

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“ ”

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.connectio	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	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		档位	gear	characteristic.ge	GET/REPORT/PUT	enum	枚举-自定义枚举范围	0-关 1-轻柔 2-舒适 3-强力		
		globalVolume	全局音量	gear		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		
		hotcompress	热敷温度	gear		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		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

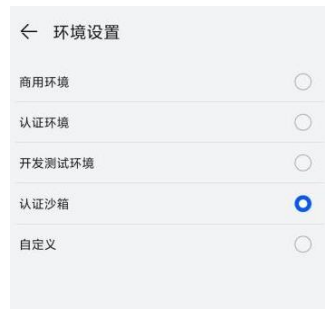
此问题是模组没有及时上报相关数据，H5 连接成功时会下发 SID allservices，模组需要在接收到这个 sid 时来主动上报全量数据同步设备状态

```
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

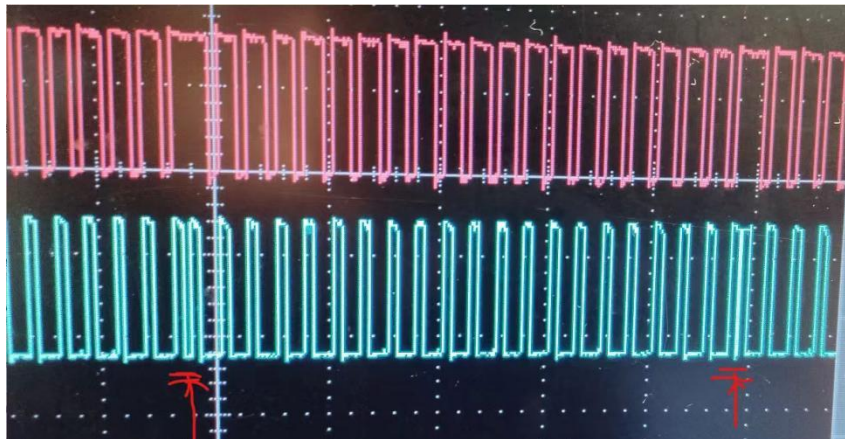


此现象常为平台数据未同步，多次刷新网页重复保存页面可以解决，清理智慧生活 APP 缓存。

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

```
(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

```
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？

[illegible]

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、hfufwrite

函数原型:

```
int HSF_API hfufwrite(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、hfufread

函数原型:

```
int HSF_API hfufread(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

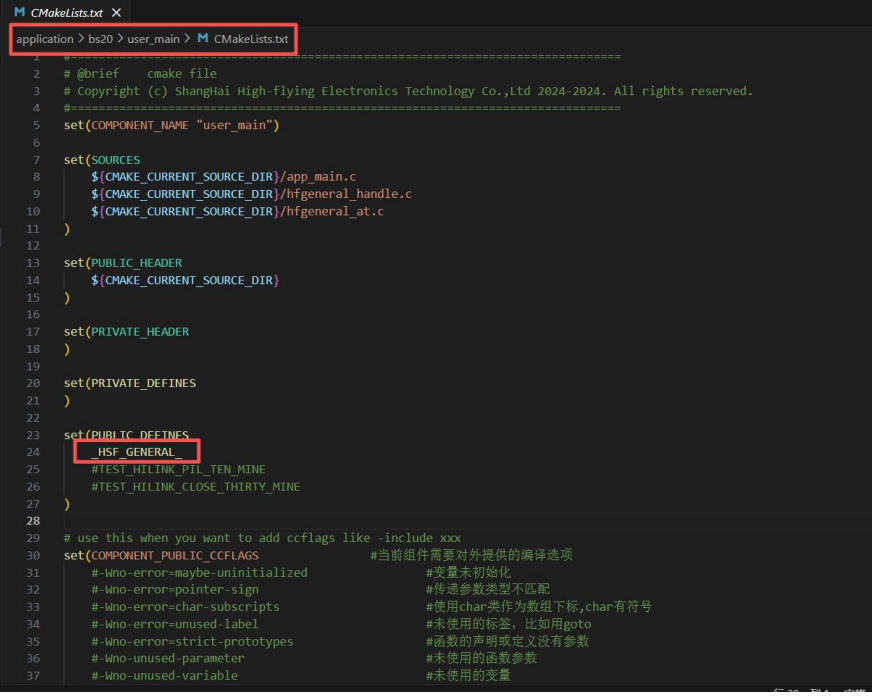
len: 待读取数据的长度

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。

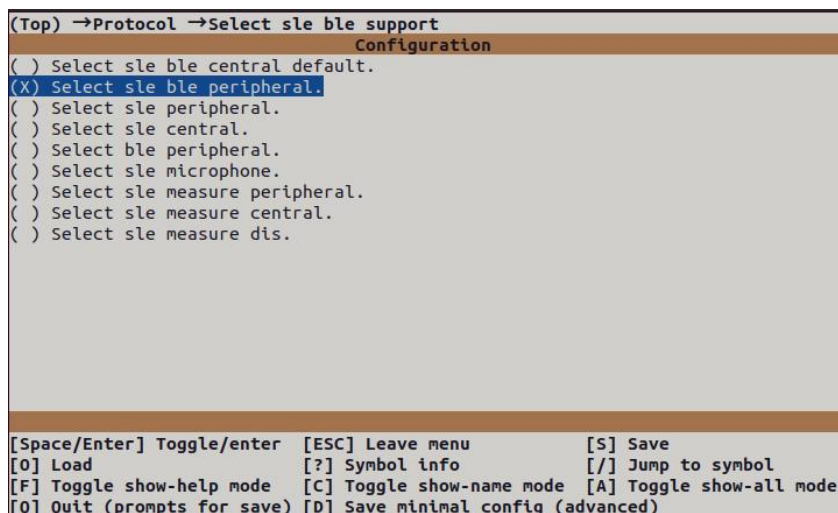


