

目录

1. 产品配网常见的问题	1
(1) 设备配网上线后, H5 页面显示的属性数值异常	1
(2) 设备配网上线后, H5 页面显示的属性数值与设备实际状态不同步	1
(3) 智慧生活 APP 添加设备时扫描不到设备, 但可以扫到模组发出的蓝牙广播 ..	1
(4) 连接设备时注册失败, 提示该设备未经过华为官方认证, 该设备 MAC 非法 ..	2
(5) 设备添加注册成功, 点击设备进入 H5 页面显示网络异常没有控制界面	3
2. SDK API 接口与使用问题	4
(1) PWM 相关接口的使用	4
(2) GPIO 模式配置及常见问题	4
(3) 重置配网接口 HILINK_BT_HardRevoke()使用注意事项	5
3. 蓝牙、星闪相关问题	6
(1) 使用蓝牙注册 gatt 客户端时, 接口返回 80006006 异常码	6
(2) 启用蓝牙靠近发现是在什么地方, 如何修改靠近发现的距离?	6
(3) 不同品类设备我们的蓝牙名称有要求, 这些字段应该在哪改?	7
(4) 对蓝牙广播的时间有要求, 在哪里进行修改?	7
(5) 使用指令 AT+CONFIG 配置完成时, APP 搜索不到设备, 并且也扫描不到蓝牙广播	7
(6) 蓝牙使用自定义 PIN 码功能, 设置的通行码未生效, 只能弹出默认 PIN 码 ..	7
(7) 在低功耗状态下, 模组做星闪/蓝牙从机时功耗偏高, 需要怎么解决	8
(8) 如何实时获取蓝牙的连接状态?	8
(9) 使用星闪连接时, MTU 大小有限制吗? 为什么设置时只允许 251 字节? ..	9
(10) 使用星闪保持长连接时, 进入低功耗状态, 在对应回调函数中增加连接间隔的数值来降低功耗, 功耗反正会增加, 是什么原因?	10
(11) 蓝牙在设置发包参数时返回异常码 0x80006007, 是什么原因?	10
4. 环境搭建和编译常见问题	11
(1) 编译报错编解码解析失败	11

1. 产品配网常见的问题

(1) 设备配网上线后，H5 页面显示的属性数值异常

此情况是该属性的值未进行初始设置或设置的值不在取值范围内，对比检查 BLE_SendCustomData() 函数中上报的属性值是否符合 Profile_XXXX.xlsx 文件中的数据类型和取值范围，特别注意 string 类型的数据上报时需要加“ ”

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
设备类型	设备类型(中文)	服务 sid	服务(中文)	服务类型 ServiceType	属性	属性中文名称	属性英文名称	属性类型 Character Type	操作权限	数据类型	数据约束 (IT系统录入时使用)	取值范围	描述	
眼部按摩器		bt.connect	外部音源连接	bt.connection										
		externalaud	外部音源连接	externalaud	characteristic.bt	GET/REPORT/PUT	enum	枚举-自定义枚举范围	枚举-自定义枚举范围	枚举-自定义枚举范围	枚举-自定义枚举范围	枚举-自定义枚举范围	枚举-自定义枚举范围	枚举-自定义枚举范围
		childlockSwi	童锁开关	switch	on	童锁	childLockSw	characteristic.on	GET/REPORT/PUT	bool	枚举-固定枚举范围	枚举-固定枚举范围	枚举-固定枚举范围	枚举-固定枚举范围
		gear	按摩档位	gear	gear	档位	gear	characteristic.ge	GET/REPORT/PUT	enum	枚举-自定义枚举范围	枚举-自定义枚举范围	枚举-自定义枚举范围	枚举-自定义枚举范围
		globalVolume	全局音量	gear	gear	档位	globalVolum	characteristic.ge	GET/REPORT/PUT	enum	枚举-自定义枚举范围	枚举-自定义枚举范围	枚举-自定义枚举范围	枚举-自定义枚举范围
		hotcompress	热敷温度	gear	gear	档位	hotcompress	characteristic.ge	GET/REPORT/PUT	enum	枚举-自定义枚举范围	枚举-自定义枚举范围	枚举-自定义枚举范围	枚举-自定义枚举范围
		mode	模式	mode	mode	模式	mode	characteristic.mo	GET/REPORT/PUT	enum	枚举-自定义枚举范围	枚举-自定义枚举范围	枚举-自定义枚举范围	枚举-自定义枚举范围
		switch	开关	switch	on	开关	switch	characteristic.on	GET/REPORT/PUT	bool	枚举-固定枚举范围	枚举-固定枚举范围	枚举-固定枚举范围	枚举-固定枚举范围

