AW883XX Android Driver(MTK)

版本： V1.7

时间： 2021 年 9 月 24 日

修订记录

日期	版本	描述	作者
2021-3-12	V1.0	初版	赵磊
2021-3-24	V1.1	1. 增加 i2c_log_en，phase_sync，spk_temp 节点描述 2. 增加 re_range 校准命令及节点描述	周慧栋
2021-6-1	V1.2	1. 修改 MTK 5G 平台移植描述	周慧栋
2021-7-2	V1.3	1. 修改驱动 misc 校准方式 re 值设置指令描述	周慧栋
2021-7-14	V1.4	1. 修改低温低压描述	周慧栋
2021-7-30	V1.5	1. 增加校准示例代码说明 2. 默认校准 Re 值范围修改为 1000-40000mohm	王萍
2021-8-3	V1.6	1. 修改 5G dai_link 描述 2. 修改 bin 文件加载描述	周慧栋
2021-9-2	V1.6.1	1. 修改校准文件描述 2. 增加声道旋转功能描述	周慧栋
2021-9-24	V1.7	1. 增加 monitor_switch control 2. 增加 dsp 节点描述	周慧栋

目录

INFORMATION

HAL File	AudioParamOptions.xml SmartPa_ParamUnitDesc.xml
Driver File	aw883xx.c, aw883xx.h, aw_pid_2049_reg.h,aw_monitor.c, aw_monitor.h,aw_log.h,aw_init.c,aw_device.c,aw_device.h, aw_data_type.h,aw_calib.h,aw_calib.c,aw_bin_parse.c, aw_bin_parse.h,aw_spin.c,aw_spin.h
Smart PA	aw88363、aw88394、aw88395
I²C Address	0x34/0x35/0x36/0x37
ADB Debug	yes
Platform	MT6739，MT6853
Android version	android 9.0，android Q

PROJECT CONFIG

在 ProjectXXX.mk 中添加

```
MTK_AUDIO_SPEAKER_PATH = smartpa_awinic_aw883xx
```

AUIDO DEVICE

添加 aw883xx 选项

在 xxx/audio_param/AudioParamOptions.xml 中添加 aw883xx 选项。（注：该文件在整编时自动生成）

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

添加 aw883xx 参数配置

在 device/mediatek/common/audio_param_smartpa/SmartPa_AudioParam.xml 添加 aw883xx 参数配置

添加 aw883xx：

```
<Param path="smartpa_awinic_aw883xx" param_id="7"/>
```

添加 aw883xx 参数：

```
<ParamUnit param_id="7">
    <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="8000,11025,12000,16000,22050,24000,32000,44100,48000"/>
    <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>
```

添加 aw883xx 描述

在 device/mediatek/common/audio_param_smartpa/SmartPa_ParamUnitDesc.xml 中添加
<Category name="smartpa_awinic_aw883xx"/>

KERNEL DRIVER

AW883XX Smart PA Driver

添加 dts 配置

打开 kernel/arch/arm/boot/dts/mediatek/mt6853.dts 文件，添加 aw883xx 的配置。

其中 aw-cali-mode 属性对应的值为校准方式，可选“aw_attr”、“aw_misc”以及“aw_class”三种，若无该属性或错填该属性的值则只有 aw_misc 起效。

sync-flag 为 PA 同步控制方式，配置为 1 时打开 PA 同步功能，不配置时默认为 0，不开启该功能。

hw-monitor-delay 为 hardware monitor 的间隔时间，单位为 ms，不配置时，默认时间为 1000ms。根据 bin 文件配置决定是否开启 hardware monitor。

re-min 与 re-max 分别为校准 Re 值范围的最小值与最大值，不配置时默认范围为 1000-40000mohm。

如下为单 PA 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
@@ -549,6 +549,8 @@
    i2c_x {      /*x 表示对应的总线号*/
+         /* AWINIC AW883XX mono Smart PA */
+         aw883xx_smartpa_0: aw883xx_smartpa@34 {
+             compatible = "awinic,aw883xx_smartpa";
+             #sound-dai-cells = <0>;
+             reg = <0x34>;
+             reset-gpio = <&pio 89 0>;
+             irq-gpio = <&pio 37 0x0>;
+             sound-channel = <0>;
+             re-min = <1000>;
+             re-max= <40000>;
+             aw-cali-mode = "aw_attr";
+             status = "okay";
+      };
+      /* AWINIC AW883XX mono Smart PA End */
```

若为多 PA 项目，则增加 i2c 节点即可，这里以双 PA 为例，注意：不同 i2c 节点 sound-channel 属性需要不同，请根据/*0:pri_l 1:pri_r 2:sec_l 3:sec_r*/设置

