全志科技
Allwinner Technology

V40 项目

音频模块说明书 V1.0

文档履历

版本号	日期	制/修订人	内容描述
V1.0	2016/06/07		初始版本

目 录

1. 前言

1.1. 编写目的

本文档目的是让开发者了解 V40 音频系统框架，能够在 V40 平台上开发新的音频方案。

1.2. 适用范围

本模块说明适用于 V40 平台。

1.3. 相关人员

音频系统开发及维护人员。

1.4. 相关术语

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.
OSS	Open Sound System
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(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。不过多数情况下，我们只需要使用交错模式就可以了。
Hdmiaudio	内置 hdmi 音频接口
SPDIF	Sony/Philips Digital Interface Format 是 SONY、PHILIPS 数字音频接口的简称，一般使用同轴电缆或光纤传输，SPDIF 分为输出（SPDIF OUT）和输入（SPDIF IN）两种，V40 只支持 spdif out
I2S	数字音频接口，一般用于外挂 codec

AGC	自动增益控制
DRC	音频输出动态范围控制
daudio	平台数字音频接口的统称，可配置成 i2s/pcm 格式标准音频接口
xrun	音频流异常状态

2. V40 音频系统框架概述

2.1. V40 原型机音频硬件框架图

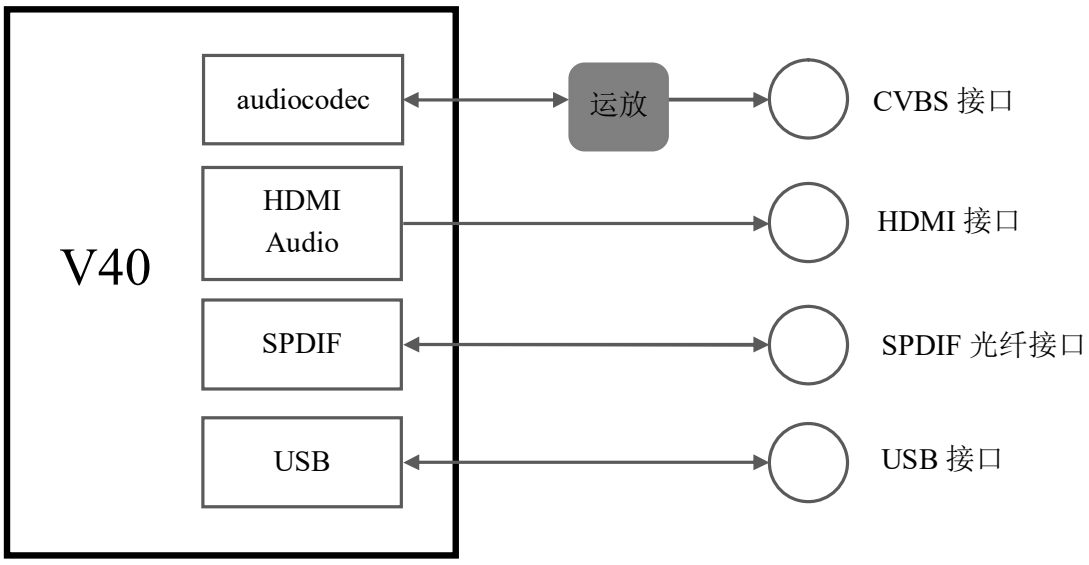


图 2-1 V40 原型机音频硬件框架图

各个设备对应的设备节点：

硬件接口/设备	设备节点	/sys/class/sound/cardX/id	备注
CVBS/耳机接口	/dev/snd/controlC0 /dev/snd/pcmC0D0c /dev/snd/pcmC0D0p	audiocodec	Controlx 为控制器 pcmCxDxc 为输入节点 pcmCxDxp 为输出节点 设备属性节点在： /sys/class/sound/cardX,
HDMI 接口	/dev/snd/controlC1 /dev/snd/pcmC1D0p	sndhdmi	
SPDIF 接口	/dev/snd/controlC2 /dev/snd/pcmC2D0c /dev/snd/pcmC2D0p	sndspdif	
USB Audio	/dev/snd/controlC3 /dev/snd/pcmC3D0c /dev/snd/pcmC3D0p	从 usb 设备中读取获得	

