3 和 LED2 翻转，在主函数里的按键回调处理函数 bsp_event_handler 里配置加入：

```

651 static void bsp_event_handler(bsp_event_t event)
652 {
653     ret_code_t err_code;
654
655     switch (event)
656     {
657         case BSP_EVENT_SLEEP:
658             sleep_mode_enter();
659             break; // BSP_EVENT_SLEEP
660
661         case BSP_EVENT_DISCONNECT:
662             err_code = sd_ble_gap_disconnect(m_conn_handle,
663                                             BLE_HCI_REMOTE_USER_TERMINATED_CONNEC
664             if (err_code != NRF_ERROR_INVALID_STATE)
665             {
666                 APP_ERROR_CHECK(err_code);
667             }
668             break; // BSP_EVENT_DISCONNECT
669
670         case BSP_EVENT_WHITELIST_OFF:
671             if (m_conn_handle == BLE_CONN_HANDLE_INVALID)
672             {
673                 err_code = ble_advertising_restart_without_whitelist(&m_advertisir
674                 if (err_code != NRF_ERROR_INVALID_STATE)
675                 {
676                     APP_ERROR_CHECK(err_code);
677                 }
678             }
679             break; // BSP_EVENT_KEY_0
680         case BSP_EVENT_KEY_3:|
681             nrf_gpio_pin_toggle(LED_2);
682             break;
683         case BSP_EVENT_KEY_4:
684
685             nrf_gpio_pin_toggle(LED_3);
686             nrf_gpio_pin_toggle(LED_2);
687             break;
688         default:
689             break;
690     }
691 }

```

按键的时间长短可以通过配置 bsp.h 文件和 bsp.config.h 文件里的 BSP_BUTTON_ACTION_LONG_PUSH 的大小决定, 如果设置为 10 表示 10ms



本例使用的协议栈为 S132 版本, NRFgo 进行下载, 可以参考软件篇介绍。首先整片擦除, 后下载协议栈。然后使用 KEIL 下载程序。LED1 广播灯闪烁。

短按按键 4 后, LED2 会翻转, 不管是在蓝牙连接状态下还是广播状态下, 按键都可以触发 LED2 灯的翻转。长按按键 4 后, 会发生 LED2 和 LED3 同时翻转的现象。