



A40i 系列 & T3 系列 Linux Audio 开发指南

版本号: 1.0

发布日期: 2022.11.09

版本历史

版本号	日期	制/修订人	内容描述
1.0	2022.11.09	AWA2077	初始版本

目 录

1 前言	1
1.1 文档简介	1
1.2 目标读者	1
1.3 适用范围	1
1.4 相关术语	1
2 模块介绍	3
2.1 sun8iw11 音频接口	3
2.1.1 时钟树	3
2.1.2 AudioCodec	4
2.1.2.1 驱动特性	4
2.1.2.2 音频流通路	5
2.1.2.3 Device Tree 配置	5
2.1.2.4 board.dts 板级配置	8
2.1.2.5 kernel menuconfig 配置	10
2.1.2.6 加载 & 卸载方法 (若编译为 ko)	11
2.1.2.7 声卡控件	11
2.1.2.8 常用使用方法	14
2.1.3 I2S/PCM	15
2.1.3.1 驱动特性	15
2.1.3.2 音频流通路	15
2.1.3.3 Device Tree 配置	16
2.1.3.4 board.dts 板级配置	18
2.1.3.5 kernel menuconfig 配置	21
2.1.3.6 加载 & 卸载方法 (若编译为 ko)	22
2.1.3.7 声卡控件	22
2.1.3.8 常用使用方法	23
2.1.3.9 HDMI 声卡控件	23
2.1.3.10 HDMI 常用使用方法	23
2.1.4 S/PDIF	24
2.1.4.1 驱动特性	24
2.1.4.2 音频流通路	24
2.1.4.3 Device Tree 配置	24
2.1.4.4 board.dts 板级配置	26
2.1.4.5 kernel menuconfig 配置	26
2.1.4.6 加载 & 卸载方法 (若编译为 ko)	27
2.1.4.7 声卡控件	28
2.1.4.8 常用使用方法	28

2.2 GPIO 功能复用配置	28
3 模块接口说明	30
3.1 源文件列表	30
3.1.1 驱动源码	30
3.1.2 设备树	31
3.1.3 源码说明	32
3.1.3.1 platform 层 -> 公共部分	32
3.1.3.2 platform 层 -> AudioCodec	32
3.1.3.3 platform 层 -> I2S/PCM	32
3.1.3.4 platform 层 -> AHUB	32
3.1.3.5 platform 层 -> SPDIF	33
3.1.3.6 platform 层 -> DMIC	33
3.1.3.7 codec 层 -> 公共部分	33
3.1.3.8 codec 层 -> AudioCodec	33
3.1.3.9 machine 层	33
3.1.3.10 特殊功能组件	34
3.1.3.11 平台基础资源	34
3.2 软件框图	34
3.3 关键数据结构	37
3.3.1 pcm 数据类结构体	38
3.3.1.1 sunxi_dma_params	38
3.3.2 platform 类结构体	38
3.3.2.1 sunxi_daudio	38
3.3.2.2 sunxi_spdif	38
3.3.2.3 sunxi_dmic	38
3.3.3 codec 类结构体	39
3.3.3.1 sunxi_codec	39
3.3.4 machine 类结构体	39
3.3.4.1 asoc_simple_priv	39
3.4 接口说明	39
3.4.1 pcm 相关接口	39
3.4.1.1 sunxi_pcm_new [linux-4.9~linux-5.4]	39
3.4.1.2 sunxi_pcm_construct [linux-5.10~linux-5.15]	40
3.4.1.3 sunxi_pcm_free [linux-4.9~linux-5.4]	40
3.4.1.4 sunxi_pcm_destruct [linux-5.10~linux-5.15]	40
3.4.1.5 sunxi_pcm_open	41
3.4.1.6 sunxi_pcm_close	41
3.4.1.7 sunxi_pcm_ioctl	41
3.4.1.8 sunxi_pcm_hw_params	41
3.4.1.9 sunxi_pcm_hw_free	42
3.4.1.10 sunxi_pcm_trigger	42
3.4.1.11 sunxi_pcm_pointer	42

3.4.1.12	sunxi_pcm_hw_params_raw	43
3.4.1.13	sunxi_pcm_hw_free_raw	43
3.4.1.14	sunxi_pcm_prepare_raw	43
3.4.1.15	sunxi_pcm_trigger_raw	44
3.4.1.16	sunxi_pcm_pointer_raw	44
3.4.1.17	sunxi_pcm_copy_raw	44
3.4.1.18	sunxi_pcm_mmap	45
3.4.2	platform 层接口 -> AudioCodec	45
3.4.2.1	sunxi_aaudio_dai_probe	45
3.4.2.2	sunxi_aaudio_dai_startup	45
3.4.3	platform 层接口 -> I2S/PCM	46
3.4.3.1	sunxi_daudio_component_probe	46
3.4.3.2	sunxi_daudio_dai_suspend [linux-4.9]	46
3.4.3.3	sunxi_daudio_component_suspend [linux-5.4~linux-5.15]	46
3.4.3.4	sunxi_daudio_dai_resume [linux-4.9]	47
3.4.3.5	sunxi_daudio_component_resume [linux-5.4~linux-5.15]	47
3.4.3.6	sunxi_daudio_dai_probe	47
3.4.3.7	sunxi_daudio_dai_remove	47
3.4.3.8	sunxi_daudio_dai_set_pll	48
3.4.3.9	sunxi_daudio_dai_set_sysclk	48
3.4.3.10	sunxi_daudio_dai_set_bclk_ratio	49
3.4.3.11	sunxi_daudio_dai_set_fmt	49
3.4.3.12	sunxi_daudio_dai_set_tdm_slot	49
3.4.3.13	sunxi_daudio_dai_startup	50
3.4.3.14	sunxi_daudio_dai_hw_params	50
3.4.3.15	sunxi_daudio_dai_prepare	50
3.4.3.16	sunxi_daudio_dai_trigger	51
3.4.3.17	sunxi_daudio_dai_shutdown	51
3.4.4	platform 层接口 -> AHUB	51
3.4.4.1	sunxi_ahub_dai_suspend [linux-4.9]	51
3.4.4.2	sunxi_ahub_dai_suspend [linux-5.4~linux-5.15]	52
3.4.4.3	sunxi_ahub_dai_resume [linux-4.9]	52
3.4.4.4	sunxi_ahub_dai_resume [linux-5.4~linux-5.15]	52
3.4.4.5	sunxi_ahub_probe	52
3.4.4.6	sunxi_ahub_dai_suspend [linux-4.9]	53
3.4.4.7	sunxi_ahub_suspend [linux-5.4~linux-5.15]	53
3.4.4.8	sunxi_ahub_dai_resume [linux-4.9]	53
3.4.4.9	sunxi_ahub_resume [linux-5.4~linux-5.15]	54
3.4.4.10	sunxi_ahub_dai_probe	54
3.4.4.11	sunxi_ahub_dai_remove	54
3.4.4.12	sunxi_ahub_dai_set_pll	54
3.4.4.13	sunxi_ahub_dai_set_sysclk	55
3.4.4.14	sunxi_ahub_dai_set_bclk_ratio	55

3.4.4.15	sunxi_ahub_dai_set_fmt	56
3.4.4.16	sunxi_ahub_dai_set_tdm_slot	56
3.4.4.17	sunxi_ahub_dai_startup	56
3.4.4.18	sunxi_ahub_dai_hw_params	57
3.4.4.19	sunxi_ahub_dai_hw_free	57
3.4.4.20	sunxi_ahub_dai_prepare	57
3.4.4.21	sunxi_ahub_dai_trigger	58
3.4.4.22	sunxi_ahub_dai_shutdown	58
3.4.5	platform 层接口 -> SPDIF	58
3.4.5.1	sunxi_spdif_component_probe	58
3.4.5.2	sunxi_spdif_dai_suspend [linux-4.9]	59
3.4.5.3	sunxi_spdif_component_suspend [linux-5.4~linux-5.15]	59
3.4.5.4	sunxi_spdif_dai_resume [linux-4.9]	59
3.4.5.5	sunxi_spdif_component_resume [linux-5.4~linux-5.15]	59
3.4.5.6	sunxi_spdif_dai_probe	60
3.4.5.7	sunxi_spdif_dai_remove	60
3.4.5.8	sunxi_spdif_dai_set_pll	60
3.4.5.9	sunxi_spdif_dai_set_clkdiv	61
3.4.5.10	sunxi_spdif_dai_startup	61
3.4.5.11	sunxi_spdif_dai_hw_params	62
3.4.5.12	sunxi_spdif_dai_prepare	62
3.4.5.13	sunxi_spdif_dai_trigger	62
3.4.5.14	sunxi_spdif_dai_shutdown	63
3.4.6	platform 层接口 -> DMIC	63
3.4.6.1	sunxi_dmic_component_probe	63
3.4.6.2	sunxi_dmic_dai_suspend [linux-4.9]	63
3.4.6.3	sunxi_dmic_component_suspend [linux-5.4~linux-5.15]	64
3.4.6.4	sunxi_dmic_dai_resume [linux-4.9]	64
3.4.6.5	sunxi_dmic_component_resume [linux-5.4~linux-5.15]	64
3.4.6.6	sunxi_dmic_dai_probe	64
3.4.6.7	sunxi_dmic_dai_set_pll	65
3.4.6.8	sunxi_dmic_dai_startup	65
3.4.6.9	sunxi_dmic_dai_hw_params	65
3.4.6.10	sunxi_dmic_dai_prepare	66
3.4.6.11	sunxi_dmic_dai_trigger	66
3.4.6.12	sunxi_dmic_dai_shutdown	66
3.4.7	codec 层接口 -> 公共部分	67
3.4.7.1	snd_sunxi_pa_pin_init	67
3.4.7.2	snd_sunxi_pa_pin_exit	67
3.4.7.3	snd_sunxi_pa_pin_enable	67
3.4.7.4	snd_sunxi_pa_pin_disable	68
3.4.8	codec 层接口 -> AudioCodec	68
3.4.8.1	sunxi_internal_codec_probe	68

3.4.8.2	sunxi_internal_codec_remove	68
3.4.8.3	sunxi_internal_codec_suspend [linux-4.9]	69
3.4.8.4	sunxi_internal_codec_suspend [linux-5.4~linux-5.15]	69
3.4.8.5	sunxi_internal_codec_resume [linux-4.9]	69
3.4.8.6	sunxi_internal_codec_resume [linux-5.4~linux-5.15]	70
3.4.8.7	sunxi_internal_codec_dai_set_pll	70
3.4.8.8	sunxi_internal_codec_dai_startup	70
3.4.8.9	sunxi_internal_codec_dai_hw_params	71
3.4.8.10	sunxi_internal_codec_dai_prepare	71
3.4.8.11	sunxi_internal_codec_dai_trigger	71
3.4.8.12	sunxi_internal_codec_dai_shutdown	72
3.4.9	machine 层接口	72
3.4.9.1	simple_soc_probe	72
3.4.9.2	simple_soc_remove	72
3.4.9.3	simple_dai_link_of	73
3.4.9.4	asoc_simple_parse_widgets	73
3.4.9.5	asoc_simple_parse_routing	73
3.4.9.6	asoc_simple_parse_pin_switches	74
3.4.9.7	asoc_simple_parse_daifmt	74
3.4.9.8	asoc_simple_parse_daistream	74
3.4.9.9	asoc_simple_parse_tdm_slot	75
3.4.9.10	asoc_simple_parse_tdm_clk	75
3.4.9.11	asoc_simple_set_dailink_name	75
3.4.9.12	asoc_simple_dai_init	76
3.4.9.13	asoc_simple_hw_params	76
3.5	软件调试接口	76
3.5.1	snd_sunxi_debug_show_reg	76
3.5.2	snd_sunxi_debug_store_reg	77
4	模块使用	78
4.1	kernel menuconfig 配置	78
4.2	加载与卸载方法	78
4.3	声卡设备查看	79
4.4	声卡控件	80
4.5	声卡测试工具使用	80
4.5.1	tinypalsa 工具	80
4.5.1.1	tinymix	81
4.5.1.2	tinypplay	81
4.5.1.3	tinypcap	81
4.5.1.4	tinypcminfo	82
4.5.1.5	tinyploop	82
4.5.2	alsa-utils 工具	83
4.5.2.1	aplay	83

