AW883XX Android Driver(MTK)

版本： V2.1

时间： 2022 年 05 月 25 日

目录

1. 驱动说明

驱动源码文件	aw883xx.c, aw883xx.h, aw883xx_pid_2049_reg.h,aw883xx_monitor.c, aw883xx_monitor.h,aw883xx_log.h,aw883xx_init.c,aw883xx_device.c, aw883xx_device.h,aw883xx_data_type.h,aw883xx_calib.h,aw883xx_calib.c, aw883xx_bin_parse.c,aw883xx_bin_parse.h,aw883xx_spin.c,aw883xx_spin.h
驱动支持产品	aw88394、aw88395
I²C 地址范围	0x34/0x35/0x36/0x37
平台信息	mt6853

2. 驱动移植

2.1 SmartPA 配置

2.1.1 添加 PA 配置

在 ProjectXXX.mk 中添加

```
MTK_AUDIO_SPEAKER_PATH = smartpa_awinic_aw883xx
```

配置以上选项会在整编时在 AudioParamOptions.xml 中生成以下参数

```
<Param name="MTK_AUDIO_SPEAKER_PATH" value="smartpa_awinic_aw883xx" />
```

SmartPa_AudioParam.xml 添加 aw883xx 参数配置

```
--- a/audio_param/SmartPa_AudioParam.xml
+++ b/audio_param/SmartPa_AudioParam.xml
@@ -8,6 +8,7 @@
     <Param path="smartpa_cirrus_cs35l35" param_id="4"/>
     <Param path="smartpa_mtk_dummy" param_id="5"/>
     <Param path="smartpa_mtk_mt6660" param_id="5"/>
+    <Param path="smartpa_awinic_aw883xx" param_id="20"/>
   </ParamTree>
   <ParamUnitPool>
     <ParamUnit param_id="0">
@@ -76,5 +77,16 @@
       <Param name="is_apll_needed" value="1"/>
       <Param name="i2s_set_stage" value="4"/>
     </ParamUnit>
+    <ParamUnit param_id="20">
```



```
+       <Param name="have_dsp" value="1"/>
+       <Param name="is_alsa_codec" value="1"/>
+       <Param name="chip_delay_us" value="22000"/>
+       <Param name="supported_rate_list"
 value="0x1F40,0x2B11,0x2EE0,0x3E80,0x5622,0x5DC0,0x7D00,0xAC44,0xBB80"/>
+       <Param name="spk_lib_path" value=""/>
+       <Param name="spk_lib64_path" value=""/>
+       <Param name="codec_ctl_name" value=""/>
+       <Param name="is_apll_needed" value="1"/>
+       <Param name="i2s_set_stage" value="5"/>
+    </ParamUnit>
   </ParamUnitPool>
 </AudioParam>
```

SmartPa_ParamUnitDesc.xml 中添加描述：

```
<Category name="smartpa_awinic_aw883xx"/>
```

2.2 AW883XX 驱动移植

2.2.1 DTS 配置

单 PA 配置方法

```
&i2c_x {                                                    /*x 表示对应的总线号*/
+       aw883xx_smartpa_0: aw883xx_smartpa@34 { /*以 I2C 地址 0x34 为例*/
+           compatible = "awinic,aw883xx_smartpa";
+           #sound-dai-cells = <0>;
+           reg = <0x34>;
+           reset-gpio = <&pio 89 0>;       /*复位引脚配置，以 gpio 89 举例*/
+           irq-gpio = <&pio 37 0x0>;       /*中断引脚配置，以 gpio 37 举例*/
+           sound-channel = <0>;
+           re-min = <1000>;                 /*校准 re 范围最小值(mohms)*/
+           re-max= <40000>;                 /*校准 re 范围最大值(mohms)*/
+           status = "okay";
+       };
        /*re 为喇叭阻抗*/
```

