Linux Audio
开发指南

版本号: 2.0
发布日期: 2020.11.11

版本历史

版本号	日期	制/修订人	内容描述
1.0	2020.06.19	AWA1636	建立初版
2.0	2020.11.11	AWA1636	新增 Linux-5.4 配置

目　录

插　图

1 前言

1.1 文档简介

本文档是让开发者了解 Sunxi 平台音频系统框架，能够在 Sunxi 平台上开发新的音频方案。

1.2 目标读者

音频系统开发人员。

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
T509	Linux-4.9	sound/soc/sunxi/*
MR813	Linux-4.9	sound/soc/sunxi/*
R818	Linux-4.9	sound/soc/sunxi/*
A133	Linux-4.9	sound/soc/sunxi/*
A133	Linux-5.4	sound/soc/sunxi/*
H616	Linux-4.9	sound/soc/sunxi/*

2 模块介绍

2.1 模块功能介绍

在 Sunxi 中，从 Linux 软件上通常存在 4 类音频设备。分别为 audiocodec，daudio，dmic，spdif。
每一类音频设备都适配 asoc 架构。

2.1.1 AudioCodec 模块功能

2.1.2 AudioCodec 模块功能

Audio Codec 驱动所具有的功能：

- 支持多种采样率格式 （8KHz，11.025KHz，12KHz，16KHz，22.0KHz，24KHz，32KHz，44.1KHz ，48KHz，96KHz，192KHz)，其中录音最大支持 48KHz；
- 支持同时 playback 和 record(全双工模式)；
- 支持 mixer 接口；
- 支持 dapm 接口；
- 支持 16bit/20bit 数据精度；
- 支持 DAC，采样率为 8KHz~192KHz，支持差分输出；
- 支持 ADC，采样率为 8KHz~48KHz，支持差分输入；

2.1.3 Daudio 模块功能

驱动所具有的功能：

- 支持多种采样率格式 (8KHz，11.025KHz，16KHz，22.05KHz，24KHz，32KHz，44.1KHz，48KHz，88.2KHz，96KHz，176.4KHz，192KHz);
- 支持 mono 和 stereo 模式，支持 1-8 通道；
- 支持同时 playback 和 record(全双工模式)；
- 支持 i2s、pcm 协议格式配置；
- 支持 16bit/24bit/32bit 数据精度；

2.1.4 DMIC 模块功能

驱动所具有的功能：

- 支持多种采样率格式 (8KHz，11.025KHz，16KHz，22.05KHz，24KHz，32KHz，44.1KHz，48KHz);
- 最多支持 8 通道;
- 只支持 record;
- 支持 64 OSR 以及 128 OSR;
- 支持 16bit/24bit 数据精度

2.1.5 S/PDIF 模块功能

驱动所具有的功能：

- 支持多种采样率格式 (44.1KHz，48KHz，96KHz，192KHz);
- 支持单声道和立体声输出;
- 支持 16bit/20bit/24bit 数据精度

2.2 相关术语介绍

2.2.1 硬件术语

表 2-1：硬件术语

相关术语	解释说明
audiocodec	芯片内置音频接口
DMIC	外置数字 MIC 接口
SPDIF	外置音响音频设备接口，一般使用同轴电缆或光纤接口
I2S	外置音频通道接口
Daudio	数字音频接口，可配置成 i2s/pcm 格式标准音频接口
Ahub	音频集线器，AW 独有的硬件模块，内部集成连接了 4 组 I2S 接口、3 组 APB 及 2 组 DAM，可

2.2.2 软件术语

表 2-2: 软件术语

相关术语	解释说明
Sunxi	全志科技使用的 Linux 开发平台
ASOC	ALSA System on Chip
ALSA	Advanced Linux Sound Architecture
DMA	直接内存存取，指数据不经 cpu 直接在设备和内存，内存和内存，设备和设备之间传输
样本长度 (sample)	样本是记录音频数据最基本的单位，常见的有 16 位
通道数 (channel)	该参数为 1 表示单声道，2 则是立体声
帧 (frame)	帧记录了一个声音单元，其长度为样本长度与通道数的乘积
采样率 (rate)	每秒钟采样次数，该次数是针对帧而言
周期 (period)	音频设备一次处理所需要的帧数，对于音频设备的数据访问以及音频数据的存储，都是以此为单位
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
DAPM	动态音频电源管理
hp	headphone 缩写，耳机/耳麦
交错模式 (interleave)	是一种音频数据的记录模式，在交错模式下，数据以连续帧的形式存放，即首先记录完帧 1 的左声道样本和右声道样本（假设为立体声格式），再开始帧 2 的记录，而在非交错模式下，首先记录的是一个周期内所有帧的左声道样本，再记录右声道样本，数据是以连续通道的方式存储。多数情况下，只需要使用交错模式。

2.3 模块配置介绍

2.3.1 Device Tree 配置说明

设备树中存在的是该类芯片所有平台的模块配置，设备树文件的路径为：kernel/内核版本/arch/arm64（32 位平台为 arm）/boot/dts/sunxi/CHIP.dtsi(CHIP 为研发代号，如 sun50iw10p1 等)，设备树配置如下所示：

- Codec 配置 Linux-4.9 内核版本配置如下：

```
        codec:codec@0x05096000 {
            compatible = "allwinner,sunxi-internal-codec";
            reg = <0x0 0x05096000 0x0 0x32c>;
            clocks = <&clk_pll_audiox4>,<&clk_codec_dac_1x>,<&clk_codec_adc_1x>,<&
    clk_pll_com>,<&clk_pll_comdiv5>;
            device_type = "codec";
            status = "disabled";
        };

        cpudai:cpudai-controller@0x050906000 {
```

```
        compatible = "allwinner,sunxi-internal-cpudai";
        reg = <0x0 0x05096000 0x0 0x32c>;
        playback_cma = <128>;
        capture_cma = <256>;
        device_type = "cpudai";
        status = "disabled";
    };

    sndcodec:sound@0 {
        compatible = "allwinner,sunxi-codec-machine";
        interrupts = <GIC_SPI 25 IRQ_TYPE_LEVEL_HIGH>;
        sunxi,cpudai-controller = <&cpudai>;
        sunxi,audio-codec = <&codec>;
        hp_detect_case = <0x00>;
        device_type = "sndcodec";
        status = "disabled";
    };
```

Linux-5.4 内核版本配置和 Linux-4.9 内核版本配置有稍许差异，如下：

```
/* codec addr: 0x05096000, the others is invalid to avoid build warining */
codec:codec@5096000 {
    #sound-dai-cells = <0>;
    compatible = "allwinner,sunxi-internal-codec";
    reg = <0x0 0x05096000 0x0 0x32c>;
    clocks = <&ccu CLK_PLL_AUDIO>,  /* 98.304M / 90.3168M */
        <&ccu CLK_AUDIO_DAC>,
        <&ccu CLK_AUDIO_ADC>,
        <&ccu CLK_PLL_COM>,
        <&ccu CLK_PLL_COM_AUDIO>,
        <&ccu CLK_BUS_AUDIO_CODEC>;
    clock-names = "pll_audio", "codec_dac", "codec_adc",
        "pll_com", "pll_com_audio", "codec_bus";
    resets = <&ccu RST_BUS_AUDIO_CODEC>;
    playback_cma    = <128>;
    capture_cma = <256>;
    device_type = "codec";
};

dummy_cpudai:dummy_cpudai@509632c {
    compatible = "allwinner,sunxi-dummy-cpudai";
    reg = <0x0 0x0509632c 0x0 0x4>;
    tx_fifo_size    = <128>;
    rx_fifo_size    = <256>;
    dac_txdata  = <0x05096020>;
    adc_txdata  = <0x05096040>;
    playback_cma    = <128>;
    capture_cma = <256>;
    device_type = "cpudai";
    dmas = <&dma 7>, <&dma 7>;
    dma-names = "tx", "rx";
};

sndcodec:sound@5096330 {
    compatible = "allwinner,sunxi-codec-machine";
    reg = <0x0 0x05096330 0x0 0x4>;
    interrupts = <GIC_SPI 25 IRQ_TYPE_LEVEL_HIGH>;
    hp_detect_case = <0x00>;
```

```
        sunxi,audio-codec = <&codec>;
        sunxi,cpudai-controller = <&dummy_cpudai>;
        device_type = "sndcodec";
    };
```

其中，codec 节点作为 ASOC 中的 Codec 部分，cpudai 节点作为 Platform 部分，sndcodec 作为 Machine 部分。

- DMIC 配置

Linux-4.9 内核版本配置如下：

```
        dmic:dmic-controller@0x05095000{
            compatible = "allwinner,sunxi-dmic";
            reg = <0x0 0x05095000 0x0 0x50>;
            clocks = <&clk_pll_audio>,<&clk_pll_audiox4>,<&clk_dmic>;
            pinctrl-names = "default","sleep";
            pinctrl-0 = <&dmic_pins_a>;
            pinctrl-1 = <&dmic_pins_b>;
            device_type = "dmic";
            status = "disabled";
        };

        snddmic:sound@2{
            compatible = "allwinner,sunxi-dmic-machine";
            sunxi,dmic-controller = <&dmic>;
            device_type = "snddmic";
            status = "disabled";
        };
```

