Seekwave Bluetooth Porting Guide

地址:

上海市浦东新区张江科技园碧波路 889 弄 S1 座 10 层

www.seekwavetech.com

版本历史

Date	Content	Author	Version
2023/06/5	First version	BT Group	V1.0
2023/09/8	Modify document format	BT Group	V1.1
2023/09/11	Add Checklist	BT Group	V1.2
2024/05/31	Add BLE ADV wakeup description	BT Group	V1.3

目录

一、 Android 蓝牙移植

1 移植清单

Android 系统蓝牙驱动的移植有如下项目需要检查，以便移植。

序号	移植要求	项目名称	备注
1	必须	SeLinux 修改	如果是 Android7.1 以下版本,不需要在蓝牙 Hal 层增加权限
2	必须	修改调用接口	
3	必须	驱动代码/库	
4	必须	编译 makefile 引入	
5	可选	A2DP 链路标识	根据项目需求。如果有 A2DP 业务需求，但不合入该选项，则容易出现卡顿
6	可选	蓝牙语音(SCO)	根据项目需求，有蓝牙 SCO 需求则合入

注:

希微蓝牙 Controller 不支持谷歌私有的 muti-ADV 命令，在 Android 协议栈编译的时候注意将 BLE_VND_INCLUDED 宏定义置为 FALSE，否则 BLE ADV 功能不能使用。

2 增加蓝牙相关权限

2.1修改设备节点权限

device/xxxx/common/rootdir/ueventd.rockchip.rc

```
/dev/BTBOOT          0666    bluetooth   net_bt
/dev/BTCMD           0666    bluetooth   net_bt
/dev/BTDATA          0666    bluetooth   net_bt
/dev/BTAUDIO         0666    bluetooth   net_bt
/dev/BTISOC          0666    bluetooth   net_bt
```

注意:

- 在 system/core/rootdir/ueventd.rc 文件中添加上述权限也可以，但进行 Google 测试时会失败
- 用户组不一定是 net_bt，有些平台是 net_bt_admin，而有些平台是 bluetooth、net_bt_stack，需要根据当前平台配置
- 有些平台是放到 rc 文件里面，内容如下请参考，使用此种方式要求平台驱动加载要在本 rc 脚本执行之前，否则会由于设备节点不存在而导致权限修改无效。

```
chmod 0666 /dev/BTCMD
chmod 0666 /dev/BTDATA
chmod 0666 /dev/BTAUDIO
chmod 0666 /dev/BTBOOT
chmod 0666 /dev/BTISOC


chown bluetooth net_bt /dev/BTCMD
chown bluetooth net_bt /dev/BTDATA
chown bluetooth net_bt /dev/BTAUDIO
chown bluetooth net_bt /dev/BTBOOT
chown bluetooth net_bt /dev/BTISOC
```

2.2设备节点归属

device/ xxxx /common/sepolicy/vendor/file_contexts

```
/dev/BTBOOT          u:object_r:skwbt_device:s0
/dev/BTCMD           u:object_r:skwbt_device:s0
/dev/BTDATA          u:object_r:skwbt_device:s0
/dev/BTAUDIO         u:object_r:skwbt_device:s0
/dev/ BTISOC         u:object_r:skwbt_device:s0
```

2.3蓝牙设备节点及文件权限修改

device/ xxxx /common/sepolicy/vendor/hal_bluetooth_default.te

```
//蓝牙设备节点访问权限
allow hal_bluetooth_default skwbt_device:chr_file { read write open ioctl };

//文件访问权限【Android12 及以上版本如果加入下面修改导致编译不过，此时可以修改 domain.te，去除对蓝牙驱动写权限的限制，但进行 Google 测试时会失败，在调试阶段尽量打开】
allow hal_bluetooth_default bluetooth_data_file:dir create_dir_perms;
allow hal_bluetooth_default bluetooth_data_file:file { open read write open getattr create ioctl rename };
```

2.4设备节点类型定义

device/ xxxx /common/sepolicy/vendor/device.te

```
type skwbt_device, dev_type;
```

注:

权限相关的修改必须要合入，否则蓝牙驱动对设备节点无读写权限。

3 修改调用接口

对应文件 bluetooth/1.0/default/vendor_interface.cc 的 VendorInterface::Open 函数，修改代码使 vendor_lib_name 变量为 libbt-vendor-seekwave.so，便于在蓝牙开启的时候加载到相应驱动

对应 patch：hardware_interfaces.patch

如果系统/SDK 没有通过属性等方式设置蓝牙地址，即 BluetoothAddress::get_local_address()函数返回为假，在 vendor_interface.cc 文件 Open 函数中，如下这行代码需要注释，否则将异常。

```
223:     return false;
224:   }
225:
226:   // Get the local BD address
227:
228:   uint8_t local_bda[BluetoothAddress::kBytes];
229:   if (!BluetoothAddress::get_local_address(local_bda)) {
230:     LOG_ALWAYS_FATAL("%s: No Bluetooth Address!", __func__);
231:   }
232:   int status = lib_interface_->init(&lib_callbacks, (unsigned char*)local_bda);
233:   if (status) {
234:     ALOGE("%s unable to initialize vendor library: %d", __func__, status);
235:     return false;
236:   }
237:
238:   ALOGD("%s vendor library loaded", __func__);
```

◆ 注：

上述仅介绍常规方法，有些 SDK 需要根据模组 PID 和 VID 动态加载对应驱动，因此需要根据当前 SDK 及客户要求动态适配。

4 蓝牙驱动(libbt)代码/库

将 hardware 文件夹中的内容拷贝到项目 hardware 目录下，这个目录是放置蓝牙驱动及相应配置的目录。一般路径为/hardware/seekwave/skwbt

◆ 注：

(a) 如果驱动路径不在 hardware 目录，需要修改 skwbt.mk 文件 CUR_PATH 变量为实际路径

(b) 如果/vendor/etc/bluetooth 目录不存在，则需要修改驱动源码，一般在低版本的 Android 系统需要修

改

5 A2DP 链路标识

A2DP 链路标识用于标识某个链路在进行 A2DP 业务，便于 BT Controller 链路的调度。

对应 patch：A2DP_cmd_patch.patch，该 patch 是基于 Android10，如果版本高于 Android10，需要将 avdt_send_a2dp_status_cmd 函数中的 p_lcb->handle 改成 p_lcb->Handle()即可。

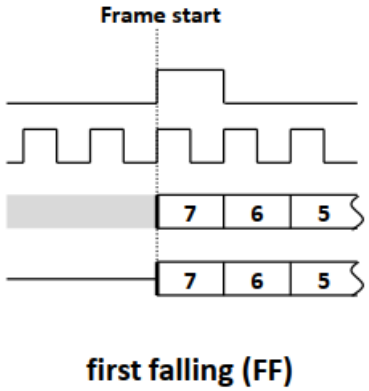
Android13 及以上版本蓝牙协议栈目录有所变化，在 packages/modules/Bluetooth/system 目录下，如 avdt_ccb_act.cc 在如下路径，修改方式与 Android12 一致。

packages/modules/Bluetooth/system/stack/avdt/avdt_ccb_act.cc

6 蓝牙语音(SCO)