多 PA 配置方法

```
&i2c_x {                                /*x 表示对应的总线号*/
+       aw883xx_smartpa_0: aw883xx@34 {
+          compatible = "awinic,aw883xx";
```

```
+            #sound-dai-cells = <0>;
+            reg = <0x34>;
+            reset-gpio = <&pio 89 0x0>;
+            irq-gpio = <&pio 37 0x0>;
+            sound-channel = <0>;        /*第 1 个 PA sound-channel 配置为 0*/
+             re-min = <1000>;
+             re-max= <40000>;
+            status = "okay";
+        };
+        aw883xx_smartpa_1: aw883xx@35 {
+            compatible = "awinic,aw883xx";
+            #sound-dai-cells = <0>;
+            reg = <0x35>;
+            reset-gpio = <&pio 17 0x0>;
+            irq-gpio = <&pio 19 0x0>;
+            sound-channel = <1>;       /*第 2 个 PA sound-channel 配置为 1*/
+             re-min = <1000>;
+             re-max= <40000>;
+            status = "okay";
+      };
+};
/*以上为双 PA 举例，多 PA 时 sound-channel 序号依次递增。其他属性参考单 PA 说明*/
```

2.2.2 驱动配置

内核编译配置

defconfig 编译配置选项：

```
CONFIG_SND_SMARTPA_AW883XX=y
```

内核 codecs 目录创建 aw883xx 目录，添加驱动文件

```
aw883xx.c,aw883xx.h,aw883xx_pid_2049_reg.h,aw883xx_monitor.c,aw883xx_monitor.h,
aw883xx_log.h,aw883xx_init.c,aw883xx_device.c,aw883xx_device.h,aw883xx_data_type.h,
aw883xx_calib.h,aw883xx_calib.c,aw883xx_bin_parse.c,aw883xx_bin_parse.h,aw883xx_spin.c,
aw883xx_spin.h
```

Kconfig 配置

```
config SND_SMARTPA_AW883XX
tristate "SoC Audio for awinic aw883xxseries"
depends on I2C
help
    This option enables support for aw883xxseries Smart PA.
```

Makefile 配置

```
#for AWINIC AW883XXSmart PA
```



```
obj-$(CONFIG_SND_SMARTPA_AW883XX) += aw883xx/aw883xx.o aw883xx/aw883xx_monitor.o
aw883xx/aw883xx_bin_parse.o aw883xx/aw883xx_device.o aw883xx/aw883xx_init.o
aw883xx/aw883xx_calib.o aw883xx/aw883xx_spin.o
```

KO 编译配置

若驱动以 KO 方式进行编译，需要配置以下宏定义

```
diff --git a/aw883xx.h b/aw883xx.h
index a377035..10bdfdc 100644
--- a/aw883xx.h
+++ b/aw883xx.h
@@ -10,7 +10,7 @@
 #define AW_MTK_PLATFORM

+#define AW_SYNC_LOAD

 #define AW883XX_CHIP_ID_REG    (0x00)
```

2.2.3 bin 文件配置

PA 需要配置寄存器等参数才能正常工作，PA bin 文件配置步骤如下:

编译配置

在项目对应位置添加 bin 文件编译选项：

```
PRODUCT_COPY_FILES += \
xxxx/aw883xx_acf.bin:$(TARGET_COPY_OUT_VENDOR)/firmware/aw883xx_acf.bin

/*xxxx 为平台路径*/
```

路径配置

在内核 firmware_class.c 中添加 bin 文件在手机中的目录，一般目录为 **vendor/firmware**

```
static const char * const fw_path[] = {
    fw_path_para,
    "/vendor/firmware",                /*添加路径*/
    "/lib/firmware/updates/" UTS_RELEASE,
    "/lib/firmware/updates",
    "/lib/firmware/" UTS_RELEASE,
    "/lib/firmware"
};
```

/* 注：调试阶段可直接 push bin 文件到**/vendor/firmware/** */

bin 文件选择

根据平台 I2S 输出格式以及 PA 数量选择 bin 文件，以 aw88394 为例：

2.3 平台驱动配置

2.3.1 DAI_LINK 配置

不同平台 dai_link 配置如下。

4G 平台配置

1）添加 awinic_codecs 数组。

单 PA 配置方法

```
struct snd_soc_dai_link_component awinic_codecs[] = {
    {
        .of_node = NULL,
        .dai_name = "aw883xx-aif-6-34",         /*以 I2C 总线 6，地址 34 举例*/
        .name = "aw883xx_smartpa.6-0034",
    },
};
```

