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 接口及相关功能使用问题 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
（4）操作用户区域 flash 相关 API 说明 5
（5）SDK 开启标准串口功能 7
3. 蓝牙、星闪相关问题 8
（1）使用蓝牙注册 gatt 客户端时，接口返回 80006006 异常码 8
（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? 8
（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? 9
（4）对蓝牙广播的时间有要求，在哪里进行修改？ 9
（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 9
（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 ..9
（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 10
（8） 如何实时获取蓝牙的连接状态？ 10
（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？ 11
（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？ 12
（11） 蓝牙在设置发包参数时返回异常码 0x80006007，是什么原因？ 12
4. 环境搭建和编译常见问题 13
（1）编译报错编解码解析失败 13
5. SDK RAM 优化方案 14
（1）方案简述 14
（2）修改方法 14
（3）修改验证 17

1. 产品配网常见的问题

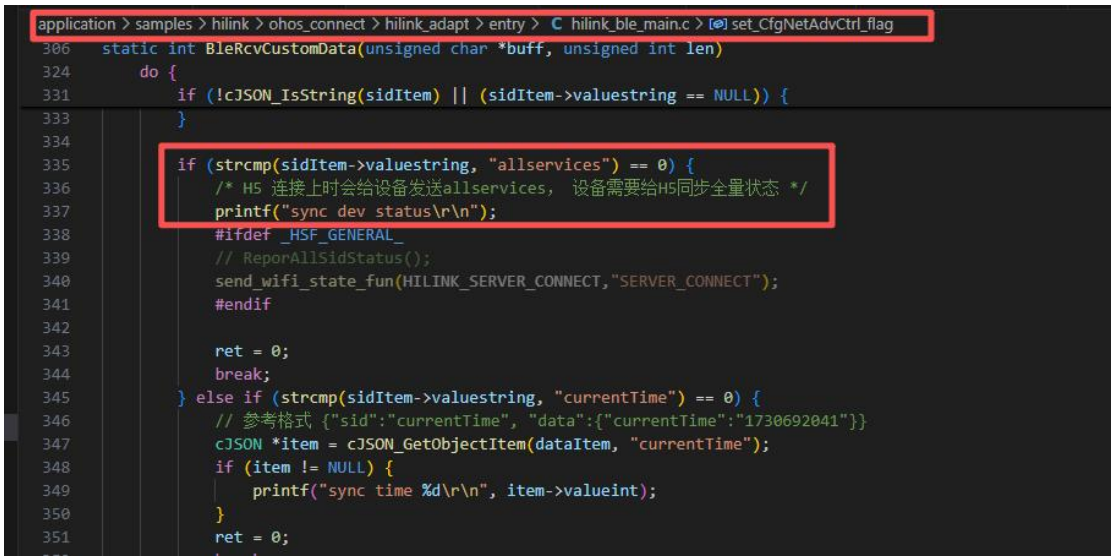
（1）设备配网上线后，H5 页面显示的属性数值异常

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”\”

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.connectic	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	0-连接 1-正在连接 2-断开	
		childlockSwi	童锁开关	switch									
					on	童锁	childLockSw	characteristic.on	GET/REPORT/PUT	bool	枚举-固定枚举范围	0-关 1-开	
		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	0-关 1-轻柔 2-舒缓 3-强力	
		globalVolume	全局音量	gear									
					globalVolum	全局音量	globalVolum	characteristic.ge	GET/REPORT/PUT	enum	枚举-自定义枚举范围	0-静音 1-一档 2-二档 3-三档	
		hotcompressI	热敷温度	gear									
					hotcompress	热敷温度	hotcompress	characteristic.ge	GET/REPORT/PUT	enum	枚举-自定义枚举范围	0-不加热 1-低温 2-中温 3-高温	
		mode	模式	mode									
					mode	模式	Mode	characteristic.mo	GET/REPORT/PUT	enum	枚举-自定义枚举范围	0-轻度模式 1-中度模式 2-高度模式 3-眼保健操 4-舒缓助眠	
		switch	开关	switch									
					on	开关	on	characteristic.on	GET/REPORT/PUT	bool	枚举-固定枚举范围	0-关 1-开	