4.5.2.2 arecord	84
4.5.2.3 amixer	85
4.6 I2S 外挂 codec	86
4.6.1 硬件连接	86
4.6.2 获取外部 codec I2S 协议格式	87
5 FAQ	88
5.1 调试方法	88
5.1.1 调试工具	88
5.1.2 调试节点	88
5.2 常见问题	89
5.2.1 录音或播放变速	89
5.2.2 AudioCodec 输入输出无声音	89
5.2.3 DMIC 录音异常（静音/通道移位）	89

表 格

表 1-1	适用产品列表	1
表 1-2	硬件术语	2
表 1-3	软件术语	2
表 2-1	AudioCodec codec 节点配置项 (linux-5.10)	7
表 2-2	AudioCodec codec_plat 节点配置项 (linux-5.10)	7
表 2-3	AudioCodec codec_mach 节点配置项 (linux-5.10)	7
表 2-4	AudioCodec 模块板级配置项	8
表 2-5	特殊功能类控件说明	12
表 2-6	音量调节类控件说明	13
表 2-7	通路开关类控件说明	13
表 2-8	I2S/PCM daudio(n)_plat 节点配置项 (linux-5.10)	18
表 2-9	I2S/PCM daudio(n)_mach 节点配置项 (linux-5.10)	18
表 2-10	I2S/PCM 模块板级配置项	20
表 2-11	控件说明	22
表 2-12	控件说明	23
表 2-13	S/PDIF spdif_plat 节点配置项 (linux-5.10)	25
表 2-14	S/PDIF spdif_mach 节点配置项 (linux-5.10)	26
表 2-15	S/PDIF 模块板级配置项	26
表 2-16	控件说明	28
表 2-17	GPIO 功能复用配置项	29
表 2-18	模块引脚组定义说明 (linux-5.4, linux-5.10)	29

1 前言

1.1 文档简介

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

- 1、“音频时钟树-> 驱动特性-> 音频流通路-> 设备树配置-> 编译配置及加载-> 声卡控件介绍-> 常见使用方法”等部分，介绍音频驱动的测试验证和使用；
- 2、“源码结构-> 驱动框架-> 关键数据结构-> 接口说明”等部分，介绍音频驱动的二次开发；
- 3、“声卡查看-> 声卡测试工具-> 外挂 codec”等部分，介绍声卡模块的测试验证和使用；
- 4、“FAQ”章节：调试方法及常见问题汇总。

1.2 目标读者

音频系统相关人员。

1.3 适用范围

表 1-1: 适用产品列表

产品名称	内核版本	驱动文件
sun8iw11	Linux-5.10	bsp/drivers/sound/platform/*

说明

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

1.4 相关术语

表 1-2: 硬件术语

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

表 1-3: 软件术语

相关术语	解释说明
ALSA	Advanced Linux Sound Architecture。
ASoC	ALSA System on Chip。
DAPM	动态音频电源管理。
samplebit	样本精度，记录音频数据最基本的单位，常见的有 16 位。
channel	通道数，该参数为 1 表示单声道，2 表示立体声，大于 2 表示多声道。
rate	采样率，每秒钟采样次数，该次数是针对帧而言。
frame	帧，记录了一个声音单元，其长度为样本长度与通道数的乘积。
period size	每次硬件中断处理音频数据的帧数。
period count	处理完一个 buffer 数据所需的硬件中断次数。
buffer size	数据缓冲区大小 (period size * period count)。
DRC	音频输出动态范围控制。
HPF	高通滤波。
XRUN	音频流异常状态，分为 underrun 和 overrun 两种状态。
交错模式	一种音频数据记录模式，数据以连续帧形式存放 (帧 1_L, 帧 1_R, 帧 2_L, 帧 2_R, ...)。
非交错模式	一种音频数据记录模式，数据是以连续通道形式存放 (L-帧 1, L-帧 2, ..., R-帧 2, R-帧 2, ...)。
tinyalsa	在 Linux 内核中与 ALSA 接口对接的库，可用于基本播录。
alsalib	在 Linux 内核中与 ALSA 接口对接的库，可用于基本播录， 并可与常见音频算法组合使用。

2 模块介绍

在 sunxi 中，从 Linux 软件上通常存在 5 类音频设备，如下。

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

以上每一类音频设备均适配 ASoC 架构。

2.1 sun8iw11 音频接口

AudioCodec x1
I2S/PCM x3
S/PDIF x1

2.1.1 时钟树

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。

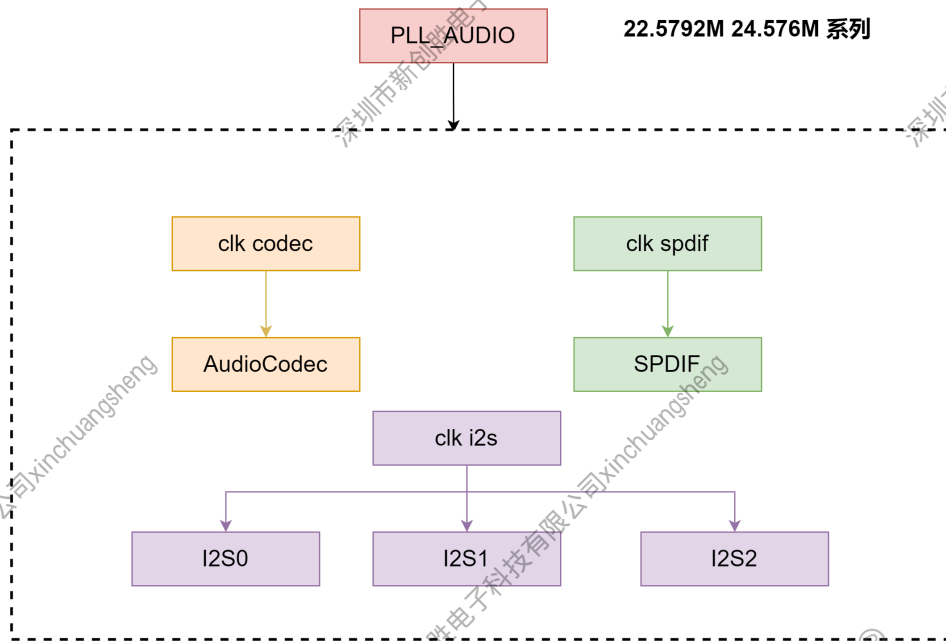


图 2-1: 时钟树 sun8iw11

2.1.2 AudioCodec

2.1.2.1 驱动特性

- 支持多种采样率格式
 - 播放：8~192kHz
 - 录音：8~48kHz
- 支持多通道播放和录音
 - 播放：1~2ch
 - 录音：1~2ch
- 支持 16/24/32bit 数据精度（硬件支持 16bit/24bit）
- 支持硬件 HPF、DRC 算法
- 支持的物理接口
 - 录音接口：MIC x 2, FMIN, LINEIN
 - 播放接口：LINEOUT, HPOUT, PHONEOUT(差分输出)
- 支持同时 playback 和 capture (全双工模式)
- 支持多声卡同源播放

2.1.2.2 音频流通路

播放流

```

Playback ----> DACL --> HPOUTL
Playback ----> DACL --> Left Output Mixer ----> HPOUTL
Playback ----> DACR -----^          \--> LINEOUTL
Playback ----> MIC1 -----^          \--> LINEOUTR
Playback ----> MIC2 -----^          \--> PHONEOUT Mixer --> PHONEOUT
Playback ----> FMINL -----^
Playback ----> LINEINL -----^
Playback ----> LINEINLR -----^
HPOUTR -----> HPOUTL

Playback --> DACR --> HPOUTR
Playback --> DACL --> Right Output Mixer ----> HPOUTR
Playback --> DACR -----^          \--> LINEOUTL
Playback --> MIC1 -----^          \--> LINEOUTR
Playback --> MIC2 -----^          \--> PHONEOUT Mixer --> PHONEOUT
Playback --> FMINR -----^
Playback --> LINEINR -----^
Playback --> LINEINLR -----^
HPOUTL -----> HPOUTR

MIC1 -----> PHONEOUT Mixer --> PHONEOUT
MIC2 -----> PHONEOUT Mixer --> PHONEOUT
LINEOUTL --> LINEOUTR

```

录音流

```

MIC1 -----> Left Input Mixer --> ADCL --> Capture
MIC2 -----^
FMINL -----^
LINEINL -----^
LINEINLR -----^
Left Output Mixer ---^
Right Output Mixer --^

MIC1 -----> Right Input Mixer --> ADCR --> Capture
MIC2 -----^
FMINR -----^
LINEINR -----^
LINEINLR -----^
Left Output Mixer ---^
Right Output Mixer --^

```

2.1.2.3 Device Tree 配置

```

codec:codec@1c22c00 {
    #sound-dai-cells    = <0>;
    compatible          = "allwinner,sunxi-snd-codec";
    reg                 = <0x0 0x01c22c00 0x0 0x300>;
    resets              = <&ccu RST_BUS_AC>;
    clocks               = <&ccu CLK_BUS_AC_DIG>,
                        <&ccu CLK_PLL_AUDIO>;
}

```

```

                                <&ccu CLK_AC_DIGITAL_GATE>;
clock-names                    = "clk_bus_audio",
                                "clk_pll_audio",
                                "clk_ac_digital_gate";
status = "disabled";
}

codec_plat:codec_plat {
    #sound-dai-cells           = <0>;
    compatible                 = "allwinner,sunxi-snd-plat-audio";
    adc-txdata                 = <0x01c22c18>;
    dac-txdata                 = <0x01c22c20>;
    dmas                       = <&dma 19>, <&dma 19>;
    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", "FMIN", "LINEIN",
                                "HPOUT", "PHONEOUT", "SPK";
    soundcard-mach,routing      = "MIC1_PIN", "MIC1",
                                "MIC2_PIN", "MIC2",
                                "FMINL_PIN", "FMIN",
                                "FMINR_PIN", "FMIN",
                                "LINEINL_PIN", "LINEIN",
                                "LINEINR_PIN", "LINEIN",
                                "HPOUT", "HPOUTL_PIN",
                                "HPOUT", "HPOUTR_PIN",
                                "PHONEOUT", "PHONEOUTP_PIN",
                                "PHONEOUT", "PHONEOUTN_PIN",
                                "SPK", "HPOUTL_PIN",
                                "SPK", "HPOUTR_PIN",
                                "SPK", "PHONEOUTP_PIN",
                                "SPK", "PHONEOUTN_PIN";

    status = "disabled";
    soundcard-mach,cpu {
        sound-dai = <&codec_plat>;
    };
    soundcard-mach,codec {
        sound-dai = <&codec>;
        soundcard-mach,pll-fs = <1>;
    };
};

