全志科技
Allwinner Technology

# A64

## 音频驱动使用说明书

# 文档履历

| 版本号 | 日期 | 制/修订人 | 制/修订记录 |
|---|---|---|---|
| V1.0 | 2015-9-28 | 韦金材 | 初始版本 |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

# 目录

A64..... 1
1. 概述.....4
1.1. 编写目的.....4
1.2. 适用范围.....4
1.3. 相关人员.....4
2. 模块介绍.....5
2.1. 模块功能介绍.....5
2.1.1. audiocodec 功能.....5
2.1.2. Daudio 功能.....5
2.1.3. Spdif 功能.....5
2.1.4. hdmiaudio 功能.....5
2.2. 相关术语介绍.....6
2.3. 音频 config 配置.....6
2.3.1. 内核配置.....6
2.3.2. sys_config.fex 配置.....8
2.4. 代码位置.....11
3. 使用描述.....12
3.1. Audiocodec.....12
3.1.1. playback 应用.....14
3.1.2. capture 应用.....15
3.1.3. 播放录音配置 ctl 说明.....15
3.2. Daudio.....17
3.2.1. playback 应用.....17
3.2.2. capture 应用.....18
3.3. Hdmi.....18
3.3.1. playback 应用.....18
3.4. Spdif.....18
3.4.1. playback 应用.....18

# 1. 概述

## 1.1. 编写目的

本文档基于 A64 平台展开分析，用于指导音频系统的开发。

## 1.2. 适用范围

适用于 A64 平台。

## 1.3. 相关人员

音频相关开发人员。

# 2. 模块介绍

## 2.1. 模块功能介绍

### 2.1.1. audiocodec 功能

A64 audiocodec 硬件上的实现是通过 I2S 接口与 cpu 进行数据流传输，cpu 通过 apb 总线对 audiocodec 进行寄存器配置，软件上采用 alsa-asoc 架构实现，具有以下功能。

（1) 支持多种采样率格式(8khz, 11.025 KHz, 12 KHz, 16 KHz, 22.05 KHz, 24 KHz, 32 KHz, 44.1 KHz , 48 KHz, 96KHz, 192KHz);

（2）支持 mono 和 stereo 模式;

（3）支持同时 playback 和 record(全双工模式);

（4）支持 3、4 段耳机插拔检测，hook 键检测，耳机音量加减键定制。

### 2.1.2. Daudio 功能

（1）支持多种采样率格式(8khz, 11.025 khz, 12 khz, 16 khz, 22.05 khz, 24 khz, 32 khz, 44.1 khz , 48 khz, 96khz, 192khz);

（2）支持 mono 和 stereo 模式;

（3）支持同时 playback 和 record(全双工模式);

（4）支持 i2s/pcm

### 2.1.3. Spdif 功能

spdif 驱动所具有的功能：

（1）支持多种采样率格式(22.05khz，24khz，32khz，44.1khz，48khz，88.2khz，96khz，176.4khz，192khz);

（2）支持 mono 和 stereo 模式;

（3）只支持 playback 模式，不支持 record 模式。

（4）支持 rawdata 模式

### 2.1.4. hdmiaudio 功能

hdmiaudio 驱动所具有的功能：

（1）支持多种采样率格式(32khz，44.1khz，48khz，96khz，192khz);

（2）支持 mono 和 stereo 模式;

