蓝牙应用开发 FAQ 说明文档

V0.5

版本记录

版本	日期	修改日志
V0.5	2019-12-5	1. URL 编码在线网站的增加，以及注意事项； 2. 新增 AEC 使用的注意事项；
V0.4	2019-10-29	3. 增加 BLE 属性配置，Attribute flags 的参数说明； 4. 蓝牙频偏值相关说明；
V0.3	2019-9-18	增加蓝牙名称(非英文格式)说明
V0.2	2019-9-16	1. 完善 BLE 工具说明(增加 UUID128 的修改方式，属性的定义和移植方式) 2. 调整某些截图和内容说明
V0.1	2019-9-6	初版发布

Q：蓝牙参数配置

A：

1) 蓝牙功能开关 CFG_APP_BT_MODE_EN 在 app_config.h 文件中；
2) 基带 EM 空间的分配参数配置在 app_config.h 文件中；
3) 蓝牙应用功能的配置，蓝牙相关参数配置数据在 bt_config.h 文件中；

Q：蓝牙地址获取方式

A：

1) 上电后从 flash 中指定位置读取蓝牙地址信息；
2) 如 flash 无数据，从芯片的 efuse 中读取 chip ID 生成蓝牙地址，然后保存到 flash；
3) 如 efuse 数据为空，则通过随机数生成器生成蓝牙地址，然后保存到 flash；(该方式基本无效，只有在前期芯片没有烧录 efuse 信息的时候才有效，后续的芯片必然会烧录 efuse，存在 chip ID 信息)

注：用户可以不遵守如上的规则，自行定义蓝牙地址；

Q：蓝牙相关数据保存位置

A：数据保存位置如下：

1) 蓝牙地址，蓝牙名称，BLE 地址，频偏参数等蓝牙相关配置参数保存在 flash 的最后 4K，即：0x1FF000 ~ 0x1FFFFF；该区域受 download 保护；
2) 蓝牙的配对记录区域：0x1FB000 ~ 0x1FCFFF(共 8K)
3) 蓝牙最后 1 次连接设备的地址：0x1FD000 ~ 0x1FDFFF(共 4K)

Q：蓝牙频偏值相关说明

A：

1) 第一次上电，系统会调用默认频偏值 BT_TRIM(bt_config.h)，并调用该删除保存到 flash 中；后续都是从 flash 中读取频偏值；
2) 可以通过串口日志信息观察当前板级的频偏值，
串口日志表现如下：Freq trim:0x17
3) 在使用蓝牙测试盒的时候，可以根据实际情况选择操作：
a. 频偏校准：将测试盒档位打到校准频偏模式(000)，在测试盒连接成功被测设备后，会自动将频偏值校准，并保存到 flash 中；测试盒上显示校准后的频率偏差值；
b. 不需要频偏校准：将测试盒档位达到不校准频偏模式(001)，在测试盒连接成功被测设备后，不校准频偏，只是将当前被测设备的频率偏差值显示在屏幕上；

Q：蓝牙名称说明

A：

1) 在 bt_config.h 头文件中配置蓝牙名称 BT_NAME
2) 如蓝牙名称为中文，或者其他语言，请将内容进行 URL 编码，再写入 bt_LocalDeviceName；
需要进行 URL 编码的数据可以在网站在线转换：http://tool.chinaz.com/tools/urlencode.aspx
注：数字和字母不需要转换；网上转换的数据为十六进制；

Q：BLE 地址、名称修改和配套工具的使用

A：

1) BLE 的地址先从 flash 的指定位置读取，如无保存的地址信息，则通过 BT 的地址来生成，并保存到 flash；

2) 修改 BLE 的名称和 UUID 在 2 处地方：

a. BLE 广播数据包的数组中 advertisement_data；

```
const uint8_t advertisement_data[] = {
    0x02, 0x01, 0x02,       //flag:LE General Discoverable
    0x03, 0x03, 0x00, 0xab, //16bit service UUIDs
    9,    0x09, 'B', 'P', '1', '0', '-','B','L','E',//
};
```

