A31 平台音频模块开发说明文档

2013-01-30

CONFIDENTIAL

Copyright © 2011-2013 Allwinner Technology. All Rights Reserved.

版本历史

版本	时间	备注
V1.0	2012-10-24	建立初始版本
V1.1	2013-01-30	增加 3G 音频通道切换

CONFIDENTIAL

Copyright © 2011-2013 Allwinner Technology. All Rights Reserved.

目 录

1. 前言......1
1.1. 编写目的......1
1.2. 适用范围......1
1.3. 相关人员......1
2. 音频模块介绍......2
2.1. audio 模块功能介绍......2
2.1.1. audio codec 功能......2
2.1.2. hdmiaudio 功能......2
2.1.3. spdif 功能......2
2.1.4. i2s 功能......2
2.1.5. pcm 功能......3
2.1.6. switch 耳机检测功能......3
2.2. 源码结构介绍......3
2.3. audio 相关术语介绍......4
2.4. audio 模块配置介绍......5
2.4.1. Menuconfig 配置......5
2.4.2. Sysconfig.fex 配置......9
3. 音频模块体系结构描述......15
4. 接口描述......16
4.1. Audiocodec 接口描述......16
4.1.1. 系统音频播放接口......16
4.1.2. 系统音频录音接口......19
4.1.3. 手机上行模拟通路接口......19
4.1.4. 手机下行模拟通路接口......22
4.1.5. 手机通话录音接口......24
4.1.6. 模拟音频扩展接口......26
4.2. Hdmiaudio 接口描述......29
4.3. spdif 接口描述......29
4.4. I2s 接口描述......29
4.5. pcm 接口描述......29
4.6. switch 接口描述......30
5. 模块开发 demo......31
5.1. 最小的 playback 应用......31
5.2. 最小的 capture 应用......35
5.3. Mixer 接口的使用......39
6. 音频常见问题调试......40
6.1. 耳机插拔检测失败......40
6.2. 喇叭耳机音量大小......40
6.3. pcm 蓝牙音频没有声音输出或者噪音......40

CONFIDENTIAL

Copyright © 2011-2013 Allwinner Technology. All Rights Reserved.

1. 前言

1.1. 编写目的

本文档目的是为了让开发者了解 A31 音频系统框架，能够在 A31 平台上开发新的音频方案。

1.2. 适用范围

硬件平台：

A31 平台。

软件平台：

exdroid4.1.1_r1-a31-v1.2 及以上版本。

1.3. 相关人员

音频系统开发人员。

CONFIDENTIAL

2. 音频模块介绍

linux 中的 audio 子系统采用 alsa 架构实现。alsa 目前已经成为了 linux 的主流音频体系结构。在内核设备驱动层，ALSA 提供了 alsa-driver，同时在应用层，ALSA 为我们提供了 alsa-lib，应用程序只要调用 alsa-lib 提供的 API，即可以完成对底层音频硬件的控制。

2.1. audio 模块功能介绍

在 a31 中，存在 5 个音频设备。分别为 audiocodec，hdmiaudio，spdif，pcm，i2s。其中芯片内置的 audiocodec 采用 alsa-pci 架构实现，hdmiaudio，spdif，pcm，i2s 采用 alsa-asoc 架构实现。switch 主要实现耳机检测的功能。Pcm，i2s 都可以配置成 pcm 和 i2s 两种模式。

2.1.1. audio codec 功能

Audio Codec 驱动所具有的功能：

- 支持多种采样率格式(8khz, 11.025 KHz, 12 KHz, 16 KHz, 22.05 KHz, 24 KHz, 32 KHz, 44.1 KHz , 48 KHz, 96KHz, 192KHz);
- 支持 mono 和 stereo 模式;
- 支持同时 playback 和 record(全双工模式);
- 支持 start, stop, pause 和 resume;
- 支持 mixer 接口
- 支持 3g 通话功能

2.1.2. hdmiaudio 功能

hdmiaudio 驱动所具有的功能：

- 支持多种采样率格式(8khz，11.025khz，12khz，16khz，22.05, 24khz，32khz，44.1, 48khz, 88.2khz，96khz，176.4khz，192khz);
- 支持 mono 和 stereo 模式;
- 只支持 playback 模式，不支持 record 模式。
- 支持 start, stop, pause 和 resume;

2.1.3. spdif 功能

spdif 驱动所具有的功能：

- 支持多种采样率格式(22.05khz，24khz，32khz，44.1khz，48khz，88.2khz，96khz，176.4khz, 192khz);
- 支持 mono 和 stereo 模式;
- 只支持 playback 模式，不支持 record 模式。
- 支持 start, stop, pause 和 resume;

2.1.4. i2s 功能

i2s 驱动所具有的功能：

- 支持多种采样率格式(8khz，11.025khz，16khz，22.05khz，24khz，32khz，44.1khz，48khz, 88.2khz，96khz，176.4khz，192khz);
- 支持 mono 和 stereo 模式;

CONFIDENTIAL

Copyright © 2011-2013 Allwinner Technology. All Rights Reserved.

- 支持同时 playback 和 record(全双工模式);
- 支持 start, stop, pause 和 resume;

I2s 驱动可以配置成 i2s 模式，也可以配置成 pcm 模式，如果配置成 pcm 模式，那么只支持 8k 采样率。在 a31 中，有两套 pcm 驱动，为了区分，分别命名为 pcm 和 i2s。

2.1.5. pcm 功能

pcm 驱动所具有的功能:

- 支持多种采样率格式(8khz，11.025khz，16khz，22.05khz，24khz，32khz，44.1khz，48khz，88.2khz，96khz，176.4khz，192khz);
- 支持 mono 和 stereo 模式;
- 支持同时 playback 和 record(全双工模式);
- 支持 start, stop, pause 和 resume;

Pcm 驱动可以配置成 i2s 模式，也可以配置成 pcm 模式，如果配置成 pcm 模式，那么只支持 8k 采样率。在 a31 中，有两套 pcm 驱动，为了区分，分别命名为 pcm 和 i2s。

2.1.6. switch 耳机检测功能

Switch 支持 3 段耳机，4 段耳机的插拔检测。

2.2. 源码结构介绍

a31 中的音频子系统存放于 soc 目录下，如图 1 音频系统源码所示。其中 Audiocodec，hdmiaudio，i2s，pcm，spdif 都是一个独立的音频驱动。耳机检测驱动源码如图 2 switch 源码所示。

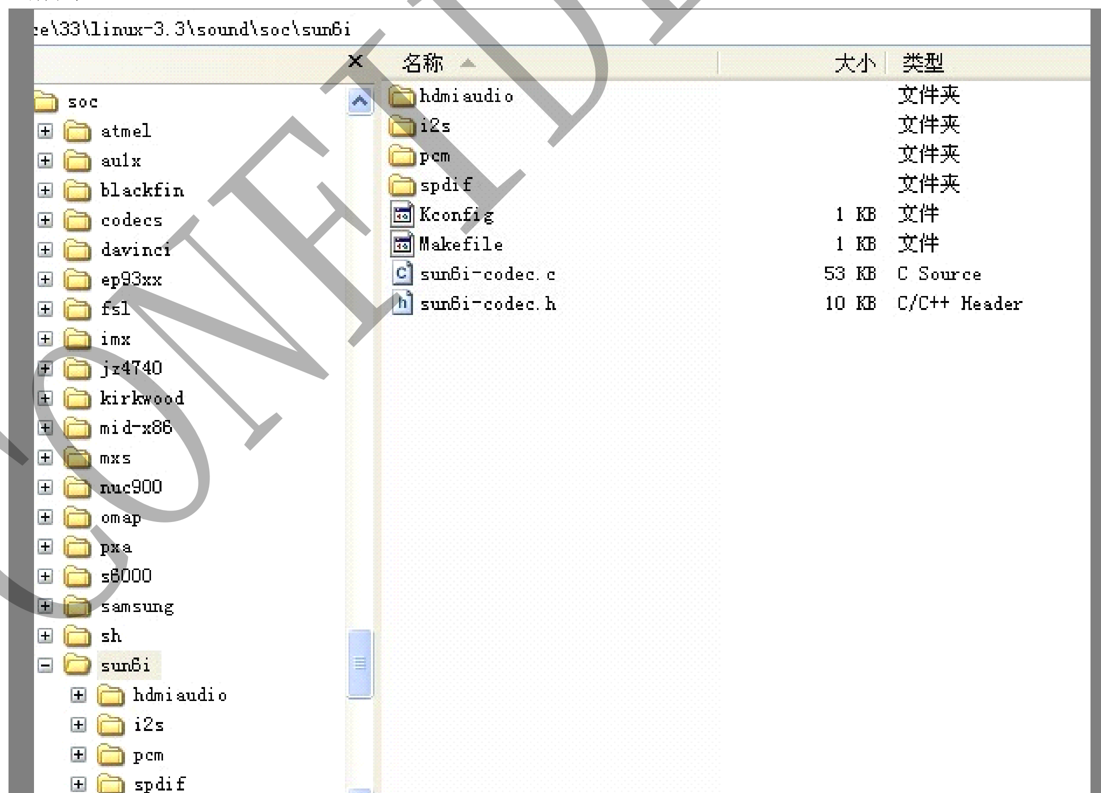

图 1 音频系统源码

CONFIDENTIAL

Copyright © 2011-2013 Allwinner Technology. All Rights Reserved.

\linux-3.3\drivers\staging\android\switch

名称	大小	类型	修改日期
.built-in.o.cmd	1 KB	Windows NT 命令...	2013-2-18 8:51
.switch_class.o.cmd	22 KB	Windows NT 命令...	2013-2-18 8:51
.switch_headset.o.cmd	26 KB	Windows NT 命令...	2013-2-18 8:51
built-in.o	123 KB	O 文件	2013-2-18 8:51
Kconfig	1 KB	文件	2013-2-18 8:43
Makefile	1 KB	文件	2013-2-18 8:43
modules.builtin	1 KB	BUILTIN 文件	2013-2-21 15:43
modules.order	0 KB	ORDER 文件	2013-2-22 17:43
switch.h	2 KB	H 文件	2013-2-18 8:43
switch_class.c	5 KB	C 文件	2013-2-18 8:43
switch_class.o	63 KB	O 文件	2013-2-18 8:51
switch_gpio.c	5 KB	C 文件	2013-2-18 8:43
switch_headset.c	11 KB	C 文件	2013-2-18 8:43
switch_headset.o	67 KB	O 文件	2013-2-18 8:51

图 2.Switch 源码

2.3. audio 相关术语介绍

Audio Driver: Acronyms	
Acronym	**Definition**
ALSA	Advanced Linux Sound Architecture
DMA	即直接内存存取，指数据不经 cpu，直接在设备和内存，内存和内存，设备和设备之间传输.
OSS	Open Sound System
样本长度 sample	样本是记录音频数据最基本的单位，常见的有 8 位和 16 位
通道数 channel	该参数为 1 表示单声道，2 则是立体声。
帧 frame	帧记录了一个声音单元，其长度为样本长度与通道数的乘积。
采样率 rate	每秒钟采样次数，该次数是针对帧而言。
周期 period	音频设备一次处理所需要的帧数，对于音频设备的数据访问以及音频数据的存储，都是以此为单位。
交错模式 interleaved	是一种音频数据的记录模式，在交错模式下，数据以连续帧的形式存放，即首先记录完帧 1 的左声道样本和右声道样本（假设为立体声格式），再开始帧 2 的记录，而在非交错模式下，首先记录的是一个周期内所有帧的左声道样本，再记录右声道样本，数据是以连续通道的方式存储。不过多数情况下，我们只需要使用交错模式就可以了。
Audiocodec	芯片内置音频接口
Hdmiaudio	内置 hdmi 音频接口
Spdif	外置音响音频设备接口
I2s	外置音频通道接口

CONFIDENTIAL

Copyright © 2011-2013 Allwinner Technology. All Rights Reserved.