（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



（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。

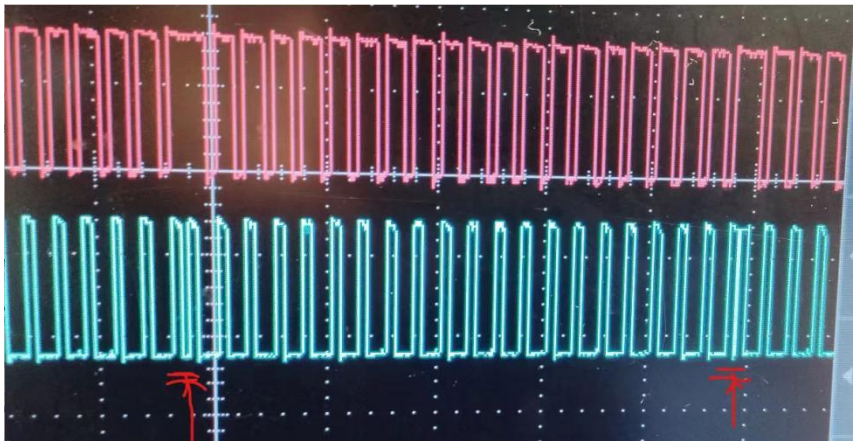
（5）设备添加注册成功，点击设备进入 H5 页面显示网络异常没有控制界面

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 接口及相关功能使用问题

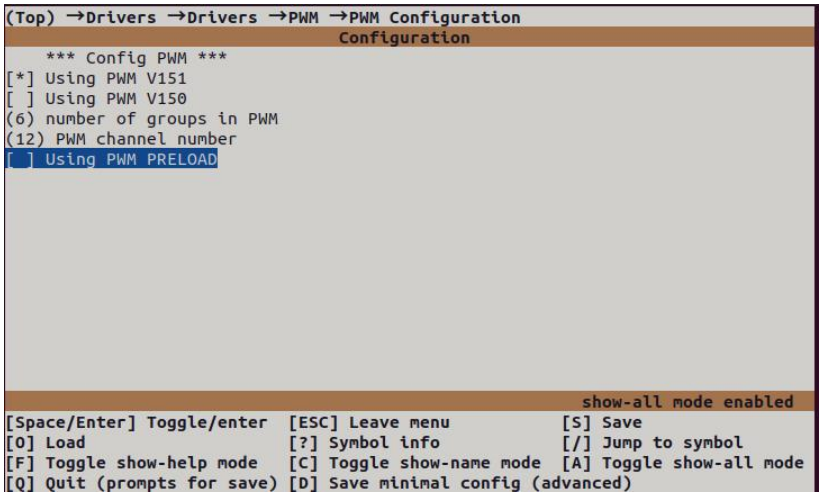
（1）PWM 相关接口的使用

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



（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

```c
static bool pmu_control_pin_reset_enable(pin_t pin)
{
    uapi_pin_set_mode(pin, 0);
    uapi_pin_set_pull(pin, PIN_PULL_UP);
    uapi_pin_set_ie(pin, PIN_IE_1);
    reg16_setbit(0x5702C51C, 0x0);
    reg16_setbits(0x5702C51C, 0x4, 0x5, (uint8_t)pin);
    uapi_tcxo_delay_ms(5);
    reg16_clrbit(0x5702C51C, 0x0);
    return true;
}
```

（3）重置配网接口 HILINK_BT_HardRevoke()使用注意事项

Q： 调用此接口之后模组打印下图中的日志，是什么原因？

```
[HSW]HILINK_MutexLock():124, mutex lock error -6
[HSW]HILINK_MilliSleep():213, delay 200 ms error -6
[HSW]HILINK_MutexLock():124, mutex lock error -6
[HSW]HILINK_MilliSleep():213, delay 200 ms error -6
[HSW]HILINK_MutexLock():124, mutex lock error -6
[HSW]HILINK_MilliSleep():213, delay 200 ms error -6
[HSW]HILINK_MutexLock():124, mutex lock error -6
[HSW]HILINK_MilliSleep():213, delay 200 ms error -6
[HSW]HILINK_MutexLock():124, mutex lock error -6
[HSW]HILINK_MilliSleep():213, delay 200 ms error -6
[HSW]HILINK_MutexLock():124, mutex lock error -6
[HSW]HILINK_MilliSleep():213, delay 200 ms error -6
[HSW]HILINK_MutexLock():124, mutex lock error -6
[HSW]HILINK_MilliSleep():213, delay 200 ms error -6
[HSW]HILINK_MutexLock():124, mutex lock error -6
[HSW]HILINK_MilliSleep():213, delay 200 ms error -6
[HSW]HILINK_MutexLock():124, mutex lock error -6
[HSW]HILINK_MilliSleep():213, delay 200 ms error -6
[HSW]HILINK_MutexLock():124, mutex lock e boot.
```

