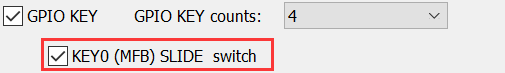
# RTK代码烧录流程

1. 使用McuConfigTool设定MUC相关设定；
   1. 选择IC对应的型号；
   2. 导入对应IC的config bin文件；
   3. 在UI界面设定参数；
   4. 调整完毕后通过“export 🡪 save as”保存成 .rcfg文件；
2. 使用DspConfigTool设定DPS相关设定；
   1. 选择IC package型号；
   2. 调整各项参数或导入其他bin文件的设定；
   3. 通过“save config”将结果保存为bin文件；
3. 使用MPPackTool打包所有配置文件；
   1. 将原厂提供的4个image文件和前两步生成的配置文件放在同一目录；
   2. 第一次打开软件时选择“RTL8763 V1”;
   3. 通过“Browse”按键选择所有配置文件；
   4. 勾选左下角的“save path”；
   5. 点击左下角的“. . .”设定最终生成的Bin文件路径；
   6. 点击“confirm”生产Bin文件；
4. 使用MPPGTool烧录软件到IC上；
   1. PC通过连接板与IC相连；
   2. 在“MP setting”中设定第三步打包生成的Bin文件路径，然后点击lock；
   3. 设定蓝牙地址；（可选）
   4. 取消勾选“读回并合并”，该项会将蓝牙上存储的地址、频偏信息读回并存储在Tool中。
   5. 在“MP download”中选择“All Erase”将IC flash擦除；
   6. 将IC断电后重连；（新版Tool不需要）
   7. 点击“download”下载程序到IC上；

# McuConfigTool使用

## HW feature

1. 设定是否为滑动开关，如果是滑动开关的话会依据key0电平决定开关机而不是按键时间。

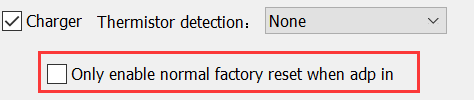


1. 如果将开关机设定为GPIO模式，则当设定的Pin脚上是低电平IC会自动关机，高电平时开机。





1. 如果勾上了这个选项，只有在充电的时候才允许复位；



## General

设定产品的一般信息。

* Device name

设定蓝牙设备在用户界面上显示的名称。

* Class of device (CoD)

格式类型字段的长度为变量，以不同于“11”的两位数结束。版本字段从 CoD 的最低有效位开始，并向上延伸，每位分别代表一个分类，可设定多类。

* Power on/off

设定开关机的按键设置。

* Link back and pairing

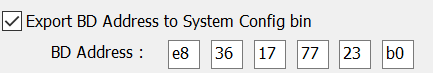
设置回连（连接之前连接的蓝牙设备）与配对超时设定。

* Multi-link

设定是否支持同时连接多个链路与回连、意外断开时的处理方式。

## System configuration

1. 这个地址只有在单个文件烧录（debug）时会用到，MP烧录时的地址以Tool中的为准；



## Legacy

设定相关协议的支持与具体设定。

## LE

设定低功耗LE相关设定。

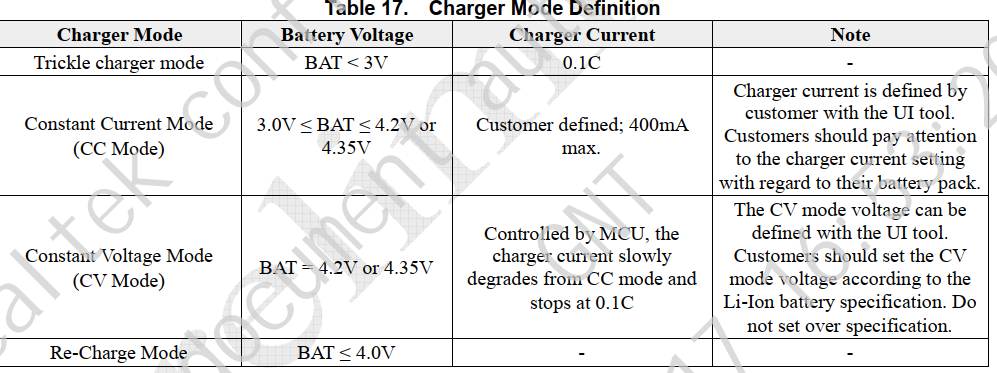
## CODEC DSP

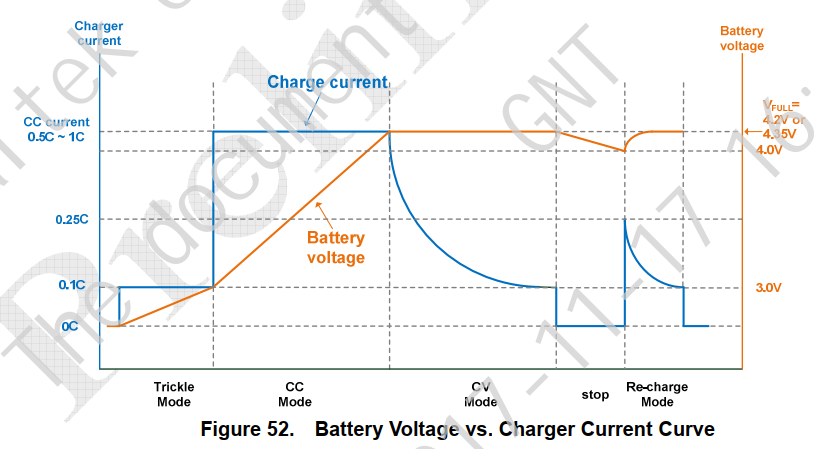
1. 如果耳机需要单独使用，则设为L+R/2，就可以让每个声道同时输出两个声道混合的声音；



## Charge

1. Pre-charge定义trickle mode阶段超时时间，如果超过则会停止充电。
2. Fast-charge定义CC mode阶段的超时时间，如果超过则会停止充电。
3. “reference battery resistance”定义了电池阻抗，不匹配可能会导致电池充不满电，如果没有提供就用默认值。
4. 充电模式如下：



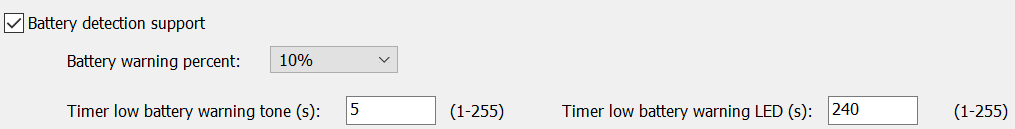


注意：图中C表示电池实际的毫安时，如1200mAh的1C表示1200mA。

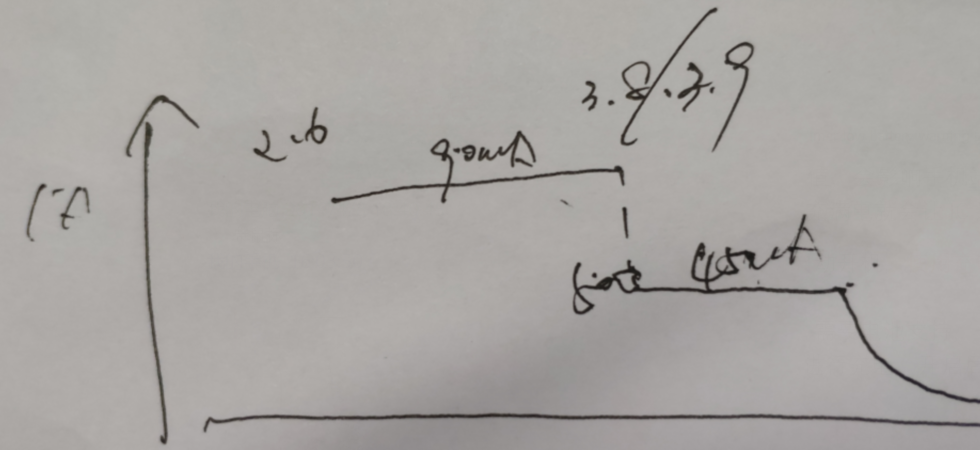
1. 温度保护设定效果如下：



1. Battery detection support定义了电量报警的提示方式，下发的选项是提示的间隔，每到一个提示周期就会触发一次提示，之后会恢复原状态的LED等设置，所以无法设定当达到某个电压后一直处于某种LED状态中。



1. Reference battery voltage定义了电池的电压阈值，会影响手机端显示的蓝牙电量，一般“0%”要设为3100以上，因为大部分电池电压低于3.1V会损伤电池，一般电池电压是2.6V ~ 4.2V。
2. “enter power down mode…………”开启后，如果充满电就会让IC进入极低功耗状态，此时充电电流约为4uA左右，拔出充电器后会自动开机，使用充电盒的设备默认都会开启，否则可能会耗尽充电盒的电量。
3. “rapid charge enable”用于开启急速充电功能，需要电池支持，一般电池只能支持最大1C的电流，特别的电池可以用大于1C的电流进行充电。开启后会以下面的曲线代替fast-charge阶段的曲线。



1. “current of fast-charger”一般设为0.8C或者50mA。

## Peripheral

设定UART、NFC、Buzzer、USB相关设定。

## Ringtone

设定各种情况下的提示音，可用内置的提示音也用自制的音频文件作为提示音。

主要事件情况：

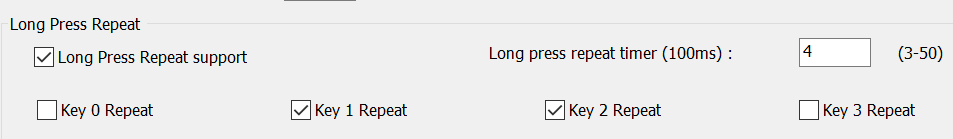
1. Power on ：开机时
2. Power off：关机时
3. Pairing：开始配对时
4. New device connected：在手机或PC连接后触发

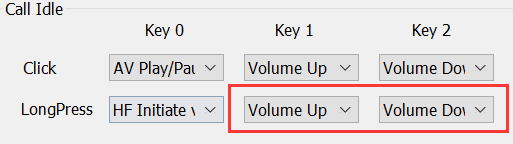
## KEY

1. 如果将active电压会改变该Pin悬空时的电平。
2. 开启该选项后会使长按功能在长按达到时间后立刻触发，否则会在松手后触发；



1. 如果开启Long press repeat功能，则需要设定对应的Long press动作。

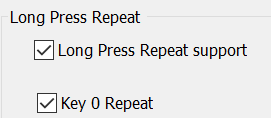




1. EQ效果切换可以通过Audio effect next/previous调整。



1. 因为工厂生产时会对每个板子进行测试功能是否正常，所以需要进入配对和清除配对（Reset）功能。一般会在“call idle”中额外添加一个按键，短按时进入配对，长按时Reset，在“call active”中添加进入配对功能断开测试设备。
2. 如果同一个按键同时开启长按重复和超长按功能，会导致长按重复失效。