2.4. audio 模块配置介绍

2.4.1. Menuconfig 配置

在编译服务器上，目录为\lichee\linux-3.3 上，输入命令：

```
make ARCH=arm menuconfig
```

如下所示：

```
/lichee/linux-3.3$ make ARCH=arm menuconfig
```

- 音频驱动配置

音频模块中包括 audiocodec，hdmiaudio，spdif，i2s，pcm 共 5 个音频驱动。他们的配置项如图 3 到图 7 所示。其中 audiocodec，hdmiaudio buildin 到内核中，spdif，i2s，pcm 配置成模块的方式。

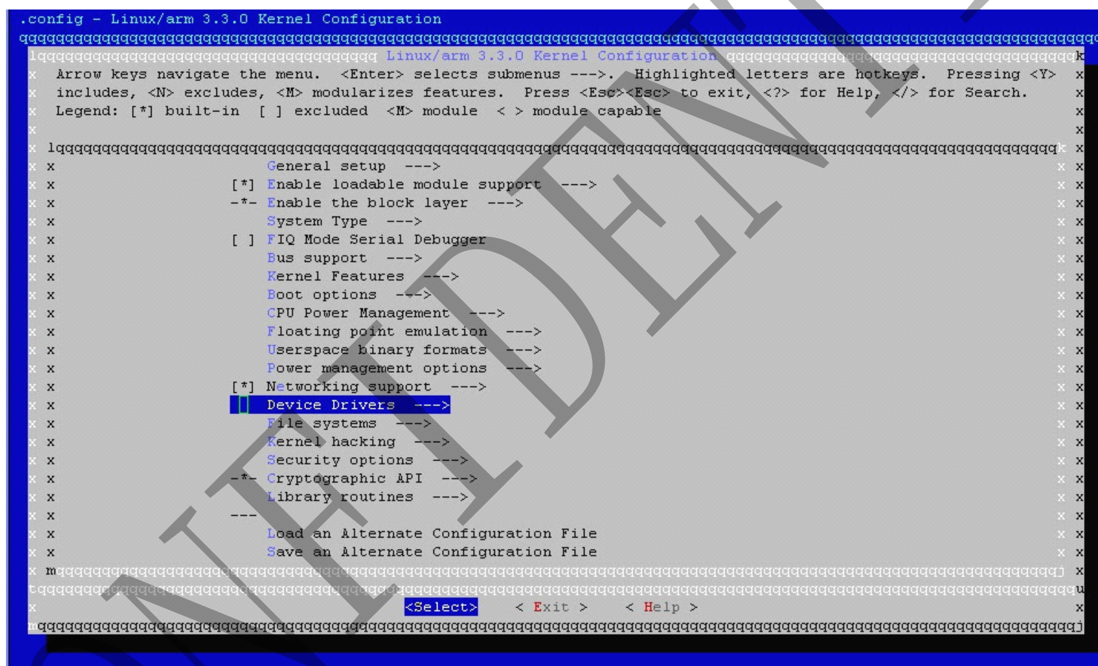

图 3.Device Drivers 选项配置

CONFIDENTIAL

Copyright © 2011-2013 Allwinner Technology. All Rights Reserved.

```
.config - Linux/arm 3.3.0 Kernel Configuration
                              Device Drivers
  Arrow keys navigate the menu.  <Enter> selects submenus --->.  Highlighted letters are hotkeys.  Pressing <Y>
  includes, <N> excludes, <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.
  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

        Generic Driver Options  --->
    < > Connector - unified userspace <-> kernelspace linker  --->
    < > Memory Technology Device (MTD) support  --->
    < > Parallel port support  --->
    [*] Block devices  --->
        Misc devices  --->
        SCSI device support  --->
    < > Serial ATA and Parallel ATA drivers  --->
    [*] Multiple devices driver support (RAID and LVM)  --->
    < > Generic Target Core Mod (TCM) and ConfigFS Infrastructure  --->
    [*] Network device support  --->
    [ ] ISDN support  --->
    < > Telephony support  --->
        Input device support  --->
        Character devices  --->
    {*} I2C support  --->
    [ ] SPI support  --->
        PPS support  --->
        PTP clock support  --->
    -*- GPIO Support  --->
    < > Dallas's 1-wire support  --->
    <*> Power supply class support  --->
    <*> Hardware Monitoring support  --->
    < > Generic Thermal sysfs driver  --->
    [ ] Watchdog Timer Support  --->
        Sonics Silicon Backplane  --->
        Broadcom specific AMBA  --->
        Multifunction device drivers  --->
    [*] Voltage and Current Regulator Support  --->
    <*> Multimedia support  --->
        Graphics support  --->
    <*> Sound card support  --->
    [*] HID Devices  --->
    [*] USB support  --->
    <*> MMC/SD/SDIO card support  --->
    < > Sony MemoryStick card support (EXPERIMENTAL)  --->
    [ ] LED Support  --->

                    <Select>    < Exit >    < Help >
```

图 4.音频模块配置

```
config - Linux/arm 3.3.0 Kernel Configuration
                              Sound card support
  Arrow keys navigate the menu.  <Enter> selects submenus --->.  Highlighted letters are hotkeys.  Pressing <Y>
  includes, <N> excludes, <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.
  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

    --- Sound card support
    <*>   Advanced Linux Sound Architecture  --->
    < >   Open Sound System (DEPRECATED)  --->

                    <Select>    < Exit >    < Help >
```

图 5.音频 ALSA 模块配置

CONFIDENTIAL

Copyright © 2011-2013 Allwinner Technology. All Rights Reserved.

```
config - Linux/arm 3.3.0 Kernel Configuration
                         Advanced Linux Sound Architecture
  Arrow keys navigate the menu.  <Enter> selects submenus --->.  Highlighted letters are hotkeys.  Pressing <Y>
  includes, <N> excludes, <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.
  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

                      --- Advanced Linux Sound Architecture
                      < >   Sequencer support
                      < >   OSS Mixer API
                      < >   OSS PCM (digital audio) API
                      < >   HR-timer backend support
                      [ ]   Dynamic device file minor numbers
                      [*]   Support old ALSA API
                      [*]   Verbose procfs contents
                      [ ]   Verbose printk
                      [ ]   Debug
                      [*]   Generic sound devices  --->
                      [*]   ARM sound devices  --->
                      [*]   USB sound devices  --->
                      <*>   ALSA for SoC audio support  --->

                        <Select>    < Exit >    < Help >
```

图 6.音频 soc 模块配置

```
config - Linux/arm 3.3.0 Kernel Configuration
                         ALSA for SoC audio support
  Arrow keys navigate the menu.  <Enter> selects submenus --->.  Highlighted letters are hotkeys.  Pressing <Y>
  includes, <N> excludes, <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.
  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

                      --- ALSA for SoC audio support
                      <*>   APB On-Chip Codec
                      <*>   HDMI Audio for the ReuuiMlla SUN6I chips
                      <M>   sun6i On-Chip spdif
                      <M>   SoC i2s interface for the ReuuiMlla SUN6I chips
                      <M>   SoC pcm interface for the ReuuiMlla SUN6I chips
                      < >   Build all ASoC CODEC drivers

                        <Select>    < Exit >    < Help >
```

图 7.音频音频驱动配置

- Hdmi 配置

如果要编译 hdmiaudio，也需要将 hdmi video 编译到内核中。如图 8，图 9 所示。进入 device driver 后，选择 graphics support，然后选择 buildin HDMI driver support（sun6i）即可。

```
config - Linux/arm 3.3.0 Kernel Configuration
                         Device Drivers
  Arrow keys navigate the menu.  <Enter> selects submenus --->.  Highlighted letters are hotkeys.  Pressing <Y>
  includes, <N> excludes, <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.
  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

                      <*> Hardware Monitoring support  --->
                      < > Generic Thermal sysfs driver  --->
                      [ ] Watchdog Timer Support  --->
                          Sonics Silicon Backplane  --->
                          Broadcom specific AMBA  --->
                          Multifunction device drivers  --->
                      [*] Voltage and Current Regulator Support  --->
                      <*> Multimedia support  --->
                          Graphics support  --->
                      <*> Sound card support  --->
                      [*] HID Devices  --->
                      [*] USB support  --->
                      <*> MMC/SD/SDIO card support  --->
                      < > Sony MemoryStick card support (EXPERIMENTAL)  --->
                      [ ] LED Support  --->
                      < > Switch class support  --->

                        <Select>    < Exit >    < Help >
```

图 8 graphics support

CONFIDENTIAL

Copyright © 2011-2013 Allwinner Technology. All Rights Reserved.

```
qqqqqqqqqqqqqqqqqqqqqqqqqqq Graphics support qqqqqqqqqqqqqqqqqqqqqqqqqqq
nus --->.  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> ex
built-in  [ ] excluded  <M> module  < > module capable

qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
 < > Direct Rendering Manager (XFree86 4.1.0 and higher DRI support)  --->
 <*> Ion Memory Manager  --->
 < > Lowlevel video output switch controls
 <*> Support for frame buffer devices  --->
 <M> LCD Driver Support(sun6i)
 <*> HDMI Driver Support(sun6i)
 < > ARM PrimeCell PL110 support
 < > Epson S1D13XXX framebuffer support
 < > SMSC UFX6000/7000 USB Framebuffer support
 < > Displaylink USB Framebuffer support
 < > Virtual Frame Buffer support (ONLY FOR TESTING!)
 < > E-Ink Metronome/8track controller support
 < > E-Ink Broadsheet/Epson S1D13521 controller support
 [ ] Backlight & LCD device support  --->
     Console display driver support  --->
 [ ] Bootup logo  --->
```

图 9 hdmi driver

- 耳机检测驱动配置

在 android 系统中，支持耳机动态检测。耳机检测驱动配置如图 10~图 12 所示。

```
config - Linux/arm 3.3.0 Kernel Configuration
 Device Drivers
  Arrow keys navigate the menu.  <Enter> selects submenus --->.  Highlighted letters are hotkeys.  Pressing <Y>
  includes, <N> excludes, <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.
  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

                    [*] HID Devices  --->
                    [*] USB support  --->
                    <*> MMC/SD/SDIO card support  --->
                    < > Sony MemoryStick card support (EXPERIMENTAL)  --->
                    [ ] LED Support  --->
                    < > Switch class support  --->
                    [ ] Accessibility support  --->
                    [*] Real Time Clock  --->
                    [ ] DMA Engine support  --->
                    [ ] Auxiliary Display support  --->
                    < > Userspace I/O drivers  --->
                        Virtio drivers  --->
                        Microsoft Hyper-V guest support  --->
                    [*] Staging drivers  --->
                        Hardware Spinlock drivers  --->
                    [ ] IOMMU Hardware Support  --->

                        <Select>    < Exit >    < Help >
```

图 10 staging driver

CONFIDENTIAL

Copyright © 2011-2013 Allwinner Technology. All Rights Reserved.

```
config - Linux/arm 3.3.0 Kernel Configuration
                         Staging drivers
 Arrow keys navigate the menu.  <Enter> selects submenus --->.  Highlighted letters are hotkeys.  Pressing <Y>
 includes, <N> excludes, <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.
 Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

                     < >   Frontier Tranzport and Alphatrack support
                     < >   Line6 USB support  --->
                     < >   USB Quatech ESU-100 8 Port Serial Driver
                     < >   USB Quatech xSU2-[14]00 USB Serial Driver
                     < >   VIA Technologies VT6656 support
                     < >   Industrial I/O support  --->
                     < >   Compressed RAM block device support
                     < >   Silicon Motion SM7XX Frame Buffer Support
                     < >   USB ENE SM card reader support
                     < >   Beceem BCS200/BCS220-3 and BCSM250 wimax support
                     < >   Drivers for Flarion ft1000 devices
                           Speakup console speech  --->
                     < >   Synaptics Clearpad TM1217
                     < >   Synaptics i2c rmi4 touchscreen
                     [ ]   Media staging drivers  --->
                     [ ]   Android  --->

                       <Select>    < Exit >    < Help >
```

图 11.android 配置

```
config - Linux/arm 3.3.0 Kernel Configuration
                         Android
 Arrow keys navigate the menu.  <Enter> selects submenus --->.  Highlighted letters are hotkeys.  Pressing <Y>
 includes, <N> excludes, <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.
 Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

                     [*] Android Drivers
                     [*]   Android Binder IPC Driver
                     [*]   Enable the Anonymous Shared Memory Subsystem
                     <*>   Android log driver
                     [*]   Android RAM buffer console
                     [ ]   Persistent function tracer
                     [*]   Timed output class driver
                     <*>     Android timed gpio driver
                     [*]   Android Low Memory Killer
                     <*>   Android Switch class support  --->
                     <*>   swith headset support
                     < >   Android GPIO Switch support

                       <Select>    < Exit >    < Help >
```

