Linux Audio
开发指南

版本号: 2.1
发布日期: 2025.06.26

版本历史

版本号	日期	制/修订人	内容描述
1.0	2022.05.22	AWA1692	初始版本
1.1	2023.11.27	AWA1458	调整优化文档结构
1.2	2024.03.08	AWA1692	减少 platform 层接口重复介绍
1.3	2024.05.4	AWA2136	修改 menuconfig 配置说明
1.4	2024.07.23	AWA2077	新增 sun300iw1 相关配置
1.5	2024.10.29	AWA2136	新增 sun251iw1 相关配置, 补充 HDMI EDP AV 驱动说明
1.6	2025.02.07	AWA1458	新增 sun60iw2 相关配置
1.7	2025.03.10	AWA2077	新增 sun50iw15 相关配置
1.8	2025.03.12	AWA2077	新增 sun65iw1 相关配置
1.9	2025.04.12	AWA2278	新增 sun8iw22 相关配置
2.0	2025.05.10	AWA2136	附录针对复杂配置项增加调试指南
2.1	2025.06.26	AWS0733	附录针对耳机配置项增加相关说明

目　录

插　图

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、**模块介绍**章节主要从 **“配置->KO 加载-> 声卡控件-> 使用方法”** 等部分，介绍 sunxi 平台各音频接口的使用；

2、**驱动架构介绍**章节主要从 **“软件框图-> 源码结构-> 关键数据结构-> 接口说明”** 等部分，介绍音频驱动源码的关键内容，方便二次开发；

3、**测试工具介绍**章节主要从 **“工具->asound.conf 文件”** 等部分，介绍模块常用测试工具的使用和配置；

4、**FAQ** 章节主要提供了模块的调试方法和常见问题。

5、**附录**章节主要介绍了各平台的 AudioCodec 声卡使用等。

1.2 目标读者

音频系统相关人员。

1.3 适用平台

表 1-1: 适用平台列表

产品名称	内核版本	驱动文件
T113	linux-5.4	sound/soc/sunxi_v2/*
R528	linux-5.4	sound/soc/sunxi_v2/*
V853	linux-4.9	sound/soc/sunxi_v2/*
T3	linux-5.10	bsp/drivers/sound/platform/*
A40I	linux-5.10	bsp/drivers/sound/platform/*
MR153	llinux-5.15-origin	bsp/drivers/sound/platform/*

产品名称	内核版本	驱动文件
T153	linux-5.10-rt	bsp/drivers/sound/platform/*
T5	linux-5.4、linux-5.10	bsp/drivers/sound/platform/*
T507	linux-5.4、linux-5.10	bsp/drivers/sound/platform/*
H618	linux-5.4、linux-5.10	bsp/drivers/sound/platform/*
H313	linux-5.4、linux-5.10	bsp/drivers/sound/platform/*
A100	linux-5.10、linux-5.15	bsp/drivers/sound/platform/*
A133	linux-5.10、linux-5.15	bsp/drivers/sound/platform/*
T509	linux-5.10、linux-5.15	bsp/drivers/sound/platform/*
R818	linux-5.10、linux-5.15	bsp/drivers/sound/platform/*
A523	linux-5.15	bsp/drivers/sound/platform/*
A527	linux-5.15	bsp/drivers/sound/platform/*
T527	linux-5.15	bsp/drivers/sound/platform/*
MR527	linux-5.15	bsp/drivers/sound/platform/*
AI985	linux-5.15	bsp/drivers/sound/platform/*
H728	linux-5.15	bsp/drivers/sound/platform/*
TI533	linux-5.15-origin	bsp/drivers/sound/platform/*
T536	linux-5.15-origin	bsp/drivers/sound/platform/*
MR536	linux-5.15-origin	bsp/drivers/sound/platform/*
MR533	linux-5.15-origin	bsp/drivers/sound/platform/*
V821	linux-5.4-andes	bsp/drivers/sound/platform/*
F135	linux-6.6-xuantie	bsp/drivers/sound/platform/*
F136	linux-6.6-xuantie	bsp/drivers/sound/platform/*
H135	linux-6.6-xuantie	bsp/drivers/sound/platform/*
H136	linux-6.6-xuantie	bsp/drivers/sound/platform/*
H137	linux-6.6-xuantie	bsp/drivers/sound/platform/*
T736	linux-6.6	bsp/drivers/sound/platform/*
A733	linux-6.6	bsp/drivers/sound/platform/*
TV323	linux-5.15	bsp/drivers/sound/platform/*
H726	linux-5.15	bsp/drivers/sound/platform/*
A537	linux-6.6	bsp/drivers/sound/platform/*
A333	linux-6.6	bsp/drivers/sound/platform/*

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。

相关术语	解释说明
DAM	数字音频混音器。
OWA	外置数组音频接口，常用于同轴电缆或光纤输出。
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，其长度为样本精度与通道数的乘积。
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。
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。
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，可用于播录。

2 模块介绍

在 sunxi 中，从 Linux 软件上通常存在 5 类音频接口，如下。

- AudioCodec
- I2S/PCM
- I2S/PCM with DAM
- DMIC
- OWA

音频接口驱动针对上述音频接口，会分别创建播放设备 pcmXp 和录音设备 pcmXc（X：声卡序号）。

表 2-1: AW SOC 音频接口分布

平台	AudioCodec	I2S/PCM	DMIC	OWA
sun8iw20	DAC x2, 8k-192kHz ADC x3, 8k-48kHz HPOUTL/R LINEOUTP/N x2 MICP/N x3 LINEINL/R FMINL/R	x4	1-8ch 8k-48kHz	x1
sun8iw21	DAC x1, 8k-192kHz ADC x2, 8k-48kHz LINEOUTP/N MICP/N x2 LINEINL/R	x2	1-8ch 8k-48kHz	\
sun8iw11	DAC x2, 8k-192kHz ADC x2, 8k-48kHz HPOUTL/R LINEOUTL/R MIC x2 LINEINL/R FMINL/R	x3	\	x1
sun8iw22	DAC x1, 8k-192kHz LINEOUTP/N x1	x3	1-8ch 8k-48kHz	x1

平台	AudioCodec	I2S/PCM	DMIC	OWA
sun50iw9	DAC x2, 8k-192kHz LINEOUTL/R LINEINL/R FMINL/R	I2S/PCM x4 DAM x2 APB x3	1-8ch 8k-48kHz	x1
sun50iw10	DAC x2, 8k-192kHz ADC x2, 8k-48kHz HPOUTL/R LINEOUTLP/N MICP/N x2	x4	1-8ch 8k-48kHz	x1
sun55iw3	DAC x2, 8k-192kHz ADC x3, 8k-48kHz HPOUTL/R x1 LINEOUTP/N x2 MICP/N x3	x4	1-8ch 8k-48kHz	x1
sun55iw6	DAC x1, 8k-192kHz LINEOUTP/N x1	x4	1-8ch 8k-48kHz	x1
sun300iw1	DAC x1, 8k-192kHz ADC x1, 8k-48kHz LINEOUTP/N x1 MICP/N x1			
sun251iw1	DAC x2, 8k-192kHz ADC x2, 8k-48kHz LINEOUTL/R HPOUTL/R MICP/N x2 LINEINL/R FMINL/R	x3	1-8ch 8k-48kHz	x1
sun60iw2	\	x5	1-8ch 8k-48kHz	x1
sun50iw15	DAC x2, 8k-192kHz ADC x2, 8k-48kHz LINEOUTL/R HPOUTL/R LINEINL/R	x1		x2
sun65iw1	DAC x2, 8k-192kHz ADC x2, 8k-48kHz HPOUTL/R x1 LINEOUTP/N x2 MICP/N x2	x4	1-8ch 8k-48kHz	x1

说明

AW SOC 通常会内置上述多种接口，接口具体特性请参考发布文档中 《XXX_User_Manual_Vx.x.pdf》对应模块的描述。

2.1 AudioCodec

2.1.1 Device Tree 配置

2.1.1.1 配置路径

设备树中定义的是该类芯片对应于 IC 规格的所有配置，设备树文件路径如下：

- linux-4.9 ~ linux-5.4：

> 32 位平台：kernel/{KERNEl_VER}/arch/arm/boot/dts/{CHIP}.dtsi
> 64 位平台：kernel/{KERNEl_VER}/arch/arm64/boot/dts/sunxi/{CHIP}.dtsi

- linux-5.10（含 linux-5.10）之后：

> 32/64 位平台：bsp/configs/{KERNEl_VER}/{CHIP}.dtsi

说明

1. **{KERNEl_VER} 为内核版本，如 linux-5.15;**
2. **{CHIP}.dtsi 为具体芯片型号，如 sun50iw10p1.dtsi。**

2.1.1.2 配置示例

```
codec:codec@{module_base_reg} {
  #sound-dai-cells = <0>;
  compatible    = "allwinner,sunxi-snd-codec";
  reg           = <0x0 {module_base_reg} 0x0 0x32C>;
  resets        = <&ccu RST_BUS_AUDIO_CODEC>;
  clocks        = <&ccu CLK_BUS_AUDIO_{module}>,
                  <&ccu CLK_PLL_xxx1>,   /* 24.576M * n*/
                  <&ccu CLK_PLL_xxx2>,   /* 22.5792M * n */
                  <&ccu CLK_AUDIO_{module}>,
  clock-names   = "clk_bus_audio_{module}",
                  "clk_pll_xxx1",
                  "clk_pll_xxx2",
                  "clk_audio_{module}";
  interrupts    = <GIC_SPI {irq_num} IRQ_TYPE_LEVEL_HIGH>;   /* jack irq */
  status        = "disabled";
};
```

```
codec_plat:codec_plat {
  #sound-dai-cells = <0>;
  compatible    = "allwinner,sunxi-snd-plat-aaudio";
  dac-txdata    = <{dac_txdata_reg}>;
  adc-rxdata    = <{adc_rxdata_reg}>;
  dmas          = <&dma {DRQ_PORT}>, <&dma {DRQ_PORT}>;
  dma-names     = "tx", "rx";
  playback-cma  = <128>;
  capture-cma   = <128>;
  tx-fifo-size  = <128>;
  rx-fifo-size  = <128>;
  status        = "disabled";
};

codec_mach:codec_mach {
  compatible                = "allwinner,sunxi-snd-mach";
  soundcard-mach,name       = "audiocodec";
  soundcard-mach,pin-switches   = "MIC1", "MIC2",
                    "LINEOUT", "HPOUT", "SPK";
  soundcard-mach,routing        = "MIC1_PIN", "MIC1",
                    "MIC2_PIN", "MIC2",
                    "LINEOUT", "LINEOUTL_PIN",
                    "HPOUT", "HPOUTL_PIN",
                    "HPOUT", "HPOUTR_PIN",
                    "SPK", "HPOUTL_PIN",
                    "SPK", "HPOUTR_PIN",
                    "SPK", "LINEOUTL_PIN";
  soundcard-mach,jack-support   = <1>;
  status                    = "disabled";
  soundcard-mach,cpu {
    sound-dai   = <&codec_plat>;
  };
  soundcard-mach,codec {
    sound-dai   = <&codec>;
    soundcard-mach,pll-fs  = <{n}>;
  };
};
```

说明

本部分仅为示例，具体内容根据实际情况填写。

配置项说明

AudioCodec 模块由 3 个设备树节点构建。

1、ASoC 层 codec: codec

表 2-2: 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。

配置项名称	配置项说明
interrupts	AudioCodec 中断号

2、ASoC 层 platform: codec_plat

表 2-3: AudioCodec codec_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。
playback-cma	设置播放流 DMA 申请 size 大小，为 (2^n)Kbyte，单位 Kb。
capture-cma	设置录音流 DMA 申请 size 大小，为 (2^n)Kbyte，单位 Kb。
tx-fifo-size	设置播放流的 fifo_size 大小，用于声卡参数限定，单位 Kb。
rx-fifo-size	设置录音流的 fifo_size 大小，用于声卡参数限定，单位 Kb。
dac-txdata	设置播放流 DMA 搬运地址 (audiocodec 模块 tx_fifo 寄存器地址)。
adc-rxdata	设置录音流 DMA 搬运地址 (audiocodec 模块 rx_fifo 寄存器地址)。
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: codec_mach

表 2-4: AudioCodec codec_mach 节点配置项

配置项名称	配置项说明
soundcard-mach, name	machine 层配置前缀。 声卡名字。
pin-switches	用于定义模块接口开关， 需参考驱动代码 dapm 进行设定。
routing	用于定义模块接口开关所链接的 dapm 通路， 需参考驱动代码 dapm 进行设定。
jack-support	指定支持的耳机检测类型，具体检测类型说明见附录耳机配置说明 0：不支持耳机，1：内置 codec 耳机检测， 2：extcon 耳机检测，3：gpio 耳机检测，4：advance 耳机检测。
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。
pll-fs	指定模块时钟源频率（24.576M or 22.5792M * pll-fs）。

2.1.2 board.dts 配置

2.1.2.1 配置路径

board.dts 用于保存每一个板级平台的设备信息（如 demo 板，perf1 板等），里面的配置信息会覆盖上面的 Device Tree 中 dtsi 默认配置信息。不同 IC、版型及内核版本对应的 board.dts 具体路径如下。

> device/config/chips/{PLATFORM}/configs/{BOARD}/{KERNEl_VER}/board.dts

2.1.2.2 配置示例

```
&codec {
    /* note: power settings */
    rglt-max            = <n>;       /* n: rglt cnt */
    rglt(n)-mode        = "xxx";     /* PMU; AUDIO; */
    rglt(n)-voltage     = <n>;       /* n: vcc voltage */
    rglt(n)-supply      = <&pmu_node>;  /* AUDIO mode unnecessary */
    /* note: volume settings */
    dac-vol             = <63>;
    dac(n)-vol          = <160>;
    adc(n)-vol          = <160>;
    adc(n)-gain         = <31>;
    lineout-gain        = <31>;
    hpout-gain          = <7>;
    /* note: pa settings */
    pa-pin-max          = <2>;
    /* 0: level contrl; 1: pulse contrl; 0xFF: user contrl. */
    pa-cfg-mode-0       = <0>;
    pa-pin-0            = <&pio xxx>;
    pa-pin-level-0      = <1>;
    pa-pin-msleep-0     = <0>;
    pa-pin-1            = <&pio xxx>;
    pa-cfg-mode-1       = <1>;
    pa-pin-level-1      = <1>;
    pa-pin-msleep-1     = <n>;
    pa-pin-duty-1       = <n>;
    pa-pin-period-1     = <n>;
    pa-pin-polarity-1   = <1>;
    pa-pin-periodcnt-1  = <n>;
    /* note: jack param -> gpio */
    hp-det-gpio         = <&pio xxx>;
    /* note: jack param -> codec */
    jack-det-level      = <0>;
    jack-det-threshold  = <8>;
    jack-det-debouce    = <250>;
    /* note: jack param -> extcon */
    extcon              = <&usb_power_supply>;
    jack-swpin-max      = <3>;
    jack-swpin-0        = <&pio xxx>;
    jack-swpin-1        = <&pio xxx>;
    jack-swpin-2        = <&pio xxx>;
```

```
    jack-mode-off        = <0xf 0 0>;
    jack-mode-usb        = <0xf 1 1>;
    jack-mode-audio      = <0xf 1 0>;
    jack-mode-micn       = <1 0xf 0xf>;
    jack-mode-mici       = <0 0xf 0xf>;
    jack-det-level       = <1>;
    jack-det-threshold   = <8>;
    jack-det-debounce    = <250>;
    /* jack-key-det-voltage = <min max> */
    jack-key-det-voltage-hook  = <0 0>;
    jack-key-det-voltage-up    = <2 2>;
    jack-key-det-voltage-down  = <4 5>;
    jack-key-det-voltage-voice = <1 1>;
    /* note: sdbp - headset detective based on headphone */
    jack-sdbp-method         = <1>;
    jack-sdbp-scan-single-time = <1000>;
    /* note: other settings */
    adc-delay-time       = <0>;
    tx-hub-en;
    rx-sync-en;
    status               = "okay";
  };

&codec_plat {
  status = "okay";
};

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

配置项说明

说明

- **功放有三类：电平控制类、脉冲控制类、用户自定义实现控制类，根据原理图确认使用功放类型；**
- **耳机的相关检测配置说明见附录耳机配置说明**
- **增益或音量的配置值范围每个平台可能存在差异，需通过 spec 或驱动确认。**

表 2-5: AudioCodec 模块板级配置项-通用类

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。
dac-vol	0~63	dac 总数字音量， 音量范围：-73.08->0dB。
dac(n)-vol	0~255	dac(n) 数字音量， 音量范围-119.25->71.25dB。
adc(n)-vol	0~255	adc(n) 数字音量， 音量范围：-119.25->71.25dB。
adc(n)-gain	0~31	adc(n) 模拟增益， 音量范围：0->36dB。
lineout-gain	0~31	lineout 输出增益， 音量范围：-43.5->0,0dB。
hpout-gain	0~7	hpout 输出增益，

配置项名称	配置值范围	配置项说明
		音量范围：-42->0dB。
fminl-gain	0~7	fminl 模拟增益， 音量范围：0->6dB。
fminr-gain	0~7	fminr 模拟增益， 音量范围：0->6dB。
lineinl-gain	0~1	lineinl 模拟增益， 音量范围：0->6dB。
lineinr-gain	0~1	lineinr 模拟增益， 音量范围：0->6dB。
adc-delay-time	0,5,10,20,30	设置 adc 录音延迟时长，单位 ms。
tx-hub-en	注释为 false, 反之为 ture	选择是否注册 txhub 控件。
rx-sync-en	注释为 false, 反之为 ture	选择是否注册 rxsync 控件。
pa-cfg-mode-(n)	0, 1, 0xff	指定第 (n) 个功放的控制方式。 0：电平控制。 1：脉冲控制。 0xff：脉冲控制。

表 2-6: AudioCodec 模块板级配置项-电源类

配置项名称	配置值范围	配置项说明
rglt-max	u32	模块需要电源的总数。
rglt(n)-mode	“PMU”，“AUDIO”	模块所需第 n 路电的供电模式。
rglt(n)-voltage	1800000，3300000	模块所需第 n 路电的电压值，单位 uV。
rglt(n)-supply	缺省，pmu 节点	模块所需第 n 路电的供电来源。

表 2-7: AudioCodec 模块板级配置项-电平控制类功放

配置项名称	配置值范围	配置项说明
pa-pin-max	u32	标定外部功放芯片使能引脚数量。
pa-pin-(n)	pio 引脚	指定第 (n) 个功放使能引脚。
pa-pin-level-(n)	0~1	指定功放芯片使能电平。
pa-pin-msleep-(n)	u32	设置功放芯片使能延迟时长， 单位 ms, 正常小于 200， 常用于规避 pop 声。
pa-pin-msleep1-(n)	u32	设置功放芯片关闭后， soc 音频输出延迟时长， 单位 ms, 正常小于 200， 常用于规避功放关闭时的 pop 声。

表 2-8: AudioCodec 模块板级配置项-脉冲控制类功放

配置项名称	配置值范围	配置项说明
pa-pin-duty-(n)	u32	脉冲的宽度, 单位 us。
pa-pin-period-(n)	u32	脉冲的周期，单位 us。
pa-pin-polarity-(n)	1	脉冲的极性。
pa-pin-periodcnt-(n)	u32	脉冲的个数。

2.1.3 AudioCodec kernel menuconfig 配置说明

linux-4.9 ~ linux-5.4，menuconfig 必选配置如下。

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

linux-5.10 及其以上内核版本，menuconfig 必选配置如下。

```
Allwinner BSP  --->
  Device Drivers  --->
    SOUND Drivers  --->
      Platform drivers  --->
        <M> Allwinner AAUDIO support
```

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）。

配置项说明

表 2-9: AudioCodec kernel menuconfig 可选配置项

配置项名称	配置项说明
Allwinner AAUDIO support	AudioCodec 模块。
Allwinner function components	功能组件模块。
Components SFX	供 cedarSE 使用功能组件（硬件算法更改寄存器）。
Components Debug	调试节点功能组件（查看音频寄存器）。
Enable audio dynamic debug	使能 AUDIO 模块 DYNAMIC DEBUG 模式。

说明

1. **Components SFX 与 Components Debug 依赖于 Allwinner function components，可按照实际需求选择。**

2.1.4 加载与卸载方法（ko 方式）

sunxi 音频驱动模块分五大类驱动如下。

- 特定功能组件（sfx）
- PCM 驱动
- ASoC platfrom 驱动
- ASoC codec 驱动
- ASoC machine 驱动

ko 加载顺序遵循 **“公共组件 -> 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。

```
# 公共组件，提供公共接口
insmod snd_soc_sunxi_common.ko

# 特殊功能组件(具体根据实际打开的component加载)
insmod snd_soc_sunxi_component_sfx.ko

# PCM 驱动
insmod snd_soc_sunxi_pcm.ko

# ASoC platfrom 驱动 和 ASoC codec 驱动
insmod snd_soc_sunxi_aaudio.ko
insmod snd_soc_sunxi_internal_codec.ko

# ASoC machine 驱动
insmod snd_soc_sunxi_machine.ko
```

2.1.5 声卡控件介绍

不同平台的 AudioCodec 的声卡控件不完全相同，声卡控件及常用使用方法见AudioCodec 声卡使用。AudioCodec 的声卡控件分为三类，分别是特殊功能类控件，音量调节类控件，通路开关类控件。

表 2-10: 特殊功能类控件说明

控件名称	功能	数值
tx hub mode	同源播放开关	Off;On
rx sync mode	同步录音开关	Off;On
ADC DRC{n} Mode	ADC DRC{n} 开关	Off;On
ADC HPF{n} Mode	ADC HPF{n} 开关	Off;On
DAC DRC Mode	DAC DRC 开关	Off;On
DAC HPF Mode	DAC HPF 开关	Off;On

控件名称	功能	数值
ADDA Loop Mode	ADC DAC 回路开关	Off;On
DAC{n} DAC{m} Swap	DAC{n}&{m} 通道交换开关	Off;On
ADC{n} ADC{m} swap	ADC{n}&{m} 通道交换开关	Off;On

表 2-11: 音量调节类控件说明

控件名称	功能	数值
DAC Volume	dac 数字音量调节	0->63 (-73.08->0dB)
DAC{n} Volume	dac{n} 数字音量调节	0->255 (-119.25->71.25dB)
ADC{n} Volume	adc1 数字音量调节	0->255 (-119.25->71.25dB)
ADC{n} Gain	adc{n} 模拟增益调节	0~31(0->36dB)
LINEOUT{n} Gain	lineout{n} 输出增益调节	0~31(-43.5->0,0dB)
HPOUT Gain	hpout 输出增益调节	0~7(-42->0dB)

表 2-12: 通路开关类控件说明

控件名称	功能	数值
MIC{n} Switch	MIC{n} 通路开关	Off;On
LINEOUT{n} Switch	LINEOUT{n} 通路开关	Off;On
HPOUT Switch	HPOUT 通路开关	Off;On
SPK Switch	SPK 通路开关	Off;On
FMIN{n} Switch	FMIN{n} 通路开关	Off;On
LINEIN{n} Switch	LINEIN{n} 通路开关	Off;On
SPK Switch	SPK 通路开关	Off;On
Output{n} Mixer DAC{n} Switch	DAC{n}—>LINEOUT{n} 通路开关	Off;On
LINEOUT{n} Output Select	LINEOUT{n} 输出模式	Single;Differ
MIC{n} Input Select	MIC{n} 输入模式	Single;Differ
Input{n} Mux	Input{n} 输入源	MIC{n};FMINL;LINEINL
X Output/Input Mixer Y Switch	Y->X 通路开关	Off;On

2.2 I2S/PCM

2.2.1 Device Tree 配置

2.2.1.1 配置路径

设备树中定义的是该类芯片对应于 IC 规格的所有配置，设备树文件路径如下：

- linux-4.9、linux-5.4：

> 32 位平台：kernel/{KERNEl_VER}/arch/arm/boot/dts/{CHIP}.dtsi
> 64 位平台：kernel/{KERNEl_VER}/arch/arm64/boot/dts/sunxi/{CHIP}.dtsi

- linux-5.10（含 linux-5.10）之后：

> 32/64 位平台：bsp/configs/{KERNEl_VER}/{CHIP}.dtsi

说明

1. **{KERNEl_VER} 为内核版本，如 linux-5.15；**
2. **{CHIP}.dtsi 为具体芯片型号，如 sun55iw3p1.dtsi。**

2.2.1.2 配置示例

- 第 n 组 I2S 配置如下。

```
i2s{n}_plat:i2s{n}_plat@{module_base_reg} {
    #sound-dai-cells = <0>;
    compatible     = "allwinner,sunxi-snd-plat-i2s";
    reg            = <0x0 {module_base_reg} 0x0 0xA0>;
    resets         = <&ccu RST_BUS_{module}>;
    clocks         = <&ccu CLK_PLL_xxx1>,
                     <&ccu CLK_PLL_xxx2>,
                     <&ccu CLK_xx1>;
    clock-names    = "clk_pll_xxx1",
                     "clk_pll_xxx2",
                     "clk_{module}";
    dmas           = <&dma1 {DRQ_PORT}>, <&dma1 {DRQ_PORT}>;
    dma-names      = "tx", "rx";
    playback-cma   = <128>;
    capture-cma    = <128>;
    tx-fifo-size   = <128>;
    rx-fifo-size   = <128>;
    status         = "disabled";
};
```

```
i2s{n}_mach:i2s{n}_mach{
    compatible              = "allwinner,sunxi-snd-mach";
    soundcard-mach,name     = "snd i2s{n}";
    soundcard-mach,format   = "i2s";
    soundcard-mach,slot-num    = <2>;
    soundcard-mach,slot-width  = <32>;
    status                  = "disabled";
    soundcard-mach,cpu {
      sound-dai = <&i2s{n}_plat>;
    };
    soundcard-mach,codec {
    };
};
```

说明

本部分仅为示例，具体内容根据实际情况填写。

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: i2s(n)_plat

表 2-13: I2S/PCM i2s(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。
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。
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。
playback-cma	设置播放流 DMA 申请 size 大小，为 (2^n)Kbyte，单位 Kb。
capture-cma	设置录音流 DMA 申请 size 大小，为 (2^n)Kbyte，单位 Kb。
tx-fifo-size	设置播放流的 fifo_size 大小，用于声卡参数限定，单位 Kb。
rx-fifo-size	设置录音流的 fifo_size 大小，用于声卡参数限定，单位 Kb。

3、ASoC 层 machine: i2s(n)_mach

表 2-14: I2S/PCM i2s(n)_mach 节点配置项

配置项名称	配置项说明
soundcard-mach, name	machine 层配置前缀。 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。

配置项名称	配置项说明
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。

2.2.2 board.dts 配置

2.2.2.1 配置路径

board.dts 用于保存每一个板级平台的设备信息（如 demo 板，perf1 板等），里面的配置信息会覆盖上面的 Device Tree 中 dtsi 默认配置信息。不同 IC、版型及内核版本对应的 board.dts 具体路径如下。

> device/config/chips/{PLATFORM}/configs/{BOARD}/{KERNEl_VER}/board.dts

2.2.2.2 配置示例

```
&i2s{n}_plat {
    tdm-num         = <{n}>;
    tx-pin          = <{m} ...>;
    tx-pin{m}-chmap     = <{num1} {num2} ...>;
    rxfifo-pinmap       = <{m} ...>;
    rxfifo-chmap        = <{num1} {num2} ...>;
    /* pinctrl-used; */
    /* pinctrl-names   = "default","sleep"; */
    /* pinctrl-0       = <&i2s{n}_pins_a>; */
    /* pinctrl-1       = <&i2s{n}_pins_b>; */
    tx-hub-en;
    rx-sync-en;
    status          = "okay";
};