UUID

BLE名称

长度

b. 在 gatt 配置的相关参数内，需要使用 BLE 的 gatt_tool 工具，在 example.gatt 中修改配置；

example.gatt

```
1  PRIMARY_SERVICE, GAP_SERVICE
2  CHARACTERISTIC, GAP_DEVICE_NAME, READ, "BP10-BLE"
3
4  PRIMARY_SERVICE, AB00
5  CHARACTERISTIC, AB01, READ | WRITE | WRITE_WITHOUT_RESPONSE | DYNAMIC,
6  CHARACTERISTIC, AB02, NOTIFY | DYNAMIC,
7  CHARACTERISTIC, AB03, NOTIFY | DYNAMIC,
8
```

BLE名称

UUID

c. gatt_tool.exe 工具生成 out.h 中的 profile_data 数组，替换掉 ble_app_func.c 中的 profile_data；

d. 将 out.h 中 list service handle ranges 和 list mapping between characteristics and handles 的定义的参数同步到 ble_app_func.c，如下图所示：

```
//
// list service handle ranges
//
#define ATT_SERVICE_GAP_SERVICE_START_HANDLE 0x0001
#define ATT_SERVICE_GAP_SERVICE_END_HANDLE 0x0003
#define ATT_SERVICE_AB00_START_HANDLE 0x0004
#define ATT_SERVICE_AB00_END_HANDLE 0x000c

//
// list mapping between characteristics and handles
//
#define ATT_CHARACTERISTIC_GAP_DEVICE_NAME_01_VALUE_HANDLE 0x0003
#define ATT_CHARACTERISTIC_AB01_01_VALUE_HANDLE 0x0006
#define ATT_CHARACTERISTIC_AB02_01_VALUE_HANDLE 0x0008
#define ATT_CHARACTERISTIC_AB02_01_CLIENT_CONFIGURATION_HANDLE 0x0009
#define ATT_CHARACTERISTIC_AB03_01_VALUE_HANDLE 0x000b
#define ATT_CHARACTERISTIC_AB03_01_CLIENT_CONFIGURATION_HANDLE 0x000c
```

3) 工具的位置：MVsB1_Base_SDK\middleware\bluetooth\ble_adv_data_tool；使用方式请参阅该文件夹内的 readme.txt；

4) Gap_service 服务中，目前只是定义了 GAP_DEVICE_NAME 属性，假如需要增加属性，则自行添加；

```
assigned_uuids =
{
    'GAP_SERVICE'                       : 0x1800,
    'GATT_SERVICE'                      : 0x1801,
    'GAP_DEVICE_NAME'                   : 0x2a00,
    'GAP_APPEARANCE'                    : 0x2a01,
    'GAP_PERIPHERAL_PRIVACY_FLAG'       : 0x2a02,
    'GAP_RECONNECTION_ADDRESS'          : 0x2a03,
    'GAP_PERIPHERAL_PREFERRED_CONNECTION_PARAMETERS'     : 0x2a04,
    'GATT_SERVICE_CHANGED'              : 0x2a05,
}
```

```
PRIMARY_SERVICE, GAP_SERVICE
CHARACTERISTIC, GAP_DEVICE_NAME, READ, "B1X-BLE-Demo"

PRIMARY_SERVICE, B75C49D2-04A3-4071-A0B5-35853EB08307     在此插入GAP/GATT相关属性的内容
CHARACTERISTIC, B85C49D2-04A3-4071-A0B5-35853EB08307, NOTIFY | DYNAMIC,
CHARACTERISTIC, BA5C49D2-04A3-4071-A0B5-35853EB08307, WRITE_WITHOUT_RESPONSE | DYNAMIC,
CHARACTERISTIC, B95C49D2-04A3-4071-A0B5-35853EB08307, NOTIFY | READ | WRITE_WITHOUT_RESPONSE | DYNAMIC,
```