可以输入一下命令查看系统挂载上的声卡：

```
cat /proc/asound/cards
 0 [audiocodec     ]: audiocodec - audiocodec
                      audiocodec
 1 [sndhdmi        ]: sndhdmi - sndhdmi
                      sndhdmi
 2 [sndspdif       ]: sndspdif - sndspdif
                      sndspdif
 3 [Device         ]: USB-Audio - USB Audio Device
```

2.2. V40 软件框架图

V40 音频软件框架，大部分沿用原生系统框架。

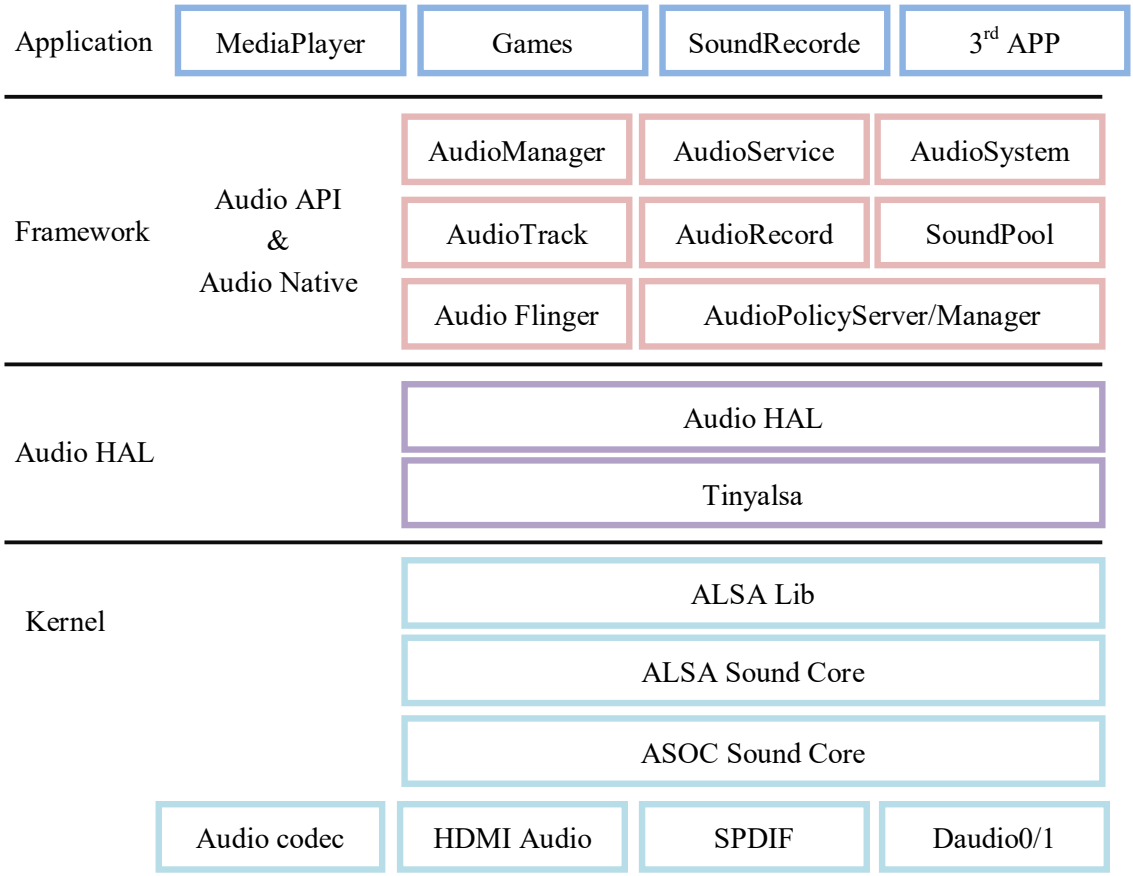


图 3-1 V40 系统软件结构图

与 Android 原生框架不同是，V40 盒子有一套独立的音频输入输出策略。同时具备原生系统所不具备的功能，例如支持 HDMI，USB，CVBS 等接口的热插拔，支持单/多路输出，支持音频透传等。