6.1PCM 接口说明

PCM 接口为 Slave 模式，主控提供时钟；支持 8KHz 及 16KHz 两种采样频率；16bit 位宽；两个通道(Channel)；支持 dsp_b 模式，即先收 MSB，且 MSB 与 fsync 对齐，如下图所示：



采样频率为 8KHz 时，对应时钟频率为 8KHz*2(channel) * 16(位宽) = 256KHz
采样频率为 16KHZ 时钟，对应时钟频率为 16KHz*2(channel)*16(位宽) = 512KHz

6.2移植步骤

6.2.1 Profile 配置

需要打开 hfp Profile，如下所示：

```
<bool name="profile_supported_hs_hfp">true</bool>
```

RK 平台的 Profile 配置文件如下所示：
device/rockchip/common/overlay/packages/apps/Bluetooth/res/values/config.xml

6.2.2 配置声卡

蓝牙语音默认走 I2S 接口，需要在 dts 文件中配置声卡，以确保进行语音通话的时候，数据通过 I2S 接口传给蓝牙。(相关 patch 需要主控平台提供)

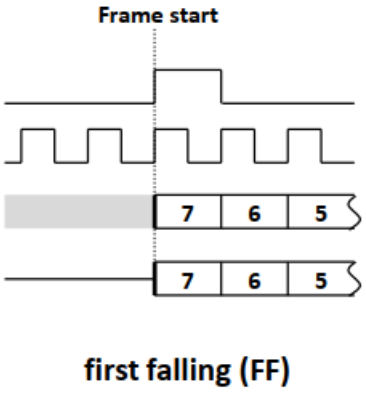
以下以 RK3399 平台举例：

```
//设备树声卡配置(kernel/arch/arm64/boot/dts/rockchip/rk3399-blend-mipi.dtsi)
        bt-sound {
                compatible = "simple-audio-card";
                simple-audio-card,format = "dsp_b";
                simple-audio-card,bitclock-inversion = <1>;
                simple-audio-card,mclk-fs = <256>;
                simple-audio-card,name = "rockchip,bt";
                simple-audio-card,cpu {
                        sound-dai = <&i2s1>;
                };
                simple-audio-card,codec {
                        sound-dai = <&bt_sco>;
                };
        };
        bt_sco: bt-sco {
                compatible = "delta,dfbmcs320";
                #sound-dai-cells = <0>;
                status = "okay";
        };

# I2s 时钟分频系数配置，
&i2s1 {
        rockchip,bclk-fs = <32>;
        status = "okay";
};
```

注意：

目前仅支持 dsp_b 模式，也就是线上数据与 fsync 线对齐，不 delay 1bit。如果声音异常，可以先在 hardware/rockchip/audio/tinyalsa_hal/audio_hw.c 里 out_write 函数里设置发送数据固定为 0x5A，然后抓取 pcm 线上数据查看波形是否正确。



目前仅支持该模式，先收 MSB，且 MSB 与 fsync 对齐

6.2.3 配置支持宽带编码

确认 I2S 驱动是否支持蓝牙宽带编码(MSBC)：rates 属性是否包含 SNDRV_PCM_RATE_16000

```
// kernel/sound/soc/codecs/bt-sco.c
static struct snd_soc_dai_driver bt_sco_dai[] = {
        {
                .name = "bt-sco-pcm",
                .playback = {
                        .stream_name = "Playback",
                        .channels_min = 1,
                        .channels_max = 2,
                        .rates = SNDRV_PCM_RATE_8000 | SNDRV_PCM_RATE_16000,
                        .formats = SNDRV_PCM_FMTBIT_S16_LE,
                },
                .capture = {
                         .stream_name = "Capture",
                        .channels_min = 1,
                        .channels_max = 2,
                        .rates = SNDRV_PCM_RATE_8000 | SNDRV_PCM_RATE_16000,
                        .formats = SNDRV_PCM_FMTBIT_S16_LE,
                },
        },
}
```

6.2.4 确认声卡信息与配置一致

(1)、adb shell cat /proc/asound/cards 查看所有声卡信息

```
D:\>adb shell cat /proc/asound/cards
 0 [rockchiphdmi   ]: rockchip_hdmi - rockchip,hdmi
                      rockchip,hdmi
 1 [rockchipbt     ]: rockchip_bt - rockchip,bt
                      rockchip,bt
 2 [rockchipes8316c]: rockchip_es8316 - rockchip,es8316-codec
                      rockchip,es8316-codec
```

在 DTS 中配置的声卡名称为 rockchipbt，因此需要查看声卡列表是否包含 rockchipbt，如上图所示，如果没有则声卡没有配置正确。一般需要查看对应的 i2s 管脚是否有冲突，是否存在，并且需要查看管脚是否配置正确。

(2)、从 logcat 中(rockchipbt)确定蓝牙使用的是哪个声卡

```
D modules.primary.audio_hal: card0 id:rockchiphdmi
D modules.primary.audio_hal: HDMI card, got card=0,device=0
D modules.primary.audio_hal: card1 id:rockchipbt
D modules.primary.audio_hal: BT card, got card=1,device=0
```

上图显示 BT 使用的是 card 1，device 0

(3)、使用 tinypcminfo -D cardId -d deviceId 查看该声卡的详细信息（tinypcminfo -D 1 -d 0）

```
PS D:\work\vscode\android\download> adb shell tinypcminfo -D 1 -d 0
Info for card 1, device 0:

PCM out:
      Access:   0x000009
   Format[0]:   0x000004
   Format[1]:   00000000
 Format Name:   S16_LE
   Subformat:   0x000001
        Rate:   min=8000Hz      max=16000Hz
    Channels:   min=2           max=2
 Sample bits:   min=16          max=16
 Period size:   min=64          max=65536
Period count:   min=2           max=2048

PCM in:
      Access:   0x000009
   Format[0]:   0x000004
   Format[1]:   00000000
 Format Name:   S16_LE
   Subformat:   0x000001
        Rate:   min=8000Hz      max=16000Hz
    Channels:   min=2           max=2
 Sample bits:   min=16          max=16
 Period size:   min=64          max=65536
Period count:   min=2           max=2048
```