5) 正常使用中，请不要开启 AI 功能(宏开关：CFG_FUNC_AI)，因为 AI 功能会串改 BLE 广播的数据(BLE 名称)；

Q：BLE 的 Attribute Flag 的说明

A：

```
// MARK: ATT Operations
// MARK: Attribute Property Flags
#define ATT_PROPERTY_BROADCAST       0x01
#define ATT_PROPERTY_READ            0x02
#define ATT_PROPERTY_WRITE_WITHOUT_RESPONSE 0x04
#define ATT_PROPERTY_WRITE           0x08
#define ATT_PROPERTY_NOTIFY          0x10
#define ATT_PROPERTY_INDICATE        0x20
#define ATT_PROPERTY_AUTHENTICATED_SIGNED_WRITE 0x40
#define ATT_PROPERTY_EXTENDED_PROPERTIES 0x80
以上的部分是gatt 的spec中定义的标准的Characteristic Properties。

// MARK: Attribute Property Flag, ble stack extension
// value is asked from client
#define ATT_PROPERTY_DYNAMIC         0x100 这个bit位如果置为的话，对应的对这个attribute的读或者写都会有callback上报到应用层，也就是说读或者写的值和长度，都是由上层控制的，协议栈自己不负责管理；如果数据中一次生成就不会变的属性，这个位可以置0
// 128 bit UUID used
#define ATT_PROPERTY_UUID128         0x200    指示128位uuid
// Authentication required
#define ATT_PROPERTY_AUTHENTICATION_REQUIRED 0x400 这个使用于属性扩展的，有些上层的profile会要求对某个属性的访问需要有authenticate，需要对链路进行加密
// Authorization from user required
#define ATT_PROPERTY_AUTHORIZATION_REQUIRED 0x800 这个使用于属性扩展的，有些上层的profile会要求对某个属性的访问需要有authorization，比如说如果有client访问这个属性，会有对应的callback事件上报，具体应用见peripheral_event_handle中SM_Security_event部分的SM_AUTHORIZATION_REQUEST的处理。

高4位，默认全1就可以了，一般不会用到，这个是用来对某个属性指定加密key的长度的。一般不需要指定，默认即可
```

Q：BLE 的服务 UUID 使用 128 位配置

A：

1) 在 example.gatt 中修改配置如下图：

```
PRIMARY_SERVICE, B75C49D2-04A3-4071-A0B5-35853EB08307
CHARACTERISTIC, B85C49D2-04A3-4071-A0B5-35853EB08307, NOTIFY | DYNAMIC,
CHARACTERISTIC, BA5C49D2-04A3-4071-A0B5-35853EB08307, WRITE_WITHOUT_RESPONSE | DYNAMIC,
CHARACTERISTIC, B95C49D2-04A3-4071-A0B5-35853EB08307, NOTIFY | READ | WRITE_WITHOUT_RESPONSE | DYNAMIC,
```

2) 在 gatt_tool.exe 工具生成 out.h 中的 profile_data 数组，替换掉 ble_app_func.c 中的 profile_data；

3) 将 out.h 中 list service handle ranges 和 list mapping between characteristics and handles 的定义的参数同步到 ble_app_func.c；

4) 在 ble_app_func.c 中，同步修改 ble 广播数据包的数据内容：

```
const uint8_t advertisement_data[] = {
长度 0x02, 0x01, 0x02,       //flag:LE General Disconverable
  0x11, 0x07, 0xB7, 0x5C, 0x49, 0xD2, 0x04, 0xA3, 0x40, 0x71, 0xA0, 0xB5, 0x35, 0x85, 0x3E, 0xB0, 0x83, 0x07,    //128bit service UUIDs
  9,   0x09, 'B', 'P', '1', '0', '-','B','L','E',//
};
```

UUID128标识位　　服务UUID128

Q：BLE 的服务属性特征值修改

A：

1) 在 example.gatt 中修改配置如下图：

```
PRIMARY_SERVICE, B75C49D2-04A3-4071-A0B5-35853EB08307
CHARACTERISTIC, B85C49D2-04A3-4071-A0B5-35853EB08307, NOTIFY | DYNAMIC,
CHARACTERISTIC, BA5C49D2-04A3-4071-A0B5-35853EB08307, WRITE_WITHOUT_RESPONSE | DYNAMIC,
CHARACTERISTIC, B95C49D2-04A3-4071-A0B5-35853EB08307, NOTIFY | READ | WRITE_WITHOUT_RESPONSE | DYNAMIC,
```