3. V40 音频模块介绍

在 V40 中，存在 5 个音频设备。存在 5 个音频设备。分别为 audiocodec，daudio0，daudio1，HDMI(daudio2)，owa(spdif)。每一个音频设备都采用 asoc 架构实现。

asoc 是建立在标准 alsa 驱动层上，为了更好地支持嵌入式处理器和移动设备中的音频 codec 的一套软件体系，asoc 将音频系统分为 3 部分：Machine，Platform 和 Codec。

（1）Codec 驱动

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

所有的 Codec 驱动都要提供以下特性：

Codec DAI (Digital Audio Interface) 和 PCM 的配置信息；

Codec 的 IO 控制方式（I2C，SPI 等）；

Mixer 和其他的音频控件；

Codec 的 ALSA 音频操作接口；

必要时，也可以提供以下功能：

DAPM 描述信息；

DAPM 事件处理程序；

DAC 数字静音控制；

（2）Platform 驱动

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

（3）Machine 驱动

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

3.1. 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， 11.025 KHz, 12 KHz, 16 KHz, 22.05 KHz, 24 KHz, 32 KHz, 44.1 KHz , 48 KHz, 96KHz, 192KHz)；
- 支持 mono 和 stereo 模式；
- 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/24bit 数据精度；
- 支持两路 ADC，采样率为 8khz~48khz，支持两路 DAC，采样率为 8khz~192khz；
- 支持三路输入；

3.2. Daudio0/1 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;
- 支持同时 playback 和 record(全双工模式)；
- 支持 i2s、pcm 配置。
- 支持 16bit/24bit 数据精度

3.3. Daudio2(hdmiaudio)模块功能

Daudio2(hdmiaudio)驱动所具有的功能：

- 支持多种采样率格式(32khz，44.1，48khz，96khz，192khz)；
- 支持 mono 和 stereo 模式；
- 只支持 playback 模式，不支持 record 模式；
- 支持 16bit/24bit/32bit 数据精度；
- 支持 raw 数据输出；

Hdmiaudio 功能采用 daudio2 接口。

3.4. Owa(Spdif)模块功能

spdif 驱动所具有的功能：

- 支持多种采样率格式(22.05khz，24khz，32khz，44.1khz，48khz，88.2khz，96khz，176.4khz，192khz)；
- 支持 mono 和 stereo 模式；
- 只支持 playback 模式，不支持 record 模式；
- 支持 16bit/24bit 数据精度；
- 支持 raw 数据输出；