（3）只支持 playback 模式，不支持 record 模式。

（4）支持 rawdata 模式

## 2.2. 相关术语介绍

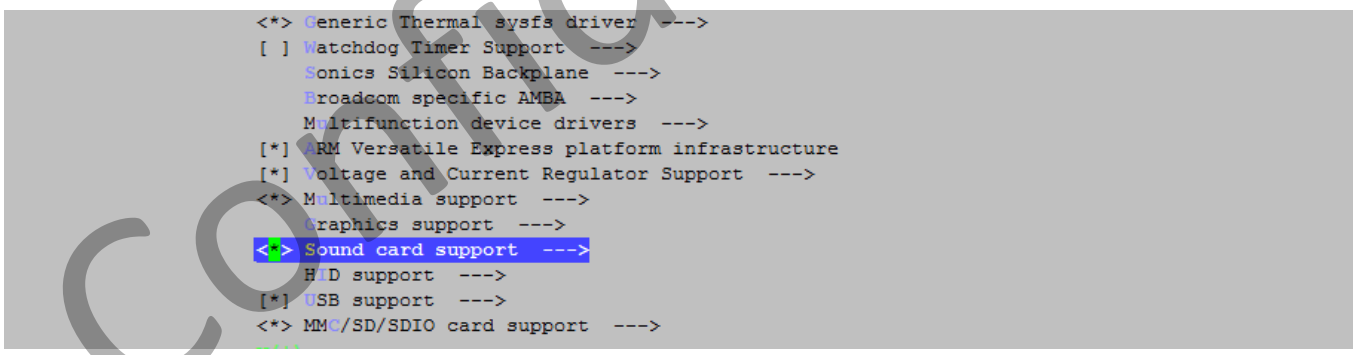
| Audio Driver: Acronyms | |
|---|---|
| **Acronym** | **Definition** |
| ALSA | Advanced Linux Sound Architecture |
| DMA | 即直接内存存取，指数据不经 cpu，直接在设备和内存，内存和内存，设备和设备之间传输. |
| Asoc | ALSA System on Chip |
| 样本长度 sample | 样本是记录音频数据最基本的单位，常见的有 8 位和 16 位 |
| 通道数 channel | 该参数为 1 表示单声道，2 则是立体声。 |
| 帧 frame | 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。 |
| 交错模式 interleaved | 是一种音频数据的记录模式，在交错模式下，数据以连续帧的形式存放，即首先记录完帧 1 的左声道样本和右声道样本（假设为立体声格式），再开始帧 2 的记录，而在非交错模式下，首先记录的是一个周期内所有帧的左声道样本，再记录右声道样本，数据是以连续通道的方式存储。不过多数情况下，我们只需要使用交错模式就可以了。 |
| Audiocodec | 芯片内置音频接口 |
| daudio | 数字音频接口，可配置成 i2s/pcm 格式标准音频接口 |

## 2.3. 音频 config 配置

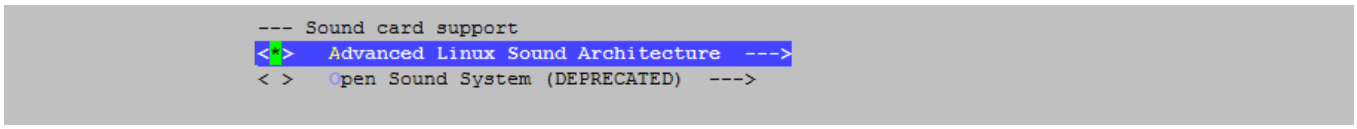
### 2.3.1. 内核配置

在 lichee/linux-3.10/ 下执行 make ARCH=arm64 menuconfig 选中如图蓝色

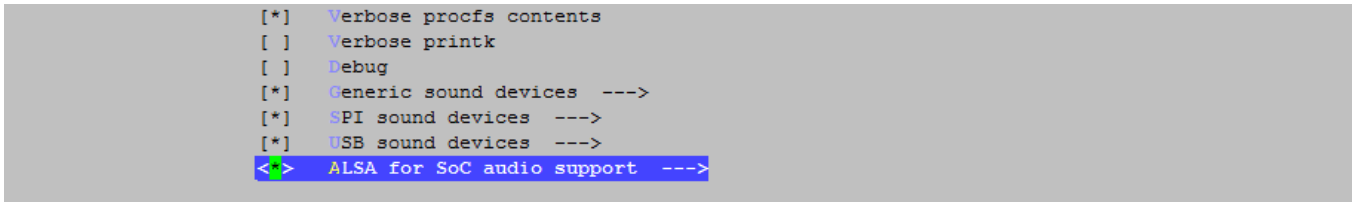

选中下图




选中下图




选中下图




选中下图

```
--- ALSA for SoC audio support
< >   SoC Audio for the Atmel System-on-Chip
< >   Synopsys I2S Device Driver
<*>   ASoC support for SUNXI  --->
< >   Build all ASoC CODEC drivers
< >   ASoC Simple sound card support
```

选中下图

```
--- ASoC support for SUNXI
<*>   ASoC support for audiocodec
<*>   ASoC support for internal-i2s
<*>   ASoC support for audiocodec machine
< >   ASoC support for daudio platform.
< >   ASoC support for vircodec.
< >   ASoC support for daudio0 machine
< >   ASoC support for daudio1 machine
<*>   ASoC support for hdmiaudio.
< >   ASoC support for spdif soundcard
[ ]   Support SUNXI AUDIO DEBUG
```

图中加载了 audiocodec 驱动和 HDMI 驱动。

audiocodec 驱动包含三部分：

ASoC support for audiocodec

ASoC support for internal-i2s

ASoC support for audiocodec machine

hdmi 驱动为：

ASoC support for hdmiaudio.