A: 不允许在硬件中断里面调用这个接口，可在线程中去调用

（4）操作用户区域 flash 相关 API 说明

1、hfuflash_size

函数原型：

int HSF_API hfuflash_size(void);

说明：

获取用户 flash 大小，单位字节

参数：

无

返回值：

返回用户 flash 大小，单位字节

2、hfuflash_erase_page

函数原型：

int HSF_API hfuflash_erase_page(uint32_t addr, int pages);

说明：

擦除用户 flash 的页

参数：

addr: 用户 flash 逻辑地址

pages: 要擦除的 flash 页数

返回值：

成功返回 HF_SUCCESS,失败返回 HF_FAIL

3、hfuflash_write

函数原型：

int HSF_API hfuflash_write(uint32_t addr, char *data, int len);

说明：

将数据写入到用户 flash 中的指定地址

参数：

addr: 用户 flash 逻辑地址

data: 指向待写入数据的指针

len: 待写入数据的长度

返回值：

如果小于零失败，否则返回实际写入到 flash 的 Bytes 数;

4、hfuflash_read

函数原型：

int HSF_API hfuflash_read(uint32_t addr, char *data, int len);

说明：

从用户 flash 中的指定地址读取数据

参数：

addr: 用户 flash 逻辑地址

data: 指向存储读取数据的指针

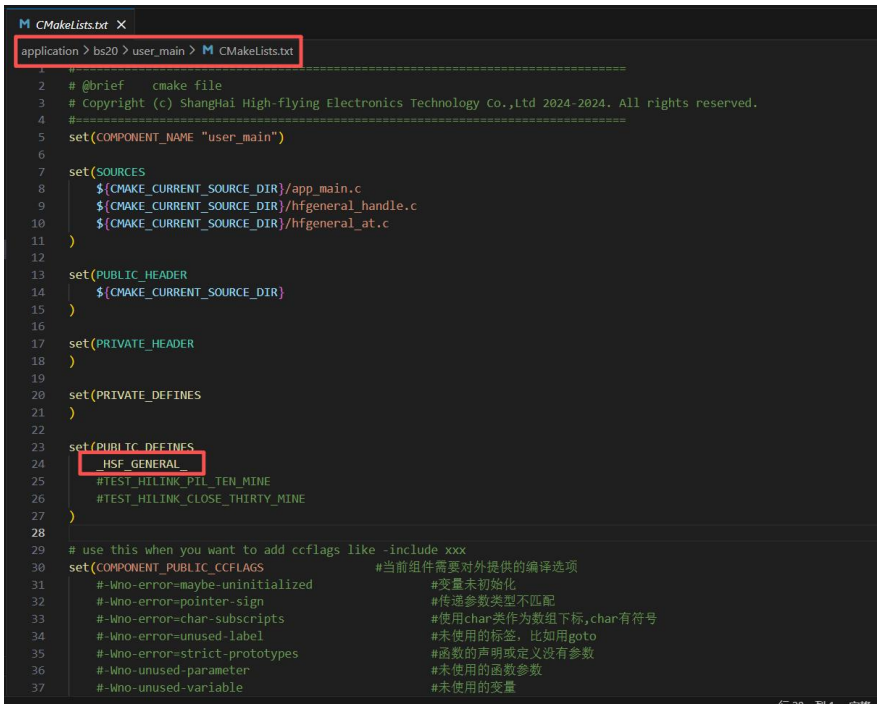
len: 待读取数据的长度

返回值：

如果小于零失败，否则返回成功读取的 Bytes 数;