图 12 耳机检测驱动配置

2.4.2. Sysconfig.fex 配置

配置文件的位置：\lichee\tools\pack\chips\sun6i\configs\android\fiber-XXX 目录下。音频中需要配置的有 audiocodec，i2s，pcm，spdif 四个音频驱动。

2.4.2.1. Audiocodec 配置

;pa_val: 31 level,1.5db/step.0x1b is the large volume.
;cap_val:0x0--0db,0x1--24db,0x2--27db,0x3--30db,0x4--33db,0x5--36db,0x6--39db,0x7--42db
;headphone_vol must least than 0x3b. 0x1 is a step.
[audio_para]
audio_used = 1
headphone_vol = 0x3b
cap_vol = 0x5
pa_single_vol = 0x19
pa_double_used = 0
pa_double_vol = 0x1f

CONFIDENTIAL

Copyright © 2011-2013 Allwinner Technology. All Rights Reserved.

headphone_direct_used = 0
headset_mic_vol = 0x5
audio_pa_ctrl = port:PA18<1><default><default><0>

配置项	配置项含义
audio_used =xx	Audiocodec 是否使用， 1：打开（默认）0：关闭
headphone_vol	耳机音量大小
cap_vol	录音音量大小
pa_single_vol	喇叭音量大小（硬件只有一个喇叭）
pa_double_used	硬件是否支持双喇叭配置
pa_double_vol	双喇叭音量大小（如果只有一个喇叭，这个可以不配置）
headphone_direct_used	耳机的直驱交驱选择（建议 3g-phone 方案选择交驱，pad 方案选择直驱）
headset_mic_vol	耳机 mic 大小
audio_pa_ctrl=xx	喇叭的 gpio 口控制。

2.4.2.2. I2s 配置

I2s 可以配置成 pcm 模式，也可以配置成 i2s 模式。
[i2s_para]
i2s_used = 1
i2s_channel = 2
i2s_master = 1
i2s_select = 0
audio_format = 4
signal_inversion = 3
over_sample_rate = 256
sample_resolution = 16
word_select_size = 32
pcm_sync_period = 256
msb_lsb_first = 0
sign_extend = 0
slot_index = 0
slot_width = 16
frame_width = 1
tx_data_mode = 0
rx_data_mode = 0
i2s_mclk =
i2s_bclk = port:PB01<2><1><default><default>
i2s_lrclk = port:PB02<2><1><default><default>
i2s_dout0 = port:PB03<2><1><default><default>
i2s_dout1 =

CONFIDENTIAL

Copyright © 2011-2013 Allwinner Technology. All Rights Reserved.

i2s_dout2 =
i2s_dout3 =
i2s_din = port:PB07<2><1><default><default>

配置项	配置项含义
i2s_used	xx 为 1 时加载该模块，为 0 时不加载[default value: 1]
i2s_channel	声道控制[default value: 2]
i2s_master	主从模式配置： [1: SND_SOC_DAIFMT_CBM_CFM(codec clk & FRM master) SOC as slave & codec as master] [4: SND_SOC_DAIFMT_CBS_CFS(codec clk & FRM slave) SOC as master & codec as slave]
i2s_select	Pcm 和 i2s 模式选择[0：pcm 1：i2s]
audio_format	音频格式选择： [1: SND_SOC_DAIFMT_I2S(standard i2s format)] [2: SND_SOC_DAIFMT_RIGHT_J(right justfied format)] [3: SND_SOC_DAIFMT_LEFT_J(left justfied format)] [4: SND_SOC_DAIFMT_DSP_A(pcm. MSB is available on 2nd BCLK rising edge after LRC rising edge)] [5: SND_SOC_DAIFMT_DSP_B(pcm. MSB is available on 1nd BCLK rising edge after LRC rising edge)]
signal_inversion	音频信号模式选择： [1: SND_SOC_DAIFMT_NB_NF(normal bit clock + frame)] [2: SND_SOC_DAIFMT_NB_IF(normal BCLK + inv FRM)] [3: SND_SOC_DAIFMT_IB_NF(invert BCLK + nor FRM)] [4: SND_SOC_DAIFMT_IB_IF(invert BCLK + FRM)]
over_sample_rate	音频过采样率选择： support 128fs/192fs/256fs/384fs/512fs/768fs [default value: 256]
sample_resolution	采样精度选择： 16bits/20bits/24bits[default value: 16]
word_select_size	位宽选择： 16bits/24bits/32bits[default value: 32]
pcm_sync_period	PCM 周期长度选择： 16/32/64/128/256 BCLKs [default value: 256]
msb_lsb_first	数据位模式选择： [0: msb first; 1: lsb first][default value: 0]
sign_extend	标志位扩展模式： [0: zero pending; 1: sign extend][default value: 0]
slot_index	时间片索引： [0: the 1st slot - 3: the 4th slot][default value: 0]
slot_width	时间片宽度： [8 bit width / 16 bit width][default value: 16]
frame_width	帧宽度： [0: long frame = 2 BCLK width; 1: short frame = 1 BCLK width][default value: 1]

CONFIDENTIAL

Copyright © 2011-2013 Allwinner Technology. All Rights Reserved.

tx_data_mode	Tx 数据传输模式： [0: 16bit linear PCM; 1: 8bit linear PCM; 2: 8bit u-law; 3: 8bit a-law][default value: 0]
rx_data_mode	Rx 数据传输模式： [0: 16bit linear PCM; 1: 8bit linear PCM; 2: 8bit u-law; 3: 8bit a-law][default value: 0]
i2s_mclk	I2sMCLK 信号的 GPIO 配置
i2s_bclk	I2sBCLK 信号的 GPIO 配置
i2s_lrclk	I2sLRCK 信号的 GPIO 配置
i2s_dout0	I2S out0 的 GPIO 配置
i2s_dout1	暂不使用
i2s_dout2	暂不使用
i2s_dout3	暂不使用
i2s_din	I2sIN 信号的 GPIO 配置

2.4.2.3. pcm 配置

pcm 可以配置成 pcm 模式，也可以配置成 i2s 模式。

```
[pcm_para]
pcm_used            = 1
pcm_channel         = 2
pcm_master          = 4
pcm_select          = 1
audio_format        = 4
signal_inversion    = 3
over_sample_rate    = 512
sample_resolution   = 16
word_select_size    = 32
pcm_sync_period     = 64
msb_lsb_first       = 0
sign_extend         = 0
slot_index          = 0
slot_width          = 16
frame_width         = 1
tx_data_mode        = 0
rx_data_mode        = 0
pcm_mclk            =
pcm_bclk            = port:PG13<3><1><default><default>
pcm_lrclk           = port:PG14<3><1><default><default>
pcm_dout            = port:PG16<3><1><default><default>
pcm_din             = port:PG15<3><1><default><default>
```

配置项	配置项含义
pcm_used	xx 为 1 时加载该模块，为 0 时不加载[default value: 1]
pcm_channel	声道控制[default value: 2]
Pcm_master	主从模式配置： [1: SND_SOC_DAIFMT_CBM_CFM(codec clk & FRM master) SOC

CONFIDENTIAL

Copyright © 2011-2013 Allwinner Technology. All Rights Reserved.

	as slave & codec as master] [4: SND_SOC_DAIFMT_CBS_CFS(codec clk & FRM slave) SOC as master & codec as slave]
pcm_select	Pcm 和 i2s 模式选择[1：pcm 0：i2s]
audio_format	音频格式选择 [default value: 4] [1: SND_SOC_DAIFMT_I2S(standard i2s format)] [2: SND_SOC_DAIFMT_RIGHT_J(right justfied format)] [3: SND_SOC_DAIFMT_LEFT_J(left justfied format)] [4: SND_SOC_DAIFMT_DSP_A(pcm. MSB is available on 2nd BCLK rising edge after LRC rising edge)] [5: SND_SOC_DAIFMT_DSP_B(pcm. MSB is available on 1nd BCLK rising edge after LRC rising edge)]
signal_inversion	音频信号模式选择 [default value: 3] [1: SND_SOC_DAIFMT_NB_NF(normal bit clock + frame)] [2: SND_SOC_DAIFMT_NB_IF(normal BCLK + inv FRM)] [3: SND_SOC_DAIFMT_IB_NF(invert BCLK + nor FRM)] [4: SND_SOC_DAIFMT_IB_IF(invert BCLK + FRM)]
over_sample_rate	音频过采样率选择： support 128fs/192fs/256fs/384fs/512fs/768fs [default value: 512]
sample_resolution	采样精度选择： 16bits/20bits/24bits[default value: 16]
word_select_size	位宽选择： 16bits/24bits/32bits[default value: 32]
pcm_sync_period	PCM 周期长度选择： 16/32/64/128/256 BCLKs [default value: 64]
msb_lsb_first	数据位模式选择： [0: msb first; 1: lsb first][default value: 0]
sign_extend	标志位扩展模式： [0: zero pending; 1: sign extend][default value: 0]
slot_index	时间片索引： [0: the 1st slot - 3: the 4th slot][default value: 0]
slot_width	时间片宽度： [8 bit width / 16 bit width][default value: 16]
frame_width	帧宽度： [0: long frame = 2 BCLK width; 1: short frame = 1 BCLK width][default value: 1]
tx_data_mode	Tx 数据传输模式： [0: 16bit linear PCM; 1: 8bit linear PCM; 2: 8bit u-law; 3: 8bit a-law][default value: 0]
rx_data_mode	Rx 数据传输模式： [0: 16bit linear PCM; 1: 8bit linear PCM; 2: 8bit u-law; 3: 8bit a-law][default value: 0]
pcm_bclk=xx	pcmBCLK 信号的 GPIO 配置

CONFIDENTIAL

Copyright © 2011-2013 Allwinner Technology. All Rights Reserved.

pcm_lrclk =xx	pcmLRCK 信号的 GPIO 配置
pcm_dout	pcm out 的 GPIO 配置
pcm_din	pcmIN 信号的 GPIO 配置
pcm_mclk =xx	暂不使用
pcm_bclk=xx	pcmBCLK 信号的 GPIO 配置
pcm_lrclk =xx	pcmLRCK 信号的 GPIO 配置

2.4.2.4. Spdif 配置

```
[spdif_para]
spdif_used          = 1
spdif_dout          = port:PH28<3><1><default><default>
spdif_din           = port:PH27<3><1><default><default>
```

配置项	配置项含义
spdif_used=xx	xx 为 0 时加载该模块，为 0 是不加载
spdif_dout =xx	Spdif out 的 gpio 控制
spdif_din=xx	

CONFIDENTIAL

Copyright © 2011-2013 Allwinner Technology. All Rights Reserved.

3. 音频模块体系结构描述

在 a31 中有 5 个音频设备驱动，分别为 audiocodec，hdmiaudio，spdif，pcm，i2s 以及一个耳机检测驱动 switch。在 a31 中，基本的音频框架如图 4.A31 音频框架所示。

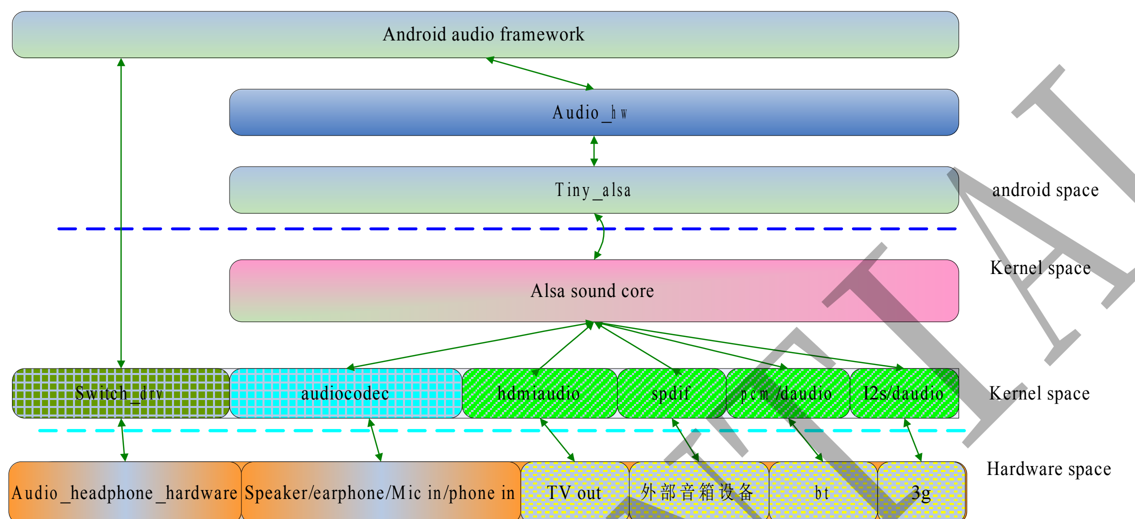

图 4.A31 音频框架

CONFIDENTIAL

Copyright © 2011-2013 Allwinner Technology. All Rights Reserved.