从上图中可知支持的最小采样频率为 8KHz，最大为 16KHz

6.2.5 确认配置与代码中配置一致

确认 Audio Hal 是否已经支持蓝牙宽带编码

```
// hardware/rockchip/audio/tinyalsa_hal/ audio_hw.c
static int adev_set_parameters(struct audio_hw_device *dev, const char *kvpairs)
{
......
    ret = str_parms_get_str(parms, AUDIO_PARAMETER_KEY_BT_SCO_WB, value, sizeof(value));
    if (ret >= 0) {
        adev->bt_wb_speech_enabled = !strcmp(value, AUDIO_PARAMETER_VALUE_ON);
    }
.......
}
```

6.3蓝牙语音数据抓取

为方便问题定位一般需要确定上层传给蓝牙之前的音频数据是否完好，同时也需要确定蓝牙传给 AP 的音频是否存在问题，一般需要在 Audio Hal 中抓取音频数据。

tinyalsa_hal/audio_hw.c 的 out_write 函数中将音频数据保存下来，即为 AP 传给蓝牙的数据。

```
if(pcm_src_out_fp == NULL)
{
        pcm_src_out_fp = fopen("/data/misc/audioserver/pcmdata_to_bt.pcm", "wb");
        if(pcm_src_out_fp != NULL)
        {
                ALOGD("Src Open ok");
        }
        else
        {
                ALOGD("Src Open fail: %s", strerror(errno));
        }
}
if(pcm_src_out_fp != -1)
{
        fwrite((char *)out_buffer, outFrameCount*2*2, 1, pcm_src_out_fp);
}
```

tinyalsa_hal/audio_hw.c 的 get_next_buffer 函数中将音频数据保存下来，即为蓝牙传给 AP 的音频数据。

```
if (in->device & AUDIO_DEVICE_IN_BLUETOOTH_SCO_HEADSET)
{
        if(pcm_src_in_fp == NULL)
        {
                pcm_src_in_fp = fopen("/data/misc/audioserver/pcmdata_from_bt.pcm", "wb");
                if(pcm_src_in_fp != NULL)
                {
                        ALOGD("Src In Open ok");
                }
                else
                {
                        ALOGD("Src In Open fail: %s", strerror(errno));
                }
        }
        if(pcm_src_in_fp != NULL)
        {
                fwrite((char *)in->buffer, read_size, 1, pcm_src_in_fp);
        }
}
```

7 蓝牙驱动编译

在 makefile 中加入蓝牙驱动 makefile，以便将蓝牙驱动打包至文件系统中。

例如在 RK 平台下在 device/rockchip/common/device.mk 文件中增加如下内容

#seekwave bt driver
include hardware/seekwave/skwbt/skwbt.mk

注意事项

Android 蓝牙编译时<<内核编译时 CONFIG_SKW_BT 一定要关掉>>

8 蓝牙相关日志

蓝牙驱动包含三部分：logcat、HCI log、BT Controller log，对应配置文件为 skwbt.conf。修改该文件的下面三个参数为 true/false 可以控制 log 开启/关闭。

SkwBtsnoopDump=true　　//蓝牙 HCI log
SkwBtcplog=true　　// BT Controller log
SkwBtDrvlog=true　　//即在 logcat 中不会显示蓝牙驱动的相关 log

skwbt.conf 对应项目源码路径为/hardware/seekwave/skwbt/vendor/etc/bluetooth 目录下；对应在 Android 系统中/vendor/etc/bluetooth/目录下。

蓝牙 log 默认存储在/data/misc/bluedroid 目录下，可以在 skwbt.conf 文件中修改默认路径，但前提需要蓝牙 Hal 层有对该文件夹的读写权限。

注:

蓝牙 log 需要写权限，Android12 已经修改安全策略，需要修改 domain.te。

8.1蓝牙 log 开关

蓝牙 log 受配置文件 skwbt.conf 文件和内核配置宏 CONFIG_SEEKWAVE_PLD_RELEASE 控制。如果在配置文件将相关配置置 false 则关闭对应 log。如果内核编译的时候 CONFIG_SEEKWAVE_PLD_RELEASE 为 y，则也会关闭蓝牙 Controller 及 HCI log，此时修改 skwbt.conf 文件无效。即不设置 CONFIG_SEEKWAVE_PLD_RELEASE 的情况下（如果在驱动放在内核，.config 文件为 # CONFIG_SEEKWAVE_PLD_RELEASE is not set）才会去检查 skwbt.conf 配置文件。真值表如下所示：

CONFIG_SEEKWAVE_PLD_RELEASE	skwbt.conf 文件相关配置	log 结果
y	true/false	无 log
不设置	false	无 log
	true	有 log

9 蓝牙地址

蓝牙地址使用规则：如果 AP 侧有地址则使用 AP 侧地址，没有则查看 CP 侧是否写有 efuse 地址，如果写入则使用该地址，否则使用随机地址。

因为有些客户有自己的地址段，希望使用自己的地址段，因此在驱动初始化的时候由主控(AP)侧写入，优先使用该地址。

调试阶段 AP 侧和 CP 侧基本上都不会写有地址，为防止调试的设备地址均相同，驱动会随机生成一个地址并保存在/data/misc/bluedroid 目录下，每次均使用该地址。如果蓝牙驱动没有对该目录的读写权限，则蓝牙临时/随机地址则无法保存，即会导致每次重启后的地址均不一样。

二、 Linux 系统移植

蓝牙驱动目录：kernel/drivers/Bluetooth

1 移植清单

Linux 系统蓝牙驱动的移植有如下项目需要检查，以便移植。

序号	移植要求	项目名称	备注
1	必须	Makefile 修改	
2	必须	编译配置	
3	必须	驱动代码	
4	必须	编译选项	
5	可选	A2DP 链路标识	根据项目需求，如果有 A2DP 业务需求，但不合入该选项，则容易出现卡顿

2 修改 makefile

修改 makefile

在 Makefile 文件中添加：

```
obj-$(CONFIG_SKW_BT)        += skwbt.o
skwbt-y                     := skw_btdriver.o
skwbt-y                     += skw_btsnoop.o
skwbt-y                     += skw_log.o
skwbt-y                     += skw_common.o
```

注：

如果驱动在内核外，则 Makefile 已经包含该改动，不需要再做修改

3 修改编译配置

在 Kconfig 文件中增加如下内容：

```
config SKW_BT
        tristate "Seekwave BT driver"
        depends on SEEKWAVE_BSP_DRIVERS && MMC
        help
          The driver for seekwave Bluetooth chipsets with SDIO/USB interface.

          Say Y here to compile support for Seekwave BT-over-SDIO/USB driver
          into the kernel or say M to compile it as module.
```

注：

如果驱动在内核外，则 Kconfig 已经包含该改动，不需要再做修改

4 增加编译选项

如果蓝牙驱动在内核内则增加编译选项需要根据项目进行：有些项目只需要运行 make menuconfig 选中 SKW_BT 即可。有些项目需要修改配置文件，CONFIG_SKW_BT 才能加入到.config 文件中。
如果在蓝牙驱动在内核外，不需要额外增加编译选项，和蓝牙驱动 skwbt 目录平级的 Makefile 已经增加了 CONFIG_SKW_BT 编译选项。

