

MVsB5_BT_Audio_SDK

系统应用开发 **FAQ** 说明文档

V1.2

版本记录

版本	作者	日期	修改日志
V1.2	Liangzx	2024-6-13	更新 B0/B1/B2/B3/B7/B8 说明
V1.1	Shanks	2024-5-27	更新 USB 声卡采样率配置
V1.0	Liangzx	2024-4-10	内容更新
V0.2	Liangzx	2024-2-21	增加 GPIOB1 说明
V0.1	Liangzx	2024-1-24	初版发布

目录

系统应用开发 FAQ 说明文档	1
版本记录	2
1.蓝牙参数配置	4
2.蓝牙 MAC 地址获取方式	4
3.蓝牙名称说明	4
4.蓝牙频偏值相关说明	4
5.系统参量化数据说明	5
6.参量化数据修改举例 1：蓝牙名修改	6
7.参量化数据修改举例 2：蓝牙重连设置	7
8.参量化数据修改举例 3：蓝牙后台设置	8
9.参量化数据支持列表（持续更新）	8
10.蓝牙 profile 支持说明	9
11.GPIOA31 打印口影响 B0B1 的 linein 指标，如何解决	9
12.蓝牙配对记录清除说明	10
13.系统低功耗配置和策略	10
14.蓝牙歌曲信息使用说明	10
15.FlashBoot 打印如何开启	11
16.如何修改 DAC 高采样率（大于 48K）输出	11
17.如何配置 USB 声卡采样率	12
18.SDK 启动信息说明	14
19.GPIOB0/B1/B2/B3/B7/B8 设置内部上拉没有作用？	15
20.GPIOB0/B1 SarADC 电压有偏差？	15
21.修改开机默认模式	15

1. 蓝牙参数配置

- 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 bt_em_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。
- 4) 蓝牙协议栈 MEM_SIZE 配置在 bt_stack_memoty.h 文件中。

2. 蓝牙 MAC 地址获取方式

- 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。

3. 蓝牙名称说明

- 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 编码的数据可以通过一些工具网站在线转换。
- 注：数字和字母不需要转换，网上转换的数据为十六进制。

4. 蓝牙频偏值相关说明

- 1) 上电时，系统会从 flash 的指定区域读取频偏值并使用；如果 flash 中没有保存频偏参数，则会使用默认频偏值 BT_TRIM(bt_config.h);
- 2) 可以通过串口日志信息观察当前板级的频偏值，蓝牙测试盒(v0.2.7 之后的固件版本)上也会显示校准后的参数；串口日志表现如下：Freq trim: 0x13
- 3) 在使用蓝牙测试盒的时候，可以根据实际情况选择操作：
 - 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；
 - 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；
 - 在频偏校准完成后，蓝牙测试盒上会显示校准后的频偏值；如下图，0x13 是校准值；4Khz 为校准后的偏差频率；



- 在客户的代码调试阶段，可以根据当前样板，校准得到默认的频偏参数，写入 SDK，可以保证当前批量的样板不会出现频偏偏差较大的问题；
- 若生产环节不需要每台机器做频偏校准，可使用试产的产品多个使用测试盒进行频率的校准后，确认测试盒显示的校准值是否保持接近，然后将频偏值写到 SDK 中；
- 如果蓝牙测试盒的频偏校正后面未显示参数，而是“---”，则说明频偏值超过一定范围，测试盒无法校准；