4. 接口描述

在音频框架中，audiocodec 属于模拟音频部分，hdmiaudio，spdif，pcm，i2s 属于数字音频。其中 audiocodec 在 3g 音频通话中支持模拟音频通路和通话录音功能接口。Pcm,i2s 都可以配置成 pcm 和 i2s 模式，代码中为了区分，一个以 pcm 命名，一个以 i2s 命名。Hdmiaudio，spdif 支持 raw data 模式。耳机检测驱动 switch 支持 android 标准的耳机检测接口。

4.1. Audiocodec 接口描述

在模拟音频驱动 audiocodec 中，支持 ADC 录音，DAC 播放，模拟音频通路。支持四路音频输入（mic1，mic2，phone in， line in），四路音频输出（phone out， headphone， earphone，Speaker）。如图 13.audiocodec 音频硬件通道所示。Audiocodec 中的音频通道接口也是根据图 13.audiocodec 音频硬件通道封装的。

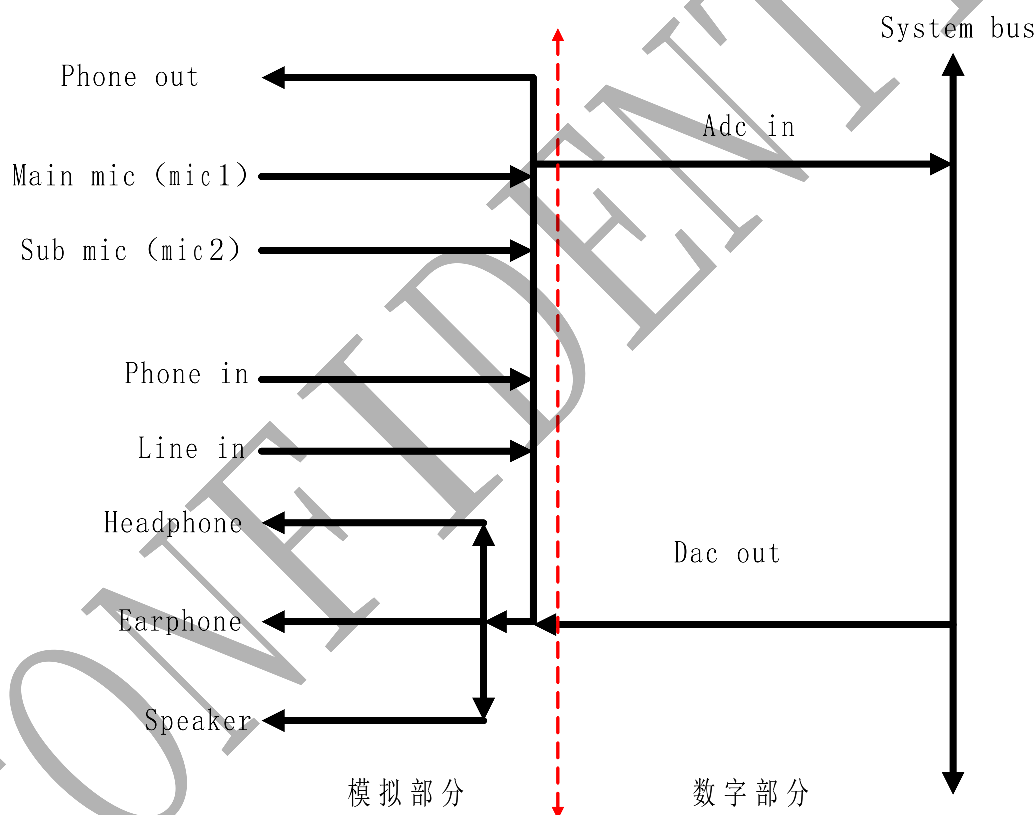

图 13.audiocodec 音频硬件通道

4.1.1. 系统音频播放接口

系统音频播放接口中，只有 headphone（耳机）跟 Speaker（喇叭）有使用到。Codec_set_spk 用于 pad 方案音频播放接口，codec_set_spk_headset 用于 phone 方案音频播放接口。

系统音频播放接口有三种：

- 1.从喇叭输出，如图 14.系统音频接口 dac out->speaker 绿色和蓝色部分所示。
- 2.从耳机输出，如图 14.系统音频接口 dac out->headphone 绿色和红色部分所示。

CONFIDENTIAL

- 3.从喇叭耳机输出，如图 14.系统音频接口 dac out->speaker&&headphone 绿色、蓝色和红色部分所示。

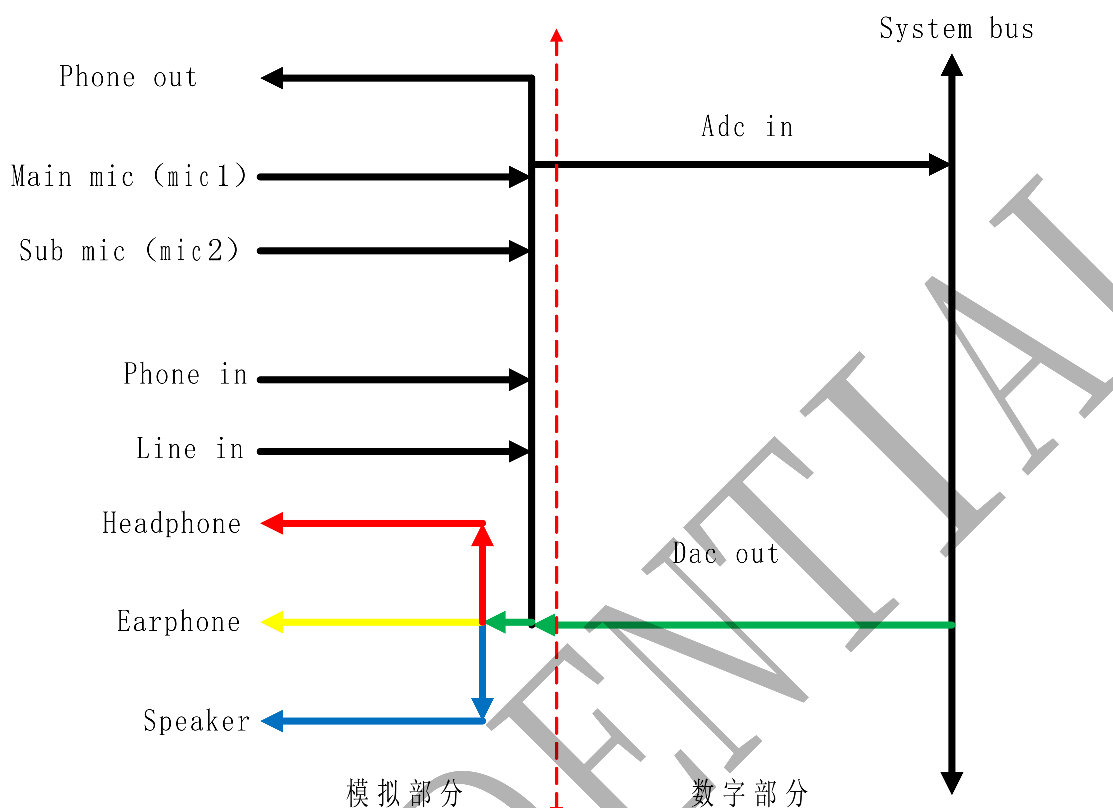

图 14.系统音频播放接口

4.1.1.1. codec_set_spk

- **Prototype:**

```
static int codec_set_spk(struct snd_kcontrol *kcontrol,
        struct snd_ctl_elem_value *ucontrol)
```

- **Arguments:**

Kcontrol:mixer 控制接口

Ucontrol：数据传输接口

- **Returns:**

Return 0.参数设置成功

- **Note:**

应用层通过 alsa lib 标准接口 mixer_ctl_set_value 进行调用，最后一个参数是传入 ucontrol 中的值，1 为喇叭，0 为耳机，参考 demo。

- **demo**

参考 android 层 audio_hw.c 中的 spk 接口封装

```
struct mixer *mixer;
struct mixer_ctl *audio_spk_switch;
/*step1.设置跟 audiocodec 音频驱动相同的名称*/
#define MIXER_AUDIO_SPK_SWITCH        "Audio Spk Switch"
```

CONFIDENTIAL

Copyright © 2011-2013 Allwinner Technology. All Rights Reserved.

```
/*step2.获取 audio spk switch 的接口*/
audio_spk_switch = mixer_get_ctl_by_name(mixer, MIXER_AUDIO_SPK_SWITCH);   if
(!audio_spk_switch) {
                ALOGE("Unable to find '%s' mixer control",MIXER_AUDIO_SPK_SWITCH);
                goto error_out;
     }
/*step3.设置音频的通路接口。
*  第三个参数表示通路选择：0：从耳机输出。打开红色部分
*                          1：从喇叭输出。打开蓝色部分。
*/
mixer_ctl_set_value(audio_spk_switch, 0, 0);
/*
*step4.设置完通路后，打开 dac 进行音频数据传输
*从耳机输出，如图 14.系统音频接口 dac out->headphone 绿色和红色部分所示。
*从喇叭输出，如图 14.系统音频接口  dac out->speaker 绿色和蓝色部分所示。
*/
```

4.1.1.2. codec_set_spk_headset

- **Prototype:**

  static int codec_set_spk_headset(struct snd_kcontrol *kcontrol,

  struct snd_ctl_elem_value *ucontrol)

- **Arguments:**

Kcontrol:mixer 控制接口

Ucontrol：数据传输接口

- **Returns:**

Return 0.参数设置成功

- **Note:**

应用层通过 alsa lib 标准接口 mixer_ctl_set_enum_by_stringe 进行调用，最后一个参数是传入 ucontrol 中的值；0：声音从耳机出，1：声音从喇叭出，2：声音从喇叭以及耳机同时出（用于插着耳机时候，电话来电铃声）。参考 demo。

- **demo**

参考 android 层 audio_hw.c 中的 spk 接口封装

```
struct mixer *mixer;
struct mixer_ctl *audio_spk_headset_switch;
/*step1.设置跟 audiocodec 音频驱动相同的名称*/
#define MIXER_AUDIO_NORMAL_SPEAKER_HEADSET                "Speaker Function"
/*step2.获取 audio_spk_headset_switch 的接口*/
     audio_spk_headset_switch = mixer_get_ctl_by_name(mixer,

MIXER_AUDIO_NORMAL_SPEAKER_HEADSET);
/*step3.设置音频的通路接口。
*  第二个参数表示通路选择：    ”spk_headset”表示从喇叭耳机同时出：打开红色和蓝色部分
*                              "spk"：表示声音从喇叭出：打开蓝色部分。
```

CONFIDENTIAL

```
*                                    "headset"：表示声音从耳机出：打开红色部分。
*/
mixer_ctl_set_enum_by_string(audio_spk_headset_switch, "spk_headset");
/*
*step4.设置完通路后，打开 dac 进行音频数据传输
*1.从喇叭输出，如图 14.系统音频接口 dac out->speaker 绿色和蓝色部分所示。
*2.从耳机输出，如图 14.系统音频接口 dac out->headphone 绿色和红色部分所示。
*3.从喇叭耳机输出，如图 14.系统音频接口 dac out->speaker&&headphone 绿色、蓝色和红色
*部分所示。
*/
```