（5）SDK 开启标准串口功能

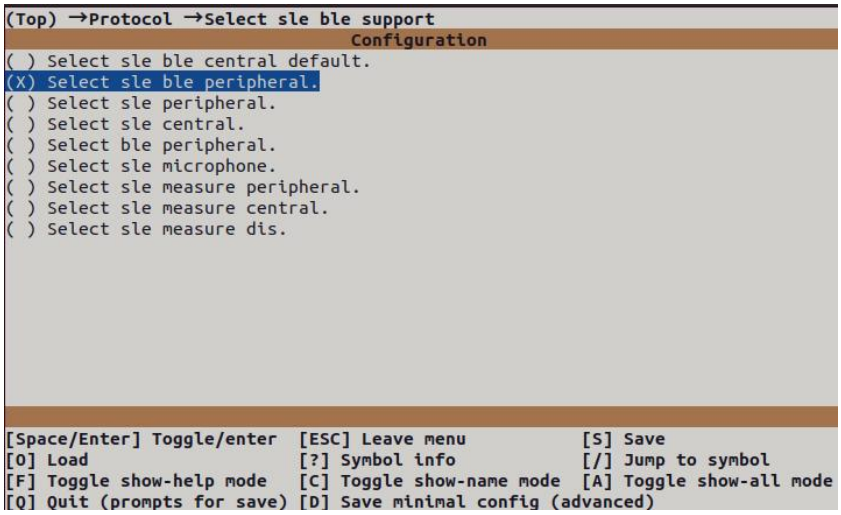
定义宏_HSF_GENERAL_即可打开，反之则关闭。



3. 蓝牙、星闪相关问题

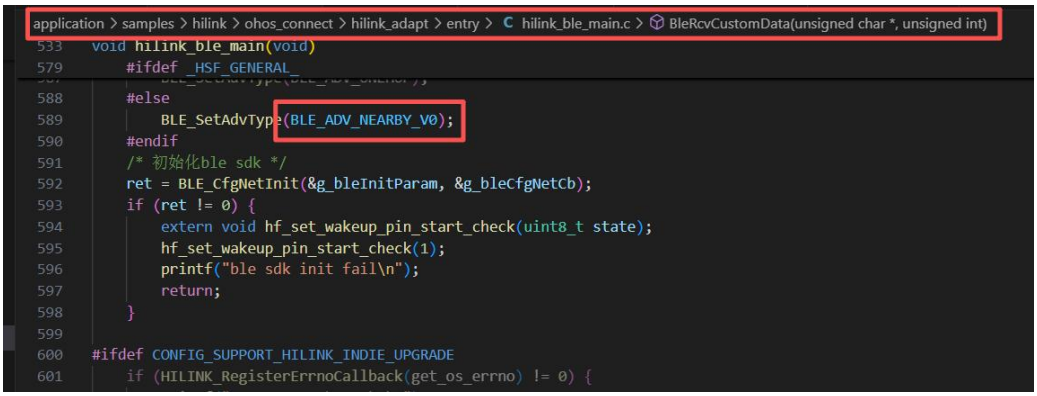
（1）使用蓝牙注册 gatt 客户端时，接口返回 80006006 异常码

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 主从机 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：拉取半模态卡片的靠近发现、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



```
/* BLE广播类型定义 */
typedef enum {
    BLE_ADV_DEFAULT,//新蓝牙靠近发现
    BLE_ADV_NEARBY_V0,//拉取半模态卡片的靠近发现
    BLE_ADV_ONEHOP,//蓝牙碰一碰
    BLE_ADV_LOCAL_NAME,//常态广播
    BLE_ADV_CUSTOM
} 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

（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 inteval 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);

}
```

