嵌入式 Linux 音频系统驱动指南

Lutts Wolf
lutts.cao@gmail.com

2011 年 8 月 12 日

摘要

本文档分为两部分。

第一部分讲解了 linux-2.6.31 内核的 ALSA 驱动的架构，特别是和嵌入式系统相关的内容，即 ASoC。硬件部分主要针对君正 4770 及其内部 codec，另外考虑到中高端客户会选用 wm8753 等外部 codec，添加了对 wm8731 及 wm8753 这两款 codec 的驱动分析。

第二部分讲解了 linux-2.6.31 内核的 OSS 驱动的架构，基于君正 4770 及内部 codec，没有提供外部 codec 的说明。

建议客户将产品的音频部分迁移到 ALSA，特别是使用外部 codec 的客户。

关键字: Audio ALSA OSS JZ4770 wm8731 wm8753

献给想了解 Linux 音频系统的人

天才在于积累，聪明在于勤奋。勤能补拙是良训，一分辛苦一分才。

——— 华罗庚

面对悬崖峭壁，一百年也看不出一条缝来，但用斧凿，能进一寸进一寸，得进一尺进一尺，不断积累，飞跃必来，突破随之。

——— 华罗庚

目录

表格

第一部分　ALSA– Advanced Linux Sound Architecture

以下是 ALSA 官网对 ALSA 的介绍：

The Advanced Linux Sound Architecture (ALSA) provides audio and MIDI functionality to the Linux operating system. ALSA has the following significant features:

- Efficient support for all types of audio interfaces, from consumer sound cards to professional multichannel audio interfaces.
- Fully modularized sound drivers.
- SMP and thread-safe design.
- User space library (alsa-lib) to simplify application programming and provide higher level functionality.
- Support for the older Open Sound System (OSS) API, providing binary compatibility for most OSS programs.

通过本文的讲解，我希望能稍微揭开 ALSA 神秘的面纱。

首先是初识 ALSA，通过这一章，我将带你熟悉 ALSA 的使用，编写 ALSA 驱动的一大难题就是不知道内核和用户态是怎样联系起来的。在这一章中，我会从整体上介绍 ALSA 所涉及到的方方面面 —哪些文件、哪些配置、哪些工具。

然后采用逐步展开的方式，由外入里地讲解 ALSA 音频驱动的实现细节，先是音频驱动的整体框架，然后是 codec 驱动、I2S 驱动，最后是 PCM 驱动 (DMA)。