(2) 设备配网上线后，H5 页面显示的属性数值与设备实际状态不同步

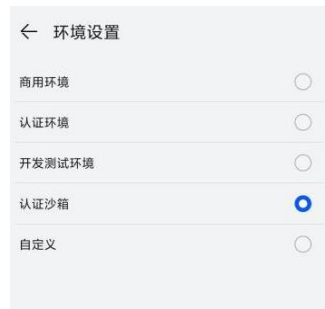
此问题是模组没有及时上报相关数据，H5 连接成功时会下发 SID allservices，模组需要在接收到这个 sid 时来主动上报全量数据同步设备状态

```
application > samples > hilink > ohos_connect > hilink_adapt > entry > C hilink_ble_main.c > [?] set_CfgNetAdvCtrl_flag
306 static int BleRcvCustomData(unsigned char *buff, unsigned int len)
324 do {
331 if (!cJSON_IsString(sidItem) || (sidItem->valuestring == NULL)) {
333 }
334
335 if (strcmp(sidItem->valuestring, "allservices") == 0) {
336 /* H5 连接上时会给设备发送allservices，设备需要给H5同步全量状态 */
337 printf("sync dev status\r\n");
338 #ifdef _HSF_GENERAL_
339 // ReporAllSidStatus();
340 send_wifi_state_fun(HILINK_SERVER_CONNECT, "SERVER_CONNECT");
341 #endif
342
343 ret = 0;
344 break;
345 } else if (strcmp(sidItem->valuestring, "currentTime") == 0) {
346 // 参考格式 {"sid": "currentTime", "data": {"currentTime": "1730692041"}}
347 cJSON *item = cJSON_GetObjectItem(dataItem, "currentTime");
348 if (item != NULL) {
349 printf("sync time %d\r\n", item->valueint);
350 }
351 ret = 0;
```

(3) 智慧生活 APP 添加设备时扫描不到设备，但可以扫到模组发出的蓝牙广播

此现象通常有以下四种状况

- a. 智慧生活 app 环境设置非认证沙箱环境
设置方法:我的-->设置-->关于-->环境设置



- b. 平台信息没同步，刷新网页和清理智慧生活 APP 的缓存之后即可解决，这种情况通常出现在平台新建的产品上。
- c. 模组内部有保存上次产品的配网信息，此次发送的蓝牙广播为不可被扫描到的广播类型，这个时候在烧录固件的时候选择 **erase all**，或者发送 **AT+RESET** 重置。
- d. 蓝牙广播名称非法，设备无法被识别，需要特别注意 空格 也属于非法字符。