多 PA 配置方法

```
struct snd_soc_dai_link_component awinic_codecs[] = {
    {
        .of_node = NULL,
        .dai_name = "aw883xx-aif-6-34",         /*以 I2C 总线 6，地址 34 举例*/
        .name = "aw883xx_smartpa.6-0034",
    },
```



```
    {
        .of_node = NULL,
        .dai_name = "aw883xx-aif-6-35",        /*以 I2C 总线 6，地址 35 举例*/
        .name = "aw883xx_smartpa.6-0035",
    },
};
/*以上以双 PA 配置举例，多 PA 时依次增加配置*/
```

2） 修改 DAI 接口

```
static struct snd_soc_dai_link mt_soc_extspk_dai[] = {
    {
        .name = "Ext_Speaker_Multimedia",
        .stream_name = MT_SOC_SPEAKER_STREAM_NAME,
        .cpu_dai_name = "snd-soc-dummy-dai",
        .platform_name = "snd-soc-dummy",
    +#ifdef CONFIG_SND_SMARTPA_AW883XX
    +   .num_codecs = ARRAY_SIZE(awinic_codecs),
    +   .codecs = awinic_codecs,
    +#endif
        .ops = &mt_machine_audio_ops,
    }
}
```

5G 平台配置

单 PA 配置方法

```
b/arch/arm/boot/dts/mediatek/mt6853.dtsi
index f22db2e..a340a32 100644
--- a/arch/arm64/boot/dts/mediatek/mt6853.dts
+++ b/arch/arm/boot/dts/mediatek/mt6853.dts
@@ -2824,7 +2824,7 @@
        mtk_spk_i2s_in = <0>;
        /* mtk_spk_i2s_mck = <3>; */
        mediatek,speaker-codec {
-           sound-dai = <&speaker_amp>;
+           sound-dai = <&aw883xx_smartpa_0>;
        };
    };
/*根据 dts 配置 I2C 节点<aw883xx_smartpa_0>添加 dai 信息*/
```

多 PA 配置方法

```
diff --git a/arch/arm/boot/dts/mediatek/mt6853.dtsi
b/arch/arm/boot/dts/mediatek/mt6853.dtsi
index f22db2e..a340a32 100644
--- a/arch/arm64/boot/dts/mediatek/mt6853.dts
+++ b/arch/arm/boot/dts/mediatek/mt6853.dts
@@ -2824,7 +2824,7 @@
       mtk_spk_i2s_in = <0>;
       /* mtk_spk_i2s_mck = <3>; */
       mediatek,speaker-codec {
-          sound-dai = <&speaker_amp>;
+          sound-dai = <&aw883xx_smartpa_0 &aw883xx_smartpa_1>;
       };
   };
```



```
/*以双 PA 举例，其中对应 I2C 节点的信息分别为<aw883xx_smartpa_0>、<aw883xx_smartpa_1>*/
```

2.4 平台通路配置

仅 5G 平台需要配置

```
diff --git a/arch/arm/boot/dts/mediatek/mt6853.dtsi
b/arch/arm/boot/dts/mediatek/mt6853.dtsi
index f22db2e..a340a32 100644
--- a/arch/arm/boot/dts/mediatek/mt6853.dtsi
+++ b/arch/arm/boot/dts/mediatek/mt6853.dtsi
    sound: sound {
        compatible = "mediatek,mt6853-mt6359-sound";
        mediatek,audio-codec = <&mt6359_snd>;
        mediatek,platform = <&afe>;
+       mtk_spk_i2s_out = <3>;
+       mtk_spk_i2s_in = <9>;
        mtk_spk_i2s_mck = <3>;
    };
```

2.5 驱动移植有效性验证

以上操作完成了驱动集成，通过以下驱动 log 确认移植有效：

1）I2C 通信成功:

```
[Awinic][6-0034]aw883xx_append_i2c_suffix: change name :aw883xx-aif-6-34
[Awinic][6-0034]aw883xx_append_i2c_suffix: change name :Speaker_Playback_6_34
[Awinic][6-0034]aw883xx_append_i2c_suffix: change name :Speaker_Capture_6_34
[Awinic][6-0034]aw883xx_dai_drv_append_suffix: dai name [aw883xx-aif-6-34]
[Awinic][6-0034]aw883xx_dai_drv_append_suffix: pstream_name [Speaker_Playback_6_34]
[Awinic][6-0034]aw883xx_dai_drv_append_suffix: cstream_name [Speaker_Capture_6_34]
[Awinic][6-0034]aw883xx_i2c_probe: dev_cnt 1 probe completed successfully
```