4. V40 音频配置

4.1. 源码结构

关键源码目录：lichee/linux-3.10/sound/soc/sunxi/

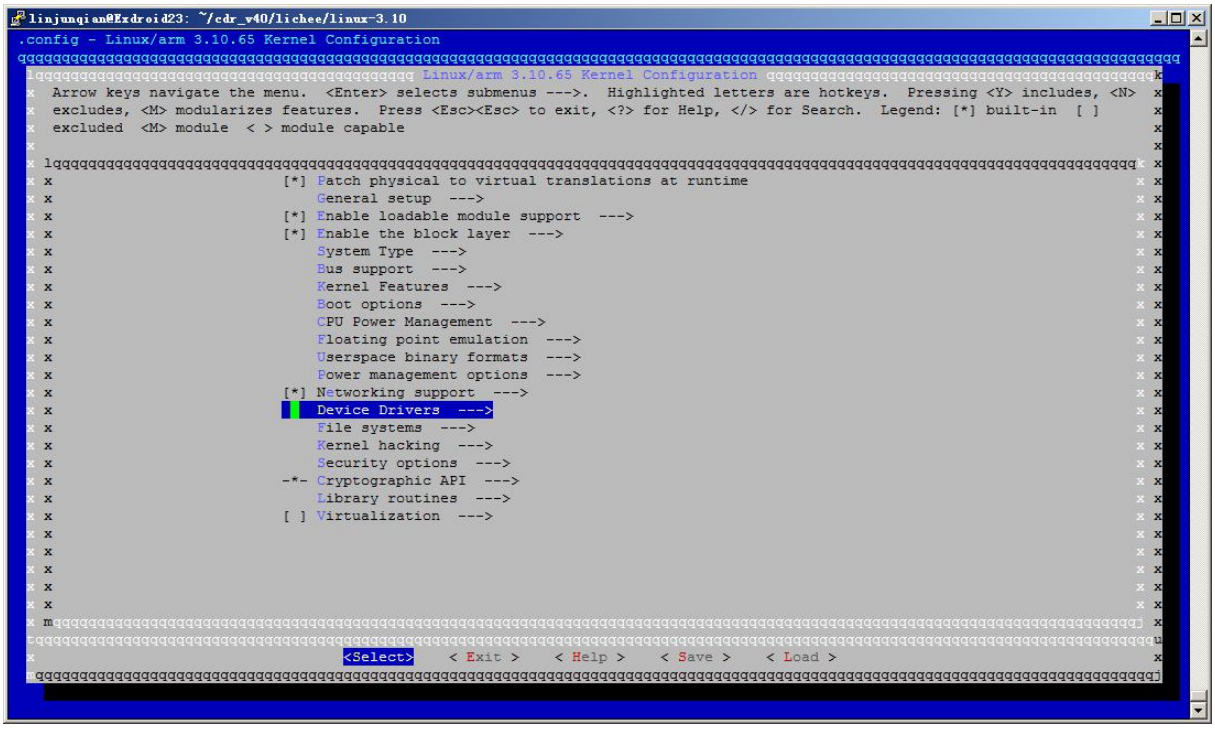
```
linjunqian@Exdroid23:~/cdr_v40/lichee/linux-3.10/sound/soc/sunxi$ l
codec_utils.c*    snddaudio.c*          sun8iw11_codec.c      sunxi_dma.c*    sunxi_i2s.h*          sunxi-sndspdif.c*
codec-utils.h*    sndhdmi.c*            sun8iw11_codec.h*     sunxi_dma.h*    sunxi_rw_func.c*      sunxi_spdif.c*
cs4385.c*         spdif-utils.c         sun8iw11_sndcodec.c*  sunxi_dmic.c*   sunxi_rw_func.h*      sunxi_spdif.h*
cs4385.h*         sun50iw2_codec.c      sunxi_codec.c*        sunxi_dmic.h*   sunxi_sndcodec.c*     sunxi_tdmhdmi.c*
d.patch           sun50iw2_codec.h      sunxi_codec.h*        sunxi_dsd.c*    sunxi-snddaudio0.c*   sunxi_tdm_utils.c*
Kconfig*          sun50iw2_sndcodec.c   sunxi_cpudai.c*       sunxi_dsd.h*    sunxi-snddaudio1.c*   sunxi_tdm_utils.h*
Makefile*         sun8iw10_codec.c      sunxi_cpudai.h*       sunxi_hub.c*    sunxi-snddmic.c*
modules.builtin   sun8iw10_codec.h*     sunxi_daudio.c*       sunxi_hub.h*    sunxi-snddsd.c*
modules.order     sun8iw10_sndcodec.c*  sunxi_daudio.h        sunxi_i2s.c*    sunxi-sndhdmi.c*
```