根据实际的属性进行修改

2) 属性如下：**BROADCAST、READ、WRITE_WITHOUT_RESPONSE、WRITE、NOTIFY、INDICATE；**

3) gatt_tool.exe 工具生成 out.h 中的 profile_data 数组，替换掉 ble_app_func.c 中的 profile_data；

4) 将 out.h 中 list service handle ranges 和 list mapping between characteristics and handles 的定义的参数同步到 ble_app_func.c；

Q：AI 功能使用场景

A：AI 功能针对于小 Q 同学 APP 进行定制开发，如有需要开启 CFG_FUNC_AI 宏定义(app_config.h)；

注意事项：

1) 此功能需要找小 Q 同学运营商申请 APP ID 和 SECRET 相关信息；

2) 开启后对 BLE 广播数据包内容有要求，并会进行修改；

Q：模式切换，不操作 RF 的快速开关蓝牙功能使用方法说明

A：

1) 开启 BT_FAST_POWER_ON_OFF_FUNC 功能和蓝牙后台功能 CFG_BT_BACKGROUND_RUN_EN；

2) 功能开启后，在进入蓝牙模式，调用 BtFastPowerOn 开启蓝牙，进入可被搜索可被连接状态，并进行蓝牙回连；

3) 退出蓝牙模式时，调用 BtFastPowerOff 断开蓝牙连接，并使蓝牙进入不可被搜索不可被连接状态；

Q：模式切换，开关 RF 的蓝牙开关操作使用方法说明

A：

1) 关闭蓝牙后台功能 CFG_BT_BACKGROUND_RUN_EN；
2) 在退出蓝牙模式，调用 BtPowerOff 函数，断开蓝牙连接，复位基带，关闭蓝牙中断，删除蓝牙协议栈任务，关闭蓝牙模块；
3) 进入蓝牙模式，调用 BtPowerOn 函数，使能蓝牙模块时钟，初始化基带和蓝牙协议栈等；

Q：初始化蓝牙后，不立即进入蓝牙可被搜索可被连接状态，也不自动回连的说明

A：

1) 在蓝牙初始化的时候，需要在蓝牙协议栈接收到 MSG_BT_MID_STACK_INIT 消息(bt_stack_service.c)时，将 GetBtManager()->btAccessModeEnable 参数配置为 0，蓝牙会进入不可被搜索不可被连接状态；
2) 可以通过调用 BTSetAccessMode 函数，来修改蓝牙工作状态；

Q：蓝牙发射功率配置

A：在 bt_config.h 中，配置 BT_TX_POWER_LEVEL 参数来调整发射功率；
1) 默认参数为 level22(+6db)；
2) 发射功率最大为 level 23(+8db)；按照 2db 递减；
3) 发射功率过大，可能会导致蓝牙回连会有干扰声的产生；

Q：蓝牙 profile 支持说明

A：

1) A2DP 和 AVRCP 必须要同时开启，由于考虑到 SDK 的使用情况，默认都是支持蓝牙音乐播放，所以必须要开启 A2DP；
2) BLE、HFP、SPP 根据实际的需要进行开启和关闭；
3) 支持不同的 profile，使用的 ram 资源是有区别的，请查看 bt_config.h 中 ram config 的相关说明；

Q：高级 AVRCP 功能说明

A：默认开启高级 AVRCP 功能：BT_AVRCP_ADVANCED
1) 通过高级 AVRCP 功能能及时更新当前的播放状态；
2) 开启 BT_AVRCP_VOLUME_SYNC，支持蓝牙音量同步功能，主要适用于苹果手机的播放音乐音量同步；
3) 开启 BT_AVRCP_SONG_PLAY_STATE，实时的刷新歌曲的播放时间；
4) 开启 BT_AVRCP_SONG_TRACK_INFOR，在歌曲开始播放、歌曲切换，获取歌曲的 ID3 信息；