| 配置选项\声卡 | audiocodec | daudio0 | daudio1 | hdmi | spdif |
|---|---|---|---|---|---|
| ASoC support for audiocodec | √ | | | | |
| ASoC support for internal-i2s | √ | | | | |
| ASoC support for audiocodec machine | √ | | | | |
| ASoC support for daudio platform. | | √ | √ | | |
| ASoC support for vircodec. | | | | | |
| ASoC support for daudio0 machine | | √ | | | |
| ASoC support for daudio1 machine | | | √ | | |
| ASoC support for hdmiaudio. | | | | √ | |
| ASoC support for spdif soundcard | | | | | √ |
| Support SUNXI AUDIO DEBUG | | | | | |

## 2.3.2. sys_config.fex 配置

| 配置项 | 说明 |
|---|---|
| **[sndcodec]** | |
| sndcodec_used = 0x1 | 是否使用 sndcodec |
| aif2fmt = 0x3 | |

| aif3fmt = 0x3 | |
|---|---|
| aif2master = 0x1 | |
| hp_detect_case = 0x0 | 0x0:低电平触发； 0x1:高电平触发 |
| **[i2s]** | |
| i2s_used = 0x1 | 是否使用 i2s |
| **[codec]** | |
| codec_used = 0x1 | 是否使用 codec |
| headphonevol = 0x3b | 耳机音量：0x0 -- 0x3f, 0db -- (-62db) 1db/step |
| spkervol = 0x1d | 喇叭音量：0x0 -- 0x1f, 0db -- (-43.5db) 1.5db/step |
| earpiecevol = 0x1e | 听筒音量：0x0 -- 0x1f, 0db -- (-43.5db) 1.5db/step |
| maingain = 0x4 | 主麦增益：0x0 -- 0x7, 0x0:0db 0x1:24db 0x1 -- 0x7 为 3db/step |
| headsetmicgain = 0x4 | 耳麦增益：0x0 -- 0x7, 0x0:0db 0x1:24db 0x1 -- 0x7 为 3db/step |
| adcagc_cfg = 0x0 | A/D 自动增益控制 |
| adcdrc_cfg = 0x0 | A/D 动态范围压缩 |
| adchpf_cfg = 0x0 | A/D 高频滤波 |
| dacdrc_cfg = 0x0 | D/A 动态范围压缩 |
| dachpf_cfg = 0x0 | D/A 高频滤波 |
| aif2config = 0x0 | 0x0:不启用 aif2 0x1:启用 aif2 |
| aif3config = 0x0 | 0x0:不启用 aif3 0x1:启用 aif3 |
| aif1_lrlk_div = 0x40 | aif1 的 lrck 分频系数 |
| aif2_lrlk_div = 0x40 | aif2 的 lrck 分频系数 |
| pa_sleep_time = 0x15e | 喇叭进入稳定状态需要的时间 |
| dac_digital_vol = 0x9898 | Dac 数字部分音量配置 |
| gpio-spk = port:PH07<2><1><default><default> | 喇叭 gpio 口配置 |

| 配置项 | 说明 |
|---|---|
| **[spdif]** | |
| spdif_used = 0 | 是否使用 spdif |
| **[sndspdif]** | |
| sndspdif_used = 0 | 是否使用 sndspdif |

| 配置项 | 说明 |
|---|---|
| **[daudio2]** | |
| daudio2_used = 1 | 是否使用 daudio2 |
| **[sndhdmi]** | |
| sndhdmi_used = 1 | 是否使用 sndhdmi |

| 配置项 | 说明 |
|---|---|
| **[snddaudio0]** | |
| snddaudio0_used = 0 | 是否使用 snddaudio0 |
| **[daudio0]** | |