1 初识 ALSA

1.1 ALSA 架构

下面是 ALSA 音频系统的架构：

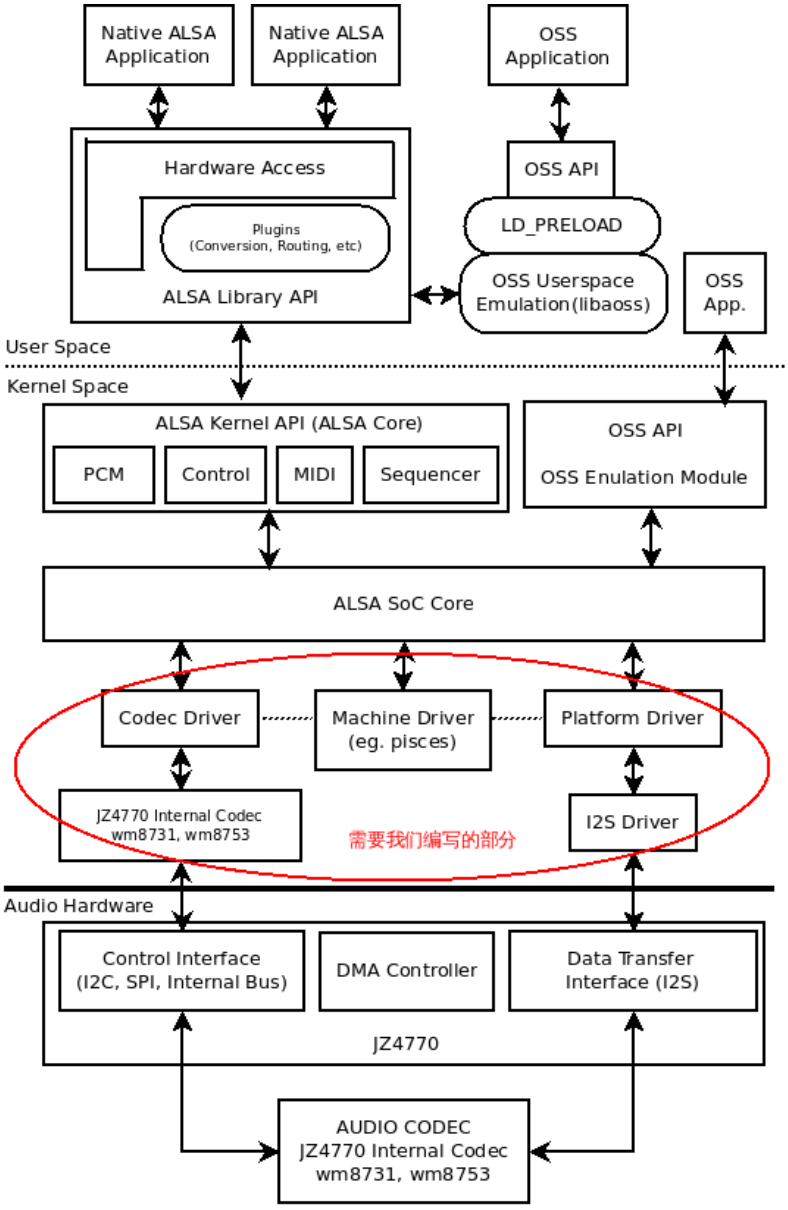

图 1: Basic Structure and Flow of ALSA System

本文接下来的内容我们将对图中提到的各个部件进行详细的讲解。

1.2 ALSA in Kernel

包括硬件相关及一些公共代码，位于 sound/core 和 sound/soc 目录下，其中 sound/core 目录是 ALSA 的核心代码，我们暂不分析，sound/soc 的主要内容如下

- **soc-core.c 和 soc-dapm.c:** 这是 ASoC 的核心代码
- **codec/:** 各种 codec 驱动，对于 JZ4770 的内部 codec, 对应源码文件为 jz4770_icdc.c 及 jz4770_icdc.h，具体实现会在后面的章节讲解。这里说明一下 Kconfig 及 Makefile. 一般情况下，添加新的 codec 支持很简单，以 jz4770 内部 codec 为例，只需在 Kconfig 中添加以下内容即可：

```
config SND_SOC_JZ4770_ICDC
        tristate
        depends on SND_SOC
```

  然后在 Makefile 中，分别在适当的位置加入***snd-soc-jz4770-icdc-objs := jz4770_icdc.o*** 和 ***obj-$(CONFIG_SND_SOC_JZ4770_ICDC) += snd-soc-jz4770-icdc.o***即可。

- **除 codec 之外的目录:** 一般存放各个芯片的 Machine Driver 和 Platform Driver，这点从他们的名字大致可以看出来，例如 jz47xx 这个目录，jz47xx-i2s.c 是 I2S 控制器驱动, jz47xx-pcm.c 是 DMA 数据传输的实现, pisces_icdc.c 是 Machine Driver，对应我们的 Pisces 开发板。

  再比如 s3c24xx 这个目录，对应有 s3c24xx 的 I2S 控制器驱动 s3c24xx-i2s.c，s3c24xx-pcm.c 是 DMA 数据传输的实现，neo1973_wm8753.c 是 openmoko 项目的 NEO 手机的 Machine Driver。

  对于 Machine Driver 的文件名，一般是***<board name>_<codec_name>.c***的形式，I2S 控制器没有在这个名字里体现出来，如果某个 codec 支持多种音频总线，你可能需要在文件名上体现出来。

- **Machine Driver 和 Platform Driver 的 Kconfig 及 Makefile 写法:** 下面是 JZ47xx 的 Kconfig:

```
config SND_JZ47XX_SOC
        tristate "SoC Audio for Ingenic jz47XX chip"
        depends on JZSOC && SND_SOC
        help
          Say Y or M if you want to add support for codecs attached to
          the Jz47XX AC97, I2S controller.

config SND_JZ47XX_SOC_I2S
        tristate

config SND_JZ4760_SOC_LEPUS_ICDC
        tristate "SoC I2S Audio support for JZ4760 Lepus reference board with internal \
            codec"
        depends on SND_JZ47XX_SOC
        depends on (SOC_JZ4760 || SOC_JZ4760B)
        select SND_JZ47XX_SOC_I2S
        select SND_SOC_JZ4760_ICDC
        help
          Say Y if you want to add audio support for JZ4760 Lepus reference board with \
              internal codec.

config SND_JZ4770_SOC_PISCES_ICDC
        tristate "SoC I2S Audio support for JZ4770 Pisces reference board with internal \
            codec"
        depends on SND_JZ47XX_SOC
        depends on SOC_JZ4770
        select SND_JZ47XX_SOC_I2S
        select SND_SOC_JZ4770_ICDC
        help
          Say Y if you want to add audio support for JZ4770 Pisces reference board with \
              internal codec.
```