* 常见情况：

1. Call idle：播放音乐时的行为；
2. Voice dial：语言拨号界面，如Siri。
3. Incoming call：接到外部来电时；
4. Outgoing call：向外拨打电话时；
5. Call active：通话中，不论接听还是拨打；
6. Call active with call waiting：正在通话中有第三方电话呼叫时；
7. Call active with call hold：正在通话中，而且有第三方来电已经被保留；
8. Mutil-link call active with call waiting：多方链接正在通话中时有其他来电等待处理；
9. Multi-link Call Active with Call Hold：多方链接正在通话中时有其他来电已经被保留；

* 常见动作：

1. AV Play/Pause：播放/暂停音乐；
2. AV Stop：停止播放音乐；
3. AV forward：前一首音乐；
4. AV backward：下一首音乐；
5. Mic mute toggle：
6. HF initial voice dial：将切换至voice dial环境；
7. HF cancel voice dial：将离开voice dial切换至call idle；
8. Switch next voice prompt language：切换至下一个语言的提示音；
9. Normal factory rest：设置复位；
10. Legacy enter pairing mode：进入配对模式；
11. Audio effect next/previous：切换EQ音效；

## LED

1. LED行为：（on duty + off duty） \* blink count + interval
2. 如果通过LED界面作为外设的电源控制端，那么在充电时最后将外设的电源关掉。
3. 长时间的闪烁行为如果没有要求的话，最好在将关闭的时间拉长些，可以省电和缓解视觉疲劳；

* LED设置惯例：

关机：红灯

开机：蓝灯

低电压：闪红灯

已连接：闪蓝灯

配对中：交替闪烁

充电中：红灯

满电：蓝灯/不亮，如果有充电仓默认设为不亮

通话/播放音乐：设置同一种灯

* 常见状态：

1. Connected single link：两个蓝牙设备单独链接时触发；

## RWS

RWS如果不存在配对信息（Reset mode），那么在开机瞬间会先进入配对模式寻找相同的RWS设备，否则会回连之前的设备，连接后才会跑到其他状态。

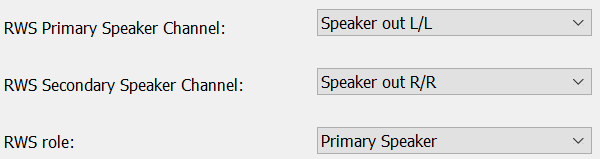
1. 设定开始时主耳和副耳之间配对或回连的时间；



1. 设定主耳和副耳之间的配对ID，只有ID一致的才能连接；



1. 设置设备在RWS中的扮演的角色和输出的声道，主耳和副耳要单独设定；



1. 默认情况下关闭主耳的电源也会关闭副耳的电源，但是关闭副耳的电源不会影响主耳的，如果勾选这项的话关闭副耳的电源也会同时关闭主耳电源；



1. 默认情况下只有主耳有语言提示，勾选后副耳也会播放语音提示；



1. 副耳断开和主耳的连接后是否进入配对模式，默认不进入；

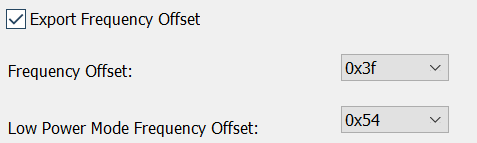


1. 设置副耳是否能被手机搜索到，勾选后副耳如果没和主耳连接也不会自动进入配对模式，

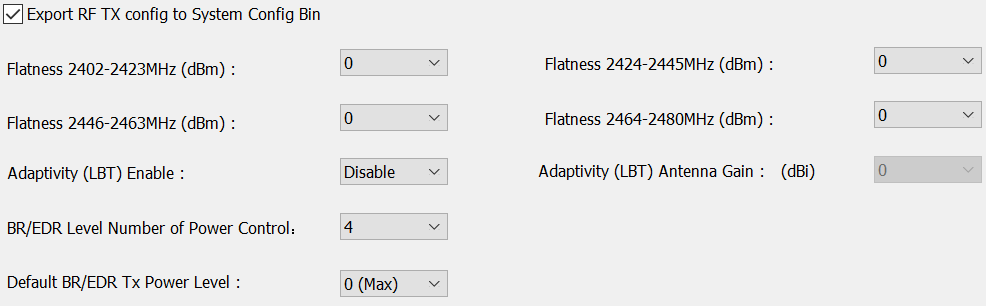


## RF TX

1. 一般蓝牙有两个晶振，主晶振约40MHz（外挂），副晶振约32.768KHz（一般内置），低功耗状态下会使用副晶振工作。两个选项分别设置主、副晶振的频偏。

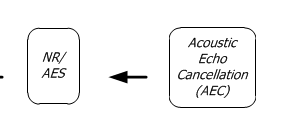


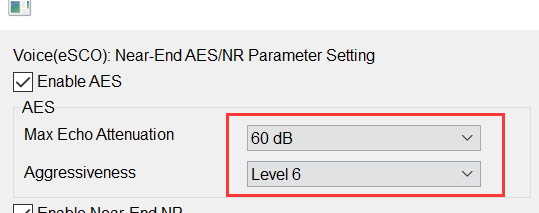
1. 设置蓝牙在2.402M~2.480M之间的频率偏移补偿，一般使用默认值；

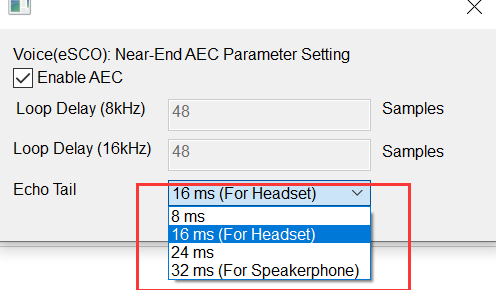


# DSP工具使用

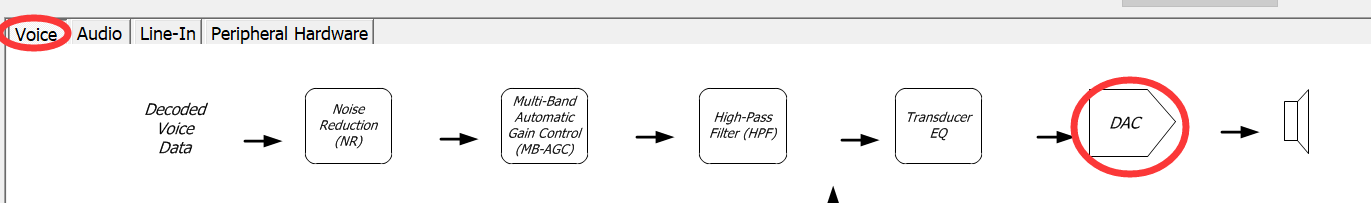
1. 遇到回音问题可以调整DSP中的下面两个，将数值拉高。

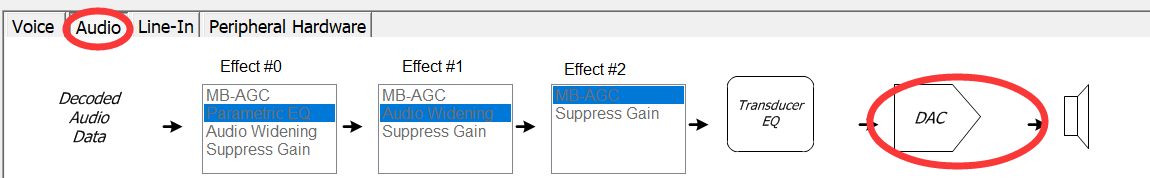


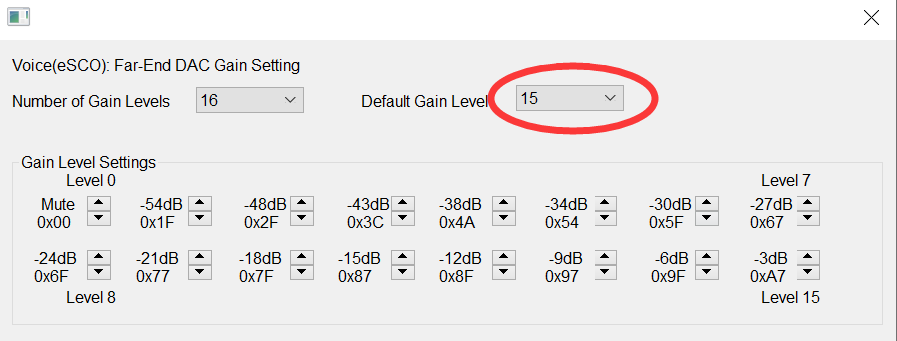




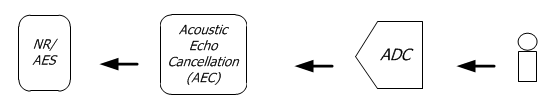
1. 如果耳机没有音量调节功能的话要把DSP中的默认音量拉到最大，不然安卓手机有可能没办法把蓝牙声音调大，苹果有音量同步则没问题；





