T113 Linux Audio
开发指南

版本号: 1.1
发布日期: 2022.05.29

版本历史

版本号	日期	制/修订人	内容描述
1.0	2022.05.22	AWA1692	添加初版音频驱动开发说明文档
1.1	2022.05.29	AWA1692	添加 sun8iw20 linux5.4 开发说明

目　录

表　格

1 前言

1.1 文档简介

本文档基于 sunxi 平台基础音频框架介绍，能够让使用者在 sunxi 平台开发使用音频驱动，内容分四大部分。

1、**“音频时钟树-> 驱动特性-> 音频流通路-> 设备树配置-> 编译配置及加载-> 声卡控件介绍-> 常见使用方法”**等部分，介绍音频驱动的测试验证和使用；

2、**“源码结构-> 驱动框架-> 关键数据结构-> 接口说明”**等部分，介绍音频驱动的二次开发；

3、**“声卡查看-> 声卡测试工具-> 外挂 codec”**等部分，介绍声卡模块的测试验证和使用；

4、**“FAQ”**章节：调试方法及常见问题汇总。

1.2 目标读者

音频系统相关人员。

1.3 适用范围

表 1-1: 适用产品列表

产品名称	内核版本	驱动文件
sun8iw20	Linux-5.4	KERNEL/sound/soc/sunxi/*

说明

该文档涵盖以上适用产品进行说明，部分规格并非平台共有，具体规格参考“模块介绍”章节。

1.4 相关术语

表 1-2: 硬件术语

相关术语	解释说明
AudioCodec	芯片内置音频接口
I2S/PCM	外置数字音频接口，常用于外接 codec 模块
AHUB	音频集线器，内部集成 I2S 接口及 DAM 混音器，可实现多路输入播放及硬件混音功能
S/PDIF	外置音响音频设备接口，一般使用同轴电缆或光纤接口
DMIC	外置数字 MIC 接口
同源播放	不同音频模块同时播放同一份音频数据
同步采样	不同音频模块同时录音（可消除线程调度时差影响）

表 1-3: 软件术语

相关术语	解释说明
ALSA	Advanced Linux Sound Architecture
ASOC	ALSA System on Chip
DAPM	动态音频电源管理
samplebit	样本精度，记录音频数据最基本的单位，常见的有 16 位
channel	通道数，该参数为 1 表示单声道，2 表示立体声，大于 2 表示多声道
rate	采样率，每秒钟采样次数，该次数是针对帧而言
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
period size	每次硬件中断处理音频数据的帧数
period count	处理完一个 buffer 数据所需的硬件中断次数
buffer size	数据缓冲区大小 (period size * period count)
DRC	音频输出动态范围控制
HPF	高通滤波
XRUN	音频流异常状态，分为 underrun 和 overrun 两种状态
交错模式	一种音频数据记录模式，数据以连续帧形式存放 (帧 1_L, 帧 1_R, 帧 2_L, 帧 2_R, · · ·)
非交错模式	一种音频数据记录模式，数据是以连续通道形式存放 (L-帧 1, L-帧 2, · · ·, R-帧 2, R-帧 2, · · ·)
tinyalsa	在 Linux 内核中与 ALSA 接口对接的库，可用于基本播录
alsalib	在 Linux 内核中与 ALSA 接口对接的库，可用于基本播录，并可与常见音频算法组合使用

2 模块介绍

在 sunxi 中，从 Linux 软件上通常存在 5 类音频设备，分别为：

- AudioCodec
- I2S/PCM
- AHUB
- DMIC
- S/PDIF

以上每一类音频设备均适配 asoc 架构。

2.1 sun8iw20 音频接口

> AudioCodec x1
> I2S/PCM x4
> S/PDIF x1
> DMIC x1
> 内核：linux5.4

2.1.1 时钟树

sun8iw20 音频模块时钟源来自 PLL_AUDIO0 和 PLL_AUDIO1_DIV5。

PLL_AUDIO0 可输出 22.5792M，PLL_AUDIO1_DIV5 可输出 24.576M 频率的时钟，分别支持 44.1K 系列、48K 系列的播放录音。

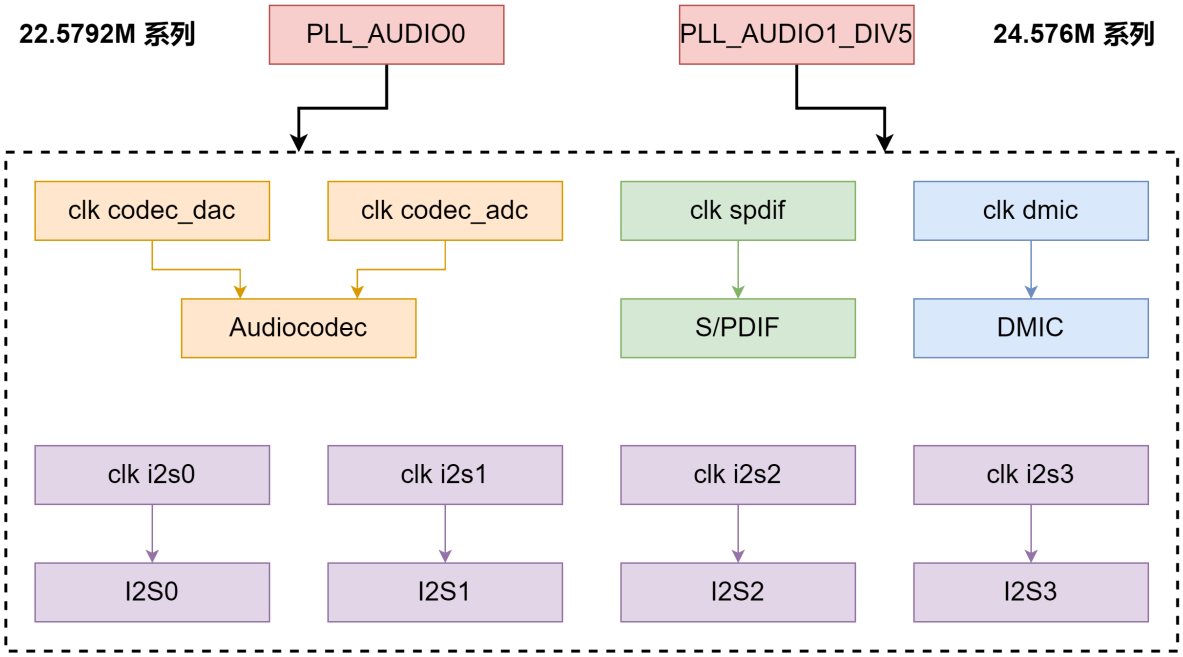

图 2-1：时钟树 sun8iw20

2.1.2 AudioCodec

2.1.2.1 驱动特性

- 支持多种采样率格式
  - 播放：8~192KHz
  - 录音：8~48KHz
- 支持多通道播放和录音
  - 播放：1~2
  - 录音：1~3
- 支持 16/24/32bit 数据精度（硬件支持 16bit/20bit）
- 支持硬件 HPF、DRC 算法
- 支持的物理接口
  - INPUT : mic, fmin, linein
  - OUTPUT: hpout, lineout
- 支持同时 playback 和 capture (全双工模式)
- 支持多声卡同源播放
- 支持多声卡同步录音

说明

***mic, lineout* 支持差分和单端模式**

2.1.2.2 音频流通路

播放流

```
Playback ──> DACL ──> HPOUTL ──> HPOUT
Playback ──> DACR ──> HPOUTR ──^

Playback ──> DACL ──> LINEOUTL Output Select ──> LINEOUTL ──> LINEOUT
Playback ──> DACR ──> LINEOUTR Output Select ──> LINEOUTR ──^

... ──> HPOUTL ──> SPK
... ──> HPOUTR ──> SPK
... ──> LINEOUTL ──> SPK
... ──> LINEOUTR ──> SPK
```

录音流

```
MIC1 ──> MIC1 Input Select ──> ADC1 ──> Capture
MIC2 ──> MIC2 Input Select ──> ADC2 ──> Capture
MIC3 ──> MIC3 Input Select ──> ADC3 ──> Capture

FMIN ──> LINEINL ──> ADC1 ──> Capture
     └─> LINEINR ──> ADC2 ──> Capture

LINEIN ──> LINEINL ──> ADC1 ──> Capture
       └─> LINEINR ──> ADC2 ──> Capture
```

2.1.2.3 Device Tree 配置

```
codec:codec@2030000 {
    #sound-dai-cells = <0>;
    compatible  = "allwinner,sunxi-internal-codec";
    reg         = <0x0 0x02030000 0x0 0x34c>;
    clocks      = <&ccu CLK_PLL_AUDIO0>,
                  <&ccu CLK_PLL_AUDIO1_DIV5>,
                  <&ccu CLK_AUDIO_DAC>,
                  <&ccu CLK_AUDIO_ADC>,
                  <&ccu CLK_BUS_AUDIO_CODEC>;
    clock-names = "pll_audio0", "pll_audio1_div5",
                  "audio_clk_dac", "audio_clk_adc",
                  "audio_clk_bus";
    resets      = <&ccu RST_BUS_AUDIO_CODEC>;
    rx_sync_en  = <0x00>;
    device_type = "codec";
    status = "okay";
};

dummy_cpudai:dummy_cpudai@203034c {
    compatible      = "allwinner,sunxi-dummy-cpudai";
    reg             = <0x0 0x0203034c 0x0 0x4>;
    tx_fifo_size    = <128>;
    rx_fifo_size    = <256>;
    dac_txdata      = <0x02030020>;
    adc_txdata      = <0x02030040>;
```

```
    playback_cma    = <128>;
    capture_cma     = <256>;
    device_type     = "cpudai";
    dmas            = <&dma 7>, <&dma 7>;
    dma-names       = "tx", "rx";
    status = "okay";
};

sndcodec:sound@2030340 {
    compatible = "allwinner,sunxi-codec-machine";
    reg         = <0x0 0x02030340 0x0 0x4>;
    interrupts  = <GIC_SPI 25 IRQ_TYPE_LEVEL_HIGH>;
    sunxi,audio-codec = <&codec>;
    sunxi,cpudai-controller = <&dummy_cpudai>;
    device_type = "sndcodec";
    status = "okay";
};
```

配置项说明：

AudioCodec 模块由 3 个设备树节点构建。

1、ASoC 层 codec: codec

表 2-1: AudioCodec codec 节点配置项

配置项名称	配置项说明
#sound-dai-cells	machine 层检测 codec 和 platform 节点的标志
reg	设置 audiocodec 寄存器起始地址和地址长度
resets	设置 audiocodec 所需的复位时钟
clocks	设置 audiocodec 所需的时钟源和模块时钟
clock-names	对 clocks 属性内容进行名称定义，用于辅助 clocks 属性获取

2、ASoC 层 platform: dummy_cpudai

表 2-2: AudioCodec dummy_cpudai 节点配置项

配置项名称	配置项说明
#sound-dai-cells	machine 层检测 codec 和 platform 节点的标志
playback-cma	设置播放流 DMA 申请 size 大小，为 (2^n)Kbyte，单位 Kb，默认 128
capture-cma	设置录音流 DMA 申请 size 大小，为 (2^n)Kbyte，单位 Kb，默认 128
tx-fifo-size	设置播放流的 fifo_size 大小，用于声卡参数限定，单位 Kb，默认 128
rx-fifo-size	设置录音流的 fifo_size 大小，用于声卡参数限定，单位 Kb，默认 128
dac-txdata	设置播放流 DMA 搬运地址（audiocodec 模块 tx_fifo 寄存器地址）
adc-txdata	设置录音流 DMA 搬运地址（audiocodec 模块 rx_fifo 寄存器地址）

3、ASoC 层 machine: sndcodec

表 2-3: AudioCodec sndcodec 节点配置项

配置项名称	配置项说明
sunxi,cpudai-controller	machine 层所绑定的 cpu 节点（即 platform 层）
sunxi,audio-codec	machine 层所绑定的 codec 节点（即 codec 层）
interrupts	耳机中断号