2）声卡注册成功：

```
[Awinic][6-0035]aw883xx_i2c_probe: dev_cnt 2 probe completed successfully
type=1400 audit(1667.319:26): avc: denied { net_raw } for pid=513 comm="pm-service" capability=13
type=1400 audit(1667.319:26): avc: denied { net_raw } for pid=513 comm="pm-service" capability=13
[Awinic][6-0034]aw883xx_codec_probe: enter
[Awinic][6-0034]aw883xx_add_codec_controls: enter
[Awinic][6-0034]aw883xx_request_firmware_file: loaded aw883xx_acf.bin - size: 167092
[Awinic]aw_dev_check_cfg_by_hdr: project name [aw88395]
```

3）bin 文件加载成功：

```
[Awinic][6-0034]aw_monitor_write_data_to_table: min_val:3700, max_val:3800, ipeak:0x9, gain:0x8, vmax:0xfffffe82
[Awinic][6-0034]aw_monitor_write_data_to_table: min_val:3500, max_val:3600, ipeak:0x8, gain:0xc, vmax:0xffffdf2
[Awinic][6-0034]aw_monitor_write_data_to_table: min_val:0, max_val:3400, ipeak:0x7, gain:0x10, vmax:0xfffffd5a
[Awinic][6-0034]aw_monitor_parse_vol_data_v_0_1_1: ===parse_vol end ===
[Awinic][6-0034]aw_dev_cfg_get_vaild_prof: get vaild profile:2
[Awinic][6-0034]aw_dev_cfg_load: parse cfg success
[Awinic][6-0034]aw_dev_fw_update: start update Music
[Awinic][6-0034]aw_dev_get_dsp_config: done
[Awinic][6-0034]aw_dev_mute: done
```

4）播放音乐，PA 发声：

```
[Awinic][6-0034]aw_dev_mute: enter
[Awinic][6-0034]aw_dev_mute: done
[Awinic][6-0034]aw_dev_get_int_status: read interrupt reg = 0x03d5
[Awinic][6-0034]aw_dev_get_int_status: read interrupt reg = 0x0000
[Awinic][6-0034]aw_dev_clear_int_status: done
[Awinic][6-0034]aw_dev_set_intmask: done
[Awinic][6-0034]aw_monitor_start: enter
[Awinic][6-0034]aw_device_start: done
[Awinic][6-0034]aw883xx_start_pa: start success
[Awinic][6-0034]aw_monitor_work_func: scene_mode 0,monitor_status:1, monitor_switch:0
```

3. 驱动校准功能

3.1 校准目的

针对喇叭保护需求，AW883XX 驱动支持在产线对 speaker 进行校准，并将符合要求的 speaker 的 re 值写入到手机的 persist 分区中。在开机加载芯片配置时，驱动会将 persist 分区中读到的校准值写入芯片 DSP 中，达到温度保护的作用。

3.2 校准适配

3.2.1 校准结果保存路径适配

1）驱动（aw883xx_calib.c）中定义了保存校准值文件的路径如下：

```
/*******************************cali re store example start*********************************/
#ifdef AW_CALI_STORE_EXAMPLE
/*write cali to persist file example*/
#define AWINIC_CALI_FILE  "/mnt/vendor/persist/factory/audio/aw_cali.bin"
#define AW_INT_DEC_DIGIT (10)
```

2）请确认手机中是否存在相同路径，**无对应路径时会导致校准失败**：

```
msm8996:/ # cd mnt/vendor/persist/factory/audio/
msm8996:/mnt/vendor/persist/factory/audio # pwd
/mnt/vendor/persist/factory/audio
msm8996:/mnt/vendor/persist/factory/audio # ls
aw_cali.bin
msm8996:/mnt/vendor/persist/factory/audio #
```