很简单，不多做解释。

Makefile 也很简单：

```
#
# Jz47xx Platform Support
#
snd-soc-jz47xx-objs := jz47xx-pcm.o
snd-soc-jz47xx-i2s-objs := jz47xx-i2s.o

obj-$(CONFIG_SND_JZ47XX_SOC) += snd-soc-jz47xx.o
obj-$(CONFIG_SND_JZ47XX_SOC_I2S) += snd-soc-jz47xx-i2s.o

# Jz4760 Machine Support
snd-soc-lepus-icdc-objs := lepus_icdc.o

```

```
13 obj-$(CONFIG_SND_JZ4760_SOC_LEPUS_ICDC) += snd-soc-lepus-icdc.o
14
15 # Jz4770 Machine Support
16 snd-soc-pisces-icdc-objs := pisces_icdc.o
17
18 obj-$(CONFIG_SND_JZ4770_SOC_PISCES_ICDC) += snd-soc-pisces-icdc.o
```

然后在 sound/soc/Kconfig 中添加如下内容：

```
source "sound/soc/jz47xx/Kconfig"
```

再在 sound/soc/Makefile 中添加如下内容:

```
obj-$(CONFIG_SND_SOC)    += jz47xx/
```

进行完上述这些步骤后，我们需要在 make menuconfig(或 make xconfig) 中选择:

```
Device Drivers ---->
   <*> Sound card support ---->
       <*>  Advanced Linux Sound Architecture ---->
           <*>  ALSA for SoC audio support ---->
               <*>  SoC Audio for Ingenic JZ47XX chip
               <*>  SoC I2S Audio support for JZ4770 Pisces reference board with \
                   internal codec
```

如果要支持 OSS Emulation，请选择以下配置：

```
Device Drivers ---->
   <*> Sound card support ---->
       <*>  Advanced Linux Sound Architecture ---->
                    <*>  OSS Mixer API
                    <*>  OSS PCM (digital audio) API
                    [*]    OSS PCM (digital audio) API - Include plugin system
```

注意：请确保 OSS，即 Device Drivers - - -> Sound - - -> Open Sound System 没有被选中

1.3 ALSA Userspace Library

下面是 ALSA 用户态库 (1.0.20) 的编译方法：

1. 首先从 ALSA 的官方网站下载 alsa-lib-1.0.20.tar.bz2

```
wget ftp://ftp.alsa-project.org/pub/lib/alsa-lib-1.0.20.tar.bz2
```

2. 依次执行以下命令：

```
tar xjf alsa-lib-1.0.20.tar.bz2
cd alsa-lib-1.0.20
./configure --host=mipsel-linux \
            --prefix=/home/slcao/alsa/alsa-1.0.20/target \
            --enable-static --enable-shared --with-alsa-devdir=/dev \
            --with-configdir=/home/slcao/alsa/local/share \
            --with-plugindir=/home/slcao/alsa/local/lib/alsa_lib
make
make install
```

其中：

--prefix: 指定安装路径，编译生成的 libasound 会在 $prefix/lib 目录下

--with-alsa-devdir: ALSA 默认会去/dev/snd/下查找音频设备文件，在这里我们强制指定为/dev

--with-configdir: 指定 alsa.conf 等配置文件所在的目录，这里要注意的一点是，在后面的 make install 过程中，安装程序会将 alsa.conf 等文件拷贝到当前机器 (即：你执行上面这些命令所在的机器)，所以，请确保你有这个目录的权限，最好不要和你机器的目录重名，也就是说，/usr/local/share 也许并不是一个很好的选择，因为你本机的 ALSA 配置文件也可能在这个位置。

--with-plugindir: 这个和 plugin 相关，如 smixer 等，暂不讨论

进行完上述步骤后，目标板的 ALSA 只是被“安装”到了本地机器，需要拷贝到目录板的相关目录，因为就一个 lib 干不了什么，因此等介绍完 alsa utils 的编译步骤后，我再详细说明拷贝的过程。

1.4 ALSA 初体验

1.4.1 ALSA 设备接口

在/dev/下，一般包含以下几种类型的设备:

- **PCM 设备:** 用于录/放数字音频;
- **Control 设备:** 用于控制声卡内部 mixer 以及 routing 等;
- **MIDI 设备:** 控制声卡的 MIDI 端口 (如果有的话)

