



FAQ_ Flash 操作与 BLE 事件的互斥

0 Keywords

Flash, 擦除、BlueNRG-1/2

1 Q&A

Q:

客户经常问：我写或者擦除 Flash 后，没有蓝牙信号了，或者是我写或者擦除 Flash 后，死机了。

A: BlueNRG-1/2 的 Flash 存储数据应用数据时，可能遇到这个问题。因为在擦除一页的过程中，中断会被关掉大概 20 多 ms,在擦除期间，radio 中断是被关掉了的，整个 radio 的状态被延时或者整个 RF 部分的状态被破坏。

ST 的 SDK 虽然提供了访问 Flash 的 Demo，但是当多个 Flash 操作时，与蓝牙事件互斥就比较难处理了，特别是对于一些应用多连接或者同时用了主从，多种蓝牙事件的场景，互斥就显得更麻烦了。为了方便用户操作 Flash，下面我提供访问 Flash 的方法供参考（前面是分析，如果想直接看操作，翻转到文档最后面）。

解决做法：使用一个双向链表来管理 Flash 操作（Flash 擦除或者是 Flash 写，Flash 读无需和 BLE 相关事件互斥），和使用一个双向链表来管理空余时间状态。在空余的时间执行 Flash 操作。

从 datasheet 可以知道，操作 Flash，特别是擦除时，会占用比较多的时间。为了避免操作 Flash 占用过多的时间片，把每次写或

者擦除 Flash，用一个列表来管理，每次写或者擦除操作，插入 Flash 操作链表的最后。在主循环中检测是否有足够的时间写，如果当前空闲的时间，如果空闲的时间足够，取出 Flash 操作链表中最前的一个节点，执行 Flash 操作，然后将节点移除。

对于如何检测空闲时刻，也用一个链表来管理空闲时刻。

首先，初始化时需要设置 radio active 事件抛到应用

```
aci_hal_set_radio_activity_mask(0x0001|0x0002|0x0004|0x0008|0x0010|0x0020);
```

```
331: }
332:
333: /* Application demo Led Init */
334: SdkEvalLedInit(LED1); //Activity led
335: SdkEvalLedOn(LED1);
336:
337: //device initialization
338: device_initialization();
339: aci_hal_set_radio_activity_mask(0x0001|0x0002|0x0004|0x0008|0x0010|0x0020);
340:
341: printf("BLE device initialized\r\n");
342:
343: while(1) {
344:     /* BlueNRG-1 stack tick */
345:     BTLE_StackTick();
346:
347:     /* Application Tick */
348: }
```

设置之后，重写函数 `aci_hal_end_of_radio_activity_event`，当有事件触发时，就会进入这个函数。

```
1290 void aci_gap_pairing_complete_event(uint16_t Connection_Handle,
1291                                     uint8_t Status,
1292                                     uint8_t Reason)
1293 {
1294     printf("Xs Reason:0x Connection_Handle:0x\n", __func__, Reason, Connection_Handle);
1295 }
1296
1297 void aci_hal_end_of_radio_activity_event(uint8_t Last_State,
1298                                         uint8_t Next_State,
1299                                         uint32_t Next_State_SystemTime)
1300 {
1301     //uint32_t cur_time = HAL_VTimerGetCurrentTime_sysT32();
1302     //printf("Last_State:0x Next_State:0x Next_State_SystemTime:0x ", Last_State, Next_State, Next_State_SystemTime);
1303     //printf("Time gap: %d\n", (Next_State_SystemTime-cur_time)*2.441d/1000 );
1304     free_list_update(Last_State, Next_State, Next_State_SystemTime);
1305 }
1306
1307 /** \endcond
1308 */
```