(4) 连接设备时注册失败，提示该设备未经过华为官方认证，该设备 MAC 非法



此现象为模组的 MAC 未录入到华为的 DP 平台，联系相关人员进行模组 MAC 录入即可解决。

(5) 设备添加注册成功，点击设备进入 H5 页面显示网络异常没有控制界面

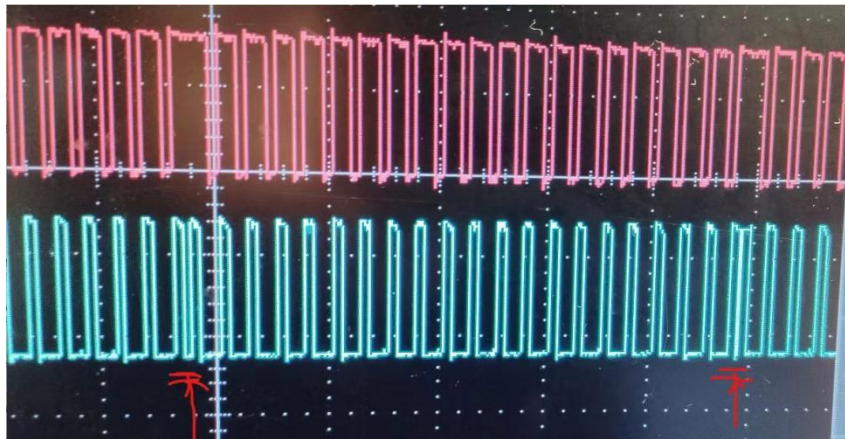


此现象常为平台数据未同步，多次刷新网页重复保存页面可以解决，清理智慧生活 APP 缓存。

2. SDK API 接口与使用问题

(1) PWM 相关接口的使用

Q: 当更新 PWM 占空比时, 就会破坏当前 PWM 周期, 现象是 LED 出现闪烁, 进行调光时有卡顿感。



A: 占空比动态调整可使用 PWM 预配置接口 `uapi_pwm_config_preload()`; 当上一个 PWM 周期完成时, 此配置会自动加载。

此功能默认关闭, 使用此接口需要添加该配置

```
(Top) → Drivers → Drivers → PWM → PWM Configuration
Configuration
*** Config PWM ***
[*] Using PWM V151
[ ] Using PWM V150
(6) number of groups in PWM
(12) PWM channel number
[ ] Using PWM PRELOAD
show-all mode enabled
[Space/Enter] Toggle/enter  [ESC] Leave menu  [S] Save
[0] Load  [?] Symbol info  [/] Jump to symbol
[F] Toggle show-help mode  [C] Toggle show-name mode  [A] Toggle show-all mode
[Q] Quit (prompts for save)  [D] Save minimal config (advanced)
```

(2) GPIO 模式配置及常见问题

Q: 复制例程中的 GPIO 引脚配置代码, 就改了引脚发现配置不生效。

A: 不同引脚的模式的复用信号对应功能不一定相同, 配置引脚模式请查看硬件用户指南。

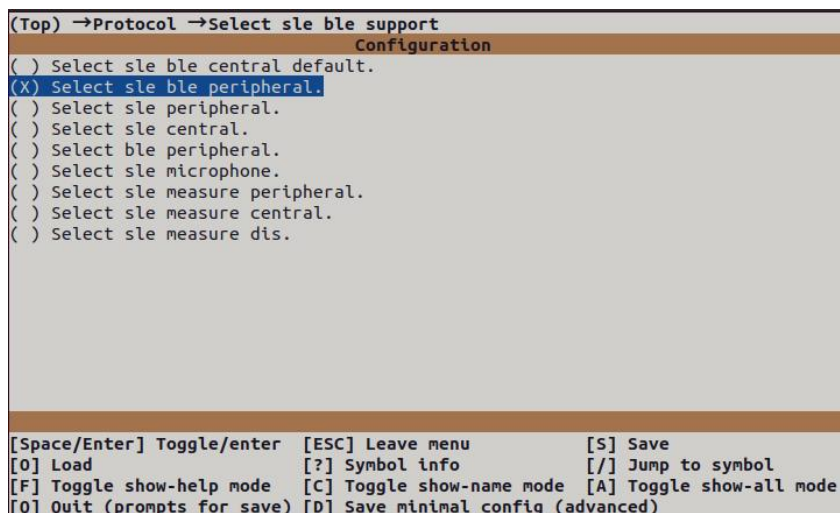
Q: 如何用软件配置 GPIO 口复位功能。

A: 参考下图 demo 配置即可, 通常选择复用于 GPIO21, 只有当软件运行后配置才会生效

3. 蓝牙、星闪相关问题

(1) 使用蓝牙注册 gatt 客户端时，接口返回 80006006 异常码

SDK 默认只开启 BLE、SLE 从机 lib(Select sle ble peripheral),如果需要使用主机功能,需要使用 python3 build.py -ninja menuconfig 指令进入图形化配置页面开启 SLE、BLE 主机 lib(Select sle ble central default)