2.1.2.4 board.dts 板级配置

```
&codec {
    /* MIC and headphone gain setting */
    mic1gain        = <0x13>;
    mic2gain        = <0x13>;
    mic3gain        = <0x13>;
    /* ADC/DAC DRC/HPF func enabled */
    /* 0x1:DAP_HP_EN; 0x2:DAP_SPK_EN; 0x3:DAP_HPSPK_EN */
    adcdrc_cfg      = <0x0>;
    adchpf_cfg      = <0x1>;
    dacdrc_cfg      = <0x0>;
    dachpf_cfg      = <0x0>;
    /* Volume about */
    digital_vol     = <0x00>;
    lineout_vol     = <0x1a>;
    headphonegain   = <0x03>;
    /* Pa enabled about */
    pa_level        = <0x01>;
    pa_pwr_level    = <0x01>;
    pa_msleep_time  = <0x78>;
    /* gpio-spk     = <&pio PF 2 GPIO_ACTIVE_HIGH>; */
    /* gpio-spk-pwr = <&pio PF 4 GPIO_ACTIVE_HIGH>; */
    /* regulator about */
    /* avcc-supply  = <&reg_aldo1>; */
    /* hpvcc-supply = <&reg_eldo1>; */
    status = "okay";
};

&dummy_cpudai {
    status = "okay";
};

&sndcodec {
    hp_detect_case  = <0x01>;
    jack_enable     = <0x01>;
    status = "okay";
};
```

配置项说明：

表 2-4: AudioCodec 模块板级配置项

配置项名称	配置值范围	配置项说明
status	“okay”, “disabled”	使能或关闭该节点驱动
mic1gain	1~31	mic1 增益
mic2gain	1~31	mic2 增益
mic3gain	1~31	mic3 增益
adcdrc_cfg	0~1	adcdrc_cfg 选择
adchpf_cfg	0~1	adchpf_cfg 选择
dacdrc_cfg	0~1	dacdrc_cfg 选择
dachpf_cfg	0~1	dachpf_cfg 选择
digital_vol	0~63	DAC 数字音量
lineout_vol	0~31(-43.5->0,0dB)	lineout 默认输出音量（增益）
headphonegain	0~7	耳机播放增益
pa_level	0~1	指定功放芯片使能电平
pa_pwr_level	0~1	指定功放芯片供电使能电平
pa_msleep_time	u32（正常小于 200）	设置功放芯片使能所需的 sleep 时长，用于规避 pop 声，单位 ms
rx-sync-en	注释为 false, 反之为 ture	选择是否注册 rxsync 控件

说明

以上配置项并非所有板型均必须包含，以板型具体规格为准，进行裁剪。

2.1.2.5 kernel menuconfig 配置

```
Device Drivers  --->
    <*> Sound card support  --->
        <*> Advanced Linux Sound Architecture  --->
            <*> ALSA for SoC audio support  --->
                Allwinner SoC Audio support  --->
                    <M> Allwinner Sun20iw1 Codec Support
                    [M] Allwinner RX SYNC Support
```

配置项说明：

选择需要的模块，可选择直接编译进内核（Y），也可编译成模块（M）。

声卡配置：

- Allwinner Sun20iw1 Codec Support
  - Audiocodec 模块

特定功能配置：

- Allwinner RX SYNC Support
  - 同步采样功能组件

编译模式：

- Y: 编入内核，跟随系统启动加载；
- M: 编译为模块，系统启动后加载，方便调试。

2.1.2.6 加载卸载方法（若编译为 ko）

sunxi 音频驱动模块分五大类驱动如下。

- 特定功能组件（rxsync、jack 等）
- pcm 驱动
- ASOC platfrom 驱动
- ASOC codec 驱动
- ASOC machine 驱动

ko 加载顺序遵循 **“特殊功能组件 -> pcm 驱动 -> ASOC platfrom 驱动或 ASOC codec 驱动 -> ASOC machine 驱动”** 顺序，卸载顺序则相反。

Audiocodec 声卡加载顺序如下。

```
# 特殊功能组件(若使能 rxsync 功能)
insmod sunxi-rx-sync.ko

# pcm 驱动
insmod snd-soc-sunxi.ko

# ASOC platfrom 驱动 和 ASOC codec 驱动
insmod snd-soc-sunxi-dummy-cpudai.ko
insmod sun20iw1-codec.ko

# ASOC machine 驱动
insmod sun20iw1-sndcodec.ko
```

2.1.2.7 声卡控件

控件列表

```
Mixer name: 'audiocodec'
Number of controls: 32
ctl     type    num     name                                    value
0       ENUM    1       codec hub mode                          >hub_disable hub_enable
1       ENUM    1       ADC1 ADC2 swap                          >Off On
2       ENUM    1       ADC3 ADC4 swap                          >Off On
3       INT     1       digital volume                          63 (dsrange 0->63)
4       INT     2       DAC volume                              160 160 (dsrange 0->255)
5       INT     1       ADC1 volume                             160 (dsrange 0->255)
6       INT     1       ADC2 volume                             160 (dsrange 0->255)
7       INT     1       ADC3 volume                             160 (dsrange 0->255)
8       INT     1       MIC1 gain volume                        0 (dsrange 0->31)
9       INT     1       MIC2 gain volume                        0 (dsrange 0->31)
10      INT     1       MIC3 gain volume                        19 (dsrange 0->31)
11      BOOL    1       FMINL gain volume                       Off
12      BOOL    1       FMINR gain volume                       Off
13      BOOL    1       LINEINL gain volume                     Off
14      BOOL    1       LINEINR gain volume                     Off
15      INT     1       LINEOUT volume                          26 (dsrange 0->31)
16      INT     1       Headphone volume                        4 (dsrange 0->7)
17      ENUM    1       LINEOUTL Output Select                  DAC_SINGLE >DAC_DIFFER
18      ENUM    1       LINEOUTR Output Select                  DAC_SINGLE >DAC_DIFFER
19      ENUM    1       MIC1 Input Select                       >MIC_DIFFER MIC_SINGLE
20      ENUM    1       MIC2 Input Select                       >MIC_DIFFER MIC_SINGLE
21      ENUM    1       MIC3 Input Select                       >MIC_DIFFER MIC_SINGLE
22      BOOL    1       ADC1 Input MIC1 Boost Switch            Off
23      BOOL    1       ADC1 Input FMINL Switch                 Off
24      BOOL    1       ADC1 Input LINEINL Switch               On
25      BOOL    1       ADC2 Input MIC2 Boost Switch            Off
26      BOOL    1       ADC2 Input FMINR Switch                 Off
27      BOOL    1       ADC2 Input LINEINR Switch               On
28      BOOL    1       ADC3 Input MIC3 Boost Switch            On
29      BOOL    1       Headphone Switch                        On
30      BOOL    1       HpSpeaker Switch                        Off
31      BOOL    1       LINEOUT Switch                          Off
```

表 2-5: 特殊功能类控件说明

控件名称	功能	数值
codec hub mode	同源播放开关	Off;On
ADC1 ADC2 swap	ADC1&2 通道交换开关	Off;On
ADC3 ADC4 swap	ADC3&4 通道交换开关	Off;On

表 2-6: 音量调节类控件说明

控件名称	功能	数值
digital volume	播放数字端音量调节	0->63 (-73.08->0dB)
DAC volume	DAC 音量调节	0->255 (-119.25->71.25dB)
ADC1 volume	ADC1 音量调节	0->255 (-119.25->71.25dB)
ADC2 volume	ADC2 音量调节	0->255 (-119.25->71.25dB)
ADC3 volume	ADC3 音量调节	0->255 (-119.25->71.25dB)
MIC1 gain volume	MIC1 增益调节	0->31 (0,6,6,6,9->36dB)

控件名称	功能	数值
MIC2 gain volume	MIC2 增益调节	0->31 (0,6,6,6,9->36dB)
MIC3 gain volume	MIC3 增益调节	0->31 (0,6,6,6,9->36dB)
FMINL gain volume	FMINL 增益调节	Off;On (0->6dB)
FMINR gain volume	FMINR 增益调节	Off;On (0->6dB)
LINEINL gain volume	LINEINL 增益调节	Off;On (0->6dB)
LINEINR gain volume	LINEINR 增益调节	Off;On (0->6dB)
Headphone volume	HPOUT 增益调节	0->7 (-42->0,0dB)
LINEOUT volume	LINEOUT 增益调节	0->31 (-43.5->0,0dB)

表 2-7: 通路开关类控件说明

控件名称	功能	数值
LINEOUTL Output Select	LINEOUTL 输出模式	single;differ
LINEOUTR Output Select	LINEOUTR 输出模式	single;differ
MIC1 Input Select	MIC1 输入模式	single;differ
MIC2 Input Select	MIC2 输入模式	single;differ
MIC3 Input Select	MIC3 输入模式	single;differ
ADC1 Input MIC1 Boost Switch	MIC1 通路开关	Off;On
ADC2 Input MIC2 Boost Switch	MIC2 通路开关	Off;On
ADC3 Input MIC3 Boost Switch	MIC3 通路开关	Off;On
ADC1 Input FMINL Switch	FMINL 通路开关	Off;On
ADC2 Input FMINR Switch	FMINR 通路开关	Off;On
ADC1 Input LINEINL Switch	LINEINL 通路开关	Off;On
ADC2 Input LINEINR Switch	LINEINR 通路开关	Off;On
Headphone Switch	HPOUT 通路开关	Off;On
LINEOUT Switch	LINEOUT 通路开关	Off;On
HpSpeaker Switch	SPK 通路开关	Off;On

2.1.2.8 常用使用方法

1. 以 tinyalsa 工具举例说明，具体使用方法见 **“模块使用-> 声卡测试工具使用->tinyalsa 工具”** 章节；
2. 假设 audiocodec 声卡序号为 0。

录音

MIC1 单通道录音

```
# 录音通路控件
tinymix -D 0 "ADC1 Input MIC1 Boost Switch" 1
tinycap mic.wav -D 0 -c 1 -T 10
```

MIC2 单通道录音

```
# 录音通路控件
tinymix -D 0 "ADC2 Input MIC2 Boost Switch" 1
tinycap mic.wav -D 0 -c 1 -T 10
```

MIC1&2&3 双通道输入

```
# 录音通路控件
tinymix -D 0 "ADC1 Input MIC1 Boost Switch" 1
tinymix -D 0 "ADC2 Input MIC2 Boost Switch" 1
tinymix -D 0 "ADC3 Input MIC3 Boost Switch" 1
tinycap mic.wav -D 0 -c 3 -T 10
```

FMIN 双通道输入

```
# 录音通路控件
tinymix -D 0 "ADC1 Input FMINL Switch" 1
tinymix -D 0 "ADC2 Input FMINR Switch" 1
tinymix -D 0 "MIC1 Input Select" 1
tinymix -D 0 "MIC2 Input Select" 1
tinycap linein.wav -D 0 -c 2 -T 10
```

LINEIN 双通道输入

```
# 录音通路控件
tinymix -D 0 "ADC1 Input LINEINL Switch" 1
tinymix -D 0 "ADC2 Input LINEINR Switch" 1
tinymix -D 0 "MIC1 Input Select" 1
tinymix -D 0 "MIC2 Input Select" 1
tinycap linein.wav -D 0 -c 2 -T 10
```

说明

***MIC, LINEIN, FMIN* 输入属于 *mux* 的关系，即一个 *ADC* 只能选一个输入源，不能同时开启 *MIC, LINEIN, FMIN* 的通路开关。**

播放

LINEOUT 双通道播放

```
# 播放通路控件
tinymix -D 0 "LINEOUT Switch" 1
tinyplay test_2ch.wav -D 0
```

HPOUT 双通道播放

```
# 播放通路控件
tinymix -D 0 "Headphone Switch" 1
tinyplay test_2ch.wav -D 0
```

Speaker 单/双通道播放

```
# 上述播放示例基础上将 "SPK Switch" 控件打开即可
tinymix -D 0 "HpSpeaker Switch" 1
```

2.1.3 I2S/PCM

2.1.3.1 驱动特性

- 支持多种采样率格式
  - 播放：8~192KHz
  - 录音：8~192KHz
- 支持多通道播放和录音
  - 播放：1~16
  - 录音：1~16
- 支持 16/20/24/32bit 数据精度（硬件支持 8/12/16/20/24/28/32bit）
- 支持 5 种 TDM 模式
  - I2S standard mode
  - Left-justified mode
  - Right-justified mode
  - DSP-A mode (short frame PCM mode)
  - DSP-B mode (long frame PCM mode)
- 支持 loopback 回环模式
- 支持同时 playback 和 capture (全双工模式)
- 支持硬件重采样
- 支持多声卡同源播放
- 支持多声卡同步录音

