密级状态：绝密(　　)　　秘密(　　)　　内部(　)　　公开(√)

RK3308 Audio Codec 介绍

（第三系统产品部）

文件状态： [　] 正在修改 [√] 正式发布	当前版本：	V0.3.0
	作　　者：	郑兴
	完成日期：	2018-09-04
	审　　核：	
	完成日期：	2018-09-04

福州瑞芯微电子有限公司
Fuzhou　Rockchips　Semiconductor　Co . , Ltd

版 本 历 史

版本	作者	日期	描述
v0.1.0	Xing Zheng	2018/05/01	添加 RK3308 Audio 介绍初始版本
v0.2.0	Xing Zheng	2018/08/16	添加 RK3308 CODEC 常用属性配置描述
v0.3.0	Xing Zheng	2018/09/04	添加 HPF 和 AGC 相关配置说明

目 录

1. RK3308 Audio 接口介绍

RK3308 有丰富的音频相关接口，包括 I2S，PCM，TDM，PDM，SPDIF，VAD(Voice Activity Detection）以及内置 CODEC：

- I2S with 2 channel
- I2S with 8 channel
- I2S with 16 channel
- PDM with 8 channel
- TDM with 8 channel
- SPDIF
- Voice Activity Detection(VAD)
- Embedded Audio Codec

更多具体 Audio 接口细节请参看《RK3308 TRM Chapter 22 Audio Subsystem》章节。

2. RK3308 EVB MIC 类型介绍

RK3308 支持不同类型接口的 MIC 矩阵是其一大亮点，EVB 上有三种接口类型的 MIC 可供客户选择：

- 模拟 MIC（以下简称 AMIC）与 RK3308 内置 CODEC 连接
- 数字 MIC（以下简称 DMIC）与 RK3308 I2S 连接
- PDM MIC 与 RK3308 PDM 连接

因此，RK3308 的 kernel 工程也是依据以上三种不同的 MIC 来区分板级 dts 文件。除了公共的部分 ACODEC 的配置需引用：

arch/arm64/boot/dts/rockchip/rk3308-evb-v10.dtsi

最末端的子板级 dts 他们分别是：

arch/arm64/boot/dts/rockchip/rk3308-evb-amic-v10.dts
arch/arm64/boot/dts/rockchip/rk3308-evb-dmic-i2s-v10.dts
arch/arm64/boot/dts/rockchip/rk3308-evb-dmic-pdm-v10.dts

下面简单介绍一下 Audio 部分相关的 dts 配置。

2.1 RK3308 AMIC Board 配置

因为 RK3308 EVB 都需要启用内置的 CODEC 功能，将 rk3308-evb-amic-v10.dts 中 CODEC 的配置挪到公共的 rk3308-evb-v10.dtsi 中，所以 rk3308-evb-amic-v10.dts 里仅仅是引用了 rk3308-evb-v10.dtsi，并作了空的描述（后期根据项目需要可根据 AMIC Board 独有的功能模块在其中自行添加）：

```
#include "rk3308-evb-v10.dtsi"

/ {
        model = "Rockchip RK3308 evb analog mic board";
        compatible = "rockchip,rk3308-evb-amic-v10", "rockchip,rk3308";
};
```

其中在 rk3308-evb-v10.dtsi 中添加的 CODEC 部分（DMIC Board 和 PDM MIC Board 也包含了它）：

```
        sound {
                compatible = "simple-audio-card";
                simple-audio-card,format = "i2s";
                simple-audio-card,name = "rockchip,rk3308-acodec";
                simple-audio-card,mclk-fs = <256>;
                simple-audio-card,codec-hp-det;
                simple-audio-card,widgets =
                        "Headphone", "Headphones";
                simple-audio-card,cpu {
                        sound-dai = <&i2s_8ch_2>;
                };
                simple-audio-card,codec {
                        sound-dai = <&acodec>;
                };
        };
```

启用了内部 8 通道的 i2s_8ch_2：

```
&i2s_8ch_2 {
        status = "okay";

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

启用内置 CODEC：

```
&acodec {
        status = "okay";

