Rockchip
瑞芯微电子

# Rockchip
# Audio 开发指南

发布版本:1.00

日期:2017.02

# 前言

## 概述

本文档主要介绍 RK 平台 Audio 框架介绍以及配置。

## 产品版本

| 芯片名称 | 内核版本 |
| --- | --- |
| RK312X | Linux 3.10 |
| RK3188 | Linux 3.10 |
| RK322X | Linux 3.10 |
| RK3328 | Linux 3.10 |
| RK3368 | Linux 3.10 |
| RV1108 | Linux 3.10 |

## 读者对象

本文档（本指南）主要适用于以下工程师：
技术支持工程师
软件开发工程师

## 修订记录

| 日期 | 版本 | 作者 | 修改说明 |
| --- | --- | --- | --- |
| 2017-02-15 | V1.0 | ZXG | 第一次临时版本发布 |

# 目录

# 1 概述

## 1.1 概述

本章主要描述 Audio 的相关的概念、代码结构。

## 1.2 概念

- CPU DAI:主控端的 Audio Data Interface，比如 I²S,Spdif,Pdm,Tdm
- CODEC DAI：即 Codec
- DAI_LINK:绑定 Cpu_Dai 和 Codec_Dai 为一个声卡，等同于 Machine Driver。
- DMAENGINE：用于 Cpu 和 I²S/Spdif 等 Dai 之间的 Dma 传输引擎，实际是通过 Dma 来进行数据的搬运。
- DAPM：动态音频电源管理，用于动态管理 Codec 等的电源管理，根据通路的开启配置开关，以达到保证功能的前提下功耗尽量小。
- JACK:耳机的接口检测,大部分使用 Codec 自身的检测机制,小部分使用 IO 来进行模拟。

## 1.3 代码结构

表 1-1 SOUND 代码构成

| 项目 | 功能 | 路径 |
|---|---|---|
| Sound soc | 主要包含公共部分代码，包括 dapm 控制， jack，dmaengine，core 等等 | sound/soc/ |
| rockchip platform | Rockchip 平台的 cpu dai 的驱动，比如 I²S, spdif 等以及自定义声卡 machine driver | sound/soc/rockchip |
| generic platform | simple card framework | sound/soc/generic |
| codec driver | 所有的 codec driver 存放位置 | sound/soc/codecs |

# 2 Audio 开发指南

## 2.1 概述

本章描述如何添加声卡，调试声卡以及通路等。

## 2.2 音频开发指南

一个声卡包含 cpu_dai, codec_dai, 以及 dai_link 组成，分别对应 cpu dai 的 dirver，比如 I²S driver, spdif driver；codec driver， 比如 rt5640 codec driver；dai_link driver，也就是 machine driver， 比如 sound/soc/rockchip/rockchip_rt5640.c。 4.4 的内核中支持两种方式创建声卡，一种是通用的 simple-card framework，一种是传统的编写自定义的 machine driver 来创建。本文档均以 rt5640 为例。

### 2.2.1 自定义的 Machine Driver

3.10 的内核，simple-card 还不完善， 所以在 3.10 的内核中，主要以 machine driver 为主。这个时候就需要编写相对应的 machine driver，比如：
sound/soc/rockchip/rockchip_rt5640.c，然后在这个 machine driver 添加特殊的控制，路由等等。这里不做举例，延续原有的格式，以及目录下均有参考代码可作为参照。

### 2.2.2 声卡调试

1. 通过如下命令确认声卡是否注册成功

```
root@rk3366:/ # cat /proc/asound/cards
 0 [rockchiprt5640c]: rockchip_rt5640 - rockchip,rt5640-codec
                      rockchip,rt5640-codec

root@rk3366:/ # ls -l /dev/snd/
crw-rw---- system    audio     116,    2 2013-01-18 08:51 controlC0
crw-rw---- system    audio     116,    4 2013-01-18 08:51 pcmC0D0c
crw-rw---- system    audio     116,    3 2013-01-18 08:51 pcmC0D0p
```

