Build a Smarter World

MDM9x07&MDM9628

Codec 驱动定制

LTE 系列

版本：MDM9x07&MDM9628 Codec 驱动定制

日期：2018-03-01

状态：受控文件

上海移远通信技术股份有限公司始终以为客户提供最及时、最全面的服务为宗旨。如需任何帮助，请随时联系我司上海总部，联系方式如下：

上海移远通信技术股份有限公司
上海市徐汇区虹梅路 1801 号宏业大厦 7 楼　　邮编：200233
电话：+86 21 51086236　　邮箱：info@quectel.com

或联系我司当地办事处，详情请登录：
http://quectel.com/cn/support/sales.htm

如需技术支持或反馈我司技术文档中的问题，可随时登陆如下网址：
http://quectel.com/cn/support/technical.htm
或发送邮件至：support@quectel.com

前言

上海移远通信技术股份有限公司提供该文档内容用以支持其客户的产品设计。客户须按照文档中提供的规范、参数来设计其产品。由于客户操作不当而造成的人身伤害或财产损失，本公司不承担任何责任。在未声明前，上海移远通信技术股份有限公司有权对该文档进行更新。

版权申明

文档历史

修订记录

Revision	Date	Author	Description
1.0	2018-03-01	Yang Yang	Initial

目录

表格索引

图片索引

Codec 定制需求

移远默认支持几款 Codec（ALC5616,NAU8814,NAU8810,TLV320AIC3104），Codec driver 支持以模块的方式加载和静态加载方式两种方式，客户可以根据自己需求，修改 Codec 加载方式。如果不使用移远推荐的 Codec，请参照此文档修改代码，集成相应 Codec driver。