&i2s{n}_mach {
    soundcard-mach,format         = "i2s";
    soundcard-mach,frame-master     = <&i2s{n}_cpu>;
    soundcard-mach,bitclock-master  = <&i2s{n}_cpu>;
    /* soundcard-mach,frame-inversion; */
    /* soundcard-mach,bitclock-inversion; */
    soundcard-mach,slot-num         = <2>;
    soundcard-mach,slot-width       = <32>;
    soundcard-mach,pin-switches     = "SPK";
    soundcard-mach,routing          = "SPK", "I2S_PIN";
    soundcard-mach,capture-only;
    status      = "okay";
    i2s{n}_cpu: soundcard-mach,cpu {
        sound-dai = <&i2s{n}_plat>;
        /* note: pll freq = 24.576M or 22.5792M * pll-fs */
```

```
    soundcard-mach,pll-fs  = <1>;
    /* note:
     * "mclk-fs"
     * if not defined it or equal to 0, disable mclk.
     *
     * "mclk-fp" (if defined "mclk-fs")
     * 1. if not defined "mclk-fp", mclk_freq = mclk-fs * sample_rate;
     * 2. if defined "mclk-fp" but no value, mclk_freq = mclk-fs * 11.2896M or 12.288M;
     * 3. if defined "mclk-fp" with 2 value, like: mclk-fp = <val1 val2>;
     *   it means: mclk_freq(44.1k fp) = mclk-fs * val1;
     *          mclk_freq(48k fp)  = mclk-fs * val2.
     */
    soundcard-mach,mclk-fs  = <1>;
    soundcard-mach,mclk-fp  = <11289600 12288000>;
  };
  i2s{n}_codec: soundcard-mach,codec {
    sound-dai        = <&ac107>;
    soundcard-mach,pll-fs  = <1>;
  };
};
```

说明

gpio 部分请参考附件GPIO 功能复用配置

配置项说明

表 2-15: 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。
tdm-num	0~1	指定 I2S 序号，需和 i2s(n)_plat 的 (n) 对应。
tx-pin	0~3	指定 I2S 所使用的 DOUT 引脚序号。
tx-pin{m}-chmap	0~15	指定第 m 个 DOUT 脚 txfifo 通道序号到 tx slot 序号的映射
rxfifo-pinmap	0~3	指定每个 slot 数据来源于哪路 DIN 引脚。
rxfifo-chmap	0~15	指定 rx slot 序号到 rxfifo 通道序号的映射。
tx-hub-en	注释为 false, 反之为 ture	选择是否注册 txhub 控件。
rx-sync-en	注释为 false, 反之为 ture	选择是否注册 rxsync 控件。
capture-only	注释为 false, 反之为 ture	选择是否注册录音的设备节点。
pin-switches	“SPK”	用于控制外挂 codec/PA。
routing	“SPK”，“I2S_PIN”	SPK 开关所链接的 dapm 通路。
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。
slot-num	1~16	slot 数量， 可简单理解为支持最大通道数。
slot-width	8, 16, 24, 32	单个 slot 宽度， 可简单理解为支持最大数据精度。

配置项名称	配置值范围	配置项说明
mclk-fp	注释为 false, 反之为 ture	ture: 指定 mclk-fp[0] 与 mclk-fp[1] 默认值：11289600 12288000 false: mclk 以采样率倍数输出。 默认值：0 0
mclk-fs	u32	固定频段：mclk = mclk-fs * mclk-fp[0] or mclk-fs * mclk-fp[1]。 采样率倍数：mclk = mclk-fs * pcm rate。

2.2.3 I2S kernel menuconfig 配置说明

linux-5.10 以下内核版本，menuconfig 必选配置如下。

```
Device Drivers  --->
  <*> Sound card support  --->
    <*> Advanced Linux Sound Architecture  --->
      <*> ALSA for SoC audio support  --->
        Allwinner SoC Audio support V2  --->
          <M> Allwinner I2S Support
```

linux-5.10 及其以上内核版本，menuconfig 必选配置如下。

```
Allwinner BSP  --->
  Device Drivers  --->
    SOUND Drivers  --->
      Platform drivers  --->
        <M> Allwinner I2S Support
```

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）。

配置项说明

表 2-16: I2S/PCM menuconfig 可选配置项

配置项名称	配置项说明
Allwinner I2S support	I2S/PCM 模块。
Allwinner HDMIAUDIO Support	HDMI AUDIO 模块。
Allwinner EDPAUDIO Support	EDP AUDIO 模块。
Allwinner AVAUDIO Support	AV AUDIO 模块 (new edp driver, 应用于 sun60iw2 以后的新平台)。
Allwinner function components	功能组件模块。
Components Debug	调试节点功能组件（查看音频寄存器与 i2s 格式调试）。
Enable audio dynamic debug	使能 AUDIO 模块 DYNAMIC DEBUG 模式。

2.2.4 加载与卸载方法（ko 方式）

sunxi 音频驱动模块分五大类驱动如下。

- PCM 驱动
- ASoC platfrom 驱动
- ASoC codec 驱动
- ASoC machine 驱动

ko 加载顺序遵循 **“公共组件 -> 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。

```
# 公共组件，提供公共接口
insmod snd_soc_sunxi_common.ko

# PCM 驱动
insmod snd_soc_sunxi_pcm.ko

# ASoC platfrom 驱动
insmod snd_soc_sunxi_i2s.ko

# HDMI Audio驱动（若使用HDMI Audio）
insmod snd_soc_codec_hdmi.ko
# EDP Audio驱动（若使用EDP Audio）
insmod snd_soc_codec_edp.ko
# AV Audio驱动（若使用AV Audio）
insmod snd_soc_codec_av.ko

# ASoC machine 驱动
insmod snd_soc_sunxi_machine.ko
```

2.2.5 声卡控件介绍

2.2.5.1 I2S 控件

```
Mixer name: 'sndi2s0'
Number of controls: 3
ctl   type   num    name                    value
0     ENUM   1     tx hub mode               Off
1     ENUM   1     rx sync mode              Off
2     BOOL   1     loopback debug              Off
3     BOOL   1     SPK Switch                Off
```

```
ahub_dam_plat:ahub_dam_plat@{module_base_reg}{
   #sound-dai-cells = <0>;
   /* sound card without pcm for hardware mix setting */
   compatible     = "allwinner,sunxi-snd-plat-ahub_dam";
   reg         = <0x0 {module_base_reg} 0x0 0xAEC>;
```

```
    resets       = <&ccu RST_BUS_AUDIO_{module}>;
    clocks       = <&ccu CLK_BUS_AUDIO_{module}>,
                   <&ccu CLK_PLL_AUDIO(n)>,
                   <&ccu CLK_AUDIO_{module}>;
    clock-names  = "clk_bus_audio_{module}",
                   "clk_pll_audio(n)",
                   "clk_audio_{module}";
    status = "disabled";
};

ahub_dam_mach:ahub_dam_mach {
    compatible          = "allwinner,sunxi-snd-mach";
    soundcard-mach,name  = "ahubdam";
    status              = "disabled";
    soundcard-mach,cpu {
        sound-dai = <&ahub_dam_plat>;
    };
    soundcard-mach,codec {
    };
};

ahub{n}_plat:ahub{0}_plat {
    #sound-dai-cells = <0>;
    compatible          = "allwinner,sunxi-snd-plat-ahub";
    apb-num             = <{n}>;
    dmas                = <&dma {DRQ_PORT}>, <&dma {DRQ_PORT}>;
    dma-names           = "tx", "rx";
    playback-cma        = <128>;
    capture-cma         = <128>;
    tx-fifo-size        = <128>;
    rx-fifo-size        = <128>;
    status              = "disabled";
};

ahub{n}_mach:ahub{n}_mach {
    compatible      = "allwinner,sunxi-snd-mach";
    soundcard-mach,name = "ahubi2s{n}";
    status          = "disabled";
    soundcard-mach,cpu {
        sound-dai = <&ahub{n}_plat>;
    };
    soundcard-mach,codec {
    };
};
```

说明

本部分仅为示例，具体内容根据实际情况填写。

配置项说明：

AHUB DAM 由 2 个设备树节点构建，AHUB 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: ahub_dam_plat 和 ahub(n)_plat

表 2-17: AHUB DAM ahub_dam_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 AHUB 寄存器起始地址和地址长度。
resets	设置 AHUB 所需的复位时钟。
clocks	设置 AHUB 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。

表 2-18: AHUB ahub(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。
apb-num	设置模块所绑定的 apb 通道号（对应一组 DMA）。
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。
playback-cma	设置播放流 DMA 申请 size 大小，为 (2^n)Kbyte，单位 Kb。
capture-cma	设置录音流 DMA 申请 size 大小，为 (2^n)Kbyte，单位 Kb。
tx-fifo-size	设置播放流的 fifo_size 大小，用于声卡参数限定，单位 Kb。
rx-fifo-size	设置录音流的 fifo_size 大小，用于声卡参数限定，单位 Kb。

3、ASoC 层 machine: daudio(n)_mach

表 2-19: 混音部分 ahub_dam_mach 节点配置项

配置项名称	配置项说明
soundcard-mach, name	machine 层配置前缀。 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。

表 2-20: I2S/PCM ahub(n)_mach 节点配置项

配置项名称	配置项说明
soundcard-mach, name	machine 层配置前缀。 声卡名字。
cpu	machine 层所绑定的 cpu 节点（即 platform 层），

配置项名称	配置项说明
codec	用 sound-dai 属性指定节点。 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。
pll-fs	指定模块时钟源频率（24.576M or 22.5792M * pll-fs）。

2.2.6 board.dts 配置

2.2.6.1 配置路径

board.dts 用于保存每一个板级平台的设备信息（如 demo 板，perf1 板等），里面的配置信息会覆盖上面的 Device Tree 中 dtsi 默认配置信息。不同 IC、版型及内核版本对应的 board.dts 具体路径如下。

> device/config/chips/{PLATFORM}/configs/{BOARD}/{KERNEl_VER}/board.dts

2.2.6.2 配置示例

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

&ahub_dam_mach {
    status = "okay";
};

&ahub{n}_plat {
    tdm-num       = <{n}>;
    tx-pin        = <{m}>;
    rx-pin        = <{m}>;
    pinctrl-used;
    pinctrl-names  = "default","sleep";
    pinctrl-0     = <&ahub_i2s{n}_pins_a>;
    pinctrl-1     = <&ahub_i2s{n}_pins_b>;
    status        = "okay";
};

&ahub{n}_mach {
    soundcard-mach,format      = "i2s";
    soundcard-mach,frame-master   = <&ahub{n}_cpu>;
    soundcard-mach,bitclock-master   = <&ahub{n}_cpu>;
    /* soundcard-mach,frame-inversion; */
    /* soundcard-mach,bitclock-inversion; */
    soundcard-mach,slot-num      = <2>;
    soundcard-mach,slot-width   = <32>;
```

```
    soundcard-mach,capture-only;
    status = "okay";
    ahub{n}_cpu: soundcard-mach,cpu {
        sound-dai          = <&ahub{n}_plat>;
        soundcard-mach,pll-fs   = <4>;
        soundcard-mach,mclk-fs   = <1>;
        soundcard-mach,mclk-fp;
    };
    ahub0_codec: soundcard-mach,codec {
        sound-dai          = <&ac107>;
        soundcard-mach,pll-fs   = <1>;
    };
};
```

说明

gpio 部分请参考附件GPIO 功能复用配置

配置项说明：

表 2-21: 带混音 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。
tdm-num	0~3	指定 I2S 序号， 需和 ahub(n)_plat 的 (n) 对应。
tx-pin	0~3	指定 I2S 所使用的 DOUT 引脚序号。
rx-pin	0~3	指定 I2S 所使用的 DIN 引脚序号。
dai-type	“I2S”，“HDMI”	指定接口用作 I2S 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。
slot-num	1~16	slot 数量（可简单理解为 支持最大通道数。）
slot-width	8, 16, 24, 32	单个 slot 宽度（可简单理解为 支持最大数据精度。）
mclk-fp	注释为 false, 反之为 ture	ture: mclk 以固定频段输出。 false: mclk 以采样率倍数输出。
mclk-fs	u32	固定频段：mclk = mclk-fs * 12.288M or 11.2896M. 采样率倍数：mclk = mclk-fs * pcm rate.

2.2.7 AHUB kernel menuconfig 配置说明

linux-4.9 ~ linux-5.4，menuconfig 必选配置如下。

```
Device Drivers  --->
  <*> Sound card support  --->
    <*> Advanced Linux Sound Architecture  --->
      <*> ALSA for SoC audio support  --->
        Allwinner SoC Audio support V2  --->
          <M> Allwinner AHUB support
          <M>  Allwinner HDMIAUDIO Support
```

linux-5.10 及其以上内核版本，menuconfig 必选配置如下。

```
Allwinner BSP  --->
  Device Drivers  --->
    SOUND Drivers  --->
      Platform drivers  --->
        <M> Allwinner AHUB support
        <M>  Allwinner HDMIAUDIO Support
```

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）。

配置项说明

表 2-22: I2S/PCM 混音 AHUB menuconfig 可选配置项

配置项名称	配置项说明
Allwinner AHUB support	AHUB 模块。
Allwinner HDMIAUDIO Support	HDMI AUDIO 模块。
Allwinner function components	功能组件模块。
Components Debug	调试节点功能组件（查看音频寄存器）。
Enable audio dynamic debug	使能 AUDIO 模块 DYNAMIC DEBUG 模式。

2.2.8 加载与卸载方法（ko 方式）

sunxi 音频驱动模块分四大类驱动如下。

- PCM 驱动
- ASoC platfrom 驱动
- ASoC codec 驱动
- ASoC machine 驱动

ko 加载顺序遵循 **“公共组件 -> 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。

AHUB DAM 和 AHUB 声卡加载顺序如下。

```
# 公共组件，提供公共接口
insmod snd_soc_sunxi_common.ko

# PCM 驱动
insmod snd_soc_sunxi_pcm.ko

# ASoC platfrom 驱动 和 ASoC codec 驱动
insmod snd_soc_sunxi_ahub_dam.ko
insmod snd_soc_sunxi_ahub.ko
# HDMI Audio驱动（若使用HDMI Audio）
insmod snd_soc_codec_hdmi.ko

# ASoC machine 驱动
insmod snd_soc_sunxi_machine.ko
```

2.2.9 声卡控件介绍

控件列表

ahub_dam 声卡

```
Mixer name: 'ahubdam'
Number of controls: 13
ctl   type   num    name                value
0     ENUM   1     APBIF0 Src Select          NONE
1     ENUM   1     APBIF1 Src Select          NONE
2     ENUM   1     APBIF2 Src Select          NONE
3     ENUM   1     I2S0 Src Select           NONE
4     ENUM   1     I2S1 Src Select           NONE
5     ENUM   1     I2S2 Src Select           NONE
6     ENUM   1     I2S3 Src Select           NONE
7     ENUM   1     DAM0C0 Src Select           NONE
8     ENUM   1     DAM0C1 Src Select           NONE
9     ENUM   1     DAM0C2 Src Select           NONE
10    ENUM   1     DAM1C0 Src Select           NONE
11    ENUM   1     DAM1C1 Src Select           NONE
12    ENUM   1     DAM1C2 Src Select           NONE

# value 可选项均为以下10个
# NONE
# APBIF_TXDIF0 APBIF_TXDIF1 APBIF_TXDIF2
# I2S0_TXDIF I2S1_TXDIF I2S2_TXDIF I2S3_TXDIF
# DAM0_TXDIF DAM1_TXDIF
```

表 2-23: 控件说明

控件名称	功能	数值
APBIF0 Src Select	APBIF0 数据源选择	0~9(对应控件 value 枚举值)
APBIF1 Src Select	APBIF1 数据源选择	0~9(对应控件 value 枚举值)
APBIF2 Src Select	APBIF2 数据源选择	0~9(对应控件 value 枚举值)
I2S0 Src Select	I2S0 数据源选择	0~9(对应控件 value 枚举值)
I2S1 Src Select	I2S1 数据源选择	0~9(对应控件 value 枚举值)

控件名称	功能	数值
I2S2 Src Select	I2S2 数据源选择	0~9(对应控件 value 枚举值)
I2S3 Src Select	I2S3 数据源选择	0~9(对应控件 value 枚举值)
DAM0C0 Src Select	DAM0C0 数据源选择	0~9(对应控件 value 枚举值)
DAM0C1 Src Select	DAM0C1 数据源选择	0~9(对应控件 value 枚举值)
DAM0C2 Src Select	DAM0C2 数据源选择	0~9(对应控件 value 枚举值)
DAM1C0 Src Select	DAM1C0 数据源选择	0~9(对应控件 value 枚举值)
DAM1C1 Src Select	DAM1C1 数据源选择	0~9(对应控件 value 枚举值)
DAM1C2 Src Select	DAM1C2 数据源选择	0~9(对应控件 value 枚举值)

ahubi2s(n) 声卡 (以 ahubi2s0 为例)

```
Mixer name: 'ahubi2s0'
Number of controls: 1
ctl    type    num    name                value
0      BOOL    1      loopback debug      Off
```

表 2-24: 控件说明

控件名称	功能	数值
loopback debug	内部回录开关	Off;On

ahubhdmi 声卡

```
Mixer name: 'ahubhdmi'
Number of controls: 2
ctl    type    num    name                value
0      ENUM    1      audio data format   PCM
1      BOOL    1      loopback debug      Off

# 控件 0 value 可选项如下。
# NULL PCM AC3 MPEG1 MP3 MPEG2 AAC DTS ATRAC ONE_BIT_AUDIO DOLBY_DIGITAL_PLUS DTS_HD MAT DST WMAPRO
```

表 2-25: 控件说明

控件名称	功能	数值
audio data format	设置音频数据格式	0~14(对应控件 value 枚举值)
loopback debug	内部回录开关	Off;On

2.2.10 常见使用方法

> 以 tinyalsa 工具举例说明，具体使用方法见tinyalsa 工具 章节。

确认声卡序号

```
cat /proc/asound/cards
0 [ahubdam   ]: ahubdam - ahubdam
          ahubdam
1 [ahubi2s0  ]: ahubi2s0 - ahubi2s0
          ahubi2s0
2 [ahubhdmi  ]: ahubhdmi - ahubhdmi
          ahubhdmi
3 [ahubi2s2  ]: ahubi2s2 - ahubi2s2
          ahubi2s2
```

无混音播录

无混音播录情况下，ahubdam 声卡控件保持默认，不做更改。

```
# ahubdam 声卡控件默认值
Mixer name: 'ahubdam'
Number of controls: 13
ctl    type    num    name                    value

0      ENUM    1      APBIF0 Src Select       I2S0_TXDIF
1      ENUM    1      APBIF1 Src Select       I2S1_TXDIF
2      ENUM    1      APBIF2 Src Select       I2S2_TXDIF
3      ENUM    1      I2S0 Src Select         APBIF_TXDIF0
4      ENUM    1      I2S1 Src Select         APBIF_TXDIF1
5      ENUM    1      I2S2 Src Select         APBIF_TXDIF2
6      ENUM    1      I2S3 Src Select         NONE
7      ENUM    1      DAM0C0 Src Select       NONE
8      ENUM    1      DAM0C1 Src Select       NONE
9      ENUM    1      DAM0C2 Src Select       NONE
10     ENUM    1      DAM1C0 Src Select       NONE
11     ENUM    1      DAM1C1 Src Select       NONE
12     ENUM    1      DAM1C2 Src Select       NONE

# ahubi2s0 声卡录音，2channel 16bit 48000hz
tinycap test.wav -D 1 -c 2 -b 16 -r 48000 -T 10

# ahubi2s0 声卡播放
tinyplay test.wav -D 1

# ahubi2s0 声卡内部回环，播录音频格式需保持一致
tinymix -D 1 "loopback debug" 1
tinyplay test_play.wav -D 1 &
tinycap test_cap.wav -D 1 -c 2 -b 16 -r 48000 -T 10

# ahubhdmi 声卡播放（PCM 数据格式）
tinymix -D 2 "audio data format" PCM
tinyplay test.wav -D 2

# ahubhdmi 声卡播放（透传数据格式，根据所需格式设置，以DTS格式为例）
tinymix -D 2 "audio data format" DTS
tinyplay test.wav -D 2
```

说明

ahubhdmi 声卡透传数据格式播放时，需所接的外部 HDMI 设备支持所选透传格式播放。

混音播录

> 前提条件 1：假设按照默认配置 I2S 和 APB 绑定关系如下。
>
> - APB0 <-> I2S0
> - APB1 <-> I2S1(HDMI AUDIO)
> - APB2 <-> I2S2
>
> 前提条件 2：所有混音的音频格式 (channel,bit,rate) 均需保持一致

混音播录情况下，ahubdam 声卡控件需根据实际需求更改，此处只做部分示例说明。

- 使用 DAM0 混音器实现 ahubi2s0 播放 + ahubhdmi 播放 + ahubi2s2 播放从 ahubhdmi 混音播放

```
# 通路需求设置
# Playback ——> APBIF0_TX ——> DAM0C0 ——> DAM0_TX ——> I2S1(HDMI AUDIO)
# Playback ——> APBIF1_TX ——> DAM0C1 ——^
# Playback ——> APBIF2_TX ——> DAM0C2 ——^

# 设置 DAM0 混音器数据源
tinymix -D 0 "DAM0C0 Src Select" APBIF_TXDIF0
tinymix -D 0 "DAM0C1 Src Select" APBIF_TXDIF1
tinymix -D 0 "DAM0C2 Src Select" APBIF_TXDIF2

# 设置 I2S1(HDMI AUDIO) 播放所需数据源
tinymix -D 0 "I2S1 Src Select" DAM0_TXDIF

# test1.wav test2.wav test3.wav 从 ahubhdmi 声卡混音播放
tinyplay test1.wav -D 1
tinyplay test2.wav -D 2
tinyplay test3.wav -D 3
```

- 使用 DAM0 混音器实现 ahubi2s0 录音 + ahubhdmi 播放 + ahubi2s2 播放从 ahubhdmi 混音播放

```
# 通路需求设置
# Capture ——> I2S0_RX  ——> DAM0C0 ——> DAM0_TX ——> I2S1(HDMI AUDIO)
# Playback ——> APBIF1_TX ——> DAM0C1 ——^
# Playback ——> APBIF2_TX ——> DAM0C2 ——^

# 设置 DAM0 混音器数据源
tinymix -D 0 "DAM0C0 Src Select" I2S0_TXDIF
tinymix -D 0 "DAM0C1 Src Select" APBIF_TXDIF1
tinymix -D 0 "DAM0C2 Src Select" APBIF_TXDIF2

# 设置 I2S1(HDMI AUDIO) 播放所需数据源
tinymix -D 0 "I2S1 Src Select" DAM0_TXDIF

# test1.wav test2.wav test3.wav 从 ahubhdmi 声卡混音播放
```

```
tinycap test1.wav -D 1 -c 2 -b 16 -r 48000 -T 10
tinyplay test2.wav -D 2
tinyplay test3.wav -D 3
```

2.3 DMIC

2.3.1 Device Tree 配置

2.3.1.1 配置路径

设备树中定义的是该类芯片对应于 IC 规格的所有配置，设备树文件路径如下：

- linux-4.9、linux-5.4：

> 32 位平台：kernel/{KERNEl_VER}/arch/arm/boot/dts/{CHIP}.dtsi
> 64 位平台：kernel/{KERNEl_VER}/arch/arm64/boot/dts/sunxi/{CHIP}.dtsi

- linux-5.10（含 linux-5.10）之后：

> 32/64 位平台：bsp/configs/{KERNEl_VER}/{CHIP}.dtsi

说明

1. **{KERNEl_VER} 为内核版本，如 linux-5.15；**
2. **{CHIP}.dtsi 为具体芯片型号，如 sun55iw3p1.dtsi。**

2.3.1.2 配置示例

```
dmic_plat:dmic_plat@{module_base_reg} {
  #sound-dai-cells = <0>;
  compatible    = "allwinner,sunxi-snd-plat-dmic";
  reg           = <0x0 {module_base_reg} 0x0 0x50>;
  resets        = <&ccu RST_BUS_DMIC>;
  clocks        = <&ccu CLK_BUS_{module}>,
                  <&ccu CLK_PLL_AUDIO{n}>,
                  <&ccu CLK_{module}>;
  clock-names   = "clk_bus_{module}",
                  "clk_pll_audio{n}",
                  "clk_{module}";
  dmas          = <&dma1 {DRQ_PORT}>;
  dma-names     = "rx";
  capture-cma   = <128>;
  rx-fifo-size  = <128>;
  status        = "disabled";
```

```
};

dmic_mach:dmic_mach{
    compatible          = "allwinner,sunxi-snd-mach";
    soundcard-mach,name    = "snddmic";
    soundcard-mach,capture-only;
    status = "disabled";
    soundcard-mach,cpu {
        sound-dai = <&dmic_plat>;
    };
    soundcard-mach,codec {
    };
};
```

说明

本部分仅为示例，具体内容根据实际情况填写。

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_plat

表 2-26: DMIC dmic_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 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。
rx-fifo-size	设置录音流的 fifo_size 大小，用于声卡参数限定，单位 Kb。
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: dmic_mach

表 2-27: DMIC dmic_mach 节点配置项

配置项名称	配置项说明
soundcard-mach, name	machine 层配置前缀。 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）。

配置项名称	配置项说明
capture-only	设置仅录音，不进行播放流设备创建。
pll-fs	指定模块时钟源频率（24.576M or 22.5792M * pll-fs）。

2.3.2 board.dts 配置

2.3.2.1 配置路径

board.dts 用于保存每一个板级平台的设备信息（如 demo 板，perf1 板等），里面的配置信息会覆盖上面的 Device Tree 中 dtsi 默认配置信息。不同 IC、版型及内核版本对应的 board.dts 具体路径如下。

```
device/config/chips/{PLATFORM}/configs/{BOARD}/{KERNEL_VER}/board.dts
```

2.3.2.2 配置示例

```
&dmic_plat {
    rx-chmap        = <0x76543210>;
    data-vol        = <0xB0>;
    rxdelaytime     = <0>;
    /* pinctrl-used; */
    /* pinctrl-names  = "default","sleep"; */
    /* pinctrl-0      = <&dmic_pins_a>; */
    /* pinctrl-1      = <&dmic_pins_b>; */
    rx-sync-en;
    status          = "disabled";
};

&dmic_mach {
    status      = "disabled";
    soundcard-mach,cpu {
        sound-dai = <&dmic_plat>;
    };
    soundcard-mach,codec {
    };
};
```

说明

gpio 部分请参考附件GPIO 功能复用配置

配置项说明

表 2-28: 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-chmap	u32(默认值为 0x76543210)	四条 DATA 线的通道映射。

2.3.3 DMIC kernel menuconfig 配置说明

linux-5.10 以下内核版本，menuconfig 必选配置如下。

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

linux-5.10 及其以上内核版本，menuconfig 必选配置如下。

```
Allwinner BSP  --->
  Device Drivers  --->
    SOUND Drivers  --->
      Platform drivers  --->
        <M> Allwinner DMIC support
```

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）。

配置项说明

表 2-29: DMIC menuconfig 可选配置项

配置项名称	配置项说明
Allwinner DMIC support	DMIC 模块。
Allwinner function components	功能组件模块。
Components Debug	调试节点功能组件（查看音频寄存器）。
Enable audio dynamic debug	使能 AUDIO 模块 DYNAMIC DEBUG 模式。

2.3.4 加载与卸载方法（ko 方式）

sunxi 音频驱动模块分四大类驱动如下。

- PCM 驱动

- ASoC platfrom 驱动
- ASoC codec 驱动
- ASoC machine 驱动

ko 加载顺序遵循 **“公共组件 -> 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。

```
# 公共组件，提供公共接口
insmod snd_soc_sunxi_common.ko

# PCM 驱动
insmod snd_soc_sunxi_pcm.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_machine.ko
```