★ 注意：

如果是 6652 系列或者是 6621XX-S 系列还需要设置 CONFIG_SEEKWAVE_BSP_DRIVERS_V20=y。否则在编译的时候提示 skw_start_bt_service 和 skw_stop_bt_service 未定义。

上面操作如果在配置文件里面设置 CONFIG_SEEKWAVE_BSP_DRIVERS_V20=y 不方便，还可以直接修改蓝牙的 Makefile，把对宏定义的判断去掉即可，修改如下所示：

```
3
4  #ifeq ($(CONFIG_SEEKWAVE_BSP_DRIVERS_V20),y)
5      ccflags-y += -DINCLUDE_NEW_VERSION=1
6  #endif
7
```

5 驱动代码拷贝

如果蓝牙驱动在内核内：将蓝牙驱动代码拷贝到 kernel/drivers/Bluetooth
如果蓝牙驱动在内核外：不需要做修改

6 蓝牙配置文件

蓝牙配置文件文件 sv6160.nvbin/sv6316.nvbin 等以 nvbin 为扩展名的文件，在蓝牙打开时会将其下发给蓝牙 Controller，使用 request_firmware 函数获取配置文件内容，因此需要将该配置文件放入到内核能读取的目录下，如/lib/firmware/。不同的系统可能对应的目录不一样。

7 蓝牙相关日志

蓝牙相关日志在 skw_common.h 文件中进行配置，其中有两个宏定义分别控制 HCI 日志及 BT Controller 日志的开启和关闭。

```
#define BT_HCI_LOG_EN    0          //HCI 日志
#define BT_CP_LOG_EN     0          //BT Controller 日志
```

注意：

SEEKWAVE_BT_LOG_PATH 宏定义为蓝牙 log 存储的路径，这个目录必须存在，并且需要有读写权限，且有足够的存储空间，log 文件大小由 MAX_BT_LOG_SIZE 宏定义决定

Linux 内核写文件会先写到缓存中，因此开启 log 时会消耗内存(不是内存泄露，会消耗到 cache 中)，若发现内存消耗过大情况可以关闭 log 确认

7.1蓝牙 log 开关

蓝牙 log 受上述宏定义控制，同时也受 CONFIG_SEEKWAVE_PLD_RELEASE 宏控制，如果在驱动编译的时候 CONFIG_SEEKWAVE_PLD_RELEASE 宏置为 y，则不会生成 log。CONFIG_SEEKWAVE_PLD_RELEASE 宏优先级高于 BT_HCI_LOG_EN/ BT_CP_LOG_EN，真值表如下所示：

CONFIG_SEEKWAVE_PLD_RELEASE	BT_HCI_LOG_EN/ BT_CP_LOG_EN	log 结果
y	0/1	无 log
不设置	0	无 log
	1	有 log

8 A2DP 链路标识

A2DP 链路标识用于标识某个链路在进行 A2DP 业务，便于 BT Controller 链路的调度。A2DP 链路标识在 Linux 系统需要修改内核协议栈(kernel/net/bluetooth/)及应用层协议栈(BlueZ)。不同的系统可能修改方式可能不一样，请联系开发人员。

9 蓝牙地址设置

Linux 系统蓝牙地址使用规则与 Android 系统下的使用规则一致，请参考 Android 章节。

驱动提供加载驱动时设置地址，设置地址为大端模式，操作方式如下所示：

```
insmod skwbt.ko bd_addr=34:9A:98:86:74:22
```

使用上述命令后通过 hciconfig 命令可以查看地址如下所示：

```
root@root:/home/root# hciconfig
hci0:   Type: Primary   Bus: SDIO
        BD Address: 34:9A:98:86:74:22   ACL MTU: 1021:9   SCO MTU: 255:4
        UP RUNNING
        RX bytes:0 acl:0 sco:0 events:64 errors:0
        TX bytes:1352 acl:0 sco:0 commands:68 errors:0
```

蓝牙地址构成如下所示：

```
BD_ADDR: 0x34-9a-98-86-74-22
    LAP: 0x86-74-22
    UAP: 0x98
    NAP: 0x34-9a
```

9.1蓝牙临时地址

蓝牙地址一般写在芯片中，但调试阶段一般不会写有地址，使用的是固件中的固定地址，这样导致调试阶段的所有蓝牙设备均为同一个地址，因此在驱动中会生成一个随机地址，保存在系统中，这样可以避免地址的重复。

如果使用临时地址，则需要确保驱动在指定目录有读写文件权限。蓝牙地址存储的目录可以修改 skw_common.h 文件中的 BD_ADDR_FILE_PATH 宏定义。默认该宏定义与 Log 一致。

10 注意事项

蓝牙驱动需要依赖平台驱动，只有加载完平台驱动才能加载蓝牙驱动，因此建议使用模块编译生成 skwbt.ko

三、 BLE 唤醒功能

支持 BLE 连接及非连接状态的唤醒。连接状态下的唤醒即当前系统处于睡眠状态，对端设备通过发送数据以达到唤醒系统的功能，此功能默认支持，无需额外操作。

BLE 非连接下的唤醒包含：关机(uboot)模式及系统 suspend 模式下的唤醒。

要求： 所有的唤醒 ADV 必须为 ADV_IND(0x00)包

1 关于 uboot 唤醒

Uboot 唤醒一般分为第一次上电和上电之后的唤醒两种模式。

1.1上电第一次唤醒

由于刚上电，系统没有初始化导致蓝牙也无法初始化，此时蓝牙芯片处于掉电状态，需要在 uboot 阶段下载固件到芯片并启动，此时配置文件也一同下发到芯片中。由于 uboot 阶段对文件的索引非常麻烦，并且不同的 SDK 所能访问的路径都是有限制的，因此无法和系统启动之后的唤醒共用一个配置文件，故将蓝牙唤醒的匹配规则放到 seekwave_boot.h 文件中的 BLE_WAKEUP_ADV_INFO 宏定义中。使用时需要留意该配置。配置如下所示，参数介绍请参考后续章节。

```
#define BLE_WAKEUP_ADV_INFO
"16;1;0;020106031980010FFF00112233AABBCCDD;0000FF0000FFFFFF000000000000000000"
```

注意：

第一次上电进入 uboot 的唤醒和系统启动正常开机后进入 uboot 之后的唤醒代码不一样。

后续章节所提到的 uboot 唤醒均为系统启动正常开机后进入 uboot 之后的唤醒。

2 BLE 非连接下的唤醒

2.1Android Uboot 模式 BLE 开机功能

在某些场景下，设备进入 shutdown 模式并非真正关机，而进入 Uboot 模式，此时蓝牙芯片不掉电，需要遥控器的开机键来开启设备，按下遥控器的开机键会发送 ADV 包，蓝牙芯片收到相应的 ADV 包后会通过拉低/高 GPIO 来唤醒设备，使设备进入 boot 模式。