1. 硬件连接图及 gpio 口配置信息

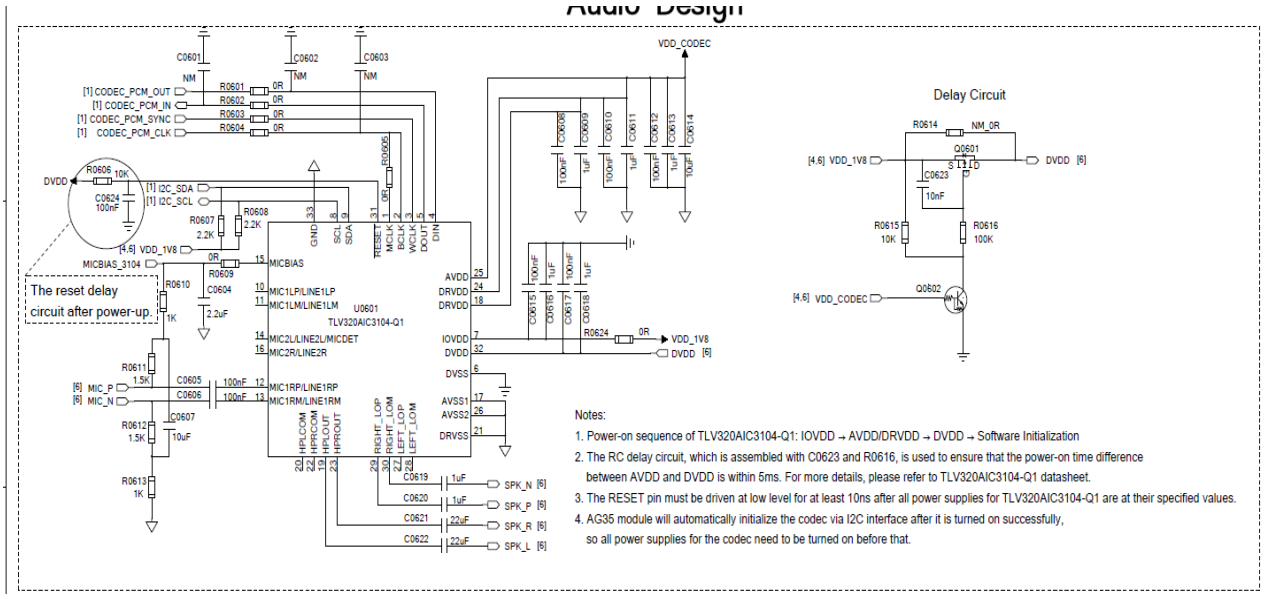

图 1：TLV320AIC3104 硬件连接图

表 1：CODEC PIN

CODEC PIN	MDM9x28 PIN
CODEC_PCM_OUT	GPIO77/GPIO22
CODEC_PCM_IN	GPIO76/GPIO21
CODEC_PCM_SYNC	GPIO79/GPIO20
CODEC_PCM_CLK	GPIO78/GPIO23
I2C_SDA	GPIO18(AGxx)，GPIO6(ECxx)，
I2C_SCL	GPIO19(AGxx)，GPIO7(ECxx)，

在表格中 CODEC_PCM_OUT ,CODEC_PCM_IN ,CODEC_PCM_SYNC ,CODEC_PCM_CLK 这个 4 个管脚是控制音频的数据和时钟信号的管脚，I2C_SDA(GPIO18),I2C_SCL(GPIO19)这两个管脚是 i2c 的数据和 clk 管脚，控制对 codec 寄存器的读写操作。

2. Codec 驱动添加方法

如果要添加一款新的 codec 驱动支持,需要经过大概如下四个步骤

1) 根据 codec datasheet 上 PCM 接口要求，确定 PCM 上的 mode,fsync,clock,format 等参数，然后根据这些参数配 9607 DTSI 来配置模块 PCM 管脚复用和模块端的 PCM 格式 。具体配置信息参考 2.2.1 章节
2) 拿到 codec 驱动代码并添加到内核中，添加方法参考 2.2.2 章节描述
3) 配置开机启动时使用的 codec 名称，配置方法参考 2.2.3 章节
4) 重新编译下载，确认新加的 codec 后是已经正常启用

下面以 ALC5616 为例子介绍具体的 codec 驱动添加方法。

2.1 根据 codec PCM 参数配置模块 PCM 接口参数

修改 msm-3.18/arch/arm/boot/dts/qcom/mdm9607.dtsi

```
sound-9306 {
        compatible = "qcom,mdm9607-audio-tapan";
        qcom,model = "mdm9607-tapan-i2s-snd-card";
        status = "disabled";
        ......
+++dai_sec_auxpcm: qcom,msm-sec-auxpcm {
        +++compatible = "qcom,msm-auxpcm-dev";
        +++qcom,msm-cpudai-auxpcm-mode = <0>, <0>;
        +++qcom,msm-cpudai-auxpcm-sync = <1>, <1>;
        +++qcom,msm-cpudai-auxpcm-frame = <5>, <5>;
        +++qcom,msm-cpudai-auxpcm-quant = <2>, <2>;
        +++qcom,msm-cpudai-auxpcm-num-slots = <1>, <1>;
        +++qcom,msm-cpudai-auxpcm-slot-mapping = <1>, <1>;
        +++qcom,msm-cpudai-auxpcm-data = <0>, <0>;
        +++qcom,msm-cpudai-auxpcm-pcm-clk-rate = <2048000>, <4096000>;
        +++qcom,msm-cpudai-afe-clk-ver = <2>;
        +++qcom,msm-auxpcm-interface = "secondary";
        +++pinctrl-names = "default", "idle";
        +++pinctrl-0 = <&sec_auxpcm_ws_active
            &sec_auxpcm_sck_active
            &sec_auxpcm_dout_active
            &sec_auxpcm_din_active>;
        +++pinctrl-1 = <&sec_auxpcm_ws_sleep
            &sec_auxpcm_sck_sleep
            &sec_auxpcm_dout_sleep
            &sec_auxpcm_din_sleep>;
    };
```