4.1.2. 系统音频录音接口

系统音频录音接口，采用主 mic 进行录音，不用相关的接口设置，利用 alsa 标准的录音程序即可进行录音。音频通路如图 15 系统录音接口所示。

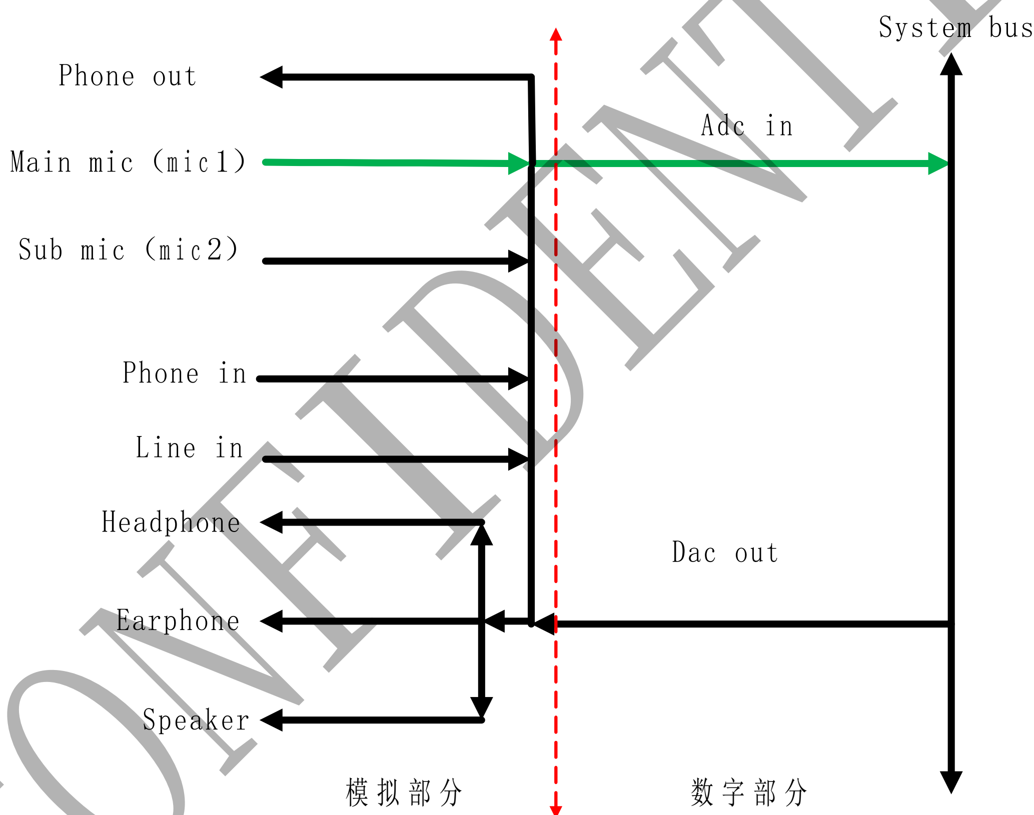

图 15 系统录音接口

4.1.3. 手机上行模拟通路接口

在手机上行接口中。可以通过 main mic，sub mic 以及从系统中来的 dac 数据通过 phone out 输出。如图 15 手机上行通路所示。设备通路分别为：

1.main mic->phone out 蓝色和绿色部分所示。
2.Sub mic->phone out 紫色和绿色部分所示。
3.Dac out->phone out 红色和绿色部分所示。

CONFIDENTIAL

Copyright © 2011-2013 Allwinner Technology. All Rights Reserved.

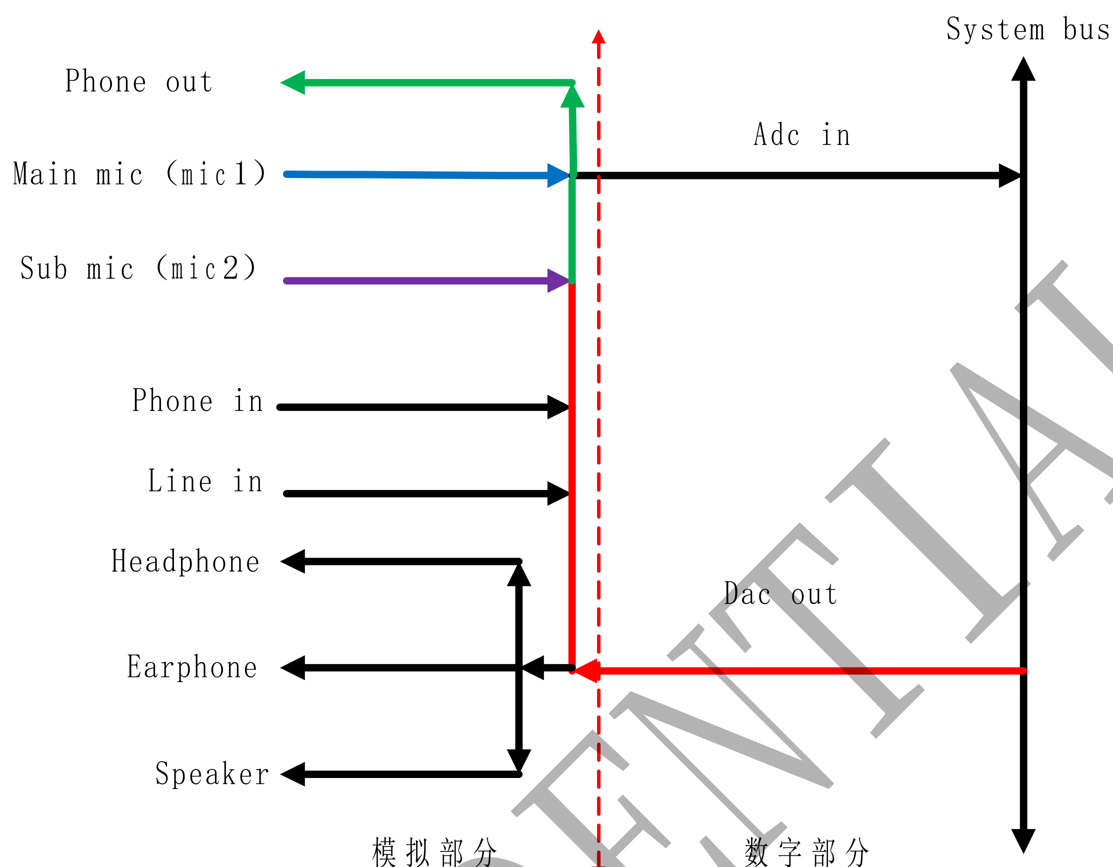

图 15 手机上行模拟通路接口

4.1.3.1. codec_set_phoneout

➢ **Prototype:**

static int codec_set_phoneout(struct snd_kcontrol *kcontrol,
    struct snd_ctl_elem_value *ucontrol)

➢ **Arguments:**

Kcontrol:mixer 控制接口

Ucontrol：数据传输接口

➢ **Returns:**

Return 0.参数设置成功

➢ **Note:**

应用层通过 alsa lib 中的 mixer_ctl_set_value 进行调用；

1：使能上行 phoneout。如图 15.手机上行模拟通路接口。绿色部分所示。

0：不使能上行 phoneout。

➢ **demo**

参考 android 层 audio_hw.c 中的 spk 接口封装

4.1.3.2. codec_set_phonemic

➢ **Prototype:**

static int codec_set_phonemic(struct snd_kcontrol *kcontrol,

Copyright © 2011-2013 Allwinner Technology. All Rights Reserved.

CONFIDENTIAL

struct snd_ctl_elem_value *ucontrol)

- **Arguments:**

Kcontrol:mixer 控制接口

Ucontrol：数据传输接口

- **Returns:**

Return 0.参数设置成功

- **Note:**

应用层通过 alsa lib 中的 mixer_ctl_set_value 进行调用；

1：打开 main mic phoneout。如图 15.手机上行模拟通路接口。蓝色部分所示。

0：关闭 main mic phoneout。

- **demo**

参考 android 层 audio_hw.c 中的 spk 接口封装

4.1.3.3. codec_set_headsetmic

- **Prototype:**

static int codec_set_headsetmic(struct snd_kcontrol *kcontrol,

struct snd_ctl_elem_value *ucontrol)

- **Arguments:**

Kcontrol:mixer 控制接口

Ucontrol：数据传输接口

- **Returns:**

Return 0.参数设置成功

- **Note:**

应用层通过 alsa lib 中的 mixer_ctl_set_value 进行调用；

1：打开 sub mic phoneout。例如耳机 mic。如图 15.手机上行模拟通路接口。紫色部分所示。

0：关闭 sub mic phoneout。

- **demo**

参考 android 层 audio_hw.c 中的 spk 接口封装

4.1.3.4. codec_set_dacphoneout

- **Prototype:**

static int codec_set_dacphoneout(struct snd_kcontrol *kcontrol,

struct snd_ctl_elem_value *ucontrol)

- **Arguments:**

Kcontrol:mixer 控制接口

Ucontrol：数据传输接口

- **Returns:**

Return 0.参数设置成功

- **Note:**

应用层通过 alsa lib 中的 mixer_ctl_set_value 进行调用；

1：打开 dac phoneout。如图 15.手机上行模拟通路接口。红色部分所示。

0：关闭 dac phoneout。

- **demo**

CONFIDENTIAL

参考 android 层 audio_hw.c 中的 spk 接口封装

4.1.4. 手机下行模拟通路接口

在手机下行接口中。可以通过 phone in 打开到耳机（headphone），听筒（earphone），喇叭（Speaker）的通路。也可以将 phone in 中的音频数据进行 adc 录音，进入系统中。如图 21 手机下行通路所示。设备通路分别为：

4.phone in->headphone 绿色和红色部分所示。

5.Phone in->earphone 绿色和黄色部分所示。

6.Phone in->Speaker 绿色和蓝色部分所示。

7.Phone in->adc in 绿色和紫色部分所示。

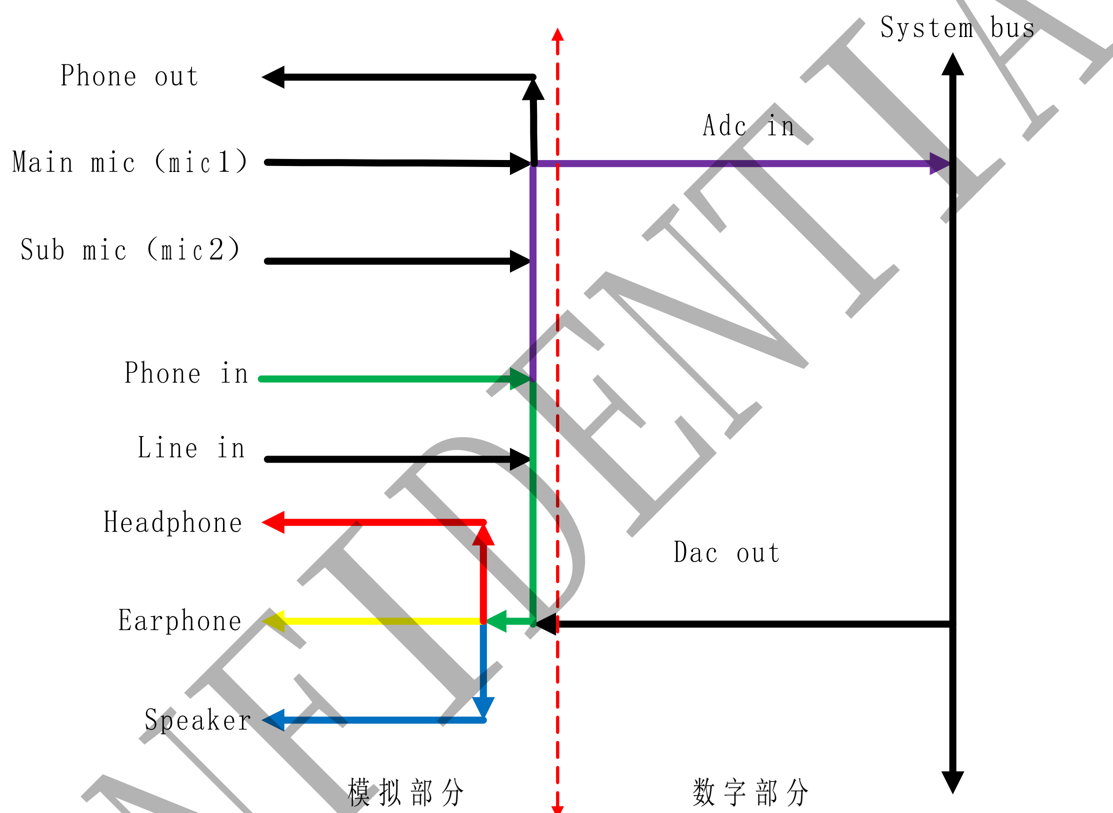

图 16.手机下行模拟通路接口

4.1.4.1. codec_set_phonein

- **Prototype:**

static int codec_set_phonein(struct snd_kcontrol *kcontrol,
    struct snd_ctl_elem_value *ucontrol)

