



D1 Tina Linux 音频 开发指南

**版本号: 1.1
发布日期: 2021.05.15**

版本历史

版本号	日期	制/修订人	内容描述
1.0	2021.04.09	AW0985	初始版本。
1.1	2021.05.15	AW0985	完善部分描述。



目 录

1 概述	1
1.1 编写目的	1
1.2 适用范围	1
1.3 相关人员	1
1.4 相关术语	1
2 模块介绍	3
2.1 驱动框架	3
2.2 D1 音频接口	4
2.2.1 时钟源	4
2.2.2 代码结构	5
2.2.3 AudioCodec	6
2.2.3.1 内核配置	6
2.2.3.2 DTS 配置	6
2.2.3.3 codec 数据通路	8
2.2.4 Daudio	10
2.2.4.1 内核配置	11
2.2.4.2 DTS 配置	11
2.2.4.3 I2S 注意事项	13
2.2.5 DMIC	13
2.2.5.1 内核配置	14
2.2.5.2 DTS 配置	14
2.2.6 SPDIF	15
2.2.6.1 内核配置	15
2.2.6.2 DTS 配置	15
2.2.7 外挂 codec:AC107	16
2.2.7.1 内核配置	16
2.2.7.2 DTS 配置	16
2.2.7.3 使用	17
2.2.8 标案音频测试方法	18
2.2.8.1 播放	18
2.2.8.2 录音	18
3 常用工具及调试方法	19
3.1 alsa-utils	19
3.1.1 amixer	19
3.1.2 aplay	20
3.1.3 arecord	20
3.1.4 alsacnf	21
3.2 tinyalsa-utils	24
3.2.1 tinymix	24
3.2.2 tinyplay	25

3.2.3 tinycap	25
3.3 dump 寄存器	26
3.3.1 dump audiocodec 寄存器	26
3.3.2 dump daudio 寄存器	28
3.3.3 dump dmic 寄存器	28
3.3.4 dump spdif 寄存器	28
3.4 sound procfs	29
4 常用接口说明	31
4.1 control 接口	31
4.2 PCM 接口	33
5 调试注意事项	34
5.1 声卡没有加载	34
5.2 播放没有声音	34



1 概述

1.1 编写目的

介绍 D1 Tina 平台音频模块的使用方法。

1.2 适用范围

Allwinner 软件平台 Tina。

Allwinner 硬件平台 D1 芯片。

1.3 相关人员

Tina 平台下进行音频模块开发的工程师。

1.4 相关术语

术语	解释说明
ALSA	Advanced Linux Sound Architecture
DMA	直接内存存取, 指数据不经 cpu, 直接在设备和内存, 内存和内存, 设备和设备之间传输
ASoC	ALSA System on Chip
样本长度 sample	样本是记录音频数据最基本的单位, 常使用 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

术语	解释说明
交错模式 interleaved	是一种音频数据的记录模式，在交错模式下，数据以连续帧的形式存放，即首先记录完帧 1 的左声道样本和右声道样本（假设为立体声格式），再开始帧 2 的记录，而在非交错模式下，首先记录的是一个周期内所有帧的左声道样本，再记录右声道样本，数据是以连续通道的方式存储。不过多数情况下，我们只需要使用交错模式就可以了
AudioCodec	芯片内置音频接口
Daudio	数字音频接口，可配置成 I2S/PCM 标准音频接口
Dmic	数字麦接口
AGC	Automatic Gain Control
DRC	Dynamic Range Control



2 模块介绍

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

2.1 驱动框架

Tina SDK 对各个平台的音频设备驱动均采用 ASoC 架构实现。ASoC 是建立在标准 alsa 驱动层上，为了更好地支持嵌入式处理器和移动设备中的音频 codec 的一套软件体系，ASoC 将音频系统分为 3 部分：Codec，Platform 和 Machine。

1. Codec 驱动

ASoC 中的一个重要设计原则就是要求 Codec 驱动是平台无关的，它包含了一些音频的控件 (Controls)，音频接口，DAMP(动态音频电源管理) 的定义和某些 Codec IO 功能。为了保证硬件无关性，任何特定于平台和机器的代码都要移到 Platform 和 Machine 驱动中。

所有的 Codec 驱动都要提供以下特性：

- Codec DAI (Digital Audio Interface) 和 PCM 的配置信息；
- Codec 的 IO 控制方式 (I2C, SPI 等)；
- Mixer 和其他的音频控件；
- Codec 和 ALSA 音频操作接口；

2. Platform 驱动

它包含了该 SoC 平台的音频 DMA 和音频接口的配置和控制 (I2S, PCM, AC97 等等)；一般不包含与板子或 codec 相关的代码。

3. Machine 驱动

单独的 Platform 和 Codec 驱动是不能工作的，它必须由 Machine 驱动把它们结合在一起才能完成整个设备的音频处理工作。

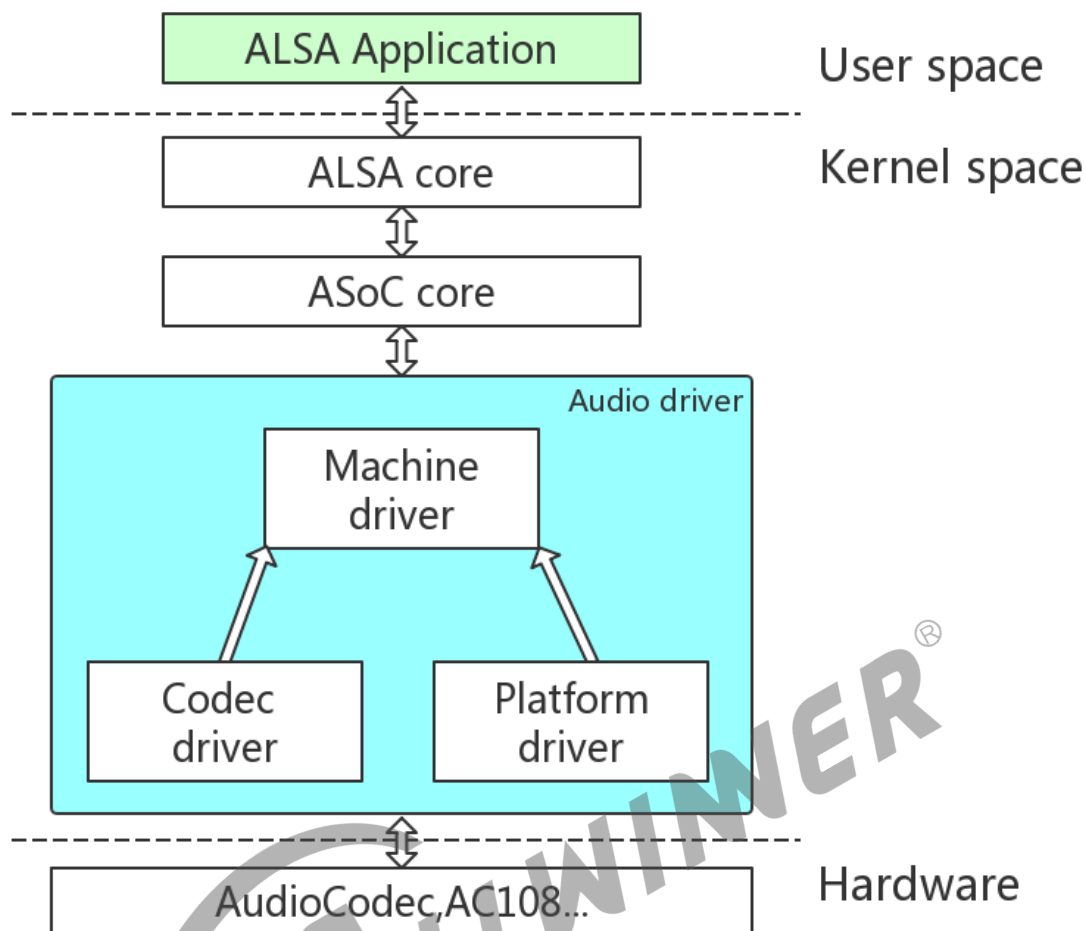


图 2-1: ASoC 框图

2.2 D1 音频接口

D1 包含多个音频模块，分别是内置 AudioCodec, I2S0, I2S1, I2S2(与 HDMI Audio 输出), DMIC, SPDIF。

2.2.1 时钟源

D1 音频模块的时钟源来自 pll_audio0 以及 pll_audio1_div5。

pll_audio0 可以输出 22.5792M 的时钟，而 pll_audio1_div5 输出 24.576M 的时钟，分别支持 44.1k 系列，48k 系列的播放录音。