2.1.3.2 音频流通路

播放流

```
Playback ──> I2S DOUT
```

录音流

```
I2S DIN ──> Capture
```

回环流

```
Playback ——> Capture
```

2.1.3.3 Device Tree 配置

```
daudio0:daudio@2032000 {
    #sound-dai-cells = <0>;
    compatible = "allwinner,sunxi-daudio";
    reg = <0x0 0x02032000 0x0 0xa0>;
    clocks = <&ccu CLK_PLL_AUDIO0>,
             <&ccu CLK_I2S0>,
             <&ccu CLK_BUS_I2S0>;
    clock-names = "pll_audio", "i2s0", "i2s0_bus";
    resets = <&ccu RST_BUS_I2S0>;
    dmas        = <&dma 3>, <&dma 3>;
    dma-names   = "tx", "rx";
    interrupts-extended = <&plic0 42 IRQ_TYPE_LEVEL_HIGH>;
    sign_extend     = <0x00>;
    tx_data_mode        = <0x00>;
    rx_data_mode        = <0x00>;
    msb_lsb_first       = <0x00>;
    pcm_lrck_period     = <0x80>;
    slot_width_select   = <0x20>;
    frametype       = <0x00>;
    tdm_config      = <0x01>;
    tdm_num         = <0x00>;
    mclk_div        = <0x00>;
    clk_parent      = <0x01>;
    capture_cma     = <128>;
    playback_cma        = <128>;
    tx_num          = <4>;
    tx_chmap1       = <0x76543210>;
    tx_chmap0       = <0xFEDCBA98>;
    rx_num          = <4>;
    rx_chmap3       = <0x03020100>;
    rx_chmap2       = <0x07060504>;
    rx_chmap1       = <0x0B0A0908>;
    rx_chmap0       = <0x0F0E0D0C>;
    asrc_function_en    = <0x00>;
    rx_sync_en              = <0x00>;
    device_type = "daudio0";
    status = "disabled";
};

sounddaudio0: sounddaudio0@20320a0 {
    reg = <0x0 0x020320a0 0x0 0x4>;
    compatible = "sunxi,simple-audio-card";
    simple-audio-card,name = "snddaudio0";
    simple-audio-card,format = "i2s";
    status = "disabled";
    /* simple-audio-card,frame-master = <&daudio0_master>; */
    /* simple-audio-card,bitclock-master = <&daudio0_master>; */
    /* simple-audio-card,bitclock-inversion; */
    /* simple-audio-card,frame-inversion; */
    simple-audio-card,cpu {
        sound-dai = <&daudio0>;
    };
};
```

配置项说明：

I2S/PCM 模块由 2 个或 3 个设备树节点构建。

1、ASoC 层 codec: 非必须节点，若无，则绑定虚拟 codec 节点。

2、ASoC 层 platform: daudio(n)

表 2-8: I2S/PCM daudio(n)_plat 节点配置项

配置项名称	配置项说明
#sound-dai-cells	machine 层检测 codec 和 platform 节点的标志
reg	设置 I2S/PCM 寄存器起始地址和地址长度
resets	设置 I2S/PCM 所需的复位时钟
clocks	设置 I2S/PCM 所需的时钟源和模块时钟
clock-names	对 clocks 属性内容进行名称定义，用于辅助 clocks 属性获取
playback-cma	设置播放流 DMA 申请 size 大小，为 (2^n)Kbyte，单位 Kb，默认 128
capture-cma	设置录音流 DMA 申请 size 大小，为 (2^n)Kbyte，单位 Kb，默认 128
tx-fifo-size	设置播放流的 fifo_size 大小，用于声卡参数限定，单位 Kb，默认 128
rx-fifo-size	设置录音流的 fifo_size 大小，用于声卡参数限定，单位 Kb，默认 128

3、ASoC 层 machine: sounddaudio(n)

表 2-9: I2S/PCM daudio(n)_mach 节点配置项

配置项名称	配置项说明
simple-audio-card,	machine 层配置前缀
name	声卡名字
cpu	machine 层所绑定的 cpu 节点（即 platform 层）， 用 sound-dai 属性指定节点
codec	machine 层所绑定的 codec 节点（即 codec 层）， 用 sound-dai 属性指定节点若该子节点下无 sound-dai 属性， 即代表使用虚拟 codec，用于辅助生成声卡

2.1.3.4 board.dts 板级配置

```
&daudio0 {
    mclk_div    = <0x01>;
    frametype   = <0x00>;
    tdm_config  = <0x01>;
    sign_extend     = <0x00>;
    msb_lsb_first   = <0x00>;
    pcm_lrck_period = <0x80>;
    slot_width_select = <0x20>;
    pinctrl-names   = "default", "sleep";
    pinctrl-0       = <&daudio0_pins_a>;
    pinctrl-1       = <&daudio0_pins_b>;
    pinctrl_used    = <0x0>;
    status = "disabled";
};
```

```
&sounddaudio0 {
    /* simple-audio-card,format = "i2s"; */
    /* simple-audio-card,frame-master = <&daudio0_master>; */
    /* simple-audio-card,bitclock-master = <&daudio0_master>; */
    /* simple-audio-card,bitclock-inversion; */
    /* simple-audio-card,frame-inversion; */
    status = "disabled";
    daudio0_master: simple-audio-card,codec {
        /* sound-dai = <&ac108>; */
    };
};
```

配置项说明：

表 2-10: I2S/PCM 模块板级配置项

配置项名称	配置值范围	配置项说明
status	“okay”, “disabled”	使能或关闭该节点驱动
pcm_lrck_period	u32	lrck 宽度
slot_width_select	u32	slot 宽度
mclk_div	u32	mclk 分频系数
daudio_type	0~1	外挂 codec 或 HDMI
format	“i2s”,“right_j”,“left_j”, “dsp_a”,“dsp_b”	选择 tdm 协议格式
frame-master	cpu 子节点，codec 子节点	选择 LRCK 信号主模式
bitclock-master	cpu 子节点，codec 子节点	选择 BCLK 信号主模式
frame-inversion	注释为 false, 反之为 ture	LRCK 信号是否翻转
bitclock-inversion	注释为 false, 反之为 ture	BCLK 信号是否翻转

2.1.3.5 kernel menuconfig 配置

```
Device Drivers  --->
    <*> Sound card support  --->
        <*> Advanced Linux Sound Architecture  --->
            <*> ALSA for SoC audio support  --->
                Allwinner SoC Audio support V2  --->
                    <M> Allwinner Audio Simple Card
                    <M>   Allwinner Digital Audio Support
                    <M>     Allwinner Digital Audio ASRC Support
                    <M>     Allwinner HDMI Audio Support
                    [M] Allwinner RX SYNC Support
```

配置项说明：

> 选择需要的模块，可选择直接编译进内核（Y），也可编译成模块（M）。

声卡配置：

- Allwinner Digital Audio Support
  - I2S/PCM 模块
- Allwinner HDMI Audio Support
  - I2S/PCM HDMI 模块

特定功能配置：

- Allwinner RX SYNC Support
  - 同步采样功能组件

编译模式：

- Y: 编入内核，跟随系统启动加载；
- M: 编译为模块，系统启动后加载，方便调试。

2.1.3.6 加载卸载方法（若编译为 ko）

sunxi 音频驱动模块分五大类驱动如下。

- 特定功能组件（rxsync、jack 等）
- pcm 驱动
- ASOC platfrom 驱动
- ASOC codec 驱动
- ASOC machine 驱动

ko 加载顺序遵循 **“特殊功能组件 -> pcm 驱动 -> ASOC platfrom 驱动或 ASOC codec 驱动 -> ASOC machine 驱动”** 顺序，卸载顺序则相反。

I2S/PCM 声卡加载顺序如下。

```
# 特殊功能组件(若使能 rxsync 功能)
insmod sunxi-rx-sync.ko

# pcm 驱动
insmod snd-soc-sunxi.ko

# ASOC platfrom 驱动 和 ASOC codec 驱动(NULL)
insmod snd-soc-sunxi-daudio.ko

# ASOC machine 驱动
insmod snd-soc-sunxi-simple-card.ko
```

2.1.3.7 声卡控件

控件列表

```
Mixer name: 'snddaudio0'
Number of controls: 2
ctl     type    num     name                                    value
0       ENUM    1       sunxi daudio audio hub mode             Disable
1       BOOL    1       sunxi daudio loopback debug             Off
```

表 2-11：控件说明

控件名称	功能	数值
sunxi daudio audio hub mode	同源播放开关	Off;On
sunxi daudio loopback debug	内部回录开关	Off;On

2.1.3.8 常用使用方法

1. 以 tinyalsa 工具举例说明，具体使用方法见 **“模块使用-> 声卡测试工具使用->tinyalsa 工具”** 章节；
2. 假设 snddaudio0 声卡序号为 0。

```
# snddaudio0 声卡录音，2channel 16bit 48000hz
tinycap test.wav -D 0 -c 2 -b 16 -r 48000 -T 10

# snddaudio0 声卡播放
tinyplay test.wav -D 0

# snddaudio0 声卡内部回环，播录音频格式需保持一致
tinymix -D 0 "sunxi daudio loopback debug" 1
tinyplay test_play.wav -D 0 &
tinycap test_cap.wav -D 0 -c 2 -b 16 -r 48000 -T 10
```

2.1.4 S/PDIF

2.1.4.1 驱动特性

- 支持多种采样率格式
  - 播放：22.05~192KHz
  - 录音：22.05~192KHz
- 支持多通道播放和录音

- 播放：1~2
- 录音：1~2

- 支持 16bit/20bit/24bit/32bit 数据精度（硬件支持 16/20/24bit）
- 支持 loopback 回环模式
- 支持同时 playback 和 capture (全双工模式)
- 支持 IEC-60958 协议
- 支持 IEC-61937 协议
- 支持多声卡同源播放
- 支持透传播放

2.1.4.2 音频流通路

播放流

```
Playback ──> SPDIF DOUT
```

录音流

```
SPDIF DIN ──> Capture
```

2.1.4.3 Device Tree 配置

```
spdif:spdif@2036000 {
    #sound-dai-cells = <0>;
    compatible = "allwinner,sunxi-spdif";
    reg = <0x0 0x02036000 0x0 0x58>;
    clocks = <&ccu CLK_PLL_AUDIO0_4X>,
             <&ccu CLK_SPDIF_TX>,
             <&ccu CLK_BUS_SPDIF>,
             <&ccu CLK_PLL_AUDIO1>,
             <&ccu CLK_PLL_AUDIO1_DIV5>,
             <&ccu CLK_PLL_PERIPH0>,
             <&ccu CLK_SPDIF_RX>;
    clock-names = "pll_audio0", "spdif", "spdif_bus",
                  "pll_audio1", "pll_audio1_div5",
                  "pll_periph","spdif_rx";
    resets = <&ccu RST_BUS_SPDIF>;
    dmas            = <&dma 2>, <&dma 2>;
    dma-names       = "tx", "rx";
    interrupts-extended = <&plic0 41 IRQ_TYPE_LEVEL_HIGH>;
    clk_parent      = <0x1>;
    playback_cma    = <128>;
    capture_cma     = <128>;
    rx_sync_en      = <0>;
    device_type     = "spdif";
    status = "disabled";
};
soundspdif:soundspdif@2036040 {
```

```
    reg = <0x0 0x02036040 0x0 0x4>;
    compatible = "sunxi,simple-audio-card";
    simple-audio-card,name = "sndspdif";
    status = "disabled";
     simple-audio-card,cpu {
         sound-dai = <&spdif>;
     };
     simple-audio-card,codec {
         /*snd-soc-dummy*/
     };
};
```

配置项说明：

S/PDIF 模块由 2 个设备树节点构建。

1、ASoC 层 codec: 无，绑定虚拟 codec 节点。

2、ASoC 层 platform: spdif

表 2-12: SPDIF spdif 节点配置项

配置项名称	配置项说明
#sound-dai-cells	machine 层检测 codec 和 platform 节点的标志
reg	设置 S/PDIF 寄存器起始地址和地址长度
resets	设置 S/PDIF 所需的复位时钟
clocks	设置 S/PDIF 所需的时钟源和模块时钟
clock-names	对 clocks 属性内容进行名称定义，用于辅助 clocks 属性获取
playback-cma	设置播放流 DMA 申请 size 大小，为 (2^n)Kbyte，单位 Kb，默认 128
capture-cma	设置录音流 DMA 申请 size 大小，为 (2^n)Kbyte，单位 Kb，默认 128
tx-fifo-size	设置播放流的 fifo_size 大小，用于声卡参数限定，单位 Kb，默认 128
rx-fifo-size	设置录音流的 fifo_size 大小，用于声卡参数限定，单位 Kb，默认 128
dmas	设置模块所绑定的 dma 通道号
dma-names	对 dmas 属性内容进行名称定义，用于辅助 dmas 属性获取

3、ASoC 层 machine: soundspdif

表 2-13: S/PDIF soundspdif 节点配置项

配置项名称	配置项说明
simple-audio-card,	machine 层配置前缀
name	声卡名字
cpu	machine 层所绑定的 cpu 节点（即 platform 层），用 sound-dai 属性指定节点
codec	machine 层所绑定的 codec 节点（即 codec 层），用 sound-dai 属性指定节点（使用虚拟 codec）