每次进入这个函数，生成一个空闲时刻的节点插入到链表中。

主循环中有个 Flash 操作的调度函数，不停的调度是否有数据需要写入。

```

gh 4.0 - [master_slave.c (D:\work\...\src)]
ptions Tools View Window Help

702: * Function Name : app_tick.
703: * Description : Device Demo state machine.
704: * Input : None.
705: * Output : None.
706: * Return : None.
707: *****
708: void APP_Tick(void)
709: {
710:     uint8_t status;
711:
712:     /* User has to add his code */
713:
714:     /* Check if Slaves discovery procedure has to be started */
715:     if ((disc_to_be_checked == TRUE) && (disc_procedure_is_ongoing == FALSE) && )
716:     {
717:         /* do connection with discovered slaves devices */
718:         if (APP_FLAG(DO_SLAVES_CONNECTION)) ...
719:         else if APP_FLAG(GET_NOTIFICATION) ...
720:         { ...
721:
722:         write_characteristics_tick();
723:         flash_operate_tick();
724:     }
725:     /* end APP_Tick */ /* end APP_Tick() */
726:     /* ***** BlueNRG-1 Stack Callbacks***** */
727: }

```

在 Flash 操作的调度函数中，主要做两件事情，一个是把空闲时间计算出来给到底层的 Flash 操作链表调度中。另一个是移除过期的节点

```

[ble_free_list.c (D:\work\...\common)]
Tools View Window Help

43: void flash_operate_tick(void)
44: {
45:     uint32_t cur_tick = 0;
46:     uint32_t next_state_sys_time = 0;
47:
48:     cur_tick = HAL_VTimerGetCurrentTime_sysT32();
49:     // check whether the node is expire ?
50:     struct ble_free *item = NULL;
51:     uint32_t gap_ticks = 0;
52:     item = dl_list_last(&q_free_list, struct ble_free, node);
53:     if(item != NULL){
54:         next_state_sys_time = item->next_state_sysTime;
55:         if(cur_tick < next_state_sys_time){
56:             // Time overflow
57:             if(abs(cur_tick - next_state_sys_time) > (~0u / 2)){
58:                 gap_ticks = ~0u - next_state_sys_time + cur_tick;
59:             }
60:             else{
61:                 gap_ticks = next_state_sys_time - cur_tick;
62:             }
63:             // printf("gap_ticks:%.2f", gap_ticks*2.4414/1000);
64:             flash_list_tick(gap_ticks);
65:         }
66:         else{
67:             // node have expired , remove it
68:             dl_list_del(&item->node);
69:             dl_list_add(&q_free_unuse, &item->node);
70:             // Recursion call it.
71:             flash_operate_tick();
72:             return;
73:         }
74:     }
75:     /* end if item!=NULL */
76:     else{
77:     }
78: }
79: /* end flash_operate_tick */

```

当前空余时间，抛送到Flash操作链表调度中，用于决定是否执行Flash操作

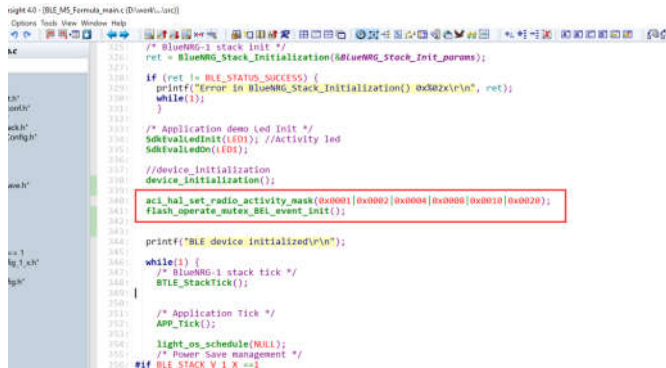
当节点过期了，直接移除，回收到空闲链表中。

flash_list_tick 函数主要是从 Flash 任务列表中去取节点，当空余时间满足时，则执行相应的 Flash 操作，然后将节点从任务列表中移除，回收到空闲列表中。