2.3.5 声卡控件介绍

```
Mixer name: 'snddmic'
Number of controls: 9
ctl   type   num    name                value
0     ENUM   1      rx sync mode        >Off On
1     INT    1      L0 volume           176 (dsrange 0->255)
2     INT    1      R0 volume           176 (dsrange 0->255)
3     INT    1      L1 volume           176 (dsrange 0->255)
4     INT    1      R1 volume           176 (dsrange 0->255)
5     INT    1      L2 volume           176 (dsrange 0->255)
6     INT    1      R2 volume           176 (dsrange 0->255)
7     INT    1      L3 volume           176 (dsrange 0->255)
8     INT    1      R3 volume           176 (dsrange 0->255)
```

表 2-30: 控件说明

控件名称	功能	数值
rx sync mode	同源播放开关	Off;On
L0 volume	DIN0 左声道音量调节	0->255 (-119.25->71.25dB)
R0 volume	DIN0 右声道音量调节	0->255 (-119.25->71.25dB)
L1 volume	DIN1 左声道音量调节	0->255 (-119.25->71.25dB)
R1 volume	DIN1 右声道音量调节	0->255 (-119.25->71.25dB)
L2 volume	DIN2 左声道音量调节	0->255 (-119.25->71.25dB)
R2 volume	DIN2 右声道音量调节	0->255 (-119.25->71.25dB)
L3 volume	DIN3 左声道音量调节	0->255 (-119.25->71.25dB)
R3 volume	DIN3 右声道音量调节	0->255 (-119.25->71.25dB)

2.3.6 常用使用方法

> 以 tinyalsa 工具举例说明，具体使用方法见tinyalsa 工具 章节。

录音

```
# 确认声卡序号
cat /proc/asound/cards
5 [sndi2s0    ]: sndhdmi - sndhdmi
                 sndhdmi

# 2channel 16bit 48000rate
tinycap test.wav -D 5 -c 2 -b 16 -r 48000 -T 10

# 8channel 16bit 48000rate
tinycap test.wav -D 5 -c 8 -b 16 -r 48000 -T 10
```

2.4 OWA

2.4.1 Device Tree 配置

2.4.1.1 配置路径

设备树中定义的是该类芯片对应于 IC 规格的所有配置，设备树文件路径如下：

- linux-4.9、linux-5.4：

> 32 位平台：kernel/{KERNEl_VER}/arch/arm/boot/dts/{CHIP}.dtsi
> 64 位平台：kernel/{KERNEl_VER}/arch/arm64/boot/dts/sunxi/{CHIP}.dtsi

- linux-5.10（含 linux-5.10）之后：

> 32/64 位平台：bsp/configs/{KERNEl_VER}/{CHIP}.dtsi

📖 说明

1. **{KERNEl_VER} 为内核版本，如 linux-5.15;**
2. **{CHIP}.dtsi 为具体芯片型号，如 sun50iw10p1.dtsi。**

2.4.1.2 配置示例

```
owa_plat:owa_plat@{module_base_reg} {
  #sound-dai-cells = <0>;
  compatible    = "allwinner,sunxi-snd-plat-owa";
  reg           = <0x0 {module_base_reg} 0x0 0x58>;
  resets        = <&ccu RST_BUS_OWA>;
  clocks        = <&ccu CLK_BUS_{module}>,
                  <&ccu CLK_PLL_DSP_AUDIO{n}>,
                  <&ccu CLK_{modele}>;
  clock-names   = "clk_bus_owa",
                  "clk_pll_audio{n}",
                  "clk_{module}";
  dmas          = <&dma1 {DRQ_PORT}>, <&dma1 {DRQ_PORT}>;
  dma-names     = "tx", "rx";
  playback-cma  = <128>;
  capture-cma   = <128>;
  tx-fifo-size  = <128>;
  rx-fifo-size  = <128>;
  status        = "disabled";
};

owa_mach:owa_mach {
  compatible          = "allwinner,sunxi-snd-mach";
  soundcard-mach,name = "sndowa";
  status              = "disabled";
  soundcard-mach,cpu {
    sound-dai = <&owa_plat>;
  };
  soundcard-mach,codec {
  };
};
```

说明

本部分仅为示例，具体内容根据实际情况填写。

配置项说明

OWA 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: owa_plat

表 2-31: OWA owa_plat 节点配置项 (linux-5.10)

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 OWA 寄存器起始地址和地址长度。
resets	设置 OWA 所需的复位时钟。
clocks	设置 OWA 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。

配置项名称	配置项说明
capture-cma	设置录音流 DMA 申请 size 大小，为 (2^n)Kbyte，单位 Kb。
tx-fifo-size	设置播放流的 fifo_size 大小，用于声卡参数限定，单位 Kb。
rx-fifo-size	设置录音流的 fifo_size 大小，用于声卡参数限定，单位 Kb。
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: owa_mach

表 2-32: OWA owa_mach 节点配置项 (linux-5.10)

配置项名称	配置项说明
soundcard-mach,	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）。
pll-fs	指定模块时钟源频率（24.576M or 22.5792M * pll-fs）。

2.4.2 board.dts 配置

2.4.2.1 配置路径

board.dts 用于保存每一个板级平台的设备信息（如 demo 板，perf1 板等），里面的配置信息会覆盖上面的 Device Tree 中 dtsi 默认配置信息。不同 IC、版型及内核版本对应的 board.dts 具体路径如下。

> device/config/chips/{PLATFORM}/configs/{BOARD}/{KERNEl_VER}/board.dts

2.4.2.2 配置示例

```
&owa_plat {
    /* pinctrl-used; */
    /* pinctrl-names   = "default","sleep"; */
    /* pinctrl-0       = <&owa_pins_a>; */
    /* pinctrl-1       = <&owa_pins_b>; */
    tx-hub-en;
    status          = "disabled";
};
```

```
&owa_mach {
  status          = "disabled";
  soundcard-mach,cpu {
    sound-dai   = <&owa_plat>;
  };
  soundcard-mach,codec {
  };
};
```

说明

gpio 部分请参考附件GPIO 功能复用配置

配置项说明

表 2-33: OWA 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
tx-hub-en	注释为 false, 反之为 ture	选择是否注册 txhub 控件

2.4.3 OWA-IN 与 OWA-OUT 调试说明

1、配置 owa-in 时，只需从原理图中查询到所使用的 GPIO，完成 board.dts 中的 pinctrl 配置，且开机后可正常注册 owa 声卡即可使用；配置 owa-out 同理；

```
&pio {
  /* 假设PH6为owa-in，PH7为owa-out */
  owa_pins_a: owa@0 {
    pins = "PH6", "PH7";
    function = "owa";
    drive-strength = <20>;
    bias-disable;
  };

  owa_pins_b: owa@1 {
    pins = "PH6", "PH7";
    function = "io_disabled";
    drive-strength = <20>;
    bias-disable;
  };
};

&owa_plat {
  pinctrl-used;
  pinctrl-names   = "default","sleep";
  pinctrl-0       = <&owa_pins_a>;
  pinctrl-1       = <&owa_pins_b>;
  tx-hub-en;
  status          = "okay";
};

&owa_mach {
```

```
    status          = "okay";
    soundcard-mach,cpu {
        sound-dai  = <&owa_plat>;
    };
    soundcard-mach,codec {
    };
};
```

2、测试 owa 输入输出功能时，无需打开任何控件，直接使用 tinyplay 或 tinycap 即可；

3、owa-in 若无法采集数据，一是需要测量 owa-in 端口是否有信号，二是需确认输入数据格式是否符合 IEC-60958 或 IEC-61937 协议格式；

3、调试 soc 端的 owa 功能是否正常，可短接 soc 的 owa-in 与 owa-out 引脚，先执行 tinyplay 使用 owa 声卡播放，再执行 tinycap 使用 owa 声卡录音，若能够录到正常声音，说明 soc 端的 owa 配置是正常的；

4、若播放时出现声音播放速度明显变慢或变快的情况，可能是时钟配置错误导致，此时请提 AService 咨询全志工程师处理。

2.4.4 OWA kernel menuconfig 配置说明

linux-5.10 以下内核版本，menuconfig 必选配置如下。

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

linux-5.10 及其以上内核版本，menuconfig 必须配置如下。

```
Allwinner BSP  --->
  Device Drivers  --->
    SOUND Drivers  --->
      Platform drivers  --->
        <M> Allwinner OWA Support
```

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）。

配置项说明

表 2-34: OWA menuconfig 可选配置项

配置项名称	配置项说明
Allwinner OWA support	OWA 模块。
Components Rx Sync	同步采样功能组件。
Allwinner function components	功能组件模块。
Components Debug	调试节点功能组件（查看音频寄存器）。

配置项名称	配置项说明
Enable audio dynamic debug	使能 AUDIO 模块 DYNAMIC DEBUG 模式。

2.4.5 加载与卸载方法（ko 方式）

sunxi 音频驱动模块分四大类驱动如下。

- PCM 驱动
- ASoC platfrom 驱动
- ASoC codec 驱动
- ASoC machine 驱动

ko 加载顺序遵循 **“公共组件 -> 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。

OWA 声卡加载顺序如下。

```
# 公共组件，提供公共接口
insmod snd_soc_sunxi_common.ko

# PCM 驱动
insmod snd_soc_sunxi_pcm.ko

# ASoC platfrom 驱动和 ASoC codec 驱动(NULL)
insmod snd_soc_sunxi_owa.ko

# ASoC machine 驱动
insmod snd_soc_sunxi_machine.ko
```

2.4.6 声卡控件

控件列表

```
Mixer name: 'sndowa'
Number of controls: 9
ctl    type    num    name                    value
      range/values
0      ENUM    1      tx hub mode               >Off On
1      BOOL    1      loopback debug              Off
2      ENUM    1      audio tx data format          >PCM RAW
3      ENUM    1      audio rx data format          >PCM RAW
4      ENUM    1      rx channel cnt              >0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
5      ENUM    1      rx word length               >Null 16bits 17bits 18bits 19bits 20bits 21bits 22bits 23bits 24bits
6      ENUM    1      rx rate                   >Null 22.05kHz 24kHz 32kHz 44.1kHz 48kHz 96kHz 176.4kHz 192kHz 768kHz
7      ENUM    1      rx origin rate              >Null 8kHz 11.025kHz 12kHz 16kHz 22.05kHz 24kHz 32kHz 44.1kHz 48kHz
       88.2kHz 96kHz 176.4kHz 192kHz
8      ENUM    1      rx data type              >Linear_PCM Nonlinear_PCM
```

表 2-35: 控件说明

控件名称	功能	数值
tx hub mode	同源播放开关	Off;On
loopback debug	内部回录开关	Off;On
audio tx data format	设置播放音频数据格式	0~1(对应控件 value 枚举值)
audio rx data format	设置录音音频数据格式	0~1(对应控件 value 枚举值)
rx channel cnt	获取当前录音通道数（IEC 格式信息）	0~15（只读）
rx word length	获取当前录音数据位深（IEC 格式信息）	16~24bits（只读）
rx rate	获取当前录音采样率（IEC 格式信息）	8~768kHz（只读）
rx origin rate	获取当前录音采样率（IEC 格式信息）	8~192kHz（只读）
rx data type	获取当前录音数据格式（IEC 格式信息）	Linear_PCM;Nonlinear_PCM（只读）

2.4.7 常用使用方法

> 以 tinyalsa 工具举例说明，具体使用方法见tinyalsa 工具 章节。

播放

```
# 确认声卡序号
cat /proc/asound/cards
1 [sndi2s0    ]: sndowa - sndowa
                 sndowa

tinyplay test.wav -D 1

#透传播放
tinymix -D 0 "audio tx data format" RAW
tinyplay test.dts -D 1
```

2.5 HDMI EDP AV

2.5.1 board.dts 配置

2.5.1.1 配置路径

board.dts 用于保存每一个板级平台的设备信息（如 demo 板，perf1 板等），里面的配置信息会覆盖上面的 Device Tree 中 dtsi 默认配置信息。不同 IC、版型及内核版本对应的 board.dts 具体路径如下。

> device/config/chips/{PLATFORM}/configs/{BOARD}/{KERNEL_VER}/board.dts

2.5.1.2 HDMI TX 配置示例

```
hdmi_codec:hdmi_codec {
    #sound-dai-cells = <0>;
    compatible  = "allwinner,sunxi-snd-codec-hdmi";
    status = "disabled";
};
&i2s{n}_plat {
    tx-pin        = <0 1 2 3>;
    dai-type      = "hdmi";
    status        = "disabled";
};
&i2s{n}_mach {
    ...
    soundcard-mach,playback-only;
    i2s{n}_cpu: soundcard-mach,cpu {
        sound-dai = <&i2s{n}_plat>;
        soundcard-mach,pll-fs  = <1>;
        soundcard-mach,mclk-fs  = <0>;
    };
    i2s{n}_codec: soundcard-mach,codec {
        sound-dai           = <&hdmi_codec>;
    };
};
```

说明

HDMI TX 连接哪一路 I2S 需根据 I2S SPEC 说明或公版 SDK DTS 示例确定。

配置项说明

HDMI TX 音频组成模块由 3 个设备树节点构建。

1、ASoC 层 codec: hdmi_codec

表 2-36: hdmi_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。
tx-pin	I2S 的各路 DOUT 引脚均与 HDMI 模块连接，为固定配置。
dai-type	指定接口类型为 HDMI，为固定配置。

2、ASoC 层 platform: i2s(n)_plat

表 2-37: i2s(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。

3、ASoC 层 machine: i2s(n)_mach

表 2-38: I2S/PCM i2s(n)_mach 节点配置项

配置项名称	配置项说明
playback-only	HDMI TX 仅用于播放
cpu	machine 层所绑定的 platform 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。

2.5.1.3 HDMI RX 配置示例

```
&i2s{n}_plat {
    ...
};
&i2s{n}_mach {
    ...
    soundcard-mach,capture-only;
    i2s{n}_cpu: soundcard-mach,cpu {
        sound-dai = <&i2s{n}_plat>;
        ...
    };
    i2s{n}_codec: soundcard-mach,codec {
    };
};
```

说明

HDMI RX 连接哪一路 I2S 需根据 I2S SPEC 说明或公版 SDK DTS 示例确定。

配置项说明

HDMI RX 音频组成模块由 2 个设备树节点构建，无需打开 hdmi_codec 节点，根据手册配置正确的 I2S 格式。

1、ASoC 层 platform: i2s(n)_plat

2、ASoC 层 machine: i2s(n)_mach

表 2-39: I2S/PCM i2s(n)_mach 节点配置项

配置项名称	配置项说明
capture-only	HDMI RX 仅用于录音
cpu	machine 层所绑定的 platform 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。

2.5.1.4 EDP TX 配置示例

```
edp_codec:edp_codec {
  #sound-dai-cells = <0>;
  compatible  = "allwinner,sunxi-snd-codec-edp";
  status = "disabled";
};
&i2s{n}_plat {
  ...
};
&i2s{n}_mach {
  ...
  soundcard-mach,playback-only;
  i2s{n}_cpu: soundcard-mach,cpu {
    sound-dai = <&i2s{n}_plat>;
    soundcard-mach,pll-fs  = <4>;
    soundcard-mach,mclk-fs  = <512>;
  };
  i2s{n}_codec: soundcard-mach,codec {
    sound-dai          = <&edp_codec>;
  };
};
```

说明

EDP TX 连接哪一路 I2S 需根据 I2S SPEC 说明或公版 SDK DTS 示例确定。

配置项说明

EDP TX 音频组成模块由 3 个设备树节点构建。

1、ASoC 层 codec: edp_codec

表 2-40: edp_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。

2、ASoC 层 platform: i2s(n)_plat

3、ASoC 层 machine: i2s(n)_mach

表 2-41: I2S/PCM i2s(n)_mach 节点配置项

配置项名称	配置项说明
playback-only	EDP TX 仅用于播放
cpu	machine 层所绑定的 platform 节点（即 platform 层），用 sound-dai 属性指定节点。
pll-fs	指定模块时钟源频率（24.576M or 22.5792M * pll-fs），需设置此值为 4。
mclk-fs	本节点无 mclk-fp 属性，故 mclk 以采样率倍数输出（mclk = mclk-fs * pcm rate），需设置此值为 512。

配置项名称	配置项说明
codec	machine 层所绑定的 codec 节点（即 codec 层），用 sound-dai 属性指定节点。

2.5.1.5 AV TX 配置示例

```
&drm_edp {
    #sound-dai-cells = <0>;
    compatible = "allwinner,drm-edp";
    status = "disabled";
    ...
};
&i2s{n}_plat {
    ...
};
&i2s{n}_mach {
    ...
    soundcard-mach,playback-only;
    i2s{n}_cpu: soundcard-mach,cpu {
        sound-dai = <&i2s{n}_plat>;
        soundcard-mach,pll-fs  = <4>;
        soundcard-mach,mclk-fs  = <512>;
    };
    i2s{n}_codec: soundcard-mach,codec {
        sound-dai          = <&drm_edp>;
    };
};
```

说明

AV TX 连接哪一路 I2S 需根据 I2S SPEC 说明或公版 SDK DTS 示例确定。

配置项说明

AV TX 音频组成模块由 2 个设备树节点构建，另外需要使用 edp 模块的设备树节点辅助构建，AV TX 由 EDP TX 升级而来，供 1903 平台及往后新平台应用。

1、ASoC 层 codec: drm_edp

表 2-42: drm_edp 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。

2、ASoC 层 platform: i2s(n)_plat

3、ASoC 层 machine: i2s(n)_mach

表 2-43: I2S/PCM i2s(n)_mach 节点配置项

配置项名称	配置项说明
playback-only	AV TX 仅用于播放
cpu	machine 层所绑定的 platform 节点（即 platform 层），用 sound-dai 属性指定节点。
pll-fs	指定模块时钟源频率（24.576M or 22.5792M * pll-fs），需设置此值为 4。
mclk-fs	本节点无 mclk-fp 属性，故 mclk 以采样率倍数输出（mclk = mclk-fs * pcm rate），需设置此值为 512。
codec	machine 层所绑定的 codec 节点（即 codec 层），用 sound-dai 属性指定节点。

2.5.2 HDMI TX kernel menuconfig 配置说明

linux-5.10 以下内核版本，menuconfig 配置如下。

```
Device Drivers  --->
  <*> Sound card support  --->
    <*> Advanced Linux Sound Architecture  --->
      <*> ALSA for SoC audio support  --->
        Allwinner SoC Audio support V2  --->
          <M> Allwinner I2S support
          <M>  Allwinner HDMIAUDIO Support
```

linux-5.10 及其以上内核版本，menuconfig 配置如下。

```
Allwinner BSP  --->
  Device Drivers  --->
    SOUND Drivers  --->
      Platform drivers  --->
        <M> Allwinner I2S support
        <M>  Allwinner HDMIAUDIO Support
```

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）。

配置项说明

表 2-44: menuconfig 配置项

配置项名称	配置项说明
Allwinner I2S support	I2S 模块。
Allwinner HDMIAUDIO Support	HDMI TX 模块。

2.5.3 HDMI RX kernel menuconfig 配置说明

linux-5.10 以下内核版本，menuconfig 配置如下。

```
Device Drivers  --->
  <*> Sound card support  --->
    <*> Advanced Linux Sound Architecture  --->
      <*> ALSA for SoC audio support  --->
        Allwinner SoC Audio support V2  --->
          <M> Allwinner I2S support
```

linux-5.10 及其以上内核版本，menuconfig 配置如下。

```
Allwinner BSP  --->
  Device Drivers  --->
    SOUND Drivers  --->
      Platform drivers  --->
        <M> Allwinner I2S support
```

说明

1. **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）。**
2. **HDMI RX 无需打开 Allwinner HDMIAUDIO Support。**

配置项说明

表 2-45: menuconfig 配置项

配置项名称	配置项说明
Allwinner I2S support	I2S 模块。

2.5.4 EDP TX kernel menuconfig 配置说明

该驱动仅支持 linux-5.10 及其以上内核版本，menuconfig 配置如下。

```
Allwinner BSP  --->
  Device Drivers  --->
    SOUND Drivers  --->
      Platform drivers  --->
        <M> Allwinner I2S support
        <M>  Allwinner EDPAUDIO support
```

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）。

配置项说明

表 2-46: menuconfig 配置项

配置项名称	配置项说明
Allwinner I2S support	I2S 模块。

配置项名称	配置项说明
Allwinner EDPAUDIO Support	EDP TX 模块。

2.5.5 AV TX kernel menuconfig 配置说明

该驱动仅支持 linux-5.10 及其以上内核版本，menuconfig 配置如下。

```
Allwinner BSP  --->
  Device Drivers  --->
    SOUND Drivers  --->
      Platform drivers  --->
        <M> Allwinner I2S support
        <M>  Allwinner AVAUDIO support
```

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）。

配置项说明

表 2-47: menuconfig 配置项

配置项名称	配置项说明
Allwinner I2S support	I2S 模块。
Allwinner AVAUDIO Support	AV TX 模块。

2.5.6 加载与卸载方法（ko 方式）

sunxi 音频驱动模块分四大类驱动如下。

- PCM 驱动
- ASoC platfrom 驱动
- ASoC codec 驱动
- ASoC machine 驱动

ko 加载顺序遵循 **“公共组件 -> 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。

```
# 公共组件，提供公共接口
insmod snd_soc_sunxi_common.ko

# PCM 驱动
insmod snd_soc_sunxi_pcm.ko
```

```
# ASoC platfrom 驱动
insmod snd_soc_sunxi_i2s.ko

# HDMI TX Audio驱动（若使用HDMI TX Audio，HDMI RX Audio无需加载）
insmod snd_soc_codec_hdmi.ko
# EDP TX Audio驱动（若使用EDP TX Audio）
insmod snd_soc_codec_edp.ko
# AV TX Audio驱动（若使用AV TX Audio）
insmod snd_soc_codec_av.ko

# ASoC machine 驱动
insmod snd_soc_sunxi_machine.ko
```

2.5.7 声卡控件

控件列表

2.5.7.1 HDMI TX 控件

```
Mixer name: 'sndhdmi'
Number of controls: 3
ctl   type   num   name              value
0     ENUM   1     audio data format     NULL >PCM AC3 MPEG1 MP3 MPEG2 AAC DTS ATRAC ONE_BIT_AUDIO
                                  DOLBY_DIGITAL_PLUS DTS_HD MAT DST WMAPRO
1     ENUM   1     tx hub mode           >Off On
2     BOOL   1     loopback debug        Off
```

表 2-48: 控件说明

控件名称	功能	数值
audio data format	设置音频数据格式	NULL PCM AC3 MPEG1 MP3 MPEG2 AAC DTS ATRAC ONE_BIT_AUDIO DOLBY_DIGITAL_PLUS DTS_HD MAT DST WMAPRO
tx hub mode	同源播放开关	Off;On
loopback debug	内部回录开关	Off;On

说明

HDMI RX or EDP TX or AV TX 均无特殊控件。

2.5.8 常用使用方法

以 tinyalsa 工具举例说明，具体使用方法见tinyalsa 工具 章节；

2.5.8.1 hdmi TX 常见使用方法

```
# 确认声卡序号
cat /proc/asound/cards
4 [sndhdmi    ]: sndhdmi - sndhdmi
                 sndhdmi

# sndhdmi 声卡播放
tinyplay test.wav -D 4

# sndhdmi 数据透传
tinymix -D 4 "audio data format" DTS
tinyplay test.dts -D 4
```