        #sound-dai-cells = <0>;

        hp-ctl-gpios = <&gpio0 RK_PA1 GPIO_ACTIVE_HIGH>;
        spk-ctl-gpios = <&gpio0 RK_PA5 GPIO_ACTIVE_HIGH>;
};
```

2.2 RK3308 DMIC Board 配置

RK3308 EVB DMIC Board 描述如下：

```
        i2s-dmic-array {
                compatible = "simple-audio-card";
                simple-audio-card,format = "i2s";
                simple-audio-card,name = "rockchip,i2s-dmic-array";
                simple-audio-card,mclk-fs = <256>;
                simple-audio-card,cpu {
                        sound-dai = <&i2s_8ch_0>;
                };
                simple-audio-card,codec {
                        sound-dai = <&dummy_codec>;
                };
        };
```

因为 i2s_8ch_0 接的是 I2S 接口的功放，所以对 sound framework 来说需要注册一个 dummy_codec 来保证音频设备的完整性：

```
&dummy_codec {
        status = "okay";
        #sound-dai-cells = <0>;
};

&i2s_8ch_0 {
        status = "okay";
        #sound-dai-cells = <0>;
};
```

2.3 RK3308 PDM MIC Board 配置

与 DMIC Board 类似，PDM MIC Board 的描述：

```
        pdm-mic-array {
                compatible = "simple-audio-card";
                simple-audio-card,name = "rockchip,pdm-mic-array";
                simple-audio-card,cpu {
                        sound-dai = <&pdm_8ch>;
                };
                simple-audio-card,codec {
                        sound-dai = <&dummy_codec>;
                };
        };
```

开启 pdm_8ch 的同时，也需要打开 dummy_codec：

```
&dummy_codec {
        status = "okay";
        #sound-dai-cells = <0>;
};

&pdm_8ch {
        status = "okay";
        #sound-dai-cells = <0>;
        pinctrl-names = "default";
        pinctrl-0 = <&pdm_m2_clk
                     &pdm_m2_clkm
                     &pdm_m2_sdi0
                     &pdm_m2_sdi1
                     &pdm_m2_sdi2
                     &pdm_m2_sdi3>;
};
```

3. RK3308 常用 Audio 相关配置

Audio 是消费电子产品中不可或缺、非常重要的一个功能模块，Linux 平台上相关的工具也非常多样。所以，RK3308 EVB 上也整合了常用的 alsa-utils、tiny-alsa 等工具来帮助我们日常开发使用。

3.1 常用音频工具的使用

我们常用的音频使用场景主要是音频的播放和录音，常用的有 alsa-utils 的 aplay/arecord，tiny-alsa 的/tinyplay/tinycap。由于 alsa-utils 一般会使用到 alsa-libs 和 alsa 相关的 route 配置，为避免其中的配置不当导致引起播放/录音结果与预期不符，建议项目开发过程中还是多使用更简单直接的 tiny-alsa 来避免这些问题。

- 播放命令

播放 wav 文件输出到声卡 0 上：

```
tinyplay /data/2k_1k0db_48K.wav -D 0
```

- 录音命令

录制采样率 44100Hz、位深 16bit、双声道格式的 wav 文件到声卡 0 上：

```
tinycap /tmp/my_record.wav -r 44100 -b 16 -D 0 -c 2
```

- 同时录放回采命令

RK3308 EVB 支持将 Lineout 2CH 输出的同时，loopback 到内置 CODEC ADC7/ADC8 通道。我们可以在后台启动一个 tinycap 进程的同时，去播放需要回采的 wav 文件：

```
tinycap /tmp/my_loopback.wav -r 44100 -b 16 -D 0 -c 8 &
tinyplay /data/2k_1k0db_48K.wav -D 0
```

3.2 CODEC 音频通路切换

目前 CODEC 支持使用脚本来直接切换通路，比如 LINEOUT 切换到 HPOUT：

```
# /data/switch_inoutput.sh hp
switch to hp-out
dac path: hp out
```

更多的使用可以不带参数查看 usage：