Linux-5.4 内核版本配置和 Linux-4.9 内核版本配置有稍许差异，如下：

```
        /* dmic addr: 0x05095000, the others is invalid to avoid build warining */
        dmic:dmic@5095000{

            #sound-dai-cells = <0>;

            #sound-dai-cells = <0>;
            compatible = "allwinner,sunxi-dmic";
            reg = <0x0 0x05095000 0x0 0x50>;
            clocks = <&ccu CLK_PLL_AUDIO>, /* 98.304M / 90.3168M */
                 <&ccu CLK_DMIC>,
                 <&ccu CLK_BUS_DMIC>;
            clock-names = "pll_audio", "dmic", "dmic_bus";
            resets = <&ccu RST_BUS_DMIC>;
            pinctrl-names = "default","sleep";
            pinctrl-0 = <&dmic_pins_a>;
            pinctrl-1 = <&dmic_pins_b>;
            clk_parent  = <0x1>;
            capture_cma = <256>;
            data_vol    = <0xB0>;
            dmic_rxsync_en  = <0x0>;
            rx_chmap    = <0x76543210>;
            device_type = "dmic";
```

```
        dmas = <&dma 8>;
        dma-names = "rx";
    };

    dmic_codec:sound@5095050{

        #sound-dai-cells = <0>;

        #sound-dai-cells = <0>;
        compatible = "dmic-codec";
        reg = <0x0 0x05095050 0x0 0x4>;
        num-channels = <6>;
    };

    sounddmic:sounddmic@5095060 {
        reg = <0x0 0x05095060 0x0 0x4>;
        compatible = "sunxi,simple-audio-card";
        simple-audio-card,name = "snddmic";
        /* simple-audio-card,format = "i2s"; */
        simple-audio-card,cpu {
        sound-dai = <&dmic>;
    };
```

其中，dmic 节点作为 ASOC 中的 Codec 部分，snddmic 作为 Machine 部分。

- S/PDIF 配置

Linux-4.9 内核版本配置如下：

```
    spdif:spdif-controller@0x05094000{
        compatible = "allwinner,sunxi-spdif";
        reg = <0x0 0x05094000 0x0 0x40>;
        clocks = <&clk_pll_audio>,<&clk_pll_audiox4>,<&clk_spdif>;
        pinctrl-names = "default","sleep";
        pinctrl-0 = <&spdif_pins_a>;
        pinctrl-1 = <&spdif_pins_b>;
        device_type = "spdif";
        status = "disabled";
    };

    sndspdif:sound@1{
        compatible = "allwinner,sunxi-spdif-machine";
        sunxi,spdif-controller = <&spdif>;
        device_type = "sndspdif";
        status = "disabled";
    };
```

Linux-5.4 内核版本配置和 Linux-4.9 内核版本配置有稍许差异，如下：

```
    /* spdif addr: 0x05094000, the others is invalid to avoid build warining */
    spdif:spdif@5094000{

        #sound-dai-cells = <0>;

        #sound-dai-cells = <0>;
```

```
        compatible = "allwinner,sunxi-spdif";
        reg = <0x0 0x05094000 0x0 0x40>;
        clocks = <&ccu CLK_PLL_AUDIO>,  /* 98.304M / 90.3168M */
             <&ccu CLK_SPDIF>,
             <&ccu CLK_BUS_SPDIF>;
        clock-names = "pll_audio", "spdif","spdif_bus";
        resets = <&ccu RST_BUS_SPDIF>;
        pinctrl-names = "default","sleep";
        pinctrl-0 = <&spdif_pins_a>;
        pinctrl-1 = <&spdif_pins_b>;
        clk_parent  = <0x1>;
        playback_cma    = <128>;
        capture_cma = <128>;
        device_type = "spdif";
        dmas = <&dma 2>, <&dma 2>;
        dma-names = "tx", "rx";
    };

    soundspdif:soundspdif@5094040 {
        reg = <0x0 0x05094040 0x0 0x4>;
        compatible = "sunxi,simple-audio-card";
        simple-audio-card,name = "sndspdif";
        /* simple-audio-card,format = "i2s"; */
        simple-audio-card,cpu {
            sound-dai = <&spdif>;
        };

        simple-audio-card,codec {
            /*snd-soc-dummy*/
        };
    };
```

其中，spdif 节点作为 ASOC 中的 Codec 部分，sndspdif 作为 Machine 部分。

- Daudio 配置

Linux-4.9 内核版本配置如下：

```
    daudio0:daudio@0x05090000 {
        compatible = "allwinner,sunxi-daudio";
        reg = <0x0 0x05090000 0x0 0x7c>;
        clocks = <&clk_pll_audio>,<&clk_pll_audiox4>,<&clk_i2s0>;
        pinctrl-names = "default", "sleep";
        pinctrl-0 = <&daudio0_pins_a>;
        pinctrl-1 = <&daudio0_pins_b>;
        device_type = "daudio0";
        tdm_num = <0x00>;
        status = "disabled";
    };

    snddaudio0:sound@3{
        compatible = "allwinner,sunxi-daudio0-machine";
        sunxi,daudio-controller = <&daudio0>;
        device_type = "snddaudio0";
        status = "disabled";
    };
```

Linux-5.4 内核版本配置和 Linux-4.9 内核版本配置有稍许差异，如下：

```
        /* daudio0 addr: 0x05090000, the others is invalid to avoid build warining */
        daudio0:daudio@5090000 {
            #sound-dai-cells = <0>;
            compatible = "allwinner,sunxi-daudio";
            reg = <0x0 0x05090000 0x0 0x7c>;
            clocks = <&ccu CLK_PLL_AUDIO>, /* 98.304M / 90.3168M */
                 <&ccu CLK_I2S0>,
                 <&ccu CLK_BUS_I2S0>;
            clock-names = "pll_audio", "i2s0", "i2s0_bus";
            resets = <&ccu RST_BUS_I2S0>;
            pinctrl-names = "default", "sleep";
            pinctrl-0 = <&daudio0_pins_a>;
            pinctrl-1 = <&daudio0_pins_b>;
            pinctrl_used        = <0x01>;
            sign_extend     = <0x00>;
            tx_data_mode        = <0x00>;
            rx_data_mode        = <0x00>;
            msb_lsb_first       = <0x00>;
            daudio_rxsync_en    = <0x00>;
            pcm_lrck_period     = <0x80>;
            slot_width_select   = <0x20>;
            frametype       = <0x00>;
            tdm_config      = <0x01>;
            tdm_num         = <0x00>;
            mclk_div        = <0x00>;
            clk_parent      = <0x01>;
            capture_cma     = <128>;
            playback_cma        = <128>;
            tx_num          = <4>;
            tx_chmap1       = <0x76543210>;
            tx_chmap0       = <0xFEDCBA98>;
            rx_num          = <4>;
            rx_chmap3       = <0x03020100>;
            rx_chmap2       = <0x07060504>;
            rx_chmap1       = <0x0B0A0908>;
            rx_chmap0       = <0x0F0E0D0C>;
            device_type = "daudio0";
            dmas = <&dma 3>, <&dma 3>;
            dma-names = "tx", "rx";
        };

        sounddaudio0: sounddaudio0@509007c {
            reg = <0x0 0x0509007c 0x0 0x4>;
            compatible = "sunxi,simple-audio-card";
            simple-audio-card,name = "snddaudio0";
            simple-audio-card,format = "i2s";
            simple-audio-card,cpu {
                sound-dai = <&daudio0>;
            };
            daudio0_master: simple-audio-card,codec {
                 /* sound-dai = <&ac108>; */
            };
        };
```