说明

HDMI RX or EDP TX or AV TX 按照普通 I2S 声卡使用即可。

3 驱动架构介绍

3.1 软件框图

3.1.1 ALSA 音频架构

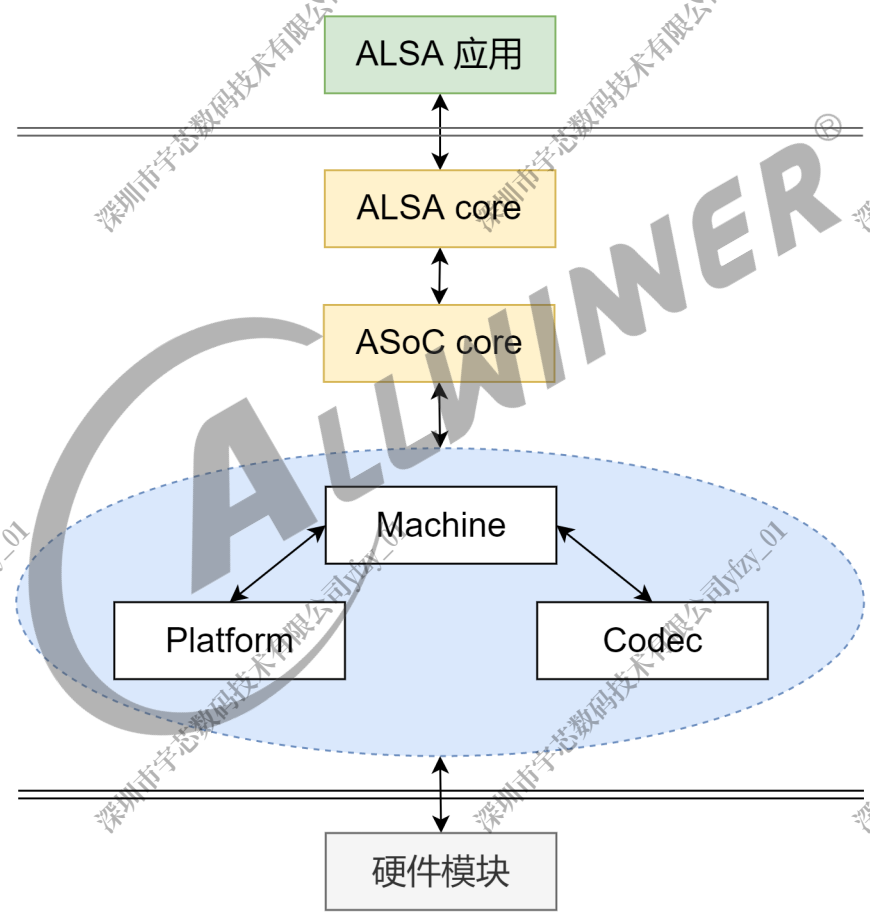

图 3-1: ALSA 音频架构

3.1.2 ASoC 驱动框架

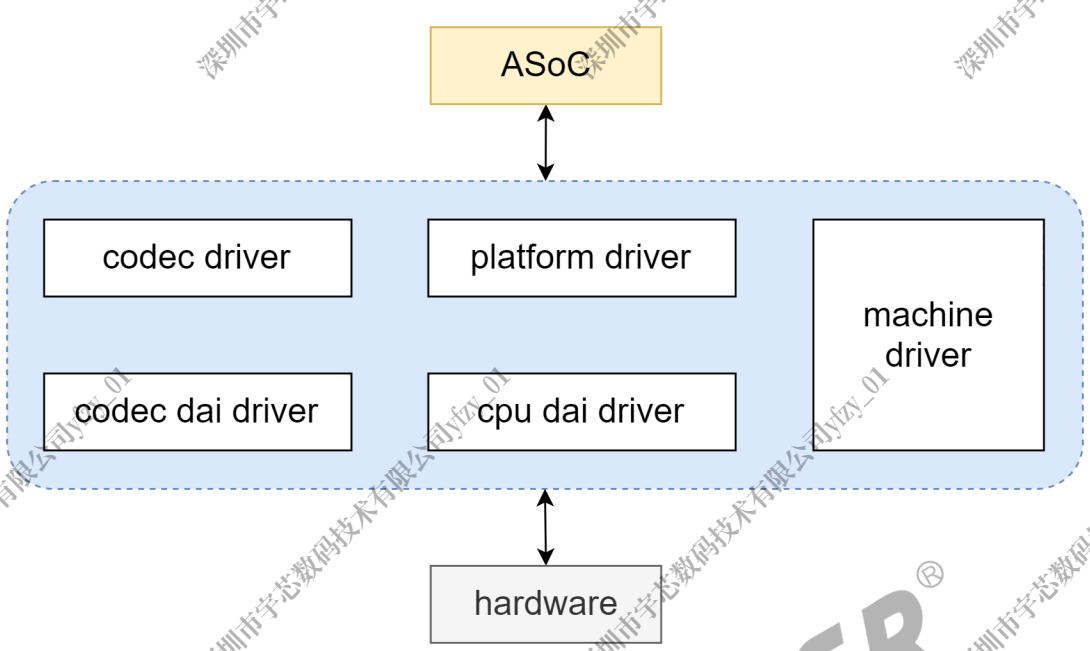

图 3-2: ASoC 驱动框架

3.1.3 基于 ASoC 的 sunxi 驱动框架

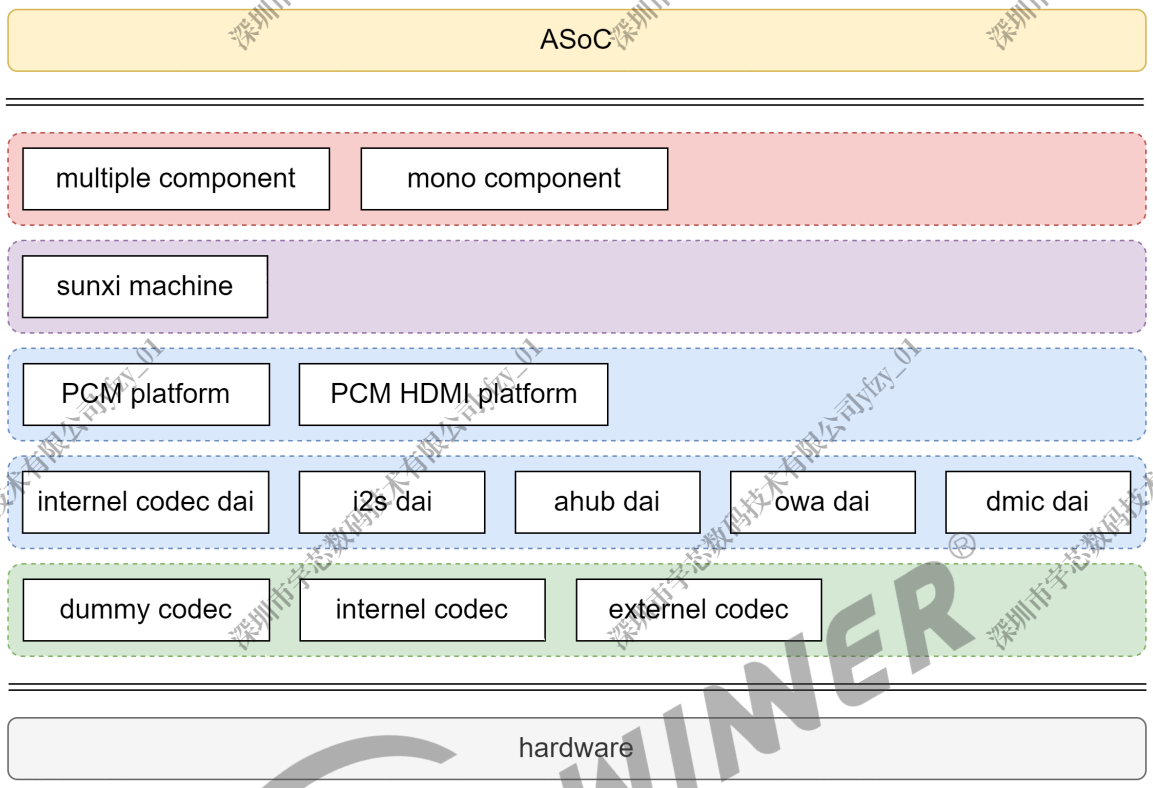

图 3-3: ASoC SUNXI 驱动框架

3.1.4 基于 ASoC 的 sunxi 代码结构

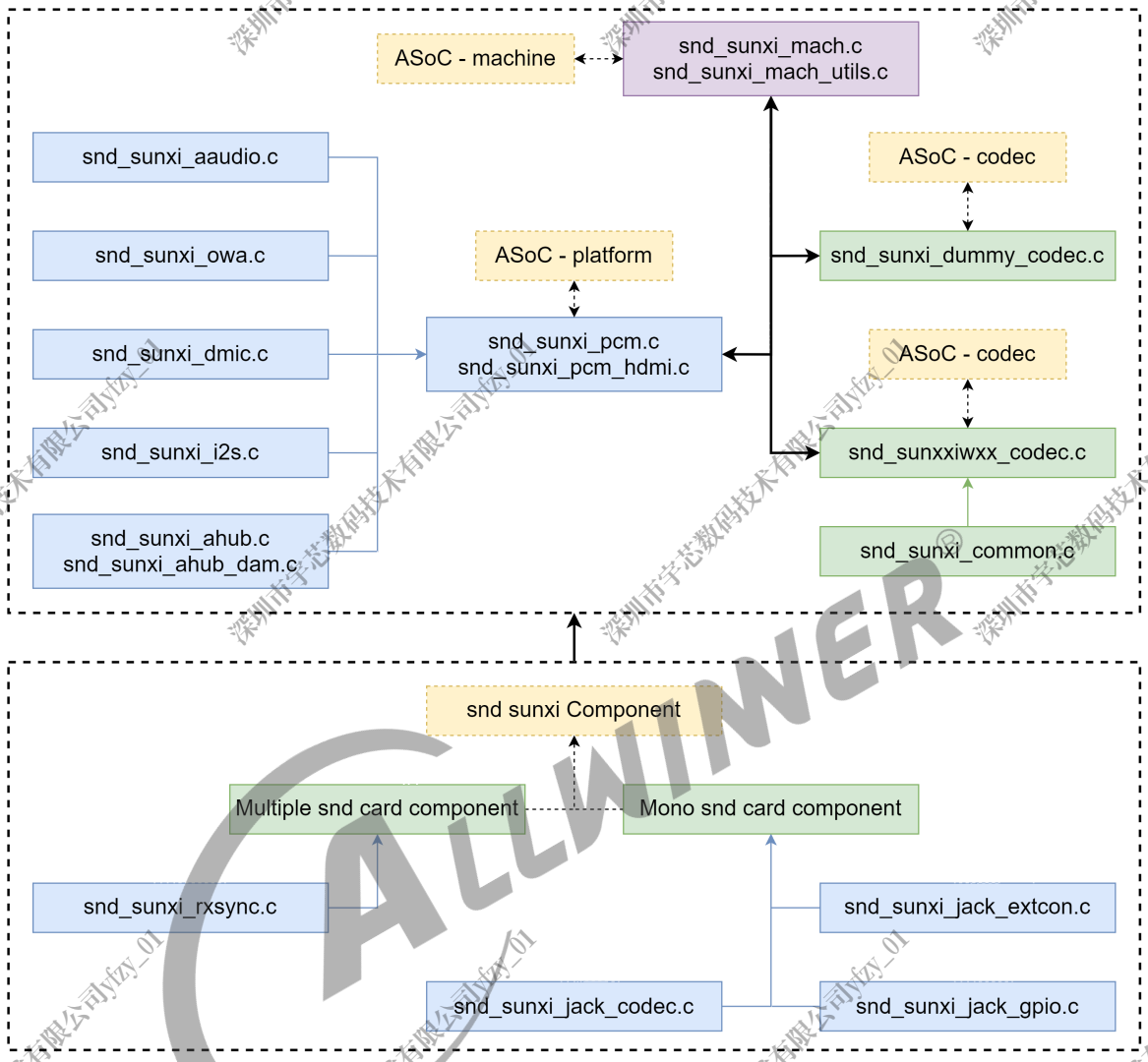

图 3-4: ASoC SUNXI 代码结构

3.2 源码结构介绍

3.2.1 驱动源码

```
.
├── Kconfig
├── Makefile
├── platforms
│   ├── snd_{CHIP}_i2s.c
│   ├── snd_{CHIP}_dmic.c
│   └── snd_{CHIP}_owa.c
└── snd_{CHIP}_codec.c
```

```
├── snd_{CHIP}_codec.h
├── snd_sunxi_aaudio.c
├── snd_sunxi_ahub.c
├── snd_sunxi_ahub_dam.c
├── snd_sunxi_ahub_dam.h
├── snd_sunxi_ahub.h
├── snd_sunxi_codec_edp.c
├── snd_sunxi_codec_hdmi.c
├── snd_sunxi_codec_av.c
├── snd_sunxi_common.c
├── snd_sunxi_common.h
├── snd_sunxi_dap.c
├── snd_sunxi_dap.h
├── snd_sunxi_dmic.c
├── snd_sunxi_dmic.h
├── snd_sunxi_dummy_codec.c
├── snd_sunxi_i2s.c
├── snd_sunxi_i2s.h
├── snd_sunxi_jack_advance.c
├── snd_sunxi_jack.c
├── snd_sunxi_jack_codec.c
├── snd_sunxi_jack_extcon.c
├── snd_sunxi_jack_gpio.c
├── snd_sunxi_jack.h
├── snd_sunxi_log.h
├── snd_sunxi_mach.c
├── snd_sunxi_mach_utils.c
├── snd_sunxi_mach_utils.h
├── snd_sunxi_owa.c
├── snd_sunxi_owa.h
├── snd_sunxi_owa_rx61937.c
├── snd_sunxi_owa_rx61937.h
├── snd_sunxi_pcm.c
├── snd_sunxi_pcm.h
├── snd_sunxi_rxsync.c
├── snd_sunxi_rxsync.h
├── snd_sunxi_sfx.c
└── snd_sunxi_sfx.h
```

3.2.2 源码说明

3.2.2.1 platform 层 –> 公共部分

```
snd_sunxi_pcm.c
snd_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。

```
snd_sunxi_hdmi.c
snd_sunxi_hdmi.h
snd_sunxi_pcm.h
```

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

3.2.2.2 platform 层 –> AudioCodec

```
snd_sunxi_aaudio.c
```

负责 AudioCodec 模块 DMA 相关配置。

3.2.2.3 platform 层 –> I2S/PCM

```
snd_sunxi_i2s.c
snd_sunxi_i2s.h
```

负责 I2S/PCM 模块硬件参数、DMA 相关配置。

3.2.2.4 platform 层 –> AHUB

```
snd_sunxi_ahub.c
snd_sunxi_ahub.h
snd_sunxi_ahub_dam.c
snd_sunxi_ahub_dam.h
```

负责 AHUB 模块硬件参数、DMA 相关配置。

3.2.2.5 platform 层 –> OWA

```
snd_sunxi_owa.c
snd_sunxi_owa.h
```

负责 OWA 模块硬件参数、DMA 相关配置。

3.2.2.6 platform 层 –> DMIC

```
snd_sunxi_dmic.c
snd_sunxi_dmic.h
```

负责 DMIC 模块硬件参数、DMA 相关配置。

3.2.2.7 codec 层 –> 公共部分

```
snd_sunxi_common.c
snd_sunxi_common.h
```

负责 AudioCodec 模块公共功能配置。

- 外部功放控制

3.2.2.8 codec 层 –> AudioCodec

```
snd_{CHIP}_codec.c
snd_{CHIP}_codec.h
```

负责 {CHIP} 平台 AudioCodec 模块硬件参数配置。

3.2.2.9 machine 层

```
snd_sunxi_mach.c
snd_sunxi_mach.h
snd_sunxi_mach_utils.c
snd_sunxi_mach_utils.h
```

负责 platform 层和 codec 层绑定。

3.2.2.10 特殊功能组件

```
snd_sunxi_rxsync.c
snd_sunxi_rxsync.h
```

负责多声卡同步录音。

```
snd_sunxi_jack_codec.c
snd_sunxi_jack.h
```

复杂内置 AudioCodec 耳机插拔和耳机按键检测。

```
snd_sunxi_jack_extcon.c
snd_sunxi_jack.h
```

复杂 extcon 事件耳机插拔和耳机按键检测。

说明

暂仅支持 typec 接口模拟耳机。

3.2.2.11 平台基础资源

```
platforms/snd_{CHIP}_i2s.h    --> {CHIP}平台I2S资源配置。
platforms/snd_{CHIP}_dmic.h   --> {CHIP}平台DMIC资源配置。
platforms/snd_{CHIP}_owa.h    --> {CHIP}平台OWA资源配置。
```

3.3 关键数据结构

> 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

sunxi_dma_params

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

定义 audio dai dma 相关参数。

3.3.2 platform 类结构体

sunxi_i2s

```
struct sunxi_i2s {
    ...
};
```

I2S/PCM 模块总结构体，包含基础平台资源、特定功能私有参数。

sunxi_owa

```
struct sunxi_owa {
    ...
};
```

OWA 模块总结构体，包含基础平台资源、特定功能私有参数。

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

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。

3.3.4 machine 类结构体

asoc_simple_priv

```
struct asoc_simple_priv {
    . . .
};
```

Machine 总结构体，包含 codec dai、cpu dai、特定功能私有参数。

3.4 接口说明

> 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

sunxi_pcm_new {linux-4.9~linux-5.4}

- 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

sunxi_pcm_construct {linux-5.10~linux-5.15}

- 函数原型：

```
int sunxi_pcm_construct(struct snd_soc_component *component, struct snd_soc_pcm_runtime *rtd)
```

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

sunxi_pcm_free {linux-4.9~linux-5.4}

- 函数原型：

```
void sunxi_pcm_free(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

sunxi_pcm_destruct {linux-5.10 or later}

- 函数原型：

```
void sunxi_pcm_destruct(struct snd_soc_component *component, struct snd_pcm *pcm)
```

- 功能描述：释放 pcm 设备
- 参数说明：
  - component: platform 层组件
  - pcm: pcm 设备
- 返回值：void

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

sunxi_pcm_close

- 函数原型：

```
void sunxi_pcm_close(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

sunxi_pcm_ioctl

- 函数原型：

```
int sunxi_pcm_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

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

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

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

sunxi_pcm_pointer

- 函数原型：

```
snd_pcm_uframes_t sunxi_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

sunxi_pcm_hw_params_raw

- 函数原型：

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

- 功能描述：设置 pcm 设备参数 (for HDMI audio)
- 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

sunxi_pcm_hw_free_raw

- 函数原型：

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

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

sunxi_pcm_prepare_raw

- 函数原型：

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

- 功能描述：触发 pcm 设备运行准备工作 (for HDMI audio)
- 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

sunxi_pcm_trigger_raw

- 函数原型：

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

- 功能描述：触发 pcm 设备运行 (for HDMI audio)
- 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

sunxi_pcm_pointer_raw

- 函数原型：

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

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

sunxi_pcm_copy_raw

- 函数原型：

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

- 功能描述：音频数据传输和透传数据处理 (for HDMI audio)
- 参数说明：
  - substream: pcm 子流信息
- 返回值：当前 DMA 缓冲指针

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

> {module} 表示模块名称，有 aaudio、i2s、ahub、owa、dmic。

sunxi_{module}_component_probe

- 函数原型：

```
int sunxi_{module}_component_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

sunxi_{module}_dai_suspend {linux-4.9}

- 函数原型：

```
int sunxi_{module}_dai_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

sunxi_{module}_component_suspend {linux-5.4 or later}

- 函数原型：

```
int sunxi_{module}_component_suspend(struct snd_soc_component *component)
```

- 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：
  - component: platform 层组件
- 返回值：0-成功，其它-失败

sunxi_{module}_dai_resume {linux-4.9}

- 函数原型：

```
int sunxi_{module}_dai_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

sunxi_{module}_component_resume {linux-5.4 or later}

- 函数原型：

```
int sunxi_{module}_component_resume(struct snd_soc_component *component)
```

- 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：
  - component: platform 层组件
- 返回值：0-成功，其它-失败

sunxi_{module}_dai_probe

- 函数原型：

```
int sunxi_{module}_dai_probe(struct snd_soc_dai *dai)
```

- 功能描述：I2S/PCM 模块接口初始化
- 参数说明：
  - dai: cpu dai 信息
- 返回值：0-成功，其它-失败

sunxi_{module}_dai_remove

- 函数原型：

```
int sunxi_{module}_dai_remove(struct snd_soc_dai *dai)
```

- 功能描述：I2S/PCM 模块接口移除
- 参数说明：

  - dai: cpu dai 信息
- 返回值：0-成功，其它-失败

sunxi_{module}_dai_set_pll

- 函数原型：

```
int sunxi_{module}_dai_set_pll(struct snd_soc_dai *dai, int pll_id, int source,
        unsigned int freq_in, unsigned int freq_out)
```

- 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

sunxi_{module}_dai_set_sysclk

- 函数原型：

```
int sunxi_{module}_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

sunxi_{module}_dai_set_bclk_ratio

- 函数原型：

```
int sunxi_{module}_dai_set_bclk_ratio(struct snd_soc_dai *dai, unsigned int ratio)
```

- 功能描述：设置模块 BCLK 时钟
- 参数说明：
  - dai: cpu dai 信息
  - ratio: BCLK 分频数
- 返回值：0-成功，其它-失败

sunxi_{module}_dai_set_fmt

- 函数原型：

```
int sunxi_{module}_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

sunxi_{module}_dai_set_tdm_slot

- 函数原型：

```
int sunxi_{module}_dai_set_tdm_slot(struct snd_soc_dai *dai, unsigned int tx_mask,
        unsigned int rx_mask, int slots, int slot_width)
```

- 功能描述：设置模块 I2S slot 数和 slot 宽度
- 参数说明：
  - dai: cpu dai 信息
  - tx_mask: tx 掩码
  - rx_mask: rx 掩码
  - slots: I2S slot 数
  - slot_width: I2S slot 宽度
- 返回值：0-成功，其它-失败

sunxi_{module}_dai_startup

- 函数原型：

```
int sunxi_{module}_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

sunxi_{module}_dai_hw_params

- 函数原型：

```
int sunxi_{module}_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

sunxi_{module}_dai_prepare

- 函数原型：

```
int sunxi_{module}_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

sunxi_{module}_dai_trigger

- 函数原型：

```
int sunxi_{module}_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

sunxi_{module}_dai_shutdown

- 函数原型：

```
void sunxi_{module}_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

3.4.3 codec 层接口 –> AudioCodec

sunxi_internal_codec_probe

- 函数原型：

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

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

sunxi_internal_codec_remove

- 函数原型：

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

- 功能描述：模块资源释放
- 参数说明：
  - component: codec 层组件
- 返回值：0-成功，其它-失败

sunxi_internal_codec_suspend {linux-4.9}

- 函数原型：

```
int sunxi_internal_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 层组件
- 返回值：0-成功，其它-失败

sunxi_internal_codec_suspend {linux-5.4~linux-5.15}

- 函数原型：

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

- 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：
  - component: codec 层组件
- 返回值：0-成功，其它-失败

sunxi_internal_codec_resume {linux-4.9}

- 函数原型：

```
int sunxi_internal_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 层组件

- 返回值：0-成功，其它-失败

sunxi_internal_codec_resume {linux-5.4~linux-5.15}

- 函数原型：

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

- 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：
  - component: codec 层组件
- 返回值：0-成功，其它-失败

sunxi_internal_codec_dai_set_pll

- 函数原型：

```
int sunxi_internal_codec_dai_set_pll(struct snd_soc_dai *dai, int pll_id, int source,
        unsigned int freq_in, unsigned int freq_out)
```

- 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

sunxi_internal_codec_dai_startup

- 函数原型：

```
int sunxi_internal_codec_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

sunxi_internal_codec_dai_hw_params

- 函数原型：

```
int sunxi_internal_codec_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

sunxi_internal_codec_dai_prepare

- 函数原型：

```
int sunxi_internal_codec_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

sunxi_internal_codec_dai_trigger

- 函数原型：

```
int sunxi_internal_codec_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

sunxi_internal_codec_dai_shutdown

- 函数原型：

```
void sunxi_internal_codec_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 machine 层接口

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

simple_soc_remove

- 函数原型：

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

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

simple_dai_link_of

- 函数原型：

```
int simple_dai_link_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

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

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

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

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

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

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

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

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

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

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

3.4.5 common 层接口 –> 公共部分

snd_sunxi_pa_pin_init

- 函数原型：

```
struct pa_config *snd_sunxi_pa_pin_init(struct platform_device *pdev, u32 *pa_pin_max)
```

- 功能描述：获取并初始化功放引脚
- 参数说明：
  - pa_pin_max: 功放使能引脚个数
- 返回值：非空-成功，空-失败

snd_sunxi_pa_pin_exit

- 函数原型：

```
void snd_sunxi_pa_pin_exit(struct platform_device *pdev,
        struct pa_config *pa_cfg,
        u32 pa_pin_max)
```

- 功能描述：释放功放引脚资源
- 参数说明：
  - pa_cfg: 功放引脚配置
  - pa_pin_max: 功放使能引脚个数
- 返回值：void

snd_sunxi_pa_pin_enable

- 函数原型：

```
int snd_sunxi_pa_pin_enable(struct pa_config *pa_cfg, u32 pa_pin_max)
```

- 功能描述：使能功放
- 参数说明：
  - pa_cfg: 功放引脚配置
  - pa_pin_max: 功放使能引脚个数
- 返回值：0-成功，其它-失败

snd_sunxi_pa_pin_disable

- 函数原型：

```
int snd_sunxi_pa_pin_disable(struct pa_config *pa_cfg, u32 pa_pin_max)
```

- 功能描述：失能功放
- 参数说明：
  - pa_cfg: 功放引脚配置
  - pa_pin_max: 功放使能引脚个数
- 返回值：0-成功，其它-失败

snd_sunxi_regulator_init

- 函数原型：

```
struct snd_sunxi_rglt *snd_sunxi_regulator_init(struct platform_device *pdev)
```

- 功能描述：电源控制初始化
- 参数说明：
  - pdev: 字符设备
- 返回值：非空-成功，空-失败

snd_sunxi_regulator_exit

- 函数原型：

```
void snd_sunxi_regulator_exit(struct snd_sunxi_rglt *rglt)
```

- 功能描述：电源控制销毁
- 参数说明：
  - rglt: 电源控制句柄
- 返回值：void

snd_sunxi_regulator_enable

- 函数原型：

```
int snd_sunxi_regulator_enable(struct snd_sunxi_rglt *rglt)
```

- 功能描述：电源控制使能

- 参数说明：
  - rglt: 电源控制句柄
- 返回值：0-成功，其它-失败

snd_sunxi_regulator_disable

- 函数原型：

```
void snd_sunxi_regulator_disable(struct snd_sunxi_rglt *rglt)
```

- 功能描述：电源控制失能
- 参数说明：
  - rglt: 电源控制句柄
- 返回值：void

3.4.6 软件调试接口

snd_sunxi_dump_register

- 函数原型：

```
int snd_sunxi_dump_register(struct snd_sunxi_dump *dump)
```

- 功能描述：注册调试
- 参数说明：
  - dump->dump_version: 版本查看函数
  - dump->dump_help: 调试帮助信息显示函数
  - dump->dump_show: 调试显示函数
  - dump->dump_store: 调试输入函数
- 返回值：0-成功，其它-失败

snd_sunxi_dump_unregister

- 函数原型：

```
void snd_sunxi_dump_unregister(struct snd_sunxi_dump *dump)
```

- 功能描述：注销调试
- 参数说明：
  - dump: 调试接口函数集
- 返回值：void

4 测试工具介绍

4.1 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。

4.1.1 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 */
```

- 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
...
```

- 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)
```

- 设置声卡 0 的控件 “lineout volume” 为 26

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

4.1.2 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]
```

- 用声卡 0 播放 test.wav

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

4.1.3 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]
```

- 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
```

4.1.4 tinypcminfo

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

- 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
  ...
```

4.1.5 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]
```

- 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

4.2 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。

4.2.1 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
...
```

- 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: sndi2s0 [sndi2s0], device 0: 2032000.i2s0_plat-snd-soc-dummy-dai snd-soc-dummy-dai-0 []
 Subdevices: 1/1
 Subdevice #0: subdevice #0
```

- 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...
```

4.2.2 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
...
```

- 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: sndi2s0 [sndi2s0], device 0: 2032000.i2s0_plat-snd-soc-dummy-dai snd-soc-dummy-dai-0 []
  Subdevices: 1/1
  Subdevice #0: subdevice #0
...
```

- 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...
```

4.2.3 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
```

- 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
```

- 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%]
...
```

- 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
```

5 FAQ

5.1 调试方法

5.1.1 查看所有声卡

可输入命令 **cat /proc/asound/cards** 查看系统挂载上的声卡.