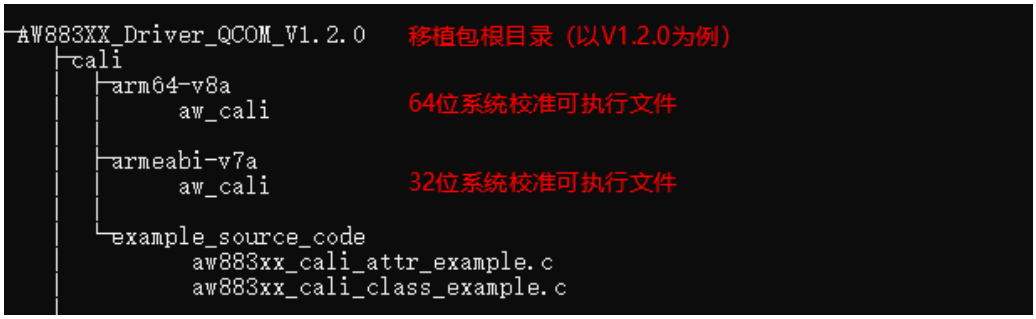
3.3 校准方式

AW883XX 驱动提供了 misc、class 和 attr 三种校准方式。

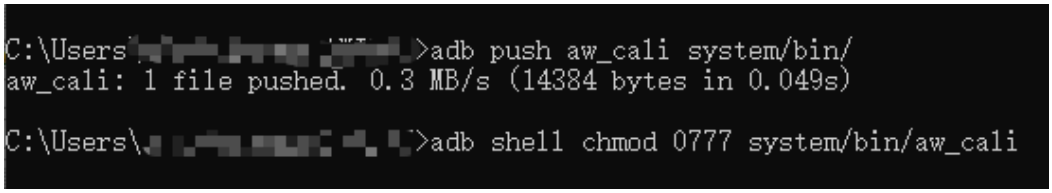
3.3.1 misc 方式

AW883XX 驱动 misc 校准通过可执行文件来实现。按照以下步骤进行：

获取可执行文件



配置可执行文件



指令介绍

参数解释（注：[]代表该选项可不填）

dev_name	用于校准单个设备，devx,x 与 dts 中配置的 sound-channel 相对应，不填写时默认校准所有设备

校准步骤

1）播放静音音乐；

2) 启动校准：

```
msm8996:/ # aw_cali start_cali
dev[0]cali_re = 6718
dev[1]cali_re = 6903
dev[0]cali_f0 = 946
dev[1]cali_f0 = 846
```

3.3.2 class 方式

节点功能

节点	功能
/sys/class/smartpa/cali_time	1.可配置校准 re 的延时时间 2.读取当前校准 re 的延时时间
/sys/class/smartpa/f0_calib	校准 f0
/sys/class/smartpa/re25_calib	1.校准 re 2.设置 re 值
/sys/class/smartpa/f0_q_calib	校准 f0 和 q 值
/sys/class/smartpa/re_range	查看 re 值设置范围

校准步骤

1) 播放静音文件；
2) 校准 re:

```
msm8996:/sys/class/smartpa #
msm8996:/sys/class/smartpa # cat re25_calib
dev[0]:6810 mOhms dev[1]:6886 mOhms
msm8996:/sys/class/smartpa #
```

校准 f0:

```
msm8996:/sys/class/smartpa #
msm8996:/sys/class/smartpa # cat f0_calib
dev[0]:1115 Hz dev[1]:949 Hz
msm8996:/sys/class/smartpa #
```

3.3.3 attr 方式

节点路径

其中 6-0034，“6”代表 i2c 总线号，“0034”代表 i2c 地址：


```
C:\Users\                    >adb shell
msm8996:/ # cd /sys/bus/i2c/drivers/aw883xx_smartpa/6-0034
msm8996:/sys/bus/i2c/drivers/aw883xx_smartpa/6-0034 # ls
awrw       cali_re    driver   dsp_re   fade_step  monitor        phase_sync reg       subsystem
cali_f0    cali_time  drv_ver  dsp_rw   i2c_log_en monitor_update power     rw        uevent
cali_f0_q  dbg_prof   dsp      fade_en  modalias   name           re_range   spk_temp
msm8996:/sys/bus/i2c/drivers/aw883xx_smartpa/6-0034 #
```

节点功能

节点	功能
cali_time	1.可配置校准 re 的延时时间 2.读取当前校准 re 的延时时间
cali_re	1.校准 re 2.设置 re 值
cali_f0	校准 f0
cali_f0_q	校准 f0、q
re_range	查看 re 设置范围

校准步骤

1）播放静音文件；
2）校准 re：

```
msm8996:/sys/bus/i2c/drivers/aw883xx_smartpa/6-0034 # cat cali_re
dev[0]: 6790mOhms dev[1]: 6859mOhms
msm8996:/sys/bus/i2c/drivers/aw883xx_smartpa/6-0034 #
```