此种场景需要以下要求：

(1) 设备进入 shutdown 模式并非真正关机，而进入 Uboot 模式，此时蓝牙芯片不掉电

(2) 设备进入 shutdown 时在调用 system/core/init/reboot.cpp 文件中的 HandlePowerctlMessage 方法时，在/dev 目录下建立一个 shutdown 设备/文件，以便蓝牙驱动(libbt)能接收到 shutdown 动作

(3) 设备端能通过 GPIO 唤醒，并进入开机模式

(4) 遥控器端的开机键所发送的 ADV 要有一定的格式

2.2Linux Uboot 模式 BLE 开机功能

Linux 驱动提供 btseekwave_write_ble_wakeup_adv_enable 函数，在准备进入关机时调用。此场景需要蓝牙处于打开状态。

2.3系统 suspend 模式下的唤醒

当系统进入 suspend 时 BT Controller 会进入 BLE Scan 模式，如果收到期望的 ADV 包时会操作 GPIO 唤醒主控。

3 ADV 唤醒的驱动配置

3.1.1 配置项/文件配置

Android 系统在/vendor/etc/Bluetooth/skwbt.conf 文件中加入/修改 WakeupADVData 节点，节点配置如下所示：

WakeupADVData=gpio No;effactive level;[addr offset;ADVData;Mask]

Linux 系统在 skw_common.h 文件中的 BLE_WAKEUP_ADV_INFO 宏定义，定义示例如下所示：

#define BLE_WAKEUP_ADV_INFO “gpio No;effactive level;[addr offset;ADVData;Mask]”

◆ 说明：

(1) 各个参数详细说明如下表所示。

(2) 参数之间使用半角分号(;)隔开

(3) 红色方括号在实际配置不使用，这里表示红色方框部分可以添加多个，最多为 3 组

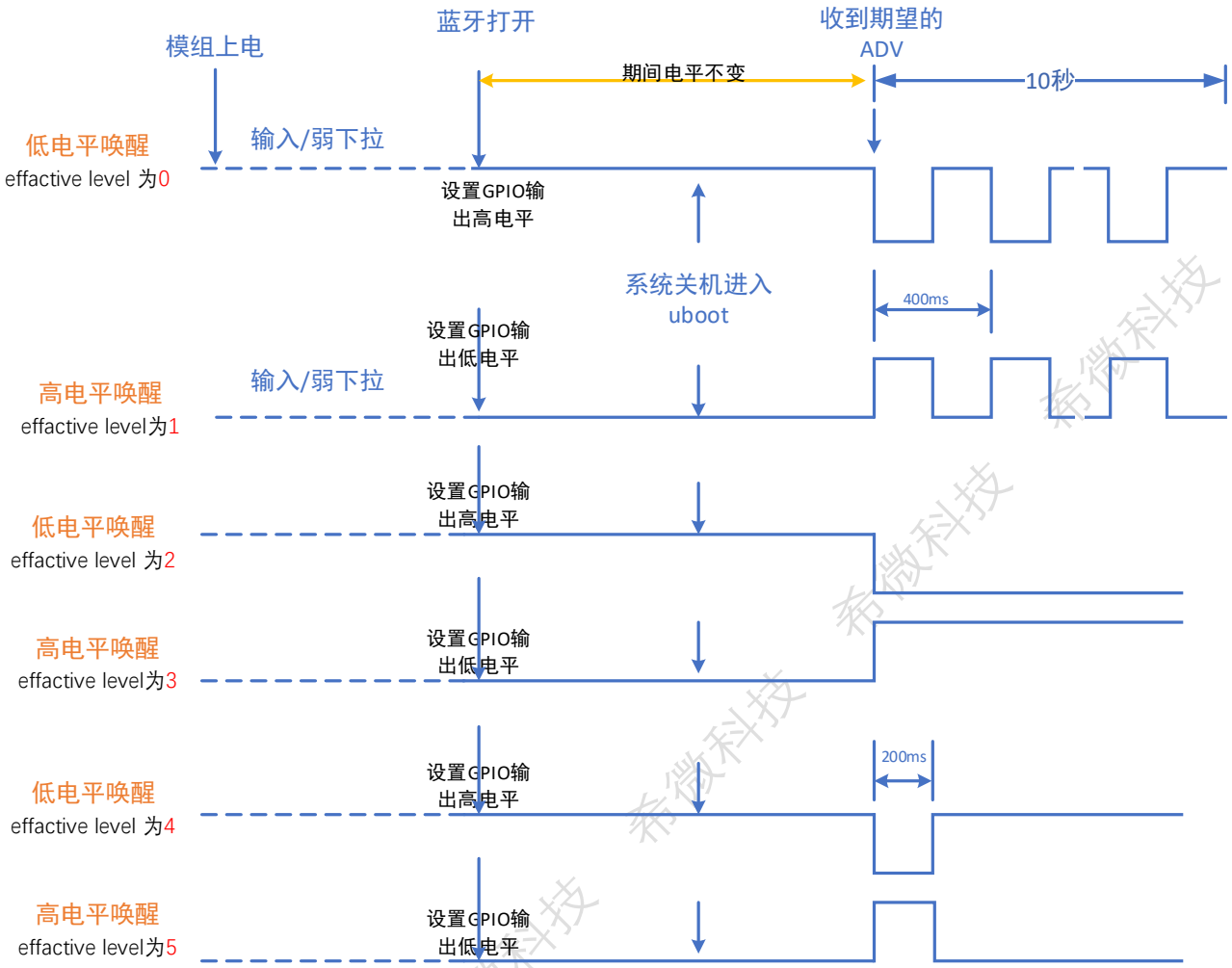
(4) 要求 ADV Data 和 Mask 的长度一定要相等

(5) 待匹配的 ADV 包格式必须为 ADV_IND 类型，因此 ADV 匹配最大长度为 31 字节

(6) Android 系统：前面加#号表示注释，例如#WakeupADVData=XXXXXXXXXXXXX.....；Linux 则使用双斜杠//来作为注释

参数	说明
gpio No	蓝牙芯片端用于唤醒的 GPIO 序号，十进制表示
effactive level	有效电平： 0 为低有效，蓝牙打开后初始化为高，收到期望 ADV 包后为 400ms 周期持续 10 秒脉冲 1 为高有效，蓝牙打开后初始化为低，收到期望 ADV 包后为 400ms 周期持续 10 秒脉冲 2 为低有效，蓝牙打开后初始化为高，收到期望 ADV 包后拉低 3 为高有效，蓝牙打开后初始化为低，收到期望 ADV 包后拉高 4 为低有效，蓝牙打开后初始化为高，收到期望 ADV 包后拉低一个 200ms 脉冲 5 为高有效，蓝牙打开后初始化为低，收到期望 ADV 包后拉高一个 200ms 脉冲 其它值为非法值
addr offset	需要比较地址的偏移，十进制，0 表示不需要进行地址比较(理论上 1，26 及以上值都是非法值，因为 ADV 最大包长为 31，地址长度为 6)，当为有效值时从该字节开始比较地址(计数时包含 0 字节)，如 17，即表示从第 17 字节比较
ADVData	将匹配的 ADV 数据，为十六进制的字符串
Mask	掩码，以便于 ADV 数据的比较，哪个比特不参与比较则将对应位置 1