2 方法

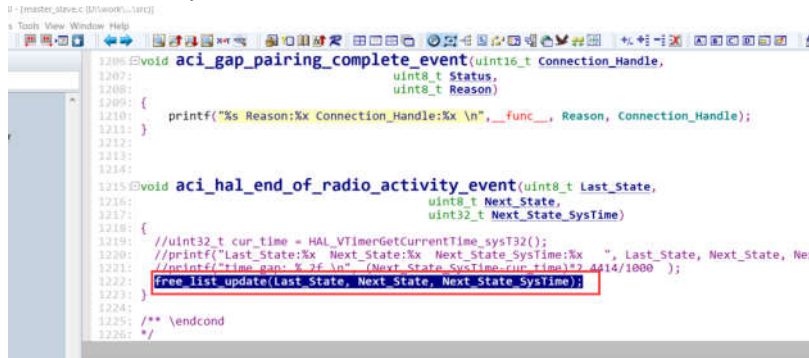
添加到其他工程的方法：

1. 初始化相关链表和初始化 radio 事件回调屏蔽



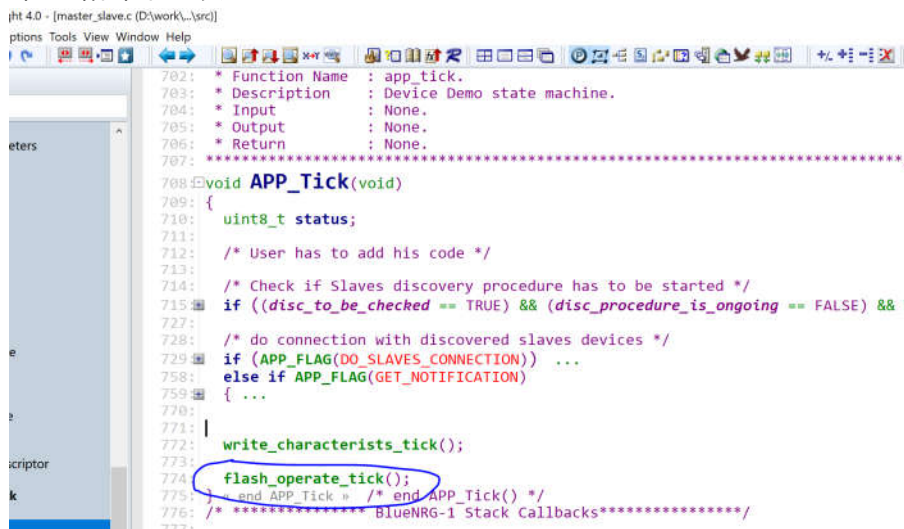
```
1221 /* BlueNRG-1 stack init */
1222 ret = BlueNRG_Stack_Init(BlueNRG_Stack_Init_Params);
1223 if (ret != BLT_STATUS_SUCCESS) {
1224     printf("Error in BlueNRG_Stack_Init: 0x%02x\n", ret);
1225     while(1);
1226 }
1227 /* Application demo led init */
1228 SdkValuedInit(100); //Activity led
1229 SdkValuedInit(100);
1230 //device initialization
1231 device_initialization();
1232 aci_hal_set_radio_activity_mask(0x0000|0x0000|0x0000|0x0010|0x0020);
1233 flash_operate_mutex_BEL_event_init();
1234
1235 printf("BLE device initialized\n");
1236 while(1) {
1237     /* BlueNRG-1 stack tick */
1238     BLT_StackTick();
1239 }
1240 /* Application Tick */
1241 APP_Tick();
1242 light_os_schedule(NULL);
1243 /* Power Save management */
1244 #if BLE_STACK_V1_X == 1
1245
```

2. 在 radio activity even 中添加 空闲链表的更新函数



```
1206 void aci_gap_pairing_complete_event(uint16_t Connection_Handle,
1207                                     uint8_t Status,
1208                                     uint8_t Reason)
1209 {
1210     printf("Reason:0x%02x Connection_Handle:0x%04x\n", __func__, Reason, Connection_Handle);
1211 }
1212
1213 void aci_hal_end_of_radio_activity_event(uint8_t Last_State,
1214                                         uint8_t Next_State,
1215                                         uint32_t Next_State_SystemTime)
1216 {
1217     //uint32_t cur_time = HAL_VTimerGetCurrentTime_sysT32();
1218     //printf("Last_State:0x%02x Next_State:0x%02x Next_State_SystemTime:0x%08x", Last_State, Next_State, Next_State_SystemTime);
1219     //printf("time_gap: %d\n", (Next_State_SystemTime - cur_time) * 2.4414 / 1000);
1220     free_list_update(Last_State, Next_State, Next_State_SystemTime);
1221 }
1222
1223 /** endcond
1224 */
1225
```