| | |
|---|---|
| pcm_lrck_period = 0x20 | |
| pcm_lrckr_period = 0x01 | 未使用 |
| slot_width_select = 0x20 | 16bits/20bits/24bits/32bits，数据 word 的宽度 |
| pcm_lsb_first = 0x0 | 0: msb first　　1: lsb first |
| tx_data_mode = 0x0 | pcm 模式中选择数据的格式 |
| rx_data_mode = 0x0 | 0: linear PCM　1: reseerved　　2: 8bit u-law　　3: 8bit a-law |
| daudio_master = 0x04 | 1:　codec 做 master, i2s 做 slave |
| | 2：codec clk slave & FRM master |
| | 3：codec clk master & frame slave |
| | 4：codec 做 slave，　i2s 做 master |
| audio_format = 0x01 | 1：标准 i2s 格式 |
| | 2：右对齐格式 |
| | 3：左对齐格式 |
| | 4：短帧格式 |
| | 5：长帧格式 |
| signal_inversion = 0x01 | 1：bclk 正常模式，　lrck 正常模式 |
| | 2：bclk 正常模式，　lrck 翻转模式 |
| | 3：bclk 翻转模式，　lrck 正常模式 |
| | 4：bclk 翻转模式，　lrck 翻转模式 |
| frametype = 0x0 | 0：短帧　　1：长帧 |
| tdm_config = 0x01 | 0：pcm 模式　1：i2s 模式 |
| daudio0_used = 0 | 是否使用 daudio0 |

| 配置项 | 说明 |
|---|---|
| **[snddaudio1]** | |
| Snddaudio1_used = 0 | 是否使用 snddaudio1 |
| **[daudio1]** | |
| pcm_lrck_period = 0x20 | |
| pcm_lrckr_period = 0x01 | 未使用 |
| slot_width_select = 0x20 | 16bits/20bits/24bits/32bits，数据 word 的宽度 |
| pcm_lsb_first = 0x0 | 0: msb first　　1: lsb first |
| tx_data_mode = 0x0 | pcm 模式中选择数据的格式 |
| rx_data_mode = 0x0 | 0: linear PCM　1: reseerved　　2: 8bit u-law　　3: 8bit a-law |
| daudio_master = 0x04 | 1:　codec 做 master, i2s 做 slave |
| | 2：codec clk slave & FRM master |
| | 3：codec clk master & frame slave |
| | 4：codec 做 slave，　i2s 做 master |
| audio_format = 0x01 | 1：标准 i2s 格式 |
| | 2：右对齐格式 |
| | 3：左对齐格式 |
| | 4：短帧格式 |
| | 5：长帧格式 |
| signal_inversion = 0x01 | 1：bclk 正常模式，　lrck 正常模式 |
| | 2：bclk 正常模式，　lrck 翻转模式 |

| | |
|---|---|
| | 3：bclk 翻转模式， lrck 正常模式 |
| | 4：bclk 翻转模式， lrck 翻转模式 |
| frametype = 0x0 | 0：短帧　　1：长帧 |
| tdm_config = 0x01 | 0：pcm 模式　　1：i2s 模式 |
| daudio1_used = 0 | 是否使用 daudio1 |

## 2.4.代码位置

代码路径：linux-3.10/sound/soc/sunxi/

| | |
|---|---|
| **公共部分** | |
| codec_utils.c | 处理 codec dai 部分， 负责提供注册 codec dai 设备的公共函数 |
| sunxi_dma.c | 处理 dma 部分， 负责提供注册 platform 设备的公共函数 |
| **Audiocodec** | |
| sunxi_codec.c | 在 asoc 框架中设计为 codec 模型 |
| sunxi_sndcodec.c | 在 asoc 框架中设计为 machine 模型 |
| sunxi_i2s.c | 在 asoc 框架中设计为 cpu_dai 模型，其中 platform 也在此注册 |
| sunxi_rw_func.c | 处理 audiocodec 读写寄存器部分 |
| **Daudio** | |
| sunxi_tdm_utils.c | Tdm 处理的公共函数 |
| sunxi_daudio.c | 在 asoc 框架中设计为 cpu_dai 模型，其中 platform 也在此注册 |
| sunxi_snddaudio0.c | 处理 daudio0 部分，在 asoc 框架中设计为 machine 模型 |
| sunxi_snddaudio1.c | 处理 daudio1 部分，在 asoc 框架中设计为 machine 模型 |
| **Hdmiaudio** | |
| sndhdmi.c | 在 asoc 框架中设计为 codec 模型 |
| sunxi_hdmi.c | 在 asoc 框架中设计为 cpu_dai 模型，其中 platform 也在此注册 |
| sunxi_sndhdmi.c | 在 asoc 框架中设计为 machine 模型 |
| **Spdif** | |
| sunxi_sndspdif.c | 在 asoc 框架中设计为 machine 模型 |
| sunxi_spdif.c | 在 asoc 框架中设计为 cpu_dai 模型，其中 platform 也在此注册 |
| **audioHub** | |
| sunxi_hub.c | 处理 audiohub 逻辑部分，在 asoc 框架中设计为 machine 模型 |