校准 f0:

```
msm8996:/sys/bus/i2c/drivers/aw883xx_smartpa/6-0034 # cat cali_f0
dev[0]:1117 Hz dev[1]:949 Hz
msm8996:/sys/bus/i2c/drivers/aw883xx_smartpa/6-0034 #
```

3.4 校准有效性验证

1）多次校准，确认校准 re 是否在有效范围内，且值不恒定。（re 有效范围与硬件同事确认）
以 attr 方式校准 2 次举例：

```
msm8996:/sys/bus/i2c/drivers/aw883xx_smartpa/6-0034 # cat cali_re
dev[0]: 6791mOhms dev[1]: 6889mOhms
msm8996:/sys/bus/i2c/drivers/aw883xx_smartpa/6-0034 # cat cali_re
dev[0]: 6796mOhms dev[1]: 6885mOhms
msm8996:/sys/bus/i2c/drivers/aw883xx_smartpa/6-0034 #
```

2）查看 re 值是否写入文件中：

```
msm8996:/ #
msm8996:/ # cat mnt/vendor/persist/factory/audio/aw_cali.bin
      6796      6885msm8996:/ #
```

3）播放音乐状态下，查看 dsp_re 节点，确认与上述校准值相同：

```
msm8996:/sys/bus/i2c/drivers/aw883xx_smartpa/6-0034 # cat cali_re
dev[0]: 6791mOhms dev[1]: 6889mOhms
msm8996:/sys/bus/i2c/drivers/aw883xx_smartpa/6-0034 # cat cali_re
dev[0]: 6796mOhms dev[1]: 6885mOhms
msm8996:/sys/bus/i2c/drivers/aw883xx_smartpa/6-0034 # cat dsp_re
dsp_re: 6795
msm8996:/sys/bus/i2c/drivers/aw883xx_smartpa/6-0034 # cd ../6-0035
msm8996:/sys/bus/i2c/drivers/aw883xx_smartpa/6-0035 # cat dsp_re
dsp_re: 6884
```

（注：以上差异为定点化计算结果的正常误差，一般范围为 1）

4）重启手机，播放音乐，再次查看 dsp_re 节点，确认 dsp_re 节点中的值与文件中的 re 值相同。

3.5 校准示例代码

AW883XX 驱动提供了 attr、class 校准节点调用参考代码，如下所示。

```
AW883XX_Driver_QCOM_V1.2.0                    /*驱动移植包根目录（以V1.2.0为例）*/
    cali
        arm64-v8a
            aw_cali
        armeabi-v7a
            aw_cali
        example_source_code
            aw883xx_cali_attr_example.c       attr校准方式示例代码
            aw883xx_cali_class_example.c      class校准方式示例代码
```

4. 调试接口

4.1 设备节点

AW883XX Driver 创建多个设备节点文件供调试，路径是 sys/bus/i2c/drivers/aw883xx_smartpa/*-00xx，其中*为 i2c bus number，xx 为 i2c address。

reg

节点名字	reg
功能描述	用于读写 aw883xx 的所有寄存器
使用方法	读寄存器值：cat reg 写寄存器值：echo reg_addr reg_data > reg （16 进制操作）
参考例程	cat reg （获取所有可读寄存器上的值） echo 0x04 0x0241 > reg （向 0x04 寄存器写值 0x0241）

rw

节点名字	rw
功能描述	用于读写 aw883xx 的单个寄存器
使用方法	读寄存器值：echo reg_addr > rw （16 进制操作） cat rw 写寄存器值：echo reg_addr reg_data > rw （16 进制操作）
参考例程	echo 0x04 > rw （读取 0x04 寄存器值） cat rw echo 0x04 0x0241 > rw （向 0x04 寄存器写值 0x0241）

drv_ver

节点名字	drv_ver
功能描述	用于获取驱动版本号
使用方法	获取版本号：cat drv_ver

dsp_rw

节点名字	dsp_rw
功能描述	用于设置或者获取 dsp 寄存器值
使用方法	读寄存器值： echo reg_addr > dsp_rw （16 进制操作） cat dsp_rw 写寄存器值： echo reg_addr reg_data > dsp_rw （16 进制操作）
参考例程	echo 0x8601 > dsp_rw （读取 dsp 的 0x8604 寄存器值） cat dsp_rw echo 0x8604 0x4011 > dsp_rw （向 dsp 的 0x8604 寄存器写值 0x4011）