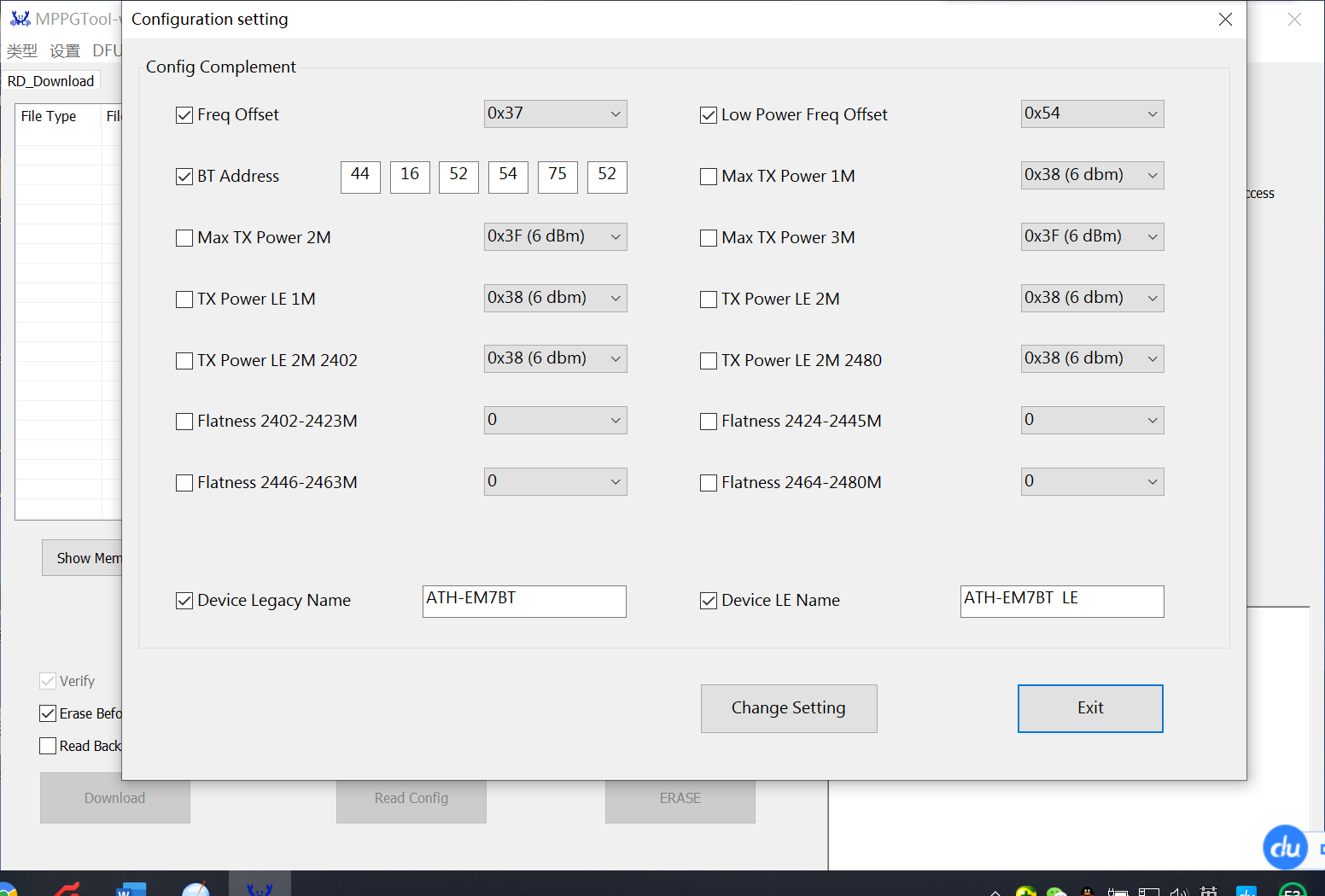


1. 如果遇到音色问题，主要调整下面三个地方，回音抑制和降噪开大了也会影响音质；



# RTK 注意事项

1. power key默认为key0，如需使用其他按键作为power key只需将其映射为key0即可。
2. 充电前先确认电池电压，大于3.7V可以先不充电，小于3.7V时可以按4.2V 1.2A设定电源充电，待电流下降至600~700mA即可停止。
3. 如果关闭里按键控制快关机功能就需要外部充电盒来唤醒开机。
4. RTK的IC时依据上电时P20 Port的电平来决定是否进入烧写模式，所以需要让P20为0.
5. IC在待机状态下（没连上其他设备也没处于pairing mode），触发“AV Play/pause”会自动回连上一个设备。
6. IC默认存储有8个已配对设备的信息，且新设备接入后会替代掉最旧的那个设备信息。
7. 一个蓝牙耳机最多可连接2个手机。
8. RWS耳机中一般只需要通过LE控制主耳，所以主耳和副耳的LE名称要注意有区别。
9. 普通蓝牙（BR）和低功耗蓝牙（LE）的地址是不一样的，RTK IC默认无法修改LE地址。
10. “normal factory Reset”只有在没有连接任何设备（包括对耳）时才会有效。
11. 软件更新后一定要清除配对记录（Reset），否则会导致无法回连手机。
12. OTA升级无法更新“system configurtion”、“Charge”、“RF TX”部分内容；
13. 要检测IC是否有烧录过程序可以直接使用烧录软件在调试模式下读取配置内容，如果地址和名称都是正常的，那么就是已烧录软件的IC；



1. 一般会在电池负极串联一个小于1.5Ω的电阻用于分压测量电流，如果电阻大于1.5Ω可能会导致无法充电。

# RTK Tool Bug

1. “battery detection support”最好都要勾上，因为目前UI Tool有bug，如果不勾选会导致充电仓离仓后无法自动开机。
2. 在TWS(RWS)配对失败后，没和其他手机连接时（包含pairing mode），如果触发“AV Play/pause”或者“HF initial voice dial”会使蓝牙重新进入TWS(RWS)配对状态。
3. KEY0的“long press repeat”与按键关机是互斥功能（V2.18可同时存在，但存在bug），如果同时开启（V2.18）会导致主耳关机时无法主副同时关机；

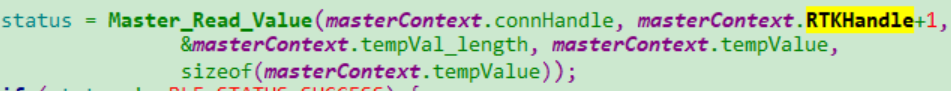


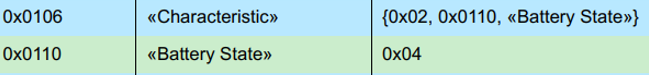
1. 即时开启了主副耳响铃同步，有些

# ST IC编程相关

## 基本知识点

1. 控制设备的API主要存放在bluenrg1\_api.h中，可以直接传参数调用；
2. 底层收到信息后会以事件的形式传递到应用层，直接重写bluenrg1\_events.h中的函数即可实现；
3. bluenrg1\_types.h 中定义了各种GATT命令需要用到的数据包格式，可以配合Core spec定义；
4. 程序路径不要有中文，不然生成Bin档会有问题；
5. 因为特性按照 UUID 🡪 值 🡪 通知/提醒（如果有）的顺序存放在属性表中，所以通过Master\_GetCharacOfService获得的Handle是指向UUID的，要读取特性的值需要+1；





1. 蓝牙时间片基本流程：
   1. 检查HCI RX Queue是否有已收到的数据包；
   2. 如果收到了HCI Event 包，则进一步判断事件类型；
   3. 如果是LE事件则调用hci\_le\_meta\_events\_table中对应函数，如果是供应商（vendor）事件则调用hci\_vendor\_specific\_events\_table中对应函数，其他事件调用hci\_events\_table中的函数。
2. 一般蓝牙IC建立连接流程为：
   1. 建立连接；
   2. 进行加密认证；
   3. 更新连接信息；
   4. 设置一些参数？？
   5. 开始通信；
3. GATT数据库长度计算：
   1. GATT Service Records：因为GAP和GATT必须包含，所以最少为2；
   2. GATT Attribute Records：GAP and GATT共有9个属性，所以最少为9；

计算规则：如果属性包含通知、指示、广播、扩展特性集时数量分别加1，所以一个特性需要2~5个字节；

* 1. Value Array：GAP和GATT共44个，最少为44；

计算规则：

1. 如果特性的UUID时16 bit的，则+5，如果是128 bit的，则+19；
2. 如果一个特性添加了一个服务配置描述符，则+2；
3. 每同时存在一个连接，如果一个特性添加了一个客户配置描述符，则+2；
4. 每有一个具备扩展性的特性，则+2；

所以每个特性应该怎加5 ~ 25个字节；

示例：

一．



GATT Attribute Records = 2（声明+数值） + 1（notify） = 3

Value Array = 19（UUID） + 20（Char length） + 2（notify） = 41

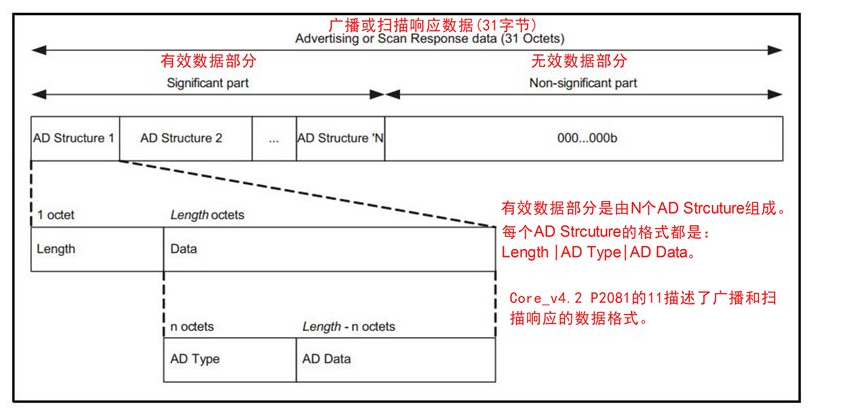
二．



GATT Attribute Records = 2（声明+数值）

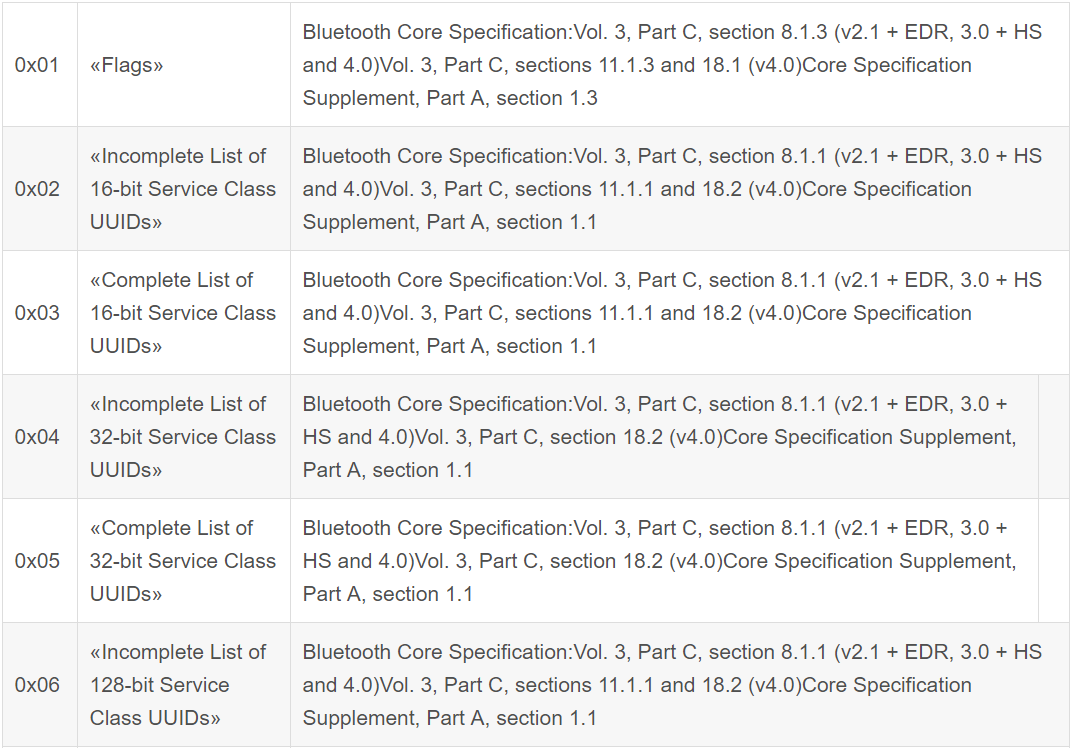
Value Array = 19（UUID） + 20（Char length）= 39；