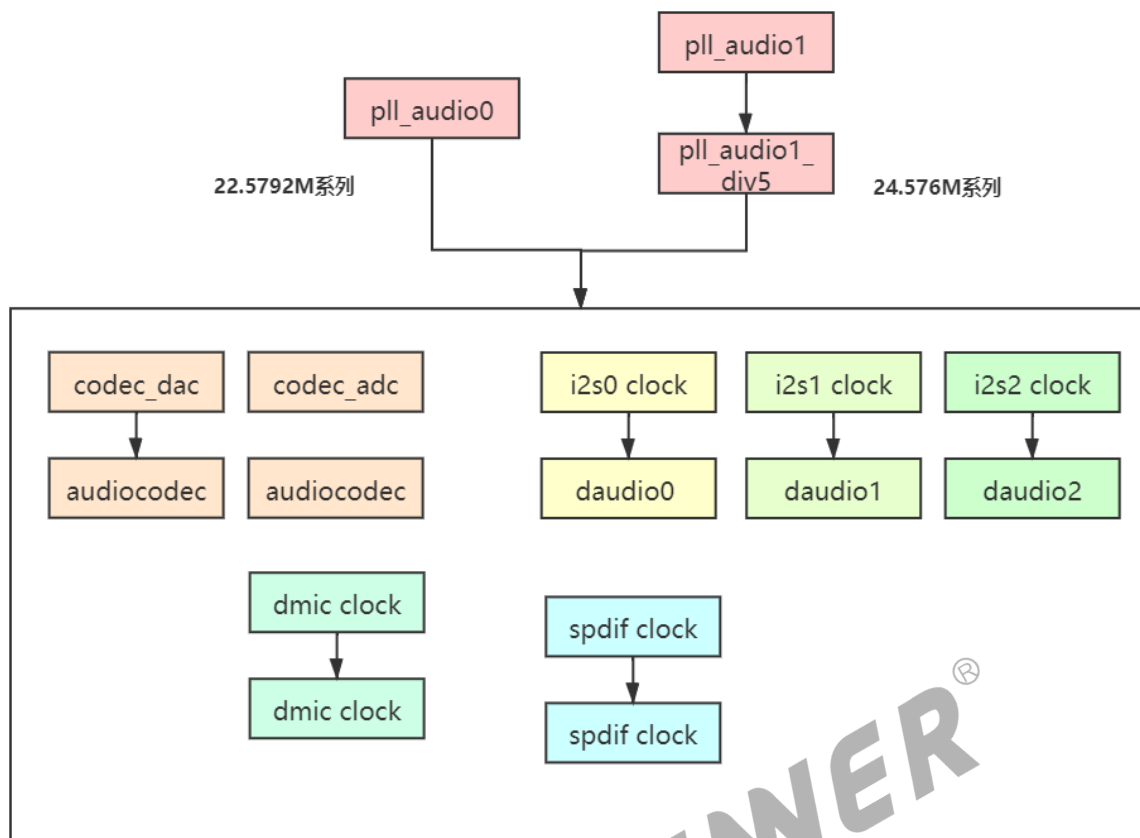


图 2-2: D1 时钟源

2.2.2 代码结构

```

linux-5.4/sound/soc/sunxi/
├── sun20iw1-codec.c           // codec驱动
├── sun20iw1-codec.h
├── sun20iw1-sndcodec.c       // codec machine驱动
├── sunxi-dummy-cpudai.c     // codec platform驱动
├── sunxi-daudio.c           // daudio platform驱动
├── sunxi-daudio.h
├── sunxi-simple-card.c      // 通用machine驱动
├── sunxi-dmic.c             // dmick platform驱动
├── sunxi-dmic.h
├── sunxi-pcm.c              // 通用文件，提供注册platform驱动接口及相关函数集
├── sunxi-pcm.h
├── sunxi-spdif.c            // spdif platform驱动
├── sunxi-spdif.h

linux-5.4/sound/soc/codecs/dmic.c // dmick codec驱动
linux-5.4/sound/soc/soc-utils.c   // daudio codec驱动(snd-soc-dummy)
  
```

2.2.3 AudioCodec

硬件特性

- 两路 DAC
 - 支持 16bit,20bit 有效采样精度
 - 支持 8KHz~192KHz 采样率
- 三路 ADC
 - 支持 16bit,20bit 有效采样精度
 - 支持 8KHz~48KHz 采样率
- 两路模拟输出：
 - 一路立体声输出 HPOUTL,HPOUTR
 - 一路立体声差分输出 LINEOUTLP/N,LINEOUTRP/N
- 五路模拟差分输入：MIC1P/N,MIC2P/N
 - 三路差分麦克风输入 MIC1P/N,MIC2P/N,MIC3P/N
 - 一路立体声 line-in 输入 LINEINL,LINEINR
 - 一路立体声 FM-in 输入 FMINL,FMINR
- 支持耳机驱动电路
- 支持同时 playback 和 record(全双工模式)
- DAC 及 ADC 均支持 DRC

2.2.3.1 内核配置

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

2.2.3.2 DTS 配置

```
&codec {
    /* MIC and headphone gain setting */
    mic1gain      = <0x1F>;
    mic2gain      = <0x1F>;
    mic3gain      = <0x1F>;
    /* 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-spkr      = <&pio PF 2 GPIO_ACTIVE_HIGH>; */
    /* gpio-spkr-pwr  = <&pio PF 4 GPIO_ACTIVE_HIGH>; */
    /* regulator about */
    /* avcc-supply    = <&reg_aldol>; */
    /* hpvcc-supply   = <&reg_eldol>; */
    status = "okay";
};

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

&dummy_cpudai {
    /* CMA config about */
    playback_cma     = <128>;
    capture_cma      = <256>;
    status = "okay";
};