解决方案：

- ① 更换晶体；
- ② 增加或者调整晶体负载电容；
- ③ 加大晶体谐振电流值；

例如：在 main 函数内，Clock_HOSCCurrentSet(10); //晶体谐振电流配置为 10

注：晶体的谐振电流参数配置可以咨询 FAE

目前蓝牙测试盒支持 BT 和 BLE 校准两种方案，详细请询 FAE。

5. 系统参量化数据说明

1) 参量化数据来源于两个地方：SDK 默认值和 flash 中保存的值。

- SDK 默认值来源：sys_param.c 文件 default_parameter 变量。

```
static const SYS_PARAMETER default_parameter =
{
    .bt_LocalDeviceName      = BT_NAME,
    .ble_LocalDeviceName     = BLE_NAME,
    .bt_TxPowerLevel         = BT_TX_POWER_LEVEL,
    .bt_PagePowerLevel       = BT_PAGE_TX_POWER_LEVEL,
    .bt_Trim                  = BT_DEFAULT_TRIM,
    .TwsVolSyncEnable        = TRUE, //主从之间音量控制同步
    .bt_CallinRingType       = SYS_DEFAULT_RING_TYPE,
    .bt_BackgroundType       = SYS_BT_BACKGROUND_TYPE,
    .bt_SimplePairingEnable   = BT_SIMPLEPAIRING_FLAG,
    .bt_PinCode               = BT_PINCODE,
    .bt_ReconnectionEnable    = TRUE,
    .bt_ReconnectionTryCounts = 5,
    .bt_ReconnectionInternalTime = 3,
    .bt_BBLostReconnectionEnable = TRUE,
    .bt_BBLostTryCounts       = 90,
    .bt_BBLostInternalTime    = 5,
    .bt_TwsReconnectionEnable = TRUE,
    .bt_TwsReconnectionTryCounts = 3,
    .bt_TwsReconnectionInternalTime = 3,
    .bt_TwsBBLostReconnectionEnable = TRUE,
    .bt_TwsBBLostTryCounts     = 3,
    .bt_TwsBBLostInternalTime  = 5,
    .bt_TwsConnectedWhenActiveDisconSupport = FALSE,
    .bt_TwsPairingWhenPhoneConnectedSupport = TRUE,
};
```

- Flash 中保存的值：SDK 编译阶段由 flash_param.c 生成 flash_param.bin，并且打包到 MVA 文件里面。可以通过 PC 工具 OCPWorkBench 在线和离线修改。也可以在编译阶段直接在 flash_param.c 文件对 SysDefaultParm 进行修改。

```
const FLASH_PARAMETER SysDefaultParm =
{
    .SysVer = {SYS_PARA_VER_ID, sizeof(SysDefaultParm.SysVer.name),
        "Ver 1.0.0"}, //版本
    .BtName = {BT_PARA_BT_NAME_ID, sizeof(SysDefaultParm.BtName.name),
        BT_NAME}, //蓝牙名称
    .BleName = {BT_PARA_BLE_NAME_ID, sizeof(SysDefaultParm.BleName.name),
        BLE_NAME}, //BLE蓝牙名称
    .bt_TxPowerLevel = {BT_PARA_RF_TX_LEVEL_ID, sizeof(SysDefaultParm.bt_TxPowerLevel.para_val),
        BT_TX_POWER_LEVEL}, //蓝牙正常工作时发射功率
    .bt_PagePowerLevel = {BT_PARA_RF_PAGE_LEVEL_ID, sizeof(SysDefaultParm.bt_PagePowerLevel.para_val),
        BT_PAGE_TX_POWER_LEVEL}, //蓝牙回连发射功率
};
```

