A31 Audio 适配层使用说明

V1.0

2013-3-5

版本历史

版本	时间	备注
V1.0	2013-03-05	建立初始版本

目 录

1. 前言

1.1. 编写目的

本文档目的是为了让开发者了解 A31 电话系统框架的 Android audio 适配层，能够在 A31 平台上开发新的电话方案。

1.2. 适用范围

硬件平台：

A31 平台。

软件平台：

exdroid4.1.1_r1-a31-v1.0 及以上版本。

1.3. 相关人员

A31 Android 开发人员。

2. Android audio hal 层介绍

Android audio hal 层是上层应用和 linux 驱动之间的纽带，功能是：负责响应上层命令，管理 linux 音频硬件驱动。提供 alsa－lib 与 linux 中 alsa-driver 架构实现控制和数据传输。实现。应用程序只要调用 Android audio hal 的 alsa-lib 提供的 API，即可以完成对底层音频硬件的控制。

2.1. Android audio hal 层功能介绍

在 a31 中，存在 5 个音频设备。分别为 audiocodec，hdmiaudio，spdif，pcm，i2s。具体说明请参见《A31 平台音频模块开发说明文档》

2.1.1. Android audio hal 层功能

Android audio hal 层所具有的功能：

- 设置多种采样率格式(8khz, 11.025 KHz, 12 KHz, 16 KHz, 22.05 KHz, 24 KHz, 32 KHz, 44.1 KHz , 48 KHz, 96KHz, 192KHz);
- 设置 mono 和 stereo 模式;
- playback 和 record(全双工模式)数据传送;
- 支持 start, stop, pause 和 resume;
- 支持 mixer 接口
- 支持 resample（重采样）功能
- 支持 3g 通话功能
- 支持录音功能