```
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
@@ -549,6 +549,8 @@
&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>;
+           re-min = <1000>;
+           re-max= <40000>;
+           aw-cali-mode = "aw_attr";
+           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>;
+           re-min = <1000>;
+           re-max= <40000>;
+           aw-cali-mode = "aw_attr";
+           status = "okay";
+      };
+};
```

添加驱动文件

在 kernel/sound/soc/codecs/aw883xx 目录下添加 aw883xx 驱动文件

aw883xx.c,aw883xx.h,aw_pid_2049_reg.h,aw_monitor.c,aw_monitor.h,aw_log.h,aw_init.c,aw_device.c,aw_device.h,aw_data_type.h,aw_calib.h,aw_calib.c,aw_bin_parse.c,aw_bin_parse.h,aw_spin.c,aw_spin.h

注意：若在 codec_probe 函数中无法加载到 bin 文件，请修改 aw883xx.h 中如下的宏，增加延时加载时间：

```
#define AW883XX_LOAD_FW_DELAY_TIME      (3000)
```

或者也可以修改 aw883xx.c 如下表示 retry 加载次数的宏来满足需求，如下：

```
#define AW_REQUEST_FW_RETRIES       5   /* 5 times */
```

更新 Kconfig 和 Makefile

1) 在 kernel/sound/soc/codecs/Kconfig 中添加

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

2) 在 kernel/sound/soc/codecs/Makefile 中添加

```
#for AWINIC AW883XXSmart PA
obj-$(CONFIG_SND_SMARTPA_AW883XX) += aw883xx/aw883xx.o
aw883xx/aw_monitor.o aw883xx/aw_bin_parse.o aw883xx/aw_device.o
aw883xx/aw_init.o aw883xx/aw_calib.o aw883xx/aw_spin.o
```


添加 AW883XXfw&cfg 文件

1) 在 kernel/drivers/base/firmware_class.c 中添加 bin 文件目录，目录由系统决定，一般目录为

```
/system/vendor/firmware 或/system/etc/firmware
static const char * const fw_path[] = {
     fw_path_para,
     "vendor/firmware/awinic",
     "/system/etc/firmware",
     "/lib/firmware/updates/" UTS_RELEASE,
     "/lib/firmware/updates",
     "/lib/firmware/" UTS_RELEASE,
     "/lib/firmware"
};
```

2) 使用 adb 将 config 文件 push 到手机中，根据需求 push ap/config/ 路径下的 bin 文件。

```
adb push aw883xx_acf.bin vendor/firmware/awinic/
```

ASoc Machine Driver

4G 平台配置

若平台为 4G 平台，在 kernel/sound/soc/mediatek/common_int/mtk-soc-machine.c 中的 mt_soc_extspk_dai 添加 aw883xx 的 dai_link 配置，其他默认保持平台不变。
若为单 PA 项目，则添加以下信息，注意 6-0034 对应的总线与地址,

```
struct snd_soc_dai_link_component awinic_codecs[] = {
    {
        .of_node = NULL,
        .dai_name = "aw883xx-aif-6-34",
        .name = "aw883xx_smartpa.6-0034",
    },
};
```

若为多 PA 项目，则在该数组中继续添加设备信息，这里以双 PA 为例，一个 PA 的总线与地址为 6-0034，另一个 PA 的总线与地址为 6-0035

```
struct snd_soc_dai_link_component awinic_codecs[] = {
    {
        .of_node = NULL,
        .dai_name = "aw883xx-aif-6-34",
        .name = "aw883xx_smartpa.6-0034",
    },
    {
        .of_node = NULL,
        .dai_name = "aw883xx-aif-6-35",
        .name = "aw883xx_smartpa.6-0035",
    },
};
```

```
{
    .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 配置时根据前边 dts 配置的 I2C 节点<aw883xx_smartpa_0>添加对应 sound-dai 信息，配置 DAI_LINK。

```
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>;
        };
    };
```

多 PA 时，根据 DTS 增加的 I2C 节点信息对应配置 DAI_LINK，这里以双 PA 为例配置 DAI_LINK。

```
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>;
        };
    };
```

以上部分主要完成了 RX 部分的驱动集成，该部分集成的目的是使 PA 出声音，客户可通过 log 等手段依次确认以下各项，若均无问题，则 RX 部分驱动集成正确，且硬件完好

1) 编译可以通过；
2) I2C 通信成功；
3) 声卡注册成功；
4) PA 可以出声音。

SPEAKER CALI

校准方式

awinic 提供了三种校准的方式，分别通过 misc、class 和 attr 三种方式进行，其中 attr 和 class 这两种方式需要在设备树进行定义，才能开启该方式的校准功能。