2.1.4.4 board.dts 板级配置

```
&spdif {
    pinctrl-names   = "default","sleep";
    pinctrl-0       = <&spdif_pins_a>;
    pinctrl-1       = <&spdif_pins_b>;
    status = "okay";
};

&soundspdif {
    status = "okay";
};
```

配置项说明：

表 2-14: S/PDIF 模块板级配置项

配置项名称	配置值范围	配置项说明
status	“okay”, “disabled”	使能或关闭该节点驱动

2.1.4.5 kernel menuconfig 配置

```
Device Drivers  --->
    <*> Sound card support  --->
        <*> Advanced Linux Sound Architecture  --->
            <*> ALSA for SoC audio support  --->
                Allwinner SoC Audio support  --->
                    <M> Allwinner SPDIF Support
```

配置项说明：

> 选择需要的模块，可选择直接编译进内核（Y），也可编译成模块（M）。

声卡配置：

- Allwinner SPDIF Support
  - SPDIF 模块

编译模式：

- Y: 编入内核，跟随系统启动加载；
- M: 编译为模块，系统启动后加载，方便调试。

2.1.4.6 加载卸载方法（若编译为 ko）

sunxi 音频驱动模块分五大类驱动如下。

- 特定功能组件（rxsync、jack 等）
- pcm 驱动
- ASOC platfrom 驱动
- ASOC codec 驱动
- ASOC machine 驱动

ko 加载顺序遵循 **“特殊功能组件 -> pcm 驱动 -> ASOC platfrom 驱动或 ASOC codec 驱动 -> ASOC machine 驱动”** 顺序，卸载顺序则相反。

S/PDIF 声卡加载顺序如下。

```
# pcm 驱动
insmod snd-soc-sunxi.ko

# ASOC platfrom 驱动 和 ASOC codec 驱动(NULL)
insmod snd-soc-sunxi-spdif.ko

# ASOC machine 驱动
insmod snd-soc-sunxi-simple-card.ko
```

2.1.4.7 声卡控件

控件列表

```
Mixer name: 'sndspdif'
Number of controls: 3
ctl     type    num     name                                    value
0       ENUM    1       spdif audio format function             >Disable Enable
1       ENUM    1       sunxi spdif hub mode                    >PCM DTS
2       BOOL    1       sunxi spdif loopback debug              Off
```

表 2-15: 控件说明

控件名称	功能	数值
spdif audio format function	同源播放开关	Disable;Enable
sunxi spdif hub mode	设置音频数据格式	0~1(对应控件 value 枚举值)
sunxi spdif loopback debug	内部回录开关	Off;On

2.1.4.8 常用使用方法

1. 以 tinyalsa 工具举例说明，具体使用方法见 **“模块使用-> 声卡测试工具使用->tinyalsa 工具”** 章节，
2. 假设 spdif 声卡序号为 0。

播放

```
tinyplay test.wav -D 0
```

2.1.5 DMIC

2.1.5.1 驱动特性

- 支持多种采样率格式
  - 8~48KHz
- 支持多通道录音
  - channels：1~8
- 支持 16bit/24bit 数据精度
- 支持硬件 HPF 算法
- 支持多声卡同步录音

2.1.5.2 音频流通路

录音流

```
DMIC DIN ——> Capture
```

2.1.5.3 Device Tree 配置

```
dmic:dmic@2031000{
    #sound-dai-cells = <0>;
    compatible = "allwinner,sunxi-dmic";
    reg = <0x0 0x02031000 0x0 0x50>;
    clocks = <&ccu CLK_PLL_AUDIO0>,
             <&ccu CLK_DMIC>,
             <&ccu CLK_BUS_DMIC>;
    clock-names = "pll_audio", "dmic", "dmic_bus";
    resets = <&ccu RST_BUS_DMIC>;
```

```
    dmas        = <&dma 8>;
    dma-names   = "rx";
    interrupts-extended = <&plic0 40 IRQ_TYPE_LEVEL_HIGH>;
    clk_parent  = <0x1>;
    capture_cma = <256>;
    data_vol    = <0xB0>;
    rx_chmap    = <0x76543210>;
    rx_sync_en  = <0x00>;
    device_type = "dmic";
    status = "disabled";
};
dmic_codec:sound@2031050{
    #sound-dai-cells = <0>;
    compatible = "dmic-codec";
    reg = <0x0 0x02031050 0x0 0x4>;
    num-channels = <8>;
    status = "disabled";
};
sounddmic:sounddmic@2031060 {
    reg = <0x0 0x02031060 0x0 0x4>;
    compatible = "sunxi,simple-audio-card";
    simple-audio-card,name = "snddmic";
    simple-audio-card,capture_only;
    status = "disabled";
    /* simple-audio-card,format = "i2s"; */
    simple-audio-card,cpu {
        sound-dai = <&dmic>;
    };
    simple-audio-card,codec {
        sound-dai = <&dmic_codec>;
    };
};
```

配置项说明：

DMIC 模块由 2 个设备树节点构建。

1、ASoC 层 codec: 无，绑定虚拟 codec 节点。

2、ASoC 层 platform: dmic

表 2-16: DMIC dmic 节点配置项

配置项名称	配置项说明
#sound-dai-cells	machine 层检测 codec 和 platform 节点的标志
reg	设置 DMIC 寄存器起始地址和地址长度
resets	设置 DMIC 所需的复位时钟
clocks	设置 DMIC 所需的时钟源和模块时钟
clock-names	对 clocks 属性内容进行名称定义，用于辅助 clocks 属性获取
capture-cma	设置录音流 DMA 申请 size 大小，为 (2^n)Kbyte，单位 Kb，默认 128
rx-fifo-size	设置录音流的 fifo_size 大小，用于声卡参数限定，单位 Kb，默认 128

3、ASoC 层 machine: sounddmic

表 2-17: DMIC sounddmic 节点配置项

配置项名称	配置项说明
simple-audio-card,	machine 层配置前缀
name	声卡名字
cpu	machine 层所绑定的 cpu 节点（即 platform 层），用 sound-dai 属性指定节点
codec	machine 层所绑定的 codec 节点（即 codec 层），用 sound-dai 属性指定节点（使用虚拟 codec）
capture-only	设置仅录音，不进行播放流设备创建

2.1.5.4 board.dts 板级配置

```
&dummy_cpudai {
    status = "okay";
};

&dmic {
    pinctrl-names   = "default","sleep";
    pinctrl-0       = <&dmic_pins_a>;
    pinctrl-1       = <&dmic_pins_b>;
    status = "okay";
};

&dmic_codec {
    status = "okay";
};

&sounddmic {
    status = "okay";
};
```

配置项说明：

表 2-18: DMIC 模块板级配置项

配置项名称	配置值范围	配置项说明
status	“okay”, “disabled”	使能或关闭该节点驱动
data-vol	0->255(-119.25->71.25dB)	录音数字端音量调节
rxdelaytime	0,5,10,20,30	设置录音延迟时长，单位 ms
rx-sync-en	注释为 false, 反之为 ture	选择是否注册 rxsync 控件

2.1.5.5 kernel menuconfig 配置

```
Device Drivers  --->
    <*> Sound card support  --->
        <*> Advanced Linux Sound Architecture  --->
            <*> ALSA for SoC audio support  --->
                Allwinner SoC Audio support V2  --->
                    <M> Allwinner DMIC Support
                    <M> Allwinner Function Components
                    [M] Allwinner RX SYNC Support
```

配置项说明：

> 选择需要的模块，可选择直接编译进内核（Y），也可编译成模块（M）。

声卡配置：

- Allwinner DMIC Support
  - DMIC 模块

特定功能配置：

- Allwinner RX SYNC Support
  - 同步采样功能组件

编译模式：

- Y: 编入内核，跟随系统启动加载；
- M: 编译为模块，系统启动后加载，方便调试。

2.1.5.6 加载卸载方法（若编译为 ko）

sunxi 音频驱动模块分五大类驱动如下。

- 特定功能组件（rxsync、jack 等）
- pcm 驱动
- ASOC platfrom 驱动
- ASOC codec 驱动
- ASOC machine 驱动

ko 加载顺序遵循 **“特殊功能组件 -> pcm 驱动 -> ASOC platfrom 驱动或 ASOC codec 驱动 -> ASOC machine 驱动”** 顺序，卸载顺序则相反。

DMIC 声卡加载顺序如下。

```
# 特殊功能组件(若使能 rxsync 功能)
insmod sunxi-rx-sync.ko

# pcm 驱动
insmod snd-soc-sunxi.ko

# ASOC platfrom 驱动 和 ASOC codec 驱动(NULL)
insmod snd-soc-sunxi-dmic.ko

# ASOC machine 驱动
insmod snd-soc-sunxi-simple-card.ko
```

2.1.5.7 声卡控件

控件列表

```
Mixer name: 'snddmic'
Number of controls: 0
```

2.1.5.8 常用使用方法

1. 以 tinyalsa 工具举例说明，具体使用方法见 **“模块使用-> 声卡测试工具使用->tinyalsa 工具”** 章节，
2. 假设 dmic 声卡序号为 0。

录音

```
# 2channel 16bit 48000rate
tinycap test.wav -D 0 -c 2 -b 16 -r 48000 -T 10

# 8channel 16bit 48000rate
tinycap test.wav -D 0 -c 8 -b 16 -r 48000 -T 10
```

2.1.6 GPIO 功能复用配置

- AudioCodec 模块：

  所用引脚功能均固化，无需进行 pin 功能复用配置。

- I2S/PCM，DMIC 模块：

可选择不进行 pin 功能复用配置，该情况仍可生成声卡，但引脚无实际功能输入输出。

```
&xxx_plat {
    ...
    pinctrl-used;
    pinctrl-names   = "default","sleep";
    pinctrl-0       = <&xxx_pins_a>;
    pinctrl-1       = <&xxx_pins_b>;
};
```

配置项说明：

表 2-19: GPIO 功能复用配置项

配置项名称	配置值范围	配置项说明
pinctrl-used	注释为 false, 反之为 ture	是否使用引脚复用功能
pinctrl-names	“default”,“sleep”	对 pinctrl 属性内容进行名称定义，用于辅助 pinctrl 属性获取
pinctrl-0	模块 pin 功能复用节点	对应 pinctrl-names 第 0 个属性
pinctrl-1	模块 pin 功能复用节点	对应 pinctrl-names 第 1 个属性

表 2-20: 模块引脚组定义说明 (linux5.4)

节点配置	解释说明
pins	模块需要使用到的引脚组定义
function	模块引脚组复用名称
drive-strength	模块引脚驱动力，可选值为 10,20,30,40，默认配置为 20 即可
bias-disable	关闭上下拉（默认选择该项）
bias-pull-up	支持上拉（默认关闭）
bias-pull-down	支持下拉（默认关闭）

3 模块接口说明

3.1 源文件列表

```
linux-5.4/sound/soc/sunxi
.
├── Kconfig
├── Makefile
├── sun20iw1-codec.c
├── sun20iw1-codec.h
├── sun20iw1-sndcodec.c
├── sun50iw10-codec.c
├── sun50iw10-codec.h
├── sun50iw10-sndcodec.c
├── sun50iw12-codec.c
├── sun50iw12-codec.h
├── sun50iw12-sndcodec.c
├── sun8iw20-codec.c
├── sun8iw20-codec.h
├── sun8iw20-sndcodec.c
├── sunxi-daudio.c
├── sunxi-daudio.h
├── sunxi-dmic.c
├── sunxi-dmic.h
├── sunxi-dummy-cpudai.c
├── sunxi-hdmi.c
├── sunxi-pcm.c
├── sunxi-pcm.h
├── sunxi-rx-sync.c
├── sunxi-rx-sync.h
├── sunxi-simple-card.c
├── sunxi_sound_log.h
├── sunxi-spdif.c
└── sunxi-spdif.h
```