```
# /data/switch_inoutput.sh
Usage:
        input <-- mi: mic-in, li: line-in
        output --> lo: line-out, hp: hp-out, lohp: both line-out and hp-out
```

3.3 CODEC 音频增益调节

RK3308 CODEC 内部支持多级输入/输出增益调节，它们的所有状态都可以用 amixer 或者 tinymix 工具来查看。

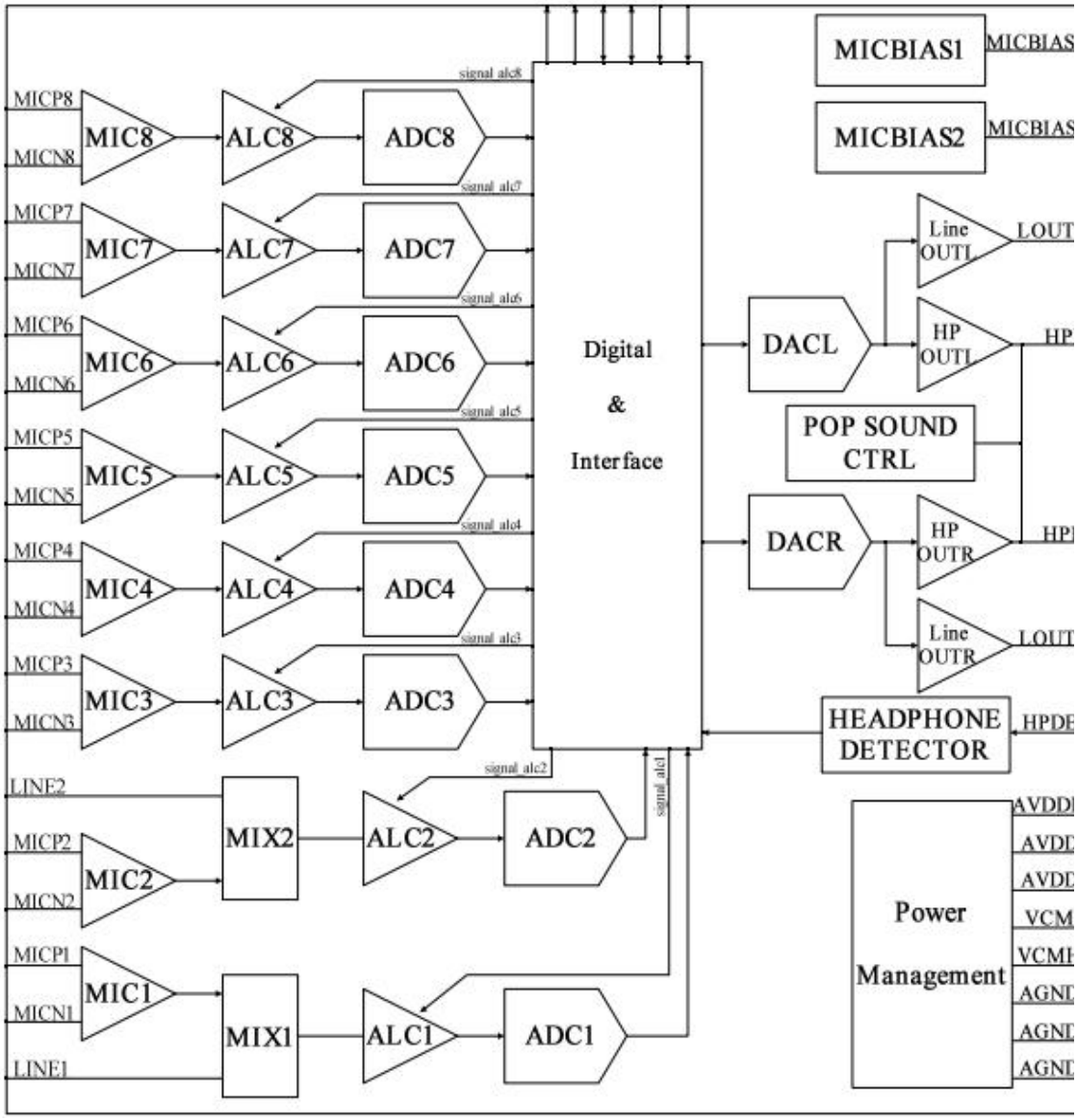
使用 amixer contents 查看，输出的内容比较多，不是很直观：