- **Arguments:**

Kcontrol:mixer 控制接口

Ucontrol：数据传输接口

- **Returns:**

Return 0.参数设置成功

- **Note:**

CONFIDENTIAL

应用层通过 alsa lib 中的 mixer_ctl_set_value 进行调用；
1：打开 phone in。图 16.手机下行模拟通路接口。绿色部分所示。
0：关闭 phone in。

- **demo**

参考 android 层 audio_hw.c 中的 spk 接口封装

4.1.4.2. codec_set_speakerout

- **Prototype:**

static int codec_set_speakerout(struct snd_kcontrol *kcontrol,
struct snd_ctl_elem_value *ucontrol)

- **Arguments:**

Kcontrol:mixer 控制接口
Ucontrol：数据传输接口

- **Returns:**

Return 0.参数设置成功

- **Note:**

应用层通过 alsa lib 中的 mixer_ctl_set_value 进行调用；
1：打开 speaker out。图 16.手机下行模拟通路接口。蓝色部分所示。
0：关闭 speaker out。

- **demo**

参考 android 层 audio_hw.c 中的 spk 接口封装

4.1.4.3. codec_set_headphoneout

- **Prototype:**

static int codec_set_headphoneout(struct snd_kcontrol *kcontrol,
struct snd_ctl_elem_value *ucontrol)

- **Arguments:**

Kcontrol:mixer 控制接口
Ucontrol：数据传输接口

- **Returns:**

Return 0.参数设置成功

- **Note:**

应用层通过 alsa lib 中的 mixer_ctl_set_value 进行调用；
1：打开 headphone out。图 16.手机下行模拟通路接口。红色部分所示。
0：关闭 headphone out。

- **demo**

参考 android 层 audio_hw.c 中的 spk 接口封装

4.1.4.4. codec_set_earpieceout

- **Prototype:**

static int codec_set_earpieceout(struct snd_kcontrol *kcontrol,
struct snd_ctl_elem_value *ucontrol)

- **Arguments:**

CONFIDENTIAL

Copyright © 2011-2013 Allwinner Technology. All Rights Reserved.

Kcontrol:mixer 控制接口

Ucontrol：数据传输接口

- **Returns:**

Return 0.参数设置成功

- **Note:**

  应用层通过 alsa lib 中的 mixer_ctl_set_value 进行调用；

  1：打开 earpiece out。图 16.手机下行模拟通路接口。黄色部分所示。

  0：关闭 earpiece out。

- **demo**

参考 android 层 audio_hw.c 中的 spk 接口封装

4.1.4.5. codec_set_adcphonein

- **Prototype:**

static int codec_set_adcphonein(struct snd_kcontrol *kcontrol,

  struct snd_ctl_elem_value *ucontrol)

- **Arguments:**

Kcontrol:mixer 控制接口

Ucontrol：数据传输接口

- **Returns:**

Return 0.参数设置成功

- **Note:**

  应用层通过 alsa lib 中的 mixer_ctl_set_value 进行调用；

  1：打开 phone in adc in。图 16.手机下行模拟通路接口。紫色部分所示。

  0：关闭 phone in adc in。

- **demo**

参考 android 层 audio_hw.c 中的 spk 接口封装

4.1.5. 手机通话录音接口

在手机通话录音中。支持通话录音。在通话的时候，设置通话录音，然后进行 adc 采集音频数据即可。Phone out 和 phone in 的混音处理，在模拟部分由硬件进行混音。

CONFIDENTIAL

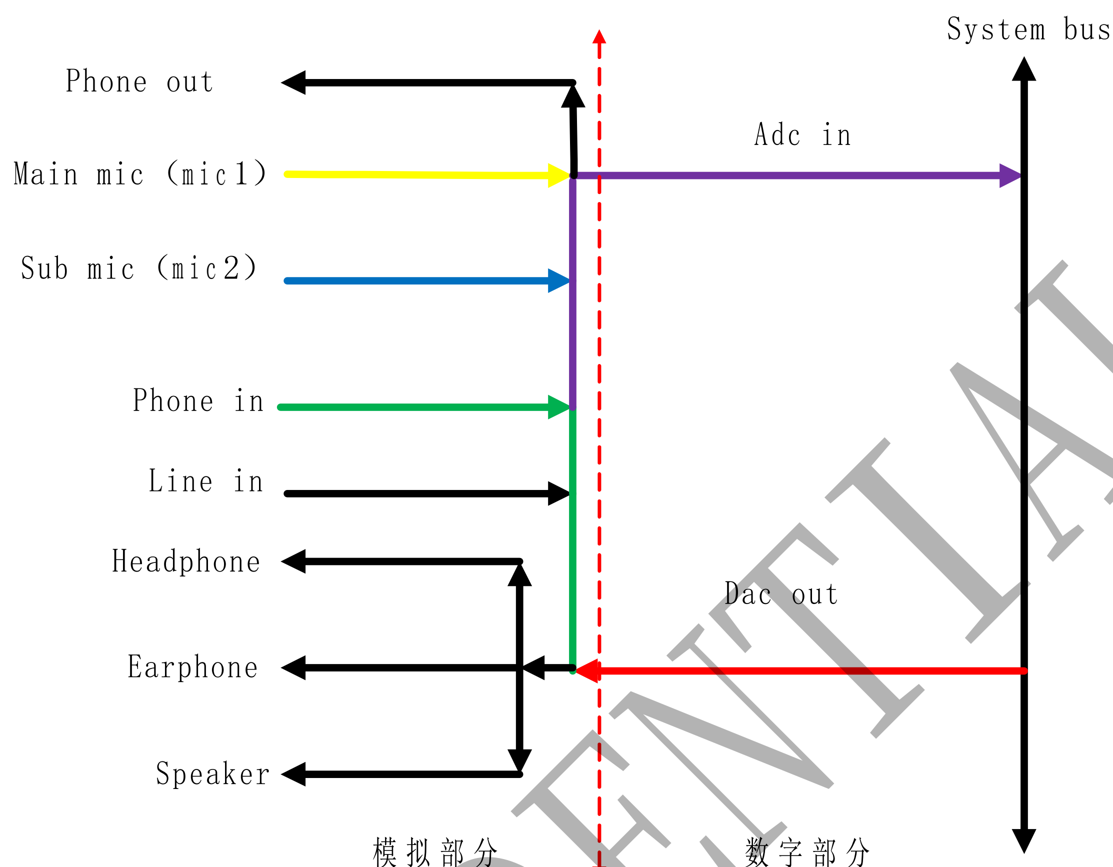

图 17 手机通话录音接口

4.1.5.1. codec_set_voicerecord

- **Prototype:**

static int codec_set_voicerecord(struct snd_kcontrol *kcontrol,
    struct snd_ctl_elem_value *ucontrol)

- **Arguments:**

Kcontrol:mixer 控制接口

Ucontrol：数据传输接口

- **Returns:**

Return 0.参数设置成功

- **Note:**

  应用层通过 alsa lib 中的 mixer_ctl_set_value 进行调用；

  1：打开通话录音。图 17.手机通话录音接口。紫色部分所示。

  0：关闭通话录音。

- **demo**

参考 android 层 audio_hw.c 中的 spk 接口封装

CONFIDENTIAL

4.1.6. 模拟音频扩展接口

4.1.6.1. codec_set_endcall

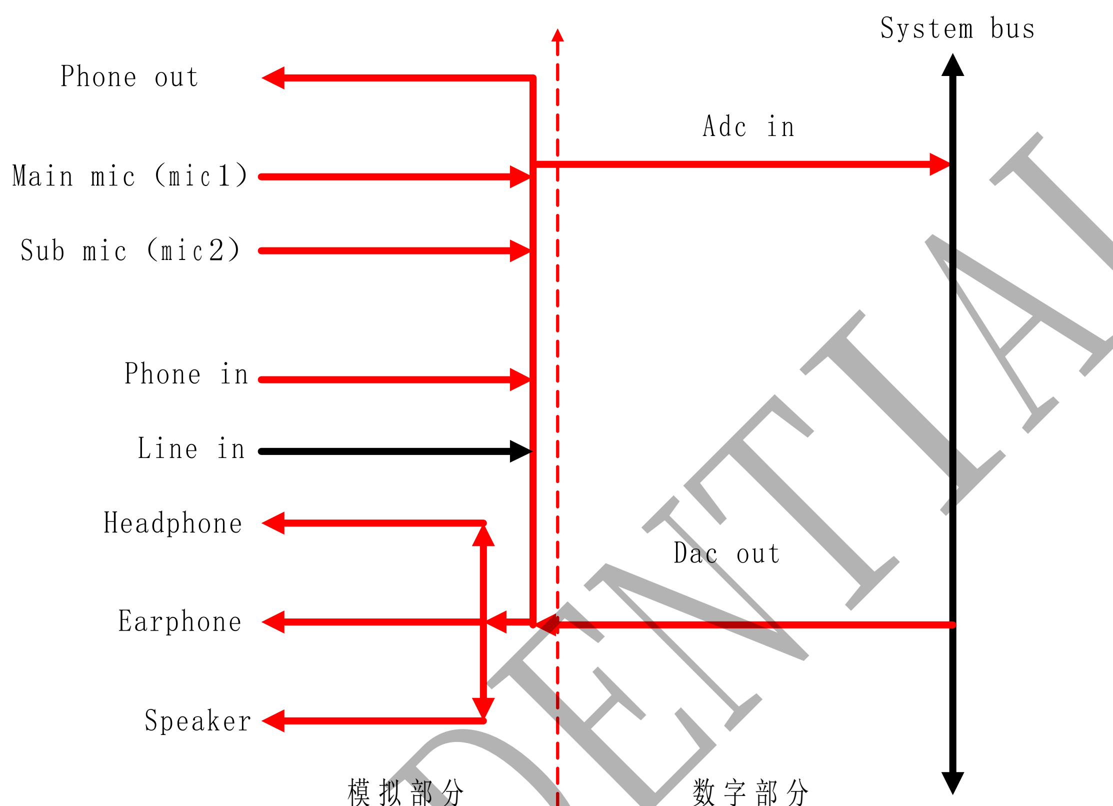

图 18 关闭音频通路

CONFIDENTIAL

- **Prototype:**

```
static int codec_set_endcall(struct snd_kcontrol *kcontrol,
    struct snd_ctl_elem_value *ucontrol)
```

- **Arguments:**

Kcontrol:mixer 控制接口

Ucontrol：数据传输接口

- **Returns:**

Return 0.参数设置成功

- **Note:**

  应用层通过 alsa lib 中的 mixer_ctl_set_value 进行调用；

  传入的参数无论是 0 还是 1，都会将所有打开的音频通路关闭。如图 18 红色部分所示。

- **demo**

参考 android 层 audio_hw.c 中的 spk 接口封装

4.1.6.2. codec_set_lineinin

Linein 通路可以用于模拟的 fm 通路，卡拉 ok 模式。

Copyright © 2011-2013 Allwinner Technology. All Rights Reserved.