dsp

节点名字	dsp
功能描述	用于获取 dsp firmware 与 dsp config
使用方法	获取 dsp firmware 与 dsp config： cat dsp
参考例程	cat dsp

fade_step

节点名字	fade_step
功能描述	设置淡入淡出步进
使用方法	设置步进 echo step > fade_step 获取步进 cat fade_step
参考例程	echo 6 > fade_step （设置步进为 6） cat fade_step (获取当前淡入淡出步进)

dbg_prof

节点名字	dbg_prof
功能描述	用于控制是否开启场景切换
使用方法	开启场景切换 echo 1 > dbg_prof 关闭场景切换 echo 0 > dbg_prof

fade_en

节点名字	fade_en
功能描述	用于控制淡入淡出使能
使用方法	开启淡入淡出 echo 1 > fade_en 关闭淡入淡出 echo 0 > fade_en

monitor

节点名字	monitor
功能描述	用于控制低温低压开关
使用方法	开启低温低压 echo 1 > monitor 关闭低温低压 echo 0 > monitor

monitor_update

节点名字	monitor_update
功能描述	用于临时更新 monitor 配置
使用方法	更新配置 echo 1 > monitor_update

dsp_re

节点名字	dsp_re
功能描述	用于获取 dsp 中的 re 值
使用方法	cat dsp_re

i2c_log_en

节点名字	I2c_log_en
功能描述	用于控制寄存器读写 log
使用方法	开启 i2c 读写 log　echo 1 > i2c_log_en 关闭 i2c 读写 log　echo 0 > i2c_log_en

phase_sync

节点名字	phase_sync
功能描述	用于控制是否每次开启 pa 时均更新寄存器
使用方法	开启更新使能标志　echo 1 > phase_sync 关闭更新使能标志　echo 0 > phase_sync

spk_temp

节点名字	spk_temp
功能描述	用于查看喇叭实时状态
使用方法	cat spk_temp

cali_re

节点名字	cali_re
功能描述	校准 re 设置校准 re 值到 bin 与 dsp 中
使用方法	校准 re: cat cali_re 设置 re 值： echo dev[0]:6848 dev[1]:6683 > cali_re （以双 PA 配置举例，多 PA 时按照格式增加）

cali_f0

节点名字	cali_f0
功能描述	校准 f0
使用方法	校准 f0: cat cali_f0

cali_f0_q

节点名字	cali_f0_q
功能描述	校准 f0,q
使用方法	校准 f0,q: cat cali_f0_q

cali_time

节点名字	cali_time
功能描述	查看校准时间 设置校准延时时间
使用方法	查看校准时间： cat cali_time 设置校准延时时间： echo 3000 > cali_time (单位为 ms)

re_range

节点名字	re_range
功能描述	查看校准 re 值范围
使用方法	查看校准 re 值范围： cat re_range

4.2 Kcontrol 控件

其中 x 代表设备号

aw_dev_x_switch

节点名字	aw_dev_x_switch
功能描述	PA 开关

使用方法	tinymix aw_dev_ x _switch Enable 第 x 个 PA 允许开启 tinymix aw_dev_ x _switch Disable 第 x 个 PA 不允许开启

aw_dev_x_prof

节点名字	aw_dev_x_prof
功能描述	模式切换(假设 bin 文件中配置了 Music 和 Receive 模式)
使用方法	tinymix aw_dev_ x _prof Music 第 x 个 PA 切换到 Music 模式 tinymix aw_dev_ x _prof Receive 切换到 Receive 模式

aw_dev_x_monitor_switch

节点名字	aw_dev_x_monitor_switch
功能描述	PA monitor 功能开关
使用方法	tinymix aw_dev_ x _switch Enable 第 x 个 PA 允许 monitor 开启 tinymix aw_dev_ x _switch Disable 第 x 个 PA 不允许 monitor 开启

aw883xx_fadein_us

节点名字	aw883xx_fadein_us
功能描述	每个步进的淡入时间设置
使用方法	tinymix aw883xx_fadein_us 500 将每个步进淡入时间间隔设置为 500us

aw883xx_fadeout_us

节点名字	aw883xx_fadeout_us
功能描述	每个步进的淡出时间设置
使用方法	tinymix aw883xx_fadeout_us 500 将每个步进淡出时间间隔设置为 500us