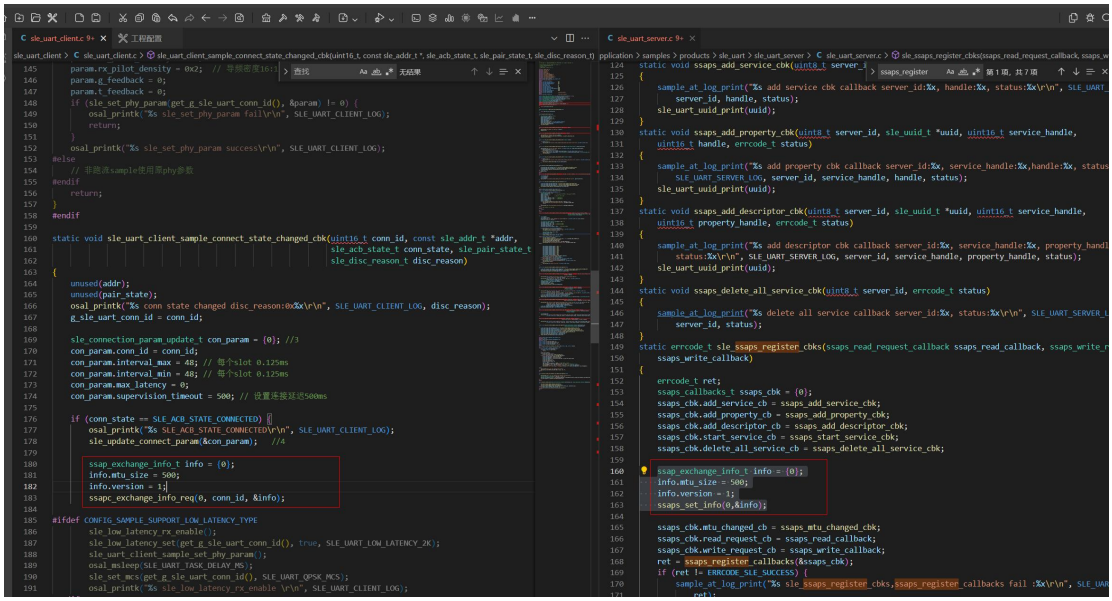
（8）如何实时获取蓝牙的连接状态？

可在下图中标注的蓝牙连接状态回调函数中来进行获取

```
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          send_wifi_state_fun(HILINK_BLE_DISCONNECT,"BLE_DISCONNECT");
444      else if(status == CFG_NET_BLE_CONNECT)
445          send_wifi_state_fun(HILINK_BLE_CONNECT,"BLE_CONNECT");
446      #endif
447      return 0;
448  }
449
```

（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，客户端再根据需要设置成想要的值，最后协商结果均为客户端设置的值。



（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：
supervision_timeout>2 * (1+max_lantency)*(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 = MAXTXOCTETS;
    param.maxtxtime = MAXTXTIME;
    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

5. SDK RAM 优化方案

（1）方案简述

将系统栈 DTCM 开头移动到 ITCM 结尾，空出的 DTCM 部分作为 RAM 的补充，可增加 4KB 内存空间

（2）修改方法

共五张修改参考图

图 1：

```
#ifdef SUPPORT_STACK_IN_ITCM
/* stack for nmi 0.5k */
#define APP_NMI_STACK_LIMIT (APP_RAMTEXT_ORIGIN + APP_ITCM_LENGTH)
#define APP_NMI_STACK_LEN 0x200
#define APP_NMI_STACK_BASEADDR (APP_NMI_STACK_LIMIT - APP_NMI_STACK_LEN)

/* stack for exception 0.5k */
#define APP_EXCP_STACK_LIMIT (APP_NMI_STACK_BASEADDR)
#define APP_EXCP_STACK_LEN 0x200
#define APP_EXCP_STACK_BASEADDR (APP_EXCP_STACK_LIMIT - APP_EXCP_STACK_LEN)

/* stack for irq 2k */
#define APP_IRQ_STACK_LIMIT (APP_EXCP_STACK_BASEADDR)
#define APP_IRQ_STACK_LEN 0x800
#define APP_IRQ_STACK_BASEADDR (APP_IRQ_STACK_LIMIT - APP_IRQ_STACK_LEN)

/* stack for normal 1.5k */
#define APP_USER_STACK_LIMIT (APP_IRQ_STACK_BASEADDR)
#define APP_USER_STACK_LEN 0x600
#define APP_USER_STACK_BASEADDR (APP_USER_STACK_LIMIT - APP_USER_STACK_LEN)

#define APP_STACK_TOTAL_LEN (APP_NMI_STACK_LEN + APP_EXCP_STACK_LEN + APP_IRQ_STACK_LEN +
APP_USER_STACK_LEN)
#else
```