3.1.1 源码说明

3.1.1.1 platform 层 -> 公共部分

```
sunxi-pcm.c
sunxi-pcm.h
```

负责音频流传输，使用 dma 方式，提供注册 platform 设备的公共函数。

3.1.1.2 platform 层 -> Audiocodec

```
sunxi-dummy-cpudai.c
```

负责 Audiocodec 模块 dma 相关配置。

3.1.1.3 platform 层 -> I2S/PCM

```
sunxi-daudio.c
sunxi-daudio.h
```

负责 I2S/PCM 模块硬件参数、dma 相关配置。

3.1.1.4 platform 层 -> SPDIF

```
sunxi-spdif.c
sunxi-spdif.h
```

负责 SPDIF 模块硬件参数、dma 相关配置。

3.1.1.5 platform 层 -> DMIC

```
sunxi-dmic.c
sunxi-dmic.h
```

负责 DMIC 模块硬件参数、dma 相关配置。

3.1.1.6 codec 层 -> Audiocodec

```
sun20iw1-codec.c
sun20iw1-codec.h
```

负责 Audiocodec 模块硬件参数配置 (sun8iw20)。

3.1.1.7 machine 层

```
sunxi-simple-card.c
```

负责 platform 层和 codec 层绑定。

3.1.1.8 特殊功能组件

```
sunxi-rx-sync.c
sunxi-rx-sync.h
```

负责多声卡同步录音。

3.2 软件框图

ALSA 音频架构

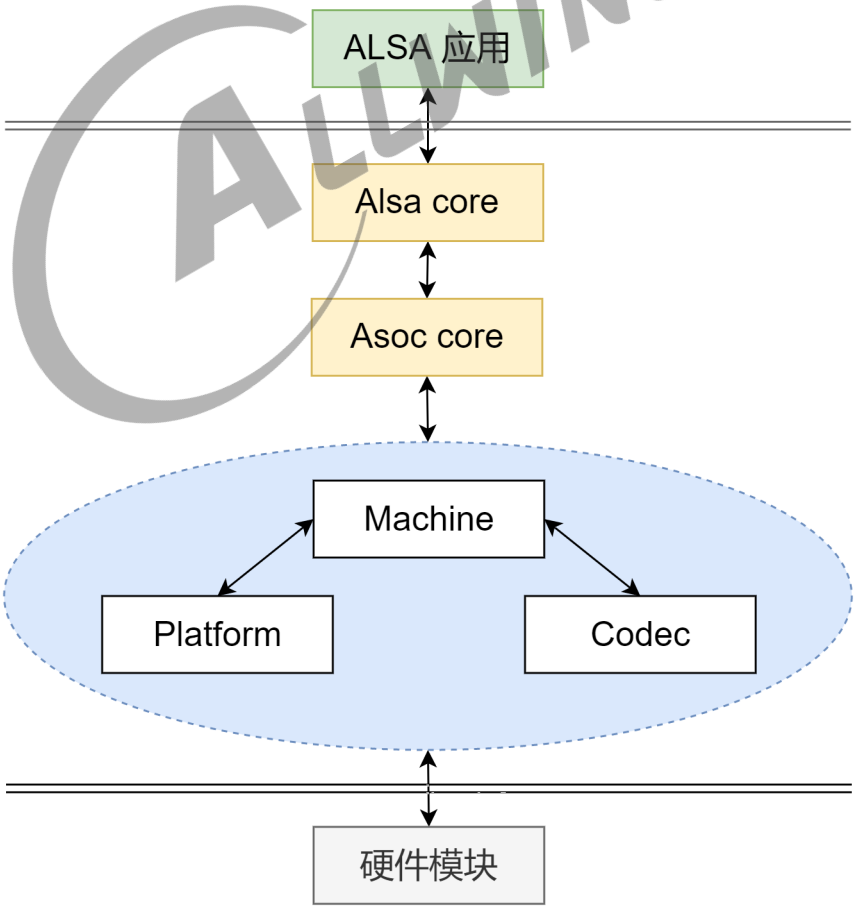

图 3-1: ALSA 音频架构

ASOC 驱动框架

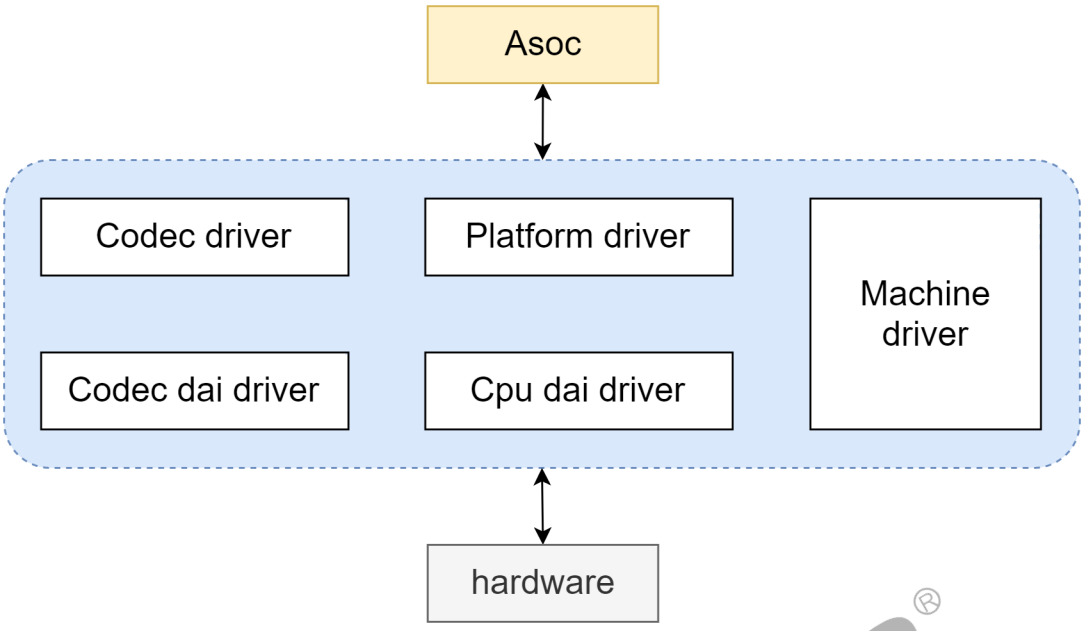

图 3-2: ASOC 驱动框架

3.3 关键数据结构（linux5.4）

仅说明自定义相关结构体和全局变量，alsa 框架内部结构体不做说明。

3.3.1 pcm 数据类结构体

3.3.1.1 sunxi_dma_params

```
struct sunxi_dma_params {
    ...
};
```

定义 audio dai dma 相关参数。

3.3.2 platform 类结构体

3.3.2.1 sunxi_daudio

```
struct sunxi_daudio {
    ...
};
```

I2S/PCM 模块总结构体，包含基础平台资源、特定功能私有参数。

3.3.2.2 sunxi_spdif

```
struct sunxi_spdif {
    ...
};
```

S/PDIF 模块总结构体，包含基础平台资源、特定功能私有参数。

3.3.2.3 sunxi_dmic

```
struct sunxi_dmic {
    ...
};
```

DMIC 模块总结构体，包含基础平台资源、特定功能私有参数。

3.3.3 codec 类结构体

3.3.3.1 sunxi_codec

```
struct sunxi_codec {
    ...
};
```

Audiocodec 模块总结构体，包含基础平台资源、特定功能私有参数。

3.4 软件重要接口（linux5.4）

> 仅说明自定义软件接口，alsa 框架内部接口不做说明。

3.4.1 pcm 相关接口

3.4.1.1 sunxi_pcm_new

- 函数原型：

```
int sunxi_pcm_new(struct snd_soc_pcm_runtime *rtd)
```

- 功能描述：创建 pcm 设备
- 参数说明：
  - rtd: pcm 流信息
- 返回值：0-成功，其它-失败

3.4.1.2 sunxi_pcm_free_dma_buffers

- 函数原型：

```
void sunxi_pcm_free_dma_buffers(struct snd_pcm *pcm)
```

- 功能描述：释放 pcm 设备
- 参数说明：
  - pcm: pcm 设备
- 返回值：void

3.4.1.3 sunxi_pcm_open

- 函数原型：

```
int sunxi_pcm_open(struct snd_pcm_substream *substream)
```

- 功能描述：开启 pcm 设备
- 参数说明：
  - substream: pcm 子流信息
- 返回值：0-成功，其它-失败

3.4.1.4 snd_dmaengine_pcm_close_release_chan

- 函数原型：

```
void snd_dmaengine_pcm_close_release_chan(struct snd_pcm_substream *substream)
```

- 功能描述：关闭 pcm 设备
- 参数说明：
  - substream: pcm 子流信息
- 返回值：void

3.4.1.5 snd_pcm_lib_ioctl

- 函数原型：

```
int snd_pcm_lib_ioctl(struct snd_pcm_substream *substream, unsigned int cmd, void *arg)
```

- 功能描述：pcm 设备操作
- 参数说明：
  - substream: pcm 子流信息
  - cmd: 操作命令
  - arg: 命令参数
- 返回值：0-成功，其它-失败

3.4.1.6 sunxi_pcm_hw_params

- 函数原型：

```
int sunxi_pcm_hw_params(struct snd_pcm_substream *substream, struct snd_pcm_hw_params *
    params)
```

- 功能描述：设置 pcm 设备参数
- 参数说明：
  - substream: pcm 子流信息
  - params: pcm 硬件参数
- 返回值：0-成功，其它-失败

3.4.1.7 sunxi_pcm_hw_free

- 函数原型：

```
int sunxi_pcm_hw_free(struct snd_pcm_substream *substream)
```

- 功能描述：释放 pcm 设备参数
- 参数说明：
  - substream: pcm 子流信息
- 返回值：0-成功，其它-失败

3.4.1.8 sunxi_pcm_trigger

- 函数原型：

```
int sunxi_pcm_trigger(struct snd_pcm_substream *substream, int cmd)
```

- 功能描述：触发 pcm 设备运行
- 参数说明：
  - substream: pcm 子流信息
  - cmd: 触发命令
- 返回值：0-成功，其它-失败

3.4.1.9 snd_dmaengine_pcm_pointer

- 函数原型：

```
snd_pcm_uframes_t snd_dmaengine_pcm_pointer(struct snd_pcm_substream *substream)
```

- 功能描述：获取 pcm 设备帧点
- 参数说明：
  - substream: pcm 子流信息
- 返回值：当前 DMA 缓冲指针

3.4.1.10 sunxi_pcm_mmap

- 函数原型：

```
int sunxi_pcm_mmap(struct snd_pcm_substream *substream, struct vm_area_struct *vma)
```

- 功能描述：创建 pcm 设备内存映射
- 参数说明：
  - substream: pcm 子流信息
  - vma: VMM 内存区域
- 返回值：0-成功，其它-失败

3.4.2 platform 层接口 -> Audiocodec

3.4.2.1 sunxi_cpudai_probe

- 函数原型：

```
int sunxi_cpudai_probe(struct snd_soc_dai *dai)
```

- 功能描述：初始化 DMA 参数
- 参数说明：
  - dai: cpu dai 信息
- 返回值：0-成功，其它-失败

3.4.2.2 sunxi_cpudai_startup

- 函数原型：

```
int sunxi_cpudai_startup(struct snd_pcm_substream *substream, struct snd_soc_dai *dai)
```

- 功能描述：更新设置 DMA 参数
- 参数说明：
  - substream：pcm 子流信息
  - dai：cpu dai 信息
- 返回值：0-成功，其它-失败

3.4.3 platform 层接口 -> I2S/PCM

3.4.3.1 sunxi_daudio_probe

- 函数原型：

```
int sunxi_daudio_probe(struct snd_soc_component *component)
```

- 功能描述：初始化 I2S/PCM 声卡相关信息（如控件初始化）
- 参数说明：
  - component：platform 层组件
- 返回值：0-成功，其它-失败

3.4.3.2 sunxi_daudio_suspend

- 函数原型：

```
int sunxi_daudio_suspend(struct snd_soc_dai *dai)
```

- 功能描述：I2S/PCM 模块休眠（保存寄存器、关闭电源、关闭时钟）
- 参数说明：
  - dai：cpu dai 信息
- 返回值：0-成功，其它-失败

3.4.3.3 sunxi_daudio_resume

- 函数原型：

```
int sunxi_daudio_resume(struct snd_soc_dai *dai)
```

- 功能描述：I2S/PCM 模块唤醒（开启电源、开启时钟、恢复寄存器）
- 参数说明：
  - dai: cpu dai 信息
- 返回值：0-成功，其它-失败

3.4.3.4 sunxi_daudio_set_clkdiv

- 函数原型：

```
int sunxi_daudio_set_clkdiv(struct snd_soc_dai *dai, int clk_id, int clk_div)
```

- 功能描述：设置模块 pllclk
- 参数说明：
  - dai: cpu dai 信息
  - pll_id: pll 辅助信息
  - source: pll 源
  - freq_in: 输入频率
  - freq_out: 输出频率
- 返回值：0-成功，其它-失败

3.4.3.5 sunxi_daudio_dai_set_sysclk

- 函数原型：

```
int sunxi_daudio_dai_set_sysclk(struct snd_soc_dai *dai, int clk_id, unsigned int freq, int
    dir)
```

- 功能描述：设置模块工作时钟
- 参数说明：
  - dai: cpu dai 信息
  - clk_id: clk 辅助信息
  - freq: 时钟频率
  - dir: 时钟输出方向
- 返回值：0-成功，其它-失败