广播包格式：



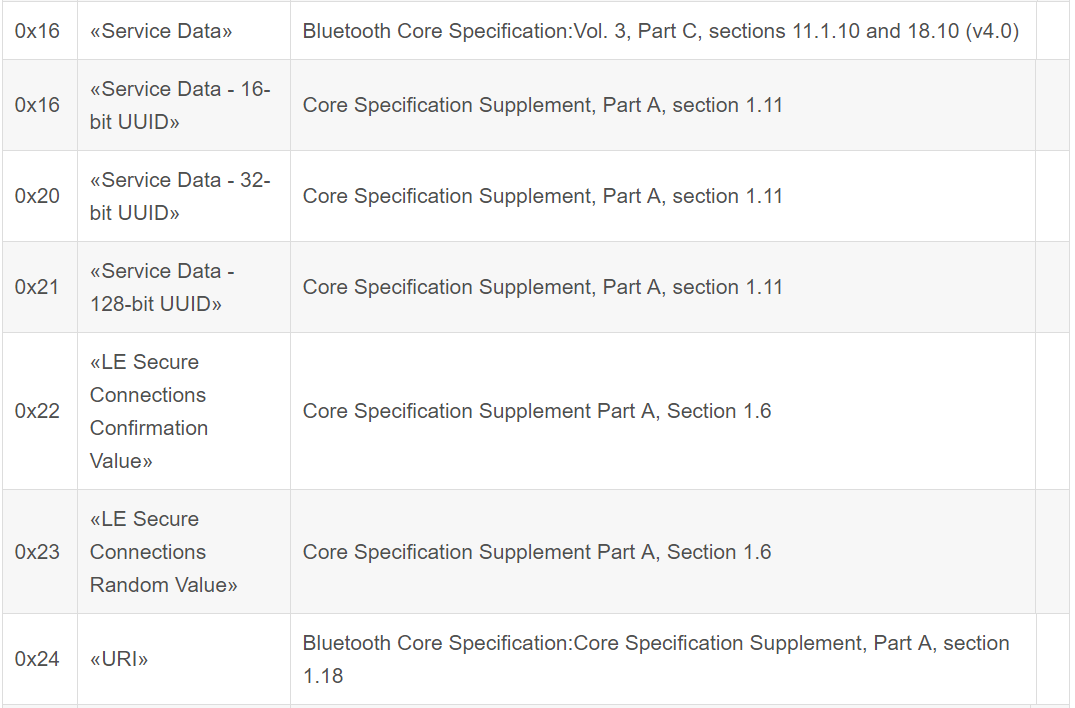
有效数据部分：包含N个AD Structure，每个AD Structure由Length，AD Type和AD Data组成。其中：

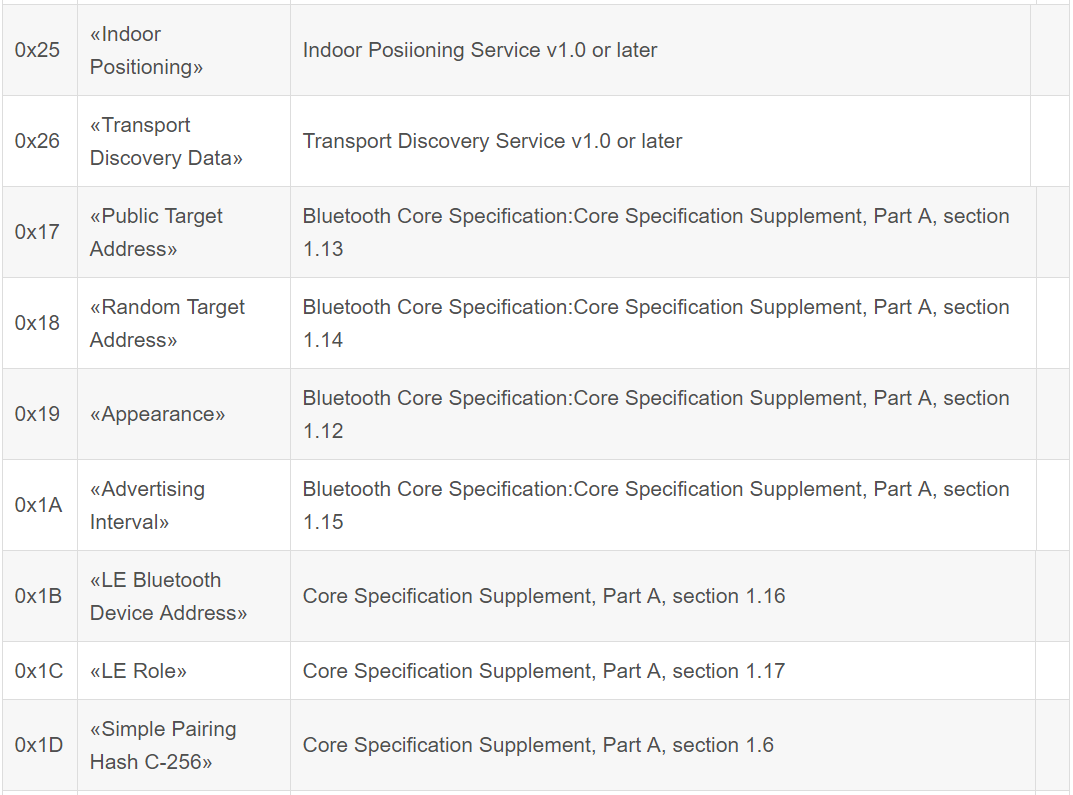
* Length：AD Type和AD Data的长度。
* AD Type：指示AD Data数据的含义

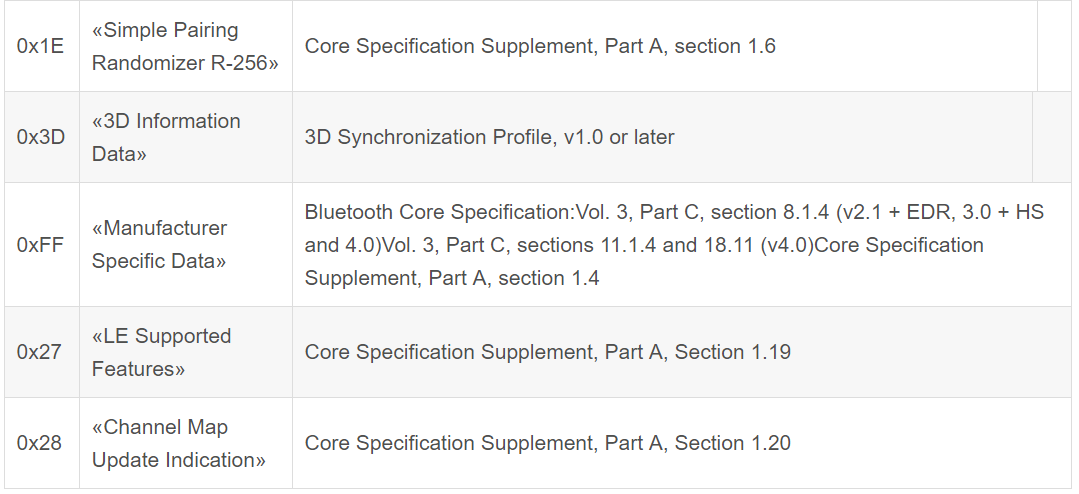












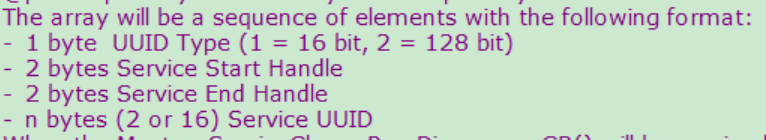
## Security Master central程序流程

* Main函数流程

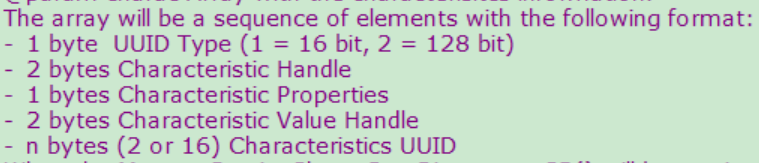
1. 初始化系统时钟、串口、蓝牙栈；
2. 在deviceInit中初始化masterContext，用于标识当前主机状态，用于在APP\_TICK中作判断条件。调用Master\_Init初始化设备地址、名称并作为主机。
3. 在Device\_Security中初始化安全数据库，通过Master\_SecuritySet设置相关安全参数；
4. 初始化按键中断，用于触发中断时断开连接；
5. 调用deviceDiscovery开始设备扫描，扫描结束或者被中止后会触发Master\_DeviceDiscovery\_CB函数；
6. 开始while死循环：
   1. 在BTLE\_StackTick中处理HCI层提交上来的Event；
   2. 在APP\_Tick中依据masterContext的各个成员进行相关的操作，共包含
   3. 调用Master\_Process获取当前处理器电源状态，并设定低功耗模式唤醒条件；

* 蓝牙连接流程

1. Master\_DeviceDiscovery\_CB中通过名称找到要连接的设备后将设备地址传递给deviceConnection进行连接；
2. deviceConnection中配置连接参数（扫描时间、连接间隔、连接延迟、超时时间等），通过Master\_DeviceConnection进行连接；
3. DeviceConnection执行后会回调Master\_Connection\_CB；
4. Master\_Connection\_CB依据结果执行不同分支，正常情况下masterContext.isconnected被置1；
5. 此时客户端应该发送security request给服务器（调用aci\_gap\_slave\_security\_req）.
6. 服务器接收到security request后会自动处理完成链路加密，并触发回调函数Master\_library\_encryption\_change\_event\_CB，置位masterContext.encryption\_enabled\_OK。
7. 在APP Tick中调用start\_primary\_service\_discovery。
8. 在start\_primary\_service\_discovery中调用Master\_GetPrimaryServices，开始获取主要服务；
9. Master\_ServiceCharacPeerDiscovery\_CB被触发，在primaryServiceFound中通过比对UUID方式查找服务，并置位masterContext.findCharacOfService；
10. 在APP Tick中执行findCharcOfService，依据UUID找出GATT或者外设服务的相关信息，并专递给Master\_GetCharacOfService；



1. Master\_ServiceCharacPeerDiscovery\_CB再次被触发，此时执行“GET\_CHARACTERISTICS\_OF\_A\_SERVICE” Case，并调用extractCharacInfo；
2. 在extractCharacInfo中通过特性的UUID找到特性的句柄，置位masterContext.enableNotif；



1. 在APP TICK中调用enableSensorNotifications;
2. 之后通过Master\_NotifIndic\_Status开启通知状态，触发回调函数Master\_PeerDataExchange\_CB，此时状态为“NOTIFICATION\_INDICATION\_CHANGE\_STATUS”，并最终置位masterContext.read\_char；
3. masterContext.read\_char被置位后在APP TICK中每次“RX\_CHAR\_TIMEOUT”就通过readRXChar读取一次外设RX特性；