# 3. 使用描述

A64 驱动中用户对音频驱动的操作，完全按照标准 alsa-lib 或者 tinyalsa 操作即可。

## 3.1. Audiocodec

驱动中将 audiocodec 封装为 card0，采用 DAPM 机制实现。例如，用 android 提供的 tinymix 工具查看 card0 的 ctl。

| | | | | |
|---|---|---|---|---|
| 0 | INT | 2 | AIF1 ADC timeslot 0 volume | 160 160 |
| 1 | INT | 2 | AIF1 ADC timeslot 1 volume | 160 160 |
| 2 | INT | 2 | AIF1 DAC timeslot 0 volume | 159 159 |
| 3 | INT | 2 | AIF1 DAC timeslot 1 volume | 160 160 |
| 4 | INT | 2 | AIF1 ADC timeslot 0 mixer gain | 0 0 |
| 5 | INT | 2 | AIF1 ADC timeslot 1 mixer gain | 0 0 |
| 6 | INT | 2 | AIF2 ADC volume | 160 160 |
| 7 | INT | 2 | AIF2 DAC volume | 160 160 |
| 8 | INT | 2 | AIF2 ADC mixer gain | 0 0 |
| 9 | INT | 2 | ADC volume | 160 160 |
| 10 | INT | 2 | DAC volume | 0 0 |
| 11 | INT | 2 | DAC mixer gain | 0 0 |
| 12 | INT | 1 | digital volume | 0 |
| 13 | INT | 2 | ADC input gain | 3 3 |
| 14 | INT | 1 | MIC1 boost amplifier gain | 4 |
| 15 | INT | 1 | MIC2 boost amplifier gain | 4 |
| 16 | INT | 1 | LINEINL-LINEINR pre-amplifier gain | 4 |
| 17 | INT | 1 | AUXI pre-amplifier gain | 4 |
| 18 | INT | 1 | AXin to L_R output mixer gain | 3 |
| 19 | INT | 1 | MIC1 BST stage to L_R outp mixer gain | 3 |
| 20 | INT | 1 | MIC2 BST stage to L_R outp mixer gain | 3 |
| 21 | INT | 1 | LINEINL/R to L_R output mixer gain | 3 |
| 22 | INT | 1 | earpiece volume | 30 |
| 23 | INT | 1 | speaker volume | 27 |
| 24 | INT | 1 | line out volume | 3 |
| 25 | INT | 1 | headphone volume | 59 |
| 26 | ENUM | 1 | ADCR Mux | ADC |
| 27 | ENUM | 1 | ADCL Mux | ADC |
| 28 | BOOL | 1 | Line Out Mixer MIC1 boost Switch | Off |
| 29 | BOOL | 1 | Line Out Mixer MIC2 boost Switch | Off |
| 30 | BOOL | 1 | Line Out Mixer Rout_Mixer_Switch | Off |
| 31 | BOOL | 1 | Line Out Mixer Lout_Mixer_Switch | Off |
| 32 | ENUM | 1 | MIC2 SRC | MIC2 |
| 33 | BOOL | 1 | RIGHT ADC input Mixer MIC1 boost Switch | Off |
| 34 | BOOL | 1 | RIGHT ADC input Mixer MIC2 boost Switch | Off |