Q：AEC 相关参数配置

A：参数配置在 bt_config.h 中：
1) MIC 增益根据 MIC 的实际的灵敏度来进行调整：
BT_HFP_MIC_PGA_GAIN：默认值为 level 15(1.5db 左右)
BT_HFP_MIC_DIGIT_GAIN：默认值为 4095(0db)
BT_HFP_INPUT_DIGIT_GAIN：默认值为 4095(0db)，用于调小 DAC 输出的通话声音大小

2) BT_HFP_AEC_ECHO_LEVEL：回音抑制，参数范围：0-5；正常情况下，设为 2 或者 3；
BT_HFP_AEC_NOISE_LEVEL：噪声抑制，参数范围：0-5；默认设为 0，不开启；
3) AEC DELAY 参数：
AEC 算法对 reference 数据和 MIC 数据的时差的要求不能超过 28ms；
通话模式下，samples_per_frame 为 256 时(默认参数)，reference 和 mic 时差在 25ms 左右，BT_HFP_AEC_DELAY_BLK 可配置为 6；
Samples_per_frame 为 128 时，时差在 17ms 左右，BT_HFP_AEC_DELAY_BLK 可配置为 4；
如上时差、delay_blk 参数仅供参考，基于 128PIN 开发板测试；
AEC 功能注意事项：
a. 通话过程中，不能使用转采样功能；
b. MIC 增益相关参数不能太大，如饱和溢出，会导致 AEC 失效；
c. AEC ECHO 的参数配置越大，对回音的抑制效果越好，但是对双工的影响越大；
d. Reference 的数据一定要早于 mic 数据，给到 AEC 音效算法进行处理；否则会导致 AEC 算法失效；
e. 在 AEC 之后的 MIC 数据，尽量不要进行其他的音效处理算法处理，可能会由于算法的原因带入某些杂音；对方听到的声音不干净；

Q：通话时音效算法支持
A：目前在通话时，只支持 AEC 和变调算法；
1) 默认系统开启 AEC 算法；
2) 变调功能，通过 BT_HFP_MIC_PITCH_SHIFTER_FUNC 开启，调用 BtHf_PitchShifterEffectConfig 来改变声调；

Q：HFP 电池电量同步功能
A：通过 BT_HFP_BATTERY_SYNC 开启电池电量同步功能
1) 在 HFP 协议连接成功后，自动同步当前设备的电量到手机端显示；
2) 需要配合 CFG_FUNC_POWER_MONITOR_EN 功能一起使用；

Q：开启 HFP 协议，但是不需要接听电话功能
A：通过 BT_HFP_MODE_DISABLE 开启该功能；
1) 在蓝牙连接成功后，正常的连接 HFP 协议；
2) 但是在呼入、呼出等通话功能开启的时候，设备端都会自动将通路切换到手机端进行操作；

Q：K 歌录音功能说明
A：通过 BT_RECORD_FUNC_ENABLE 开启该功能
1) 适用范围：苹果手机的全民 K 歌等 K 歌软件，安卓手机使用相应 APP 暂时不会需要该功能；
2) K 歌录音时，数据通路是基于 HFP 通话链路基础上；
3) 此功能开启后，建议关闭蓝牙后台常驻功能，或者切换模式开启快速开关功能；避免应用场景相对复杂；

Q：蓝牙自动重连功能

A：首先需要在 bt_config.h 头文件中开启 BT_RECONNECTION_FUNC 功能：
1) 开机回连(BT_POWERON_RECONNECTION)：回连次数(BT_POR_TRY_COUNTS)，间隔时间(BT_POR_INTERNAL_TIME)；
2) 连接丢失后自动重连(BT_BB_LOST_RECONNECTION)：回连次数(BT_BLR_TRY_COUNTS)，间隔时间(BT_BLR_INTERNAL_TIME)；

Q：蓝牙连接成功、断开连接，提示音导入
A：蓝牙协议栈会在蓝牙连接成功和断开连接的时候，将 MSG_BT_STATE_CONNECTED 和 MSG_BT_STATE_DISCONNECT 2 个消息发送到 main_task.c 中，可在消息处理中加入相关的提示音；