## 从Master切换至Broadcast流程

1. 先断开现有的连接；
2. 调用hci\_reset，重置HCI层；
3. 重新设定发射功率、GAP、GATT协议；

## Event

### 服务器属性被修改

void aci\_gatt\_attribute\_modified\_event(uint16\_t Connection\_Handle,uint16\_t Attr\_Handle,uint16\_t Offset,uint16\_t Attr\_Data\_Length,uint8\_t Attr\_Data[]);

## 常用API

### 写入操作

#### 设置蓝牙地址

tBleStatus aci\_hal\_write\_config\_data(uint8\_t Offset,uint8\_t Length,uint8\_t Value[]);

#### 初始化GATT

tBleStatus aci\_gatt\_init(void)

#### 初始化GAP

tBleStatus aci\_gap\_init(uint8\_t Role,uint8\_t privacy\_enabled,uint8\_t device\_name\_char\_len,uint16\_t \*Service\_Handle,uint16\_t \*Dev\_Name\_Char\_Handle,uint16\_t \*Appearance\_Char\_Handle)

#### 创建连接

tBleStatus aci\_gap\_create\_connection(uint16\_t LE\_Scan\_Interval,

uint16\_t LE\_Scan\_Window,

uint8\_t Peer\_Address\_Type,

uint8\_t Peer\_Address[6],

uint8\_t Own\_Address\_Type,

uint16\_t Conn\_Interval\_Min,

uint16\_t Conn\_Interval\_Max,

uint16\_t Conn\_Latency,

uint16\_t Supervision\_Timeout,

uint16\_t Minimum\_CE\_Length,

uint16\_t Maximum\_CE\_Length);

#### 终止连接

tBleStatus hci\_disconnect(uint16\_t Connection\_Handle,uint8\_t Reason);

会触发：Master\_Connection\_CB

#### 设置HCI事件屏蔽

tBleStatus hci\_set\_event\_mask(uint8\_t Event\_Mask[8]);

#### HCI链路层复位

tBleStatus hci\_reset(void);

#### 修改属性值

tBleStatus aci\_gatt\_update\_char\_value\_ext(uint16\_t Conn\_Handle\_To\_Notify,

uint16\_t Service\_Handle,

uint16\_t Char\_Handle,

uint8\_t Update\_Type,

uint16\_t Char\_Length,

uint16\_t Value\_Offset,

uint8\_t Value\_Length,

uint8\_t Value[]);

#### 修改描述符

tBleStatus aci\_gatt\_write\_char\_desc(uint16\_t Connection\_Handle,

uint16\_t Attr\_Handle,

uint8\_t Attribute\_Val\_Length,

uint8\_t Attribute\_Val[]);

#### 设置扫描响应数据包

tBleStatus hci\_le\_set\_scan\_response\_data(uint8\_t Scan\_Response\_Data\_Length,

uint8\_t Scan\_Response\_Data[31]);

#### 设置可发现性

tBleStatus aci\_gap\_set\_discoverable(uint8\_t Advertising\_Type,

uint16\_t Advertising\_Interval\_Min,

uint16\_t Advertising\_Interval\_Max,

uint8\_t Own\_Address\_Type,

uint8\_t Advertising\_Filter\_Policy,

uint8\_t Local\_Name\_Length,

uint8\_t Local\_Name[],

uint8\_t Service\_Uuid\_length,

uint8\_t Service\_Uuid\_List[],

uint16\_t Slave\_Conn\_Interval\_Min,

uint16\_t Slave\_Conn\_Interval\_Max);

#### 设置广播数据

tBleStatus hci\_le\_set\_advertising\_data(uint8\_t Advertising\_Data\_Length,

uint8\_t Advertising\_Data[31]);

### 读取操作

#### 读取蓝牙地址

tBleStatus hci\_read\_bd\_addr(uint8\_t BD\_ADDR[6]);

#### 读取传输功率等级

tBleStatus hci\_read\_transmit\_power\_level(uint16\_t Connection\_Handle,uint8\_t Type,uint8\_t \*Transmit\_Power\_Level);

#### 读取本地控制器版本

tBleStatus hci\_read\_local\_version\_information(uint8\_t \*HCI\_Version,uint16\_t \*HCI\_Revision,uint8\_t \*LMP\_PAL\_Version,uint16\_t \*Manufacturer\_Name,uint16\_t \*LMP\_PAL\_Subversion);

#### 读取本地控制器支持的HCI命令

tBleStatus hci\_read\_local\_supported\_commands(uint8\_t Supported\_Commands[64]);

#### 读取本地控制器支持的LMP特性

tBleStatus hci\_read\_local\_supported\_features(uint8\_t LMP\_Features[8]);

#### 读取控制器接收信号强度

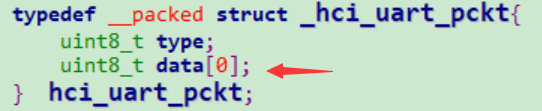
tBleStatus hci\_read\_rssi(uint16\_t Connection\_Handle,

#### 读取远程设备版本信息

tBleStatus hci\_read\_remote\_version\_information(uint16\_t Connection\_Handle);

# 程序注意点

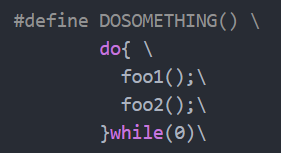
1. 如果在结构体的中的最后一项定义一个长度为0的数组，如果使用sizeof测量结构体则该项可以忽略，如果使用malloc申请，就申请了一段长度为结构体长度加可变长度的内存空间给结构体类型的指针，这段可变长度的内存空间和前面的结构体长度的内存空间是连续的。



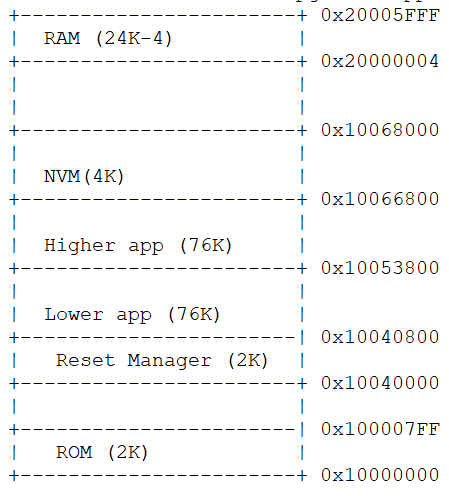
1. \_\_packed\_\_ 前标要求编译器进行1字节对齐（连续存放）。



1. do/while(0)的作用是在宏定义中调用多条语句时保证调用的正确；

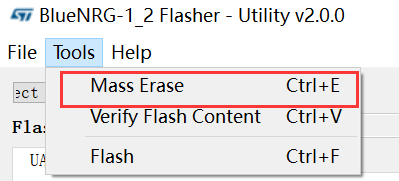


# 发布BIN档生成步骤



**图. BlueNRG-1 内存映射图**

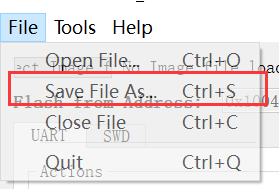
1. **打开“BlueNRG-1\_2 Flasher Utility”导入软件Bin档；**
2. **擦除Flash；**



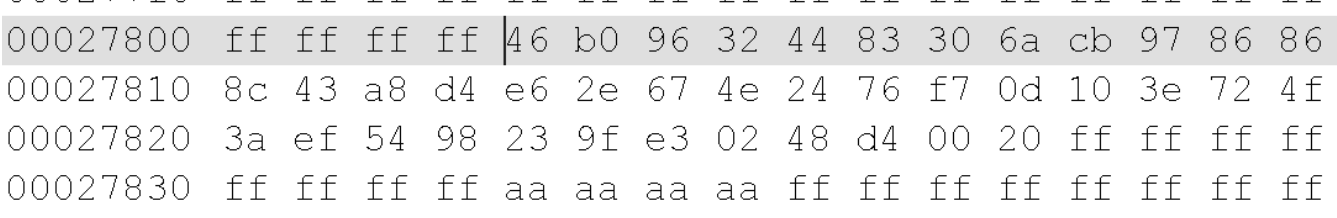
1. **烧录到IC中；**
2. **设置读取Size为0x28000，点击Read;**



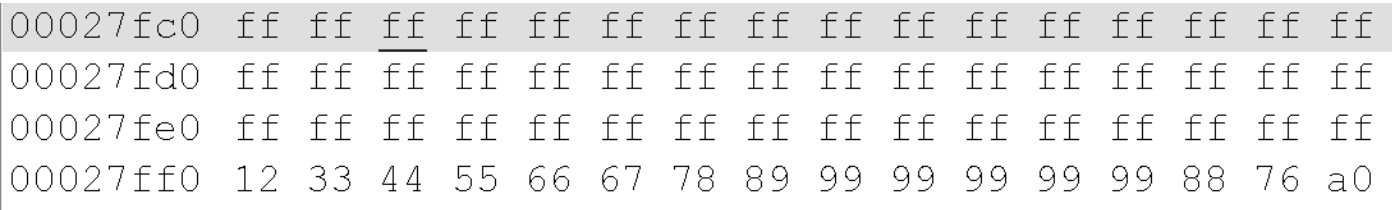
1. **将Bin文件另存到其他位置；**



1. **转到0x00027800处将非FF的数据全改为FF，此处存放的是IC运行过程中产生的密钥，只要下面“aa aa aa aa”处是FF，每次IC运行都会随机生成密钥，为了让每个板子的密钥不同，此处要改为ff；**