| | | | | |
|---|---|---|---|---|
| 35 | BOOL | 1 | RIGHT ADC input Mixer LINEINL-R Switch | Off |
| 36 | BOOL | 1 | RIGHT ADC input Mixer LINEINR Switch | Off |
| 37 | BOOL | 1 | RIGHT ADC input Mixer AUXINR Switch | Off |
| 38 | BOOL | 1 | RIGHT ADC input Mixer Rout_Mixer_Switch | Off |
| 39 | BOOL | 1 | RIGHT ADC input Mixer Lout_Mixer_Switch | Off |
| 40 | BOOL | 1 | LEFT ADC input Mixer MIC1 boost Switch | Off |
| 41 | BOOL | 1 | LEFT ADC input Mixer MIC2 boost Switch | Off |
| 42 | BOOL | 1 | LEFT ADC input Mixer LININL-R Switch | Off |
| 43 | BOOL | 1 | LEFT ADC input Mixer LINEINL Switch | Off |
| 44 | BOOL | 1 | LEFT ADC input Mixer AUXINL Switch | Off |
| 45 | BOOL | 1 | LEFT ADC input Mixer Lout_Mixer_Switch | Off |
| 46 | BOOL | 1 | LEFT ADC input Mixer Rout_Mixer_Switch | Off |
| 47 | ENUM | 1 | AIF2 DAC SRC Mux | Left_s right_s AIF2 |
| 48 | ENUM | 1 | AIF3OUT Mux | |
| 49 | BOOL | 1 | AIF2 ADR Mixer AIF1 DA0R Switch | Off |
| 50 | BOOL | 1 | AIF2 ADR Mixer AIF1 DA1R Switch | Off |
| 51 | BOOL | 1 | AIF2 ADR Mixer AIF2 DACL Switch | Off |
| 52 | BOOL | 1 | AIF2 ADR Mixer ADCR Switch | Off |
| 53 | BOOL | 1 | AIF2 ADL Mixer AIF1 DA0L Switch | Off |
| 54 | BOOL | 1 | AIF2 ADL Mixer AIF1 DA1L Switch | Off |
| 55 | BOOL | 1 | AIF2 ADL Mixer AIF2 DACR Switch | Off |
| 56 | BOOL | 1 | AIF2 ADL Mixer ADCL Switch | Off |
| 57 | ENUM | 1 | AIF2INR Mux | AIF2_DACR |
| 58 | ENUM | 1 | AIF2INL Mux | AIF2_DACL |
| 59 | ENUM | 1 | AIF2OUTR Mux | AIF2_ADCR |
| 60 | ENUM | 1 | AIF2OUTL Mux | AIF2_ADCL |
| 61 | ENUM | 1 | EAR Mux | DACR |
| 62 | ENUM | 1 | SPK_L Mux | MIXEL Switch |
| 63 | ENUM | 1 | SPK_R Mux | MIXER Switch |
| 64 | ENUM | 1 | HP_L Mux | DACL HPL Switch |
| 65 | ENUM | 1 | HP_R Mux | DACR HPR Switch |
| 66 | BOOL | 1 | Right Output Mixer DACL Switch | Off |
| 67 | BOOL | 1 | Right Output Mixer DACR Switch | Off |
| 68 | BOOL | 1 | Right Output Mixer AUXINR Switch | Off |
| 69 | BOOL | 1 | Right Output Mixer LINEINR Switch | Off |
| 70 | BOOL | 1 | Right Output Mixer LINEINL-LINEINR Switc | Off |
| 71 | BOOL | 1 | Right Output Mixer MIC2Booststage Switch | Off |
| 72 | BOOL | 1 | Right Output Mixer MIC1Booststage Switch | Off |
| 1689 | BOOL | 1 | Left Output Mixer DACR Switch | Off |
| 74 | BOOL | 1 | Left Output Mixer DACL Switch | Off |
| 75 | BOOL | 1 | Left Output Mixer AUXINL Switch | Off |
| 76 | BOOL | 1 | Left Output Mixer LINEINL Switch | Off |
| 77 | BOOL | 1 | Left Output Mixer LINEINL-LINEINR Switch | Off |
| 78 | BOOL | 1 | Left Output Mixer MIC2Booststage Switch | Off |
| 79 | BOOL | 1 | Left Output Mixer MIC1Booststage Switch | Off |