关于电平的配置及对应状态转换如下图所示。模组上电后，唤醒的 GPIO 为输入脚，表现为弱下拉，不会影响对应主控 GPIO 管脚的状态。



举例 1：GPIO 序号为 16，高电平有效，没有地址比较，包含一组匹配数据

WakeupADVData=16;1;0;020106031980010DFF5D0003000101F815B5896D3C;00000000000000000000000000000FF000000000000

举例 2：GPIO 序号为 16，高电平有效，包含两组匹配数据。组 1 没有地址比较；组 2 包含地址比较，地址部分从第 19 字节开始比较

WakeupADVData=16;1;0;020106031980010FFF00112233AABBCCDD;0000FF0000FFFFFF000000000000000000;19;020104030312180319C1030DFF1122330001;00000000000000000000000000000000000000

注：

如果需要匹配希微 BLE 唤醒 ADV 包格式，请参考后续章节

3.1.2 驱动代码修改

Android 系统：将 skwbt/include/skw_ext.h 文件中的 BLE_ADV_WAKEUP_ENABLE 宏定义置 1，表示开启 ADV 唤醒功能。

Linux 系统：配置好 BLE_WAKEUP_ADV_INFO 宏定义即表示开启 ADV 唤醒功能。

★注：

默认发布的版本该功能关闭。

4 希微 BLE 唤醒 ADV 包格式

ADV 格式如下表所示，其遵循 SIG ADV 格式，具体请参阅《Core Spec》Vol 3, Part C ,11 ADVERTISING AND SCAN RESPONSE DATA FORMAT 部分，其中类型(AD Type)定义请参阅《Generic Access Profile》(https://www.bluetooth.com/specifications/assigned-numbers/)中 2.3 Common Data Types 部分。下表中蓝色部分为可更改部分，黑色字体部分不能更改，总长度为 23 字节，ADV 最大包长为 31 字节，用户可以根据情况扩展使用剩余的 8 个字节，如下表绿色字体部分。

长度	类型	值(LSB --- MSB)	备注
0x02	0x01	0X06	值可变
0x03	0x19	0X80, 0X01	值可变
0x0F	0xFF	0x00, 0x11, 0x22, 0x33, 0xAA, 0xBB, 0xCC, 0XDD	唤醒识别，不能更改
		0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF	最后连接的蓝牙地址
xx	xx	xx, xx, xx, xx, xx, xx	扩展部分，长度域最大值为 7

本端收到 ADV 包后如果黑色字体部分均匹配，则会接受最后连接的蓝牙地址为全 F 的包，否则会和本端地址相比较，如果相等则接受，立即唤醒系统，否则丢弃。驱动请使用如下配置，其中 GPIO 编号、电平信号需要根据实际情况配置。地址偏移部分始终为 0。

WakeupADVData=16;1;0;020106031980010FFF00112233AABBCCDD;0000FF0000FFFFFF000000000000000000

注：

如果使用希微 BLE 唤醒 ADV 包格式，则可以实现 1 对 1 功能，即如果对端设备已经和本端连接过，

其发送的 ADV 包不会被其它设备所识别，仅能被最后一次连接的设备所识别。

5 BLE 连接请求唤醒

进入此模式后本端会发送 ADV，收到对端请求(收到 CONNECT_IND 包)后操作所配置的 GPIO 唤醒主控。ADV 参数来源有两种：一是在关机之前使能的 ADV 参数；一个是使用希微自定义的参数，该参数 ADV 周期为 500ms，ADV data 带有 SeekwaveBT 名字。

6 BLE 唤醒使能

默认只支持 BLE Scan 模式下的唤醒，如果需要支持 BLE 连接请求唤醒请根据情况使用如下参数。

```
typedef enum{
    WAKEUP_OP_DISABLE = 0x00,
    WAKEUP_OP_SCAN_ONLY,
    WAKEUP_OP_SCAN_ADV_HOST,//use host paramater
    WAKEUP_OP_SCAN_ADV_SKW //use skw data: SeekwaveBT
}le_wakeup_op_enum;
```

四、 常见问题

1 Android 版本驱动和 Linux 版本驱动混淆

Google 在 Android 系统中已经将蓝牙驱动从 Linux 内核中提取出来，专门设置为一个动态库(libbt)，所以两种操作系统的蓝牙驱动完全不一致。因此在实现上也使用了不同的模式，同时在平台驱动层上也做了改动，两种不同的版本使用 CONFIG_SKW_BT 宏来区分，如果检查到 CONFIG_SKW_BT 宏有定义则认为是 Linux 版本，否则为 Android 版本。

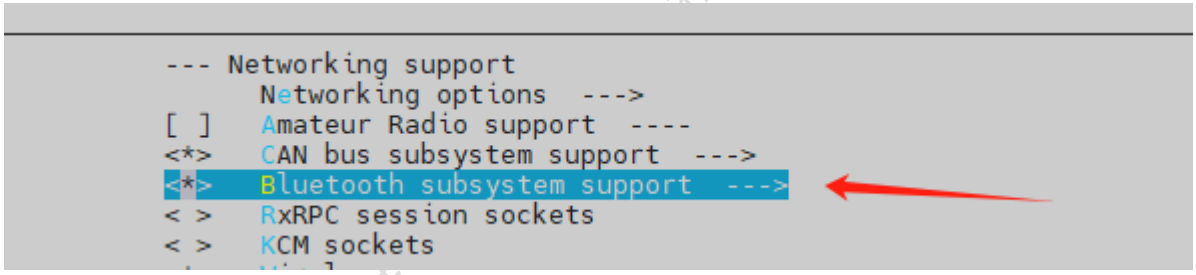
Linux 系统中需要将 CONFIG_SKW_BT 置成 m，Android 系统将将其去掉。