3.4.3.6 sunxi_daudio_dai_set_fmt

- 函数原型：

```
int sunxi_daudio_dai_set_fmt(struct snd_soc_dai *dai, unsigned int fmt)
```

- 功能描述：设置模块 I2S 格式
- 参数说明：
  - dai: cpu dai 信息
  - fmt: I2S 格式信息
- 返回值：0-成功，其它-失败

3.4.3.7 sunxi_daudio_dai_startup

- 函数原型：

```
int sunxi_daudio_dai_startup(struct snd_pcm_substream *substream, struct snd_soc_dai *dai)
```

- 功能描述：设置模块开启工作资源 (dma 参数、组件功能等)
- 参数说明：
  - substream: pcm 子流信息
  - dai: cpu dai 信息
- 返回值：0-成功，其它-失败

3.4.3.8 sunxi_daudio_dai_hw_params

- 函数原型：

```
int sunxi_daudio_dai_hw_params(struct snd_pcm_substream *substream, struct
    snd_pcm_hw_params *params, struct snd_soc_dai *dai)
```

- 功能描述：设置模块硬件参数
- 参数说明：
  - substream: pcm 子流信息
  - params: 硬件参数
  - dai: cpu dai 信息
- 返回值：0-成功，其它-失败

3.4.3.9 sunxi_daudio_dai_prepare

- 函数原型：

```
int sunxi_daudio_dai_prepare(struct snd_pcm_substream *substream, struct snd_soc_dai *dai)
```

- 功能描述：清除模块 fifo
- 参数说明：
  - substream: pcm 子流信息
  - dai: cpu dai 信息
- 返回值：0-成功，其它-失败

3.4.3.10 sunxi_daudio_dai_trigger

- 函数原型：

```
int sunxi_daudio_dai_trigger(struct snd_pcm_substream *substream, int cmd, struct
    snd_soc_dai *dai)
```

- 功能描述：触发模块工作
- 参数说明：
  - substream: pcm 子流信息
  - cmd: 触发命令
  - dai: cpu dai 信息
- 返回值：0-成功，其它-失败

3.4.3.11 sunxi_daudio_dai_shutdown

- 函数原型：

```
void sunxi_daudio_dai_shutdown(struct snd_pcm_substream *substream, struct snd_soc_dai *dai
    )
```

- 功能描述：设置模块关闭工作资源 (组件功能等)
- 参数说明：
  - substream: pcm 子流信息
  - dai: cpu dai 信息
- 返回值：0-成功，其它-失败

3.4.4 platform 层接口 -> SPDIF

3.4.4.1 sunxi_spdif_component_probe

- 函数原型：

```
int sunxi_spdif_component_probe(struct snd_soc_component *component)
```

- 功能描述：初始化 SPDIF 声卡相关信息（如控件初始化）
- 参数说明：
  - component: platform 层组件
- 返回值：0-成功，其它-失败

3.4.4.2 sunxi_spdif_probe

- 函数原型：

```
int sunxi_spdif_probe(struct snd_soc_dai *dai)
```

- 功能描述：初始化 cpu dai (dma、模块寄存器)
- 参数说明：
  - dai: cpu dai 信息
- 返回值：0-成功，其它-失败

3.4.4.3 sunxi_spdif_remove

- 函数原型：

```
int sunxi_spdif_remove(struct snd_soc_dai *dai)
```

- 功能描述：移除 cpu dai (模块寄存器失能)
- 参数说明：
  - dai: cpu dai 信息
- 返回值：0-成功，其它-失败

3.4.4.4 sunxi_spdif_suspend

- 函数原型：

```
int sunxi_spdif_suspend(struct snd_soc_dai *dai)
```

- 功能描述：模块休眠（保存寄存器、关闭电源、关闭时钟）
- 参数说明：
  - dai: cpu dai 信息
- 返回值：0-成功，其它-失败

3.4.4.5 sunxi_spdif_resume

- 函数原型：

```
int sunxi_spdif_resume(struct snd_soc_dai *dai)
```

- 功能描述：模块唤醒（开启电源、开启时钟、恢复寄存器）
- 参数说明：
  - dai: cpu dai 信息
- 返回值：0-成功，其它-失败

3.4.4.6 sunxi_spdif_dai_set_clkdiv

- 函数原型：

```
int sunxi_spdif_dai_set_clkdiv(struct snd_soc_dai *dai, int clk_id, int clk_div)
```

- 功能描述：设置模块 pllclk
- 参数说明：
  - dai: cpu dai 信息
  - pll_id: pll 辅助信息
  - source: pll 源
  - freq_in: 输入频率
  - freq_out: 输出频率
- 返回值：0-成功，其它-失败

3.4.4.7 sunxi_spdif_dai_set_sysclk

- 函数原型：

```
int sunxi_spdif_dai_set_sysclk(struct snd_soc_dai *dai, int clk_id, unsigned int freq, int
    dir)
```

- 功能描述：设置模块工作时钟
- 参数说明：
  - dai: cpu dai 信息
  - clk_id: clk 辅助信息
  - freq: 时钟频率
  - dir: 时钟输出方向
- 返回值：0-成功，其它-失败

3.4.4.8 sunxi_spdif_dai_startup

- 函数原型：

```
int sunxi_spdif_dai_startup(struct snd_pcm_substream *substream, struct snd_soc_dai *dai)
```

- 功能描述：设置模块开启工作资源 (dma 参数、组件功能等)
- 参数说明：
  - substream: pcm 子流信息
  - dai: cpu dai 信息
- 返回值：0-成功，其它-失败

3.4.4.9 sunxi_spdif_dai_hw_params

- 函数原型：

```
int sunxi_spdif_dai_hw_params(struct snd_pcm_substream *substream, struct snd_pcm_hw_params
     *params, struct snd_soc_dai *dai)
```

- 功能描述：设置模块硬件参数
- 参数说明：

- substream: pcm 子流信息
- params: 硬件参数
- dai: cpu dai 信息

- 返回值：0-成功，其它-失败

3.4.4.10 sunxi_spdif_prepare

- 函数原型：

```
int sunxi_spdif_prepare(struct snd_pcm_substream *substream, struct snd_soc_dai *dai)
```

- 功能描述：清除模块 fifo
- 参数说明：

  - substream: pcm 子流信息
  - dai: cpu dai 信息

- 返回值：0-成功，其它-失败

3.4.4.11 sunxi_spdif_trigger

- 函数原型：

```
int sunxi_spdif_trigger(struct snd_pcm_substream *substream, int cmd, struct snd_soc_dai *
    dai)
```

- 功能描述：触发模块工作
- 参数说明：

  - substream: pcm 子流信息
  - cmd: 触发命令
  - dai: cpu dai 信息

- 返回值：0-成功，其它-失败

3.4.4.12 sunxi_spdif_dai_shutdown

- 函数原型：

```
void sunxi_spdif_dai_shutdown(struct snd_pcm_substream *substream, struct snd_soc_dai *dai)
```

- 功能描述：设置模块关闭工作资源 (组件功能等)
- 参数说明：
  - substream：pcm 子流信息
  - dai：cpu dai 信息
- 返回值：0-成功，其它-失败

3.4.5 platform 层接口 -> DMIC

3.4.5.1 sunxi_dmic_probe

- 函数原型：

```
int sunxi_dmic_probe(struct snd_soc_dai *dai)
```

- 功能描述：初始化 cpu dai (dma、模块寄存器)
- 参数说明：
  - dai：cpu dai 信息
- 返回值：0-成功，其它-失败

3.4.5.2 sunxi_dmic_suspend

- 函数原型：

```
int sunxi_dmic_suspend(struct snd_soc_dai *dai)
```

- 功能描述：模块休眠（保存寄存器、关闭电源、关闭时钟）
- 参数说明：
  - dai：cpu dai 信息
- 返回值：0-成功，其它-失败

3.4.5.3 sunxi_dmic_resume

- 函数原型：

```
int sunxi_dmic_resume(struct snd_soc_dai *dai)
```

- 功能描述：模块唤醒（开启电源、开启时钟、恢复寄存器）
- 参数说明：
  - dai: cpu dai 信息
- 返回值：0-成功，其它-失败

3.4.5.4 sunxi_dmic_set_sysclk

- 函数原型：

```
int sunxi_dmic_set_sysclk(struct snd_soc_dai *dai, int clk_id, unsigned int freq, int dir)
```

- 功能描述：设置模块 pllclk
- 参数说明：
  - dai: cpu dai 信息
  - pll_id: pll 辅助信息
  - source: pll 源
  - freq_in: 输入频率
  - freq_out: 输出频率
- 返回值：0-成功，其它-失败

3.4.5.5 sunxi_dmic_startup

- 函数原型：

```
int sunxi_dmic_startup(struct snd_pcm_substream *substream, struct snd_soc_dai *dai)
```

- 功能描述：设置模块开启工作资源 (dma 参数、组件功能等)
- 参数说明：
  - substream: pcm 子流信息
  - dai: cpu dai 信息
- 返回值：0-成功，其它-失败

3.4.5.6 sunxi_dmic_hw_params

- 函数原型：

```
int sunxi_dmic_hw_params(struct snd_pcm_substream *substream, struct snd_pcm_hw_params *
    params, struct snd_soc_dai *dai)
```

- 功能描述：设置模块硬件参数
- 参数说明：
  - substream: pcm 子流信息
  - params: 硬件参数
  - dai: cpu dai 信息
- 返回值：0-成功，其它-失败

3.4.5.7 sunxi_dmic_prepare

- 函数原型：

```
int sunxi_dmic_prepare(struct snd_pcm_substream *substream, struct snd_soc_dai *dai)
```

- 功能描述：清除模块 fifo
- 参数说明：
  - substream: pcm 子流信息
  - dai: cpu dai 信息
- 返回值：0-成功，其它-失败

3.4.5.8 sunxi_dmic_trigger

- 函数原型：

```
int sunxi_dmic_trigger(struct snd_pcm_substream *substream, int cmd, struct snd_soc_dai *
    dai)
```

- 功能描述：触发模块工作
- 参数说明：
  - substream: pcm 子流信息
  - cmd: 触发命令
  - dai: cpu dai 信息
- 返回值：0-成功，其它-失败

3.4.5.9 sunxi_dmic_shutdown

- 函数原型：

```
void sunxi_dmic_shutdown(struct snd_pcm_substream *substream, struct snd_soc_dai *dai)
```

- 功能描述：设置模块关闭工作资源 (组件功能等)
- 参数说明：
  - substream: pcm 子流信息
  - dai: cpu dai 信息
- 返回值：0-成功，其它-失败

3.4.6 codec 层接口 -> Audiocodec

3.4.6.1 sunxi_codec_probe

- 函数原型：

```
int sunxi_codec_probe(struct snd_soc_codec *snd_codec)
```

- 功能描述：初始化 Audiocodec 声卡相关信息（如控件初始化）
- 参数说明：
  - snd_codec: codec 驱动定义
- 返回值：0-成功，其它-失败

3.4.6.2 sunxi_codec_remove

- 函数原型：

```
int sunxi_codec_remove(struct snd_soc_codec *snd_codec)
```

- 功能描述：模块资源释放
- 参数说明：
  - snd_codec: codec 驱动定义
- 返回值：0-成功，其它-失败

3.4.6.3 sunxi_codec_suspend

- 函数原型：

```
int sunxi_codec_suspend(struct snd_soc_codec *snd_codec)
```

- 功能描述：模块休眠（保存寄存器、关闭电源、关闭时钟）
- 参数说明：
  - snd_codec: codec 驱动定义
- 返回值：0-成功，其它-失败

3.4.6.4 sunxi_codec_resume

- 函数原型：

```
int sunxi_codec_resume(struct snd_soc_codec *snd_codec)
```

- 功能描述：模块唤醒（开启电源、开启时钟、恢复寄存器）
- 参数说明：
  - snd_codec: codec 驱动定义
- 返回值：0-成功，其它-失败

3.4.6.5 sunxi_codec_set_sysclk

- 函数原型：

```
int sunxi_codec_set_sysclk(struct snd_soc_dai *dai, int clk_id, unsigned int freq, int dir)
```

- 功能描述：设置模块 pllclk
- 参数说明：
  - dai: cpu dai 信息
  - pll_id: pll 辅助信息
  - source: pll 源
  - freq_in: 输入频率
  - freq_out: 输出频率
- 返回值：0-成功，其它-失败

3.4.6.6 sunxi_codec_startup

- 函数原型：

```
int sunxi_codec_startup(struct snd_pcm_substream *substream, struct snd_soc_dai *dai)
```

- 功能描述：设置模块开启工作资源 (dma 参数、组件功能等)
- 参数说明：
  - substream: pcm 子流信息
  - dai: cpu dai 信息
- 返回值：0-成功，其它-失败