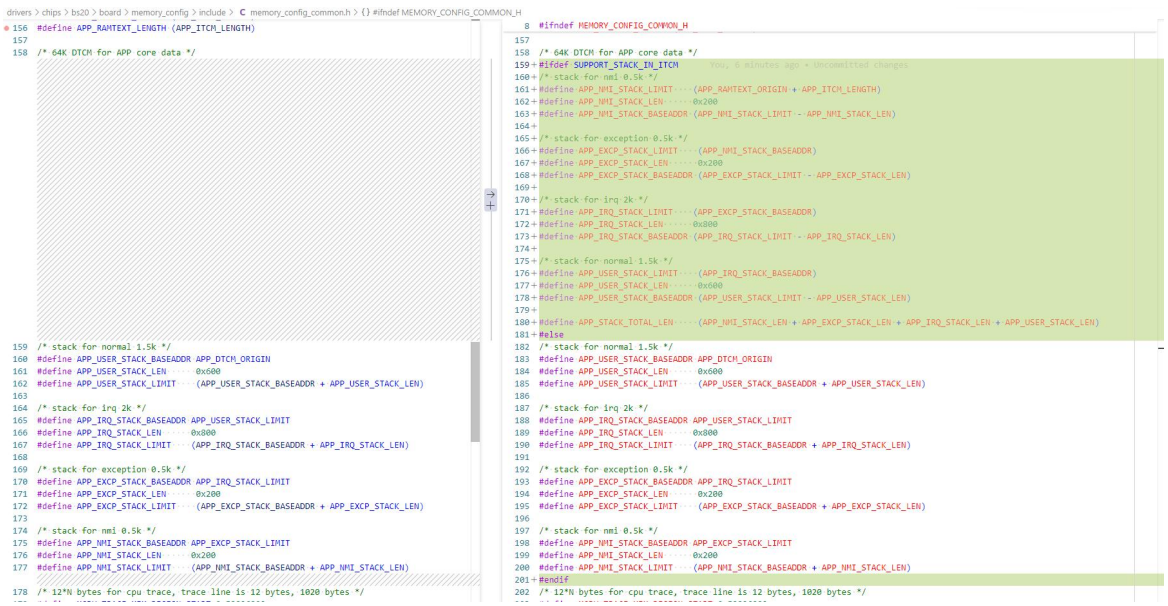


图 2：

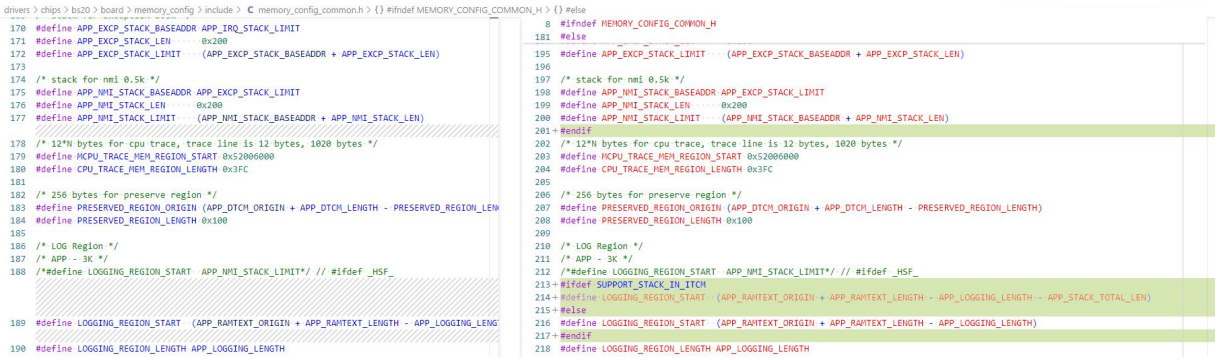


图 3：

```
drivers > chips > bs20 > board > linker > standard > linker.prelds
35  #define PLAT_ROM_FUNC(func) \
48      func(*secureprintoutput_a*.o*) \
49
50  #define ROM_CHECK_OFFSET 0x80
51  #define ITCM_DFX_LEN 0x400
52  #define SYMBOL_ORIGIN 0xFFFFF000
53  #define SYMBOL_LEN 0x200
54
55  /* FLASH sector size is 2K */
56  /* make this table consistent with the configuration given by the security core */
57  MEMORY
58  {
59      ROM                     : ORIGIN = ROM_START,                   LENGTH = ROM_LENGTH
60  #ifdef OS_DFX_SUPPORT
61      ITCM                    : ORIGIN = APP_RAMTEXT_ORIGIN, LENGTH = APP_RAMTEXT_LENGTH - ITCM_DFX_LEN - APP_LOGGING_LENGTH
62      ITCM_DFX                : ORIGIN = ORIGIN(ITCM) + LENGTH(ITCM), LENGTH = ITCM_DFX_LEN
63  #elif defined(SUPPORT_STACK_IN_ITCM)
64      ITCM                    : ORIGIN = APP_RAMTEXT_ORIGIN, LENGTH = APP_RAMTEXT_LENGTH - APP_STACK_TOTAL_LEN - APP_LOGGING_LENGTH
65      ITCM_STACK              : ORIGIN = APP_USER_STACK_BASEADDR, LENGTH = APP_STACK_TOTAL_LEN
66  #else
67      ITCM                    : ORIGIN = APP_RAMTEXT_ORIGIN, LENGTH = APP_RAMTEXT_LENGTH - APP_LOGGING_LENGTH
68  #endif
69      DTCM                    : ORIGIN = APP_DTCM_ORIGIN,             LENGTH = APP_DTCM_LENGTH - PRESERVED_REGION_LENGTH
70      PRESERVE                : ORIGIN = PRESERVED_REGION_ORIGIN,     LENGTH = PRESERVED_REGION_LENGTH
71      FLASH_STARTUP           : ORIGIN = APP_PROGRAM_ORIGIN,          LENGTH = 0x300
72      FLASH_PROGRAM           : ORIGIN = APP_PROGRAM_ORIGIN + 0x300,  LENGTH = APP_PROGRAM_LENGTH - 0x300
73      CPUTRACE_RAM            : ORIGIN = MCPU_TRACE_MEM_REGION_START, LENGTH = CPU_TRACE_MEM_REGION_LENGTH