其中，daudio0 节点作为 ASOC 中的 Codec 部分，snddaudio0 作为 Machine 部分。

- Ahub 配置

Linux-4.9 内核版本配置如下：

```
        ahub_cpudai0:cpudai0-controller@0x05097000 {
            compatible = "allwinner,sunxi-ahub-cpudai";
            reg = <0x0 0x05097000 0x0 0xADF>;
            id = <0x0>;
            status = "okay";
        };

        ahub_cpudai1:cpudai1-controller@0x05097000 {
            compatible = "allwinner,sunxi-ahub-cpudai";
            reg = <0x0 0x05097000 0x0 0xADF>;
            id = <0x1>;
            status = "okay";
        };

        ahub_cpudai2:cpudai2-controller@0x05097000 {
            compatible = "allwinner,sunxi-ahub-cpudai";
            reg = <0x0 0x05097000 0x0 0xADF>;
            id = <0x2>;
            status = "okay";
        };

        ahub_cpudai3:cpudai3-controller@0x05097000 {
            compatible = "allwinner,sunxi-ahub-cpudai";
            reg = <0x0 0x05097000 0x0 0xADF>;
            id = <0x3>;
            status = "okay";
        };

        ahub_codec:ahub_codec@0x05097000{
            compatible = "allwinner,sunxi-ahub";
            reg = <0x0 0x05097000 0x0 0xADF>;
            clocks = <&clk_pll_audio>,<&clk_pll_audiox4>,<&clk_ahub>;
            status = "okay";
        };

        ahub_daudio0:ahub_daudio0@0x05097000{
            compatible = "allwinner,sunxi-ahub-daudio";
            reg = <0x0 0x05097000 0x0 0xADF>;
            clocks = <&clk_pll_audio>,<&clk_pll_audiox4>,<&clk_ahub>;
            pinctrl-names = "default","sleep";
            pinctrl-0 = <&ahub_daudio0_pins_a>;
            pinctrl-1 = <&ahub_daudio0_pins_b>;
            tdm_num = <0x00>;
            device_type = "ahub_daudio0";
            status = "disabled";
        };

        ahub_daudio1:ahub_daudio1@0x05097000{
            compatible = "allwinner,sunxi-ahub-daudio";
            reg = <0x0 0x05097000 0x0 0xADF>;
            clocks = <&clk_pll_audio>,<&clk_pll_audiox4>,<&clk_ahub>;
            tdm_num = <0x01>;
            device_type = "ahub_daudio1";
            status = "okay";
```

```
};

ahub_daudio2:ahub_daudio2@0x05097000{
    compatible = "allwinner,sunxi-ahub-daudio";
    reg = <0x0 0x05097000 0x0 0xADF>;
    clocks = <&clk_pll_audio>,<&clk_pll_audiox4>,<&clk_ahub>;
    pinctrl-names = "default","sleep";
    pinctrl-0 = <&ahub_daudio2_pins_a>;
    pinctrl-1 = <&ahub_daudio2_pins_b>;
    tdm_num = <0x02>;
    device_type = "ahub_daudio2";
    status = "disabled";
};

ahub_daudio3:ahub_daudio3@0x05097000{
    compatible = "allwinner,sunxi-ahub-daudio";
    reg = <0x0 0x05097000 0x0 0xADF>;
    clocks = <&clk_pll_audio>,<&clk_pll_audiox4>,<&clk_ahub>;
    pinctrl-names = "default","sleep";
    pinctrl-0 = <&ahub_daudio3_pins_a>;
    pinctrl-1 = <&ahub_daudio3_pins_b>;
    tdm_num = <0x03>;
    device_type = "ahub_daudio3";
    status = "disabled";
};

snddaudio0:sound@0{
    compatible = "allwinner,sunxi-daudio0-machine";
    sunxi,cpudai-controller = <&ahub_daudio0>;
    device_type = "snddaudio0";
    status = "disabled";
};

sndhdmi:sound@1{
    compatible = "allwinner,sunxi-hdmi-machine";
    sunxi,cpudai-controller = <&ahub_daudio1>;
    device_type = "sndhdmi";
    status = "okay";
};

snddaudio2:sound@2{
    compatible = "allwinner,sunxi-daudio2-machine";
    sunxi,cpudai-controller = <&ahub_daudio2>;
    device_type = "snddaudio2";
    status = "disabled";
};

snddaudio3:sound@3{
    compatible = "allwinner,sunxi-daudio3-machine";
    sunxi,cpudai-controller = <&ahub_daudio3>;
    device_type = "snddaudio3";
    status = "disabled";
};

sndahub:sound@7{
    compatible = "allwinner,sunxi-ahub-machine";
    sunxi,cpudai-controller0 = <&ahub_cpudai0>;
    sunxi,cpudai-controller1 = <&ahub_cpudai1>;
    sunxi,cpudai-controller2 = <&ahub_cpudai2>;
    sunxi,audio-codec = <&ahub_codec>;
```

```
        device_type = "sndahub";
        status = "okay";
    };
```

2.3.2 board.dts 板级配置

board.dts 用于保存每一个板级平台设备差异化的信息的补充（如 demo 板，demo2.0 板，ver1 板等等），里面的配置信息会覆盖上面的 device tree 默认配置信息。

board.dts 的路径为/device/config/chips/{IC}/configs/{BOARD}/board.dts，其中的具体配置如下：

::: note 在 Linux-5.4 内核版本中对 board.dts 语法做了修改，不再支持同名节点覆盖，使用“&”符号引用节点。:::

Linux-4.9 内核版本配置如下：

- Codec 的具体配置

```
    /* Audio Driver Modules */
    codec:codec@0x05096000 {
        /* MIC and headphone gain setting */
        mic1gain    = <0x17>;//mic1增益设置
        mic2gain    = <0x17>;//mic2增益设置
        headphonegain   = <0x00>;//耳机增益设置
        /* adc/dac DRC/HPF func enabled */
        adcdrc_cfg  = <0x00>;//是否配置drc
        adchpf_cfg  = <0x00>;//是否配置hpf
        dacdrc_cfg  = <0x00>;
        dachpf_cfg  = <0x00>;
        /* Volume about */
        digital_vol     = <0x00>;//默认数字音量
        lineout_vol     = <0x1a>;//默认输出音量
        /* Pa enabled about */
        pa_level    = <0x01>;//是否使用pa
        pa_msleep_time  = <0xa0>;//喇叭pa进入稳定状态需要的时间
        gpio-spk = <&pio PH 18 1 1 1 1>;//外部功放使能脚，一般用于了外放喇叭控制
        status = "okay";
    };

    sndcodec:sound@0 {
        hp_detect_case = <0x01>;
        status = "okay";
    };
```

- DMIC 的具体配置

```
        dmic:dmic-controller@0x05095000{
            status = "okay";//开启dmic
        };

        snddmic:sound@2{
            status = "okay";//开启dmic
        };
```

- S/PDIF 的具体配置

```
        spdif:spdif-controller@0x05094000{
            status = "okay";//开启spdif
        };

        sndspdif:sound@1{
            status = "okay";//开启spdif
        };
```

- Daudio 的具体配置

```
        daudio0:daudio@0x05090000 {
            mclk_div    = <0x01>;
            frametype   = <0x00>;
            tdm_config  = <0x01>;
            sign_extend     = <0x00>;
            tx_data_mode    = <0x00>;
            rx_data_mode    = <0x00>;
            msb_lsb_first   = <0x00>;
            pcm_lrck_period = <0x80>;
            audio_format    = <0x01>;
            daudio_master   = <0x04>;
            signal_inversion  = <0x01>;
            slot_width_select = <0x20>;
            status = "okay";
        };

        snddaudio0:sound@3 {
        /*
            sunxi,snddaudio-codec = "ac108.0-003b";
            sunxi,snddaudio-codec-dai = "ac108-pcm0";
        */
            status = "okay";
        };
```

相应配置说明如下：

```
daudio_master:
    1: SND_SOC_DAIFMT_CBM_CFM(codec clk & FRM master)
    2: SND_SOC_DAIFMT_CBS_CFM(codec clk slave & FRM master)
    3: SND_SOC_DAIFMT_CBM_CFS(codec clk master & frame slave)
```

```
    4: SND_SOC_DAIFMT_CBS_CFS(codec clk & FRM slave)
tdm_config:
    0 is pcm; 1 is i2s
audio_format:
    1:SND_SOC_DAIFMT_I2S(standard i2s format)
    2:SND_SOC_DAIFMT_RIGHT_J(right justfied format)
    3:SND_SOC_DAIFMT_LEFT_J(left justfied format)
    4:SND_SOC_DAIFMT_DSP_A(pcm. MSB is available on 2nd BCLK rising edge after LRC rising
    edge)
    5:SND_SOC_DAIFMT_DSP_B(pcm. MSB is available on 1nd BCLK rising edge after LRC rising
    edge)
signal_inversion:
    1:SND_SOC_DAIFMT_NB_NF(normal bit clock + frame)
    2:SND_SOC_DAIFMT_NB_IF(normal BCLK + inv FRM)
    3:SND_SOC_DAIFMT_IB_NF(invert BCLK + nor FRM)
    4:SND_SOC_DAIFMT_IB_IF(invert BCLK + FRM)
word_select_size    :16bits/20bits/24bits/32bits
pcm_lrck_period     :16/32/64/128/256 表示多少个bclk，具体关系见sunxi-daudio.c中关于set_clk函数部
    分
msb_lsb_first       :0: msb first; 1: lsb first
sign_extend         :0: zero pending; 1: sign extend
slot_width_select   :8 bit width / 16 bit width / 32 bit width 必须大于或等于使用的采样精度
frametype       :0: short frame = 1 clock width;  1: long frame = 2 clock width
mclk_div        :0: not output(normal setting this)
    1/2/4/6/8/12/16/24/32/48/64/96/128/176/192:
            setting mclk as input clock to external codec,  freq is pll_audio/mclk_div
tx_data_mode        :0: 16bit linear PCM;  1: reserved;  2: 8bit u-law;  3: 8bit a-law
rx_data_mode        :0: 16bit linear PCM;  1: reserved;
            :2: 8bit u-law;  3: 8bit a-law
playback_cma        :dma memory size(kB) for playback;
capture_cma     :dma memory size(kB) for capture;
status = "okay"打开，"disabled"关闭.
```