3.4.6.7 sunxi_codec_hw_params

- 函数原型：

```
int sunxi_codec_hw_params(struct snd_pcm_substream *substream, struct snd_pcm_hw_params *
    params, struct snd_soc_dai *dai)
```

- 功能描述：设置模块硬件参数
- 参数说明：
  - substream: pcm 子流信息
  - params: 硬件参数
  - dai: cpu dai 信息
- 返回值：0-成功，其它-失败

3.4.6.8 sunxi_codec_prepare

- 函数原型：

```
int sunxi_codec_prepare(struct snd_pcm_substream *substream, struct snd_soc_dai *dai)
```

- 功能描述：清除模块 fifo
- 参数说明：
  - substream: pcm 子流信息
  - dai: cpu dai 信息
- 返回值：0-成功，其它-失败

3.4.6.9 sunxi_codec_trigger

- 函数原型：

```
int sunxi_codec_trigger(struct snd_pcm_substream *substream, int cmd, struct snd_soc_dai *
    dai)
```

- 功能描述：触发模块工作
- 参数说明：
  - substream: pcm 子流信息
  - cmd: 触发命令
  - dai: cpu dai 信息
- 返回值：0-成功，其它-失败

3.4.6.10 sunxi_codec_shutdown

- 函数原型：

```
void sunxi_codec_shutdown(struct snd_pcm_substream *substream, struct snd_soc_dai *dai)
```

- 功能描述：设置模块关闭工作资源 (组件功能等)
- 参数说明：
  - substream: pcm 子流信息
  - dai: cpu dai 信息
- 返回值：0-成功，其它-失败

3.4.7 machine 层接口

3.4.7.1 simple_soc_probe

- 函数原型：

```
int simple_soc_probe(struct snd_soc_card *card)
```

- 功能描述：初始化声卡相关信息（如 jack）
- 参数说明：
  - card: 声卡
- 返回值：0-成功，其它-失败

3.4.7.2 asoc_simple_card_parse_of

- 函数原型：

```
int asoc_simple_card_parse_of(struct device_node *node, struct asoc_simple_priv *priv)
```

- 功能描述：解析 codec 和 platform 节点，并绑定
- 参数说明：
  - node: machine 节点
  - priv: simple card 信息
- 返回值：0-成功，其它-失败

3.4.7.3 asoc_simple_parse_widgets

- 函数原型：

```
int asoc_simple_parse_widgets(struct snd_soc_card *card, char *prefix)
```

- 功能描述：解析 widget 部件
- 参数说明：
  - card: 声卡
  - prefix: 设备树属性名称前缀
- 返回值：0-成功，其它-失败

3.4.7.4 asoc_simple_parse_routing

- 函数原型：

```
int asoc_simple_parse_routing(struct snd_soc_card *card, char *prefix)
```

- 功能描述：解析 route 路径
- 参数说明：
  - card: 声卡
  - prefix: 设备树属性名称前缀
- 返回值：0-成功，其它-失败

3.4.7.5 asoc_simple_parse_pin_switches

- 函数原型：

```
int asoc_simple_parse_pin_switches(struct snd_soc_card *card, char *prefix)
```

- 功能描述：解析 dai 开关
- 参数说明：
  - card: 声卡
  - prefix: 设备树属性名称前缀
- 返回值：0-成功，其它-失败

3.4.7.6 asoc_simple_parse_daifmt

- 函数原型：

```
int asoc_simple_parse_daifmt(struct device_node *node, struct device_node *codec, char *
    prefix, unsigned int *retfmt)
```

- 功能描述：解析 I2S 格式
- 参数说明：
  - node: machine 节点
  - codec: codec 节点
  - prefix: 设备树属性名称前缀
  - retfmt: I2S 格式
- 返回值：0-成功，其它-失败

3.4.7.7 asoc_simple_parse_daistream

- 函数原型：

```
int asoc_simple_parse_daistream(struct device_node *node, char *prefix, struct
    snd_soc_dai_link *dai_link)
```

- 功能描述：解析音频流
- 参数说明：

- node: machine 节点
- prefix: 设备树属性名称前缀
- dai_link: dai 链接信息

- 返回值：0-成功，其它-失败

3.4.7.8 asoc_simple_parse_tdm_slot

- 函数原型：

```
int asoc_simple_parse_tdm_slot(struct device_node *node, char *prefix, struct
    asoc_simple_dai *dais)
```

- 功能描述：解析 I2S slot 个数和 slot 宽度
- 参数说明：
  - node: machine 节点
  - prefix: 设备树属性名称前缀
  - dais: I2S 信息
- 返回值：0-成功，其它-失败

3.4.7.9 asoc_simple_parse_tdm_clk

- 函数原型：

```
int asoc_simple_parse_tdm_clk(struct device_node *cpu, struct device_node *codec, char *
    prefix, struct simple_dai_props *dai_props)
```

- 功能描述：解析 I2S clk
- 参数说明：
  - node: cpu 节点
  - node: codec 节点
  - prefix: 设备树属性名称前缀
  - dai_props: dai 参数
- 返回值：0-成功，其它-失败

3.4.7.10 asoc_simple_set_dailink_name

- 函数原型：

```
int asoc_simple_set_dailink_name(struct device *dev, struct snd_soc_dai_link *dai_link,
    const char *fmt, ...)
```

- 功能描述：设置声卡名字
- 参数说明：
  - dai_link: dai 链接信息
  - fmt: cpu dai 和 codec dai 名
- 返回值：0-成功，其它-失败

3.4.7.11 asoc_simple_dai_init

- 函数原型：

```
int asoc_simple_dai_init(struct snd_soc_pcm_runtime *rtd)
```

- 功能描述：初始化 dai
- 参数说明：
  - rtd: dai 运行信息
- 返回值：0-成功，其它-失败

3.4.7.12 asoc_simple_hw_params

- 函数原型：

```
int asoc_simple_hw_params(struct snd_pcm_substream *substream, struct snd_pcm_hw_params *
    params)
```

- 功能描述：dai 硬件参数设置
- 参数说明：
  - substream: pcm 子流信息
  - params: 硬件参数
- 返回值：0-成功，其它-失败

3.5 软件调试接口

3.5.1 snd_sunxi_debug_show_reg

- 函数原型：

```
ssize_t snd_sunxi_debug_show_reg(struct device *dev, struct device_attribute *attr, char *
    buf)
```

- 功能描述：显示寄存器 dump 用法
- 参数说明：
  - attr: 设备属性
  - buf: printk buf
- 返回值：buf 大小

3.5.2 snd_sunxi_debug_store_reg

- 函数原型：

```
ssize_t snd_sunxi_debug_show_reg(struct device *dev, struct device_attribute *attr, char *
    buf)
```

- 功能描述：显示寄存器名称及值
- 参数说明：
  - attr: 设备属性
  - buf: printk buf
- 返回值：buf 大小

4 模块使用

一个声卡的简单测试使用，包含 3 部分，分别为声卡的加载、控件设置、测试工具。本章节将从以下 5 个通用小节和 1 个外挂 codec 小节介绍声卡如何使用。

1. kernel menuconfig 配置
2. 加载卸载方法
3. 声卡设备查看
4. 声卡控件
5. 声卡测试工具使用
6. I2S 外挂 codec

4.1 kernel menuconfig 配置

进入 menuconfig 配置命令：

```
#方式一：
#arm 32位平台，进入kernel目录执行以下命令
make menuconfig ARCH=arm

#arm 64位平台，进入kernel目录执行以下命令
make menuconfig ARCH=arm64

#方式二：
#进入longan目录执行以下命令
./build.sh menuconfig
```

具体配置选项，根据芯片平台、内核版本、所需音频模块，查看**“模块介绍”**章节的**“kernel menuconfig 配置”**说明。

4.2 加载卸载方法

具体配置选项，根据芯片平台、内核版本、所需音频模块，查看**“模块介绍”**章节的**“加载卸载方法（若编译为 ko）”**说明。

4.3 声卡设备查看

可输入以下命令查看系统挂载上的声卡

```
cat /proc/asound/cards
0 [audiocodec     ]: audiocodec - audiocodec
                     audiocodec
1 [sndspdif       ]: sndspdif - sndspdif
                     sndspdif
2 [snddmic        ]: snddmic - snddmic
                     snddmic
3 [snddaudio0     ]: snddaudio0 - snddaudio0
                     snddaudio0
4 [sndhdmi        ]: sndhdmi - sndhdmi
                     sndhdmi
5 [snddaudio2     ]: snddaudio2 - snddaudio2
                     snddaudio2
6 [ahubdam        ]: ahubdam - ahubdam
                     ahubdam
7 [ahubi2s0       ]: ahubi2s0 - ahubi2s0
                     ahubi2s0
8 [ahubhdmi       ]: ahubhdmi - ahubhdmi
                     ahubhdmi
9 [ahubi2s2       ]: ahubi2s2 - ahubi2s2
                     ahubi2s2
```

技巧

可通过修改 ***“soundcard-mach,name”*** 属性，设定声卡名称。

说明

该声卡列表仅为示范作用，部分声卡并非平台共有，取决于芯片规格。

查看更加详细的声卡信息如下（以 audiocodec 声卡为例）。

```
# 查看声卡号
cat /proc/asound/audiocodec/id
audiocodec

# 查看播放设备信息
cat /proc/asound/audiocodec/pcm0p/info
card: 0                 # 声卡0
device: 0               # 设备0
stream: PLAYBACK        # 播放流
...

# 查看录音设备信息
cat /proc/asound/audiocodec/pcm0c/info
card: 0
device: 0
stream: CAPTURE         # 录音流
...
```

查看播放设备硬件参数

```
cat /proc/asound/card0/pcm0p/sub0/hw_params
access: RW_INTERLEAVED
format: S16_LE          # 位深
subformat: STD
channels: 1             # 通道数
rate: 48000 (48000/1)   # 采样率
period_size: 1024
buffer_size: 4096
```

查看播放设备软件参数

```
cat /proc/asound/card0/pcm0p/sub0/sw_params
tstamp_mode: ENABLE
period_step: 1
avail_min: 1
start_threshold: 2048
stop_threshold: 4096
silence_threshold: 0
silence_size: 0
boundary: 4611686018427387904
```

查看播放设备状态

```
cat /proc/asound/card0/pcm0p/sub0/status
state: RUNNING
owner_pid   : 29385
trigger_time: 1477225.134078038
tstamp      : 1477265.465387349
delay       : 3520
avail       : 576
avail_max   : 1024
-----
hw_ptr      : 1935936
appl_ptr    : 1939456
```

4.4 声卡控件

具体配置选项，根据芯片平台、内核版本、所需音频模块，查看**“模块介绍”**章节的**“声卡控件”**说明。

4.5 声卡测试工具使用

4.5.1 tinyalsa 工具

tinyalsa 主要提供五个工具 1. tinymix: 可以得到音频通路相关的各项配置参数，也可通过传参设置参数 2. tinyplay: 是一个简易的音乐播放器，一般用于播放测试 3. tinycap: 是一个简易的录音软件，一般用于录音测试 4. tinypcminfo: 用于查看 pcm 通道的相关信息 5. tinyloop: 通

过软件的方式，将录制的声音实时播放

4.5.1.1 tinymix

```
tinymix -D cardx        /* 查看声卡x的控件列表 */
tinymix -D cardx y      /* 查看声卡x序号为y的控件的可选值 */
tinymix -D cardx y z    /* 设置声卡x序号为y的控件的值为z */
```

e.g. 查看声卡 0 的控件

```
/ # tinymix -D 0
Mixer name: 'audiocodec'
Number of controls: 8
ctl     type    num     name                                    value
0       ENUM    1       tx hub mode                             Off
1       INT     1       digital volume                          63
2       INT     1       lineout volume                          31
...
```

e.g. 查看声卡 0 的控件 “lineout volume” 可设置范围

```
/ # tinymix -D 0 2
LINEOUT volume: 31 (range 0->31)
```

e.g. 设置声卡 0 的控件 “lineout volume” 为 26

```
/ # tinymix -D 0 2 26
或
/ # tinymix -D 0  “lineout volume”  26
```

4.5.1.2 tinyplay

```
/* 播放wav文件，[]选项为可选项，不带则为默认值 */
tinyplay file.wav [-D card] [-d device] [-p period_size] [-n n_periods]
```

e.g. 用声卡 0 播放 test.wav

```
/ # tinyplay test.wav -D 0
Playing sample: 2 ch, 48000 hz, 16 bit 36678428 bytes
```