```
cat /proc/asound/cards

 0 [audiocodec ]: audiocodec - audiocodec
                  audiocodec
 1 [sndowa     ]: sndowa - sndowa
                  sndowa
 2 [snddmic    ]: snddmic - snddmic
                  snddmic
 3 [sndi2s0    ]: sndi2s0 - sndi2s0
                  sndi2s0
 4 [sndhdmi    ]: sndhdmi - sndhdmi
                  sndhdmi
 5 [sndi2s2    ]: sndi2s2 - sndi2s2
                  sndi2s2
 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

1. **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；**
2. **sndi2s0 和 ahubi2s0 均为 i2s 接口，ahub 前缀代表该声卡具备混音功能。**

5.1.2 查看声卡的具体信息

```
# 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
...
```

5.1.3 查看播放设备参数

5.1.3.1 播放设备的硬件参数

```
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
```

5.1.3.2 播放设备的软件参数

```
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
```

5.1.3.3 播放设备的状态

```
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
```

说明

录音设备的信息查看方法类似。

5.1.4 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。

进入调试节点目录

```
cd /sys/class/snd_sunxi
/sys/class/snd_sunxi # ls
dump    help    module  version
```

- version: 查看所使用的各个音频模块版本；
- module: 模块查看和模块选择节点；
- help: 当前模块使用帮助信息；
- dump: 模块操作节点。

使用说明

1. 通过 cat module 确认可用模块，通过 echo {module name} > module 选择模块；
2. 通过 cat help 查看当前模块使用方式；
3. 根据使用帮助信息操作 dump 节点。

示例 1：读写 AudioCodec 模块寄存器

```
# 查看可使用模块
/sys/class/snd_sunxi # cat module
optional modules:
1. AudioCodec
2. I2S0
3. machine-sndi2s0
current module(NULL)

# 选择AudioCodec模块
/sys/class/snd_sunxi # echo AudioCodec > module
```

```
# 查看当前模块帮助信息
/sys/class/snd_sunxi # cat help
 == module help ==
1. get optional modules: cat module
2. set current module  : echo {module name} > module
== current module(AudioCodec) help ==
1. reg read : echo {num} > dump && cat dump
num: 0(all)
2. reg write: echo {reg} {value} > dump
eg. echo 0x00 0xaa > dump

# 根据节点提示说明查看相应寄存器值，如下
/sys/class/snd_sunxi # echo 0 > dump && cat dump
module(AudioCodec)
[0x000]: 0x    100
[0x004]: 0x   1a0a0
[0x010]: 0x    4000
...

# 根据节点提示说明设置相应寄存器值，如下
/sys/class/snd_sunxi # echo 0x4 0x1a0a1 > dump
reg[0x004]: 0x1a0a0 (old)
reg[0x004]: 0x1a0a1 (new)
```

示例 2：修改 I2S 格式

machine-sndi2s{n} 调试节点可修改 I2S 格式。
注：修改 I2S 格式，需要重新播放或录音，播放或录音过程中，I2S 格式修改不会立即生效。

```
# 查看可使用模块
/sys/class/snd_sunxi # cat module
optional modules:
1. AudioCodec
2. I2S0
3. machine-sndi2s0
current module(AudioCodec)

# 选择machine-sndi2s0模块
/sys/class/snd_sunxi # echo machine-sndi2s0 > module

# 查看当前模块帮助信息
/sys/class/snd_sunxi # cat help
== module help ==
1. get optional modules: cat module
2. set current module  : echo {module name} > module
== current module(machine-sndi2s0) help ==
1. get daifmt: echo {num} > dump && cat dump
num: 0(all)
2. set daifmt: echo {Opt num} {Val} > dump
Opt num         Val                     note
1 CPUPLL FS     -> 1~n                    -> 22.5792 or 24.576 * fs MHz
2 CODECPLL FS   -> 1~n                    -> 22.5792 or 24.576 * fs MHz
3 MCLK FP       -> Off On                 -> mclk fs flag
4 MCLK FS       -> 0~n                    -> pcm rate * fs(fp Off), (11.2896 or 12.288)MHz * fs(fp On)
5 FMT           -> i2s right_j left_j dsp_a dsp_b   -> i2s/pcm format config
6 MASTER        -> CBM_CFM CBS_CFM CBM_CFS CBS_CFS  -> bclk&lrck master
7 INVERT        -> NB_NF NB_IF IB_NF IB_IF          -> bclk&lrck invert
8 SLOTS         -> 2~32 (must be 2*n)               -> slot number
9 SLOT WIDTH    -> 16 24 32                         -> slot width
```

```
10 ML SEL      -> RM_TM RM_TL RL_TM RL_TL       -> RX&TX MSB/LSB first select
11 DATA LATE   -> 0~3                           -> data is offset by n BCLKS to LRCK

# 根据节点提示说明查看当前I2S格式，如下
/sys/class/snd_sunxi # echo 0 > dump && cat dump
module(machine-sndi2s0)
CPUPLL FS   -> 1
CODECPLL FS -> 1
MCLK FP     -> On
MCLK FS     -> 1
FMT         -> i2s
MASTER      -> CBS_CFS
INVERT      -> NB_NF
SLOTS       -> 2
SLOT WIDTH  -> 32
ML SEL      -> RM_TM
DATA LATE   -> 1

# 根据节点提示说明设置I2S格式，如下
/sys/class/snd_sunxi # echo 8 4 > dump
/sys/class/snd_sunxi # echo 0 > dump && cat dump
module(machine-sndi2s0)
CPUPLL FS   -> 1
CODECPLL FS -> 1
MCLK FP     -> On
MCLK FS     -> 1
FMT         -> i2s
MASTER      -> CBS_CFS
INVERT      -> NB_NF
SLOTS       -> 4      # 该值已修改为4
SLOT WIDTH  -> 32
ML SEL      -> RM_TM
DATA LATE   -> 1
```

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。

5.2.4 I2S 外挂 codec

【分析步骤一】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。

【分析步骤二】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）；
10. 位编号：选择 MSB first 或 LSB first。
11. data late 模式：数据采样时相对 LRCK 脉冲距离多少个 BCLK 脉冲。

查看各模块的 **board.dts 板级配置**配置项说明，根据 I2S 协议格式进行配置。

6 附录

6.1 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。
- 通用 I2S/PCM、带混音 I2S/PCM、OWA、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 6-1: 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 6-2: 模块引脚组定义说明 (linux-4.9)

节点配置	解释说明
allwinner,pins	模块需要使用到的引脚组定义。
allwinner,function	模块引脚组复用名称。
allwinner,muxsel	模块引脚组复用类型，需和 function 对应。
allwinner,driver	模块引脚驱动力，可选值为 0,1,2,3，默认配置为 1 即可。
allwinner,pull	0: 关闭上下拉，1: 支持上拉，2: 支持下拉（默认配置为 0）。

表 6-3: 模块引脚组定义说明 (linux-5.4, linux-5.10, linux-5.15, linux-6.6)

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）。

6.2 耳机配置说明

6.2.1 jack-support 参数说明及配置方法

耳机检测有多种方式，需根据具体硬件配置 jack-support。

表 6-4: jack-support 参数说明

jack-support 值	参数说明
0	不支持耳机检测
1	内置 codec 耳机检测
2	extcon 耳机检测
3	gpio 耳机检测
4	adv 耳机检测

确认 jack-support 类型的方法如下：

1. 内置 codec

对于内置 codec，jack-support 的值可选范围为 0、1、2、3。具体选择的方法如下：

表 6-5: jack-support 值配置

	3.5mm 耳机孔	typec 口
有 MIC-DET	1	2
无 MIC-DET	3	2

2. 外挂 codec

对于外挂 codec，jack-support 的值可选为 0、4，详细的配置方法可参照如下说明：

说明

若外挂 codec 的型号为 AC101, 则查阅《AC101_ 开发指南》；
若外挂 codec 的型号为 AC101B, 则查阅《AC101B_ 开发指南》；

6.2.2 耳机具体配置项说明

根据 jack-support 值，进一步选择耳机配置。耳机配置有

表 6-6: 3.5mm 耳机配置 (使用 micdet)-optional

配置项名称	配置值范围	配置项说明
jack-det-level	0~1	耳机插入检测电平。
jack-det-threshold	u32	耳机 mic 检测阈值，默认 8。
jack-det-debounce	u32	耳机检测防抖时间 ms，默认值 250。

表 6-7: 3.5mm 耳机配置 (使用 gpio)-optional

配置项名称	配置值范围	配置项说明
hp-det-gpio	pio 引脚	耳机插入 gpio 检测引脚。
jack-det-level	0~1	耳机插入检测电平。

表 6-8: type-c 模拟耳机配置-optional

配置项名称	配置值范围	配置项说明
extcon	事件链提供的节点	通过耳机插拔事件链通知节点。
jack-swpin-max	u32	标定 type-c 耳机检测输出引脚数量。
jack-swpin-(n)	pio 引脚	usb audio 转换控制使能引脚。
jack-mode-off	0,1,0xf	usb audio 转换关闭真值。
jack-mode-usb	0,1,0xf	usb 模式真值。
jack-mode-hp	0,1,0xf	audio 模式真值。
jack-mode-micn	0,1,0xf	正插模式真值。
jack-mode-mici	0,1,0xf	反插模式真值。

表 6-9: 耳机通用配置-optional

配置项名称	配置值范围	配置项说明
jack-key-det-voltage-hook	u32	耳机 hook 按键检测电压范围， 默认值：mic: 0, max: 0。

配置项名称	配置值范围	配置项说明
jack-key-det-voltage-up	u32	耳机 up 按键检测电压范围， 默认值：mic: 2, max: 2。
jack-key-det-voltage-down	u32	耳机 down 按键检测电压范围， 默认值：mic: 4, max: 5。
jack-key-det-voltage-voice	u32	耳机 voice 按键检测电压范围， 默认值：mic: 1, max: 1。
jack-sdbp-method	0~2	三节耳机检测四节耳机的检测方式， 0：关闭检测，1：中断检测，2：轮询检测。
jack-sdbp-scan-single-time	u32（>1000）	轮询检测的时间间隔，单位 ms。

6.3 调试指南

6.3.1 pinctrl

6.3.1.1 典型问题 1：引脚复用配置错误导致声卡注册失败

由于每个平台针对模块引脚定义中的 function 定义可能不同，或者 pinctrl 驱动升级时各模块未完成 function 适配，若 pinctrl 出现类似如下报错：

```
sunxi:pinctrl_sunxi@2000000.pinctrl[ERR]: unsupported function i2s0_i on pin PB7
```

说明 function 属性配置错误，需要从对应平台 pinctrl 驱动中获取准确的 function 名称，如：a523 项目（sun55iw3）配置 i2s0 模块引脚 i2s0_pins_a 时，pins 属性中使用到了 “PB7”，function 属性使用 “i2s0”，而引入了如上报错继而导致 i2s0 声卡无法注册，需要从如下路径中获取 PB7 引脚准确的 i2s 引脚复用名称：

> 原配置

```
...
    i2s0_pins_a: i2s0@0 {
        pins = "PB4", "PB5", "PB6", "PB7";
        function = "i2s0";
        drive-strength = <20>;
        bias-disable;
    };
    i2s0_pins_b: i2s0@1 {
        pins = "PB4", "PB5", "PB6", "PB7";
        function = "io_disabled";
        drive-strength = <20>;
        bias-disable;
    };
...
&i2s0_plat {
...
```

```
    pinctrl-used;
    pinctrl-names  = "default","sleep";
    pinctrl-0      = <&i2s0_pins_a>;
    pinctrl-1      = <&i2s0_pins_b>;
...
};
```

从驱动中找出 PB7 准确的 i2s 引脚复用名称

```
longan/bsp/drivers/pinctrl/pinctrl-sun55iw3.c
...
    SUNXI_PIN(SUNXI_PINCTRL_PIN(B, 7),
...
        SUNXI_FUNCTION(0x3, "i2s0_dout"),     /* i2s0_dout0 */
        SUNXI_FUNCTION(0x4, "i2s0_din"),      /* i2s0_din1 */
...
```

该引脚即可复用为 DOUT 或 DIN，若需要使用 DOUT 功能，可作如下修改

```
    i2s0_pins_a: i2s0@0 {
        pins = "PB4", "PB5", "PB6";
        function = "i2s0";
        drive-strength = <20>;
        bias-disable;
    };
    i2s0_pins_b: i2s0@1 {
        pins = "PB4", "PB5", "PB6", "PB7";
        function = "io_disabled";
        drive-strength = <20>;
        bias-disable;
    };
    i2s0_pins_c: i2s0@2 {
        pins = "PB7";
        function = "i2s0_dout";
        drive-strength = <20>;
        bias-disable;
    };
...
&i2s0_plat {
...
    pinctrl-used;
    pinctrl-names  = "default","sleep";
    pinctrl-0      = <&i2s0_pins_a &i2s0_pins_c>;
    pinctrl-1      = <&i2s0_pins_b>;
...
```

6.3.1.2 典型问题 2：多模块引脚占用导致声卡注册失败

当两个模块同时引用同一个引脚时，会出现引脚占用而导致模块加载异常问题，参考日志：

```
[   1.055601] sun251iw1-pinctrl 2000000.pinctrl: pin PF0 already requested by 4020000.sdmmc; cannot claim for
      2034000.i2s2_plat
```

确认引脚占用模块

1. 被占用引脚可能位于 pinctrl 属性中，如日志中提示 PF0 被占用，过滤 **PF0**; 2. 被占用引脚亦可能位于 GPIO 属性中，如日志中提示 PF0 被占用，过滤 **PF 0**;

> 确认占用模块是否作为关键功能

1. 假设占用模块非必须打开或可在测试阶段关闭，则关闭此模块；2. 假设占用模块必须打开，则需硬件改版解决引脚占用问题。

6.3.1.3 典型问题 3：I2S 启动或关闭时外挂 CODEC 产生 pop 音

产生此问题的原因可能是 I2S 关闭状态下引脚处于悬空状态导致，可配置引脚为上拉或下拉解决此问题：

```
i2s0_pins_a: i2s0@0 {
    ...
    /* bias-disable:关闭时悬空; bias-pull-up:关闭时上拉; bias-pull-down:关闭时下拉 */
    bias-pull-up;
}
```

6.3.2 耳机检测配置说明

6.3.2.1 codec 内置耳机检测

说明

以下属性可用于支持圆孔模拟耳机检测场景。

属性 1：jack-det-level

- 调试方法 1：

1. 将 **jack-det-level** 值配置为 1；2. 使用示波器或万用表测量 **HP-DET** 引脚，预期结果：

```
插入耳机前：高电平
插入耳机后：低电平
```

3. 若电平变化方向与预期相反，将 **jack-det-level** 设置为 0。

- 调试方法 2：

1. 异常现象：

```
插入耳机时打印：`jack report -> OUT`
拔出耳机时打印：`jack report -> HEADSET`
```

解决方案：将 jack-det-level 的当前值取反。

属性 2：jack-det-threshold

1. 前提条件：audiocodec 驱动中打印出 “headset_basedata” 变量的值；2. 异常现象 1：

```
插入耳机时：headset_basedata != jack-det-threshold
```

解决方案：将 jack-det-threshold 的当前值设置为 headset_basedata 插入耳机时的值。

3. 异常现象 2：

```
插入耳机时：headset_basedata == 0
```

解决方案：检查 HP-DET 引脚有没有连接耳机座子，检查耳机检测外围电路。

属性 3：jack-det-debounce

1. 设置一个较小的延时值（默认 250ms），测试插入/拔出的响应速度;

```
/* 默认值 */
jack-det-debounce = <250>;
```

2. 反复插拔耳机，观察是否有误触发现象（如插入后立即检测为拔出）;

```
/* 使用以下命令查看日志 */
dmesg | grep "jack report"
```

3. 如果出现误触发，逐步增加延时值（如 300ms、350ms），直到误触发消失。

6.3.2.2 extcon 耳机检测

说明

以下属性可用于支持 TYPE-C 模拟耳机检测场景。

属性 1：extcon

1. 从原理图中查看 CC 引脚所连接的 CC logic 模块；2. 确认该模块所属驱动是否已支持 extcon 检测 type-c 耳机插拔，若未支持或程序未调用该函数，需要 PMU 模块负责人协助支持;

```
/* 相关代码 */
extcon_set_state_sync(chip->edev, EXTCON_JACK_HEADPHONE, true);
```

3. 设置属性值为 CC logic 模块驱动 dts 节点：

```
extcon = <&{CC logic模块dts节点}>;
/* 示例 */
extcon = <&usb_power_supply>;

usb_power_supply: usb_power_supply {
  ...
  pmu_usb_typec_used = <1>;
  ...
};
```

属性 2：jack-swpin-max

从原理图中查看 TYPE-C AUDIO 的相关电路，包括如下两个转换模块，分别为 MIC/GND 转换模块与 USB/AUDIO 转换模块，确认所需配置的引脚数：

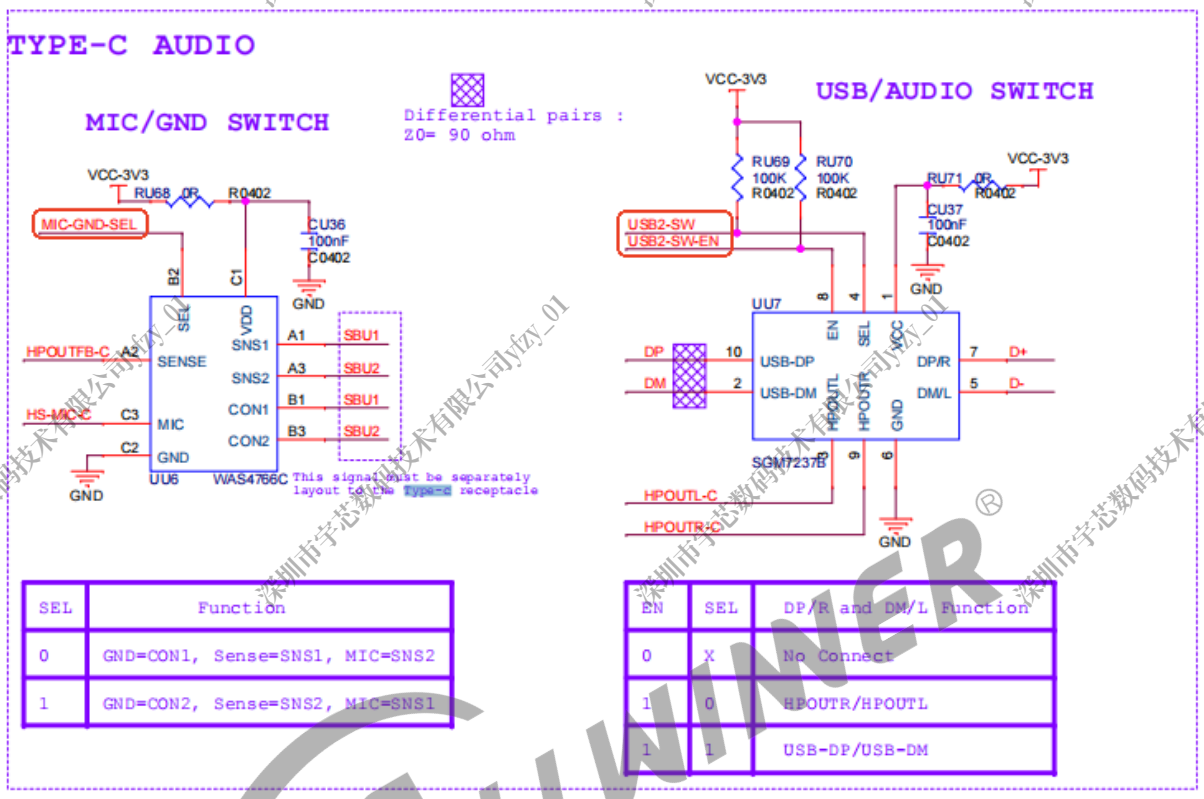

图 6-1: A523 EVB TYPE-C 耳机检测电路

- 例如：如图中红框圈出，A523 EVB TYPE-C 耳机检测电路中控制模式具有 3 个引脚，故将 **jack-swpin-max** 设置为 3。

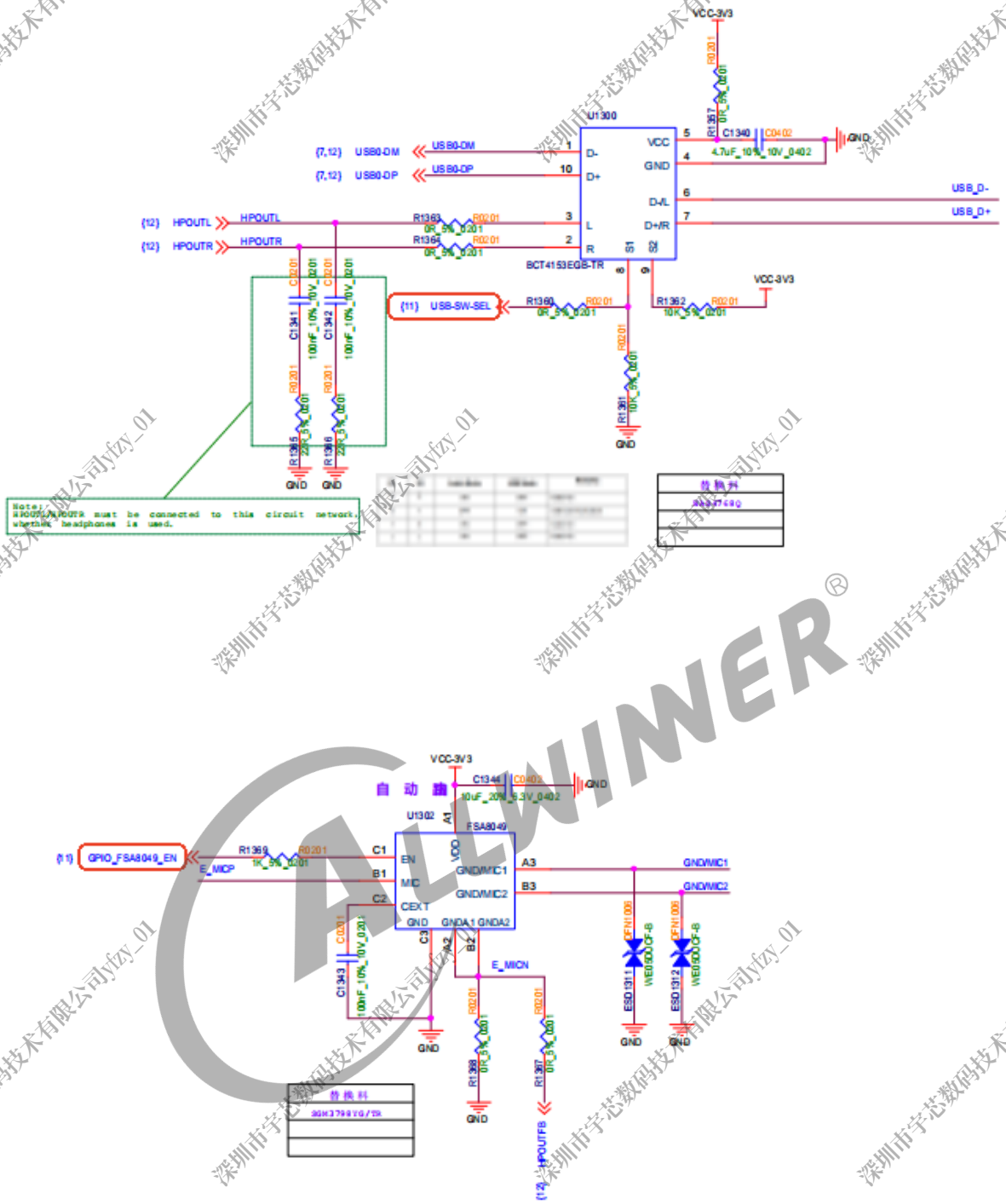

图 6-2: AI985 SCANP TYPE-C 耳机检测电路

- 例如：如图中红框圈出，AI985 SCANP TYPE-C 耳机检测电路中控制模式具有 2 个引脚，故将 **jack-swpin-max** 设置为 2。

属性 3：jack-swpin-{n}

从原理图中查看各个 CC 检测电路与 USB AUDIO 转换电路的引脚序号，并逐一配置。

```
jack-swpin-0     = <&pio PH 8 GPIO_ACTIVE_HIGH>;
...
/* {n}等于jack-swpin-max的属性值 */
jack-swpin-{n-1}     = ...;
```

属性 4：jack-mode-off

从原理图的真值表中查看 TYPE-C 识别为拔出状态时各个引脚的电平状态，若无真值表需要查看芯片手册或咨询硬件。

```
低电平：0； 高电平：1；保持默认：0xf。
示例配置：
/* 属性中 value 个数等于jack-swpin-max */
/* 第 n 个 value 对应第 n 个 pin 的电平 */
jack-mode-off = <0xf 0xf>;
```

属性 5：jack-mode-usb

从原理图的真值表中查看 TYPE-C 识别为 USB 状态时各个引脚的电平状态，若无真值表需要查看芯片手册或咨询硬件。

```
低电平：0； 高电平：1；保持默认：0xf。
示例配置：
/* 属性中 value 个数等于jack-swpin-max */
/* 第 n 个 value 对应第 n 个 pin 的电平 */
jack-mode-usb = <0xf 0>;
```

属性 6：jack-mode-hp

从原理图的真值表中查看 TYPE-C 识别为耳机状态时各个引脚的电平状态，若无真值表需要查看芯片手册或咨询硬件。

```
低电平：0； 高电平：1；保持默认：0xf。
示例配置：
/* 属性中 value 个数等于jack-swpin-max */
/* 第 n 个 value 对应第 n 个 pin 的电平 */
jack-mode-hp = <0xf 1>;
```

属性 7：jack-mode-mici & jack-mode-micn

从原理图的真值表中查看 TYPE-C 识别为正插或反插时各个引脚的电平状态，若无真值表需要查看芯片手册或咨询硬件。

```
低电平：0； 高电平：1；保持默认：0xf。
示例配置：
/* 属性中 value 个数等于jack-swpin-max */
/* 第 n 个 value 对应第 n 个 pin 的电平 */
jack-mode-mici = <0x1 0xf>;
jack-mode-micn = <0x0 0xf>;
```

6.3.2.3 gpio 耳机检测

用于支持圆孔模拟耳机检测，IC 无 HP-DET 引脚时需使用此方式。

属性 1：hp-det-gpio

从原理图中查看耳机插入 gpio 检测引脚。

```
示例配置：
hp-det-gpio = <&pio PF 3 GPIO_ACTIVE_HIGH>;
```

属性 2：jack-det-level

参考内置 “codec 耳机检测” 一章，功能一致。

6.3.2.4 四段耳机属性

属性：jack-key-det-voltage-hook & jack-key-det-voltage-up & jack-key-det-voltage-down & jack-key-det-voltage-voice

默认按照出厂经验值即可，若出现按键误识别的情况，需要按如下步骤调试。

1. 前提条件：在 audiocodec 驱动中打印出经过计算的 “SUNXI_HMIC_STA” 寄存器的值;

```
...
regmap_read(regmap, SUNXI_HMIC_STA, &reg_val);
reg_val = (reg_val & 0x1f00) >> 8;
/* 增加如下打印 */
SND_LOG_ERR("reg_val:%u\n", reg_val);
...
```

2. 使用多款不同厂商的耳机，逐一按下各个耳机按键，统计各款耳机不同按键下的 “reg_val”；3. 逐步调整阈值范围，统计出一套能够兼容所有耳机的按键阈值范围，并确保各个按键阈值互不交错; 4. 若出现阈值交错的情况，需确认 SWITCH 器件或耳机检测外围电路是否异常。

6.3.3 PA 配置说明

> 电平使能方式配置

属性 1：pa-pin-max

1. 从原理图中查看外部功放控制引脚数量。2. 部分功放具备电源控制引脚与使能引脚等，注意识别。

属性 2：pa-pin-{n}

从原理图中查看各个外部功放控制引脚。

```
示例配置：
pa-pin-0 = <&pio PH 6 GPIO_ACTIVE_HIGH>;
...
/* {n}等于pa-pin-max的属性值 */
```

```
pa-pin-{n-1} = <&pio PH 6 GPIO_ACTIVE_HIGH>;
```

属性 3：pa-pin-level-{n}

从原理图或功放手册中查看各个外部功放控制引脚使能电平，一般为高电平。

属性 4：pa-pin-msleep-{n}

若启动播放时喇叭存在 pop 音，可逐步调整此配置用于规避 pop 音。此方法仅可规避 IC 内部产生的 pop 音。

属性 5：pa-pin-msleep1-{n}

若结束播放时喇叭存在 pop 音，可逐步调整此配置用于规避 pop 音。此方法仅可规避 IC 内部产生的 pop 音。

> 脉冲使能方式附加配置

属性 6：pa-pin-duty-{n}

一个周期内电平为 polarity 时的时长，从功放手册中确定使能脉冲宽度。

属性 7：pa-pin-period-{n}

一个周期的时长，从功放手册中确定使能脉冲周期。

属性 8：pa-pin-polarity-{n}

一个周期的起始电平，从功放手册中确定使能脉冲极性。

属性 9：pa-pin-periodcnt-{n}

单次使能脉冲的周期个数，从功放手册中确定使能脉冲个数。

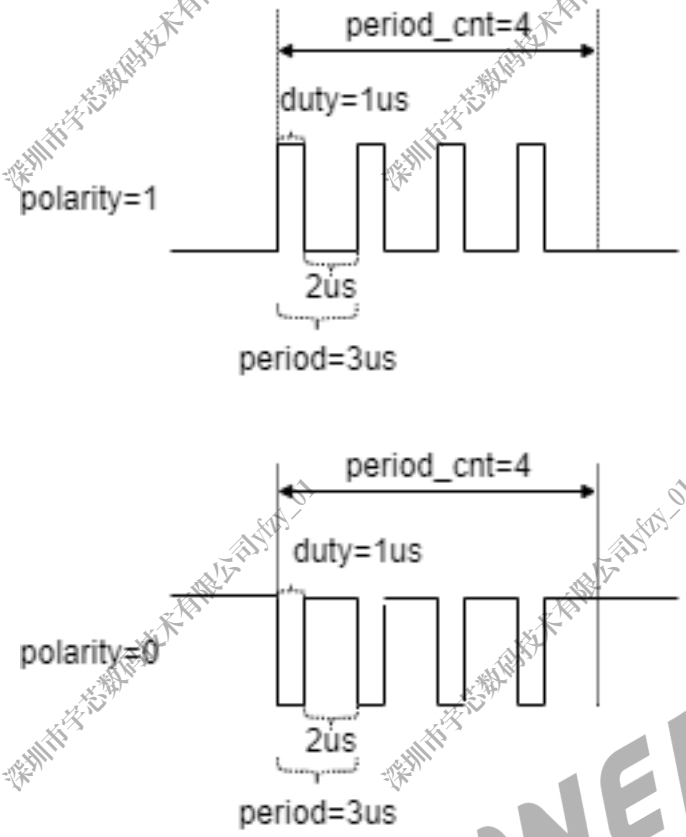

图 6-3: 脉冲使能方式示例

6.3.4 I2S 通道映射

6.3.4.1 TX 通道映射

TX 通道映射定义了音频数据流中每个逻辑声道（如左声道、右声道）在物理输出接口（如 I2S 的 DATA 线）上的传输顺序和位置关系。具体来说，它规定了每个声道的音频采样值在输出数据流中的排列方式，即 txfifo channel 序号映射到 DOUT 的 tx slot 序号。

1. tx-pin 表示所需使能的 data 线;
2. tx-pin{n}-chmap[i] 的参数个数为通道数，表示第 n 路 data 的第 i 通道音频数据对应哪个 slot(0~15)；