Android 蓝牙驱动文件为：libbt-vendor-seekwave.so
Linux 蓝牙驱动文件为:skwbt.ko

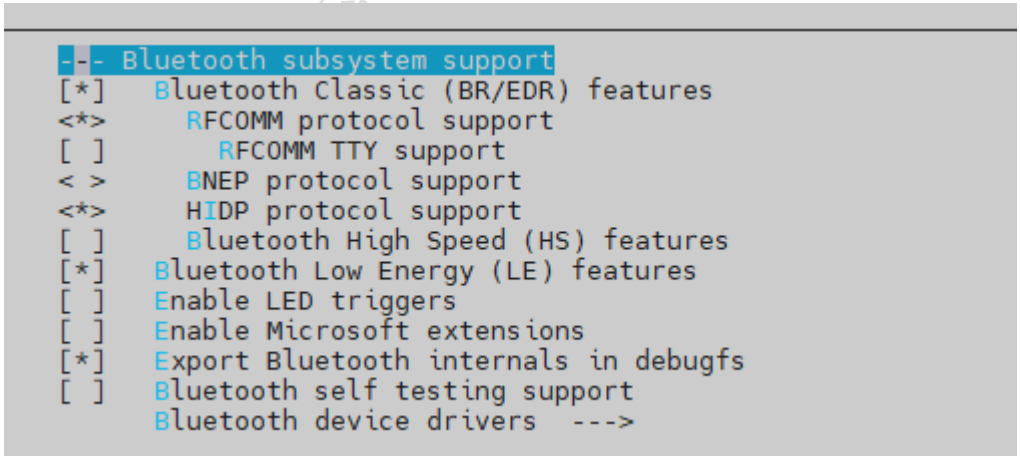
2 Linux 蓝牙驱动编译相关函数未定义

2.1hci_register_dev、hci_recv_frame、hci_unregister_dev、hci_alloc_dev、hci_free_dev 函数未定义

这些函数为内核函数，需要在内核打开蓝牙相关选项：CONFIG_BT=y



如下图所示选项请实际选择：



2.2skw_start_bt_service、skw_stop_bt_service 函数未定义

这两个函数是 BSP 驱动的导出函数，请检查 BSP 驱动编译时 CONFIG_SKW_BT 是否置成 m，同时请查看 BSP 驱动编译之后的 Module.symvers 文件中是否存在这两个函数，如果没有存在则是 BSP 驱动编译有问题，如果存在则需要查看蓝牙驱动的 Makefile 是否存在下面定义：

```
ifneq ($(skw_extra_symbols),)
KBUILD_EXTRA_SYMBOLS += $(skw_extra_symbols)
endif
```

skw_extra_symbols 在主 Makefile 有定义，如下所示：

```
SKW_SRC_ROOT := $(shell pwd)
SKW_EXTRA_INC := $(SKW_SRC_ROOT)/include/linux/platform_data
SKW_EXTRA_SYMBOLS := $(SKW_SRC_ROOT)/drivers/seekwaveplatform/Module.symvers

ARCH ?= $(shell uname -m)
KSRC ?= /usr/src/linux-headers-$(shell uname -r)
CROSS_COMPILE ?= arm-linux-gnueabi-

all:
    make CONFIG_SEEKWAVE_BSP_DRIVERS="m" CONFIG_SKW_BT="m" CONFIG_SKW_BSP_UCOM="m"
     -C $(KSRC) $ M=$(SKW_SRC_ROOT)/drivers/seekwaveplatform modules
    make CONFIG_SEEKWAVE_BSP_DRIVERS="m" CONFIG_SKW_BT="m"  skw_extra_flags="-I$(SK
    /drivers/skwbt modules skw_extra_symbols=$(SKW_EXTRA_SYMBOLS)
    make ARCH=$(ARCH) CROSS_COMPILE=$(CROSS_COMPILE) -C $(KSRC) M=$(SKW_SRC_ROOT)/d
        CONFIG_WLAN_VENDOR_SEEKWAVE=m \
        skw_extra_flags="-I$(SKW_EXTRA_INC) -I$(SKW_SRC_ROOT)/drivers/skwifi -inclu
        skw_extra_symbols=$(SKW_EXTRA_SYMBOLS)
```

3 Linux 的蓝牙配置文件

Linux 系统的蓝牙配置文件 sv6160.nvbin/6316.nvbin 等以 nvbin 为扩展名的文件，在驱动中使用 request_firmware 接口获取，因此配置文件的路径需使驱动能访问到，如/lib/firmware 目录。

如果 request_firmware 接口不是映射到/lib/firmware 目录，则需要将配置文件放置到相应的目录。

4 驱动是否加载成功

4.1Android 版本

启动后直接打开蓝牙，如果蓝牙能打开说明加载成功，否则可以从以下三点排查问题。

查看是否存在蓝牙设备节点 BTBOOT/BTCMD/BTDATA/BTAUDIO。如果存在则表示平台驱动已加载并且设备成功注册。

```
rk3399_Android10:/ # ls -l /dev
total 12
crw------- 1 system    system 235,   1 2023-07-14 02:10 ATC
crw-rw-rw- 1 bluetooth net_bt 235,   5 2023-07-14 02:10 BTAUDIO
crw-rw-rw- 1 bluetooth net_bt 235,  13 2023-07-14 02:10 BTBOOT
crw-rw-rw- 1 bluetooth net_bt 235,   4 2023-07-14 02:10 BTCMD
crw-rw-rw- 1 bluetooth net_bt 235,   3 2023-07-14 02:10 BTDATA
crw------- 1 root      root   235,   2 2023-07-14 02:10 LOG
crw------- 1 root      root   235,   8 2023-07-14 02:10 LOOPCHECK
```

在 logcat 中查找是否存在字符串：libbt-vendor-seekwave.so，如果存在则表示已经蓝牙驱动库已存在

4.2Linux 版本

加载蓝牙驱动后使用 dmesg 查看 log，如果存在字符串 init cmd response: 0xfc8001，则表示蓝牙驱动已经响应，驱动和蓝牙内核协议栈已经链接上。

使用 bluetoothctrl show，查看是否显示蓝牙地址。【需要安装 BlueZ 协议栈】

```
root@ubuntu:/# bluetoothctl show
Controller A0:59:50:31:50:5F (public)
        Name: ubuntu
        Alias: ubuntu
        Class: 0x006c010c
        Powered: yes
        Discoverable: no
        DiscoverableTimeout: 0x000000b4
```

或者使用使用 hcitool dev 命令查看，是否显示蓝牙地址。

```
root@ubuntu:/# hcitool dev
Devices:
        hci0    A0:59:50:31:50:5F
```