- **timer**：

设备名	描述
/dev/controlC0	用于控制声卡内部 mixer 以及 routing 等
/dev/pcmC0D0c	PCM Card 0 Device 0 Capture device
/dev/pcmC0D0p	PCM Card 0 Device 0 Playback device
/dev/timer	ALSA timer interface

表 1: ALSA 设备接口

1.4.2 alsa-utils-1.0.20.tar.bz2 编译

1. 下载 alsa-utils-1.0.20.tar.bz2

```
wget ftp://ftp.alsa-project.org/pub/utils/alsa-utils-1.0.20.tar.bz2
```

2. 依次执行以下命令：

```
tar xjf alsa-lib-1.0.20.tar.bz2
cd alsa-lib-1.0.20
./configure --host=mipsel-linux \
            --prefix=/home/slcao/alsa/alsa-1.0.20/target \
            --enable-shared \
            CFLAGS="-I/home/slcao/alsa/alsa-1.0.20/target/include" \
          LDFLAGS="-L/home/slcao/alsa/alsa-1.0.20/target/lib -lasound" \
            --disable-alsamixer --disable-xmlto \
            --with-alsa-inc-prefix=/home/slcao/alsa/alsa-1.0.20/target/\
                include \
            --with-configdir=/home/slcao/alsa/local/share \
            --with-plugindir=/home/slcao/alsa/local/lib/alsa_lib
make
make install
```

其中：

- **-prefix:** 指定安装路径，编译生成的 aplay,amixer 会在 $prefix/bin 目录下,alsactl 在 $prefix/sbin 下

- **-disable-alsamixer:** 不编译 alsamixer，alsamixer 是一个终端下基于文本的图形化音量控制界面，我们不需要这个

- **-disable-xmlto:** 去掉对 xmlto 的依赖

其他选项请参见ALSA Lib的编译说明。

1.4.3 amixer 使用

amixer 是一个命令行工具，用于配置音频的各个参数，例如设置需要的音量、开关某个 switch（开关）等等。

因此，对于 amixer 的使用，你首先需要搞懂你要设置的参数是哪些，然后才可能去了解，如何去配置对应的值，下面简要介绍其具体用法：

1. 先看看 amixer 支持哪些命令, 大概有哪些功能

```
# amixer —-help
Usage: amixer <options> [command]

Available options:
  -h,--help       this help
  -c,--card N     select the card
  -D,--device N   select the device, default 'default'
  -d,--debug      debug mode
  -n,--nocheck    do not perform range checking
  -v,--version    print version of this program
  -q,--quiet      be quiet
  -i,--inactive   show also inactive controls
  -a,--abstract L select abstraction level (none or basic)
  -s,--stdin      Read and execute commands from stdin sequentially

Available commands:
  scontrols       show all mixer simple controls
  scontents       show contents of all mixer simple controls (default command)
  sset sID P      set contents for one mixer simple control
  sget sID        get contents for one mixer simple control
  controls        show all controls for given card
  contents        show contents of all controls for given card
  cset cID P      set control contents for one control
  cget cID        get control contents for one control
```

呵呵，help 已经很详细了。至于什么是 *simple controls*，后文再解释，我们暂时用不到这个

2. 再看看当前你的音频系统（不同的音频驱动对应不同的内容和操作接口）提供了那些供你使用的接口去操作

   关于驱动里面已经提供了多少接口可以去操作，可以用命令 amixer contents 查看，例如：

```
# amixer controls
numid=2,iface=MIXER,name='Master Playback Volume'
numid=1,iface=MIXER,name='PCM Volume'
numid=6,iface=MIXER,name='Linein Bypass Capture Volume'
numid=4,iface=MIXER,name='Mic1 Capture Volume'
numid=5,iface=MIXER,name='Mic2 Capture Volume'
numid=3,iface=MIXER,name='ADC Capture Volume'
numid=8,iface=MIXER,name='Capture Left Mux'
numid=7,iface=MIXER,name='Capture Right Mux'
numid=12,iface=MIXER,name='Playback HP Left Mux'
numid=11,iface=MIXER,name='Playback HP Right Mux'
numid=10,iface=MIXER,name='Playback Lineout Left Mux'
numid=9,iface=MIXER,name='Playback Lineout Right Mux'
numid=13,iface=MIXER,name='Pisces Mode'
```