```
# amixer contents
numid=47,iface=CARD,name='Headphones Jack'
  ; type=BOOLEAN,access=r--------,values=1
  : values=off
numid=33,iface=MIXER,name='ADC ALC Group 0 Left Volume'
  ; type=INTEGER,access=rw---R--,values=1,min=0,max=31,step=0
  : values=12
  | dBscale-min=-18.00dB,step=1.50dB,mute=1
numid=34,iface=MIXER,name='ADC ALC Group 0 Right Volume'
  ; type=INTEGER,access=rw---R--,values=1,min=0,max=31,step=0
  : values=12
  | dBscale-min=-18.00dB,step=1.50dB,mute=1
```

我们可以使用 tinymix contents，这样每个通道的每个增益范围都一目了然：

```
# tinymix contents
Number of controls: 47
ctl     type    num     name                                    value
0       INT     1       ALC AGC Group 0 Left Volume             12 (range 0->31)
1       INT     1       ALC AGC Group 0 Right Volume            12 (range 0->31)
2       INT     1       ALC AGC Group 1 Left Volume             12 (range 0->31)
3       INT     1       ALC AGC Group 1 Right Volume            12 (range 0->31)
4       INT     1       ALC AGC Group 2 Left Volume             12 (range 0->31)
5       INT     1       ALC AGC Group 2 Right Volume            12 (range 0->31)
6       INT     1       ALC AGC Group 3 Left Volume             12 (range 0->31)
7       INT     1       ALC AGC Group 3 Right Volume            12 (range 0->31)
8       INT     1       ALC AGC Group 0 Left Max Volume         7 (range 0->7)
9       INT     1       ALC AGC Group 0 Right Max Volume        7 (range 0->7)
10      INT     1       ALC AGC Group 1 Left Max Volume         7 (range 0->7)
11      INT     1       ALC AGC Group 1 Right Max Volume        7 (range 0->7)
12      INT     1       ALC AGC Group 2 Left Max Volume         7 (range 0->7)
13      INT     1       ALC AGC Group 2 Right Max Volume        7 (range 0->7)
14      INT     1       ALC AGC Group 3 Left Max Volume         7 (range 0->7)
15      INT     1       ALC AGC Group 3 Right Max Volume        7 (range 0->7)
16      INT     1       ALC AGC Group 0 Left Min Volume         0 (range 0->7)
17      INT     1       ALC AGC Group 0 Right Min Volume        0 (range 0->7)
18      INT     1       ALC AGC Group 1 Left Min Volume         0 (range 0->7)
19      INT     1       ALC AGC Group 1 Right Min Volume        0 (range 0->7)
20      INT     1       ALC AGC Group 2 Left Min Volume         0 (range 0->7)
21      INT     1       ALC AGC Group 2 Right Min Volume        0 (range 0->7)
22      INT     1       ALC AGC Group 3 Left Min Volume         0 (range 0->7)
23      INT     1       ALC AGC Group 3 Right Min Volume        0 (range 0->7)
24      INT     1       ADC MIC Group 0 Left Volume             0 (range 0->3)
25      INT     1       ADC MIC Group 0 Right Volume            0 (range 0->3)
26      INT     1       ADC MIC Group 1 Left Volume             0 (range 0->3)
27      INT     1       ADC MIC Group 1 Right Volume            0 (range 0->3)
28      INT     1       ADC MIC Group 2 Left Volume             0 (range 0->3)
29      INT     1       ADC MIC Group 2 Right Volume            0 (range 0->3)
30      INT     1       ADC MIC Group 3 Left Volume             0 (range 0->3)
31      INT     1       ADC MIC Group 3 Right Volume            0 (range 0->3)
32      INT     1       ADC ALC Group 0 Left Volume             12 (range 0->31)
33      INT     1       ADC ALC Group 0 Right Volume            12 (range 0->31)
34      INT     1       ADC ALC Group 1 Left Volume             12 (range 0->31)
35      INT     1       ADC ALC Group 1 Right Volume            12 (range 0->31)
36      INT     1       ADC ALC Group 2 Left Volume             12 (range 0->31)
37      INT     1       ADC ALC Group 2 Right Volume            12 (range 0->31)
38      INT     1       ADC ALC Group 3 Left Volume             12 (range 0->31)
39      INT     1       ADC ALC Group 3 Right Volume            12 (range 0->31)
40      INT     1       DAC LINEOUT Left Volume                 0 (range 0->3)
41      INT     1       DAC LINEOUT Right Volume                0 (range 0->3)
42      INT     1       DAC HPOUT Left Volume                   0 (range 0->30)
43      INT     1       DAC HPOUT Right Volume                  0 (range 0->30)
44      INT     1       DAC HPMIX Left Volume                   0 (range 0->1)
45      INT     1       DAC HPMIX Right Volume                  0 (range 0->1)
46      BOOL    1       Headphones Jack                         Off
```

从上面的 contents dump 可以看到，可调节的 Volume Gain 有不少，这里简单说明一下。来自一张 RK3308 TRM ACODEC 章节的 Block 图：



可以看到，RK3308 CODEC 包含了 8 个 ADC 输入，将其分组的话，ADC1/ADC2 作为 Group 0，ADC3/ADC4 作为 Group 1，ADC5/ADC6 作为 Group 2，ADC7/ADC8 作为 Group 3。

“ALC AGC”前缀的表示对 AGC 自动调节增益的配置，以及调节对应通道的“Max”和“Min”部分，来控制 AGC 的范围。目前默认 AGC 是关闭状态，所以可以暂时忽略这些 Volume。

“ADC MIC”前缀表示调节前级 MIC PGA 线性放大增益。

“ADC ALC”前缀表示调节后级 ALC 线性放大增益。

“DAC LINEOUT”前缀表示调节后级 LINEOUT 线性放大增益。

“DAC HPOUT”前缀表示调节后级 HPOUT 线性放大增益。

“DAC HPMIX”前缀表示调节前级线性放大增益，它作为 LINEOUT 和 HPOUT 共同的前级。

另外，每个通道都有各自的调节范围，比如：