```

sndcodec 配置，即 machine 驱动的相关配置

sndcodec 配置	sndcodec 配置说明
status	是否使用 sndcodec 驱动。disabled：不使用；okay：使用
hp_detect_case	耳机检测电平，0：低电平；1：高电平
jack_enable	是否初始化耳机相关代码。0：不使用；1：使用

dummy_cpudai 配置，即 platform 驱动的相关配置

dummy_cpudai 配置	dummy_cpudai 配置说明
playback_cma	配置 playback 的 dma buffer 大小，单位 KB
capture_cma	配置 capture 的 dma buffer 大小，单位 KB
status	是否使用 cpudai 驱动。disabled：不使用；okay：使用

codec 配置，即内置 audiocodec 驱动的相关配置

codec 配置	codec 配置说明
status	是否使用 codec 驱动。disabled: 不使用; okay: 使用
mic1gain	mic1 增益, 可设定范围 0~0x1f, 0:0dB, 1~0x3:6dB, 0x4~0x1f:9~36dB, 1dB/step, 一般设置 0x13, 即 24dB
mic2gain	mic2 增益, 可设定范围 0~0x1f, 0:0dB, 1~0x3:6dB, 0x4~0x1f:9~36dB, 1dB/step 一般设置 0x13, 即 24dB
mic3gain	mic3 增益, 可设定范围 0~0x1f, 0:0dB, 1~0x3:6dB, 0x4~0x1f:9~36dB, 1dB/step, 一般设置 0x13, 即 24dB
adchpf_cfg	是否使用 ADC HPF 功能, 1: 使用; 0: 不使用
digital_vol	初始化 digital volume, 可设定范围 0~0x3f, 表示 0~-73.08dB, -1.16dB/step
lineout_vol	lineout volume, 可设定范围 0~0x1f, 表示-43.5dB~0dB, 1.5dB/step
headphonegain	headphone 增益, 可设定范围 0~0x07, 表示 0dB~-42dB, -6dB/step
pa_level	PA 引脚使能方式。0: 低电平有效; 1: 高电平有效
pa_pwr_level	PA 供电引脚使能方式。0: 低电平有效; 1: 高电平有效
gpio-spk	PA 使能引脚
gpio-spk-pwr	PA 供电使能引脚

2.2.3.3 codec 数据通路

通过Lineout播歌

Playback --> DACL --> LINEOUTL Output Select --> LINEOUTL --> LINEOUT
 Playback --> DACR --> LINEOUTR Output Select --> LINEOUTR --> LINEOUT

通过HPOUT播歌 (D1开发板默认使用耳机播放)

Playback --> DACL --> HPOUTL --> Headphone
 Playback --> DACR --> HPOUTR --> Headphone

如果HPOUT输出到模拟功放:

Playback --> DACL --> HPOUTL --> HpSpeaker
 Playback --> DACR --> HPOUTR --> HpSpeaker

录音(3通道)

MIC1 --> MIC1 Input Select --> ADC1 Input --> ADC1 --> Capture
 MIC2 --> MIC2 Input Select --> ADC2 Input --> ADC2 --> Capture
 MIC3 --> MIC3 Input Select --> ADC3 Input --> ADC3 --> Capture

耳机录音 (D1开发板默认使用耳麦)

MIC3 --> MIC3 Input Select --> ADC3 Input --> ADC3 --> Capture

录制回采 (D1开发板硬件上可以通过LINEINL/R录制回采信号)

```
LINEINL --> ADC1 Input --> ADC1 --> Capture
LINEINR --> ADC2 Input --> ADC2 --> Capture
```

LINE-in录音

```
LINEINL --> ADC1 Input --> ADC1 --> Capture
LINEINR --> ADC2 Input --> ADC2 --> Capture
```

FM-in录音

```
FMINL --> ADC1 Input --> ADC1 --> Capture
FMINR --> ADC2 Input --> ADC2 --> Capture
```

D1 所有控件如下表：

控件名称	功能	数值
MIC1 gain volume	MIC1 增益	0~31, 表示 0~36dB, 0:0dB, 1~3:6dB, 4~31:9~36dB, 1dB/step
MIC2 gain volume	MIC2 增益	0~31, 表示 0~36dB, 0:0dB, 1~3:6dB, 4~31:9~36dB, 1dB/step
MIC3 gain volume	MIC3 增益	0~31, 表示 0~36dB, 0:0dB, 1~3:6dB, 4~31:9~36dB, 1dB/step
FMINL gain volume	FMINL 增益	0:0dB; 1:6dB
FMINR gain volume	FMINR 增益	0:0dB; 1:6dB
LINEINL gain volume	LINEINL 增益	0:0dB; 1:6dB
LINEINR gain volume	LINEINR 增益	0:0dB; 1:6dB
MIC1 Input Select	MIC1 输入模式	0: 差分输入; 1: 单端输入
MIC2 Input Select	MIC2 输入模式	0: 差分输入; 1: 单端输入
MIC3 Input Select	MIC3 输入模式	0: 差分输入; 1: 单端输入
ADC1 volume	ADC1 数字音量设置	0~0xFF, 0:Mute; 1~0xFF:- 119.25dB~71.24dB, 0.75dB/step, 默认 0xA0=0dB
ADC2 volume	ADC2 数字音量设置	0~0xFF, 0:Mute; 1~0xFF:- 119.25dB~71.24dB, 0.75dB/step, 默认 0xA0=0dB
ADC3 volume	ADC3 数字音量设置	0~0xFF, 0:Mute; 1~0xFF:- 119.25dB~71.24dB, 0.75dB/step, 默认 0xA0=0dB

控件名称	功能	数值
ADC1 Input MIC1 Boost Switch	是否使能 ADC1->MIC1 的通路	0: 关闭; 1: 使能
ADC2 Input MIC2 Boost Switch	是否使能 ADC2->MIC2 的通路	0: 关闭; 1: 使能
ADC3 Input MIC3 Boost Switch	是否使能 ADC3->MIC3 的通路	0: 关闭; 1: 使能
ADC1 Input FMINL Switch	是否使能 ADC1->FMINL 的通路	0: 关闭; 1: 使能
ADC2 Input FMINR Switch	是否使能 ADC2->FMINR 的通路	0: 关闭; 1: 使能
ADC1 Input LINEINL Switch	是否使能 ADC1->LINEINL 的通路	0: 关闭; 1: 使能
ADC2 Input LINEINR Switch	是否使能 ADC2->LINEINR 的通路	0: 关闭; 1: 使能
LINEOUT volume	Lineout 音量设置	0~31, 表示-43.5~0dB
digital volume	数字端音量设置	0~63, 表示-73.08~0dB
DAC volume	DACL,DACR 音量设置	0~0xFF, 0:Mute; 1~0xFF:- 119.25dB~71.24dB, 0.75dB/step, 默认 0xA0=0dB
Headphone Volume	Headphone 音量设置	0~7, 表示-0dB~-42dB, -6dB/step
LINEOUTL Output Select	Lineout left 输出选择	0: 单端; 1: 差分
LINEOUTR Output Select	Lineout right 输出选择	0: 单端; 1: 差分
LINEOUT Switch	是否使能 Lineout 通路	0: 关闭; 1: 使能
Headphone Switch	是否使能 Headphone 通路	0: 关闭; 1: 使能
HpSpeaker Switch	是否使能 Speaker 通路 (使用功放)	0: 关闭; 1: 使能

2.2.4 Daudio

硬件特性

- 四路 I2S/PCM, 可用于蓝牙通话, 语音采集, 数字功放;
- 支持主从模式
- 支持 Left-justified, Right-justified, Standar mode I2S, PCM mode

- 支持 i2s,pcm 协议格式配置
- 支持 mono 和 stereo 模式，最高支持 8 通道
- 支持同时 playback 和 record(全双工模式)
- 支持 8~192KHz 采样率
- 支持 16,24,32bit 采样精度
- 支持 3 路 MCLK 输出