而对于所有的可能的配置的值，可以通过这个查看：

```
# amixer contents
numid=2,iface=MIXER,name='Master Playback Volume'
  ; type=INTEGER,access=rw----R--,values=2,min=0,max=31,step=0
  : values=16,16
  | dBscale-min=-25.00dB,step=1.00dB,mute=0
numid=1,iface=MIXER,name='PCM Volume'
  ; type=INTEGER,access=rw----R--,values=2,min=0,max=31,step=0
  : values=20,20
  | dBscale-min=-31.00dB,step=1.00dB,mute=0
numid=6,iface=MIXER,name='Linein Bypass Capture Volume'
  ; type=INTEGER,access=rw----R--,values=2,min=0,max=31,step=0
  : values=25,25
  | dBscale-min=-25.00dB,step=1.00dB,mute=0
numid=4,iface=MIXER,name='Mic1 Capture Volume'
  ; type=INTEGER,access=rw----R--,values=1,min=0,max=5,step=0
  : values=0
  | dBscale-min=0.00dB,step=4.00dB,mute=0
numid=5,iface=MIXER,name='Mic2 Capture Volume'
  ; type=INTEGER,access=rw----R--,values=1,min=0,max=5,step=0
  : values=0
  | dBscale-min=0.00dB,step=4.00dB,mute=0
numid=3,iface=MIXER,name='ADC Capture Volume'
  ; type=INTEGER,access=rw----R--,values=2,min=0,max=23,step=0
  : values=0,0
  | dBscale-min=0.00dB,step=1.00dB,mute=0
numid=8,iface=MIXER,name='Capture Left Mux'
  ; type=ENUMERATED,access=rw-------,values=1,items=3
  ; Item #0 'Mic 1'
  ; Item #1 'Mic 2'
  ; Item #2 'Line In'
  : values=0
numid=7,iface=MIXER,name='Capture Right Mux'
```

```
  ; type=ENUMERATED,access=rw------,values=1,items=3
  ; Item #0 'Mic 1'
  ; Item #1 'Mic 2'
  ; Item #2 'Line In'
  : values=0
numid=12,iface=MIXER,name='Playback HP Left Mux'
  ; type=ENUMERATED,access=rw------,values=1,items=4
  ; Item #0 'Mic 1b'
  ; Item #1 'Mic 2b'
  ; Item #2 'Line Inb'
  ; Item #3 'Stereo DAC'
  : values=3
numid=11,iface=MIXER,name='Playback HP Right Mux'
  ; type=ENUMERATED,access=rw------,values=1,items=4
  ; Item #0 'Mic 1b'
  ; Item #1 'Mic 2b'
  ; Item #2 'Line Inb'
  ; Item #3 'Stereo DAC'
  : values=3
numid=10,iface=MIXER,name='Playback Lineout Left Mux'
  ; type=ENUMERATED,access=rw------,values=1,items=4
  ; Item #0 'Mic 1_lo'
  ; Item #1 'Mic 2_lo'
  ; Item #2 'Line In_lo'
  ; Item #3 'Stereo DAC'
  : values=3
numid=9,iface=MIXER,name='Playback Lineout Right Mux'
  ; type=ENUMERATED,access=rw------,values=1,items=4
  ; Item #0 'Mic 1_lo'
  ; Item #1 'Mic 2_lo'
  ; Item #2 'Line In_lo'
  ; Item #3 'Stereo DAC'
  : values=3
numid=13,iface=MIXER,name='Pisces Mode'
  ; type=ENUMERATED,access=rw------,values=1,items=16
  ; Item #0 'Off'
  ; Item #1 'Headphone'
  ; Item #2 'Lineout'
  ; Item #3 'HP+Lineout'
  ; Item #4 'Mic1'
  ; Item #5 'Mic2'
  ; Item #6 'Mic1+Mic2'
  ; Item #7 'HP+Mic1'
  ; Item #8 'HP+Mic2'
  ; Item #9 'HP+MIC1+Mic2'
  ; Item #10 'Lineout+Mic1'
  ; Item #11 'Lineout+Mic2'
  ; Item #12 'Lineout+Mic1+Mic2'
  ; Item #13 'HP+LO+Mic1'
  ; Item #14 'HP+LO+Mic2'
```

```
; Item #15 'HP+LO+Mic1+Mic2'
: values=1
```

