一、前言

我们在使用BLE芯片时，有的时候我们需要把一些传感器或者标志位等数据存到BLE芯片内部的flash，以便掉电重新上电我们可以利用这些数据来实现一些自定义的功能，现在我就以Nordic 52832的BLE芯片为例，谈谈我在使用这个功能上遇到的问题和心得。

二、使用步骤

1、初始化一个pstorage模块

直接调用以下这个API函数即可初始化一个pstorage模块

```
1.      err_code = pstorage_init();                                    //初始化一个pstora
    ge模块，必须在调用相关pstorage模块的API函数前先调用该初始化函数
2.      RTT_PRINTF("[pstorage_init]-->%d\r\n",err_code);
```

2、注册分配一个自定义大小的空间

```
1.          uint32_t err_code;
2.          pstorage_module_param_t module_param;                          //保
    存申请的block的大小和数据
3.
4.          module_param.block_count = BLOCK_COUNTS;
5.          module_param.block_size  = BLOCK_SIZE;
6.          module_param.cb                  = flash_wr_cb;
7.
8.
9.          err_code = pstorage_init();                                    //注
    册一个pstorage模块
10.         RTT_PRINTF("[pstorage_init]-->%d\r\n",err_code);
11.
12.
13.         err_code = pstorage_register(&module_param,&m_block_id);       //申
    请了3个大小为16字节的Block,最小的空间单元是16字节
14.         RTT_PRINTF("[m_block_id.block_id]-->0x%08X\r\n",m_block_id.block_id);
15.         RTT_PRINTF("[m_block_id.module_id]-->%d\r\n",m_block_id.module_id);
16.
17.         pstorage_block_identifier_get(&m_block_id,BLOCK1,&led1_dest_block_id);  //
    获取存放led1状态的block空间首地址
18.         RTT_PRINTF("[led1_dest_block_id.block_id]-->0x%08X\r\n",led1_dest_block_id
    .block_id);
19.         RTT_PRINTF("[led1_dest_block_id.module_id]-->%d\r\n",led1_dest_block_id.mo
    dule_id);
```

```
20.
21.         pstorage_block_identifier_get(&m_block_id,BLOCK2,&led2_dest_block_id);  //获取存放led2状态的block空间首地址
22.         RTT_PRINTF("[led2_dest_block_id.block_id]-->0x%08X\r\n",led2_dest_block_id.block_id);
23.         RTT_PRINTF("[led2_dest_block_id.module_id]-->%d\r\n",led2_dest_block_id.module_id);
24.
25.        pstorage_block_identifier_get(&m_block_id,BLOCK3,&led3_dest_block_id);   //获取存放led3状态的block空间首地址
26.         RTT_PRINTF("[led3_dest_block_id.block_id]-->0x%08X\r\n",led3_dest_block_id.block_id);
27.         RTT_PRINTF("[led3_dest_block_id.module_id]-->%d\r\n",led3_dest_block_id.module_id);
```

3、读或者写申请到的flash空间

```
1.  err_code = pstorage_update(&led1_dest_block_id,&led1_staus,4,0);//把当前灯的开关状态写至flash中，且一定要是4字节写入，不然，初始化pstorage模块时设置的回调函数无法进入
2.  RTT_PRINTF("pstorage_update is %d\r\n",err_code);
```

```
1.  err_code = pstorage_load(&led1_dest_block_id,&led1_staus,1,0);//从flash中读取当前灯的开关状态
2.  RTT_PRINTF("pstorage_loadis %d\r\n",err_code);
```

三、注意事项

1、如果代码有使用了device_manager_init()函数，那么就要看看该函数是不是也调用了pstorage_init()，如果是的话就要注意避免调用两次，否则

```
1.  module_param.cb                 = flash_wr_cb;
```

这个回调函数就会进不去

2、保存读写申请到的flash中的变量一定要4字节对齐，否则在调用读写函数时会报错误内存地址的错误信息，这个也是我在使用这个pstorage模块功能遇到的问题，解决方法如下：

```
1.  static uint8_t          led1_staus __attribute__ ((aligned (4)));           //4字节对齐，不然调用pstorage_update或者pstorage_store会报错.
2.  static uint8_t          led2_staus __attribute__ ((aligned (4)));           //4字节对齐，不然调用pstorage_update或者pstorage_store会报错.
```

```
3.  static uint8_t          led3_staus __attribute__ ((aligned (4)));          //4字节对齐，不然调用pstorage_update或者pstorage_store会报错.
```

3、读取pstorage模块的数据需要等，写完成了读取的数据才是有效的，否则读取到的数据是上一次的无效数据

```
1.  /**@brief 将想要的数据写入pstorage模块中
2.   */
3.  static void flash_wr_cb(pstorage_handle_t  * handle,uint8_t op_code,uint32_t result,uint8_t  * p_data, uint32_t   data_len)
4.  {
5.  //  uint8_t led_staus;
6.      switch(op_code)
7.          {
8.              case PSTORAGE_UPDATE_OP_CODE://将数据写写入pstorage模块时，才触发该事件
9.                  if(result == NRF_SUCCESS)//只要当返回的结果是NRF_SUCCESS时，此时读取数据才是有效正确的数据，否则读出来的数据是无效的
10.                 {
11. //                  flash_w_flag = 1;
12.                 }
13.             break;
14.             case PSTORAGE_LOAD_OP_CODE:////从pstorage模块读取数据时，才触发该事件
15.
16.             break;
17.             default:
18.             break;
19.         }
20.
21. }
```