2) 参量化数据使用规则：上电以后会优先读取 flash 中的参数，如果没有找到对应参数会使用 SDK 中提供的默认值。
sys_param.c 文件中 sys_parameter_init 完成了该操作。sys_param.c 中定义了 FlashParamReadMap 数组，通过 ID 从 flash 中读取数据，如果读取成功会从 flash 中复制 len 个字节到指定的地方。如果不成功，会从指定默认值的地方复制 len 个字节。

```
//flash中参数读取
//从flash中读取参数成功，将参数拷贝到dest_para
//读取不成功，拷贝默认参数default_para到dest_para
static const struct
{
    SYS_PARAMETER_ID id; //参数ID
    void * dest_para; //参数地址
    void * default_para; //默认参数地址
    uint8_t len; //参数长度
}FlashParamReadMap[] =
{
    {BT_PARA_BT_NAME_ID, (void *)&sys_parameter.bt_LocalDeviceName, (void *)&default_parameter.bt_LocalDeviceName, BT_NAME_SIZE},
    {BT_PARA_BLE_NAME_ID, (void *)&sys_parameter.ble_LocalDeviceName, (void *)&default_parameter.ble_LocalDeviceName, BLE_NAME_SIZE},
    {BT_PARA_RF_TX_LEVEL_ID, (void *)&sys_parameter.bt_TxPowerLevel, (void *)&default_parameter.bt_TxPowerLevel, 1},
    {BT_PARA_RF_PAGE_LEVEL_ID, (void *)&sys_parameter.bt_PagePowerLevel, (void *)&default_parameter.bt_PagePowerLevel, 1},
    {BT_PARA_TRIM_VAL_ID, (void *)&sys_parameter.bt_Trim, (void *)&default_parameter.bt_Trim, 1},
    {BT_PARA_CallinRingType_ID, (void *)&sys_parameter.bt_CallinRingType, (void *)&default_parameter.bt_CallinRingType, 1},
    {BT_PARA_BackgroundType_ID, (void *)&sys_parameter.bt_BackgroundType, (void *)&default_parameter.bt_BackgroundType, 1},
    {BT_PARA_SimplePairingEnable_ID, (void *)&sys_parameter.bt_SimplePairingEnable, (void *)&default_parameter.bt_SimplePairingEnable, 1},
    {BT_PARA_PinCode_ID, (void *)&sys_parameter.bt_PinCode, (void *)&default_parameter.bt_PinCode, BT_PIN_CODE_LEN},
    {BT_PARA_ReconnectionEnable_ID, (void *)&sys_parameter.bt_ReconnectionEnable, (void *)&default_parameter.bt_ReconnectionEnable, 6},
    {TWS_PARA_TWS_VOL_SYNC_ID, (void *)&sys_parameter.TwsVolSyncEnable, (void *)&default_parameter.TwsVolSyncEnable, 1},
    {TWS_PARA_ReconnectionEnable_ID, (void *)&sys_parameter.bt_TwsReconnectionEnable, (void *)&default_parameter.bt_TwsReconnectionEnable, 6},
    {TWS_PARA_TwsPairingWhenPhoneConnectedSupport_ID, (void *)&sys_parameter.bt_TwsPairingWhenPhoneConnectedSupport, (void *)&default_parameter.bt_TwsPairingWhenPhoneConnectedSupport, 1},
    {TWS_PARA_TwsConnectedWhenActiveDisconSupport_ID, (void *)&sys_parameter.bt_TwsConnectedWhenActiveDisconSupport, (void *)&default_parameter.bt_TwsConnectedWhenActiveDisconSupport, 1}
};
```

3) 增加自己的参量化数据。

- flash_param.h 文件中 SYS_PARAMETER_ID 增加一个自定义参数的参数 ID，在 FLASH_PARAMETER 中增加参数的结构体类型（包含 ID+长度+参数值）。
- flash_param.c 文件中 SysDefaultParm，对这个参数赋默认值。
- sys_param.c 读取参数值，并且对参数做合法判断，非法的时候设置一个默认值。

6. 参量化数据修改举例 1：蓝牙名修改

1) 离线修改：PC 运行 OCPWorkBench_V1.0.4(2023.10.28)软件。

- 导入 MVA 文件。
- 修改蓝牙名称。
- 保存为新的 MVA 文件。
- 用新的 MVA 文件下载/烧录到目标板上既可以生效。



- 2) 在线修改：PC 用 USB 线连接目标设备，然后运行 OCPWorkBench_V1.0.4(2023.10.28)软件。
 - a. 点击 Receive_data，读取设备中的参量化数据。
 - b. 修改蓝牙名称。
 - c. 点击 Send_data，写入数据到设备中。
 - d. 重新断电上电设备既可以生效。



7. 参量化数据修改举例 2：蓝牙重连设置

- 1) 参考 6，蓝牙名称的修改步骤。同样支持在线/离线修改 2 种方式。
- 2) 参数说明（参考 SYS_PARAMETER 定义的结构体顺序）：

[1, 5, 3, 1, 90, 5]

参数 1 BT 自动重连(开机或者切换模式)----- 1 开启

参数 2 自动重连尝试次数 ----- 5 次

参数 3 自动重连每两次间隔时间(in seconds) ----- 间隔 3S

参数 4 BB Lost 之后自动重连 1-> 打开/0->关闭 ----- 1 打开

参数 5 BB Lost 尝试重连次数 ----- 90 次

参数 6 BB Lost 重连每两次间隔时间(in seconds) ----- 间隔 5S

- 3) 点击参数修改的时候，在工具下方也会有对应的参数说明。



8. 参量化数据修改举例 3：蓝牙后台设置

- 1) 参考 6，蓝牙名称的修改步骤。同样支持在线/离线修改 2 种方式。
- 2) 参数说明：

BT 后台设置
0 -> BT 后台不能连接手机
1 -> BT 后台可以连接手机
2 -> 无后台

9. 参量化数据支持列表（持续更新）

ID	长度	默认参数	范围	说明
0x00	40	BP15_BT		字符串，蓝牙名称
0x01	40	BP15_BLE		字符串，BLE 蓝牙名称
0x02	2	23	0-2 3	蓝牙正常工作时发射功率
0x03	2	16	0-2 3	蓝牙回连发射功率
0x04	2	20	0-3 1	频偏 trim 值