4.2. 内核配置

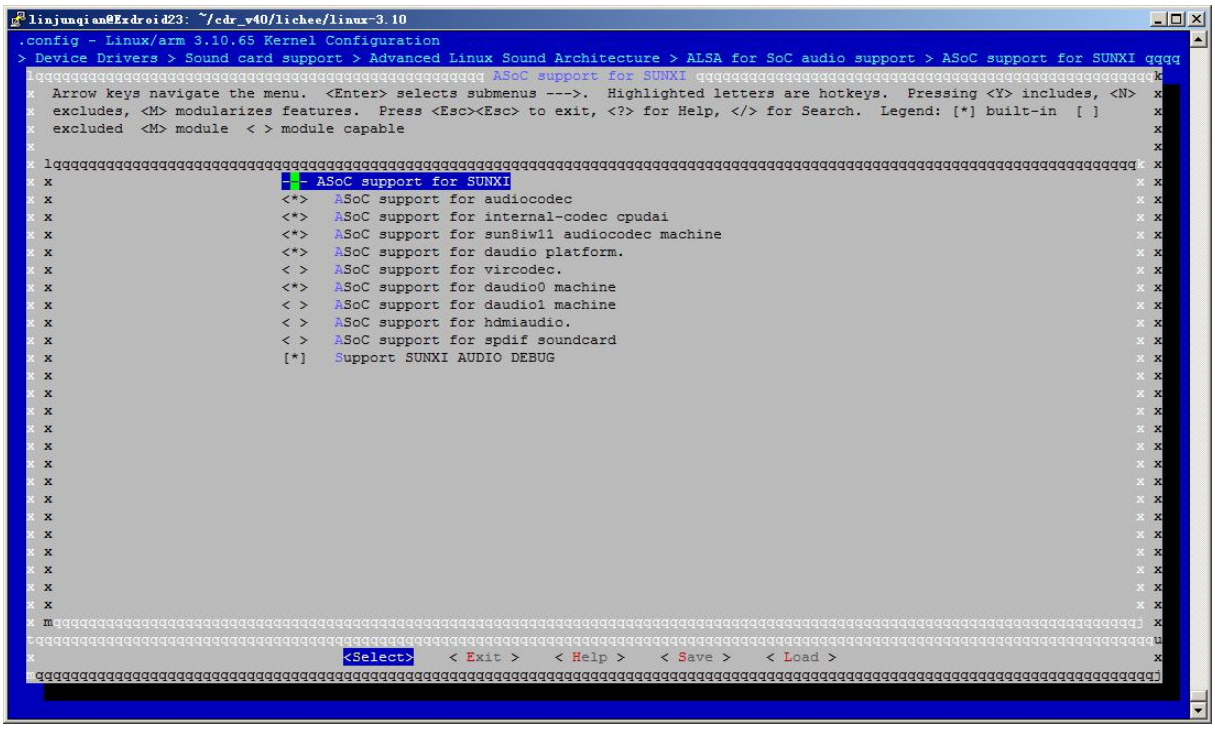
4.2.1. Menuconfig 配置

在编译服务器上，目录为\lichee\linux-3.10 上，输入命令：

```
make ARCH=menuconfig
```



- 音频驱动配置：



V40 原型机上，CODEC 直接编入内核，其他默认不编译，一般 HDMI 和 CODEC 是编译为 built-in，spdif 或者 USB Audio 的驱动着编成 ko。

- USB Audio 的驱动加载在：

android\device\softwinner\magton-common\init.sun8iw11p1.rc

```
on post-fs
#insmod usb audio card ko
    insmod /system/vendor/modules/snd-hwdep.ko
    insmod /system/vendor/modules/snd-usbmidi-lib.ko
    insmod /system/vendor/modules/snd-usb-audio.ko
```

4.3. 系统配置（sys_config.fex）

配置文件的位置：不同的方案配置不一样，以 magton-p1 方案为例，其配置文件位于 lichee\tools\pack\chips\sun8iw11p1\configs\magton-p1\sys_config.fex 目录下。

4.3.1. V40 内置模拟音频通道(codec)配置

- [codec]

配置项	配置项含义
codec_used	是否使用 V40 模拟音频输入输出 0x1：使用，0x0：不使用
headphonevol	HP 默认音量设置，最大值是 0x3f
spkervol	SPK 默认音量设置，最大值是 0x1f
maingain	Mic1 前端增益，最大值是 0x7
hp_dirused	HP 开关，0x1：开启，0x0：不开启

adcagc_cfg	Adc 自动增益控制，0x1：开启，0x0：不开启
adcdrc_cfg	Drc 动态范围控制，0x1：开启，0x0：不开启
adchpf_cfg	Adc 端高通滤波，0x1：开启，0x0：不开启
dacdrc_cfg	播放动态音效调节，0x1：开启，0x0：不开启
dachpf_cfg	播放通路高通滤波开启，0x1：开启，0x0：不开启
gpio-spk	外部功放使能脚