从上面的打印信息可以看出，对于 JZ4770 的 Pisces 开发板，我们支持以下设置：

- DAC 音量设置 (PCM Volume)
- 耳机音量设置 (Master Playback Volume)
- Mic1 音量设置 (Mic1 Capture Volume，即 Mic1 Boost)
- Mic2 音量设置 (Mic2 Capture Volume, 即 Mic2 Boost)
- Linein bypass 音量设置 (Linein Bypass Capture Volume)
- ADC 音量设置 (ADC Capture Volume)
- 录音源选择 (Capture Left Mux 和 Capture Right Mux)
- 耳机放音源选择 (Playback HP Left Mux 和 Playback HP Right Mux)
- Lineout 放音源选择 (Playback Lineout Left Mux 和 Playback Lineout Right Mux)
- 功能接口启用/禁用，在 Pisces 开发板上，我们支持以下功能接口组合:
  (a) **Headphone:** 通过耳机放音
  (b) **Lineout:** 通过 Lineout 接口放音
  (c) **HP+Lineout:** 耳机和 Lineout 同时放音，此时耳机和 Lineout 的放音源必须相同
  (d) **Mic1:** 通过 Mic1 录音
  (e) **Mic2:** 通过 Mic2 录音
  (f) **Mic1+Mic2:** Mic1 和 Mic2 同时录音
  (g) **HP+Mic1:** 耳机和 Mic1 功能启用，可用于一边放音，一边录音，也可用于通过 Mic1 录音同时 bypass 到耳机
  (h) **HP+Mic2:** 同上，不同在于通过 Mic2 录音
  (i) **HP+MIC1+Mic2:** 同上，不同在于同时从 Mic1 和 Mic2 录音
  (j) **Lineout+Mic1:** lineout 和 Mic1 功能启用，可用于一般通过 lineout 放音，一边录音，也可用于通过 Mic1 录音的同时 bypass 到 lineout
  (k) **Lineout+Mic2:** 同上，录音源是 Mic2
  (l) **Lineout+Mic1+Mic2:** 同上，录音源是 Mic1 和 Mic2

(m) **HP+LO+Mic1:** 同上，同时 bypass 到耳机和 lineout

(n) **HP+LO+Mic2:** 同上，录音源是 Mic2

(o) **HP+LO+Mic1+Mic2:** 同上，录音源是 Mic1 和 Mic2

以上只是启用/禁用相关的功能接口，具体如何工作还需配合录音源的选择以及放音源的选择

3. 搞懂如何去设置某个参数

总结起来就是，先要用 get 系列命令去看懂有哪些接口，然后再去用 set 系列的命令，去设置对应你所要设置的值。

比如想要设置上面的的 DAC 音量，即 controls 中显示的：

numid=1,iface=MIXER,name='PCM Volume'

那么，可以先看看当前的值：

```
#  amixer cget numid=1,iface=MIXER,name='PCM Volume'
numid=1,iface=MIXER,name='PCM Volume'
  ; type=INTEGER,access=rw——R--,values=2,min=0,max=31,step=0
  : values=31,31
  | dBscale-min=-31.00dB,step=1.00dB,mute=0
```

显示的是最大的 31(即 max=31)，假设想要设置为 25，那么就用 cset 去设置：

```
# amixer cset numid=1,iface=MIXER,name='PCM Volume' 25
numid=1,iface=MIXER,name='PCM Volume'
  ; type=INTEGER,access=rw——R--,values=2,min=0,max=31,step=0
  : values=25,25
  | dBscale-min=-31.00dB,step=1.00dB,mute=0
```

再来一个例子，依然是先查看，后设置:

```
# amixer cget numid=10,iface=MIXER,name='Left Out Mux'
numid=10,iface=MIXER,name='Left Out Mux'
  ; type=ENUMERATED,access=rw------,values=1,items=4
  ; Item #0 'Mic 1b'
  ; Item #1 'Mic 2b'
  ; Item #2 'Line Inb'
  ; Item #3 'Stereo DAC'
  : values=0
```

```
# amixer cset numid=10,iface=MIXER,name='Left Out Mux' 3
numid=10,iface=MIXER,name='Left Out Mux'
  ; type=ENUMERATED,access=rw------,values=1,items=4
  ; Item #0 'Mic 1b'
  ; Item #1 'Mic 2b'
  ; Item #2 'Line Inb'
  ; Item #3 'Stereo DAC'
  : values=3
```

总结一下用法，就是：

```
amixer cget    controls 中所输出的某个参数
amixer cset    controls 中所输出的某个参数       具体的值 (比如,0,1,On,Off 等)
```

1.4.4 alsa 配置文件

ALSA 有两类配置文件，一类是 controls 状态文件，一类录放音及插件配置相关

controls 状态文件:

每次都需要使用 amixer 去设置的话，效率会很低，ALSA 提供了保存/恢复状态文件的功能。

使用 amixer 配置好需要的值之后，如果以后想一直用这些值，那么可以使用以下命令将配置好的值保存到一个 state 文件中

```
alsactl -f test.state store
```

以上命令将当前的所有配置保存到 test.state 文件中。在需要的时候，可以使用以下命令恢复配置

```
alsactl -f test.state restore
```