如下示例，假设需要使用 DOUT0 与 DOUT1 输出音频数据，其中 DOUT0 的第一通道数据来源于播放流的 slot0, 第二通道数据来源于播放流的 slot1; DOUT1 的第一通道数据来源于播放流的 slot1, 第二通道数据来源于播放流的 slot0;

```
tx-pin       = <0 1>;
tx-pin0-chmap   = <0 1>;
tx-pin1-chmap   = <1 0>;
```

6.3.4.2 RX 通道映射

RX 通道映射与 TX 通道映射功能类似，映射方向相反。即该属性用于设置 DIN 的 rx slot 序号映射到 rxfifo channel 的序号。

1. rxfifo-pinmap 和 rxfifo-chmap 的参数个数，即为通道数;
2. rxfifo-pinmap[i] 表示第 i 个通道的音频数据采集于哪个 data 线;
3. rxfifo-chmap[j] 表示第 j 个通道的音频数据来源于 rxfifo-pinmap[j] 对应 data 线的哪个 solt(0~15)。

如下示例，假设需录制 2ch 音频，rxfifo-pinmap 设置了录音流的第一通道数据采集于 DIN0, 录音流的第二通道数据采集于 DIN1；rxfifo-chmap 设置了录音流的第一通道数据来源于 DIN0 的 slot0, 录音流的第二通道数据来源于 DIN1 的 slot1:

```
rxfifo-pinmap = <0 1>;
rxfifo-chmap  = <0 1>;
```

6.3.5 DMIC 通道映射

DMIC 每路 DATA 可采集 2ch 数据，属性值采用十六进制表示，其中 bit0-bit3 代表录音流的第一通道，bit4-bit7 代表录音流的第二通道，后续依此类推；数字 0 代表 DATA0 的 slot 0，数字 1 代表 DATA0 的 slot1；数字 2 代表 DATA1 的 slot 0，数字 3 代表 DATA1 的 slot1；数字 4 代表 DATA2 的 slot 0，数字 5 代表 DATA2 的 slot1；数字 6 代表 DATA3 的 slot 0，数字 7 代表 DATA3 的 slot1；

如下示例，使用 DATA1 录制 2ch 音频，录音流的第一通道映射 DATA1 的左声道，第二通道映射 DATA1 的右声道，对应的 dts 配置为：

```
rx-chmap = <0x32>;
```

6.4 时钟树

6.4.1 sun8iw20

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。

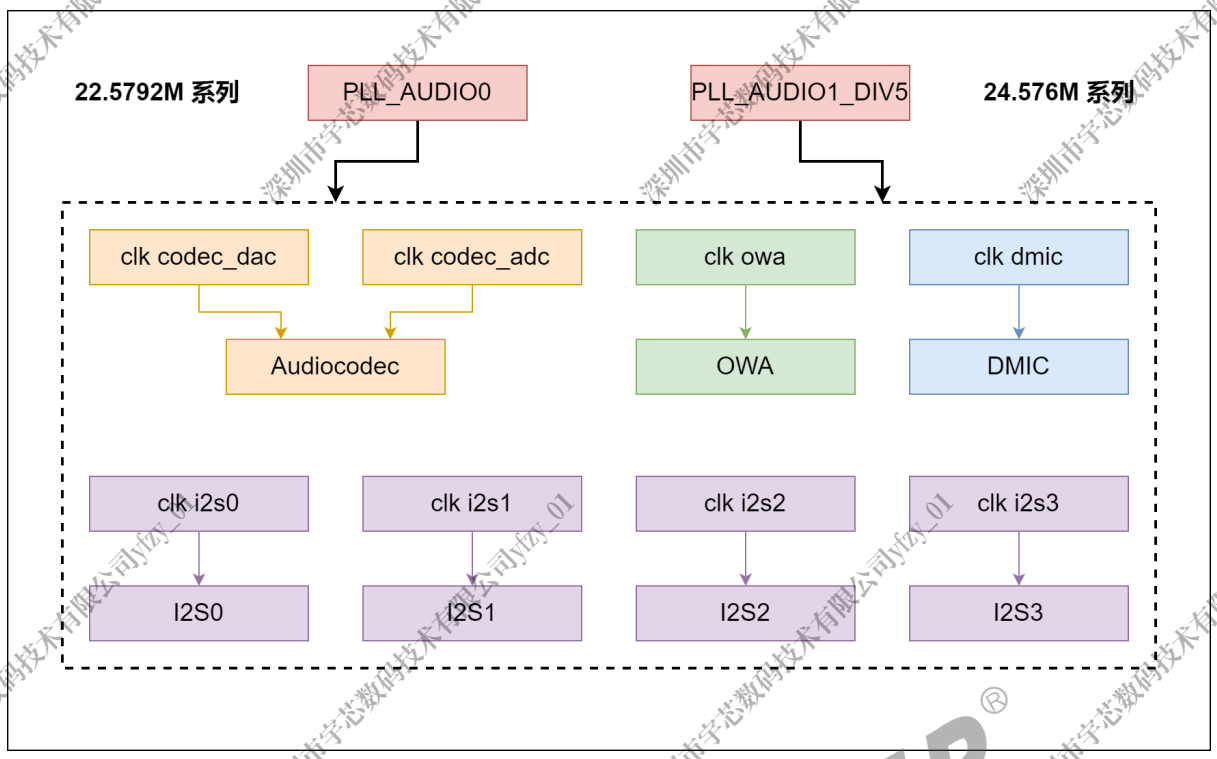

图 6-4: 时钟树 sun8iw20

6.4.2 sun8iw21

sun8iw21 音频模块时钟源来自 PLL_AUDIO_4X。

PLL_AUDIO 可输出 22.5792M 和 24.576M 频率的时钟，分别支持 44.1K 系列、48K 系列的播放录音，但无法同时输出。

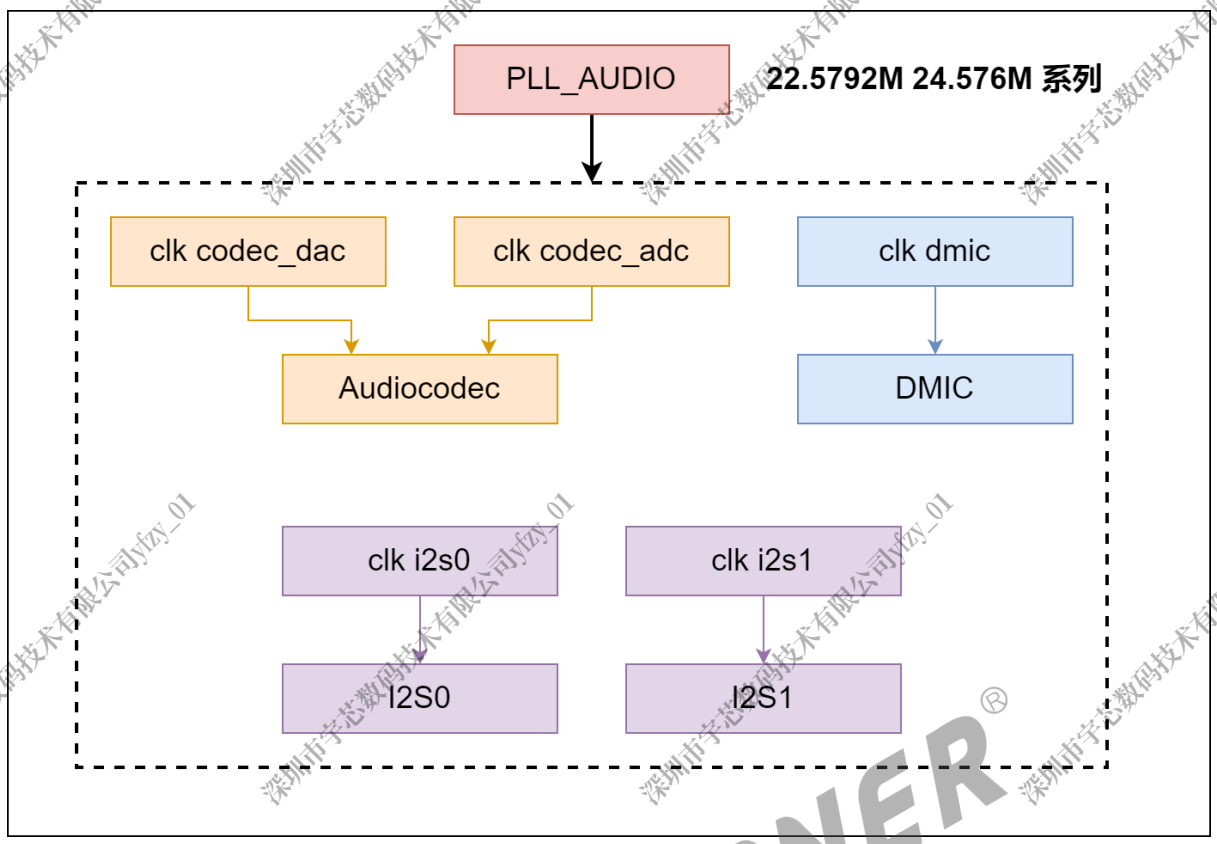

图 6-5: 时钟树 sun8iw21

6.4.3 sun8iw11

sun8iw11 音频模块时钟源来自 PLL_AUDIO。

PLL_AUDIO 可输出 22.5792M 和 24.576M 频率的时钟，分别支持 44.1K 系列、48K 系列的播放录音，但无法同时输出。

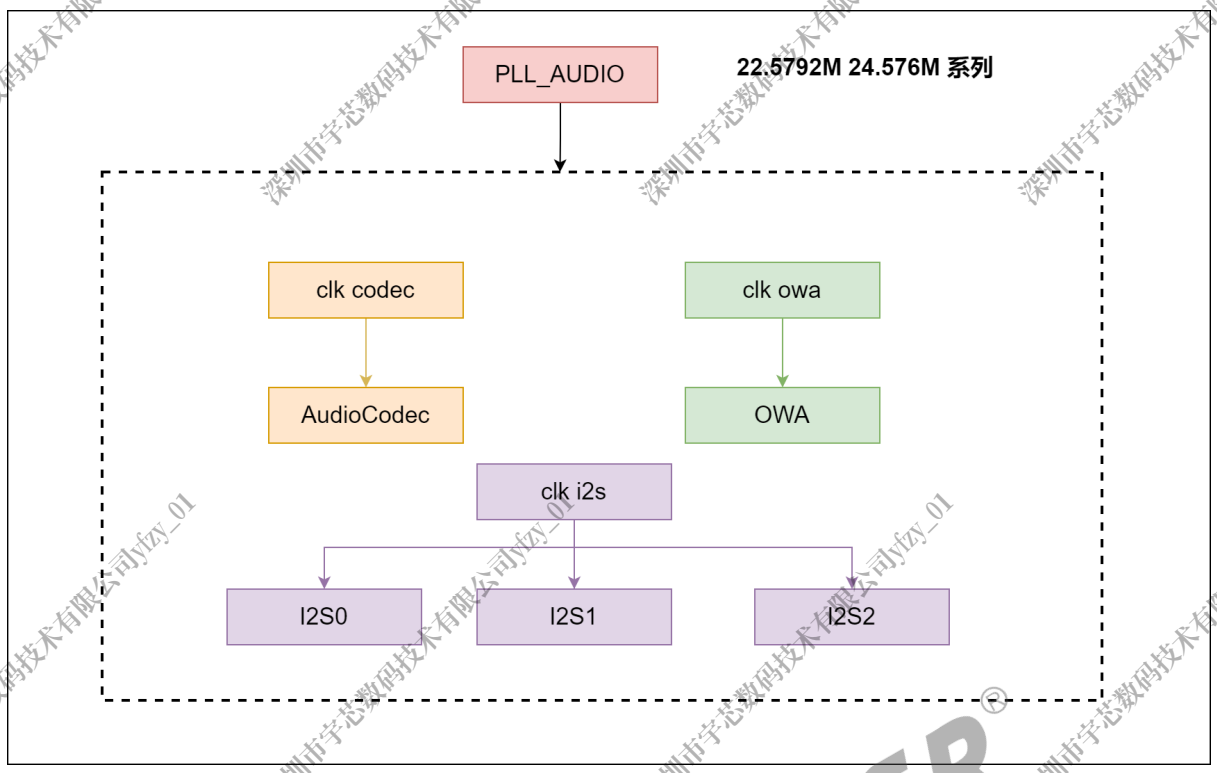

图 6-6: 时钟树 sun8iw11

6.4.4 sun8iw22

sun8iw22 音频模块时钟源来自 PLL_AUDIO0 和 PERI1_600M。

PLL_AUDIO0 可输出 22.5792M，PERI1_600M 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。

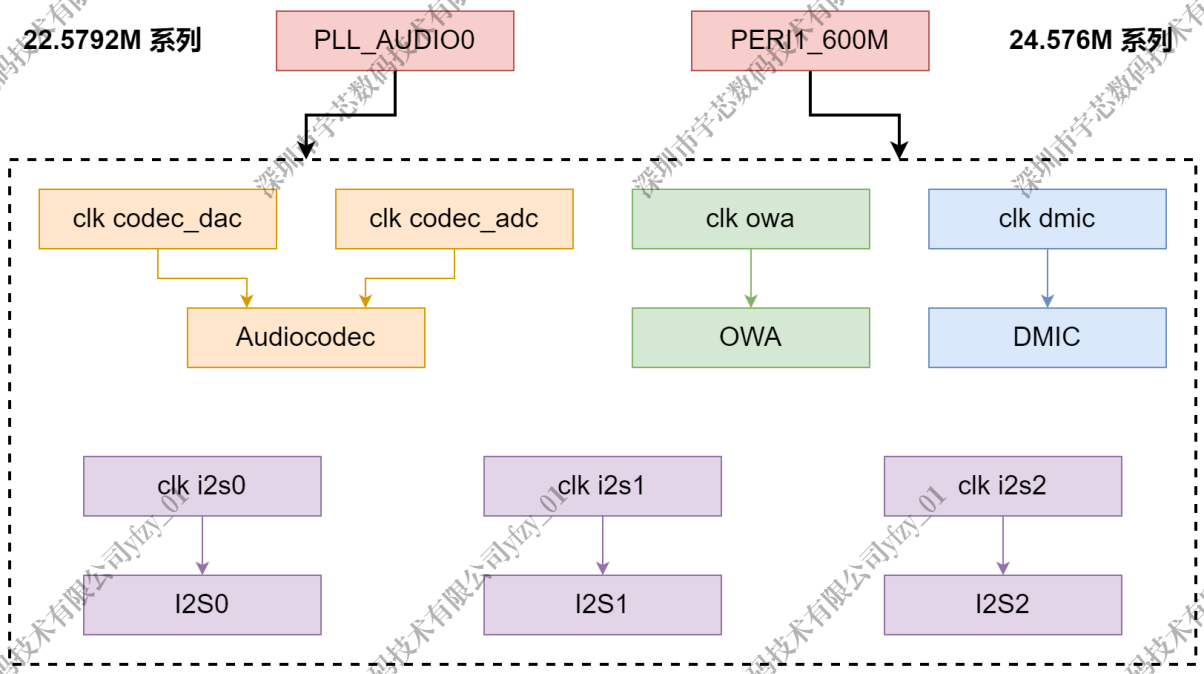

图 6-7: 时钟树 sun8iw22

6.4.5 sun50iw9

sun50iw9 音频模块时钟源来自 PLL_AUDIO_4X。

PLL_AUDIO_4X 可输出 90.3168M 和 98.304M 频率的时钟，分别支持 44.1K 系列、48K 系列的播放录音，但无法同时输出。

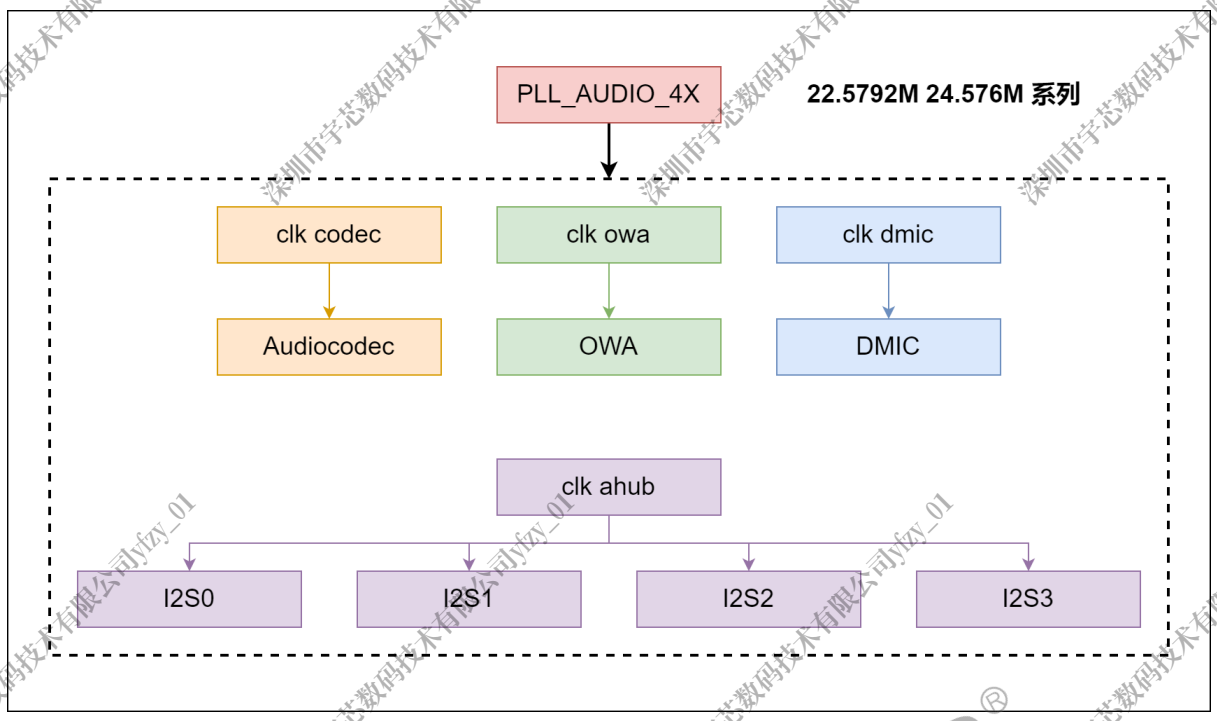

图 6-8: 时钟树 sun50iw9

6.4.6 sun50iw10

sun50iw10 音频模块时钟源来自 PLL_COM_AUDIO 和 PLL_AUDIO。

PLL_COM_AUDIO 可输出 90.3168M，PLL_AUDIO 可输出 98.304M 频率的时钟，分别支持 44.1K 系列、48K 系列的播放录音。

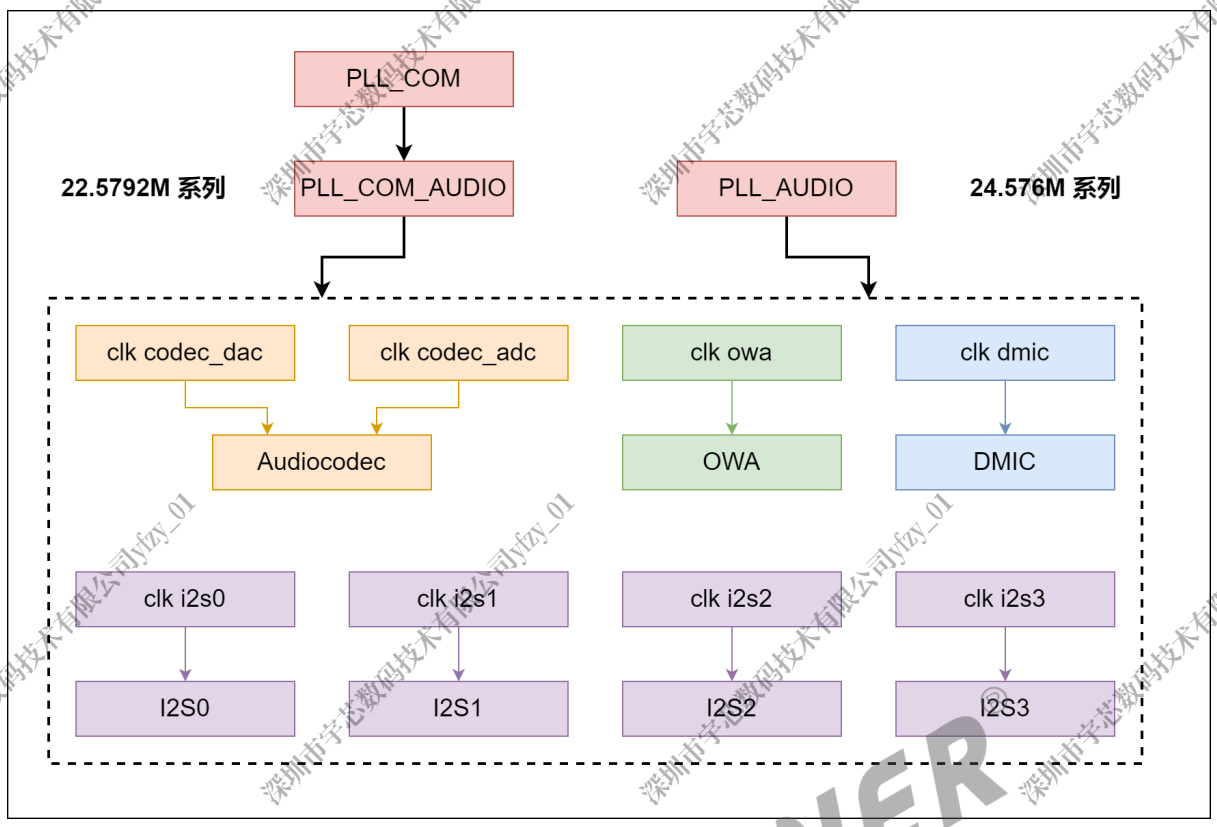

图 6-9: 时钟树 sun50iw10

说明

PLL_xxx 到模块时钟，模块时钟将其 4 分频，最终输出 22.5792M 和 24.576M 的时钟。

6.4.7 sun55iw3

sun55iw3 音频模块时钟源来自 PLL_AUDIO0_4X，PLL_AUDIO1_DIV2，PLL_AUDIO1_DIV5，PLL_PERI0_300M。

PLL_AUDIO0、PLL_AUDIO1_DIV2 和 PLL_AUDIO1_DIV5 均可输出 22.5792M 和 24.576M 频率的时钟，分别支持 44.1K 系列、48K 系列的播放录音。PLL_AUDIO1_DIV2 和 PLL_AUDIO1_DIV5 同时使用时只能输出相同的频率。PLL_PERI0_300M 输出 300M。