- Ahub 的具体配置

```
            /* this set of daudio0 pins is used for t507 board */
            ahub_daudio0_pins_a: ahub_daudio0@0 {
                allwinner,pins = "PA6", "PA7", "PA8", "PA9";
                allwinner,function = "h_i2s0";
                allwinner,muxsel = <3>;
                allwinner,drive = <1>;
                allwinner,pull = <0>;
            };

            ahub_daudio0_pins_b: ahub_daudio0_sleep@0 {
                allwinner,pins = "PA6", "PA7", "PA8", "PA9";
                allwinner,function = "io_disabled";
                allwinner,muxsel = <7>;
                allwinner,drive = <1>;
                allwinner,pull = <0>;
            };

            ahub_daudio2_pins_a: ahub_daudio2@0 {
                allwinner,pins = "PG11", "PG12", "PG13", "PG14";
                allwinner,function = "h_i2s2";
                allwinner,muxsel = <2>;
                allwinner,drive = <1>;
```

```
            allwinner,pull = <0>;
        };

        ahub_daudio2_pins_b: ahub_daudio2_sleep@0 {
            allwinner,pins = "PG11", "PG12", "PG13", "PG14";
            allwinner,function = "io_disabled";
            allwinner,muxsel = <7>;
            allwinner,drive = <1>;
            allwinner,pull = <0>;
        };

        ahub_daudio3_pins_a: ahub_daudio3@0 {
            allwinner,pins = "PH5", "PH6", "PH7", "PH8", "PH9";
            allwinner,function = "h_i2s3";
            allwinner,muxsel = <3>;
            allwinner,drive = <1>;
            allwinner,pull = <0>;
        };

        ahub_daudio3_pins_b: ahub_daudio3_sleep@0 {
            allwinner,pins = "PH5", "PH6", "PH7", "PH8", "PH9";
            allwinner,function = "io_disabled";
            allwinner,muxsel = <7>;
            allwinner,drive = <1>;
            allwinner,pull = <0>;
        };
```

相应配置说明如下：

表 2-3: 模块引脚组定义说明

节点配置	解释说明
pins	模块需要使用到的引脚组定义，一般分别对应模块引脚定义为 “MCLK/BCLK/LRCK/DIN/DOUT”
function	模块引脚组复用名称
muxsel	模块引脚组具体复用
drive	模块引脚驱动力，默认配置为 1 即可
pull	模块引脚下拉选择，默认配置 1 即可

Linux-5.4 内核版本配置如下：

- Codec 的具体配置

```
    /* Audio Driver Modules */
&codec {
    /* MIC and headphone gain setting */
    mic1gain    = <0x1F>;//mic1增益设置
    mic2gain    = <0x1F>;//mic2增益设置
    /* ADC/DAC DRC/HPF func enabled */
    /* 0x1:DAP_HP_EN; 0x2:DAP_SPK_EN; 0x3:DAP_HPSPK_EN */
    adcdrc_cfg  = <0x2>;//是否配置drc
    adchpf_cfg  = <0x1>;//是否配置hpf
```

```
    dacdrc_cfg  = <0x2>;
    dachpf_cfg  = <0x0>;
    /* Volume about */
    digital_vol     = <0x00>;//默认数字音量
    lineout_vol     = <0x1a>;//默认输出音量
    headphonegain   = <0x00>;//耳机增益设置
    /* Pa enabled about */
    pa_level    = <0x01>;//是否使用pa
    pa_msleep_time  = <0x78>;
    gpio-spk    = <&pio PH 6 GPIO_ACTIVE_HIGH>;//外部功放使能脚，一般用于了外放喇叭控制
    /* CMA config about */
    playback_cma    = <128>;//播放CMA大小配置
    capture_cma = <256>;//录音CMA大小配置
    /* regulator about */
    avcc-supply = <&reg_aldo1>;
    cpvin-supply    = <&reg_eldo1>;
    status = "okay";
};

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

- DMIC 的具体配置

```
&dmic {
    capture_cma = <128>;//录音CMA大小配置
    data_vol    = <0xB0>;//录音音量
    rx_chmap    = <0x76543210>;//录音通道映射设置
    status = "okay";//开启dmic
};

&dmic_codec {
    status = "okay";//开启dmic
};
```

- S/PDIF 的具体配置

```
&spdif {
    playback_cma    = <128>;
    capture_cma = <128>;
    status = "okay";//开启spdif
};
```

- Daudio 的具体配置

```
&daudio0 {
    mclk_div    = <0x01>;
    frametype   = <0x00>;
```

```
    tdm_config  = <0x01>;
    sign_extend     = <0x00>;
    tx_data_mode    = <0x00>;
    rx_data_mode    = <0x00>;
    msb_lsb_first   = <0x00>;
    pcm_lrck_period = <0x80>;
    slot_width_select = <0x20>;
    status = "okay";//开启daudio
};

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

其中：

- simple-audio-card,frame-master 代表 codec 做主，soc 做从。
- simple-audio-card,bitclock-master 代表 bitclock 由 codec 发出。
- simple-audio-card,bitclock-inversion 代表 bitclock 极性取反。
- simple-audio-card,frame-inversion 代表 lrclock 极性取反。

daudio 相应配置说明如下：

```
daudio_master:
    1: SND_SOC_DAIFMT_CBM_CFM(codec clk & FRM master)
    2: SND_SOC_DAIFMT_CBS_CFM(codec clk slave & FRM master)
    3: SND_SOC_DAIFMT_CBM_CFS(codec clk master & frame slave)
    4: SND_SOC_DAIFMT_CBS_CFS(codec clk & FRM slave)
tdm_config:
    0 is pcm; 1 is i2s
audio_format:
    1:SND_SOC_DAIFMT_I2S(standard i2s format)
    2:SND_SOC_DAIFMT_RIGHT_J(right justfied format)
    3:SND_SOC_DAIFMT_LEFT_J(left justfied format)
    4:SND_SOC_DAIFMT_DSP_A(pcm. MSB is available on 2nd BCLK rising edge after LRC rising
    edge)
    5:SND_SOC_DAIFMT_DSP_B(pcm. MSB is available on 1nd BCLK rising edge after LRC rising
    edge)
signal_inversion:
    1:SND_SOC_DAIFMT_NB_NF(normal bit clock + frame)
    2:SND_SOC_DAIFMT_NB_IF(normal BCLK + inv FRM)
    3:SND_SOC_DAIFMT_IB_NF(invert BCLK + nor FRM)
    4:SND_SOC_DAIFMT_IB_IF(invert BCLK + FRM)
word_select_size    :16bits/20bits/24bits/32bits
pcm_lrck_period     :16/32/64/128/256 表示多少个bclk，具体关系见sunxi-daudio.c中关于set_clk函数部
    分
msb_lsb_first       :0: msb first; 1: lsb first
sign_extend         :0: zero pending; 1: sign extend
```

```
slot_width_select   :8 bit width / 16 bit width / 32 bit width 必须大于或等于使用的采样精度
frametype       :0: short frame = 1 clock width;  1: long frame = 2 clock width
mclk_div        :0: not output(normal setting this)
    1/2/4/6/8/12/16/24/32/48/64/96/128/176/192:
            setting mclk as input clock to external codec,  freq is pll_audio/mclk_div
tx_data_mode        :0: 16bit linear PCM;  1: reserved;  2: 8bit u-law;  3: 8bit a-law
rx_data_mode        :0: 16bit linear PCM;  1: reserved;
            :2: 8bit u-law;  3: 8bit a-law
playback_cma        :dma memory size(kB) for playback;
capture_cma     :dma memory size(kB) for capture;
status = "okay"打开，"disabled"关闭.
```

2.3.3 kernel menuconfig 配置

在命令行进入内核根目录，执行 make ARCH=arm64 menuconfig (32 位平台执行：make ARCH=arm menuconfig) 进入配置主界面 (Linux-5.4 内核执行：./build.sh menuconfig)，并按以下步骤操作：