```

配置项说明：

AudioCodec 模块由 3 个设备树节点构建。

1、ASoC 层 codec: codec

表 2-1: AudioCodec codec 节点配置项 (linux-5.10)

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

2、ASoC 层 platform: codec_plat

表 2-2: AudioCodec codec_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。
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-txdata	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 2-3: AudioCodec codec_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。
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）。
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）。

2.1.2.4 board.dts 板级配置

```
&codec {
    avcc-external;
    avcc-supply = <&reg_aldo3>;
    avcc-vol    = <3000000>;
    dvcc-external;
    dvcc-supply = <&reg_dcdc1>;
    dvcc-vol    = <3000000>;

    dac-vol      = <63>;          /* default value:63 range:0->63 */
    adc-gain      = <3>;          /* default value:3 range:0->7 */
    mic1-gain     = <4>;          /* default value:4 range:0->7 */
    mic2-gain     = <4>;          /* default value:4 range:0->7 */
    mic1-to-omix-gain = <3>;      /* default value:3 range:0->7 */
    mic2-to-omix-gain = <3>;      /* default value:3 range:0->7 */
    fmin-to-omix-gain = <3>;      /* default value:3 range:0->7 */
    linein-to-omix-gain = <3>;    /* default value:3 range:0->7 */
    lineinl-to-romix-gain = <3>; /* default value:3 range:0->7 */
    lineinr-to-lomix-gain = <3>; /* default value:3 range:0->7 */
    phoneout-gain = <3>;          /* default value:3 range:0->7 */
    hpout-gain    = <63>;          /* default value:59 range:0->63 */

    pa-pin-max    = <1>;
    pa-pin-0      = <&pio PB 16 GPIO_ACTIVE_HIGH>;
    pa-pin-level-0 = <1>;
    pa-pin-msleep-0 = <0>;
    /* pa-pin-1    = <&pio PB 3 GPIO_ACTIVE_HIGH>; */
    /* pa-pin-level-1 = <1>; */
    /* pa-pin-msleep-1 = <0>; */
    status = "okay";
};

&codec_plat {
    status = "okay";
};

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

配置项说明：

表 2-4: AudioCodec 模块板级配置项

配置项名称	配置值范围	配置项说明
status	"okay", "disabled"	使能或关闭该节点驱动。
avcc-external	注释为 false, 反之 为 true	AVCC 电源是否为外围电路提供。
avcc-supply	注释,	AVCC 若为外部 PMU 供电,

配置项名称	配置值范围	配置项说明
	引用 PMU 提供的电源节点	可选择该项指定对应的 PMU 电源。
avcc-vol	u32(缺省值: 1800000)	AVCC 电压值设定, 单位 uV, 需符合实际硬件需求。
dvcc-external	注释为 false, 反之为 ture	DVCC 电源是否为外围电路提供。
dvcc-supply	注释, 引用 PMU 提供的电源节点	DVCC 若为外部 PMU 供电, 可选择该项指定对应的 PMU 电源。
dvcc-vol	u32(缺省值: 1800000)	DVCC 电压值设定, 单位 uV, 需符合实际硬件需求。
dac-vol	0~63 (-73.08->0dB)	DAC 数字音量。
adc-gain	0~7 (-4.5->6dB)	ADC 模拟增益。
mic1-gain	0~7 (24->42dB)	MIC1 模拟增益。
mic2-gain	0~7 (24->42dB)	MIC2 模拟增益。
mic1-to-omix-gain	0~7 (-4.5->6dB)	MIC1 到 Output Mixer 的模拟增益。
mic2-to-omix-gain	0~7 (-4.5->6dB)	MIC2 到 Output Mixer 的模拟增益。
fmin-to-omix-gain	0~7 (-4.5->6dB)	FMIN 到 Output Mixer 的模拟增益。
linein-to-omix-gain	0~7 (-4.5->6dB)	LINEIN 到 Output Mixer 的模拟增益。
lineinl-to-romix-gain	0~7 (-4.5->6dB)	LINEINL 到 Right Output Mixer 的模拟增益。
lineinr-to-lomix-gain	0~7 (-4.5->6dB)	LINEINR 到 Left Output Mixer 的模拟增益。
dac-drc-en	注释为 false, 反之为 ture	DAC-DRC 的使能开关。
dac-hpf-en	注释为 false, 反之为 ture	DAC-HPF 的使能开关。
adc-drc-en	注释为 false, 反之为 ture	ADC-DRC 的使能开关。
adc-hpf-en	注释为 false, 反之为 ture	ADC-HPF 的使能开关。
phoneout-gain	0~7 (-4.5->6dB)	PHONEOUT 模拟增益。
hpout-gain	0~63 (-63->0dB)	HPOUT 模拟增益。
pa-pin-max	u32(正常为 1 或 2)	标定外部功放芯片使能引脚数量。
pa-pin-(n)	pio 提供的引脚节点	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(正常小于 200)	设置功放芯片使能所需。

配置项名称	配置值范围	配置项说明
		的 sleep 时长，用于规避 pop 声，单位 ms。

说明

以上配置项并非所有板型均必须包含，以板型具体规格为准，进行裁剪。

2.1.2.5 kernel menuconfig 配置

```
Allwinner BSP --->
Device Drivers --->
SOUND Drivers ---->
Platform drivers --->
  <M> Allwinner AAUDIO support
    jack detection method --->
      <X> none
  <M> Allwinner Function Components
  <M> Components Debug
```

配置项说明：

选择需要的模块，可选择直接编译进内核（Y），也可编译成模块（M）。

声卡配置：

- Allwinner AAUDIO support
 - AudioCodec 模块

特定功能配置：

- Components Debug
 - 调试节点功能组件（查看音频寄存器）

编译模式：

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

2.1.2.6 加载 & 卸载方法（若编译为 ko）

sunxi 音频驱动模块分五大类驱动如下。

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

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

AudioCodec 声卡加载顺序如下。

```
# 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
```

2.1.2.7 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                     >0ff 0n
1      ENUM     1      DAC HPF Switch                     >0ff 0n
2      ENUM     1      ADC DRC Switch                     >0ff 0n
3      ENUM     1      ADC HPF Switch                     >0ff 0n
4      ENUM     1      tx hub mode                         >0ff 0n
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	B00L	1	LINEIN Switch	Off
21	B00L	1	HPOUT Switch	Off
22	B00L	1	PHONEOUT Switch	Off
23	B00L	1	SPK Switch	Off
24	B00L	1	Left Output Mixer DACL Switch	Off
25	B00L	1	Left Output Mixer DACR Switch	Off
26	B00L	1	Left Output Mixer MIC1 Switch	Off
27	B00L	1	Left Output Mixer MIC2 Switch	Off
28	B00L	1	Left Output Mixer FMINL Switch	Off
29	B00L	1	Left Output Mixer LINEINL Switch	Off
30	B00L	1	Left Output Mixer LINEINLR Switch	Off
31	B00L	1	Right Output Mixer DACL Switch	Off
32	B00L	1	Right Output Mixer DACR Switch	Off
33	B00L	1	Right Output Mixer MIC1 Switch	Off
34	B00L	1	Right Output Mixer MIC2 Switch	Off
35	B00L	1	Right Output Mixer FMINR Switch	Off
36	B00L	1	Right Output Mixer LINEINR Switch	Off
37	B00L	1	Right Output Mixer LINEINLR Switch	Off
38	B00L	1	Left Input Mixer MIC1 Switch	Off
39	B00L	1	Left Input Mixer MIC2 Switch	Off
40	B00L	1	Left Input Mixer FMINL Switch	Off
41	B00L	1	Left Input Mixer LINEINL Switch	Off
42	B00L	1	Left Input Mixer LINEINLR Switch	Off
43	B00L	1	Left Input Mixer LOMIX Switch	Off
44	B00L	1	Left Input Mixer ROMIX Switch	Off
45	B00L	1	Right Input Mixer MIC1 Switch	Off
46	B00L	1	Right Input Mixer MIC2 Switch	Off
47	B00L	1	Right Input Mixer FMINR Switch	Off
48	B00L	1	Right Input Mixer LINEINR Switch	Off
49	B00L	1	Right Input Mixer LINEINLR Switch	Off
50	B00L	1	Right Input Mixer LOMIX Switch	Off
51	B00L	1	Right Input Mixer ROMIX Switch	Off
52	B00L	1	PHONEOUT Mixer MIC1 Switch	Off
53	B00L	1	PHONEOUT Mixer MIC2 Switch	Off
54	B00L	1	PHONEOUT Mixer LOMIX Switch	Off
55	B00L	1	PHONEOUT Mixer ROMIX Switch	Off
56	ENUM	1	HPL Source	>DACL LOMIX HPR
57	ENUM	1	HPR Source	>DACR ROMIX HPL

表 2-5: 特殊功能类控件说明

控件名称	功能	数值
DAC DRC Switch	DAC DRC 开关	Off;On
DAC HPF Switch	DAC HPF 开关	Off;On
ADC DRC Switch	ADC DRC 开关	Off;On
ADC HPF Switch	ADC HPF 开关	Off;On
tx hub mode	同源播放开关	Off;On
SPK Switch	功放开关	Off;On
HPL Source	HPOUTL 的数据源	DACL LOMIX HPR
HPR Source	HPOUTR 的数据源	DACR ROMIX HPL

表 2-6: 音量调节类控件说明

控件名称	功能	数值
dac-vol	DAC 数字音量	0->63 (-73.08->0dB)
adc-gain	ADC 模拟增益	0->7 (-4.5->6dB)
mic1-gain	MIC1 模拟增益	0->7 (24->42dB)
mic2-gain	MIC2 模拟增益	0->7 (24->42dB)
mic1-to-omix-gain	MIC1 到 Output Mixer 的模拟增益	0->7 (-4.5->6dB)
mic2-to-omix-gain	MIC2 到 Output Mixer 的模拟增益	0->7 (-4.5->6dB))
fmin-to-omix-gain	FMIN 到 Output Mixer 的模拟增益	0->7 (-4.5->6dB)
linein-to-omix-gain	LINEIN 到 Output Mixer 的模拟增益	0->7 (-4.5->6dB)
lineinl-to-romix-gain	LINEINL 到 Right Output Mixer 的模拟增益	0->7 (-4.5->6dB)
lineinr-to-lomix-gain	LINEINR 到 Left Output Mixer 的模拟增益	0->7 (-4.5->6dB)
phoneout-gain	PHONEOUT 模拟增益	0->7 (-4.5->6dB)
hpout-gain	HPOUT 模拟增益	0->63 (-63->0dB)

表 2-7: 通路开关类控件说明

控件名称	功能	数值
MIC1 Switch	MIC1 通路开关	Off;On
MIC2 Switch	MIC2 通路开关	Off;On
FMIN Switch	FMIN 通路开关	Off;On
LINEIN Switch	LINEIN 通路开关	Off;On
HPOUT Switch	HPOUT 通路开关	Off;On
PHONEOUT Switch	PHONEOUT 通路开关	Off;On
Left Output Mixer DACL Switch	DACL->LOMIX 通路开关	Off;On
Left Output Mixer DACR Switch	DACR->LOMIX 通路开关	Off;On
Left Output Mixer MIC1 Switch	MIC1->LOMIX 通路开关	Off;On
Left Output Mixer MIC2 Switch	MIC2->LOMIX 通路开关	Off;On
Left Output Mixer FMINL Switch	FMINL->LOMIX 通路开关	Off;On
Left Output Mixer LINEINL Switch	LINEINL->LOMIX 通路开关	Off;On
Left Output Mixer LINEINLR Switch	LINEINLR->LOMIX 通路开关	Off;On
Right Output Mixer DACL Switch	DACL->ROMIX 通路开关	Off;On
Right Output Mixer DACR Switch	DACR->ROMIX 通路开关	Off;On
Right Output Mixer MIC1 Switch	MIC1->ROMIX 通路开关	Off;On
Right Output Mixer MIC2 Switch	MIC2->ROMIX 通路开关	Off;On
Right Output Mixer FMINR Switch	FMINR->ROMIX 通路开关	Off;On
Right Output Mixer LINEINR Switch	LINEINR->ROMIX 通路开关	Off;On
Right Output Mixer LINEINLR Switch	LINEINLR->ROMIX 通路开关	Off;On
Left Input Mixer MIC1 Switch	MIC1->LIMIX 通路开关	Off;On
Left Input Mixer MIC2 Switch	MIC2->LIMIX 通路开关	Off;On
Left Input Mixer FMINL Switch	FMINL->LIMIX 通路开关	Off;On

控件名称	功能	数值
Left Input Mixer LINEINL Switch	LINEINL->LIMIX 通路开关	Off;On
Left Input Mixer LINEINLR Switch	LINEINLR->LIMIX 通路开关	Off;On
Left Input Mixer LOMIX Switch	LOMIX->LIMIX 通路开关	Off;On
Left Input Mixer ROMIX Switch	ROMIX->LIMIX 通路开关	Off;On
Right Input Mixer MIC1 Switch	MIC1->RIMIX 通路开关	Off;On
Right Input Mixer MIC2 Switch	MIC2->RIMIX 通路开关	Off;On
Right Input Mixer FMINR Switch	FMINR->RIMIX 通路开关	Off;On
Right Input Mixer LINEINR Switch	LINEINR->RIMIX 通路开关	Off;On
Right Input Mixer LINEINLR Switch	LINEINLR->RIMIX 通路开关	Off;On
Right Input Mixer LOMIX Switch	LOMIX->RIMIX 通路开关	Off;On
Right Input Mixer ROMIX Switch	ROMIX->RIMIX 通路开关	Off;On
PHONEOUT Mixer MIC1 Switch	MIC1->PMIX 通路开关	Off;On
PHONEOUT Mixer MIC2 Switch	MIC2->PMIX 通路开关	Off;On
PHONEOUT Mixer LOMIX Switch	LOMIX->PMIX 通路开关	Off;On
PHONEOUT Mixer ROMIX Switch	ROMIX->PMIX 通路开关	Off;On

2.1.2.8 常用使用方法

1. 以 tinyalsa 工具举例说明，具体使用方法见 [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
```