2. 通过命令行播放录制调试声卡：

播放：一般播放 1khz 0db 正弦波，然后在 codec 输出端示波器简单测量是否失真，杂音，然后再使用音频分析仪测试指标。

```
root@rk3366:/ # tinyplay
Usage: tinyplay file.wav [-D card] [-d device] [-p period_size] [-n n_periods]

|root@rk3366:/ # tinyplay /sdcard/test44.wav -D 0 -d 0 -p 1024 -n 3
Playing sample: 2 ch, 44100 hz, 32 bit
```

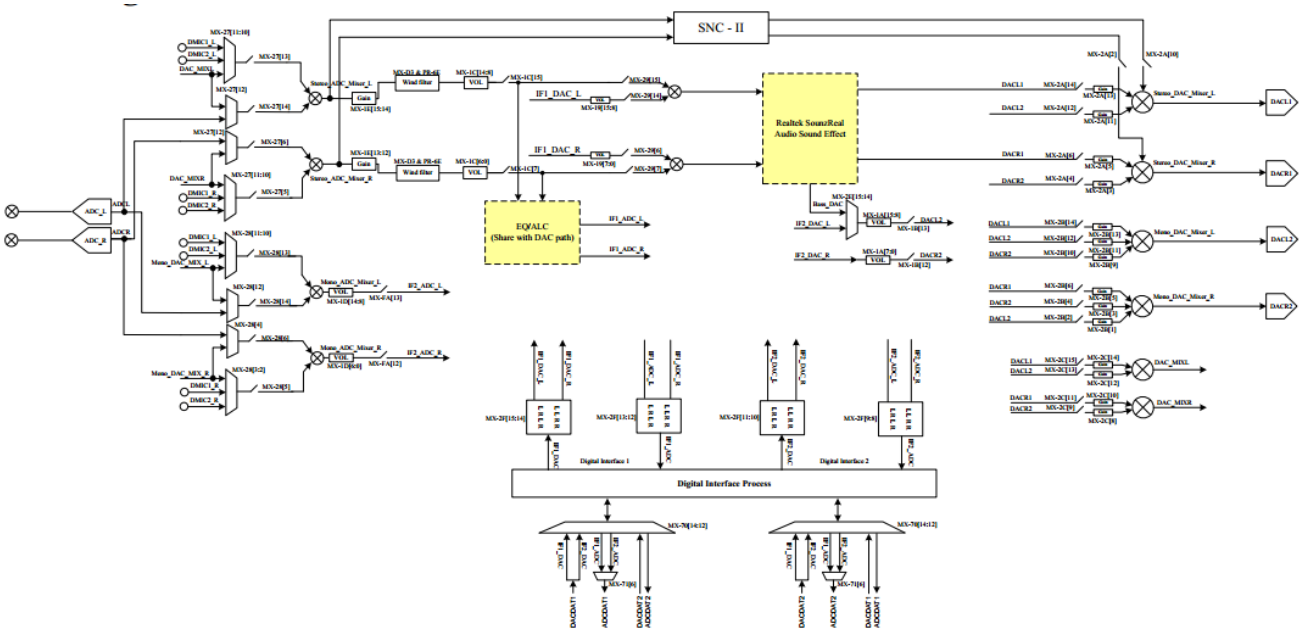
录制：

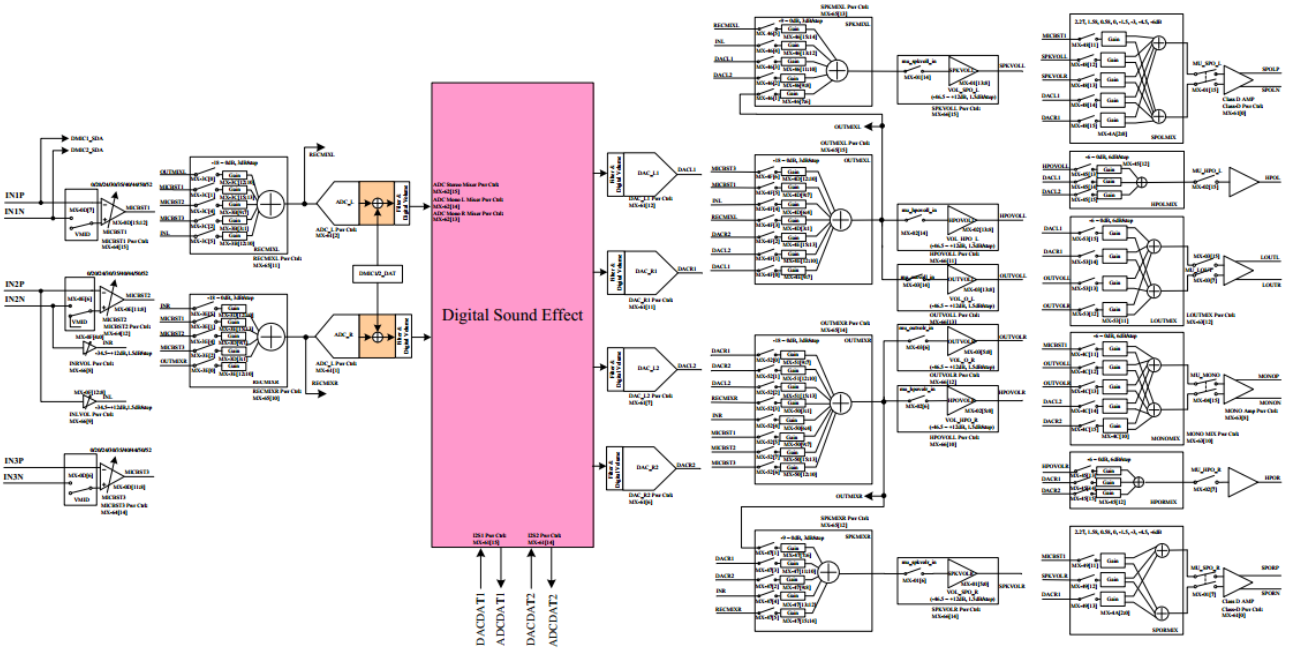
```
root@rk3366:/ # tinycap
Usage: tinycap file.wav [-D card] [-d device] [-c channels] [-r rate] [-b bits] [-p
period_size] [-n n_periods]
```

```
|root@rk3366:/ # tinycap /sdcard/rec.wav -D 0 -d 0 –c 2 –r 44100 –b 16 –p 1024 –n 3
```

3. 通过命令行调试声卡的通路：

一般复杂的 codec 可提供各种通路的配置，如下图：分别是数字部分通路和模拟部分通路，通路旁边都有标注控制的寄存器 bit，codec driver 负责将这些控制实例化为 kcontrol，提供给上层设置切换通路使用，实际的调试方法为从数字部分的通路开始，比如 DACDAT 然后顺着找出一条最优的路径到达模拟输出端，比如 HPOUT。然后通过 tinymix 控制路径上的相应节点开关，打通通路。




4. tinymix 调试通路：

```
root@rk3366:/ # tinymix
Mixer name: 'rockchip,rt5640-codec'
```

```
Number of controls: 123
ctl      type    num     name                                value
0        BOOL    1        Mono Playback Switch                Off
1        INT     2       Mono DAC Playback Volume             175 175
2        BOOL    2        Speaker Channel Switch              Off Off
3        INT     2       Speaker Playback Volume              31 31
4        BOOL    2        HP Channel Switch                   Off Off
```

可通路 ctr id 或者 name 来控制，例子如下，不带 val 设置时，为查询该 mix 的当前状态

```
root@rk3366:/ # tinymix 0 1
root@rk3366:/ # tinymix 0
Mono Playback Switch: On

root@rk3366:/ # tinymix "Mono Playback Switch" 1
root@rk3366:/ # tinymix "Mono Playback Switch"
Mono Playback Switch: On
```