(2) 启用蓝牙靠近发现是在什么地方，如何修改靠近发现的距离？

如图示路径下，通过 BLE_SetAdvType() 函数设置 hilink 广播类型，目前已经支持的广播类型有：拉取半模式卡片的靠近发现、蓝牙碰一碰以及常态广播三种。靠近发现的距离可通过修改广播的功率来修改，修改位置：文件 hilink_ble_main.c 中的 ADV_TX_POWER

```
application > samples > hilink > ohos_connect > hilink_adapt > entry > C hilink_ble_main.c > BleRcvCustomData(unsigned char *, unsigned int)
533 void hilink_ble_main(void)
579 #ifdef HSF_GENERAL
588 #else
589     BLE_SetAdvType(BLE_ADV_NEARBY_V0);
590 #endif
591 /* 初始化ble sdk */
592 ret = BLE_CfgNetInit(&g_bleInitParam, &g_bleCfgNetCb);
593 if (ret != 0) {
594     extern void hf_set_wakeup_pin_start_check(uint8_t state);
595     hf_set_wakeup_pin_start_check(1);
596     printf("ble sdk init fail\n");
597     return;
598 }
599
600 #ifdef CONFIG_SUPPORT_HILINK_INDIE_UPGRADE
601 if (HILINK_RegisterErrnoCallback(get_os_errno) != 0) {
```

```

65  /* BLE广播类型定义 */
66  typedef enum {
67      BLE_ADV_DEFAULT, //新蓝牙靠近发现
68      BLE_ADV_NEARBY_V0, //拉取半模态卡片的靠近发现
69      BLE_ADV_ONEHOP, //蓝牙碰一碰
70      BLE_ADV_LOCAL_NAME, //常态广播
71      BLE_ADV_CUSTOM
72  } BLE_AdvType;

```

(3) 不同品类设备我们的蓝牙名称有要求，这些字段应该在哪改？

蓝牙广播名称对应程序位置：

application/samples/hilink/ohos_connect/hilink_adapt/entry/hilink_ble_main.c



(4) 对蓝牙广播的时间有要求，在哪里进行修改？

更改 application/samples/hilink/ohos_connect/hilink_adapt/entry/hilink_ble_main.c 中 BLE_ADV_TIME 的值

```

application > samples > hilink > ohos_connect > hilink_adapt > entry > C hilink_ble_main.c > BLE_ADV_TIME
72  #else
83  #define FIRMWAREVER "1.0.5"
84  #define SOFTWAREVER "14.2.0.311"
85  #define HARDWAREVER "1.0.0"
86  /* 蓝牙sdk单独使用的定义 */
87  #define SUB_PRODUCT_ID "00"
88  #define ADV_TX_POWER 0xF8
89  #define BLE_ADV_TIME 0x900B0
90  /* 厂商自定义蓝牙广播，设备型号信息 */
91  #define BT_CUSTOM_INFO "12345678"
92  #define DEVICE_MANU_ID "01C"
93