2.2.4.1 内核配置

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

2.2.4.2 DTS 配置

```
&audio0 {
    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     = <&audio0_pins_a>;
    pinctrl-1     = <&audio0_pins_b>;
    pinctrl_used   = <0x0>;
    status = "disabled";
};

&sndaudio0 {
    status = "disabled";
    simple-audio-card,name = "sndaudio0";
    /* simple-audio-card,frame-master = <&audio0_master>; */
    /* simple-audio-card,bitclock-master = <&audio0_master>; */
    /* simple-audio-card,bitclock-inversion; */
    /* simple-audio-card,frame-inversion; */
    audio0_master: simple-audio-card,codec {
        /* sound-dai = <&acl08>; */
    };
};
```

audio0 配置，即 audio0 platform 驱动的相关配置

daudio 配置	daudio 配置说明
mclk_div	0: not output(normal setting this); 1/2/4/6/8/12/16/24/32/48/64/96/128/176/192: 给外部 codec 提供时钟, 频率是 pll_audio/mclk_div
frametype	0: short frame = 1 clock width; 1: long frame = 2 clock width
tdm_config	0: pcm mode; 1: i2s mode
sign_extend	0: zero pending; 1: sign extend
msb_lsb_first	0: msb first; 1: lsb first
pcm_lrck_period	一般可配置 16/32/64/128/256 个 bclk
slot_width_select	支持 8bit, 16bit, 32bit 宽度
tx_data_mode	0: 16bit linear PCM;1: reserved;2: 8bit u-law;3: 8bit a-law
rx_data_mode	0: 16bit linear PCM;1: reserved;2: 8bit u-law;3: 8bit a-law
playback_cma	配置 playback 的 dma buffer 大小, 单位 KB
capture_cma	配置 capture 的 dma buffer 大小, 单位 KB

soundaudio0 配置, 即 daudio0 machine 驱动的相关配置

soundaudio 配置	soundaudio 配置说明
status	是否使用 sndaudio 驱动。disabled: 不使用; okay: 使用
simple-audio-card,name	声卡名称
simple-audio-card,format	i2s,right_j,left_j,dsp_a,dsp_b
simple-audio-card,frame-master	配置 frame clk 主从关系, 不配置则是 SoC 作为主,codec 作为 slave; 如果配置了 codec 节点, 则 codec 作为主, SoC 作为从
simple-audio-card,bitclock-master	配置 bit clk 主从关系, 不配置则是 SoC 作为主,codec 作为 slave; 如果配置了 codec 节点, 则 codec 作为主, SoC 作为从
simple-audio-card,bitclock-inversion	配置 bit clk 极性取反; 不配置则是正常极性
simple-audio-card,frame-inversion	配置 frame clk 极性取反; 不配置则是正常极性
simple-audio-card,capture_only	仅支持录音流
simple-audio-card,playback_only	仅支持播放流

2.2.4.3 I2S 注意事项

HDMI audio 功能只能使用 daudio2 这一组 I2S, 默认 DTS 也已经配置好, 主要配置如下:

```
&daudio2 {
    mclk_div      = <0x00>;
    frametype     = <0x00>;
    tdm_config    = <0x01>;
    sign_extend   = <0x00>;
    tx_data_mode  = <0x00>;
    rx_data_mode  = <0x00>;
    msb_lsb_first = <0x00>;
    pcm_lrck_period = <0x20>;
    slot_width_select = <0x20>;
    asrc_function_en = <0x00>;
    pinctrl-names = "default", "sleep";
    pinctrl-0     = <&daudio2_pins_a &daudio2_pins_b &daudio2_pins_c>;
    pinctrl-1     = <&daudio2_pins_d>;
    pinctrl_used   = <0x0>;
    daudio_type    = <0x1>;
    status = "okay";
};
&hdmaudio {
    status = "okay";
};
&sounddaudio2 {
    status = "okay";
    simple-audio-card,name = "sndhdm1";
    daudio2_master: simple-audio-card,codec {
        sound-dai = <&hdmaudio>;
    };
};
```

其中需要注意的点: slot_width_select 设置 0x20 tdm_config 设置 0x1 pinctrl_used 设置 0 daudio_type 设置 1

进入系统后, 通过命令 `cat /proc/asound/cards` 列出当前声卡信息, 如果发现 `sndhdm1` 声卡, 说明已经正常加载驱动

无需额外设置音频通路, 可直接用下面命令进行播放:

```
推送wav音频文件到小机端, 例如/mnt/UDISK/test.wav
aplay -Dhw:sndhdm1 /mnt/UDISK/test.wav
```

```
默认/etc/asound.conf包含了HDMI audio的配置, 也可以用下面命令播放:
aplay -DPlaybackHDMI /mnt/UDISK/test.wav
```

2.2.5 DMIC

硬件特性

- 支持 8 路输入

- 支持 8~48KHz 采样率
- 支持 16/24bit 采样精度

2.2.5.1 内核配置

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

2.2.5.2 DTS 配置

配置如下:

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

&dmic_codec {
    status = "okay";
};

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

dmic 配置, 即 platform 驱动的相关配置

dmic 配置	dmic 配置说明
status	是否使用 dmic 驱动。disabled: 不使用; okay: 使用
capture_cma	配置 capture 的 dma buffer 大小, 单位 KB
data_vol	DATA0~DATA3 的数字增益, 默认配置 0xB0 即 12dB
rx_chmap	通道映射, 默认配置 0x76543210 表示按照默认通道顺序

dmic_codec 配置, 即 codec 驱动的相关配置

dmic_codec 配置	dmic_codec 配置说明
status	是否使用 dmic_codec 驱动。disabled: 不使用; okay: 使用

sounddmic 配置，即 machine 驱动的相关配置

sounddmic 配置	sounddmic 配置说明
status	是否使用 sounddmic 驱动。disabled：不使用； okay：使用
simple-audio-card,name	声卡名称
simple-audio-card,capture_only	仅支持录音流

2.2.6 SPDIF

硬件特性

- 支持 S/PDIF_OUT 和 S/PDIF_IN
- 支持 mono 和 stereo 模式
- 输出支持 22.05kHz, 24kHz, 32kHz, 44.1kHz, 48kHz, 88.2kHz, 96kHz, 176.4kHz, 192kHz 采样率
- 输入支持 44.1KHz, 48KHz 采样率
- 输出和输入支持 16bit, 24bit 采样精度

2.2.6.1 内核配置

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

2.2.6.2 DTS 配置

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

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

spdif 配置，即 platform 驱动的相关配置

spdif 配置	spdif 配置说明
status	是否使用 spdif 驱动。disabled：不使用；okay：使用

soundspdif 配置，即 machine 驱动的相关配置

soundspdif 配置	soundspdif 配置说明
status	是否使用 sndspdif 驱动。disabled：不使用；okay：使用

2.2.7 外挂 codec:AC107

AC107 是一颗含有两个 ADC 的 Codec 芯片，下面对 D1 如何配置使用 AC107 作简单介绍, 如果使用其他的 Codec 芯片，也可作配置参考。

2.2.7.1 内核配置

```
Device Drivers --->
<*> Sound card support --->
  <*> Advanced Linux Sound Architecture --->
    <*> ALSA for SoC audio support --->
      Allwinner SoC Audio support --->
        <*> Allwinner Audio Simple Card
        <*> Allwinner Digital Audio Support
      CODEC drivers --->
        <*> Sunxi AC107 Codec
```