74      SYMBOL_UNREACH          : ORIGIN = SYMBOL_ORIGIN,               LENGTH = SYMBOL_LEN
75  }
```

图 4：

drivers > chips > bs20 > board > linker > standard > linker.prelds

```
78   {
98       {
105          __irq_stack_top__ = .;
106          __irq_stack_top = .;
107          . += APP_EXCP_STACK_LEN;
108          . = ALIGN(16);
109          __excp_stack_top__ = .;
110          __exc_stack_top = .;
111          . += APP_NMI_STACK_LEN;
112          . = ALIGN(16);
113          __nmi_stack_top__ = .;
114          __nmi_stack_top = .;
115          g_system_stack_end = .;
116  #ifdef SUPPORT_STACK_IN_ITCM
117      } > ITCM_STACK
118  #else
119  //#ifdef CONFIG_SUPPORT_LOG_THREAD
120  //          . += LOGGING_REGION_LENGTH;
121  //          . = ALIGN(16);
122  //#endif
123      } > DTCM
124  #endif
125      g_system_stack_size = g_system_stack_end - g_system_stack_begin;
126      g_stack_system = g_system_stack_size;
127  }
```

图 5：

build > config > target_config > bs20 > config.py

```
1  #!/usr/bin/env python3
2  # encoding=utf-8
3  # ============================================================================
4  # @brief    Target Definitions File
5  # Copyright CompanyNameMagicTag 2022-2022. All rights reserved.
6  # ============================================================================
7
8  target = {
9      'standard-bs20-n1200': {
10         'base_target_name': 'target_bs20_application_template',
11         'pkg_chip': 'bs20-n1200',
12         'defines': [
13             'FLASH_1M', "USE_RSA3072_SIGN", 'BGLE_TASK_EXIST', 'SUPPORT_MULTI_LIBS',
14             'SW_UART_DEBUG', 'XO_32M_CALI', 'SUPPORT_CHIP_N1200','SUPPORT_STACK_IN_ITCM',
15             #, 'SUPPORT_CFBB_UPG' 'AT_COMMAND','OS_DFX_SUPPORT',
16         ],
```

（3）修改验证

参考下图：多出新的区域 ITCM_STACK 且 DTCM 的使用为空