```

(5) 使用指令 AT+CONFIG 配置完成时，APP 搜索不到设备，并且也扫描不到蓝牙广播

需要根据文档来检查参数是否有误，参数设置错误会造成蓝牙初始化失败

(6) 蓝牙使用自定义 PIN 码功能，设置的通行码未生效，只能弹出默认 PIN 码

hilink_ble_adapter.c 文件中 BleGattsSetEncryption() 回调中会关闭 PIN 码配对，需要修改对应代码来使能安全配对能力

```

application > samples > hilink > ohos_connect > hilink_adapt > adapter > C hilink_ble_adapter.c > BleGattsSetEncryption(BdAddr, BleSecAct)
1087 int BleGattsSendIndication(int serverId, GattsSendIndParam *param)
1126
1127 /*
1128  * @brief Set the encryption level of the data transmission channel during connection
1129  * @param[in] <bdAddr> remote address
1130  * @param[in] <secAct> BleSecAct
1131  * @return 0-success, other-fail
1132  */
1133 int BleGattsSetEncryption(BdAddr bdAddr, BleSecAct secAct)
1134 {
1135     printf("%s BleGattsSetEncryption enter, secAct:%d.\r\n", BLE_HILINK_SERVER_LOG, secAct);
1136
1137     gap_ble_sec_params_t sec_params = {0};
1138     sec_params.bondable = 1;
1139     sec_params.io_capability = g_io_cap_mode;
1140     sec_params.sc_enable = 0;
1141     sec_params.sc_mode = secAct;
1142     int ret = gap_ble_set_sec_param(&sec_params);
1143     if (ret != 0) {
1144         printf("%s gap_ble_set_sec_param fail, ret:%d.\r\n", BLE_HILINK_SERVER_LOG, ret);
1145     }
1146     return 0;
1147 }
1148

```

(7) 在低功耗状态下，模组做星闪/蓝牙从机时功耗偏高，需要怎么解决

在星闪/蓝牙处于未连接状态时，需要在模组进入深睡状态时，在对应的回调函数中更新广播参数（增加广播间隔时间）来降低功耗，退出低功耗时恢复即可。处于长连接状态时，保连电流会受参数 `interval` 和 `latency` 影响比较大，也需要在模组进入深睡状态时更新对应的连接参数

```

void hf_pm_low_power_entry(void)
{
    pm_state_trans_handler_t handler = {
        .work_to_standby = pm_state_work_to_standby,
        .standby_to_sleep = pm_state_standby_to_sleep,
        .standby_to_work = pm_state_standby_to_work,
        .sleep_to_work = pm_state_sleep_to_work,
    };
    uapi_pm_state_trans_handler_register(&handler);

    uapi_pm_work_state_reset();
    uapi_pm_set_state_trans_duration(0xFFFFFFFF, 0xFFFFFFFF);
}

```

(8) 如何实时获取蓝牙的连接状态？

可在下图中标注的蓝牙连接状态回调函数中进行获取

```

application > samples > hilink > ohos_connect > hilink_adapt > entry > C hilink_ble_main.c > ...
379 void HILINK_ReportSidState(char * sid_char, char * data_char)
420 }
421
422 static int BleCfgNetProcess(BLE_CfgNetStatus status)
423 {
424     printf("BleCfgNetProcess status[%d]\n", status);
425     #ifdef _HSF_GENERAL_
426     if(status == CFG_NET_SPEKE_SUCCESS)
427     {
428         send_wifi_state_fun(HILINK_SERVER_CONNECT, "SERVER_CONNECT");
429         int adv_type = hf_get_hilink_adv_type();
430         if ((adv_type == HF_BLE_ADV_NEARBY_NO_RETURN || adv_type == HF_BLE_ADV_LOCAL))
431         {
432             extern void BLE_SetAdvType(int type);
433             BLE_SetAdvType(BLE_ADV_LOCAL_NAME);
434         }
435         else if(adv_type == HF_BLE_ADV_NEARBY_V0)
436         {
437             extern void BLE_SetAdvType(int type);
438             BLE_SetAdvType(BLE_ADV_NEARBY_V0);
439             hf_change_adv_timer();
440         }
441     }
442     else if(status == CFG_NET_BLE_DIS_CONNECT)
443     {
444         send_wifi_state_fun(HILINK_BLE_DISCONNECT, "BLE_DISCONNECT");
445     }
446     else if(status == CFG_NET_BLE_CONNECT)
447     {
448         send_wifi_state_fun(HILINK_BLE_CONNECT, "BLE_CONNECT");
449     }
450     #endif
451     return 0;
452 }
453