2.2.7.2 DTS 配置

假设 D1 是通过 twi0 控制 AC107, 而 i2s2 用于音频数据的传输

twi 配置:

```
&twi0 {
    clock-frequency = <400000>;
    pinctrl-0 = <&twi0_pins_a>;
    pinctrl-1 = <&twi0_pins_b>;
    pinctrl-names = "default", "sleep";
    status = "okay";

    ac107: ac107@36 {
        #sound-dai-cells = <0>;
        compatible = "Allwinnertech,ac107_0";
        /*compatible = "ac107_0";*/
        reg = <0x36>;
    }
}
```

```
        status = "okay";
    };
};
```

i2s 配置:

```
&audio2 {
    mclk_div        = <0x02>;
    frametype       = <0x00>;
    tdm_config      = <0x01>;
    sign_extend     = <0x00>;
    tx_data_mode    = <0x00>;
    rx_data_mode    = <0x00>;
    msb_lsb_first   = <0x00>;
    pcm_lrck_period = <0x80>;
    slot_width_select = <0x20>;
    asrc_function_en = <0x00>;
    pinctrl-names   = "default", "sleep";
    pinctrl-0       = <&audio2_pins_a &audio2_pins_b &audio2_pins_c>;
    pinctrl-1       = <&audio2_pins_d>;
    pinctrl_used    = <0x1>;
    audio_type      = <0x0>;
    status = "okay";
};

&soundaudio2 {
    status = "okay";
    simple-audio-card,name = "ac107";
    simple-audio-card,format = "i2s";
    simple-audio-card,capture_only;
    audio2_master: simple-audio-card,codec {
        sound-dai = <&ac107>;
    };
};
```

其中注意配置mclk_div=2,ac107要求MCLK为11.288M或者11.2896M;

使用标准i2s格式;

AC107作为从设备

BCLK,LRCK都是normal模式,即不用配置bitclock-inversion,frame-inversion

simple-audio-card,capture_only表示只注册录音流,因为ac107声卡仅支持录音流

simple-audio-card,name声卡名字是ac107

2.2.7.3 使用

进入系统后, 通过命令 `cat /proc/asound/cards` 列出当前声卡信息, 如果发现 ac107 相关声卡, 说明已经正常加载驱动

无需额外设置音频通路, 可直接用下面命令进行录音:

```
arecord -Dhw:ac107 -f S16_LE -r 16000 -c 2 /tmp/test.wav
```

另外可以通过下面命令调整增益

```
amixer -D hw:ac107 cset name='Channel 1 PGA Gain' 25
```

```
amixer -D hw:ac107 cset name='Channel 2 PGA Gain' 25
```

2.2.8 标案音频测试方法

该章节主要介绍在标案上进行播歌，录音的测试命令

2.2.8.1 播放

通过Headphone播放

```
amixer -D hw:audiocodec cset name='Headphone Switch' 1  
amixer -D hw:audiocodec cset name='Headphone Volume' 3
```

```
aplay -Dhw:audiocodec /mnt/UDISK/1KHz_0dB_16000.wav
```

或者利用默认/etc/asound.conf配置的pcm设备进行播放：

```
aplay -Ddefault /mnt/UDISK/1KHz_0dB_16000.wav
```

2.2.8.2 录音

通过耳麦(MIC3)录制单声道

```
amixer -D hw:audiocodec cset name='ADC3 Input MIC3 Boost Switch' 1  
amixer -D hw:audiocodec cset name='MIC3 Input Select' 0  
amixer -D hw:audiocodec cset name='MIC3 gain volume' 19
```

```
arecord -Dhw:audiocodec -f S16_LE -r 16000 -c 1 /tmp/test.wav
```

注意不要使能MIC1,MIC2的音频通路，否则录制的数据会有问题。

如果MIC1,MIC2,MIC3的通路都使能了，那么可以用下面方式是进行录音：

```
arecord -DCaptureMic -f S16_LE -r 16000 -c 1 /tmp/test.wav
```

默认在/etc/asound.conf中配置了CaptureMic，硬件上以3声道录音，但是arecord只拿MIC3这一声道数据

3 常用工具及调试方法

3.1 alsa-utils

标准 ALSA 工具, 它使用到 alsa-lib 标准库, 一般常用到的有 amixer, aplay, arecord 等。

3.1.1 amixer

amixer 是命令行的 ALSA 声卡驱动调节器工具, 用于设置 mixer control。

使用方法:

- 常用选项

选项	功能
-D, -device	指定声卡设备, 默认使用 default

- 常用命令

命令	功能
controls	列出指定声卡的所有控件
contents	列出指定声卡的所有控件的具体信息
cget	获取指定控件的信息
cset	设定指定控件的值

举例:

```
获取audiocodec声卡的所有控件名
amixer -Dhw:audiocodec controls

获取当前硬件音量
amixer -Dhw:audiocodec cget name='LINEOUT volume'

设置当前硬件音量
amixer -Dhw:audiocodec cset name='LINEOUT volume' 25
```

3.1.2 aplay

aplay 是命令行的 ALSA 声卡驱动的播放工具，用于播放功能。

使用方法：

选项	功能
-D,-device	指定声卡设备, 默认使用 default
-l,-list-devices	列出当前所有声卡
-t,-file-type	指定播放文件的格式, 如 voc,wav,raw, 不指定的情况下会去读取文件头部作识别
-c,-channels	指定通道数
-f,-format	指定采样格式
-r,-rate	采样率
-d,-duration	指定播放的时间
-period-size	指定 period size
-buffer-size	指定 buffer size

如果播放的是 wav 文件，可以解析头部，识别通道数，采样率等参数。

举例：

```
aplay -Dhw:audiocodec /mnt/UDISK/test.wav
```

3.1.3 arecord

arecord 是命令行的 ALSA 声卡驱动的录音工具，用于录音功能。

使用方法：

选项	功能
-D,-device	指定声卡设备, 默认使用 default
-l,-list-devices	列出当前所有声卡
-t,-file-type	指定播放文件的格式, 如 voc,wav,raw, 不指定的情况下会去读取文件头部作识别
-c,-channels	指定通道数
-f,-format	指定采样格式
-r,-rate	采样率
-d,-duration	指定播放的时间
-period-size	指定 period size
-buffer-size	指定 buffer size

举例：

```
录制5s,通道数为2, 采样率为16000, 采样精度为16bit, 保存为wav文件  
arecord -Dhw:audiocodec -f S16_LE -r 16000 -c 2 -d 5 /mnt/UDISK/test.wav
```

3.1.4 alsacnf

alsacnf 指的是 ALSA configuration file, 使用 alsa-lib 打开声卡, 操作 pcm, mixer 时, 会加载相关位置上的配置文件, 用于指导操作 pcm,mixer 设备。

首先会读取配置文件/usr/share/alsa/alsa.conf, 其中有下面一段 hooks。

```
@hooks [  
  {  
    func load  
    files [  
      {  
        @func concat  
        strings [  
          { @func datadir }  
            "/alsa.conf.d/"  
        ]  
      }  
      "/etc/asound.conf"  
      "~/.asoundrc"  
    ]  
    errors false  
  }  
]
```

这里设定了一个钩子, 去读取相关目录配置文件:

```
/usr/share/alsa/alsa.conf.d/  
/etc/asound.conf  
~/.asoundrc
```

这些配置文件可以设定 default 声卡, 自定义 pcm 设备,alsa 插件等功能, 具体可以参考:

<https://www.alsa-project.org/alsa-doc/alsa-lib/conf.html>

https://www.alsa-project.org/alsa-doc/alsa-lib/pcm_plugins.html

Tina sdk 下有相关软件包会设置/etc/asound.conf, 可以用作参考。

使用方法:

Tina 根目录下执行 make menuconfig, 选择 alsa-conf-aw 软件包。

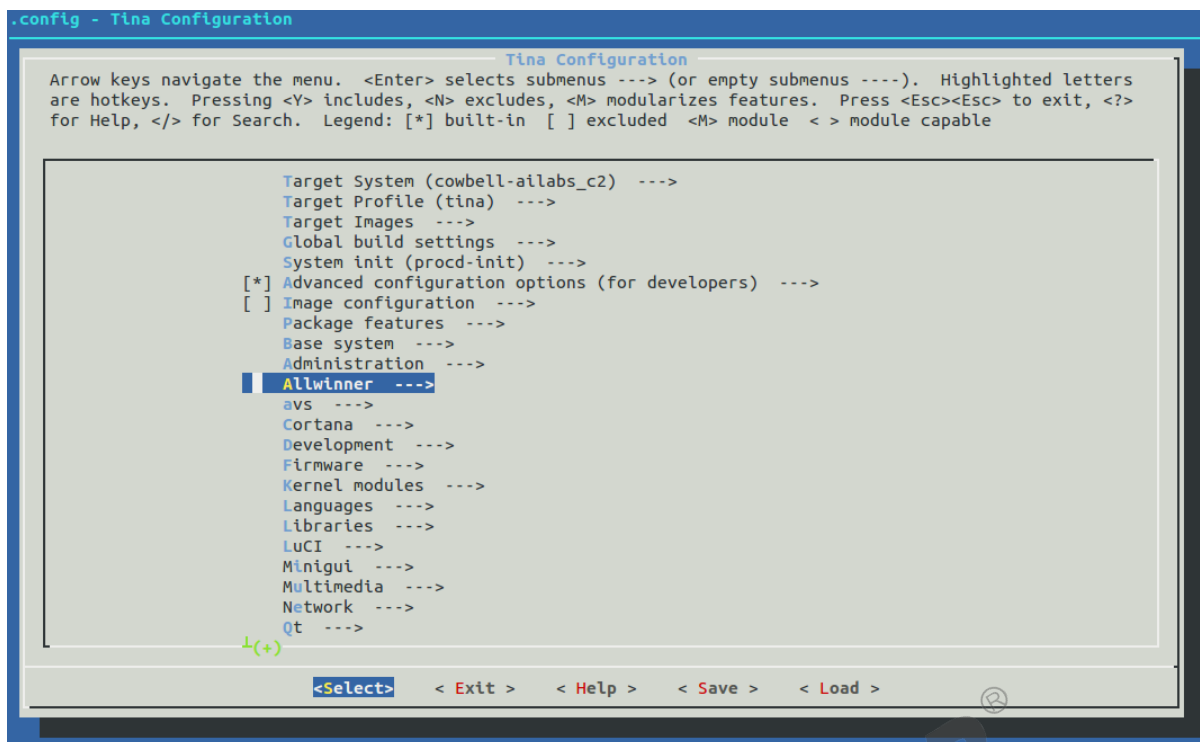


图 3-1: menuconfig allwinner

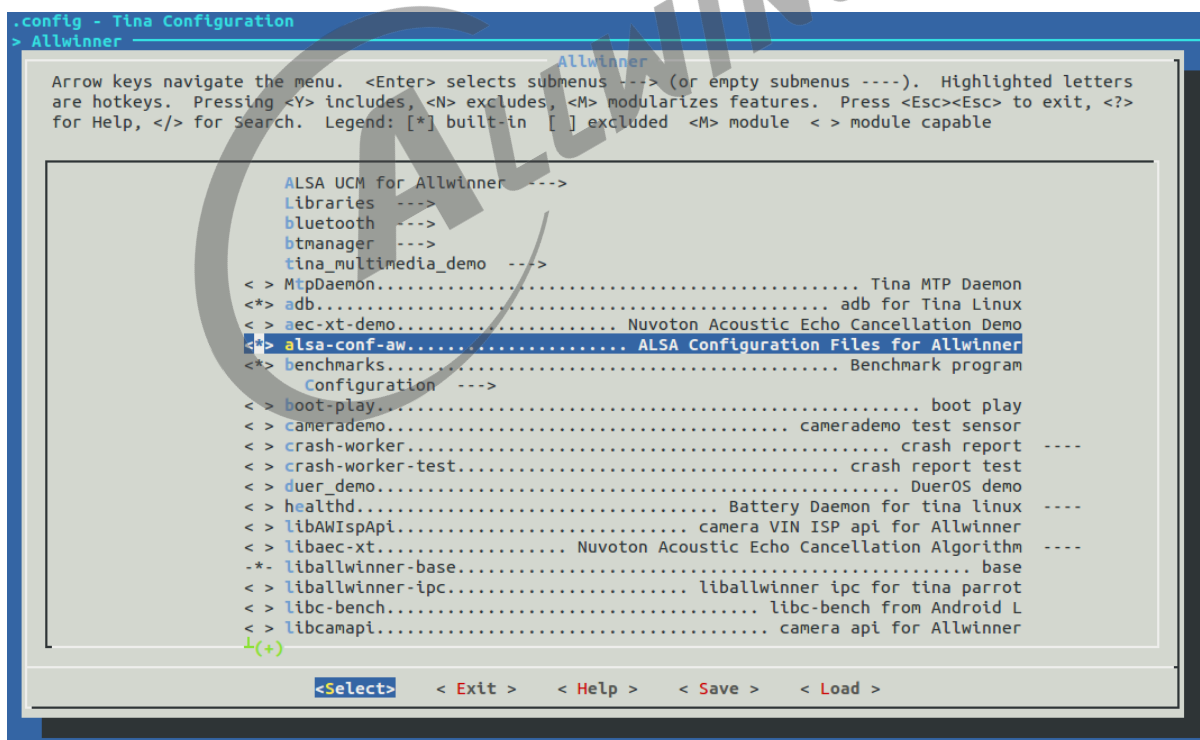


图 3-2: menuconfig alsa-conf-aw

它会生成/etc/asound.conf 文件，下面作简单介绍：

设定amixer操作的default声卡(执行snd_hctl_open会获取该配置)

```
ctl.!default {  
    type hw  
    card audiocodec  
}
```

设定default声卡(执行snd_pcm_open会获取该配置)

```
pcm.!default {  
    type asym  
    playback.pcm "PlaybackDmix"  
    capture.pcm "CaptureDsnoop"  
}
```

使用dmix插件，可以混合播歌，即支持多次打开声卡进行播歌

```
pcm.PlaybackDmix {  
    type plug  
    slave.pcm {  
        type dmix  
        ipc_key 1111  
        ipc_perm 0666  
        slave {  
            pcm "hw:audiocodec"  
            rate 48000  
            channels 2  
        }  
    }  
}
```

使用dsnoop插件，可以混合录音，即支持多次打开声卡进行录音

```
pcm.CaptureDsnoop {  
    type plug  
    slave.pcm {  
        type dsnoop  
        ipc_key 1111  
        ipc_perm 0666  
        slave {  
            pcm "hw:audiocodec,0"  
            rate 48000  
            channels 2  
        }  
    }  
}
```

使用dmix插件以及softvol插件，softvol插件可以增加一个control，用于控制音量(软件上作调节)

```
pcm.PlaybackDmix {  
    type plug  
    slave.pcm {  
        type softvol  
        slave.pcm {  
            type dmix  
            ipc_key 1111  
            ipc_perm 0666  
            slave {  
                pcm "hw:audiocodec,0"  
                rate 48000  
                channels 1  
            }  
        }  
    }  
    control {  
        name "Soft Volume Master"  
    }  
}
```

```

        card audiocodec
    }
    min_dB -51.0
    max_dB 0.0
    resolution 256
}
}

```

3.2 tinyalsa-utils

tinyalsa 是 alsa-lib 的一个简化版。它提供了 pcm 和 control 的基本接口；没有太多太复杂的操作、功能。可以按需使用接口。tinyalsa-utils 是基于 tinyalsa 的一些工具，下面对几个常用的工具作介绍。