2.1.3 I2S/PCM

2.1.3.1 驱动特性

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

2.1.3.2 音频流通路

播放流

```
Playback → I2S DOUT
```

录音流

I2S DIN → Capture

回环流

Playback → Capture

2.1.3.3 Device Tree 配置

```
audio0_plat:audio0_plat@1c22000 {
    #sound-dai-cells      = <0>;
    compatible            = "allwinner,sun8iw11-daudio";
    reg                   = <0x0 0x01C22000 0x0 0x58>;
    resets                 = <&ccu RST_BUS_DAUDIO0>;
    clocks                 = <&ccu CLK_BUS_DAUDIO0>,
                          <&ccu CLK_PLL_AUDIO0>,
                          <&ccu CLK_DAUDIO0>;
    clock-names           = "clk_bus_daudio",
                          "clk_pll_audio",
                          "clk_daudio";
    dmas                   = <&dma 3>, <&dma 3>;
    dma-names             = "tx", "rx";
    playback-cma           = <128>;
    capture-cma           = <128>;
    tx-fifo-size          = <128>;
    rx-fifo-size          = <128>;
    status                 = "disabled";
};
```

```
audio0_mach:audio0_mach {
    compatible            = "allwinner,sunxi-snd-mach";
    soundcard-mach,name    = "snddaudio0";
    soundcard-mach,format  = "i2s";
    soundcard-mach,slot-num = <2>;
    soundcard-mach,slot-width = <32>;
    status                 = "disabled";
    soundcard-mach,cpu {
        sound-dai = <&audio0_plat>;
    };
    soundcard-mach,codec {
    };
};
```

```
audio1_plat:audio1_plat@1c22400 {
    #sound-dai-cells      = <0>;
    compatible            = "allwinner,sun8iw11-daudio";
    reg                   = <0x0 0x01C22400 0x0 0x58>;
    resets                 = <&ccu RST_BUS_DAUDIO1>;
    clocks                 = <&ccu CLK_BUS_DAUDIO1>,
                          <&ccu CLK_PLL_AUDIO0>,
                          <&ccu CLK_DAUDIO1>;
    clock-names           = "clk_bus_daudio",
                          "clk_pll_audio",
                          "clk_daudio";
    dmas                   = <&dma 4>, <&dma 4>;
    dma-names             = "tx", "rx";
    playback-cma           = <128>;
    capture-cma           = <128>;
```

```
tx-fifo-size    = <128>;
rx-fifo-size    = <128>;
status = "disabled";
};

daudio1_mach:daudio1_mach {
    compatible = "allwinner,sunxi-snd-mach";
    soundcard-mach,name      = "snddaudio1";
    soundcard-mach,format    = "i2s";
    soundcard-mach,slot-num   = <2>;
    soundcard-mach,slot-width = <32>;
    status = "disabled";
    soundcard-mach,cpu {
        sound-dai = <&daudio1_plat>;
    };
    soundcard-mach,codec {
    };
};
```

```
daudio2_plat:audio2_plat@1c22800 {
    #sound-dai-cells = <0>;
    compatible = "allwinner,sun8iw11-daudio";
    reg = <0x0 0x01c22800 0x0 0x58>;
    resets = <&ccu RST_BUS_DAUDIO2>;
    clocks = <&ccu CLK_BUS_DAUDIO2>,
            <&ccu CLK_PLL_AUDIO>,
            <&ccu CLK_DAUDIO2>;
    clock-names = "clk_bus_audio",
                  "clk_pll_audio",
                  "clk_daudio";
    dmas = <&dma 6>, <&dma 6>;
    dma-names = "tx", "rx";
    playback-cma = <128>;
    capture-cma = <128>;
    tx-fifo-size = <128>;
    rx-fifo-size = <128>;
    status = "disabled";
};
```

```
daudio2_mach:audio2_mach{
    compatible = "allwinner,sunxi-snd-mach";
    soundcard-mach,name      = "sndhdmi";
    soundcard-mach,format    = "i2s";
    soundcard-mach,slot-num   = <2>;
    soundcard-mach,slot-width = <32>;
    status = "disabled";
    soundcard-mach,cpu {
        sound-dai = <&daudio2_plat>;
    };
    soundcard-mach,codec {
    };
};
```

配置项说明：

I2S/PCM 模块由 2 个或 3 个设备树节点构建。

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

2、ASoC 层 platform: daudio(n)_plat

表 2-8: I2S/PCM daudio(n)_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 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。
dmass	设置模块所绑定的 DMA 通道号。
dma-names	对 dmass 属性内容进行名称定义，用于辅助 dmass 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 2-9: I2S/PCM daudio(n)_mach 节点配置项 (linux-5.10)

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。
pll-fs	指定模块时钟源频率（24.576M or 22.5792M * pll-fs）。

2.1.3.4 board.dts 板级配置

```

&daudio0_plat {
    tdm-num = <0>;
    tx-pin = <0>;
    rx-pin = <0>;
    /* pinctrl-used; */
    /* pinctrl-names      = "default","sleep"; */
    /* pinctrl-0          = <&daudio0_pins_a>; */
    /* pinctrl-1          = <&daudio0_pins_b>; */
    tx-hub-en;
    status = "okay";
};

&daudio0_mach {
    soundcard-mach,format = "i2s";

```

```

soundcard-mach,frame-master = <&audio0_cpu>;
soundcard-mach,bitclock-master = <&audio0_cpu>;
/* soundcard-mach,frame-inversion; */
/* soundcard-mach,bitclock-inversion; */
soundcard-mach,slot-num = <2>;
soundcard-mach,slot-width = <32>;
status = "okay";
audio0_cpu: soundcard-mach,cpu {
    sound-dai = <&audio0_plat>;
    /* note: pll freq = 24.576M or 22.5792M * pll-fs */
    soundcard-mach,pll-fs = <1>;
    soundcard-mach,mclk-fp;
    soundcard-mach,mclk-fs = <0>;
};
audio0_codec: soundcard-mach,codec {
};
};

```

```

&audio1_plat {
    tdm-num = <0>;
    tx-pin = <0>;
    rx-pin = <0>;
    /* pinctrl-used; */
    /* pinctrl-names = "default","sleep"; */
    /* pinctrl-0 = <&audio1_pins_a>; */
    /* pinctrl-1 = <&audio1_pins_b>; */
    tx-hub-en;
    status = "okay";
};

&audio1_mach {
    soundcard-mach,format = "i2s";
    soundcard-mach,frame-master = <&audio1_cpu>;
    soundcard-mach,bitclock-master = <&audio1_cpu>;
    /* soundcard-mach,frame-inversion; */
    /* soundcard-mach,bitclock-inversion; */
    soundcard-mach,slot-num = <2>;
    soundcard-mach,slot-width = <32>;
    status = "okay";
    audio1_cpu: soundcard-mach,cpu {
        sound-dai = <&audio1_plat>;
        /* note: pll freq = 24.576M or 22.5792M * pll-fs */
        soundcard-mach,pll-fs = <1>;
        soundcard-mach,mclk-fp;
        soundcard-mach,mclk-fs = <0>;
    };
    audio1_codec: soundcard-mach,codec {
    };
};
};

```

```

&audio2_plat {
    tdm-num = <0>;
    tx-pin = <0 1 2 3>;
    rx-pin = <0>;
    dai-type = "hdmi";
    /* pinctrl-used; */
    /* pinctrl-names = "default","sleep"; */
    /* pinctrl-0 = <&audio2_pins_a>; */
    /* pinctrl-1 = <&audio2_pins_b>; */
    tx-hub-en;
};