使用 alc5616 的 codec 驱动 rt5616_set_dai_fmt 接口来配置 codec 驱动参数

```
static int rt5616_set_dai_fmt(struct snd_soc_dai *dai, unsigned int fmt)
{
    struct snd_soc_codec *codec = dai->codec;
    struct rt5616_priv *rt5616 = snd_soc_codec_get_drvdata(codec);
    unsigned int reg_val = 0;

    switch (fmt & SND_SOC_DAIFMT_MASTER_MASK) {
    case SND_SOC_DAIFMT_CBM_CFM:
        rt5616->master[dai->id] = 1;
        break;
    case SND_SOC_DAIFMT_CBS_CFS:
        reg_val |= RT5616_I2S_MS_S;
        rt5616->master[dai->id] = 0;
        break;
    default:
        return -EINVAL;
    }

    switch (fmt & SND_SOC_DAIFMT_INV_MASK) {
    case SND_SOC_DAIFMT_NB_NF:
        break;
    case SND_SOC_DAIFMT_IB_NF:
        reg_val |= RT5616_I2S_BP_INV;
        break;
    default:
        return -EINVAL;
    }

    switch (fmt & SND_SOC_DAIFMT_FORMAT_MASK) {
    case SND_SOC_DAIFMT_I2S:
        break;
    case SND_SOC_DAIFMT_LEFT_J:
        reg_val |= RT5616_I2S_DF_LEFT;
        break;
    case SND_SOC_DAIFMT_DSP_A:
        reg_val |= RT5616_I2S_DF_PCM_A;
        break;
    case SND_SOC_DAIFMT_DSP_B:
        reg_val |= RT5616_I2S_DF_PCM_B;
9867/ql-ol-sdk/ql-ol-kernel/msm-3.18/sound/soc/codecs/alc5616.c [FORMAT=unix] [T
```

2.2 codec 驱动添加到内核

1)根据 alc5616 codec datasheet 来编写的 alc5616 的 codec 的驱动源代码。
2)把编写好的 codec 驱动 alc5616.c,alc5616.h,放到 msm-3.18/sound/soc/codecs 目录下
3)修改 msm-3.18/arch/arm/config/mdm9607_degconfig(debug) mdm9607-perf_defconfig(release)
+++CONFIG_SND_SOC_ALC5616=y 或 +++CONFIG_SND_SOC_ALC5616=y
修改 msm-3.18/sound/soc/codec/Makefile ,
snd-soc-tas2552-objs := tas2552.o
+++snd-soc-alc5616-objs := alc5616.o,
+++obj-$(CONFIG_SND_SOC_ALC5616) += snd-soc-alc5616.o
修改 msm-3.18/sound/soc/codecs/Kconfig
Config SND_SOC_TLV320AIC3XX
Tristate”Texas Instruments TLV320AIC31xx”
Depends on I2C
Select REGMAP_I2c
+++config SND_SOC_ALC5616
+++tristate “alc5616 codecs”

2.3 选中开机加载的 codec

codec_name 在注册 codec 驱动的时候,代码获取了 i2c_driver 的 name 和 i2c bus 和 slave addr 把三个字符串拼接在一起了。加载 codec 新的驱动的时候 codec_name = “i2c->driver->name.i2c_bus-i2c->addr”,对于 rx_dai_name,tx_dai_name 只要跟 codec 驱动代码 snd_soc_dai_driver 的 name 相同就可以了.

```
struct snd_soc_dai_driver rt5616_dai[] = {
    {
        .name = "rt5616-aif1",
        .id = RT5616_AIF1,
        .playback = {
            .stream_name = "AIF1 Playback",
            .channels_min = 1,
            .channels_max = 2,
```

修改 msm-3.18/sound/soc/msm/mdm9607.c

---static char quec_codec_name[32] = {'a'};

---static char quec_rx_dai_name[32]= {'a'};