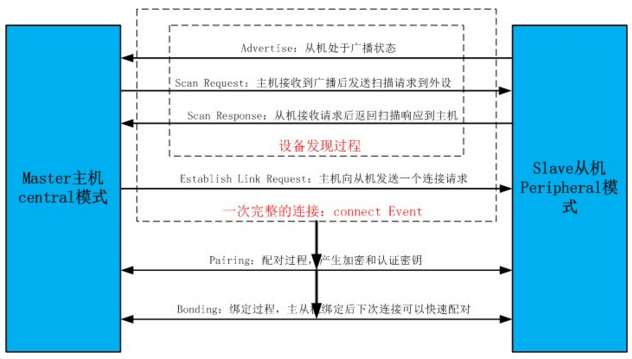


1. **将结尾处最后一行的FF改为其他内容；**



# 基本知识点

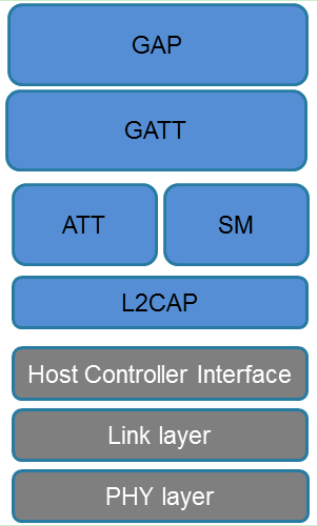
## 蓝牙连接过程



## 服务、特征、属性关系

一个服务（service）可以包含多个特征（character），一个特征可以包含多个属性（attribute）。

## BLE结构



## 蓝牙模块主要组成

* 1. UI包含：HSP（音频）、HFP（音频）、A2DP（音频）、AVRCP（控制）、SPP（数据）、BLE（数据）。

其中SPP在iPhone上使用时需要添加MFY认证IC，所以在iPhone上多用BLE传输数据文件。

* 1. Audio包含：通话、音乐两大类。



* 1. RF包含：功率、频偏两类。

功率部分蓝牙主要工作在class2（-6db ~ 4db），频偏主要通过调整IC内部电容以补偿频率偏移。

## RF主要测试内容

gain、offset、sensitive。Sensitive表征蓝牙模块能接到到的最小信号的能力，灵敏度越高，能识别的信号幅度越小。

## 蓝牙工作频段

在2.4GHz ISM频段上，不同国家或地区会有些不同，我国的蓝牙频率在2.402GHz～2.483GHz,蓝牙每个频道的宽度为1MHz，为了减少带外辐射的干扰，保留上、下保护为3.5MHz和2MHz，79个跳频点中至少75个伪随机码跳动，30S内任何一个频点使用时长不能超过0.4S。



## **发射功率**

蓝牙是能工作在以下三种功率级下的短距离无线网络技术：

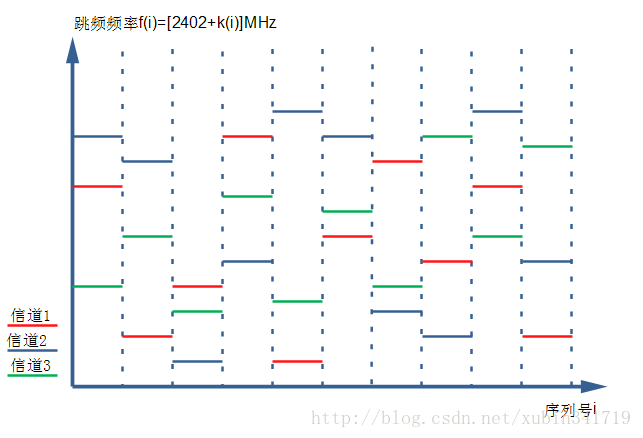
功率级1(最高功率电平+20dBm，有效范围100m), 即0.1W 是GSM手机最大功率的1/20。

功率级2(最高功率电平+4dBm，有效范围20m), 即2.5mW 是蓝牙耳机常用功率等级.

功率级3(最高功率电平0dBm，有效范围10m)。 即1mW是GSM手机最大功率的1/2000。

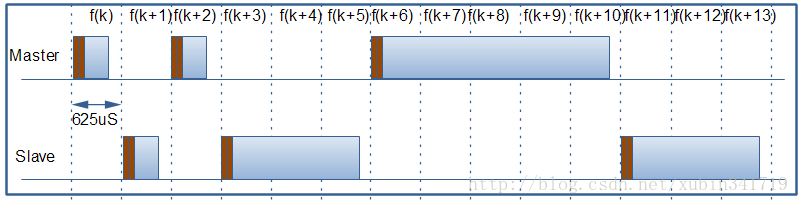
## **物理信道**

蓝牙物理信道有伪随机序列控制的79个跳频点构成，不同跳频序列代表不同的信道。



## **时隙**

蓝牙跳频速率为1600次/s,每个时间为625uS(1S/1600)称为一个时隙。



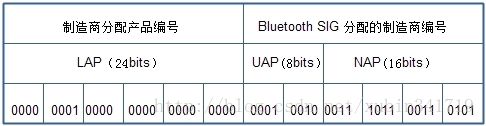
## **蓝牙地址**

BD\_ADDR：BluetoothDevice Address；

LAP:LowerAddress Part 低地址部分，是制造商可以自己分配的；

UAP: UpperAddress Part 高地址部分，由SIG分配；

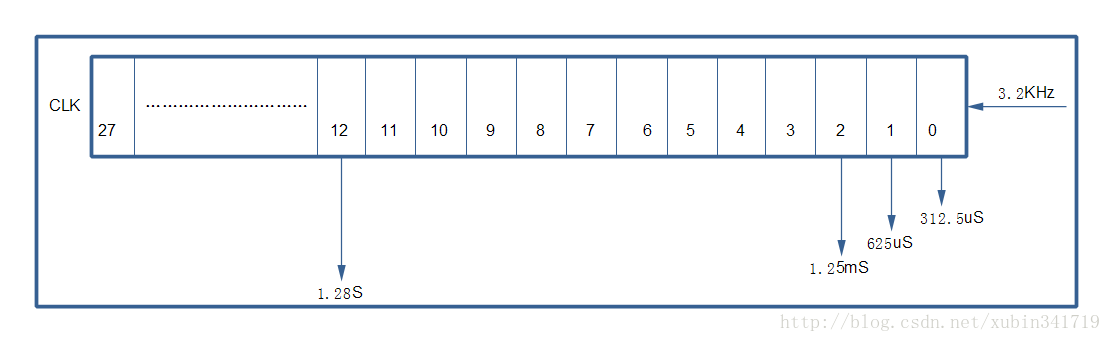
NAP: Non-significantAddress Part 无效地址部分，由SIG分配。

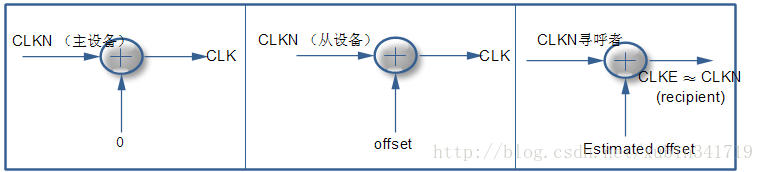


## **蓝牙时钟**

每个蓝牙设备都有一个独立运行的内部系统时钟，称为本地时钟（Local Clock），决定定时器的收发跳频。为了与其他设备同步，本地时钟要加一个偏移量（offset），提供给其他设备同步。

蓝牙基带四个关键周期：**312.5uS、625uS、1.25mS、1.28S**。



**CLKN：**本地时钟：  
**CLKE:**预计时钟，扫描寻呼过程中用到；  
**CLK：**设备实际运行的时钟频率。  
CLKE、CLK由CLKN加上一个偏移量得到的。  


## **蓝牙物理链路**

通信设备间物理层的数据连接通道就是物理链路。

ACL（Asynchronous Connectionless）异步无连接链路；对时间要求不敏感的数据通信，如文件数据、控制信令等。

SCO（Synochronous Connection Oriented）同步面向连接链路；对时间比较敏感的通信，如：语音；最多只支持3条SCO链路，不支持重传。

ACL用于数据传输；

## **蓝牙基带分组**

基带分组至少包括：接入码、分组头、有效载荷。



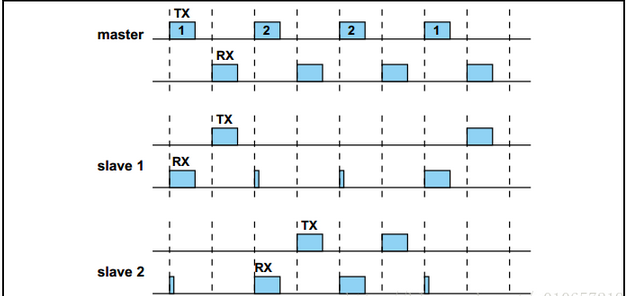
## 链路控制器状态

1. 寻呼page： 向远端蓝牙设备发起蓝牙连接请求。
2. 寻呼扫描page scan：蓝牙设备开启page scan模式，才能够相应其他蓝牙设备的连接请求。例如蓝牙耳机只有处于page scan才允许其他设备来连接。
3. 查询inquiry：开始发起搜索周围的蓝牙设备
4. 查询扫描inquiry scan：蓝牙设备开启inquiry scan模式，才能够被别的蓝牙设备搜索到。
5. 主设备相应Master Response：master收到slave response的msg后，他就会进入到master response的状态，同时他会发送一个FHS的packet。
6. 从设备相应Slave Response：这个就是在page的过程中，slave收到了master的page msg，它会回应对应的page response msg，同时自己就进入到了slave response的状态。
7. 查询相应inquiry response：就是在inquiry scan的设备在收到inquiry的msg后，就会发送inquiry response的msg，在这之后它就会进入到了inquiry response的状态了。

## 蓝牙连接状态

1. 激活模式active

在主动模式下，在微微网内部所有的从设备都可以和主设备通信，最多只能有七个从设备。所有的通信都有主设备来主导。微微网所有的从设备都会在主设备-> 从设备时隙上监听数据包。如果一个从设备没有被寻址，它将等待下一个数据传输。从设备能从主设备传输的包头获取传输占用的时隙，在此期间没有被寻址的设备将会等待传输时隙。具体可以查看下图,多从设备传输时序图：



1. 呼吸模式sniff

如果在主动模式下，从设备要时刻监听主设备发送过来的数据包，但是在Sniff模式下不需要，从而降低设备的功耗。在sniff模式下主设备将每隔Tsniff向从设备发送数据包，所以每隔Tsniff去监听主设备的数据包即可！slave就是只在下图中所示的sniff anchor point时监听。sniff mode只能应用于异步传输，不能应用于同步逻辑传输。

1. 保持模式hold

从机和主机协商一个保持时间，在此期间从设备进入低功耗模式但仍然保持LT\_ADDR。异步传输在此模式下，不响应当然微微网的任何数据包。但在同步传输模式下(SCO,eSCO)需要支持保留时隙的数据包。在此模式下的设备可以scanning, paging, inquiring, 或者加入其它的微微网。

1. 休眠模式park

当一个从设备不需要一直参与微微网的信道，但是却需要保持跟信道的同步时，从设备可以进入Park state。处于Park state状态下的从设备只有很少的活动，它会选择放弃它的逻辑传输地址 LT\_ADDR,取而代之的是，它会使用两个新的地址：

PM\_ADDR: 8-bit Parked Member Address

AR\_ADDR: 8-bit Access Request Address

另外，为了达到低功耗的目的，在Park state下，一个master可以连接多于七个slaves. 在一时刻， 只有七个slaves可以开始连接状态。但是，通过交换active slave 和park slave, 在微微网中的slaves的数目可以达到更多（255 如果PM\_ADDR被使用的时候，并且如果用了BD\_ADDR,这个数目还可以随机大）；