```

```

};
status = "okay";
};

&audio2_mach {
    soundcard-mach,format = "i2s";
    soundcard-mach,frame-master = <&audio2_cpu>;
    soundcard-mach,bitclock-master = <&audio2_cpu>;
    /* soundcard-mach,frame-inversion; */
    /* soundcard-mach,bitclock-inversion; */
    soundcard-mach,slot-num = <2>;
    soundcard-mach,slot-width = <32>;
    status = "okay";
    audio2_cpu: soundcard-mach,cpu {
        sound-dai = <&audio2_plat>;
        /* note: pll freq = 24.576M or 22.5792M * pll-fs */
        soundcard-mach,pll-fs = <1>;
        soundcard-mach,mclk-fp;
        soundcard-mach,mclk-fs = <0>;
    };
    audio2_codec: soundcard-mach,codec {
    };
};

```

配置项说明：

表 2-10: I2S/PCM 模块板级配置项

配置项名称	配置值范围	配置项说明
status	"okay", "disabled"	使能或关闭该节点驱动。
tdm-num	0~2	指定 I2S 序号，需和 audio(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。
tx-hub-en	注释为 false, 反之为 true	选择是否注册 txhub 控件。
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 true	LRCK 信号是否翻转。
bitclock-inversion	注释为 false, 反之为 true	BCLK 信号是否翻转。
slot-num	1~8	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 true	true: 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 =

配置项名称	配置值范围	配置项说明
		mclk-fs * 12.288M or 11.2896M 采样率倍数：MCLK = mclk-fs * pcm rate。

2.1.3.5 kernel menuconfig 配置

```
Allwinner BSP --->
  Device Drivers --->
    SOUND Drivers --->
      Platform drivers --->
        <M> Allwinner DAUDIO Support
          <M> Allwinner HDMIAUDIO Support
        <M> Allwinner Function Components
          <M> Components Debug
```

配置项说明：

选择需要的模块，可选择直接编译进内核（Y），也可编译成模块（M）。

声卡配置：

- Allwinner DAUDIO Support
 - I2S/PCM 模块
- Allwinner HDMIAUDIO Support
 - HDMI 模块

特定功能配置：

- Components Debug
 - 调节点功能组件（查看音频寄存器）

编译模式：

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

2.1.3.6 加载 & 卸载方法（若编译为 ko）

sunxi 音频驱动模块分五大类驱动如下。

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

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

I2S/PCM 声卡加载顺序如下。

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

# ASoC platfrom 驱动 和 ASoC codec 驱动(NULL)
insmod snd_soc_sunxi_daudio.ko

# ASoC machine 驱动
insmod snd_soc_sunxi_machine.ko
```

2.1.3.7 声卡控件

控件列表

```
Mixer name: 'snddaudio0'
Number of controls: 2
ctl      type    num    name                      value
0        ENUM    1      tx hub mode               >0ff 0n
1        BOOL    1      loopback debug            Off
```

表 2-11: 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

2.1.3.8 常用使用方法

1. 以 tinyalsa 工具举例说明，具体使用方法见 [tinyalsa 工具](#) 章节。
2. 假设 snddaudio0 声卡序号为 0。

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

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

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

2.1.3.9 HDMI 声卡控件

控件列表

```
Mixer name: 'sndhdm1'
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 2-12: 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

2.1.3.10 HDMI 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 sndhdm1 声卡序号为 0。

播放


```
# sndhdmi 声卡播放
tinyplay test.wav -D 0

# sndhdmi 数据透传
tinymix -D 0 "audio data format" DTS
tinyplay test.dts -D 0
```

2.1.4 S/PDIF

2.1.4.1 驱动特性

- 支持多种采样率格式
 - 播放：22.05~192kHz
 - 录音：22.05~192kHz
- 支持多通道播放和录音
 - 播放：1~2ch
 - 录音：1~2ch
- 支持 16bit/20bit/24bit/32bit 数据精度（硬件支持 16/20/24bit）
- 支持 loopback 回环模式
- 支持同时 playback 和 capture（全双工模式）
- 支持 IEC-60958 协议
- 支持多声卡同源播放

2.1.4.2 音频流通路

播放流

Playback → S/PDIF DOUT

录音流

S/PDIF DIN → Capture

2.1.4.3 Device Tree 配置

```
spdif_plat:spdif_plat@1c21000 {
    #sound-dai-cells = <0>;
    compatible = "allwinner,sun8iw11-spdif";
    reg = <0x0 0x01c21000 0x0 0x38>;
    resets = <&ccu RST_BUS_SPDIF>;
    clocks = <&ccu CLK_BUS_SPDIF>,
            <&ccu CLK_PLL_AUDIO>,
            <&ccu CLK_SPDIF>;
}
```

```

clock-names      = "clk_bus_spdif",
                  "clk_pll_audio",
                  "clk_spdif";
dmas              = <&dma 2>, <&dma 2>;
dma-names        = "tx", "rx";
playback-cma     = <128>;
capture-cma      = <128>;
tx-fifo-size     = <128>;
rx-fifo-size     = <128>;
status = "disabled";
};

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

```

配置项说明：

S/PDIF 模块由 2 个设备树节点构建。

1、ASoC 层 codec: 无，绑定虚拟 codec 节点。

2、ASoC 层 platform: spdif_plat

表 2-13: S/PDIF spdif_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 S/PDIF 寄存器起始地址和地址长度。
resets	设置 S/PDIF 所需的复位时钟。
clocks	设置 S/PDIF 所需的时钟源和模块时钟。
clock-names	对 clocks 属性内容进行名称定义，用于辅助 clocks 属性获取。
playback-cma	设置播放流 DMA 申请 size 大小，为 (2^n)Kbyte，单位 Kb。
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: spdif_mach

表 2-14: S/PDIF spdif_mach 节点配置项 (linux-5.10)

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）。
pll-fs	指定模块时钟源频率（24.576M or 22.5792M * pll-fs）。

2.1.4.4 board.dts 板级配置

```
&spdif_plat {
    pinctrl-used;
    pinctrl-names = "default", "sleep";
    pinctrl-0 = <&spdif_pins_a>;
    pinctrl-1 = <&spdif_pins_b>;
    tx-hub-en;
    status = "okay";
};

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

配置项说明：

表 2-15: S/PDIF 模块板级配置项

配置项名称	配置值范围	配置项说明
status	"okay", "disabled"	使能或关闭该节点驱动。
tx-hub-en	注释为 false, 反之为 true	选择是否注册 txhub 控件。

2.1.4.5 kernel menuconfig 配置

```
Allwinner BSP --->
  Device Drivers --->
    SOUND Drivers --->
      Platform drivers --->
        <M> Allwinner SPDIF Support
        <M> Allwinner Function Components
        <M> Components Debug
```

配置项说明：

选择需要的模块，可选择直接编译进内核（Y），也可编译成模块（M）。

声卡配置：

- Allwinner SPDIF Support
 - S/PDIF 模块

特定功能配置：

- Components Debug
 - 调试节点功能组件（查看音频寄存器）

编译模式：

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

2.1.4.6 加载 & 卸载方法（若编译为 ko）

sunxi 音频驱动模块分五大类驱动如下。

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

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

S/PDIF 声卡加载顺序如下。

```
# PCM 驱动
insmod snd_soc_sunxi_pcm.ko

# ASoC platfrom 驱动 和 ASoC codec 驱动(NULL)
insmod snd_soc_sunxi_spdif.ko

# ASoC machine 驱动
insmod snd_soc_sunxi_machine.ko
```

2.1.4.7 声卡控件

控件列表

```
Mixer name: 'sndspdif'
Number of controls: 2
ctl      type    num    name                      value
0        ENUM    1      tx hub mode               Off
1        ENUM    1      audio data format        >PCM RAW
```

表 2-16: 控件说明

控件名称	功能	数值
tx hub mode	同源播放开关	Off;On
audio data format	设置音频数据格式	0~1(对应控件 value 枚举值)

2.1.4.8 常用使用方法

1. 以 tinyalsa 工具举例说明，具体使用方法见[tinyalsa 工具](#) 章节。
2. 假设 spdif 声卡序号为 0。

播放

```
tinypplay test.wav -D 0
```

2.2 GPIO 功能复用配置

- AudioCodec 模块：
 - 所用引脚功能均固化，无需进行 pin 功能复用配置。
- I2S/PCM、AHUB、S/PDIF、DMIC 模块：
 - 可选择不进行 pin 功能复用配置，该情况仍可生成声卡，但引脚无实际功能输入输出。

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

配置项说明：

表 2-17: 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 true	是否使用引脚复用功能。
pinctrl-names	"default", "sleep"	对 pinctrl 属性内容进行名称定义, 用于辅助 pinctrl 属性获取。
pinctrl-0	模块 pin 功能复用节点	对应 pinctrl-names 第 0 个属性。
pinctrl-1	模块 pin 功能复用节点	对应 pinctrl-names 第 1 个属性。

表 2-18: 模块引脚组定义说明 (linux-5.4, linux-5.10)

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

3 模块接口说明

3.1 源文件列表

3.1.1 驱动源码

```
# linux-4.9 : linux4.9/sound/soc/sunxi_v2  
# linux-5.4 : linux-5.4/sound/soc/sunxi_v2  
# linux-5.10: bsp/drivers/sound/platform  
# linux-5.15: bsp/drivers/sound/platform
```

```
├─ Kconfig  
├─ Makefile  
├─ platforms  
│   ├── snd_sun50iw10_daudio.h  
│   ├── snd_sun50iw10_dmic.h  
│   ├── snd_sun50iw10_spdif.h  
│   ├── snd_sun50iw9_dmic.h  
│   ├── snd_sun50iw9_spdif.h  
│   ├── snd_sun55iw3_daudio.h  
│   ├── snd_sun55iw3_dmic.h  
│   ├── snd_sun55iw3_spdif.h  
│   ├── snd_sun8iw11_daudio.h  
│   ├── snd_sun8iw11_spdif.h  
│   ├── snd_sun8iw21_daudio.h  
│   └── snd_sun8iw21_dmic.h  
├─ snd_sun50iw10_codec.c  
├─ snd_sun50iw10_codec.h  
├─ snd_sun50iw9_codec.c  
├─ snd_sun50iw9_codec.h  
├─ snd_sun8iw11_codec.c  
├─ snd_sun8iw11_codec.h  
├─ snd_sun8iw21_codec.c  
├─ snd_sun8iw21_codec.h  
├─ snd_sunxi_aaudio.c  
├─ snd_sunxi_ahub.c  
├─ snd_sunxi_ahub.h  
├─ snd_sunxi_ahub_dam.c  
├─ snd_sunxi_ahub_dam.h  
├─ snd_sunxi_common.c  
├─ snd_sunxi_common.h  
├─ snd_sunxi_daudio.c  
├─ snd_sunxi_daudio.h  
├─ snd_sunxi_dmic.c  
├─ snd_sunxi_dmic.h  
├─ snd_sunxi_dummy_codec.c  
├─ snd_sunxi_hdmi.c  
├─ snd_sunxi_hdmi.h  
└─ snd_sunxi_jack.h
```

```
|— snd_sunxi_jack_codec.c
|— snd_sunxi_jack_extcon.c
|— snd_sunxi_log.h
|— snd_sunxi_mach.c
|— snd_sunxi_mach.h
|— snd_sunxi_mach_utils.c
|— snd_sunxi_mach_utils.h
|— snd_sunxi_pcm.c
|— snd_sunxi_pcm.h
|— snd_sunxi_pcm_hdmi.c
|— snd_sunxi_rxsync.c
|— snd_sunxi_rxsync.h
|— snd_sunxi_spdif.c
|— snd_sunxi_spdif.h
|— snd_sunxi_spdif_rx61937.c
|— snd_sunxi_spdif_rx61937.h
```

3.1.2 设备树

Device Tree 配置文件路径