```
32      INT     1       ADC ALC Group 0 Left Volume                12 (range 0->31)
```

表示”ADC ALC Group 0 Left Volume”量化了 0 ~ 31 共 32 个量化等级，当前的 Volume 值是 12。如果我们想改变这个 Volume 为 20，可以通过 tinymix set 去设置它：

```
# tinymix set "ADC ALC Group 0 Left Volume" 20
```

通过 tinymix get 可以去读取指定 Volume 的状态：

```
# tinymix get "ADC ALC Group 0 Left Volume"
20 (range 0->31)
```

3.4 CODEC 高通滤波器配置

```
commit 304e48e978d4a2337022e29020afdd1c4a8c2698
Author: Xing Zheng <zhengxing@rock-chips.com>
Date:   Tue Aug 28 14:02:41 2018 +0800

    ASoC: rk3308_codec: disable high pass filter by default
```

从这个提交开始，acodec driver 将默认关闭 HPF 功能，客户需要根据实际项目情况去关闭/打开 HPF：

```
56      ENUM    1       ADC Group 0 HPF Cut-off                    20Hz245Hz612Hz, Off
57      ENUM    1       ADC Group 1 HPF Cut-off                    20Hz245Hz612Hz, Off
58      ENUM    1       ADC Group 2 HPF Cut-off                    20Hz245Hz612Hz, Off
59      ENUM    1       ADC Group 3 HPF Cut-off                    20Hz245Hz612Hz, Off
```

因为 tinymix contents 可以比较方便的一行显示 controls 的状态，但上图由于是 tinymix contents 显示排版不是很友好的原因，逗号分割在 Off 的前面，表示该 item 是选择 Off。这个可以通过 amixer sget 命令确认。比如：