```
1 #-----
2 # @brief  cmake file
3 # Copyright (c) Shanghai High-flying Electronics Technology Co.,Ltd 2024-2024. All rights reserved.
4 #-----
5 set(COMPONENT_NAME "user_main")
6
7 set(SOURCES
8     ${CMAKE_CURRENT_SOURCE_DIR}/app_main.c
9     ${CMAKE_CURRENT_SOURCE_DIR}/hfgeneral_handle.c
10    ${CMAKE_CURRENT_SOURCE_DIR}/hfgeneral_at.c
11 )
12
13 set(PUBLIC_HEADER
14     ${CMAKE_CURRENT_SOURCE_DIR}
15 )
16
17 set(PRIVATE_HEADER
18 )
19
20 set(PRIVATE_DEFINES
21 )
22
23 set(PUBLIC_DEFINES
24     _HSF_GENERAL_
25     #TEST_HILINK_PIL_TEN_MINE
26     #TEST_HILINK_CLOSE_THIRTY_MINE
27 )
28
29 # use this when you want to add ccflags like -include xxx
30 set(COMPONENT_PUBLIC_CCFLAGS #当前组件需要对外提供的编译选项
31     #-Wno-error=maybe-uninitialized #变量未初始化
32     #-Wno-error=pointer-sign #传递参数类型不匹配
33     #-Wno-error=char-subscripts #使用char类作为数组下标, char有符号
34     #-Wno-error=unused-label #未使用的标签, 比如用goto
35     #-Wno-error=strict-prototypes #函数的声明或定义没有参数
36     #-Wno-unused-parameter #未使用的函数参数
37     #-Wno-unused-variable #未使用的变量
```


3. 蓝牙、星闪相关问题

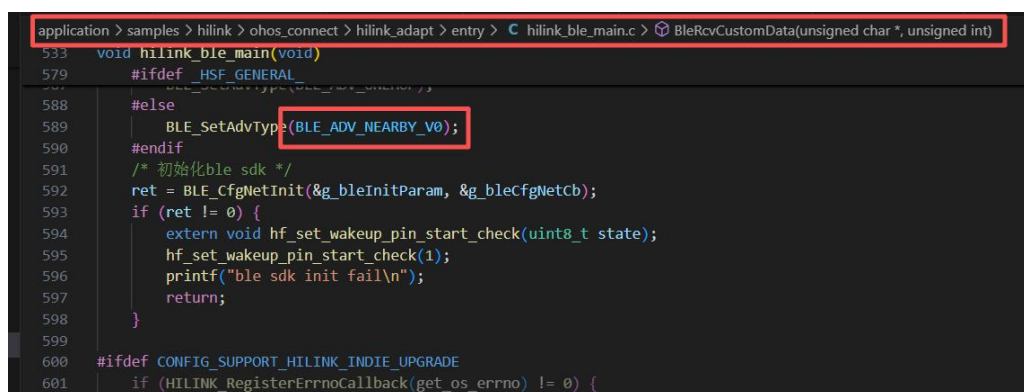
(1) 使用蓝牙注册 gatt 客户端时，接口返回 80006006 异常码

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