1) misc 方式

该方式不需要在设备树中进行配置，默认有效。

AW883XX 的校准是通过/cali/aw_cali 的可执行文件来实现。将该文件放到 system/bin 目录中,并修改权限：

adb shell chmod 0777 system/bin/aw_cali

指令介绍，adb shell aw_cali 会出现指令介绍：

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

校准操作步骤：

1) 正常播放静音音乐；

2) 启动校准，结束后会输出校准值：

./system/bin/aw_cali start_cali

```
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) 如果校准值在合理范围内，驱动会默认将 Re 值设置到系统 bin 文件中。客户有客制化需求时可以通过以下命令来对 re 进行设置。

./system/bin/aw_cali set_cali_re 6718 6903

```
msm8996:/ # aw_cali set_cali_re 6718 6903
dev[0]:set cali re 6718
dev[1]:set cali re 6903
```


2) Class 方式

class 方式需在设备树的 aw-cali-mode 属性中赋“aw_class”值，才能起效。该方式利用 class 文件系统在/sys/class/smartpa 目录下创建了相关目录与节点：

节点	功能
/sys/calss/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/calss/smartpa/f0_calib	校准 f0
/sys/calss/smartpa/re25_calib	1.校准 re 2.设置 re 值
/sys/calss/smartpa/f0_q_calib	校准 f0 和 q 值
/sys/calss/smartpa/re_range	查看 re 设置值的范围

客户可以利用 fgets/fread 读取节点的值， fwrite cali_time 节点，可写入校准 re 的延时时间；

3) attr 方式

attr 方式需要在设备树的 aw-cali-mode 属性中配置“aw_attr”才可以生效。利用 device 设备属性节点，在/sys/bus/i2c/drivers/aw883xx_smartpa/*-00xx/目录下创建了相关节点，其中*为 i2c bus number，xx 为 i2c address。

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 设置值的范围

客户播放静音文件，open cali_re 节点，read 该节点，即可开启 re 校准，通过 write dev[0]:xxxx dev[1]:xxxx (根据实际设备个数写入，这里以两个设备为例)字符串到到该节点，能够写入校准 RE 值；
open cali_f0 节点， read 该节点，即可开启 f0 校准；
此外，可通过 cali_time 节点配置校准 re 的延时时间。

cali_Re 写入验证

a. 查看 re 值是否写入文件中,可直接 cat 保存 re 值的文件。文件路径默认使用 aw_calib.c 中的定义，根据客户情况可以修改：

```
#define AWINIC_CALI_FILE  "/mnt/vendor/persist/factory/audio/aw_cali.bin"
```

b.重新播放音乐后，cat dsp_re 节点，确认 dsp 中的值与写入值相同
c.重启手机，播放音乐，再次 cat dsp_re 节点，确认 dsp 中的值与文件中的 re 值

校准值的保存（示例）

由于 PA 内部没有用于保存校准 re 值的内存，故校准结束后需要保存校准数据到平台，在启动 PA 时读

取 re 值，将其设置到 mec 算法中。下面是 awinic 提供了一个将校准值写入 persist 分区文件的参考示例（代码位于 aw_calib.c）

```
/* customer need add function to set cali_re to nv or get cali_re from nv */
int aw_cali_write_re_to_nvram(int32_t cali_re, int32_t channel)
{
#ifdef AW_CALI_STORE_EXAMPLE
    return aw_cali_write_cali_re_to_file(cali_re, channel);
#else
    return -EBUSY;
#endif
}