3.2.1 tinymix

与 amixer 作用类似，用于操作 mixer control。

- 常用选项

选项	功能
-D,-card	指定声卡设备, 默认使用 card0

- 常用命令

命令	功能
controls	列出指定声卡的所有控件
contents	列出指定声卡的所有控件的具体信息
get	获取指定控件的信息
set	设定指定控件的值

举例：

```

获取card0的所有控件名
tinymix -D 0 controls

获取card0当前硬件音量
tinymix -D 0 get 'LINEOUT volume'

设置card0当前硬件音量

```

```
tinymix -D 0 set 'LINEOUT volume' 25
```

3.2.2 tinypplay

与 aplay 作用类似, 用于操作声卡设备进行播放

- 常用选项

选项	功能
-D,-card	指定声卡设备, 默认使用 card0
-p,-period-size	指定 period 大小, 单位为帧
-c,-channels	指定通道数
-r,-rate	指定采样率
-b,-bits	指定采样精度

如果播放的是 wav 文件, 可以解析头部, 识别通道数, 采样率等参数

举例:

```
tinypplay -D 0 /tmp/16000-stere-10s.wav
```

3.2.3 tinycap

与 arecord 作用类似, 用于操作声卡进行录音功能

- 常用选项

选项	功能
-D,-device	指定声卡设备, 默认使用 card0
-p,-period-size	指定 period 大小, 单位为帧
-c,-channels	指定通道数
-r,-rate	指定采样率
-b,-bits	指定采样精度

举例:

```
录制通道数为2，采样率为16000，采样精度为16bit，保存为wav文件
tinycap -D 0 -b 16 -r 16000 -c 2 /mnt/UDISK/test.wav
```

3.3 dump 寄存器

我们 sunxi 平台均提供了 sunxi_dump 驱动，用于查看读写寄存器。

节点位于/sys/class/sunxi_dump 目录。可以根据 spec 查看相关模块的寄存器地址，去进行读写操作。

3.3.1 dump audiocodec 寄存器

audiocodec 驱动的寄存器调试节点一般名字为 audio_reg，可通过 find 命令查找：

```
root@TinaLinux:/# find -name audio_reg
./sys/devices/platform/soc@3000000/2030000.codec/audio_reg_debug/audio_reg
root@TinaLinux:/#
```

使用方法：

通过 echo 写入下列参数

参数 1: 0-read; 1-write

参数 2: reg value

参数 3: write value

举例：

查看所有寄存器状态：

```
cat /sys/devices/platform/soc@3000000/2030000.codec/audio_reg_debug/audio_reg
```

REG NAME	OFFSET	VALUE
	31-28 27-24 23-20 19-16 15-12 11-08 07-04 03-00	save_value
SUNXI_DAC_DPC	0x 0 0x	0 0000 0000 0000 0000 0000 0000 0000
SUNXI_DAC_VOL_CTRL	0x 4 0x	1a0a0 0000 0000 0000 0001 1010 0000 1010
SUNXI_DAC_FIFOC	0x 10 0x	4000 0000 0000 0000 0000 0100 0000 0000
SUNXI_DAC_FIFOS	0x 14 0x	808008 0000 0000 1000 0000 1000 0000 0000
SUNXI_DAC_TXDATA	0x 20 0x	0 0000 0000 0000 0000 0000 0000 0000
SUNXI_DAC_CNT	0x 24 0x	0 0000 0000 0000 0000 0000 0000 0000
SUNXI_DAC_DG	0x 28 0x	0 0000 0000 0000 0000 0000 0000 0000

查看某个寄存器状态:

表示0x10寄存器的值为0x00004000

改写某个寄存器:

```
echo 1,0x24,0 > /sys/devices/platform/soc@3000000/2030000.codec/audio_reg_debug/audio_reg
```

表示将0x24寄存器写为0x0

3.3.2 dump daudio 寄存器

查看 spec 可以知道 i2s 模块的寄存器基地址

```
i2s0: 0x02032000
i2s1: 0x02033000
i2s2: 0x02034000
```

可以通过 sunxi_dump 节点查询寄存器状态, 例如查看 i2s0 的寄存器:

```
cd /sys/class/sunxi_dump
echo 0x02032000,0x020320a0 > dump
cat dump
```

注意在录音、播放状态下, 不要查看 RX DATA(0x10), TX DATA(0x20) 寄存器, 否则导致数据异常 (会把 FIFO 中的数据读出来了)

3.3.3 dump dmic 寄存器

查看 spec 可以知道 dmic 模块的寄存器基地址

```
dmic: 0x02031000
```

可以通过 sunxi_dump 节点查询寄存器状态:

```
cd /sys/class/sunxi_dump
echo 0x02031000,0x02031050 > dump
cat dump
```

注意在录音状态下, 不要查看 RX DATA(0x10) 寄存器, 否则导致数据异常 (会把 FIFO 中的数据读出来了)

3.3.4 dump spdif 寄存器

查看 spec 可以知道 spdif 模块的寄存器基地址

```
spdif: 0x02036000
```

可以通过 sunxi_dump 节点查询 spdif 寄存器状态:


```
cd /sys/class/sunxi_dump
echo 0x02036000,0x02036040 > dump
cat dump
```

3.4 sound procfs

通过 procfs 文件系统下面的声卡相关节点，可以得到各个声卡各个音频流的状态。实际调试中会非常有用。

内核需要选中下面选项才能在 procfs 下生成对应节点：

```
Device Drivers --->
<*> Sound card support --->
  <*> Advanced Linux Sound Architecture --->
    [*] Sound Proc FS Support
    [*] Verbose procfs contents
```

以 card0 为例看下提供的节点信息：

```
ddd
/proc/asound/card0/
├── id                /* 声卡名称 */
├── pcm0c             /* pcm0 录音流 */
│   ├── info         /* pcm信息 */
│   └── sub0
│       ├── hw_params /* 硬件参数信息 */
│       ├── info      /* pcm信息 */
│       ├── status    /* pcm流运行状态 */
│       └── sw_params /* 软件参数信息 */
└── pcm0p            /* pcm0 播放流 */
    ├── info
    └── sub0
```

其中，hw_params, status 都能拿到比较有用的信息：

```
cat /proc/asound/card0/pcm0c/sub0/hw_params
access: RW_INTERLEAVED          /* 交错模式排列通道 */
format: S16_LE                  /* 当前音频流的采样精度 */
subformat: STD
channels: 2                     /* 通道数 */
rate: 16000 (16000/1)           /* 采样率 */
period_size: 320                /* 周期(决定dma中断时间,例如这里period_time=320/16000=20ms) */
buffer_size: 2560               /* 内核ALSA框架中环形缓冲区大小,决定能够缓存多少个period */

cat /proc/asound/card0/pcm0c/sub0/status
state: RUNNING                  /* 音频流运行状态,RUNNING, SETUP等状态 */
owner_pid : 22653
trigger_time: 81828.078175765
tstamp : 82373.796969347        /* 开始运行后的时间戳信息 */
delay : 256
avail : 256                     /* 当前可用音频数据帧数 */
avail_max : 320
```

```
-----  
hw_ptr      : 8731456          /* 硬件逻辑指针, 单位(帧) */  
appl_ptr    : 8731200          /* 应用逻辑指针, 单位(帧) */
```

- 从 period_size 可以知道当前 dma 中断频率，太快会影响系统响应速度，太慢可能就存在一定延时。
- buffer_size 可以知道缓存区大小，太小容易因调度不及时出现 xrun, 太大同样存在一定延时。
- 从 hw_ptr, appl_ptr 可以知道当前录音/播音的帧数，是否发生过 xrun 等。