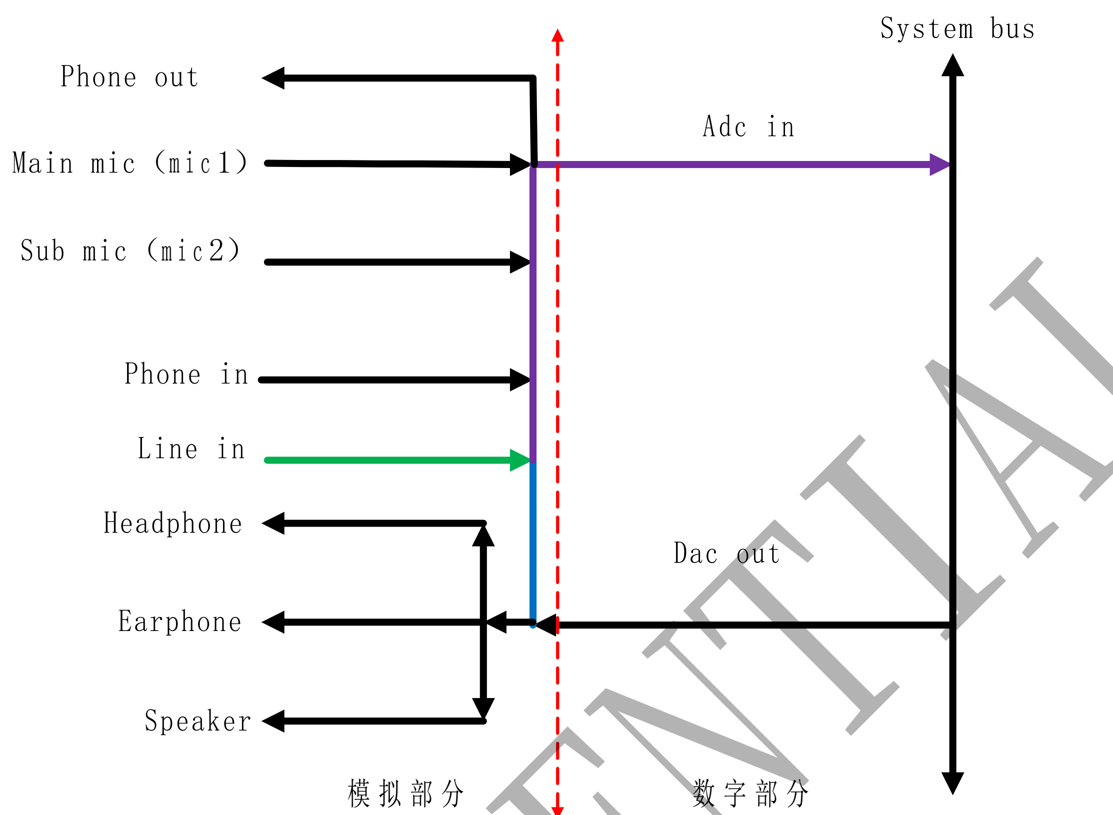

图 18.Linein 通路

➢ **Prototype:**

static int codec_set_lineinin(struct snd_kcontrol *kcontrol,
    struct snd_ctl_elem_value *ucontrol)

➢ **Arguments:**

Kcontrol:mixer 控制接口

Ucontrol：数据传输接口

➢ **Returns:**

Return 0.参数设置成功

➢ **Note:**

1：打开 linein。图 18.linein 通路。绿色和蓝色部分所示。

0：关闭 linein。

如果需要将 linein 里面的音频通过耳机，喇叭，听筒输出，参考手机现象模拟通路接口部分。

➢ **demo**

参考 android 层 audio_hw.c 中的 spk 接口封装

4.1.6.3. Codec_set_lineincap

➢ **Prototype:**

static int codec_set_lineincap(struct snd_kcontrol *kcontrol,
    struct snd_ctl_elem_value *ucontrol)

➢ **Arguments:**

CONFIDENTIAL

Kcontrol:mixer 控制接口

Ucontrol：数据传输接口

➢ **Returns：**

Return 0.参数设置成功

➢ **Note：**

1：打开 linein capture。图 18linein 通路。绿色和紫色部分所示。

0：关闭 linein capture。

可将 linein 进来的音频流通过 adc 录音进入系统处理。

➢ **demo**

参考 android 层 audio_hw.c 中的 spk 接口封装

4.1.6.4. codec_set_fm_speaker

Fm 的通路可以经过 linein 进来，也可以通过数字音频驱动 i2s 进入系统，然后再将数据通过 dac 输出到 headphone（耳机）或者 speaker（喇叭）。Fm 的系统接口，可以通过系统音频音频播放接口替代，这里为了兼容 phone 方案，保留这个接口。

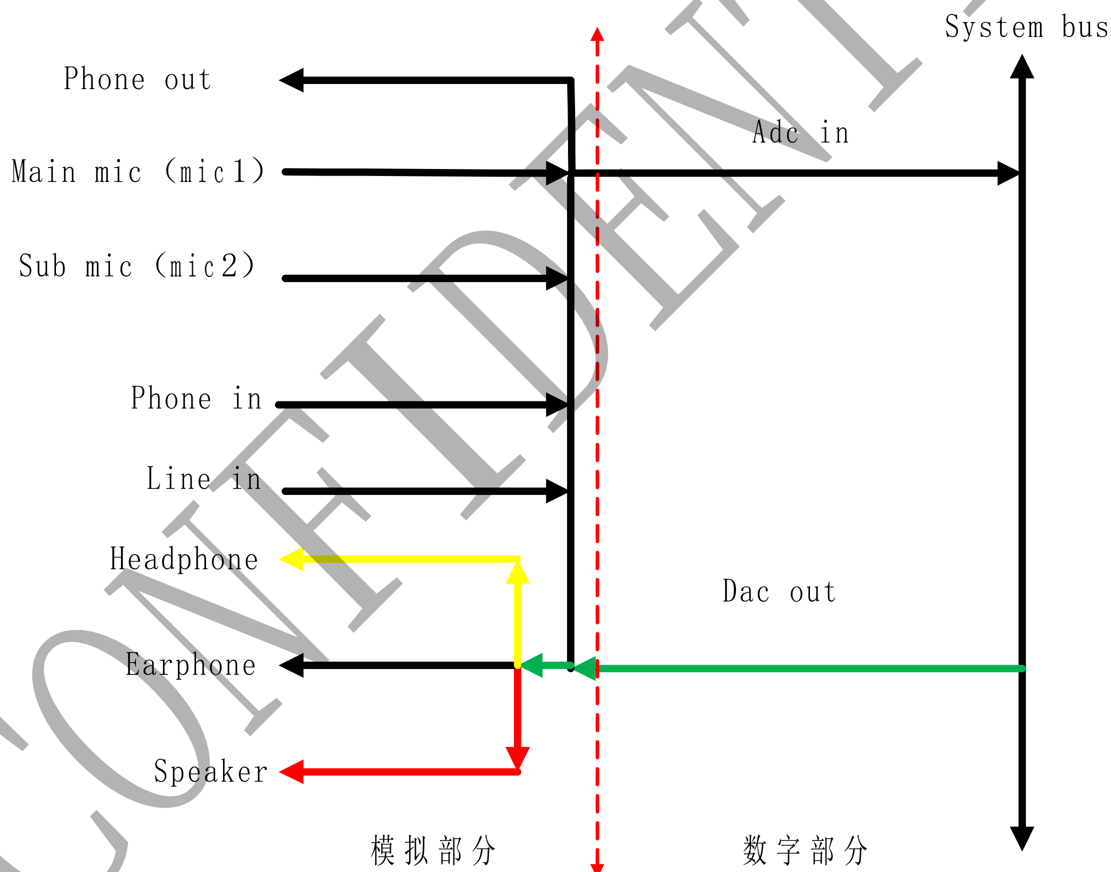

图 19fm 系统接口

➢ **Prototype：**

static int codec_set_fm_speaker(struct snd_kcontrol *kcontrol,

struct snd_ctl_elem_value *ucontrol)

➢ **Arguments：**

Kcontrol:mixer 控制接口

Copyright © 2011-2013 Allwinner Technology. All Rights Reserved.

Ucontrol：数据传输接口

- **Returns:**

Return 0.参数设置成功

- **Note:**

  1：打开 dac->speaker 的通路，图 19fm 系统接口，绿色和红色部分所示。
  0：关闭 dac->speaker 的通路

- **demo**

参考 android 层 audio_hw.c 中的 spk 接口封装

4.1.6.5. codec_set_fm_headset

- **Prototype:**

```
static int codec_set_fm_headset(struct snd_kcontrol *kcontrol,
	struct snd_ctl_elem_value *ucontrol)
```

- **Arguments:**

Kcontrol:mixer 控制接口
Ucontrol：数据传输接口

- **Returns:**

Return 0.参数设置成功

- **Note:**

  1：打开 dac->speaker 的通路，图 19fm 系统接口，绿色和黄色部分所示。
  0：关闭 dac->speaker 的通路

- **demo**

参考 android 层 audio_hw.c 中的 spk 接口封装

4.2. Hdmiaudio 接口描述

Hdmiaudio 中，支持 raw data 和 pcm data 模式。
当音频 channel 设置成 channels = 4 的时候，hdmiaudio 会相应设置成 raw data 模式。当音频 channel 设置成 channels = 2 或者 1 的时候，hdmiaudio 会设置成 pcm data 模式。
Hdmiaudio 是一个独立的音频驱动，接口支持 alsa lib 中的标准接口，不在这里一一列举。

4.3. spdif 接口描述

spdif 中，支持 raw data 和 pcm data 模式。
当音频 channel 设置成 channels = 4 的时候，spdif 会相应设置成 raw data 模式。当音频 channel 设置成 channels = 2 或者 1 的时候，spdif 会设置成 pcm data 模式。
spdif 是一个独立的音频驱动，接口支持 alsa lib 中的标准接口，不在这里一一列举。

4.4. I2s 接口描述

I2s 可以配置成 pcm 模式和 i2s 模式，具体参考 2.4.2.2 中 i2s 的 sysconfig 配置。

CONFIDENTIAL

Copyright © 2011-2013 Allwinner Technology. All Rights Reserved.

4.5. pcm 接口描述

I2s 可以配置成 pcm 模式和 i2s 模式，具体参考 2.4.2.3 中 pcm 的 sysconfig 配置。

4.6. switch 接口描述

Switch 接口支持 android 标准的耳机检测。支持 3 段耳机和四段耳机的插拔检测功能。不再描述。

CONFIDENTIAL

Copyright © 2011-2013 Allwinner Technology. All Rights Reserved.

5. 模块开发 demo

音频的外部接口跟普通的驱动不同。音频的 application 需要额外的 alsa-lib 进行外部接口的封装。在 android2.3.4 中，用的是 small alsa 应用库，而在 android4.0 以后，采用 tiny alsa 应用库进行外部接口的封装。所有的接口都是音频的标准接口。在这里不一一罗列。下面给出播放和录音的应用.

写一个音频应用程序涉及到以下几步

- opening the audio device
- set the parameters of the device
- receive audio data from the device or deliver audio data to the device
- close the device

在 a31 中，涉及 5 个音频驱动，alsa 的 lib 库支持任何一个音频驱动。请参考最小 playback 应用和录音应用以及 mixer 接口的使用方法。
demo 采用 tiny_alsa 库，可以从\android4.1\external\tinyalsa 中获得。

5.1. 最小的 playback 应用

```
/*
 * spdif test
 * play_high_rate-1633.c
 * (C) Copyright 2010-2016
 * reuuimllatech Technology Co., Ltd. <www.reuuimllatech.com>
 * huangxin <huangxin@reuuimllatech.com>
 *
 * some simple description for this code
 *
 * This program is free software; you can redistribute it and/or
 * modify it under the terms of the GNU General Public License as
 * published by the Free Software Foundation; either version 2 of
 * the License, or (at your option) any later version.
 *
 */

#include <include/tinyalsa/asoundlib.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <string.h>

#define ID_RIFF 0x46464952
#define ID_WAVE 0x45564157
```

CONFIDENTIAL

Copyright © 2011-2013 Allwinner Technology. All Rights Reserved.

```
#define ID_FMT     0x20746d66
#define ID_DATA 0x61746164

#define FORMAT_PCM 1

struct wav_header {
        uint32_t riff_id;
        uint32_t riff_sz;
        uint32_t riff_fmt;
        uint32_t fmt_id;
        uint32_t fmt_sz;
        uint16_t audio_format;
        uint16_t num_channels;
        uint32_t sample_rate;
        uint32_t byte_rate;
        uint16_t block_align;
        uint16_t bits_per_sample;
        uint32_t data_id;
        uint32_t data_sz;
};

void play_sample(FILE *file, unsigned int device, unsigned int channels,
                                 unsigned int rate, unsigned int bits);

int main(int argc, char **argv)
{
        FILE *file;
        struct wav_header header;
        unsigned int device = 0;

        if (argc < 2) {
                fprintf(stderr, "Usage: %s file.wav [-d device]\n", argv[0]);
                return 1;
        }

        file = fopen(argv[1], "rb");
        if (!file) {
                fprintf(stderr, "Unable to open file '%s'\n", argv[1]);
                return 1;
        }

        /* parse command line arguments */
        argv += 2;
        while (*argv) {
```