```

(9) 使用星闪连接时，MTU 大小有限制吗？为什么设置时只允许 251 字节？

MTU 可以设置 251-520 字节，连接前要先通过对外接口 `ssaps_set_info` 服务端设置 MTU 自定义最大值，例如 500，客户端再根据需求设置成想要的值，最后协商结果均为客户端设置的值。

```

C de_uart_client > C de_uart_client_sample_connect_state_changed_cbk
145 param.rx_pilot_density = 0.02; // 导频密度10%
146 param.g_feedback = 0;
147 param.t_feedback = 0;
148 if (sle_set_phy_param(get_g_sle_uart_com_id(), &param) != 0) {
149     oal_printk("sle_set_phy_param fail\n", SLE_UART_CLIENT_LOG);
150     return;
151 }
152 oal_printk("sle_set_phy_param success\n", SLE_UART_CLIENT_LOG);
153
154 // 非阻塞采样使用phy参数
155 #ifndef _NON_BLOCK_SAMPLE_USE_PHY_PARAM_
156 #endif
157 #endif
158 #endif
159
160 static void sle_uart_client_sample_connect_state_changed_cbk(uint16_t com_id, const sle_addr_t *addr,
161     sle_acb_state_t com_state, sle_pair_state_t
162     sle_disc_reason_t disc_reason)
163 {
164     unused(addr);
165     unused(pair_state);
166     oal_printk("com state changed disc_reason:0x%x\n", SLE_UART_CLIENT_LOG, disc_reason);
167     g_sle_uart_com_id = com_id;
168
169     sle_connection_param_update_t con_param = {0};
170     con_param.com_id = com_id;
171     con_param.interval_max = 48; // 每个slot 0.125ms
172     con_param.interval_min = 48; // 每个slot 0.125ms
173     con_param.max_latency = 0;
174     con_param.supervision_timeout = 500; // 设置连接超时500ms
175
176     if (com_state == SLE_ACB_STATE_CONNECTED) {
177         oal_printk("sle_sle_acb_state_connected\n", SLE_UART_CLIENT_LOG);
178         sle_update_connect_param(&con_param);
179     }
180
181     ssaps_exchange_info_t info = {0};
182     info.mtu_size = 500;
183     info.version = 1;
184     ssaps_exchange_info_req(0, com_id, &info);
185
186 #ifdef CONFIG_SAMPLE_SUPPORT_LOW_LATENCY_TYPE
187     sle_low_latency_rx_enable();
188     sle_low_latency_set_get_g_sle_uart_com_id(), true, SLE_UART_LOW_LATENCY_2K;
189     sle_uart_client_sample_set_phy_param();
190     oal_msleep(SLE_UART_TASK_DELAY_MS);
191     sle_set_acb_get_g_sle_uart_com_id(), SLE_UART_QPSK_PCS;
192     oal_printk("sle_low_latency_rx_enable\n", SLE_UART_CLIENT_LOG);
193 }
194
195 application > samples > products > de_uart > de_uart_server > C de_uart_server > C de_ssaps_register_cbk(ssaps_read_request_callback, ssaps_w
196 static void ssaps_add_service_cbk(uint16_t server_id) { ssaps_register
197     sample_at_log_print("sle add service cbk callback server_id:0x, handle:0x, status:0x\n", SLE_UART
198     server_id, handle, status);
199     sle_uart_uid_print(uid);
200 }
201
202 static void ssaps_add_property_cbk(uint16_t server_id, sle_uid_t *uid, uint16_t service_handle,
203     uint16_t handle, errcode_t status)
204 {
205     sample_at_log_print("sle add property cbk callback server_id:0x, service_handle:0x, handle:0x, status
206     sle_uart_uid_print(uid);
207     sle_uart_uid_print(handle);
208 }
209
210 static void ssaps_add_descriptor_cbk(uint16_t server_id, sle_uid_t *uid, uint16_t service_handle,
211     uint16_t property_handle, errcode_t status)
212 {
213     sample_at_log_print("sle add descriptor cbk callback server_id:0x, service_handle:0x, property_hand
214     status:0x\n", SLE_UART_SERVER_LOG, server_id, service_handle, property_handle, status);
215     sle_uart_uid_print(uid);
216 }
217
218 static void ssaps_delete_all_service_cbk(uint16_t server_id, errcode_t status)
219 {
220     sample_at_log_print("sle delete all service callback server_id:0x, status:0x\n", SLE_UART_SERVER_L
221     server_id, status);
222 }
223
224 static errcode_t sle_ssaps_register_cbks(ssaps_read_request_callback ssaps_read_callback, ssaps_write_r
225     ssaps_write_callback)
226 {
227     errcode_t ret;
228     ssaps_callbacks_t ssaps_cbk = {0};
229     ssaps_cbk.add_service_cb = ssaps_add_service_cbk;
230     ssaps_cbk.add_property_cb = ssaps_add_property_cbk;
231     ssaps_cbk.add_descriptor_cb = ssaps_add_descriptor_cbk;
232     ssaps_cbk.start_service_cb = ssaps_start_service_cbk;
233     ssaps_cbk.delete_all_service_cb = ssaps_delete_all_service_cbk;
234
235     ssaps_exchange_info_t info = {0};
236     info.mtu_size = 500;
237     info.version = 1;
238     ssaps_set_info(0, &info);
239
240     ssaps_cbk.mtu_changed_cb = ssaps_mtu_changed_cbk;
241     ssaps_cbk.read_request_cb = ssaps_read_callback;
242     ssaps_cbk.write_request_cb = ssaps_write_callback;
243     ret = ssaps_register_callbacks(&ssaps_cbk);
244     if (ret != ERR_SUCCESS_SLE_SUCCESS) {
245         sample_at_log_print("sle_ssaps_register_cbks,ssaps_register_callbacks fail :0x\n", SLE_UART
246         ret);
247     }
248 }