配置举例：

```
;-------------------------------------------------------------------------------------
;       NOTE :Make sure audiocodec_machine_used = 0x1,sun50i2s_used = 0x1
;         sun50codec_used = 0x1,if register the sound card audiocodec.
;-------------------------------------------------------------------------------------
[sndcodec]
sndcodec_used = 0x1
;-------------------------------------------------------------------------------------
[codec]
codec_used = 0x1
headphonevol =0x3b
spkervol =   0x1b
maingain =   0x4
hp_dirused = 0x1
adcagc_cfg =   0x0
adcdrc_cfg =   0x0
adchpf_cfg =   0x0
dacdrc_cfg =   0x0
dachpf_cfg =   0x0
gpio-spk = port:PB05<1><1><default><default>
```

4.3.2. HDMI 音频

HDMI 音频设备在 IC 内部直接挂在 Daudio2 上。

- [audiohdmi]

配置项	配置项含义
audiohdmi_used	是否开启 HDMI codec，1：开启，0：不开启

- [hdmi_machine_used]

配置项	配置项含义
hdmi_machine_used	是否开户 HDMI machine，1：开启，0：不开启

配置举例：

```
;-------------------------------------------------------------------------------------
;       NOTE :Make sure audiohdmi_used = 0x1,hdmi_machine_used = 0x1,
;         if register the sound card hdmi.
;-------------------------------------------------------------------------------------
[audiohdmi]
audiohdmi_used = 0
```

```
[hdmi_machine_used]
hdmi_machine_used = 01
```

4.3.3. 数字音频总线（S/PDIF）

Spdif 数字音频总线，一般有同轴电缆和光纤接口两种，使用光纤接口较多。Spdif 接口提供输入和输出功能。可根据需求配置。

- [audiospdif]

配置项	配置项含义
audiospdif_used	是否开启 spdif codec，1：开启，0：不开启

- [spdif_machine]

配置项	配置项含义
spdif_machine_used	是否开启 spdif machine，1：开启，0：不开启

配置举例：

```
;-----------------------------------------------------------------------------------
;       NOTE :Make sure audiospdif_used = 0x1,spdif_machine_used = 0x1,
;         if register the sound card spdif.
;-----------------------------------------------------------------------------------
[audiospdif]
audiospdif_used          = 0
[spdif_machine]
spdif_machine_used   = 0
```

4.3.4. 外挂音频芯片接口(I2S/PCM)

以 daudio0 为例，配置说明如下：

- [daudio0]

Daudio0 配置	Daudio0 tdm 配置说明
snddaudio0_used	snddaudio0_used==1 使用 daudio0 machine； snddaudio0_used ==0 不使用 daudio0 machine；
pcm_lrck_period	it is used to program the number of BCLKS per channel of sample frame. This value is interpreted as follow PCM mode: Number of BCLKs within(Left + Right)channel width 注意在 pcm 模式下，pcm_lrck_period 代表左和右声道相加，2 个声道的大小； I2S/Left-Justified/Right-Justified mode: Number of BCLKs within each individual channel width(Left or Right)，pcm_lrck_period 代表左或者右声道，一个声道的大小； 在 i2s 模式下,一个 lrck 的宽度:2*32。假如 fs=48k,那么需要的 bclk 是 3.072M = 2*32*48k; bclk_div = 24.576M/3.072M=8; 在 pcm 模式下，一个 lrck 的宽度就是 32。假如 fs=8k，那么需要的 bclk 是：32*8k=256k