CONFIDENTIAL

Copyright © 2011-2013 Allwinner Technology. All Rights Reserved.

```
            if (strcmp(*argv, "-d") == 0) {
                argv++;
                device = atoi(*argv);
            }
            argv++;
        }

        fread(&header, sizeof(struct wav_header), 1, file);

        if ((header.riff_id != ID_RIFF) ||
            (header.riff_fmt != ID_WAVE) ||
            (header.fmt_id != ID_FMT) ||
            (header.audio_format != FORMAT_PCM) ||
            (header.fmt_sz != 16)) {
            fprintf(stderr, "Error: '%s' is not a PCM riff/wave file\n", argv[1]);
            fclose(file);
            return 1;
        }

        /* install signal handler and begin capturing */
        play_sample(file, device, header.num_channels, header.sample_rate,
                    header.bits_per_sample);
        fclose(file);

        return 0;
}

void play_sample(FILE *file, unsigned int device, unsigned int channels,
                 unsigned int rate, unsigned int bits)
{
        struct pcm_config config;
        struct pcm *pcm0;
        char *buffer;
        int size;
        int num_read;
        int dtime = 15;
        int loop_time = 0;
        int i=0;

        config.channels = channels;
        config.rate = rate;
        config.period_size = 1024;
        config.period_count = 4;
        if (bits == 32)
```

CONFIDENTIAL

Copyright © 2011-2013 Allwinner Technology. All Rights Reserved.

```
        config.format = PCM_FORMAT_S32_LE;
    else if (bits == 16)
        config.format = PCM_FORMAT_S16_LE;
    config.start_threshold = 0;
    config.stop_threshold = 0;
    config.silence_threshold = 0;

    /*0 is audiocodec, 1 is hdmiaudio, 2 is spdif, 3 is i2s，4 is pcm*/
    pcm0 = pcm_open(3, device, PCM_OUT, &config);
    if (!pcm0 || !pcm_is_ready(pcm0)) {
        fprintf(stderr, "Unable to open PCM device %u (%s)\n", device, pcm_get_error(pcm0));
        return;
    }

    size = pcm_get_buffer_size(pcm0);
    buffer = malloc(size);
    if (!buffer) {
        fprintf(stderr, "Unable to allocate %d bytes\n", size);
        free(buffer);
        pcm_close(pcm0);
        return;
    }

    printf("Playing sample: %u ch, %u hz, %u bit\n", channels, rate, bits);
    loop_time = dtime*rate/1024;
    printf("loop_time:%d\n", loop_time);
    do {
        num_read = fread(buffer, 1, size, file);
        if (num_read > 0) {
            if (pcm_write(pcm0, buffer, num_read)) {
                fprintf(stderr, "Error playing sample\n");
                    break;
            }
            if (feof(file)) {
                fseek(file, 0L, SEEK_SET);
                //break;
            }
        }
        i++;
        if(i > loop_time)
            break;
    } while ((num_read > 0));

    free(buffer);
```

CONFIDENTIAL

```
	pcm_close(pcm0);
}
```

5.2. 最小的 capture 应用

```
/* tinycap.c
**
** Copyright 2011, The Android Open Source Project
**
** Redistribution and use in source and binary forms, with or without
** modification, are permitted provided that the following conditions are met:
**     * Redistributions of source code must retain the above copyright
**       notice, this list of conditions and the following disclaimer.
**     * Redistributions in binary form must reproduce the above copyright
**       notice, this list of conditions and the following disclaimer in the
**       documentation and/or other materials provided with the distribution.
**     * Neither the name of The Android Open Source Project nor the names of
**       its contributors may be used to endorse or promote products derived
**       from this software without specific prior written permission.
**
*/

#include <include/tinyalsa/asoundlib.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <signal.h>

#define ID_RIFF 0x46464952
#define ID_WAVE 0x45564157
#define ID_FMT   0x20746d66
#define ID_DATA 0x61746164

#define FORMAT_PCM 1

struct wav_header {
	uint32_t riff_id;
	uint32_t riff_sz;
	uint32_t riff_fmt;
	uint32_t fmt_id;
	uint32_t fmt_sz;
	uint16_t audio_format;
```

CONFIDENTIAL

Copyright © 2011-2013 Allwinner Technology. All Rights Reserved.

```
        uint16_t num_channels;
        uint32_t sample_rate;
        uint32_t byte_rate;
        uint16_t block_align;
        uint16_t bits_per_sample;
        uint32_t data_id;
        uint32_t data_sz;
};

int capturing = 1;
static char filename[64] = "record1.wav";

unsigned int capture_sample(FILE *file, unsigned int device,
                            unsigned int channels, unsigned int rate,
                            unsigned int bits);

void sigint_handler(int sig)
{
        capturing = 0;
}

int main(int argc, char **argv)
{
        FILE *file;
                struct mixer *mixer;
        struct wav_header header;
        unsigned int device = 0;
        unsigned int channels = 2;
        unsigned int rate = 44100;
        unsigned int bits = 16;
        unsigned int frames;

        rate = atoi(argv[1]);
        channels = atoi(argv[2]);

        if (argc < 2) {
                fprintf(stderr, "Usage: %s file.wav [-d device] [-c channels] "
                                "[-r rate] [-b bits]\n", argv[0]);
                return 1;
        }

        file = fopen(filename, "wb");
        if (!file) {
                fprintf(stderr, "Unable to create file '%s'\n", argv[1]);
```

CONFIDENTIAL

Copyright © 2011-2013 Allwinner Technology. All Rights Reserved.

```
        return 1;
    }

    /* parse command line arguments */
    argv += 2;
    while (*argv) {
        if (strcmp(*argv, "-d") == 0) {
            argv++;
            device = atoi(*argv);
        } else if (strcmp(*argv, "-c") == 0) {
            argv++;
            channels = atoi(*argv);
        } else if (strcmp(*argv, "-r") == 0) {
            argv++;
            rate = atoi(*argv);
        } else if (strcmp(*argv, "-b") == 0) {
            argv++;
            bits = atoi(*argv);
        }
        argv++;
    }

    header.riff_id = ID_RIFF;
    header.riff_sz = 0;
    header.riff_fmt = ID_WAVE;
    header.fmt_id = ID_FMT;
    header.fmt_sz = 16;
    header.audio_format = FORMAT_PCM;
    header.num_channels = channels;
    header.sample_rate = rate;
    header.byte_rate = (header.bits_per_sample / 8) * channels * rate;
    header.block_align = channels * (header.bits_per_sample / 8);
    header.bits_per_sample = bits;
    header.data_id = ID_DATA;

    /* leave enough room for header */
    fseek(file, sizeof(struct wav_header), SEEK_SET);

    mixer = mixer_open(0);
    if (!mixer) {
        fprintf(stderr, "Failed to open mixer\n");
        return EXIT_FAILURE;
    }
    /* install signal handler and begin capturing */
```

CONFIDENTIAL

Copyright © 2011-2013 Allwinner Technology. All Rights Reserved.

```
    signal(SIGINT, sigint_handler);
    frames = capture_sample(file, device, header.num_channels,
                            header.sample_rate, header.bits_per_sample);
    printf("Captured %d frames\n", frames);

    /* write header now all information is known */
    header.data_sz = frames * header.block_align;
    fseek(file, 0, SEEK_SET);
    fwrite(&header, sizeof(struct wav_header), 1, file);
    mixer_close(mixer);
    fclose(file);

    return 0;
}

unsigned int capture_sample(FILE *file, unsigned int device,
                            unsigned int channels, unsigned int rate,
                            unsigned int bits)
{
    struct pcm_config config;
    struct pcm *pcm;
    char *buffer;
    unsigned int size;
    unsigned int bytes_read = 0;
    int dtime = 1;
    int loop = 0;

    config.channels = channels;
    config.rate = rate;
    config.period_size = 1024;
    config.period_count = 4;
    if (bits == 32)
        config.format = PCM_FORMAT_S32_LE;
    else if (bits == 16)
        config.format = PCM_FORMAT_S16_LE;
    config.start_threshold = 0;
    config.stop_threshold = 0;
    config.silence_threshold = 0;

    pcm = pcm_open(0, device, PCM_IN, &config);
    if (!pcm || !pcm_is_ready(pcm)) {
        fprintf(stderr, "Unable to open PCM device (%s)\n",
                pcm_get_error(pcm));
        return 0;
```

CONFIDENTIAL

Copyright © 2011-2013 Allwinner Technology. All Rights Reserved.

```
    }

    size = pcm_get_buffer_size(pcm);
    buffer = malloc(size);
    if (!buffer) {
        fprintf(stderr, "Unable to allocate %d bytes\n", size);
        free(buffer);
        pcm_close(pcm);
        return 0;
    }

    printf("Capturing sample: %u ch, %u hz, %u bit\n", channels, rate, bits);


    dtime = 12*60*60;
    loop = dtime*rate/1024;
    printf("loop:%d\n", loop);
    for(loop;loop > 0;loop--)
    {
        int ret = 0;
        //printf("hello,size:%d\n",size);
        ret = pcm_read(pcm, buffer, size);
        if (fwrite(buffer, 1, size, file) != size) {
            fprintf(stderr,"Error capturing sample\n");
            break;
        }
        bytes_read += size;
    }

    free(buffer);
    pcm_close(pcm);
    return bytes_read / ((bits / 8) * channels);
}
```

5.3. Mixer 接口的使用

参考 audiocodec 接口描述中的 4.1.1 系统音频播放接口部分。

CONFIDENTIAL

6. 音频常见问题调试

6.1. 耳机插拔检测失败

耳机插拔检测的时候，声音会通过喇叭，耳机进行切换，如果是电话模式，还有根听筒的切换。如果声音没有正常切换，首先检查耳机是否有检测到。将 debug 信息打开。

\lichee\linux-3.3\drivers\staging\android\switch

```
#if (1)
	#define SWITCH_DBG(format,args...)    printk("[SWITCH] "format,##args)
#else
	#define SWITCH_DBG(...)
#endif
```

检查插拔耳机的时候是否有中断产生，在 audio_hmic_irq 已经加入相关打印信息。

前提是生成一个 debug 固件。

1.将\lichee\tools\pack\chips\sun6i\configs\android\default\env.cfg

改成 loglevel=4 改成 loglevel=8

2.将\android4.1\device\softwinner\fiber-common\BoardConfigCommon.mk 中的

BOARD_KERNEL_CMDLINE := console=ttyS0,115200 rw init=/init loglevel=4

改成 BOARD_KERNEL_CMDLINE := console=ttyS0,115200 rw init=/init loglevel=8

重新打包生成一个 debug 固件即可看到内核的打印信息。

6.2. 喇叭耳机音量大小

参考 2.4.2.1 中的 sysconfig 配置，调整喇叭的音量即可。

6.3. pcm 蓝牙音频没有声音输出或者噪音

如果不确定是上行还是下行 BT 传输出现问题，在系统层将音频数据抓出来，然后利用 pc 工具分析即可。请参考

~\android4.1\device\softwinner\fiber-common\hardware\audio\audio_digital.c

将 SUNXI_DIGITAL_TEST 打开即可。

如果是额外编写的音频传输，抓起音频数据的程序可以参考如下：

```
#ifdef   PCM_TEST
	{
		static int flag = 0;
		static int num = 0;
		static FILE *fptest = 0;
		if((flag==0)&&(num==0))
		{
			if(fptest)
			{
				fclose(fptest);
				fptest = 0;
```

CONFIDENTIAL

```
        }
        fptest = fopen("/data/camera/trackdown.pcm","wb");
        if(!fptest)
        {
            ALOGE("ERROR opening V4L interface: %s", strerror(errno));
        }
        ALOGD("####FLAG,trackweb,FP=0X%0x,",fptest);
    }
    if((flag==1)&&(num==0))
    {
        if(fptest)
        {
            fclose(fptest);
            fptest = 0;
        }
        fptest = fopen("/data/camera/trackdown1.pcm","wb");
        ALOGD("####FLAG,trackweb1,FP=0X%0x,",fptest);

    }
    fwrite(codec_in_buffer,1,size3,fptest);
    num ++;
    //LOGD("####FLAG,pcmweb,FP=0X%0x,num=%d,cout=%d",fptest,num,userSize);
    if(num>2000)
    {
        if(flag==0)
            flag =1;
        else
          flag = 0;
        num =0;
    }
}
```

然后利用命令 adb pull /data/camera/trackdown.bin e:/将文件拷贝到 pc 端。可以利用 Adobe Audition 3.0 软件，查看音频数据是否正确，利用耳机直接听即可分辨（注意采样率和通道数要跟录制的一致）。

CONFIDENTIAL