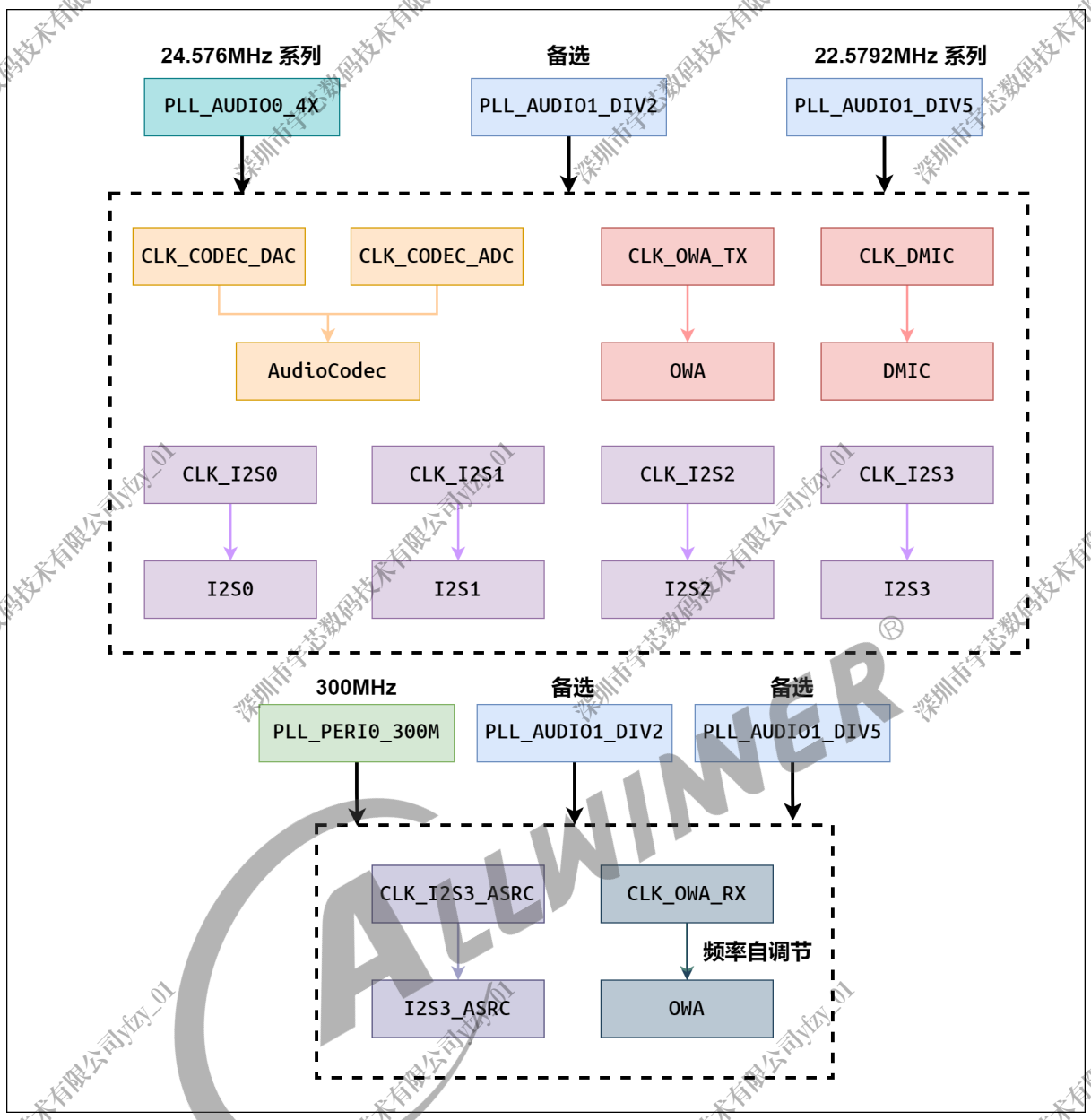

图 6-10: 时钟树 sun55iw3

6.4.8 sun55iw6

sun55iw6 音频模块时钟源来自 PLL_AUDIO0_4X, PLL_AUDIO1_4X。

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。PLL_PERI0_300M 输出 300M。

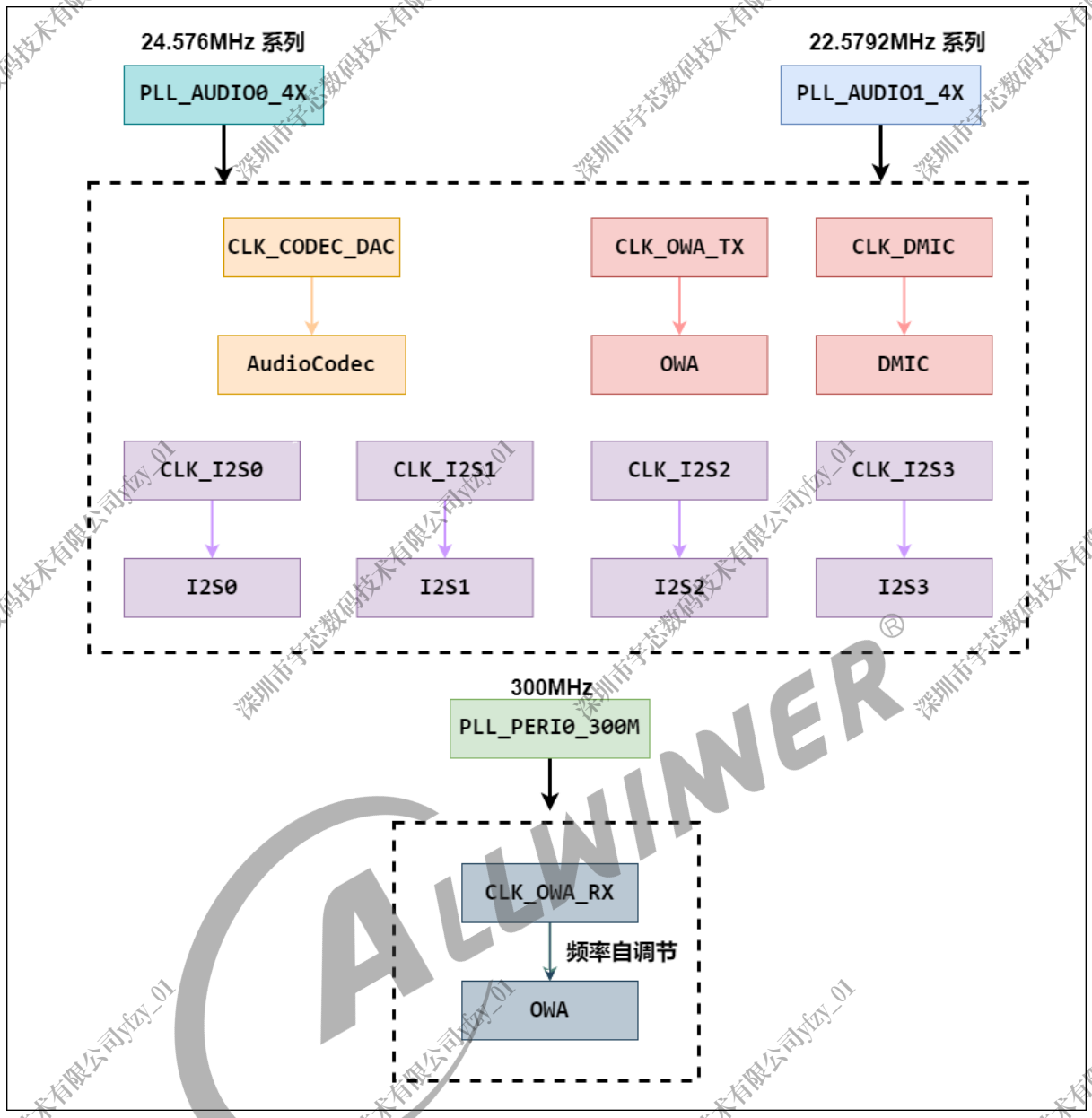

图 6-11: 时钟树 sun55iw6

6.4.9 sun300iw1

sun300iw1 音频模块时钟源来自 PLL_AUDIO0_1X。

PLL_AUDIO0_1X 可输出 24.576M，支持 48K 系列的播放录音。

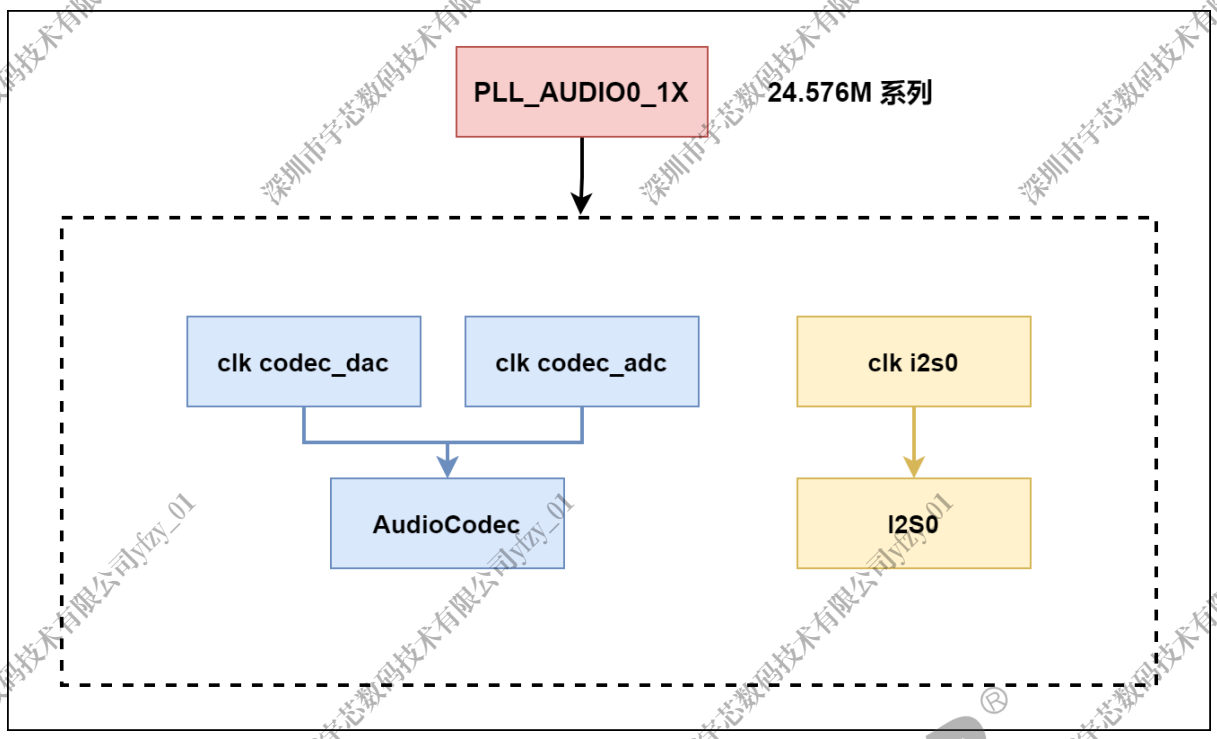

图 6-12: 时钟树 sun300iw1

6.4.10 sun251iw1

sun251iw1 音频模块时钟源来自 PLL_AUDIO0_1X，PLL_AUDIO1_DIV5，PLL_AUDIO0_4X，PLL_PERI_1X。

PLL_AUDIO0_1X 和 PLL_AUDIO0_4X 可输出 22.5792M 频率的时钟，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。PLL_PERI_1X 输出 600M。

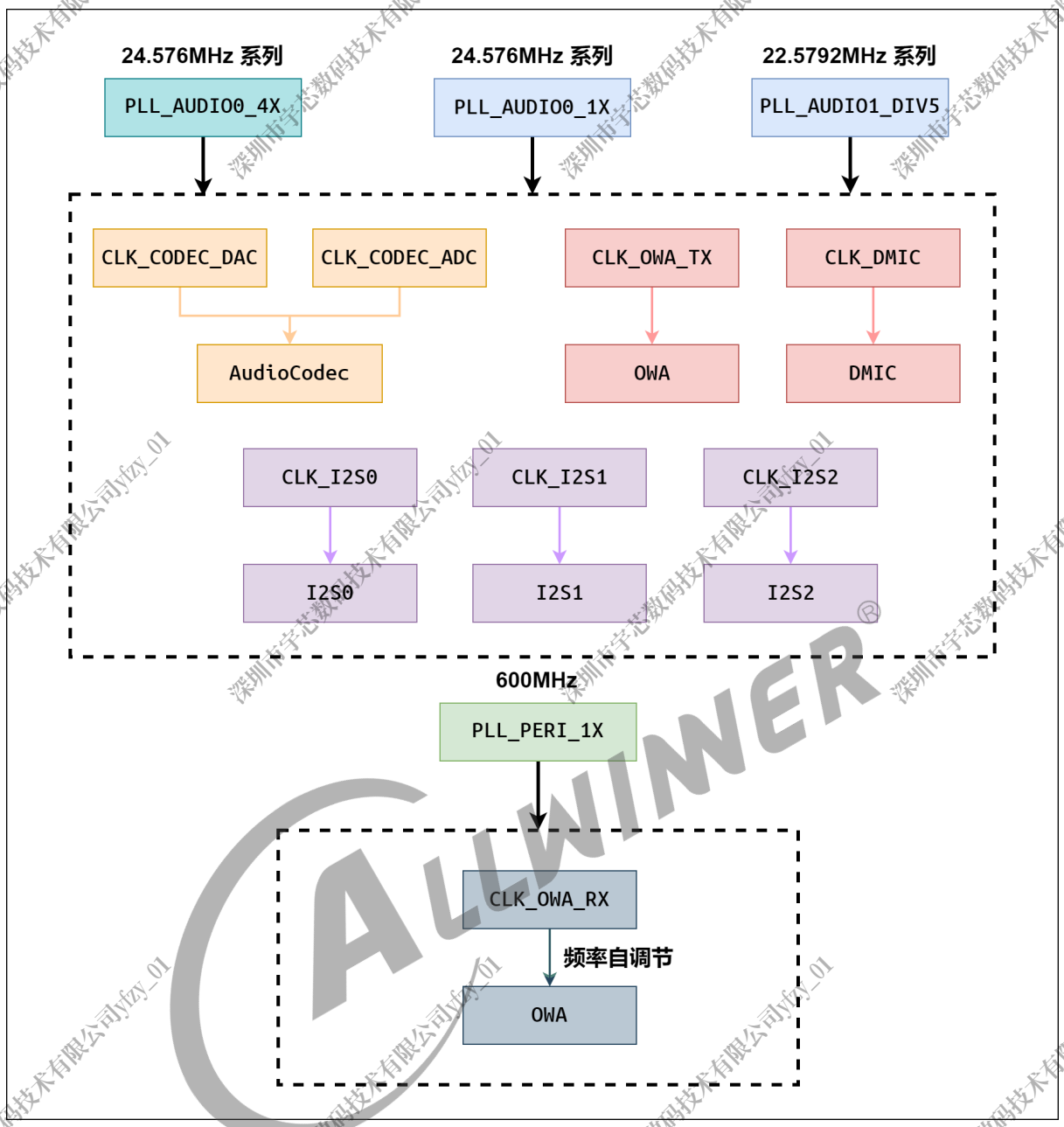

图 6-13: 时钟树 sun251iw1

6.4.11 sun60iw2

sun60iw2 音频模块时钟源来自 PLL_AUDIO0_4X, PLL_AUDIO1_DIV5。

PLL_AUDIO0_4X 可输出 22.5792M 频率的时钟，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。PLL_PERI0_200X 输出 200M。

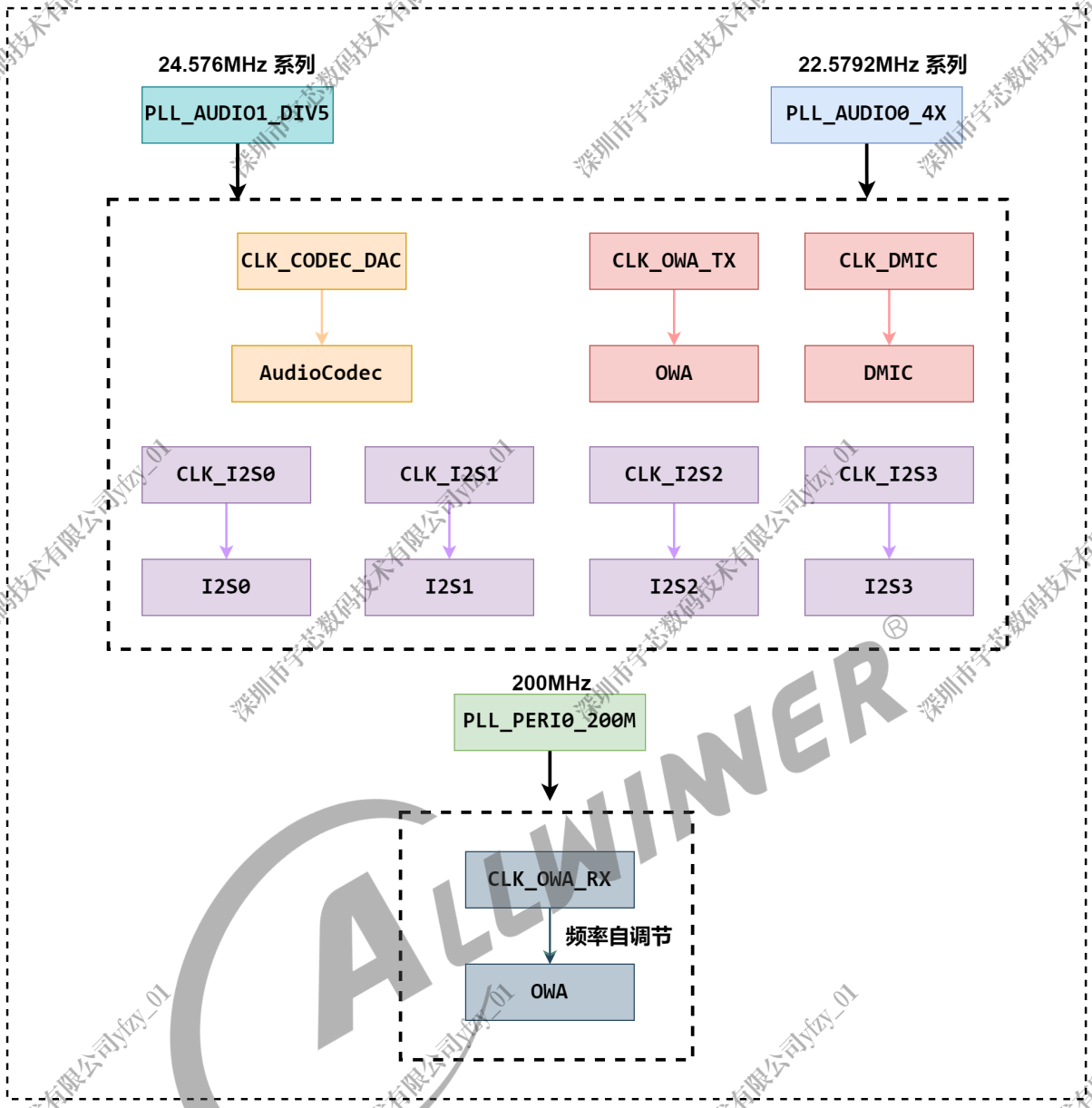

图 6-14: 时钟树 sun60iw2

6.4.12 sun50iw15

sun50iw15 音频模块时钟源来自 PLL_AUDIO_4X。

PLL_AUDIO_4X 可输出 22.5792M、24.576M 频率的时钟，分别支持 44.1K 系列、48K 系列的播放录音。PLL_PERI0_400X 输出 400M。

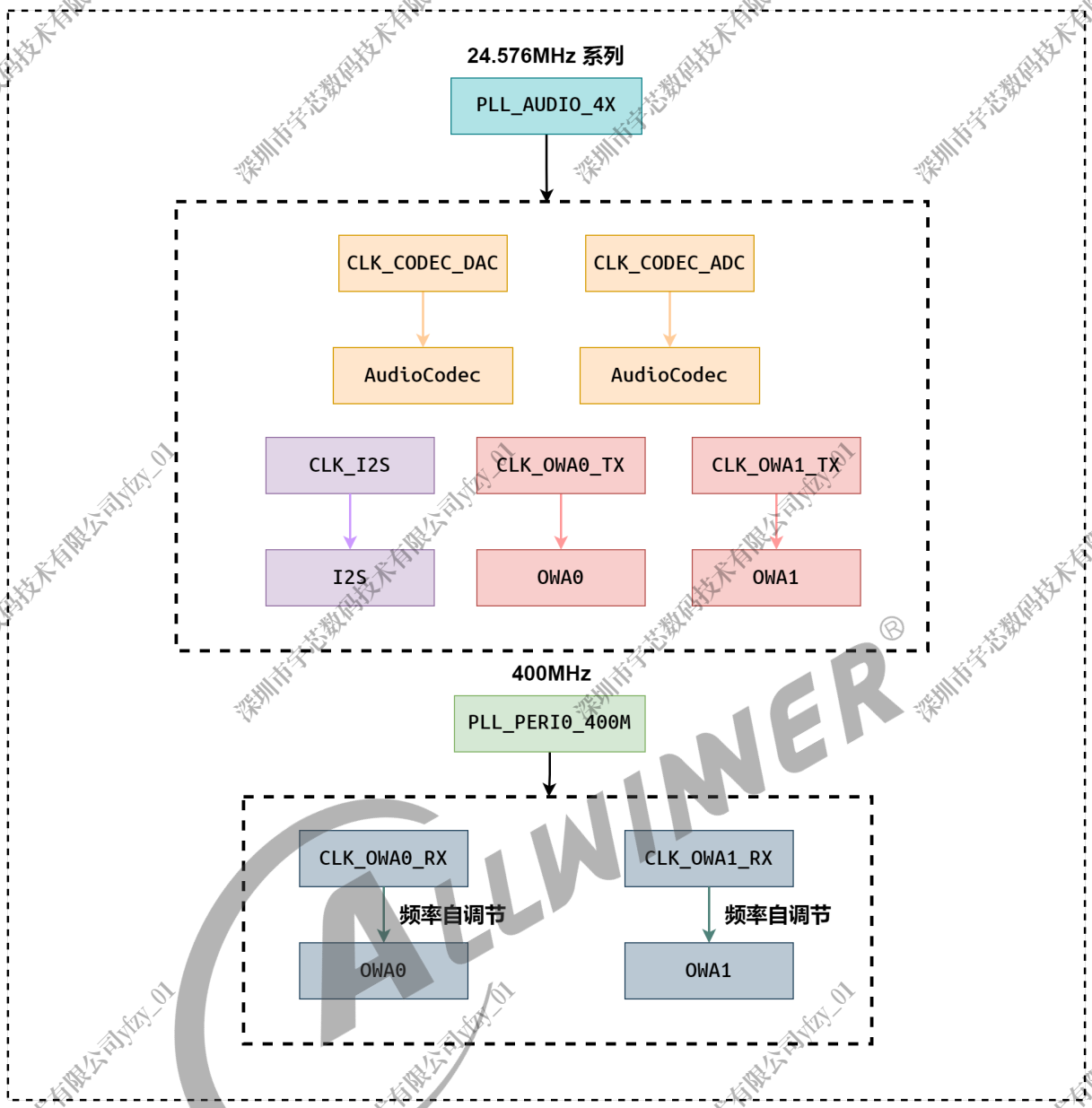

图 6-15: 时钟树 sun50iw15

6.4.13 sun65iw1

sun65iw1 音频模块时钟源来自 PLL_AUDIO0, PLL_AUDIO1_5X。

PLL_AUDIO0 可输出 22.5792M 频率的时钟，PLL_AUDIO1_5X 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。PLL_PERI0_300X 输出 300M。

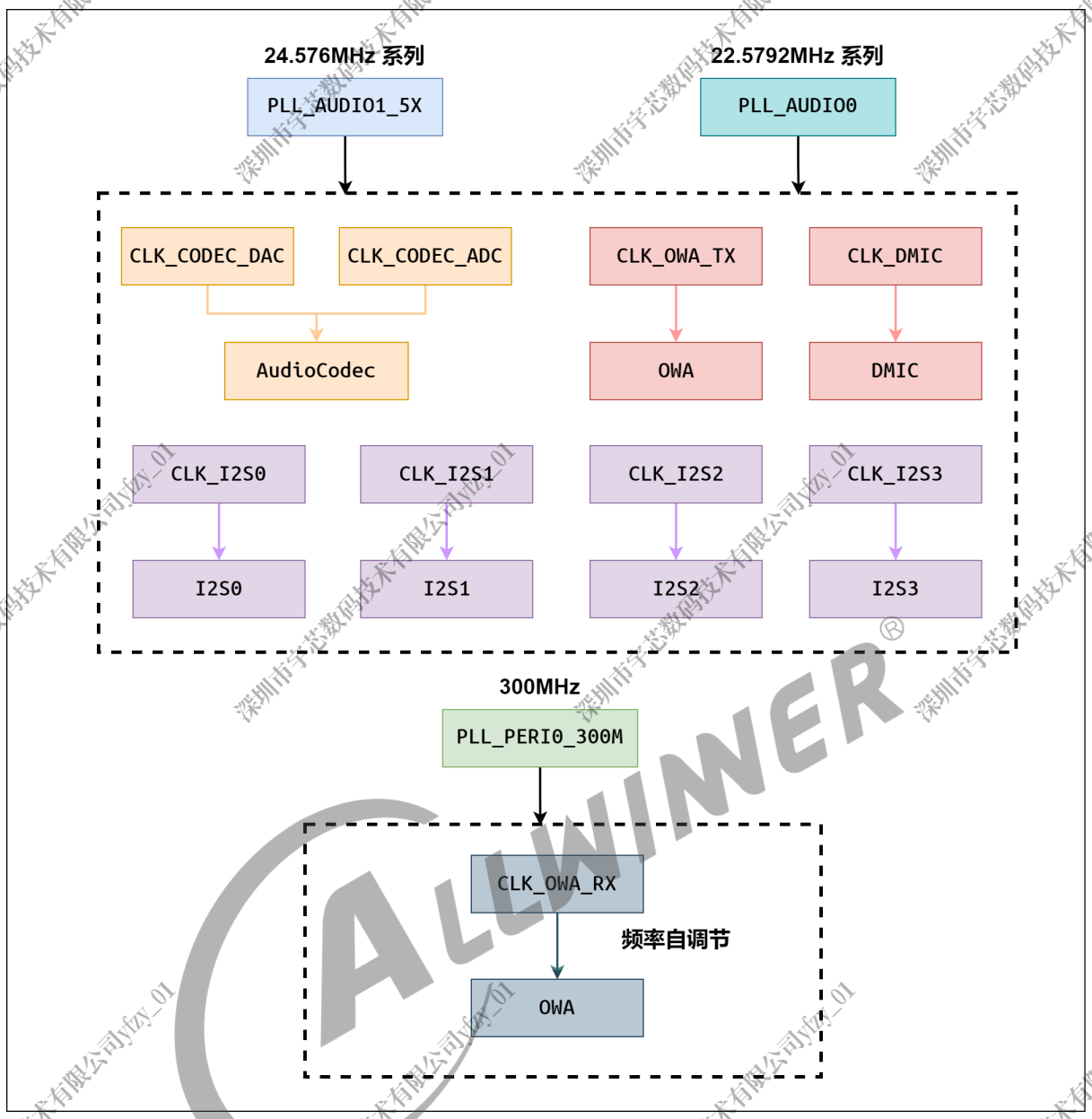

图 6-16: 时钟树 sun65iw1

6.5 AudioCodec 声卡使用

6.5.1 sun8iw20

6.5.1.1 声卡控件

控件列表

```
Mixer name: 'audiocodec'
Number of controls: 42
ctl   type   num    name                value
0     ENUM   1     DAC DRC Mode                , OffOn
1     ENUM   1     DAC HPF Mode                , OffOn
2     ENUM   1     ADC DRC0 Mode               , OffOn
3     ENUM   1     ADC HPF0 Mode               , OffOn
4     ENUM   1     ADC DRC1 Mode               , OffOn
5     ENUM   1     ADC HPF1 Mode               , OffOn
6     ENUM   1     ADC1 ADC2 Swap              , OffOn
7     ENUM   1     ADC3 ADC4 Swap              , OffOn
8     ENUM   1     LINEOUTL Output Select          , singlediffer
9     ENUM   1     LINEOUTR Output Select          , singlediffer
10    ENUM   1     MIC1 Input Select           , singlediffer
11    ENUM   1     MIC2 Input Select           , singlediffer
12    ENUM   1     MIC3 Input Select           single, differ
13    INT    1     DAC Digital Volume          63 (range 0->63)
14    INT    1     DACL Volume                 160 (range 0->255)
15    INT    1     DACR Volume                 160 (range 0->255)
16    INT    1     ADC1 Volume                 160 (range 0->255)
17    INT    1     ADC2 Volume                 160 (range 0->255)
18    INT    1     ADC3 Volume                 160 (range 0->255)
19    INT    1     ADC1 Gain                   31 (range 0->31)
20    INT    1     ADC2 Gain                   31 (range 0->31)
21    INT    1     ADC3 Gain                   31 (range 0->31)
22    INT    1     LINEOUT Volume              20 (range 0->31)
23    INT    1     HPOUT Gain                  7 (range 0->7)
24    BOOL   1     FMINL Gain                  Off
25    BOOL   1     FMINR Gain                  Off
26    BOOL   1     LINEINL Gain                Off
27    BOOL   1     LINEINR Gain                Off
28    BOOL   1     MIC1 Switch                 Off
29    BOOL   1     MIC2 Switch                 Off
30    BOOL   1     MIC3 Switch                 On
31    BOOL   1     FMINL Switch                Off
32    BOOL   1     FMINR Switch                Off
33    BOOL   1     LINEINL Switch              Off
34    BOOL   1     LINEINR Switch              Off
35    BOOL   1     LINEOUTL Switch             On
36    BOOL   1     LINEOUTR Switch             On
37    BOOL   1     HPOUT Switch                On
38    BOOL   1     SPK Switch                  On
39    ENUM   1     Input1 Mux                  , MIC1FMINLLINEINL
40    ENUM   1     Input2 Mux                  , MIC2FMINRLINEINR
41    ENUM   1     Input3 Mux                  , MIC3
```

6.5.1.2 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。声卡序号通过 **cat /proc/asound/cards** 查看。

录音

MIC1 单通道录音

```
# 录音通路控件
tinymix -D 0 "MIC1 Switch" 1
tinymix -D 0 "Input1 Mux" 0
tinycap mic.wav -D 0 -c 1 -T 10
```

MIC2 单通道录音

```
# 录音通路控件
tinymix -D 0 "MIC2 Switch" 1
tinymix -D 0 "Input2 Mux" 0
tinycap mic.wav -D 0 -c 1 -T 10
```

MIC 双通道输入（从 mic1、mic2、mic3 中任选 2 个即可）

```
# 录音通路控件（mic1 & mic2）
tinymix -D 0 "MIC1 Switch" 1
tinymix -D 0 "MIC2 Switch" 1
tinymix -D 0 "Input1 Mux" 0
tinymix -D 0 "Input2 Mux" 0
tinycap mic.wav -D 0 -c 3 -T 10
```

FMIN 双通道输入

```
# 录音通路控件
tinymix -D 0 "FMINL Switch" 1
tinymix -D 0 "FMINR Switch" 1
tinymix -D 0 "Input1 Mux" 1
tinymix -D 0 "Input2 Mux" 1
```

LINEIN 双通道输入

```
# 录音通路控件
tinymix -D 0 "LINEINL Switch" 1
tinymix -D 0 "LINEINR Switch" 1
tinymix -D 0 "Input1 Mux" 2
tinymix -D 0 "Input2 Mux" 2
```

播放

LINEOUT 双通道播放

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

HPOUT 双通道播放

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

SPK 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 "SPK Switch" 1
```

说明

MIC 和 LINEOUT 的单端和差分方式，按需操作相应控件即可。

6.5.2 sun8iw21

6.5.2.1 声卡控件

```
Mixer name: 'audiocodec'
Number of controls: 24
ctl   type   num    name                    value
0     ENUM   1     tx hub mode              >Off On
1     ENUM   1     rx sync mode             >Off On
2     ENUM   1     DACDRC                  >Off On
3     ENUM   1     ADCDRC                  >Off On
4     ENUM   1     DACHPF                  >Off On
5     ENUM   1     ADCHPF                  >Off On
6     ENUM   1     ADC1 ADC2 swap           >Off On
7     INT    1    digital volume           63 (dsrange 0->63)
8     INT    1    DAC volume              160 (dsrange 0->255)
9     INT    1    ADC1 volume              160 (dsrange 0->255)
10    INT    1     ADC2 volume             160 (dsrange 0->255)
11    INT    1     MIC1 gain volume         31 (dsrange 0->31)
12    INT    1     MIC2 gain volume         31 (dsrange 0->31)
13    BOOL   1      LINEINL gain volume      Off
14    BOOL   1      LINEINR gain volume      Off
15    INT    1    LINEOUT volume           31 (dsrange 0->31)
16    BOOL   1     MIC1 Switch             Off
17    BOOL   1     MIC2 Switch             Off
18    BOOL   1     LINEIN Switch           Off
19    BOOL   1     LINEOUT Switch           Off
20    BOOL   1     SPK Switch              Off
21    ENUM   1      LINEOUT Output Select       single >differ
22    ENUM   1      MIC1 Input Select          >differ single
23    ENUM   1      MIC2 Input Select          >differ single
```

6.5.2.2 常见使用说明

1. 以 tinyalsa 工具举例说明，具体使用方法见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 "MIC1 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 "MIC2 Switch" 1
tinycap mic.wav -D 0 -c 1 -T 10
```