```
/ # amixer sget 'ADC Group 0 HPF Cut-off'
Simple mixer control 'ADC Group 0 HPF Cut-off',0
  Capabilities: enum
  Items: '20Hz' '245Hz' '612Hz' 'Off'
  Item0: 'Off'
```

如果需要是能 acodec 的 HPF Cut-off 20Hz 功能（一般作用是去直流分量），则可以：

```
amixer sset 'ADC Group 0 HPF Cut-off' 20Hz
amixer sset 'ADC Group 1 HPF Cut-off' 20Hz
amixer sset 'ADC Group 2 HPF Cut-off' 20Hz
amixer sset 'ADC Group 3 HPF Cut-off' 20Hz
```

对 4 组/8 个 ADC 通道开启截止频率 20Hz 的 HPF：

```
56      ENUM    1       ADC Group 0 HPF Cut-off                 , 20Hz245Hz612HzOff
57      ENUM    1       ADC Group 1 HPF Cut-off                 , 20Hz245Hz612HzOff
58      ENUM    1       ADC Group 2 HPF Cut-off                 , 20Hz245Hz612HzOff
59      ENUM    1       ADC Group 3 HPF Cut-off                 , 20Hz245Hz612HzOff
```

这时逗号是分割在 20Hz 前面，item 选择的是 20Hz：

```
/ # amixer sget 'ADC Group 0 HPF Cut-off'
Simple mixer control 'ADC Group 0 HPF Cut-off',0
  Capabilities: enum
  Items: '20Hz' '245Hz' '612Hz' 'Off'
  Item0: '20Hz'
```

3.5 CODEC AGC 相关配置简要说明

默认情况下，acodec 中 8 个 ADC AGC 是处于关闭状态：

```
24      ENUM    1       ALC AGC Group 0 Left Switch             , OffOn
25      ENUM    1       ALC AGC Group 0 Right Switch            , OffOn
26      ENUM    1       ALC AGC Group 1 Left Switch             , OffOn
27      ENUM    1       ALC AGC Group 1 Right Switch            , OffOn
28      ENUM    1       ALC AGC Group 2 Left Switch             , OffOn
29      ENUM    1       ALC AGC Group 2 Right Switch            , OffOn
30      ENUM    1       ALC AGC Group 3 Left Switch             , OffOn
31      ENUM    1       ALC AGC Group 3 Right Switch            , OffOn
```

如果客户想使能 acodec 的 AGC 功能。这里以 ADC3 通道为例：

ADC3 属于 ADC Group 1 Left 通道，我们先要根据目前采样率设置 AGC 对应的 ASR（Approximate Sample Rate）。比如，当前录音的采样率是 44.1KHz，ADC3 的 ASR 就需要配置成 44.1KHz：

amixer sset 'AGC Group 1 Left Approximate Sample Rate' 44.1KHz

```
/ # amixer sset 'AGC Group 1 Left Approximate Sample Rate' 44.1KHz
Simple mixer control 'AGC Group 1 Left Approximate Sample Rate',0
  Capabilities: enum
  Items: '96KHz' '48KHz' '44.1KHz' '32KHz' '24KHz' '16KHz' '12KHz' '8KHz'
  Item0: '44.1KHz
```

然后打开 ADC3 的 AGC：

amixer sset 'ALC AGC Group 1 Left' On