2.2. 源码结构介绍

Android audio hal 层代码存放在 audio 目录，如图 1 所示。

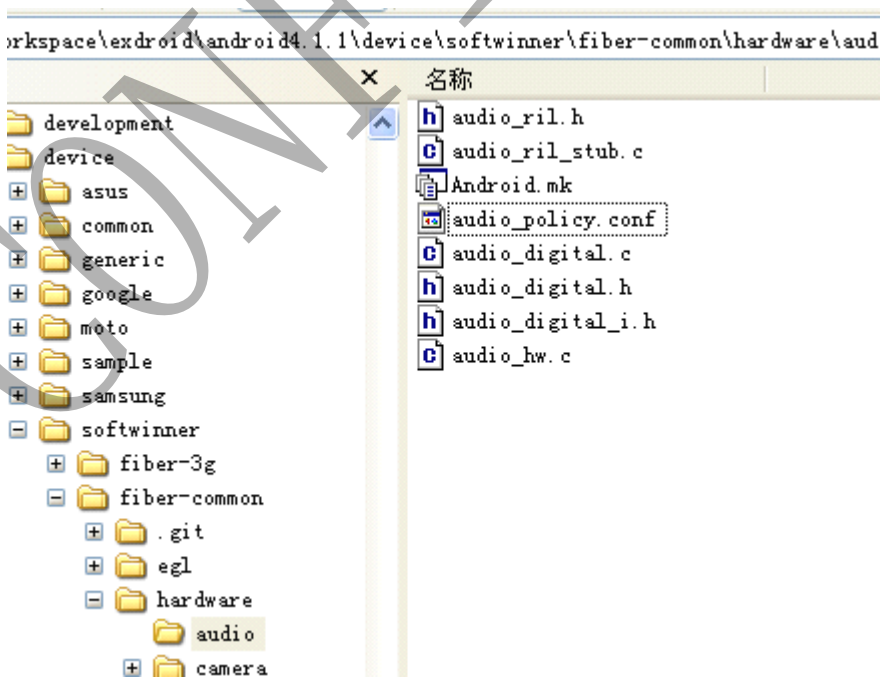

图 1 android audio hal 层源码

2.3. Android audio 框架图如图 2 所示：

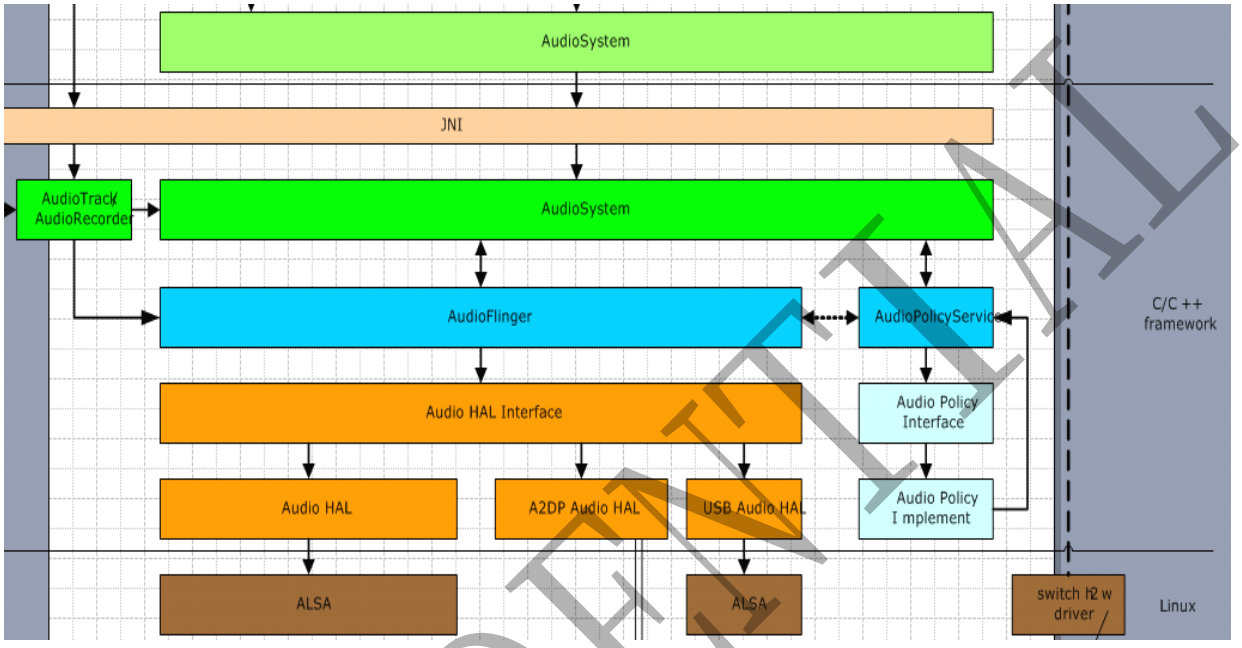

图 1 android audio 框架图

2.4. audio 相关术语介绍

Audio Driver: Acronyms	
Acronym	**Definition**
ALSA	Advanced Linux Sound Architecture
A2DP	Advanced Audio Distribution Profile(蓝牙音频传输高保真模式)
EARPIECE	打电话的听筒.
headphone	耳机
headset	headphone 上加了一个话筒，即带话筒的耳机
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。

3. 模块接口描述

3.1. 音频设备定义

```
enum audio_devices {
// output devices
DEVICE_OUT_EARPIECE = 0x1, // 听筒
DEVICE_OUT_SPEAKER = 0x2, // 扬声器
DEVICE_OUT_WIRED_HEADSET = 0x4, // 耳机
DEVICE_OUT_WIRED_HEADPHONE = 0x8, // 耳机（另一种耳机，双耳听筒）
DEVICE_OUT_BLUETOOTH_SCO = 0x10, // 蓝牙 SCO, 用于语音通话
DEVICE_OUT_BLUETOOTH_SCO_HEADSET = 0x20, // 蓝牙 SCO 耳机
DEVICE_OUT_BLUETOOTH_SCO_CARKIT = 0x40, // 蓝牙 SCO 车载
DEVICE_OUT_BLUETOOTH_A2DP = 0x80, // 蓝牙高保真设备, 用于听音乐
DEVICE_OUT_BLUETOOTH_A2DP_HEADPHONES = 0x100, // 蓝牙高保真耳机
DEVICE_OUT_BLUETOOTH_A2DP_SPEAKER = 0x200, // 蓝牙高保真扬声器
DEVICE_OUT_AUX_DIGITAL = 0x400, // 辅助数字输出, HDMI
DEVICE_OUT_ANLG_DOCK_HEADSET = 0x800, // 模拟底座
DEVICE_OUT_DGTL_DOCK_HEADSET = 0x1000, // 数字底座, spdif
DEVICE_OUT_DEFAULT = 0x8000,
}
```