---static char quec_tx_dai_name[32] = {'a'};

+++static char quec_codec_name[32] = {"alc5616-codec.4-001b"};

+++static char quec_rx_dai_name[32]= {"rt5616-aif1"};

+++static char quec_tx_dai_name[32] = {"rt5616-aif1"}

2.4 编译下载，调试确认新的 codec 驱动是否加载起来

SDK 的开发环境编译

a. make kernel_menuconfig (修改 xxx_defconfig 此步骤必须要)

b. make kernel

c. 查看我们的驱动代码是否编译生成,alc5616.o

```
yang@server2:~/yang_9867/ql-ol-sdk/ql-ol-kernel/msm-3.18/build/sound/soc/codecs$ ls
alc5616.o          max9867.o          snd-soc-alc5616.o     snd-soc-wcd9306.o  tlv320aic3x.o      wcd9330-tables.o  wcdcal-hwdep.o
audio-ext-clk.o    modules.builtin    snd-soc-max9867.o     snd-soc-wcd9330.o  wcd9306.o          wcd9xxx-common.o  wcd_cpe_core.o
audio-ext-clock.o  modules.order      snd-soc-msm-stub.o    snd-soc-wcd9xxx.o  wcd9306-tables.o   wcd9xxx-mbhc.o    wcd_cpe_services.o
built-in.o         msm_stub.o         snd-soc-tlv320aic3x.o snd-soc-wcd-cpe.o  wcd9330.o          wcd9xxx-resmgr.o
```

d. 查看声卡 pcm 设备注册 ls /dev/snd

```
~ # ls /dev/snd
controlC0  hwC0D33    pcmC0D12p  pcmC0D17c  pcmC0D21c  pcmC0D3c  pcmC0D9c
hwC0D10    hwC0D39    pcmC0D13c  pcmC0D17p  pcmC0D22p  pcmC0D3p  timer
hwC0D11    hwC0D40    pcmC0D13p  pcmC0D18c  pcmC0D23c  pcmC0D4c
hwC0D12    hwC0D9     pcmC0D14c  pcmC0D18p  pcmC0D24p  pcmC0D4p
hwC0D13    pcmC0D0c   pcmC0D14p  pcmC0D19c  pcmC0D25c  pcmC0D5p
hwC0D15    pcmC0D0p   pcmC0D15c  pcmC0D1c   pcmC0D26p  pcmC0D6c
hwC0D16    pcmC0D10p  pcmC0D16c  pcmC0D1p   pcmC0D2c   pcmC0D7p
hwC0D32    pcmC0D11c  pcmC0D16p  pcmC0D20p  pcmC0D2p   pcmC0D8c
~ #
```

3. Codec Driver Test

a.第一路 PCM Test

playback

amix 'AUX_PCM_RX Audio Mixer MultiMedia1' 1

aplay /data/ringtone1.wav

recording

amix 'MultiMedia1 Mixer AUX_PCM_UL_TX' 1

arec -C 1 -R 8000 data/rec.wav

voice call

amix 'AUX_PCM_RX_Voice Mixer CSVoice' 1

amix 'Voice_Tx Mixer AUX_PCM_TX_Voice' 1

aplay -D hw:0,2 -P&arec -D hw:0,2 -P -R 8000 -C 1

b.第二路 PCM Test

playback

amix 'SEC_AUX_PCM_RX Audio Mixer MultiMedia1' 1

aplay /data/ringtone1.wav

recording

amix 'MultiMedia1 Mixer SEC_AUX_PCM_UL_TX' 1

arec -C 1 -R 8000 data/rec.wav

voice call

amix 'SEC_AUX_PCM_RX_Voice Mixer CSVoice' 1

amix 'Voice_Tx Mixer SEC_AUX_PCM_TX_Voice' 1

aplay -D hw:0,2 -P&arec -D hw:0,2 -P -R 8000 -C 1