3. 在主循环中调度



```
702 /* Function Name : app tick.
703 /* Description : Device Demo state machine.
704 /* Input : None.
705 /* Output : None.
706 /* Return : None.
707 *****
708 void APP_Tick(void)
709 {
710     uint8_t status;
711     /* User has to add his code */
712     /* Check if Slaves discovery procedure has to be started */
713     if ((disc_to_be_checked == TRUE) && (disc_procedure_is_ongoing == FALSE) && !
714         if (APP_FLAG(DO_SLAVES_CONNECTION)) ...
715     else if APP_FLAG(GET_NOTIFICATION)
716     { ...
717 }
718 write_characteristics_tick();
719 flash_operate_tick();
720 } end APP_Tick /* end APP_Tick */
721 /* ***** BlueNRG-1 Stack Callbacks*****
722
```

附件上其他文件源码。



4. 应用操作 Flash 调用接口

```
19:
20:
21:
22:
23: void flash_list_init(void);
24: |
25: /*
26:  erase flash page operate api
27: @input
28:  pageNum: page number of wanting to erase
29:  cb:      callback of when finish erase
30:  param:   param of when callback
31: */
32: bool flash_list_erase_page(uint16_t pageNum, FLASH_DONE_CALLBACK cb, void *param);
33:
34: /*
35:  write flash page operate api
36: @input
37:  Address: address for write to flash
38:  Data:    Data for write to flash
39:  length:  length for write to flash
40:  cb:      callback of when finish erase
41:  param:   param of when callback
42: */
43: bool flash_list_program_write(uint32_t Address, uint32_t* Data, uint16_t length,
44:                               FLASH_DONE_CALLBACK cb, void *param);
45:
46: int flash_list_empty(void);
47:
48:
49: void flash_list_tick(uint32_t *addr, tick);
```

备注: 如果使用此方法中的 Flash 擦除, 没有任何 BLE 事件时, 直接只调用此方法不会执行任何操作 Flash 的操作。所以当没有蓝牙事件时, 可以使用 SDK 默认的 Flash 访问接口或者是手工调度函数类似这样:

```
flash_list_tick (~0u) ;
```

应用操作 Flash 使用示例代码 1:


```

273:
274: #include "flash_list.h"
275:
276:
277:
278: #define BLE_TEST_PAGE_ADDRESS          0x10066800
279:
280: uint8_t data_to_write[] = {0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,};
281:
282:
283:
284: void *write_done(void*param)
285: {
286:     printf("%s\n", __func__);
287:     return NULL;
288: }
289:
290: void *erase_done(void*param)
291: {
292:     printf("%s\n", __func__);
293:     return NULL;
294: }
295:
296: void test_flash_list1(void)
297: {
298:     uint16_t page = (BLE_TEST_PAGE_ADDRESS - MEMORY_FLASH_BEGIN_) / _MEMORY_BYTES_PER_PAGE_;
299:     flash_list_erase_page(page, erase_done, NULL);
300:     flash_list_program_write(BLE_TEST_PAGE_ADDRESS, data_to_write, sizeof(data_to_write), write_done, NULL);
301: }
302:
303:
304:
305:

```

参考使用接口示例代码 2:

```

274: #include "flash_list.h"
275:
276:
277:
278: #define BLE_TEST_PAGE_ADDRESS          0x10066800
279:
280: uint8_t data_to_write[] = {0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,0x12,};
281:
282:
283:
284: static void *write_done(void*param)
285: {
286:     printf("%s\n", __func__);
287:     return NULL;
288: }
289:
290: static void *erase_done(void*param)
291: {
292:     printf("%s\n", __func__);
293:     flash_list_program_write(BLE_TEST_PAGE_ADDRESS, data_to_write, ARRAY_LEN(data_to_write), write_done, NULL);
294:     return NULL;
295: }
296:
297: void test_flash_list2(void)
298: {
299:     uint16_t page = (BLE_TEST_PAGE_ADDRESS - MEMORY_FLASH_BEGIN_) / _MEMORY_BYTES_PER_PAGE_;
300:     flash_list_erase_page(page, erase_done, NULL);
301: }
302:
303:
304:
305:
306:

```

如果使用此文档有问题，可以发送邮件到 lucien.kuang@st.com

3 Revision history

Date	Author	Type	Device	Version	Changes
05-09-2019	Lucien KUANG	Firmware	BlueNRG- 1/2	1.0	Initial release