音频设备的切换是有 AudioPolicyService 管理的。

3.2. 函数接口说明

3.2.1. start_call

PROTOTYPE

static int start_call(struct tuna_audio_device *adev)

ARGUMENTS

adev　　整个 audio 驱动接口的总指针；含有所有结构体参数；

RETURNS

成功返回 0，失败返回-ENOMEM；-12；

DESCRIPTION

打开电话通话通路，如果上行和下行都可以打开返回 0 成功，否则返回失败。

3.2.2. end_call

PROTOTYPE

static void end_call(struct tuna_audio_device *adev)

ARGUMENTS

adev　　整个 audio 驱动接口的总指针；含有所有结构体参数；

RETURNS

无返回；

DESCRIPTION

关闭 start_call 打开电话通话通路。

3.2.3. set_incall_device

PROTOTYPE

static void set_incall_device(struct tuna_audio_device *adev)

ARGUMENTS

adev 整个 audio 驱动接口的总指针；含有所有结构体参数；

RETURNS

无返回；

DESCRIPTION

设置录音通路

3.2.4. set_input_volumes

PROTOTYPE

static void set_input_volumes(struct tuna_audio_device *adev, int main_mic_on,
int headset_mic_on, int sub_mic_on)

ARGUMENTS

adev 整个 audio 驱动接口的总指针；含有所有结构体参数；

main_mic_on 主麦

headset_mic_on 耳麦

sub_mic_on 副麦

RETURNS

无返回

DESCRIPTION

根据 mic 驱动的类型设置输入的声音的音调；

3.2.5. force_all_standby

PROTOTYPE

static void force_all_standby(struct tuna_audio_device *adev)

ARGUMENTS

adev 整个 audio 驱动接口的总指针；含有所有结构体参数；

RETURNS

无返回；

DESCRIPTION

调用 do_output_standy 和 do_input_standy 关闭所有 audio 驱动，包括 input 和 output，echo 等

3.2.6. select_mode

PROTOTYPE

static void select_mode(struct tuna_audio_device *adev)

ARGUMENTS

adev 整个 audio 驱动接口的总指针；含有所有结构体参数；

RETURNS

无返回；

DESCRIPTION

如果输入/输出设置改变，先关闭输入/输出设备，然后打开要输入/输出的设备

3.2.7. select_output_device

PROTOTYPE

static void select_output_device(struct tuna_audio_device *adev)

ARGUMENTS

adev 整个 audio 驱动接口的总指针；含有所有结构体参数；

RETURNS

无返回；

DESCRIPTION

根据策略设置输出设备的寄存器，设置 eq，音量大小，策略设置等

3.2.8. select_input_device

PROTOTYPE

static void select_input_device(struct tuna_audio_device *adev)

ARGUMENTS

adev 整个 audio 驱动接口的总指针；含有所有结构体参数；

RETURNS

无返回；

DESCRIPTION

根据策略设置输入设备的寄存器，音量大小，策略设置等

3.2.9. start_output_stream

PROTOTYPE

static int start_output_stream(struct tuna_stream_out *out)

ARGUMENTS

out 输出驱动总指针，在 adev 结构体内；

RETURNS

成功返回 0，失败返回-ENOMEM；-12

DESCRIPTION

输出模式 low_latency,调用 select_output_device 选择要打开的设备，调用 pcm_open 打开硬件驱动，开 android 和 linux 之前重采样的 buffer：out->buffer

3.2.10. check_input_parameters

PROTOTYPE

static int check_input_parameters(uint32_t sample_rate, audio_format_t format, int channel_count)