```

(10) 使用星闪保持长连接时，进入低功耗状态，在对应回调函数中增加连接间隔的数值来降低功耗，功耗反正会增加，是什么原因？

需要满足该公式，才能正常更新连接参数：

$$\text{supervision_timeout} > 2 * (1 + \text{max_latency}) * (\text{interval})$$

(11) 蓝牙在设置发包参数时返回异常码 0x80006007，是什么原因？

需要排查连接句柄是否有误，可使用回调函数中传入的 `conn_id`

```
static void ble_uart_mtu_changed_cbk(uint8_t server_id, uint16_t conn_id, uint16_t mtu_size, errcode_t status)
{
    osal_printk("%s MtuChanged--server_id:%d conn_id:%d\n", BLE_UART_SERVER_LOG, server_id, conn_id);
    osal_printk("%s mtusize:%d, status:%d\n", BLE_UART_SERVER_LOG, mtu_size, status);
    gap_le_set_data_length_t param = {0};
    param.conn_handle = conn_id;
    param.maxtxoctets = MAX_TX_OCTETS;
    param.maxtxtime = MAX_TX_TIME;
    errcode_t ret = gap_ble_set_data_length(&param);
    if (ret != ERRCODE_BT_SUCCESS) {
        osal_printk("%s set_data length failed ret = %d\n", BLE_UART_SERVER_ERROR, ret);
    }
    else
        osal_printk("MTU setup successful ret = %d\n", ret);
}
```

4. 环境搭建和编译常见问题

(1) 编译报错编解码解析失败

Q: 编译报错 `UnicodeDecodeError: 'utf-8' codec can't decode byte...`

A: 尝试如下方法:

方法 1:

终端命令行输入: `sudo gedit ~/.bashrc`

最后一行输入 `export LC_ALL=C.UTF-8`

保存退出, 重启虚拟机

方法 2:

终端命令行输入:

`sudo apt-get update`

`sudo apt-get install locales`

安装/修复完成后输入 `locale -a` 查看安装结果

重启虚拟机