```

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

SDK下载&集成开发

蓝牙广播设置

* 设备添加方式

☒ 支持扫码设备添加 [下载二维码](#)

* 蓝牙名称

MANUFACTURER

HI — BSL600 — 1 — 2P3W — 00 序列号 保存

[BLE广播规范](#)

(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_adapter > 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
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(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     ssap_exchange_info_t info = {0};
182     info.mtu_size = 500;
183     info.version = 1;
184     ssap_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_callbacks
196 static void ssaps_add_service_cbk(uint16_t server_id) {
197     ssaps_register_callbacks(ssaps_read_request_callback, ssaps_w
198     sample_at_log_print("sle_uart_server_id:0x%x, handle:0x%x, status:0x%x\n", SLE_UART_
199     server_id, handle, status);
200     sle_uart_uid_print(uid);
201 }
202
203 static void ssaps_add_property_cbk(uint16_t server_id, sle_uid_t *uid, uint16_t service_handle,
204 uint16_t handle, errcode_t status) {
205     sample_at_log_print("sle_uart_server_id:0x%x, service_handle:0x%x, handle:0x%x, status
206     SLE_UART_SERVER_LOG, server_id, service_handle, handle, status);
207     sle_uart_uid_print(uid);
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     sample_at_log_print("sle_uart_server_id:0x%x, service_handle:0x%x, property_hand
213     SLE_UART_SERVER_LOG, server_id, service_handle, property_handle, status);
214     sle_uart_uid_print(uid);
215 }
216
217 static void ssaps_delete_all_service_cbk(uint16_t server_id, errcode_t status) {
218     sample_at_log_print("sle_uart_server_id:0x%x, status:0x%x\n", SLE_UART_SERVER_
219     server_id, status);
220 }
221
222 static errcode_t sle_ssaps_register_callbacks(ssaps_read_request_callback ssaps_read_callback, ssaps_wri
223 ssaps_write_callback) {
224     errcode_t ret;
225     ssaps_callbacks_t ssaps_cbk = {0};
226     ssaps_cbk.add_service_cb = ssaps_add_service_cbk;
227     ssaps_cbk.add_property_cb = ssaps_add_property_cbk;
228     ssaps_cbk.add_descriptor_cb = ssaps_add_descriptor_cbk;
229     ssaps_cbk.start_service_cb = ssaps_start_service_cbk;
230     ssaps_cbk.delete_all_service_cb = ssaps_delete_all_service_cbk;
231
232     ssap_exchange_info_t info = {0};
233     info.mtu_size = 500;
234     info.version = 1;
235     ssaps_set_info(0, &info);
236
237     ssaps_cbk.mtu_changed_cb = ssaps_mtu_changed_cbk;
238     ssaps_cbk.read_request_cb = ssaps_read_callback;
239     ssaps_cbk.write_request_cb = ssaps_write_callback;
240     ret = ssaps_register_callbacks(&ssaps_cbk);
241     if (ret != ERR_SUCCESS) {
242         sample_at_log_print("sle_ssaps_register_callbacks fail:0x%x\n", SLE_UART_
243         ret);
244     }
245 }
246

```

(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 = 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
```



```

drivers > chips > bs20 > board > memory_config > include > C memory_config_common.h > {} #ifndef MEMORY_CONFIG_COMMON_H
156 #define APP_RAMTEXT_LENGTH (APP_ITCM_LENGTH)
157
158 /* 64K DTCM for APP core data */
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

图 2:

```

drivers > chips > bs20 > board > memory_config > include > C memory_config_common.h > {} #ifndef MEMORY_CONFIG_COMMON_H
170 #define APP_EXCP_STACK_BASEADDR APP_IRQ_STACK_LIMIT
171 #define APP_EXCP_STACK_LEN 0x200
172 #define APP_EXCP_STACK_LIMIT (APP_EXCP_STACK_BASEADDR + APP_EXCP_STACK_LEN)
173
174 /* stack for mmi 0.5k */
175 #define APP_MMI_STACK_BASEADDR APP_EXCP_STACK_LIMIT
176 #define APP_MMI_STACK_LEN 0x200
177 #define APP_MMI_STACK_LIMIT (APP_MMI_STACK_BASEADDR + APP_MMI_STACK_LEN)
178
179 /* 12N bytes for cpu trace, trace line is 12 bytes, 1020 bytes */
180 #define CPU_TRACE_MEM_REGION_START 0x20000000
181 #define CPU_TRACE_MEM_REGION_LENGTH 0x3FC
182
183 /* 256 bytes for preserve region */
184 #define PRESERVED_REGION_ORIGIN (APP_DTCM_ORIGIN + APP_DTCM_LENGTH - PRESERVED_REGION_LEN)
185 #define PRESERVED_REGION_LENGTH 0x100
186
187 /* LOG Region */
188 #define APP - 3K */
189 #define LOGGING_REGION_START APP_MMI_STACK_LIMIT // #ifdef _HSF_
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

图 3:

```

drivers > chips > bs20 > board > linker > standard > linker.prelis
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 0xFFFFFFFF000
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
61     #ifdef OS_DFX_SUPPORT
62         ITCM : ORIGIN = APP_RAMTEXT_ORIGIN, LENGTH = APP_RAMTEXT_LENGTH - ITCM_DFX_LEN - APP_LOGGING_LENGTH
63         ITCM_DFX : ORIGIN = ORIGIN(ITCM) + LENGTH(ITCM), LENGTH = ITCM_DFX_LEN
64     #endif
65     ITCM : ORIGIN = APP_RAMTEXT_ORIGIN, LENGTH = APP_RAMTEXT_LENGTH - APP_STACK_TOTAL_LEN - APP_LOGGING_LENGTH
66     ITCM_STACK : ORIGIN = APP_USER_STACK_BASEADDR, LENGTH = APP_STACK_TOTAL_LEN
67 #else
68     ITCM : ORIGIN = APP_RAMTEXT_ORIGIN, LENGTH = APP_RAMTEXT_LENGTH - APP_LOGGING_LENGTH
69 #endif
70     DTCM : ORIGIN = APP_DTCM_ORIGIN, LENGTH = APP_DTCM_LENGTH - PRESERVED_REGION_LENGTH
71     PRESERVE : ORIGIN = PRESERVED_REGION_ORIGIN, LENGTH = PRESERVED_REGION_LENGTH
72     FLASH_STARTUP : ORIGIN = APP_PROGRAM_ORIGIN, LENGTH = 0x300
73     FLASH_PROGRAM : ORIGIN = APP_PROGRAM_ORIGIN + 0x300, LENGTH = APP_PROGRAM_LENGTH - 0x300
74     CPUTRACE_RAM : ORIGIN = MCU_TRACE_MEM_REGION_START, LENGTH = CPU_TRACE_MEM_REGION_LENGTH
75     SYMBOL_UNREACH : ORIGIN = SYMBOL_ORIGIN, LENGTH = SYMBOL_LEN
76 }

```


图 4:

```
drivers > chips > bs20 > board > linker > standard > linker.prellds
78 {
98 {
104     . = ALIGN(16);
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

```
Generating /opt/liangjunwei/workspace/BSL600/drivers/chips/bs2l
Memory region      Used Size  Region Size  %age Used
ROM:                0 GB      169 KB      0.00%
ITCM:               73656 B     74240 B     99.21%
ITCM_STACK:         4608 B      4608 B     100.00%
DTCM:               0 GB      65280 B     0.00%
PRESERVE:           252 B       256 B     98.44%
FLASH_STARTUP:      488 B       768 B     63.54%
FLASH_PROGRAM:      525186 B    611584 B    85.87%
CPUTRACE_RAM:       1020 B      1020 B    100.00%
SYMBOL_UNREACH:     220 B       512 B     42.97%
```