```
# linux-4.9
linux-4.9/arch/arm/boot/dts/{platform}.dtsi
linux-4.9/arch/arm64/boot/dts/sunxi/{platform}.dtsi

# linux-5.4
linux-5.4/arch/arm/boot/dts/{platform}.dtsi
linux-5.4/arch/arm64/boot/dts/sunxi/{platform}.dtsi
linux-5.4/arch/riscv/boot/dts/sunxi/{platform}.dtsi

# linux-5.10
bsp/configs/linux-5.10/{platform}.dtsi

# linux-5.15
bsp/configs/linux-5.15/{platform}.dtsi

{platform}: 芯片平台型号, 如 sun8iw21。
```

board.dts 板级配置文件路径

```
# linux-4.9
device/config/chips/{chip}/configs/{board}/linux-4.9/board.dts

# linux-5.4
device/config/chips/{chip}/configs/{board}/linux-5.4/board.dts

# linux-5.10
device/config/chips/{chip}/configs/{board}/linux-5.10/board.dts

# linux-5.15
device/config/chips/{chip}/configs/{board}/linux-5.15/board.dts

{chip} : 芯片型号
{board}: 具体板型
```


3.1.3 源码说明

3.1.3.1 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_hdmi.c  
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。

3.1.3.2 platform 层 -> AudioCodec

```
snd_sunxi_aaudio.c
```

负责 AudioCodec 模块 DMA 相关配置。

3.1.3.3 platform 层 -> I2S/PCM

```
snd_sunxi_daudio.c  
snd_sunxi_daudio.h
```

负责 I2S/PCM 模块硬件参数、DMA 相关配置。

3.1.3.4 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。

3.1.3.5 platform 层 -> SPDIF

```
snd_sunxi_spdif.c  
snd_sunxi_spdif.h
```

负责 SPDIF 模块硬件参数、DMA 相关配置。

3.1.3.6 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。

3.1.3.7 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

3.1.3.8 codec 层 -> AudioCodec

```
snd_sun50iw9_codec.c  
snd_sun50iw9_codec.h
```

负责 AudioCodec 模块硬件参数配置 (sun50iw9)。

3.1.3.9 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。

3.1.3.10 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。

3.1.3.11 平台基础资源

```
platforms/snd_sun50iw9_spdif.h --> aw1823 基础平台资源配置 (S/PDIF)  
platforms/snd_sun50iw9_dmic.h --> aw1823 基础平台资源配置 (DMIC)
```

3.2 软件框图

ALSA 音频架构

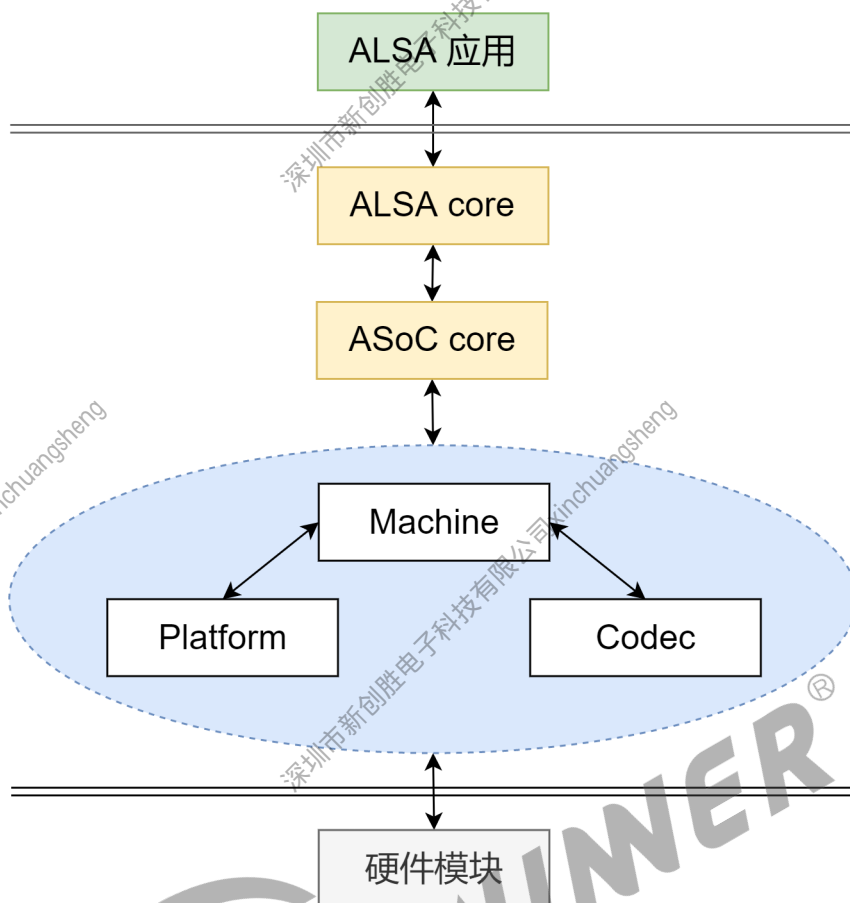


图 3-1: ALSA 音频架构

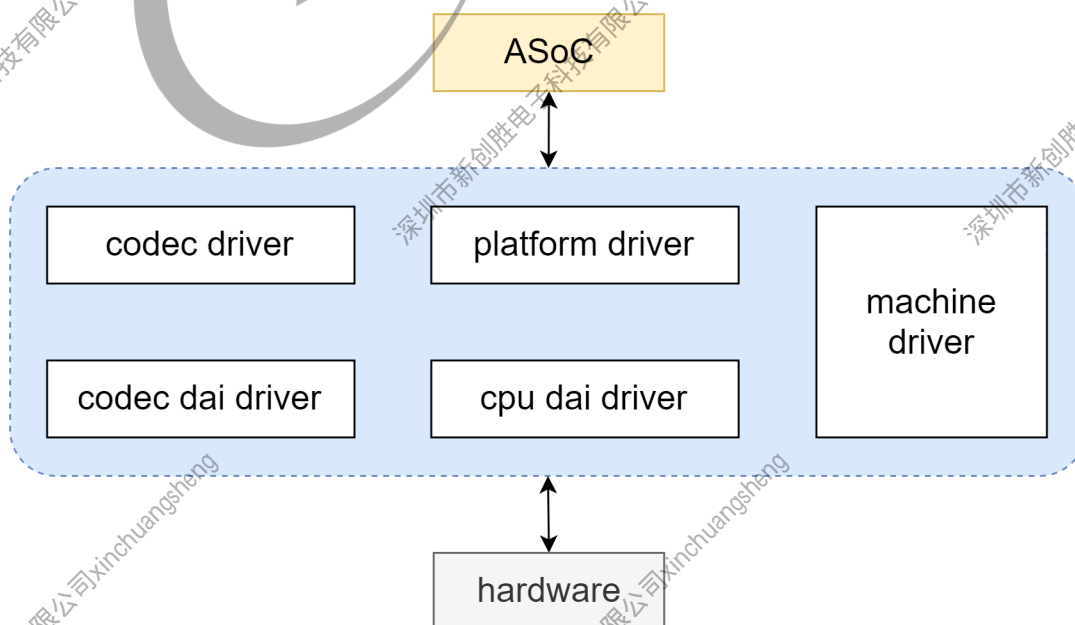
ASoC 驱动框架

图 3-2: ASoC 驱动框架

基于 ASoC 的 sunxi 驱动框架

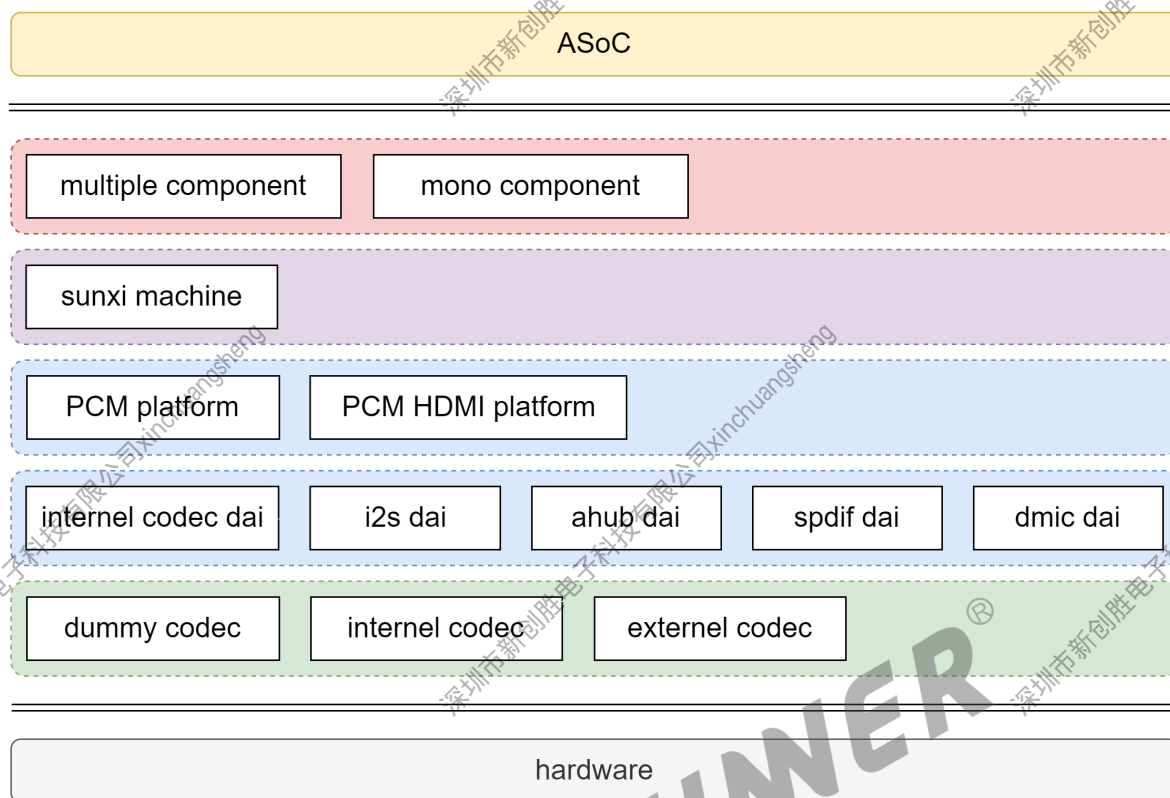


图 3-3: ASoC SUNXI 驱动框架

基于 ASoC 的 sunxi 代码结构

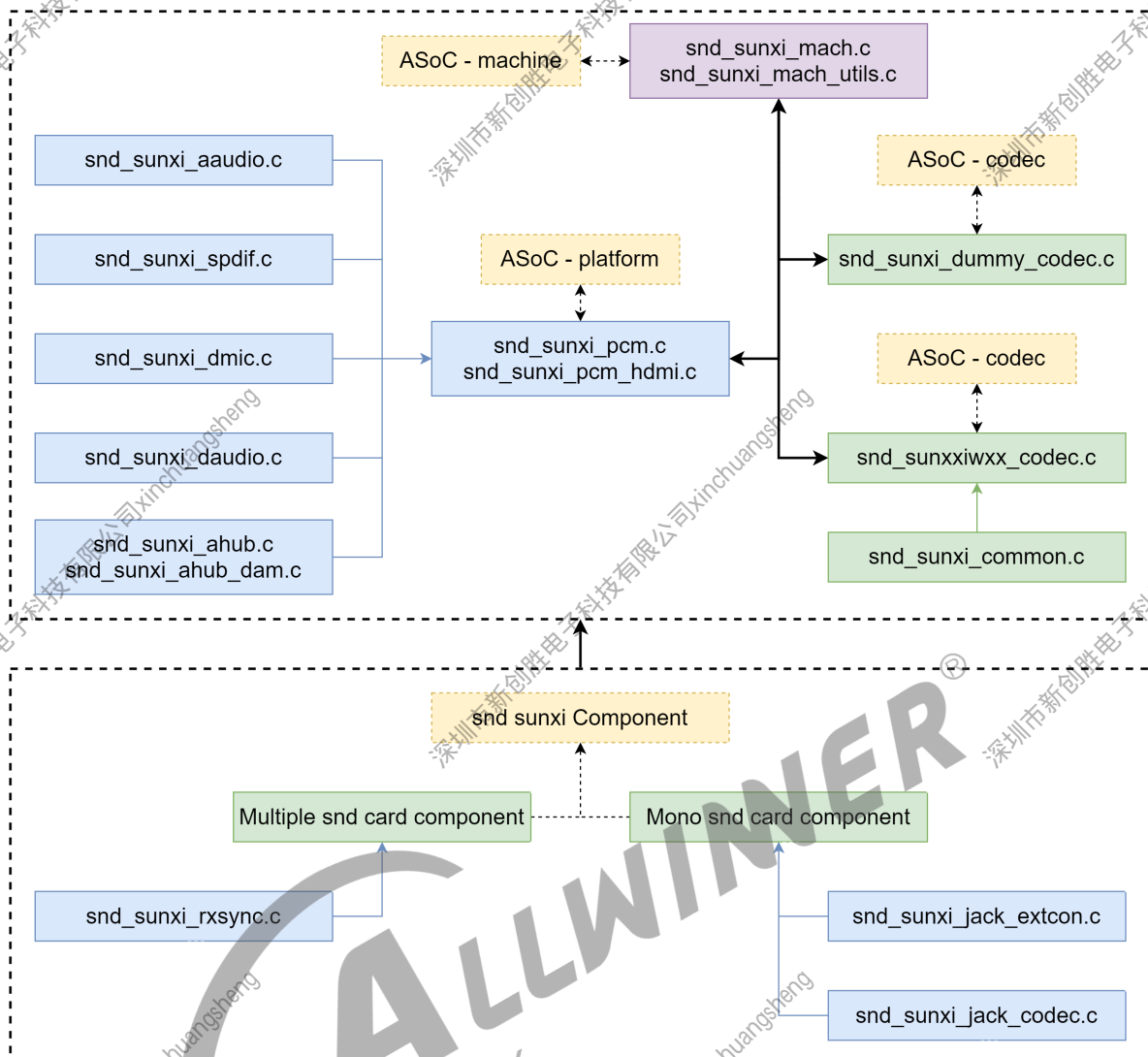


图 3-4: ASoC SUNXI 代码结构

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

3.3.1.1 sunxi_dma_params

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

定义 audio dai dma 相关参数。

3.3.2 platform 类结构体

3.3.2.1 sunxi_daudio

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

I2S/PCM 模块总结结构体，包含基础平台资源、特定功能私有参数。

3.3.2.2 sunxi_spdif

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

S/PDIF 模块总结结构体，包含基础平台资源、特定功能私有参数。

3.3.2.3 sunxi_dmic

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

DMIC 模块总结结构体，包含基础平台资源、特定功能私有参数。

3.3.3 codec 类结构体

3.3.3.1 sunxi_codec

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

AudioCodec 模块总结构体，包含基础平台资源、特定功能私有参数。

3.3.4 machine 类结构体

3.3.4.1 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

3.4.1.1 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

3.4.1.2 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

3.4.1.3 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

3.4.1.4 sunxi_pcm_destruct [linux-5.10~linux-5.15]

- 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

3.4.1.5 sunxi_pcm_open

- 函数原型：

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

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

3.4.1.6 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

3.4.1.7 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

3.4.1.8 sunxi_pcm_hw_params

- 函数原型：

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

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

3.4.1.9 sunxi_pcm_hw_free

- 函数原型：

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

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

3.4.1.10 sunxi_pcm_trigger

- 函数原型：

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

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

3.4.1.11 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

3.4.1.12 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

3.4.1.13 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

3.4.1.14 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

3.4.1.15 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

3.4.1.16 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

3.4.1.17 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

3.4.1.18 sunxi_pcm_mmap

- 函数原型：

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

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

3.4.2 platform 层接口 -> AudioCodec

3.4.2.1 sunxi_aaudio_dai_probe

- 函数原型：

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

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

3.4.2.2 sunxi_aaudio_dai_startup

- 函数原型：

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

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

3.4.3 platform 层接口 -> I2S/PCM

3.4.3.1 sunxi_daudio_component_probe

- 函数原型：