5. 声卡功能以及通路调试 ok 后，需要把通路配置配置到 hal 层，然后可以配置不同场景下的通路路由，通路的配置即为 tinymix 配置成功后的通路列表的值，把这些值做成相应 codec_config.h 加入到 hal 中，举例如下：

```
hardware/rockchip/audio/tinyalsa_hal/codec_config/rt5640_config.h
#ifndef _RT5640_CONFIG_H_
#define _RT5640_CONFIG_H_

#include "config.h"

const struct config_control rt5640_speaker_normal_controls[] = {
    {
        .ctl_name = "DAI select",
        .str_val = "1:2|2:1",
    },
    {
        .ctl_name = "Mono DAC Playback Volume",
        .int_val = {175, 175},
    },
    {
        .ctl_name = "DAC2 Playback Switch",
        .int_val = {on, on},
},
……
hardware/rockchip/audio/tinyalsa_hal/codec_config/config_list.h
struct alsa_sound_card_config sound_card_config_list[] = {
    ……
    {
        .sound_card_name = "rockchiprt5640c",
        .route_table = &rt5640_config_table,
```

```
        },
            ……
```

通过以上步骤即完成基本的声卡创建，简单调试， 以上使用的 tinyplay, tinycap, tinymix 代码位于 android/external/tinyalsa 中，如果系统中没有该命令，可进到该目录执行 mm 生成相应的命令。

# 3 常用调试方法

## 3.1 常用调试方法：

1. 查看 codec 寄存器，I²S 寄存器，spdif 寄存器等等，出现问题时，往往需要常看寄存器的状态是否正常，来定位分析问题。

a，凡是使用 regmap 的驱动， 在/sys/kernel/debug/regmap 都有相应的查询入口，如下：

```
root@rk3366:/sys/kernel/debug/regmap # ls
0-001c
0-0040
1-001c
ff880000.spdif
ff898000.i2s-8ch
```

例如：1-001c 为 rt5640 的 i2c 地址，挂载在 i2c1, codec 地址为 0x1c，那么此目录中的 registers 即为 codec 的 register，其他类似。

2. Xrun debug， 一般用于 debug underrun 或者 overrun，出现此两者情况时内核会打印 log 协助问题的定位分析。Menuconfig 中需要开启如下选项：

```
[*]   Advanced Linux Sound Architecture  --->
[*]   Debug
[*]     More verbose debug
[*]     Enable PCM ring buffer overrun/underrun debugging
```

然后在对应声卡/proc/asound/card0/xrun 中写入相应的值，值如下：

```
#define XRUN_DEBUG_BASIC          (1<<0)
#define XRUN_DEBUG_STACK          (1<<1)  /* dump also stack */
#define XRUN_DEBUG_JIFFIESCHECK (1<<2)  /* do jiffies check */
```

比如 echo 1 > xrun 或者 echo 3 > xrun 或者 echo 7 > xrun 开启所有 debug 信息检测。

3. 通过查看 clk tree 确认相应的 audio clk 是否正常，比如 mclk：如下为采样率为 44100hz 的 mclk： 11.2896M。

```
c at /sys/kernel/debug/clk/clk_summary | grep i2s
              i2s_2ch_src                 0        0   576000000
0 0
                 i2s_2ch_frac             0        0    28800000
0 0
              i2s_8ch_src                 0        0   576000000
0 0
                 i2s_8ch_frac             0        0    11289600
0 0
                    i2s_8ch_pre           0        0    11289600
0 0
                       sclk_i2s_8ch       0        0    11289600
0 0
                       i2s_8ch_clkout        0           0    11289600
```

0 0

4. 要学会使用示波器测量音频的信号， 软件方式的确认有时会有误差，最精确最根本的方式就是确认音频 clk 是否正常，满足规范。音频的信号包含 mclk, bclk, lrck, data。需要确认信号幅度是否正常，如果 io 电压为 3.3v，测试出来的信号幅值应当在 3.3v 左右。如果幅值太低，则会照成采集不到数据而无声。Clk 的频偏也不宜过大，有可能会照成杂音。Bclk, lrck 要符合设置的采样率，如果不相符，则会照成音频快进或者播放缓慢。
5. 播放测试：一般播放 1khz 0db 正弦波，然后使用示波器确认输出是否有削顶失真，相位失真，杂音等。
6. 录音测试：可使用信号发生器产生 1khz 的波形从 codec 模拟端导入，然后录制波形，可以通过回放来确认波形是否正常，无失真，或者使用电脑上的软件工具 adobe audition 来分析底噪等等基本指标。
7. 基本功能过完后，需要使用音频分析仪进行 codec 后续的指标测试以及调优。