0x05	2	2	0-3	bt 铃声设置 0 -> 不支持来电铃声 1 -> 来电报号和铃声 2 -> 使用手机铃声，若没有则播本地铃声 3 -> 强制使用本地铃声
0x06	2	0	0-2	BT 后台设置 0 -> BT 后台不能连接手机 1 -> BT 后台可以连接手机 2 -> 无后台
0x07	2	1	0-1	简易配对开启/关闭 0 -> 关闭 1 -> 开启
0x08	8	0000		字符串，配对 pin code
0x09	6	1, 5, 3, 1, 90, 5		蓝牙回连设置 参数 1 BT 自动重连(开机或者切换模式) --- 1 开启 参数 2 自动重连尝试次数 --- 5 次 参数 3 自动重连每两次间隔时间(in seconds) --- 间隔 3S 参数 4 BB Lost 之后自动重连 1-> 打开/0->关闭 --- 1 打开 参数 5 BB Lost 尝试重连次数 --- 90 次 参数 6 BB Lost 重连每两次间隔时间(in seconds) --- 间 隔 5S
0x200	40	Ver 1.0.0		参量化数据版本
0x100 0				用户自定义数据

10. 蓝牙 profile 支持说明

- 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_stack_memoty.h 中 ram config 的相关说明;

11. GPIOA31 打印口影响 B0B1 的 linein 指标，如何解决

可更换其他打印口；调试好以后关闭掉打印；使用 USB 口的打印功能。

12. 蓝牙配对记录清除说明

- 1) 蓝牙配对记录保存区域，是系统根据编译结果自动生成的位置；不是固定的地址范围；此区域的大小为 4K；
- 2) 获取蓝牙配对记录偏移地址函数接口：get_bt_data_addr()；
- 3) 清除蓝牙配对记录的函数接口：BtDdb_Erase()；

13. 系统低功耗配置和策略

- 1) 在 app_config.h 中，宏定义 CFG_FUNC_SYSTEM_LOW_POWER 关联了系统常用降低功耗的修改策略；
- 2) 芯片支持 DCDC 功能的情况下，优先选择 DCDC；即：在 chip_config.h 中定义 CHIP_USE_DCDC；
- 3) 根据系统功能，设定系统时钟选择的模式，SYS_CORE_SET_MODE 为 2 or 3 (power_config.h)
- 4) 注意事项：

当系统功耗要求较高，主频 < 240M，同时 corevdd < 1.05v，此时按键功能，不能使用 SarADC，需要改成 GPIO 方式进行扫描。如在原有基础上，功耗还不满足需求，请咨询 FAE。

14. 蓝牙歌曲信息使用说明

- 1) 蓝牙歌曲信息功能和蓝牙播放状态、时间的获取关联：

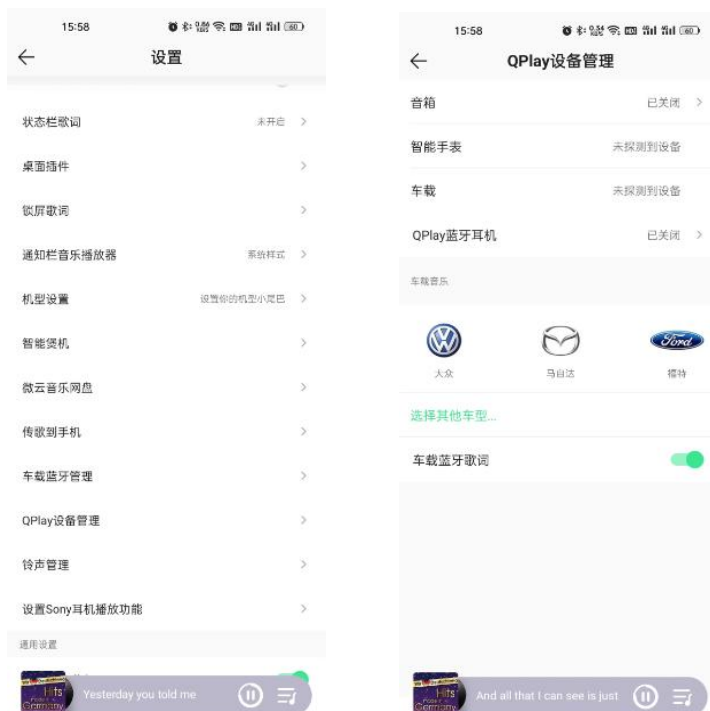
```
BT_AVRCP_SONG_TRACK_INFOR    ENABLE    //开启
BT_AVRCP_SONG_PLAY_STATE     ENABLE    //开启
```