4.5.1.3 tinycap

```
/* 录音并保存数据到wav文件，[]选项为可选项，不带则为默认值 */
tinycap file.wav [-D card] [-d device] [-c channels] [-r rate] [-d bits] [-p period_size]
    [-n n_periods] [-T capture time]
```

e.g. 用声卡 3 录音并将数据保存到 test.wav

```
/ # tinycap test.wav -D 3
Capturing sample: 2 ch, 44100 hz, 16 bit
^CCaptured 131072 frames
```

4.5.1.4 tinypcminfo

```
/* 查看指定声卡、设备的pcm通道信息 */
tinypcminfo -D card -d device
```

e.g. 查看声卡 2 设备 0 的 pcm 通道信息

```
/ # tinypcminfo -D 2 -d 0
Info for card 2, device 0:

PCM out:
    Access:   0x000009
    Format[0]:   0x000444
    Format[1]:   00000000
    ...

PCM in:
    Access:   0x000009
    Format[0]:   0x000444
    Format[1]:   00000000
    ...
```

4.5.1.5 tinyloop

```
# 用指定的声卡、设备进行录音播放回路测试，[]选项为可选项，不带为默认值
tinyloop -PD playback card -Pd playback device -CD capture card -Cd capture device
[-p period_size] [-n n_periods] [-c num_channels] [-r sample_rate]
[-b format_bit] [-T playback/capture time]
```

e.g. 用声卡 0 设备 0 和声卡 3 设备 0 进行回路测试

```
/ # tinyloop -PD 0 -Pd 0 -CD 3 -Cd 0
Loopback: Playing device 0, Capture Device 0
Sample: 2 ch, 48000 hz, 16 bit
Duration in sec: forever
```

说明

不同版本 *tinyalsa* 工具，使用方法可能存在细微差别，可使用以下命令查看其对应版本的具体使用方法。

tinymix -h

tinyplay

tinycap

4.5.2 alsa-utils 工具

alsa-utils 主要提供三个工具：

1. aplay: 用于完成与播放相关的操作
2. arecord: 用于完成与录音相关的操作
3. amixer: 用于设置相关参数

4.5.2.1 aplay

```
# 输入 aplay 或 aplay -h 可打印出使用方法
/# aplay
Usage: aplay [OPTION]... [FILE]...

-h, --help              help
    --version           print current version
-l, --list-devices      list all soundcards and digital audio devices
-L, --list-pcms         list device names
-D, --device=NAME       select PCM by name
-q, --quiet             quiet mode
-t, --file-type TYPE    file type (voc, wav, raw or au)
-c, --channels=#        channels
-f, --format=FORMAT     sample format (case insensitive)
-r, --rate=#            sample rate
-d, --duration=#        interrupt after # seconds
...
```

e.g. 查看可以用于播放的声卡

```
/# aplay -l
**** List of PLAYBACK Hardware Devices ****
card 0: audiocodec [audiocodec], device 0: soc@03000000:codec_plat-sunxi-snd-codec sunxi-
    snd-codec-0 []
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 2: snddaudio0 [snddaudio0], device 0: 2032000.daudio0_plat-snd-soc-dummy-dai snd-soc-
    dummy-dai-0 []
  Subdevices: 1/1
  Subdevice #0: subdevice #0
```

e.g. 用声卡 0 设备 0 播放 test.wav(用 ctrl c 退出)

```
/# aplay -D hw:0,0 test.wav
Playing WAVE 'test.wav' : Signed 16 bit Little Endian, Rate 8000 Hz, Mono
^CAborted by signal Interrupt...
```

4.5.2.2 arecord

```
# 输入 arecord 或 arecord -h 可打印出使用方法
/# arecord
Usage: arecord [OPTION]... [FILE]...

-h, --help              help
    --version           print current version
-l, --list-devices      list all soundcards and digital audio devices
-L, --list-pcms         list device names
-D, --device=NAME       select PCM by name
-q, --quiet             quiet mode
-t, --file-type TYPE    file type (voc, wav, raw or au)
-c, --channels=#        channels
-f, --format=FORMAT     sample format (case insensitive)
-r, --rate=#            sample rate
-d, --duration=#        interrupt after # seconds
-M, --mmap              mmap stream
-N, --nonblock          nonblocking mode
-F, --period-time=#     distance between interrupts is # microseconds
-B, --buffer-time=#     buffer duration is # microseconds
...
```

e.g. 查看可以用于录音的声卡

```
/# arecord -l
**** List of CAPTURE Hardware Devices ****
card 0: audiocodec [audiocodec], device 0: soc@03000000:codec_plat-sunxi-snd-codec sunxi-
    snd-codec-0 []
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 1: snddmic [snddmic], device 0: 2031000.dmic_plat-snd-soc-dummy-dai snd-soc-dummy-dai
    -0 []
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 2: snddaudio0 [snddaudio0], device 0: 2032000.daudio0_plat-snd-soc-dummy-dai snd-soc-
    dummy-dai-0 []
  Subdevices: 1/1
  Subdevice #0: subdevice #0
...
```

e.g. 用声卡 1 的设备 0 进行采样位数为 16 的录音，并把数据保存在 test.wav(用 ctrl c 退出)

```
/# arecord -D hw:1,0 -f S16_LE test.wav
Recording WAVE 'test.wav' : Signed 16 bit Little Endian, Rate 8000 Hz, Mono
^CAborted by signal Interrupt...
```

4.5.2.3 amixer

```
# 输入 amixer 或 amixer -h 可打印出使用方法
/# amixer -h
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
  -R,--raw-volume Use the raw value (default)
  -M,--mapped-volume Use the mapped volume

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
```

e.g. 查看声卡 1 的控件

```
/# amixer -c 1 scontrols
Simple mixer control 'L0 volume',0
Simple mixer control 'L1 volume',0
Simple mixer control 'L2 volume',0
Simple mixer control 'L3 volume',0
Simple mixer control 'R0 volume',0
Simple mixer control 'R1 volume',0
Simple mixer control 'R2 volume',0
Simple mixer control 'R3 volume',0
Simple mixer control 'rx sync mode',0
```

e.g. 查看声卡 1 的控件的具体配置

```
/# amixer -c 1 scontents
Simple mixer control 'L0 volume',0
  Capabilities: volume volume-joined
  Playback channels: Mono
  Capture channels: Mono
  Limits: 0 - 255
  Mono: 176 [69%]
Simple mixer control 'L1 volume',0
  Capabilities: volume volume-joined
  Playback channels: Mono
  Capture channels: Mono
  Limits: 0 - 255
```

```
  Mono: 176 [69%]
Simple mixer control 'L2 volume',0
  Capabilities: volume volume-joined
  Playback channels: Mono
  Capture channels: Mono
  Limits: 0 - 255
  Mono: 176 [69%]
...
```

e.g. 设置声卡 1 第一个控件的值

```
# 拿到声卡1所有控件
/# amixer -c 1 controls
numid=2,iface=MIXER,name='L0 volume'
numid=4,iface=MIXER,name='L1 volume'
numid=6,iface=MIXER,name='L2 volume'
numid=8,iface=MIXER,name='L3 volume'
numid=3,iface=MIXER,name='R0 volume'
numid=5,iface=MIXER,name='R1 volume'
numid=7,iface=MIXER,name='R2 volume'
numid=9,iface=MIXER,name='R3 volume'
numid=1,iface=MIXER,name='rx sync mode'

# 拿到控件内容
/# amixer cget numid=2,iface=MIXER,name='L0 volume'
numid=2,iface=MIXER,name='rx sync mode'
  ; type=ENUMERATED,access=rw------,values=1,items=2
  ; Item #0 'Off'
  ; Item #1 'On'
  : values=0

# 设置控件值
/# amixer cset numid=2,iface=MIXER,name='L0 volume' 1
numid=2,iface=MIXER,name='rx sync mode'
  ; type=ENUMERATED,access=rw------,values=1,items=2
  ; Item #0 'Off'
  ; Item #1 'On'
  : values=1
```

4.6 I2S 外挂 codec

4.6.1 硬件连接

确保外部 codec 芯片与 soc I2S 接口正确连接，具体确认连接如下。

- LRCK, BCLK: 确认该两线是否连接；
- mclk: 确认外部 codec 是否需要 mclk，若需要，则确认 mclk 信号线连接；
- DIN: 确认外部 codec 是否需要录音功能，若需要，则确认 DIN 信号线连接；
- DOUT: 确认外部 codec 是否需要播放功能，若需要，则确认 DOUT 信号线连接。

4.6.2 获取外部 CODEC I2S 协议格式

确认外部 codec I2S 协议格式如下。

1. 功能需求：只录音、只播放、录音播放；
2. 引脚确认：I2S 序号、data 引脚序号；
3. 主从模式：SOC 做主 (由 SOC 提供 BCLK,LRCK)、外挂 codec 做主 (由外挂 codec 提供 BCLK,LRCK)；
4. I2S 模式：标准 I2S、I2S_L、I2S_R、DSP_A、DSP_B；
5. LRCK 信号是否翻转；
6. BCLK 信号是否翻转；
7. MCLK 信号：MCLK 频率；
8. slot 个数：最高要支持多少 slot（音频通道数）；
9. slot 宽度：最高要支持多少 slot 宽度（音频采样位深）。

查看 **“模块介绍”** 章节的 **“AHUB” 或 “I2S/PCM” -> “board.dts 板级配置”** 中的配置项说明，根据 I2S 协议格式进行配置。

5 FAQ

5.1 调试方法

5.1.1 调试工具

常用调试工具有 tinyalsa 和 alsa-utils 工具，具体使用和调试方法查看**“模块介绍”**章节的**“常见使用方法”**说明和**“模块使用-> 声卡测试工具使用”**章节。

5.1.2 调试节点

开启调试配置

```
Device Drivers  --->
    <*> Sound card support  --->
        <*> Advanced Linux Sound Architecture  --->
            <*> ALSA for SoC audio support  --->
                Allwinner SoC Audio support V2  --->
                    <M> Allwinner Function Components
                    <M>   Components Debug
```

将“Components Debug”选为 Y 或 M，使能调试节点加载。

寄存器 dump 搜索

```
cd /sys/devices/platform
find -name audio_reg
```

- 带 codec 路径为 Audiocodec 模块调试节点；
- 带 daudio 路径为 I2S/PCM 模块调试节点；
- 带 ahub 路径为 AHUB 模块调试节点；
- 带 spdif 路径为 S/PDIF 模块调试节点；
- 带 dmic 路径为 DMIC 模块调试节点。

查看寄存器值

```
cat audio_debug

# 根据节点提示说明查看相应寄存器值
```

```
# e.g.
echo 0 > audio_reg
SUNXI_DAC_DPC                   [0x000]: 0x       0 :0x0
SUNXI_DAC_FIFO_CTL              [0x010]: 0x    4000 :0x0
SUNXI_DAC_FIFO_STA              [0x014]: 0x  808008 :0x0
SUNXI_DAC_CNT                   [0x024]: 0x       0 :0x0
SUNXI_DAC_DG_REG                [0x028]: 0x       0 :0x0
AC_DAC_REG                      [0x310]: 0x  15141f :0x0
AC_MIXER_REG                    [0x314]: 0x     133 :0x0
AC_RAMP_REG                     [0x31c]: 0x       0 :0x0
```

查看寄存器显示结果分别为寄存器名称、寄存器偏移地址、寄存器值、休眠前寄存器值。

5.2 常见问题

5.2.1 录音或播放变速

【分析步骤一】：确认录音和播放采样率和父时钟 PLL_AUDIO 是否属于同一频段。

【分析步骤二】：以上无法定位，请联系 FAE 协助分析定位。

5.2.2 audiocodec 输入输出无声音

【分析步骤一】：确认通路设置。

通过 tinymix 查看 route 状态，通过 debugfs 查看 dapm 状态，是否设置了需要的上下电通路。

【分析步骤二】：对于喇叭，确认功放芯片使能设置。

查看设备树 codec 节点中 pa-pin-n 的 gpio 配置和硬件原理图比对，是否适配了对应的 gpio。

【分析步骤三】：以上无法定位，请联系 FAE 协助分析定位。

5.2.3 DMIC 录音异常（静音/通道移位）

【分析步骤一】：确认 GPIO 是否正常。

1. 通过 datasheet 核对 board.dts 部分的 dmic pin 设置；
2. 通过 sunxi_dump 来打印出 dmic 的 gpio 设置是否正常（dump 寄存器的时候请在 DMIC 正在录音的时候）。

【分析步骤二】：确认 clk 的频率。

以上正常情况下，示波器查看 dmic clk 的频率是否满足如下关系。

```
clk = sample * over_sample_rate
```

【分析步骤三】：排查硬件连接和 dmic 物料问题。

【分析步骤四】：以上无法定位，请联系 FAE 协助分析定位。

著作权声明

商标声明

免责声明