下面是在 Pisces 开发板上通过 Headphone 放音的配置文件，以供参考

```
state.Pisces {
        control.1 {
                comment.access 'read write'
                comment.type INTEGER
                comment.count 2
                comment.range '0 - 31'
                comment.dbmin -3100
                comment.dbmax 0
                iface MIXER
                name 'PCM Volume'
                value.0 20
                value.1 20
        }
        control.2 {
```

```
        comment.access 'read write'
        comment.type INTEGER
        comment.count 2
        comment.range '0 - 31'
        comment.dbmin -2500
        comment.dbmax 600
        iface MIXER
        name 'Master Playback Volume'
        value.0 16
        value.1 16
}
control.3 {
        comment.access 'read write'
        comment.type INTEGER
        comment.count 2
        comment.range '0 - 23'
        comment.dbmin 0
        comment.dbmax 2300
        iface MIXER
        name 'ADC Capture Volume'
        value.0 0
        value.1 0
}
control.4 {
        comment.access 'read write'
        comment.type INTEGER
        comment.count 1
        comment.range '0 - 5'
        comment.dbmin 0
        comment.dbmax 2000
        iface MIXER
        name 'Mic1 Capture Volume'
        value 0
}
control.5 {
        comment.access 'read write'
        comment.type INTEGER
        comment.count 1
        comment.range '0 - 5'
        comment.dbmin 0
        comment.dbmax 2000
        iface MIXER
        name 'Mic2 Capture Volume'
        value 0
}
control.6 {
        comment.access 'read write'
        comment.type INTEGER
        comment.count 2
        comment.range '0 - 31'
```

```
        comment.dbmin -2500
        comment.dbmax 600
        iface MIXER
        name 'Linein Bypass Capture Volume'
        value.0 25
        value.1 25
}
control.7 {
        comment.access 'read write'
        comment.type ENUMERATED
        comment.count 1
        comment.item.0 'Mic 1'
        comment.item.1 'Mic 2'
        comment.item.2 'Line In'
        iface MIXER
        name 'Capture Right Mux'
        value 'Mic 1'
}
control.8 {
        comment.access 'read write'
        comment.type ENUMERATED
        comment.count 1
        comment.item.0 'Mic 1'
        comment.item.1 'Mic 2'
        comment.item.2 'Line In'
        iface MIXER
        name 'Capture Left Mux'
        value 'Mic 1'
}
control.9 {
        comment.access 'read write'
        comment.type ENUMERATED
        comment.count 1
        comment.item.0 'Mic 1_lo'
        comment.item.1 'Mic 2_lo'
        comment.item.2 'Line In_lo'
        comment.item.3 'Stereo DAC'
        iface MIXER
        name 'Playback Lineout Right Mux'
        value 'Stereo DAC'
}
control.10 {
        comment.access 'read write'
        comment.type ENUMERATED
        comment.count 1
        comment.item.0 'Mic 1_lo'
        comment.item.1 'Mic 2_lo'
        comment.item.2 'Line In_lo'
        comment.item.3 'Stereo DAC'
        iface MIXER
```

```
        name 'Playback Lineout Left Mux'
        value 'Stereo DAC'
}
control.11 {
        comment.access 'read write'
        comment.type ENUMERATED
        comment.count 1
        comment.item.0 'Mic 1b'
        comment.item.1 'Mic 2b'
        comment.item.2 'Line Inb'
        comment.item.3 'Stereo DAC'
        iface MIXER
        name 'Playback HP Right Mux'
        value 'Stereo DAC'
}
control.12 {
        comment.access 'read write'
        comment.type ENUMERATED
        comment.count 1
        comment.item.0 'Mic 1b'
        comment.item.1 'Mic 2b'
        comment.item.2 'Line Inb'
        comment.item.3 'Stereo DAC'
        iface MIXER
        name 'Playback HP Left Mux'
        value 'Stereo DAC'
}
control.13 {
        comment.access 'read write'
        comment.type ENUMERATED
        comment.count 1
        comment.item.0 Off
        comment.item.1 Headphone
        comment.item.2 Lineout
        comment.item.3 HP+Lineout
        comment.item.4 Mic1
        comment.item.5 Mic2
        comment.item.6 Mic1+Mic2
        comment.item.7 HP+Mic1
        comment.item.8 HP+Mic2
        comment.item.9 HP+MIC1+Mic2
        comment.item.10 Lineout+Mic1
        comment.item.11 Lineout+Mic2
        comment.item.12 Lineout+Mic1+Mic2
        comment.item.13 HP+LO+Mic1
        comment.item.14 HP+LO+Mic2
        comment.item.15 HP+LO+Mic1+Mic2
        iface MIXER
        name 'Pisces Mode'
        value Headphone
```

```
        }
}
```