1、选择 Device Drivers 选项进入下一级配置，如下图所示：

图 2-1: Device Driver

2、选择 Sound card support 选项，进入下一级配置，如下图所示：

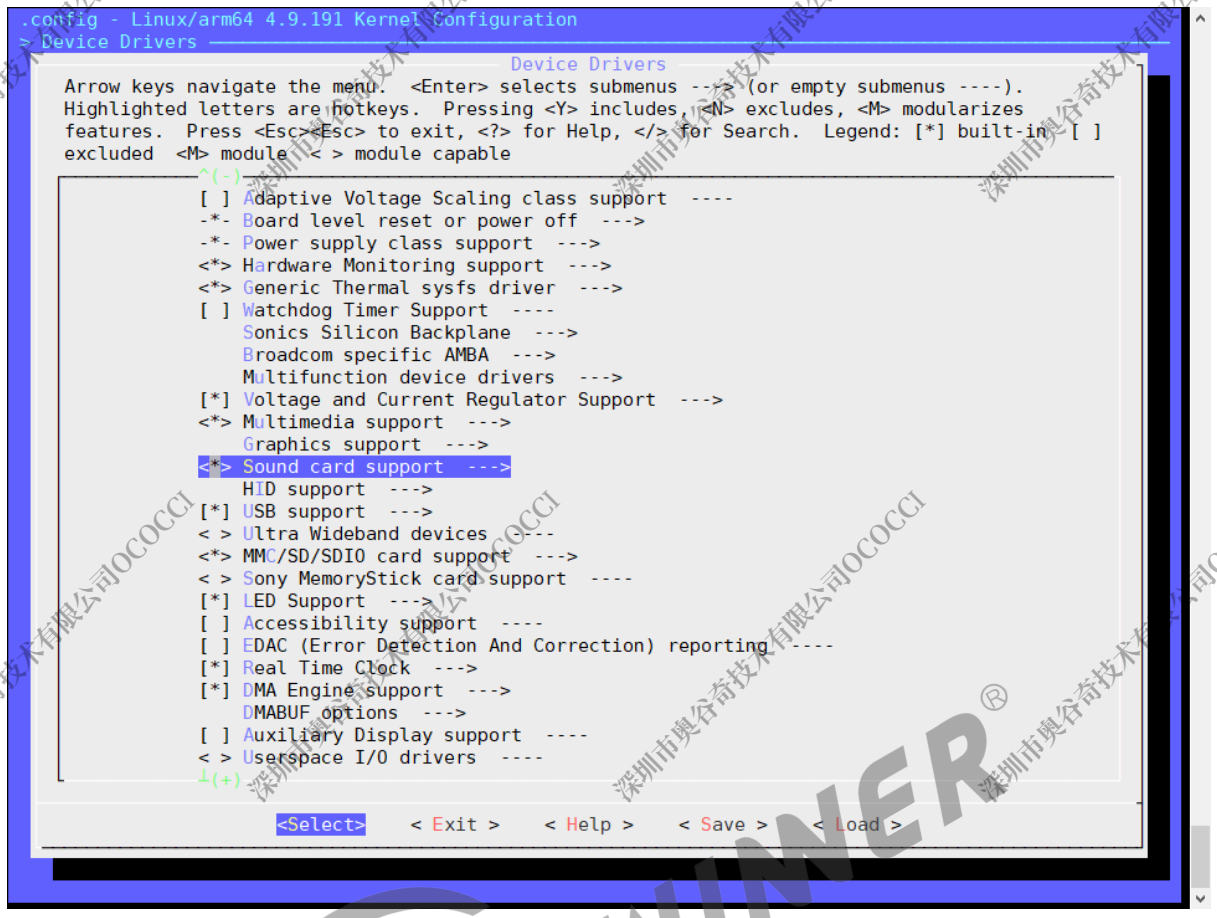

图 2-2: Sound

3、选择 ALSA 框架，即 Advanced Linux Sound Architecture 选项，如下图所示：

图 2-3: Advanced

4、选择 ALSA for SoC audio support 选项，进入下一级配置，如下图所示：

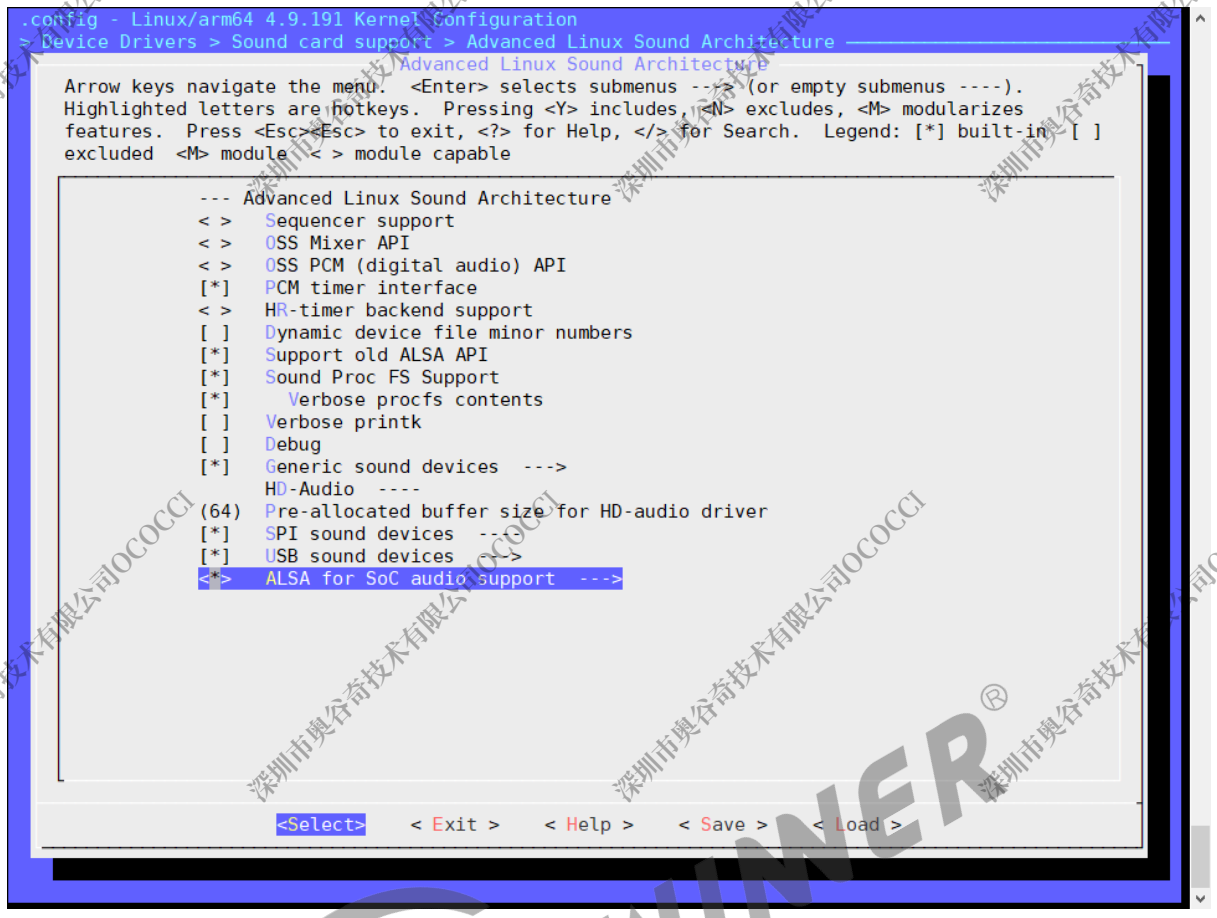

图 2-4: ALSA

5、选择 Allwinner SoC Audio support 选项，如下图所示：

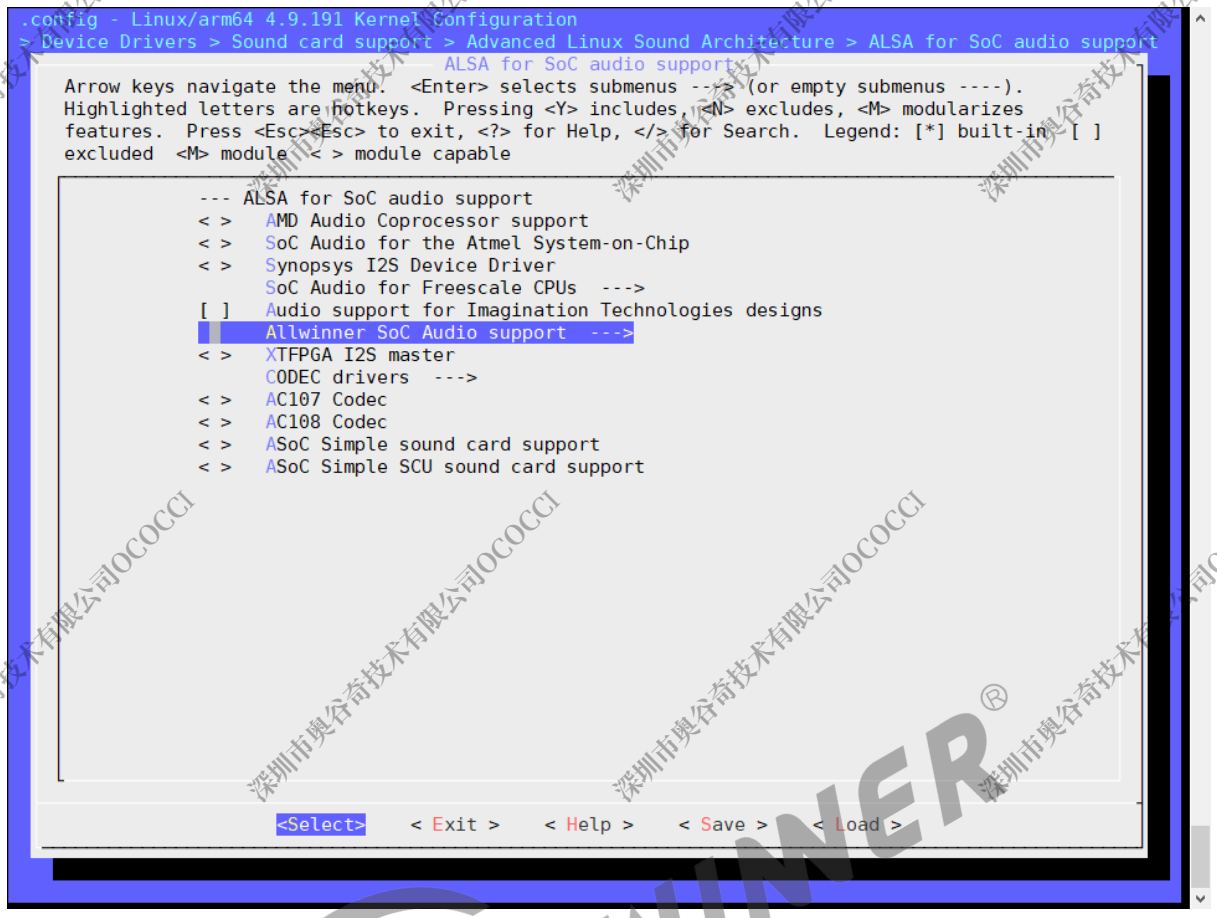

图 2-5: Allwinner

6、选择需要的模块，可选择直接编译进内核，也可编译成模块。如下图所示：

图 2-6: module

7、Linux-5.4 需要先选择 Allwinner Audio Simple Card 选项。如下图所示：

图 2-7: module

2.4 源码模块结构

Audio 驱动的源代码位于内核的 sound/soc/sunxi/目录下：

Linux-4.9 内核版本如下

```
kernel/linux-4.9/sound/soc/
├── sunxi                          // Sunxi平台
│   ├── sun50iw10-codec.c          // Sunxi平台具体芯片codec解码器代码
│   ├── sunxi-cpudai.c                 // Sunxi平台的虚拟cpudai驱动代码
│   ├── sun50iw10-sndcodec.c       // Sunxi平台具体芯片Codec machine部分代码
│   ├── sunxi-dmic.c                 // Sunxi平台DMIC接口代码
│   ├── sunxi-snddmic.c            // Sunxi平台DMIC machine部分代码
│   ├── sunxi-spdif.c              // Sunxi平台S/PDIF接口代码
│   ├── spdif-utils.c              // Sunxi平台的虚拟S/PDIF解码器代码
│   ├── sunxi-sndspdif.c           // Sunxi平台S/PDIF machine部分代码
│   ├── sunxi-daudio.c             // Sunxi平台Daudio接口代码
│   ├── sunxi-snddaudio.c          // Sunxi平台Daudio machine部分代码
│   └── sunxi-pcm.c                // Sunxi平台platform部分dma代码
│   ├── sunxi-ahub.c               // Sunxi平台ahub接口代码
│   ├── sunxi-ahub.h               // Sunxi平台ahub驱动头文件
│   ├── sunxi-sndahub.c            // Sunxi平台ahub Machine部分代码
│   ├── sunxi-snddaudio.h          // Sunxi平台ahub Machine部分头文件