- 2) 手机端蓝牙配置注意事项：

APP 不推送歌曲信息到蓝牙外设，需要先核对手机的蓝牙歌词功能是否开启：

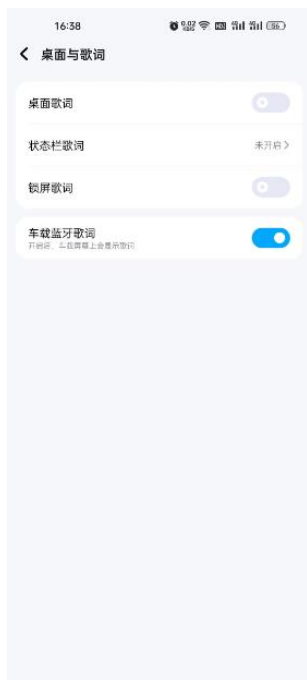
- QQ 音乐设置：

设置->QPlay 设备管理->车载蓝牙歌词 -- 开启



● 酷狗音乐设置：

设置->桌面与歌词->车载蓝牙歌词 -- 开启



15. FlashBoot 打印如何开启

app_config.h 中对宏 CFG_FLASHBOOT_DEBUG_EN 进行配置。不支持 USB 打印方式。

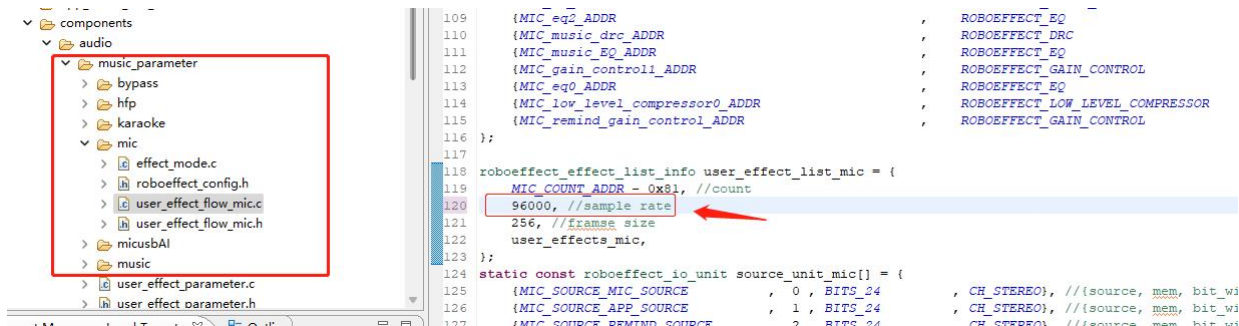
```
#include "debug.h"
#define CFG_FUNC_DEBUG_EN
// #define CFG_FUNC_USBDEBUG_EN
#ifdef CFG_FUNC_DEBUG_EN
    #define CFG_UART_TX_PORT          DEBUG_TX_A10
    #define CFG_UART_BANDRATE        DEBUG_BAUDRATE_2000000//DEBUG_BAUDRATE_115200
    #define CFG_FLASHBOOT_DEBUG_EN    ENABLE//ENABLE
#endif
```

flash_boot.h 中可以对 UART 的 GPIO/波特率进行配置。默认跟随 SDK 打印配置。

16. 如何修改 DAC 高采样率（大于 48K）输出

修改 96K 为例：

- 1) 找到当前 V3 框图使用的 effect_mode，在 user_effect_list 结构体中修改采样率为 96K



- 2) 修改 clk.h 音频时钟频率，需要 4 倍的 11.2896M 或者 12.288M 频率。

```
190     SPDIF_FRAME_FLAG,          //3:   clk1   spdif_frame_flag
191     CLK_12M_MDM_MUX,           //4:   clk1   clk_12m_mdm_mux
192     CLK_48M_MDM_MUX,           //5:   clk1   clk_48m_mdm_mux
193 }CLOCK_OUT_MODE;
194
195
196 //建议MCLK0 配置为11.2896M, MCLK1配置为12.288M
197 #define AUDIO_PLL_CLK1_FREQ    11289600*4//PLL1,11.2896MHz
198 #define AUDIO_PLL_CLK2_FREQ    12288000*4//PLL2,12.288MHz
199 #define IsSelectMcIkClk1(freq) ((AUDIO_PLL_CLK1_FREQ%freq) == 0)
200
201
202 /**
203  * @brief 系统参考时钟源配置选择
```