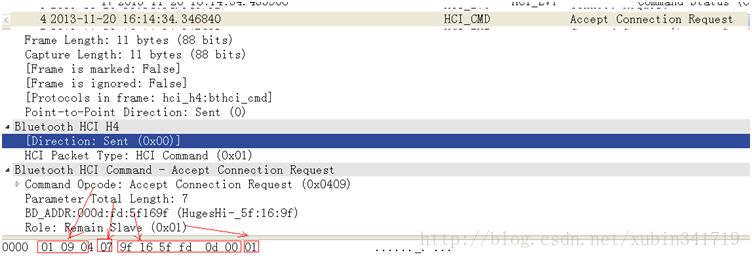
1. 待机状态

待机状态是蓝牙设备缺省低功耗状态，此状态下本地时钟以低精度运行。蓝牙从待机转入寻呼扫描状态，对其他寻呼进行响应成为从设备；也可以从待机状态进入查询扫描状态，完成一个完整的寻呼，成为主设备。

## HCI数据分组

### **指令分组**

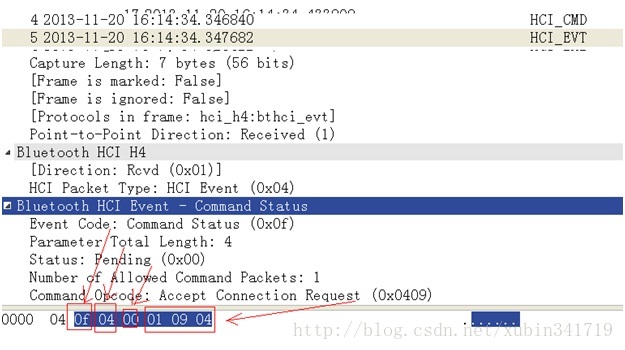
****



如：Accpet Connection Request  
Opcode为：0x0409  
参数长度为: 07   
参数中蓝牙地址为：00:0d:fd:5f:16:9f   
角色为：从设备  0x01  
大端数据模式  
指令为：09 04 07 9f 16 5f fd 0d 00 01

### 事件分组





如上图：  
Opcode :0x0409  
状态： 0x00  
总长度： 4字节  
命令状态：0x0f

### **数据分组**



## L2CAP指令

### 指令格式



### **信令指令代码**



### 示例

代码：06 00 01 00 0a 01 02 00 02 00

解析：



# 常见词汇

## Core Specification

核心规范，用于规定蓝牙设备必须实现的通用功能和协议层次。它由软件和硬件模块组成，两个模块之间的信息和数据通过主机控制接口（HCI）的解释才能进行传递。 **这个是必选**。

## Profiles

蓝牙应用规范，它从应用场景的角度为蓝牙技术的使用制定了不同的规范。这也是和大众日常生活接触最多的一部分，**profile是可选**。

## Attribute protocol（ATT）

属性协议，实现了属性客户端和服务器之间的点对点协议。ATT客户端给ATT服务器发送请求命令，ATT服务器端向ATT客户端发送回复和通知。

## Advances Audio Distribution Profile**（A2DP）**

高质量音频分发规范，定义了如何将立体质量的音频通过流媒体的方式从媒体源传输到接收器上，A2DP使用Asynchronous Connectionless Link（ACL，蓝牙异步传输）信道传输高质量音频内容，它依赖于Generic Audio/Video Distribution Profile（GAVDP，通用音频/视频分发规范）。A2DP必须支持低复杂度及Sub-bandCodec（SBC，低带宽编解码），可选支持MPEG1，2音频，MPEG2、4AAC。

## Audio/Video Remote Control Profile（AVRCP）

音频*/*视频远程控制配置文件。

*AVRCP*设计用于提供控制*TV*、*Hi-Fi*设备等的标准接口。此配置文件用于许可单个远程控制设备（或其它设备）控制所有用户可以接入的*A/V*设备。它可以与*A2DP*或*VDP*配合使用。

　　AVRCP 定义了如何控制流媒体的特征。包括暂停、停止、启动重放、音量控制及其它类型的远程控制操作。AVRCP 定义了两个角色，即控制器和目标设备。控制器通常为远程控制设备，而目标设备为特征可以更改的设备。在 AVRCP 中，控制器将检测到的用户操作翻译为 A/V 控制信号，然后再将其传输至远程 Bluetooth 设备。对于“随身听”类型的媒体播放器，控制设备可以是允许跳过音轨的耳机，而目标设备则是实际的播放器。常规红外遥控器的可用功能可以在此协议中实现。

AVRCP 协议规定了AV/C 数字接口命令集（AV/C 命令集，由1394 行业协会定义）的应用范围，实现了简化实施和易操作性。此协议为控制消息采用了AV/C 设备模式和命令格式，这些消息可以通过音频/视频控制传输协议 (AVCTP) 传输。

## Alternate Media Access Control（MAC）/PHY（AMP）

当两个BR/EDR蓝牙设备L2CAP连接建立后，AMP管理器能检测到另外一个设备的AMP管理器。当两个蓝牙设备都有AMP控制器，蓝牙核心系统提供一种机制，让数据流从主要控制器迁移到次要控制器。

## Dial-up Networking Profile (DUN)

拨号网络配置文件，实现一台蓝牙设备通过另外一个带无线功能的蓝牙设备共享上网。



## Enhanced Data Rate（EDR）

蓝牙增强速率，大大提高了蓝牙技术的数据传输速率，达到了2.1Mbps ；



## Generic Access Profile（GAP)

通用访问配置文件，GAP是所有其他配置文件的基础,它定义了在蓝牙设备间建立基带链路的通用方法.除此之外,GAP还定义了下列内容:

　 ① 必须在所有蓝牙设备中实施的功能

　 ② 发现和链接设备的通用步骤

　 ③ 基本用户界面术语.

GAP确保了应用程序和设备间的高度互操作性,还允许开发人员利用现有的定义更加容易地定义新的配置文件.GAP处理未连接的两个设备间的发现和建立连接过程.此配置文件定义了一些通用的操作,这些操作可供引用GAP的配置文件,以及实施多个配置文件的设备使用.GAP确保了两个蓝牙设备可通过蓝牙技术交换信息,以发现彼此支持的应用程序.不符合任何其他蓝牙配置文件的蓝牙设备必须与GAP符合以确保基本的互操作性和共存.

## Generic attribute profile（GATT）

通用属性规范，表示服务器属性和客户端属性，描述了属性服务器中使用的服务层次，特点和属性。BLE设备使用它作为蓝牙低功耗应用规范的服务发现。

## Generic Object Exchange Profile（GOEP）

通用对象交换。它定义的是数据的传输，包括同步，文件传输，或者推送其它的数据。可以理解为与内容无关的传输层协议，可以被任何应用用来传输自己定义的数据对象。

## Human Interface Device Profile（HID）

人机接口设备协议，

## **Basic Rate（BR）/Low Energy（LE）**

**BR是经典蓝牙，蓝牙4.0之前的蓝牙的统称。LE是低功耗蓝牙，蓝牙4.0之后才支持的，这两种技术，都包括搜索（discovery）管理、连接（connection）管理等机制，相互独立的；**

## Logical Link Control and Adaptation Protocol（L2CAP）

逻辑链路控制和适配协议，L2CAP负责适配基带中的上层协议，它同LM并行工作，向上层协议提供面向连接和无连接的数据服务，并提供多路复用，分段和重组操作，允许高层次的协议和应用能够以64KB的长度发送和接收数据包(L2CAP Serveice Data Units, SDU)。

## Link Management Protocol（LMP）

链路管理协议，负责蓝牙各设备间连接的建立和设置。LMP通过连接的发起，交换和核实进行身份验证和加密，通过协商确定基带数据分组大小；还控制无线设备的节能模式和工作周期，以及微微网络内设备单元的连接状态。

## Generic Object Exchange Profile(GOEP)

通用对象交换配置文件，GOEP可用于将对象从一个设备传输到另一个设备.对象可以是任意的.如:图片,文档,名片等.此配置文件定义了两个角色:提供拉提或推送对象位置的服务器及启动操作的客户端.使用GOEP的应用程序假定链路和信道已按GAP的定义建立.GOEP依赖于串行端口配置文件.

GOEP为使用OBEX协议的其他配置文件提供了通用蓝图,并为设备定义了客户端和服务器角色.对于所有的OBEX事务.GOEP规定应由客户端启动所有事务.但是此配置文件并没有描述应用程序就如何定义要交换的对象或如何实施交换.这些细节留给属于GOEP的配置文件.即OPP,FTP和SYNC去完成.通常使用此配置文件的蓝牙设备为笔记本电脑,PDA,手机及智能电话.

## Service Discovery Application Profile（SDAP)

服务发现应用配置文件，SDAP描述了应用程序如何使用SDP发现远程设备上的服务.由于GAP的要求,任何蓝牙设备都应能够连接至其他蓝牙设备.基于此,SDAP要求任何应用程序都应当能够发现它要连接的其他蓝牙设备上的可用服务.此配置文件可承担搜索已知和特定服务及一般的任务.SDAP涉及了称为“服务发现用户应用程序”的一个应用程序,这是蓝牙设备查找服务所必需的.此应用程序可与向/从其他蓝牙设备发送/接收服务查询的SDP相接.SDAP依赖于GAP,并可以重新使用部分GAP.

## Security manager protocol（SMP）

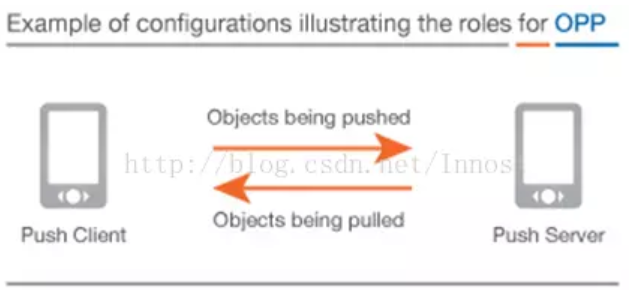
安全管理协议，用于生成对等协议的加密密钥和身份密钥。SMP管理加密密钥和身份密钥的存储，它通过生成和解析设备的地址来识别蓝牙设备。

## Serial Port Profile（SPP)

串行端口配置文件，SPP定义了如何设置虚拟串行端口及如何连接两个蓝牙设备.SPP基于ETSI TS 07.10规格,使用RFCOMM协议提供串行商品仿真. 能在蓝牙设备之间创建串口进行数据传输的一种设备.SPP依赖于GAP.