│   ├── sunxi_ahub_daudio.c        // Sunxi平台ahub Machine部分头文件
│   ├── sunxi_ahub_cpudai.c        // Sunxi平台ahub Machine部分头文件
└── codecs                         // 解码器存放路径
    ├── dmic.c                     // DMIC解码器驱动
```

```
    └── ac108.c                 // AC108解码器codec驱动
```

Linux-5.4 内核版本如下

```
kernel/linux-5.4/sound/soc/
├── sunxi                       // Sunxi平台
│   ├── sun50iw10-codec.c       // Sunxi平台具体芯片codec解码器代码
│   ├── sunxi-dummy-cpudai.c     // Sunxi平台的虚拟cpudai驱动代码
│   ├── sun50iw10-sndcodec.c    // Sunxi平台具体芯片Codec machine部分代码
│   ├── sunxi-simple-card.c // Sunxi平台通用Codec machine框架部分代码
│   ├── sunxi-dmic.c              // Sunxi平台DMIC接口代码
│   ├── sunxi-spdif.c           // Sunxi平台S/PDIF接口代码
│   ├── sunxi-daudio.c          // Sunxi平台Daudio接口代码
│   └── sunxi-pcm.c             // Sunxi平台platform部分dma代码
└── codecs                      // 解码器存放路径
    ├── dmic.c                  // DMIC解码器驱动
    └── ac108.c                 // AC108解码器codec驱动
```

2.5 驱动框架介绍

2.5.1 音频驱动硬件框架图

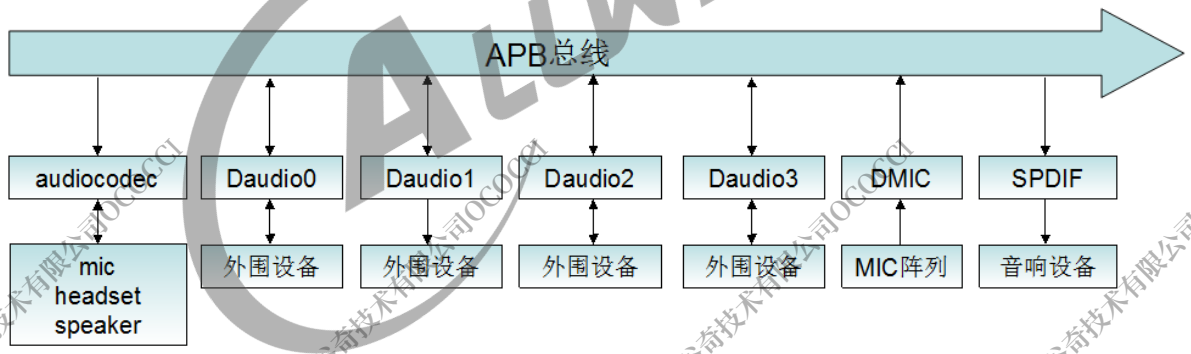

图 2-8: hardware

各个设备对应的设备节点：

表 2-4: 设备节点

硬件接口/设备	设备节点	/sys/class/sound/cardX/id
模拟 codec	/dev/snd/controlC0 /dev/snd/pcmC0D0c /dev/snd/pcmC0D0p /dev/snd/controlC1	audiocodec
spdif 接口	/dev/snd/pcmC1D0c	sndspdif/xxxx

硬件接口/设备	设备节点	/sys/class/sound/cardX/id
	/dev/snd/pcmC1D0p /dev/snd/controlC2	
dmic 接口	/dev/snd/pcmC2D0c 注：DMIC 模块无播放功能 /dev/snd/controlC3	snddmic/xxx
daudio 接口	/dev/snd/pcmC3D0c /dev/snd/pcmC3D0p /dev/snd/controlC4	snddaudio/xxx
ahub 接口	/dev/snd/pcmC4D0c /dev/snd/pcmC4Dnc（第 n 个录音设备） /dev/snd/pcmC4D0p /dev/snd/pcmC4Dnp（第 n 个播放设备）	sndahub/xxx