4 常用接口说明

这里主要介绍 alsa-lib 中的常用接口

4.1 control 接口

为了方便操作访问，alsa-lib 中封装了相关接口，通过 control 节点 (/dev/snd/controlCX) 去获取、设置 control elements

主要涉及到的接口：

```
snd_ctl_open  
snd_ctl_elem_info_get_id  
snd_ctl_elem_info_set_id  
snd_ctl_elem_info  
snd_ctl_ascii_value_parse  
snd_ctl_elem_read  
snd_ctl_elem_write  
snd_ctl_close
```

详细 control 接口说明请查阅：

<https://www.alsa-project.org/alsa-doc/alsa-lib/control.html>

https://www.alsa-project.org/alsa-doc/alsa-lib/group__control.html

下面是一个设置音量接口的例子：

```
#include <stdio.h>  
#include <stdlib.h>  
#include <stdbool.h>  
#include <stdint.h>  
#include <unistd.h>  
#include <string.h>  
  
#include <alsa/asoundlib.h>  
  
#define DEV_NAME "hw:audiocodec"  
#define VOLUME_CONTROL "name='LINEOUT volume'"  
  
/* Fuction to convert from percentage to volume. val = volume */  
static int convert_volume(int percent, long min, long max)  
{  
    long range = max - min;  
  
    if (range == 0)  
        return 0;
```

```
        return (int)((range * percent / 100) + min);
    }

bool controlVolume(int volume_percent)
{
    int err = -1;
    snd_ctl_t *handle = NULL;
    char *card = DEV_NAME;
    char *volume_control = VOLUME_CONTROL;
    char volume_string[4];
    long min, max, raw;

    snd_ctl_elem_info_t *info = NULL;
    snd_ctl_elem_id_t *id = NULL;
    snd_ctl_elem_value_t *control = NULL;

    if (volume_percent > 100 || volume_percent < 0)
        return false;

    snd_ctl_elem_info_alloca(&info);
    snd_ctl_elem_id_alloca(&id);
    snd_ctl_elem_value_alloca(&control);

    err = snd_ctl_ascii_elem_id_parse(id, volume_control);
    if (err < 0) {
        fprintf(stderr, "Wrong control identifier: %s\n", volume_control);
        goto failed;
    }
    err = snd_ctl_open(&handle, card, 0);
    if (err < 0) {
        fprintf(stderr, "Control device %s open error:%s\n", card, snd_strerror(err));
        goto failed;
    }
    snd_ctl_elem_info_set_id(info, id);
    err = snd_ctl_elem_info(handle, info);
    if (err < 0) {
        fprintf(stderr, "Cannot find the given element from control %s\n", card);
        goto failed;
    }
    snd_ctl_elem_info_get_id(info, id);
    snd_ctl_elem_value_set_id(control, id);
    err = snd_ctl_elem_read(handle, control);
    if (err < 0) {
        fprintf(stderr, "Cannot read the given element from control %s\n", card);
        goto failed;
    }
    min = snd_ctl_elem_info_get_min(info);
    max = snd_ctl_elem_info_get_max(info);
    snprintf(volume_string, sizeof(volume_string), "%d", convert_volume(volume_percent, min, max));
    /*printf("set volume %s, [%u%]\n", volume_string, volume_percent);*/
    err = snd_ctl_ascii_value_parse(handle, control, info, volume_string);
    if (err < 0) {
        fprintf(stderr, "Control %s parse error: %s\n", card, snd_strerror(err));
        goto failed;
    }
    err = snd_ctl_elem_write(handle, control);
    if (err < 0) {
        fprintf(stderr, "Control %s write error: %s\n", card, snd_strerror(err));
    }
}
```

```
        goto failed;
    }
failed:
    if (info)
        snd_ctl_elem_info_free(info);
    if (id)
        snd_ctl_elem_id_free(id);
    if (control)
        snd_ctl_elem_value_free(control);
    if (handle)
        snd_ctl_close(handle);

    return ((err < 0) ? false : true);
}
```

4.2 PCM 接口

为了方便操作访问，alsa-lib 中封装了相关接口，通过 pcmCXDXp/pcmCXDXc 节点 (/dev/snd/pcmCXDXx) 去实现播放、录音功能。

主要涉及到的接口：

```
snd_pcm_open
snd_pcm_info
snd_pcm_hw_params_any
snd_pcm_hw_params_set_access
snd_pcm_hw_params_set_format
snd_pcm_hw_params_set_channels
snd_pcm_hw_params_set_rate_near
snd_pcm_hw_params_set_buffer_size_near
snd_pcm_hw_params
snd_pcm_sw_params_current
snd_pcm_sw_params
snd_pcm_readi
snd_pcm_writeti
snd_pcm_close
```

详细 pcm 接口说明请查阅：

<https://www.alsa-project.org/alsa-doc/alsa-lib/pcm.html>

https://www.alsa-project.org/alsa-doc/alsa-lib/group__p_c_m.html

接口使用例子可以参考 aplay,arecord 的实现，代码可以在 alsa-utils 中找到 (dl/alsa-utils-1.1.0.tar.bz2)

5 调试注意事项

5.1 声卡没有加载

1. 确认 ASoC 框架中 codec,platform,machine 驱动的加载情况; 可以根据/sys/kernel/debug/asoc/的节点进行确认
2. 确认驱动的内核配置是否选了 (例如 Daudio, DMIC 等); 确认 dts 是否配置上了相关模块;
3. 驱动初始化失败了; 根据开机打印确认, 通常是 pinctrl 申请失败导致的, 请确认 dts 是否存在 IO 复用

5.2 播放没有声音

1. 检查音频通路是否开启了
2. 检查模拟功放 PA 使能引脚是否使能了 (dts 配置)




著作权声明

版权所有 © 2021 珠海全志科技股份有限公司。保留一切权利。

本文档及内容受著作权法保护，其著作权由珠海全志科技股份有限公司（“全志”）拥有并保留一切权利。

本文档是全志的原创作品和版权财产，未经全志书面许可，任何单位和个人不得擅自摘抄、复制、修改、发表或传播本文档内容的部分或全部，且不得以任何形式传播。

商标声明

、、**全志科技**、（不完全列举）均为珠海全志科技股份有限公司的商标或者注册商标。在本文档描述的产品中出现的其它商标，产品名称，和服务名称，均由其各自所有人拥有。

免责声明

您购买的产品、服务或特性应受您与珠海全志科技股份有限公司（“全志”）之间签署的商业合同和条款的约束。本文档中描述的全部或部分产品、服务或特性可能不在您所购买或使用的范围内。使用前请认真阅读合同条款和相关说明，并严格遵循本文档的使用说明。您将自行承担任何不当使用行为（包括但不限于如超压，超频，超温使用）造成的不利后果，全志概不负责。

本文档作为使用指导仅供参考。由于产品版本升级或其他原因，本文档内容有可能修改，如有变更，恕不另行通知。全志尽全力在本文档中提供准确的信息，但并不确保内容完全没有错误，因使用本文档而发生损害（包括但不限于间接的、偶然的、特殊的损失）或发生侵犯第三方权利事件，全志概不负责。本文档中的所有陈述、信息和建议并不构成任何明示或暗示的保证或承诺。

本文档未以明示或暗示或其他方式授予全志的任何专利或知识产权。在您实施方案或使用产品的过程中，可能需要获得第三方的权利许可。请您自行向第三方权利人获取相关的许可。全志不承担也不代为支付任何关于获取第三方许可的许可费或版税（专利税）。全志不对您所使用的第三方许可技术做出任何保证、赔偿或承担其他义务。