```
int sunxi_daudio_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

3.4.3.2 sunxi_daudio_dai_suspend [linux-4.9]

- 函数原型：

```
int sunxi_daudio_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

3.4.3.3 sunxi_daudio_component_suspend [linux-5.4~linux-5.15]

- 函数原型：

```
int sunxi_daudio_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

3.4.3.4 sunxi_daudio_dai_resume [linux-4.9]

- 函数原型：

```
int sunxi_daudio_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

3.4.3.5 sunxi_daudio_component_resume [linux-5.4~linux-5.15]

- 函数原型：

```
int sunxi_daudio_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

3.4.3.6 sunxi_daudio_dai_probe

- 函数原型：

```
int sunxi_daudio_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

3.4.3.7 sunxi_daudio_dai_remove

- 函数原型：

```
int sunxi_daudio_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

3.4.3.8 sunxi_daudio_dai_set_pll

- 函数原型：

```
int sunxi_daudio_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

3.4.3.9 sunxi_daudio_dai_set_sysclk

- 函数原型：

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

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

3.4.3.10 sunxi_daudio_dai_set_bclk_ratio

- 函数原型：

```
int sunxi_daudio_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

3.4.3.11 sunxi_daudio_dai_set_fmt

- 函数原型：

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

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

3.4.3.12 sunxi_daudio_dai_set_tdm_slot

- 函数原型：

```
int sunxi_daudio_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

3.4.3.13 sunxi_daudio_dai_startup

- 函数原型：

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

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

3.4.3.14 sunxi_daudio_dai_hw_params

- 函数原型：

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

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

3.4.3.15 sunxi_daudio_dai_prepare

- 函数原型：

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

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

3.4.3.16 sunxi_daudio_dai_trigger

- 函数原型：

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

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

3.4.3.17 sunxi_daudio_dai_shutdown

- 函数原型：

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

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

3.4.4 platform 层接口 -> AHUB

3.4.4.1 sunxi_ahub_dam_dai_suspend [linux-4.9]

- 函数原型：

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

- 功能描述：AHUB 模块休眠（关闭时钟）
- 参数说明：
 - dai: cpu dai 信息
- 返回值：0-成功，其它-失败

3.4.4.2 sunxi_ahub_dam_suspend [linux-5.4~linux-5.15]

- 函数原型：

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

- 功能描述：AHUB 模块休眠（关闭时钟）
- 参数说明：
 - component: platform 层组件
- 返回值：0-成功，其它-失败

3.4.4.3 sunxi_ahub_dam_dai_resume [linux-4.9]

- 函数原型：

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

- 功能描述：AHUB 模块唤醒（开启时钟）
- 参数说明：
 - dai: cpu dai 信息
- 返回值：0-成功，其它-失败

3.4.4.4 sunxi_ahub_dam_resume [linux-5.4~linux-5.15]

- 函数原型：

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

- 功能描述：AHUB 模块唤醒（开启时钟）
- 参数说明：
 - component: platform 层组件
- 返回值：0-成功，其它-失败

3.4.4.5 sunxi_ahub_probe

- 函数原型：


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

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

3.4.4.6 sunxi_ahub_dai_suspend [linux-4.9]

- 函数原型：

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

- 功能描述：AHUB 模块休眠（保存寄存器）
- 参数说明：
 - dai: cpu dai 信息
- 返回值：0-成功，其它-失败

3.4.4.7 sunxi_ahub_suspend [linux-5.4~linux-5.15]

- 函数原型：

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

- 功能描述：AHUB 模块休眠（保存寄存器）
- 参数说明：
 - component: platform 层组件
- 返回值：0-成功，其它-失败

3.4.4.8 sunxi_ahub_dai_resume [linux-4.9]

- 函数原型：

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

- 功能描述：AHUB 模块唤醒（恢复寄存器）
- 参数说明：
 - dai: cpu dai 信息
- 返回值：0-成功，其它-失败

3.4.4.9 sunxi_ahub_resume [linux-5.4~linux-5.15]

- 函数原型：

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

- 功能描述：AHUB 模块唤醒（恢复寄存器）
- 参数说明：
 - component: platform 层组件
- 返回值：0-成功，其它-失败

3.4.4.10 sunxi_ahub_dai_probe

- 函数原型：

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

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

3.4.4.11 sunxi_ahub_dai_remove

- 函数原型：

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

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

3.4.4.12 sunxi_ahub_dai_set_pll

- 函数原型：

```
int sunxi_ahub_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

3.4.4.13 sunxi_ahub_dai_set_sysclk

- 函数原型：

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

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

3.4.4.14 sunxi_ahub_dai_set_bclk_ratio

- 函数原型：

```
int sunxi_ahub_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

3.4.4.15 sunxi_ahub_dai_set_fmt

- 函数原型：

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

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

3.4.4.16 sunxi_ahub_dai_set_tdm_slot

- 函数原型：

```
int sunxi_ahub_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

3.4.4.17 sunxi_ahub_dai_startup

- 函数原型：

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

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

3.4.4.18 sunxi_ahub_dai_hw_params

- 函数原型：

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

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

3.4.4.19 sunxi_ahub_dai_hw_free

- 函数原型：

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

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

3.4.4.20 sunxi_ahub_dai_prepare

- 函数原型：

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

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

3.4.4.21 sunxi_ahub_dai_trigger

- 函数原型：

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

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

3.4.4.22 sunxi_ahub_dai_shutdown

- 函数原型：

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

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

3.4.5 platform 层接口 -> SPDIF

3.4.5.1 sunxi_spdif_component_probe

- 函数原型：

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

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

3.4.5.2 sunxi_spdif_dai_suspend [linux-4.9]

- 函数原型：

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

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

3.4.5.3 sunxi_spdif_component_suspend [linux-5.4~linux-5.15]

- 函数原型：

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

3.4.5.4 sunxi_spdif_dai_resume [linux-4.9]

- 函数原型：

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

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

3.4.5.5 sunxi_spdif_component_resume [linux-5.4~linux-5.15]