- 3) 注意事项:

- 需要关闭 MIC/LINEIN/BT 模式，因为 ADC 最高只能支持 48K，也可以对这些做单独配置转采样。
- 提示音 44.1K 需要转采样，提示音通道在 ModeCommonInit 中修改为增加转采样。
- USB 声卡模式，UsbDevicePlayResMalloc 关联了 CFG_PARA_SAMPLE_RATE 宏，可能存在内存申请失败，需要单独修改。

17. 如何配置 USB 声卡采样率

otg_device_standard_request.h 中修改对应配置:

- 1) USB 总带宽不要超过 1K，有编译报错提醒。

```
#define DEVICE_FS_ISO_IN_MPS      (USBD_AUDIO_MIC_FREQ*MIC_CHANNELS_NUM*MAX(MIC_ALT1_BITS,MIC_ALT2_BITS)/1000)
#define DEVICE_FS_ISO_OUT_MPS    (USBD_AUDIO_FREQ*PACKET_CHANNELS_NUM*MAX(SPEAKER_ALT1_BITS,SPEAKER_ALT2_BITS)/1000)
#if (DEVICE_FS_ISO_IN_MPS + DEVICE_FS_ISO_OUT_MPS > 1000)
#error usb带宽不够
#endif
```

- 2) MIC_ALT2_EN 和 SPEAKER_ALT2_EN 使能第二种 PCM 格式，不需要支持两种格式可以注释。
- 3) MIC_FREQ_NUM 和 SPEAKER_FREQ_NUM 声卡采样率个数，最大只能配 6 个。
- 4) USBD_AUDIO_FREQ 和 USBD_AUDIO_MIC_FREQ 只能填支持的最大采样率。并且此参数需要能够被 1000 整除。
- 5) 每次修改参数后需要修改 PID 或 VID，因为电脑会根据这两个记住上次的参数，从而导致无法正常使用。或者使用设备清除工具清除。

```
#ifdef CFG_OTG_MODE_MIC_EN
#define MIC_ALT2_EN //麦克风第二种PCM格式使能

#define MIC_FREQ_NUM 6 //采样率个数 MAX: 6
#define USBD_AUDIO_MIC_FREQ 192000 //192000 : bits per seconds 麦克风最大采样率
#define USBD_AUDIO_MIC_FREQ1 176400 //176400 : bits per seconds
#define USBD_AUDIO_MIC_FREQ2 96000 //48000 : bits per seconds
#define USBD_AUDIO_MIC_FREQ3 88200 //48000 : bits per seconds
#define USBD_AUDIO_MIC_FREQ4 48000 //48000 : bits per seconds
#define USBD_AUDIO_MIC_FREQ5 44100 //48000 : bits per seconds

#define MIC_CHANNELS_NUM 1 //麦克风声道数 1 or 2
#define MIC_ALT1_BITS PCM16BIT //PCM16BIT or PCM24BIT
#define MIC_ALT2_BITS PCM24BIT //PCM16BIT or PCM24BIT
#endif

#ifdef CFG_OTG_MODE_AUDIO_EN
#define SPEAKER_ALT2_EN //扬声器第二种PCM格式使能

#define SPEAKER_FREQ_NUM 2 //采样率个数 MAX: 6
#define USBD_AUDIO_FREQ 48000 //48000 : bits per seconds 扬声器最大采样率
#define USBD_AUDIO_FREQ1 44100 //44100 : bits per seconds
#define USBD_AUDIO_FREQ2 44100 //44100 : bits per seconds
#define USBD_AUDIO_FREQ3 44100 //44100 : bits per seconds
#define USBD_AUDIO_FREQ4 44100 //44100 : bits per seconds
#define USBD_AUDIO_FREQ5 44100 //44100 : bits per seconds

#define PACKET_CHANNELS_NUM 2 //扬声器声道数 1 or 2
#define SPEAKER_ALT1_BITS PCM16BIT //PCM16BIT or PCM24BIT
#define SPEAKER_ALT2_BITS PCM24BIT //PCM16BIT or PCM24BIT
#endif
```