5. 附录

5.1 平台 I2C 总线动态变更

若平台 I2C 总线号会发生变更，I2C 总线号的动态变更会导致 dai_link 匹配失败，可通过修改设备树配置与 dai_link 配置解决。

5.1.1DTS 配置

驱动节点增加 rename-flag 属性，并配置属性值为 1。

单 PA 配置方法

```
&i2c_x {                                                    /*x 表示对应的总线号*/
+       aw883xx_smartpa_0: aw883xx_smartpa@34 { /*以 I2C 地址 0x34 为例*/
+           compatible = "awinic,aw883xx_smartpa";
+           #sound-dai-cells = <0>;
+           reg = <0x34>;
+           reset-gpio = <&pio 89 0>;       /*复位引脚配置，以 gpio 89 举例*/
+           irq-gpio = <&pio 37 0x0>;       /*中断引脚配置，以 gpio 37 举例*/
+           sound-channel = <0>;
+           re-min = <1000>;                 /*校准 re 范围最小值(mohms)*/
+           re-max= <40000>;                 /*校准 re 范围最大值(mohms)*/
+           rename-flag= <1>;
+           status = "okay";
+       };
        /*re 为喇叭阻抗*/
```

多 PA 配置方法

```
&i2c_x {                                    /*x 表示对应的总线号*/
+       aw883xx_smartpa_0: aw883xx@34 {
+           compatible = "awinic,aw883xx";
+           #sound-dai-cells = <0>;
+           reg = <0x34>;
+           reset-gpio = <&pio 89 0x0>;
+           irq-gpio = <&pio 37 0x0>;
+           sound-channel = <0>;        /*第 1 个 PA sound-channel 配置为 0*/
+           re-min = <1000>;
+           re-max= <40000>;
+           rename-flag= <1>;
+           status = "okay";
+       };
+       aw883xx_smartpa_1: aw883xx@35 {
+           compatible = "awinic,aw883xx";
+           #sound-dai-cells = <0>;
+           reg = <0x35>;
+           reset-gpio = <&pio 17 0x0>;
+           irq-gpio = <&pio 19 0x0>;
+           sound-channel = <1>;        /*第 2 个 PA sound-channel 配置为 1*/
+           re-min = <1000>;
+           re-max= <40000>;
+           rename-flag= <1>;
```

```
+           status = "okay";
+       };
+};
/*以上为双 PA 举例，多 PA 时 sound-channel 序号依次递增。其他属性参考单 PA 说明*/
```

5.1.2 DAI_LINK 配置

codec_name 与 codec_dai_name 修改后缀为 sound-channel，不同平台的 dai_link 配置区分如下。

4G 平台配置

1） 添加 awinic_codecs 数组。

单 PA 配置方法

```
struct snd_soc_dai_link_component awinic_codecs[] = {
    {
        .of_node = NULL,
        .dai_name = "aw883xx-aif-0",        /*修改后缀为 dts 节点对应的 sound-channel */
        .name = "aw883xx_smartpa_0",
    },
};
```

多 PA 配置方法

```
struct snd_soc_dai_link_component awinic_codecs[] = {
    {
        .of_node = NULL,
        .dai_name = "aw883xx-aif-0",        /*修改后缀为 dts 节点对应的 sound-channel */
        .name = "aw883xx_smartpa_0",
    },
    {
        .of_node = NULL,
        .dai_name = "aw883xx-aif-1",        /*修改后缀为 dts 节点对应的 sound-channel */
        .name = "aw883xx_smartpa_1",
    },
};
/*以上以双 PA 配置举例，多 PA 时依次增加配置*/
```

2） 修改 DAI 接口

```
static struct snd_soc_dai_link mt_soc_extspk_dai[] = {
    {
        .name = "Ext_Speaker_Multimedia",
        .stream_name = MT_SOC_SPEAKER_STREAM_NAME,
        .cpu_dai_name = "snd-soc-dummy-dai",
        .platform_name = "snd-soc-dummy",
    +#ifdef CONFIG_SND_SMARTPA_AW883XX
    +   .num_codecs = ARRAY_SIZE(awinic_codecs),
    +   .codecs = awinic_codecs,
    +#endif
        .ops = &mt_machine_audio_ops,
    }
}
```

5G 平台配置

5G 平台不需要做额外修改；