5 蓝牙 HCI 命令被拒绝

在应用程序中发送 HCI 命令返回 0x0C(Command disallowed)，如果该命令为 ADV 或者 Scan 相关命令，大概率是命令不匹配。

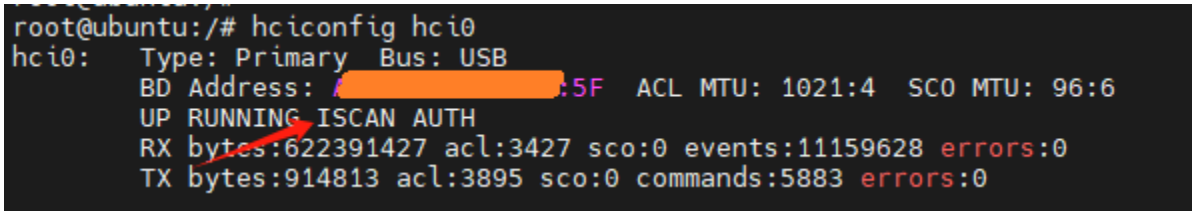
如果 Controller 即支持 Legacy 命令也支持 Ext 命令时，如果使用了一次 Legacy 的命令，则不能再使用 ext 命令；反之如果首先用了 ext 命令，则不能再使用 legacy 命令。即 legacy 命令和 ext 命令不能混用。因此如果遇到命令被拒绝，则可以查看当前内核是不是太老或者太新了，与当前应用程序的命令不匹配。此时需要应用程序根据内核或者 Controller 情况发送不同命令。

6 蓝牙不能被搜索到

如果使用应用程序操作，大概率是下发的 HCI 命令被拒绝，可以参考上述“蓝牙 HCI 命令被拒绝”章节”

如果使用的是系统命令或者 App，则需要查看可发现模式(Inquiry Scan)是否打开。

Linux 系统可以使用 hciconfig 命令查看 Inquiry Scan 是否打开，如下图所示。



Android 系统，有些手机有可发现模式按钮，有些需要进入到蓝牙设置界面才会打开可发现。

7 Log 提供

当对蓝牙进行测试时可能出现一些问题，此时需要研发参与分析解决，因此需要提供蓝牙相关的 log。蓝牙 HCI log 及 CP log 的配置，可以参阅移植部分的 log 配置部分。

7.1Android 版本

Android 系统需要提供以下 log，HCI log 及 CP log 在/data/misc/bluedroid 目录下，可将整个目录一同打包提供。

Android log，即 Logcat

内核 log

HCI log

CP log

如果发生异常(如发生 Assert 等)还需要提供平台 log(memory log)，路径为 data 目录下：log000/log111

7.2Linux 版本

Linux 系统下需要提供 kernel log，HCI log 及 CP log，其中 HCI log 及 CP log 路径在驱动中配置，可参阅前面所述。

8 通话音质差或无声

如果出现无声或者声音类似于电流声，一般是接口问题(如果是蓝牙问题只能是卡顿)，可能是声卡配置不正确(Android 系统的声卡配置可参阅移植部分)，也可能 PCM 接口通路未通，此时可以飞线抓 PCM 接口信号以确定。

如果出现卡顿，可能是上层传输过来的音频本来就卡顿，此时可以改成有线网络以确定是否是网络问题，如果改成有线网络仍然卡顿，可以到屏蔽室等环境测试，如果卡顿得以解决则说明是环境造成。

9 蓝牙设备无法连接

蓝牙设备连接后就断开，一般出现这种情况都是使用如手机这类设备，这类设备默认不支持长连接，即连接后交互完双方信息就断开连接，可以更换蓝牙耳机或者音箱再试。

10 蓝牙音乐卡顿

首先确认 A2DP 链路标识相关 Patch 是否已经合入。
其次关掉 WiFi 查看卡顿是否还存在。
更换蓝牙音箱/耳机确认是否为兼容性问题。

11 蓝牙偶尔搜索不到设备

Android12 开始修改了搜索策略，即配对和搜索不能同时进行，而 BLE 连接基本上都以白名单形式连接，其连接超时时间为 30 秒，因此会出现 30 秒左右无法搜索到设备的问题。

12 BLE 遥控器无法回连

目前在 Android 平台出现 BLE 遥控器无法回连问题，原因是 Android 蓝牙协议栈在进行背景连接设置本端地址时仅查看本端 controller 是否支持 privacy，并不查看对端是否支持 privacy，当查看本端支持时直接使用随机地址，导致本端发起的 Connect_IND 包中的地址并非之前连接的地址，此时对端使用的是 direct ADV，从而对端拒绝接受导致无法连接成功。

当前修改方案为：每连接一个 BLE 设备均会保存其 Features，进行背景连接时遍历所有设备，发现有设备不支持 privacy，一律使用公有地址。对应 patch 为 ble_privacy.patch。

BLE Features 中的 privacy 对应位如下图所示。参见《core spec》V6 Low Energy Controller, Part B LINK LAYER SPECIFICATION, 4.6 FEATURE SUPPORT。

Bit position	Link Layer Feature	Valid from Controller to Controller	Masked to Peer	Host Controlled
0	LE Encryption	Y	N	N
1	Connection Parameters Request Procedure	Y	N	N
2	Extended Reject Indication	Y	N	N
3	Slave-initiated Features Exchange	Y	N	N
4	LE Ping	N	N	N
5	LE Data Packet Length Extension	Y	N	N
6	LL Privacy	N	N	N
7	Extended Scanner Filter Policies	N	N	N
8	LE 2M PHY	Y	N	N

Table 4.6: FeatureSet field's bit mapping to Controller features

★ 注意：

有些设备 features 设置为支持 privacy，而实际却不支持，当使用 RPA 地址时对端无法解析而导致连接失败。这种设置为支持而实际不支持的行为是违反协议的，遇到这种设备只能关闭本端 privacy 功能，但也丧失了该功能，因此需要根据实际评估。

13 低功耗蓝牙无法搜索到设备，write scan enable 返回 invalid parameters

这种情况一般是设置 scan 参数时地址使用随机地址，而在 write scan enable 之前没有初始化随机地址导致。如果是 Android 可能一打开蓝牙扫描时则协议栈会崩溃重启，表现为蓝牙不停打开关闭，需要在在协议栈代码里面的 system\btif\src\btif_dm.cc 文件里面的 void BTIF_dm_enable() 函数 BTA_DmBleConfigLocalPrivacy 函数的入参初始化为 TRUE。

14 Android BLE ADV 无法开启

Muti-ADV 是谷歌早期为实现多个 ADV 而增加的，SIG 后续使用 ext ADV 来实现相同的功能(Core 4.2)，我们模组是遵循 SIG 标准，即支持 ext ADV。bluedroid 是谷歌的代码，其使用宏定义将两个功能/场景隔开。

希微蓝牙 Controller 不支持谷歌私有的 muti-ADV 命令，在 Android 协议栈编译的时候注意将 BLE_VND_INCLUDED 宏定义置为 FALSE，否则 BLE ADV 功能不能使用。

BLE_VND_INCLUDED 宏定义一般在 system/bt/internal_include/bt_target.h，有些 SDK 可能会在 device 或者 vendor 目录下的 bdroid_buildcfg.h 中修改