| | | | | |
|---|---|---|---|---|
| 80 | BOOL | 1 | DACR Mixer ADCR Switch | Off |
| 81 | BOOL | 1 | DACR Mixer AIF2DACR Switch | Off |
| 82 | BOOL | 1 | DACR Mixer AIF1DA1R Switch | Off |
| 83 | BOOL | 1 | DACR Mixer AIF1DA0R Switch | Off |
| 84 | BOOL | 1 | DACL Mixer ADCL Switch | Off |
| 85 | BOOL | 1 | DACL Mixer AIF2DACL Switch | Off |
| 86 | BOOL | 1 | DACL Mixer AIF1DA1L Switch | Off |
| 87 | BOOL | 1 | DACL Mixer AIF1DA0L Switch | Off |
| 88 | BOOL | 1 | AIF1 AD1R Mixer AIF2 DACR Switch | Off |
| 89 | BOOL | 1 | AIF1 AD1R Mixer ADCR Switch | Off |
| 90 | BOOL | 1 | AIF1 AD1L Mixer AIF2 DACL Switch | Off |
| 91 | BOOL | 1 | AIF1 AD1L Mixer ADCL Switch | Off |
| 92 | BOOL | 1 | AIF1 AD0R Mixer AIF1 DA0R Switch | Off |
| 93 | BOOL | 1 | AIF1 AD0R Mixer AIF2 DACR Switch | Off |
| 94 | BOOL | 1 | AIF1 AD0R Mixer ADCR Switch | Off |
| 95 | BOOL | 1 | AIF1 AD0R Mixer AIF2 DACL Switch | Off |
| 96 | BOOL | 1 | AIF1 AD0L Mixer AIF1 DA0L Switch | Off |
| 97 | BOOL | 1 | AIF1 AD0L Mixer AIF2 DACL Switch | Off |
| 98 | BOOL | 1 | AIF1 AD0L Mixer ADCL Switch | Off |
| 99 | BOOL | 1 | AIF1 AD0L Mixer AIF2 DACR Switch | Off |
| 100 | ENUM | 1 | AIF1IN1R Mux | AIF1_DA1R |
| 101 | ENUM | 1 | AIF1IN1L Mux | AIF1_DA1L |
| 102 | ENUM | 1 | AIF1IN0R Mux | AIF1_DA0R |
| 103 | ENUM | 1 | AIF1IN0L Mux | AIF1_DA0L |
| 104 | ENUM | 1 | AIF1OUT1R Mux | AIF1_AD1R |
| 105 | ENUM | 1 | AIF1OUT1L Mux | AIF1_AD1L |
| 106 | ENUM | 1 | AIF1OUT0R Mux | AIF1_AD0R |
| 107 | ENUM | 1 | AIF1OUT0L Mux | AIF1_AD0L |
| 108 | BOOL | 1 | AIF2INR Mux VIR switch aif2inr aif3Switc | Off |
| 109 | BOOL | 1 | AIF2INL Mux VIR switch aif2inl aif3Switc | Off |
| 110 | BOOL | 1 | AIF2INR Mux switch aif2inr aif2Switch | Off |
| 111 | BOOL | 1 | AIF2INL Mux switch aif2inl aif2Switch | Off |
| 112 | BOOL | 1 | External Speaker Switch | On |
| 113 | BOOL | 1 | Headphone Switch | On |
| 114 | BOOL | 1 | Earpiece Switch | On |

### 3.1.1. playback 应用

(1) 打开相关设备的 mixer 接口：

struct mixer *mixer_open(unsigned int card)

(2) 配置音频输出通路：

tinymix_set_value(mixer, 57,0)

该示例中 mixer 参数为打开的 mixer 接口，57 代表 57 号 ctl

```
static void tinymix_set_value(struct mixer *mixer, unsigned int id,
```

```
                                unsigned int value)
{
    struct mixer_ctl *ctl;
    enum mixer_ctl_type type;
    unsigned int num_values;
    unsigned int i;

    ctl = mixer_get_ctl(mixer, id);
    type = mixer_ctl_get_type(ctl);
    num_values = mixer_ctl_get_num_values(ctl);

    for (i = 0; i < num_values; i++) {
        if (mixer_ctl_set_value(ctl, i, value)) {
            fprintf(stderr, "Error: invalid value\n");
            return;
        }
    }
}
```

（3）struct pcm *pcm_open(unsigned int card, unsigned int device,
unsigned int flags, struct pcm_config *config)

：打开相应的音频流接口

(4) int pcm_write(struct pcm *pcm, void *data, unsigned int count)

：向打开的音频流中送入音频数据

应用程序一直通过 pcm_write 向 buffer 中添加数据，直至音频播放结束。

(5) int pcm_close(struct pcm *pcm)：关闭打开的音频接口

(6) void mixer_close(struct mixer *mixer)：关闭打开的 mixer

### 3.1.2. capture 应用

Capture 的配置与播放类似，详细方式可以参考 tinyalsa。

### 3.1.3. 播放录音配置 ctl 说明

#### 3.1.3.1. Headphone 输出

播放通路设置：

| number | ctl_name | value |
|---|---|---|
| 1 | Headphone volume | 0------63 |
| 2 | AIF1IN0L Mux | AIF1_DA0L |
| 3 | AIF1IN0R Mux | AIF1_DA0R |
| 4 | DACL Mixer AIF1DA0L Switch | 1 |
| 5 | DACR Mixer AIF1DA0R Switch | 1 |

| | | |
|---|---|---|
| 6 | HP_R Mux | DACR HPR Switch |
| 7 | HP_L Mux | DACL HPL Switch |
| 8 | Headphone Switch | 1 |