MIC1&2 双通道输入

```
# 录音通路控件
tinymix -D 0 "MIC1 Switch" 1
tinymix -D 0 "MIC2 Switch" 1
tinycap mic.wav -D 0 -c 2 -T 10
```

LINEIN 双通道输入

```
# 录音通路控件
tinymix -D 0 "LINEIN 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

LINEIN 和 MIC 输入属于 mux 的关系，即只能二选一，不能同时开启 LINEIN 和 MIC 的通路开关。

播放

LINEOUT 单/双通道播放

```
# 播放通路控件
tinymix -D 0 "LINEOUT Switch" 1
tinyplay test_1ch.wav -D 0
# 或
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 "SPK Switch" 1
```

6.5.3 sun8iw11

6.5.3.1 声卡控件

控件列表

```
Mixer name: 'audiocodec'
Number of controls: 58
ctl   type   num    name                    value
0     ENUM   1      DAC DRC Switch             >Off On
1     ENUM   1      DAC HPF Switch             >Off On
2     ENUM   1      ADC DRC Switch             >Off On
3     ENUM   1      ADC HPF Switch             >Off On
4     ENUM   1      tx hub mode               >Off On
5     INT    1     DAC Volume                 63 (range 0->63)
6     INT    1     ADC Gain                  3 (range 0->7)
7     INT    1     MIC1 Gain                 4 (range 0->7)
8     INT    1     MIC2 Gain                 4 (range 0->7)
9     INT    1     MIC1 to OMIX Gain          3 (range 0->7)
10    INT    1     MIC2 to OMIX Gain          3 (range 0->7)
11    INT    1     FMIN to OMIX Gain          3 (range 0->7)
12    INT    1     LINEIN to OMIX Gain        3 (range 0->7)
13    INT    1     LINEINL to ROMIX Gain      3 (range 0->7)
```

```
14  INT   1   LINEINR to LOMIX Gain          3 (range 0->7)
15  INT   1   PHONEOUT Gain                  3 (range 0->7)
16  INT   1   HPOUT Gain                     63 (range 0->63)
17  BOOL  1   MIC1 Switch                    Off
18  BOOL  1   MIC2 Switch                    Off
19  BOOL  1   FMIN Switch                    Off
20  BOOL  1   LINEIN Switch                  Off
21  BOOL  1   HPOUT Switch                   Off
22  BOOL  1   PHONEOUT Switch                Off
23  BOOL  1   SPK Switch                     Off
24  BOOL  1   Left Output Mixer DACL Switch  Off
25  BOOL  1   Left Output Mixer DACR Switch  Off
26  BOOL  1   Left Output Mixer MIC1 Switch  Off
27  BOOL  1   Left Output Mixer MIC2 Switch  Off
28  BOOL  1   Left Output Mixer FMINL Switch  Off
29  BOOL  1   Left Output Mixer LINEINL Switch  Off
30  BOOL  1   Left Output Mixer LINEINLR Switch  Off
31  BOOL  1   Right Output Mixer DACL Switch  Off
32  BOOL  1   Right Output Mixer DACR Switch  Off
33  BOOL  1   Right Output Mixer MIC1 Switch  Off
34  BOOL  1   Right Output Mixer MIC2 Switch  Off
35  BOOL  1   Right Output Mixer FMINR Switch  Off
36  BOOL  1   Right Output Mixer LINEINR Switch  Off
37  BOOL  1   Right Output Mixer LINEINLR Switch  Off
38  BOOL  1   Left Input Mixer MIC1 Switch    Off
39  BOOL  1   Left Input Mixer MIC2 Switch    Off
40  BOOL  1   Left Input Mixer FMINL Switch   Off
41  BOOL  1   Left Input Mixer LINEINL Switch  Off
42  BOOL  1   Left Input Mixer LINEINLR Switch  Off
43  BOOL  1   Left Input Mixer LOMIX Switch   Off
44  BOOL  1   Left Input Mixer ROMIX Switch   Off
45  BOOL  1   Right Input Mixer MIC1 Switch   Off
46  BOOL  1   Right Input Mixer MIC2 Switch   Off
47  BOOL  1   Right Input Mixer FMINR Switch  Off
48  BOOL  1   Right Input Mixer LINEINR Switch  Off
49  BOOL  1   Right Input Mixer LINEINLR Switch  Off
50  BOOL  1   Right Input Mixer LOMIX Switch  Off
51  BOOL  1   Right Input Mixer ROMIX Switch  Off
52  BOOL  1   PHONEOUT Mixer MIC1 Switch     Off
53  BOOL  1   PHONEOUT Mixer MIC2 Switch     Off
54  BOOL  1   PHONEOUT Mixer LOMIX Switch    Off
55  BOOL  1   PHONEOUT Mixer ROMIX Switch    Off
56  ENUM  1   HPL Source                     >DACL LOMIX HPR
57  ENUM  1   HPR Source                     >DACR ROMIX HPL
```

6.5.3.2 常用使用方法

1. 以 tinyalsa 工具举例说明，具体使用方法见tinyalsa 工具 章节。
2. 假设 audiocodec 声卡序号为 0。

录音

```
# MIC1 单通道录音
tinymix -D 0 "MIC1 Switch" 1
tinymix -D 0 "Left Input Mixer MIC1 Switch" 1
tinycap mic.wav -D 0 -c 1 -T 10
```

```
# MIC2 单通道录音
tinymix -D 0 "MIC2 Switch" 1
tinymix -D 0 "Right Input Mixer MIC2 Switch" 1
tinycap mic.wav -D 0 -c 1 -T 10
```

```
# MIC1&2 双通道输入
tinymix -D 0 "MIC1 Switch" 1
tinymix -D 0 "MIC2 Switch" 1
tinymix -D 0 "Left Input Mixer MIC1 Switch" 1
tinymix -D 0 "Right Input Mixer MIC2 Switch" 1
tinycap mic.wav -D 0 -c 2 -T 10
```

播放

```
# HPOUT 单/双通道播放
tinymix -D 0 "HPOUT Switch" 1
tinyplay test_1ch.wav -D 0
# 或
tinyplay test_2ch.wav -D 0
```

```
# PHONEOUT 单/双通道播放（外接喇叭）
tinymix -D 0 "PHONEOUT Switch" 1
tinymix -D 0 "SPK Switch" 1
tinymix -D 0 "Left Output Mixer DACL Switch" 1
tinymix -D 0 "Right Output Mixer DACR Switch" 1
tinymix -D 0 "PHONEOUT Mixer LOMIX Switch" 1
tinymix -D 0 "PHONEOUT Mixer ROMIX Switch" 1
tinyplay test_1ch.wav -D 0
# 或
tinyplay test_2ch.wav -D 0
```

6.5.4 sun8iw22

6.5.4.1 声卡控件

控件列表

```
Mixer name: 'audiocodec'
Number of controls: 8
ctl    type    num    name                    value

0      ENUM    1      tx hub mode             Off
1      ENUM    1      DAC DRC Mode            Off
2      ENUM    1      DAC HPF Mode            Off
3      INT     1      DAC Volume              63
4      INT     1      DACL Volume             160
5      INT     1      LINEOUT Gain            31
6      BOOL    1      LINEOUTL Switch         Off
7      BOOL    1      SPK Switch              Off
```

6.5.4.2 常用使用方法

1. 以 tinyalsa 工具举例说明，具体使用方法见tinyalsa 工具 章节。
2. 假设 audiocodec 声卡序号为 0。

播放

LINEOUT 单通道播放

```
# 播放通路控件
tinymix -D 0 "LINEOUTL Switch" 1
tinymix -D 0 "SPK Switch" 1
tinyplay test_1ch.wav -D 0
```

6.5.5 sun50iw9

6.5.5.1 声卡控件

控件列表

```
Mixer name: 'audiocodec'
Number of controls: 9
ctl   type   num    name                    value
0     ENUM   1      tx hub mode             >Off On
1     INT    1      digital volume          63 (dsrange 0->63)
2     INT    1      lineout volume          31 (dsrange 0->31)
3     BOOL   1      LINEOUT Switch          Off
4     BOOL   1      SPK Switch              Off
5     BOOL   1      OutputL Mixer DACL Switch       Off
6     BOOL   1      OutputL Mixer DACR Switch       Off
7     BOOL   1      OutputR Mixer DACL Switch       Off
8     BOOL   1      OutputR Mixer DACR Switch       Off
```

6.5.5.2 常用使用方法

1. 以 tinyalsa 工具举例说明，具体使用方法见tinyalsa 工具 章节。
2. 假设 audiocodec 声卡序号为 0。

播放

LINEOUT 左单通道播放

```
# 播放通路控件
tinymix -D 0 "OutputL Mixer DACL Switch" 1
tinymix -D 0 "LINEOUT Switch" 1
tinyplay test_1ch.wav -D 0
```

LINEOUT 右单通道播放

```
# 播放通路控件
tinymix -D 0 "OutputR Mixer DACR Switch" 1
tinymix -D 0 "LINEOUT Switch" 1
tinyplay test_1ch.wav -D 0
```

LINEOUT 左右双通道播放

```
# 播放通路控件
tinymix -D 0 "OutputL Mixer DACR Switch" 1
tinymix -D 0 "OutputR Mixer DACL Switch" 1
tinymix -D 0 "LINEOUT Switch" 1
tinyplay test_2ch.wav -D 0
```

LINEOUT 左右双通道交换播放

```
# 播放通路控件
tinymix -D 0 "OutputL Mixer DACL Switch" 1
tinymix -D 0 "OutputR Mixer DACR Switch" 1
tinymix -D 0 "LINEOUT Switch" 1
tinyplay test_2ch.wav -D 0
```

Speaker 播放

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

6.5.6 sun50iw10

6.5.6.1 声卡控件

控件列表

```
Mixer name: 'audiocodec'
Number of controls: 24
ctl   type   num    name                 value
0     ENUM   1      tx hub mode            >Off On
1     ENUM   1      rx sync mode           >Off On
2     ENUM   1      ADCDRC                >Off On
3     ENUM   1      ADCHPF                >Off On
4     ENUM   1      DACDRC                >Off On
5     ENUM   1      DACHPF                >Off On
6     ENUM   1      ADC Swap               >Off On
7     ENUM   1      DAC Swap               >Off On
8     ENUM   1      LINEOUT Output Select          single >differ
9     BOOL   1      Loop ADDA              Off
10    INT    1      ADC1 volume            160 (dsrange 0->255)
11    INT    1      ADC2 volume            160 (dsrange 0->255)
12    INT    1      DAC digital volume       63 (dsrange 0->63)
13    INT    1      DACL volume            160 (dsrange 0->255)
14    INT    1      DACR volume            160 (dsrange 0->255)
15    INT    1      MIC1 volume            31 (dsrange 0->31)
16    INT    1      MIC2 volume            31 (dsrange 0->31)
17    INT    1      LINEOUT volume           31 (dsrange 0->31)
```

```
18    INT    1    HPOUT volume                7 (dsrange 0->7)
19    BOOL   1    MIC1 Switch                 Off
20    BOOL   1    MIC2 Switch                 Off
21    BOOL   1    LINEOUT Switch              Off
22    BOOL   1    HPOUT Switch                Off
23    BOOL   1    SPK Switch                  Off
```

6.5.6.2 常用使用方法

1. 以 tinyalsa 工具举例说明，具体使用方法见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 "MIC1 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 "MIC2 Switch" 1
tinycap mic.wav -D 0 -c 1 -T 10
```

MIC1&2 双通道输入

```
# 录音通路控件
tinymix -D 0 "MIC1 Switch" 1
tinymix -D 0 "MIC2 Switch" 1
tinycap mic.wav -D 0 -c 2 -T 10
```

LINEIN 双通道输入

```
# 录音通路控件
tinymix -D 0 "LINEIN 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

LINEIN 和 MIC 输入属于 mux 的关系，即只能二选一，不能同时开启 LINEIN 和 MIC 的通路开关。

播放

LINEOUT 单/双通道播放

```
# 播放通路控件
tinymix -D 0 "LINEOUT Switch" 1
tinyplay test_1ch.wav -D 0
# 或
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 "SPK Switch" 1
```

6.5.7 sun55iw3

6.5.7.1 声卡控件

控件列表

```
Mixer name: 'audiocodec'
Number of controls: 30
ctl   type   num    name                value
0     ENUM   1      tx hub mode         Off
1     ENUM   1      rx sync mode        Off
2     ENUM   1      DAC DRC Mode        Off
3     ENUM   1      DAC HPF Mode        Off
4     ENUM   1      ADC DRC0 Mode       Off
5     ENUM   1      ADC HPF0 Mode       Off
6     ENUM   1      ADC DRC1 Mode       Off
7     ENUM   1      ADC HPF1 Mode       Off
8     ENUM   1      ADDA Loop Mode      Off
9     ENUM   1      DACL DACR Swap      Off
10    ENUM   1      ADC1 ADC2 Swap      Off
11    ENUM   1      ADC3 ADC4 Swap      Off
12    INT    1      DAC Volume          63
13    INT    1      DACL Volume         160
14    INT    1      DACR Volume         160
15    INT    1      ADC1 Volume         160
16    INT    1      ADC2 Volume         160
17    INT    1      ADC3 Volume         160
18    INT    1      LINEOUT Gain        31
19    INT    1      HPOUT Gain          7
20    INT    1      ADC1 Gain           31
21    INT    1      ADC2 Gain           31
22    INT    1      ADC3 Gain           31
23    BOOL   1      MIC1 Switch         Off
24    BOOL   1      MIC2 Switch         Off
25    BOOL   1      MIC3 Switch         Off
26    BOOL   1      LINEOUTL Switch     Off
27    BOOL   1      LINEOUTR Switch     Off
28    BOOL   1      HPOUT Switch        Off
29    BOOL   1      SPK Switch          Off
```

6.5.7.2 常用使用方法

1. 以 tinyalsa 工具举例说明，具体使用方法见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 "MIC1 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 "MIC2 Switch" 1
tinycap mic.wav -D 0 -c 1 -T 10
```

MIC1&2 双通道输入

```
# 录音通路控件
tinymix -D 0 "MIC1 Switch" 1
tinymix -D 0 "MIC2 Switch" 1
tinycap mic.wav -D 0 -c 2 -T 10
```

播放

LINEOUT 单通道播放

```
# 播放通路控件
tinymix -D 0 "LINEOUTL Switch" 1
tinymix -D 0 "SPK Switch" 1
tinyplay test_1ch.wav -D 0
```

LINEOUT 双通道播放

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

HPOUT 双通道播放

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

6.5.8 sun55iw6

6.5.8.1 声卡控件

控件列表

```
Mixer name: 'audiocodec'
Number of controls: 7
ctl    type    num    name                    value

0      ENUM    1      DAC DRC Mode            Off
1      ENUM    1      DAC HPF Mode            Off
```

```
2    INT   1    DAC Volume                63
3    INT   1    DACL Volume               160
4    INT   1    LINEOUT Gain              31
5    BOOL  1    LINEOUTL Switch              Off
6    BOOL  1    SPK Switch                Off
```

6.5.8.2 常用使用方法

1. 以 tinyalsa 工具举例说明，具体使用方法见tinyalsa 工具 章节。
2. 假设 audiocodec 声卡序号为 0。

播放

LINEOUT 单/双通道播放

```
# 播放通路控件
tinymix -D 0 "LINEOUTL Switch" 1
tinymix -D 0 "SPK Switch" 1
tinyplay test_1ch.wav -D 0
# 或
tinyplay test_2ch.wav -D 0
```

6.5.9 sun300iw1

6.5.9.1 声卡控件

控件列表

```
Mixer name: 'audiocodec'
Number of controls: 18
0    ENUM   1    tx hub mode               >Off On
1    ENUM   1    rx sync mode              >Off On
2    ENUM   1    DAC DRC Mode              >Off On
3    ENUM   1    DAC HPF Mode              >Off On
4    ENUM   1    ADC DRC0 Mode             >Off On
5    ENUM   1    ADC HPF0 Mode             >Off On
6    ENUM   1    ADC DRC1 Mode             >Off On
7    ENUM   1    ADC HPF1 Mode             >Off On
8    ENUM   1    LINEOUT Output Select        DIFFER >SINGLE
9    ENUM   1    ADDA Loop Mode            >Off DAC-to-ADC
10   INT    1    DAC Volume                63 (dsrange 0->63)
11   INT    1    DACL Volume               160 (dsrange 0->255)
12   INT    1    ADC Volume                160 (dsrange 0->255)
13   INT    1    LINEOUT Gain              31 (dsrange 0->31)
14   INT    1    MIC Gain                  31 (dsrange 0->31)
15   BOOL   1    MIC Switch                Off
16   BOOL   1    LINEOUT Switch               Off
17   BOOL   1    SPK Switch                Off
```

6.5.9.2 常用使用方法

1. 以 tinyalsa 工具举例说明，具体使用方法见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 "MIC Switch" 1
tinycap mic.wav -D 0 -c 1 -T 10
```

播放

LINEOUT 单通道播放

```
# 播放通路控件
tinymix -D 0 "LINEOUT Switch" 1
tinymix -D 0 "SPK Switch" 1
tinyplay test_1ch.wav -D 0
```

6.5.10 sun251iw1

6.5.10.1 声卡控件

控件列表

```
Mixer name: 'audiocodec'
Number of controls: 35
ctl   type   num    name                 value

0     ENUM   1      DAC DRC Mode         Off
1     ENUM   1      DAC HPF Mode         Off
2     ENUM   1      ADC DRC0 Mode        Off
3     ENUM   1      ADC HPF0 Mode        Off
4     ENUM   1      ADDA Loop Mode       Off
5     ENUM   1      DACL DACR Swap       Off
6     ENUM   1      ADC1 ADC2 Swap       Off
7     ENUM   1      MIC1 Input Select    differ
8     ENUM   1      MIC2 Input Select    differ
9     INT    1      DAC Volume           63
10    INT    1      DACL Volume          160
11    INT    1      DACR Volume          160
12    INT    1      ADC1 Volume          160
13    INT    1      ADC2 Volume          160
14    ENUM   1      FMINL Gain           0dB
15    ENUM   1      FMINR Gain           0dB
16    ENUM   1      LINEINL Gain         0dB
17    ENUM   1      LINEINR Gain         0dB
18    INT    1      LINEOUT Volume       7
```

```
19   INT   1    HPOUT Gain                7
20   INT   1    ADC1 Gain                 0
21   INT   1    ADC2 Gain                 0
22   BOOL  1     MIC1 Switch               Off
23   BOOL  1     MIC2 Switch               Off
24   BOOL  1     FMINL Switch              Off
25   BOOL  1     FMINR Switch              Off
26   BOOL  1     LINEINL Switch            Off
27   BOOL  1     LINEINR Switch            Off
28   BOOL  1     LINEOUTL Switch             Off
29   BOOL  1     LINEOUTR Switch             Off
30   BOOL  1     HPOUT Switch              Off
31   BOOL  1     SPK Switch              Off
32   ENUM  1     Input1 Mux              NONE
33   ENUM  1     Input2 Mux              NONE
34   INT   2    Soft Volume Master          75 75
```

6.5.10.2 常用使用方法

1. 以 tinyalsa 工具举例说明，具体使用方法见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 "MIC1 Switch" 1
tinymix -D 0 "Input1 Mux" 1
tinycap mic.wav -D 0 -c 1 -T 10
```

MIC1&2 双通道输入

```
# 录音通路控件
tinymix -D 0 "MIC1 Switch" 1
tinymix -D 0 "MIC2 Switch" 1
tinymix -D 0 "Input1 Mux" 1
tinymix -D 0 "Input2 Mux" 1
tinycap mic.wav -D 0 -c 2 -T 10
```

LINEIN 双通道输入

```
# 录音通路控件
tinymix -D 0 "LINEINL Switch" 1
tinymix -D 0 "LINEINR Switch" 1
tinymix -D 0 "Input1 Mux" 3
tinymix -D 0 "Input2 Mux" 3
tinycap linein.wav -D 0 -c 2 -T 10
```

FMIN 双通道输入

```
# 录音通路控件
tinymix -D 0 "FMINL Switch" 1
tinymix -D 0 "FMINR Switch" 1
tinymix -D 0 "Input1 Mux" 2
```

```
tinymix -D 0 "Input2 Mux" 2
tinycap linein.wav -D 0 -c 2 -T 10
```

播放

LINEOUT 单通道播放

```
# 播放通路控件
tinymix -D 0 "LINEOUTL Switch" 1
tinymix -D 0 "SPK Switch" 1
tinyplay test_1ch.wav -D 0
```

LINEOUT 双通道播放

```
# 播放通路控件
tinymix -D 0 "LINEOUTL Switch" 1
tinymix -D 0 "LINEOUTR Switch" 1
tinymix -D 0 "SPK Switch" 1
tinyplay test_1ch.wav -D 0
```

HPOUT 播放

```
# 播放通路控件
tinymix -D 0 "HPOUT Switch" 1
tinyplay test_1ch.wav -D 0
```

6.5.11 sun50iw15

6.5.11.1 声卡控件

控件列表

```
Mixer name: 'audiocodec'
Number of controls: 31
ctl   type   num    name                value
      range/values
0     ENUM   1      tx hub mode         >Off On
1     ENUM   1      rx sync mode        >Off On
2     ENUM   1      DAC DRC Mode          >Off On
3     ENUM   1      DAC HPF Mode          >Off On
4     ENUM   1      ADC DRC Mode          >Off On
5     ENUM   1      ADC HPF Mode          >Off On
6     ENUM   1      DA2AD Loop Mode         >Off DACLR-to-ADC12
7     ENUM   1      DACL DACR Swap        >Off On
8     ENUM   1      ADC1 ADC2 Swap        >Off On
9     INT    1    DAC Volume          63 (dsrange 0->63)
10    INT    1     DACL Volume          160 (dsrange 0->255)
11    INT    1     DACR Volume          160 (dsrange 0->255)
12    INT    1     ADC1 Volume          160 (dsrange 0->255)
13    INT    1     ADC2 Volume          160 (dsrange 0->255)
14    INT    1    LINEOUT Gain          7 (dsrange 0->7)
15    INT    1    HPOUT Gain          7 (dsrange 0->7)
16    ENUM   1     LINEINL Gain          0dB >6dB
17    ENUM   1     LINEINR Gain          0dB >6dB
18    ENUM   1     DAC Src Select        >APB I2S
```

```
19   ENUM   1    ADC Dst Select                >APB I2S
20   ENUM   1    I2S DAC OUT SEL                >Null HPOUT SPK HPOUT-SPK
21   BOOL   1    LINEINL Switch                Off
22   BOOL   1    LINEINR Switch                Off
23   BOOL   1    LINEOUTL Switch                Off
24   BOOL   1    LINEOUTR Switch                Off
25   BOOL   1    HPOUT Switch                Off
26   BOOL   1    SPK Switch                Off
27   BOOL   1    LINEINL Mixer LINEINL1 Switch        Off
28   BOOL   1    LINEINL Mixer LINEINL2 Switch        Off
29   BOOL   1    LINEINR Mixer LINEINR1 Switch        Off
30   BOOL   1    LINEINR Mixer LINEINR2 Switch        Off
```

6.5.11.2 常用使用方法

1. 以 tinyalsa 工具举例说明，具体使用方法见tinyalsa 工具 章节。
2. 假设 audiocodec 声卡序号为 0。

录音

LINEINL 单通道录音

```
# 录音通路控件
tinymix -D 0 "LINEINL Switch" 1
tinymix -D 0 "LINEINL Mixer LINEINL1 Switch" 1
tinycap mic.wav -D 0 -c 1 -T 10
```

LINEINL/R 双通道输入

```
# 录音通路控件
tinymix -D 0 "LINEINL Switch" 1
tinymix -D 0 "LINEINR Switch" 1
tinymix -D 0 "LINEINL Mixer LINEINL1 Switch" 1
tinymix -D 0 "LINEINR Mixer LINEINR1 Switch" 1
tinycap mic.wav -D 0 -c 2 -T 10
```

播放

LINEOUTL 单通道喇叭播放

```
# 播放通路控件
tinymix -D 0 "LINEOUTL Switch" 1
tinymix -D 0 "SPK Switch" 1
tinyplay test_1ch.wav -D 0
```

LINEOUTL/R 双通道喇叭播放

```
# 播放通路控件
tinymix -D 0 "LINEOUTL Switch" 1
tinymix -D 0 "LINEOUTR Switch" 1
tinymix -D 0 "SPK Switch" 1
tinyplay test_1ch.wav -D 0
```

HPOUT 播放

```
# 播放通路控件
tinymix -D 0 "HPOUT Switch" 1
tinyplay test_1ch.wav -D 0
```

6.5.12 sun65iw1

6.5.12.1 声卡控件

控件列表

```
Mixer name: 'audiocodec'
Number of controls: 25
ctl    type    num    name                    value
       range/values
0      ENUM    1      tx hub mode             >Off On
1      ENUM    1      rx sync mode            >Off On
2      ENUM    1      DAC DRC Mode            >Off On
3      ENUM    1      DAC HPF Mode            >Off On
4      ENUM    1      ADC DRC Mode            >Off On
5      ENUM    1      ADC HPF Mode            >Off On
6      ENUM    1      AD2DA Loop Mode         >Off ADC12-to-DACLR
7      ENUM    1      DA2AD Loop Mode         >Off DACLR-to-ADC12
8      ENUM    1      DACL DACR Swap          >Off On
9      ENUM    1      ADC1 ADC2 Swap          >Off On
10     INT     1      DAC Volume              63 (dsrange 0->63)
11     INT     1      DACL Volume             160 (dsrange 0->255)
12     INT     1      DACR Volume             160 (dsrange 0->255)
13     INT     1      ADC1 Volume             160 (dsrange 0->255)
14     INT     1      ADC2 Volume             160 (dsrange 0->255)
15     INT     1      LINEOUT Gain            31 (dsrange 0->31)
16     INT     1      HPOUT Gain              7 (dsrange 0->7)
17     INT     1      ADC1 Gain               31 (dsrange 0->31)
18     INT     1      ADC2 Gain               31 (dsrange 0->31)
19     BOOL    1      MIC1 Switch             Off
20     BOOL    1      MIC2 Switch             Off
21     BOOL    1      LINEOUTL Switch         Off
22     BOOL    1      LINEOUTR Switch         Off
23     BOOL    1      HPOUT Switch            Off
24     BOOL    1      SPK Switch              Off
```

6.5.12.2 常用使用方法

1. 以 tinyalsa 工具举例说明，具体使用方法见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 "MIC1 Switch" 1
tinycap mic.wav -D 0 -c 1 -T 10
```

MIC1/2 双通道输入

```
# 录音通路控件
tinymix -D 0 "MIC1 Switch" 1
tinymix -D 0 "MIC2 Switch" 1
tinycap mic.wav -D 0 -c 2 -T 10
```

播放

LINEOUTL 单通道喇叭播放

```
# 播放通路控件
tinymix -D 0 "LINEOUTL Switch" 1
tinymix -D 0 "SPK Switch" 1
tinyplay test_1ch.wav -D 0
```

LINEOUTL/R 双通道喇叭播放

```
# 播放通路控件
tinymix -D 0 "LINEOUTL Switch" 1
tinymix -D 0 "LINEOUTR Switch" 1
tinymix -D 0 "SPK Switch" 1
tinyplay test_1ch.wav -D 0
```

HPOUT 播放

```
# 播放通路控件
tinymix -D 0 "HPOUT Switch" 1
tinyplay test_1ch.wav -D 0
```

著作权声明

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





（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。