	并需要确保：i2s 模式下:channel*slot_width_select <= 2 * pcm_lrck_period; pcm 模式下：channel*slot_width_select <= pcm_lrck_period;
pcm_lrckr_period	未使用
slot_width_select	16bits/20bits/24bits/32bits 数据 word 的宽度。slot_width_select 必须大于等于采样精度，要不然会出现数据丢失。相当于 slot_width_select 装载 sample_resolution 传输，所以 slot_width_select > sample_resolution;这里 slot_width_select 对 i2s 模式，pcm 模式都有效。
pcm_lsb_first	0: msb first; 1: lsb first 对应每一个 slot，数据有效位从 lsb 算（数据的低位传开始）还是从 msb 算（数据的高位传开始）。
tx_data_mode	0: linear PCM; 1: reserved; 2: 8bit u-law; 3: 8bit a-law pcm 传输模式中选择数据的格式；注意：如果数据格式选择的是 u-law,a-law，而 slot_width 选择 16bit，但是 u-law，a-law 是 8bit，后续 8bit 用 0 填充
rx_data_mode	0: linear PCM; 1:reserved ; 2: 8bit u-law; 3: 8bit a-law pcm 传输模式中选择数据的格式
daudio_master	1 SND_SOC_DAIFMT_CBM_CFM(codec clk & FRM master) 表示 codec 做 master，i2s 做 slave 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) use 表示 codec 做 slave，i2s 做 master
audio_format	1 SND_SOC_DAIFMT_I2S(standard i2s format). use 表示标准 i2s 格式 2 SND_SOC_DAIFMT_RIGHT_J(right justfied format). 表示右对齐格式 3 SND_SOC_DAIFMT_LEFT_J(left justfied format) 表示左对齐格式 4 SND_SOC_DAIFMT_DSP_A 短帧模式 并设置 frametype 为 0.短帧 5 SND_SOC_DAIFMT_DSP_B 长帧模式 并设置 frametype 为 1.长帧
signal_inversion	1 SND_SOC_DAIFMT_NB_NF(normal bit clock + frame) use 表示 bclk 采用正常模式，lrck 也正常模式 2 SND_SOC_DAIFMT_NB_IF(normal BCLK + inv FRM) 表示 bclk 采用正常模式，lrck 采用翻转模式 3 SND_SOC_DAIFMT_IB_NF(invert BCLK + nor FRM) use 表示 bclk 采用翻转模式，lrck 采用正常模式 4 SND_SOC_DAIFMT_IB_IF(invert BCLK + FRM) 表示 bclk 采用翻转模式，lrck 采用翻转模式 信号的翻转，比如标准的 I2S 模式，如果 lrck 翻转是模式，那么用示波器测量，左右声道是跟标准 i2s 模式相反的。如果 bclk 是翻转模式，那么用示波器测量，BCLK 信号是翻转的。参考上面的标准 i2s 时序图。

frametype	0: short frame = 1 clock width; 1: long frame = 2clock width 长帧还是短帧。2 个 bclk 代表长帧（SND_SOC_DAIFMT_DSP_B 模式），1 个 bclk 代表短帧（SND_SOC_DAIFMT_DSP_A 模式）。用于 pcm 模式的配置中，i2s 无效
tdm_config	0 is pcm.1 is i2s 0 配置成 pcm 模式；1 配置成 i2s 模式
daudio0_used	Daudio0_used==1 使用 daudio0；daudio0_used ==0 不使用 daudio0；

配置举例：

```
;-----------------------------------------------------------------------------------------
;       NOTE :Make sure snddaudio0_used = 0x1,daudio0_used = 0x1,
;         if register the sound card daudio0.
;-----------------------------------------------------------------------------------------
[snddaudio0]
snddaudio0_used = 0
;-----------------------------------------------------------------------------------------
[daudio0]
pcm_lrck_period =   0x20
pcm_lrckr_period =   0x01
slot_width_select =   0x20
pcm_lsb_first =   0x0
tx_data_mode =   0x0
rx_data_mode =   0x0
daudio_master =   0x04
audio_format =   0x01
signal_inversion =   0x01
frametype =   0x0
tdm_config =   0x01
mclk_div = 0x0
daudio0_used = 0
```

同理，[daudio1]配置为：

```
;-----------------------------------------------------------------------------------------
;       NOTE :Make sure snddaudio_used = 0x1,daudio1_used = 0x1,
;         if register the sound card DAUDIO1.
;-----------------------------------------------------------------------------------------
[snddaudio1]
snddaudio_used = 1
;-----------------------------------------------------------------------------------------
[daudio1]
pcm_lrck_period =   0x20
pcm_lrckr_period =   0x01
slot_width_select =   0x10
pcm_lsb_first =   0x0
tx_data_mode =   0x0
```

```
rx_data_mode =   0x0
daudio_master =   0x04
audio_format =   0x01
signal_inversion =   0x01
frametype =   0x0
tdm_config =   0x01
daudio1_used = 1
```