```
/ # amixer sset 'ALC AGC Group 1 Left' On
Simple mixer control 'ALC AGC Group 1 Left',0
  Capabilities: volume volume-joined enum
  Items: 'Off' 'On'
  Item0: 'On'
```

这样，ADC3 就使能了 AGC 功能。

4. RK3308 CODEC 常用属性配置

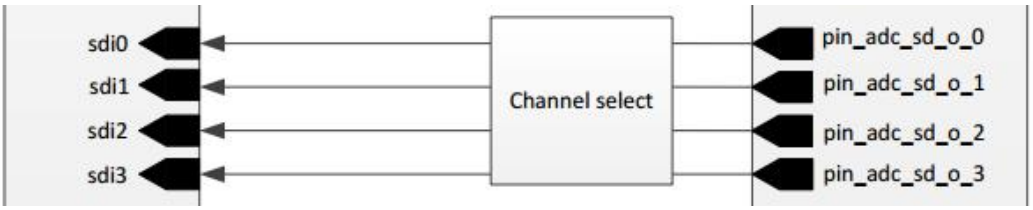
RK3308 CODEC 的属性比较多，dts 里能使用到的属性都可以在 kernel 代码的：

Documentation/devicetree/bindings/sound/rockchip,rk3308-codec.txt

找到，这里介绍一些平时开发中常用的属性，以便于理解：

- rockchip,adc-grps-route

这个属性是可选项，可以调整各路 ADC 与 i2s sdi 的映射关系。注意，这里不是指 MIC 和 ADC 的映射，是 CODEC IP 出来的 sdo 与连接的 i2s IP 的 sdi 的映射关系。具体框图可以参考 RK3308 TRM Chapter 6 Audio Subsystem：



如果不指定，默认为一一映射的关系。比如 CODEC 和 i2s_8ch_2 连接，录制 8ch 时。ADC_MIC 0-7 分别对应 i2s_8ch_2 的 sdi 0-7:

```
* sdi_0 <-- sdo_0 <-- MIC_0_1
* sdi_1 <-- sdo_1 <-- MIC_2_3
* sdi_2 <-- sdo_2 <-- MIC_4_5
* sdi_3 <-- sdo_3 <-- MIC_6_7
```

如果有这样一个场景，我们希望再录制 4ch 的时候，MIC_2_3 移到最后作为回采功能，MIC_4_5 移到最前面，dts 里可以增加这样的描述：

```
rockchip,adc-grps-route = <2 1 3 0>;
```

此时的 ADC 通路的映射关系：

```
* sdi_0 <-- sdo_1 <-- MIC_2_3   // ch0 and ch1
* sdi_1 <-- sdo_0 <-- MIC_0_1   // ch2 and ch3
* sdi_2 <-- sdo_3 <-- MIC_6_7   // not used
* sdi_3 <-- sdo_2 <-- MIC_4_5   // not used
```

由于我们只用到 4ch，i2s 的 sdi2 和 sdi3 并没有被使用到，所以后面 2 租的描述可以

忽略。

- rockchip,loopback-grp

  这个属性指定的是模拟 PA 对应的连接的 ADC group，通过这个属性，codec driver 会在合适的时间打开回采，以节省功耗。比如，回采电路连接在 ADC_0_1 上，我们可以在 dts 指定：

  rockchip,loopback-grp = <0>;

- rockchip,no-hp-det

  该属性表明 CODEC 就不会去使能 hp-det 的功能。如果目标板硬件上没有用 CODEC 的耳机检测功能，CODEC hp-det pin 悬空，该属性强烈建议加上，否则会引起耳机插入误报的现象。

- hp-ctl-gpios

  该属性指定了控制耳机通路的 gpio pin。在耳机通路使能下，播放/关闭音乐的时候，打开/关闭耳机通路模块。比如：

  hp-ctl-gpios = <&gpio0 1 GPIO_ACTIVE_HIGH>;

- spk-ctl-gpios

  该属性指定了控制喇叭通路的 gpio pin。在喇叭通路使能下，播放/关闭音乐的时候，打开/关闭 PA 通路模块。比如：

  spk-ctl-gpios = <&gpio0 5 GPIO_ACTIVE_HIGH>;

- rockchip,en-always-grps

  该属性可以让指定的 ADC group 打开一次之后就常开，主要应用于与 VAD 配合的场景，即在休眠的时候不关闭与 VAD 相关的 ADC，达到快速响应的功能。

- rockchip,delay-loopback-handle-ms

  因为在实际开发过程中，选用的 PA 的启动时延不同。该属性指定了打开回采后，需要等待的稳定时延才重新打开对应的 ADC，避免回采数据抖动。比如：

rockchip,delay-loopback-handle-ms = <200>;

- rockchip,no-deep-low-power

  该属性表明在系统休眠的时候 CODEC 不进入低功耗模式，以适应更快速的响应需求，适合对功耗不是很在意的场景。