- 函数原型：


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

3.4.5.6 sunxi_spdif_dai_probe

- 函数原型：

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

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

3.4.5.7 sunxi_spdif_dai_remove

- 函数原型：

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

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

3.4.5.8 sunxi_spdif_dai_set_pll

- 函数原型：

```
int sunxi_spdif_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

3.4.5.9 sunxi_spdif_dai_set_clkdiv

- 函数原型：

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

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

3.4.5.10 sunxi_spdif_dai_startup

- 函数原型：

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

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

3.4.5.11 sunxi_spdif_dai_hw_params

- 函数原型：

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

- 功能描述：设置模块硬件参数
- 参数说明：

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

- 返回值：0-成功，其它-失败

3.4.5.12 sunxi_spdif_dai_prepare

- 函数原型：

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

- 功能描述：清除模块 fifo
- 参数说明：

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

- 返回值：0-成功，其它-失败

3.4.5.13 sunxi_spdif_dai_trigger

- 函数原型：

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

- 功能描述：触发模块工作
- 参数说明：

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

- 返回值：0-成功，其它-失败

3.4.5.14 sunxi_spdif_dai_shutdown

- 函数原型：

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

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

3.4.6 platform 层接口 -> DMIC

3.4.6.1 sunxi_dmic_component_probe

- 函数原型：

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

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

3.4.6.2 sunxi_dmic_dai_suspend [linux-4.9]

- 函数原型：

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

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

3.4.6.3 sunxi_dmic_component_suspend [linux-5.4~linux-5.15]

- 函数原型：

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

3.4.6.4 sunxi_dmic_dai_resume [linux-4.9]

- 函数原型：

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

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

3.4.6.5 sunxi_dmic_component_resume [linux-5.4~linux-5.15]

- 函数原型：

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

3.4.6.6 sunxi_dmic_dai_probe

- 函数原型：

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

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

3.4.6.7 sunxi_dmic_dai_set_pll

- 函数原型：

```
int sunxi_dmic_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

3.4.6.8 sunxi_dmic_dai_startup

- 函数原型：

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

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

3.4.6.9 sunxi_dmic_dai_hw_params

- 函数原型：

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

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

3.4.6.10 sunxi_dmic_dai_prepare

- 函数原型：

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

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

3.4.6.11 sunxi_dmic_dai_trigger

- 函数原型：

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

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

3.4.6.12 sunxi_dmic_dai_shutdown

- 函数原型：

```
void sunxi_dmic_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

3.4.7 codec 层接口 -> 公共部分

3.4.7.1 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

3.4.7.2 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

3.4.7.3 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

3.4.7.4 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

3.4.8 codec 层接口 -> AudioCodec

3.4.8.1 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

3.4.8.2 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

3.4.8.3 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

3.4.8.4 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

3.4.8.5 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

3.4.8.6 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

3.4.8.7 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

3.4.8.8 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

3.4.8.9 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

3.4.8.10 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

3.4.8.11 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

3.4.8.12 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

3.4.9 machine 层接口

3.4.9.1 simple_soc_probe

- 函数原型：

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

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

3.4.9.2 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

3.4.9.3 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

3.4.9.4 asoc_simple_parse_widgets

- 函数原型：

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

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

3.4.9.5 asoc_simple_parse_routing

- 函数原型：

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

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

3.4.9.6 asoc_simple_parse_pin_switches

- 函数原型：

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

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

3.4.9.7 asoc_simple_parse_daifmt

- 函数原型：

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

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

3.4.9.8 asoc_simple_parse_daistream

- 函数原型：

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

- 功能描述：解析音频流
- 参数说明：
 - node: machine 节点
 - prefix: 设备树属性名称前缀
 - dai_link: dai 链接信息
- 返回值：0-成功，其它-失败

3.4.9.9 asoc_simple_parse_tdm_slot

- 函数原型：

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

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

3.4.9.10 asoc_simple_parse_tdm_clk

- 函数原型：

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

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

3.4.9.11 asoc_simple_set_dailink_name

- 函数原型：

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

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

3.4.9.12 asoc_simple_dai_init

- 函数原型：

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

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

3.4.9.13 asoc_simple_hw_params

- 函数原型：

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

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

3.5 软件调试接口

3.5.1 snd_sunxi_debug_show_reg

- 函数原型：

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

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

3.5.2 snd_sunxi_debug_store_reg

- 函数原型：

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

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

4 模块使用

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

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

4.1 kernel menuconfig 配置

进入 menuconfig 配置命令：

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

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

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

具体配置选项，根据芯片平台、内核版本、所需音频模块，查看各模块的 **kernel menuconfig 配置说明**。

4.2 加载与卸载方法

具体配置选项，根据芯片平台、内核版本、所需音频模块，查看各模块的 **加载与卸载方法（若编译为 ko）说明**。

4.3 声卡设备查看

可输入以下命令查看系统挂载上的声卡

```
cat /proc/asound/cards
0 [audiocodec      ]: audiocodec - audiocodec
                        audiocodec
1 [sndspdif        ]: sndspdif - sndspdif
                        sndspdif
2 [snddmic         ]: snddmic - snddmic
                        snddmic
3 [snddaudio0      ]: snddaudio0 - snddaudio0
                        snddaudio0
4 [sndhdmix        ]: sndhdmix - sndhdmix
                        sndhdmix
5 [snddaudio2      ]: snddaudio2 - snddaudio2
                        snddaudio2
6 [ahubdam         ]: ahubdam - ahubdam
                        ahubdam
7 [ahubi2s0        ]: ahubi2s0 - ahubi2s0
                        ahubi2s0
8 [ahubhdmix       ]: ahubhdmix - ahubhdmix
                        ahubhdmix
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. **snddaudio0** 和 **ahubi2s0** 均为 **i2s** 接口，**ahub** 前缀代表该声卡具备混音功能。

查看更加详细的声卡信息如下（以 audiocodec 声卡为例）。

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

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

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

查看播放设备硬件参数

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

查看播放设备软件参数

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

查看播放设备状态

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

4.4 声卡控件

具体配置选项，根据芯片平台、内核版本、所需音频模块，查看各模块的 **声卡控件** 说明。

4.5 声卡测试工具使用

4.5.1 tinyalsa 工具

tinyalsa 主要提供五个工具。

1. tinymix: 可以得到音频通路相关的各项配置参数，也可通过传参设置参数；
2. tinypplay: 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. tinyloup: 通过软件的方式，将录制的声音实时播放。

4.5.1.1 tinymix

```
tinymix -D cardx /* 查看声卡x的控件列表 */
tinymix -D cardx y /* 查看声卡x序号为y的控件的可选项 */
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
```

4.5.1.2 tinypplay

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

- 用声卡 0 播放 test.wav

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

4.5.1.3 tinycap

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

- 用声卡 3 录音并将数据保存到 test.wav

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

4.5.1.4 tinypcminfo

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

- 查看声卡 2 设备 0 的 pcm 通道信息

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

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

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

4.5.1.5 tinyloop

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

- 用声卡 0 设备 0 和声卡 3 设备 0 进行回路测试

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



说明

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

tinymix -h

tinyplay

tinycap

4.5.2 alsa-utils 工具

alsa-utils 主要提供三个工具：

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

4.5.2.1 aplay

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

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

- 查看可以用于播放的声卡

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

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


```

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

```

4.5.2.2 arecord

```

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

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

```

- 查看可以用于录音的声卡

```

/# arecord -l
**** List of CAPTURE Hardware Devices ****
card 0: audiocodec [audiocodec], device 0: soc@03000000:codec_plat-sunxi-snd-codec sunxi-
snd-codec-0 []
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 1: snddmic [snddmic], device 0: 2031000.dmic_plat-snd-soc-dummy-dai snd-soc-dummy-dai
-0 []
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 2: snddaudio0 [snddaudio0], device 0: 2032000.dauidio0_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...

```

4.5.2.3 amixer

```
# 输入 amixer 或 amixer -h 可打印出使用方法
/# amixer -h
Usage: amixer <options> [command]

Available options:
-h,--help          this help
-c,--card N        select the card
-D,--device N      select the device, default 'default'
-d,--debug         debug mode
-n,--nocheck       do not perform range checking
-v,--version       print version of this program
-q,--quiet         be quiet
-i,--inactive      show also inactive controls
-a,--abstract L    select abstraction level (none or basic)
-s,--stdin         Read and execute commands from stdin sequentially
-R,--raw-volume    Use the raw value (default)
-M,--mapped-volume Use the mapped volume

Available commands:
scontrols          show all mixer simple controls
scontents          show contents of all mixer simple controls (default command)
sset sID P         set contents for one mixer simple control
sget sID           get contents for one mixer simple control
controls          show all controls for given card
contents          show contents of all controls for given card
cset cID P         set control contents for one control
cget cID          get control contents for one control
```

- 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
```

4.6 I2S 外挂 codec

4.6.1 硬件连接

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

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

4.6.2 获取外部 codec I2S 协议格式

确认外部 codec I2S 协议格式如下。

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

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

5 FAQ

5.1 调试方法

5.1.1 调试工具

常用调试工具有 tinyalsa 和 alsa-utils 工具，具体使用和调试方法查看各模块的 **常见使用方法说明**和 **声卡测试工具使用说明**。

5.1.2 调试节点

开启调试配置

```
Allwinner BSP --->
  Device Drivers --->
    SOUND Drivers ---->
      Platform drivers --->
        <M> Allwinner Function Components
        <M> Components Debug
```

将“Components Debug”选为 Y 或 M，使能调试节点加载。

寄存器 dump 搜索

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

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

查看寄存器值

```
cat audio_debug

# 根据节点提示说明查看相应寄存器值，如下。
echo 0 > audio_reg
SUNXI_DAC_DPC [0x000]: 0x 0 :0x0
SUNXI_DAC_FIFO_CTL [0x010]: 0x 4000 :0x0
```

SUNXI_DAC_FIFO_STA	[0x014]: 0x	808008	:0x0
SUNXI_DAC_CNT	[0x024]: 0x	0	:0x0
SUNXI_DAC_DG_REG	[0x028]: 0x	0	:0x0
AC_DAC_REG	[0x310]: 0x	15141f	:0x0
AC_MIXER_REG	[0x314]: 0x	133	:0x0
AC_RAMP_REG	[0x31c]: 0x	0	:0x0

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

5.2 常见问题

5.2.1 录音或播放变速

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

【分析步骤二】：以上无法定位，请联系 FAE 协助分析定位。

5.2.2 AudioCodec 输入输出无声音

【分析步骤一】：确认通路设置。

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

【分析步骤二】：对于喇叭，确认功放芯片使能设置。

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

【分析步骤三】：以上无法定位，请联系 FAE 协助分析定位。

5.2.3 DMIC 录音异常（静音/通道移位）

【分析步骤一】：确认 GPIO 是否正常。

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

【分析步骤二】：确认 clk 的频率。

以上正常情况下，示波器查看 DMIC clk 的频率是否满足如下关系。

$$\text{clk} = \text{sample} * \text{over_sample_rate}$$

【分析步骤三】：排查硬件连接和 DMIC 物料问题。

【分析步骤四】：以上无法定位，请联系 FAE 协助分析定位。






著作权声明

版权所有 © 2022 珠海全志科技股份有限公司。保留一切权利。

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

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

商标声明

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

免责声明

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

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

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