int aw_cali_read_re_from_nvram(int32_t *cali_re, int32_t channel)
{
/*custom add, if success return value is 0 , else -1*/
#ifdef AW_CALI_STORE_EXAMPLE
    return aw_cali_get_cali_re_from_file(cali_re, channel);
#else
    return -EBUSY;
#endif
}
```

DEBUG INTERFACE

Node

AW883XX Driver 会创建多个不同功能的设备节点文件，路径是 sys/bus/i2c/drivers/aw883xx_smartpa/*-00xx，其中*为 i2c bus number，xx 为 i2c address。可以使用 adb 读写节点调试 aw883xx。

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	用于设置或者获取算法中设定的 re 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

Kcontrol

其中 x 代表设备号

aw_dev_x_switch

PA 开关
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

模式切换(假设 bin 文件中配置了 Music 和 Receive 模式)
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

aw883xx_fadein_us

每个步进的淡入时间设置
tinymix aw883xx_fadein_us 500　将每个步进淡入时间间隔设置为 500us

aw883xx_fadeout_us

每个步进的淡出时间设置
tinymix aw883xx_fadeout_us 500　将每个步进淡出时间间隔设置为 500us

aw_dev_x_monitor_switch

PA monitor 功能开关
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 ANDROID DRIVER 注意事项

关于校准节点示例代码

Awinic 在 cali\example_source_code 里提供了 attr 属性节点和 class 属性节点的校准调用示例代码；FAE 和客户可以参考使用。

AW883XX 声道旋转功能

旋转方案

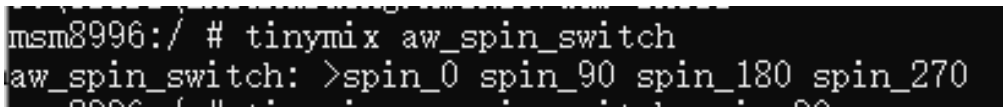
AW883XX 驱动提供了三种方案的声道旋转功能，分别为 ADSP 声道旋转、HAL 声道旋转、REG 声道旋转。通过 dts 的配置来进行选择不同方案，dts 不配置时不会开启旋转功能。

ADSP 声道旋转

ADSP 声道旋转适用于算法在平台 DSP 运行的情况，通过调用算法接口实现声道旋转。所有 PA 的 dts 中需要添加 **dsp_spin** 的配置，并在平台侧集成 DSP 通信函数。

```
&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>;
+           re-min = <1000>;
+           re-max= <40000>;
+           spin-mode = "dsp_spin";
+           aw-cali-mode = "aw_attr";
+           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>;
+           re-min = <1000>;
+           re-max= <40000>;
+           spin-mode = "dsp_spin";
+           aw-cali-mode = "aw_attr";
+           status = "okay";
+      };
+};
```

使用方法：通过调用 aw_spin_switch 控件来控制声道旋转。可设置值为 spin_0，spin_90，spin_180，spin_270，依次表示旋转 0 度，旋转 90 度，旋转 180 度，旋转 270 度。根据算法中的配置实现不同角度的旋转。



HAL 声道旋转

HAL 声道旋转适用于算法在 HAL 层的情况，旋转时通过调用算法接口混音，再根据 dts 中 spin-data 的配

置来选择 PA 处理的声道，最后结束混音完成声道旋转。所有 PA 的 dts 中需要添加 **reg_spin** 的配置。

```
&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>;
+           re-min = <1000>;
+           re-max= <40000>;
+           spin-mode = "reg_spin";
+           spin-data = "l r l r";
+           aw-cali-mode = "aw_attr";
+           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>;
+           re-min = <1000>;
+           re-max= <40000>;
+           spin-mode = "reg_spin";
+           spin-data = "r l r l";
+           aw-cali-mode = "aw_attr";
+           status = "okay";
+      };
+};
```

使用方法：通过调用 aw_spin_switch 控件来控制声道旋转。可设置值为 spin_0，spin_90，spin_180，spin_270，依次表示旋转 0 度，旋转 90 度，旋转 180 度，旋转 270 度。根据 dts 中 spin-data 的配置实现不同角度的旋转，“lrlr”代表该 PA 在 0 度、90 度、180 度、270 度分别输出左、右、左、右的声道情况。使用时根据实际需求修改配置。

```
msm8996:/ # tinymix aw_spin_switch
aw_spin_switch: >spin_0 spin_90 spin_180 spin_270
```

REG 声道旋转

REG 声道旋转适用于算法运行在平台 DSP 中的情况，与 PA 声道寄存器的配置来共同实现旋转功能。旋转时先调用算法接口混音，再根据 dts 中 spin-data 的配置来选择 PA 处理的声道，最后结束混音完成声道旋转。所有 PA 的 dts 中需要添加 reg_mixer_spin 的配置，并在平台侧集成 DSP 通信函数。

```
&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>;
+           re-min = <1000>;
+           re-max= <40000>;
+           spin-mode = "reg_mixer_spin";
+           spin-data = "l r l r";
+           aw-cali-mode = "aw_attr";
+           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>;
+           re-min = <1000>;
+           re-max= <40000>;
+           spin-mode = "reg_mixer_spin";
+           spin-data = "r l r l";
+           aw-cali-mode = "aw_attr";
+           status = "okay";
+       };
+};
```

使用方法：通过调用 aw_spin_switch 控件来控制声道旋转。可设置值为 spin_0，spin_90，spin_180，spin_270，依次表示旋转 0 度，旋转 90 度，旋转 180 度，旋转 270 度。根据 dts 中 spin-data 的配置实现不同角度的旋转，“lrlr”代表该 PA 在 0 度、90 度、180 度、270 度分别输出左、右、左、右的声道情况。使用时根据实际需求修改配置。