ARGUMENTS

sample_rate 采样率

Format bits of sample 模型

channel_count 通道数

RETURNS

成功返回 0，错误返回-EINVAL；-22；

DESCRIPTION

检验三个输入参数是否正确。

3.2.11. get_input_buffer_size

PROTOTYPE

static size_t get_input_buffer_size(uint32_t sample_rate, audio_format_t format, int

channel_count)

ARGUMENTS

sample_rate 采样率

Format bits of sample 模型

channel_count 通道数

RETURNS

错误返回长度 0，否则返回 input buffer

DESCRIPTION

调用 check_input_parameters 检查输入参数是否正确，根据 period_sizes 值算出 AudioRecordbuffer 大小，单位 bytes

3.2.12. do_output_standby

PROTOTYPE

static int do_output_standby(struct tuna_stream_out *out)

ARGUMENTS

out 输出驱动总指针，在 adev 结构体内。

RETURNS

返回 0；

DESCRIPTION

关闭所有 output 通路，包括 echo 等。

3.2.13. out_standby

PROTOTYPE

static int out_standby(struct audio_stream *stream)

ARGUMENTS

stream 输出驱动总指针，在 adev 结构体内。

RETURNS

返回状态；

DESCRIPTION

调用 do_output_standby,关闭所有 output 通路。

3.2.14. out_set_parameters

PROTOTYPE

static int out_set_parameters(struct audio_stream *stream, const char *kvpairs)

ARGUMENTS

stream 强制指针，其实是输出驱动总指针，在 audio 驱动 adrv 结构体内

Kvpairs 设置参数字符串

RETURNS

返回 ret

DESCRIPTION

返回 ret，是否有这个参数，本函数设置参数，本函数赋值 audio 驱动指针，可以供上层调用。

3.2.15. out_get_sample_rate

PROTOTYPE

static uint32_t out_get_sample_rate(const struct audio_stream *stream)

ARGUMENTS

stream 强制指针，其实是输出驱动总指针，在 audio 驱动 adrv 结构体内

RETURNS

返回采样率

DESCRIPTION

返回采样率，本函数赋值 audio 驱动指针，可以供上层调用。

3.2.16. out_get_parameters

PROTOTYPE

static char * out_get_parameters(const struct audio_stream *stream, const char *keys)

ARGUMENTS

stream 强制指针，其实是输出驱动总指针，在 audio 驱动 adrv 结构体内

Keys 字符串指针

RETURNS

返回字符串

DESCRIPTION

返回字符串，本函数赋值 audio 驱动指针，可以供上层调用。

3.2.17. out_write

PROTOTYPE

static ssize_t out_write(struct audio_stream_out *stream, const void* buffer,
size_t bytes)

ARGUMENTS

stream 强制指针，其实是输出驱动总指针，在 audio 驱动 adrv 结构体内

Buffer 写指针的首地址

Bytes 写 buffer 的长度

RETURNS

返回写的长度

DESCRIPTION

如果没有打开 audio 驱动，先打开 audio 驱动，如果需要重采样，进行重采样，如果需要进行 echo 处理，进行 echo 处理，往 linux 写数据 pcm_write 方式。本函数赋值 audio 驱动指针，可以供上层调用。

3.2.18. in_read

PROTOTYPE

static ssize_t in_read(struct audio_stream_in *stream, void* buffer, size_t bytes))

ARGUMENTS

stream 强制指针，其实是输出驱动总指针，在 audio 驱动 adrv 结构体内

Buffer 读指针的首地址

Bytes 读 buffer 的长度

RETURNS

返回读的长度

DESCRIPTION

如果没有打开 audio 录音驱动，先打开 audio 录音驱动，如果需要重采样，进行重采样，如果需要进行 echo 处理，进行 echo 处理，从 linux 读数据 pcm_read 方式。本函数可以供上层调用。

3.3. Android 3G

3.3.1. 声音切换

3.3.1.1. AP 端音频切换

请参考《A31 平台音频模块开发说明文档》文档。

3.3.1.2. BP 端音频切换

请参考《A31 平台电话系统开发说明文档》文档。

3.3.2. 3G 开发环境搭建

请参考《A31 平台电话系统开发说明文档》文档

3.3.3. 蓝牙通话

请参考《A31 平台电话系统开发说明文档》文档