可以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 [sndahub        ]: sndahub - sndahub
                      sndahub
```

2.5.2 音频驱动软件框架图

音频软件框架使用 ASOC，它是在 ALSA 驱动程序上封装的一层，如下图：

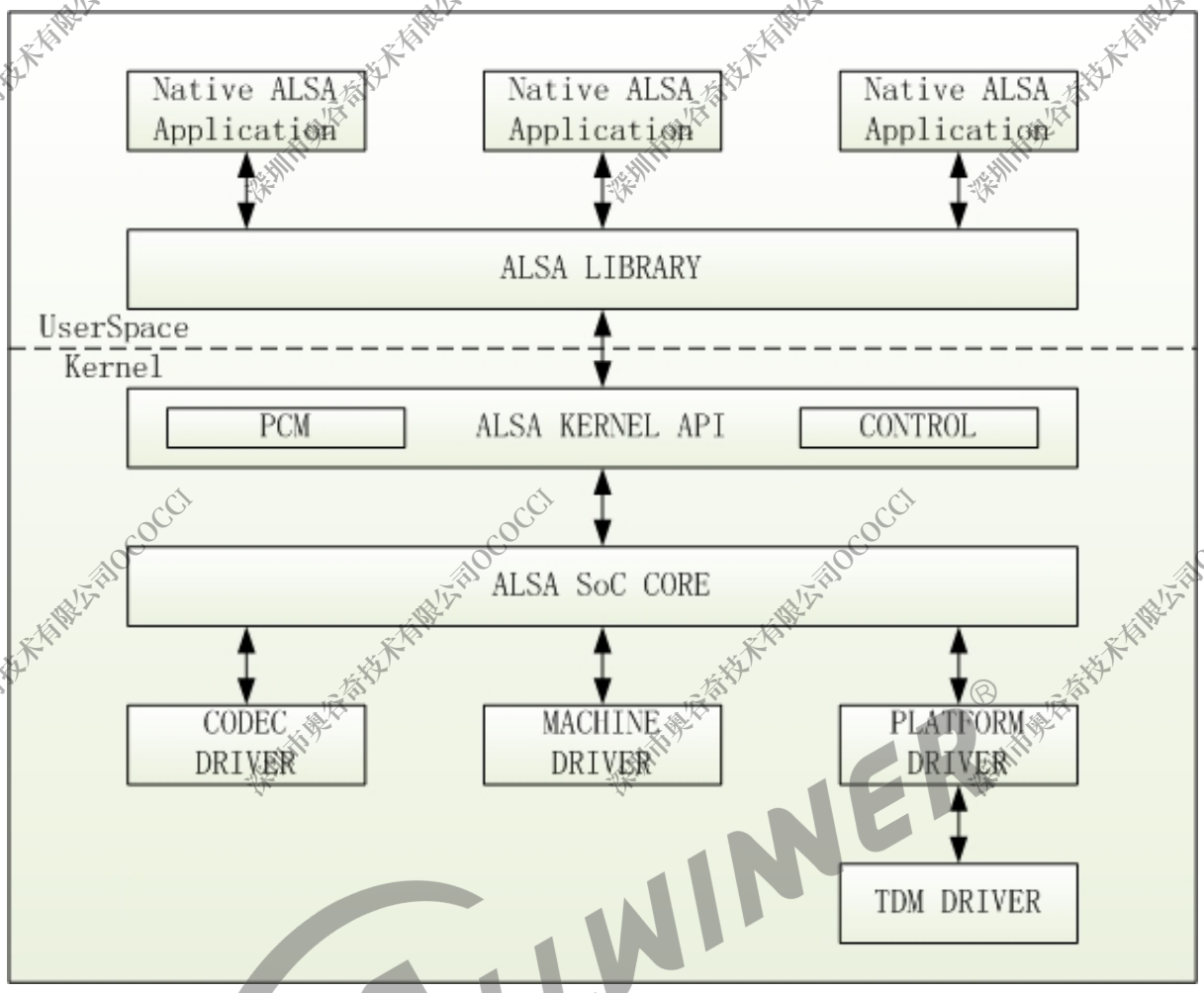

图 2-9: 软件框架图

为了更好地支持嵌入式处理器和移动设备中的音频 codec 的一套软件体系，ASOC 将音频系统分为 3 部分：Machine，Platform 和 Codec。

- Codec 驱动

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

所有的 Codec 驱动都要提供以下特性：

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

必要时，也可以提供以下功能：

5. DAPM 描述信息；
6. DAPM 事件处理程序；
7. DAC 数字静音控制；

- Platform 驱动

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

- Machine 驱动

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

3 模块接口说明

3.1 asoc_dma_platform_register()

- 函数原型：int asoc_dma_platform_register(struct device *dev, unsigned int flags)
- 作用：注册 asoc 里 platform 部分
- 参数：
  - dev：指向所属的设备；
  - flag：可选为：
  - SND_DMAENGINE_PCM_FLAG_COMPAT、SND_DMAENGINE_PCM_FLAG_NO_DT
  - SND_DMAENGINE_PCM_FLAG_HALF_DUPLEX
  - SND_DMAENGINE_PCM_FLAG_CUSTOM_CHANNEL_NAME
- 返回：
  - 0：成功；
  - <0：失败；

3.2 snd_soc_register_component()

- 函数原型：int snd_soc_register_component(struct device *dev, const struct snd_soc_component_driver *cmpnt_drv, struct snd_soc_dai_driver *dai_drv, int num_dai)
- 作用：注册 dai 组件
- 参数：
  - dev：指向所属的设备；
  - cmpnt_drv：组件结构体；
  - dai_drv：dai 的描述；
  - num_dai：dai 的数量
- 返回：
  - 0：成功；
  - <0：失败；

3.3 snd_soc_register_codec

- 函数原型：int snd_soc_register_codec(struct device *dev, const struct snd_soc_codec_driver *codec_drv, struct snd_soc_dai_driver *dai_drv, int num_dai)
- 作用: 注册 codec
- 参数：
  - dev：指向所属的设备；
  - codec_drv：codec 结构体；
  - dai_drv：dai 的描述；
  - num_dai：dai 的数量
- 返回：
  - 0：成功；
  - <0：失败；

3.4 snd_soc_register_card()

- 函数原型：int snd_soc_register_card(struct snd_soc_card *card)
- 作用: 注册一个声卡
- 参数说明：
  - card：描述声卡的结构体
- 返回：
  - 0: 成功；
  - <0: 失败;

3.5 snd_soc_dapm_add_routes()

- 函数原型：int snd_soc_dapm_add_routes(struct snd_soc_dapm_context *dapm, const struct snd_soc_dapm_route *route, int num)
- 作用: 在 DAPM 中添加音频路由表
- 参数说明：
  - dapm：dapm 结构体；
  - route：需要添加的音频路由；
  - num：路由的数量
- 返回：
  - 0: 成功；
  - <0: 失败;

3.6 snd_soc_dapm_new_controls()

- 函数原型：int snd_soc_dapm_new_controls(struct snd_soc_dapm_context *dapm, const struct snd_soc_dapm_widget *widget, int num)
- 作用: 在 DAPM 中添加 control 控制项
- 参数说明：
  - dapm：dapm 结构体；
  - widget：控制的小部件；
  - num：小部件的数量
- 返回：
  - 0: 成功；
  - <0: 失败;

4 FAQ

4.1 调试方法

4.1.1 调试工具

正常情况下 Linux 固件都会配置由 tinyalsa 工具，如果是 Android 固件，可以在 Android 下编译生成。
Android 在 android/external/tinyalsa 目录下使用 mm 编译，会生成 tinycap tinyplay tinymix tinypcminfo tinyhostless 这五个调试工具。
调试工具的用途与用法：

- tinycap
  录音测试工具。用于操作声卡里音频录音设备节点。

```
Usage: tinycap file.wav [-D card] [-d device] [-c channels] [-r rate] [-b bits] [-p
    period_size] [-n n_periods] [-T capture time]
```

例如：

```
tinycap record.wav -D 0 -d 0 -c 2 -r 48000 -b 16
```

这条指令将会使用声卡 0 的第 0 个设备录制一条 48K 双通道 16bit 的音频数据，并命名为 record.wav 保存在当前路径。

- tinyplay
  播放测试工具。用于操作声卡里音频播放设备节点。

```
Usage: tinyplay file.wav [-D card] [-d device] [-p period_size] [-n n_periods]
```

例如：

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

这条指令将会使用声卡 0 的第 0 个设备播放测试音频 test.wav。

- tinymix
  查看音频通路相关的各项配置参数，并通过命令修改参数配置。

```
Usage: tinymix [-D card] [control id] [value to set]
```

例如：

```
tinymix -D 0
```

这条指令可以查看声卡 0 的配置参数。

例如：

```
tinymix -D 0 5 1
```

这条指令可以修改声卡 0 中序号为 5 的参数配置为 1。

4.1.2 调试节点

- 寄存器 dump 搜索

```
/ # find /sys/ -name "audio_reg"
/sys/devices/platform/soc/codec/audio_reg_debug/audio_reg
/ # find /sys/ -name "daudio_reg"
/sys/devices/platform/soc/daudio2/daudio_debug/daudio_reg
/sys/devices/platform/soc/r_daudio0/daudio_debug/daudio_reg
/sys/devices/platform/soc/daudio1/daudio_debug/daudio_reg
```

- codec 输出

```
/ # tinyplay music-44100-2ch.wav -D 0 -p 1024 -n 8&
/ # cat /sys/devices/platform/soc/codec/audio_reg_debug/audio_reg
dump audio reg:
SUNXI_DAC_DPC       [0x000]: 0x80000000    Save:0x0
SUNXI_DAC_VOL_CTL   [0x004]: 0x1a0a0       Save:0x0
SUNXI_DAC_FIFO_CTL  [0x010]: 0x3004010     Save:0x0
SUNXI_DAC_FIFO_STA  [0x014]: 0x3e04        Save:0x0
SUNXI_DAC_TXDATA    [0x020]: 0x0           Save:0x0
SUNXI_DAC_CNT       [0x024]: 0x4f334       Save:0x0
SUNXI_DAC_DG        [0x028]: 0x0           Save:0x0
SUNXI_ADC_FIFO_CTL  [0x030]: 0xe000800     Save:0x0
SUNXI_ADC_VOL_CTL1  [0x034]: 0xa0a0a0a0    Save:0x0
SUNXI_ADC_FIFO_STA  [0x038]: 0x1           Save:0x0
SUNXI_ADC_VOL_CTL2  [0x03c]: 0xa0          Save:0x0
```