录放音及插件配置:

ALSA 有很多插件可以使用，提供诸如混音，环绕，resample 等支持。具体请参见 ALSA 网站的 asoundrc 以及 pcm plugin 相关的文档。

下面给出一份参考配置，支持混音，resample，保存为.asoundrc 放在用户主目录下 ($HOME) 即可。在这我不对这个配置文件多做解释，感兴趣的读者请先看懂 asoundrc 及 pcm plugin 相关的文档

```
pcm.!default {
        type hw
        card 0
}

ctrl.!default {
        type hw
        card 0
}

pcm.primary {
        type hw
        card 0
        device 0
}

pcm_slave.sl2 {
        pcm "hw:0,0"
        rate 48000
}

pcm.rate_convert {
        type rate
        slave sl2
}

pcm.asymed {
        type asym
        playback.pcm "dmixer"
        capture.pcm "dsnoop0"
}

pcm.!default{
        type plug
        slave.pcm "asymed"
}
```

```
pcm.dmixer {
        type dmix
        ipc_key 1024
        slave {
                pcm "hw:0,0"
                rate 44100
                channels 2
                format S16_LE
        }
        bindings {
                0 0
                1 1
        }
}

ctl.dmixer {
        type hw
        card 0
}

pcm.dsnoop0 {
    type dsnoop
    ipc_key 1025
    ipc_key_add_uid true
    slave.pcm {
        type hw
        card 0
        device 0
        }
}

pcm.dmixoss {
        type dmix
        ipc_key 1026
        ipc_key_add_uid true
        slave {
                pcm "hw:0,0"
                period_time 0
                period_time 1024 # must be power of 2
                buffer_size 8192 # must be power of 2
        }
        bindings {
                0 0
                1 1
        }
}

ctl.dmixoss {
        type hw
        card 0
```

```
}

pcm.!oss_mix_dev {
        type plug
        slave.pcm "dmixer"
}
```

1.4.5 放音及录音—aplay 和 arecord 的使用

至此，ALSA 的配置已经完成了，我们可以使用 aplay 和 arecord 来测试录放音，他们的使用都很简单

使用 aplay 播放 test.wav 的命令如下

```
aplay test.wav
```

arecord 使用方法如下

```
arecord -r 44100 -c 2 -f S16_LE -t wav -d 30 test.wav
```

上面的命令将录制采样率为 44100，两通道，格式为 signed 16bit 的录音，保存文件格式是 WAV，文件名是 test.wav

通过适当的配置，ALSA 是支持同时录放的。

在君正提供的示例 rootfs 中，在/alsa/目录下有两个目录，分别是 60b 和 70，分别对应 JZ4760B 和 JZ4770 的测试例程.

/alsa/70 目录说明

1.5 ALSA Library API 编程

可以参考 alsa-utils 中的 aplay 和 arecord 的实现

http://www.suse.de/~mana/alsa090_howto.html详细介绍了 ALSA 的编程方法。

1.6 alsa-plugins

一般情况下您不需要这个包，但如果你觉得 alsa-lib 自带的 dmix 不好，想用 libsamplerate[1]的话，你也需要 alsa-plugins 的支持。

[1] http://www.mega-nerd.com/SRC/download.html

1.7 What's Next?

通过本节的介绍，我相信你对 ALSA 应该有了初步的认识，并且已经学会使用了。接下来将详细分析 JZ4770 及其内部 Codec 的 ALSA 实现，期间我也会详细讲解 wm8731 以及 wm8753 codec 驱动实现。

2 驱动总体框架

从图 1可以看出，Machine Driver 是联系其他驱动模块的纽带，我们的 Pisces 对应的文件是 pisces_icdc.c，我们来看看它的具体实现。

3 codec 驱动分析

3.1 概述

3.2 codec 寄存器管理

3.3 kcontrols 分析

3.4 DAPM 分析

3.5 总结

4 I2S 控制器驱动分析

4.1 驱动的基本框架

4.2 具体实现分析

5 PCM – 音频数据传输流程

5.1 驱动的基本框架

5.2 具体实现分析

5.3 pcm_lib 的实现分析

6 深入分析

6.1 ALSA lib 分析

第二部分 OSS – Open Sound System

A Jz4770 Codec Driver Source Code

B Jz4770 I2S Controller Driver Source Code

C Jz4770 PCM(DMA) Driver Source Code

D JZ4770 Lepus Board Driver Source Code