### 3.1.3.2. Speaker 输出

播放通路设置：

| number | ctl_name | value |
|---|---|---|
| 1 | speaker volume | 0------31 |
| 2 | AIF1IN0R Mux | AIF1_DA0R |
| 3 | AIF1IN0L Mux | AIF1_DA0L |
| 4 | DACR Mixer AIF1DA0R Switch | 1 |
| 5 | DACL Mixer AIF1DA0L Switch | 1 |
| 6 | Right Output Mixer DACR Switch | 1 |
| 7 | Left Output Mixer DACL Switch | 1 |
| 8 | SPK_L Mux | MIXEL Switch |
| 9 | SPK_R Mux | MIXER Switch |
| 10 | External Speaker Switch | 1 |

### 3.1.3.3. 主 mic 录音

录音通路设置：

| number | ctl_name | value |
|---|---|---|
| 1 | AIF1OUT0L Mux | AIF1_AD0L |
| 2 | AIF1OUT0R Mux | AIF1_AD0R |
| 3 | AIF1 AD0L Mixer ADCL Switch | 1 |
| 4 | AIF1 AD0R Mixer ADCR Switch | 1 |
| 5 | ADCR Mux | ADC |
| 6 | ADCL Mux | ADC |
| 7 | LADC input Mixer MIC1 boost Switch | 1 |
| 8 | RADC input Mixer MIC1 boost Switch | 1 |
| 9 | MIC1 boost amplifier gain | 0----7 |

### 3.1.3.4. 耳 mic 录音

录音通路设置：

| number | ctl_name | value |
|---|---|---|
| 1 | AIF1OUT0L Mux | AIF1_AD0L |
| 2 | AIF1OUT0R Mux | AIF1_AD0R |
| 3 | AIF1 AD0L Mixer ADCL Switch | 1 |
| 4 | AIF1 AD0R Mixer ADCR Switch | 1 |
| 5 | ADCR Mux | ADC |
| 6 | ADCL Mux | ADC |
| 7 | LADC input Mixer MIC2 boost Switch | 1 |
| 8 | RADC input Mixer MIC2 boost Switch | 1 |
| 9 | MIC2 SRC | MIC2 |
| 10 | MIC2 boost amplifier gain | 0---------7 |

#### 3.1.3.5. D-MIC 录音

录音通路设置：

| number | ctl_name | value |
|---|---|---|
| 1 | AIF1OUT0L Mux | AIF1_AD0L |
| 2 | AIF1OUT0R Mux | AIF1_AD0R |
| 3 | AIF1 AD0L Mixer ADCL Switch | 1 |
| 4 | AIF1 AD0R Mixer ADCR Switch | 1 |
| 5 | ADCR Mux | DMIC |
| 6 | ADCL Mux | DMIC |

其他详细的通路设计操作请参考 hal 层设计，以及 audio-path 文件。

## 3.2. Daudio

在 A64 平台中，存在两个 daudio 模块，可以外挂相应的设备，支持 pcm 和 i2s 格式传输，目前 pcm 格式只支持 8k，i2s 格式支持 8k 至 192k。

同样 daudio 的使用同其他声卡一致

### 3.2.1. playback 应用

（1）struct pcm *pcm_open(unsigned int card, unsigned int device,
unsigned int flags, struct pcm_config *config)

：打开相应的音频流接口

（2）int pcm_write(struct pcm *pcm, void *data, unsigned int count)

：向打开的音频流中送入音频数据

应用程序一直通过 pcm_write 向 buffer 中添加数据，直至音频播放结束。

(3) int pcm_close(struct pcm *pcm)：关闭打开的音频接口

### 3.2.2. capture 应用

Capture 的配置与播放类似，详细方式可以参考 tinyalsa。

## 3.3. Hdmi

Hdmi 声卡注册为 card1。

### 3.3.1. playback 应用

（1）struct pcm *pcm_open(unsigned int card, unsigned int device,
unsigned int flags, struct pcm_config *config)
：打开相应的音频流接口
（2）int pcm_write(struct pcm *pcm, void *data, unsigned int count)
：向打开的音频流中送入音频数据
应用程序一直通过 pcm_write 向 buffer 中添加数据，直至音频播放结束。
（3）int pcm_close(struct pcm *pcm)：关闭打开的音频接口

## 3.4. Spdif

Spdif 声卡注册为 card2。

### 3.4.1. playback 应用

（1）struct pcm *pcm_open(unsigned int card, unsigned int device,
unsigned int flags, struct pcm_config *config)
：打开相应的音频流接口
（2）int pcm_write(struct pcm *pcm, void *data, unsigned int count)
：向打开的音频流中送入音频数据
应用程序一直通过 pcm_write 向 buffer 中添加数据，直至音频播放结束。
（3）int pcm_close(struct pcm *pcm)：关闭打开的音频接口