在 V40 CDR 产品中，daudio1 是与蓝牙相连，配置为 pcm 接口。

5. 音频输入输出切换策略

5.1. V40 音频输出策略

目前在 V40 CDR 产品中，输出只有 speaker 和 headphone，一般情况 headphone 将会接 FM 发射模块；而在原型机上默认是从 speaker 输出。可自定义输出策略。

5.2. 音频输入策略

目前 V40 CDR 产品中，输入是双 mic 输入和蓝牙输入。蓝牙输入是用于蓝牙模块输入信号给到 V40 Codec。蓝牙输入只有在蓝牙音乐和蓝牙电话时才有效。而双 mic 输入是主要用于语音识别和作为蓝牙通话的输入。

后续有新的需求可以自行扩展。

6. 蓝牙通话模式

蓝牙模块请阅读相应说明文档，在此只简单说明一下蓝牙通话模式的实现。在 V40 中，蓝牙有两种工作模式：蓝牙音乐模式和蓝牙通话模式。

蓝牙音乐模式，具体实现可以查阅 android/packages/apps/Bluetooth/src/com/android/bluetooth/a2dp 的代码，通话 AudioRecord 录 uart 的数据，再将其用 AudioTrack 播放输出。

在此重点说一下蓝牙通话模式，在 hal 层中开两线程来实现数据的传输，具体实现请阅读 android/device/softwinner/common/hardware/audio/audio_bt.c 和 audio_hw.c。

蓝牙的上行（从从蓝牙到机器）在此实现：

```
// for bt call read from bt in pcm, send data to codec output
void* btcall_thread1(void *data)
```

蓝牙的下行（从机器的 mic 到蓝牙）在此实现：

```
// for bt call read from codec mic ,send data to bt pcm
void* btcall_thread2(void *data)
```

当上层接收到蓝牙通话的请求时，会以 hfp_enable 为 key，然后通过 setParameter 接口通知 Audio hal 层进入蓝牙通话模式：

```
    ret = str_parms_get_str(parms, AUDIO_PARAMETER_CDR_BTCALL, value, sizeof(value));
       if (ret >= 0) {
           if (strcmp(value, AUDIO_PARAMETER_CDR_BTCALL_ON) == 0) {
               pthread_mutex_lock(&adev->bt_data->lock);
               pthread_mutex_lock(&adev->lock);
               adev->bt_on = true;
               adev->bt_data->start_work = 1;
               adev->bt_data->exit_work = 0;
               pthread_mutex_unlock(&adev->lock);
               pthread_cond_signal(&adev->bt_data->cond);
               ALOGV("pthread_cond_signal");
               pthread_mutex_unlock(&adev->bt_data->lock);

               pthread_mutex_lock(&adev->bt_data->lock2);
               pthread_mutex_lock(&adev->lock);
               adev->bt_data->start_work2 = 1;
               adev->bt_data->exit_work2 = 0;
               pthread_mutex_unlock(&adev->lock);
               pthread_cond_signal(&adev->bt_data->cond2);
               ALOGV("pthread_cond_signal2");
               pthread_mutex_unlock(&adev->bt_data->lock2);
           } else {
               pthread_mutex_lock(&adev->bt_data->lock);
               adev->bt_on = false;
               adev->bt_data->start_work = 0;
               adev->bt_data->exit_work = 0;
```

```
                ALOGV("pthread1 waiting...");
                pthread_mutex_unlock(&adev->bt_data->lock);

                pthread_mutex_lock(&adev->bt_data->lock2);
                adev->bt_data->start_work2 = 0;
                adev->bt_data->exit_work2 = 0;
                ALOGV("pthread2 waiting...");
                pthread_mutex_unlock(&adev->bt_data->lock2);
            }
        }
```

7. Declaration