- 6) 默认如下配置:



7) 修改测试:

```
#ifdef CFG_OTG_MODE_MIC_EN
// #define MIC_ALT2_EN //麦克风第二种PCM格式使能

#define MIC_FREQ_NUM 2 //采样率个数 MAX: 6
#define USB_AUDIO_MIC_FREQ 48000 //192000 : bits per seconds 麦克风最大采样率
#define USB_AUDIO_MIC_FREQ1 44100 //96000 : bits per seconds
#define USB_AUDIO_MIC_FREQ2 44100 //48000 : bits per seconds
#define USB_AUDIO_MIC_FREQ3 44100 //48000 : bits per seconds
#define USB_AUDIO_MIC_FREQ4 44100 //48000 : bits per seconds
#define USB_AUDIO_MIC_FREQ5 44100 //48000 : bits per seconds

#define MIC_CHANNELS_NUM 2 //麦克风声道数 1 or 2
#define MIC_ALT1_BITS PCM16BIT //PCM16BIT or PCM24BIT
#ifdef MIC_ALT2_EN
#define MIC_ALT2_BITS PCM24BIT //PCM16BIT or PCM24BIT
#endif
#endif

#ifdef CFG_OTG_MODE_AUDIO_EN
// #define SPEAKER_ALT2_EN //扬声器第二种PCM格式使能

#define SPEAKER_FREQ_NUM 6 //采样率个数 MAX: 6
#define USB_AUDIO_FREQ 192000 //48000 : bits per seconds 扬声器最大采样率
#define USB_AUDIO_FREQ1 176400 //44100 : bits per seconds
#define USB_AUDIO_FREQ2 96000 //44100 : bits per seconds
#define USB_AUDIO_FREQ3 88200 //44100 : bits per seconds
#define USB_AUDIO_FREQ4 48000 //44100 : bits per seconds
#define USB_AUDIO_FREQ5 44100 //44100 : bits per seconds

#define PACKET_CHANNELS_NUM 2 //扬声器声道数 1 or 2
#define SPEAKER_ALT1_BITS PCM16BIT //PCM16BIT or PCM24BIT
#ifdef SPEAKER_ALT2_EN
#define SPEAKER_ALT2_BITS PCM24BIT //PCM16BIT or PCM24BIT
#endif
#endif
#endif
```



18. SDK 启动信息说明

```
*****
|           MVsB5_BT_Audio_SDK           | 1
| Mountain View Silicon Technology Co.,Ltd. |
| SDK Version: 0.2.12                     |
*****
sys_clk =120000000 2
read flash Capacity = 0x200000 0x200000,FlashTable: 1 3
sys_parameter = 0x1fc000
bp_data = 0x1ce000
bt_data = 0x1d0000 4
remind = 0x1d2000
user_config_addr = 0x1fe000
bt_config_addr = 0x1ff000
flash_parameter_ver: Ver 1.0.0
BB_EM_SIZE=20480,EM_BT_END=17988
bt em size:20KB
RstFlag = 1 5
Audio Decoder Version: 8.10.4 build @ Jan 23 2024 18:59:14
Audio Effect Lib Version: 2.36.0
Roboeffect Lib Version: 2.14.0 6
Driver Version: 0.1.17 build @ Jan 17 2024 09:33:54 0
BtLib Version: 15.0.2.6 build @ Dec 21 2023 18:46:11
Fatfs presearch acc Lib Version: 1.6.1 build @ Dec 12 2023 11:01:31

Load BP INFO START
EffectMode:1,1
MusicVolume:24,24
MicVolume:32,32
HfVolume:32,32
MicEffectDelayStep:32,32
power on mode LinePlay
[SYS]: Loading control vars as default
AudioCore init
bluetooth stack service init.
MainApp:run
MainApp:AudioCore service creaed
---MSG_TASK_START---
[17:46:29.771]收◆---
header error
used default bt trim value
*****
Local Device Infor:
Bt Name:BP15_BT
FlashBtAddre(NAP-UAP-LAP):78:2d:01:43:2b:52 7
BtAddr(LAP-UAP-NAP):52:2b:43:01:2d:78
BleAddr:f8:2d:01:43:2b:d2
Freq trim:0x7
*****
```