```
SUNXI_ADC_RXDATA    [0x040]: 0x0           Save:0x0
SUNXI_ADC_CNT       [0x044]: 0x0           Save:0x0
SUNXI_ADC_DG        [0x04c]: 0x0           Save:0x0
SUNXI_ADC_DIG_CTL   [0x050]: 0x0           Save:0x0
SUNXI_VAR1_SPEEDUP_DOWN_CTL[0x054]: 0x10          Save:0x0
SUNXI_DAC_DAP_CTL   [0x0f0]: 0x0           Save:0x0
SUNXI_ADC_DAP_CTL   [0x0f8]: 0x99000000    Save:0x0
SUNXI_ADC1_REG      [0x300]: 0xcd055       Save:0x0
SUNXI_ADC2_REG      [0x304]: 0xc1055       Save:0x0
SUNXI_ADC3_REG      [0x308]: 0xc1055       Save:0x0
SUNXI_ADC4_REG      [0x30c]: 0xc0055       Save:0x0
SUNXI_DAC_REG       [0x310]: 0x15fd6a      Save:0x0
SUNXI_MICBIAS_REG   [0x318]: 0x30          Save:0x0
SUNXI_RAMP_REG      [0x31c]: 0x1           Save:0x0
SUNXI_BIAS_REG      [0x320]: 0x0           Save:0x0
SUNXI_ADC5_REG      [0x330]: 0xc0055       Save:0x0
```

- codec 输入

```
/ # tinycap rec-mic.wav -D 0 -c 2 -r 16000 -p 1024 -n 8&
/ # cat /sys/devices/platform/soc/codec/audio_reg_debug/audio_reg
dump audio reg:
SUNXI_DAC_DPC       [0x000]: 0x0           Save:0x0
SUNXI_DAC_VOL_CTL   [0x004]: 0x1a0a0       Save:0x0
SUNXI_DAC_FIFO_CTL  [0x010]: 0x3004000     Save:0x0
SUNXI_DAC_FIFO_STA  [0x014]: 0x80800c      Save:0x0
SUNXI_DAC_TXDATA    [0x020]: 0x0           Save:0x0
SUNXI_DAC_CNT       [0x024]: 0x6487c       Save:0x0
SUNXI_DAC_DG        [0x028]: 0x0           Save:0x0
SUNXI_ADC_FIFO_CTL  [0x030]: 0x1f000808    Save:0x0
SUNXI_ADC_VOL_CTL1  [0x034]: 0xa0a0a0a0    Save:0x0
SUNXI_ADC_FIFO_STA  [0x038]: 0x807f01      Save:0x0
SUNXI_ADC_VOL_CTL2  [0x03c]: 0xa0          Save:0x0
SUNXI_ADC_RXDATA    [0x040]: 0x6           Save:0x0
SUNXI_ADC_CNT       [0x044]: 0x46eaa       Save:0x0
SUNXI_ADC_DG        [0x04c]: 0x0           Save:0x0
SUNXI_ADC_DIG_CTL   [0x050]: 0x3           Save:0x0
SUNXI_VAR1_SPEEDUP_DOWN_CTL[0x054]: 0x10          Save:0x0
SUNXI_DAC_DAP_CTL   [0x0f0]: 0x0           Save:0x0
SUNXI_ADC_DAP_CTL   [0x0f8]: 0x99000000    Save:0x0
SUNXI_ADC1_REG      [0x300]: 0xc00cd055    Save:0x0
SUNXI_ADC2_REG      [0x304]: 0xc00c1055    Save:0x0
SUNXI_ADC3_REG      [0x308]: 0xc1055       Save:0x0
SUNXI_ADC4_REG      [0x30c]: 0xc0055       Save:0x0
SUNXI_DAC_REG       [0x310]: 0x15016a      Save:0x0
SUNXI_MICBIAS_REG   [0x318]: 0xb0          Save:0x0
SUNXI_RAMP_REG      [0x31c]: 0x0           Save:0x0
SUNXI_BIAS_REG      [0x320]: 0x0           Save:0x0
SUNXI_ADC5_REG      [0x330]: 0xc0055       Save:0x0
```

4.2 常见问题

4.2.1 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。

【分析步骤二】：对于喇叭，查看设备树 audiocodec 节点中 spk 的 gpio 配置和硬件原理图比对，代码是否适配了对应的 gpio。

【分析步骤三】：以上无法定位，请联系 FAE 协助分析定位。

4.2.2 录音或播放变速

【分析步骤一】：确认录音和播放采样率是否一致。

【分析步骤二】：以上无法定位，请联系 FAE 协助分析定位。

【问题解析】常见问题在于录音和播放不在同一采样点时钟上，备注：spdif 录音不支持单通道。

4.2.3 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 arch/arm(64 位为 arm64)/boot/dts/CHIP-pinctrl.dtsi 部分的 dmic 的 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;

关于过采样率有两个选项，具体意义查看 datasheet.

【分析步骤三】：dump 寄存器

（1）由于布线问题和语音算法需要，针对通道需要移位情况，寄存器查看和修改 chan_map，将对应的 MIC 的数据移到指定的通道。
（2）一一比对下寄存器是否有明显遗漏部分。

【分析步骤四】：排查硬件连接和 dmic 物料问题。

【分析步骤五】：以上无法定位，请联系 FAE 协助分析定位。

【问题解析】常见问题在于 GPIO 和通道修改的问题上。

4.3 常见问题

4.3.1 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。

【分析步骤二】：对于喇叭，查看设备树 audiocodec 节点中 spk 的 gpio 配置和硬件原理图比对，代码是否适配了对应的 gpio。

【分析步骤三】：以上无法定位，请联系 FAE 协助分析定位。

4.3.2 录音或播放变速

【分析步骤一】：确认录音和播放采样率是否一致。

【分析步骤二】：以上无法定位，请联系 FAE 协助分析定位。

【问题解析】常见问题在于录音和播放不在同一采样点时钟上，备注：spdif 录音不支持单通道。

4.3.3 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 arch/arm(64 位为 arm64)/boot/dts/CHIP-pinctrl.dtsi 部分的 dmic 的 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;

关于过采样率有两个选项，具体意义查看 datasheet.

【分析步骤三】：dump 寄存器

（1）由于布线问题和语音算法需要，针对通道需要移位情况，寄存器查看和修改 chan_map，将对应的 MIC 的数据移到指定的通道。
（2）一一比对下寄存器是否有明显遗漏部分。

【分析步骤四】：排查硬件连接和 dmic 物料问题。

【分析步骤五】：以上无法定位，请联系 FAE 协助分析定位。

【问题解析】常见问题在于 GPIO 和通道修改的问题上。

4.3.4 录音或播放变速

【分析步骤一】：确认录音和播放采样率是否一致。

【分析步骤二】：以上无法定位，请联系 FAE 协助分析定位。

【问题解析】常见问题在于录音和播放不在同一采样点时钟上，备注：spdif 录音不支持单通道。

4.3.5 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 arch/arm(64 位为 arm64)/boot/dts/CHIP-pinctrl.dtsi 部分的 dmic 的 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;

关于过采样率有两个选项，具体意义查看 datasheet.

【分析步骤三】：dump 寄存器

（1）由于布线问题和语音算法需要，针对通道需要移位情况，寄存器查看和修改 chan_map，将对应的 MIC 的数据移到指定的通道。
（2）一一比对下寄存器是否有明显遗漏部分。

【分析步骤四】：排查硬件连接和 dmic 物料问题。

【分析步骤五】：以上无法定位，请联系 FAE 协助分析定位。

【问题解析】常见问题在于 GPIO 和通道修改的问题上。

4.3.6 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 arch/arm(64 位为 arm64)/boot/dts/CHIP-pinctrl.dtsi 部分的 dmic 的 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;

关于过采样率有两个选项，具体意义查看 datasheet.

【分析步骤三】：dump 寄存器

（1）由于布线问题和语音算法需要，针对通道需要移位情况，寄存器查看和修改 chan_map，将对应的 MIC 的数据移到指定的通道。

（2）一一比对下寄存器是否有明显遗漏部分。

【分析步骤四】：排查硬件连接和 dmic 物料问题。

【分析步骤五】：以上无法定位，请联系 FAE 协助分析定位。

【问题解析】常见问题在于 GPIO 和通道修改的问题上。

著作权声明

商标声明

免责声明