## **Object Push Profile（**OPP**）**

对象推送规范，定义了推送服务器和客户端之间基于Generic Object Exchange Profile（GOEP，通用对象交换规范）进行对象交换的规范，主要用于手机与手机或者手机与电脑之间通过蓝牙进行文件操作；



## HCI

主机控制接口

## **Internet Of Things（IOT）**

全联网

## ConnectionInterval（连接间隔）

ConnectionInterval（GAPROLE\_MIN\_CONN\_INTERVAL&&GAPROLE\_MAX\_CONN\_INTERVAL）连接间隔，在BLE的两个设备的连接中使用跳频机制。两个设备使用特定的信道发送和接收数据，然后过一段时间后再使用新的信道（BLE协议栈的链路层处理信道的切换）。两个设备在切换信道后发送和接收数据称为一个连接事件。尽管没有应用数据被发送和接收，两个设备仍旧会交换链路层数据（空包 EmptyPDU）来维持连接。

这个连接间隔就是指在一个连接事件（Connectionevents）的开始到下一个连接事件（Connectionevents）的开始的时间间隔。连接间隔以1.25ms为单元，连接间隔的范围是6~3200既7.5ms~4s之间。

## SlaveLatency（从设备延迟或者从设备时延）

允许Slave（从设备）在没有数据要发的情况下，跳过一定数目的连接事件（Connectionevents），在这些连接事件（Connectionevents）中不必回复Master（主设备）的包，这样就能更加省电。

范围可以是0 ~ 499

更详细的使用解析如下：

SlaveLatency=OFF也就是SlaveLatency为0时，Master发包，Slave必须回复，如果不回复，Master就会认为Slave那边接收不正常。

SlaveLatency=ON也就是SlaveLatency不为0的时候，图中SlaveLatency为3。Master发包，Slave没有数据要回复的时候，就会忽略3个连接事件，在第4个连接事件接收到Master发送的数据之后，回复Master。如果Slave有数据要发送就会唤醒，也就是说即使SlaveLatency为3，但是在Master发第二包的时候Slave有数据要回复，这个时候就会立即回复Master而不是等到3个连接事件之后的第4个连接事件去回复。

## SupervisionTimeout（超时时间或者监控超时）

这个参数设定了一个超时时间，如果BLE在这个时间内没有发生通信的话，就会自动断开。

单位是10ms，该变量的范围是10 ~ 3200，折算成时间范围是100ms ~ 32s。

连接间隔、从机时延以及超时时间这三者必须满足如下公式：

SupervisionTimeout >（1+slaveLatency）\*（connectionInterval）

上述公式必须满足，否则连接就会不正常断开。

## Beacon(灯塔)

Beacon 会每隔一定的时间（SKYLAB的beacon为100毫秒）广播一个数据包到周围，作为独立的蓝牙主机（比如手机等）在执行扫描动作时，会间隔地接收到 Beacon 广播出来的数据包。该数据包内容最多可以包含 31 个字节的内容。

同时，在主机接收到广播包时，其中会指示该广播包来自于哪一个蓝牙从机 MAC 地址(每个 Beacon 拥有唯一的 MAC 地址)的从机设备和当前的接收发送信号强度指示值RSSI为多少。这时候，如果手机上安装有beacon对应的app，接收到该ID的app会根据该ID的设置条件采取相应的动作！

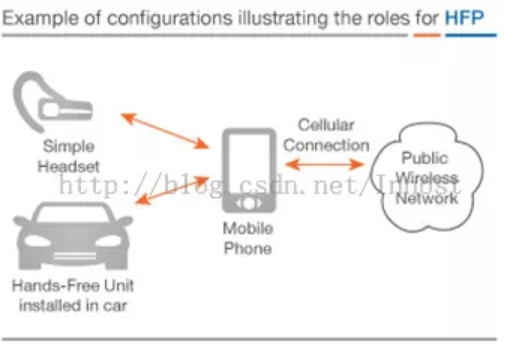
## 声压

主要由输出电信号的电压、喇叭、腔体构造决定。

# 常用协议

## **Hands-free Profile（HFP）**

免提规范，定义了蓝牙音频网关设备如何通过蓝牙免提设备拨打和接听电话。



HFP包括两个角色：

* Audio Gateway（AG，音频网关）和Hands-Free Unit（HF，免提设备）。AG是音频输入和输出的设备，典型的AG设备是手机。HF是执行音频网关的远程音频输入输出设备。
* HFP常见的场景是汽车上的车载套件，当车载套件和耳机通过蓝牙方式连接到手机时，通过无线蓝牙耳机拨打和接听电话。

## AT、GATT

AT：属性协议，定义了客户端与服务器端如何互相发送符合标准的信息；

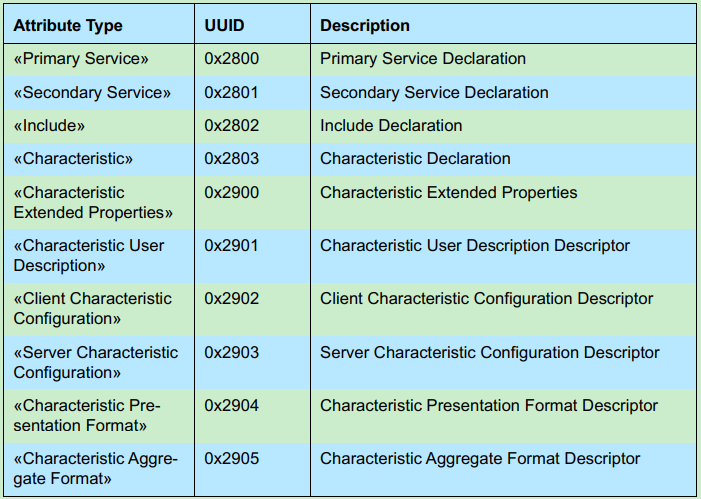
GATT：通用属性协议，定义了如何发现和使用服务、特性、描述符的标准方法；

### GATT属性类型UUID（page 2367）

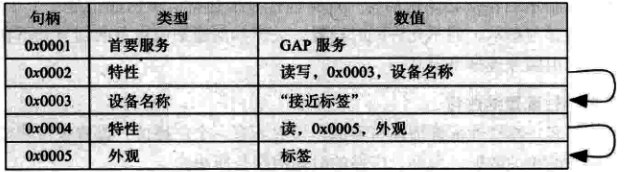
* 大类划分

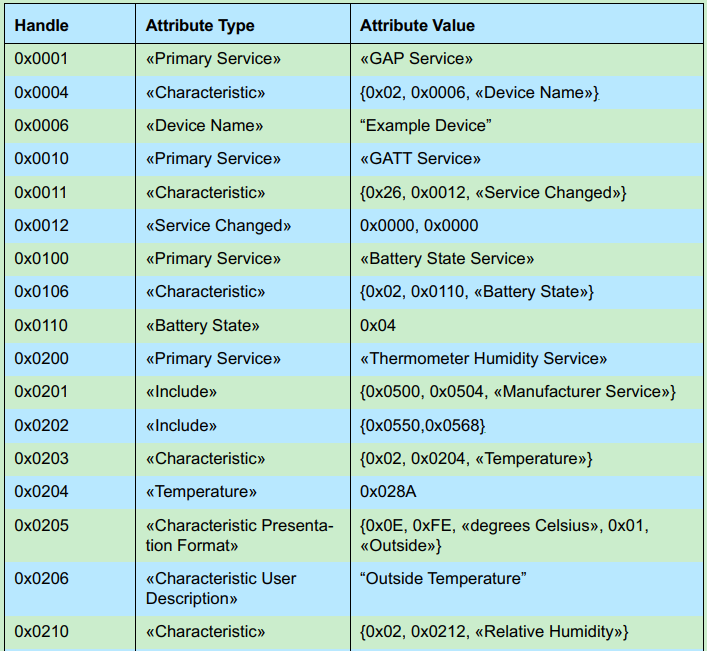


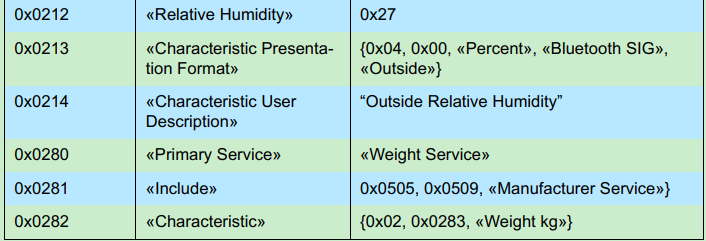
* 属性、特性描述

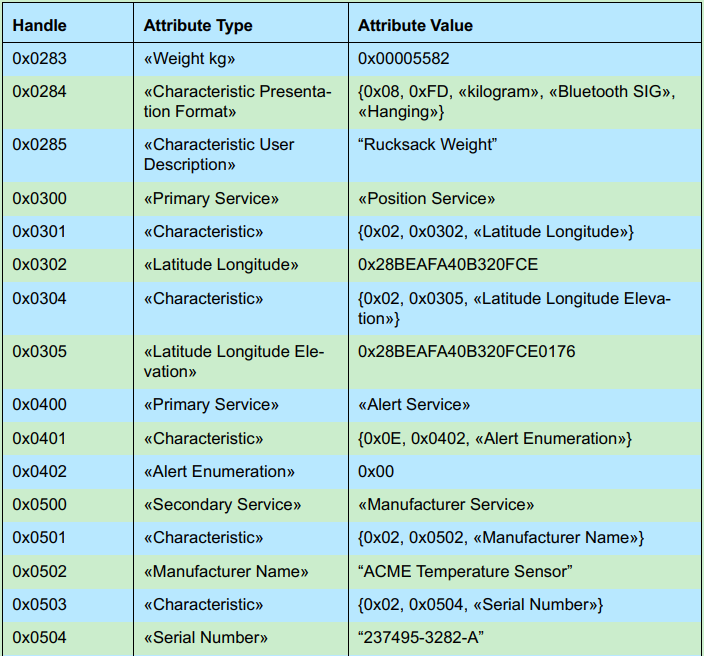


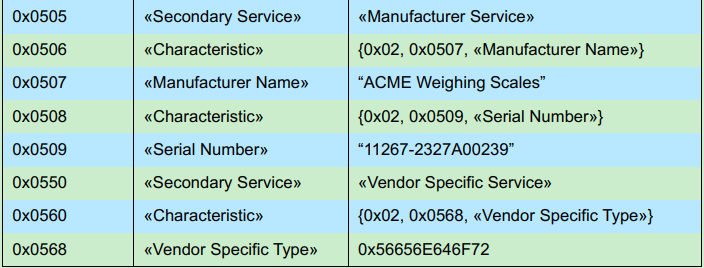
### 服务属性表模板（page 2414）











# 参考资料

1. 蓝牙Profile的概念和常见种类：<https://blog.csdn.net/daydayupfromnowon/article/details/6041988>
2. 蓝牙核心技术了解（蓝牙协议、架构、硬件和软件笔记）：<http://www.cnblogs.com/shaobojiao/p/7883330.html?tdsourcetag=s_pcqq_aiomsg>
3. 蓝牙技术谈之跳频技术：<https://blog.csdn.net/dxdxsmy/article/details/9968247>
4. 蓝牙核心技术概述：<https://blog.csdn.net/xubin341719/article/details/38145507>