1) SDK 名称: MVsB5_BT_Audio_SDK; 版本: 0.2.12

2) sys_clk 频率 120M

3) Flash 和索引表信息:

read flash Capacity = 0x200000 (FLASH 实际容量) 0x200000 (SDK 配置的容量),FlashTable: 1 (索引表有效)

4) SDK 常用数据地址

sys_parameter = 0x1fc000 //参量化数据

bp_data = 0x1ce000 //断点记忆数据

bt_data = 0x1d0000 //蓝牙配对信息

remind = 0x1d2000 //提示音

user_config_addr = 0x1fe000 //用户数据, 比如 SN 码

bt_config_addr = 0x1ff000 //系统数据, 蓝牙 MAC 地址, 频偏 trim 值

5) RstFlag 启动标志位

bit0=1 Power On Reset

bit1=1 Pin Reset

bit2=1 Powerkey long_press Reset

bit3=1 Charge Reset

bit4=1 WatchDog Reset

```
bit5=1 LVD(Low Voltage Detected On Vdd33) Reset
bit6=1 CPU SW Debug Reset
bit7=1 System Reset
```

- 6) 库版本信息
- 7) 蓝牙信息

19. GPIOB0/B1/B2/B3/B7/B8 设置内部上拉没有作用？

BP15 系列芯片 B0/B1/B2/B3/B7/B8 这几个 GPIO，内部连接了 audio adc 的输入。这些 IO 如果用于 GPIO 的时候（audio adc 模块关闭），因为 audio 输入上有内部的下拉 50k 阻抗，GPIO 本身的下拉是无效的（有固定下拉了），上拉很强上拉才有效。

使能 RPU 寄存器（普通的 PU 是不行的，RPU 是强上拉），强上拉 10K：

```
GPIO_RegBitsSet(GPIO_B_REG_RPU,GPIOB1);
```

或者调用专用 API 接口：

```
GPIO_PortB10KPullupSet(GPIOB1,ENABLE_10K_OHMS_PULL_UP);
```

20. GPIOB0/B1 SarADC 电压有偏差？

B0/B1 用于 SarADC 功能的时候，需要关闭 SW 功能和对应 audio adc 的复用功能。如上问题 19，有内部的下拉 50k 阻抗，做 ADC 采样的时候，需要将这个下拉电阻考虑进来进行计算。

21. 修改开机默认模式

mode_task.c 中 PowerOnModeGenerate 函数，最终决定进入哪个模式。

- 1) CFG_FUNC_BREAKPOINT_EN 开启的情况下，查找上次记忆的模式。第一次上电默认的模式在 breakpoint.c 中结构体 sInitSysInfo 中修改：

```
46
47 // global sysInfo default init nvm value
48 const static BP_SYS_INFO sInitSysInfo =
49 {
50     #ifdef CFG_APP_BT_MODE_EN
51         (uint8_t)ModeBtAudioPlay,           // CurModuleId
52     #else
53         (uint8_t)ModeUDiskAudioPlay,        // CurModuleId
54     #endif
55     CFG_PARA_SYS_VOLUME_DEFAULT,           //Music Volume default value
56     EFFECT_MODE_DEFAULT,                   //Audio effect mode default value
57     MAX_MIC_DIG_STEP,                      //Mic Volume default value
58     #ifdef CFG_FUNC_MUSIC_EQ_MODE_EN
59         0,                                  //Eq mode default value
60     #endif
61 }
62
63
64
```

- 2) 没有开启 CFG_FUNC_BREAKPOINT_EN 或者记忆区非法值的情况，查找 SysMode 表中第一个 ModeStateInit 状态的模式；
- 3) 按需求，可以强制修改进入某一个指定模式


```
242     ,  
243     mainAppCt.SysPrevMode = SysMode[SYS_MODE_MAX_NUMBER-1].ModeNumber;  
244  
245     if (Mode == ModeIdle) // Do not have memory mode, set a default mode  
246     {  
247         Mode = SysMode[default_count].ModeNumber;  
248     }  
249  
250     Mode = ModeLineAudioPlay;  
251  
252     APP_DBG("power on mode %s\n", GetModeNameStr(Mode));  
253  
254     if (GetModeDefineState(ModeIdle))  
255     {
```